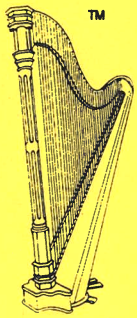


HarPE™

User's Manual

Optimization Systems Associates Inc.

HarPE™

User's Manual

Version 2.0

August 1995

Optimization Systems Associates Inc.

LIABILITY AND WARRANTY

NEITHER OPTIMIZATION SYSTEMS ASSOCIATES INC. NOR ITS EMPLOYEES, OFFICERS, DIRECTORS OR ANY OTHER PERSON, COMPANY, AGENCY OR INSTITUTION: (1) MAKES ANY WARRANTY, EXPRESS OR IMPLIED AS TO ANY MATTER WHATSOEVER REGARDING THIS MATERIAL, INCLUDING BUT NOT LIMITED TO THE GENERALITY THEREOF, ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR THOSE ARISING BY STATUTE OR OTHERWISE IN LAW OR FROM THE COURSE OF DEALING OR USAGE OF TRADE HAVE BEEN AND ARE HEREBY EXPRESSLY EXCLUDED; OR (2) ASSUMES ANY LEGAL RESPONSIBILITY WHATSOEVER FOR THE ACCURACY, COMPLETENESS OR USEFULNESS OF THIS MATERIAL; OR (3) REPRESENTS THAT ITS USE WOULD NOT INFRINGE UPON PRIVATELY OWNED RIGHTS OF THIRD PARTIES. IT IS EXPRESSLY UNDERSTOOD AND AGREED THAT ANY RISKS, LIABILITIES OR LOSSES ARISING OUT OF ANY USE, TRANSFER OR LEASE OF THIS MATERIAL WILL NOT BE ATTRIBUTED TO OPTIMIZATION SYSTEMS ASSOCIATES INC. OR ANY INDIVIDUAL ASSOCIATED WITH THE COMPANY. ACCURACY, COMPLETENESS OR USEFULNESS FOR ANY APPLICATION SHALL BE DETERMINED INDEPENDENTLY BY THE PARTY UNDERTAKING SUCH AN APPLICATION.

IN NO EVENT WHATSOEVER WILL OPTIMIZATION SYSTEMS ASSOCIATES INC., ITS EMPLOYEES, OFFICERS, DIRECTORS, OR AGENTS BE LIABLE FOR ANY DAMAGES, INCLUDING, BUT WITHOUT LIMITATION, DIRECT, INDIRECT, INCIDENTAL AND CONSEQUENTIAL DAMAGES AND DAMAGES FOR LOST DATA OR PROFITS, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS MATERIAL.

CONTENTS ARE SUBJECT TO CHANGE WITHOUT NOTICE.

Copyright

HarPE User's Manual Version 2.0 first published in 1995. Printed in Canada.

Optimization Systems Associates Inc.
P.O. Box 8083, Dundas, Ontario
Canada L9H 5E7

Tel 905 628 8228
Fax 905 628 8225

Trademarks of Optimization Systems Associates Inc.

HarPE
OSA90
OSA90/hope
Datapipe
FAST
Empipe

Other Trademarks

HP is a trademark of Hewlett-Packard Company.
Sun and Sun Workstation are trademarks of Sun Microsystems, Inc.
UNIX is a registered trademark of AT&T.
Touchstone is a registered trademark of EEsof Inc.
Libra is a trademark of EEsof Inc.
Super-Compact is a registered trademark of Compact Software Inc.
Microwave Harmonica is a trademark of Compact Software Inc.
MicroCAT is a trademark of Cascade Microtech Inc.
em is a trademark of Sonnect Software, Inc.

HarPE™ User's Manual

Table of Contents

4 Expressions

5 Circuit Model

6 Nonlinear Elements

7 Linear Elements

8 Simulation

9 Optimization

10 Monte Carlo Analysis

11 Statistical Modeling

12 Operation Control

13 Examples

Appendix Diagnostic Messages

Index

1

Introduction

1
Introduction

1.1 Overview

HarPE™ is an advanced, powerful and user-friendly integrated CAD system dedicated to complete nonlinear device characterization and single device circuit simulations and design optimizations. Emerged from many years of theoretical research by Optimization Systems Associates Inc. [1-10], HarPE was introduced as the world's first software system offering parameter extraction from harmonic measurements.

Unified DC, Small-Signal and Large-Signal Simulation

HarPE offers a comprehensive approach to device-oriented CAD. It seamlessly integrates [6] large-signal frequency-domain harmonic balance simulations with small-signal/DC simulations, using a unified circuit description. When needed, the small-signal parameters are automatically linearized from the nonlinear model. HarPE can simulate, optimize and accept as measurement data any combination of frequency-domain power spectra, DC IV characteristics, small-signal *S* parameters, as well as time-domain waveforms. The system allows you to carry out arbitrary combinations of bias, frequency and input power sweeps to investigate device behaviour or to accommodate multiple specifications for parameter extraction or design optimization. Sweeping arbitrary parameters is also possible.

HarPE offers a truly nonlinear model device parameter extraction procedure which utilizes spectrum measurements, including DC bias information and output power at different harmonics measured under large-signal RF excitations. If large-signal measurements are not available, HarPE can extract the nonlinear model parameters from small-signal *S* parameters measured at a number of bias points, or even just from DC data (which alone is not sufficient to determine some parameters such as nonlinear capacitors). In general, HarPE accepts and utilizes all information from any combination of the three types of measurement data. The unique multi-circuit approach employed by HarPE increases the model identifiability and ensures a reliable and consistent solution suitable for large-signal, small-signal and DC simulation.

State-of-the-Art Optimizers

The state-of-the-art gradient-based ℓ_1, ℓ_2 (least-squares), minimax and Huber optimizers [11-14] featured in HarPE have a proven track record in circuit optimization. Combined with OSA's unique nonlinear FAST™ adjoint sensitivity analysis [5], these optimizers provide expedient and consistent solutions to your modeling and design problems. In addition to a number of built-in responses, arbitrary user-defined functions can be optimized. The goals and window specifications can be defined both directly and through expressions. When used for parameter extraction, HarPE's unique approach of exploiting multi-input-power-level, multi-bias, multi-frequency measurements and integrated large- and small-signal modeling produces solutions with unparalleled speed, accuracy and robustness.

Typically, the ℓ_1 optimizer is the most suitable one for modeling and parameter extraction tasks. For design optimization tasks minimax is the optimizer of choice. The traditional ℓ_2 optimizer (least-squares) with its smooth properties is by some used for both purposes. The Huber optimizer, by combining the desirable properties of the ℓ_1 and ℓ_2 approaches, can provide a smooth model from data which contains many small variations and such a model is also robust against gross errors.

The parameter sensitivity analysis feature in HarPE provides a very useful aid in the selection of optimizable variables. The sensitivity of the objective function with respect to each and every optimization variable is computed and displayed. Based on this information, you will be able to select the most sensitive parameters for optimization.

Preprocessing "Cold" Measurements

HarPE offers the option of extracting the extrinsic parameters analytically from cold measurements (unbiased and pinched-off measurements) without optimization. It provides a fast way to determine the parasitic elements.

This option is very useful in two-stage parameter extraction of MESFET and HEMT models. You can first extract the extrinsic parameters analytically from cold measurements and then extract the intrinsic parameters from the "hot" measurements using optimization while keeping the extrinsic parameters fixed.

Libraries of Device Models and Linear Components

You can use the built-in nonlinear intrinsic models and linear extrinsic super-components to achieve the maximum computational efficiency. You can also create custom device models using HarPE's unique Expression Block for user-defined models.

HarPE supports a number of built-in nonlinear intrinsic device models including a diode model, a heterojunction bipolar transistor (HBT) model, several FET and HEMT models, as well as bipolar transistor models. For FETs, both the equivalent circuit models and physics based models are available. An effort has been made to ensure as much compatibility of the built-in models as possible with the models available in other commercial CAD software.

Currently, the nonlinear intrinsic device model library includes:

- Curtice and Ettenberg FET symmetrical and asymmetrical models [15]
- Materka and Kacprzak FET model [16]
- Raytheon (Statz *et al.*) FET model [17]
- Gummel and Poon bipolar transistor model [18]
- Khatibzadeh and Trew physics-based FET model [19,20]
- Modified physics-based FET model for uniform doping [21]
- Semiconductor diode model [22]

- Curtice HEMT model [23]
- Advanced Curtice HEMT model [23]
- High order beta degradation HEMT model [24]
- Double parabolic HEMT model [24]
- Heterojunction bipolar transistor model [25]

HarPE is the first commercial software system to offer characterization, simulation and optimization of physics based models to the microwave community. Since physics based models relate device responses directly to geometrical, process and material parameters, they promise to be more accurate and reliable compared with equivalent circuit models, especially in statistical modeling, analysis and optimization.

The linear library models include extrinsic super-components, lumped resistors, capacitors and inductors, transmission lines, microstrip components, as well as all types of controlled sources. HarPE also supports linear subcircuits described by "black box" S, Y or Z data imported from external simulators.

Customized Models

Using typical mathematical expressions, you can create your own device models or modify the built-in models directly in the circuit file to define process, geometrical, physical or abstract functional dependence of any model parameters. No subroutines need to be written, compiled or linked. This allows you to test, verify and even optimize in-house models effortlessly.

Expressions can be formulated using any combinations of algebraic operations, standard mathematical functions, user-defined labels and a set of predefined labels which allows you to refer to the program's internal variables, such as the frequency, bias voltages and time-domain (instantaneous) intrinsic voltages.

HarPE supports several generic topologies for user-defined nonlinear device models. Within the predefined topologies you have total flexibility in creating your own model parameters and equations. For your convenience, a number of preprogrammed device models using the user-defined model feature are supplied together with HarPE. You can use these files directly, make modifications to fit your specific needs, or use them as templates for developing your own models.

Built-in and Customized Responses

The built-in large-signal, small-signal and DC responses include:

- ▷ frequency spectra of powers, voltages and currents
- ▷ time-domain waveforms of voltages and currents
- ▷ DC IV curves
- ▷ small-signal linearized model parameters at a given bias point
- ▷ *S* parameters
- ▷ insertion loss, maximum available gain and stability factor

Customized response functions can also be created using expressions. User-defined response functions can be optimized just like the built-in responses.

Statistical Modeling

The advanced statistical modeling capabilities in HarPE include indirect and direct statistical modeling techniques.

In indirect statistical modeling, multiple device models are automatically extracted through repeated optimization from measurements. The sample of extracted models is postprocessed to produce a multidimensional statistical model. The statistical model can be back annotated into the circuit file and used in Monte Carlo analysis. Parameter correlations and discretized marginal density functions are included in the statistical model.

In direct statistical modeling, the statistical model is determined by fitting the cumulative probability distributions or the histograms of the model responses to those of the measured data [26].

Running as a Datapipe™ Child of OSA90/hope™

As a specific feature of HarPE with OSA90/hope, HarPE can be used as a Datapipe child of OSA90/hope. In other words, from OSA90/hope you can invoke HarPE via Datapipe.

This opens up new opportunities for directly and dynamically integrating the comprehensive general-purpose circuit-level features of OSA90/hope with the efficient and dedicated device modeling capabilities of HarPE to create a sophisticated environment for device modeling, simulation and optimization.

Running as a child of OSA90/hope, HarPE can be used as a device simulator for nominal and statistical Monte Carlo simulations, or as a parameter extractor for nominal and statistical device modeling. The simulation results or the extracted parameters are directly returned to the parent OSA90/hope through Datapipe.

The details of running HarPE as a child of OSA90/hope is described in the manual "HarPE as a Child of OSA90/hope".

Compatibility with Other Software

HarPE's parameter extraction capabilities can help you make the most accurate use of other CAD software you use. You will be able to supply model parameters that truly belong to your own devices for which you have access to measurement data. The built-in nonlinear device models can be directly exported into Libra™ [27] and Microwave Harmonica™ [28] input files. HarPE supports linear subcircuits described by "black box" data imported from other simulators. HarPE also directly accepts measured data from Cascade Microtech's MicroCAT™ Test Executive system, data in the MDIF (Measurement Data Interchange Format) format and data in the EEsof format [27].

Friendly User Interface

The sophisticated capabilities of HarPE are presented through a polished and friendly user-interface. It is menu-driven, easy to learn and pleasant to use. The following highlights some of the features.

- ▷ Logically designed menu-driven and window-driven commands with on-line help for every command.
- ▷ Mouse support facilitates a quick and fluent operation. However, the program is fully operational from a keyboard without a mouse.
- ▷ Clear circuit file syntax with flexible and user-designable data formats.
- ▷ Circuit file preprocessor featuring include files and macro definitions.
- ▷ Built-in dual-window, dual-file full screen editor with search and replace, cut and paste, buffered undo, macros, and more.
- ▷ Circuit file parser fully integrated into the screen editor. The location of any syntax error is automatically highlighted on screen by the editor.
- ▷ On-line User's Manual. You can add your own comments to the Manual, or copy syntax definitions and examples in the Manual and paste them directly into your circuit file, or define and search for arbitrary keywords (dynamic indexing).
- ▷ The program can be operated in several modes: fully interactive, driven by a macro file, silent (the graphics are suppressed) or as a separate background process.
- ▷ An automatically generated log file maintains a complete record of the operations, including clock time, user commands and messages.
- ▷ Graphics zooming through user-defined display ranges.
- ▷ Graphics documentation is supported by means of HPGL plotter files.

Other OSA Products and Services

Optimization Systems Associates Inc. has been developing state-of-the-art CAE software and technologies since 1983.

As a user of HarPE, you can rely on the expedient and professional support from our technical experts, including authors of the program.

OSA90/hope™ is a powerful general-purpose CAD/CAE system for linear and nonlinear simulation, nominal and yield optimization. It features Datapipe™ for high-speed data communication with external simulators to create an integrated CAD/CAE environment and provide multi-domain optimization solutions.

Empipe™ is a powerful and friendly software system for automated electromagnetic (EM) design optimization, driving the EM simulator *em*™ from Sonnet Software, Inc. Empipe allows you to designate geometrical and material parameters as candidate variables for optimization in an intuitive and friendly manner. Any arbitrary structures that you can simulate using *em* you can now optimize using Empipe!

OSA also provides consulting services for specialized CAE solutions.

For further information:
Phone 905 628 8228
Fax 905 628 8225

1.2 Installation

The installation of HarPE can be done by either a system administrator or a user.

In a multi-user environment, the system administrator should install HarPE so that it is accessible to all users. If, on the other hand, HarPE is to be used by a single user, then that user can carry out the installation himself/herself.

This section describes the procedure of loading the software. Only the person who performs the installation needs to follow this procedure. However, *every* user of HarPE must follow the instructions provided in Section 1.3 regarding setting UNIX environment variables and copying the demonstration examples.

Creating the OSA Installation Directory

First, you must allocate a directory in which the HarPE program and other associated files will be stored. We will refer to this directory as the OSA Installation Directory.

Starting with

HarPE Version 2.0
Empipe Version 3.1
OSA90/hope Version 3.1

all the CAD software from OSA will reside within the same OSA Installation Directory.

If you already have an installation of any of these OSA products (or a higher version), then the OSA Installation Directory should already exist on your computer. You should find that directory and install HarPE within it.

Otherwise, you can create the OSA Installation Directory using the UNIX command:

`mkdir` *<OSA Installation Directory>*

If you are a system administrator installing HarPE for multiple users, you should create the OSA Installation Directory from the root or another publicly accessible file system. For example, *<OSA Installation Directory>* can be "`/osa_home`" or "`/usr/osa_home`".

System Administrator: please make sure that the OSA Installation Directory has the "read" and "execute" permission for all HarPE users.

If you install HarPE from your user account, then the OSA Installation Directory will be created within your home directory. In this case, access to HarPE is by default restricted to yourself.

Loading the Software

Change directory to the OSA Installation Directory:

```
cd <OSA Installation Directory>
```

Then follow the instructions for the specific media in your case.

Sun SPARCstations SunOS 4.x, Floppy Disks

```
bar xvpfZ /dev/rfd0c
```

Sun SPARCstations SunOS 4.x, Tape

```
tar xvpf /dev/rst0
```

Sun SPARCstations Solaris 2.x, Floppy Disks

For Solaris 2.2 and higher, you need to disable the volume manager first. Log in as root and type

```
/etc/init.d/volmgt stop
```

Then load the software in the OSA Installation Directory:

```
cpio -id -H bar -I /dev/rdiskette
```

After the loading is finished, eject the floppy disk:

```
eject floppy
```

For Solaris 2.2 and higher, you may restart the volume manager (as root):

```
/etc/init.d/volmgt start
```

Sun SPARCstations Solaris 2.x, Tape

```
tar xvpf /dev/rmt/0m
```

HP 700 Series, DAT Tape

```
tar xvpf /dev/rmt/0m
```

Listing the Files Loaded

You can visually inspect the files loaded by typing

```
ls -la
```

which lists the files in the OSA Installation Directory.

You should be able to see the following subdirectory names listed:

```
bin
harpe_examples
harpemsg
```

There may also be other files if you have other OSA products.

You can also list the files in the "bin" subdirectory:

```
ls -la bin
```

The list should include the executable program

```
harpe
```

License File

The authorization of using HarPE is controlled by an encoded license file. Usually the appropriate license file is included in the package supplied to you.

Under some circumstances, such as when the software expiry date is being extended, you may be required to enter a new key in the license file.

In this case, change directory to the OSA Installation Directory, then type

```
cd harpemsg
mv license.osa license.osa.old
vi license.osa
```

Type "i" to enter the "insert" mode; type the key code *exactly* as provided; press <Esc> to exit the "insert" mode; type ":wq" to write to the file and quit the editor.

1.3 Setting UNIX Environment Variables

Every user of HarPE must follow the instructions in this section before attempting to run HarPE for the first time.

You must define a UNIX environment variable OSA_DIR to point to the OSA Installation Directory. If you do not know the name of that directory, please ask the person who carried out the software installation.

You should also modify the PATH environment variable so that you can execute HarPE from your own directory.

The UNIX environment variable DISPLAY must also be properly defined. For this you need to know the host name of your computer. To obtain the host name, type

```
uname -n
```

Instructions for csh Users

Include the following lines in the .login file in your home directory:

```
setenv OSA_DIR <OSA Installation Directory>
setenv DISPLAY host_name:0
set path=($path $OSA_DIR/bin)
```

Instructions for sh and ksh Users

Include the following lines in the .profile file in your home directory (HP VUE users should include these lines in the .vueprofile file instead):

```
OSA_DIR=<OSA Installation Directory>;  export OSA_DIR
DISPLAY=host_name:0;  export DISPLAY
PATH=$PATH:$OSA_DIR/bin;  export PATH
```

Effecting the Environment Changes

After completing the changes to the environment variables, you need to log out and then log in again in order for the changes to take effect.

Copying the Demo Examples

A set of HarPE examples are provided to you in the subdirectory

<OSA Installation Directory>/harpe_examples

Chapter 13 of this Manual utilizes a few selected demo examples in a step-by-step guided tutorial. You are encouraged to follow the tutorial on the computer. A separate manual entitled *HarPE Applications Illustrated* provides more details on the demo examples.

It is not advisable to practice with the original set of the examples for they may get inadvertently altered. You should make a copy of the examples.

First, create a suitable subdirectory within your own home directory. You can use the UNIX command:

mkdir *<example directory>*

where *<example directory>* is an arbitrary name, such as "harpe_examples".

Then change directory to *<example directory>* by typing

cd *<example directory>*

and copy the HarPE examples by typing

cp *<OSA Installation Directory>*/harpe_examples/* .

The total size of all the examples for HarPE Version 2.0 is about 1.3 megabytes.

1.4 Simulation Environment

In order to efficiently perform its tasks HarPE is designed to handle specific one- and two-port configurations with a single nonlinear device embedded in a linear environment reflecting parasitic or packaging effects. A number of predetermined circuit quantities that are represented by program internal variables can thus be made available to the user for requesting simulation outputs, defining optimization goals, or for postprocessing - all in a simple and convenient, but flexible enough manner. A typical overall schematic of a single-device circuit is shown in Fig. 1.1.

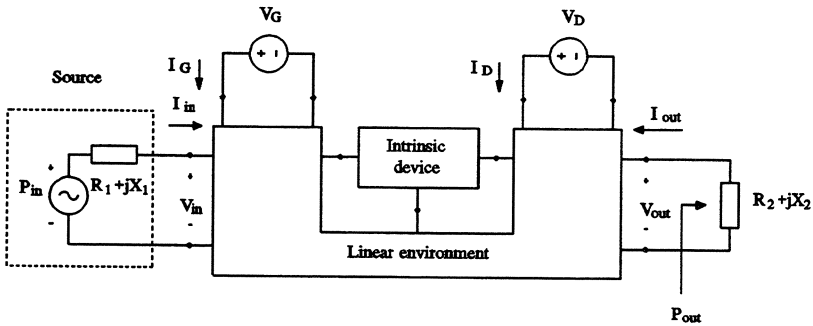


Fig. 1.1 Schematic representation of the device and its environment.

In Fig. 1.1, P_{in} represents the RF sinusoidal source described by its available power, R_1+jX_1 is the source impedance, P_{out} is the actual AC power delivered to the load R_2+jX_2, V_G generically represents the gate (base) DC bias voltage, and V_D the drain (collector) DC bias voltage. At AC, HarPE calculates the input and output voltages and currents V_{in}, V_{out}, I_{in} and I_{out}, output power P_{out}, and small-signal *S* parameters. At DC, HarPE calculates the currents at the bias ports, namely I_G and I_D. Fig. 1.1 shows a typical two-port circuit configuration with voltage bias sources. Variations of this basic topology include one- and two-port configurations and an option of base current bias source designed to accommodate measurement setups used for bipolar devices (see Chapter 5).

Bias

Bias refers to the bias voltages V_G and/or V_D, as shown in Fig. 1.1. To simplify the notation, the FET terminology "gate" and "drain" is used in a generic sense. FET and bipolar transistors require both bias voltages. For diodes, only V_G is needed.

Input Power and Output Power

Input Power refers to the available input power P_{in}, and Output Power refers to P_{out}, as illustrated in Fig. 1.1.

DC Simulation

DC data refers to computed or measured DC currents I_G and/or I_D, as shown in Fig. 1.1. The FET terminology "gate" and "drain" is used in a generic sense. For diodes, only I_D is defined.

For DC simulations only the bias voltages are taken into account. The input RF source is short-circuited, so it becomes terminated simply with R_1. The output port is terminated with R_2. For perfectly isolated DC and AC circuits a convenient short-cut description is available (see Chapters 5 and 7).

Small-Signal Simulation

For small-signal simulations the intrinsic device is linearized at the operating point determined by the bias voltages. Then the bias sources are short-circuited and the one- or two-port linear circuit defined by the input and output ports is simulated. The input power does not play any role in these calculations. Also, the actual terminations are not reflected in the resulting S parameters, unless the circuit is terminated with the reference impedance used in the calculations. Only insertion loss considers the actual terminations.

Harmonic Balance Simulation

For harmonic balance analysis, a single-tone RF excitation at the fundamental frequency is considered. Both bias voltages and the input power, as well as actual port terminations are simultaneously taken into account.

1.5 Interactive Operations

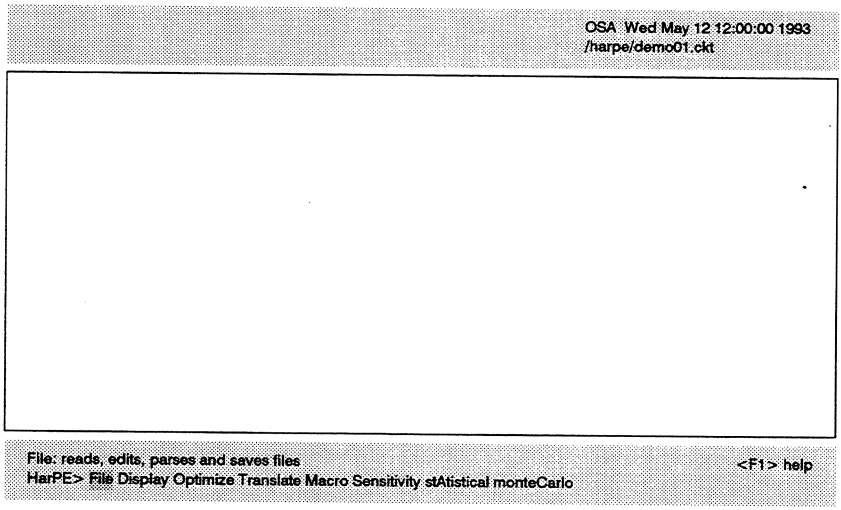
This section describes the interactive operations of HarPE. (You can also run HarPE from a macro command file, see Chapter 12). In the interactive mode, you give instructions to HarPE through the keyboard or mouse.

Keyboard Entries

Throughout this manual, keyboard entries are enclosed in a pair of brackets "<>". For instance, <ESC> represents the key labelled ESC, and <Y> represents the key producing the letter "Y".

Menus

When the program is ready for the next instruction, an appropriate menu of available options is shown at the bottom of the screen. A typical screen may look like this:



The leading field on the second menu line indicates the level of the menu, followed by the list of available options. One of the options is highlighted as the current or default option. On the line above it is a brief description of the highlighted option.

On-Line Help

To obtain on-line help messages for the highlighted option, press the <F1> key.

Menu Hierarchy

The different options (features) of HarPE are organized into a hierarchy of menus, which is summarized in Table 1.1 and illustrated in Figure 1.2.

TABLE 1.1 HarPE MENU OPTIONS

Menu Option	Brief Description
HarPE.File	reads, edits, parses and saves files
HarPE.Display	simulates and displays the responses
HarPE.Optimize	optimizes circuit model parameters
HarPE.Translate	translates model into external format
HarPE.Macro	executes a macro definition file
HarPE.Sensitivity	displays parameter sensitivities
HarPE.stAtistical	statistical modeling from multi-device measurements
HarPE.monteCarlo	performs and displays Monte Carlo analyses
HarPE.Learn	learns user inputs to create a macro command file
HarPE.Preprocess	extracts extrinsic parameters from cold measurements
HarPE.Display.specTrum	displays frequency-domain spectra
HarPE.Display.Waveform	displays time-domain waveforms
HarPE.Display.Psweep	displays power sweep (sweep fundamental input power)
HarPE.Display.Fsweep	displays frequency sweep (sweep fundamental frequency)
HarPE.Display.DC	displays DC IV curves and linearized model parameters
HarPE.Display.S-par	displays small-signal scattering parameters
HarPE.Display.Error	displays errors between model and measured responses
HarPE.Display.Xsweep	displays response vs. arbitrary parameter sweep
HarPE.Statistical.Extract	extracts multiple device models from measurements
HarPE.Statistical.Sample	plots statistical diagrams from the SAMPLE Block
HarPE.Statistical.Postprocess	postprocesses extracted statistical model
HarPE.Statistical.Data	plots statistical diagrams of the measured data
HarPE.Statistical.Match	performs direct statistical modeling
HarPE.MonteCarlo.Histogram	plots histogram of a single response
HarPE.MonteCarlo.RunChart	plots run chart of a single response
HarPE.MonteCarlo.Sweep	plots multiple sweep curves
HarPE.MonteCarlo.Yield	displays yield estimated by Monte Carlo analysis
HarPE.MonteCarlo.Cumulative	plots cumulative probability distributions of a response
HarPE.Statistical.Sample.Histogram	histogram of a single statistical variable
HarPE.Statistical.Sample.Scatter	scatter diagram between two statistical variables
HarPE.Statistical.Sample.Cumulative	cumulative probability distribution of a variable
HarPE.Statistical.Data.Histogram	histogram of the measured data
HarPE.Statistical.Data.Cumulative	cumulative probability distribution of the data

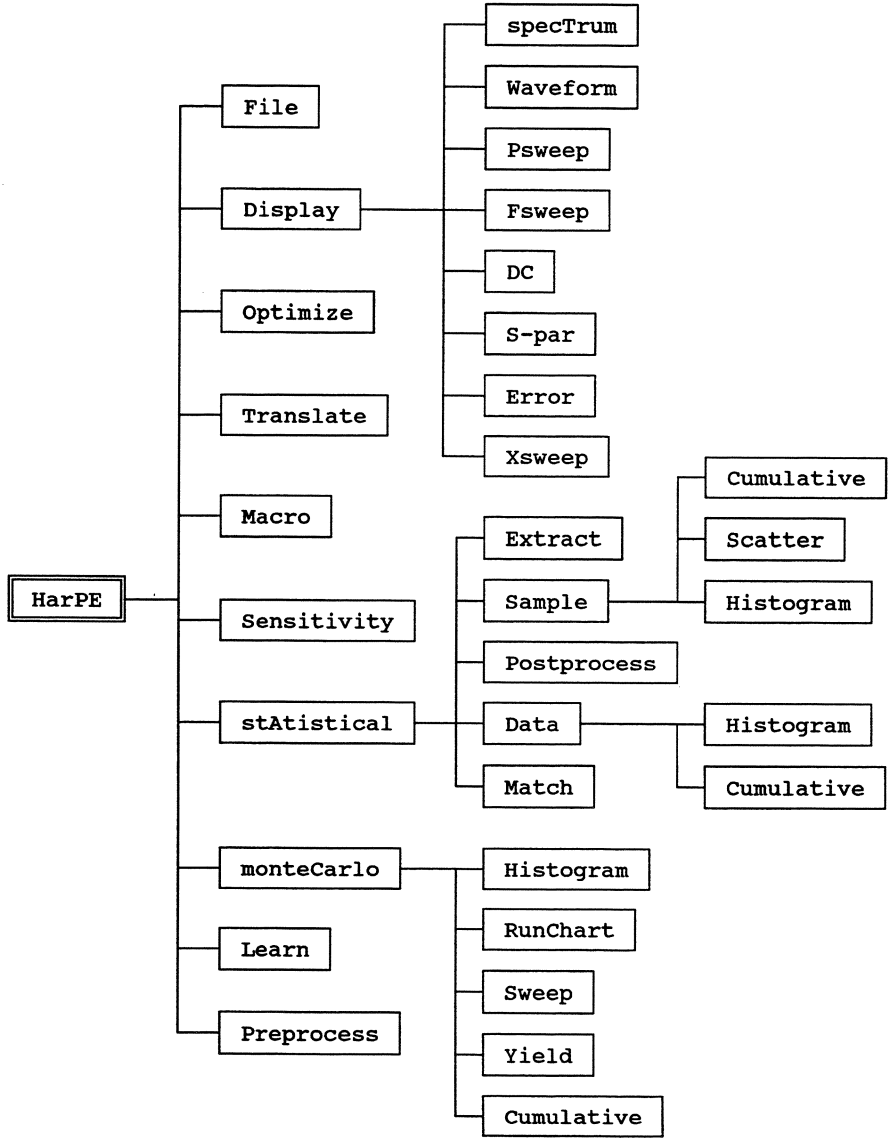


Fig. 1.2 HarPE menu overview.

Select Menu Options

You select options from a menu by one of the following methods:

- use the left and right arrow keys to move the cursor to highlight the desired option, and then press the <ENTER> key to select it
- use the mouse to move the cursor to highlight the desired option, and then click the left-hand mouse button to select it
- type the first capital letter in the name of the desired option, e.g., press <D> to select the option Display

Note that a menu option can be identified by the first capital letter in its name which may not be the first letter, if the name of another option begins with the same letter. For example, the options Macro and monteCarlo are identified by <M> and <C>, respectively.

Exit and Cancelation

To exit the current menu, use the <ESC> key or the right-hand mouse button. This will return you to the next higher-up menu level. If you are already at the top level (i.e., the HarPE> level), you will see this prompt:

```
Terminate the program Y/<N>?
```

If you respond with <Y> or the left-hand mouse button, HarPE will terminate and return to the operating system.

Throughout the program, the <ESC> key and the right-hand mouse button always signal cancelation of a command or operation. The <ENTER> key and the left-hand mouse button signal confirmation of a command or acceptance of an option. When responding to a "Yes" or "No" prompt, the left-hand button means "Yes", and the right-hand button means "No".

Windows

Once you have selected a menu option, a pop-up window may appear, showing the parameters related to that particular menu option. The following illustrates a typical pop-up window:

```
Output form:    Graphical
Zoom scale:     <TAB> to set scale
Sweep set:      sweep set 1
X-axis:         Input Power
Y-axis:         Pout & IDO
Sweep set:      1 of 2
Gate bias:      -1.74V
Drain bias:     4V
Frequency:      6GHZ
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You may be able to modify some of the parameters, depending on the circumstances. Using the arrow keys or the mouse, you can move the cursor to highlight the parameter of interest. You can also select a desired parameter by typing the first capital letter in the name of the parameter.

Note that a parameter can be identified by the first capital letter in its name which may not be the first letter, if the name of another parameter begins with the same letter. For example, the parameters Ymin and yMax are identified by <Y> and <M>, respectively.

On-Line Help

You can request on-line help messages regarding the parameters shown in the pop-up window using the <F1> key.

Change Parameters in Window

As you move the cursor to highlight different parameters, one of the following messages will be shown at the bottom of the window.

- **<ENTER> = go or <ESC> = cancel.** You see this message when the cursor is at the "Ready to go" line. Press <ENTER> or the left-hand mouse button to accept the current parameter setting as shown in the window. Press <ESC> or the right-hand mouse button to exit from (cancel) the window.

- **Predefined feature.** This indicates that the item highlighted represents a predefined parameter which cannot be modified by the user.

- **Toggle data with <LEFT>/<RIGHT>.** This means that the parameter highlighted has multiple values to choose from. Use the left and right arrow keys to toggle between the different choices, or press the left-hand mouse button to select from a pull-down menu. For example, the parameter "Output form" may offer you three choices, namely Graphical, Numerical and Numerical Append.

- **Only one value supplied.** This indicates that the item highlighted represents a user-definable parameter. But in that particular case, there is only one value available for it. For example, the parameter "Y-axis" allows you to select different responses for display. But if you have defined only one response, then you will see this message.

- **Specify a value for this item.** This means that the parameter highlighted is user-definable and you can assign it a new value if you wish. For example, the "zoom" feature allows you to specify the "Ymin", "yMax", "Xmin" and "xmAx" parameters for the graphical display. You can change any of these parameters by typing an appropriate numerical value.

- **Invoke this feature with <TAB>.** This indicates that the item highlighted represents a feature which you can activate by pressing <TAB> or the left-hand mouse button. For example, you can invoke the graphics zoom feature by pressing <TAB> at the "Zoom scale" line. Then a separate pop-up window will appear to allow you to specify further details related to the activated feature.

1.6 A Sample Session

This sample session is intended to help you familiarize yourself with the basic operations of HarPE. You are shown, step by step, how you might proceed through the program from beginning to end. Remember, wherever the <ENTER> key is referred to, you can choose to use the left-hand mouse button instead. Similarly, you may use the right-hand mouse button instead of the <ESC> key.

1 Type **harpe** at the system prompt to start the program. The HarPE title screen is displayed. Press any key to continue.

2 You need to select an input file. We will use demo01.ckt for illustration. You select a file by (1) moving the cursor to highlight file demo01.ckt and pressing <ENTER>, or (2) pressing <N> to indicate that you would like to select the file by name, and then typing **demo01.ckt** followed by <ENTER>.

3 Now the input file demo01.ckt is loaded into the screen editor. You can view the contents of the file by moving the cursor. Click the left-hand mouse button to activate the editor menu and select "Exit from editor" (by clicking the left-hand mouse button). You can also press <F7> to exit from the editor. Upon exit from the editor, the input file will be parsed.

4 Now you are at the HarPE> menu. Commands available from this menu are File, Display, Optimize, etc. Move the cursor to highlight Display and press <ENTER> to select the display option.

5 Now you are at the HarPE.Display> menu. Commands available from this menu are specTrum, Waveform, etc. Move the cursor to highlight S-par and press <ENTER> to select the *S* parameter display (Smith Chart and polar plot).

6 A pop-up window appears. Press <ENTER> to select the default bias point and frequency. A Smith chart and two polar plots showing the *S* parameters appear on the screen. The computed responses are shown as smooth curves, and the measured data is super-imposed on the display as discrete points.

7 Press <ESC> to go back to the HarPE> menu. Move the cursor to highlight File and press <ENTER>. You are once again in the screen editor.

8 If you wish to try another example, click the left-hand mouse button to activate the editor menu and select "Read a new file" (you can also press <Ctrl-I> to read a file). Read the desired input file and then repeat the steps starting from step **2.**

9 If you wish to terminate the sample session and exit the HarPE program, press <ESC>. You will see the prompt "`Terminate the program (Y/<N>)?`". Press <Y>.

1.7 Size Limits

The following is a list of the technical and syntactical size limits of HarPE. If any of these limits presents a serious problem for your application, please contact OSA's technical support.

Maximum number of harmonics for simulation and optimization	8
Maximum number of waveform time samples	21
Maximum number of steps in a single sweep range	1024
Maximum number of points in a single graphical plot	1024
Maximum number of optimization variables	50
Maximum number of statistical variables	50
Maximum number of user-defined responses per sweep set	16
Maximum length of a single circuit or data file	256K
Maximum length of a single macro command file	16K
Maximum number of combinations of frequency, bias and input power levels for nonlinear optimization	64
Maximum number of combinations of frequency and bias points for small-signal optimization	256
Maximum sample size for statistical measurement data	512

1.8 References

[1] J.W. Bandler and S.H. Chen, "Circuit optimization: the state of the art", *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 424-443.

[2] J.W. Bandler, S.H. Chen, S. Ye and Q.J. Zhang, "Integrated model parameter extraction using large-scale optimization concepts", *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 1629-1638.

[3] J.W. Bandler, Q.J. Zhang, S. Ye and S.H. Chen, "Efficient large-signal FET parameter extraction using harmonics", *IEEE Trans. Microwave Theory Tech.*, vol. 37, 1989, pp. 2099-2108.

[4] J.W. Bandler, Q.J. Zhang and R.M. Biernacki, "A unified theory for frequency domain simulation and sensitivity analysis of linear and nonlinear circuits", *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 1661-1669.

[5] J.W. Bandler, Q.J. Zhang, J. Song and R.M. Biernacki, "FAST gradient based yield optimization of nonlinear circuits", *IEEE Trans. Microwave Theory Tech.*, vol. 38, 1990, pp. 1701-1710.

[6] J.W. Bandler, R.M. Biernacki, S.H. Chen, J. Song, S. Ye and Q.J. Zhang "Analytically unified DC/small-signal/large-signal circuit design", *IEEE Trans. Microwave Theory Tech.*, vol. 39, 1991, pp. 1076-1082.

[7] J.W. Bandler, R.M. Biernacki, S.H. Chen, J. Loman, M. Renault and Q.J. Zhang, "Combined discrete/normal statistical modeling of microwave devices," *Proc. European Microwave Conf.* (Wembley, England), 1989, pp. 205-210.

[8] J.W. Bandler, R.M. Biernacki, S.H. Chen, J. Song, S. Ye and Q.J. Zhang, "Statistical modeling of GaAs MESFETs," *IEEE MTT-S Int. Microwave Symp. Dig.* (Boston, MA), 1991, pp. 87-90.

[9] J.W. Bandler, Q.J. Zhang, J. Song and R.M. Biernacki, "Yield optimization of nonlinear circuits with statistically characterized devices," *IEEE MTT-S Int. Microwave Symp. Dig.* (Long Beach, CA), 1989, pp. 649-652.

[10] J.W. Bandler, S. Ye, Q. Cai, R.M. Biernacki and S.H. Chen, "Predictable yield-driven circuit optimization," *IEEE MTT-S Int. Microwave Symp. Dig.* (Albuquerque, NM), 1992, pp. 837-840.

[11] J.W. Bandler, W. Kellermann and K. Madsen, "A superlinearly convergent minimax algorithm for microwave circuit design", *IEEE Trans. Microwave Theory Tech.*, vol. MTT-33, 1985, pp. 1519-1530.

[12] J.W. Bandler, S.H. Chen and S. Daijavad, "Microwave device modeling using efficient ℓ_1 optimization: a novel approach", *IEEE Trans. Microwave Theory Tech.*, vol. MTT-34, 1986, pp. 1282-1293.

[13] J.W. Bandler, W. Kellermann and K. Madsen, "A nonlinear ℓ_1 optimization algorithm for design, modeling and diagnosis of networks", *IEEE Trans. Circuits and Systems*, vol. CAS-34, 1987, pp. 174-181.

[14] J.W. Bandler, S.H. Chen, R.M. Biernacki, L. Gao, K. Madsen and H. Yu, "Huber optimization of circuits: a robust approach," *IEEE Trans. Microwave Theory Tech.*, vol. 41, 1993, pp. 2279-2287.

[15] W.R. Curtice and M. Ettenberg, "A nonlinear GaAs FET model for use in the design of output circuits for power amplifiers", *IEEE Trans. Microwave Theory Tech.*, vol. MTT-33, 1985, pp. 1383-1394.

[16] A. Materka and T. Kacprzak, "Computer calculation of large-signal GaAs FET amplifier characteristics", *IEEE Trans. Microwave Theory Tech.*, vol. MTT-33, 1985, pp. 129-135.

[17] H. Statz, P. Newman, I.R. Smith, R.A. Pucel and H.A. Haus, "GaAs FET device and circuit simulation in SPICE", *IEEE Trans. Electron Devices*, vol. ED-34, 1987, pp. 160-169.

[18] H.K. Gummel and H.C. Poon, "An integral charge control model of bipolar transistors", *Bell Syst. Tech. Journal*, vol. 49, 1970, pp. 827-852.

[19] M.A. Khatibzadeh and R.J. Trew, "A large-signal, analytic model for the GaAs MESFET", *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 231-238.

[20] J.W. Bandler, Q.J. Zhang and Q. Cai, "Nonlinear circuit optimization with dynamically integrated physical device models", *1990 IEEE MTT-S Int. Microwave Symp. Digest* (Dallas, TX, 1990), pp. 303-306.

[21] J.W. Bandler, R.M. Biernacki, Q. Cai, S.H. Chen, S. Ye and Q.J. Zhang, "Integrated physics-oriented statistical modeling, simulation and optimization," *IEEE Trans. Microwave Theory Tech.* vol. 40, 1992, pp. 1374-1400.

[22] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design.* New York, NY: Van Nostrand Reinhold, 1983, pp. 309-312.

[23] J.M. Golio, *Microwave MESFETs and HEMTs*, Norwood, MA: Artech House, 1991.

[24] D. Halchin, M. Miller, M. Golio and S. Tehrani, "HEMT models for large signal circuit simulation," *IEEE MTT-S Int. Microwave Symp. Digest* (San Diego, CA), 1994, pp. 985-988.

[25] H. Wang, C. Algani, A. Konczykowska and W. Zuberek, "Temperature dependence of DC currents in HBT," *IEEE MTT-S Int. Microwave Symp. Digest* (Albuquerque, NM), 1992, pp. 731-734.

[26] J.W. Bandler, R.M. Biernacki, Q. Cai and S.H. Chen, "A novel approach to statistical modeling using cumulative probability distribution fitting," *IEEE MTT-S Int. Microwave Symp. Dig.* (San Diego, CA), 1994, pp. 385-388.

[27] *Libra Reference Manual*, EEsof Inc., Westlake Village, CA 91362.

[28] *Microwave Harmonica User's Manual*, Compact Software Inc., Paterson, NJ 07504.

2

File Editor

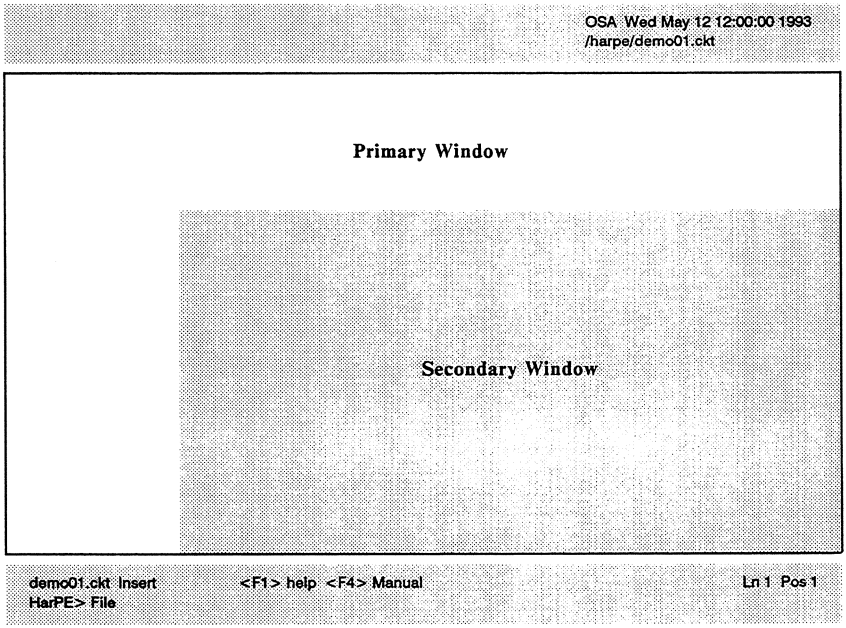
2

File Editor

2.1 Overview

HarPE has a built-in full screen ASCII text file editor. Primarily, it allows you to create, retrieve, modify and save circuit files for HarPE.

When you run HarPE, it automatically starts in the editor mode. Otherwise, you invoke the editor by selecting the menu option HarPE.File. The editor screen looks like this:



You can edit two different files concurrently in two separate screen windows. The two editor windows are designated as the **Primary Window** and **Secondary Window**, and the files in these windows are correspondingly designated as the **Primary File** and **Secondary File.**

Switch between Two Windows

You switch (toggle) between the two windows (files) by clicking the left-hand mouse button to activate the editor menu and selecting "Toggle file". You can also use the <F8> key. Text can be copied between the two windows (files).

Primary File Is Circuit File

The Primary File is assumed to be the circuit file, i.e., its contents are relevant to the operation of HarPE and written in compliance with the syntax of HarPE. Extra safeguards are built-in to protect it from being accidentally discarded or overwritten. For instance, if the Primary File has been modified but not yet saved, and you want to read in a new file from the disk into the Primary Window, the editor will prompt you for confirmation.

Secondary File Is Working Area

The Secondary Window is treated as a "working area" or "scratch pad". While working on the Primary File, you can use the secondary window to look up information stored in other files. For instance, you may want to copy some text from another circuit file or data file.

The Secondary File is not automatically protected. It may be overwritten during some operations. For instance, the on-line User's Manual may be displayed in the Secondary Window upon request, and the circuit file parser may use it as a working space. Therefore, if the Secondary File is of importance, save it explicitly as soon as you finish editing.

Function Keys

The screen editor features search and replace, cut and paste, undo, editor macros for repeated operations, text printing, and more. The editor is integrated with the file parser to make syntax error detection and correction convenient and friendly. You can also access the on-line User's Manual within the editor.

The various editor function keys and the features they represent are listed in Table 2.1.

TABLE 2.1 EDITOR FUNCTION KEYS

Key	Description
<LEFT>	move cursor to the left by one character
<RIGHT>	move cursor to the right by one character
<UP>	move cursor up by one line
<DOWN>	move cursor down by one line
<PGUP>	move cursor up by one page
<PGDN>	move cursor down by one page
<HOME>	move cursor to the beginning of the line
<Ctrl-HOME>	move cursor to the end of the line
<Ctrl-UP>	move cursor to the beginning of the file
<Ctrl-DOWN>	move cursor to the end of the file
<Ctrl-LEFT>	move cursor to the left by one word
<Ctrl-RIGHT>	move cursor to the right by one word
<F8>	toggle between the primary and secondary windows
<F1>	request help messages
<F4>	request for on-line User's Manual
<Ctrl-I>	input (read in) a new file to the editor
<Ctrl-S>	save the file or the marked block to a disk file
<Ctrl-N>	discard the current file to start a new (empty) file
<F5>	change the default path (working directory)
<F7>	exit the editor and invoke the circuit file parser
<ESC>	terminate the program or cancel a command
<DEL>	delete the character under cursor or the marked block
<BACKSPACE>	delete the character to the left of the cursor
<Ctrl-D>	delete from cursor to the end of line
<Ctrl-W>	delete the word under cursor
<Ctrl-U>	undo the last sequence of consecutive deletions
<Ctrl-X>	delete trailing blanks throughout the file (trimming)
<INS>	toggle between insert and typeover modes
<F6>	toggle between edit and read-only modes
<F2>	mark/unmark a block of text for copying or moving
<F3>	copy the marked block, or paste the copied text
<Ctrl-F>	search forward for a specified string
<Ctrl-B>	search backward for a specified string
<Ctrl-R>	replace a string and with another string
<F9>	begin/end the definition of an editing macro
<F10>	execute the macro defined
<Ctrl-P>	print the file or the marked block of text
<Ctrl-Y>	change the display color map

Pull-Down Menu

In the editor, you can click the left-hand mouse button to activate a pull-down menu of the options listed in Table 2.2. To select an option, move the cursor to highlight the desired option and click the left-hand mouse button. To cancel the menu without selecting an option, use the right-hand mouse button.

TABLE 2.2 EDITOR PULL-DOWN MENU

Option	Equivalent Function Key
Exit from editor	<F7>
Terminate program	<ESC>
Help	<F1>
Toggle file	<F8>
Read a new file	<Ctrl-I>
Save file to disk	<Ctrl-S>
Search	<Ctrl-F>
Replace	<Ctrl-R>
Change directory	<F5>
Print text	<Ctrl-P>
Clear file buffer	<Ctrl-N>
On-line Manual	<F4>
Trim file	<Ctrl-X>
New color map	<Ctrl-Y>

On-Line Manual

Press <F4> to request the on-line User's Manual. A list of chapter titles will appear on the screen for you to make a selection.

The on-line Manual will be read into the Secondary Window and overwrite whatever text may already exist in the Secondary Window. For this reason, the <F4> key is ignored in the Secondary Window, since the window itself is to be used as the working space. In other words, the editor will respond to your request for the on-line Manual only when you are in the Primary Window.

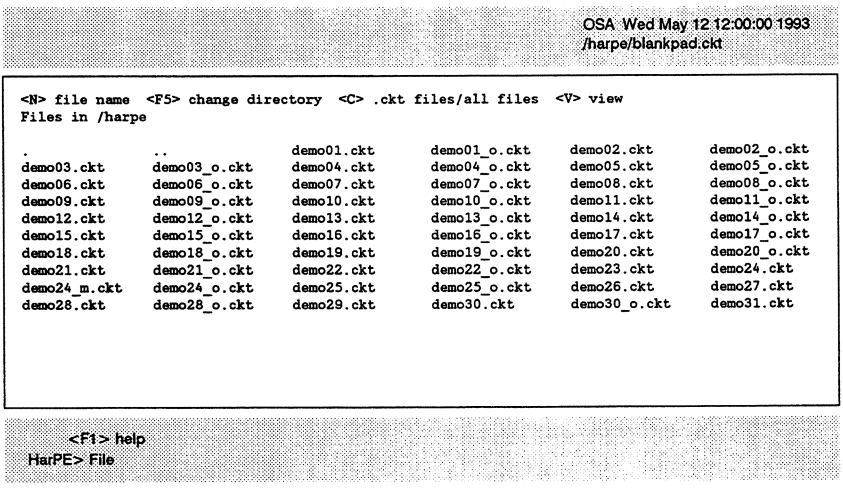
The on-line User's Manual is stored as an ASCII text file and as such it does not contain the graphical information in the printed Manual. Once the Manual is read into the editor, it can be manipulated just like any other files. For example, you can search for a keyword or phrase of interest. It is possible for you to edit the on-line Manual, adding your own notes, for instance, and save it as a personalized Manual. Care must be exercised, though, if you are sharing the same Manual with other users.

2.2 Read and Save Files

Input (Read) a File: <Ctrl-I>

To read a disk file into the editor, use <Ctrl-I>. When a file is being edited, the editor does not actually change the original file on the disk. Rather, it reads (copies) the contents of the file into a memory buffer.

To help you select the file to be read, the editor lists on the screen the files in the current working directory, as illustrated in the following (when you start HarPE, it automatically goes to this screen):



You move the cursor to highlight the desired file, and press the <ENTER> key or the left-hand mouse button to select the file for the editor to read.

To select a file by name instead of by moving the cursor, you press <N> and then type the exact file name followed by <ENTER>.

Be aware that file names on UNIX workstations are case sensitive, e.g., demo01.ckt and DEMO01.CKT do not refer to the same file.

By default, the editor lists the files in the directory from which you run HarPE (which is called the "working directory"). To read a file outside the working directory, press <N> and then enter the full path name, such as "/user/john/project1/ex01.ckt".

To change the working directory, press <F5>. You will be prompted for the name of the new working directory.

To create a new file from scratch, press <N> and enter a unique name (i.e., different from any existing file names). Since the editor cannot find a file by the name specified, you will be prompted:

```
Create File (Y/<N>):
```

If you respond with <Y>, the editor will create a new (blank) file with the name specified. Answer <N> if you wish to re-type the name.

List .ckt files only: when the editor lists the files in the working directory, it can list all the files or list only those files with the extension .ckt (the recommended extension for HarPE circuit files). You can switch (toggle) between these two listings by pressing <C>.

File preview is a convenience feature. If you do not remember the contents of a file from its name, the "preview" feature may be helpful. Move the cursor to highlight a file and press <V>, and the editor will display the first few lines of the file in a small window for you to preview. If it is the file you want, you can then read it into the editor, otherwise you can move on to preview other files.

Primary file is protected from being unintentionally overwritten. The file newly read by the editor replaces (overwrites) the existing file in the current editor window. In the Primary Window, the existing file is automatically protected if it has been changed and not yet saved. You will be prompted for confirmation:

```
Discard the modified file (Y/<N>)?
```

Maximum file size is limited to 256K bytes. The virtual size of the circuit file can be extended by means of include files, see Chapter 3.

Save File: <Ctrl-S>

To save the file in the current editor window to the disk, press <Ctrl-S>. You will be prompted for a file name, with the original file name given as the default.

If you save the edited file under the original name or the new name you specified coincides with the name of another file which already exists on the disk, then the file existing on the disk will be renamed as a backup file. Assuming the name of the file to be saved is xxxxxx.yyy, the backup file will be named xxxxxx.bak. If a previous backup file xxxxxx.bak already exists, it will be overwritten (i.e., only one backup version is kept).

Change Directory: <F5>

The directory from which you start HarPE becomes the working directory. By default, the editor reads files from and saves files to this directory.

To change the working directory, press <F5>. You will be prompted for the name of the new working directory.

Exit the Editor: <F7>

To exit from the current editor window, press <F7>.

In the Primary Window, <F7> automatically invokes the integrated circuit file parser, because the Primary File is assumed to be a HarPE circuit file, i.e., it contains information relevant to the operation of HarPE and complies with the syntax rules of HarPE. If a syntax error is detected during parsing, the editor will display an error message, pinpoint the location of the erroneous text, and automatically position the cursor so that you can make the correction conveniently.

In the Secondary Window, <F7> switches the editor to the Primary Window.

Terminate HarPE: <ESC>

To exit the editor and terminate HarPE, press <ESC> in the Primary Window. You will be prompted for confirmation.

In the Secondary Window, <ESC> switches the editor to the Primary Window.

Clear the Buffer for a New File: <Ctrl-N>

Press <Ctrl-N> if you wish to discard the file in the current editor window, empty the file buffer and start with a new blank file ("start from scratch").

The Primary File is safeguarded from accidental loss. If it has been modified but not yet saved, you will be prompted for confirmation.

2.3 Delete and Undelete Text

Delete Character: <DEL>, <BACKSPACE>

The <DEL> key deletes a single character under the cursor.

The <BACKSPACE> key deletes a single character before (to the left of) the cursor.

Delete to the End of Line: <Ctrl-D>

The <Ctrl-D> key deletes all text from the cursor position to the end of the line.

To delete a complete line, position the cursor to the beginning of the line, press <Ctrl-D> to delete the visible text, and then press <DEL> to delete the new-line character.

Delete Word: <Ctrl-W>

The <Ctrl-W> key deletes the word under the cursor. Words are delimited by spaces, new-line characters, commas, periods, colons, parentheses, etc.

Undelete Text: <Ctrl-U>

The editor automatically buffers the text deleted by the <DEL>, <BACKSPACE>, <Ctrl-D> and <Ctrl-W> keys. The <Ctrl-U> key retrieves the text from the undo buffer and inserts it at the current cursor position (which may be different from the original position of the deleted text).

The editor uses only one undo buffer. Only the latest deletion or sequence of consecutive deletions is buffered.

Consecutive deletions refer to a sequence of consecutive <DEL>, <BACKSPACE>, <Ctrl-D> and/or <Ctrl-W> keys, uninterrupted by any other keystrokes. For example, you press consecutively <Ctrl-D>, <DEL>, <DEL>, <DEL>, then all the text deleted by these keystrokes is saved in the undo buffer. On the other hand, if you press <DEL>, <RIGHT>, <DEL>, then only the character deleted by the last <DEL> is buffered.

Use undelete for copying. The "undelete" feature is primarily useful for recovering text deleted by mistake. However, you can also use it as a sort of "quick cut and paste" function for copying or moving a small amount of text.

To copy text, you "delete" the text to be copied, undelete it immediately using <UNDO>, move the cursor to where you wish to copy the text to, and press <Ctrl-U> again.

To move text, you delete the text to be moved, move the cursor to the new location, and press <Ctrl-U>.

⚠ **The undo buffer size** (i.e., the amount of text that can be undeleted) is limited to 512 bytes. This is especially important if you intend to use <Ctrl-U> for copying or moving text.

Trim Trailing Blanks: <Ctrl-X>

Trailing blanks refer to blanks (spaces) between the last significant character on a text line and the end-of-line. Trailing blanks inadvertently introduced during file editing take up unnecessary space. If you wish to remove all the trailing blanks from the file in the current editor window ("trim the file"), press <Ctrl-X>.

☝ You cannot undo the "trim" command.

2.4 Editing Modes

Insert and Typeover: <INS>

By default, the text you typed at the keyboard is inserted at the cursor position, and the file size increases by the number of characters you have typed. This is called the **Insert Mode** and it is indicated at the bottom of the editor window by the status word **Insert**.

You can switch to the **Typeover Mode** using the <INS> key. In the typeover mode, the text you typed at the keyboard replaces (overwrites) the text under the cursor. This is indicated at the bottom of the editor window by the status word **Typeover**.

The <INS> key toggles between Insert Mode and Typeover Mode.

Read-Only Mode: <F6>

If the file in the current editor window is for viewing only and you do not wish it to be altered in any way, press <F6>. This switches the editor into the **Read-Only Mode** which is indicated at the bottom of the editor window by the status word **Read-Only**.

No text can be added to or deleted from the file in the Read-Only Mode. To switch back to the normal editing mode, press <F6> again.

☝ The editing modes of the Primary and Secondary windows can be independently assigned.

2.5 Block Operations

Mark a Block: <F2>

To mark a block of text in the file for coping or moving, position the cursor to the one end of the text block and press <F2>. Now as you move the cursor towards the other end of the block, the text within the block (i.e., between the two ends) is shown in inverse color.

Copy the Marked Block: <F3>

After you have marked one end of the text block using <F2>, move the cursor to the other end of the block, and press <F3>. The marked text is copied to the editor's block buffer.

Then, move the cursor to the location where you wish to copy the text block to, and press <F3>. The editor retrieves the text in the block buffer and inserts it at the current cursor position (also known as "paste").

Note that there is only one block buffer and the copying command overwrites any text that may already exist in the buffer.

Delete or Move the Marked Block: <DEL>

To delete or move a block of text, mark one end of the block using <F2>, move the cursor to the other end, and press <DEL>. The marked text will be removed (deleted) from its current position in the file and copied to the editor's block buffer.

If you wish to move the text block, point the cursor to the new location for the text block, and press <F3>. The editor retrieves the text in the block buffer and inserts it at the current cursor position.

Unmark a Block: <F2>

If after you marked one end of a text block, you change your mind and do not wish to copy or move the text, you can unmark the text by pressing <F2> again.

Save a Block as a Disk File: <Ctrl-S>

You can save a marked block of text as a disk file using <Ctrl-S>. You will be prompted for a file name (☞ also see Section 2.2).

2.6 Search and Replace

Search for a String: <Ctrl-F>, <Ctrl-B>

The <Ctrl-F> key instructs the editor to scan through the file from the current cursor towards the end of the file for a specific text string. The <Ctrl-B> key performs the same function but in the backward direction, i.e., from the current cursor towards the beginning of the file.

After pressing <Ctrl-F>, you specify the string to search for at the prompt:

```
Search ->:
```

The corresponding prompt for <Ctrl-B> is

```
Search <-:
```

The arrow in the prompt indicates the search direction: -> indicates search forward and <- indicates search backward. You can reverse the search direction while you are entering the string to search for. Use the <UP> cursor key to change from "search forward" to "search backward". Use the <DOWN> cursor key to change from "search backward" to "search forward". The arrow in the prompt will change to indicate the new search direction.

Carriage-return and tab characters can be included in the search string. A carriage-return is shown as [CR] and a tab is shown as [TAB]. The inclusion of carriage-return and tab characters may help you to uniquely identify the search string. For example, the search string "[CR]A" tells the editor to look for a line beginning with the letter "A".

After you have completed typing the search string, press <Ctrl-F>, <Ctrl-B> or the left-hand mouse button. Do not use <ENTER>, because it will be treated as the carriage-return [CR] and included in the search string.

Lower case letter in the search string you specified will match the same letter of either lower or upper case in the file text. For instance, an "a" will match an "a" or an "A".

Upper case letter matches only an upper case letter. For example, an "A" in the search string will match an "A" but not an "a" in the file text.

For case-insensitive search, use all lower-case letters in the search string.

Repeated search. The current search string will be offered as the default for the next time you invoke the search feature, hence to repeat the search for the same string, just press <Ctrl-F> (or <Ctrl-B>) twice.

Utilize the block buffer. If you have a string in the file and wish to search for other occurrences of the same string, you can utilize the block buffer (☞ Section 2.5) to capture the search string. Press <F2> to mark the search string in the file, then press <F3> to

copy it to the editor's block buffer. Press <Ctrl-F>, and when you are prompted for the search string, press <F3> to retrieve the string from the block buffer. This is particularly convenient if the string is long and/or complicated.

Replace a String by Another: <Ctrl-R>

To search for a specific string in the file and replace it with another string, press <Ctrl-R>. First, specify the string to search for at the prompt:

```
Search ->:
```

☞ Details on how to specify the search string are given earlier in this Section.

Then, you will be prompted for the replacement (substitution) string:

```
Replace with ->:
```

After that you will see this prompt:

```
With confirmation <Y>/N?
```

If you respond with <N>, then all the occurrences of the search string along the search direction will be replaced without further prompt. If your answer is <Y>, then for each and every occurrence of the search string, you will be asked for confirmation:

```
Replace <Y>/N?
```

Press <Y> or <ENTER> to confirm the replacement, press <N> to skip over that particular occurrence and continue searching for the next one, or press <ESC> to terminate the search and replace process.

2.7 Editor Macros

The editor macro feature can be likened to a "typing robot". It learns (records) an arbitrary sequence of keystrokes and then replays the sequence of keystrokes on command as if you were repeating the keystrokes at the keyboard. It can be very useful in saving you the trouble of duplicating complicated sequence of editing commands.

Record a Macro: <F9>

The <F9> key switches on and off the editor macro "recording" ("learning") mode.

To create (define) an editor macro, you press <F9> to switch on the "recording" mode, which is indicated by the status word **DefMacro** shown at the bottom of the editor window.

In the recording mode, the keystrokes you type at the keyboard are stored in the editor macro buffer at the same time as the corresponding functions are carried out by the editor.

When you have completed the macro definition, press the <F9> key again to switch off the recording mode.

Note that there is only one macro buffer and the recording command overwrites the last macro definition in the buffer if one exists.

Replay a Macro: <F10>

To play back the editor macro, you press the <F10> key. The keystrokes previously recorded in the macro buffer will be repeated as if they were entered from the keyboard and in exactly the same sequence as they were originally recorded.

2.8 Print Text

To print text, press <Ctrl-P>. Normally, the entire file in the current editor window is printed. But you can also print a selected portion of the text by using the <F2> key to mark a block of text (☞ see Section 2.5) and then pressing <Ctrl-P>.

HarPE does not control the printer directly. Rather, it relies on the standard UNIX printer server (queue and spooler) on your system.

2.9 Change Color Map

You can use the <Ctrl-Y> key to change the color map used by HarPE for text and graphics display. ☞ Color map file format is described in Chapter 12.

3

Circuit Files

3

Circuit Files

3.1 Overview

A circuit file is an ASCII text file prepared by the user to supply HarPE with the essential information needed for simulation and optimization.

The circuit file describes the equivalent circuit topology, defines the intrinsic and extrinsic models, initializes the model parameter values, designates optimizable parameters, specifies bias, frequency and input power ranges for simulation, supplies measured data, goals and specifications for optimization.

This chapter outlines the basic structure of the circuit file and describes the syntax rules that are applicable throughout the circuit file. Details of the various file blocks are given in subsequent chapters when the appropriate features are discussed.

Extension .ckt Is Recommended

It is strongly recommended for all HarPE circuit files to have the extension .ckt. Some computation saving features, such as the "HarPE acceleration files" described in Chapter 8, apply only to circuit files with the .ckt extension.

3.2 Keywords

Keywords are names designated by HarPE to denote built-in features and options.

For example, in the circuit file, a resistor is defined as

```
RES  1 2  R = 5.0;
```

where RES is the keyword representing the built-in resistor element model, and R is the keyword representing the resistance parameter of the resistor element model.

Keywords are case insensitive. For example, RES, res and Res are treated by the file parser as identical keywords.

3.3 Circuit File Blocks

The contents of a circuit file are divided into sections called file blocks. Each file block is designated to supply information related to a particular aspect of the operation.

Syntax:

Block_Name
...
...
End

where *Block_Name* represents a designated file block name, and End is a keyword which marks the end of a file block.

TABLE 3.1 CIRCUIT FILE BLOCKS

Block Name	Contains Definitions of	Required for	☞ Chapter
CONTROL	user-controllable default options	-	3
STATISTICS	statistical correlation coefficients	-	10,11
SAMPLE	sample of statistical outcomes	-	10,11
IMPORTDATA	imported linear subcircuit data	-	5
EXPRESSION	labels, variables, functions	-	4
MODEL	circuit model and parameters	circuit simulation	5,6,7
DATA	measurement data	modeling	9,11
SWEEP	simulation ranges and responses	simulation/display	8
SPECIFICATION	optimization ranges and goals	optimization	9,11
MONTECARLO	statistical analysis ranges, outputs	statistical analysis	10

Block names are case insensitive, like all other keywords. For example, MODEL, Model and model all identify the same file block.

Block names can be abbreviated to the first four or more characters. For example, you can use Spec as the name for the Specification block.

Order of entry is arbitrary. The file blocks may appear in the circuit file in any order, although the file parser always processes the blocks in the order given in Table 3.1, automatically rearranging the text (in the parser's memory buffer) if necessary.

The Model block is required. It describes the circuit topology, the intrinsic device model, and the extrinsic elements. The Sweep block is also required for simulation. The other blocks are optional. They are needed only for certain operations as noted in Table 3.1 in the "Required for" column. For instance, you need to define the Specification block in the circuit file only if you wish to perform optimization.

3.4 Circuit File Templates

This section contains a number of templates for creating HarPE circuit files. The templates are designed for several typical combinations of features. In each case, the *minimal* configuration of file blocks necessary to set up a problem properly is shown.

The templates contain only an outline of the circuit file. Details of the relevant file blocks are discussed in the appropriate chapters as listed in Table 3.1.

☞ See also the demo examples in Chapter 13.

Circuit Simulation Using Built-In Models

```
Model
    define circuit topology, model and parameters (Chapters 5,6,7)
End

Sweep
    define simulation ranges and responses (Chapter 8)
End
```

Circuit Simulation with User-Defined Models and Responses

```
Expression
    define labels, model equations and user-defined responses (Chapter 4)
End

Model
    define circuit topology, model and parameters (Chapters 5,6,7)
End

Sweep
    define simulation ranges and responses (Chapter 8)
End
```

Device Modeling

```
Expression
    define labels, model equations and user-defined responses (Chapter 4)
End

Model
    define circuit topology, model and parameters (Chapters 5,6,7)
End

Data
    supply measured data (Chapter 9)
End

Sweep
    define simulation ranges and responses (Chapter 8)
End

Specification
    define optimization ranges, goals and weights (Chapter 9)
End
```

Monte Carlo Analysis

```
Expression
    define labels, model equations and user-defined responses (Chapter 4)
End

Model
    define circuit topology, model and parameters (Chapters 5,6,7)
End

Sweep
    define simulation ranges and responses (Chapter 8)
End

MonteCarlo
    define statistical simulation ranges, number of outcomes and responses (Chapter 10)
End
```

3.5 Statements

The logical unit of the information within a circuit file block is called a statement.

Statements are delimited (separated) by a semicolon ";".

A statement may occupy one text line:

```
RES  1 2  R = 5.0;
```

Or, a long statement may occupy several text lines:

```
Value_DB:  if (Value > Value_Max) (20.0 * log10(Value_Max))
           else (if (Value < Value_Min) (20.0 * Log10(Value_Min))
           else (20.0 * log10(Value)));
```

On the other hand, several short statements may share a single text line:

```
V1 = 50;   V2 = ?1.5?;   V3 = (V1 + V2) / 2;
```

Since a statement does not always correspond to one text line, make sure it is properly terminated by a semicolon. Otherwise information from different lines can get mixed up.

Comments

Comments can be included in the circuit file following the exclamation mark "!". The file parser will skip over any text following an exclamation mark "!" until the end of the line.

You can have full lines of comments, such as

```
!  File: demo01.ckt
!  HarPE circuit file for demonstration
```

or in-line comments, such as

```
FETC  1 2 3  ...      ! Curtice cubic model
```

Blank lines can also be used to space the circuit file text for clarity.

3.6 Preprocessor Macros

HarPE's file parser has a built-in preprocessor. As the first phase of file parsing, the circuit file is scanned for preprocessor directives which are marked by the number sign (hash) "#". The preprocessor directives include text macros and include files.

Text macros are created by the #define directive to represent lengthy, complex and/or repetitive portions of text. At parsing time, references to text macros are expanded into the full text the macros represent. The use of text macros can make the circuit file more readable and easier to modify.

The preprocessor macros in HarPE are quite similar to those of the "C" language, except

- In HarPE, text macro names are case insensitive unless enclosed within quotation marks.
- In HarPE, macros apply to the whole circuit file regardless of the location where the macros are defined. In other words, the file parser recognizes references to a macro even if such references appear *before* the macro is defined. This is necessary because, as described in Section 3.3, the circuit file blocks may be rearranged by the file parser into a certain order.
- HarPE allows two alternative styles for continuation lines.
- HarPE implements a special "$" argument.

Macro Constants

Syntax:

```
#define  macro_name   substitution_text
```

Example:

```
#define  Boltzmann_Constant    1.38066E-23
```

During preprocessing, the file parser will replace all references to *macro_name* in the circuit file with *substitution_text*. In the example, the substitution text is "1.38066E-23". The file preprocessor will replace all references to Boltzmann_Constant with "1.38066E-23".

Macro names are case insensitive unless enclosed within double quotation marks. For example, MACRO1, macro1 and Macro1 are indistinguishable as macro names, whereas "macro1" and "Macro1" would be considered as different and distinct names.

Macros are useful for associating critical constants with meaningful names, such as the Boltzmann constant.

Aliases for built-in keywords can be created using macros. For instance, the macro

```
#define  Resistor   RES
```

creates an alias for the built-in resistor element keyword RES. Using this alias, you can write

```
Resistor  1 2  R = 5.0;
```

which the file parser will translate into the standard form of

```
RES  1 2  R = 5.0;
```

Macro Expressions

Syntax:

#define *macro_name* *substitution_text*

Example:

```
#define  AREA     LENGTH * WIDTH
```

Here, the substitution text is an expression rather than a constant.

Recursive references, i.e., macros which contain references to other macros, are permissible. For example,

```
#define  VOLUME   AREA * HEIGHT
```

where the substitution text includes a reference to the macro AREA defined above.

Care must be exercised with recursive references to macro expressions. You may have to use parentheses to preserve the proper precedence of algebraic operations.

Example:

```
#define  X_SUM   X1 + X2

#define  Y2      Y1 * X_SUM
```

Y2 will be expanded into

```
Y1 * X1 + X2
```

which is unlikely to be the intended result.

A proper definition of Y2 should be

```
#define  Y2    Y1 * (X_SUM)
```

Remember that the preprocessor does not actually evaluate the macro expressions but merely carries out text substitutions.

Macro Functions

Macro functions are the most sophisticated form of preprocessor macros. In addition to a name for identification, a macro function is defined with a set of arguments and the substitution text is typically defined with dependence on the arguments.

Syntax:

#define *macro_name(arguments) substitution_text*

Example:

```
#define  MAGNITUDE(REAL,IMAG)   sqrt(REAL*REAL + IMAG*IMAG)
```

The opening parenthesis of the argument list must immediately follow the macro name. No space can separate the macro name from the opening parenthesis, otherwise the parenthesis and the arguments would be mistaken as part of the substitution text. Multiple arguments must be separated by commas.

Formal and actual arguments: the arguments listed in the macro definition are called the formal arguments; each reference to a macro function must supply a set of actual arguments to match the list of formal arguments. The formal arguments which appear in the substitution text will be replaced by the actual arguments. In other words, for each reference, a specific version of the substitution text is generated according to the actual arguments.

For example, the following reference to the macro function shown above

```
MAGNITUDE(x1,x2)
```

will be expanded during preprocessing to

```
sqrt(x1*x1 + x2*x2)
```

Recursive references in actual arguments, i.e., the actual arguments for a macro function which contain references to other macros, are permissible. For example, consider a macro function defined as

```
#define  MULTI(X1,X2)     X1 * X2
```

and the following reference to this macro function:

```
MULTI(MAGNITUDE(x1,x2), x3)
```

where the actual arguments refer to another macro function MAGNITUDE which is defined earlier. This reference will be expanded to

```
sqrt(x1*x1 + x2*x2) * x3
```

after preprocessing.

Care must be exercised with recursive macro references. You may have to use parentheses to preserve the proper precedence of algebraic operations.

Example:

```
#define  SUM(X1,X2)      X1 + X2
#define  MULTI(X1,X2)    X1 * X2

MULTI(SUM(x1,x2), x3)
```

The macro reference will lead to erroneous results after preprocessing:

```
x1 + x2 * x3
```

You can avoid this kind of problem by using parentheses either in the macro definition, such as

```
#define  SUM(X1,X2)    (X1 + X2)
```

or, in the actual arguments

```
MULTI((SUM(x1,x2)), x3)
```

or both.

Formal arguments are local identifiers within the macro functions, since they are merely placeholders for the actual arguments. For instance,

```
#define  MAGNITUDE(REAL,IMAG)     sqrt(REAL*REAL + IMAG*IMAG)
```

the formal arguments REAL and IMAG are local identifiers recognized only within the macro function MAGNITUDE.

Hence, you can assign the same names to the formal arguments of different macro functions without causing conflicts.

Continuation Lines for Long Substitution Text

The substitution text of some macros may be longer than one line. Similar to the "C" language, HarPE's preprocessor allows long text to be continued onto the next line by ending the current line with a backslash "\":

```
#define  EXP_SAFE(X)   if (X > X_MAX) (exp(X_MAX))    \
                       else (if (X < X_MIN) (0)       \
                       else (exp(X)))
```

Also, HarPE implements an alternative style for continuation lines: using a pair of curly brackets to enclose the lines, such as

```
#define  EXP_SAFE(X)  {  if (X > X_MAX) (exp(X_MAX))
                         else (if (X < X_MIN) (0)
                         else (exp(X)))
                      }
```

This is especially useful for macros which contain multiple statements, such as

```
#define  SPAR_IN_DB {
            MS11_DB = 20 * log10(MS11);
            MS21_DB = 20 * log10(MS21);
            MS12_DB = 20 * log10(MS12);
            MS22_DB = 20 * log10(MS22);
         }
```

The two styles for continuation lines can be mixed for different macros, but they cannot be mixed within the same macro definition.

Null Substitution Text

A macro can be defined with null (empty) substitution text:

```
#define  macro_name
```

The result is that all appearances of *macro_name* are removed from the circuit file.

An interesting application is the design of a "compilation switch". You may wish to include or exclude part of the circuit file for testing purpose. A convenient way for you to do this is to mark (precede) the text you wish to include/exclude with a macro name, say, FLAG. Then you can "comment out" the marked text by having

```
#define  FLAG   !
```

or, you can "activate" the marked text by having

```
#define  FLAG
```

which in effect removes all occurrences of FLAG from the circuit file.

The $ Argument

The "$" argument is a special feature of HarPE for macro functions. You can include "$" as a formal argument of a macro function, and the $ argument will be treated differently from any other arguments, in the sense that it can be concatenated with other arguments to form unique identifiers (names).

Example:

```
#define  OFFSET($)     FACTOR$ * A$09 + J$
```

Then

```
OFFSET(3)
```

will be expanded into

```
FACTOR3 * A309 + J3
```

The $ argument may appear anywhere in the formal argument list. The corresponding actual argument can be a number, a character string or even null (empty). For example,

```
#define  CONCAT(A, $, B)      A$B

CONCAT(Drive_, Path, _File)
CONCAT(Spec, , ification)
```

The first reference will be expanded into Drive_Path_File and the second reference will be expanded into Specification.

Macro Subcircuits

One of the most useful applications of macro functions is the definition of symbolic subcircuits. ☞ See Chapter 5 for details.

3.7 Include Files

Syntax:

```
#include  file_name
```

Examples:

```
#include "my_model.inc"

#include "/fet_lab/dev_3305.dat"
```

where "my_model.inc" and "/fet_lab/dev_3305.dat" are file names.

The preprocessor will incorporate the contents of the named file into the circuit file at the point where the #include directive appears.

The files to be included must already exist on the disk and must be accessible to HarPE (i.e., the user must have the file read privilege).

Include files are useful for

- breaking up a very long file into several smaller files
- importing measurements and subcircuit data
- storing frequently used definitions to be shared among different circuit files

Restrictions on include files:

- The #include directive must appear within the body of a file block. The file block name (which begins the block) and the END keyword (which closes a file block) must be in the main circuit file and cannot be imported from an include file. In other words, a file block cannot begin or end within an include file.
- The circuit file may contain any number of #include directives, but an include file cannot contain any #include directives. In other words, you can have many files to be included into the circuit file, but you cannot have nested include files.
- An include file cannot contain definitions of optimization variables (☞ Chapter 4), because of back-annotation problems. After optimization, the circuit file text is updated with the new values of the optimizable variables. Since the text of an include file is utilized only during preprocessing, it cannot be updated after optimization.

3.8 Physical Units

When entering numerical constants in a circuit file, you can include physical units from those listed in Table 3.2. For example, you can use 3.5CM to represent 3.5 centimeters.

TABLE 3.2 PHYSICAL UNITS

Quantity	Available Keywords			
capacitance	F	(farad)	MF	(millifarad)
	UF	(microfarad)	NF	(nanofarad)
	PF	(picofarad)		
conductance	/OH	(siemens)	/KOH	(millisiemens)
	/MOH	(microsiemens)		
frequency	HZ	(hertz)	KHZ	(kilohertz)
	MHZ	(megahertz)	GHZ	(gigahertz)
inductance	H	(henry)	MH	(millihenry)
	UH	(microhenry)	NH	(nanohenry)
	PH	(picohenry)		
resistance	OH	(ohm)	KOH	(kilo-ohm)
	MOH	(mega-ohm)		
time	SEC	(second)	MS	(millisecond)
	US	(microsecond)	NS	(nanosecond)
	PS	(picosecond)		
power	DBM	(dBm)	W	(watt)
	MW	(milliwatt)		
current	A	(ampere)	MA	(milliampere)
	UA	(microampere)	NA	(nanoampere)
voltage	V	(volt)	MV	(millivolt)
	UV	(microvolt)		
phase	DEG	(degree)	RAD	(radian)
length	M	(meter)	CM	(centimeter)
	MM	(millimeter)	UM	(micron)
	IN	(inch)	MIL	(milli-inch)
	UIN	(microinch)		

The physical units are case insensitive, e.g., MHZ, MHz and mhz all represent megahertz.

The unit must not be separated by space from the numerals. For instance, 3.5cm cannot be entered as 3.5 cm.

Default Units

Numerical constants without explicit units in the circuit file are assigned default units. Two systems of default units are available, as listed in Table 3.3. The difference between these two systems is their default units for capacitance, frequency, inductance and time.

TABLE 3.3 DEFAULT UNITS

Quantity	Microwave	Non-Microwave
capacitance	NF	F
conductance	/OH	/OH
frequency	GHZ	HZ
inductance	NH	H
resistance	OH	OH
time	NS	SEC
power	DBM	DBM
current	A	A
voltage	V	V
phase	DEG	DEG
length	M	M

The Microwave unit system is the default system. You can change the default unit system to Non-Microwave in the Control block (☞ Section 3.9).

Syntax:

```
Control
   Non_Microwave_Units;
   ...
   ...
End
```

For upward compatibility you can also change the default unit system to Non-Microwave in the Expression block.

Syntax:

```
Expression
   Non_Microwave_Units;
   ...
   ...
End
```

The control keyword Non_Microwave_Units, if given, must be the first statement in the Expression block.

To avoid ambiguity and ensure compatibility, it is highly recommended that you assign explicit units wherever possible.

3.9 Control Block

Many of the operation parameters of HarPE are user-controllable by means of the Control block in the input file.

Syntax:

```
Control
   control_keyword = choice;

   control_keyword;
   ...
End
```

TABLE 3.4 CONTROL KEYWORDS

Keyword	Type	To Select
Accuracy	choice	solution accuracy required for optimization
Huber_Threshold	choice	threshold value for Huber optimization
Keep_DDF	choice	options for keeping DDF for statistical model
Matching_Goal	choice	matching goal for statistical modeling
Non_Microwave_Units	flag	non-microwave physical unit system
N_Hist_Bins	choice	number of bins of histogram for direct statistical modeling using histogram matching
N_Iterations	choice	number of iterations for optimization
Optimizer	choice	choosing the optimizer
Reset_Variables	flag	reset variables in multi-device parameter extraction for indirect statistical modeling

The keywords listed as the "choice" type represent options with multiple choices and you must enter a specific choice following the keyword. The keywords listed as the "flag" type activate the options they represent.

Solution Accuracy Required for Optimization

When you invoke optimization, you can interactively select the accuracy required for the solution in the pop-up window (☞ Chapter 9).

You can also designate your choice in the Control block.

Syntax:

`Accuracy = v;`

where *v* is one of the choices: `0.01, 0.001, 1.0e-4, 1.0e-5, 1.0e-6.`

Other values of *v* will be rounded to the closest choice which is smaller than *v*, e.g., 0.003 will be rounded to 0.001. If *v* < 1.0e-6, it will be set to 1.0e-6.

Threshold Value for Huber Optimization

When you invoke optimization and select the Huber optimizer, you can interactively select the threshold value in the pop-up window (☞ Chapter 9).

You can also designate the threshold value in the Control block.

Syntax:

`Huber_Threshold = v;`

where *v* must be a positive value.

Discrete Density Function for Statistical Modeling

When you invoke HarPE.Statistical.Postprocess option, you can interactively press <Y> or <N> to keep or discard the discrete density functions as a discrete approximation of the marginal density function of a statistical parameter (☞ Chapter 11).

You can also designate your choice in the Control block.

Syntax:

`Keep_DDF` = *`selection`*;

selection can be: `All, None, Interactive`

`All` for keeping DDFs for all statistical parameters.
None for discarding all DDFs.
`Interactive` for interactively selecting DDFs for individual statistical parameters.

When you select the `Interactive` option HarPE must run in the interactive mode.

Matching Goal for Statistical Modeling

When you invoke HarPE.Statistical.Match menu, you can interactively select "Histogram matching" or "CPD matching" for the Goal option in the pop-up window (☞ Chapter 11).

You can also designate your choice in the Control block.

Syntax:

`Matching_Goal` = *`goal_type`*;

goal_type can be: `CPD, HST, PEP`

`CPD` for cumulative probability distribution matching.
`HST` for histogram matching.
`PEP` for indirect statistical modeling using parameter extraction and postprocessing.

This control keyword is particularly designed for HarPE running as a Datapipe child of OSA90/hope for statistical modeling (☞ see the manual "HarPE as a Child of OSA90/hope"). When the choice is selected as "PEP", the indirect method using multi-device parameter extraction and postprocessing is used for statistical modeling. When the choice is selected as "CPD" or "HST", the direct method is used for statistical modeling with CPD fitting or histogram fitting (☞ Chapter 11).

When you run HarPE in the interactive mode and invoke the HarPE.Statistical.Extract menu, this keyword will be ignored if "CPD" or "HST" is selected. On the other hand, if you select "PEP" but invoke the HarPE.Statistical.Match menu, it is also ignored.

Non_Microwave_Units

This keyword controls the default physical unit system to be used, see Section 3.8.

Number of Bins of Histograms for Direct Statistical Modeling

When you invoke HarPE.Statistical.Match menu and select "Histogram matching" for the Goal option, you can interactively enter, in the pop-up window, the number of bins of the histograms to be matched (☞ Chapter 11).

You can also designate your choice in the Control block.

Syntax:

`N_Hist_Bins = n`

where n must be an integer and $n \geq 2$. The default value is 20.

Maximum Number of Optimization Iterations

When you invoke optimization, you can interactively select the maximum number of optimization iterations in the pop-up window (☞ Chapter 9).

You can also designate your choice in the Control block.

Syntax:

`N_Iterations = n;`

where n is one of the choices: 1, 2, 5, 10, 15, 30, 50, 100, 999.

Other values of n will be rounded to the closest choice which is greater than n, e.g., 45 will be rounded to 50. If $n > 999$, it will be set to 999.

Default Optimizer

HarPE offers several optimizers (minimax, ℓ_1, ℓ_2, Huber, ☞ see Chapter 9 for more details). You can designate the default optimizer in the Control block.

Syntax:

`Optimizer = optimizer_name;`

optimizer_name can be: `L1, L2, Minimax, Huber`

Reset Variables in Statistical Parameter Extraction

When you invoke HarPE.Statistical.Extract menu, you can interactively select "Yes" or "No" to reset the variables to the initial values defined in the circuit file for parameter extraction of every device or to use the parameter extraction solution of previous device as the starting point (☞ Chapter 11).

You can also designate your choice in the Control block.

Syntax:

`Reset_Variables;`

This will instruct the program to reset the variables during statistical parameter extraction. In other words, the optimization will have a common starting point for all the devices.

4

Expressions

4

Expressions

4.1 Overview

In the Expression block of the circuit file, you can define labels to represent constants, optimization variables, formulas, model equations and user-defined responses.

Syntax:

```
Expression

   label_name: constant;
   label_name: optimization variable;
   label_name: expression;
   ...

End
```

4.2 Label Names

Labels are defined by statements which begin with a label name, followed by a colon ":" or an equal sign "=", and ends with a semicolon ";".

For example,

```
V1: 2.5;
Formula1 = V1 * V1 - 0.76 * V1 + 2;
```

where V1 and Formula1 are label names. The label V1 represents a constant, and the label Formula1 represents a formula (an expression).

Case Sensitivity

Label names are case insensitive by default, e.g., FORMULA1, formula1 and Formula1 are considered to be referring to the same label.

Reserved Symbols

Label names can be arbitrary character strings but they must not contain any of the reserved symbols listed in Table 4.1, and they must not begin with a digit (0, ..., 9).

For instance, "3_VAR" or "ABC!" cannot be used as a label name.

TABLE 4.1 RESERVED SYMBOLS

+	-	.	,	*	/	\	!	@	#	$
%	^	&	(	)	=	{	}	[	]	:
;	'	<	>	?	|	~	‘			

Literal Label Names

Literal label names can be created by enclosing the label names in quotation marks. Literal label names are case sensitive, e.g., "formula1" and "Formula1" will be treated as distinct and different label names. Also, literal label names are not restricted by reserved symbols, e.g.,

```
"/harpe/fet_dc.dat"
"1989-device-33-55-7"
```

are legitimate label names. This is especially relevant for file names, because they are case sensitive on UNIX systems and often contain the reserved symbols "." and "/".

4.3 Constant Labels

In the simplest form, a label represents a numerical constant, such as

```
Boltzmann_Constant: 1.3806E-23;
```

Constant labels and preprocessor macro constants (Chapter 3) are nearly identical in their usage. For example, a macro constant defined as

```
#define  Boltzman_Constant  1.3806E-23
```

can be used almost identically to the constant label defined above.

The subtle difference lies in their internal structure: a macro is meaningful only to the file preprocessor as an alias of its substitution text, whereas a label is structured as a permanent identifier. A label can be used as an output for graphical and numerical display, but a macro cannot. On the other hand, a label requires slightly more memory and processing time than a macro.

4.4 Predefined Labels

Predefined labels are automatically created by HarPE to allow the user access to certain internal variables and built-in responses, such as the frequency, bias voltages, intrinsic voltages, output power and *S* parameters, as listed in Table 4.2.

TABLE 4.2 PREDEFINED LABELS

Label	Definition	Notes
PI	constant 3.141592654	
FREQ	frequency or harmonic frequency	1
FFREQ	fundamental frequency	1
VD	drain (collector) bias voltage in Volts	2
VG	gate (base) bias voltage in Volts	
VDS_DC	intrinsic drain-source DC voltage in Volts	2, 3
VGS_DC	intrinsic gate-source DC voltage in Volts	3
VDS_T	intrinsic drain-source instantaneous voltage in Volts	2, 4
VGS_T	intrinsic gate-source instantaneous voltage in Volts	4
VGG_T	intrinsic internal gate instantaneous voltage in Volts	4
VGS_TAU	intrinsic gate-source voltage delayed by TAU, in Volts	4
PIN	input power in DBM	
PINW	input power in Watts	
IDO	DC component of drain (collector) current in Amperes	5
POUT*k*	output power at the *k*th harmonic, in DBM	5
POUTW*k*	output power at the *k*th harmonic, in Watts	5
MS*ij*	magnitude of *S* parameters $i,j = 1,n$	6
PS*ij*	phase of *S* parameters in degrees $i,j = 1,n$	6
RS*ij*	real part of *S* parameters $i,j = 1,n$	6
IS*ij*	imaginary part of *S* parameters $i,j = 1,n$	6
IG	DC gate (base) current in Amperes	7
ID	DC drain (collector) current in Amperes	7

1. The unit of frequency is GHZ in the default microwave unit system, but you can change the default unit to HZ by including the control keyword Non_Microwave_Unit as the first statement in the Expression block (also see Chapter 3).
2. Not applicable if only one bias port is defined (see Chapter 5).
3. Useful for defining bias-dependent small-signal models.
4. To be used in user-defined nonlinear intrinsic models only.
5. Large-signal responses.
6. Small-signal *S* parameters, where *n* represents the number of ports (see Chapter 5).
7. DC responses.

User-Defined Model Equations

By referring to the predefined labels, you can create expressions for user-defined models.

For example, the drain current in the Curtice cubic FET model can be expressed as

```
Vl = VGS_TAU * (1 + Beta * (Vds0 - VDS_T));

Drain_Current = (A0 + A1*Vl + A2*Vl^2 + A3*Vl^3) *
                tanh(Gamma * VDS_T);
```

where the predefined labels VDS_T and VGS_TAU represent the time-domain (instantaneous) drain voltage and the delayed gate voltage, respectively. A0, A1, A2, A3, Vds0, Beta and Gamma are user-defined model parameters.

This equation can then be incorporated into the circuit definition as a user-defined model in the Model block:

```
FETU1  1 2 3  IDS=Drain_Current ... ;
```

☞ See also Chapters 5 and 6.

User-Defined Responses

The predefined labels can also be used to create user-defined responses.

Examples:

```
Power_Added_Efficiency = 100 * (POUTW1 - PINW) / (ID0 * VD);

ID_in_mA:  ID * 1000;

PS11_in_Radian:  PS11 * PI / 180;
```

where Power_Added_Efficiency, ID_in_mA and PS11_in_Radian represent user-defined large-signal, DC and small-signal responses, respectively.

User-defined response functions can be calculated through circuit simulation, displayed and optimized just like the built-in responses (☞ Chapters 8 and 9).

4.5 Optimization Variables

Optimization variables in the circuit file are marked by question marks.

Syntax:

label_name: ?*initial_value*?;

Example:

```
X1: ?2.5?;
```

After optimization, the variables will be automatically updated with their optimized values. For instance, consider the variable labelled X1 in the above example. Suppose that its value is changed to 1.234 after optimization. Then the corresponding text in the circuit file will be automatically updated to be

```
X1: ?1.234?;
```

Optimization variables can also be directly defined for circuit model parameters in the Model block without using a label (☞ Chapter 5).

Bounds on Variables

Optional bounds can be assigned to an optimization variable:

Syntax:

label_name: ?*lower_bound* *initial_value* *upper_bound*?;

Example:

```
X1: ?0.1  2.5  10?;
```

This will force the value of the variable to remain within *lower_bound* and *upper_bound* during optimization. If you specify bounds, you must specify both bounds.

Relative bounds can also be specified.

Syntax:

label_name: ?*lower*% *initial_value* *upper*%?;

The effective bounds are calculated by the file parser as

lower_bound = *initial_value* - ABS(*lower* * *initial_value*) / 100

upper_bound = *initial_value* + ABS(*upper* * *initial_value*) / 100

Example:

```
X1: ?80%  10  70%?;
```

The effective lower bound will be 2 and the effective upper bound 17.

Note that the effective bounds are evaluated based on the *initial_value* when the circuit file is parsed and they will not change during optimization.

Default Bounds

An optimization variable defined without explicit bounds will be assigned the default bounds: (0 +∞) if the initial value is positive or zero, or (-∞ 0) otherwise.

For example,

```
X1: ?2.5?;
```

X1 will be given the default bounds (0 +∞), i.e., its value will remain positive.

The default bounds are based on the assumption of natural bounds which exist in many physical problems. For example, when inductances, capacitances and resistances are defined as optimization variables, their values should never become negative.

4.6 Expressions

Expressions generically refer to user-defined formulas, model equations and response functions. Expressions consist of constants, algebraic operators, mathematical functions, and references to labels that are already defined.

For example,

```
Euclidean_Norm = SQRT(X * X + Y * Y);
```

where Euclidean_Norm is the label representing the expression, SQRT is the mathematical function of square root, "*" and "+" are algebraic operators, and X and Y refer to labels that should be already defined.

Optimization variables cannot be created within an expression. For example,

```
F1 = FREQ * ?2.5?;
```

is not acceptable. Optimization variables must be defined separately, such as

```
X1: ?2.5?;

F1 = FREQ * X1;
```

Algebraic Operators

Table 4.3 lists the set of algebraic operators available.

TABLE 4.3 ALGEBRAIC OPERATORS

Symbol	Operation
+	addition
-	subtraction
>	greater than
<	less than
=	equal to
*	multiplication
/	division
^	exponentiation
( )	precedence

Logical expressions can be constructed using the operators ">", "<" and "=". In HarPE, logical expressions evaluate to 1 if the logical condition is true, or to 0 if false.

The operators are listed in ascending order of precedence in Table 4.3. Parentheses can be used to override the natural order of precedence, e.g.,

```
Error = Weight * (Response - Goal);
```

Mathematical Function Library

Table 4.4 lists the library of mathematical functions.

TABLE 4.4 LIBRARY OF MATHEMATICAL FUNCTIONS

Function	Description
ABS(x)	absolute value of x
EXP(x)	exponential function of x
LOG(x)	natural logarithm of x
LOG10(x)	base-10 logarithm of x
SQRT(x)	square root of x
SIN(x)	sine of x
COS(x)	cosine of x
TAN(x)	tangent of x
ACOS(x)	arccosine of x
ASIN(x)	arcsine of x
ATAN(x)	arctangent of x
SINH(x)	hyperbolic sine of x
COSH(x)	hyperbolic cosine of x
TANH(x)	hyperbolic tangent of x
FLOOR(x)	the largest integer $\leq x$
CEIL(x)	the smallest integer $\geq x$
J0(x)	Bessel function of the first kind, order 0
J1(x)	Bessel function of the first kind, order 1
Y0(x)	Bessel function of the second kind, order 0
Y1(x)	Bessel function of the second kind, order 1
POS(x)	returns 1 if $x > 0$, returns 0 otherwise
NEG(x)	returns 1 if $x < 0$, returns 0 otherwise
ZERO(x)	returns 1 if $x = 0$, returns 0 otherwise

4.7 Conditional Expressions

Conditional expressions can be used to assign different values to a label according to a given set of conditions. HarPE supports both explicit if - else structures and implicit conditional expressions using the >, < and = operators and the POS(), NEG() and ZERO() functions.

If - Else Structure

Syntax:

```
label: if (condition) (expression1) [else (expression2)];
```

Examples:

```
Absolute_Difference: if (X > Y) (X - Y) else (Y - X);

Drain_Current = if (VDS_T > 0) (Id_Model) else (0);
```

condition, *expression1* and *expression2* must all be enclosed within parentheses.

If *condition* is true, then *label* is evaluated from *expression1*, otherwise it is evaluated from *expression2*.

If *expression2* is omitted and *condition* is false, then *label* is set to zero.

Algebraic condition: if *condition* is an algebraic expression, then it is "true" if it evaluates to a nonzero value, or "false" if it evaluates to zero.

For example,

```
Y:  if (X - 3.5) (4.44) else (7.77);
```

Y will be set to 4.44 if X equals to 3.5, otherwise Y is set to 7.77.

Algebraic conditions can be used to implement AND and OR logics. For example,

```
Y: if ((X > 2) + (X < -2)) (X);
```

Y is set to X if X > 2 or X < -2.

Another example:

```
Y: if ((X < 2) * (X > -2)) (X);
```

Y is set to X if -2 < X < 2.

Nested if - else structures are supported:

```
Y: if (X > X_MAX) (exp(X_MAX))
   else (
         if (X < X_MIN) (exp(X_MIN)) else (exp(X))
        );
```

The inner if - else structure(s) must be entirely enclosed within parentheses.

Only the value of a label but not the existence of the label itself can be made conditional. Hence the following "statement" is not permissible:

```
if (X > X_MAX) (
   Y: exp(X_MAX);
)
```

Implicit Conditional Expression

In addition to explicit if - else structures, implicit conditional expressions can be defined using the >, < and = operators and the POS(), NEG() and ZERO() functions.

For example,

```
Y = POS(X) * Y1 + ZERO(X) * Y2 + NEG(X) * Y3;
```

is equivalent to

```
Y = if (X > 0) (Y1) else (if (X = 0) (Y2) else (Y3));
```

The same conditions can also be expressed as

```
Y = (X > 0) * Y1 + (X = 0) * Y2 + (X < 0) * Y3;
```

Implicit conditional expressions are generally less efficient. In the above example, using an explicit if - else structure only one of the three formulas, namely Y1, Y2 or Y3, needs to be evaluated under any given condition. Using implicit expressions all three formulas are evaluated first and then multiplied by the corresponding conditions (1 or 0). This is especially vulnerable if one of the formulas may result in a floating-point error under certain conditions. For example, expression

```
Y = POS(X) * log(X);
```

will cause an error when X is zero or negative. The correct expression should be

```
Y = if (X > 0) (log(X));
```

On the other hand, implicit conditional expressions can be useful for complex expressions which depend on multi-layer and inter-related algebraic conditions.

5

Circuit Model

5

Circuit Models

5.1 Overview

The Model block of the circuit file contains the definition of the circuit model.

For HarPE, the circuit model typically consists of a single nonlinear intrinsic device embedded in a linear extrinsic environment. Purely linear circuits, i.e., circuits which do not contain any nonlinear device model, are also accommodated.

The overall circuit must be defined as a one- or two-port. The external (input and output) ports do not have to be grounded. The bias circuit is defined either implicitly through the input and output ports, or explicitly as separate bias ports.

Syntax:

```
Model
   element_name  nodes  parameters;
   ...
   element_name  nodes  parameters;
   bias_ports  nodes;
   input_output_ports  nodes;
End
```

Each *element_name* identifies a library linear or nonlinear element model.
The optional *bias_ports* can be defined by `1BIASPORT` or `2BIASPORT`.
The *input_output_ports* can be defined as `1POR`, `2POR`, `1PORT` or `2PORT`.

5.2 Nodes

The circuit components in the Model block are interconnected through nodes.

A node can be designated by an arbitrary positive integer, such as

```
RES  1 2  ...
```

which means that a RES element is connected between the nodes designated as 1 and 2.

Node numbers do not need to be consecutive or in any particular order.

Nodes Designated by Names

If you prefer, nodes can also be designated by character strings with the prefix @. For example,

```
RES  @input  @ground ...

FETM  @gate  @drain  @source  ...
```

Node names are case insensitive unless they are enclosed within quotation marks. For example,

```
@gate,  @GATE,  @Gate
```

designate the same node, but

```
@"gate",  @"GATE"
```

are considered as two separate and distinct node names.

You can mix numerical and string nodes in the same file:

```
RES  1  @output  ...
```

The Ground Node

The node number 0 and the node name @ground are automatically created and they always designate the ground node.

5.3 Elements

HarPE provides a library of linear and nonlinear elements from which you can select components of a circuit model.

Syntax:

element_name n1 n2 ... nk par1=x1 par2=x2 ... parm=xm;

where *element_name* is a keyword identifying a library element, *ni* is the *i*th connection node, *parj* is a keyword identifying one of the parameters of the element model, and *xj* is a value or a label assigned by the user to the parameter *parj*.

Examples:

```
RES  3 5  R=50;

CAP  @input @ground  C=10pF;
```

Catalogues of the library elements are contained in Chapter 6 (nonlinear elements) and Chapter 7 (linear elements), which include the schematics, keywords, number of nodes, parameters and model equations where appropriate.

Each element definition is a statement and must end with a semicolon ";".

Element Nodes

Each element has a predefined number of nodes. For example, the element RES (resistor) has 2 nodes. Nodes that are not explicitly specified will be grounded (i.e., default to the node 0 or @ground).

However, each element requires a minimum number of nonzero nodes for the element to be meaningfully connected, e.g., the element RES requires at least one nonzero node.

Element Parameters

Each element has a number of parameters. An element parameter can be assigned a constant value, defined as an optimization variable or specified by a label.

Example:

```
SRLC  1 0  R=15  L=?0.2nH?  C=Label_C1;
```

where R, L and C are keywords representing the parameters for the element SRLC. R is assigned a constant value, L is defined as an optimization variable, and C is specified by the label Label_C1 which must have been defined prior to this reference.

Numerical constants can have physical units, e.g., 10pF and 0.2nH (☞ Chapter 3).

Parameters Defined As Optimization Variables

Element parameters can be defined as optimization variables by enclosing their initial values within question marks.

Syntax:

parameter_name = ?initial_value?

Example:

```
RES  1 2  R = ?25?;
```

After optimization, the variables will be automatically updated with their optimized values. For instance, suppose that the value of the optimization variable in the above example is changed to 34.56 after optimization, then the corresponding text in the circuit file will be automatically updated to be

```
RES  1 2  R = ?34.56?;
```

You can also define a labelled optimization variable (☞ Chapter 4) in the Expression block and then assign the label to one or more circuit model parameters.

Example:

```
Expression
   X1: ?25?;
   ...
End

Model
   RES  1 2  R = X1;
   ...
End
```

By assigning a label to several circuit parameters, you can tie the parameters to one optimization variable.

Optional bounds can be assigned to optimization variables. For example:

```
RES  1 2  R = ?5  25  100?;
```

where 5 is the lower bound and 100 is the upper bound. This will force the value of the variable to remain within the bounds during optimization (☞ Chapter 4 for further details on optimization variable bounds and default bounds).

Parameters Defined by Expressions

An element parameter can also be defined by an expression (formula), such as

```
RES  1 0  R = (Resistivity * Length / Cross_Section);
```

Such an expression within an element definition must be enclosed in a pair of parentheses to clearly delimit the expression from the rest of the text. Otherwise, the expression may be confused with the other parameter keywords in the same statement.

For example,

```
SRL  1 0  R = Resistivity * Length / Cross_Section  L = A + B;
```

will be considered as an error by the circuit file parser.

Alternatively, you can define the expression as a label in a separate statement:

```
Label_R: Resistivity * Length / Cross_Section;

RES  1 0  R=Label_R;
```

5.4 Input and Output Ports

The overall circuit must be defined as either a one-port or a two-port. For a two-port, the first port is designated as the input port, and the second port as the output port.

Syntax:

`1POR` *n1*;

`1PORT` *n1 n2*;

`2POR` *n1 n2*;

`2PORT` *n1 n2 n3 n4*;

where *n1*, *n2*, *n3* and *n4* represent nodes.
The 1POR, 1PORT, 2POR or 2PORT statement must be the last statement in the MODEL block, after all the elements have been defined.

The 1POR and 2POR statements define grounded one- and two-ports, and the 1PORT and 2PORT statements define general one- and two-ports, respectively. They are illustrated in Figs. 5.1 through 5.4.

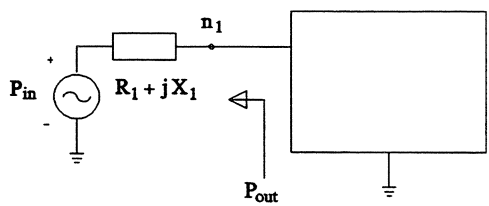


Fig. 5.1 A grounded 1-port defined by the 1POR *statement.*

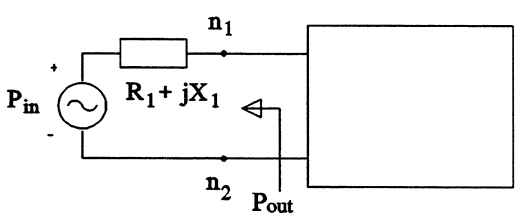


Fig. 5.2 A general 1-port defined by the 1PORT *statement.*

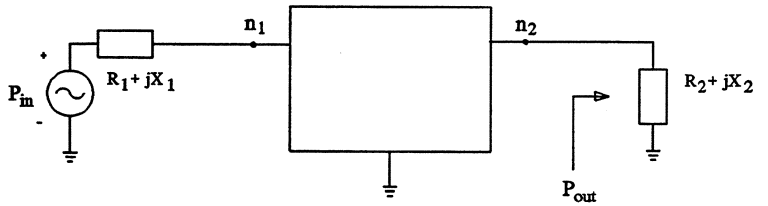


Fig. 5.3 A grounded 2-port defined by the 2POR *statement.*

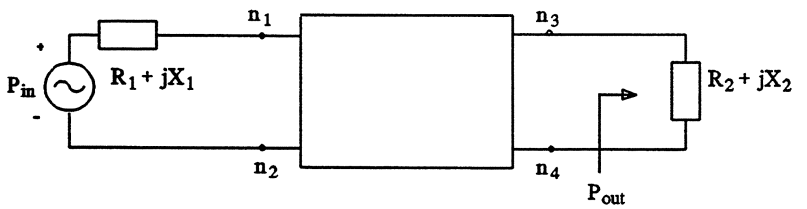


Fig. 5.4 A general 2-port defined by the 2PORT *statement.*

Port Terminating Impedances

You can specify the port terminating impedances in the 1POR, 1PORT, 2POR or 2PORT statement.

Syntax:

`1POR` *n1* `R1=`*v1* `X1=`*v2*;

`1PORT` *n1 n2* `R1=`*v1* `X1=`*v2*;

`2POR` *n1 n2* `R1=`*v1* `X1=`*v2* `R2=`*v3* `X2=`*v4*;

`2PORT` *n1 n2 n3 n4* `R1=`*v1* `X1=`*v2* `R2=`*v3* `X2=`*v4*;

R1 and X1 (R2 and X2) are keywords representing the real and imaginary parts of the terminating impedance at port 1 (port 2), respectively.
v1, v2, v3 and *v4* represent user entries which can be numerical constants, optimization variables, labels or expressions.

The default terminating impedance of the input/output ports is 50OH.

Example:

```
2POR  1 2  R1=50  R2=?35?  X1=label_1  X2=(0.2 * FREQ);
```

where R1 is defined as a constant, R2 is defined as an optimization variable, X1 is assigned through a label named label_1 which must be already defined in the Expression block, and X2 is defined by an expression which must be enclosed within parentheses.

Termination Specified by Imported Data

A port terminating impedance can also be specified by imported frequency-dependent data by replacing the keywords R1 and X1 (or R2 and X2) with the keyword Z1 (or Z2).

Syntax:

1POR *n1* Z1=*data_label1*;

1PORT *n1 n2* Z1=*data_label1*;

2POR *n1 n2* Z1=*data_label1* Z2=*data_label2*;

2PORT *n1 n2 n3 n4* Z1=*data_label1* Z2=*data_label2*;

Z1 and Z2 are keywords representing the complex terminating impedances at port 1 and port 2, respectively.
data_label1 and *data_label2* each represents a set of imported impedance data.

The impedance data can be supplied in the ImportData block, for example,

```
ImportData
   PARAMETER  NAME=Impedance_A;
   FORMAT  FREQ(GHZ)  MZ11  PZ11;
           0.000      xxx   xxx
           0.500      xxx   xxx
           ...
          10.000      xxx   xxx
End

Model
   ...

   2POR  1 2  Z1=Impedance_A;
End
```

☞ See Section 5.8 for further details on the ImportData block.

5.5 Bias Ports

In addition to the input/output ports, you can define one or two bias ports.

Syntax:

`1BIASPORT` *n5 n6*;

`2BIASPORT` *n5 n6 n7 n8*;

where *n5*, *n6*, *n7* and *n8* represent nodes.

We generically use the FET terminology by labelling the first bias port as the "gate" bias port and the second bias port as the "drain" bias port.

The number of bias ports depends on the type of nonlinear device. Typically, transistors require 2BIASPORT and diodes require 1BIASPORT. The number of bias ports is totally independent of the number of input and output ports. For instance, a diode circuit usually requires one bias port, but the overall circuit can be configured as either a one-port or a two-port. Different combinations of PORT and BIASPORT definitions together with the corresponding notation are explained in Figs. 5.5 through 5.8.

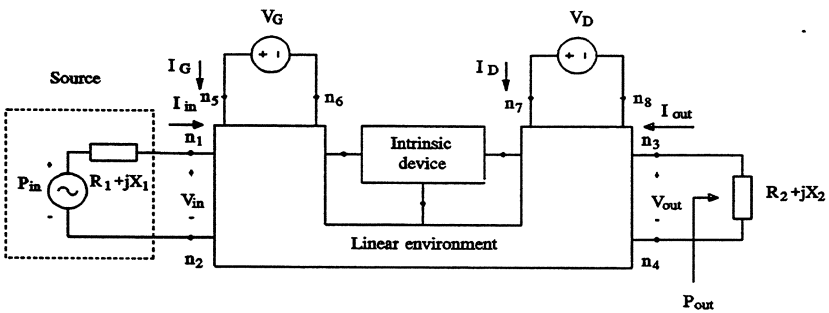


Fig. 5.5 2PORT *circuit with* 2BIASPORT*: topology and notation.*

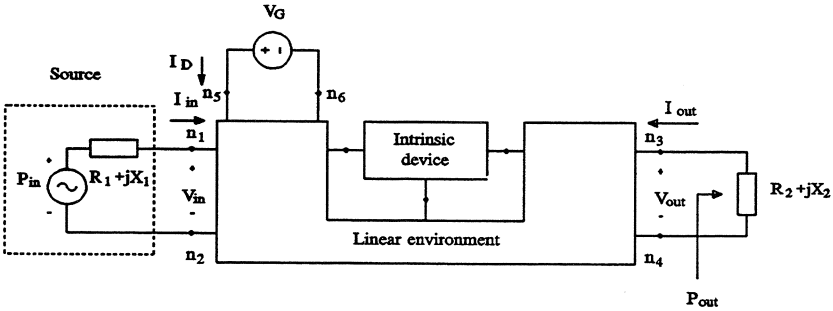


Fig. 5.6 2PORT *circuit with* 1BIASPORT*: topology and notation.*

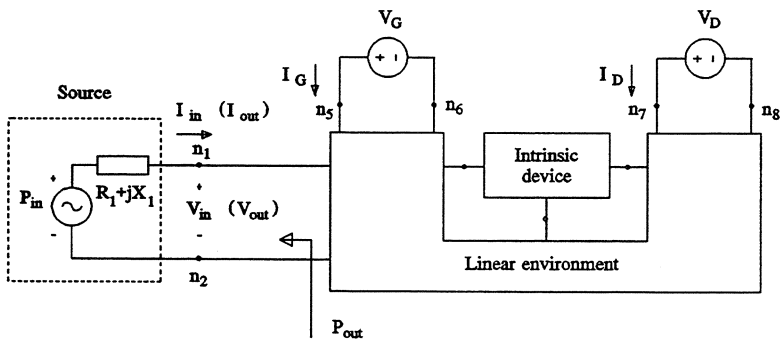


Fig. 5.7 1PORT *circuit with* 2BIASPORT*: topology and notation.*

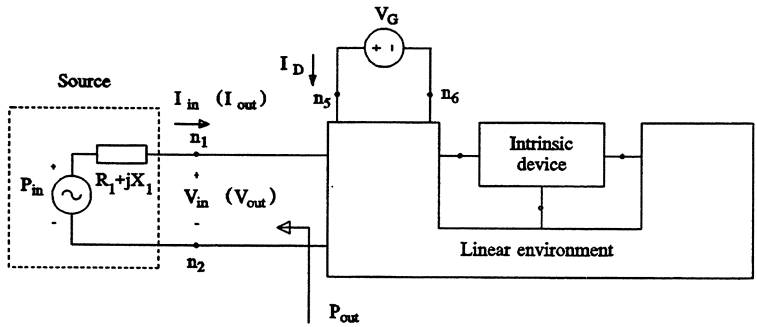


Fig. 5.8 1PORT *circuit with* 1BIASPORT*: topology and notation.*

Combinations of POR and BIASPORT definitions are similar to those of Figs. 5.5 - 5.8, except that the input and output ports are grounded. In this case, the nodes of the input and output ports are numbered as illustrated in Figs. 5.1 and 5.3.

If explicitly defined, the bias ports must be different from the input and output ports.

Implicitly Defined Bias Ports

If the bias ports are not explicitly specified, bias voltages will be applied directly to the input and output ports. In other words, the input and output ports also serve as bias ports. This implies that the circuit has the same number of bias ports and input/output ports.

The circuit topologies in the DC and AC cases for 1PORT and 2PORT circuits are depicted in Figs. 5.9 through 5.12. The cases of 1POR and 2POR circuits are similar except that all ports are grounded and the nodes are numbered as in Fig. 5.1 and Fig. 5.3.

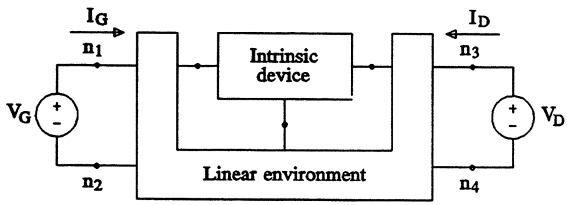


Fig. 5.9 2PORT *DC circuit when bias ports are not explicitly specified.*

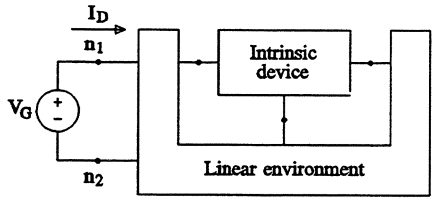


Fig. 5.10 1PORT *DC circuit when bias ports are not explicitly specified.*

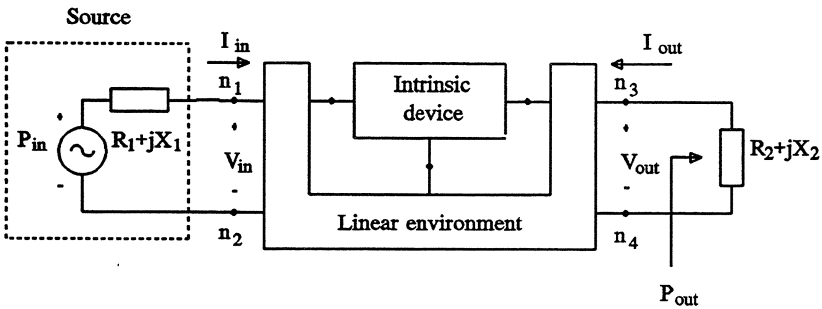


Fig. 5.11 2PORT *AC circuit when bias ports are not explicitly specified.*

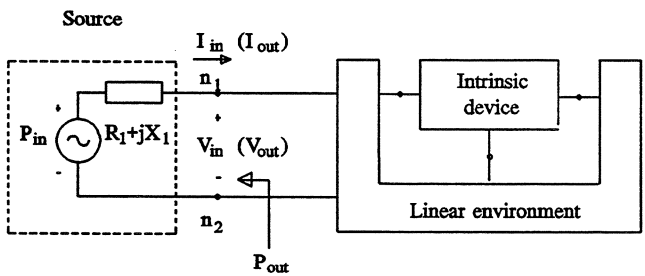


Fig. 5.12 1PORT *AC circuit when bias ports are not explicitly specified.*

Summary of Explicit and Implicit Bias Ports

HarPE allows for both an explicit description of biasing circuits and an implicit "short-cut" approach when perfect isolation of DC and RF ports can be assumed.

The four cases handled by HarPE are as follows.

Explicit one bias port - a 1BIASPORT statement is included in the MODEL block.

Implicit one bias port - a 1POR or 1PORT statement is used and neither 1BIASPORT nor 2BIASPORT statement is included in the MODEL block.

Explicit two bias ports - a 2BIASPORT statement is included in the MODEL block.

Implicit two bias ports - a 2POR or 2PORT statement is used and neither 1BIASPORT nor 2BIASPORT statement is included in the MODEL block.

5.6 Current-Driven Devices

This section describes the option of base current bias source. It is designed to accommodate measurement setups used for bipolar devices. Base current bias source can be specified by adding the keyword CURRENT_DRIVEN to the input/output port definition.

Syntax:

1POR *n1* CURRENT_DRIVEN;

1PORT *n1 n2* CURRENT_DRIVEN;

2POR *n1 n2* CURRENT_DRIVEN;

2PORT *n1 n2 n3 n4* CURRENT_DRIVEN;

where *n1*, *n2*, *n3* and *n4* represent nodes. Port terminating impedances can be specified as described in Section 5.4.

This instructs HarPE to treat the first bias port (or the only bias port in the case of one bias port) as a current source. Its current will become an input parameter and the voltage across it will be an output response.

Predefined Labels and Keywords

The meaning of the predefined labels VG, IG, ID and ID0 (☞ Chapter 4) is changed when the CURRENT_DRIVEN option is specified, as illustrated in Figs. 5.13 and 5.14.

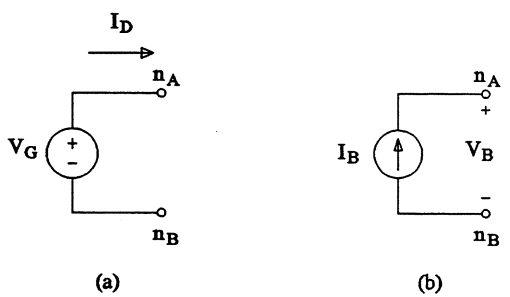


Fig. 5.13 Corresponding notation in the case of explicit or implicit one bias port for (a) voltage bias source and (b) current bias source. When the CURRENT_DRIVEN *option is specified (case (b)), the label* VG *represents the bias source current I_B and the label* ID *represents the voltage V_B.*

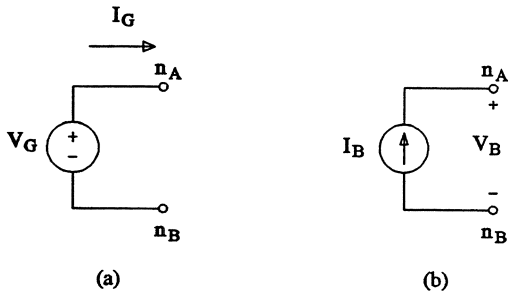


Fig. 5.14 Corresponding notation in the case of explicit or implicit two bias ports for (a) voltage bias source and (b) current bias source. When the CURRENT_DRIVEN *option is specified (case (b)), the label* VG *represents the bias source current I_B and the label* IG *represents the voltage V_B.*

Fig. 5.13 applies to the situations depicted in Figs. 5.6, 5.8 and 5.10. Fig. 5.14 applies to the situations depicted in Figs. 5.5, 5.7 and 5.9. The nodes n_A and n_B in Figs. 5.13 and 5.14 correspond to the nodes n_5 and n_6, respectively, in Figs. 5.5 through 5.8, and to the nodes n_1 and n_2, respectively, in Figs. 5.9 and 5.10.

The source current of Fig. 5.13(b) corresponds to the source voltage of Fig. 5.13(a) and the resulting voltage of Fig. 5.13(b) corresponds to the resulting current of Fig. 5.13(a). The same correspondence exists between Fig. 5.14(b) and 5.14(a).

For example, if the circuit contains two bias ports and the option CURRENT_DRIVEN is specified, then you use the predefined label VG to specify the source current IB (in amperes) and the predefined label IG to specify VB (in volts), as shown in Fig. 5.14.

VG, IG and ID are also used as keywords for defining simulation ranges, measured data and optimization goals (see Chapters 8 and 9). When CURRENT_DRIVEN is specified, these keywords are interpreted by HarPE in the same way as for the predefined labels.

Include Files Containing Macros for Your Convenience

Although VB and IB are not directly available as predefined labels in HarPE, you can create them as aliases using the file preprocessor macros (☞ Chapter 3 and Section 5.9). For your convenience two include files: "current1.inc" and "current2.inc" are supplied together with HarPE. They contain the macro definitions to rename VG and IG (ID) to IB and VB, as illustrated in Figs. 5.13 and 5.14.

Notation for the Outputs

The notation for the graphical and numerical outputs from HarPE is automatically selected. When CURRENT_DRIVEN is specified, HarPE produces the outputs in accordance with the notation shown in Fig. 5.13(b) or 5.14(b).

5.7 User-Defined Device Models

User-defined device models allow you to specify the model equations using expressions. The nonlinear currents and charges in the model are expressed as functions of arbitrary model parameters and the predefined voltage labels within a fixed topology.

A number of user-defined intrinsic device models with different topologies are available (☞ Chapter 6).

For example, consider the Curtice cubic FET model. It is available as a built-in model (FETC), but you can also recreate it using the user-definable model FETU1:

```
Expression
   Beta: ...;
   Vds0: ...;
   A0: ...;
   A1: ...;
   A2: ...;
   A3: ...;
   Gamma: ...;

   V1 = VGS_TAU * (1 + Beta * (Vds0 - VDS_T));
   Ids_Model = (A0 + A1*V1 + A2*V1^2 + A3*V1^3)
               * tanh(Gamma*VDS_T);
   ...
End

Model
   FETU1 @gate @drain @source
      TAU = ...
      IGS = ...
      IDS = Ids_Model
      QGS = ...;

   ...

End
```

where Beta, Vds0, A0, A1, A2, A3 and Gamma represent user-defined model parameters (they can be given arbitrary names), and VGS_TAU and VDS_T are predefined labels representing the time-domain intrinsic voltages.

In this example, using FETU1 instead of the built-in FETC gives you the opportunity to modify the model equations if there is a need to do so.

Preprogrammed Models

A number of nonlinear FET models have been preprogrammed as user-defined models. They are contained in the include files listed in Table 5.1 and supplied together with the HarPE. You can incorporate these preprogrammed models directly into your own circuit file using the #include preprocessor directive (☞ Chapter 3).

TABLE 5.1 PREPROGRAMMED MODELS

File	Model
curtsym.inc	Curtice and Ettenberg cubic symmetrical FET model
curtasy.inc	Curtice and Ettenberg cubic asymmetrical FET model
materka.inc	Materka and Kacprzak FET model
npn.inc	NPN bipolar transistor model (Gummel and Poon)
plessey.inc	Plessey FET model
pnp.inc	PNP bipolar transistor model (Gummel and Poon)
statz.inc	Raytheon (Statz *et al.*) FET model
tom.inc	TriQuint's Own Model (TOM) for FETs

5.8 ImportData Block

In addition to the built-in library of linear elements, HarPE also allows you to import linear n-ports ("black-box data") as part of the circuit description.

A linear subcircuit to be imported must be an n-port with a common reference node (i.e., all the ports are defined w.r.t. to the same reference node). The subcircuit must be represented by its scattering, admittance or impedance matrix ($S/Y/Z$ matrix) supplied in the ImportData block.

Syntax:

```
ImportData
   PARAMETER ...;
   FORMAT ...;
   S/Y/Z data

   ...

   PARAMETER ...;
   FORMAT ...;
   S/Y/Z data
End
```

The block may contain a number of subcircuits, each represented by a data set which consists of a PARAMETER statement, a FORMAT statement and the $S/Y/Z$ data.

PARAMETER Statement

Each import data set begins with a PARAMETER statement:

Syntax:

```
PARAMETER  NAME=label  RREF=x;
```

where *label* is a character string to identify the set of imported data.
If the data is S parameters, RREF can be used to specify the reference impedance.
The default reference impedance is 50OH.

The *label* following the keyword NAME is required, so that when the imported subcircuit is used in the Model block, it can be identified by *label*.

The QUIET Keyword

An optional keyword QUIET can be used to instruct HarPE to ignore unrecognizable entries. This feature makes it more friendly for you to share data files between HarPE and other programs, because you will be able to keep keywords which are unknown to HarPE but meaningful to other applications.

Syntax:

PARAMETER NAME=*label* **RREF=***x* **QUIET** *foreign_keyword=...*;

Example:

```
PARAMETER  ...  QUIET  TEMPERATURE=273  CODE=12;
```

where TEMPERATURE and CODE are keywords unknown to HarPE but may be useful for another program. With the QUIET keyword, the entries TEMPERATURE=273 and CODE=12 will be ignored. Otherwise they would be treated as errors.

FORMAT Statement

For each data set, following the PARAMETER statement, a FORMAT statement is required.

Syntax:

FORMAT *keyword(unit) keyword(unit) ...*;

where *keyword* represents a data keyword and *unit* is optional.

Example:

```
PARAMETER  NAME=Import_2_Port;
FORMAT  FREQ(GHZ)  RY11 IY11 RY12 IY12 RY21 IY21 RY22 IY22;

1.0  0.0412 -0.6259 -0.0392 0.6266 -0.0392 0.6266 0.0392 -0.6266
2.0  0.0118 -0.3120 -0.0098 0.3133 -0.0098 0.3133 0.0098 -0.3133
5.0  0.0035 -0.1222 -0.0015 0.1253 -0.0015 0.1253 0.0015 -0.1253
6.0  0.0031 -0.1007 -0.0011 0.1044 -0.0011 0.1044 0.0011 -0.1044
7.0  0.0028 -0.0851 -0.0008 0.0895 -0.0008 0.0895 0.0008 -0.0895
```

Each keyword in the FORMAT statement represents a column of data. Using the FORMAT statement, you can indicate the data type, order and units, without being restricted to a rigid order of data entry.

The keywords for imported data are listed in Table 5.1.

TABLE 5.1 KEYWORDS FOR IMPORTED DATA

Keyword	Default Unit	Description
FREQ	GHZ	frequency
MS*ij*		magnitude of *Sij*
PS*ij*	DEG	phase of *Sij*
RS*ij*		real part of *Sij*
IS*ij*		imaginary part of *Sij*
MY*ij*	/OH	magnitude of *Yij*
PY*ij*	DEG	phase of *Yij*
RY*ij*	/OH	real part of *Yij*
IY*ij*	/OH	imaginary part of *Yij*
MZ*ij*	OH	magnitude of *Zij*
PZ*ij*	DEG	phase of *Zij*
RZ*ij*	OH	real part of *Zij*
IZ*ij*	OH	imaginary part of *Zij*

☞ See Chapter 3 for the complete list of physical units.

All the keywords used in one FORMAT statement (one data set) must be consistent. In other words, you cannot mix *S* parameters with *Y* parameters, or mix rectangular form with polar form, in the same data set.

The number of ports is automatically determined from the highest index among the keywords specified. The maximum number of ports is 9.

You must supply complete *S/Y/Z* matrices. For instance, a set of *Y* parameters for a two-port must include Y_{11}, Y_{12}, Y_{21}, and Y_{22}.

The *S/Y/Z* matrix for each frequency may occupy as many text lines as necessary, for example,

```
PARAMETER  NAME=Import_2_Port;
FORMAT  FREQ(GHZ)  RY11 IY11 RY12 IY12 RY21 IY21 RY22 IY22;

        1.0    0.0412   -0.6259   -0.0392    0.6266
              -0.0392    0.6266    0.0392   -0.6266

        2.0    0.0118   -0.3120   -0.0098    0.3133
              -0.0098    0.3133    0.0098   -0.3133

        ...
```

However, each matrix (frequency) must begin on a new line.

Data Interpolation

The frequencies supplied for the imported data need not be exactly the same frequencies involved in the simulation by HarPE. If necessary, HarPE will interpolate or extrapolate the imported data using cubic splines. This requires that the imported data include at least three different frequencies. It is desirable and recommended that the imported data cover the frequency range of the simulation. For harmonic simulation, the frequency range is from DC to the highest harmonic frequency.

Super Keywords

For your convenience, a set of super keywords are defined to represent typical sets of keywords for imported subcircuit data. These are listed in Table 5.2.

TABLE 5.2 SUPER KEYWORDS FOR IMPORTED DATA

Super Keyword		Sequence of Keywords Represented
S2MP		MS11, PS11, MS21, PS21, MS12, PS12, MS22, PS22
S2RI		RS11, IS11, RS21, IS21, RS12, IS12, RS22, IS22
SnMP	n ≠ 2	MS11, PS11, MS12, PS12, ..., MS1n, PS1n, ..., MSnn, PSnn
SnRI	n ≠ 2	RS11, IS11, RS12, IS12, ..., RS1n, IS1n, ..., RSnn, ISnn
YnMP		MY11, PY11, MY12, PY12, ..., MY1n, PY1n, ..., MYnn, PYnn
YnRI		RY11, IY11, RY12, IY12, ..., RY1n, IY1n, ..., RYnn, IYnn
ZnMP		MZ11, PZ11, MZ12, PZ12, ..., MZ1n, PZ1n, ..., MZnn, PZnn
ZnRI		RZ11, IZ11, RZ12, IZ12, ..., RZ1n, IZ1n, ..., RZnn, IZnn

All the keyword sequences represent matrices entered row-wise,
except S2MP and S2RI which are arranged column-wise as conventionally done.

Example:

```
FORMAT  FREQ Y2RI;
```

which is equivalent to

```
FORMAT  FREQ RY11 IY11 RY12 IY12 RY21 IY21 RY22 IY22;
```

You can also specify a data unit, such as

```
FORMAT  FREQ  Z2RI(KOH);
```

which is equivalent to

```
FORMAT  FREQ  RZ11(KOH) IZ11(KOH) RZ12(KOH) IZ12(KOH)
              RZ21(KOH) IZ21(KOH) RZ22(KOH) IZ22(KOH);
```

The QUIET Keyword in FORMAT

An optional keyword QUIET can be used to instruct HarPE to ignore unrecognizable entries. This feature makes it more friendly for you to share data files between HarPE and other programs, because you will be able to keep keywords which are unknown to HarPE but meaningful to other applications.

Syntax:

```
FORMAT  keyword(unit) ...  QUIET foreign_keyword(unit) ...;
```

Example:

```
FORMAT  QUIET  FREQ  MS11  PS11  RETURN_LOSS(DB);
```

where RETURN_LOSS(DB) is unknown to HarPE as an imported data keyword. With QUIET, you instruct HarPE to ignore such foreign keywords and to skip over the corresponding columns of data.

Reference to Imported Data

Imported linear subcircuits can be used as nPORDATA elements in the Model block.

Syntax:

*n*PORDATA *m1 m2 ... mk* DATA=*label*;

where n is the number of ports, $1 \le n \le 9$.
label identifies an imported data set in the ImportData block.
m1, *m2*, ... *mk* are nodes, $k = n$ or $(n + 1)$. If $k = n + 1$, then the reference node of the imported subcircuit is connected to the node *mk* which can be nonzero. If $k = n$, then the reference node is grounded by default.

Example:

```
ImportData
   PARAMETER  NAME=Imported_2_Port;
   FORMAT  FREQ(GHZ)  Y2RI(/KOH);
           0.000   xxx xxx xxx xxx xxx xxx xxx xxx
           0.500   xxx xxx xxx xxx xxx xxx xxx xxx
           ...
          10.000   xxx xxx xxx xxx xxx xxx xxx xxx
End

Model
   2PORDATA  1 2 3  DATA=Imported_2_Port;
   ...
End
```

The imported data must describe an n-port with a common reference node (i.e., all the ports are defined w.r.t. to the same reference node). The nPORDATA statement and the import data it refers to must have the same number of ports.


Also see Chapter 7 for nPORDATA.

Imported One-Ports as Terminations

Imported one-ports can be used to define frequency-dependent terminations for the input and output ports of the circuit (Section 5.4).

Example:

```
ImportData
   PARAMETER  NAME=Imported_Impedance;
   FORMAT  FREQ(GHZ)  MZ11  PZ11;
           0.000      xxx   xxx
           0.500      xxx   xxx
           ...
          10.000      xxx   xxx
End

Model
   ...

   2PORT  10 0 11 0  Z1=Imported_Impedance;
End
```

The use of the keywords Z1 and Z2 in the port definition does not imply that the data supplied in the ImportData block can only be *Z* (impedance) data. The data supplied can also be *S* or *Y* parameters, HarPE will automatically make the necessary conversion. However, the imported data must be convertible to a finite and nonzero impedance value at all the frequencies involved in the circuit simulation.

Direct Use of Data in EEsof Format

Data in EEsof format can be directly included in the ImportData block and used for nPORDATA elements.

Syntax:

```
ImportData
   #include "file_name.snp"
End

Model
  ...

   nPORDATA  m1 m2 ... mk  DATA=file_name;

   ...
End
```

where n is the number of ports, $1 \le n \le 9$.
file_name is the prefix of the name of the data file in EEsof format.
m1, *m2*, ... *mk* are nodes, $k = n$ or $(n + 1)$. If $k = n + 1$, then the reference node of the imported subcircuit is connected to the node *mk* which can be nonzero. If $k = n$, then the reference node is grounded by default.

Example:

```
ImportData
   #include "nec700.s2p"     ! EEsof S-parameter data
End

Model
   ...

   2PORDATA  1 2 3  Data=nec700;

   ...
End
```

5.9 Macros and Symbolic Subcircuits

The text macros supported by HarPE's file parser are very useful for simplifying circuit files and customizing the file syntax (☞ Chapter 3).

Text macros are identifiers (names) created by the #define preprocessor directive to replace complex and repetitive portions of text.

Customized Keywords

Text macros can be used to customize the syntax by replacing the built-in keywords with identifiers which are more readable, more familiar to you, or compatible with some other programs.

Example:

```
#define  User_Definable_FET_Model      FETU1
#define  Gate_Source_Current           IGS
#define  Gate_Drain_Current            IGD
#define  Drain_Source_Current          IDS
#define  Gate_Source_Charge            QGS
#define  Gate_Drain_Charge             QGD

User_Definable_FET_Model  1  2  3
   Gate_Source_Current = expression1
   Gate_Drain_Current = expression2
   Drain_Source_Current = expression3
   Gate_Source_Charge = expression4
   Gate_Drain_Charge = expression5;
```

These macros will be expanded by the file parser into

```
FETU1  1 2 3
   IGS = expression1
   IGD = expression2
   IDS = expression3
   QGS = expression4
   QGD = expression5;
```

In this example, the built-in keywords are replaced by lengthier but less cryptic text.

Another example:

```
#define  Resistor   RES
#define  Resistance(length, area)   R=(Resistivity*length/area)

Resistor  @n1 @n2  Resistance(L1,A1);
Resistor  @n3 @n4  Resistance(L2,A2);
```

The expanded text becomes

```
RES  @n1 @n2  R=(Resistivity*L1/A1);
RES  @n3 @n4  R=(Resistivity*L2/A2);
```

You can even redefine circuit file block names, such as

```
#define  CKT   Model
```

which allows you to use a CKT block instead of a Model block for circuit definition.

Symbolic Subcircuits

A more sophisticated usage of macros is to define symbolic subcircuits.

A symbolic subcircuit can be quite simple, such as

```
#define Transmission_Line(n1, n2)     TEM n1 n2  Z=40 E=34 F=10GHz
```

which defines a subcircuit containing a single transmission line with constant parameters. The two arguments are the nodes. By using Transmission_line with different arguments, you can place different lines at the appropriate nodes. For example, Transmission_Line(10,14) places a line between the nodes 10 and 14. By modifying the macro definition, you can change all the lines simultaneously.

The $ argument (☞ Chapter 3) is very useful for defining symbolic subcircuits. For example,

```
#define  T_SECTION(n1, n2, $) {
             IND  n1 @center_node$  L=10nH;
             IND  n2 @center_node$  L=10nH;
             CAP  @center_node$ @ground  C=0.5pF
}
```

This defines a grounded LC T-section between the nodes n1 and n2. The T-section contains an internal node which must be a different node each time the subcircuit is placed. Instead of passing a new node, a new index is passed as the $ argument, and a new internal node is constructed using a template and the $ argument.

For example,

```
T_SECTION(17, 20, 1);
T_SECTION(33, 14, 2);
```

The corresponding internal nodes will be @center_node1 and @center_node2.

Notes and Cautions

Symbolic subcircuits are not actual subcircuits. They are not treated as separate structures in circuit simulation. Rather, symbolic subcircuits are expanded by the preprocessor into the overall circuit. The overall circuit is simulated on a single level.

The use of symbolic subcircuits provides convenience but does not in general lead to computational savings.

Symbolic subcircuits can be nested, i.e., a symbolic subcircuit can contain other symbolic subcircuits, up to 6 levels (layers).

Symbolic subcircuits cannot contain explicit definition of optimization variables, otherwise the explicit definition will be duplicated each time the subcircuit is used. This gives rise to ambiguity and difficulties in updating the variables after optimization. Optimization variables should be *defined* as labels outside the symbolic subcircuit and then *referenced* in the symbolic subcircuit.

6

Nonlinear Elements

6

Nonlinear Elements

This chapter describes HarPE's library of nonlinear element models.

The description for each model includes a schematic diagram, the number of nodes, the set of parameters and their default values, an example of the syntax, the model equations and some application notes if appropriate.

The nodes for each element can be divided into two subsets. The first subset represents nodes which must be nonzero for the element to be meaningfully connected (e.g., the element DIODE requires at least one nonzero node). The other set represents nodes which are optional. In the description, the optional nodes are enclosed within square brackets. The default for optional nodes that are not explicitly specified is the ground node.

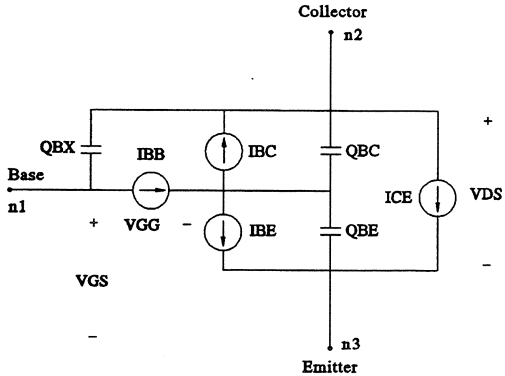
All the device model parameters are optional except where noted. Model parameters can be specified by constant values, optimization variables and labels. Parameters that are omitted will be assigned the default values.

The nonlinear element library includes

- built-in device models for FETs, HEMTs, bipolar transistors and diodes, with fixed topology and built-in model equations
- user-definable device models for FETs, bipolar transistors and diodes, with fixed topology and user-defined model equations

☞ See Chapter 5 for further details on creating user-defined models in the Model block.

BJTU user-definable bipolar junction transistor model BJTU



Keywords

		Defaults
IBE	base-emitter current	0
IBC	base-collector current	0
ICE	collector-emitter current	0
IBB	external base-internal base current	0
QBE	base-emitter capacitor charge	0
QBC	base-collector capacitor charge	0
QBX	external base-collector capacitor charge	0

Form

```
BJTU  n1 n2 [n3]  IBE = expression1  IBC = expression2  ICE = expression3
                  IBB = expression4  QBE = expression5  QBC = expression6
                  QBX = expression7;
```

To utilize BJTU, you can create a set of model parameters of arbitrary meaning and names, and then define the nonlinear currents and charges as functions (expression1, ..., expression7) of these model parameters and the time-domain intrinsic voltages which are represented by the predefined labels VGS_T, VDS_T and VGG_T.

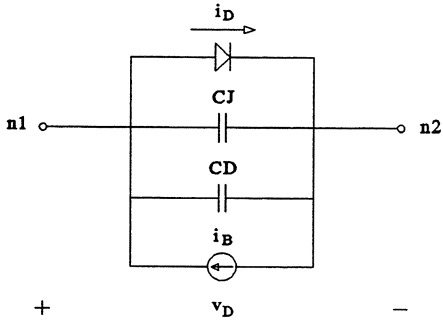
VGS_T	represents the instantaneous external base-emitter voltage.
VDS_T	represents the instantaneous collector-emitter voltage.
VGG_T	represents the instantaneous external base-internal base voltage.

BJTU user-definable bipolar junction transistor model BJTU

The nonlinear charge sources are represented in the schematic diagram by capacitors to emphasize their capacitive nature. The charge source QBE is directed from the base toward the emitter. The charge source QBC is directed from the base toward the collector. The charge source QBX is directed from the external base toward the collector.

Examples are provided in the include files npn.inc and pnp.inc which implement the Gummel-Poon NPN and PNP models using BJTU.

DIODE — semiconductor diode model — DIODE



Keywords

		Defaults
IS	saturation current	1.0E-14
N	emission coefficient	1.0
TEMP	temperature	298°K
VB	breakdown voltage	$-\infty$
IBO	breakdown current at v_D = VB — 1	0
EB	breakdown power-law exponent	10
CJ0	zero-bias junction capacitance	0
VJ	junction potential	0.8
FC	coefficient for forward bias in capacitance equation	0.5
EJ	junction capacitance exponent	0.5
CD0	zero-bias diffusion capacitance	0

Form

```
DIODE  n1 [n2]  IS = x1   N = x2   TEMP = x3  VB = x4   IBO = x5  EB = x6
                CJO = x7  VJ = x8  FC = x9    EJ = x10  CDO = x11;
```

Example

```
DIODE  1 2  IS=6.4e-15  N=1.6  CJO=0.04PF  VJ=0.9  FC=0.75;
```

DIODE semiconductor diode model DIODE

Model Equations

DIODE is a generic semiconductor diode model suitable for both PN junction diodes and Schottky barrier diodes [1].

All components of the equivalent circuit are assumed to be functions of the diode voltage v_D indicated in the diagram. The nonlinear model equations are as follows.

$$i_D = IS \cdot [\ \exp((v_D \cdot q)/(N \cdot k \cdot TEMP)) - 1\ ]$$

where q is the electron charge and k is the Boltzmann constant. IS, N and TEMP are model parameters.

$$i_B = \begin{cases} IB0 \cdot (VB - v_D)^{EB} & \text{for } v_D < VB \\ 0 & \text{for } v_D \geq VB \end{cases}$$

where IB0, VB and EB are model parameters.

$$C_J = \begin{cases} CJ0 \cdot (1 - v_D/VJ)^{-EJ} & \text{for } v_D < FC \cdot VJ \\ CJ0 \cdot (1 - FC)^{-EJ} & \text{for } v_D \geq FC \cdot VJ \end{cases}$$

where CJ0, VJ, EJ and FC are model parameters.

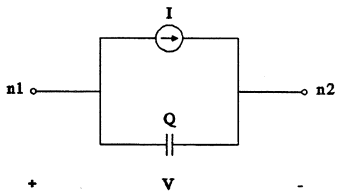
$$C_D = CD0 \cdot \exp((v_D \cdot q)/(N \cdot k \cdot TEMP))$$

where q is the electron charge and k is the Boltzmann constant. CD0, N and TEMP are model parameters.

Reference

[1] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design.* New York, NY: Van Nostrand Reinhold, 1983, pp. 309-312.

DIODEU user-definable diode model DIODEU



Keywords

		Defaults
I	diode current	0
Q	diode capacitor charge	0

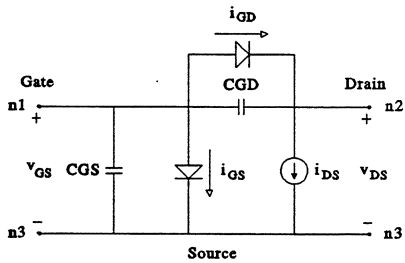
Form

```
DIODEU  n1 [n2]   I = expression1   Q = expression2;
```

To utilize DIODEU, you can create a set of model parameters of arbitrary names and meaning, and then define the nonlinear diode current and charge as functions (expression1 and expression2) of these model parameters and the time-domain voltage across the diode which is represented by the predefined label VGS_T.

The nonlinear charge source is represented in the schematic diagram by a capacitor to emphasize its capacitive nature. The charge source Q is directed from the node n1 toward the node n2.

FETC — Curtice and Ettenberg FET model — FETC



Keywords

		Defaults
A0	coefficient of the cubic formula for i_{DS}	0
A1	coefficient of the cubic formula for i_{DS}	0
A2	coefficient of the cubic formula for i_{DS}	0
A3	coefficient of the cubic formula for i_{DS}	0
GAMMA	hyperbolic tangent function parameter	0
BETA	coefficient for pinch-off change with respect to v_{DS}	0
VDS0	v_{DS} at which A0, A1, A2 and A3 were evaluated	0
TAU	transit time under gate	0
IS	gate junction saturation current	1.0E-14
N	gate-drain and gate-source emission coefficient	1.0
CGD0	zero-bias gate-drain junction capacitance	0
CGS0	zero-bias gate-source junction capacitance	0
FC	coefficient for forward bias in capacitance equation	0.5
GMIN	linear conductance associated with the Schottky junctions	1.0E-12
VBI	built-in gate potential	0.8
VBR	gate reverse bias breakdown voltage	+∞
TEMP	temperature	298°K

Form

```
FETC  n1 n2 n3  A0 = x1     A1 = x2     A2 = x3    A3 = x4     GAMMA = x5
                BETA = x6   VDS0 = x7   TAU = x8   IS = x9     N = x10
                CGD0 = x11  CGS0 = x12  FC = x13   GMIN = x14  VBI = x15
                VBR = x16   TEMP = x17;
```

Example

```
FETC  1 2 3  A0=0.06  A1=0.1  A2=0.06  GAMMA=2  VDS0=2  IS=3nA;
```

FETC — Curtice and Ettenberg FET model — FETC

Model Equations

FETC implements the cubic symmetrical Curtice and Ettenberg model based on [1].

All components of the equivalent circuit are assumed to be functions of the gate and drain voltages v_{GS} and v_{DS} as shown in the diagram. The nonlinear model equations are as follows.

$$i_{DS} = \begin{cases} (A0+A1 \cdot v1+A2 \cdot v1^2+A3 \cdot v1^3) \cdot \tanh(GAMMA \cdot v_{DS}) & \text{for } v_{DS} \geq 0 \\ 0 & \text{for } v_{DS} < 0 \end{cases}$$

where

$$v1(t) = v_{GS}(t - TAU) \cdot (1 + BETA \cdot (VDS0 - v_{DS}(t)))$$

and A0, A1, A2, A3, BETA, GAMMA, TAU and VDS0 are model parameters.

$$i_{GS} = \begin{cases} IS \cdot [\exp(v_{GS}/(N \cdot v_t)) - 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \geq -5N \cdot v_t \\ -IS + GMIN \cdot v_{GS} & \text{for } -VBR+50v_t < v_{GS} < -5N \cdot v_t \\ -IS \cdot [\exp(-(v_{GS}+VBR)/v_t) + 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \leq -VBR+50v_t \end{cases}$$

where $v_t = k$·TEMP$/q$ is the thermal voltage. IS, N, GMIN, VBR and TEMP are model parameters.

The diode current i_{GD} is similarly defined.

$$C_{GS} = \begin{cases} CGS0 \cdot (1-v_{GS}/VBI)^{-0.5} & \text{for } v_{GS} < FC \cdot VBI \\ CGS0 \cdot (1-FC)^{-0.5} \cdot (1+(v_{GS}-FC \cdot VBI)/(2 \cdot VBI \cdot (1-FC))) & \text{otherwise} \end{cases}$$

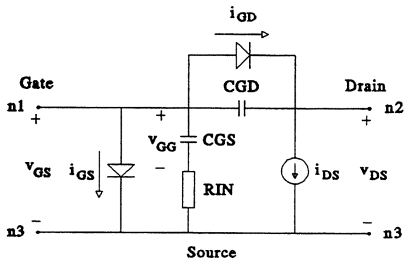
where CGS0, VBI and FC are model parameters.

The gate-drain capacitance C_{GD} is similarly defined.

Reference

[1] W.R. Curtice and M. Ettenberg, "A nonlinear GaAs FET model for use in the design of output circuits for power amplifiers", *IEEE Trans. Microwave Theory Tech.*, vol. MTT-33, 1985, pp. 1383-1394.

FETCA — Curtice asymmetrical FET model — FETCA



Keywords

		Defaults
A0	coefficient of the cubic formula for i_{DS}	0
A1	coefficient of the cubic formula for i_{DS}	0
A2	coefficient of the cubic formula for i_{DS}	0
A3	coefficient of the cubic formula for i_{DS}	0
GAMMA	hyperbolic tangent function parameter	0
BETA	coefficient for pinch-off change with respect to v_{DS}	0
VDS0	v_{DS} at which A0, A1, A2 and A3 were evaluated	0
TAU	transit time under gate	0
IS	gate junction saturation current	1.0E-14
N	gate-drain and gate-source emission coefficient	1.0
CGD0	zero-bias gate-drain junction capacitance	0
CGS0	zero-bias gate-source junction capacitance	0
FC	coefficient for forward bias in capacitance equation	0.5
GMIN	linear conductance associated with the Schottky junctions	1.0E-12
VBI	built-in gate potential	0.8
VBR	gate reverse bias breakdown voltage	+∞
TEMP	temperature	298°K
RIN	intrinsic channel gate-source resistance	1.0

Form

```
FETCA  n1 n2 n3  A0 = x1     A1 = x2     A2 = x3     A3 = x4     GAMMA = x5
                 BETA = x6   VDS0 = x7   TAU = x8    IS = x9     N = x10
                 CGD0 = x11  CGS0 = x12  FC = x13    GMIN = x14  VBI = x15
                 VBR = x16   TEMP = x17  RIN = x18;
```

Example

```
FETCA  1 2 3  A0=0.06  A1=0.1  A2=0.06  GAMMA=2  VDS0=2  IS=3nA  RIN=2.5;
```

FETCA — Curtice asymmetrical FET model — FETCA

Model Equations

FETCA implements the cubic asymmetrical Curtice and Ettenberg model based on [1].

The nonlinear components of the equivalent circuit are functions of the voltages v_{GS}, v_{GG} and v_{DS} as shown in the diagram. The nonlinear model equations are as follows.

$$i_{DS} = \begin{cases} (A0+A1 \cdot v1+A2 \cdot v1^2+A3 \cdot v1^3) \cdot \tanh(GAMMA \cdot v_{DS}) & \text{for } v_{DS} \geq 0 \\ 0 & \text{for } v_{DS} < 0 \end{cases}$$

where

$$v1(t) = v_{GS}(t - TAU) \cdot (1 + BETA \cdot (VDS0 - v_{DS}(t)))$$

and A0, A1, A2, A3, BETA, GAMMA, TAU and VDS0 are model parameters.

$$i_{GS} = \begin{cases} IS \cdot [\exp(v_{GS}/(N \cdot v_t)) - 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \geq -5N \cdot v_t \\ -IS + GMIN \cdot v_{GS} & \text{for } -VBR+50v_t < v_{GS} < -5N \cdot v_t \\ -IS \cdot [\exp(-(v_{GS}+VBR)/v_t) + 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \leq -VBR+50v_t \end{cases}$$

where $v_t = k \cdot TEMP/q$ is the thermal voltage. IS, N, GMIN, VBR and TEMP are model parameters.

The diode current i_{GD} is similarly defined.

$$C_{GS} = \begin{cases} CGS0 \cdot (1-v_{GG}/VBI)^{-0.5} & \text{for } v_{GG} < FC \cdot VBI \\ CGS0 \cdot (1-FC)^{-0.5} \cdot (1+(v_{GG}-FC \cdot VBI)/(2 \cdot VBI \cdot (1-FC))) & \text{otherwise} \end{cases}$$

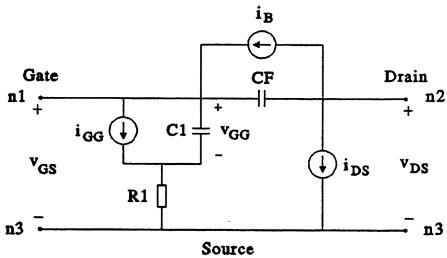
where CGS0, VBI and FC are model parameters.

The gate-drain capacitance C_{GD} is similarly defined.

Reference

[1] W.R. Curtice and M. Ettenberg, "A nonlinear GaAs FET model for use in the design of output circuits for power amplifiers", *IEEE Trans. Microwave Theory Tech.*, vol. MTT-33, 1985, pp. 1383-1394.

FETM — Materka and Kacprzak FET model — FETM



Keywords

		Defaults
IDSS	drain saturation current for v_{GG}=0	6.0E-2
VP0	pinch-off voltage for v_{DS}=0	-2.0
VBC	channel breakdown voltage	+∞
SL	slope of drain current in the linear region for v_{GG}=0	6.0E-2
ALPHAB	slope factor of breakdown current	0
TAU	channel transit-time delay	0
IB0	channel breakdown current for v_{DS} - v_{GS} = VBC	0
KG	parameter describing drain current in the linear region	0
ALPHAG	slope factor of gate conduction current	38.696
E	parameter describing drain saturation current	2.0
IG0	saturation current of gate Schottky barrier	0
KR	voltage slope factor of intrinsic channel resistance	0
GAMMA	voltage-slope parameter of pinch-off voltage	0
KE	parameter describing drain saturation current	0
R10	intrinsic channel resistance for zero gate voltage	3.5
SS	slope of drain current in the saturation region for v_{GG}=0	0
C10	gate Schottky-barrier capacitance for v_{GG}=0	4.0E-4
K1	voltage-slope parameter of gate capacitance	1.25
C1S	linear parasitic gate capacitance	0
CF0	feedback capacitance for v_{DS}=v_{GS}	2.0E-5
KF	voltage-slope parameter of feedback capacitance	0

Form

```
FETM  n1 n2 n3  IDSS = x1  VP0 = x2   VBC = x3     SL = x4      ALPHAB = x5
                TAU = x6   IB0 = x7   KG = x8      ALPHAG = x9  E = x10
                IG0 = x11  KR = x12   GAMMA = x13  KE = x14     R10 = x15
                SS = x16   C10 = x17  K1 = x18     C1S = x19    CF0 = x20
                KF = x21;
```

FETM Materka and Kacprzak FET model FETM

Example

```
FETM  1 2 3  IDSS=6.0E-2  VP0=-1.9  SL=0.0676  R10=3.5
             C10=0.42PF   K1=1.282  CF0=0.02PF;
```

Model Equations

FETM implements the Materka and Kacprzak model based on [1].

All components of the equivalent circuit are assumed to be functions of voltages v_{GS}, v_{GG} and v_{DS} as shown in the diagram. The nonlinear model equations are as follows.

$$C_1 = \begin{cases} C10 \cdot (1 - K1 \cdot v_{GG})^{-0.5} + C1S & \text{for } K1 \cdot v_{GG} < 0.8 \\ C10 \cdot \text{sqrt}(5) + C1S & \text{for } K1 \cdot v_{GG} \geq 0.8 \end{cases}$$

where C10, K1 and C1S are model parameters.

$$C_F = \begin{cases} CF0 \cdot (1 - KF \cdot (v_{GS} - v_{DS}))^{-0.5} & \text{for } KF \cdot (v_{GS} - v_{DS}) < 0.8 \\ CF0 \cdot \text{sqrt}(5) & \text{for } KF \cdot (v_{GS} - v_{DS}) \geq 0.8 \end{cases}$$

where CF0 and KF are model parameters.

$$R_1 = \begin{cases} R10 \cdot (1 - KR \cdot v_{GG}) & \text{for } KR \cdot v_{GG} < 1.0 \\ 0 & \text{for } KR \cdot v_{GG} \geq 1.0 \end{cases}$$

where R10 and KR are model parameters.

$$i_{GG} = IG0 \cdot [\exp(ALPHAG \cdot v_{GG}) - 1]$$

where IG0 and ALPHAG are model parameters.

$$i_B = IB0 \cdot \exp(ALPHAB \cdot (v_{DS} - v_{GS} - VBC))$$

where IB0, ALPHAB and VBC are model parameters.

$$i_{DS} = IDSS \cdot [1 - v_{GG}(t-TAU)/(VP0 + GAMMA \cdot v_{DS})]^{(E + KE \cdot v_{GG}(t-TAU))} \cdot \tanh[SL \cdot v_{DS}/(IDSS \cdot (1 - KG \cdot v_{GG}(t-TAU)))] \cdot [1 + SS \cdot v_{DS}/IDSS]$$

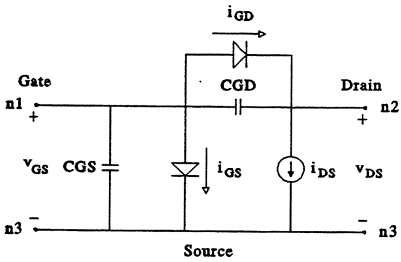
where IDSS, TAU, VP0, GAMMA, E, KE, SL, KG and SS are model parameters.

FETM Materka and Kacprzak FET model FETM

Reference

[1] A. Materka and T. Kacprzak, "Computer calculation of large-signal GaAs FET amplifier characteristics", *IEEE Trans. Microwave Theory Tech.*, vol. MTT-33, 1985, pp. 129-135.

FETR — Raytheon FET model — FETR



Keywords

		Defaults
ALPHA	hyperbolic tangent function parameter	0
BETA	transconductance parameter	0
LAMBDA	channel length modulation parameter	0
THETA	transconductance parameter for large v_{GS}	0
VT0	threshold voltage	-2.0
TAU	transit time under gate	0
IS	gate junction saturation current	1.0E-14
N	gate-drain and gate-source emission coefficient	1.0
CGD0	zero bias gate-drain junction capacitance	0
CGS0	zero bias gate-source junction capacitance	0
FC	coefficient for forward bias in the capacitance equation	0.5
GMIN	linear conductance associated with the Schottky junctions	1.0E-12
VBI	built-in gate potential	0.8
VBR	gate reverse bias breakdown voltage	+∞
TEMP	temperature	298°K

Form

```
FETR  n1 n2 n3  ALPHA = x1   BETA = x2    LAMBDA = x3   THETA = x4
                VT0 = x5     TAU = x6     IS = x7       N = x8
                CGS0 = x9    CGD0 = x10   FC = x11      GMIN = x12
                VBI = x13    VBR = x14    TEMP = x15;
```

Example

```
FETR  1 2 3  BETA=0.02  VT0=-1.2  IS=3.7nA;
```

FETR — Raytheon FET model — FETR

Model Equations

FETR implements the Raytheon model based on the work of Statz *et al.* [1].

All components of the equivalent circuit are assumed to be functions of the gate and drain voltages v_{GS} and v_{DS} as shown in the diagram. The nonlinear model equations are as follows.

$$i_{DS} = \begin{cases} xx1 \cdot xx2 \cdot (1 - (1-ALPHA \cdot v_{DS}/3)^3) & \text{for } 0 < v_{DS} < 3/ALPHA \\ xx1 \cdot xx2 & \text{for } v_{DS} \geq 3/ALPHA \end{cases}$$

where

$$xx1 = (BETA \cdot [v_{GS}(t-TAU) - VT0]^2) \; / \; (1 + THETA \cdot [v_{GS}(t-TAU) - VT0])$$

$$xx2 = 1 + LAMBDA \cdot v_{DS}$$

ALPHA, BETA, LAMBDA, THETA, TAU and VT0 are model parameters.

$$i_{GS} = \begin{cases} IS \cdot [\exp(v_{GS}/(N \cdot v_t)) - 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \geq -5N \cdot v_t \\ -IS + GMIN \cdot v_{GS} & \text{for } -VBR+50v_t < v_{GS} < -5N \cdot v_t \\ -IS \cdot [\exp(-(v_{GS}+VBR)/v_t)+1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \leq -VBR+50v_t \end{cases}$$

where $v_t = k \cdot TEMP/q$ is the thermal voltage. IS, N, GMIN, VBR and TEMP are model parameters.

The diode current i_{GD} is similarly defined.

$$C_{GS} = \begin{cases} CGS0 \cdot (1-v_{GS}/VBI)^{-0.5} & \text{for } v_{GS} < FC \cdot VBI \\ CGS0 \cdot (1-FC)^{-0.5} \cdot (1+(v_{GS}-FC \cdot VBI)/(2 \cdot VBI \cdot (1-FC))) & \text{otherwise} \end{cases}$$

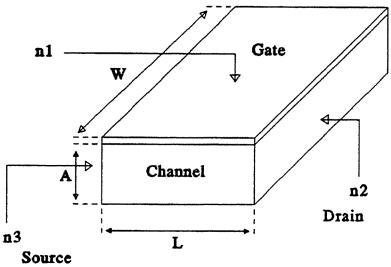
where CGS0, VBI and FC are model parameters.

The gate-drain capacitance C_{GD} is similarly defined.

Reference

[1] H. Statz, P. Newman, I.W. Smith, R.A. Pucel and H.A. Haus, "GaAs FET device and circuit simulation in SPICE", *IEEE Trans. Electron Devices*, vol. ED-34, 1987, pp. 160-169.

FETT physics-based Khatibzadeh and Trew FET model FETT



Keywords

		Defaults
L	gate length in m	0.56E-6
W	gate width in m	1.2E-3
A	channel thickness in m	0.35E-6
U0	low-field mobility in $m^2/(Vns)$	4.0E-10
D	high-field diffusion coefficient in m^2/ns	1.0E-12
EPSR	relative dielectric constant	12.5
VS	saturation electron velocity in m/ns	1.5E-4
VBI	built-in potential in V	0.7
ND	doping density (for uniform doping) in m^{-3}	7.5E22
LAMBDA	parameter for the transition function in m	1.0E-8
ALPHA	exponent for calculating the average mobility	0.5
TAU	time delay in ns	0.0
ND1	parameter for doping profile in m^{-3}	1.0E22
ND2	parameter for doping profile in m^{-3}	1.0E22
Y0	parameter for doping profile in m	1.0E-7
YNOM	parameter for doping profile in m	$+\infty$
YNOM1	parameter for doping profile in m	$+\infty$
AA	parameter for electron velocity curve in V/m	1.0E6
BB	parameter for electron velocity curve	1
CC	parameter for electron velocity curve	1
DD	parameter for electron velocity curve	0

FETT physics-based Khatibzadeh and Trew FET model FETT

Form

```
FETT  n1 n2 n3  L = x1        W = x2        A = x3        U0 = x4
                D = x5        EPSR = x6     VS = x7       VBI = x8
                ND = x9       LAMBDA = x10  ALPHA = x11   TAU = x12
                ND1 = x13     ND2 = x14     Y0 = x15      YNOM = x16
                YNOM1 = x17   AA = x18      BB = x19      CC = x20
                DD = x21;
```

Example

```
FETT  1 2 3  L=0.42UM  W=600UM  A=0.3UM  VBI=0.7  ND=1.0E23  TAU=3PS;
```

Model Equations

FETT implements the physics-based Khatibzadeh and Trew analytical model based on [1]. It is dynamically integrated into the harmonic balance simulation and optimization as described by Bandler, Zhang and Cai [2].

Doping Profile

FETT accommodates uniform and nonuniform doping profiles. Three types of doping profiles, namely uniform, quadratic and piecewise (Khatibzadeh [3]), are available.

Let N(y) represent the doping density function, where y is the depth under the FET gate.

If the model parameter YNOM is not specified, then uniform doping is assumed:

$$N(y) = ND$$

In this case, the parameters ND1, ND2, Y0, YNOM and YNOM1 are ignored.

If YNOM is specified but YNOM1 is not, then quadratic doping approximation is used:

$$N(y) = ND \cdot \exp\{ -0.5 \cdot [(y - Y0) / YNOM]^2 \}$$

In this case, ND1, ND2 and YNOM1 are ignored.

If both YNOM and YNOM1 are specified, then piecewise doping approximation is used:

$$N(y) = \begin{cases} ND & \text{if } y \le Y0 \\ ND1 \cdot \exp\{(Y0-y)/YNOM\} + ND2 \cdot \exp\{(Y0-y)/YNOM1\} & \text{otherwise} \end{cases}$$

FETT physics-based Khatibzadeh and Trew FET model FETT

Electron Velocity - Electric Field Curve

The electron velocity as a function of the electric field is given by

$$v(E) = (1 - DD) \cdot v_1(E) + DD \cdot v_2(E)$$

where E is the electric field, and DD is a model parameter.

$v_1(E)$ is the formula originally used by Khatibzadeh and Trew [1]:

$$v_1(E) = \begin{cases} VS & \text{if } E > 2 \cdot EC \\ U0 \cdot E - VS \cdot E^2 / (4 \cdot EC^2) & \text{if } E \le 2 \cdot EC \end{cases}$$

where VS, U0 are model parameters and EC = VS / U0 is the critical electric field.

The function $v_2(E)$ is added to model the overshooting behaviour as experimentally observed:

$$v_2(E) = VS \cdot (BB/CC) \cdot \{(E/AA - CC)/[(E/AA)^4 + BB] + CC/BB\}$$

where VS, AA, BB and CC are model parameters. AA can be used to adjust the location of the overshoot. BB and CC can be used to adjust the sharpness of the curve peak. A larger value of the ratio BB/CC will lead to a sharper peak.

The parameter DD serves as a weighting factor for combining $v_1(E)$ and $v_2(E)$. Its value should be between 0 and 1.

Transition Function

The transition function models the transition of the doping profile from almost zero near the gate ($y \approx 0$) to the background doping concentration in the channel ($y \approx A$):

$$T(d(x), y) = 1 - \frac{1}{1 + \exp\{(y - d(x))/LAMBDA\}}$$

where d(x) is the effective depletion-layer width calculated internally by the model, and LAMBDA is a model parameter whose value must be of the order of the Debye length.

ALPHA

ALPHA is an empirical model parameter used to calculate the weighted average mobility for the evaluation of the conduction currents [3]. Setting ALPHA = 0.5 is experimentally found to give good results for the current level.

FETT physics-based Khatibzadeh and Trew FET model FETT

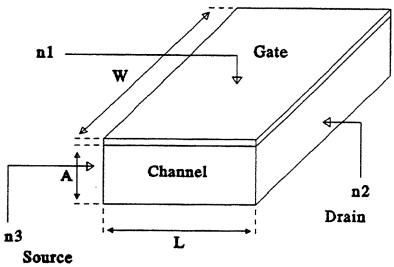
References

[1] M.A. Khatibzadeh and R.J. Trew, "A large-signal, analytic model for the GaAs MESFET", *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 231-238.

[2] J.W. Bandler, Q.J. Zhang and Q. Cai, "Nonlinear circuit optimization with dynamically integrated physical device models", *IEEE MTT-S Int. Microwave Symp. Dig.* (Dallas, TX), 1990, pp. 303-306.

[3] M.A. Khatibzadeh, "Large-signal modeling of gallium-arsenide field effect transistors", Ph.D. dissertation, North Carolina State University, Raleigh, NC, 1987.

FETT1 modified physics-based FET model FETT1



Keywords

		Defaults
L	gate length in m	0.56E-6
W	gate width in m	1.2E-3
A	channel thickness in m	0.35E-6
U0	low-field mobility in $m^2/(Vns)$	4.0E-10
D	high-field diffusion coefficient in m^2/ns	1.0E-12
EPSR	relative dielectric constant	12.5
VS	saturation electron velocity in m/ns	1.5E-4
VBI	built-in potential in V	0.7
ND	doping density (for uniform doping) in m^{-3}	7.5E22
LAMBDA	parameter for the transition function in m	1.0E-8
ALPHA	exponent for calculating the average mobility	0.5
TAU	time delay in ns	0.0
AA	parameter for electron velocity curve in V/m	1.0E6
BB	parameter for electron velocity curve	1
CC	parameter for electron velocity curve	1
DD	parameter for electron velocity curve	0

Form

```
FETT  n1 n2 n3  L = x1     W = x2       A = x3       U0 = x4
                D = x5     EPSR = x6    VS = x7      VBI = x8
                ND = x9    LAMBDA = x10 ALPHA = x11  TAU = x12
                AA = x13   BB = x14     CC = x15     DD = x16;
```

FETT1 modified physics-based FET model FETT1

Example

```
FETT1  1 2 3  L=0.42UM  W=600UM  A=0.3UM  VBI=0.7  ND=1.0E23
              TAU=3PS;
```

Model Equations

FETT1 implements the physics-based analytical model by Khatibzadeh and Trew [1] and OSA's modifications to improve the model efficiency for uniform doping profiles. Under suitable conditions (uniform doping), FETT1 can be evaluated much faster than FETT. However, it cannot be used for nonuniform doping profiles (use FETT instead).

The model equations are dynamically integrated into the harmonic balance simulation and optimization as described by Bandler, Zhang and Cai [2].

Doping Density

FETT1 is suitable for uniform doping profile only. The doping density is specified by the model parameter ND.

Electron Velocity - Electric Field Curve

The electron velocity as a function of the electric field is given by

$$v(E) = (1 - DD)\cdot v_1(E) + DD\cdot v_2(E)$$

where E is the electric field, and DD is a model parameter.

$v_1(E)$ is the formula originally used by Khatibzadeh and Trew [1]:

$$v_1(E) = \begin{cases} VS & \text{if } E > 2\cdot EC \\ U0\cdot E - VS\cdot E^2 / (4\cdot EC^2) & \text{if } E \le 2\cdot EC \end{cases}$$

where VS, U0 are model parameters and $EC = VS / U0$ is the critical electric field.

The function $v_2(E)$ is added to model the overshooting behaviour as experimentally observed:

$$v_2(E) = VS\cdot(BB/CC)\cdot\{(E/AA - CC)/[(E/AA)^4 + BB] + CC/BB\}$$

where VS, AA, BB and CC are model parameters. AA can be used to adjust the location of the overshoot. BB and CC can be used to adjust the sharpness of the curve peak. A larger value of the ratio BB/CC will lead to a sharper peak.

The parameter DD serves as a weighting factor for combining $v_1(E)$ and $v_2(E)$. Its value should be between 0 and 1.

Transition Function

The transition function models the transition of the doping profile from almost zero near the gate ($y \approx 0$) to the background doping concentration in the channel ($y \approx A$). For uniform doping, FETT1 approximates the transition function by [3]

$$T(d(x),y) = \begin{cases} 0.5\cdot(1 + \sin(\pi(y-d(x))/(6\cdot LAMBDA))) \\ \qquad \text{if} \quad d(x) - 3\cdot LAMBDA < y < d(x) + 3\cdot LAMBDA \\ 0 \qquad \text{if} \quad y < d(x) - 3\cdot LAMBDA \\ 1 \qquad \text{if} \quad y > d(x) + 3\cdot LAMBDA \end{cases}$$

where d(x) is the effective depletion-layer width calculated internally by the model, and LAMBDA is a model parameter whose value must be of the order of the Debye length (for example, LAMBDA = 0.01UM).

Using this formula, the integrations involved in the calculation of currents and charges at the gate, drain and source electrodes are substantially simplified.

ALPHA

ALPHA is an empirical model parameter used to calculate the weighted average mobility for the evaluation of the conduction currents [4]. Setting ALPHA = 0.5 is experimentally found to give good results.

References

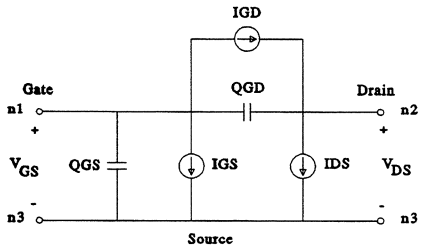
[1] M.A. Khatibzadeh and R.J. Trew, "A large-signal, analytic model for the GaAs MESFET", *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 231-238.

[2] J.W. Bandler, Q.J. Zhang and Q. Cai, "Nonlinear circuit optimization with dynamically integrated physical device models", *IEEE MTT-S Int. Microwave Symp. Dig.* (Dallas, TX), 1990, pp. 303-306.

[3] J.W. Bandler, R.M. Biernacki, Q. Cai, S.H. Chen, S. Ye and Q.J. Zhang, "Integrated physics-oriented statistical modeling, simulation and optimization," *IEEE Trans. Microwave Theory Tech.* vol. 40, 1992, pp. 1374-1400.

[4] M.A. Khatibzadeh, "Large-signal modeling of gallium-arsenide field effect transistors", Ph.D. dissertation, North Carolina State University, Raleigh, NC, 1987.

FETU1 user-definable FET model 1 FETU1



Keywords

		Defaults
IGS	gate-source current	0
IDS	drain-source current	0
IGD	gate-drain current	0
QGS	gate-source capacitor charge	0
QGD	gate-drain capacitor charge	0
TAU	transit time under gate	0

Form

```
FETU1  n1 n2 [n3]  IGS = expression1  IDS = expression2
                   IGD = expression3  QGS = expression4
                   QGD = expression5  TAU = x1;
```

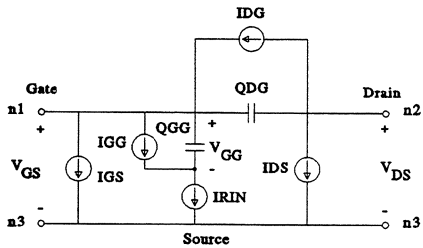
To utilize FETU1, you can create a set of model parameters of arbitrary meaning and names, and then define the nonlinear currents and charges as functions (expression1, ..., expression5) of these model parameters and the time-domain intrinsic voltages which are represented by the predefined labels VGS_T, VGS_TAU and VDS_T.

VGS_T represents the instantaneous gate-source voltage.
VGS_TAU represents the delayed gate-source voltage. The time delay is specified by the model parameter TAU.
VDS_T represents the instantaneous drain-source voltage.

IGS, IGD, QGS and QGD can be functions of VGS_T and VDS_T.
IDS can be a function of VGS_T, VGS_TAU and VDS_T.

The nonlinear charge sources are represented in the schematic diagram by capacitors to emphasize their capacitive nature. The charge source QGD is directed from the gate toward the drain. The charge source QGS is directed from the gate toward the source.

FETU2 user-definable FET model 2 FETU2



Keywords

		Defaults
IGS	gate-source current	0
IDS	drain-source current	0
IDG	drain-gate current	0
IGG	current between gate and node n4	0
IRIN	current between node n4 and source	0
QGG	gate capacitor charge	0
QDG	drain-gate capacitor charge	0
TAU	transit time under gate	0

Form

```
FETU2  n1 n2 [n3]  IGS = expression1  IDS = expression2   IDG = expression3
                   IGG = expression4  IRIN = expression5  QGG = expression6
                   QDG = expression7  TAU = x1;
```

To utilize FETU2, you can create a set of model parameters of arbitrary meaning and names, and then define the nonlinear currents and charges as functions (expression1, ..., expression7) of these model parameters and the time-domain intrinsic voltages which are represented by the predefined labels VGS_T, VGS_TAU, VGG_T and VDS_T.

VGS_T represents the instantaneous gate-source voltage.
VGS_TAU represents the delayed gate-source voltage. The time delay is specified by the model parameter TAU.
VGG_T represents the instantaneous voltage V_{GG} as shown in the schematic diagram.
VDS_T represents the instantaneous drain-source voltage.

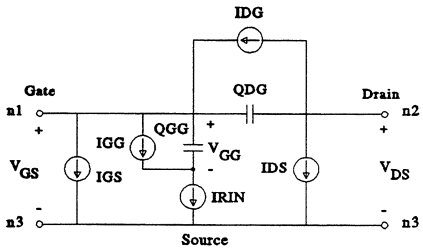
IGS, IDG, IGG, IRIN, QGG and QDG can be functions of VGS_T, VGG_T and VDS_T.
IDS can be a function of VGS_T, VGS_TAU, VGG_T and VDS_T.

FETU2 user-definable FET model 2 FETU2

The nonlinear charge sources are represented in the schematic diagram by capacitors to emphasize their capacitive nature. The charge source QDG is directed from the drain toward the gate. The charge source QGG is directed from the gate toward the source.

The only difference between FETU2 and FETU3 is which intrinsic voltage has an associated time delay. For FETU2, the voltage with time delay is V_{GS}. For FETU3, the voltage with time delay is V_{GG}. In both cases, the delayed voltage is represented by the predefined label VGS_TAU.

FETU3 user-definable FET model 3 FETU3



Keywords

		Defaults
IGS	gate-source current	0
IDS	drain-source current	0
IDG	drain-gate current	0
IGG	current between gate and node n4	0
IRIN	current between node n4 and source	0
QGG	gate capacitor charge	0
QDG	drain-gate capacitor charge	0
TAU	transit time under gate	0

Form

```
FETU3  n1 n2 [n3]  IGS = expression1  IDS = expression2   IDG = expression3
                   IGG = expression4  IRIN = expression5  QGG = expression6
                   QDG = expression7  TAU = x1;
```

To utilize FETU3, you can create a set of model parameters of arbitrary meaning and names, and then define the nonlinear currents and charges as functions (expression1, ..., expression7) of these model parameters and the time-domain intrinsic voltages which are represented by the predefined labels VGS_T, VGG_T, VGS_TAU and VDS_T.

VGS_T represents the instantaneous gate-source voltage.
VGG_T represents the instantaneous voltage V_{GG} as shown in the schematic diagram.
VGS_TAU represents the voltage V_{GG} delayed by TAU, where TAU is a model parameter.
VDS_T represents the instantaneous drain-source voltage.

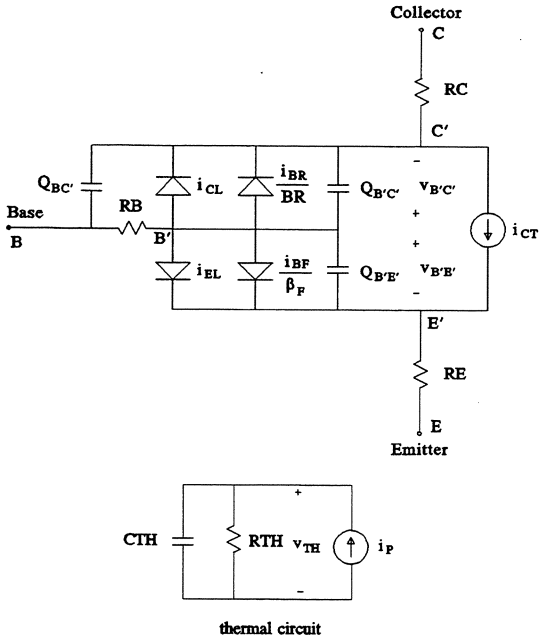
IGS, IDG, IGG, IRIN, QGG and QDG can be functions of VGS_T, VGG_T and VDS_T.
IDS can be a function of VGS_T, VGG_T, VGS_TAU and VDS_T.

FETU3 user-definable FET model 3 FETU3

The nonlinear charge sources are represented in the schematic diagram by capacitors to emphasize their capacitive nature. The charge source QDG is directed from the drain toward the gate. The charge source QGG is directed from the gate toward the source.

The only difference between FETU2 and FETU3 is which intrinsic voltage has an associated time delay. For FETU2, the voltage with time delay is V_{GS}. For FETU3, the voltage with time delay is V_{GG}. In both cases, the delayed voltage is represented by the predefined label VGS_TAU.

HBT heterojunction bipolar transistor model HBT



Keywords

		Defaults
CJE	base-emitter zero-bias depletion capacitance	0
VJE	base-emitter junction built-in potential	0.75
MJE	base-emitter junction exponential factor	0.33
TF	ideal forward-transit time	0
CJC	base-collector zero-bias depletion capacitance	0
VJC	base-collector junction built-in potential	0.75
MJC	base-collector junction exponential factor	0.33
XCJC	fraction of CJC that goes to internal base node	1.0
TR	ideal reverse-transit time	0
FC	coefficient for forward-bias depletion capacitance formula	0.5
IS0	transport saturation current	1.0E-23
BF0	ideal maximum forward beta	100
NF	forward-current emission coefficient	1.0
VAF	forward early voltage	∞

HBT heterojunction bipolar transistor model HBT

Keywords

		Defaults
IKF	corner for forward-beta high current roll-off	∞
ISE0	base-emitter leakage saturation current	0
NE	base-emitter leakage emission coefficient	1.5
BR	ideal maximum reverse beta	1.0
NR	reverse-current emission coefficient	1.0
VAR	reverse early voltage	∞
IKR	corner for reverse-beta high-current roll-off	∞
ISC0	base-collector leakage saturation current	0
NC	base-collector leakage emission coefficient	2.0
EG	parameter for calculating the thermal effect	0
EINF	parameter for calculating the thermal effect	0
P	exponential parameter for calculating the thermal effect	0
XTI	exponential parameter for calculating the thermal effect	0
RTH	thermal resistance	1.0
CTH	thermal capacitance	0
RB	base resistance	10
RC	collector resistance	10
RE	emitter resistance	10

Form

```
HBT  n1 n2 n3  CJE = x1    VJE = x2    MJE = x3    TF = x4    CJC = x5
               VJC = x6    MJC = x7    XCJC = x8   TR = x9    FC = x10
               ISO = x11   BF0 = x12   NF = x13    VAF = x14  IKF = x15
               ISE0 = x16  NE = x17    BR = x18    NR = x19   VAR = x20
               IKR = x21   ISC0 = x22  NC = x23    RB = x24   RC = x25
               RE = x26    EG = x27    EINF = x28   P = x29    XTI = x30
               RTH = x31   CTH = x32;
```

Example

```
HBT  1 2 3  VJE=0.75   MJE=0.33   VJC=0.75       MJC=0.33
            XCJC=1.0   FC=0.5     ISO=8.216E-16  BF0=408.1
            NF=0.983   VAF=105.8  IKF=7.581E-3   ISE0=8.233E-16
            NE=1.283   BR=1.0     NR=1.0         VAR=1000.0
            IKR=1.0E3  NC=2.0     RB=10.3        P=-1
            XTI=6.5    RTH=180    CTH=1.0E-6;
```

HBT heterojunction bipolar transistor model HBT

Model Description

HBT is a modification of the BJT model by introducing the thermal effect following the thermal-electrical model of HBT [1]. A thermal subcircuit is used to evaluate the device temperature which is integrated into the DC, small-signal and large-signal harmonic balance simulation.

The temperature is calculated by

$$T = T_0 + v_{TH}$$

where T_0 is the room temperature. The thermal effect on the device performance is characterized by the following formulas.

$$I_S = ISO \cdot (T/T_0)^{XTI} \cdot \exp[EG/(kT_0) - EG/(kT)]$$

$$\beta_F = BF0 \cdot (T/T_0)^{P} \cdot \exp[EINF/(kT) - EINF/(kT_0)]$$

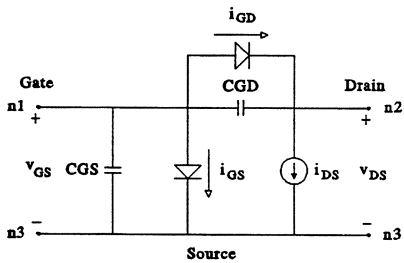
where ISO, XTI, EG, BF0, P and EINF are model parameters. k is the Boltzmann constant. The saturation current I_S and forward beta β_F are required for evaluating the nonlinear currents passing through the diodes as shown in the equivalent circuit for HBT.

Reference

[1] H. Wang, C. Algani, A. Konczykowska and W. Zuberek, "Temperature dependence of DC currents in HBT," *IEEE MTT-S Int. Microwave Symp. Digest* (Albuquerque, NM), 1992, pp. 731-734.

[2] P.C. Grossman and J. Choma, "Large signal modeling of HBT's including self-heating and transit time effects," *IEEE Trans. Microwave Theory Tech.*, vol. 40, 1992, pp. 449-464.

HEMTAC advanced Curtice HEMT model HEMTAC



Keywords

		Defaults
ALPHA	hyperbolic tangent function parameter	0
BETA	transconductance parameter	0
LAMBDA	channel length modulation parameter	0
VT0	threshold voltage	-1.0
PSI	empirical transconductance degradation parameter (exponential)	0
XI	empirical transconductance degradation parameter (linear)	0
VPF	gate voltage where transconductance begins to degrade	0
GAMMA	voltage-slope parameter of pinch-off voltage	0
UCRIT	parameter representing the critical field for mobility degradation	0
VGEXP	gate voltage exponential parameter	2
VDS0	drain voltage where current saturation occurs with zero gate bias	1
CGD0	gate-drain fixed fringing capacitance	0
CGS0	gate-source fixed fringing capacitance	0
CM0	empirical total gate electrode capacitance parameter	0
CHI	empirical gate electrode coefficient parameter	1.0
TAU	transit time under gate	0
IS	gate junction saturation current	1.0E-14
N	gate-drain and gate-source emission coefficient	1.0
GMIN	linear conductance associated with the Schottky junctions	1.0E-12
VBR	gate reverse bias breakdown voltage	+∞
TEMP	temperature	298°K

Form

```
HEMTAC  n1 n2 n3  ALPHA = x1   BETA = x2    LAMBDA = x3  VT0 = x4
                  PSI = x5     XI = x6      VPF = x7     GAMMA= x8
                  UCRIT = x9   VGEXP = x10  VDS0 = x11   CGD0 = x12
                  CGS0 = x13   CM0 = x14    CHI = x15    TAU = x16
                  IS = x17     N = x18      GMIN = x19   VBR = x20
                  TEMP = x21;
```

HEMTAC advanced Curtice HEMT model HEMTAC

Example

```
HEMTAC  1 2 3  ALPHA=0.5   BETA=0.02   VTO=-1.2  PSI=2  VPF=-1
               CGDO=0.1PF  CGSO=0.2PF;
```

Model Equations

HEMTAC implements the advanced Curtice HEMT model [1].

All components of the equivalent circuit are assumed to be functions of the gate and drain voltages v_{GS} and v_{DS} as shown in the diagram. The nonlinear model equations are as follows.

$$i_{DS} = \begin{cases} i_{DSFET} & \text{for } v_{GS}(t-TAU) \le VPF \\ i_{DSFET} - XI \cdot \beta_{eff} \cdot (v_{GS}(t-TAU) - VPF)^{PSI} \cdot f(v_{DS}) & \text{for } v_{GS}(t-TAU) > VPF \end{cases}$$

where

$$i_{DSFET} = \begin{cases} 0 & \text{for } v_{GS}(t-TAU) \le V_T \\ \beta_{eff} \cdot (v_{GS}(t-TAU) - V_T)^{VGEXP} \cdot f(v_{DS}) & \text{for } v_{GS}(t-TAU) > V_T \end{cases}$$

$$f(v_{DS}) = \tanh(ALPHA \cdot v_{DS}) \cdot (1+LAMBDA \cdot v_{DS})$$

$$\beta_{eff} = BETA \;/\; [1 + UCRIT \cdot (v_{GS}(t-TAU) - V_T)]$$

$$V_T = VTO + GAMMA \cdot v_{DS}$$

and ALPHA, BETA, LAMBDA, XI, PSI, VPF, VTO, GAMMA, UCRIT, VGEXP and TAU are model parameters.

$$C_{GS} = \begin{cases} CGSO + 2 \cdot C_G \cdot [1 - (V_{DSS}-v_{DS})^2/(2 \cdot V_{DSS}-v_{DS})^2]/3 & \text{for } v_{DS} < V_{DSS} \\ 2 \cdot C_G/3 + CGSO & \text{for } v_{DS} \ge V_{DSS} \end{cases}$$

$$C_{GD} = \begin{cases} CGDO + 2 \cdot C_G \cdot [1 - V_{DSS}^2/(2 \cdot V_{DSS}-v_{DS})^2]/3 & \text{for } v_{DS} < V_{DSS} \\ CGDO & \text{for } v_{DS} \ge V_{DSS} \end{cases}$$

where

$$C_G = \begin{cases} 0 & \text{for } v_{GS} \le VTO \\ CMO \cdot (v_{GS}-VTO)^{1/CHI} & \text{for } v_{GS} > VTO \end{cases}$$

HEMTAC advanced Curtice HEMT model HEMTAC

$$V_{DSS} = \begin{cases} 0 & \text{for } v_{GS} \le VT0 \\ VDS0 \cdot (1 - v_{GS}/VT0) & \text{for } v_{GS} > VT0 \end{cases}$$

and CGS0, CGD0, CM0, VDS0, CHI and VT0 are model parameters.

$$i_{GS} = \begin{cases} IS \cdot [\exp(v_{GS}/(N \cdot v_t)) - 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \ge -5N \cdot v_t \\ -IS + GMIN \cdot v_{GS} & \text{for } -VBR+50v_t < v_{GS} < -5N \cdot v_t \\ -IS \cdot [\exp(-(v_{GS}+VBR)/v_t) + 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \le -VBR+50v_t \end{cases}$$

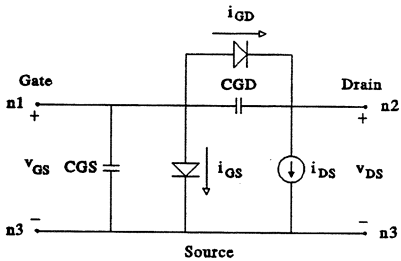
where $v_t = k \cdot TEMP/q$ is the thermal voltage. IS, N, GMIN, VBR and TEMP are model parameters.

The diode current i_{GD} is similarly defined.

Reference

[1] J.M. Golio, *Microwave MESFETs and HEMTs*, Norwood, MA: Artech House, 1991.

HEMTC — Curtice HEMT model — HEMTC



Keywords

		Defaults
ALPHA	hyperbolic tangent function parameter	0
BETA	transconductance parameter	0
LAMBDA	channel length modulation parameter	0
VTO	threshold voltage	-1.0
PSI	empirical transconductance degradation parameter (exponential)	0
XI	empirical transconductance degradation parameter (linear)	0
VPF	gate voltage where transconductance begins to degrade	0
VDSO	drain voltage where current saturation occurs with zero gate bias	1
CGDO	gate-drain fixed fringing capacitance	0
CGSO	gate-source fixed fringing capacitance	0
CMO	empirical total gate electrode capacitance parameter	0
CHI	empirical gate electrode coefficient parameter	1.0
TAU	transit time under gate	0
IS	gate junction saturation current	1.0E-14
N	gate-drain and gate-source emission coefficient	1.0
GMIN	linear conductance associated with the Schottky junctions	1.0E-12
VBR	gate reverse bias breakdown voltage	+∞
TEMP	temperature	298°K

Form

```
HEMTC  n1 n2 n3  ALPHA = x1  BETA = x2   LAMBDA = x3  VTO = x4
                 PSI = x5    XI = x6     VPF = x7     VDSO = x8
                 CGDO = x9   CGSO = x10  CMO = x11    CHI = x12
                 TAU = x13   IS = x14    N = x15      GMIN = x16
                 VBR = x17   TEMP = x18;
```

HEMTC — Curtice HEMT model — HEMTC

Example

```
HEMTC  1 2 3  BETA=0.02  VTO=-1.2  PSI=2;
```

Model Equations

HEMTC implements the Curtice HEMT model [1].

All components of the equivalent circuit are assumed to be functions of the gate and drain voltages v_{GS} and v_{DS} as shown in the diagram. The nonlinear model equations are as follows.

$$i_{DS} = \begin{cases} i_{DSFET} & \text{for } v_{GS}(t-TAU) \le VPF \\ i_{DSFET} - XI \cdot (v_{GS}(t-TAU) - VPF)^{PSI+1} \cdot f(v_{DS})/(PSI+1) & \text{for } v_{GS}(t-TAU) > VPF \end{cases}$$

where

$$i_{DSFET} = \begin{cases} 0 & \text{for } v_{GS}(t-TAU) \le VTO \\ BETA \cdot (v_{GS}(t-TAU) - VTO)^2 \cdot f(v_{DS}) & \text{for } v_{GS}(t-TAU) > VTO \end{cases}$$

$$f(v_{DS}) = \tanh(ALPHA \cdot v_{DS}) \cdot (1+LAMBDA \cdot v_{DS})$$

and ALPHA, BETA, LAMBDA, XI, PSI, VPF, VTO and TAU are model parameters.

$$C_{GS} = \begin{cases} CGSO + 2 \cdot C_G \cdot [1 - (V_{DSS}-v_{DS})^2/(2 \cdot V_{DSS}-v_{DS})^2]/3 & \text{for } v_{DS} < V_{DSS} \\ 2 \cdot C_G/3 + CGSO & \text{for } v_{DS} \ge V_{DSS} \end{cases}$$

$$C_{GD} = \begin{cases} CGDO + 2 \cdot C_G \cdot [1 - V_{DSS}^2/(2 \cdot V_{DSS}-v_{DS})^2]/3 & \text{for } v_{DS} < V_{DSS} \\ CGDO & \text{for } v_{DS} \ge V_{DSS} \end{cases}$$

where

$$C_G = \begin{cases} 0 & \text{for } v_{GS} \le VTO \\ CMO \cdot (v_{GS}-VTO)^{1/CHI} & \text{for } v_{GS} > VTO \end{cases}$$

$$V_{DSS} = \begin{cases} 0 & \text{for } v_{GS} \le VTO \\ VDSO \cdot (1 - v_{GS}/VTO) & \text{for } v_{GS} > VTO \end{cases}$$

and CGSO, CGDO, CMO, VDSO, CHI and VTO are model parameters.

HEMTC — Curtice HEMT model — HEMTC

$$i_{GS} = \begin{cases} IS \cdot [\exp(v_{GS}/(N \cdot v_t)) - 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \geq -5N \cdot v_t \\ -IS + GMIN \cdot v_{GS} & \text{for } -VBR+50v_t < v_{GS} < -5N \cdot v_t \\ -IS \cdot [\exp(-(v_{GS}+VBR)/v_t) + 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \leq -VBR+50v_t \end{cases}$$

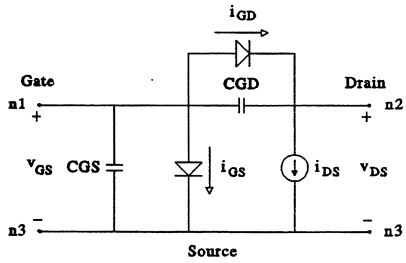
where $v_t = k \cdot TEMP/q$ is the thermal voltage. IS, N, GMIN, VBR and TEMP are model parameters.

The diode current i_{GD} is similarly defined.

Reference

[1] J.M. Golio, *Microwave MESFETs and HEMTs*, Norwood, MA: Artech House, 1991.

HEMTG1 high order beta degradation HEMT model HEMTG1



Keywords

		Defaults
ALPHA	hyperbolic tangent function parameter	0
BETA	transconductance parameter	0
LAMBDA	channel length modulation parameter	0
VT0	threshold voltage	-1.0
GAMMA	voltage-slope parameter of pinch-off voltage	0
UCRIT	parameter representing the critical field for mobility degradation	0
VGEXP	gate voltage exponential parameter	2
GMEXP	beta degradation exponential parameter	1
VDS0	drain voltage where current saturation occurs with zero gate bias	1
CGD0	gate-drain fixed fringing capacitance	0
CGS0	gate-source fixed fringing capacitance	0
CM0	empirical total gate electrode capacitance parameter	0
CHI	empirical gate electrode coefficient parameter	1.0
TAU	transit time under gate	0
IS	gate junction saturation current	1.0E-14
N	gate-drain and gate-source emission coefficient	1.0
GMIN	linear conductance associated with the Schottky junctions	1.0E-12
VBR	gate reverse bias breakdown voltage	+∞
TEMP	temperature	298°K

Form

```
HEMTG1  n1 n2 n3  ALPHA = x1  BETA = x2   LAMBDA = x3  VT0 = x4
                  GAMMA= x5   UCRIT = x6  VGEXP = x7   GMEXP = x8
                  VDS0 = x9   CGD0 = x10  CGS0 = x11   CM0 = x12
                  CHI = x13   TAU = x14   IS = x15     N = x16
                  GMIN = x17  VBR = x18   TEMP = x19;
```

HEMTG1 high order beta degradation HEMT model HEMTG1

Example

```
HEMTG1  1 2 3  ALPHA=0.5   BETA=0.02   VT0=-1.2  GMEXP=2  CGD0=0.1PF
               CGS0=0.2PF;
```

Model Equations

HEMTG1 implements the high order beta degradation HEMT model [1].

All components of the equivalent circuit are assumed to be functions of the gate and drain voltages v_{GS} and v_{DS} as shown in the diagram. The nonlinear model equations are as follows.

$$i_{DS} = \begin{cases} 0 & \text{for } v_{GS}(t\text{-}TAU) \le V_T \\ \beta_{eff} \cdot (v_{GS}(t\text{-}TAU) - V_T)^{VGEXP} \cdot f(v_{DS}) & \text{for } v_{GS}(t\text{-}TAU) > V_T \end{cases}$$

where

$$f(v_{DS}) = \tanh(ALPHA \cdot v_{DS}) \cdot (1+LAMBDA \cdot v_{DS})$$

$$\beta_{eff} = BETA \;/\; [1 + UCRIT \cdot (v_{GS}(t\text{-}TAU) - V_T)^{GMEXP}]$$

$$V_T = VT0 + GAMMA \cdot v_{DS}$$

and ALPHA, BETA, LAMBDA, VT0, GAMMA, UCRIT, VGEXP, GMEXP and TAU are model parameters.

$$C_{GS} = \begin{cases} CGS0 + 2 \cdot C_G \cdot [1 - (V_{DSS}-v_{DS})^2/(2 \cdot V_{DSS}-v_{DS})^2]/3 & \text{for } v_{DS} < V_{DSS} \\ 2 \cdot C_G/3 + CGS0 & \text{for } v_{DS} \ge V_{DSS} \end{cases}$$

$$C_{GD} = \begin{cases} CGD0 + 2 \cdot C_G \cdot [1 - V_{DSS}^2/(2 \cdot V_{DSS}-v_{DS})^2]/3 & \text{for } v_{DS} < V_{DSS} \\ CGD0 & \text{for } v_{DS} \ge V_{DSS} \end{cases}$$

where

$$C_G = \begin{cases} 0 & \text{for } v_{GS} \le VT0 \\ CM0 \cdot (v_{GS}-VT0)^{1/CHI} & \text{for } v_{GS} > VT0 \end{cases}$$

$$V_{DSS} = \begin{cases} 0 & \text{for } v_{GS} \le VT0 \\ VDS0 \cdot (1 - v_{GS}/VT0) & \text{for } v_{GS} > VT0 \end{cases}$$

and CGS0, CGD0, CM0, VDS0, CHI and VT0 are model parameters.

HEMTG1 high order beta degradation HEMT model HEMTG1

$$i_{GS} = \begin{cases} IS \cdot [\exp(v_{GS}/(N \cdot v_t)) - 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \geq -5N \cdot v_t \\ -IS + GMIN \cdot v_{GS} & \text{for } -VBR+50v_t < v_{GS} < -5N \cdot v_t \\ -IS \cdot [\exp(-(v_{GS}+VBR)/v_t) + 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \leq -VBR+50v_t \end{cases}$$

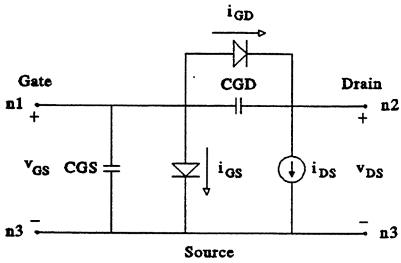
where $v_t = k \cdot TEMP/q$ is the thermal voltage. IS, N, GMIN, VBR and TEMP are model parameters.

The diode current i_{GD} is similarly defined.

Reference

[1] D. Halchin, M. Miller, M. Golio and S. Tehrani, "HEMT models for large signal circuit simulation," *IEEE MTT-S Int. Microwave Symp. Digest* (San Diego, CA), 1994, pp. 985-988.

HEMTG2 double parabolic HEMT model HEMTG2



Keywords

		Defaults
ALPHA	hyperbolic tangent function parameter	0
BETA	transconductance parameter	0
LAMBDA	channel length modulation parameter	0
VT0	threshold voltage	-1.0
GAMMA	voltage-slope parameter of pinch-off voltage	0
VC	gate voltage where the peak of transconductance occurs	0
VDS0	drain voltage where current saturation occurs with zero gate bias	1
CGD0	gate-drain fixed fringing capacitance	0
CGS0	gate-source fixed fringing capacitance	0
CM0	empirical total gate electrode capacitance parameter	0
CHI	empirical gate electrode coefficient parameter	1.0
TAU	transit time under gate	0
IS	gate junction saturation current	1.0E-14
N	gate-drain and gate-source emission coefficient	1.0
GMIN	linear conductance associated with the Schottky junctions	1.0E-12
VBR	gate reverse bias breakdown voltage	+∞
TEMP	temperature	298°K

Form

```
HEMTG2  n1 n2 n3  ALPHA = x1  BETA = x2  LAMBDA = x3  VT0 = x4
                  GAMMA= x5   VC = x6    VDS0 = x7    CGD0 = x8
                  CGS0 = x9   CM0 = x10  CHI = x11    TAU = x12
                  IS = x13    N = x14    GMIN = x15   VBR = x16
                  TEMP = x17;
```

HEMTG2 double parabolic HEMT model HEMTG2

Example

```
HEMTG2  1 2 3  ALPHA=0.5   BETA=0.02   VT0=-1.2  VDS0=-1  CGD0=0.1PF
               CGS0=0.2PF;
```

Model Equations

HEMTG2 implements the double parabolic HEMT model [1].

All components of the equivalent circuit are assumed to be functions of the gate and drain voltages v_{GS} and v_{DS} as shown in the diagram. The nonlinear model equations are as follows.

$$i_{DS} = \begin{cases} 0 & \text{for } v_{GS}(t-TAU) < V_T \\ BETA \cdot (v_{GS}(t-TAU) - GAMMA \cdot v_{DS} - VT0)^3 \cdot f(v_{DS})/3 & \text{for } V_T \le v_{GS}(t-TAU) < V_{PF} \\ BETA \cdot [(VC-VT0)^2 \cdot (v_{GS}(t-TAU) - V_{PF})/2 - (v_{GS}(t-TAU) - GAMMA \cdot v_{DS} - VC)^3/3] \cdot f(v_{DS}) & \text{for } v_{GS}(t-TAU) \ge V_{PF} \end{cases}$$

where

$$f(v_{DS}) = \tanh(ALPHA \cdot v_{DS}) \cdot (1 + LAMBDA \cdot v_{DS})$$

$$V_{PF} = (VT0+VC)/2 + GAMMA \cdot v_{DS}$$

$$V_T = VT0 + GAMMA \cdot v_{DS}$$

and ALPHA, BETA, LAMBDA, VT0, GAMMA, VC and TAU are model parameters.

$$C_{GS} = \begin{cases} CGS0 + 2 \cdot C_G \cdot [1 - (V_{DSS} - v_{DS})^2/(2 \cdot V_{DSS} - v_{DS})^2]/3 & \text{for } v_{DS} < V_{DSS} \\ 2 \cdot C_G/3 + CGS0 & \text{for } v_{DS} \ge V_{DSS} \end{cases}$$

$$C_{GD} = \begin{cases} CGD0 + 2 \cdot C_G \cdot [1 - V_{DSS}^2/(2 \cdot V_{DSS} - v_{DS})^2]/3 & \text{for } v_{DS} < V_{DSS} \\ CGD0 & \text{for } v_{DS} \ge V_{DSS} \end{cases}$$

where

$$C_G = \begin{cases} 0 & \text{for } v_{GS} \le VT0 \\ CM0 \cdot (v_{GS} - VT0)^{1/CHI} & \text{for } v_{GS} > VT0 \end{cases}$$

HEMTG2 double parabolic HEMT model HEMTG2

$$V_{DSS} = \begin{cases} 0 & \text{for } v_{GS} \le VT0 \\ VDS0 \cdot (1 - v_{GS}/VT0) & \text{for } v_{GS} > VT0 \end{cases}$$

and CGS0, CGD0, CM0, VDS0, CHI and VT0 are model parameters.

$$i_{GS} = \begin{cases} IS \cdot [\exp(v_{GS}/(N \cdot v_t)) - 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \ge -5N \cdot v_t \\ -IS + GMIN \cdot v_{GS} & \text{for } -VBR+50v_t < v_{GS} < -5N \cdot v_t \\ -IS \cdot [\exp(-(v_{GS}+VBR)/v_t) + 1] + GMIN \cdot v_{GS} & \text{for } v_{GS} \le -VBR+50v_t \end{cases}$$

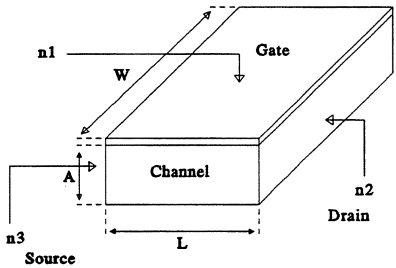
where $v_t = k$·TEMP$/q$ is the thermal voltage. IS, N, GMIN, VBR and TEMP are model parameters.

The diode current i_{GD} is similarly defined.

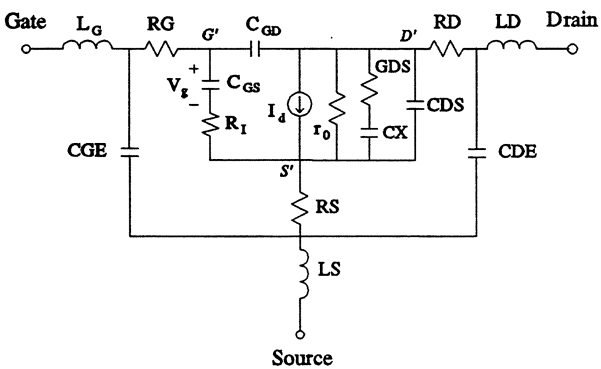
Reference

[1] D. Halchin, M. Miller, M. Golio and S. Tehrani, "HEMT models for large signal circuit simulation," *IEEE MTT-S Int. Microwave Symp. Digest* (San Diego, CA), 1994, pp. 985-988.

KTL physics-based bias-dependent small-signal FET model KTL



The device schematic for the KTL model.



The small-signal equivalent circuit for the KTL model.

Notes

- KTL combines the Khatibzadeh and Trew physics-based nonlinear model (FETT1, for solving for the DC operating point) and the Ladbrooke small-signal equivalent circuit.
- KTL assumes uniform doping.
- KTL can only be used for small-signal simulation.

KTL physics-based bias-dependent small-signal FET model KTL

Keywords

		Defaults
L	gate length in m	0.56E-6
W	gate width in m	1.2E-3
A	channel thickness in m	0.35E-6
EPSR	relative dielectric constant	12.5
VS	saturation electron velocity in m/ns	1.5E-4
VBI	built-in potential in V	0.7
U0	low-field mobility in $m^2/(Vns)$	4.0E-10
D	high-field diffusion coefficient in m^2/ns	1.0E-12
ND	doping density (for uniform doping) in m^{-3}	7.5E22
LAMBDA	parameter for the transition function in m	1.0E-8
ALPHA	exponent for calculating the average mobility	0.5
AA	parameter for electron velocity curve in V/m	1.0E6
BB	parameter for electron velocity curve	1
CC	parameter for electron velocity curve	1
DD	parameter for electron velocity curve	0
A0	proportional constant	1
R01	parameter for output resistance in Ohm/V^2	0.01
R02	parameter for output resistance in V	14
R03	parameter for output resistance in Ohm	600
M	number of gate fingers of the MESFET	4
LGG0	parameter for gate inductance in nH	0.02
CDS	drain-source capacitance	0
GDS	drain-source conductance	0
RG	gate resistance	0
RD	drain resistance	0
RS	source resistance	0
CGE	external gate capacitance	0
CDE	external drain capacitance	0
CX	drain-source capacitance	1.5pF
LD	drain lead inductance	0
LS	source lead inductance	0

KTL physics-based bias-dependent small-signal FET model KTL

Model Equations

The model equations for defining doping density, electron velocity and transition function are the same as those in FETT1.

In the small-signal equivalent circuit we have

$$I_d = g_m \cdot V_g \cdot e^{-j\omega\cdot\tau}.$$

The bias-dependent small-signal parameters are L_G, C_{GS}, C_{GD}, R_I, g_m, τ and r_0. They are derived using the modified Ladbrooke formulas once the DC operating point is solved for.

$$L_G = 4\times10^2 \cdot \pi \cdot d \cdot W \;/\; (M^2 \cdot L) + LGG0 \qquad (nH)$$

$$\varepsilon = 8.854\times10^{-12} \cdot EPSR$$

$$C_{GS} = 10^9 \cdot \varepsilon \cdot W \cdot L \cdot (1 + X/(2\cdot L) - 2\cdot d/(L + 2\cdot X)) \;/\; d \qquad (nF)$$

$$C_{GD} = 2\times10^9 \cdot \varepsilon \cdot W \;/\; (1 + 2\cdot X/L) \qquad (nF)$$

$$R_I = VS \cdot L \;/\; (U0 \cdot I_{CH}) \qquad (Ohm)$$

$$g_m = \varepsilon \cdot VS \cdot W \;/\; d \qquad (1/Ohm)$$

$$\tau = (X/2 - 2\cdot d/(1 + 2\cdot X/L)) \;/\; VS \qquad (nS)$$

$$r_0 = R01 \cdot V_{D'S'} \cdot (R02 - V_{G'S'}) + R03 \qquad (Ohm)$$

where the equivalent depletion depth, space-charge layer extension and channel current are

$$d = [2\cdot\varepsilon\cdot(VBI - V_{G'S'})/(q\cdot ND)]^{0.5} \qquad (m)$$

$$X = A0 \cdot \{2\cdot\varepsilon/[q\cdot ND(VBI - V_{G'S'})]\}^{0.5}(V_{D'S'} - V_{G'S'} + VBI) \qquad (m)$$

$$I_{CH} = q\cdot ND\cdot VS\cdot 10^9\cdot(A - d)\cdot W \qquad (A)$$

$V_{D'S'}$ and $V_{G'S'}$ are DC intrinsic voltages from D′ to S′ and from G′ to S′, respectively, as shown in the small-signal equivalent circuit diagram. Their values at the solution of the DC operating point can be obtained using HarPE's predefined labels VDS_DC and VGS_DC, respectively.

KTL physics-based bias-dependent small-signal FET model KTL

Example

```
! Example demo28.ckt
! illustrate KTL: nonlinear physics-based Ladbrooke-Trew model.

Expression
   L:    ?0.55UM?;
   W:    300UM;
   A:    0.13UM;
   EPSR: 12.5;
   VSAT: 9.5e-5;
   VBI:  ?0.2 0.6 1.2?;
   U0:   0.5e-9;
   ND_X: ?1.5 2.2 5?;
   ND:   ND_X * 1E23;
   A0:   1.0;
   R01:  ?0.078?;
   R02:  ?7.8?;
   R03:  ?550?;
   M:    4;
   LGG0: ?0.03?;

   EPSI = 8.854E-12 * EPSR;
   Q = 1.6E-19;

   D = SQRT(2 * EPSI * (-VGS_DC + VBI) / (Q * ND));
   X = A0 * SQRT(2 * EPSI / (Q * ND * (-VGS_DC + VBI))) * (VDS_DC - VGS_DC + VBI);
   ICH = Q * ND * VSAT * (A - D) * W * 1.0E9;

   LGG = 400 * PI * D * W / (M * M * L) + LGG0;

   CGS = 1.0E9 * EPSI * W * L / D * (1.0 + X / (L + L) - (D + D) / (L + X + X));
   CGD = 1.0E9 * (EPSI + EPSI) * W / (1.0 + (X + X) / L);
   RI = VSAT * L / (U0 * ICH);
   TAU = (X / 2.0 - (D + D) / (1.0 + (X + X) / L)) / VSAT;
End

Model
   Extrinsic2  1 2 3 4 5
      LG=LGG        RG=7.8      RD=0.5    LD=0.06NH    RS=3.8
      LS=0.022NH    GDS=0.0023  CX=2.0PF  CDS=0.060PF  CGE=0.055PF
      CDE=0.0002PF;

   SRC  1 3  R=RI  C=CGS;
   CAP  1 2  C=CGD;

   FETLT  1 2 3
      TAU=TAU  L=L    W=W      A=A      EPSR=EPSR  VS=VSAT  VBI=VBI
      U0=U0    A0=A0  R01=R01  R02=R02  R03=R03    ND=ND;

   2POR  4 5;
End
```

KTL physics-based bias-dependent small-signal FET model KTL

References

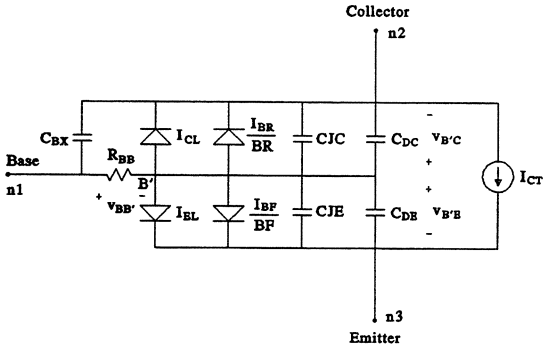
[1] M.A. Khatibzadeh and R.J. Trew, "A large-signal, analytic model for the GaAs MESFET", *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 231-238.

[2] P.H. Ladbrooke, *MMIC Design: GaAs FETs and HEMTs*. Norwood, MA: Artech House, 1989.

[3] J.W. Bandler, R.M. Biernacki, S.H. Chen, J. Song, S. Ye and Q.J. Zhang, "Statistical modeling of GaAs MESFETs", *IEEE MTT-S Int. Microwave Symp. Dig.* (Boston, MA), 1991, pp. 87-90.

[4] M.A. Khatibzadeh, "Large-signal modeling of gallium-arsenide field effect transistors", Ph.D. dissertation, North Carolina State University, Raleigh, NC, 1987.

NPN NPN Gummel and Poon bipolar transistor model NPN



Keywords

		Defaults
CJE	base-emitter zero-bias depletion capacitance	0
VJE	base-emitter junction built-in potential	0.75
MJE	base-emitter junction exponential factor	0.33
TF	ideal forward-transit time	0
CJC	base-collector zero-bias depletion capacitance	0
VJC	base-collector junction built-in potential	0.75
MJC	base-collector junction exponential factor	0.33
XCJC	fraction of CJC that goes to internal base node	1.0
TR	ideal reverse-transit time	0
FC	coefficient for forward-bias depletion capacitance formula	0.5
IS	transport saturation current	1.0E-16
BF	ideal maximum forward beta	100
NF	forward-current emission coefficient	1.0
VAF	Forward Early voltage	∞
IKF	corner for forward-beta high current roll-off	∞
ISE	base-emitter leakage saturation current	0
NE	base-emitter leakage emission coefficient	1.5
BR	ideal maximum reverse beta	1.0
NR	reverse-current emission coefficient	1.0
VAR	Reverse Early voltage	∞
IKR	corner for reverse-beta high-current roll-off	∞
ISC	base-collector leakage saturation current	0
NC	base-collector leakage emission coefficient	2.0
RB	zero-bias base resistance	10
RBM	minimum base resistance at high currents	10
IRB	current when base resistance falls half way to RBM	∞
TEMP	temperature	298°K

NPN NPN Gummel and Poon bipolar transistor model NPN

Form

```
NPN  n1 n2 n3  CJE = x1   VJE = x2   MJE = x3   TF = x4    CJC = x5
               VJC = x6   MJC = x7   XCJC = x8  TR = x9    FC = x10
               IS = x11   BF = x12   NF = x13   VAF = x14  IKF = x15
               ISE = x16  NE = x17   BR = x18   NR = x19   VAR = x20
               IKR = x21  ISC = x22  NC = x23   RB = x24   RBM = x25
               IRB = x26  TEMP = x27;
```

Example

```
NPN  1 2 3  VJE=0.75   MJE=0.33   VJC=0.75      MJC=0.33
            XCJC=1.0   FC=0.5     IS=8.216E-16  BF=408.1
            NF=0.983   VAF=105.8  IKF=7.581E-3  ISE=8.233E-16
            NE=1.283   BR=1.0     NR=1.0        VAR=1000.0
            IKR=1.0E3  NC=2.0     RB=199.3      RBM=52.05
            IRB=2.001E-3;
```

Model Description

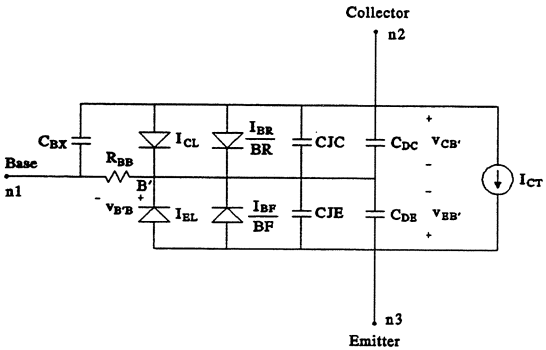
NPN is adapted from the integral charge control model of Gummel and Poon [1]. It becomes the simpler Ebers-Moll model when certain parameters required for the Gummel and Poon model are not specified.

Model parameters CJE, MJE and VJE determine the nonlinear depletion-layer capacitance for the base-emitter junction. CJC, MJC and VJC determine the nonlinear depletion-layer capacitance for the base-collector junction. TR and TF along with the depletion-layer capacitances model the base charge storage effects.

Reference

[1] I.E. Getreu, *CAD of Electronic Circuits 1: Modeling the Bipolar Transistor*. Amsterdam: Elsevier Scientific Publishing Company, 1978, pp. 69-123.

PNP PNP Gummel and Poon bipolar transistor model PNP



Keywords

		Defaults
CJE	base-emitter zero-bias depletion capacitance	0
VJE	base-emitter junction built-in potential	0.75
MJE	base-emitter junction exponential factor	0.33
TF	ideal forward-transit time	0
CJC	base-collector zero-bias depletion capacitance	0
VJC	base-collector junction built-in potential	0.75
MJC	base-collector junction exponential factor	0.33
XCJC	fraction of CJC that goes to internal base node	1.0
TR	ideal reverse-transit time	0
FC	coefficient for forward-bias depletion capacitance formula	0.5
IS	transport saturation current	1.0E-16
BF	ideal maximum forward beta	100
NF	forward-current emission coefficient	1.0
VAF	Forward Early voltage	∞
IKF	corner for forward-beta high current roll-off	∞
ISE	base-emitter leakage saturation current	0
NE	base-emitter leakage emission coefficient	1.5
BR	ideal maximum reverse beta	1.0
NR	reverse-current emission coefficient	1.0
VAR	Reverse Early voltage	∞
IKR	corner for reverse-beta high-current roll-off	∞
ISC	base-collector leakage saturation current	0
NC	base-collector leakage emission coefficient	2.0
RB	zero-bias base resistance	10
RBM	minimum base resistance at high currents	10
IRB	current when base resistance falls half way to RBM	∞
TEMP	temperature	298°K

PNP PNP Gummel and Poon bipolar transistor model PNP

Form

```
PNP  n1 n2 n3  CJE = x1   VJE = x2   MJE = x3   TF = x4    CJC = x5
               VJC = x6   MJC = x7   XCJC = x8  TR = x9    FC = x10
               IS = x11   BF = x12   NF = x13   VAF = x14  IKF = x15
               ISE = x16  NE = x17   BR = x18   NR = x19   VAR = x20
               IKR = x21  ISC = x22  NC = x23   RB = x24   RBM = x25
               IRB = x26  TEMP = x27;
```

Example

```
PNP  1 2 3  VJE=0.75   MJE=0.33   VJC=0.75      MJC=0.33
            XCJC=1.0   FC=0.5     IS=8.216E-16  BF=408.1
            NF=0.983   VAF=105.8  IKF=7.581E-3  ISE=8.233E-16
            NE=1.283   BR=1.0     NR=1.0        VAR=1000.0
            IKR=1.0E3  NC=2.0     RB=199.3      RBM=52.05
            IRB=2.001E-3;
```

Model Description

PNP is adapted from the integral charge control model of Gummel and Poon [1]. It becomes the simpler Ebers-Moll model when certain parameters required for the Gummel and Poon model are not specified.

Model parameters CJE, MJE and VJE determine the nonlinear depletion-layer capacitance for the base-emitter junction. CJC, MJC and VJC determine the nonlinear depletion-layer capacitance for the base-collector junction. TR and TF along with the depletion-layer capacitances model the base charge storage effects.

Reference

[1] I.E. Getreu, *CAD of Electronic Circuits 1: Modeling the Bipolar Transistor*. Amsterdam: Elsevier Scientific Publishing Company, 1978, pp. 69-123.

7

Linear Elements

7

Linear Elements

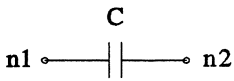
This chapter describes HarPE's library of linear element models.

The description for each element includes a schematic diagram, the number of nodes, the set of parameters and their default values, an example of the syntax, and some application notes if appropriate.

The nodes for each element can be divided into two subsets. The first subset represents nodes which must be nonzero for the element to be meaningfully connected (e.g., the element RES requires at least one nonzero node). The other set represents nodes which are optional. In the description, the optional nodes are enclosed within square brackets. The default for optional nodes that are not explicitly specified is the ground node.

All the element parameters are optional except where noted. Element parameters can be specified by constant values, optimization variables and labels. Parameters that are omitted will be assigned the default values.

CAP capacitor CAP



Keywords

C capacitance	

Defaults

0

Form

```
CAP  n1  [n2]  C = x1;
```

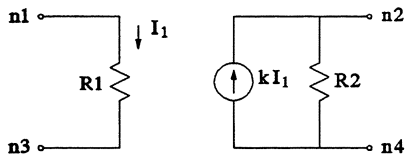
Example

```
CAP  1  2  C = 35PF;
```

Notes

▷ If C = 0 or is not specified the element becomes an open circuit.

CCCS current controlled current source CCCS



Keywords

		Defaults
R1	input resistance	0
R2	output resistance	∞
M	DC value of the controlling coefficient k	0
A	angle of the controlling coefficients	0
F	3 dB cut-off frequency of the controlling coefficient	∞
T	time delay of the controlling coefficient	0

Form

```
CCCS  n1 [n2 n3 n4]  R1 = x1  R2 = x2  M = x3  A = x4  F = x5  T = x6;
```

Example

```
CCCS  1 2 3 4  R1=50KOH  R2=5KOH  M=50;
```

Notes

- The controlling coefficient k is evaluated as

  $$k = M \cdot \exp(j(A - 2\pi f T)) / (1 + jf/F)$$

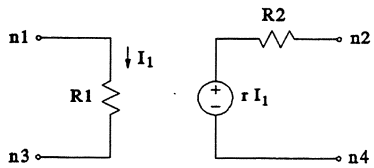
  where M, A (converted to radians, if necessary), T and F are model parameters, and f is the current frequency.

- F, if specified, must be greater than 0.

- An ideal CCCS is defined by the default values for R1 and R2.

- A source of reverse polarity can be specified with a negative value for M.

CCVS current controlled voltage source CCVS



Keywords

		Defaults
R1	input resistance	0
R2	output resistance	0
M	DC value of the transimpedance r	0
A	angle of the transimpedance r	0
F	3 dB cut-off frequency of the transimpedance r	∞
T	time delay of the transimpedance r	0

Form

```
CCVS  n1 [n2 n3 n4]  R1 = x1  R2 = x2  M = x3  A = x4  F = x5  T = x6;
```

Example

```
CCVS  1 2 3 4  R1=50KOH  R2=5KOH  M=50;
```

Notes

- The transimpedance r is evaluated as

  $$r = M \cdot \exp(j(A - 2\pi fT)) \; / \; (1 + jf/F)$$

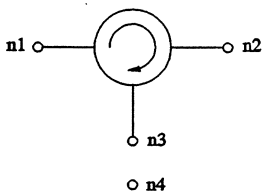
  where M, A (converted to radians, if necessary), T and F are model parameters, and f is the current frequency.

- F, if specified, must be greater than 0.

- An ideal CCVS is defined by the default values for R1 and R2.

- A source of reverse polarity can be specified with a negative value for M.

CIR3 ideal three-port circulator CIR3



Keywords

		Defaults
RREF	reference impedance	50

Form

```
CIR3  n1  n2  n3  [n4]  RREF = x1;
```

Example

```
CIR3  1  2  3;
```

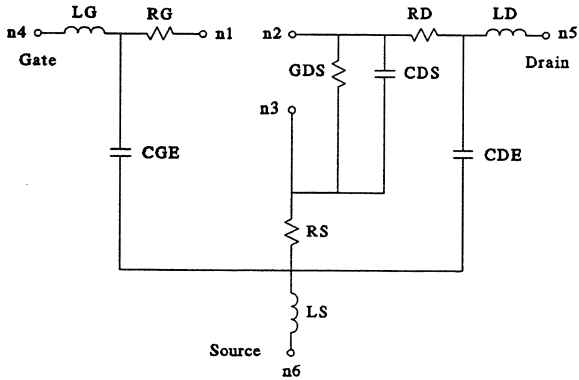
Notes

▷ The ideal three-port circulator is modelled by the following scattering matrix:

$$S = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

The reference impedance of the S parameters can be specified by RREF.

EXTRINSIC1 extrinsic parasitic model 1 EXTRINSIC1



Keywords

		Defaults
CDS	drain-source (collector-emitter) capacitance	0
GDS	drain-source (collector-emitter) conductance	0
RG	gate (base) resistance	0
RD	drain (collector) resistance	0
RS	source (emitter) resistance	0
CGE	external gate (base) capacitance	0
CDE	external drain (collector) capacitance	0
LG	gate (base) lead inductance	0
LD	drain (collector) lead inductance	0
LS	source (emitter) lead inductance	0

Form

```
EXTRINSIC1  n1 n2 n3 n4 n5 [n6]  CDS = x1  GDS = x2  RG = x3  RD = x4
                                 RS = x5   CGE = x6  CDE = x7 LG = x8
                                 LD = x9   LS = x10;
```

Example

```
EXTRINSIC1  1 2 3 4 5  RG=2.4  RD=1.54  RS=1.26
                       CDS=0.09PF  LG=0.001NH;
```

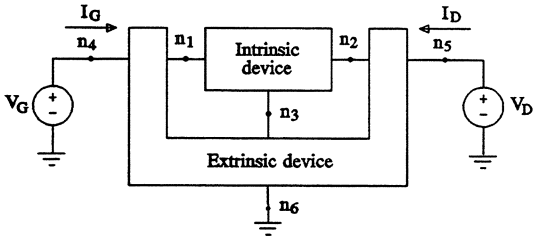
EXTRINSIC1 extrinsic parasitic model 1 EXTRINSIC1

Notes

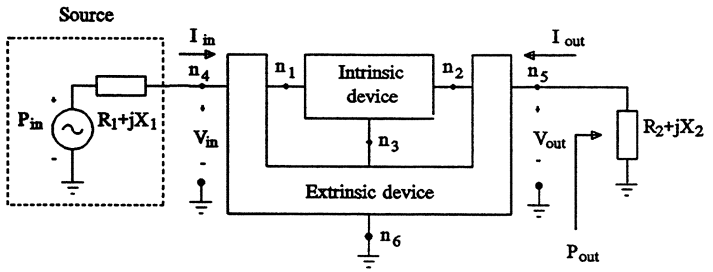
▷ When EXTRINSIC1 is used as the only linear element in the Model block, the nodes n_1, n_2 and n_3 must be connected to the intrinsic device as follows:

n_1 - gate (base)
n_2 - drain (collector)
n_3 - source (emitter)

and the nodes n_4 and n_5 must define the input and the output port, respectively. The overall circuit is then defined as a grounded 2-port, and bias sources are directly connected to the n_4 and n_5 terminals (also see Chapter 5). The overall circuit at DC is:

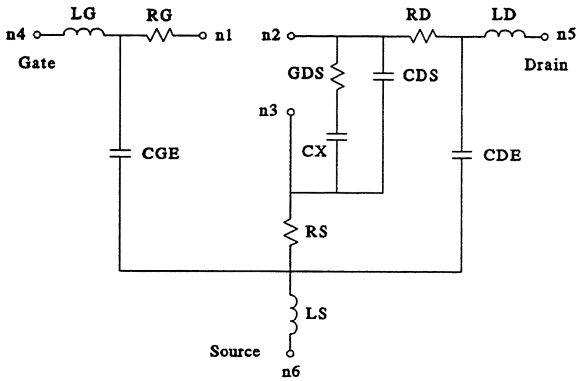


The overall circuit at AC is:



▷ This rule does not apply when EXTRINSIC1 is used together with some other linear elements, in which case all six nodes are freely available for connection to other elements.

EXTRINSIC2 extrinsic parasitic model 2 EXTRINSIC2



Keywords

		Defaults
CDS	drain-source (collector-emitter) capacitance	0
GDS	drain-source (collector-emitter) conductance	0
RG	gate (base) resistance	0
RD	drain (collector) resistance	0
RS	source (emitter) resistance	0
CGE	external gate (base) capacitance	0
CDE	external drain (collector) capacitance	0
CX	drain-source (collector-emitter) capacitance	1.5pF
LG	gate (base) lead inductance	0
LD	drain (collector) lead inductance	0
LS	source (emitter) lead inductance	0

Form

```
EXTRINSIC2  n1 n2 n3 n4 n5 [n6]  CDS = x1  GDS = x2  RG = x3   RD = x4
                                 RS = x5   CGE = x6  CDE = x7  CX = x8
                                 LG = x9   LD = x10  LS = x11;
```

Example

```
EXTRINSIC2  1 2 3 4 5  RG=2.4  RD=1.54  RS=1.26
                       CX=2PF  LG=0.001NH;
```

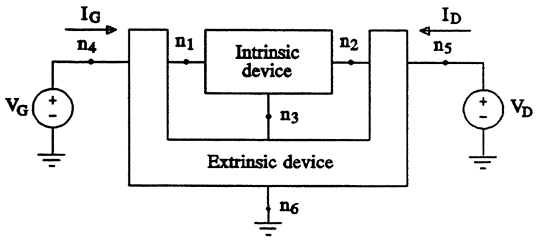
EXTRINSIC2 extrinsic parasitic model 2 EXTRINSIC2

Notes

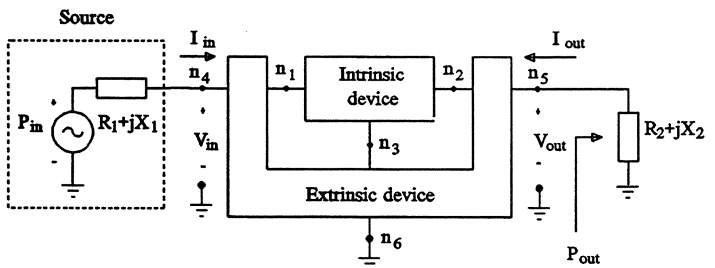
▷ When EXTRINSIC2 is used as the only linear element in the Model block, the nodes n_1, n_2 and n_3 must be connected to the intrinsic device as follows:

n_1 - gate (base)
n_2 - drain (collector)
n_3 - source (emitter)

and the nodes n_4 and n_5 must define the input and the output port, respectively. The overall circuit is then defined as a grounded 2-port, and bias sources are directly connected to the n_4 and n_5 terminals (also see Chapter 5). The overall circuit at DC is:

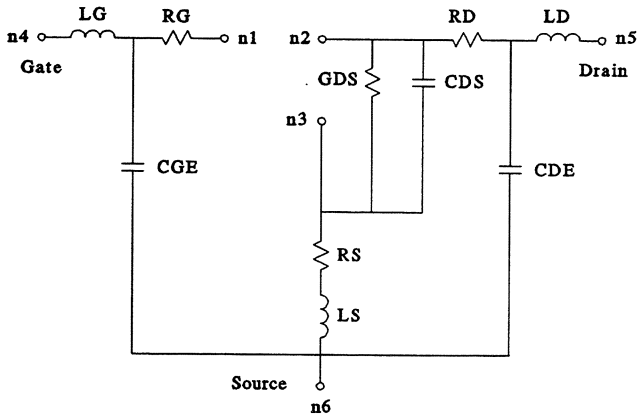


The overall circuit at AC is:



▷ This rule does not apply when EXTRINSIC2 is used together with some other linear elements, in which case all six nodes are freely available for connection to other elements.

EXTRINSIC3 extrinsic parasitic model 3 EXTRINSIC3



Keywords

		Defaults
CDS	drain-source (collector-emitter) capacitance	0
GDS	drain-source (collector-emitter) conductance	0
RG	gate (base) resistance	0
RD	drain (collector) resistance	0
RS	source (emitter) resistance	0
CGE	external gate (base) capacitance	0
CDE	external drain (collector) capacitance	0
LG	gate (base) lead inductance	0
LD	drain (collector) lead inductance	0
LS	source (emitter) lead inductance	0

Form

```
EXTRINSIC3  n1 n2 n3 n4 n5 [n6]  CDS = x1  GDS = x2  RG = x3   RD = x4
                                 RS = x5   CGE = x6  CDE = x7  LG = x8
                                 LD = x9   LS = x10;
```

Example

```
EXTRINSIC3  1 2 3 4 5  RG=2.4  RD=1.54  RS=1.26
                       CDS=0.09PF  LG=0.001NH;
```

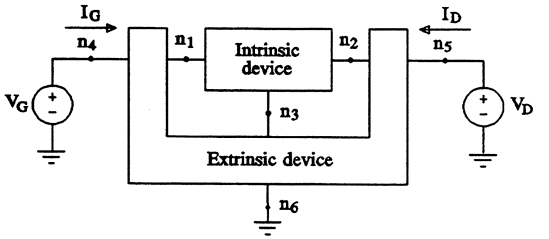
EXTRINSIC3 extrinsic parasitic model 3 EXTRINSIC3

Notes

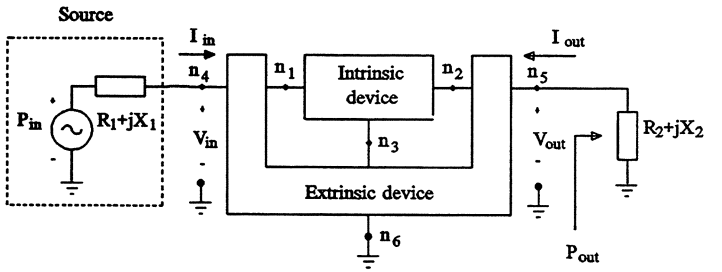
▷ When EXTRINSIC3 is used as the only linear element in the Model block, the nodes n_1, n_2 and n_3 must be connected to the intrinsic device as follows:

n_1 - gate (base)
n_2 - drain (collector)
n_3 - source (emitter)

and the nodes n_4 and n_5 must define the input and the output port, respectively. The overall circuit is then defined as a grounded 2-port, and bias sources are directly connected to the n_4 and n_5 terminals (also see Chapter 5). The overall circuit at DC is:

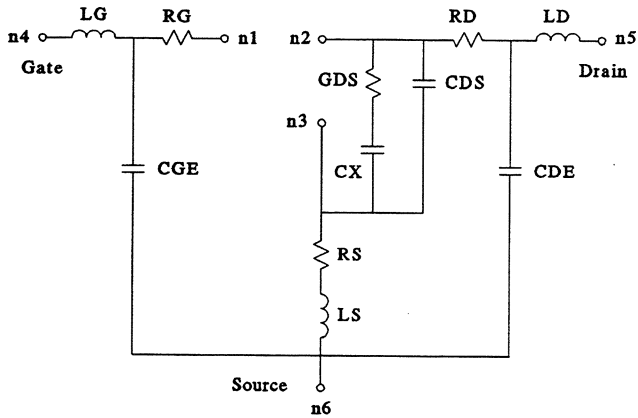


The overall circuit at AC is:



▷ This rule does not apply when EXTRINSIC3 is used together with some other linear elements, in which case all six nodes are freely available for connection to other elements.

EXTRINSIC4 extrinsic parasitic model 4 EXTRINSIC4



Keywords

		Defaults
CDS	drain-source (collector-emitter) capacitance	0
GDS	drain-source (collector-emitter) conductance	0
RG	gate (base) resistance	0
RD	drain (collector) resistance	0
RS	source (emitter) resistance	0
CGE	external gate (base) capacitance	0
CDE	external drain (collector) capacitance	0
CX	drain-source (collector-emitter) capacitance	1.5pF
LG	gate (base) lead inductance	0
LD	drain (collector) lead inductance	0
LS	source (emitter) lead inductance	0

Form

```
EXTRINSIC4  n1 n2 n3 n4 n5 [n6]  CDS = x1  GDS = x2  RG = x3   RD = x4
                                 RS = x5   CGE = x6  CDE = x7  CX = x8
                                 LG = x9   LD = x10  LS = x11;
```

Example

```
EXTRINSIC4  1 2 3 4 5  RG=2.4  RD=1.54  RS=1.26  CX=2PF  LG=0.001NH;
```

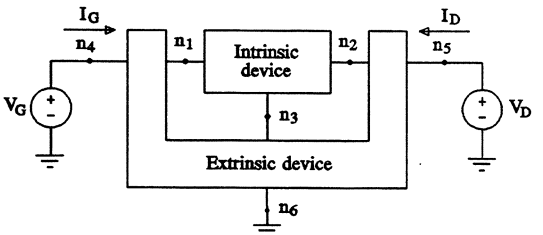
EXTRINSIC4 extrinsic parasitic model 4 EXTRINSIC4

Notes

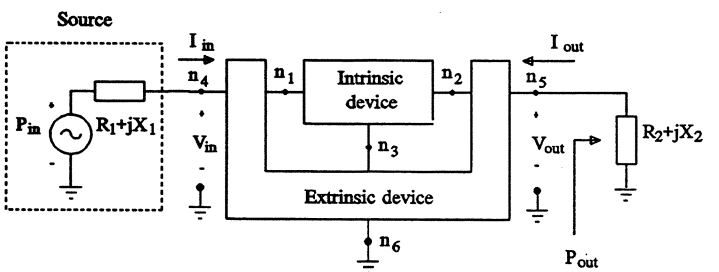
▷ When EXTRINSIC4 is used as the only linear element in the Model block, the nodes n_1, n_2 and n_3 must be connected to the intrinsic device as follows:

n_1 - gate (base)
n_2 - drain (collector)
n_3 - source (emitter)

and the nodes n_4 and n_5 must define the input and the output port, respectively. The overall circuit is then defined as a grounded 2-port, and bias sources are directly connected to the n_4 and n_5 terminals (also see Chapter 5). The overall circuit at DC is:

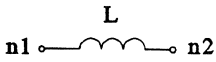


The overall circuit at AC is:



▷ This rule does not apply when EXTRINSIC4 is used together with some other linear elements, in which case all six nodes are freely available for connection to other elements.

IND — inductor — IND



Keywords

		Defaults
L	inductance	0

Form

```
IND  n1 [n2]  L = x1;
```

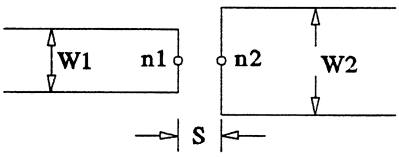
Example

```
IND  1 2  L=7NH;
```

Notes

- If L = 0 or is not specified the element becomes a short circuit and the nodes n1 and n2 are effectively short circuited. However, short circuits are preferably defined by naming the two nodes identically throughout the circuit file.

MAGAP — microstrip asymmetric gap — MAGAP



Keywords

		defaults
W1	line width of conductor 1	required
W2	line width of conductor 2	required
S	gap spacing	required

Form

```
MAGAP  n1 n2  W1 = x1  W2 = x2  S = x3;
```

Example

```
MAGAP  1 2  W1=0.65mm  W2=1mm  S=0.2mm;
```

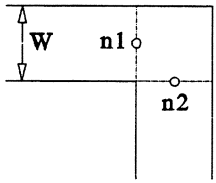
Notes

- An MSUB statement should be placed before MAGAP to provide the substrate definition.
- MAGAP is valid for the range of parameters
  $0.2 \leq S/H$
  $0.1 \leq W1/H \leq 3.0$
  $0.1 \leq W2/H \leq 3.0$
  $1 \leq W2/W1$ (or $W1/W2$) ≤ 3
  $6 \leq EPSR \leq 13$
  where H and EPSR are substrate parameters.

Reference

[1] M. Kirschning, R.H. Jansen and N.H.L. Koster, "Measurement and computer-aided modeling of microstrip discontinuities by an improved resonator method", *IEEE MTT-S Int. Microwave Symp. Dig.* (Boston, MA), May 1983, pp. 495-497.

MBEND1 microstrip 90 degree bend MBEND1



Keywords

		defaults
W	line width	required

Form

```
MBEND1  n1 n2  W = x1;
```

Example

```
MBEND1  1 2  W=1mm;
```

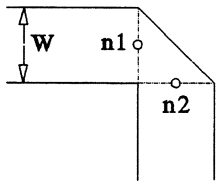
Notes

- An MSUB statement should be placed before MBEND1 to provide the substrate definition.
- MBEND1 is valid for the range of parameters
  $0.2 \leq W/H \leq 6.0$
  $2 \leq EPSR \leq 13$
  where H and EPSR are substrate parameters.
- MBEND1 was validated in [1] for the frequencies up to 14GHZ.

Reference

[1] M. Kirschning, R. Jansen and N. Koster, "Measurement and computer-aided modeling of microstrip discontinuities by an improved resonator method", *IEEE MTT-S Int. microwave Symp. Dig.* (Boston, MA), 1983, pp. 495-497.

MBEND2 microstrip chamfered 90 degree bend MBEND2



Keywords

		defaults
W	line width	required

Form

```
MBEND2  n1 n2  W = x1;
```

Example

```
MBEND2  1 2  W=1mm;
```

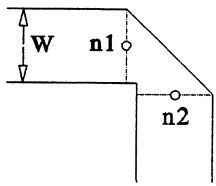
Notes

- An MSUB statement should be placed before MBEND2 to provide the substrate definition.
- MBEND2 is valid for the range of parameters
  $0.2 \leq W/H \leq 6.0$
  $2 \leq EPSR \leq 13$
  where H and EPSR are substrate parameters.
- MBEND2 was validated in [1] for the frequencies up to 14GHZ.

Reference

[1] M. Kirschning, R. Jansen and N. Koster, "Measurement and computer-aided modeling of microstrip discontinuities by an improved resonator method", *IEEE MTT-S Int. microwave Symp. Dig.* (Boston, MA), 1983, pp. 495-497.

MBEND3 microstrip optimally-chamfered 90 degree bend MBEND3



Keywords

		defaults
W	line width	required

Form

```
MBEND3  n1 n2  W = x1;
```

Example

```
MBEND3  1 2  W=1mm;
```

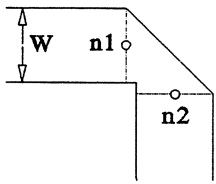
Notes

▷ An MSUB statement should be placed before MBEND3 to provide the substrate definition.

Reference

[1] E. Hammerstad, "Computer-aided design of microwave couplers with accurate discontinuity models", *IEEE MTT-S Int. Microwave Symp. Dig.* (Los Angeles, CA), 1981, pp. 54-56.

MBEND3A microstrip optimally-chamfered 90 degree bend MBEND3A



Keywords

defaults

W line width — required

Form

```
MBEND3A  n1 n2  W = x1;
```

Example

```
MBEND3A  1 2  W=1mm;
```

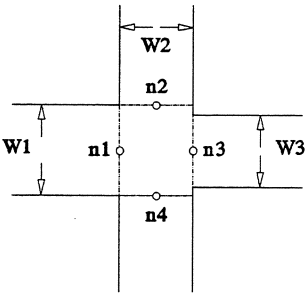
Notes

▷ An MSUB statement should be placed before MBEND3A to provide the substrate definition.

Reference

[1] I.J. Bahl and P. Bhartia, *Microwave Solid State Circuit Design*, New York, NY: John Wiley & Sons, 1989.

MCROSS microstrip cross-junction MCROSS



Keywords

		defaults
W1	line width at node n1	required
W2	line width at node n2 and node n4	required
W3	line width at node n3	required

Form

```
MCROSS  n1 n2 n3 n4  W1 = x1  W2 = x2  W3 = x3;
```

Example

```
MCROSS  1 2 3 4  W1=1mm  W2=1mm  W3=0.7mm;
```

Notes

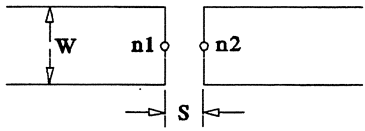
- An MSUB statement should be placed before MCROSS to provide the substrate definition.
- MCROSS is valid for the range of parameters: $0.5 \leq$ (W1/H, W2/H, W3/H) ≤ 2.0, where H is a substrate parameter.
- Scaling is used for EPSR $\neq 9.9$, where EPSR is a substrate parameter.

References

[1] K.C. Gupta, R. Garg and I.J. Bahl, *Microstrip Lines and Slotlines*, Dedham, MA: Artech House, 1979.

[2] K. Chang (ed.), *Handbook of Microwave and Optical Components*, vol. 1, Microwave Passive and Antenna Components, New York, NY: John Wiley & Sons, 1989.

MGAP microstrip symmetric gap MGAP



Keywords

		defaults
W	line width	required
S	gap spacing	required

Form

```
MGAP  n1 n2  W = x1  S = x2;
```

Example

```
MGAP  1 2  W=0.65mm  S=0.2mm;
```

Notes

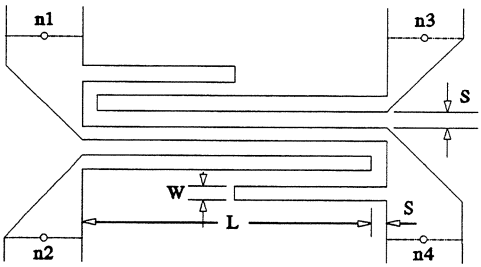
- An MSUB statement should be placed before MGAP to provide the substrate definition.
- MGAP was formulated in [1] by fitting data from [2] for the range of parameters
  $0.5 \leq W/H \leq 2.0$
  $1.0 \leq EPSR \leq 15$
  $0.1 \leq S/H \leq 2.0$
  where H and EPSR are substrate parameters.

References

[1] E. Hammerstad, "Computer-aided design of microwave couplers with accurate discontinuity models", *IEEE MTT-S Int. Microwave Symp. Dig.* (Los Angeles, CA), 1981, pp. 54-56.

[2] P. Benedek and P. Silvester, "Equivalent capacitances for microstrip gaps and steps", *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-20, 1972, pp. 729-733.

MLANG4 four-finger microstrip Lange coupler MLANG4



Keywords

		defaults
W	finger width	required
L	finger length	required
S	finger spacing	required

Form

```
MLANG4  n1 n2 n3 n4  W = x1  L = x2  S = x3;
```

Example

```
MLANG4  1 2 3 4  W=0.6mm  L=10mm  S=0.3mm;
```

Notes

- An MSUB statement should be placed before MLANG4 to provide the substrate definition.
- MLANG4 was validated in [2] for the range of parameters
  $0.1 \leq W/H \leq 3.0$
  $1.0 \leq EPSR \leq 20$
  $0.068 \leq S/H \leq 0.9$
  where H and EPSR are substrate parameters.

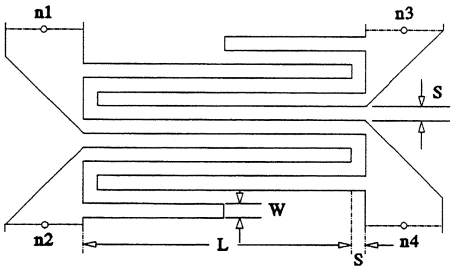
MLANG4 four-finger microstrip Lange coupler MLANG4

References

[1] D. Kajfez, Z. Paunovic and S. Pavlin, "Simplified design of Lange coupler", *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-26, 1977, pp. 806-808.

[2] F. Sellberg, "Simple and accurate algorithms to include Lange and asymmetric microstrip couplers in CAD-packages", *Proc. 18th European Microwave Conf.* (Stockholm, Sweden), 1988, pp. 1157-1162.

MLANG6 six-finger microstrip Lange coupler MLANG6



Keywords

		defaults
W	finger width	required
L	finger length	required
S	finger spacing	required

Form

```
MLANG6  n1 n2 n3 n4  W = x1  L = x2  S = x3;
```

Example

```
MLANG6  1 2 3 4  W=0.6mm  L=10mm  S=0.3mm;
```

Notes

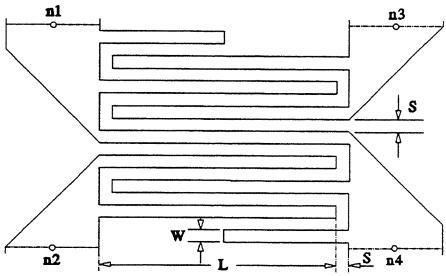
- An MSUB statement should be placed before MLANG6 to provide the substrate definition.
- MLANG6 was validated in [2] for the range of parameters
  $0.1 \leq W/H \leq 3.0$
  $1.0 \leq EPSR \leq 20$
  $0.068 \leq S/H \leq 0.9$
  where H and EPSR are substrate parameters.

References

[1] D. Kajfez, Z. Paunovic and S. Pavlin, "Simplified design of Lange coupler", *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-26, 1977, pp. 806-808.

[2] F. Sellberg, "Simple and accurate algorithms to include Lange and asymmetric microstrip couplers in CAD-packages", *Proc. 18th European Microwave Conf.* (Stockholm, Sweden), 1988, pp. 1157-1162.

MLANG8 eight-finger microstrip Lange coupler MLANG8



Keywords

		defaults
W	finger width	required
L	finger length	required
S	finger spacing	required

Form

```
MLANG8  n1 n2 n3 n4  W = x1  L = x2  S = x3;
```

Example

```
MLANG8  1 2 3 4  W=0.6mm  L=10mm  S=0.3mm;
```

Notes

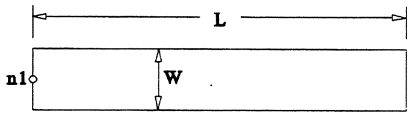
- An MSUB statement should be placed before MLANG8 to provide the substrate definition.
- MLANG8 was validated in [2] for the range of parameters
  $0.1 \leq W/H \leq 3.0$
  $1.0 \leq EPSR \leq 20$
  $0.068 \leq S/H \leq 0.9$
  where H and EPSR are substrate parameters.

References

[1] D. Kajfez, Z. Paunovic and S. Pavlin, "Simplified design of Lange coupler", *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-26, 1977, pp. 806-808.

[2] F. Sellberg, "Simple and accurate algorithms to include Lange and asymmetric microstrip couplers in CAD-packages", *Proc. 18th European Microwave Conf.* (Stockholm, Sweden), 1988, pp. 1157-1162.

MOPEN — microstrip open stub — MOPEN



Keywords

		defaults
W	line width	required
L	line length	0

Form

```
MOPEN  n1  W = x1  L = x2;
```

Example

```
MOPEN  1  W=0.5mm  L=2mm;
```

Notes

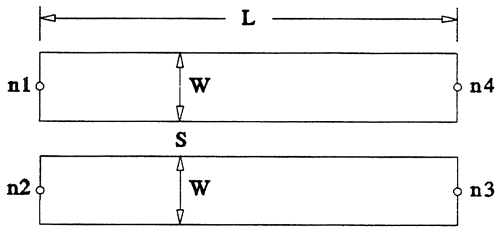
- An MSUB statement should be placed before MOPEN to provide the substrate definition.
- The valid range of the parameters is given by [2] as
  $0.1 \leq W/H \leq 10$
  $1.0 \leq EPSR \leq 18$
  $f(GHz)\text{-}H(mm) \leq 30$
  where H and EPSR are substrate parameters, and f is the frequency.

Reference

[1] M. Kirschning, R. Jansen and N. Koster, "Accurate model for open end effect of microstrip lines", *Electronics Letters*, vol. 17, Feb. 5, 1981, pp. 123-125.

[2] R. Jansen and M. Kirschning, "Arguments and an accurate model for the power-current formulation of microstrip characteristic impedance", *Arch. Elek. Ubertragung (AEU)*, vol. 37, 1983, pp.108-112.

MSCL two-conductor symmetrical coupled microstrip lines MSCL



Keywords

		Defaults
W	line width	required
L	line length	required
S	line spacing	required

Form

```
MSCL  n1 n2 n3 n4  W = x1  L = x2  S = x3;
```

Example

```
MSCL  1 2 3 4  W=0.5mm  L=2mm  S=0.2mm;
```

Notes

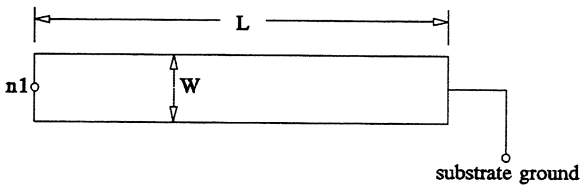
- An MSUB statement should be placed before MSCL to provide the substrate definition.
- MSCL is accurate for the range of parameters
  $0.1 \leq W/H \leq 10$
  $1.0 \leq EPSR \leq 18$
  $f(GHz)\cdot H(mm) \leq 30$
  where H and EPSR are substrate parameters, and f is the frequency.

References

[1] S. March, "Microstrip packaging: watch the last step", *Microwaves*, Dec. 1981, pp. 83-94.

[2] M. Kirschning and R. Jansen, "Accurate wide-range design equations for the frequency-dependent characteristics of parallel coupled microstrip lines", *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-32, 1984, pp. 83-90. Corrections: *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-33, 1985, p. 288.

MSHORT — microstrip short stub — MSHORT



Keywords

		defaults
W	line width	required
L	line length	required

Form

```
MSHORT  n1  W = x1  L = x2;
```

Example

```
MSHORT  1  W=0.5mm  L=3mm;
```

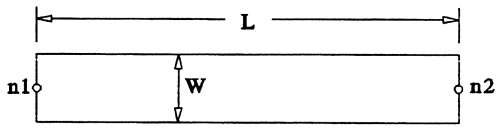
Notes

- An MSUB statement should be placed before MSHORT to provide the substrate definition.
- The valid range of the parameters is given by [1] as
  $0.1 \leq W/H \leq 10$
  $1.0 \leq EPSR \leq 18$
  $f(GHz) \cdot H(mm) \leq 30$
  where H and EPSR are substrate parameters, and f is the frequency.
- Ideal short circuit is assumed for this component.

References

[1] R. Jansen and M. Kirschning, "Arguments and an accurate model for the power-current formulation of microstrip characteristic impedance", *Arch. Elek. Ubertragung (AEU)*, vol. 37, 1983, pp.108-112.

MSL microstrip line MSL



Keywords

		Defaults
W	line width	required
L	line length	required

Form

```
MSL  n1 n2  W = x1  L = x2;
```

Example

```
MSL  1 2  W=0.5mm  L=3.5mm;
```

Notes

- An MSUB statement should be placed before MSL to provide the substrate definition.
- MSL is accurate for the range of parameters
  $0.1 \leq W/H \leq 10$
  $1.0 \leq EPSR \leq 18$
  $f(GHz) \cdot H(mm) \leq 30$
  where H and EPSR are substrate parameters, and f is the frequency.

References

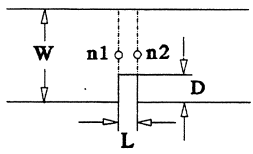
[1] E. Hammerstad and O. Jensen, "Accurate models for microstrip computer-aided design", *IEEE MTT-S Int. Microwave Symp. Dig.* (Washington, DC), 1980, pp. 407-409.

[2] S. March, "Microstrip packaging: watch the last step", *Microwaves*, Dec. 1981, pp. 83-94.

[3] M. Kirschning and R. Jansen, "Accurate model for effective dielectric constant of microstrip with validity up to millimetre-wave frequencies", *Electronics Letters*, vol. 18, no. 6, 1982, pp. 272-273.

[4] R. Jansen and M. Kirschning, "Arguments and an accurate model for the power-current formulation of microstrip characteristic impedance", *Arch. Elek. Ubertragung (AEU)*, vol. 37, 1983, pp.108-112.

MSLIT narrow transverse slit in microstrip MSLIT



Keywords

		defaults
W	line width	required
D	slit depth	0
L	slit length	required

Form

```
MSLIT  n1 n2  W = x1  D = x2  L = x3;
```

Example

```
MSLIT  1 2  W=0.5mm  D=0.15mm  L=0.1mm;
```

Notes

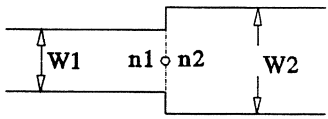
- An MSUB statement should be placed before MSLIT to provide the substrate definition.
- MSLIT is valid for the range of parameters
  $0.1 \leq W/H \leq 10$
  $0.0 \leq D/W \leq 0.9$
  $1.0 \leq EPSR \leq 18$
  $f(GHz) \cdot H(mm) \leq 30$
  where H and EPSR are substrate parameters, and f is the frequency.
- L should be smaller than H and should be much smaller than the propagation wavelength.

References

[1] W.J.R. Hoefer, "Theoretical and experimental characterization of narrow transverse slits in microstrip", *NTZ*, Bd. 30, 1977, pp. 582-585.

[2] W.J.R. Hoefer, "Equivalent series inductivity of a narrow transverse slit in microstrip", *IEEE Trans. Microwave Theory and Tech.*, vol. MTT-25, 1977, pp. 822-824.

MSTEP — microstrip step — MSTEP



Keywords

		defaults
W1	line width at node n1	required
W2	line width at node n2	required

Form

```
MSTEP  n1 n2  W1 = x1  W2 = x2;
```

Example

```
MSTEP  1 2  W1=0.6mm  W2=1mm;
```

Notes

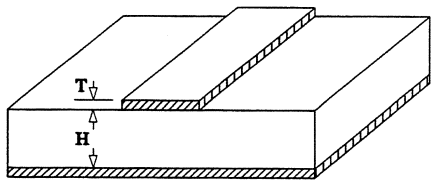
▷ An MSUB statement should be placed before MSTEP to provide the substrate definition.

References

[1] G. Kompa, "Design of stepped microstrip components", *The Radio and Electronic Engineer*, vol. 48, no. 1/2, Jan./Feb. 1978, pp.53-63.

[2] I.J. Bahl and P. Bhartia, *Microwave Solid State Circuit Design*, New York, NY: John Wiley & Sons, 1989.

MSUB microstrip substrate definition MSUB



Keywords

		Defaults
EPSR	dielectric constant of the substrate	2.35
H	substrate height	0.635mm
H2	shielding cover height	∞
T	conducting metal thickness	0
TAND	substrate dielectric loss tangent	0
ROC	resistivity of the conducting strip in Ωm	0
RHS	surface roughness of the conducting strip in m	0

Form

```
MSUB  [n]  EPSR = x1  H = x2  H2 = x3  T = x4  TAND = x5  ROC = x6  RHS = x7;
```

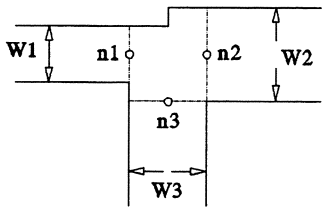
Example

```
MSUB  EPSR=9.7  H=0.635mm  T=0.006mm  TAND=0.0002  ROC=0.0172E-6  RHS=0.5E-6;
```

Notes

- MSUB is required to define the substrate information for microstrip components.
- MSUB should be placed before any microstrip components.
- Multiple MSUB can be used in one circuit file with the current one overriding the previous one.
- The node n is used to define the connection of the ground plane. If not defined, the ground plane of the substrate is by default connected to the global ground node 0.

MTEE — microstrip T-junction — MTEE



Keywords

		defaults
W1	line width at node n1	required
W2	line width at node n2	required
W3	line width at node n3	required

Form

```
MTEE  n1 n2 n3  W1 = x1  W2 = x2  W3 = x3;
```

Example

```
MTEE  1 2 3  W1=0.65mm  W2=0.65mm  W3=0.5mm;
```

Notes

▷ An MSUB statement should be placed before MTEE to provide the substrate definition.

Reference

[1] E. Hammerstad, "Computer aided design of microstrip couplers with accurate discontinuity models", *IEEE MTT-S Int. Microwave Symp. Dig.* (Los Angeles, CA), 1981, pp. 54-56.

OPEN — open circuit — OPEN

Form

```
OPEN  n1 [n2];
```

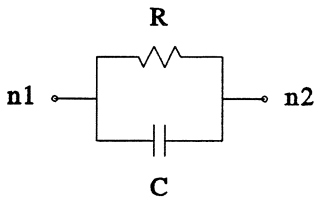
Example

```
OPEN  1 2;
```

Notes

▷ **You can use OPEN to override the circuit file parser's error checking in cases of intentionally introduced floating nodes, for example, at the open end of a transmission line.**

PRC parallel connection of resistor and capacitor PRC



Keywords

		Defaults
R	resistance	∞
C	capacitance	0

Form

```
PRC  n1 [n2]  R = x1  C = x2;
```

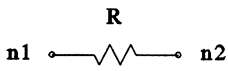
Example

```
PRC  1 2  R=3.5  C=100PF;
```

Notes

- R, if specified, must be greater than 0.
- If the keyword R or C is missing the element defaults to the CAP or RES element, respectively.

RES resistor RES



Keywords

Defaults

R	resistance	0.01ohm

Form

```
RES  n1 [n2]  R = x1;
```

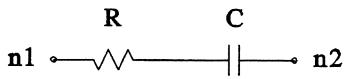
Example

```
RES  1 2  R=3.5;
```

Notes

▷ R, if specified, must be greater than 0.

SRC series connection of resistor and capacitor SRC



Keywords

		Defaults
R	resistance	0.01ohm
C	capacitance	∞

Form

```
SRC  n1 [n2]  R = x1  C = x2;
```

Example

```
SRC  1 2  R=3.5  C=100PF;
```

Notes

▷ If the keyword C is missing the element defaults to the RES element.

SRL series connection of resistor and inductor SRL

Keywords

		Defaults
R	resistance	0.01ohm
L	inductance	0

Form

```
SRL  n1 [n2]  R = x1  L = x2;
```

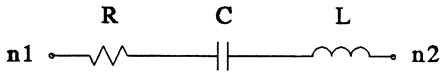
Example

```
SRL  1 2  R=3.5  L=10UH;
```

Notes

- R, if specified, must be greater than 0.
- If the keyword L is missing the element defaults to the RES element.

SRLC series connection of resistor, inductor and capacitor SRLC



Keywords

		Defaults
R	resistance	0.01ohm
L	inductance	0
C	capacitance	∞

Form

```
SRLC  n1 [n2]  R = x1  L = x2  C = x3;
```

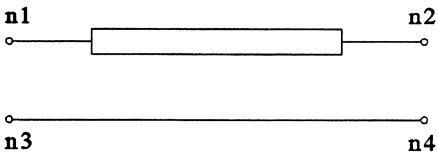
Example

```
SRLC  1 2  R=3.5  L=10UH  C=100PF;
```

Notes

- R, if specified, must be greater than 0.
- If the keyword L or C is missing the element defaults to the SRC or SRL element, respectively.

TEM — ideal transmission line — TEM



Keywords

		Defaults
Z	characteristic impedance	50ohm
E	electrical length	90deg
F	frequency at which E is specified	1GHz

Form

```
TEM  n1 [n2 n3 n4]  Z = x1  E = x2  F = x3;
```

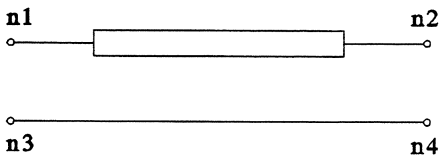
Example

```
TEM  1 2  Z=45  E=1RAD  F=10GHZ;
```

Notes

- Z, E and F, if specified, must be greater than 0.
- The default unit for electrical length is degree.

TRL transmission line, physical model TRL



Keywords

		Defaults
Z	characteristic impedance	50ohm
L	physical length	0.01m
K	effective dielectric constant	1
A	attenuation per unit length	0
F	frequency at which A is specified	0

Form

```
TRL  n1 [n2 n3 n4]  Z = x1  L = x2  K = x3  A = x4  F = x5;
```

Example

```
TRL  1 2  Z=45  L=15MM  K=9.7  A=3  F=10GHZ;
```

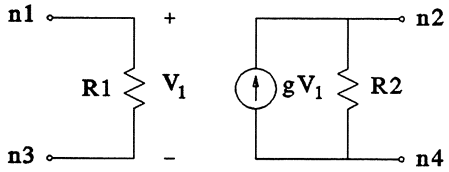
Notes

- Z and L, if specified, must be greater than 0.
- K, if specified, must be greater than or equal to 1.
- A and F, if specified, must be greater than or equal to 0.
- Attenuation A is scaled with the square root of the current frequency f relative to the frequency F, i.e.,

  $$A(f) = A \cdot \sqrt{f / F} \quad \text{if } F > 0,$$

  and is constant with frequency if F = 0 or is not specified.
- The default unit for attenuation is dB per meter.

VCCS — voltage controlled current source — VCCS



Keywords

		Defaults
R1	input resistance	∞
R2	output resistance	∞
M	DC value of the transadmittance g	0
A	angle of the transadmittance g	0
F	3 dB cut-off frequency of the transadmittance g	∞
T	time delay of the transadmittance g	0

Form

```
VCCS  n1 [n2 n3 n4]  R1 = x1  R2 = x2  M = x3  A = x4  F = x5  T = x6;
```

Example

```
VCCS  1 2 3 4  R1=50KOH  R2=5KOH  M=50;
```

Notes

- The transadmittance g is evaluated as

  $$g = M \cdot \exp(j(A - 2\pi f T)) \; / \; (1 + jf/F)$$

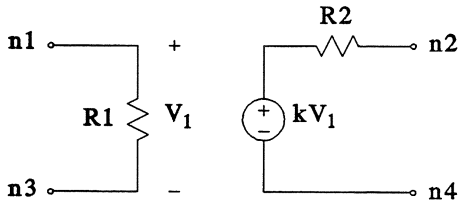
  where M, A (converted to radians, if necessary), T and F are model parameters, and f is the current frequency.

- F, if specified, must be greater than 0.

- An ideal VCCS is defined by the default values for R1 and R2.

- A source of reverse polarity can be specified with a negative value for M.

VCVS voltage controlled voltage source VCVS



Keywords

		Defaults
R1	input resistance	∞
R2	output resistance	0
M	DC value of the controlling coefficient k	0
A	angle of the controlling coefficients	0
F	3 dB cut-off frequency of the controlling coefficient	∞
T	time delay of the controlling coefficient	0

Form

```
VCVS  n1 [n2 n3 n4]  R1 = x1  R2 = x2  M = x3  A = x4  F = x5  T = x6;
```

Example

```
VCVS  1 2 3 4  R1=50KOH  R2=50  M=10;
```

Notes

- The controlling coefficient k is evaluated as

  $$k = M \cdot \exp(j(A - 2\pi f T)) / (1 + jf/F)$$

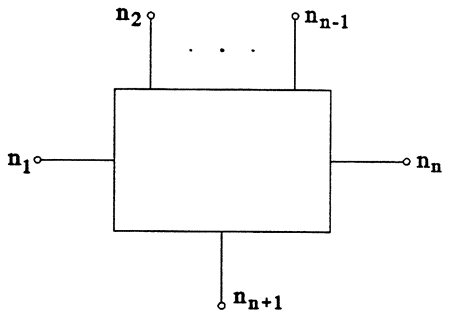
  where M, A (converted to radians, if necessary), T and F are model parameters, and f is the current frequency.

- F, if specified, must be greater than 0.

- An ideal VCVS is defined by the default values for R1 and R2.

- A source of reverse polarity can be specified with a negative value for M.

nPORDATA — imported n-port subcircuit — nPORDATA



Keywords

Defaults

DATA name of the imported data	required

Form

```
nPORDATA  n1 ... nn  [nn+1]  DATA = name;
```

Examples

```
1PORDATA  1 0  DATA=import_impedance;

2PORDATA  1 2 3  DATA=import_2port;
```

Notes

- The number of ports n must be between 1 and 9.
- The name specified after the keyword DATA must have been defined in the ImportData block (see Chapter 5).
- The imported data must represent an n-port with the $(n+1)$th node being the reference node (i.e., the data must be an $n \times n$ S, Y or Z matrix). When the data is used as an nPORDATA element the reference node, i.e., the $(n+1)$th node, can be connected to a nonzero node.

8

Simulation

8

Simulation

8.1 Overview

This chapter discusses the circuit file Sweep block and the HarPE.Display menu option.

The Sweep block allows you to define the ranges of sweep labels for circuit simulation. Sweep labels include bias voltages, frequency and input power level. User-defined labels can also be swept.

The HarPE.Display menu option invokes circuit simulation according to the sweep ranges defined in the Sweep block and then presents you with a menu of different display formats and options.

8.2 Sweep Block

Syntax:

```
Sweep
   sweep_set;
   sweep_set;
   ...

   sweep_set;
End
```

where each *sweep_set* is a statement delimited by a semicolon ";".

Syntax of sweep sets:

```
sweep_label: sweep_range
...
sweep_label: sweep_range
user_label  user_label  ...  user_label;
```

where each *sweep_label* represents a built-in or user-defined sweep parameter, *sweep_range* defines one or more values for the corresponding *sweep_label*, and each *user_label* (optional) represents a user-defined response.

Sweep Labels

Table 8.1 lists the set of built-in sweep labels (which is a subset of the predefined labels described in Chapter 4).

TABLE 8.1 BUILT-IN SWEEP LABELS

Label	Description
VD	drain (collector) bias voltage
VG	gate (base) bias voltage
FREQ	frequency or fundamental frequency
PIN	input power level

Note: VD is not applicable if only one bias port is defined.

Circuit Simulation Type

The type of circuit simulation to be performed (large-signal, small-signal or DC) is implied by the sweep labels defined, as shown in Table 8.2.

TABLE 8.2 SWEEP LABELS AND SIMULATION TYPES

Sweep Labels	Implied Simulation Type
VG, VD	DC
VG, VD, FREQ	small-signal AC
VG, VD, FREQ, PIN	large-signal (harmonic balance)

Example:

```
Sweep
   VG: from -3 to 0 step=0.5   VD: from 0 to 6 step=0.2;

   VG: -1.74   VD: 3 4.5 5   FREQ: from 2GHZ to 18GHZ step=1GHZ;

   VG: -2  VD: 4  FREQ: 6GHZ  PIN: from -15dBm to 5dBm step=5dBm;
End
```

DC simulation is assumed for the first sweep set since it contains sweep ranges for the bias voltages VG and VD only. Similarly, the simulation types for the second and third sweep sets are assumed to be small-signal AC and large-signal HB, respectively.

Define Discrete Values for Sweep Labels

Syntax:

sweep_label: *x1*, *x2*, ... *xn*

where *x1*, *x2*, ..., *xn* represent numerical values.

Example:

```
FREQ: 2, 3.2, 5.7, 13, 15
```

The comma separating two discrete values is optional.

Sweep an Interval with Uniform Step Size

Syntax:

sweep_label: from *x1* to *x2* step=*x3*

where *x1*, *x2* and *x3* are numerical values. This translates into a set of values as: *x1*, *x1+x3*, *x1+2xx3*, ..., *x2*.

Example:

```
VD: from 0 to 6 step=0.2
```

This translates into a set of values for VD as 0, 0.2, 0.4, ..., 6.

The upper limit of the interval, namely *x2*, is always included in the translated values, even if the step size does not divide the interval evenly. For example,

```
VD: from 1.5 to 2.75 step=0.5
```

The translated values are 1.5, 2, 2.5, 2.75, which include the upper limit 2.75.

You can specify a negative step size, such as

```
VD:  from 5 to 1.5 step=-0.5
```

The translated values are 5, 4.5, 4, ..., 1.5. In this case, the lower limit must be greater than the upper limit, i.e., $x1 > x2$.

The number of subintervals is $n = (x2 - x1) / x3$, if $x3$ divides $(x2 - x1)$ evenly, or else $n = (x2 - x1) / x3 + 1$, (the number of values is $n + 1$). The limit is $n \leq 1023$.

Specify the Number of Steps

Syntax:

sweep_label: `from` *x1* `to` *x2* `N=`*n*

where *x1* and *x2* are numerical values, *n* is an integer, $1 \leq n \leq 1023$.
The step size is $(x2 - x1) / n$. The number of values is $n + 1$.

Example:

```
VD:  from 0 to 4 N=8
```

This translates into a step size of (4 - 0) / 8 = 0.5, and a set of values as 0, 0.5, 1, ..., 4. Note that the total number of values is 9.

Combine Intervals and Points

You can specify a combination of sweep intervals and discrete values for a sweep label. For example,

```
VD:  0  0.1  from 0.2 to 1 step=0.2  from 1.5 to 6 step=0.5
```

which translates into the following set of values:

```
0  0.1  0.2 0.4 0.6 0.8 1  1.5 2 2.5 3 3.5 4 4.5 5 5.5 6
```

This can be useful if you wish to have different step sizes in different sweep ranges. For example, an accurate representation of the circuit DC characteristics usually requires more densely spaced bias points below the "knee".

The set of values assigned to a sweep label will not be sorted by the file parser. During simulation, the values are assigned in the exact order as they appear in the circuit file. If the values need to be processed in a particular order (ascending, descending, etc.), they must be entered in the desired order.

Multiple Sweep Labels

When multiple sweep labels are defined in a sweep set, the circuit simulation is performed at all the possible combinations of their values.

Example:

```
VG: 0, -1  VD: 1, 2;
```

For this sweep set, DC simulation is performed at four bias points, namely

```
(VG, VD) = (0, 1), (0, 2), (-1, 1), (-1, 2)
```

If, however, you wish to consider only two points given by

```
(VG, VD) = (0, 1), (-1, 2)
```

then you should define two separate sweep sets as

```
VG:  0  VD: 1;

VG: -1  VD: 2;
```

If the Sweep Block Is Omitted

If you omit the SWEEP block from the circuit file, then you must supply the Data block and the circuit simulation will be performed at the bias points, frequencies and input power levels contained in the measured data (☞ the Data block is described in Chapter 9).

8.3 User-Defined Sweep Labels

The built-in sweep labels allow you to define bias voltage sweep, frequency sweep and input power sweep. In addition, user-defined labels defined in the circuit file can be swept.

For example, you can define a label Delay in the Expression block, use it to represent the time delay of the FET model in the Model block, and then include it as a sweep label in the Sweep block:

```
Expression
   Delay: 3PS;
   ...
End

Model
   FETC  ...  TAU=Delay  ...;
   ...
End

Sweep
   Delay: from 2PS to 10PS step=1PS  ... ;
   ...
End
```

User-defined sweep labels can be combined with bias sweep, frequency sweep and power sweep. For example,

```
Delay: from 2PS to 10PS step=1PS  VG: -1.7  VD: 4  FREQ: 6GHZ
PIN: from -10DBM to 10DBM step=5DBM;
```

Only one user-defined sweep label is allowed per sweep set. You can not sweep more than one user-defined label in one sweep set. You can, however, sweep different user-defined labels in different sweep sets.

User-Defined Sweep Labels Must Be Constants or Variables

User-defined sweep labels must be defined in the Expression block to represent constants or optimization variables. Labels which represent expressions (formulas) cannot be swept.

Example:

```
Cross_Section: 3;
Length: ?10cm?;
Resistance = Resistivity * Length / Cross_Section;
```

You can use Cross_Section and Length as sweep labels, because they represent a constant and an optimization variable, respectively. But Resistance cannot be a sweep label because it represents a formula.

8.4 Keywords HARM and RREF

Highest Harmonic in Large-Signal Simulation

For large-signal sweep sets, you can specify the highest harmonic to be included in harmonic balance simulation using the keyword HARM.

Syntax:

HARM=*m* ...;

m specifies the highest harmonic for the sweep set, $1 \leq m \leq 8$.
The default is HARM = 4.

Example:

```
VG: -2  VD: 4  PIN: from -10DBM to 10DBM step=5DBM
FREQ: 6GHZ  HARM=5;
```

For this sweep set, the first 5 harmonics and the DC component will be included in the harmonic balance simulation.

A higher value for HARM may improve the convergence of the Newton iteration for solving the nonlinear harmonic balance equations. A higher value for HARM can also be used to improve the accuracy of simulation, at the expense of a longer simulation time.

Reference Impedance for S Parameters

For small-signal AC sweep sets, you can specify the reference impedance for *S* parameters using the keyword RREF.

Syntax:

RREF=*x* ...;

where *x* specifies the *S*-parameter reference impedance for the sweep set,
x must be a positive constant value, and the default is 50OH.

Example:

```
VG: -1.5 -2  VD: 4  FREQ: from 2GHZ to 18GHZ step=1GHZ  RREF=10;
```

where the reference impedance for the *S* parameters is specified as 10OH.

8.5 Responses

The purpose of simulation is to calculate certain responses of the circuit model for display. HarPE can calculate and display both built-in and user-defined circuit responses.

Built-In Circuit Responses

Corresponding to each type of simulation (DC, small-signal or large-signal) is a set of built-in circuit responses, as listed in Table 8.3.

TABLE 8.3 BUILT-IN RESPONSES

Simulation Type	Built-In Responses
DC	DC currents at the bias ports linearized small-signal parameters
small-signal	*S* parameters max available gain and stability factor insertion loss
large-signal	output power spectra input and output voltage spectra input and output voltage waveforms input and output current spectra input and output current waveforms

The simulation type of a sweep set is implied by the sweep labels defined in that sweep set, as shown in Table 8.2.

The built-in circuit responses are automatically calculated during the circuit simulation and made available for display. You do not need to explicitly specify built-in circuit responses in the Sweep block.

Example:

```
VG: -1.7  VD: 4  FREQ: From 2GHZ to 20GHZ Step=1GHZ;
```

This defines a sweep set for small-signal simulation. Although no response keyword is explicitly mentioned, all the small-signal built-in responses (*S* parameters, stability factor, maximum available gain and insertion loss) will be automatically calculated and available for display.

User-Defined Responses

In addition to the built-in responses, arbitrary user-defined responses can be created using expressions.

User-defined responses should be defined as labels in the Expression block.

Example:

```
PS11_in_Radian:  PS11 * PI / 180;
```

where PS11 is a predefined label representing a built-in response, namely the phase of S_{11} in degrees.

Details of the Expression block, including predefined labels, algebraic operators and mathematical functions are described in Chapter 4.

As a special case, you can explicitly specify predefined labels as user-defined responses. This may seem redundant since the predefined labels represent the built-in responses which are automatically included in the display. But it can be useful if you wish to see a built-in response displayed in a way that is different from the default format.

For example, by default, all the spectral components of the output power are displayed in one graph. If you wish to see the individual harmonics displayed as separate plots, you can explicitly specify the predefined labels POUT1, POUT2, etc.

Another example is given in Section 8.12, where the predefined labels representing the S parameters are explicitly specified in order to display the S parameters as rectangular plots, in addition to the built-in displays of Smith Chart and polar plots.

Simulation Types of User-Defined Responses

The simulation type of a user-defined response is determined by the built-in responses involved in its definition.

Example:

```
Power_Added_Efficiency = 100 * (POUTW1 - PINW) / (ID0 * VD);
ID_in_mA:  ID * 1000;
PS11_in_Radian:  PS11 * PI / 180;
```

where Power_Added_Efficiency is a user-defined large-signal response, ID_in_mA is a user-defined DC response and PS11_in_Radian is a user-defined small-signal response.

To ensure a unique simulation type, all the predefined labels involved in a user-defined response must have the same simulation type. For example,

```
Label_1: POUT1 + MS11;
```

This is not permissible since the definition of Label_1 attempts to mix a large-signal response POUT1 with a small-signal response MS11.

This rule also applies when user-defined responses are used in expressions to define other labels. In other words, a user-defined response inherits the simulation type of the built-in responses it refers to, and all the built-in and user-defined responses involved in an expression must have the same simulation type.

User-defined labels which do not depend on any circuit responses are "type-less" and therefore free from the restriction regarding simulation types.

User-Defined Responses Must Be Explicitly Requested

For user-defined responses to be calculated and displayed, they must be explicitly requested in the Sweep block.

Example:

```
VG=-1.7 VD=4 FREQ=6GHZ  PIN: 5DBM 10DBM   Power_Added_Efficiency;
```

This defines a sweep set for large-signal simulation and requests the calculation of the user-defined response Power_Added_Efficiency (which must have been defined in the Expression block).

Another example:

```
VG: -0.7  VD: From 0V to 4V Step=0.5V   ID_in_mA;
```

where ID_in_mA is a user-defined response to be calculated in this DC sweep set.

All User-defined responses included in a sweep set must have the same simulation type as that of the sweep set.

Limit on the Number of User-Defined Responses

Up to 16 user-defined responses can be specified *per sweep set.*

If necessary, you can define several sweep sets with identical ranges and request a different set of user-defined responses for each sweep set. For example,

```
VG=-1.7  VD=4  FREQ=6GHZ  PIN=5DBM   Label1 ... Label16;
VG=-1.7  VD=4  FREQ=6GHZ  PIN=5DBM   Label17 ... Label32;
```

This example achieves the same results as if a single sweep set were defined with 32 user-defined responses.

8.6 HarPE.Display Menu Option

The HarPE.Display menu option invokes circuit simulation according to the sweep ranges defined in the Sweep block and then presents you with a menu of different display formats and options.

☞ see Chapter 1 for a general discussion on menus and menu operations.

The steps needed for simulation and display are outlined as follows.

- ▷ Prepare an appropriate circuit file using the file editor (Chapter 2).
- ▷ Parse the circuit file by pressing the <F7> editor function key.
- ▷ Invoke the menu option HarPE.Display (by moving the cursor to highlight the option and pressing <ENTER>, see Chapter 1).
- ▷ Select from the menu of available display options.

Simulation

When you invoke the HarPE.Display menu option, HarPE first performs the necessary simulation to calculate the responses according to the Sweep block.

During simulation, HarPE displays a bar graphics indicator of the progress in the upper-left-hand corner of the screen, as well as one of the following messages:

```
HB Simulation ...  Press any key to interrupt
```

```
Small-Signal Simulation ...  Press any key to interrupt
```

```
DC Simulation ...  Press any key to interrupt
```

depending on the type of simulation in progress.

Interrupt the Simulation

You can interrupt the simulation at any time by pressing a key (any key). Then the following prompt will be displayed:

```
Terminate Operation (Y/<N>)?
```

and you can decide whether to terminate or to continue the operation.

HarPE Acceleration File

Simulation, especially nonlinear harmonic balance simulation, can be very time-consuming. An intelligent "HarPE acceleration file" feature is built into HarPE to avoid duplicate and unnecessary simulation.

Once the simulation is successfully completed, the results are saved in an encoded "HarPE acceleration file" (.haf file). This information can then be retrieved during subsequent sessions instead of repeating the simulation.

The creation and usage of HarPE acceleration files are transparent to the user. However, an explanation of the basic rules can be helpful.

A HarPE acceleration file will be created only if the circuit file has the extension .ckt. The HarPE acceleration file is named after the associated circuit file by replacing the extension .ckt with .haf. This rule is necessary to ensure a unique name for the .haf file.

For example, if you have two circuit files named example.ckt and example.old, then the HarPE acceleration file for example.ckt will be named example.haf, but example.old will not have a HarPE acceleration file. For this reason, it is strongly recommended that you name all the circuit files with the extension .ckt.

To ensure that the information stored is up to date, a "time stamp" is encoded into the .haf file which reflects the date and time of the corresponding circuit file. HarPE always checks this time stamp in a .haf file before using it. Therefore, any operation which changes the date and time of the circuit file after a .haf file is created will invalidate the contents of the .haf file.

Editors such as vi may modify the date and time of files. Even the cp (copy) command may change the date and time of the file copied.

A new HarPE acceleration file is created in three situations:

- A new circuit file is read, and simulated without editing or optimization.
- A circuit file is edited or optimized, then saved, and simulated.
- A circuit file is edited or optimized, then simulated, and saved.

Available Display Options

After the circuit simulation is successfully completed or the information stored in a .haf file from a previous simulation is retrieved, you will be at the HarPE.Display> menu level:

```
HarPE.Display> specTrum Waveform Psweep Fsweep DC S-par Error Xsweep
```

You can move the cursor to select the different display options from this menu.

The display options are summarized in Table 8.4.

TABLE 8.4 DISPLAY OPTIONS

Menu Option	Brief Description
HarPE.Display.specTrum	displays frequency-domain spectra
HarPE.Display.Waveform	displays time-domain waveforms
HarPE.Display.Psweep	displays power sweep (sweep fundamental input power)
HarPE.Display.Fsweep	displays frequency sweep (sweep fundamental frequency)
HarPE.Display.DC	displays DC IV curves and linearized model parameters
HarPE.Display.S-par	displays small-signal scattering parameters
HarPE.Display.Error	displays errors between model and measured responses
HarPE.Display.Xsweep	displays responses vs. arbitrary parameter sweep

For a given circuit file, the display options that are actually available depend on the definition of the Sweep block. For instance, if you have defined only one sweep set of the DC simulation type, then only the HarPE.Display.DC option is available (the menu option HarPE.Display.Xsweep may also be available if the sweep set includes a user-defined parameter sweep, see Section 8.3).

8.7 Spectrum Display

The menu option HarPE.Display.specTrum displays the frequency-domain spectra of the output power, input and output voltages and input and output currents of the circuit.

This option is available only if you have defined one or more large-signal sweep sets in the Sweep block in the circuit file.

When you select HarPE.Display.specTrum, a pop-up window will appear as

```
Output form:    Graphical
Zoom scale:     <TAB> to set scale
X-axis:         Frequency
Y-axis:         Pout & IDO
Sweep set:      1 of 3
Gate bias:      -1.7V
Drain bias:     4V
Frequency:      6GHZ
Input Power:    5DBM
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can move the cursor to select and change the various options.

The X-Axis and Y-Axis

The X-axis of the spectrum display is the harmonic frequency, from DC to the highest harmonic of the fundamental frequency.

The choices for the Y-axis are listed in Table 8.5.

TABLE 8.5 SPECTRUM DISPLAY Y-AXIS

Choice	Description
Pout & IDO	output power spectrum
Vin	input voltage spectrum
Vout	output voltage spectrum
Iin	input current spectrum
Iout	output current spectrum

A spectrum display is illustrated in Fig. 8.1.

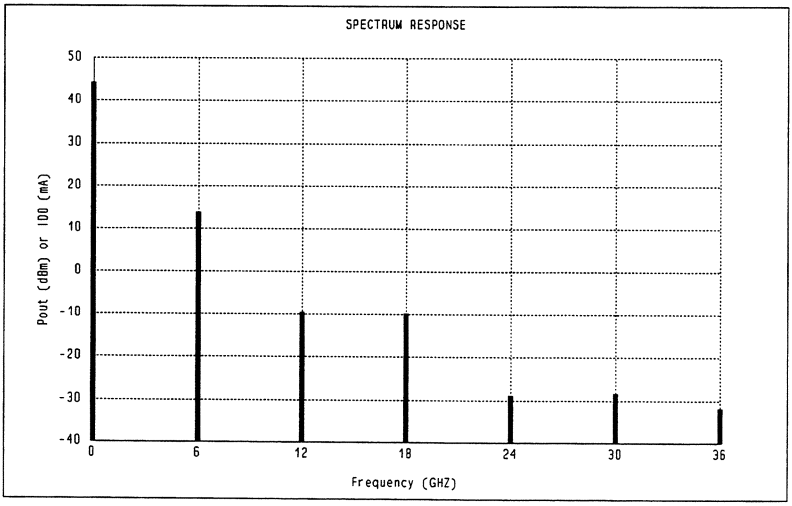


Fig. 8.1 Spectrum display.

Select Bias, Frequency and Input Power

The HarPE.Display.specTrum option displays the circuit response at a single point.

In the pop-up window, you can select the bias, fundamental frequency and input power values from those defined for the sweep set. If the Sweep block contains a number of large-signal sweep sets, any one of them can be selected for display.

Other Options

You can choose Numerical output instead of Graphical (see Section 8.15). You can also change the display scales using the Zoom scale option (see Section 8.16).

8.8 Waveform Display

The menu option HarPE.Display.Waveform displays the time-domain waveforms of the input and output voltages and currents of the circuit.

This option is available only if you have defined one or more large-signal sweep sets in the Sweep block in the circuit file.

When you select HarPE.Display.Waveform, a pop-up window will appear as

```
Output form:    Graphical
Zoom scale:     <TAB> to set scale
X-axis:         Time
Y-axis:         Vout_T
Sweep set:      1 of 3
Gate bias:      -1.7V
Drain bias:     4V
Frequency:      6GHZ
Input Power:    5DBM
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can move the cursor to select and change the various options.

The X-Axis and Y-Axis

The X-axis of the waveform display is time, with a width of two periods.

The choices for the Y-axis are listed in Table 8.6.

TABLE 8.6 WAVEFORM DISPLAY Y-AXIS

Choice	Description
Vin_T	input voltage waveform
Vout_T	output voltage waveform
Iin_T	input current waveform
Iout_T	output current waveform

A waveform display is illustrated in Fig. 8.2.

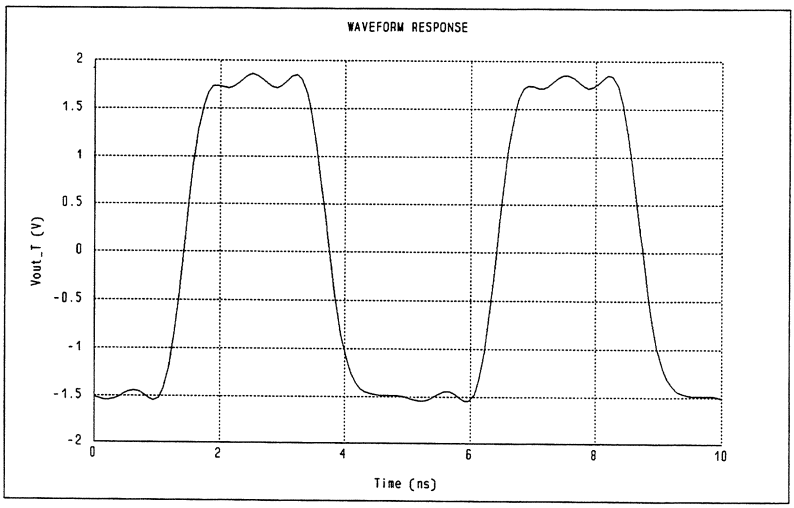


Fig. 8.2 Waveform display.

Select Bias, Frequency and Input Power

The HarPE.Display.Waveform option displays the circuit response at a single point.

In the pop-up window, you can select the bias, fundamental frequency and input power values from those defined for the sweep set. If the Sweep block contains a number of large-signal sweep sets, any one of them can be selected for display.

Display Scales

You can change the display scales using the Zoom scale option (see Section 8.16).

8.9 Power Sweep Display

The menu option HarPE.Display.Psweep displays the frequency-domain spectrum responses of the circuit versus the input power.

This option is available only if you have defined in the Sweep block one or more large-signal sweep sets which contain multiple input power values (power sweep).

When you select HarPE.Display.Psweep, a pop-up window will appear as

```
Output form:    Graphical
Zoom scale:     <TAB> to set scale
X-axis:         Input Power
Y-axis:         Pout & IDO
Sweep set:      1 of 3
Gate bias:      -1.7V
Drain bias:     4V
Frequency:      6GHZ
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can move the cursor to select and change the various options.

The X-Axis and Y-Axis

The X-axis of the power sweep display is the input power with the range defined in the sweep set.

The choices for the Y-axis are listed in Table 8.7.

TABLE 8.7 POWER SWEEP DISPLAY Y-AXIS

Choice	Description
`Pout & IDO`	output power spectrum
`Vin`	input voltage spectrum
`Vout`	output voltage spectrum
`Iin`	input current spectrum
`Iout`	output current spectrum
label	user-defined response identified by *label*
`All User Labels`	all user-defined responses

A power sweep display is illustrated in Fig. 8.3, where the multiple curves are shown using different line patterns. On the screen, they are displayed in the different colors.

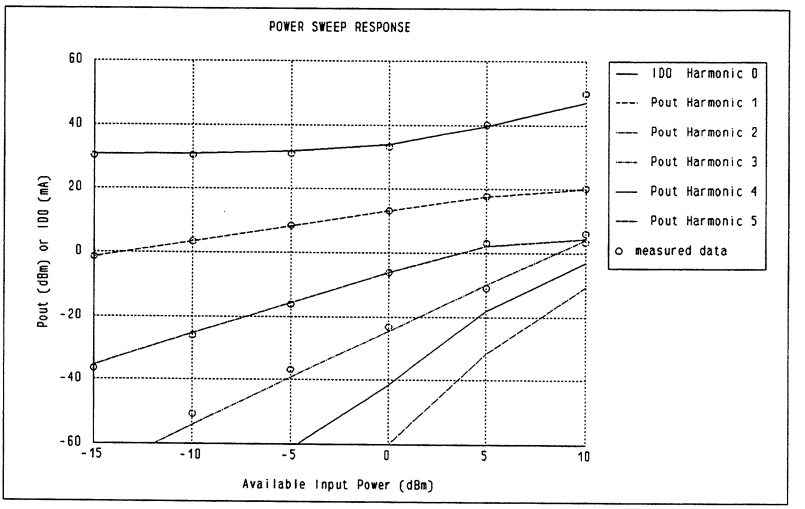


Fig. 8.3 Power sweep display.

Select Bias and Frequency

In the pop-up window for power sweep display, you can select the bias and fundamental frequency values from those defined for the sweep set. If the Sweep block contains a number of large-signal sweep sets with power sweep, any one of them can be selected for display.

Display Measured Data

If measured spectrum data corresponding to the selected responses (for the same bias, frequency and input power) are supplied in the Data block, they will be super-imposed on the display as circles, as illustrated in Fig. 8.3.

Other Options

You can choose Numerical output instead of Graphical (see Section 8.15). You can also change the display scales using the Zoom scale option (see Section 8.16).

Display User-Defined Responses

If one or more user-defined responses are included in the sweep set (Section 8.5), then you can select one or all of them for display.

Example:

```
Expression
   Power_Gain_1 = Pout1 - Pin;
   Power_Gain_2 = Pout2 - Pin;
   Power_Gain_3 = Pout3 - Pin;
End
...

Sweep
   VG=-1.7  VD=4  FREQ=6GHZ  PIN: from -15dBm to 10dBm step=5dBm
   Power_Gain_1  Power_Gain_2  Power_Gain_3;
End
```

Three user-defined responses, namely Power_Gain_1, Power_Gain_2 and Power_Gain_3, are included in the sweep set (with power sweep).

You can choose to display any one of them, as illustrated in Fig. 8.4, where Power_Gain_1 is displayed. You can also include all the user-defined responses in the same display by selecting the choice "All User Labels" for the Y-Axis option in the pop-up window.

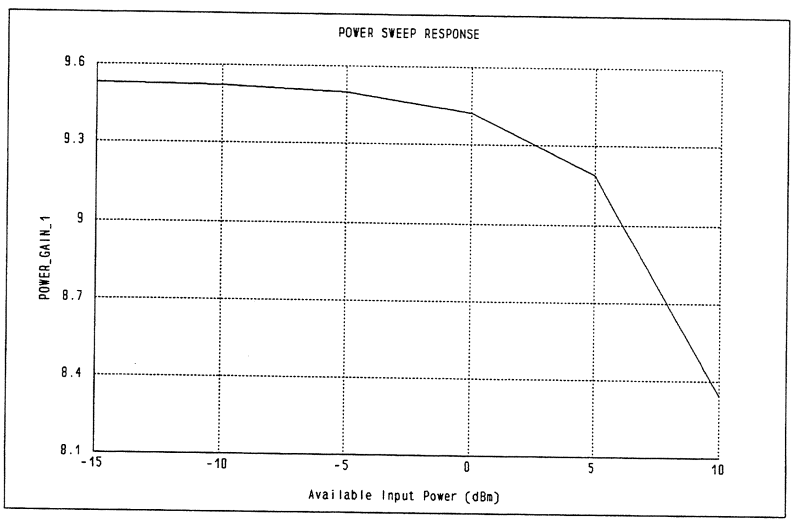


Fig. 8.4 Power sweep display of user-defined response.

8.10 Frequency Sweep Display

The menu option HarPE.Display.Fsweep displays the frequency-domain spectrum responses of the circuit versus the fundamental frequency.

This option is available only if you have defined in the Sweep block one or more large-signal sweep sets which contain multiple fundamental frequencies (frequency sweep).

When you select HarPE.Display.Fsweep, a pop-up window will appear as

```
Output form:    Graphical
Zoom scale:     <TAB> to set scale
X-axis:         Frequency
Y-axis:         Pout & ID0
Sweep set:      1 of 3
Gate bias:      -1.7V
Drain bias:     4V
Input Power:    5DBM
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can move the cursor to select and change the various options.

The X-Axis and Y-Axis

The X-axis of the frequency sweep display is the fundamental frequency with the range defined in the sweep set.

The choices for the Y-axis are listed in Table 8.8.

TABLE 8.8 FREQUENCY SWEEP DISPLAY Y-AXIS

Choice	Description
`Pout & ID0`	output power spectrum
`Vin`	input voltage spectrum
`Vout`	output voltage spectrum
`Iin`	input current spectrum
`Iout`	output current spectrum
label	user-defined response identified by *label*
`All User Labels`	all user-defined responses

A frequency sweep display is illustrated in Fig. 8.5, where the multiple curves are shown using different line patterns. On the screen, they are displayed in the different colors.

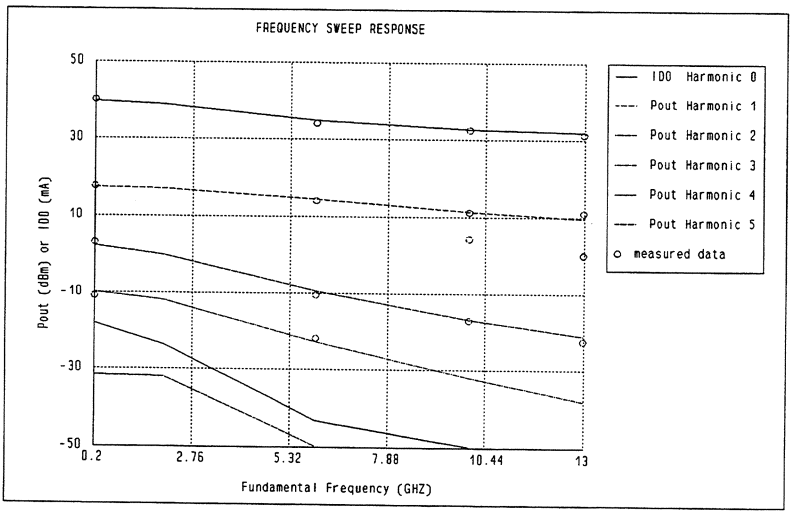


Fig. 8.5 Frequency sweep display.

Select Bias and Input Power

In the pop-up window for frequency sweep display, you can select the bias and input power values from those defined for the sweep set. If the Sweep block contains a number of large-signal sweep sets with frequency sweep, any one of them can be selected for display.

Display Measured Data

If measured spectrum data corresponding to the selected responses (for the same bias, frequency and input power) are supplied in the Data block, they will be super-imposed on the display as circles, as illustrated in Fig. 8.5.

Other Options

If one or more user-defined responses are included in the sweep set, you can select one or all of them for display (see the description for power sweep in Section 8.9).

You can choose Numerical output instead of Graphical (see Section 8.15). You can also change the display scales using the Zoom scale option (see Section 8.16).

8.11 DC Display

The menu option HarPE.Display.DC displays the DC responses of the circuit versus the bias voltages.

This option is available only if the Sweep block includes at least one DC sweep set.

When you select HarPE.Display.DC, a pop-up window will appear as

```
Output form:    Graphical
Zoom scale:     <TAB> to set scale
X-axis:         VD
Y-axis:         ID
Sweep set:      1 of 1
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can move the cursor to select and change the various options.

The X-Axis and Y-Axis

The X-axis of the DC display is a bias voltage. If the circuit contains two bias ports, you can select either VG or VD for the X-axis.

The choices for the Y-axis are listed in Table 8.9.

TABLE 8.9 DC DISPLAY Y-AXIS

Choice	Description
ID	drain (collector) current
IG	gate (base) current
VGS_DC	gate-source intrinsic voltage
VDS_DC	drain-source intrinsic voltage
parameter	linearized small-signal model parameters
label	user-defined response identified by *label*

If the Sweep block contains more than one DC sweep set, you can select one of DC sweep sets for display in the pop-up window.

A DC display is illustrated in Fig. 8.6, where the DC drain current is displayed versus the drain bias voltage at different gate bias voltages.

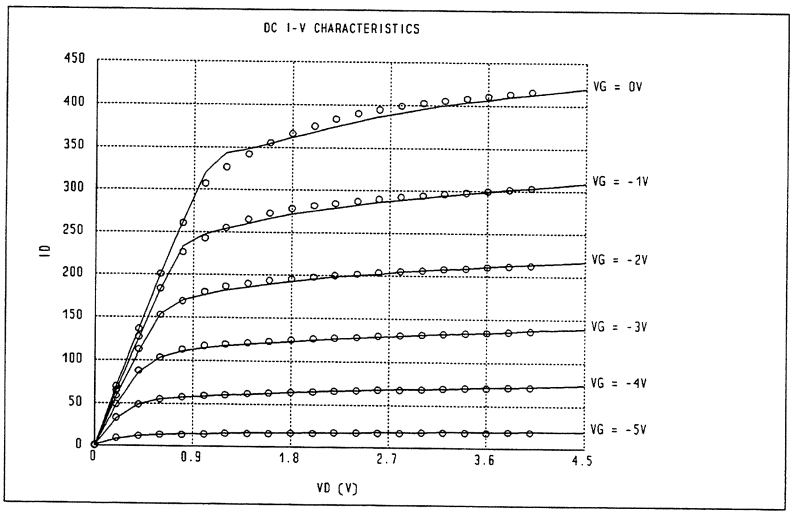


Fig. 8.6 DC display.

Display Measured Data

If measured DC data corresponding to the selected response for the same bias points are supplied in the Data block, it will be super-imposed on the display as circles, as illustrated in Fig. 8.6.

Other Options

You can choose Numerical output instead of Graphical (see Section 8.15). You can also change the display scales using the Zoom scale option (see Section 8.16).

Current-Driven Devices

For current-driven devices (i.e., the circuit includes a current bias source, see Chapter 5), the choices for the X-axis and the Y-axis are suitably modified.

Linearized Small-Signal Model Parameters

Linearized small-signal model parameters include linearized conductances, transconductances and capacitances. They are actually the derivatives of the model nonlinear currents and charges w.r.t. the intrinsic voltages evaluated at given bias points.

The linearized small-signal parameters available for the different nonlinear models are listed in Table 8.10.

TABLE 8.10 LINEARIZED PARAMETERS

Model	Available Linearized Parameters
FETC	Gm Cgs Cgd Gds Ggs Ggd
FETM	Gm Cgs Cgd Gds Ggs Ggd Gin
HEMTS	Gm Gds
NPN, PNP	GmF GmR
DIODE	Gj Cj

If the circuit contains one of the models listed in Table 8.10, then you can display the available linearized parameters using the HarPE.Display.DC option.

Display User-Defined Responses

If one or more user-defined responses are included in the sweep set (Section 8.5), then they can be selected for DC display, one at a time. You cannot display all the user-defined responses at the same time, because DC display may already contain multiple curves if both bias voltages are swept.

8.12 S-Parameter Display

The menu option HarPE.Display.S-par displays the S parameters and other small-signal responses of the circuit.

This option is available only if you have defined in the Sweep block one or more small-signal sweep sets.

When you select HarPE.Display.S-par, a pop-up window will appear as

```
Output form:    Graphical
Zoom scale:     <TAB> to set scale
Y-axis:         All S-pars
Sweep set:      1 of 2
Gate bias:      -1.7V
Drain bias:     4V
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can move the cursor to select and change the various options.

The Y-Axis

In the S-parameter display pop-up window, there is no explicit "X-axis" option. This is because that the S-parameters are displayed on the Smith Chart and polar plot instead of a rectangular plot. The underlying sweep parameter is always the frequency.

The choices for the Y-axis are listed in Table 8.11.

TABLE 8.11 S-PARAMETER DISPLAY Y-AXIS

Choice	Description
`All S-pars`	S_{11} and S_{22} on the Smith Chart, S_{21} and S_{12} on polar plots
`S11 & S22`	S_{11} and S_{22} on the Smith Chart
`S21, S12`	S_{21} or S_{12} on a polar plot
`Stability Factor`	stability factor (for two-ports), rectangular plot
`GMAX`	maximum available gain (for two-ports), rectangular plot
`Insertion Loss`	insertion loss (for two-ports), rectangular plot
label	user-defined response identified by *label*, rectangular plot
`All User Labels`	all user-defined responses, rectangular plot

The display illustrated in Fig. 8.7 includes a Smith Chart showing S_{11} and S_{22}, a polar plot showing S_{21} and a polar plot showing S_{12}.

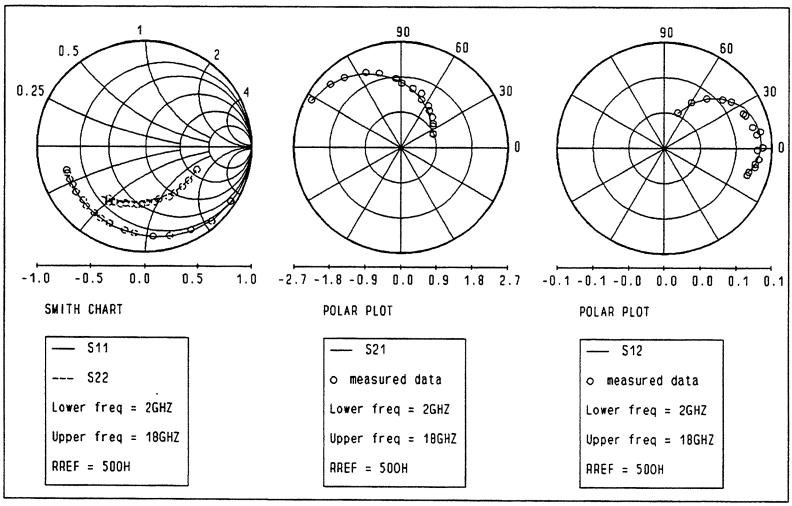


Fig. 8.7 S-parameter display.

You can also choose the Smith Chart and polar plots separately to display the individual S parameters.

If the circuit is defined as a two-port, a number of other small-signal responses are available as listed in Table 8.11.

Select the Bias Voltages

In the pop-up window for S-parameter display, you can select the bias voltages from those defined for the sweep set. If the Sweep block contains a number of small-signal sweep sets, any one of them can be selected for display.

Display Measured Data

If measured S parameters at the same bias and frequencies as those selected for display are supplied in the Data block, they will be super-imposed on the display as circles, as shown in Fig. 8.7.

Display User-Defined Responses

If one or more user-defined responses are included in the sweep set (Section 8.5), then you can select one or all of them for display.

Display S Parameters on Rectangular Plots

You can display the S parameters on a rectangular plot by explicitly specifying the response labels in a sweep set.

Example:

```
Sweep
   VG=-1.7  VD=4  FREQ: from 2GHZ to 18GHZ step=1GHZ  PS11 PS22;
   ...
End
```

where two predefined labels, namely PS11 and PS22, representing the phases of S_{11} and S_{22}, respectively, are explicitly included in the sweep set. This way, they are treated as user-defined responses. You can select to display either or both of them on a rectangular plot, as illustrated in Fig. 8.8, where both are displayed.

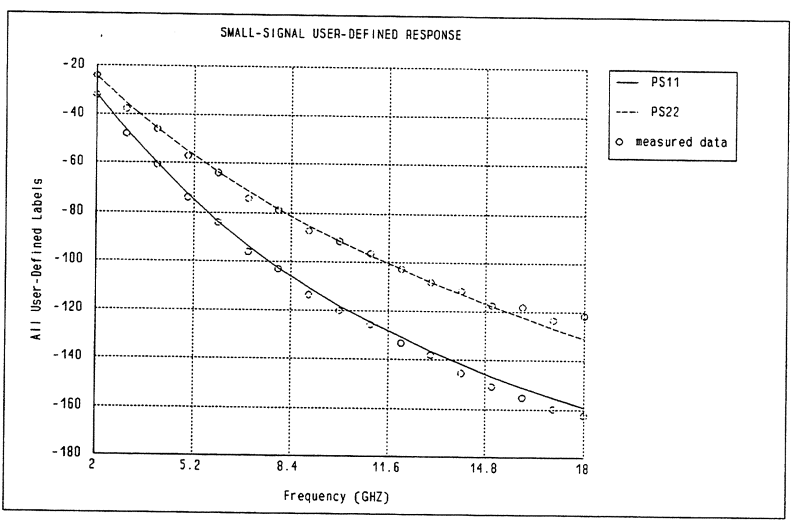


Fig. 8.8 Display S parameters on a rectangular plot.

8.13 Error Display

The menu option HarPE.Display.Error displays the discrepancies between simulated and measured large-signal power spectra, i.e., matching errors, for all the input power levels and fundamental frequencies defined in a sweep set, as illustrated in Fig. 8.9.

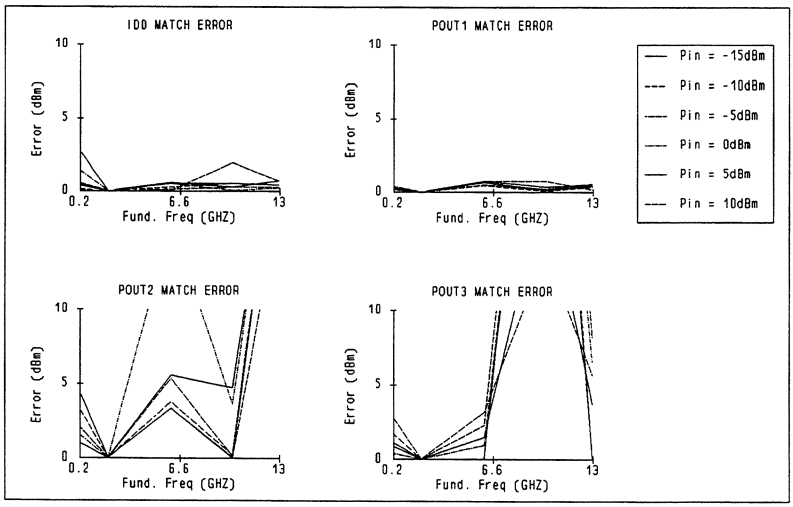


Fig. 8.9 Error display.

This option is available only if you have defined in the Sweep block one or more large-signal sweep sets which contain multiple fundamental frequencies (frequency sweep) and the corresponding power spectrum measurements are supplied in the Data block.

When you select HarPE.Display.Error, a pop-up window will appear to allow you to select a sweep set and a bias point from those available.

```
Output form:    Graphical
X-axis:         Frequency
Y-axis:         Pout & IDD
Sweep set:      1 of 3
Gate bias:      -1.7V
Drain bias:     4V
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

8.14 Xsweep Display

The menu option HarPE.Display.Xsweep displays the circuit responses versus a user-defined sweep label.

This option is available only if you have defined in the Sweep block one or more sweep sets which contain a user-defined sweep label (see Section 8.3).

Example:

```
Sweep
   IDSS: from 0.05 to 0.15 step=0.025  VG=-1.74  VD=4
   FREQ: from 2GHZ to 18GHZ step=1GHZ;
End
```

where the small-signal sweep set contains a user-defined sweep label IDSS.

When you select HarPE.Display.Xsweep, a pop-up window will appear as

```
Output form:    Graphical
Zoom scale:     <TAB> to set scale
X-axis:         IDSS
Y-axis:         MS11
Sweep set:      1 of 1
Gate bias:      -1.7V
Drain bias:     4V
Frequency:      2GHZ
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

The X-axis is the user-defined sweep label (IDSS in this example).

In general, the options in the pop-up window vary according to the simulation type of the sweep set (DC, small-signal or large-signal). For example, if the sweep set selected is a large-signal set, then one of the options is "input Power" and the choices for the Y-axis will be large-signal responses. If there are multiple sweep sets which contain user-defined sweep labels, then you can select any one of them for display.

An **Xsweep** display is illustrated in Fig. 8.10, where the user-defined sweep label IDSS in the preceding example represents the DC saturation current of the FET model, and the Y-axis is chosen to show the magnitudes of the small-signal *S* parameters.

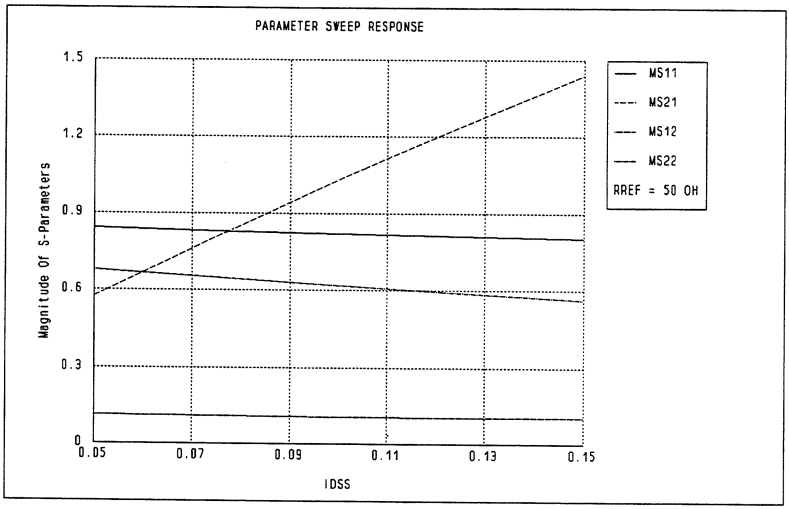


Fig. 8.10 Xsweep display.

In Fig. 8.10, the multiple curves are shown using different line patterns. On the screen, they are displayed in different colors.

You can choose **Numerical** output instead of **Graphical** (see Section 8.15). You can also change the display scales using the **Zoom scale** option (see Section 8.16).

8.15 Numerical Display

In addition to graphical display, many of the circuit responses can also be displayed numerically (i.e., as ASCII text).

Numerical outputs are available for spectrum, power sweep, frequency sweep, DC, *S* parameters and Xsweep (user-defined parameter sweep).

When you select numerical display through the "Output form" option in the pop-up window, the values of the circuit responses are printed in ASCII format, and the resulting text is loaded into HarPE's file editor. You can use all the editor features described in Chapter 2 to view, edit and save the numerical outputs.

OSA Wed May 12 12:00:00 1993
/harpe/display.dat

```
! Circuit File: /harpe/demo01_o.ckt    Wed May 12 12:00:00 1993
! Small-Signal S-Parameters
!
 PARAMETER VG=-1.74V VD=4V RREF=500H;
 FORMAT FREQ(GHZ) MS11 PS11(DEG) MS21 PS21(DEG) MS12 PS12(DEG) MS22 PS22(DEG);
     2.00  0.9698   -36.82  3.3098   150.24  0.0425    65.88  0.5707   -24.33
     3.00  0.9403   -53.40  3.0910   136.90  0.0593    55.35  0.5569   -34.85
     4.00  0.9088   -68.28  2.8457   124.74  0.0723    46.00  0.5436   -44.48
     5.00  0.8792   -81.45  2.6007   113.73  0.0820    37.82  0.5329   -53.22
     6.00  0.8535   -93.03  2.3710   103.77  0.0888    30.69  0.5260   -61.19
     7.00  0.8322  -103.21  2.1628    94.69  0.0934    24.45  0.5228   -68.50
     8.00  0.8150  -112.20  1.9774    86.35  0.0963    18.97  0.5233   -75.29
     9.00  0.8015  -120.18  1.8135    78.61  0.0979    14.11  0.5269   -81.65
    10.00  0.7910  -127.31  1.6688    71.38  0.0985     9.77  0.5331   -87.66
    11.00  0.7831  -133.73  1.5409    64.57  0.0983     5.88  0.5415   -93.36
    12.00  0.7772  -139.57  1.4274    58.11  0.0975     2.39  0.5516   -98.81
    13.00  0.7731  -144.90  1.3261    51.96  0.0961    -0.76  0.5631  -104.04
    14.00  0.7703  -149.81  1.2354    46.08  0.0943    -3.59  0.5756  -109.08
```

display.dat Insert <F1> help
HarPE:Display> S-par Ln 1 Pos 1

Fig. 8.11 Numerical Display.

The numerical output file is created in the editor's memory buffer. It is not a disk file unless you explicitly save it to the disk.

Numerical and Numerical Append

Two different choices of numerical display, namely Numerical and Numerical Append are available. If you choose Numerical, the current numerical outputs will overwrite any previous ones. The choice Numerical Append, on the other hand, will append the current numerical outputs to any previous ones. In other words, all numerical outputs are accumulated in one file with the current outputs at the end.

8.16 Graphics Zoom

By default, graphics displays are scaled to the value range of the responses to be displayed. You can change the display scale ("zooming") using the "Zoom scale" option in the pop-up window.

The Zoom scale option leads you to another pop-up window as

```
Zoom feature:       Enabled
Ymin of the plot:   0
yMax of the plot:   1.2
Xmin of the plot:   0
xmAx of the plot:   1.5
N x-axis ticks:     6
n y-axis Ticks:     6
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

The parameters "Ymin", "yMax", "Xmin" and "xmAx" define the corners of the display. The parameters "N x-axis ticks" and "n y-axis Ticks" specify the number of ticks (grids) on the X-axis and Y-axis, respectively.

The capitalized letter in the option names is not always the first character, because it must be unique as an identifier of the option (☞ Chapter 1).

You can turn the display upside down by specifying Ymin > yMax. You can also produce a mirror image of the display by specifying Xmin > xmAx.

8.17 HPGL and PostScript Files

HarPE can generate HPGL (Hewlett-Packard Graphics Language) and PostScript plotter files for capturing the graphical displays for documentation.

HPGL and PostScript files are ASCII files which can be directly sent to a compatible printer or plotter or imported into word processing programs.

To obtain an HPGL or PostScript file of the graphics currently displayed on the screen, press <Ctrl-P>. A pop-up window will appear:

```
Format:          HPGL for WORD
Name of file:    harpe.plt
With frame:      No
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

Selecting the File Format

The Format option in the pop-up window allows you to select from the available formats listed in Table 8.12. The differences between the HPGL formats are their interpretation of colors and font sizes.

To select a format, point the cursor to the option "Format" and use the <RIGHT> cursor key to toggle the choice.

TABLE 8.12 HPGL FILE FORMATS

Format	Description
HPGL for WORD	HPGL file to be imported into Microsoft Word
HPGL for WP	HPGL file to be imported into WordPerfect
HPGL 6-Pen Plotter	HPGL file for HP 6-pen plotter
PS Portrait	PostScript file, portrait orientation
PS Landscape	PostScript file, landscape orientation

Plotter File Name

The option "Name of file" in the pop-up window allows you to specify a name for the HPGL or PostScript file. The default file name is "harpe.plt".

To specify a file name, point the cursor to the option "Name of file" and type in the desired file name.

The file name can be arbitrary and the extension does not need to be ".plt".

Including or Excluding the Frame of the Graphical Display

On the screen, HarPE displays the graphics with a frame (white border lines). This frame can be included in or excluded from the HPGL or PostScript file generated.

You can include or exclude the frame by choosing "Yes" or "No" for the option "With frame". The default is "No", i.e., the frame is excluded from the plotter file.

Generating the Plotter File

Once you are satisfied with the options in the pop-up window, press <ENTER>. The pop-up window will disappear and the HPGL or PostScript file is generated.

HPGL for 6-Pen Plotter

The HPGL plotter files generated can be copied to the computer which is connected to an HP 6-pen plotter and then sent to the plotter.

If you wish the colors to closely match the colors displayed on the screen, you should load the plotter pen carousel in the following color sequence (pen 1 to pen 6): black, red, green, blue, yellow and pink.

HPGL for WordPerfect

The HPGL plotter files can be directly imported into WordPerfect. However, we recommend that you convert the HPGL file into the ".WPG" format using the graphcnv utility program supplied with WordPerfect:

```
graphcnv /W harpe.plt
```

where the option "/W" specifies single line width. This prevents irregular line widths which may occur if the HPGL files are directly imported into WordPerfect. An additional benefit is that the size of the converted ".WPG" file is smaller than the size of the HPGL file.

9

Optimization

9

Optimization

9.1 Overview

This chapter discusses the use of HarPE's powerful optimization capabilities for device modeling and optimization.

The following steps are needed in order to perform optimization:

- Define a set of optimization variables in the Expression and/or Model block (see Chapters 4 and 5).
- Define the circuit model in the Model block (see Chapter 5).
- For modeling, supply the measured data in the Data block (described in this chapter).
- Define the simulation ranges and optimization specifications in the Specification block (described in this chapter).
- Invoke the menu option HarPE.Optimize to start optimization.
- Save the optimized circuit file through the file editor (see Chapter 2).

This chapter describes the circuit file Data and Specification blocks as well as the menu options HarPE.Optimize and HarPE.Sensitivity.

The Data block allows you to supply measured data. In modeling, the circuit model is optimized to match the measurements (parameter extraction).

The Specification block allows you to define the simulation ranges (bias voltages, frequency and input power level), the responses to be optimized and weighting factors.

The HarPE.Optimize menu option allows you to select one of the four state-of-the-art gradient-based optimizers offered: ℓ_1, ℓ_2, minimax and Huber. You can also modify, if necessary, a number of parameters which control the optimization process.

A useful feature related to optimization is the parameter sensitivity display through the menu option OSA90.Sensitivity. It compares graphically the influence of different variables on the objective function to help you select the most influential variables for optimization.

Another advanced feature of HarPE is statistical modeling. This feature is described separately in Chapter 11.

9.2 Data Block

The Data block allows you to supply measured data for parameter extraction.

Syntax:

```
Data
   PARAMETER xxx xxx xxx
   FORMAT xxx xxx xxx xxx
      data data data ...
      ...
      data data data ...

   ...

   PARAMETER xxx xxx xxx
   FORMAT xxx xxx xxx xxx
      data data data ...
      ...
      data data data ...
End
```

Each data set consists of a PARAMETER statement, a FORMAT statement and the actual body of data.

You can supply separate data files and link them to the circuit file using the "#include" preprocessor directive (☞ see Chapter 3). Any part or parts of the Data block can be contained in one or more include files.

Keywords for Measured Data

The types of measurements acceptable to HarPE include power spectra, waveforms, *S* parameters and DC IV data. The data contained in each data set must be of the same type. The data types are summarized in Table 9.1.

TABLE 9.1 MEASUREMENT DATA TYPES

Data Type	Input Keywords	Response Keywords
DC	VG, VD	IG, ID
S parameters	VG, VD, FREQ, RREF	MS*ij*, PS*ij*, RS*ij*, IS*ij*
power spectrum	VG, VD, FREQ, PIN	IDO, POUT*k*
waveform	VG, VD, FREQ, PIN	VOUT_T*k*

In Table 9.1, the input keywords represent measurement conditions: bias, frequency, input power level and reference impedance. The response keywords represent measured responses. The keywords are further defined in Table 9.2.

TABLE 9.2 MEASUREMENT DATA KEYWORDS

Keyword	Default Unit	Description
FREQ	GHZ	frequency
PIN	DBM	input power
VD	V	drain (collector) bias voltage
VG	V	gate (base) bias voltage
IDO	A	DC component of drain (collector) current
POUT*k*	DBM	output power spectrum, the *k*th harmonic component
VOUT_T*k*	V	output voltage waveform, the *k*th time sample
ID	A	DC drain (collector) current
IG	A	DC gate (base) current
MS*ij*		magnitude of S_{ij}, $i,j = 1, n$
PS*ij*	DEG	phase of S_{ij}, $i,j = 1, n$
RS*ij*		real part of S_{ij}, $i,j = 1, n$
IS*ij*		imaginary part of S_{ij}, $i,j = 1, n$
RREF	OH	reference impedance for S parameters
NAME		character string identifier of the data set

- ▷ n is the number of ports (either 1 or 2).
- ▷ The default frequency unit is HZ for Non_Microwave_Units (see Chapter 3).

The PARAMETER Statement

The PARAMETER statement is optional and if used it must be the first statement in a data set. It is used to specify inputs (bias, frequency, power level, reference impedance) which remain constant within the data set.

Syntax:

PARAMETER *keyword=value ... keyword=value*;

where *keyword* represents an input keyword and *value* represents a constant.

Example:

```
PARAMETER  VG=-1.7V  VD=4V  RREF=1000H;
```

which specifies the measurement conditions for an S-parameter data set.

Only input keywords (see Table 9.1) can be specified in the PARAMETER statement. Response keywords cannot be used, because it is not logical for a response to remain constant throughout the data set.

The FORMAT Statement

Each data set must have a FORMAT statement which allows you to describe what data entries are being supplied as well as their order of appearance and units.

Syntax:

FORMAT *keyword(unit) ... keyword(unit);*

where *keyword* represents an input or response keyword with an optional *unit*.

Example:

```
FORMAT  PIN POUT1(DBM) POUT2(DBM) IDO(MA);
```

which indicates that the data entries are large-signal power spectrum measurements and they are entered in the order of PIN, POUT1, POUT2 and IDO. The units of some of the data entries are also specified, for instance, the unit of IDO is specified as MA which is different from the default unit of A. (See Chapter 3 for the list of units available).

By means of the FORMAT statement, HarPE avoids imposing a rigid order on data entry. For example, according to the following statement

```
FORMAT  MS11 MS12 MS21 MS22 FREQ(MHZ) PS11 PS12 PS21 PS22;
```

the data is entered in an unconventional order but it is perfectly acceptable.

Data can be entered one record per line, such as

```
FORMAT  PIN POUT1(DBM) POUT2(DBM) IDO(MA);
     -5.0     4.3      -20.4       30.5
      5.0    15.1        2.1       38.9
```

Or, the data may spread over a number of lines:

```
FORMAT  PIN POUT1(DBM) POUT2(DBM) IDO(MA);
        -5.0    4.3
               -20.4
                30.5
         5.0   15.1
                2.1
               38.9
```

Each data record must start on a new line.

The Completeness of Data

Each data set supplies one and only one type of data (DC data, *S* parameters, power spectra or waveforms). All the input keywords for that data type as listed in Table 9.1 must be specified to provide a complete description for the measurement conditions. Each input keyword can be specified in either the PARAMETER or the FORMAT statement.

Each data set must contain at least one measured response, i.e., the FORMAT statement must contain at least one response keyword.

The amount of data in each data set is arbitrary. HarPE will keep reading the data until it encounters a PARAMETER or FORMAT statement which marks the beginning of the next data set, or until the End of the Data block is reached.

The QUIET Option

An optional keyword QUIET can be used to instruct HarPE to ignore unrecognizable data entries.

Syntax:

```
PARAMETER  ...  QUIET  ... ;
FORMAT  ...  QUIET  ... ;
```

Example:

```
PARAMETER  QUIET  VG=-0.673  VD=4  FREQ=6GHZ  LENGTH=1.2;
FORMAT     PIN(DBM) POUT1(DBM) POUT2(DBM) ID0(MA) TEMP(C);
           +10.0     +18.1      -1.5       42.8     22.0
            +5.0     +13.9     -10.7       34.0     22.0
```

HarPE will ignore the parameter "LENGTH=1.2", the keyword "TEMP", and skip over the data entries corresponding to TEMP. Without the QUIET keyword, these unrecognizable entries will be treated as syntax errors.

The location of the QUIET keyword is important: it benefits only the entries *following* it but not those preceding it.

This user-friendly feature allows HarPE to share data files with other programs.

The QUIET keyword must be used with caution, because misspelled keywords following the QUIET keyword will not be detected as errors.

Super Keywords

To make data entry more convenient, a set of super keywords are defined to represent typical combinations of data keywords, as listed in Table 9.3.

TABLE 9.3 SUPER KEYWORDS FOR MEASUREMENT DATA

Super Keyword	Sequence of Keywords Represented
POUT*k*H	POUT1, POUT2, ..., POUT*k*
VOUT*k*T	VOUT_T1, VOUT_T2, ..., VOUT_T*k*
S1MP	MS11, PS11
S1RI	RS11, IS11
S2MP	MS11, PS11, MS21, PS21, MS12, PS12, MS22, PS22
S2RI	RS11, IS11, RS21, IS21, RS12, IS12, RS22, IS22

Example:

```
FORMAT  FREQ  S2MP;
```

which is equivalent to

```
FORMAT  FREQ  MS11 PS11 MS21 PS21 MS12 PS12 MS22 PS22;
```

Super keywords can be followed by a unit designator (see Chapter 3 for the list of unit designators). For example,

```
FORMAT  PIN  POUT4H(mW);
```

is equivalent to

```
FORMAT  PIN  POUT1(mW) POUT2(mW) POUT3(mW) POUT4(mW);
```

Cascade Microtech MicroCAT Data Format

Data from Cascade Microtech's MicroCAT Test Executive measurement system can be accepted directly by HarPE.

The following is an example of a MicroCAT data file:

```
NOTE MicroCAT sample data file.
NAME MicroCAT test run
DATE 31-JAN-89
NOTE This example shows measured S-parameters
SITE -5.000 0.000
SUBSITE 220 775
BIAS Vds=3.000
BIAS Ids=0.0210
BIAS Vgs=0.000
S2RITABLE 5
 5.0E9  0.1462 -0.8846 -1.2388 2.3593 0.0537  0.0513  0.6062 -0.3692
10.0E9 -0.3997 -0.7135  0.2803 1.6696 0.0898  0.0327  0.4051 -0.5222
15.0E9 -0.5872 -0.5211  0.7001 0.9781 0.0968  0.0125  0.2379 -0.6282
20.0E9 -0.6796 -0.3917  0.7564 0.4929 0.0968  0.0008  0.0750 -0.7027
25.0E9 -0.7203 -0.3161  0.6863 0.2040 0.0890 -0.0008 -0.0869 -0.7499
ENDTABLE
```

The information contained in a MicroCAT file which is actually picked up and utilized by HarPE includes the bias data, namely Vds, Vgs and Ids, and the S2RITABLE (*S* parameters, 2-port, Real and Imaginary data Table) which contains the frequency, RS11, IS11, RS21, IS21, RS12, IS12, RS22 and IS22.

MicroCAT data files can be directly included in the Data block.

Syntax:

`#include "`*`file_name`*`"  Source=Cascade`

where *file_name* is the name of the data file, the keyword Source indicates that the data is supplied in an external format, and the keyword Cascade represents the Cascade Microtech's MicroCAT data format.

MDIF Data Format

Measurement data in the MDIF (Measurement Data Interchange Format) format can be directly included into HarPE's circuit file.

The following is an example of data files in the MDIF format:

```
NOTE MDIF sample data file.
VAR Vds=3.000
VAR Ids=0.0210
VAR Vgs=0.000
BEGIN S2RITABLE
 5.0E9  0.1462 -0.8846 -1.2388 2.3593 0.0537  0.0513  0.6062 -0.3692
10.0E9 -0.3997 -0.7135  0.2803 1.6696 0.0898  0.0327  0.4051 -0.5222
15.0E9 -0.5872 -0.5211  0.7001 0.9781 0.0968  0.0125  0.2379 -0.6282
20.0E9 -0.6796 -0.3917  0.7564 0.4929 0.0968  0.0008  0.0750 -0.7027
25.0E9 -0.7203 -0.3161  0.6863 0.2040 0.0890 -0.0008 -0.0869 -0.7499
END
```

The information contained in the MDIF files which is actually picked up and utilized by HarPE includes the bias data, namely Vds, Vgs and Ids, and the S2RITABLE (*S* parameters, 2-port, Real and Imaginary data Table) which contains the frequency, RS11, IS11, RS21, IS21, RS12, IS12, RS22 and IS22.

MDIF data files can be directly included in the Data block.

Syntax:

`#include "`*file_name*`"  Source=MDIF`

where *file_name* is the name of the data file, the keyword Source indicates that the data is supplied in an external format, and the keyword MDIF identifies the format.

Data in EEsof Format

Data in EEsof format can be directly included into HarPE's circuit file.

The following is a typical example of a data file in EEsof format:

```
! NEC70000
! VDS=3V, IDS=30mA
# GHZ S MA R 50
! SCATTERING PARAMETERS:
2 .95 -26 3.57 157 .04 76 .66 -14
3 .93 -40 3.53 147 .05 69 .65 -20
4 .89 -52 3.23 136 .06 62 .63 -26
5 .86 -63 3.08 127 .07 56 .60 -31
6 .83 -71 2.98 121 .08 55 .59 -36
7 .80 -80 2.70 113 .09 50 .58 -40
8 .78 -87 2.58 108 .09 47 .57 -42
9 .75 -95 2.39 101 .09 44 .56 -46
10 .73 -102 2.27 96 .10 42 .54 -49
11 .68 -109 2.17 88 .10 41 .51 -52
12 .70 -116 2.11 84 .10 39 .48 -56
13 .70 -122 1.94 78 .10 37 .49 -64
14 .69 -126 1.79 77 .11 40 .51 -70
15 .68 -129 1.71 68 .11 36 .52 -74
16 .67 -130 1.62 68 .12 36 .53 -75
17 .66 -134 1.53 61 .12 38 .53 -76
18 .65 -136 1.51 57 .11 40 .54 -77
19 .64 -138 1.49 56 .11 40 .54 -78
22 .60 -144 1.30 40 .14 40 .56 -85
! NOISE PARAMETERS:
4    .7  .64  69   .38
8   1.2  .55  115  .20
12  2.0  .48  155  .20
18  2.7  .46  -33  .40
```

The information contained in the data of EEsof format which is actually picked up and utilized by HarPE includes the frequencies and S parameters. The noise parameters are ignored.

Data files in EEsof format can be directly included in the Data block.

Syntax:

#include "*file_name*"

where *file_name* is the name of the data file in EEsof format with the extension "*.snp*" where n ($1 \leq n \leq 9$) is the number of ports represented by the data.

9.3 Specification Block

The Specification block of the circuit file allows you to define the simulation ranges (bias voltages, frequency and input power level), the responses to be optimized, the goals or measured data to be matched, and weighting factors.

Syntax:

```
Specification
   specification_set;
   specification_set;
   ...

   specification_set;
End
```

where each *specification_set* is a statement delimited by a semicolon ";".

Syntax of specification sets:

```
sweep_label: sweep_range
...
sweep_label: sweep_range
response W=weight  ...
response = goal  W=weight  ...
response > goal  W=weight  ...
response < goal  W=weight  ...;
```

where each *sweep_label* represents a built-in sweep parameter,
sweep_range defines one or more values for the corresponding *sweep_label*,
response represents a built-in or user-defined response,
goal represents a constant or measured data,
and *weight* is an optional weighting factor.

The block name Specification can be abbreviated to Spec.

There are many similarities between the Sweep and Specification blocks. The Sweep block (see Chapter 8) controls circuit simulation for display whereas the Specification block applies to optimization.

Sweep Labels

Table 9.4 lists the set of built-in sweep labels (which is a subset of the predefined labels described in Chapter 4).

TABLE 9.4 BUILT-IN SWEEP LABELS

Label	Description
VD	drain (collector) bias voltage
VG	gate (base) bias voltage
FREQ	frequency or fundamental frequency
PIN	input power level

Note: VD is not applicable if only one bias port is defined.

Circuit Simulation Type

The type of circuit simulation required for calculating the responses to be optimized (large-signal, small-signal or DC) is implied by the sweep labels defined, as shown in Table 9.5.

TABLE 9.5 SWEEP LABELS AND SIMULATION TYPES

Sweep Labels	Implied Simulation Type
VG, VD	DC
VG, VD, FREQ	small-signal AC
VG, VD, FREQ, PIN	large-signal (harmonic balance)

Example:

```
Spec
   VG: from -3 to 0 step=0.5  VD: from 0 to 6 step=0.2  ID;

   VG: -1.74   VD: 3 4.5 5   FREQ: from 2GHZ to 18GHZ step=1GHZ
   MS11 PS11 MS21 PS21 MS12 PS12 MS22 PS22;

   VG: -2  VD: 4  FREQ: 6GHZ  PIN: from -15dBm to 5dBm step=5dBm
   POUT1 POUT2 POUT3 ID0;
End
```

DC simulation is assumed for the first specification set since it contains sweep ranges for the bias voltages VG and VD only. Similarly, the simulation types for the second and third specification sets are assumed to be small-signal AC and large-signal HB, respectively.

Define Discrete Values for Sweep Labels

Syntax:

sweep_label: *x1, x2, ... xn*

where *x1, x2, ..., xn* represent numerical values.

Example:

```
FREQ: 2, 3.2, 5.7, 13, 15
```

The comma separating two discrete values is optional.

Sweep an Interval with Uniform Step Size

Syntax:

sweep_label: **from** *x1* **to** *x2* **step=***x3*

where *x1, x2* and *x3* are numerical values. This translates into a set of values as: *x1, x1+x3, x1+2×x3, ..., x2.*

Example:

```
VD: from 0 to 6 step=0.2
```

This translates into a set of values for VD as 0, 0.2, 0.4, ..., 6.

The upper limit of the interval, namely *x2*, is always included in the translated values, even if the step size does not divide the interval evenly. For example,

```
VD:  from 1.5 to 2.75 step=0.5
```

The translated values are 1.5, 2, 2.5, 2.75, which include the upper limit 2.75.

You can specify a negative step size, such as

```
VD:  from 5 to 1.5 step=-0.5
```

The translated values are 5, 4.5, 4, ..., 1.5. In this case, the lower limit must be greater than the upper limit, i.e., $x1 > x2$.

The number of subintervals is $n = (x2 - x1) / x3$, if $x3$ divides $(x2 - x1)$ evenly, or else $n = (x2 - x1) / x3 + 1$, (the number of values is $n + 1$). The limit is $n \leq 1023$.

Specify the Number of Steps

Syntax:

sweep_label: **from** *x1* **to** *x2* **N=***n*

where *x1* and *x2* are numerical values, *n* is an integer, $1 \leq n \leq 1023$.
The step size is $(x2 - x1) / n$. The number of values is $n + 1$.

Example:

```
VD:  from 0 to 4 N=8
```

This translates into a step size of (4 - 0) / 8 = 0.5, and a set of values as 0, 0.5, 1, ..., 4. Note that the total number of values is 9.

Combine Intervals and Points

You can specify a combination of sweep intervals and discrete values for a sweep label. For example,

```
VD:  0  0.1  from 0.2 to 1 step=0.2  from 1.5 to 5 step=0.5
```

which translates into the following set of values:

```
0  0.1  0.2  0.4  0.6  0.8  1  1.5  2  2.5  3  3.5  4  4.5  5
```

This can be useful if you wish to have different step sizes in different sweep ranges. For example, an accurate representation of the circuit DC characteristics usually requires more densely spaced bias points below the "knee".

The set of values assigned to a sweep label will not be sorted by the file parser. During simulation, the values are assigned in the exact order as they appear in the circuit file. If the values need to be processed in a particular order (ascending, descending, etc.), they must be entered in the desired order.

Multiple Sweep Labels

When multiple sweep labels are defined in a specification set, the circuit simulation is performed at all the possible combinations of their values.

Example:

```
VG: 0, -1  VD: 1, 2;
```

For this specification set, DC simulation is performed at four bias points, namely

```
(VG, VD) = (0, 1), (0, 2), (-1, 1), (-1, 2)
```

If, however, you wish to consider only two points given by

```
(VG, VD) = (0, 1), (-1, 2)
```

then you should define two separate specification sets as

```
VG:  0  VD: 1;

VG: -1  VD: 2;
```

Keywords HARM: Highest Harmonic in Large-Signal Simulation

For large-signal specification sets, you can specify the highest harmonic to be included in harmonic balance simulation using the keyword HARM.

Syntax:

HARM=*m* ...;

m specifies the highest harmonic for the specification set, $1 \leq m \leq 8$.
The default is HARM = 4.

Example:

```
VG: -2  VD: 4  PIN: from -10DBM to 10DBM step=5DBM
FREQ: 6GHZ  HARM=5  ... ;
```

For this specification set, the first 5 harmonics and the DC component will be included in the harmonic balance simulation.

A higher value for HARM may improve the convergence of the Newton iteration for solving the nonlinear harmonic balance equations. A higher value for HARM can also be used to improve the accuracy of simulation, at the expense of a longer simulation time.

Keyword RREF: Reference Impedance for S Parameters

For small-signal AC specification sets, you can specify the reference impedance for *S* parameters using the keyword RREF.

Syntax:

RREF=*x* ...;

where *x* specifies the *S*-parameter reference impedance for the specification set, *x* must be a positive constant value, and the default is 50OH.

Example:

```
VG: -1.5 -2  VD: 4  FREQ: from 2GHZ to 18GHZ step=1GHZ
RREF=10  ... ;
```

where the reference impedance for the *S* parameters is specified as 10OH.

If the Specification Block Is Omitted

If you omit the SPECIFICATION block from the circuit file, then you must supply the Data block and the circuit model will be optimized to match *all* the measured data contained in the Data block.

9.4 Responses and Goals

Each specification set must contain at least one response for optimization. HarPE can optimize both built-in and user-defined circuit responses.

Built-In Circuit Responses

Corresponding to each type of simulation (DC, small-signal or large-signal) is a set of built-in circuit responses, as listed in Table 9.6.

TABLE 9.6 BUILT-IN RESPONSES

Simulation Type	Keyword	Description
DC	`ID`	DC drain (collector) current
	`IG`	DC gate (base) current
small-signal	`MS`*ij*	magnitude of S_{ij}, $i,j = 1, n$
	`PS`*ij*	phase of S_{ij}, $i,j = 1, n$
	`RS`*ij*	real part of S_{ij}, $i,j = 1, n$
	`IS`*ij*	imaginary part of S_{ij}, $i,j = 1, n$
	`GMAX`	max available gain
	`SK`	stability factor
	`INSL`	insertion loss
large-signal	`IDO`	DC component of drain (collector) current
	`POUT`*k*	output power magnitude in dBm, *k*th harmonic
	`POUTW`*k*	output power magnitude in Watts, *k*th harmonic
	`VOUT_T`*k*	output voltage waveform, *k*th time sample

- The simulation type of a specification set is implied by the sweep labels defined in that specification set, as shown in Table 9.5.
- n is the number of ports (either 1 or 2, see Chapter 5).
- If there is only one bias port, then `IG` is not available.

User-Defined Responses

In addition to the built-in responses, arbitrary user-defined responses can be created using expressions.

User-defined responses should be defined as labels in the Expression block.

Example:

```
PS11_in_Radian:  PS11 * PI / 180;
```

where PS11 is a predefined label representing a built-in response, namely the phase of S_{11} in degrees.

Details of the Expression block, including predefined labels, algebraic operators and mathematical functions are described in Chapter 4.

Simulation Types of User-Defined Responses

The simulation type of a user-defined response is determined by the built-in responses involved in its definition.

Example:

```
Power_Added_Efficiency = 100 * (POUTW1 - PINW) / (ID0 * VD);
ID_in_mA:  ID * 1000;
PS11_in_Radian:  PS11 * PI / 180;
```

where Power_Added_Efficiency is a user-defined large-signal response, ID_in_mA is a user-defined DC response and PS11_in_Radian is a user-defined small-signal response.

To ensure a unique simulation type, all the predefined labels involved in a user-defined response must have the same simulation type. For example,

```
Label_1: POUT1 + MS11;
```

This is not permissible since the definition of Label_1 attempts to mix a large-signal response POUT1 with a small-signal response MS11.

This rule also applies when user-defined responses are used in expressions to define other labels. In other words, a user-defined response inherits the simulation type of the built-in responses it refers to, and all the built-in and user-defined responses involved in an expression must have the same simulation type.

User-defined labels which do not depend on any circuit responses are "type-less" and therefore free from the restriction regarding simulation types.

Specifications and Goals

Single, upper and lower specifications are defined by

response = *goal*

response < *goal*

response > *goal*

respectively.

The goals can be constants or response keywords. When a goal is given by a response keyword, it means the *measured* response.

Example:

```
VG: -3 -1.7 0  VD: from 0 to 6 step=0.2  ID=ID;
```

This DC specification set instruct HarPE to match the simulated response ID to the measured response ID, within the range defined.

When measured responses are used as goals, the Data block must contain the needed measurements for all the points (bias, frequency, input power) within the sweep range defined in the specification set. For instance, in the preceding example, you must supply the measured ID data for all the bias points resulting from all possible combinations of VG = -3, -1.7, 0 and VD = 0, 0.2, 0.4, ..., 6.

Another example:

```
VG=-2  VD=4  FREQ: from 6GHZ to 12GHZ step=0.5GHZ
MS11 < 0.1  MS21_DB > 7  MS21_dB < 9;
```

This small-signal specification set appears to define a design optimization problem, where MS21_DB is a user-defined response defined in the Expression block as

```
MS21_DB = 20 * log10(MS21);
```

Goals Can Be Omitted for Parameter Extraction

In parameter extraction, the circuit model is optimized to match the measurements. In this case, the typical specification is

response = *response*

which means to match a simulated *response* to the corresponding measurement.

You can simplify the syntax for this type of specifications by omitting the equal sign and the goal.

Example:

```
VG: -1.74   VD: 3 4.5 5   FREQ: from 2GHZ to 18GHZ step=1GHZ
MS11 PS11 MS21 PS21 MS12 PS12 MS22 PS22;
```

which means that you wish to match the simulated *S* parameters to the measured ones. This is equivalent to

```
VG: -1.74   VD: 3 4.5 5   FREQ: from 2GHZ to 18GHZ step=1GHZ
MS11=MS11  PS11=PS11  MS21=MS21  PS21=PS21
MS12=MS12  PS12=PS12  MS22=MS22  PS22=PS22;
```

You can even omit the SPECIFICATION block completely, then *all* the measurements contained in the Data block will be used to extract the circuit model parameters through optimization.

Weighting Factors

Typically, a number of responses are to be optimized simultaneously. Weights can be used to balance the scales (orders of magnitude) of the different responses.

An optional weight can be assigned to each specification. For example:

```
VG: -1.74   VD: 3 4.5 5   FREQ: from 2GHZ to 18GHZ step=1GHZ
MS11 PS11  MS21 W=5  PS21  MS12 PS12  MS22 PS22;

VG: -3 -1.74 0  VD: from 0 to 5 step=0.5  ID W=20;
```

where the response MS21 is assigned a weight of 5 and the response ID is assigned a weight of 20.

A weight must be a positive constant.

Limits on the Number of Specifications

The maximum number of responses per specification set is limited to 12.

If necessary, you can define several specification sets with identical ranges and include a different set of responses for each specification set. For example,

```
VG=-1.7  VD=4  FREQ=6GHZ  PIN=5DBM   Response1 ... Response12;
VG=-1.7  VD=4  FREQ=6GHZ  PIN=5DBM   Response13 ... Response24;
```

This example achieves the same results as if a single specification set were defined with 24 responses.

9.5 Optimizers and Objective Functions

According to the specifications defined in the Specification block, error functions are created to measure the deviation of the responses from the goals, as summarized in Table 9.7.

TABLE 9.7 ERROR FUNCTIONS

Specification	Error Function
response W=*weight*	$F = weight \times \lvert response - measurement \rvert$
response = *goal* W=*weight*	$F = weight \times \lvert response - goal \rvert$
response < *goal* W=*weight*	$F = weight \times (response - goal)$
response > *goal* W=*weight*	$F = weight \times (goal - response)$

Error functions are formulated in such a way that a positive value of F would indicate that the specification is violated, and a zero or negative value of F would indicate that the specification is satisfied. In other words, error functions are expected to be minimized.

Objective Functions

Objective function is a scalar criterion by which an optimization algorithm compares one set of variables with another to decide which one is more desirable.

Given a set of error functions

$$F_1,\ F_2,\ \ldots,\ F_m$$

where F_j represents an error function derived from a specification at one of the sweep points and m represents the total number of error functions, different objective functions can be constructed.

Available in HarPE are the ℓ_1, ℓ_2, minimax and Huber objective functions. For each of these objective functions, HarPE employs a dedicated, state-of-the-art optimizer.

A few key references are listed at the end of this chapter.

L1 Objective Function

The ℓ_1 objective function is formulated as

$$U = \sum_{j=1}^{m} |F_j|$$

The ℓ_1 objective function is most suitable for parameter extraction and modeling in general, especially in the presence of measurement errors and uncertainties. It has been shown to have the unique ability to bypass isolated large errors and concentrate on error functions which are closer to zero (see, for example, references [1] and [3] at the end of this chapter).

Since the ℓ_1 objective function is a sum of absolute values, its minimization means that the individual error functions are driven towards a zero value. This is perfect for single specifications. But, if F_j is derived from an upper or lower specification, then a negative value of F_j would indicate that the specification is satisfied and therefore it should not be included in the ℓ_1 objective function.

If you have upper and lower specifications and choose the ℓ_1 objective function, then HarPE will automatically truncate any negative errors (from upper and lower specifications) to a zero value.

L2 Objective Function

The ℓ_2 objective function is the sum of squares:

$$U = \sum_{j=1}^{m} F_j^2$$

Another conventional definition uses the square root of the sum of squares, which is equivalent to the above formulation in the sense that both have the same minimizers (solutions).

The ℓ_2 objective function is widely used for its smooth convergence properties. One of the classical applications is to obtain unbiased estimates in approximation problems. However, in circuit modeling, the ℓ_2 objective function can be more susceptible to measurement errors than ℓ_1.

Since the ℓ_2 objective function is a sum of squares, its minimization means that the individual error functions are driven towards a zero value. This is perfect for single specifications. But, if F_j is derived from an upper or lower specification, then a negative value of F_j would indicate that the specification is satisfied and therefore it should not be included in the ℓ_2 objective function.

If you have upper and lower specifications, the generalized ℓ_2 objective function is more suitable. If you choose the ℓ_2 objective function, HarPE will automatically truncate any negative error function values (from upper and lower specifications) to a zero value.

Minimax Objective Function

The minimax objective is formulated as

$$U = \max_j \{ F_j \}$$

The minimax objective function represents the worst case among the error functions. Its minimization typically leads to an equal-ripple solution, i.e., at the solution there are several worst-case error functions which are equal in value to the objective function.

The minimax objective function is most suitable for design optimization where all the performance specifications are equally important. A good example is the design of filters where the loss must be uniformly minimized across the passband.

The minimax objective function is the least suitable for optimizing model responses to match measured data, especially when significant residual errors and/or measurement errors are expected. If used in such cases, the optimization can be dominated by a few error functions and easily trapped in local minima (false solutions).

Huber Objective Function

The Huber objective function is defined as [4,5]

$$U = \sum_{j=1}^{m} \rho_k(F_j)$$

where

$$\rho_k(F) = \begin{cases} F^2/2 & if\ |F| \le k \\ k|F| - k^2/2 & if\ |F| > k \end{cases}$$

where k is a positive constant threshold value.

The Huber function ρ_k is a hybrid of the least-squares (ℓ_2) (when $|F| \le k$) and the ℓ_1 (when $|F| > k$) functions.

The Huber function is robust against errors above the threshold (i.e., $|F| > k$), since those errors are treated in the ℓ_1 sense. The choice of k defines the threshold between "large" and "small" errors. Varying k alters the proportion of error functions to be treated in the ℓ_1 or ℓ_2 sense.

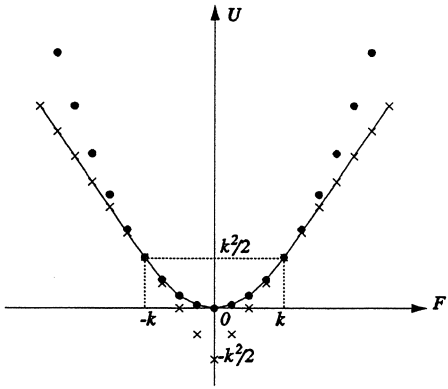


Fig. 9.1 *Comparison of the Huber, ℓ_1 and ℓ_2 objective functions in the one-dimensional case. The continuous curve, the strikes and the dots illustrate the Huber, ℓ_1 and ℓ_2 objective functions, respectively.*

9.6 HarPE.Optimize Menu Option

Once an optimization problem is properly formulated in the circuit file, you can start the optimization process using the HarPE.Optimize menu option (☞ see Chapter 1 for a general discussion on menus and menu operations).

When you invoke HarPE.Optimize, a pop-up window will appear:

```
Optimizer:                L1
Data type:                Small-Signal/DC
Number of iterations:     30
Accuracy of solution:     0.0001
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

The options in the pop-up window are summarized in Table 9.8.

TABLE 9.8 OPTIONS FOR HarPE.Optimize

Option	Available Choices
Optimizer	L1, L2, Minimax, Huber
Data type	Abstract, Large-Signal, Small-Signal/DC, All Data
Number of iterations	1, 2, 5, 10, 15, 30, 50, 100, 999
Accuracy of solution	0.01, 0.001, 1.0E-4, 1.0E-5
Threshold	threshold value for Huber objective functions

Optimizer

The choices available for the Optimizer option in the pop-up window are L1, L2, Minimax and Huber. The choice of optimizer also implies the use of the corresponding ℓ_1, ℓ_2, minimax or Huber objective function (see Section 9.5).

Data Type

The choices available for the Data type option in the pop-up window depend on the definition of the Specification block of the circuit file:

- If none of the functions and goals for optimization as defined in the Specification block involves any circuit response, then Abstract is the only choice for Data type. In this case, all the functions to be optimized are defined by generic expressions and the optimization requires no circuit simulation.
- If the Specification block contains only large-signal circuit responses for optimization, then Large-Signal is the only choice for Data type.
- If the Specification block contains only small-signal and/or DC circuit responses for optimization, then Small-Signal/DC is the only choice for Data type.
- If the Specification block contain both large-signal and small-signal/DC responses, then the Data type option has three choices: Large-Signal, Small-Signal/DC and All Data. You can choose to optimize the large-signal and small-signal/DC responses separately or simultaneously.

Number of Iterations

This option in the pop-up window allows you to limit the maximum number of iterations. The optimizer will stop once this limit is reached even if a solution within the desired accuracy has not been found. If this happens, the best set of values of the variables (in terms of minimizing the objective function) is presented as the solution.

Accuracy of Solution

This option allows you to specify the desired accuracy of the solution. The optimization will stop when the step size suggested by the algorithm is smaller than the specified accuracy. The step size is relative to the norm of the variable vector.

Interrupt the Optimization Process

During optimization, a bar graphics indicator of the progress as well as the following message are displayed in the upper-left-hand corner of the screen:

```
Optimization ... Press any key to interrupt
```

While the optimization is in progress, you can interrupt it at any time by pressing a key (any key) on the keyboard. Then the following prompt will be displayed:

```
Terminate Operation (Y/<N>)?
```

and you can decide whether to terminate or to continue the optimization process.

Display During Optimization

During optimization, HarPE displays the iteration count and current value of the ℓ_1, ℓ_2, minimax, or Huber objective function on the screen, such as

```
Small-Signal/DC L1 Optimization
Iteration   1/30  L1 Error=2.34101
Iteration   2/30  L1 Error=2.26455
Iteration   3/30  L1 Error=2.11538
Iteration   4/30  L1 Error=1.82175
Iteration   5/30  L1 Error=1.3006
Iteration   6/30  L1 Error=0.837255
Iteration   7/30  L1 Error=0.592952
Iteration   8/30  L1 Error=0.404969
Iteration   9/30  L1 Error=0.290885
Iteration  10/30  L1 Error=0.266347
Iteration  11/30  L1 Error=0.264805
Iteration  12/30  L1 Error=0.264814
Solution  L1 Error=0.264805
```

Of the iteration count, the first number is the current iteration and the second one is the maximum number of iterations specified. For example, 10/30 means that the current iteration is the 10th out of a maximum of 30.

It is sometimes possible to observe the objective function value getting larger than that of a previous iteration. This is not unusual. It is a part of the optimization algorithm.

Termination of Optimization

Optimization is terminated under one of the following conditions:

- ▷ Interrupted and terminated by the user.
- ▷ A local minimum of the objective function is found.
- ▷ The specified accuracy is reached, i.e., the step size suggested by the algorithm is smaller than the specified accuracy.
- ▷ The maximum number of optimization iterations is reached.
- ▷ A severe error condition is encountered such as a floating-point divide by zero error.

A message indicating the cause of termination will be displayed. Regardless of the cause the solution will always be the best result obtained up to the point of termination. The circuit file is automatically updated with the solution, i.e., the variables are updated with their optimized values.

The circuit file that is updated with the solution is actually the copy kept by HarPE in the computer's memory. The original disk file is not changed. If you wish to save the updated file to the disk, do so with the file editor (☞ Chapter 2).

9.7 HarPE.Sensitivity Menu Option

The HarPE.Sensitivity menu option provides sensitivity displays to assist you in selecting optimization variables. Large-change sensitivities of the ℓ_1, ℓ_2 and minimax objective functions are calculated and displayed.

☞ Objective functions are defined in Section 9.5.

Large-Change Sensitivities

The term "large-change" refers to changing (perturbing) the values of the optimization variables by a large (significant) amount.

More specifically, all the variables defined in the circuit file are perturbed by -10%, -5%, 5% and 10% of their nominal values. One variable is perturbed at a time, while the other variables are kept constant at their nominal values. The total number of perturbed points is 4 times the number of variables.

After each perturbation is made, the ℓ_1, ℓ_2 and minimax objective functions are evaluated and the changes with respect to the corresponding objective function values at the nominal point are recorded.

Simulation for Computing the Sensitivities

When you invoke the HarPE.Sensitivity menu option, the program will perform the necessary simulation to compute the sensitivities.

During simulation, a bar graphics indicator of the progress as well as the following message are displayed in the upper-left-hand corner of the screen:

```
Sensitivity Analysis ...  Press any key to interrupt
```

While the sensitivity analysis is in progress, you can interrupt it at any time by pressing a key (any key) on the keyboard. Then the following prompt will be displayed:

```
Terminate Operation (Y/<N>)?
```

and you can decide whether to terminate or to continue the operation.

Sensitivity Display Options

When the sensitivity analysis is completed, a pop-up window will appear:

```
Zoom scale:     <TAB> to set scale
Optimizer:      L1
Data type:      Small-Signal/DC
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can select L1, L2 or Minimax for the Optimizer option, and the sensitivity with respect to the corresponding objective function will be displayed.

The choices for the Data type option may be Abstract, Small-Signal/DC, Large-Signal and All Data, depending on the definition of the Specification block (☞ see Section 9.6 for further details on the Data type option).

The "Zoom scale" option allows you to change the display scale by specifying the parameters Ymin, yMax, Xmin and xmAx (☞ see Chapter 8 for further details on graphics zoom).

A sensitivity display is shown in Fig. 9.2 as an illustration.

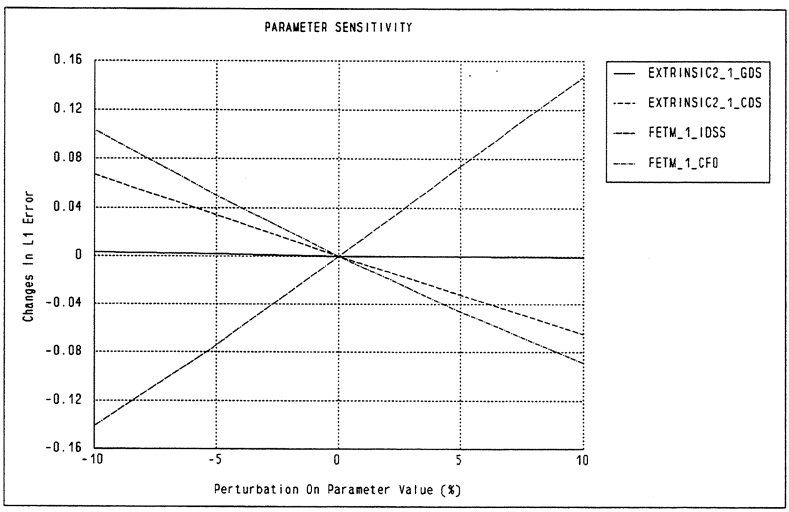


Fig. 9.2 Sensitivity display.

The X-axis of the display is marked by the numbers -10, -5, 0, 5 and 10, representing the percentage perturbations on the variables.

The Y-axis of the display represents the changes in the selected objective function with respect to the nominal (unperturbed) point. A negative value means that the objective function value decreases from the nominal point to the perturbed point. A positive value means an increase.

For example, a point at x = 10 and y = -0.2 would indicate that the value of the selected objective function decreases by 0.2 when the value of the variable increases by 10% from its nominal value.

Names of the Variables

The sensitivities with respect to different variables are displayed as separate curves, as illustrated in Fig. 9.2. The name of variable associated with each curve is shown in a legend.

Variables defined in the Expression block are identified by the label names. For example,

```
Expression
   X1: ?0.25?;
   X2: ?5  50  100?;
   ...
End
```

These two variables are identified by the label names X1 and X2.

Circuit parameters that are directly defined as optimization variables in the Model block are not labelled. The name for such a variable is created using the following template:

element_k_parameter

where *element* identifies the element model which contains the variable, *k* is an index which counts the number of occurrences of the same element, and *parameter* identifies the parameter which is defined as the variable.

Example:

```
FETM  1 2 3  IDSS=?0.2?  CF0=?0.023PF?  ...;
```

The two variables will be named FETM_1_IDSS and FETM_1_CF0, indicating that the element is FETM, the first occurrence of FETM, and the parameters are IDSS and CF0, respectively.

In Fig. 9.2, the four variables are identified as EXTRINSIC2_1_GDS, EXTRINSIC2_1_CDS, FETM_1_IDSS and FETM_1_CF0, respectively, which indicate that the variables are defined for the parameters GDS and CDS of the element EXTRINSIC2, and the parameters IDSS and CF0 of the element FETM.

If an element is used more than once in the circuit model, then the index k identifies which one of them contains the variable. For example,

```
Model
   RES  1 2  R: ?50?;
   ...
   RES  5 6  R: ?2KOH?;
   ...
   RES  8 9  R: ?50?;
   ...
End
```

The variables will be named as RES_1_R, RES_2_R and RES_3_R, which associate them with the first, second and third RES elements, respectively, in the order of appearance.

The number of different variables that can be accommodated in a single display is limited to 6 to keep the display legible. If necessary, HarPE will automatically organize the sensitivities into multiple displays, each showing a subset of the variables.

Use Sensitivity in Selecting Variables

The sensitivity display reveals not only which direction (increase or decrease) the objective function would change when a variable is perturbed, but also the magnitude of the change. By comparing the sensitivities with respect to different variables, you can estimate the relative influence that each variable has on the objective function value.

This information can be utilized to help determine which parameters should be selected as optimization variables.

First, you define all the parameters which are *potentially optimizable* as variables (by enclosing them within question marks). Then, by examining the sensitivity display you can select only the most influential parameters as variables for optimization.

For example, Fig. 9.3 shows that among the four variables considered, FETM_1_IDSS has the most significant influence on the ℓ_1 objective function and EXTRINSIC2_1_GDS has the least significant influence.

After you have optimized the circuit using a set of variables selected according to the sensitivity analysis, if the optimized solution is substantially different from the starting point, you should repeat the sensitivity analysis and augment the set of selected variables if necessary.

A proper selection of variables can be important. Having too many variables increases the computation time and may even cause the optimization problem to become ill-conditioned. On the other hand, failing to include some crucial parameters in optimization would lead to solutions that are not the best possible.

Verification of an Optimized Solution

The sensitivity display can also provide visual verification of an optimized solution. Theoretically, the sensitivity of the objective function should be zero at an optimized solution (a local minimum). Such a sensitivity display resembles a V, with the solution point at the bottom of the valley, as illustrated in Fig. 9.3.

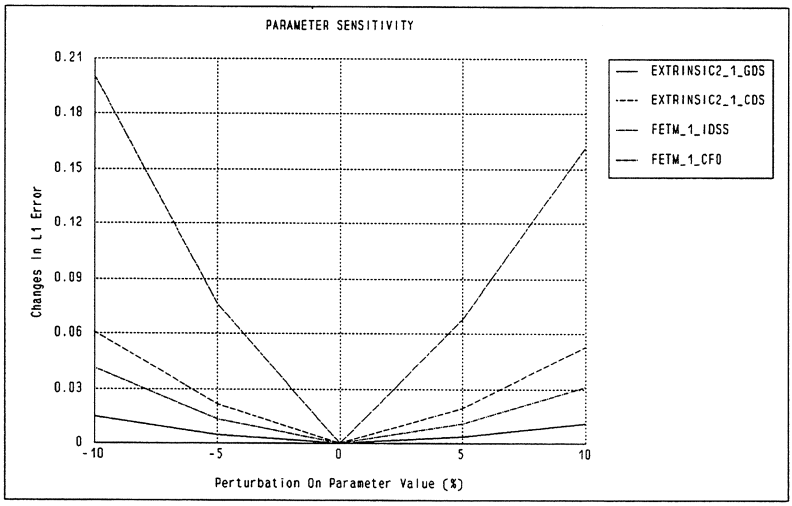


Fig. 9.3 Sensitivity display at an optimized solution.

The .sen Files

Sensitivity analyses can be time-consuming. To avoid duplicated analyses, a feature is built into HarPE to save the results of a sensitivity analysis in an encoded file, which can be retrieved and displayed in a subsequent session.

The encoded file is named after the circuit file by replacing the extension .ckt with .sen. To ensure that the .sen file is uniquely associated with the circuit file, a .sen file will be created only if the circuit file has the extension .ckt.

For example, if you have two circuit files named example.ckt and example.old, then the sensitivity analysis results for example.ckt will be saved in example.sen, but example.old will not have a .sen file.

Furthermore, to ensure that the information stored is up to date, a "time stamp" is encoded into the .sen file which reflects the date and time of the corresponding circuit file. HarPE always checks this time stamp in a .sen file before using it. Therefore, any operation which

changes the date and time of the circuit file after a .sen file is created will invalidate the stored sensitivity data.

Editors such as vi may modify the date and time of files. Even the cp (copy) command may change the date and time of the file copied.

A new .sen file is created in three situations:

- A new circuit file is read, and analyzed without editing or optimization.
- An circuit file is edited or optimized, then saved, and analyzed.
- An circuit file is edited or optimized, then analyzed, and saved.

Here "analyzed" refers to sensitivity analysis.

9.8 Preprocessing Cold Measurements

The Preprocess feature of HarPE allows you to extract parameters of the extrinsic circuits (EXTRINSIC1, EXTRINSIC2, EXTRINSIC3 and EXTRINSIC4) before employing optimization for extraction of intrinsic parameters. It is very useful in two-stage parameter extraction of FET and HEMT devices. First, the extrinsic parameters are analytically extracted from the cold measurements (VD = 0V). Then, the intrinsic parameters are extracted from the hot measurements while keeping the already extracted extrinsic parameters fixed. In our implementation we use the unbiased (VD = 0V and VG = 0V) and pinched-off (VD = 0V and VG < Vp, where Vp is the pinched-off voltage) measurements for preprocessing.

The following steps are needed in order to perform preprocess:

- Define an extrinsic circuit (EXTRINSIC1, EXTRINSIC2, EXTRINSIC3 or EXTRINSIC4) in the Model block (see Chapters 5 and 7).
- Define a FET or HEMT model for the device in the Model block (see Chapters 5 and 6).
- Define the circuit model in the Model block (see Chapter 5).
- Supply the cold (unbiased and pinched-off) measurement data in the Data block (see Section 9.2).
- Parse the circuit file and Invoke the menu option HarPE.Preprocess to extract the extrinsic parameters.
- Save the updated circuit file through the file editor (see Chapter 2).

The Extrinsic Circuit and Intrinsic Model

The extrinsic circuit and the intrinsic model must be defined in the Model block in order to perform preprocessing. The extrinsic circuit can be one of the four extrinsic circuits (EXTRINSIC1, EXTRINSIC2, EXTRINSIC3 or EXTRINSIC4) defined in Chapter 7.

The initial parameter values of the extrinsic circuit are not important since they will be updated after preprocessing. The user can simply provide the name and the connection nodes of the extrinsic circuit.

The intrinsic model must be one of the FET or HEMT models defined in Chapter 6.

The Cold Measurement Data

The cold measurement data used for preprocessing are S parameters measured under unbiased and pinched-off conditions. The frequencies in these two sets of S parameters must be consistent.

In order to proceed with preprocessing you must supply the unbiased and pinched-off measurement data in the Data block. The unbiased and pinched-off measurement data have to be indicated by the keywords UNBIASED and PINCHED, respectively, in the PARAMETER

statement. For example,

```
PARAMETER VG=0V VD=0V RREF=500H UNBIASED;
FORMAT FREQ(GHZ) MS11 PS11(DEG) MS21 PS21(DEG) MS12 PS12(DEG) MS22 PS22(DEG);
     2  0.9647   -48.34  0.0259  62.87  0.0259  62.87  0.6778  178.43
     4  0.9048   -86.00  0.0418  42.61  0.0418  42.61  0.6788  176.82
     6  0.8615  -112.66  0.0499  28.84  0.0499  28.84  0.6800  175.18
     8  0.8358  -131.82  0.0539  19.14  0.0539  19.14  0.6812  173.53
    10  0.8212  -146.28  0.0561  11.83  0.0561  11.83  0.6826  171.89
    12  0.8129  -157.74  0.0572   5.97  0.0572   5.97  0.6842  170.24
    14  0.8086  -167.22  0.0578   1.04  0.0578   1.04  0.6860  168.60
    16  0.8068  -175.33  0.0580  -3.26  0.0580  -3.26  0.6881  166.96
    18  0.8065   177.54  0.0580  -7.12  0.0580  -7.12  0.6904  165.33

 PARAMETER VG=-5V VD=0V RREF=500H PINCHED;
 FORMAT FREQ(GHZ) MS11 PS11(DEG) MS21 PS21(DEG) MS12 PS12(DEG) MS22 PS22(DEG);
     2  0.9907   -24.34  0.0421   71.28  0.0421   71.28  0.9907  -16.68
     4  0.9664   -47.43  0.0777   53.68  0.0777   53.68  0.9848  -32.88
     6  0.9355   -68.44  0.1036   37.94  0.1036   37.94  0.9766  -48.21
     8  0.9049   -87.09  0.1201   24.27  0.1201   24.27  0.9676  -62.47
    10  0.8786  -103.47  0.1289   12.59  0.1289   12.59  0.9588  -75.54
    12  0.8576  -117.85  0.1325    2.65  0.1325    2.65  0.9507  -87.44
    14  0.8417  -130.52  0.1327   -5.85  0.1327   -5.85  0.9437  -98.22
    16  0.8302  -141.77  0.1307  -13.15  0.1307  -13.15  0.9376  -107.99
    18  0.8223  -151.84  0.1276  -19.49  0.1276  -19.49  0.9326  -116.86
```

Please see Section 9.2 for details of the Data block.

HarPE.Preprocess Menu Option

The menu option HarPE.Preprocess preprocesses the cold measurement data supplied in the Data block and extracts the extrinsic parameters analytically.

After preprocessing, the program automatically returns to the file editor with the updated circuit file loaded. The extracted extrinsic parameters are placed just before the Model block and all of them are commended out.

To utilize the extracted extrinsic circuit, the user needs to replace the elements of the old extrinsic circuit by the extracted extrinsic elements.

9.9 Technical References

[1] J.W. Bandler and S.H. Chen, "Circuit optimization: the state of the art", *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 424-443.

[2] J.W. Bandler, W. Kellermann and K. Madsen, "A superlinearly convergent minimax algorithm for microwave circuit design", *IEEE Trans. Microwave Theory Tech.*, vol. MTT-33, 1985, pp. 1519-1530.

[3] J.W. Bandler, W. Kellermann and K. Madsen, "A nonlinear ℓ_1 optimization algorithm for design, modeling and diagnosis of networks", *IEEE Trans. Circuits and Systems*, vol. CAS-34, 1987, pp. 174-181.

[4] H. Ekblom and K. Madsen, "Algorithms for nonlinear Huber estimation", *BIT* 29, 1989, pp. 60-76.

[5] J.W. Bandler, S.H. Chen, R.M. Biernacki, L. Gao, K. Madsen and H. Yu, "Huber optimization of circuits: a robust approach," *IEEE Trans. Microwave Theory Tech.*, vol. 41, 1993, pp. 2279-2287.

[6] V.I. Cojocaru and T.J. Brazil, "A large-signal model for GaAs MESFET's and HEMT's valid at multiple DC bias-points," *Proc. European Gallium Arsenide Application Symp.* (Turin, Italy), 1994, pp. 419-422.

[7] M. Berroth and R. Bosch, "Broad-band determination of the FET small-signal equivalent circuit," *IEEE Trans. Microwave Theory Tech.*, vol. 38, 1990, pp. 891-895.

[8] G. Dambrine, A. Cappy, F. Heliodore and E. Playez, "A new method for determining the FET small-signal equivalent circuit," *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 1151-1159.

10

Monte Carlo Analysis

10

Monte Carlo Analysis

10.1 Overview

The Monte Carlo analysis features in HarPE are very useful for studying the statistical behaviour of devices and circuits in the presence of parameter tolerances and model uncertainties.

The statistical models that can be used include uniform, normal (Gaussian), lognormal, exponential and sample distributions. Parameter correlations can be accommodated in multidimensional normal distributions.

The HarPE.MonteCarlo menu option invokes Monte Carlo simulation to calculate the circuit responses at a number of random outcomes generated from a specified statistical model. The statistical responses that can be displayed include histograms, run charts, bias sweep, frequency sweep, input power sweep and cumulative probability distributions. Yield can be estimated with respect to user-defined specifications.

This chapter describes the syntax for defining statistical parameters in the Expression and Model blocks, defining correlation matrices in the Statistics block, supplying statistical sample in the Sample block, and defining simulation ranges, responses and specifications in the MonteCarlo block. A description of the HarPE.MonteCarlo menu option and the different types of statistical displays follows.

You obtain the statistical model needed for Monte Carlo analysis using HarPE's statistical modeling capability which is covered in Chapter 11.

10.2 Statistical Parameters

Labels in the circuit file Expression block can represent statistical parameters.

Syntax:

label_name: *nominal* { *distribution keyword=value ... keyword=value* };

Circuit model parameters in the Model block can also be defined as statistical parameters.

Syntax:

parameter: *nominal* { *distribution keyword=value ... keyword=value* };

The available distributions and keywords are listed in Table 10.1.

TABLE 10.1 STATISTICAL PARAMETER KEYWORDS

Distribution	Keywords
Uniform	TOL, HIGH, LOW
Exponent	TOL, HIGH, LOW
LogNormal	TOL, HIGH, LOW
Normal	SIGMA, HIGH, LOW, CORRELATION, DDF
Sample	*none*

Examples:

```
X1: 2.5pF {Uniform TOL=0.8pF};

RES  1 2  R=35 {Normal SIGMA=5%};
```

The label X1 represents a statistical parameter with a nominal value of 2.5pF and a uniform distribution with a tolerance of 0.8pF. In the second example, the parameter R of the element RES is defined as a statistical parameter with normal distribution. The standard deviation of R is defined as 5 percent of its mean value.

The details of the individual distributions are described in Sections 10.3 to 10.8.

10.3 Uniform Distribution

Syntax:

nominal {Uniform TOL=*tolerance* LOW=*low* HIGH=*high*}

where *nominal* represents the nominal value, *tolerance* represents the tolerance, and *low* and *high* specify the distribution interval (the default: *low* = -1 and *high* = 1).

The tolerance can be specified by an absolute value, such as

```
V1: 5 {Uniform TOL=0.5};
```

or as a percentage of the nominal value, such as

```
V1: 5 {Uniform TOL=10%};
```

Generation of Outcomes

First, a normalized (dimensionless) *deviate* is generated from a uniform distribution in the interval [*low high*] (the default interval is [-1 1]).

For an absolute tolerance, the outcome is given by

$$outcome = nominal + tolerance \times deviate$$

For a relative (percentage) tolerance, the outcome is given by

$$outcome = nominal \times (1 + tolerance \times deviate)$$

Example:

```
V1: 5 {Uniform TOL=2 LOW=-0.5 HIGH=0.7};
```

The interval of the normalized uniform distribution deviates is [-0.5 0.7], and the interval of the outcomes is [4 6.4].

10.4 Exponential Distribution

Syntax:

nominal {Exponent TOL=*tolerance* LOW=*low* HIGH=*high*}

where *nominal* represents the nominal value, *tolerance* represents the tolerance, and *low* and *high* specify the optional truncation limits (the default: *low* = -1 and *high* = 4).

The tolerance can be specified by an absolute value, such as

```
V1: 5 {Exponent TOL=0.5};
```

or as a percentage of the nominal value, such as

```
V1: 5 {Exponent TOL=10%};
```

Generation of Outcomes

First, a normalized (dimensionless) *deviate* is generated from an exponential distribution with a mean value of zero.

In text books, the exponential distribution is normalized to have a mean value of 1 and an interval of [0 ∞). In HarPE the exponential deviates are shifted to the interval [-1 ∞), so that the mean value of the deviates is zero.

Then, *deviate* is truncated to be within the limits [*low high*]. In other words, deviates generated outside the truncation limits are discarded. Truncation limits can significantly alter the distribution of the statistical outcomes.

For an absolute tolerance, the outcome is given by

$$outcome = nominal + tolerance \times deviate$$

For a relative (percentage) tolerance, the outcome is given by

$$outcome = nominal \times (1 + tolerance \times deviate)$$

10.5 Lognormal Distribution

Syntax:

nominal {Lognormal TOL=*tolerance* LOW=*low* HIGH=*high*}

where *nominal* represents the nominal value, *tolerance* represents the tolerance, and *low* and *high* specify the optional truncation limits (the default: *low* = -1.6 and *high* = 20).

The tolerance can be specified by an absolute value, such as

```
V1: 5 {Lognormal TOL=0.5};
```

or as a percentage of the nominal value, such as

```
V1: 5 {Lognormal TOL=10%};
```

Generation of Outcomes

First, a normalized (dimensionless) *deviate* is generated from a lognormal distribution with a mean value of zero.

In text books, the lognormal distribution is normalized to have a mean value of 1.6487213 ($e^{0.5}$) and an interval of (0 ∞). In HarPE the lognormal deviates are shifted to the interval (-1.6487213 ∞), so that the mean value of the deviates is zero.

Then, *deviate* is truncated to be within the limits [*low high*]. In other words, deviates generated outside the truncation limits are discarded. Truncation limits can significantly alter the distribution of the statistical outcomes.

For an absolute tolerance, the outcome is given by

$$outcome = nominal + tolerance \times deviate$$

For a relative (percentage) tolerance, the outcome is given by

$$outcome = nominal \times (1 + tolerance \times deviate)$$

10.6 Normal Distribution

This section focuses on normal (Gaussian) distribution without parameter correlations. Correlations are described in Section 10.7.

Syntax:

nominal {Normal SIGMA=σ LOW=*low* HIGH=*high* DDF=*k1 k2 ... kn*}

where *nominal* represents the nominal value, σ specifies the standard deviation, *low* and *high* are optional truncation limits (the default: *low* = -3 and *high* = 3), and *k1, k2, ..., kn* are optional integers representing a discrete density function.

The standard deviation can be specified by an absolute value, such as

```
V1: 5  {Normal SIGMA=0.25};
```

or as a percentage of the mean value, such as

```
V1: 5  {Normal SIGMA=5%};
```

Generation of Outcomes

First, a normalized (dimensionless) *deviate* is generated from a normal distribution with a zero mean value and unit standard deviation.

Then, *deviate* is truncated to be within the limits [*low high*]. In other words, deviates generated outside the truncation limits are discarded. Truncation limits can significantly alter the distribution of the statistical outcomes.

If the standard deviation σ is specified by an absolute value, the outcome is given by

$$outcome = nominal + \sigma \times deviate$$

If σ represents a relative (percentage) value, the outcome is given by

$$outcome = nominal \times (1 + \sigma \times deviate)$$

Discrete Density Function

The random deviates generated from a normal distribution can be mapped into a discrete distribution through a histogram specified by the keyword DDF. The histogram serves as a discrete density function which approximates a non-Gaussian distribution.

Using this approach, the density function derived from a non-Gaussian sample distribution can be reproduced reasonably well in the Monte Carlo analysis.

To obtain the discrete density function (DDF) from the histogram of a statistical sample, the sample outcomes are divided into *N* bins (subintervals), where *N* is typically between 10 and 20. The number of outcomes which fall into each bin is recorded, and the numeric record is used as the DDF.

Example:

```
V1:  5  {Normal SIGMA=10%  DDF=2 8 15 16 18 18 12 4 2 4};
```

where the discrete density function corresponds to a 10-bin histogram as shown in Fig. 10.1.

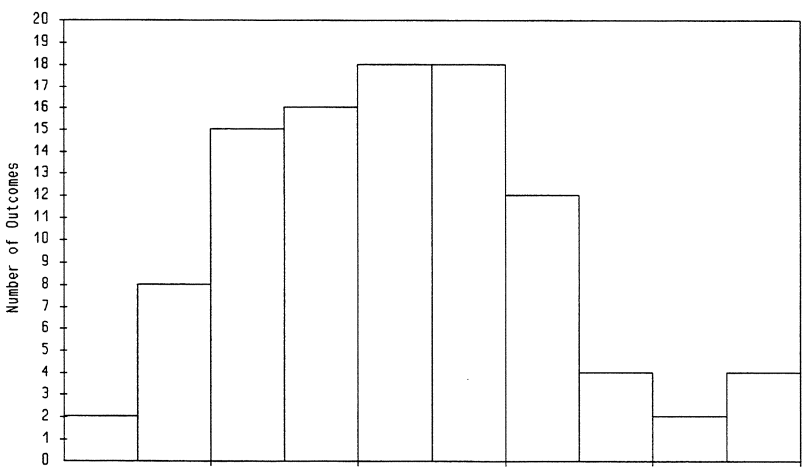


Fig. 10.1 Histogram used to specify the discrete density function.

If DDF is specified, the outcomes are mapped to the discrete density function through the respective cumulative distribution functions. Appropriate scaling and shifting may be performed to preserve the mean and standard deviation.

When you use HarPE for statistical modeling, DDFs can be automatically generated for some or all of the statistical parameters (see Chapter 11 for further details).

10.7 Statistics Block and Correlations

You can define correlations between statistical parameters with a normal distribution. First, you supply a correlation matrix in the Statistics block, and then link the matrix to a set of statistical parameters using the CORRELATION keyword.

Syntax:

```
Statistics
   CORRELATION: name  Dimension=n  Format=format;
      c11  c12  ...  c1n
      c21  c22  ...  c2n
      ...
      cn1  cn2  ...  cnn
End

Expression

   label: nominal (Normal SIGMA=σ CORRELATION=name[i] ... );
   ...

End
```

where *name* is a string identifying a correlation matrix defined in the Statistics block, *cij* is the correlation coefficient between two statistical parameters indexed *i* and *j*.

The values of the correlation coefficients must be between -1 and +1. The matrix must be symmetrical and positive semidefinite, with unit diagonal elements.

Correlation Matrix Formats

Correlation coefficients can be entered in one of the four available formats, namely, FULL, UPPER, LOWER and SPARSE.

The default is the FULL format, which means that the complete matrix is entered.

Example:

```
CORRELATION: Cor_Mat  Dimension=4  Format=FULL;
   1.0   0.2   0.4   0.6
   0.2   1.0   0.8   0.0
   0.4   0.8   1.0   0.1
   0.6   0.0   0.1   1.0
```

Syntax of the UPPER format

```
CORRELATION: name  Dimension=n  Format=UPPER;
      c11  c12  ...  c1n
           c22  ...  c2n
                ...
                     cnn
```

Only the upper-right triangular portion of the correlation matrix is entered. The diagonal elements must be included.

Example:

```
CORRELATION: Cor_Mat  Dimension=4  Format=UPPER;
   1.0   0.2   0.4   0.6
         1.0   0.8   0.0
               1.0   0.1
                     1.0
```

Syntax of the LOWER format

```
CORRELATION: name  Dimension=n  Format=LOWER;
      c11
      c21  c22
      ...
      cn1  cn2  ...  cnn
```

Only the lower-left triangular portion of the correlation matrix is entered. The diagonal elements must be included.

Example:

```
CORRELATION: Cor_Mat  Dimension=4  Format=LOWER;
   1.0
   0.2   1.0
   0.4   0.8   1.0
   0.6   0.0   0.1   1.0
```

Syntax of the SPARSE format:

```
CORRELATION: name  Dimension=n  Format=SPARSE;
      i1  j1  c(i1,j1)
      i2  j2  c(i2,j2)
      ...
      im  jm  c(im,jm)
```

Only the nonzero and off-diagonal elements are entered. *i1, i2, ..., im* are row indices, *j1, j2, ..., jm* are column indices, and *c(i1,j1), ..., c(im,jm)* are the corresponding elements of the correlation matrix.

Example:

```
CORRELATION: Cor_Mat  Dimension=4  Format=SPARSE;
   1  2   0.2
   1  3   0.4
   1  4   0.6
   2  3   0.8
   3  4   0.1
```

Indices of Correlated Statistical Parameters

The CORRELATION keyword is used to link a set of statistical parameters to a correlation matrix defined in the Statistics block.

Example:

```
V1:  5  {Normal SIGMA=10%  CORRELATION=Cor_Mat[2]};
V2:  2  {Normal SIGMA=5%   CORRELATION=Cor_Mat[3]};
```

This indicates that the correlation coefficients of V1 and V2 are given by the 2nd and 3rd rows (columns), respectively, of the correlation matrix defined in the Statistics block and identified by the string name "Cor_Mat". The correlation coefficient between V1 and V2, for instance, is the element at position (2,3).

Multiple Correlation Matrices

If necessary, you can define multiple correlation matrices in the Statistics block, each matrix representing the correlations among a subset of statistical parameters. The subsets of statistical parameters cannot overlap.

Correlation Matrix Generated by Statistical Modeling

When you use HarPE for statistical modeling, a correlation matrix is automatically generated for the statistical model (see Chapter 11 for further details).

10.8 Sample Distribution

HarPE allows you to define statistical parameters by a sample of outcomes supplied in the Sample block.

Syntax:

```
Sample
   FORMAT  name1   name2   ...   namen;
           x11     x12     ...    x1n
           x21     x22     ...    x2n
                   ...
           xm1     xm2     ...    xmn
End

Expression

   label: nominal (Sample);
   ...
End
```

where *name1, name2, ..., namen* identify a set of *n* statistical parameter,
x1i, ..., xmi are a sample of outcomes for the statistical parameter identified by *namei*, and the sample size is *m*.
To define a statistical parameter of sample distribution, *label* must be one of the *names*.

Example:

```
Sample
   FORMAT  X1     VAR2    RDS;
           4.987  ...     ...
           5.123  ...     ...
                  ...
           4.855  ...     ...
End

Expression

   X1:  5  (Sample);
   ...
End
```

where X1 is defined as a statistical parameter of sample distribution and a sample of outcomes is supplied for X1 in the Sample block.

Statistical parameters of sample distribution can only be defined in the Expression block as labels, since the label names are needed to identify with the names defined in the Sample block. If you wish to define sample distribution for circuit model parameters, you must first define the statistical parameters as labels and then assign the labels to circuit model parameters.

Generation of Outcomes

For statistical parameters of sample distribution, the outcomes supplied in the Sample block are directly used.

Consider the statistical parameter X1 in the preceding example. In Monte Carlo analysis, the outcomes for X1 are directly taken from the Sample block as 4.987, 5.123, ..., 4.855.

The *nominal* value defined for a statistical parameter of sample distribution is irrelevant in Monte Carlo analysis, since the parameter value is directly given by the outcomes. The *nominal* value is used only in the nominal simulation described in Chapter 8.

Recycling of the Outcomes

If the number of outcomes specified in the MonteCarlo block for Monte Carlo analysis (Section 10.10) exceeds the number of outcomes supplied in the Sample block (i.e., the sample size), then the supplied outcomes are recycled (reused).

To illustrate, if 50 outcomes are supplied in the Sample block for a statistical parameter as *x1, x2, ..., x50*, and 200 outcomes are specified in the MonteCarlo block, then in the Monte Carlo analysis the set of 50 outcomes will be used 4 times. For example, the 1st, 51st, 101st and 151st outcomes will have identical values.

Sample Correlations Are Preserved

When statistical parameters are defined by sample distribution, the Monte Carlo outcomes generated will reflect any parameter correlations present in the Sample block data. This is because for each outcome, all the statistical parameters defined by sample distribution are assigned values from the same row in the Sample block.

Sample Block Generated by Statistical Modeling

When you use HarPE for statistical modeling, a Sample block is automatically created to contain the sample of extracted parameter values from multi-device measurements (see Chapter 11 for further details). The statistical sample can be postprocessed to obtain a multidimensional normal distribution or you can choose to keep the Sample block to represent the statistical parameters and use it directly for Monte Carlo analysis.

10.9 User-Created Hybrid Distributions

You can combine different statistical distributions in an expression to create a hybrid distribution, such as

```
V_Normal:  5  {Normal SIGMA=5%};
V_Uniform:  3  {Uniform TOL=10%};

V_Hybrid = V_Normal + V_Uniform;
```

Another example:

```
V_Normal:  0  {Normal SIGMA=1};
V_Computed = 50 + exp(V_Normal);
```

In this example, V_Normal represents a normal distribution with a zero mean and a unit standard deviation. V_Computed actually represents a lognormal distribution.

Hierarchical (multi-level) distributions can also be defined. For instance,

```
V_Low_Level:  5  {Normal SIGMA=10%};

V_High_Level:  V_Low_Level  {Normal SIGMA=15%};
```

This is a two-level hierarchical distribution. The nominal of V_High_Level is itself a statistical parameter with the normal distribution specified by V_Low_Level. V_Low_Level may represent, for example, a "global" statistical uncertainty, and V_High_Level may model a "local" uncertainty on top of the global uncertainty.

10.10 MonteCarlo Block

The MonteCarlo block allows you to define the number of outcomes, simulation ranges, responses and yield specifications for Monte Carlo analysis.

Syntax:

```
MonteCarlo
   N_Outcomes=n;
   sweep_set;
   sweep_set;
   ...
   sweep_set;
End
```

Each *sweep_set* is a statement delimited by a semicolon ";".
The statement N_Outcomes=*n* defines the number of statistical outcomes to be generated in the Monte Carlo analysis (the same number of outcomes for all sweep sets).

Syntax of Monte Carlo sweep sets:

```
sweep_label: sweep_range
sweep_label=value  ...  sweep_label=value
response          ...
response < goal   ...
response > goal   ...;
```

where each *sweep_label* represents a built-in sweep parameter,
one *sweep_label* can have a *sweep_range* containing multiple values,
the other *sweep_labels* are assigned a single *value*,
each *response* represents a built-in or user-defined response,
and each *goal* represents a specification for yield calculation.

Example:

```
MonteCarlo
   N_Outcomes = 200;

   VG=-2  VD=4  FREQ: from 2GHZ to 18GHZ step=1GHZ  MS21 PS21;

   VG=-1.7  VD=4  FREQ=6GHz  PIN: -10dBm -5dBm 0dBm
   Spectral_Purity > 19  Conversion_Gain > 2.5;
End
```

There are many similarities between the MonteCarlo, Sweep and Specification blocks. The Sweep block is described in Chapter 8 and the Specification block is described in Chapter 9.

Sweep Labels

Table 10.2 lists the set of built-in sweep labels (which is a subset of the predefined labels described in Chapter 4).

TABLE 10.2 BUILT-IN SWEEP LABELS

Label	Description
VD	drain (collector) bias voltage
VG	gate (base) bias voltage
FREQ	frequency or fundamental frequency
PIN	input power level

Note: VD is not applicable if only one bias port is defined.

Circuit Simulation Type

The type of circuit simulation to be performed (large-signal, small-signal or DC) is implied by the sweep labels defined, as shown in Table 10.3.

TABLE 10.3 SWEEP LABELS AND SIMULATION TYPES

Sweep Labels	Implied Simulation Type
VG, VD	DC
VG, VD, FREQ	small-signal AC
VG, VD, FREQ, PIN	large-signal (harmonic balance)

Example:

```
MonteCarlo
   N_Outcomes = 200;

   VG=-2  VD: from 0 to 5 step=0.2  ID;

   VG=-1.7  VD=4  FREQ=6GHz  PIN: -10dBm -5dBm 0dBm
   Spectral_Purity > 19  Conversion_Gain > 2.5;
End
```

DC simulation is assumed for the first Monte Carlo sweep set since it contains the bias voltages VG and VD only. The simulation type for the second sweep set is assumed to be large-signal HB, since it includes the input power PIN as a sweep parameter.

Define Discrete Values for Sweep Labels

Syntax:

sweep_label: *x1, x2, ... xn*

where *x1, x2, ..., xn* represent numerical values.

Example:

```
FREQ: 2, 3.2, 5.7, 13, 15
```

The comma separating two discrete values is optional.

Sweep an Interval with Uniform Step Size

Syntax:

sweep_label: `from` *x1* `to` *x2* `step=`*x3*

where *x1, x2* and *x3* are numerical values. This translates into a set of values as: *x1, x1+x3, x1+2×x3, ..., x2.*

Example:

```
VD: from 0 to 6 step=0.2
```

This translates into a set of values for VD as 0, 0.2, 0.4, ..., 6.

The upper limit of the interval, namely *x2*, is always included in the translated values, even if the step size does not divide the interval evenly. For example,

```
VD:  from 1.5 to 2.75 step=0.5
```

The translated values are 1.5, 2, 2.5, 2.75, which include the upper limit 2.75.

You can specify a negative step size, such as

```
VD:  from 5 to 1.5 step=-0.5
```

The translated values are 5, 4.5, 4, ..., 1.5. In this case, the lower limit must be greater than the upper limit, i.e., *x1* > *x2*.

The number of subintervals is $n = (x2 - x1) / x3$, if $x3$ divides $(x2 - x1)$ evenly, or else $n = (x2 - x1) / x3 + 1$, (the number of values is $n + 1$). The limit is $n \leq 1023$.

Specify the Number of Steps

Syntax:

sweep_label: `from` *x1* `to` *x2* `N=`*n*

where *x1* and *x2* are numerical values, *n* is an integer, $1 \leq n \leq 1023$.
The step size is $(x2 - x1) / n$. The number of values is $n + 1$.

Example:

```
VD:  from 0 to 4 N=8
```

This translates into a step size of (4 - 0) / 8 = 0.5, and a set of values as 0, 0.5, 1, ..., 4. Note that the total number of values is 9.

Combine Intervals and Points

You can specify a combination of sweep intervals and discrete values for a sweep label. For example,

```
VD:  0  0.1  from 0.2 to 1 step=0.2  from 1.5 to 6 step=0.5
```

which translates into the following set of values:

```
0  0.1  0.2 0.4 0.6 0.8 1  1.5 2 2.5 3 3.5 4 4.5 5 5.5 6
```

This can be useful if you wish to have different step sizes in different sweep ranges. For example, an accurate representation of the circuit DC characteristics usually requires more densely spaced bias points below the "knee".

The set of values assigned to a sweep label will not be sorted by the file parser. During simulation, the values are assigned in the exact order as they appear in the circuit file. If the values need to be processed in a particular order (ascending, descending, etc.), they must be entered in the desired order.

Only One Multi-Step Sweep Range per Monte Carlo Sweep Set

Unlike the sweep sets for the Sweep and Specification blocks, a Monte Carlo sweep set can contain only one sweep range with multiple steps (values) for one of the sweep labels. The other sweep labels must be assigned a single value. For example, you can define frequency sweep in a Monte Carlo sweep set and power sweep in another sweep set, but you cannot define both frequency sweep and power sweep in the same sweep set.

Keywords HARM: Highest Harmonic in Large-Signal Simulation

For large-signal sweep sets, you can specify the highest harmonic to be included in the harmonic balance simulation using the keyword HARM.

Syntax:

HARM=*m* ...;

m specifies the highest harmonic for the sweep set, $1 \leq m \leq 8$.
The default is HARM = 4.

Example:

```
VG: -2  VD: 4  PIN: from -10DBM to 10DBM step=5DBM
FREQ: 6GHZ  HARM=5;
```

For this sweep set, the first 5 harmonics and the DC component will be included in the harmonic balance simulation.

A higher value for HARM may improve the convergence of the Newton iteration for solving the nonlinear harmonic balance equations. A higher value for HARM can also be used to improve the accuracy of simulation, at the expense of a longer simulation time.

Keyword RREF: Reference Impedance for S Parameters

For small-signal AC sweep sets, you can specify the reference impedance for *S* parameters using the keyword RREF.

Syntax:

RREF=*x* ...;

where *x* specifies the *S*-parameter reference impedance for the sweep set,
x must be a positive constant value, and the default is 50OH.

Example:

```
VG: -2  VD: 4  FREQ: from 2GHZ to 18GHZ step=1GHZ  RREF=10;
```

where the reference impedance for the *S* parameters is specified as 10OH.

10.11 Responses and Yield

Each Monte Carlo sweep set must contain one or more responses. These responses will be calculated during Monte Carlo analysis, resulting in a set of values which is collectively referred to as the statistical responses of the circuit.

The responses can be built-in and user-defined responses.

Built-In Circuit Responses

Corresponding to each type of simulation (DC, small-signal or large-signal) is a set of built-in circuit responses, as listed in Table 10.4.

TABLE 10.4 BUILT-IN RESPONSES

Simulation Type	Label	Description
DC	`ID`	DC drain (collector) current
	`IG`	DC gate (base) current
small-signal	`MS`*ij*	magnitude of S_{ij}, $i,j = 1, n$
	`PS`*ij*	phase of S_{ij}, $i,j = 1, n$
	`RS`*ij*	real part of S_{ij}, $i,j = 1, n$
	`IS`*ij*	imaginary part of S_{ij}, $i,j = 1, n$
large-signal	`IDO`	DC component of drain (collector) current
	`POUT`*k*	output power magnitude in dBm, *k*th harmonic
	`POUTW`*k*	output power magnitude in Watts, *k*th harmonic

- The simulation type of a sweep set is implied by the sweep labels defined in that sweep set, as shown in Table 10.3.
- n is the number of ports (see Chapter 5).
- If there is only one bias port, then `IG` is not available.

User-Defined Responses

In addition to the built-in responses, arbitrary user-defined responses can be created using expressions.

User-defined responses should be defined as labels in the Expression block.

Example:

```
PS11_in_Radian:  PS11 * PI / 180;
```

where PS11 is a predefined label representing a built-in response, namely the phase of S_{11} in degrees.

Details of the Expression block, including predefined labels, algebraic operators and mathematical functions are described in Chapter 4.

Simulation Types of User-Defined Responses

The simulation type of a user-defined response is determined by the built-in responses involved in its definition.

Example:

```
Power_Added_Efficiency = 100 * (POUTW1 - PINW) / (ID0 * VD);
ID_in_mA:  ID * 1000;
PS11_in_Radian:  PS11 * PI / 180;
```

where Power_Added_Efficiency is a user-defined large-signal response, ID_in_mA is a user-defined DC response and PS11_in_Radian is a user-defined small-signal response.

To ensure a unique simulation type, all the predefined labels involved in a user-defined response must have the same simulation type. For example,

```
Label_1: POUT1 + MS11;
```

This is not permissible since the definition of Label_1 attempts to mix a large-signal response POUT1 with a small-signal response MS11.

This rule also applies when user-defined responses are used in expressions to define other labels. In other words, a user-defined response inherits the simulation type of the built-in responses it refers to, and all the built-in and user-defined responses involved in an expression must have the same simulation type.

User-defined labels which do not depend on any circuit responses are "type-less" and therefore free from the restriction regarding simulation types.

Specifications for Yield

If you wish to estimate the yield by Monte Carlo analysis, you must define at least one specification in the MonteCarlo block.

You can define upper specifications, such as

response < goal

and lower specifications, such as

response > goal

respectively.

The yield estimated by Monte Carlo analysis is given by

Yield = N_Pass / N_Outcomes

where *N_Pass* represents the number of "passed" outcomes, and *N_Outcomes* is the total number of outcomes.

A "passed" outcome means that for that outcome all the responses which have associated goals, from all the sweep sets in the MonteCarlo block, satisfy the specifications at all the sweep points. If any of the responses associated with a goal from any of the sweep sets violates the specification at any of the sweep points, then that outcome is considered failed.

Single Specifications Are Not Allowed

Single specifications of the form

response = goal

are not allowed because such exact equality specifications will almost always lead to a zero yield which does not provide any useful information. Any single specification in nominal design should be replaced by a pair of upper and lower specifications to allow a "window" appropriate for the response.

Responses without Goals

If you do not specify goals for some of the responses, it means that you wish them to be calculated and displayed but not to be included in the yield calculation. Such responses do not affect the yield, i.e., regardless of their values, they neither cause an outcome to pass nor cause it to fail.

If none of the responses is associated with a goal, then the yield will not be calculated.

10.12 HarPE.MonteCarlo Menu Option

The menu option HarPE.MonteCarlo is designed to manage Monte Carlo simulation and the resulting statistical displays (☞ see Chapter 1 for a general discussion on menus and menu operations).

The steps needed for Monte Carlo simulation and display are outlined as follows.

- ▷ Define in the circuit file one or more statistical parameters and a MonteCarlo block using the file editor (Chapter 2).
- ▷ Parse the circuit file by exiting the editor.
- ▷ Invoke the menu option HarPE.MonteCarlo (by moving the cursor to highlight the option and pressing <ENTER>, see Chapter 1).
- ▷ Select from the menu of available display formats as listed in Table 10.5.

TABLE 10.5 STATISTICAL DISPLAY FORMATS

Menu Option	Brief Description
HarPE.MonteCarlo.Histogram	displays histograms of individual statistical responses
HarPE.MonteCarlo.RunChart	displays run charts of individual statistical responses
HarPE.MonteCarlo.Sweep	displays statistical responses versus parameter sweeps
HarPE.MonteCarlo.Yield	displays the yield estimated by Monte Carlo analysis
HarPE.MonteCarlo.Cumulative	displays cumulative probability distribution of a response

Monte Carlo Simulation

When you invoke the HarPE.MonteCarlo menu option, HarPE first performs the necessary simulation according to the MonteCarlo block.

During the Monte Carlo simulation, HarPE displays a bar graphics indicator of the progress as well as the following message in the upper-left-hand corner of the screen:

```
MonteCarlo Analysis ...  Press any key to interrupt
```

Using a random number generator, statistical outcomes are generated. Each statistical parameter is assigned a statistically perturbed value according to the distribution defined in the circuit file. The responses specified in the MonteCarlo block are calculated through circuit simulation and the yield estimate is updated for each outcome.

Interrupt the Monte Carlo Analysis

While the Monte Carlo analysis is in progress, you can interrupt it at any time by pressing a key (any key) on the keyboard. Then the following prompt will be displayed:

```
Terminate Operation (Y/<N>)?
```

and you can decide whether to terminate or to continue the operation.

The .mca Files

Monte Carlo analyses can be time-consuming. To avoid duplicated analyses, a feature is built into HarPE to save the results of the Monte Carlo analysis in an encoded file, which can be retrieved and displayed in a subsequent session.

The encoded file is named after the circuit file by replacing the extension .ckt with .mca. To ensure that the .mca file is uniquely associated with the circuit file, a .mca file will be created only if the circuit file has the extension .ckt.

For example, if you have two circuit files named example.ckt and example.old, then the Monte Carlo analysis results for example.ckt will be saved in example.mca, but example.old will not have a .mca file.

Furthermore, to ensure that the information stored is up to date, a "time stamp" is encoded into the .mca file which reflects the date and time of the corresponding circuit file. HarPE always checks this time stamp in a .mca file before using it. Therefore, any operation which changes the date and time of the circuit file after a .mca file is created will invalidate the stored statistical data.

Editors such as vi may modify the date and time of files. Even the cp (copy) command may change the date and time of the file copied.

A new .mca file is created in three situations:

- A new circuit file is read, and simulated without editing or optimization.
- A circuit file is edited or optimized, then saved, and simulated.
- A circuit file is edited or optimized, then simulated, and saved.

Here "simulated" refers to Monte Carlo simulation.

10.13 Histograms

The menu option HarPE.MonteCarlo.Histogram displays the histograms of individual responses at selected sweep points.

A histogram divides the values of a response into a number of evenly spaced intervals (bins). The number of values (outcomes) which fall into each bin is counted, and the result is displayed as a bar chart, as illustrated in Fig. 10.2.

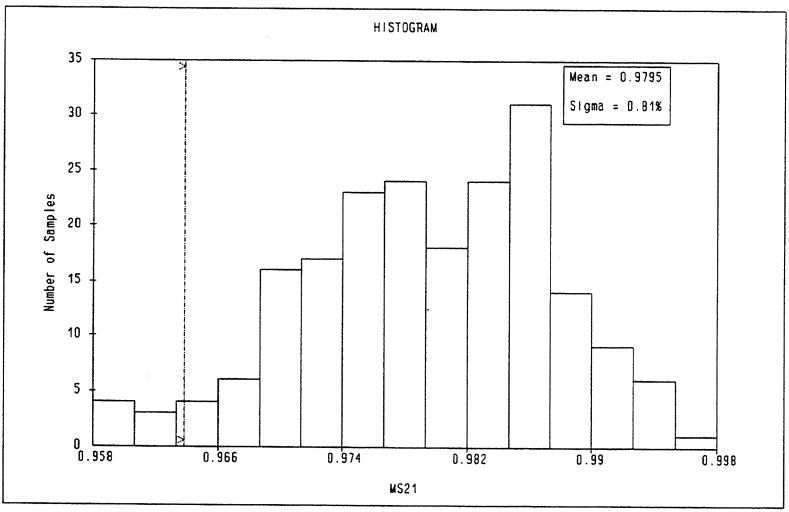


Fig. 10.2 Histogram.

The vertical line in Fig. 10.2 represents the specification (for yield) associated with the selected response, if defined in the MonteCarlo block. The arrows attached to the line indicate whether the line represents an upper or a lower specification.

Histograms of Model Responses and Data

The histograms of statistical model responses and of the corresponding measurement data can be displayed in the same diagram. This is very useful if the user wants to compare the statistics of the simulated responses with the statistics of the measurements.

Fig. 10.3 shows the histograms of a model response and of the corresponding data.

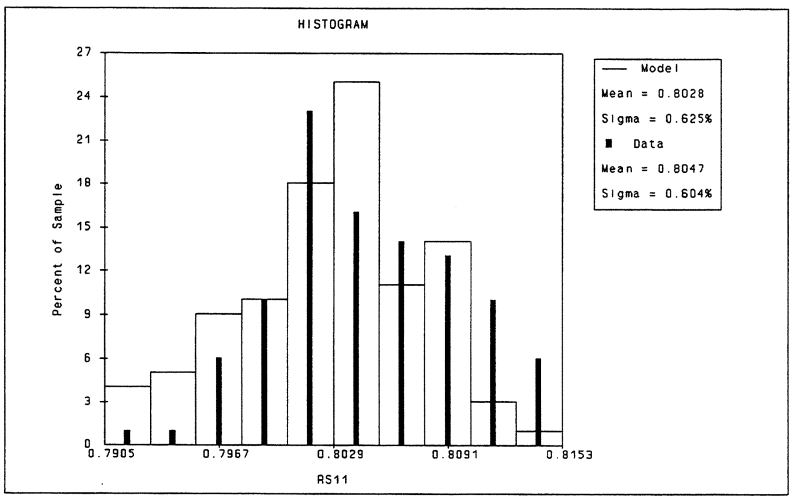


Fig. 10.3 Histograms of a model response and the corresponding data.

The histogram of the model response is displayed by wide bars while the histogram of the corresponding data is plotted using solid bars. It clearly shows the difference between the distribution of the model response and the distribution of the corresponding data.

Pop-Up Window

When you select HarPE.MonteCarlo.Histogram, a pop-up window will appear as

```
Output form:        Graphical
Zoom scale:         <TAB> to set scale
Response:           MS21
Sweep set:          1 of 2
Gate bias:          -2V
Drain bias:         4V
Frequency:          5GHZ
Input Power:        n/a
High truncation:
Low truncation:
Number of bins:     15
Mode:               Model and data
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can move the cursor to select and change the various options.

If the MonteCarlo block contains more than one sweep set, you can select a sweep set for display through the "Sweep set" option in the pop-up window.

If the selected sweep set contains more than one response, you can select the response to be displayed through the Response option in the pop-up window.

If the sweep set contains a bias, frequency or power sweep, you can select a value for that sweep label through the appropriate option in the pop-up window.

The choice for some options may be shown as "n/a" which means that those options are not applicable. For example, the "Input Power" option is not applicable to small-signal sweep sets.

Mode

The "Mode" option in the pop-up window offers two choices: "Model only" and "Model and data".

If you choose "Model only", then only the histogram of the model responses are displayed as illustrated in Fig. 10.2.

If you choose "Model and Data", then the histograms of the model responses and the corresponding measurement data are displayed in the same diagram as shown in Fig. 10.3.

Number of Bins

You can specify the number of intervals (bins) for dividing the values of the selected response. Available choices are 5, 10, 15 and 20 bins. The histogram shown in Fig. 10.2 consists of 15 bins.

High and Low Truncation Limits

Generally, the limits (minimum and maximum) of the response values are not rounded figures. Consequently, the width of the bins is not a rounded figure either. For example, if the response values range from 1.012 to 2.983, and the number of bins is selected as 10, then the width of the bins would be 0.1971. This can be inconvenient, especially if you try to compare two histograms (such as the histograms of simulated and measured responses).

The "High truncation" and "Low truncation" options in the pop-up window allow you to define rounded limits for the bin ranges. For instance, if you round the low limit from 1.012 to 1 and the high limit from 2.983 to 3, and keep the number of bins as 10, then the width of the bins will be 0.2.

The truncation limits of the histogram shown in Fig. 10.2 are 0.958 and 0.998.

Graphics Zoom

By default, histogram displays are scaled to the range of the response values. You can change the display scale ("zooming") using the "Zoom scale" option in the pop-up window.

The Zoom scale option leads you to another pop-up window as

```
Zoom feature:       Enabled
Ymin of the plot:   0
yMax of the plot:   1.2
Xmin of the plot:   0
xmAx of the plot:   1.5
N x-axis ticks:     6
n y-axis Ticks:     6
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

The parameters "Ymin", "yMax", "Xmin" and "xmAx" define the corners of the display. The parameters "N x-axis ticks" and "n y-axis Ticks" specify the number of ticks (grids) on the X-axis and Y-axis, respectively.

The zoom limits and the truncation limits can be defined independently. The zoom limits control the scales of the display, while the truncation limits determine the width of the bins.

Display Mean and Standard Deviation

If you choose "Model only" for the Mode option, the sample mean and standard deviation of the selected response is estimated and shown on the histogram display.

If you choose "Model and data" for the Mode option, the sample means and standard deviations of both the selected response and the corresponding data are estimated and shown on the histogram display.

Numerical Display

The "Output form" option in the pop-up window offers two choices: Graphical and Numerical. If you choose Numerical, then the values of the response are printed in ASCII text format, one outcome on each line. The result is loaded into HarPE's file editor to allow you to view, edit and save the numerical outputs using the features described in Chapter 2.

10.14 Cumulative Probability Distributions

The menu option HarPE.MonteCarlo.Cumulative displays the cumulative probability distributions of individual responses at selected sweep points.

Definition of Cumulative Probability Distribution

Given a sample of data

$$S = [X_1\ X_2\ \ldots\ X_n]$$

the cumulative probability distribution of S, denoted by $C(x)$, is defined as [2]

$$C(x) = n_x / n$$

where n_x is the number of data points in S which are smaller than or equal to x. When n is adequately large $C(x)$ provides a very good approximation to the theoretical probability distribution from which the sample was drawn. It can be used to test whether two samples of data come from the same probability distribution.

The cumulative probability distribution of a response is shown in Fig. 10.4.

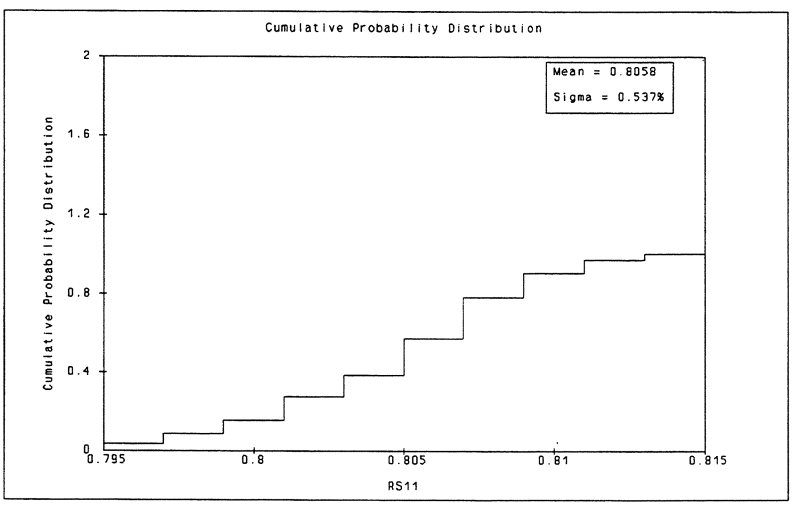


Fig. 10.4 Cumulative probability distribution of a response.

Cumulative Probability Distributions of Model Responses and Data

The cumulative probability distributions of statistical model responses and of the corresponding measurement data can be displayed in the same diagram. This is very useful if the user wants to compare the statistics of the simulated responses with the statistics of the measurements.

Fig. 10.5 shows the cumulative probability distributions of a model response and of the corresponding data.

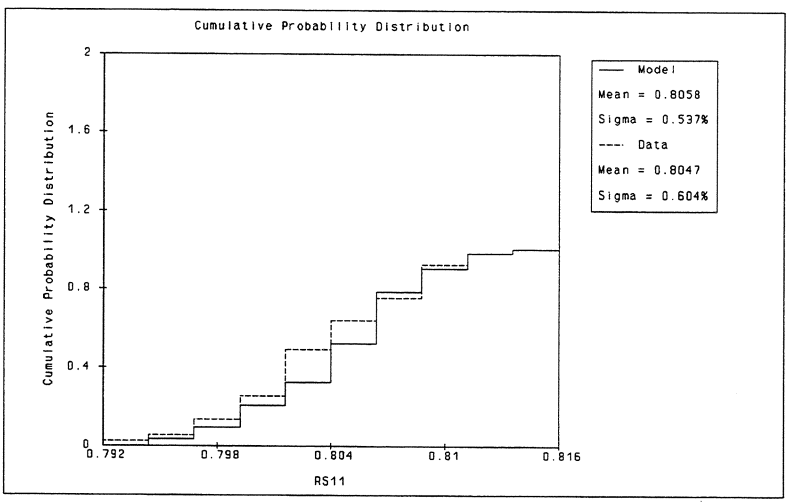


Fig. 10.5 Cumulative probability distributions of a response and the corresponding data.

Pop-Up Window

When you select HarPE.MonteCarlo.Cumulative, a pop-up window will appear as

```
Output form:        Graphical
Zoom scale:         <TAB> to set scale
Response:           MS21
Sweep set:          1 of 2
Gate bias:          -2V
Drain bias:         4V
Frequency:          5GHZ
Input Power:        n/a
High truncation:
Low truncation:
Number of bins:     10
Mode:               Model only
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can move the cursor to select and change the various options.

If the MonteCarlo block contains more than one sweep set, you can select a sweep set for display through the "Sweep set" option in the pop-up window.

If the selected sweep set contains more than one response, you can select the response to be displayed through the Response option in the pop-up window.

If the sweep set contains a bias, frequency or power sweep, you can select a value for that sweep label through the appropriate option in the pop-up window.

The choice for some options may be shown as "n/a" which means that those options are not applicable. For example, the "input Power" option is not applicable to small-signal sweep sets.

Mode

The Mode option in the pop-up window offers two choices: "Model only" and "Model and data".

If you choose "Model only", then only the cumulative probability distributions of the model responses are displayed as illustrated in Fig. 10.4.

If you choose "Model and data", then the cumulative probability distributions of the model responses and the corresponding measurement data are displayed in the same diagram as shown in Fig. 10.5.

Number of Bins

You can specify the number of intervals (bins) for dividing the values of the selected response. Available choices are 5, 10, 15 and 20 bins. The cumulative probability distribution shown in Fig. 10.4 consists of 10 bins.

High and Low Truncation Limits

Generally, the limits (minimum and maximum) of the response values are not rounded figures. Consequently, the width of the bins is not a rounded figure either. For example, if the response values range from 0.7956 to 0.8145, and the number of bins is selected as 10, then the width of the bins would be 0.00189. This can be inconvenient at times.

The "High truncation" and "Low truncation" options in the pop-up window allow you to define rounded limits for the bin ranges. For instance, if you round the low limit from 0.7956 to 0.795 and the high limit from 0.8145 to 0.815, and keep the number of bins as 10, then the width of the bins will be 0.002.

The truncation limits of the cumulative probability distribution shown in Fig. 10.4 are 0.795 and 0.815.

Graphics Zoom

By default, cumulative probability distribution displays are scaled to the range of the response values. You can change the display scale ("zooming") using the "Zoom scale" option in the pop-up window.

The "Zoom scale" option leads you to another pop-up window as

```
Zoom feature:       Enabled
Ymin of the plot:   0
yMax of the plot:   1.2
Xmin of the plot:   0
xmAx of the plot:   1.5
N x-axis ticks:     6
n y-axis Ticks:     6
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

The parameters "Ymin", "yMax", "Xmin" and "xmAx" define the corners of the display. The parameters "N x-axis ticks" and "n y-axis Ticks" specify the number of ticks (grids) on the X-axis and Y-axis, respectively.

The zoom limits and the truncation limits can be defined independently. The zoom limits control the scales of the display, while the truncation limits determine the width of the bins.

Display Mean and Standard Deviation

If you select "Model only" for the Mode option, the sample mean and standard deviation of the selected response is estimated and shown on the cumulative probability distribution display.

If you select "Model and data" for the Mode option, the sample means and standard deviations of both the selected response and the corresponding data are estimated and shown on the cumulative probability distribution display.

Numerical Display

The "Output form" option in the pop-up window offers two choices: Graphical and Numerical. If you choose Numerical, then the values of the response are printed in ASCII text format, one outcome on each line. The result is loaded into HarPE's file editor to allow you to view, edit and save the numerical outputs using the features described in Chapter 2.

10.15 Run Charts

The menu option HarPE.MonteCarlo.RunChart displays the run charts of individual responses at selected sweep points.

A run chart plots the value of a response versus the outcome index. The X-axis is the outcome index. The logical range of the outcome index is from 1 to n, where n is the total number of outcomes. However, the X-axis of the display is shown from 0 to n so that the grids (divisions) can be labelled with rounded figures, as illustrated in Fig. 10.6.

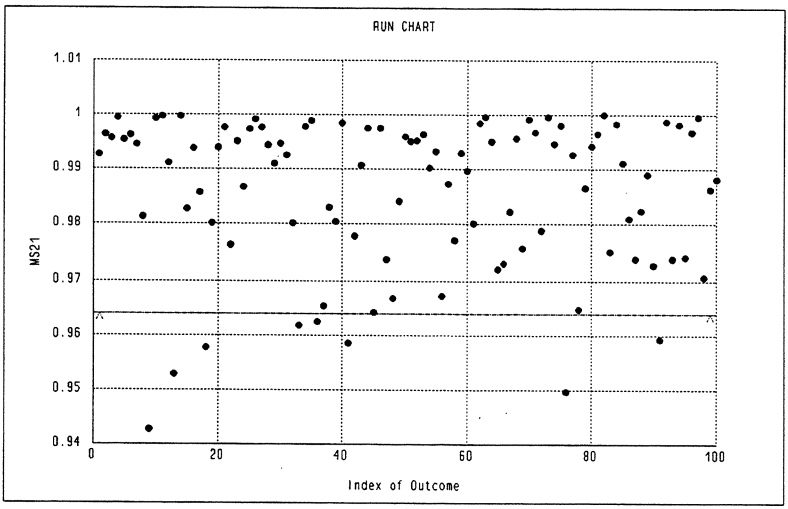


Fig. 10.6 Run chart.

The horizontal line in Fig. 10.6 represents the specification (for yield) associated with the selected response, if defined in the MonteCarlo block. The arrows attached to the line indicate whether the line represents an upper or a lower specification.

Pop-Up Window

When you select HarPE.MonteCarlo.RunChart, a pop-up window will appear as

```
Output form:        Graphical
Zoom scale:         <TAB> to set scale
Response:           MS21
Sweep set:          1 of 2
Gate bias:          -2V
Drain bias:         4V
Frequency:          5GHZ
Input Power:        n/a
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can move the cursor to select and change the various options.

If the MonteCarlo block contains more than one sweep set, you can select a sweep set for display through the "Sweep set" option in the pop-up window.

If the selected sweep set contains more than one response, you can select the response to be displayed through the Response option in the pop-up window.

If the sweep set contains a bias, frequency or power sweep, you can select a value for that sweep label through the appropriate option in the pop-up window.

The choice for some options may be shown as "n/a" which means that those options are not applicable. For example, the "input Power" option is not applicable to small-signal sweep sets.

Graphics Zoom

By default, run charts are scaled to the range of the response values. You can change the display scale ("zooming") using the "Zoom scale" option in the pop-up window.

The Zoom scale option leads you to another pop-up window as

```
Zoom feature:      Enabled
Ymin of the plot:  0
yMax of the plot:  1.2
Xmin of the plot:  0
xmAx of the plot:  1.5
N x-axis ticks:    6
n y-axis Ticks:    6
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

The parameters "Ymin", "yMax", "Xmin" and "xmAx" define the corners of the display. The parameters "N x-axis ticks" and "n y-axis Ticks" specify the number of ticks (grids) on the X-axis and Y-axis, respectively.

Numerical Display

The "Output form" option in the pop-up window offers two choices: Graphical and Numerical. If you choose Numerical, then the values of the response are printed in ASCII text format, one outcome on each line. The result is loaded into HarPE's file editor to allow you to view, edit and save the numerical outputs using the features described in Chapter 2.

10.16 Monte Carlo Sweep Display

The menu option HarPE.MonteCarlo.Sweep displays a statistical response versus bias, frequency or power sweep. Each curve on the display represents the bias, frequency or power sweep response for one statistical outcome. The total number of curves displayed is equal to the number of outcomes specified in the MonteCarlo block.

A Monte Carlo sweep display is illustrated in Fig. 10.7.

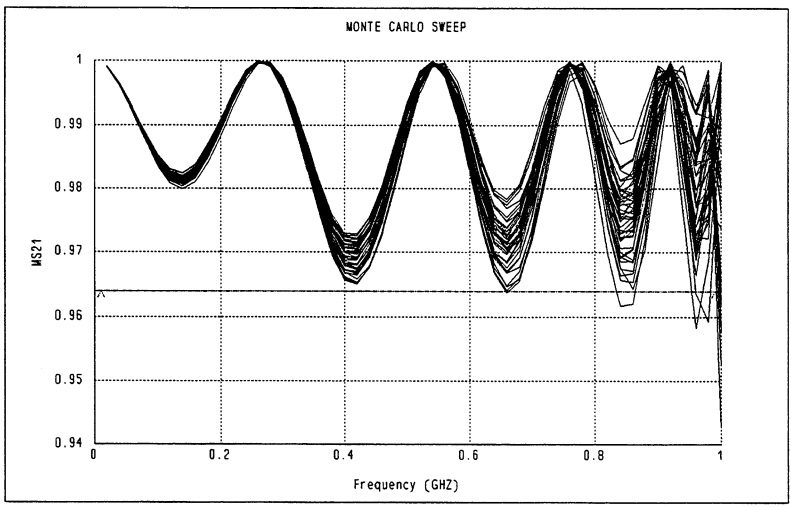


Fig. 10.7 Monte Carlo sweep display.

The horizontal line in Fig. 10.7 represents the specification (for yield) associated with the selected response, if defined in the MonteCarlo block. The arrows attached to the line indicate whether the line represents an upper or a lower specification.

Pop-Up Window

When you select HarPE.MonteCarlo.Sweep, a pop-up window will appear as

```
Output form:    Graphical
Zoom scale:    <TAB> to set scale
X-axis:         Frequency
Y-axis:         MS21
Sweep set:      1 of 1
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

You can move the cursor to select and change the various options.

Select a Sweep Set

For Monte Carlo sweep display to be available, the MonteCarlo block must contain at least one sweep set which includes a bias, frequency or power sweep, i.e., one of these sweep labels is defined with a multi-step sweep range.

If the MonteCarlo block contains a number of sweep sets which include a bias, frequency or power sweep, then you can select any one of the sweep sets for display through the Sweep set option in the pop-up window.

The X-axis

The X-axis of the Monte Carlo sweep display is the sweep label which is defined with multiple values. The X-axis is VG, VD, Frequency or Input Power, depending on whether the sweep set contains a bias, frequency or power sweep.

Since each Monte Carlo sweep set can include only one sweep range, the X-axis for the Monte Carlo sweep display is fixed once a sweep set has been selected.

Select a Response for the Y-axis

If the selected sweep set contains more than one response, you can select the response to be displayed through the Y-axis option in the pop-up window.

Graphics Zoom

By default, Monte Carlo sweep displays are scaled to the range of the response values. You can change the display scale ("zooming") using the "Zoom scale" option in the pop-up window.

The Zoom scale option leads you to another pop-up window as

```
Zoom feature:      Enabled
Ymin of the plot:  0
yMax of the plot:  1.2
Xmin of the plot:  0
xmAx of the plot:  1.5
N x-axis ticks:    6
n y-axis Ticks:    6
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

The parameters "Ymin", "yMax", "Xmin" and "xmAx" define the corners of the display. The parameters "N x-axis ticks" and "n y-axis Ticks" specify the number of ticks (grids) on the X-axis and Y-axis, respectively.

Numerical Display

The "Output form" option in the pop-up window offers two choices: Graphical and Numerical. If you choose Numerical, then the values of the response are printed in ASCII text format, one outcome on each line. The result is loaded into HarPE's file editor to allow you to view, edit and save the numerical outputs using the features described in Chapter 2.

The size of the numerical data file can be very large because it contains the response values for all the outcomes.

If you wish to obtain numerical display at a single sweep point instead of at all the sweep points, you can choose the numerical display option for HarPE.MonteCarlo.Histogram, HarPE.MonteCarlo.Cumulative or HarPE.MonteCarlo.RunChart (Sections 10.13, 10.14 and 10.15).

10.17 Yield

The menu option HarPE.MonteCarlo.Yield prints the yield numerically on the screen.

The yield is estimated and printed for every 10 outcome interval. The percentage of passed outcomes from the first 10 outcomes is calculated and printed, followed by the percentage from the first 20 outcomes, and so on.

For example, if the total number of outcomes is 200, the display on the screen may look like the illustration in Fig. 10.8.

Only those responses which are associated with goals are taken into account in the yield calculation (☞ Section 10.11).

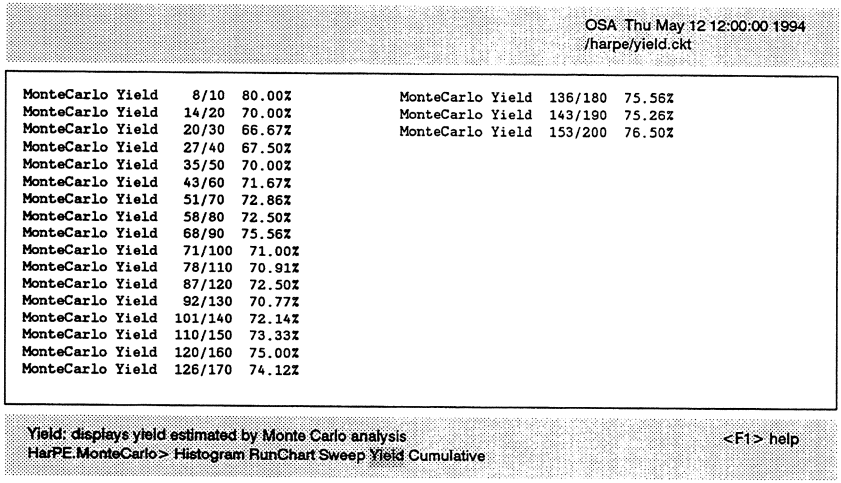


Fig. 10.8 Yield printed numerically.

10.18 Technical References

[1] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes. The Art of Scientific Computing*. Cambridge University Press: Cambridge, 1986.

[2] J.W. Bandler, R.M. Biernacki, Q. Cai and S.H. Chen, "A novel approach to statistical modeling using cumulative probability distribution fitting," *IEEE MTT-S Int. Microwave Symp. Dig.* (San Diego, CA), 1994, pp. 385-388.

11

Statistical Modeling

11
Statistical Modeling

11.1 Overview

The advanced statistical modeling feature available in HarPE is based on OSA's pioneering research work (a few key references are listed at the end of this chapter).

The capability to extract statistical models of nonlinear devices from measurements is a prerequisite for yield and cost analysis and optimization. HarPE implements two methods for statistical modeling: indirect statistical modeling and direct statistical modeling.

A two-stage method of multi-device parameter extraction and statistical postprocessing is used for indirect statistical modeling. Cumulative probability distribution matching or histogram matching is used in direct statistical modeling.

Multi-Device Parameter Extraction

Suitable measurements (large-signal power spectra, small-signal *S* parameters and/or DC data) are collected from a sample of (typically 100 to 300) devices. This may be automated by some measurement systems (such as the Cascade Microtech's MicroCAT system which is supported by HarPE). From the measured data, multiple device models are extracted through repeated optimization.

Statistical Postprocessing

The sample of extracted model parameters is postprocessed to generate a consolidated statistical model, which is then back annotated to produce a HarPE circuit file suitable for Monte Carlo analysis. Parameter correlations are automatically estimated and included in the model. For parameters exhibiting substantially non-Gaussian distributions, you can choose to keep the histograms as discrete approximations of the marginal density functions. This combined discrete/normal approach developed by OSA preserves the means, standard deviations, correlations and marginal distributions derived from the sample. It provides enhanced accuracy of the model while retaining the simplicity of the normal distribution.

The statistical postprocessor can also be applied to user-supplied arbitrary raw data.

Direct Statistical Modeling

The statistical model is determined by fitting the cumulative probability distributions or the histograms of the model responses to those of the measured data. The optimization variables include the statistical parameters such as mean values and standard deviations of a normal distribution. Thus, the model parameter statistics are obtained directly instead of from postprocessing a set of individually extracted models.

This approach can also be used to refine the statistical model obtained by multi-device parameter extraction and statistical postprocessing (i.e., the indirect statistical modeling).

Statistical Displays

Histograms and cumulative probability distributions of the measured data and the extracted model parameters can be displayed. Parameter correlations can also be shown graphically as scatter diagrams.

Steps Needed for Indirect Statistical Modeling

- ▷ Define a set of optimization variables in the Expression and/or Model block (see Chapters 4 and 5).
- ▷ Define the circuit model in the Model block (see Chapter 5).
- ▷ Supply multi-device (statistical) measurements in the Data block (described in this chapter, also see Chapter 9).
- ▷ Define the sweep ranges and optimization specifications in the Specification block (see Chapter 9).
- ▷ Invoke the menu option HarPE.Statistical.Extract to start the multi-device parameter extraction.
- ▷ Invoke the menu option HarPE.Statistical.Postprocess to generate the consolidated statistical model.
- ▷ Save the back-annotated circuit file through the file editor (see Chapter 2).

Steps Needed for Direct Statistical Modeling

- ▷ Define a set of optimization variables in the Expression and/or Model block (see Chapters 4 and 5). The variables can include statistical parameters such as mean values and standard deviations.
- ▷ Define the statistical parameters in the Expression and/or Model block (see Chapter 10). Parameter correlation matrix can also be used if it is available.
- ▷ Define the circuit model in the Model block (see Chapter 5).

- Supply multi-device (statistical) measurements in the Data block (see Chapter 11).
- Define the Monte Carlo sweep in the MonteCarlo block according to the measurements (see Chapter 10).
- Assign initial values to optimization variables including statistical parameters such as mean values and standard deviations.
- Invoke the menu option HarPE.Statistical.Match to start optimization.

This chapter describes the format of multi-device statistical measurements in the Data block and the HarPE.Statistical menu option.

Single-device modeling and optimization are covered in Chapter 9 where the Data block, the Specification block, the optimizers and objective functions are described in detail. Since most of these concepts also apply to statistical modeling, it is essential that you should have read Chapter 9 before reading this chapter. In other words, you should be familiar with the syntax and operation for single-device modeling before using HarPE for statistical modeling.

11.2 Multi-Device Measurements

Statistical modeling requires measurement data taken on a sufficiently large sample of nominally identical devices produced by the same process (*multi-device measurements*). Typically, a sample of 100 to 300 devices is deemed sufficiently large.

The maximum number of devices that can be processed by HarPE using a single circuit file is limited to 512.

The Data Block

Measurement data is supplied in the Data block of the circuit file.

Syntax:

```
Data
   PARAMETER xxx xxx xxx
   FORMAT xxx xxx xxx xxx
      data data data ...
      ...
      data data data ...

   ...

   PARAMETER xxx xxx xxx
   FORMAT xxx xxx xxx xxx
      data data data ...
      ...
      data data data ...
End
```

The measurements may include large-signal power spectra, small-signal *S* parameters and DC IV data. You can supply separate data files and link them to the circuit file using the "#include" preprocessor directive (see Chapter 3).

☞ The Data block syntax for single-device modeling is described in detail in Chapter 9.

The only addition to the Data block syntax for statistical modeling is a new keyword INDEX to be used to identify the data taken from different devices.

Syntax:

```
PARAMETER  INDEX=k  ...  ;
```

where *k* is an integer representing a unique device index.

The device index k can be an arbitrary integer, as long as it is unique for each device. The simplest way is to use consecutive integers 1, 2, 3, ..., m, where m is the total number of devices.

Example:

```
PARAMETER INDEX=1 VG=-1.74V VD=4V;
FORMAT FREQ S2RI;
    2  0.7815 -0.5718 -2.87   1.754  0.01673  0.03549  0.5183 -0.2733
    6 -0.0363 -0.8518 -0.5194 2.372  0.07108  0.04199  0.2542 -0.4683
   10 -0.4835 -0.6226  0.621  1.605  0.09085  0.01503  0.0227 -0.5372
   14 -0.6747 -0.365   0.9452 0.8582 0.0879  -0.00667 -0.19   -0.5467
   18 -0.7532 -0.1425  0.9245 0.3272 0.07678 -0.01788 -0.3916 -0.4998
PARAMETER INDEX=1 VG=-3.1V VD=4V;
FORMAT FREQ S2RI;
    2  0.6483 -0.7165 0.00148 0.00723 0.00148 0.00723 -0.8093 0.04526
    6 -0.328  -0.8011 0.00697 0.01679 0.00697 0.01679 -0.8003 0.1368
   10 -0.6873 -0.4572 0.01229 0.02409 0.01229 0.02409 -0.7823 0.226
   14 -0.7944 -0.1724 0.01842 0.03039 0.01842 0.03039 -0.7559 0.3119
   18 -0.8094  0.0476 0.02551 0.03538 0.02551 0.03538 -0.7219 0.3935
PARAMETER INDEX=2 VG=-1.74V VD=4V;
FORMAT FREQ S2RI;
    2  0.7851 -0.5705 -2.768  1.616  0.01642  0.03588  0.5309 -0.2561
    6 -0.0351 -0.86   -0.6142 2.256  0.07139  0.04453  0.2928 -0.4427
   10 -0.4912 -0.628   0.4863 1.599  0.09341  0.01832  0.0801 -0.519
   14 -0.684  -0.3636  0.8418 0.9255 0.09253 -0.00415 -0.1193 -0.543
   18 -0.7595 -0.1365  0.872  0.4269 0.08269 -0.01703 -0.3145 -0.5158
PARAMETER INDEX=2 VG=-3.1V VD=4V;
FORMAT FREQ S2RI;
    2  0.6483 -0.7165 0.00148 0.00723 0.00148 0.00723 -0.8093 0.04549
    6 -0.328  -0.8011 0.00697 0.0168  0.00697 0.0168  -0.8001 0.1375
   10 -0.6873 -0.4572 0.0123  0.02412 0.0123  0.02412 -0.7818 0.2271
   14 -0.7944 -0.1724 0.01843 0.03042 0.01843 0.03042 -0.7551 0.3133
   18 -0.8094  0.0476 0.02554 0.03541 0.02554 0.03541 -0.7207 0.395
PARAMETER INDEX=3 VG=-1.74V VD=4V;
FORMAT FREQ S2RI;
    2  0.7847 -0.5697 -2.74   1.646  0.01661  0.03573  0.5271 -0.2657
    6 -0.0324 -0.8565 -0.535  2.26   0.07148  0.04317  0.2754 -0.4599
   10 -0.4851 -0.6276  0.5619 1.556  0.0923   0.01623  0.0501 -0.5349
   14 -0.6785 -0.3669  0.8889 0.8534 0.09005 -0.00605 -0.1601 -0.5517
   18 -0.7565 -0.1419  0.8853 0.3466 0.0792  -0.01803 -0.3622 -0.5126
PARAMETER INDEX=3 VG=-3.1V VD=4V;
FORMAT FREQ S2RI;
    2  0.6483 -0.7165 0.00148 0.0072  0.00148 0.00723 -0.8093 0.04535
    6 -0.328  -0.8011 0.00697 0.0168  0.00697 0.0168  -0.8002 0.1371
   10 -0.6873 -0.4572 0.0123  0.0241  0.0123  0.0241  -0.7821 0.2265
   14 -0.7944 -0.1724 0.01842 0.0304  0.01842 0.0304  -0.7556 0.3125
   18 -0.8094  0.0476 0.02552 0.0353  0.02552 0.03539 -0.7214 0.3941
PARAMETER INDEX=4 VG=-1.74V VD=4V;
FORMAT FREQ S2RI;
    2  0.7796 -0.5746 -3.009  1.801 0.01665  0.03543  0.515  -0.271
    6 -0.0428 -0.8527 -0.6089 2.454 0.07078  0.04217  0.2529 -0.4623
   10 -0.4899 -0.6195  0.5684 1.696 0.09068  0.01563  0.0254 -0.5287
   14 -0.6791 -0.3595  0.9234 0.951 0.08814 -0.00585 -0.1825 -0.5381
   18 -0.755  -0.1362  0.9303 0.414 0.07749 -0.01713 -0.3797 -0.4934
PARAMETER INDEX=4 VG=-3.1V VD=4V;
FORMAT FREQ S2RI;
    2  0.6483 -0.7165 0.00148 0.00723 0.00148 0.00723 -0.8093 0.04531
    6 -0.328  -0.8011 0.00697 0.01679 0.00697 0.01679 -0.8002 0.137
   10 -0.6873 -0.4572 0.01229 0.0241  0.01229 0.0241  -0.7822 0.2263
   14 -0.7944 -0.1724 0.01842 0.03039 0.01842 0.03039 -0.7557 0.3122
   18 -0.8094  0.0476 0.02551 0.03539 0.02551 0.03539 -0.7216 0.3938
```

In this example, measured S parameters are supplied for four devices which are identified by the indices 1, 2, 3 and 4. For each device, two data sets are supplied which contain the measured S parameters for two different bias points.

Circuit files which contain multi-device measurement data can still be used for single-device modeling. In that case only the data from the first device (the first device index to appear in the Data block) is used.

Cascade Microtech MicroCAT Data Format

Data from Cascade Microtech's MicroCAT system can be directly included in the Data block.

Syntax:

`#include "file_name" Source=Cascade`

where *file_name* is the name of the data file, the keyword Source indicates that the data is supplied in an external format, and the keyword Cascade represents the Cascade Microtech's MicroCAT data format.

An example of a MicroCAT data file containing multi-device measurements:

```
NOTE MicroCAT sample data file.
NAME MicroCAT test run
DATE 31-JAN-89
SITE 5.000 0.000
BIAS Vds=3.000
BIAS Ids=0.0210
BIAS Vgs=0.000
S2RITABLE 6
45.000E6 1.0050 -0.0194 -3.662720 0.040894 -0.000080 0.000970 0.7965 -0.0076
 5.336E9 0.1462 -0.8846 -1.238892 2.359375 0.053711 0.051359 0.6062 -0.3692
10.627E9 -0.3997 -0.7135 0.280334 1.669678 0.089882 0.032745 0.4051 -0.5222
15.918E9 -0.5872 -0.5211 0.700195 0.978119 0.096882 0.012543 0.2379 -0.6282
21.209E9 -0.6796 -0.3917 0.756439 0.492950 0.096874 0.000851 0.0750 -0.7027
26.500E9 -0.7203 -0.3161 0.686371 0.204041 0.089085 -0.000854 -0.0869 -0.7499
ENDTABLE
SITE 4.000 1.000
SUBSITE 220 775
BIAS Vds=3.000
BIAS Ids=0.0394
BIAS Vgs=0.000
S2RITABLE 6
45.000E6 1.0041 -0.0181 -3.826660 0.048096 0.000105 0.000278 0.8454 -0.0050
 5.336E9 0.1950 -0.8993 -1.423218 2.491699 0.031494 0.034666 0.7165 -0.3345
10.627E9 -0.4012 -0.7502 0.282349 1.854736 0.058233 0.025927 0.5421 -0.5294
15.918E9 -0.6143 -0.5454 0.778137 1.085999 0.062037 0.013163 0.3824 -0.6673
21.209E9 -0.7236 -0.3884 0.869995 0.517761 0.064667 0.006695 0.2167 -0.7726
26.500E9 -0.7646 -0.2987 0.780518 0.180756 0.047050 0.013371 0.0516 -0.8484
ENDTABLE
SITE 3.000 2.000
SUBSITE 220 775
BIAS Vds=3.000
BIAS Ids=0.0464
BIAS Vgs=0.000
S2RITABLE 6
45.000E6 1.0040 -0.0170 -3.844971 0.035034 0.000180 0.000485 0.8460 -0.0059
 5.336E9 0.2629 -0.8913 -1.619507 2.518066 0.029911 0.034388 0.7216 -0.3345
10.627E9 -0.3558 -0.7808 0.193970 1.995117 0.057085 0.026552 0.5448 -0.5289
15.918E9 -0.5966 -0.5757 0.773621 1.211670 0.061361 0.013147 0.3797 -0.6645
21.209E9 -0.7109 -0.4142 0.917053 0.609497 0.063061 0.003658 0.2176 -0.7643
26.500E9 -0.7625 -0.3196 0.845215 0.249634 0.049517 0.011354 0.0548 -0.8461
ENDTABLE
```

The information contained in a MicroCAT file which is actually picked up and utilized by HarPE includes the bias data, namely Vds, Vgs and Ids, and the S2RITABLE (S parameters, 2-port, Real and Imaginary data Table) which contains the frequency, RS11, IS11, RS21, IS21, RS12, IS12, RS22 and IS22.

The different devices are identified by the SITE data. The SITE data given by MicroCAT has the form

SITE x y

HarPE combines the x and y coordinates into a single index as

$$k = 1000\ x + y$$

In the preceding example, the (x,y) coordinates of the three different devices are (5,0), (4,1) and (3,2). The corresponding indices given by HarPE are 5000, 4001 and 3002.

11.3 HarPE.Statistical.Extract Menu Option

The menu option HarPE.Statistical.Extract invokes multi-device parameter extraction as the first phase of the statistical modeling process (☞ see Chapter 1 for a general discussion on menus and menu operations).

The steps needed before you invoke HarPE.Statistical.Extract include:

- ▷ Supply multi-device measurements in the Data block (Section 11.2).
- ▷ Define a set of optimization variables in the Expression and/or Model block (see Chapters 4 and 5). The variables will be optimized to match the measured data from each device. The optimization will be repeated for every device, resulting in a sample of optimized values which can then be statistically postprocessed.
- ▷ Define the sweep ranges and specifications in the Specification block (see Chapter 9). The same set of specifications is applied to each and every device. If you omit the Specification block, all the measurements will be used in the optimization.
- ▷ Parse the circuit file by exiting the file editor (see Chapter 2). Because a circuit file for statistical modeling contains a large amount of measured data, parsing it may take considerably longer time than parsing other circuit files.

When you invoke HarPE.Statistical.Extract, a pop-up window will appear:

```
Optimizer:          L1
Reset Variables:    NO
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

Optimizer

Three choices are available for the Optimizer option in the pop-up window: L1, L2 and Huber. The ℓ_1, ℓ_2 and Huber optimizers are described in Chapter 9.

Reset Variables

The Reset variables option in the pop-up window allows you to control the starting point for the optimization of each device.

If you choose "YES" for this option, the optimization for each and every device will start from the initial values defined in the circuit file. In other words, the parameter extraction will have a common starting point for all the devices. The option name Reset variables comes from the fact that after each device optimization, the variables are "reset" to the common starting point.

If you choose "NO", then the solution of each device optimization will be used as the starting point for the next device. This can be beneficial if the initial values in the circuit file provide a very poor starting point.

Our recommendation is to first perform single-device modeling using the measurements from one of the devices (you may wish to choose a "typical" device in the sense that "wild" measurements are avoided). Then use that solution as the common starting point for statistical modeling (i.e., choose "YES" for the Reset variables option).

Display During Optimization

During the multi-device parameter extraction process, a bar graphics indicator of the progress as well as the following message are displayed in the upper-left-hand corner of the screen:

```
Optimization ... Press any key to interrupt
```

On the first line of the screen, the index of the device currently being optimized is displayed:

```
Statistical Modeling INDEX=k
```

where k is the current device index. The iteration count and the current value of the ℓ_1, ℓ_2 or Huber objective function are also displayed during the optimization, similar to those displayed for single-device optimization (see Chapter 9).

Interrupt the Optimization Process

While the optimization is in progress, you can interrupt it at any time by pressing a key (any key) on the keyboard. Then the following prompt will be displayed:

```
Terminate Operation (Y/<N>)?
```

and you can decide whether to terminate or to continue the optimization process.

Termination of Optimization

Once the multi-device parameter extraction is finished, a Sample block will be automatically created in the circuit file to store the optimized parameter values. You can examine the sample statistics through the graphical displays available from the HarPE.Statistical.Sample menu option (Section 11.4). You can also use the menu option HarPE.Statistical.Postprocess to generate a consolidated statistical model.

If you wish to restart the HarPE.Statistical.Extract option with a circuit file which is already optimized, you will have to delete the Sample block from the circuit file. Otherwise you will get an error message which says "`Statistical Model Already Exists`".

11.4 HarPE.Statistical.Sample Menu Option

The menu option HarPE.Statistical.Sample displays the histograms, scattering diagrams and cumulative probability distributions of the statistical sample contained in the Sample block of the circuit file.

☞ see Chapter 1 for a general discussion on menus and menu operations.

The Sample Block

When you invoke the HarPE.Statistical.Extract option, a Sample block is automatically created after the multi-device parameter extraction process is finished.

Syntax:

```
Sample
   FORMAT  INDEX  var_1  var_2  ...  var_n;
           k_1    x11    x12    ...   x1n
           k_2    x21    x22    ...   x2n
                  ...
                  ...

           k_m    xm1    xm2    ...   xmn
End
```

where *var_1*, *var_2*, ..., *var_n* represent the names of optimized variables,
k_1, *k_2*, ..., *k_m* represent the device indices,
and *xij* represents the value of the *j*th variable extracted from the *i*th device.

Example:

```
Sample
   FORMAT  INDEX  X1  EXTRINSIC2_1_CDS(PF)  FETC_1_TAU(PS);
             1   32.2      0.1035          3.522
             2   29.7      0.1105          3.266
             ...
             ...

           100   34.1      0.0983          3.471
End
```

where X1, EXTRINSIC2_1_CDS and FETC_1_TAU are the names of the variables and device indices are 1, 2, ..., 100 (i.e., measurements are supplied for 100 devices).

Each line within the Sample block contains the parameter values extracted from the measurements from one of the devices. The total number of lines (i.e., sets of extracted parameter values) is the same as the total number of different devices.

Names of the Variables

Variables defined in the Expression block are identified by the label names. For example,

```
Expression
   X1: ?0.25?;
   X2: ?5  50  100?;
   ...
End
```

These two variables are identified by the label names X1 and X2.

Circuit parameters that are directly defined as optimization variables in the Model block are not labelled. The name for such a variable is created using the following template:

element_k_parameter

where *element* identifies the element model which contains the variable, *k* is an index which counts the number of occurrences of the same element, and *parameter* identifies the parameter which is defined as the variable.

Example:

```
EXTRINSIC2  1 2 3 4 5  CDS=?0.1PF?  ... ;

FETC  1 2 3  TAU=?3PS?  ... ;
```

The two variables will be named EXTRINSIC2_1_CDS and FETC_1_TAU, which identify the variables with the parameter CDS of the element EXTRINSIC2 and the parameter TAU of the element FETC, respectively.

If an element is used more than once in the circuit model, then the index *k* identifies which one of them contains the variable. For example,

```
Model
   RES  1 2  R: ?50?;
   ...
   RES  5 6  R: ?2KOH?;
   ...
   RES  8 9  R: ?50?;
   ...
End
```

The variables will be named as RES_1_R, RES_2_R and RES_3_R, which associate them with the first, second and third RES elements, respectively, in the order of appearance.

The Sample block created by the HarPE.Statistical.Extract option can be directly used to define statistical parameters of sample distribution for Monte Carlo analysis (see Chapter 10), provided that such statistical parameters are defined by labels in the Expression block.

HarPE.Statistical.Sample.Histogram

Using the menu option HarPE.Statistical.Sample.Histogram, you can display the histograms of the statistical parameters. Histogram shows graphically the marginal distribution of a statistical parameter as represented by the data in the Sample block.

The histogram of a variable named GM is shown in Fig. 11.1 as an illustration.

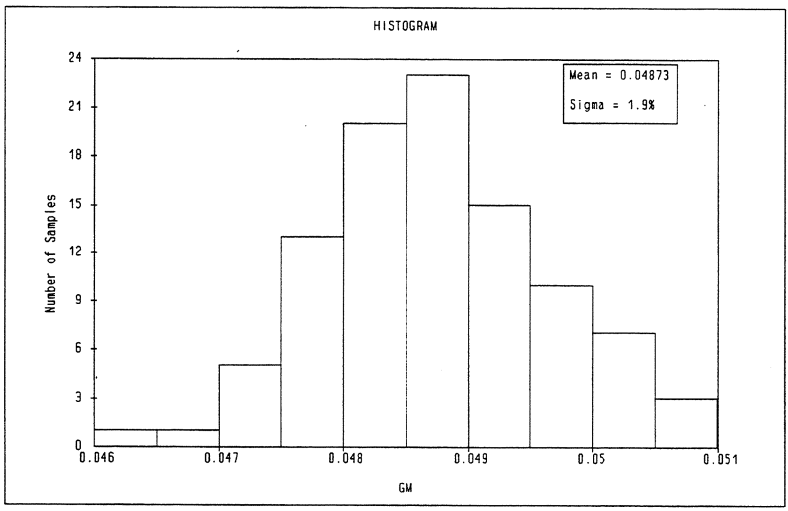


Fig. 11.1 Histogram of a statistical parameter.

When you invoke HarPE.Statistical.Sample.Histogram, a pop-up window will appear:

```
Zoom scale:        <TAB> to set scale
Variable:          GM
High truncation:
Low truncation:
Number of bins:    10
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

The Variable option in the pop-up window allows you to select any one of the variables contained in the Sample block.

The "Zoom scale" option allows you to change the display scale by specifying the parameters Ymin, yMax, Xmin and xmAx (☞ see Chapter 8 for further details on graphics zoom).

Number of Bins

You can specify the number of intervals (bins) into which the parameter values are divided. Available choices are 5, 10, 15 and 20 bins. The histogram shown in Fig. 11.1 consists of 10 bins.

High and Low Truncation Limits

Generally, the limits (minimum and maximum) of the values of the statistical parameter are not rounded figures. Consequently, the width of the bins is not a rounded figure either. For example, if the values range from 0.0462 to 0.05083, and the number of bins is selected as 10, then the width of the bins would be 0.000463. This can be inconvenient at times.

The "High truncation" and "Low truncation" options in the pop-up window allow you to define rounded limits for the bin ranges. For instance, if you round the low limit from 0.0462 to 0.046 and the high limit from 0.05083 to 0.051, and keep the number of bins as 10, then the width of the bins would be 0.0005.

The truncation limits of the histogram shown in Fig. 11.1 are 0.046 and 0.051.

The truncation limits are separate from the graphics zoom limits. The truncation limits determine the bin width, while the zoom limits determine the scale of the graphics.

Sample Means and Standard Deviations

The sample mean and standard deviation of the statistical parameter are super-imposed on the histogram. For example, in Fig. 11.1, the sample mean of GM is 0.04873 and the standard deviation is 1.9% (the standard deviation is denoted by "Sigma").

HarPE.Statistical.Sample.Cumulative

Using the menu option HarPE.Statistical.Sample.Cumulative, you can display the cumulative probability distributions of the statistical parameters as represented by the data in the Sample block (☞ see Chapter 10 for further details on cumulative probability distributions).

The cumulative probability distribution of a variable named GM is shown in Fig. 11.2.

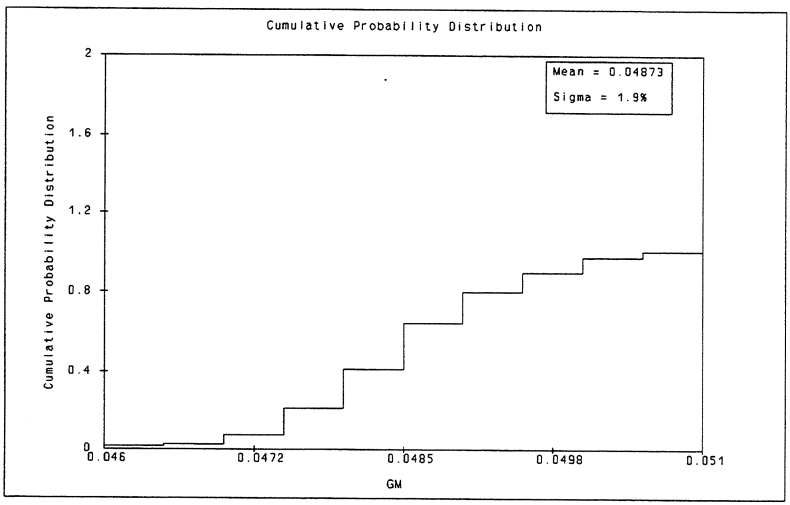


Fig. 11.2 Cumulative probability distribution of a statistical parameter.

When you invoke HarPE.Statistical.Sample.Cumulative, a pop-up window will appear:

```
Zoom scale:        <TAB> to set scale
Variable:           T
High truncation:
Low truncation:
Number of bins:    10
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

The Variable option in the pop-up window allows you to select any one of the variables contained in the Sample block.

The "Zoom scale" option allows you to change the display scale by specifying the parameters Ymin, yMax, Xmin and xmAx (☞ see Chapter 8 for further details on graphics zoom).

Number of Bins

You can specify the number of intervals (bins) into which the parameter values are divided. Available choices are 5, 10, 15 and 20 bins. The cumulative probability distribution shown in Fig. 11.2 consists of 10 bins.

High and Low Truncation Limits

Generally, the limits (minimum and maximum) of the values of the statistical parameter are not rounded figures. Consequently, the width of the bins is not a rounded figure either. For example, if the values range from 0.0462 to 0.05083, and the number of bins is selected as 10, then the width of the bins would be 0.000463. This can be inconvenient at times.

The "High truncation" and "Low truncation" options in the pop-up window allow you to define rounded limits for the bin ranges. For instance, if you round the low limit from 0.0462 to 0.046 and the high limit from 0.05083 to 0.051, and keep the number of bins as 10, then the width of the bins would be 0.0005.

The truncation limits of the cumulative probability distribution shown in Fig. 11.2 are 0.046 and 0.051.

The truncation limits are separate from the graphics zoom limits. The truncation limits determine the bin width, while the zoom limits determine the scale of the graphics.

Sample Means and Standard Deviations

The sample mean and standard deviation of the statistical parameter are super-imposed on the cumulative probability distribution. For example, in Fig. 11.2, the sample mean of GM is 0.04873 and the standard deviation is 1.9% (the standard deviation is denoted by "Sigma").

HarPE.Statistical.Sample.Scatter

Using the menu option HarPE.Statistical.Sample.Scatter, you can display scatter diagrams which show graphically the sample correlations between pairs of statistical parameters contained in the Sample block.

The scatter diagram between two statistical parameters, namely TAU and CDG, is shown in Fig. 11.3 as an illustration.

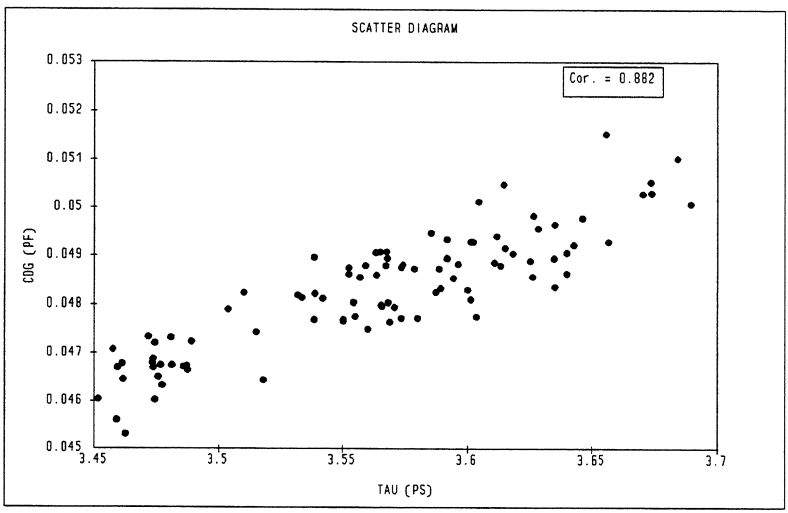


Fig. 11.3 Scatter diagram between two statistical parameters.

When you invoke HarPE.Statistical.Sample.Scatter, a pop-up window will appear:

```
Zoom scale:     <TAB> to set scale
X-axis:         TAU
Y-axis:         CDG
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

The X-axis and Y-axis options in the pop-up window allow you to select any pair of the variables contained in the Sample block for display.

The "Zoom scale" option allows you to change the display scale by specifying the parameters Ymin, yMax, Xmin and xmAx (☞ see Chapter 8 for further details on graphics zoom).

The correlation coefficient of the pair of statistical parameters is super-imposed on the scatter diagram. Fig. 11.3 shows graphically significant correlation between TAU and CDG, as confirmed by the value of the correlation coefficient (0.882).

Edit the Sample Block

Normally, the creation and processing of the Sample block is transparent to the user. However, you can edit the contents of the Sample block if so desired. For instance, after a visual inspection of the parameter variations, you may decide to delete some of the points that seem to have "wild" values (unreasonably large or small). This will alter the sample statistics and consequently the postprocessed statistical model.

User-Created Sample Block

You can even create your own Sample block, if you wish to apply the statistical display and postprocessing capabilities of HarPE to externally generated samples.

A user-created Sample block may contain a sample of values (raw data) of an arbitrary nature and with arbitrary names (labels) for the statistical parameters. Then you can use HarPE to display histograms, cumulative probability distributions and scatter diagrams, and to postprocess the sample to obtain a statistical model.

11.5 HarPE.Statistical.Postprocess Menu Option

The menu option HarPE.Statistical.Postprocess generates a consolidated statistical model from the sample contained in the Sample block and produces a back-annotated circuit file suitable for Monte Carlo analysis.

Statistical Sample

The statistical sample (a set of outcomes for the statistical parameters) to be postprocessed is expected from the Sample block, which is automatically generated by the menu option HarPE.Statistical.Extract through multi-device parameter extraction (Section 11.3).

The sample statistics can be visually examined through histograms and scatter diagrams displayed by the HarPE.Statistical.Sample menu option (Section 11.4).

The Sample block can be directly used to define statistical parameters of sample distribution for Monte Carlo analysis (see Chapter 10), provided that such statistical parameters are defined by labels in the Expression block.

Multi-Dimensional Normal Distribution

A multidimensional normal (Gaussian) distribution is used for the postprocessed statistical model. However, some of the statistical parameters may exhibit substantially non-Gaussian sample distributions. For such parameters, you may choose to keep their histograms as discrete approximations of the marginal density functions.

Pioneered by OSA (see, for instance, the technical reference [2] at the end of this chapter), this combined discrete/normal approach is capable of preserving the means, standard deviations, correlations and marginal distributions derived from the sample. It provides enhanced accuracy of the model while retaining the simplicity of the normal distribution.

General Syntax of Statistical Parameters

The general syntax of defining statistical parameters, distributions and correlation matrices are described in Chapter 10.

In general, you can directly define statistical parameters in the circuit file, as described in Chapter 10. The statistical postprocessing features discussed in this section provide a means of automatically and systematically generating statistical models suitable for Monte Carlo analysis.

Back-Annotated Circuit File

The HarPE.Statistical.Postprocess option automatically back annotates the statistical model to produce a circuit file suitable for HarPE's Monte Carlo analysis (see Chapter 10).

Syntax:

An optimization variable in the original circuit file as

name: ? ... ?

will be redefined as a statistical parameter as

name: *x* {Normal Sigma=σ% Correlation=CORMAT[*k*]}

where *name* represents a label name or an element parameter keyword, the keywords Normal, Sigma and Correlation identify normal distribution, standard deviation and correlations, respectively. x is the mean value, σ is the standard deviation as a percentage of x, and k is a row index of the correlation matrix named CORMAT.

For example, the variables in the following circuit file

```
Expression
   X1: ?30?;
End

Model
   CAP  1 2  C=?0.1PF?;

   FETC 5 6 7  TAU=?3PS? ...;
   ...
End

Sample
   FORMAT  INDEX X1 CAP_1_C(PF) FETC_1_TAU(PS);
       1   32.2   0.1035    3.522
       2   29.7   0.1105    3.266
       ...
     100   34.1   0.0983    3.471
End
```

will be redefined in the back-annotated circuit file as

```
Expression
   X1: 31.05 {Normal Sigma=4.3% Correlation=CORMAT[1]};
End

Model
   CAP  1 2  C=0.102PF {Normal Sigma=6.1% Correlation=CORMAT[2]};

   FETC  5 6 7  TAU=3.47PF {Normal Sigma=8.44% Correlation=CORMAT[3]} ...;
   ...
End

Statistics
   CORRELATION: CORMAT DIMENSION=3 FORMAT=FULL;
          1    0.4638      0.82
     0.4638         1    0.7997
       0.82    0.7997         1
End
```

The back-annotated circuit file is automatically loaded into the file editor. You can view and edit the file using any of the editing features described in Chapter 2.

Be cautioned that the back-annotated file is not automatically saved as a disk file. It exists only in the editor buffer and will disappear once you exit from the editor. You will have to explicitly save it to the disk if so desired.

The back-annotated file is kept in a separate memory buffer from the original circuit file (i.e., the file before postprocessing). Hence the editing performed on the back-annotated file will not affect the original circuit file. You can think of the back-annotated file as an output from HarPE, just like the numerical display data file described in Chapter 8.

The Data, Specification and Sample blocks are removed from the back-annotated file. This is because they have served their purposes in the modeling stage and are no longer relevant to the postprocessed file.

Correlation Matrix in the Statistics Block

In the back-annotated file a new Statistics block is created. It contains the correlation matrix which represents pair-wise correlations between the parameters of the postprocessed statistical model.

The correlation matrix in the Statistics block has a name as an identifier, a DIMENSION keyword which defines the size of the matrix and an optional FORMAT keyword.

The general definition of correlation matrices in the Statistics block is described in detail in Chapter 10.

The correlation matrix generated by postprocessing is assigned the name "CORMAT", its dimension is the same as the number of statistical parameters in the postprocessed model, and its format is always "FULL".

The individual statistical parameters are linked to the correlation matrix through an index. For example,

```
X1: 31.05 (Normal Sigma=4.3% Correlation=CORMAT[1]);
```

indicates that the correlation coefficients between X1 and the other statistical parameters are contained in the first row (first column) of the correlation matrix.

Keep Histograms as DDFs

When you invoke the HarPE.Statistical.Postprocess option, the histogram of each of the parameters being postprocessed will be displayed, such as the one shown in Fig. 11.4.

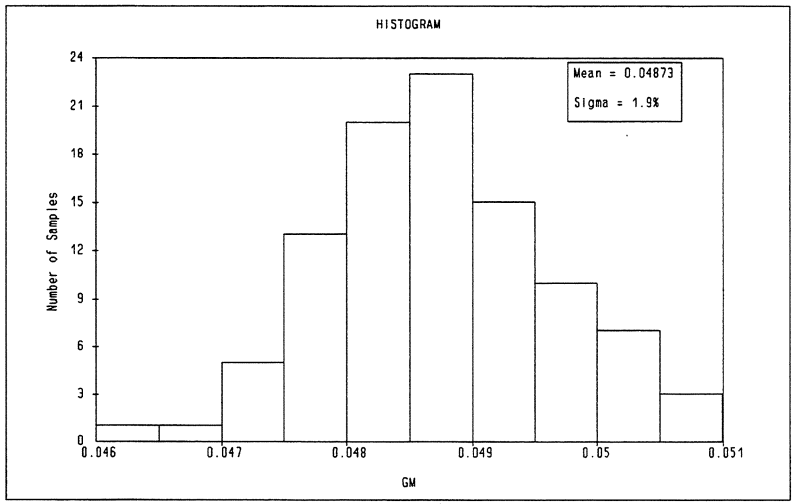


Fig. 11.4 Histogram of a statistical parameter.

The histogram helps you to judge how close the sample distribution of the parameter is to a normal distribution. The following prompt is displayed after each histogram:

```
Keep the histogram as an approximate DDF (Y/<N>):
```

Press <Y> if you judge the histogram to be substantially different from that of a normal distribution, then the histogram will be kept as a discrete approximation of the marginal density function of the statistical parameter.

Example:

```
X1: 31.04 {Normal Sigma=5% Correlation=CORMAT[1] DDF=7 16 29 37 9 2};
```

where the keyword DDF stands for "discrete density function". DDF is defined by n integers, where n is the number of bins in the histogram, and the integers are the numbers of outcomes in the corresponding bins.

☞ The definition and use of DDF are discussed in more general terms in Chapter 10.

Postprocess User-Supplied Samples

The HarPE.Statistical.Postprocess option is primarily designed to work in conjunction with the HarPE.Statistical.Extract option as an integrated statistical modeling system.

You can, however, utilize HarPE's capabilities to postprocess arbitrary statistical samples. To do this, you must create a Sample block in the circuit file to contain the statistical data. Furthermore, you must create in the Expression block a set of optimization variables as place-holders for back annotation. These variables must have label names identical to the names contained in the Sample block.

Example:

```
Expression
   VAR1: ?10?;
   VAR2: ?10?;
   XV: ?10?;
End
...

Sample
   FORMAT  INDEX  VAR1  VAR2    XV;
              1   32.2  0.1035  -131
              2   29.7  0.1105  -140
                  ...
                  ...
            100   34.1  0.0983  -127
End
```

The labels VAR1, VAR2 and XV in the Expression correspond to the names contained in the Sample block and serve as place-holders for back annotation.

After you utilize HarPE to postprocess this example, the back-annotated file may appear as

```
Expression
   VAR1: 31.65 {Normal Sigma=7.2% Correlation=CORMAT[1]};
   VAR2: 0.0998 {Normal Sigma=6.03% Correlation=CORMAT[2]};
   XV: -133.9 {Normal Sigma=2.77% Correlation=CORMAT[3]};
End
...

Statistics
   CORRELATION: CORMAT DIMENSION=3 FORMAT=FULL;
   ...
   ...
End
```

11.6 HarPE.Statistical.Data Menu Option

The menu option HarPE.Statistical.Data displays the histograms and cumulative probability distributions of statistical measurement data contained in the Data block. To view the histograms and CPDs of the data displayed together with those of the circuit model see Chapter 10.

HarPE.Statistical.Data.Histogram

Using the menu option HarPE.Statistical.Data.Histogram, you can display the histograms of the measurement data.

When you invoke HarPE.Statistical.Data.Histogram, a pop-up window will appear:

```
Zoom scale:          <TAB> to set scale
Gate bias:           -1.74V
Drain bias:          4V
Input power:         n/a
Frequency:           14GHZ
High truncation:
Low truncation:
Number of bins:      15
data Point:          4 of 10
Response:            IS11
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

Select a Data Point for Display

The pop-up window allows you to select a data point for display from those contained in the Data block.

To illustrate, suppose that the Data block contains the following measurements:

```
PARAMETER INDEX=1 VG=-1.74V VD=4V;
FORMAT FREQ RS11 IS11 RS21 IS21 RS12 IS12 RS22 IS22;
    2  0.7815 -0.5718 -2.87   1.754  0.01673  0.03549  0.5183 -0.2733
    6 -0.0363 -0.8518 -0.5194 2.372  0.07108  0.04199  0.2542 -0.4683
   10 -0.4835 -0.6226  0.621  1.605  0.09085  0.01503  0.0227 -0.5372
   14 -0.6747 -0.365   0.9452 0.8582 0.0879  -0.00667 -0.19   -0.5467
   18 -0.7532 -0.1425  0.9245 0.3272 0.07678 -0.01788 -0.3916 -0.4998
PARAMETER INDEX=1 VG=-3.1V VD=4V;
FORMAT FREQ RS11 IS11 RS21 IS21 RS12 IS12 RS22 IS22;
    2  0.6483 -0.7165 0.00148 0.00723 0.00148 0.00723 -0.8093 0.04526
    6 -0.328  -0.8011 0.00697 0.01679 0.00697 0.01679 -0.8003 0.1368
   10 -0.6873 -0.4572 0.01229 0.02409 0.01229 0.02409 -0.7823 0.226
   14 -0.7944 -0.1724 0.01842 0.03039 0.01842 0.03039 -0.7559 0.3119
   18 -0.8094  0.0476 0.02551 0.03538 0.02551 0.03538 -0.7219 0.3935

...
```

The measurements include 2 bias points and 5 frequencies per bias point, providing a total of 10 data points. You can select any of these points for display through the "data Point" option in the pop-up window. The corresponding bias voltages, frequency and input power level (if applicable) are automatically updated in the pop-up window as you toggle through the different data points.

Select a Response for Display

Once you have selected one of data points, you can then select a response from those available. In the preceding example, for instance, you can select from RS11, IS11, RS21, IS21, RS12, IS12, RS22 and IS22.

The histogram of IS11 at a selected data point is illustrated in Fig. 11.5.

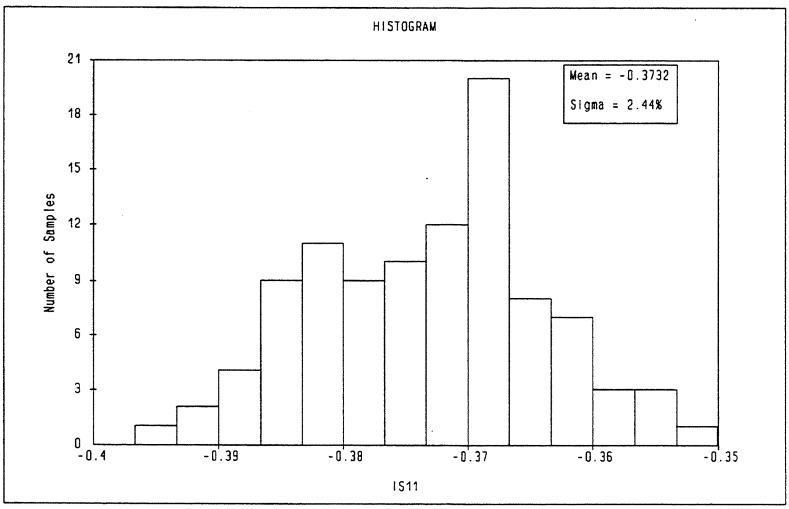


Fig. 11.5 Histogram of measured data.

Number of Bins

You can specify the number of intervals (bins) into which the data values are divided. Available choices are 5, 10, 15 and 20 bins. The histogram shown in Fig. 11.5 consists of 15 bins.

Graphics Zoom

By default, the histogram display is scaled according to the data value range. You can change the display scale using the "Zoom scale" option in the pop-up window by specifying the parameters Ymin, yMax, Xmin and xmAx (☞ see Chapter 8 for further details).

High and Low Truncation Limits

Generally, the limits (minimum and maximum) of the data values are not rounded figures. Consequently, the width of the bins is not a rounded figure either. For example, if the values range from 0.0462 to 0.05083, and the number of bins is selected as 10, then the width of the bins would be 0.000463. This can be inconvenient at times.

The "High truncation" and "Low truncation" options in the pop-up window allow you to define rounded limits for the bin ranges. For instance, if you round the low limit from 0.0462 to 0.046 and the high limit from 0.05083 to 0.051, and keep the number of bins as 10, then the width of the bins would be 0.0005.

The truncation limits of the histogram shown in Fig. 11.5 are -0.4 and -0.35.

The truncation limits are separate from the graphics zoom limits. The truncation limits determine the bin width, while the zoom limits determine the scale of the graphics.

Sample Means and Standard Deviations

The sample mean and standard deviation of the measured data are super-imposed on the histogram. For example, in Fig. 11.5, the sample mean of IS11 at the selected data point is -0.3732 and the standard deviation is 2.44%.

HarPE.Statistical.Data.Cumulative

Using the menu option HarPE.Statistical.Data.Cumulative, you can display the cumulative probability distributions of the measurement data.

When you invoke HarPE.Statistical.Data.Cumulative, a pop-up window will appear:

```
Zoom scale:          <TAB> to set scale
Gate bias:           -1.74V
Drain bias:          4V
Input power:         n/a
Frequency:           2GHZ
High truncation:
Low truncation:
Number of bins:      10
data Point:          1 of 5
Response:            RS11
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

Select a Data Point for Display

The pop-up window allows you to select a data point for display from those contained in the Data block.

To illustrate, suppose that the Data block contains the following measurements:

```
PARAMETER INDEX=1 VG=-1.74V VD=4V;
FORMAT FREQ RS11 IS11 RS21 IS21 RS12 IS12 RS22 IS22;
    2  0.7986 -0.5675 -2.6758  1.4424  0.0168  0.0391  0.4301 -0.2241
    6 -0.0318 -0.8986 -0.6043  2.1909  0.0766  0.0454  0.1804 -0.4522
   10 -0.5474 -0.6685  0.5007  1.5601  0.0916  0.0128 -0.0189 -0.5243
   14 -0.7901 -0.3598  0.8397  0.8883  0.0759 -0.0072 -0.1762 -0.5567
   18 -0.8866 -0.0641  0.8440  0.4078  0.0523 -0.0073 -0.3179 -0.5651

...
```

The measurements include 1 bias points and 5 frequencies, providing a total of 5 data points. You can select any of these points for display through the "data Point" option in the pop-up window. The corresponding bias voltages, frequency and input power level (if applicable) are automatically updated in the pop-up window as you toggle through the different data points.

Select a Response for Display

Once you have selected one of data points, you can then select a response from those available. In the preceding example, for instance, you can select from RS11, IS11, RS21, IS21, RS12, IS12, RS22 and IS22.

The cumulative probability distribution of RS11 at a selected data point is illustrated in Fig. 11.6.

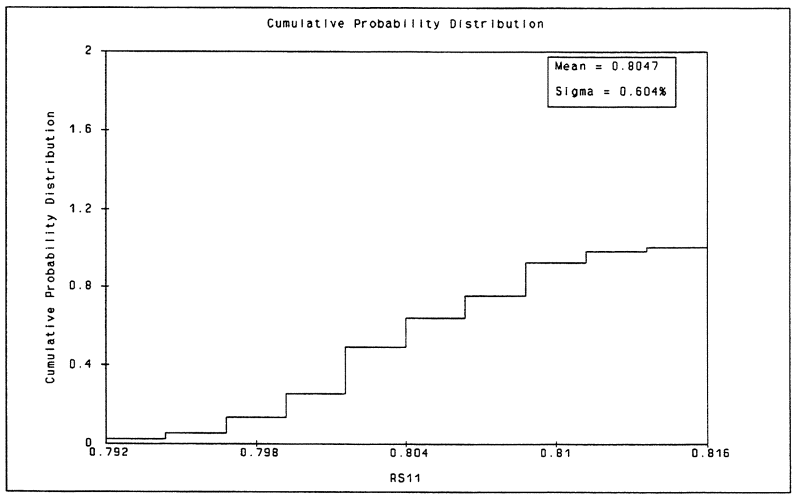


Fig. 11.6 Cumulative probability distribution of measured data.

Number of Bins

You can specify the number of intervals (bins) into which the data values are divided. Available choices are 5, 10, 15 and 20 bins. The cumulative probability distribution shown in Fig. 11.6 consists of 10 bins.

Graphics Zoom

By default, the cumulative probability distribution display is scaled according to the data value range. You can change the display scale using the "Zoom scale" option in the pop-up window by specifying the parameters Ymin, yMax, Xmin and xmAx (☞ see Chapter 8 for further details).

High and Low Truncation Limits

Generally, the limits (minimum and maximum) of the data values are not rounded figures. Consequently, the width of the bins is not a rounded figure either. For example, if the values range from 0.7926 to 0.8153, and the number of bins is selected as 10, then the width of the bins would be 0.00227. This can be inconvenient at times.

The "High truncation" and "Low truncation" options in the pop-up window allow you to define rounded limits for the bin ranges. For instance, if you round the low limit from 0.7926 to 0.792 and the high limit from 0.8153 to 0.816, and keep the number of bins as 10, then the width of the bins would be 0.0024.

The truncation limits of the cumulative probability distribution shown in Fig. 11.6 are 0.792 and 0.816.

The truncation limits are separate from the graphics zoom limits. The truncation limits determine the bin width, while the zoom limits determine the scale of the graphics.

Sample Means and Standard Deviations

The sample mean and standard deviation of the measured data are super-imposed on the cumulative probability distribution. For example, in Fig. 11.6, the sample mean of RS11 at the selected data point is 0.8407 and the standard deviation is 0.604%.

11.7 HarPE.Statistical.Match Menu Option

The menu option HarPE.Statistical.Match invokes direct statistical modeling by matching the cumulative probability distributions or the histograms of the model responses to those of the measured data.

The steps needed before you invoke HarPE.Statistical.Match include:

- Supply multi-device measurements in the Data block (Section 11.2).
- Define a set of optimization variables in the Expression and/or Model block (see Chapters 4 and 5). The variables will be optimized to match the statistics of model responses to those of the measured data.
- Define the Monte Carlo sweep in the MonteCarlo block (☞ see Chapter 10) according to the measured responses supplied in the Data block.
- Parse the circuit file by exiting the file editor (see Chapter 2). Because a circuit file for statistical modeling contains a large amount of measured data, parsing it may take considerably longer time than parsing other circuit files.

Optimization Variables

The optimization variables for direct statistical modeling can be statistical parameters such as the mean values and standard deviations of normal distributions, the nominal values and tolerances of uniform distributions. The optimization variables can also include non-statistical parameters.

In general, you can assign any initial values to the optimization variables. However, a good starting point can significantly reduce the computation time required for optimization. A two stage method can be used in statistical modeling.

First, use the multi-device parameter extraction and postprocessing approach with a small number of devices to obtain an initial statistical model. Then, use the direct statistical modeling technique with a reasonable number of outcomes to optimize the model to increase its accuracy.

Statistical Parameters

Uniform, exponential, lognormal and normal distributions (☞ see Chapter 10) can be assigned to statistical parameters. This provides more flexibility for statistical modeling. For instance, you can assign normal distributions to the physical parameters such as the FET gate length, channel thickness and doping density, and assign uniform distributions to linear elements such as resistors, capacitors and inductors.

Monte Carlo Simulation

In direct statistical modeling, the error functions for optimization are defined by comparing the statistics of the model responses with the statistics of the measurement data. Therefore, the MonteCarlo block must be defined in order to obtain the statistics of the model responses.

The sweep parameters and responses in the MonteCarlo block should be defined according to the measurements given in the Data block. In other words, the responses obtained by Monte Carlo simulation should correspond to the measured responses.

The number of outcomes defined in the MonteCarlo block can be different from the number of data sets in the measurement data.

☞ For details of the MonteCarlo block please refer to Chapter 10.

Multi-Device Measurements

Multi-device measurements are required for statistical modeling. Unlike the indirect statistical modeling technique where the supplied data can be taken at different bias conditions and at different frequencies for each data set, direct statistical modeling requires the measurements taken at the same bias conditions and at the same frequencies for each device.

☞ The syntax of multi-device measurement data is described in Chapter 11.

Pop-Up Window

When you invoke HarPE.Statistical.Match, a pop-up window will appear:

```
Optimizer:                L1
Goal:                     Histogram matching
Number of iterations:     30
Accuracy of solution:     0.0001
number of Bins:           50
Ready to go:

<ENTER> = go  or  <ESC> = cancel
Select item with <UP>/<DOWN>
<F1> help
```

Optimizer

The choices available for the Optimizer option in the pop-up window are L1, L2 and Huber. The choice of optimizer also implies the use of the corresponding ℓ_1, ℓ_2 or Huber objective function (see Section 9.5).

Goal

The choices available for the Goal option in the pop-up window are "CPD matching" and "Histogram matching".

When you select "CPD matching", the model will be optimized by matching the cumulative probability distributions of the model responses to those of the measured data.

When you select "Histogram matching", the model will be optimized by matching the histograms of the model responses to those of the measured data.

Number of Iterations

This option in the pop-up window allows you to limit the maximum number of iterations. The optimizer will stop once this limit is reached even if a solution within the desired accuracy has not been found. If this happens, the best set of values of the variables (in terms of minimizing the objective function) is presented as the solution. Available values for this option are 1, 2, 5, 10, 15, 30, 50, 100 and 999.

Accuracy of Solution

This option allows you to specify the desired accuracy of the solution. The optimization will stop when the step size suggested by the algorithm is smaller than the specified accuracy. The step size is relative to the norm of the variable vector. Available values for this option are 0.01, 0.001, 0.0001, and 1e-5.

Number of Bins

When you select "Histogram matching" for the Goal option in the pop-up window, you are required to specify the number of bins of the histograms to be matched by entering an integer number from the keyboard.

The number of bins determines the accuracy of the optimized statistical model. The larger the number of bins the more accurate is the model. It may, however, significantly increase the computational time required for optimization. The user should select a suitable number according to the experience.

Threshold

When you select Huber in the Optimizer option in the pop-up window, you are required to specify the threshold for calculating the Huber objective function. The default value is 0.25.

Interrupt the Optimization Process

During optimization, a bar graphics indicator of the progress as well as the following message are displayed in the upper-left-hand corner of the screen:

```
Optimization ...  Press any key to interrupt
```

While the optimization is in progress, you can interrupt it at any time by pressing a key (any key) on the keyboard. Then the following prompt will be displayed:

```
Terminate Operation (Y/<N>)?
```

and you can decide whether to terminate or to continue the optimization process.

Display During Optimization

During optimization, HarPE displays the iteration count and current value of the ℓ_1, ℓ_2 or Huber objective function on the screen, such as

```
Iteration   1/30  L1 Error=2.34101
Iteration   2/30  L1 Error=2.26455
Iteration   3/30  L1 Error=2.11538
Iteration   4/30  L1 Error=1.82175
Iteration   5/30  L1 Error=1.3006
Iteration   6/30  L1 Error=0.837255
Iteration   7/30  L1 Error=0.592952
Iteration   8/30  L1 Error=0.404969
Iteration   9/30  L1 Error=0.290885
Iteration  10/30  L1 Error=0.266347
Iteration  11/30  L1 Error=0.264805
Iteration  12/30  L1 Error=0.264814
Solution  L1 Error=0.264805
```

Of the iteration count, the first number is the current iteration and the second one is the maximum number of iterations specified. For example, 10/30 means that the current iteration is the 10th out of a maximum of 30.

It is sometimes possible to observe the objective function value getting larger than that of a previous iteration. This is not unusual. It is a part of the optimization algorithm.

Termination of Optimization

Optimization is terminated under one of the following conditions:

- Interrupted and terminated by the user.
- A local minimum of the objective function is found.
- The specified accuracy is reached, i.e., the step size suggested by the algorithm is smaller than the specified accuracy.
- The maximum number of optimization iterations is reached.
- A severe error condition is encountered such as a floating-point divide by zero error.

A message indicating the cause of termination will be displayed. Regardless of the cause the solution will always be the best result obtained up to the point of termination. The circuit file is automatically updated with the solution, i.e., the variables are updated with their optimized values.

The circuit file that is updated with the solution is actually the copy kept by HarPE in the computer's memory. The original disk file is not changed. If you wish to save the updated file to the disk, do so with the file editor (☞ Chapter 2).

11.8 Technical References

[1] J.W. Bandler and S.H. Chen, "Circuit optimization: the state of the art", *IEEE Trans. Microwave Theory Tech.*, vol. 36, 1988, pp. 424-443.

[2] J.W. Bandler, R.M. Biernacki, S.H. Chen, J. Loman, M. Renault and Q.J. Zhang, "Combined discrete/normal statistical modeling of microwave devices," *Proc. European Microwave Conf.* (Wembley, England), 1989, pp. 205-210.

[3] J.W. Bandler, R.M. Biernacki, S.H. Chen, J. Song, S. Ye and Q.J. Zhang, "Statistical modeling of GaAs MESFETs," *IEEE Int. Microwave Symp. Dig.* (Boston, MA), 1991, pp. 87-90.

[4] J.W. Bandler, Q.J. Zhang, J. Song and R.M. Biernacki, "Yield optimization of nonlinear circuits with statistically characterized devices," *IEEE Int. Microwave Symp. Dig.* (Long Beach, CA), 1989, pp. 649-652.

[5] J.W. Bandler, S. Ye, Q. Cai, R.M. Biernacki and S.H. Chen, "Predictable yield-driven circuit optimization," *IEEE Int. Microwave Symp. Dig.* (Albuquerque, NM), 1992.

[6] J.W. Bandler, W. Kellermann and K. Madsen, "A nonlinear ℓ_1 optimization algorithm for design, modeling and diagnosis of networks", *IEEE Trans. Circuits and Systems*, vol. CAS-34, 1987, pp. 174-181.

[7] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes. The Art of Scientific Computing*. Cambridge University Press: Cambridge, 1986.

[8] J.W. Bandler, R.M. Biernacki, Q. Cai and S.H. Chen, "A novel approach to statistical modeling using cumulative probability distribution fitting," *IEEE MTT-S Int. Microwave Symp. Dig.* (San Diego, CA), 1994, pp. 385-388.

12

Operation Control

12

Operation Control

12.1 Overview

This chapter describes the features for monitoring and controlling the operations of HarPE.

Message files contain the on-line User's Manual and on-line help messages which can be displayed upon request.

ID files allow you to run multiple copies of HarPE simultaneously.

A **log file** is created to maintain a record of the operations performed by HarPE during each session.

Color map files can be defined to accommodate permanent and temporary modifications to the display color schemes.

A **user list file** can be created to maintain a list of users authorized to use HarPE in a multi-user environment.

HarPE can operate in one of the following modes:

- Local interactive mode. This is the default operation mode for HarPE. In this mode, HarPE receives user inputs from and displays graphics on the console of the computer on which HarPE is installed.
- Remote interactive mode. In this mode, HarPE is controlled from a remote X terminal through a network. HarPE still runs on the local host, but the inputs and outputs are redirected to the remote X terminal.
- Macro mode. In this mode, HarPE receives instructions from a macro command file instead of the keyboard. It can operate in an automated, unattended fashion.
- Silent mode. In this mode, all the screen displays are suppressed. The simulation results are saved in disk files. Also, silent mode implies macro mode, i.e., HarPE receives instructions from a macro command file instead of the keyboard.
- Background mode. HarPE runs in a background process and in silent mode.

12.2 Message Files

The on-line User's Manual and the on-line help message files are contained in the harpemsg subdirectory within the OSA Installation Directory (☞ Chapter 1).

The on-line help messages are stored in the "*.hlp" files. If you press <F1> when HarPE is running, a help message appropriate to the context will be displayed.

The on-line User's Manual is stored in the "*.man" files. You can view the on-line User's Manual from within the HarPE editor.

12.3 ID Files

In a multi-tasking environment under UNIX, it is possible to run multiple copies of HarPE simultaneously in separate processes.

Depending on the HarPE license you have acquired, you may be able to run multiple copies of HarPE simultaneously.

When multiple copies of HarPE are running at the same time, each copy identifies itself by means of an associated ID file.

When you start HarPE, an ID file will be created in the "/tmp" directory. The name of the ID file is HarPE-*i*, where *i* is an index between 0 and 9. In other words, when multiple copies of HarPE are running at the same time, the first copy is identified by the ID file HarPE-0, the second copy by HarPE-1, and so on.

12.4 Log Files

Each time you run HarPE, a log file is created which chronologically records the major operations of the program. The messages generated by the program, including any error messages, are recorded in the log file. If optimization is performed, the progress of the optimization is also recorded.

Log files are ASCII text files. They are given names like harpe_*i*.log, where *i* is the ID file index described in Section 12.3. If there are multiple copies of HarPE running at the same time, each one will have its own log file.

Example:

```
HarPE Version 2.0 for Sun workstations
 (c) Optimization Systems Associates Inc. 1995
 Mon Aug 21 16:29:47 1995
16:30:04 Read in File /harpe/my_file1.ckt
16:30:08 Parsing Circuit File ...
16:30:08 File Parsing Completed
16:30:09 Small-Signal Simulation ...  Press any key to interrupt
16:30:10 DC Simulation ...  Press any key to interrupt
16:30:11 HB Simulation ...  Press any key to interrupt
16:30:24 Ready for Display
16:30:30 Read in File /harpe/my_file2.ckt
16:30:31 Parsing Circuit File ...
16:30:31 File Parsing Completed
16:30:32 Small-Signal/DC L1 Optimization
16:30:33 Iteration   1/30  L1 Error=12.3456
16:30:35 Iteration   2/30  L1 Error=...
   .
   .

16:30:50 Iteration  16/30  L1 Error=0.0707176
16:30:50 Solution  L1 Error=0.0707176
16:30:50 Optimization Completed
16:30:50 Elapsed Real Time: 00:00:18   CPU Time 00:00:17
16:30:51 Save File /harpe/my_file3.ckt
   .
```

12.5 Color Map Files

The display colors of HarPE can be redefined by color map files. Color map files are ASCII text files and contain RGB intensity definitions and color label definitions.

RGB Intensity Definitions

HarPE uses 16 basic colors. Each basic color is defined by a unique combination of RGB (Red, Green, Blue) intensities. The default intensities are listed in Table 12.1.

TABLE 12.1 DEFAULT RGB INTENSITIES

Color	R	G	B
Black	0	0	0
Red	255	0	0
Green	0	230	0
Blue	0	0	230
LightBlue	0	255	255
Yellow	245	245	0
Pink	255	0	255
White	255	255	230
Cream	240	255	190
DarkGreen	30	155	0
MidCream	255	240	180
DarkRed	210	20	0
WhiteText	255	255	255
DarkBlue	60	60	200
DarkCream	240	240	170
Brown	150	90	0

The three RGB intensity indices must be integers between 0 and 255. A larger value represents a higher intensity, and the value 255 represents the full intensity.

You can redefine the RGB intensities of the colors in a color map file.

Syntax:

k1 k2 k3 color

where *k1*, *k2* and *k3* are the new RGB intensities for *color*.

You can redefine as many colors as you wish, one color on each line of the color map file. You cannot create new colors (i.e., the names of the colors are fixed).

Example:

```
200   200   200   White
```

which redefines the color White into a shade of grey.

Color Labels

A color label represents the foreground or background color for a specific display in HarPE. Table 12.2 lists the color labels and the default color assignments.

TABLE 12.2 COLOR LABELS

Label	Default Color
TextBackground1	WhiteText
TextBackground2	Cream
TextNormalForeground	Black
TextHighlightForeground	Red
MenuForeground1	Black
MenuForeground2	WhiteText
MenuBackground1	Yellow
MenuBackground2	DarkGreen
MenuHighlightForeground	DarkRed
TimeForeground	WhiteText
TimeBackground	Pink
IndicatorForeground	Green
IndicatorBackground	DarkRed
HelpTitleForeground	Yellow
GraphicsBackground	Black
GraphicsTitleForeground	White
GraphicsLabelForeground	White
GraphicsAxisForeground	White
GraphicsGridForeground	Brown
GraphicsSpecForeground	Yellow
GraphicsSingleCurveForeground	Green
GraphicsMultiCurve1Foreground	Yellow
GraphicsMultiCurve2Foreground	Red
GraphicsMultiCurve3Foreground	White
GraphicsMultiCurve4Foreground	Green
GraphicsMultiCurve5Foreground	Pink
GraphicsMultiCurve6Foreground	LightBlue
WindowFrameForeground	DarkGreen

For example, the screen editor displays text in the primary window in the foreground color specified by the label TextNormalForeground with the background color specified by the label TextBackground1. By default, the basic colors Black and WhiteText are assigned to these two labels, respectively. Therefore, the text is displayed as Black on WhiteText.

The color assignments of the color labels can be changed in a color map file.

Syntax:

color_label color

where *color_label* represents one of the color labels listed in Table 12.2, and *color* represents one of the colors listed in Table 12.1.

Example:

```
GraphicsGridForeground   Black
```

which reassign the color label GraphicsGridForeground to the color Black. As a consequence, the grid lines will be invisible on the graphics display, if the color label GraphicsBackground is also Black (which is the default).

You can change as many color labels as you wish, one color label on each line of the color map file.

RGB Colors and Color Labels

When you change an RGB color, all the color labels which are assigned that color will be affected.

For example, if the RGB intensities of the color Yellow are modified, then all the color labels which are assigned the color Yellow will be affected.

One of the reasons for redefining the colors is to print the graphics on a single-color printer which translates colors into grey scales. You may have to experiment with your printer to get the desired results.

Comments in Color Map Files

Color map files may contain comments. Similar to the comments in input files, any text following an exclamation mark "!" is treated as a comment until the end of the line.

Example:

```
! change color labels

TextNormalForeground   Blue     ! default is black
TextBackground1        Yellow   ! default is WhiteText
```

The Default Color Map File

The default color map is the file "colorxs.map" in the harpemsg subdirectory within the OSA Installation Directory.

You can modify colorxs.map directly if you would like to make permanent changes to the color map for *all* users of HarPE.

You can also create other color map files under different names and then instruct HarPE to use such a color map file instead of colorxs.map.

Specify a Color Map File when Starting HarPE

You can specify a color map file when you start HarPE.

Command:

`% harpe -c` *color_map*

where "-c" is the run-time option for specifying a color map file,
and *color_map* is the name of the desired color map file.

Switch Color Map Files On-Line

You can switch color maps within the HarPE file editor using the function key <Ctrl-Y>. You will be prompted for the color map file name:

```
Color map file:
```

12.6 User List File

If you wish to restrict the use of HarPE to a specific group of users on your system, you can create an ASCII file named "userlist.osa" in the harpemsg subdirectory within the OSA Installation Directory. List in this file the login names of all the authorized users.

When you start HarPE, it will look for the file userlist.osa in the harpemsg subdirectory.

If the file userlist.osa does not exist, any user who has access to the file harpe will be able to run it.

If the file userlist.osa is found, HarPE will check the user's login name against the list contained in the file. If no match is found, the user will not be able to run HarPE.

For example, if the file userlist.osa contains

```
john
root
mary
```

then a user whose login name is not "john", "root" or "mary" will not be able to run HarPE.

If you create userlist.osa, you should also make sure that it has the proper file protection against unauthorized modification.

12.7 X Terminals

X-Windows allows an application to run on one computer and display the graphics on another computer (which is set up as an X terminal).

From an X terminal through a network, you can operate HarPE in the remote interactive mode. The mathematical operations ("number crunching") are still performed on the host computer on which HarPE is installed, but the graphical user interface is displayed on the remote X terminal.

Only the computers authorized in the License Agreement can be used as X terminals to operate HarPE. Attempted operation from unauthorized computers will produce an "access denied" message. Also, some X-Windows managers may have their own host access control mechanism. Check with your system administrator.

For example, suppose that you wish to use an authorized computer with the host name "charlie" as an X terminal to operate HarPE. The following steps are involved:

1 Login to "charlie", start X-Windows and open an X terminal.

2 Login through a network to the computer on which HarPE is installed.

3 You can redirect the X-Windows display by setting the DISPLAY environment variable (see Chapter 1). Alternatively, you can use the HarPE run-time option "-d" as

```
% harpe -d charlie:0.0
```

Command:

`% harpe -d` *host_name*`:0.0`

where "-d" is the run-time option for redirecting the X-Windows display and *host_name* must be the host name of an authorized computer.

12.8 Macro Command Files

In the interactive mode, you control HarPE by means of the keyboard and the mouse. In the macro mode, HarPE reads the instructions from a macro command file.

Macro command files are ASCII text files. Their contents can be interpreted by HarPE into appropriate key codes as if they were typed on the keyboard. This allows you to operate HarPE in an automated, unattended fashion.

The entries of macro command files can be categorized as literal strings, key codes and control codes.

There are similarities and differences between the macro mode and the editor macros described in Chapter 2. Both macro features serve to replace the keyboard in supplying a sequence of instructions. But, editor macros are recorded in memory and limited to the functions within the screen editor. The macro mode described here relies on a disk file (macro command file) and applies to the general operations of HarPE, including those of the screen editor.

Literal Strings

Literal strings represent character strings exactly as they would appear if typed from the keyboard. Literal strings must be enclosed in quotation marks.

For example,

```
"/user/john/demo01.ckt"
```

is a literal string.

Key Codes

Key codes represent function keys. Each function key is represented by a unique key code. The set of key codes for macro command files are listed in Table 12.3.

Key codes must not be enclosed in quotation marks. For example, the key code LEFT represents the <LEFT> arrow key, but if it is enclosed within quotation marks "LEFT" would be interpreted as a literal string.

TABLE 12.3 MACRO KEY CODES

Key Code	Key Represented
BACKSPACE	<BACKSPACE>
CHG_PATH	<F5>
CLEAR_FILE	<Ctrl-N>
COLOR_MAP	<Ctrl-Y>
DEL	<DEL>
DEL_WORD	<Ctrl-W>
DOWN	<DOWN>
EDIT	<F6>
END	<Ctrl-HOME>
ENTER	<ENTER>
ESC	<ESC>
EXIT	<F7>
FILE_END	<Ctrl-DOWN>
FILE_TOP	<Ctrl-UP>
HELP	<F1>
HOME	<HOME>
INS	<INS>
LEFT	<LEFT>
LINE_DEL	<Ctrl-D>
MACRO_DEF	<F9>
MACRO_USE	<F10>
MANUAL	<F4>
MARK	<F2>
PASTE	<F3>
PGDN	<PGDN>
PGUP	<PGUP>
PRINT	<Ctrl-P>
READ	<Ctrl-I>
REPLACE	<Ctrl-R>
RIGHT	<RIGHT>
SAVE	<Ctrl-S>
SEARCH_BWD	<Ctrl-B>
SEARCH_FWD	<Ctrl-F>
TOGGLE	<F8>
TRIM_FILE	<Ctrl-X>
UNDO	<Ctrl-U>
UP	<UP>
WORD_LEFT	<Ctrl-LEFT>
WORD_RIGHT	<Ctrl-RIGHT>

The meaning of a key code may depend on the context. For example, the key code ESC represents the <ESC> key, but the consequence of this entry is context dependent. Key codes which are not meaningful in the context are ignored.

Control Codes

Control codes do not represent any keys, instead they specify some special options for the program operating in the macro mode. The control codes are listed in Table 12.4.

TABLE 12.4 MACRO MODE CONTROL CODES

Control Code	Function
PAUSE	pauses (i.e., suspends HarPE) and prompts the user to press a key to continue
WAIT	pauses for 1 second and then continues automatically
WAIT*n*	pauses for *n* seconds, *n* = 1, ..., 5, then continues automatically
WAITM	pauses for 1 minute and then continues automatically
DISABLE_DISPLAY	disables screen output, i.e., suspends displaying text or graphics onto the screen
RESUME_DISPLAY	resumes screen output after DISABLE_DISPLAY
IF_ERROR_PAUSE	terminates the macro mode and returns to the interactive mode if an error occurs in the macro mode (this is the default)
IF_ERROR_EXIT	terminates HarPE and exits to the operating system if an error occurs in the macro mode
REPEAT	repeats all the macro commands in the current macro file
MESSAGE	displays a message box on the screen

An Example of Macro Command Files

Example:

```
"F", READ, "N", "demo01.ckt", ENTER
EXIT, "D", "S", ENTER
```

The meaning of the entries is as follows.

"F"	the first capital letter of File to select the menu option HarPE.File
READ	key <Ctrl-I> to read a circuit file into the editor
"N"	to specify the circuit file by name
"demo01.ckt"	the name of the circuit file
ENTER	key <ENTER> to enter the name of the circuit file
EXIT	key <F7> to exit the editor and parse the circuit file
"D"	the first capital letter of Display to select the menu option HarPE.Display
"S"	the first capital letter of S-par to select HarPE.Display.S-par
ENTER	key <ENTER> to accept the default options for S-par

Use Explicit Cursor Key Codes with Caution

In the interactive mode, the various menu and window options can be selected either by cursor or by using the option identifiers (i.e., the first capital letter in the option name).

Explicit cursor key codes, such as UP and DOWN, should be used with caution in macro command files. The use of cursor key code makes your macro command file explicitly dependent on the positions of the options which may change in future versions of HarPE as new options are added.

It is recommended that you use option identifiers in macro command files for selecting menu and window options, e.g., use "D" to select the option Display.

Macro Control Code IF_ERROR_EXIT

If an error occurs in the macro mode, the macro mode is immediately terminated. By default, HarPE returns to the interactive mode. You can use the macro control code IF_ERROR_EXIT to instruct HarPE to exit to the operating system if an error occurs in the macro mode.

Macro Control Code REPEAT

The macro control code REPEAT instructs HarPE to repeat the execution of a macro file.

For example, suppose that a macro file contains the following commands:

```
...
...
...  Repeat
```

When HarPE encounters the code REPEAT, it will repeat the execution of the entire macro file from the beginning.

Repeated execution of a macro file will continue indefinitely, until you interrupt it by pressing a key (any key) on the keyboard.

Commands following REPEAT are meaningless, since they will never be reached.

Macro Messages

The macro control code MESSAGE instructs HarPE to display the message that follows.

Example:

```
..., MESSAGE, "Welcome to the Show", PAUSE, ...
```

When these macro commands are executed, HarPE will display "Welcome to the Show" on the screen, then pause for the user to press a key before continuing the macro execution.

A message can have multiple lines delimited by "&". For example:

```
MESSAGE, "Demonstration of Project A&FET Parameter Extraction",
         "&Optimization Using HarPE", ...
```

This message consists of three lines, and it will be displayed as

```
Demonstration of Project A
 FET Parameter Extraction
 Optimization Using HarPE
```

Macro messages are shown in a large font and superimposed on the normal display of graphics or text.

The macro command immediately following the message determines what HarPE will do after the message is displayed. It can be PAUSE (pause and wait for the user to press a key), WAIT (pause for 1 second), WAIT*n* (pause for *n* seconds, $1 \le n \le 5$), or WAITM (pause for 1 minute). If it is anything else, the message may disappear immediately.

12.9 HarPE.Learn Menu Option

Macro command files (☞ Section 12.8) can be created using any ASCII text editor, including the file editor in HarPE. The menu option HarPE.Learn can assist you in creating a macro command file on-line.

When you invoke HarPE.Learn, you will be prompted to name the macro command file to be created:

```
Macro File Name:
```

A disk file by the name specified will be created and all the keystrokes you enter from that point on will be translated into the proper macro command entries and recorded in the file.

The Learn process (i.e., macro command file creation) continues until you invoke the menu option HarPE.Learn again or until you terminate HarPE, whichever comes first.

Record Operations by Mouse

If you use the mouse to select and invoke the menu and window options during the Learn process, it will be translated correctly into appropriate entries in the macro command file.

However, if you use the mouse simply to move the cursor so that you can edit the text in different locations, then such cursor movements will not be translated or recorded in the macro command file.

Pauses Must Be Added Manually

You cannot create pause codes (PAUSE and WAIT codes, ☞ Section 12.8) during the Learn process. The pause codes, if desired, must be added to the macro command file manually.

After the Learn process is finished, you can use the file editor in HarPE to read in the macro command file and insert the pause codes at the desired location.

12.10 HarPE.Macro Menu Option

While HarPE is running, you can switch from interactive mode to macro mode using the HarPE.Macro menu option. You can also start HarPE in the macro mode using a run-time option.

Switching from Interactive Mode to Macro Mode

To switch from interactive mode to macro mode, use the menu option HarPE.Macro.

When you invoke HarPE.Macro, you will be prompted for the name of the macro command file to be executed:

```
Macro File Name:
```

The macro command file is read and checked for syntax errors. If no error is found, HarPE will execute the instructions contained in the macro command file.

Macro Mode Always Starts from the HarPE.xxx Menu

The starting point of execution in the macro mode is always the top menu level, namely the HarPE.xxx menu level.

After all the instructions in the macro command file are carried out, HarPE switches back to the interactive mode automatically, unless the last macro command executed is to terminate the program.

Errors will Terminate the Macro Mode

If an error occurs in the macro mode, the macro mode is immediately terminated. By default, HarPE returns to the interactive mode. But this may cause a problem if the program is unattended, because it will block other people from using the system. You can avoid such a situation by using the control code IF_ERROR_EXIT which will terminate the program if an error occurs in the macro mode.

Interrupt the Macro Mode

While HarPE is running in the macro mode, you can interrupt it at any time by pressing a key (any key) on the keyboard. Then the following prompt will be displayed:

```
Terminate Operation (Y/<N>)?
```

and you can decide whether to terminate or to continue operating in the macro mode.

Chained Macro Command Files

Macro command files can be chained (cascaded), i.e., a macro command file may contain an instruction to invoke the menu option HarPE.Macro and then specify another macro command file. The operation will stay in the macro mode, and subsequent instructions will be read from the new macro command file.

As a special case, a macro command file can invoke itself, resulting in a perpetual cycle of operations, until it is interrupted from the keyboard.

Macro command files can be chained, but not nested. For example, if macro command file *A* contains instructions to invoke macro command file *B*, then HarPE will execute the macro command file *B* but will not return to the macro command file *A* after *B* is finished.

Start HarPE in the Macro Mode

Instead of switching HarPE from interactive mode to macro mode, you can start HarPE directly in the macro mode by supplying a macro command file name as an argument.

Command:

`% harpe` *macro_file*

where *macro_file* is the name of a macro command file.

After all the instructions in *macro_file* are carried out, HarPE switches back to the interactive mode, unless it is terminated by an instruction in *macro_file*.

12.11 Silent Mode

In the silent mode, all the screen outputs from HarPE are suppressed.

You can choose the silent mode only when you start HarPE and, if you do, the whole session must be run in the silent mode. In other words, you cannot switch between interactive and silent modes.

To start HarPE in the silent mode, use the run-time option -s.

Command:

`% harpe -s` *macro_file*

where "-s" is the run-time option for silent mode and
macro_file is the name of a macro command file.

A macro command file is required for the silent mode, because all the menus, windows and prompts are suppressed.

When HarPE runs in the silent mode, no text or graphics will be displayed on the screen. But you can still obtain numerical outputs from simulation and save them in data files.

Also, the simulation results are saved in encoded .haf, .sen and .mca files, as described in Chapters 8, 9 and 10. The saved results can then be retrieved and displayed later in an interactive session of HarPE.

To properly complete a silent mode session, the last instruction in the macro command file should terminate the program and return to the operating system. Whether this is the case or not, HarPE will terminate after all the instructions in the macro command file are carried out, or when an error occurs. It is not possible for the program to switch to the interactive mode from the silent mode.

12.12 Background Mode

In the background mode, HarPE runs in a background process.

Command:

`% harpe -b` *macro_file*

where "-b" is the run-time option for background mode and *macro_file* is the name of a macro command file.

HarPE will automatically create (fork) a background process and run in that new process. The foreground process will then return to the operating system and you will be able to perform other tasks.

The background mode implies the silent mode, so that the program in the background will not interfere with the keyboard or the screen operations of the foreground process. The program operates by the instructions contained in the *macro_file*.

When HarPE runs in the background mode, no text or graphics will be displayed on the screen. But you can still obtain numerical outputs from simulation and save them in data files. Also, the simulation results are saved in encoded .haf, .sen and .mca files, as described in Chapters 8, 9 and 10. The saved results can then be retrieved and displayed later in an interactive session of HarPE.

To properly complete a session in the background mode, the last instruction in the macro command file should terminate the program. Whether this is the case or not, HarPE will terminate after all the instructions in the macro command file are carried out, or when an error occurs. The background process terminates with the program. It is not possible to switch from the background mode to the foreground or interactive mode.

12.13 Time Updating Options

On the top-right-hand corner of the window, HarPE displays the time like this:

```
Mon Aug 21 12:34:56 1995
```

The time is also printed to the log file (Section 12.4) to provide a chronological record of the operations.

Two options are available for updating the time: event driven or continuous update.

The event driven option means that HarPE will update the time only when it is not idling, i.e., when there is a user input from the keyboard or when it is performing simulation, optimization, etc.

The continuous update option means that HarPE will keep updating the time even when it is doing nothing else. This can be a waste of resources in a multi-tasking environment if HarPE is left idling for an extended period of time.

The default time updating option is event driven. If you wish to choose the continuous update option, use the run-time option "-t".

Command:

```
% harpe -t
```

where "`-t`" is the run-time option for continuous time updating.

13

Examples

13

Examples

13.1 Overview

A set of demonstration examples is supplied with the HarPE program, as summarized in Table 13.1. The circuit files of the examples are named demo*xx*.ckt, where *xx* is an index. For those examples which involve optimization, the optimized solutions are contained in files named demo*xx*_o.ckt. For instance, the file demo01_o.ckt contains the optimized solution and the corresponding starting point is contained in the file demo01.ckt.

These examples serve a dual purpose: as illustrations of HarPE's features and as templates for creating your own applications.

TABLE 13.1 DEMONSTRATION EXAMPLES

Example	Model	Highlights
demo01	FETM	Optimization to match *S* parameters and DC bias data.
demo02	FETU1	The Curtice symmetrical model is recreated using the user-definable model capabilities (the formulas are in "curtsym.inc"). Optimization to match large-signal power spectrum data.
demo03	FETC	Optimization to match DC IV data.
demo04	FETM	Large-scale optimization to match 30 sets of power spectrum measurements at different bias points, fundamental frequencies and input power levels.
demo05	FETT	Physics-based FET model. Optimization to match *S* parameters.
demo06	FETU2	The Curtice asymmetric model is created as a user-defined model (the formulas are in "curtasy.inc"). Optimization to match *S* parameters and DC data.
demo07	FETU1	Similar to demo02 except that the linear extrinsic subcircuit is defined using discrete components (SRL, CAP, etc.) instead of using the super-component EXTRINSIC2.
demo08	FETM	Modeling by including the small-signal two-port stability factor SK and maximum available gain GMAX in the optimization.

TABLE 13.1 DEMONSTRATION EXAMPLES (cont'd)

Example	Model	Highlights
demo09	FETU1	Similar to demo02 except that the linear extrinsic subcircuit is defined by imported "black box" data.
demo10	FETM	Optimization of simultaneously matching large-signal power spectrum data and small-signal *S* parameters.
demo11	FETM	Optimization to match small-signal *S* parameters using the rectangular form (i.e., RS*ij* and IS*ij*) instead of the polar form (i.e., MS*ij* and PS*ij*).
demo12	FETC	Optimization to match large-signal power spectrum data in Watts (i.e., POUTW*k*) instead in dBm.
demo13	FETU1	Optimization of simultaneously matching large-signal power spectrum and DC data.
demo14	FETM	Importing measured data in Cascade Microtech and MDIF formats directly.
demo15	FETR	Optimization using the Raytheon (Statz) FET model to match measured *S* parameters.
demo16	NPN	Optimization of the bipolar NPN model to match large-signal power spectrum data.
demo17	PNP	Optimization of the bipolar PNP model to match large-signal power spectrum data.
demo18	NPN	Optimization of the bipolar NPN model to match small-signal *S* parameters and DC bias data, using the "Current_Driven" bias setup.
demo19	PNP	Optimization of the bipolar PNP model to match small-signal *S* parameters and DC bias data, using the "Current_Driven" bias setup.
demo20	NPN	Optimization of the bipolar NPN model to match DC IV data, using the "Current_Driven" bias setup.
demo21	PNP	Optimization of the bipolar PNP model to match DC IV data, using the "Current_Driven" bias setup.
demo22	DIODE	Optimization of the diode model to match measured *S* parameters.

TABLE 13.1 DEMONSTRATION EXAMPLES (cont'd)

Example	Model	Highlights
demo23	FETT	DC simulation using a physics-based FET model.
demo24	FETU1	Illustration of statistical modeling using a small-signal user-defined FET model and *S*-parameter data. The sample of models from multi-device parameter extraction is saved in the file "demo24_o.ckt" and the postprocessed statistical model is saved in the file "demo24_m.ckt".
demo25	FETU1	Illustration of using preprocessor macros to customize the circuit file syntax.
demo26	linear	Monte Carlo simulation of a linear LC filter with statistical parameters.
demo27	FETM	Monte Carlo simulation of nonlinear FET frequency doubler with yield specifications on spectral purity and conversion gain.
demo28	KTL	Illustration of using the nonlinear physics-based Ladbrooke-Trew FET model.
demo29	FETU1	The Raytheon Statz model is recreated using the user-definable model (the formulas are in "statz.inc").
demo30	FETU1	The TriQuint's Own Model (TOM) for FET is created using the user-definable model (the formulas are in "tom.inc").
demo31	FETU1	The Plessey FET model is created using the user-definable model (the formulas are in "plessey.inc").
demo32	BJTU	The Gummel-Poon NPN bipolar transistor model is recreated using the user-definable model (the formulas are in "npn.inc").
demo33	BJTU	The Gummel-Poon PNP bipolar transistor model is recreated using the user-definable model (the formulas are in "pnp.inc").
demo34	HEMTC	DC and small-signal simulation using the Curtice HEMT model.
demo35	HEMTAC	Optimization to match DC data and small-signal *S* parameters using the advanced Curtice HEMT model.
demo36	FETU1	Statistical modeling using cumulative probability distribution fitting with a user-defined small-signal FET model.
demo37	linear	Using EEsof *S*-parameter data file directly in imported 2-port subcircuit.

TABLE 13.1 DEMONSTRATION EXAMPLES (cont'd)

Example	Model	Highlights
demo38	FETR	Optimization to fit S-parameters imported in EEsof format.
demo39	HBT	Illustration of using the HBT model in circuit simulation.
demo40	FETM	Minimax optimization of a small-signal FET amplifier using OSA90. The FET is simulated by the child HarPE using the Materka and Kacprzak model. No label substitution is involved. Only the matching circuits are optimized.
demo41	KTL	Minimax optimization of a small-signal FET amplifier using OSA90. The FET is simulated by the child HarPE using the physics-based KTL model. Label substitution is in olved. Both the matching circuits and the FET are optimized.
demo42	FETR	Minimax optimization of a small-signal FET amplifier using OSA90. The FET is modelled by FETR with parameters extracted by the child HarPE from measurements (NE71000 data). The starting point for parameter extraction is supplied by OSA90 and passed to the child HarPE through Datapipe.
demo43	FETM	Using Macro command to run HarPE as a child of OSA90. Instead of straight simulations as illustrated in demo40 and demo41 the child HarPE first performs parameter extraction from measurements to obtain the optimized model parameters and then simulates the FET and returns the simulation results to OSA90.
demo44	KTL	Nominal and yield optimization of a small-signal amplifier using OSA90. Two child HarPEs are invoked for nominal and statistical modeling. The first child HarPE performs nominal parameter extraction and returns the optimized parameter values to OSA90. The second child HarPE performs indirect statistical modeling using the parameter values extracted by the first child HarPE as the starting point for multi-device parameter extraction. The statistical parameters including the mean values, standard deviations and correlation coefficients are returned by the second child HarPE to OSA90 for nominal and yield optimization.

As a case study, two of the examples, namely demo01 and demo02 are analyzed in the following sections.

The details of examples demo40 through demo44 related to HarPE running as a Datapipe child of OSA90/hope can be found in Chapter 3 of the manual "HarPE as a Child of OSA90/hope".

13.2 Example demo01

This example optimizes the built-in Materka and Kacprzak nonlinear FET model (FETM) to match the measured small-signal *S* parameters and DC bias data.

The *S* parameters are measured at 3 bias points and 17 frequencies between 2GHZ and 18GHZ. Nine of the model parameters are defined as optimization variables.

Circuit file demo01.ckt

```
!  Example  demo01.ckt
!  Optimization to fit S-parameter and DC measurements.
!  Model used: Built-in Materka and Kacprzak model.
!  Variables are selected based on sensitivity analysis

Model
   Extrinsic2  1 2 3 4 5
      LG: 0.1NH    RG: 0.0119    RD: 0.0006
      LD: 0.1NH    RS: 0.33      CX: 10PF
      GDS: ?0.004?
      LS: ?0.1NH?
      CDS: ?0.1PF?;

   FETM  1 2 3
      GAMMA: -0.1    KE: 0          SL: 0.2
      KG: -0.25      TAU: 3PS       SS: 0
      IG0: 5E-10     ALPHAG: 20     IB0: 8e-12
      ALPHAB: 1      VBC: 20        R10: 5.2
      KR: 0          K1: 1          C1S: 0.0048PF
      IDSS: ?0.2?
      VP0: ?-4?
      E: ?2?
      C10: ?0.67PF?
      CF0: ?0.023PF?
      KF: ?-0.12?;

   2POR  4  5;
end

Data
! S-parameter measurement data
#include "spar_1.dat"
end

Sweep
   ! S-parameter simulation

   FREQ: from 2GHZ to 18GHZ step=1GHZ  VG: 0 -1.74 -3.1  VD: 4V;

   ! DC IV simulation

   VG: 0 -1.74 -3.1  VD: from 0 to 1.6 step=0.2 from 2 to 6 step=0.5;

   ! HB simulation

   FREQ: 0.2GHZ 6GHZ 10GHZ  PIN: 5DBM    VG: -0.673  VD: 4;
   FREQ: 6GHZ  PIN: from -15DBM to 10DBM step=5DBM VG: -0.673  VD: 4;
end

Specification
   FREQ: from 2GHZ to 18GHZ step=3GHZ VG: 0 -1.74 -3.1  VD: 4
   MS11 PS11 MS21 PS21 MS12 PS12 MS22 PS22;

   VG: 0 -1.74 -3.1  VD: 4  ID W=50;  ! weighting factor for ID is 50
end
```

Measurement Data File spar_1.dat

```
! File: spar_1.dat
!       S-parameter and DC measurement data at 3 bias points
!
PARAMETER VG = 0  VD = 4;
FORMAT  FREQ(GHZ)  MS11   PS11   MS21   PS21   MS12   PS12   MS22   PS22;
2.  0.9546   -46.72   4.0405   145.54   0.0291   62.95   0.6010  -21.43
3.  0.9392   -66.98   3.6149   129.27   0.0388   52.47   0.5808   -32.82
4.  0.8944   -83.24   3.3323   118.50   0.0458   42.58   0.5718   -39.93
5.  0.8789   -97.95   2.9539   102.85   0.0507   33.91   0.5701   -48.94
6.  0.8598  -108.88   2.6428    95.28   0.0518   28.76   0.5808   -54.82
7.  0.8460  -120.97   2.2946    82.85   0.0517   18.44   0.5683   -63.42
8.  0.8388  -127.78   2.0989    78.94   0.0503   18.99   0.5845   -68.54
9.  0.8340  -137.66   1.9225    67.49   0.0505   12.84   0.5867   -75.58
10. 0.8350  -143.32   1.7984    58.95   0.0525   10.51   0.6089   -79.69
11. 0.8326  -149.51   1.5910    56.35   0.0499   12.37   0.6166   -83.21
12. 0.8326  -155.26   1.5106    46.49   0.0507    6.11   0.6360   -91.66
13. 0.8196  -161.01   1.3838    40.82   0.0473    8.76   0.6427   -93.39
14. 0.8301  -166.19   1.2945    34.54   0.0466    4.40   0.6633  -101.10
15. 0.8242  -172.00   1.1615    24.82   0.0442    3.98   0.6718  -106.42
16. 0.8332  -174.63   1.1335    23.98   0.0436    7.27   0.6845  -108.67
17. 0.8190  -179.96   0.9365    11.88   0.0408    5.08   0.7045  -113.75
18. 0.8394  -180.64   0.9688     8.12   0.0420    6.08   0.7098  -115.22

PARAMETER VG = -1.74  VD = 4;
FORMAT  FREQ(GHZ)  MS11   PS11   MS21   PS21   MS12   PS12   MS22   PS22;
2.   0.9585   -36.75   3.1389   150.53   0.0408   66.80   0.5439   -23.91
3.   0.9457   -54.12   2.8906   135.84   0.0567   56.58   0.5303   -36.95
4.   0.9004   -67.99   2.7296   125.90   0.0688   47.29   0.5204   -45.35
5.   0.8847   -82.22   2.4969   111.25   0.0787   37.94   0.5232   -55.70
6.   0.8597   -92.49   2.2895   103.05   0.0836   32.18   0.5309   -62.18
7.   0.8375  -104.76   2.0371    91.36   0.0868   21.53   0.5177   -71.76
8.   0.8262  -111.74   1.8851    86.36   0.0873   20.33   0.5312   -76.63
9.   0.8144  -122.27   1.7462    75.87   0.0896   12.49   0.5321   -84.21
10.  0.8145  -128.60   1.6536    66.69   0.0951    9.33   0.5543   -88.84
11.  0.8080  -134.73   1.4918    64.58   0.0906    7.99   0.5553   -92.92
12.  0.8023  -141.64   1.4118    53.94   0.0942    0.67   0.5692   -99.86
13.  0.7920  -146.83   1.3213    49.51   0.0896    0.20   0.5759  -102.41
14.  0.8003  -153.41   1.2339    42.40   0.0905   -5.70   0.5958  -109.06
15.  0.7885  -159.07   1.1325    33.85   0.0856   -8.18   0.6067  -114.76
16.  0.7926  -162.96   1.0965    32.61   0.0857   -9.30   0.6098  -115.86
17.  0.7871  -167.84   0.9808    21.65   0.0811  -13.31   0.6270  -121.67
18.  0.7972  -169.40   0.9671    20.98   0.0799  -14.38   0.6318  -120.02

PARAMETER VG = -3.1  VD = 4;
FORMAT FREQ(GHZ) MS11 PS11 MS21 PS21    MS12      PS12    MS22    PS22;
    2.0    .9614  -32.46  2.5494  152.25   .0491   68.66   .5383  -24.26
    3.0    .9490  -48.34  2.3749  138.14   .0691   58.95   .5287  -37.94
    4.0    .9026  -60.84  2.2681  128.46   .0853   49.75   .5227  -46.46
    5.0    .8840  -74.37  2.1038  114.14   .0993   40.23   .5255  -57.50
    6.0    .8604  -84.23  1.9597  105.74   .1070   34.70   .5312  -64.10
    7.0    .8324  -96.28  1.7474   93.93   .1124   23.84   .5203  -74.55
    8.0    .8145 -103.10  1.6293   88.90   .1147   22.02   .5292  -79.12
    9.0    .7972 -113.68  1.5213   78.38   .1191   13.54   .5331  -87.56
   10.0    .7953 -120.22  1.4587   68.97   .1275    9.82   .5479  -91.61
   11.0    .7859 -126.03  1.3230   66.72   .1225    8.00   .5523  -96.66
   12.0    .7767 -133.54  1.2602   55.70   .1286     .46   .5566 -102.75
   13.0    .7621 -138.24  1.1803   51.49   .1214   -1.44   .5681 -108.18
   14.0    .7681 -145.50  1.1179   44.23   .1252   -7.09   .5876 -111.68
   15.0    .7587 -151.00  1.0290   35.72   .1211  -10.03   .5939 -117.61
   16.0    .7610 -155.48  1.0001   33.95   .1213  -12.09   .5907 -118.18
   17.0    .7570 -160.04   .8957   23.17   .1145  -16.18   .5972 -123.92
   18.0    .7594 -162.61   .8860   21.81   .1139  -18.16   .6233 -121.86

FORMAT  VG    VD   ID;
      0.0   4.0  0.177
     -1.74  4.0  0.092
     -3.10  4.0  0.037
```

Step by Step "Guided Tour"

1 Start HarPE. A "title page" appears on the screen. Press <ENTER>, and the files in the working directory are listed on the screen. Move the cursor to select demo01.ckt and press <ENTER>.

2 You are in the screen editor. The contents of the circuit file demo01.ckt are displayed. Click the left-hand mouse button to activate the editor menu and select "Exit from editor" (or press <F7>) to exit from the editor. You should see the message

```
Parsing Circuit File ...
```

After the file is successfully parsed, you will be at the HarPE> menu.

3 Press "D" to select the menu option HarPE.Display (you can also move the cursor to highlight the option and then press <ENTER>). HarPE performs circuit simulation, or, if a .haf file is available, the results from a previous simulation will be retrieved (Chapter 8).

4 You are at the HarPE.Display> menu. Press "S" to select the S-par display option. A pop-up window appears. Press <ENTER> to accept the default options and you will see the Smith Chart/polar plot display which shows the match between the simulated and measured *S* parameters at one of the bias points, as illustrated in Fig. 13.1.

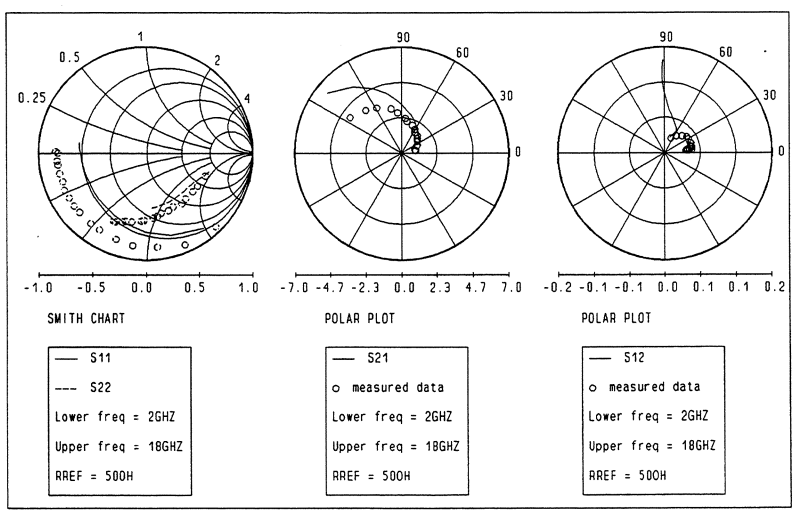


Fig. 13.1 S-parameter match before optimization.

5 Press "D" to select the DC display option. A pop-up window appears. Press <ENTER> to accept the default options and you will see the match between the simulated and measured DC bias data as illustrated in Fig. 13.2.

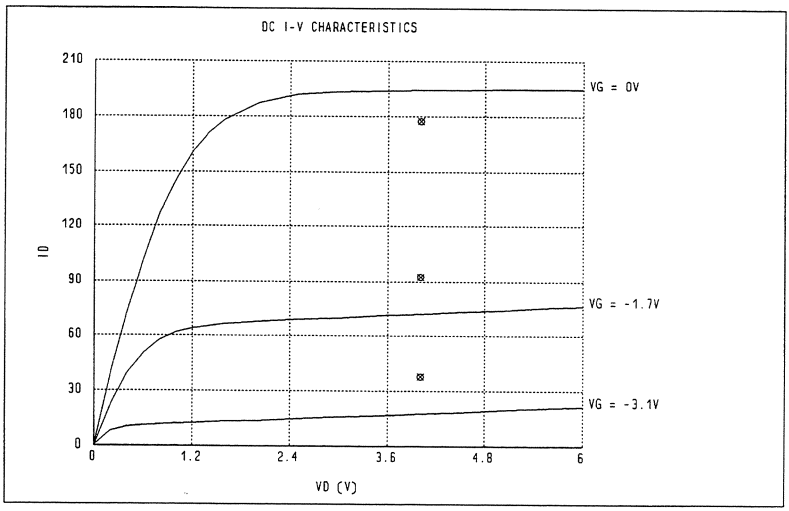


Fig. 13.2 Match between simulated and measured DC data before optimization.

6 The match in Figs. 13.1 and 13.2 are obviously quite poor. We need to optimize the model. Press <ESC> to exit from the HarPE.Display> menu and go back to the HarPE> menu. Press "O" to invoke the HarPE.Optimize menu option. A pop-up window appears. Press <ENTER> to accept the default options and start the optimization process. You will see the progress of optimization displayed on the screen.

7 After the optimization is completed, press "F" to return to the editor. You can see that the variables in the circuit file are updated with the optimized values. The updated file is only a copy in the editor's memory buffer. If you wish to save it to the disk, click the left-hand mouse button to activate the editor menu and select "Save file to disk" (or press <Ctrl-S>) and specify a new file name. The optimized solution for this example is already saved in the file demo01_o.ckt.

8 Click the left-hand mouse button to activate the editor menu and select "Exit from editor" (or press <F7>) and press "D" for HarPE.Display. Wait for the simulation to complete, then press "S" for S-par and press <ENTER>. You will see the match between the simulated and measured *S* parameters at one of the bias points, as illustrated in Fig. 13.3. It is significantly better than the match in Fig. 13.1.

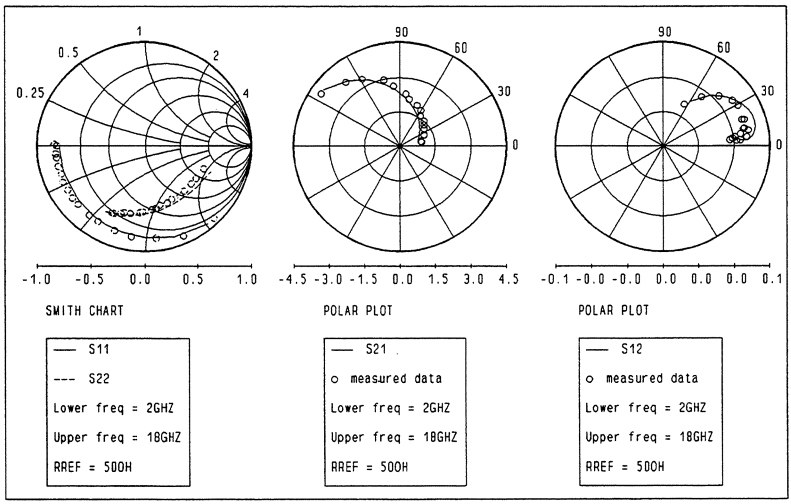


Fig. 13.3 S-parameter match after optimization.

9 You can select a different bias point or a different response for display. For instance, press "G" for Gate bias and press <RIGHT> (the right cursor key) to toggle the gate bias voltage to the second available value. Then press "Y" for Y-axis, press <RIGHT> to toggle the choice to "S11 & S22", and press <ENTER> twice to accept the modified options. The resulting display is illustrated in Fig. 13.4.

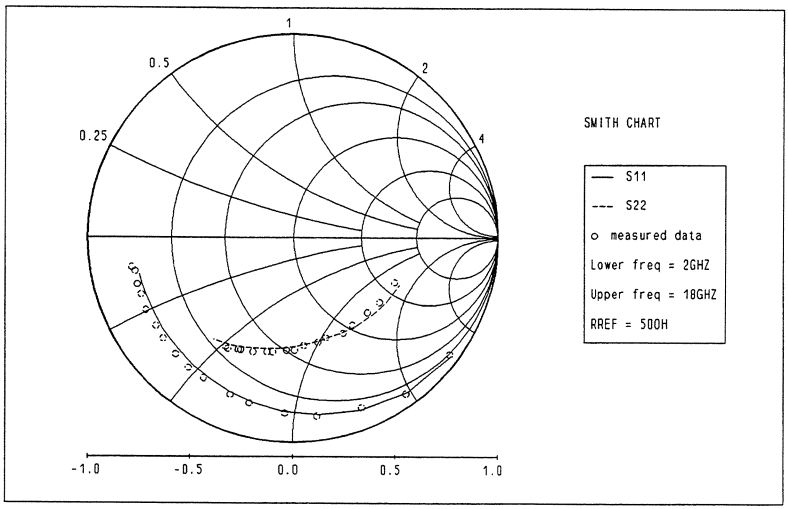


Fig. 13.4 Smith Chart of S11 and S22 match.

10 Now, press "D" to select the DC display option. A pop-up window appears. Press <ENTER> to accept the default options and you will see the optimized match between the simulated and measured DC bias data as illustrated in Fig. 13.5, which is obviously better than the match shown in Fig. 13.2.

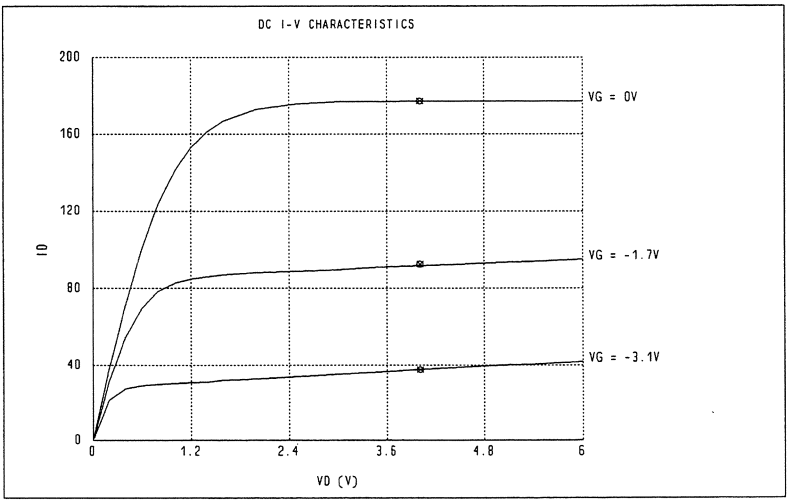


Fig. 13.5 DC match after optimization.

11 Press <ESC> to exit from the HarPE.Display> menu and go back to the HarPE> menu. Press "F" to select HarPE.File, and you are again in the file editor. Click the left-hand mouse button to activate the editor menu and select "Read a new file" (or press <Ctrl-I>) to read another circuit file.

13.3 Example demo02

This example demonstrates how to create a user-defined device model.

The model chosen for illustration is the Curtice and Ettenberg cubic symmetric model. This model is available as a built-in model, namely FETC. In demo02.ckt, we recreate the model using the topology provided by FETU1. The model equations are defined using expressions and saved in the include file "curtsym.inc". By using an include file, the user-defined model can be shared by other circuit files.

The parameters of the user-defined model are extracted from measured data through optimization. The measurements include the drain current at DC and the output power at the first three harmonics at three different fundamental frequencies, three different bias points, and six input power levels. The measurements selected for optimization correspond to the fundamental frequency of 6GHZ and two input power levels of -5dBm and 5dBm.

Circuit File demo02.ckt

```
! Example  demo02.ckt
! Harmonic balance optimization to fit power spectrum measurement.
! Model used: User-created Curtice cubic symmetrical model.

Expression
   !  user created model parameters

   BETA: 0.03;       VDS0: 2;          IS: 1e-14;    N: 1;
   CGS0: 0.545PF;    CGD0: 0.092PF;    FC: 0.77;     GMIN: 1.0E-07;
   VBI: 0.8;         VBR: 30;          K: 1.381;     Q: 16020;
   TEMP: 298;
   A0: ?0.15?;
   A1: ?0.15?;
   A2: ?-0.02?;
   A3: ?-0.02?;
   GAMMA: ?1?;

   #include "curtsym.inc"   ! formulas of the Curtice symmetric model
end

Model
   Extrinsic2    1 2 3 4 5
      LG: 0.338NH    RG: 4.063    RD: 0.01
      LD: 0.254NH    RS: 2.49     LS: 0.08NH
      GDS: 0.2e-3    CX: 1.5PF    CDS: 0.1PF;

   FETU1    1 2 3
      TAU: 3PS          IGS: IGS_MODEL    IDS: IDS_MODEL
      IGD: IGD_MODEL    QGS: QGS_MODEL    QGD: QGD_MODEL;

   2POR  4 5;
end

Data
#include "fetb_02.dat"     ! power spectrum measurement data
#include "fetb_6.dat"
#include "fetb_10.dat"
end

Sweep
   ! HB simulation

   FREQ: 0.2GHZ 6GHZ 10GHZ  PIN: 5DBM   VG: -0.673  VD: 4;
   FREQ: 6GHZ  PIN: from -15DBM to 10DBM step=5DBM VG: -0.673  VD: 4;
```

```
   ! S-parameter simulation

   FREQ: from 2GHZ to 18GHZ n=16  VG: 0 -1.74  VD: 4V;

   ! DC IV simulation

   VG: from -1.5 to 0 step=0.25  VD: from 0 to 2 step=0.2 from 2 to 5 step=1;
end

Specification
   FREQ: 6GHZ  PIN: -5DBM 5DBM  VG: -0.673  VD: 4  ID0 POUT1 POUT2 POUT3;
end
```

Include File curtsym.inc

```
! File: curtsym.inc
! Example of user-defined FET models
! Curtice and Ettenberg cubic symmetrical model is created in this file
!
! List of model parameters:
!
!    A0  A1  A2  A3  GAMMA  BETA  VDS0  IS  N  CGS0  CGD0
!    FC  GMIN  VBI  VBR  K  Q  TEMP
!
! List of model currents and charges computed from the model:
!
!    IGS_MODEL   IDS_MODEL  IGD_MODEL  QGS_MODEL  QGD_MODEL
!
! Example of usage in the MODEL block:
!
!    FETU1  @gate @drain @source
!           TAU: 3PS          IGS: IGS_MODEL  IDS: IDS_MODEL
!           IGD: IGD_MODEL  QGS: QGS_MODEL  QGD: QGD_MODEL;
!
! drain current:
   V1 = VGS_TAU * (1 + BETA * (VDS0 - VDS_T));
   IDS_CUBIC = IF (VDS_T > 0)
                  ((A0 + (A1 + (A2 + A3*V1)*V1)*V1) * tanh(GAMMA * VDS_T));
   IDS_MODEL = IF (IDS_CUBIC > 0) (IDS_CUBIC);       ! pinchoff truncation

!  gate current:
   VT = K * TEMP/Q;
   AA = VGS_T + 5 * N * VT;
   BB = VGS_T + VBR - 50 * VT;
   BBB1 = VGS_T / (N * VT);
   BBB2 = IF (BBB1 < 40) (BBB1) ELSE (40);
   BBB3 = (-VGS_T - VBR) / VT;
   BBB4 = IF (BBB3 < 40) (BBB3) ELSE (40);
   IGS_MODEL = GMIN * VGS_T + IS * (IF (AA > 0) (EXP(BBB2) - 1)
               ELSE (IF (BB > 0) (-1) ELSE (-1 - EXP(BBB4))));

! gate-drain current:
   VGD_T = VGS_T - VDS_T;
   CC = VGD_T + 5 * N * VT;
   DD = VGD_T + VBR - 50 * VT;
   DDD1 = VGD_T / (N * VT);
   DDD2 = IF (DDD1 < 40) (DDD1) ELSE(40);
   DDD3 = (-VGD_T - VBR) / VT;
   DDD4 = IF (DDD3 < 40) (DDD3) ELSE (40);
   IGD_MODEL = GMIN * VGD_T + IS * (IF (CC > 0) (EXP(DDD2) - 1)
               ELSE (IF (DD > 0) (-1) ELSE (-1 - EXP(DDD4))));

! gate charge (charge of the gate capacitor)
   FF = 1 - FC;
   GG = SQRT(FF);
   QGS_MODEL = IF (VGS_T < FC * VBI)
                  (CGS0 * 2 * VBI * (1 - SQRT(1 - VGS_T/VBI)))
```

```
                ELSE (CGS0 * (2 * VBI * (1 - GG)
                   + (VGS_T * (1 + 0.5 * (0.5 * VGS_T/VBI - FC)/FF)
                   - FC * VBI * (1 - 0.25 * FC/FF))/GG));

! gate-drain charge (charge of the gate-drain capacitor)
   QGD_MODEL = IF (VGD_T < FC * VBI)
                   (CGD0 * 2 * VBI * (1 - SQRT(1 - VGD_T/VBI)))
                ELSE (CGD0 * (2 * VBI * (1 - GG)
                   + (VGD_T * (1 + 0.5 * (0.5 * VGD_T/VBI - FC)/FF)
                   - FC * VBI * (1 - 0.25 * FC/FF) )/GG));
```

Measurement Data File fetb_6.dat

Three data files are included in the circuit file demo02.ckt, namely fetb_02.dat, fetb_6.dat and fetb_10.dat. They contain large-signal power spectrum data measured at three different fundamental frequencies (A.M. Pavio, Texas Instruments, Dallas, TX, Dec. 1988, private communication).

The subset of data actually used for optimization is at the frequency of 6GHZ and the bias point VG = -0.673V, VD = 4V, which is contained in the data file fetb_6.dat.

```
!  File: fetb_6.dat
!  FET power spectrum measurement data
!  fundamental freq is 6 GHz
!
 PARAMETER VG = -0.372  VD = 2  FREQ = 6GHZ;
 FORMAT PIN(DBM) POUT1(DBM) POUT2(DBM)  POUT3(DBM) ID0(MA);
       +10.0     +15.1      +2.4       -5.7       38.9
        +5.0     +13.0       -5.2      -11.9       42.3
         0.0      +9.6      -19.5      -27.3       44.3
        -5.0      +4.9      -32.4      -45.6       44.7
       -10.0       0.0      -42.7      -60.1       44.9
       -15.0      -5.2      -52.8      -99.9       45.1
 PARAMETER VG = -0.673  VD = 4  FREQ = 6GHZ;
 FORMAT PIN(DBM) POUT1(DBM) POUT2(DBM)  POUT3(DBM) ID0(MA);
       +10.0     +18.1      -1.5       -7.3       42.8
        +5.0     +13.9      -10.7      -22.1       34.0
         0.0      +9.5      -21.2      -36.1       31.0
        -5.0      +4.6      -31.5      -49.9       30.2
       -10.0      -0.3      -41.4      -62.1       30.0
       -15.0      -5.5      -54.4      -99.9       30.0
 PARAMETER VG = -1.073  VD = 6  FREQ = 6GHZ;
 FORMAT PIN(DBM) POUT1(DBM) POUT2(DBM)  POUT3(DBM) ID0(MA);
       +10.0     +16.1      +1.9       -10.2      31.2
        +5.0     +11.7       -5.8      -20.6       21.3
         0.0      +7.3      -14.8      -33.5       17.1
        -5.0      +2.4      -24.6      -47.8       15.6
       -10.0      -2.6      -34.4      -61.1       15.1
       -15.0      -7.8      -46.9      -99.9       15.0
```

Step by Step "Guided Tour"

It is assumed that you are quite familiar with the general operations of HarPE, otherwise you can practice with demo01, as described in detail in Section 13.2.

1 In the editor, read the circuit file demo02.ckt and press <F7>).

2 Press "D" for HarPE.Display to examine the match between the simulated and measured power spectrum responses. Press "P" for Psweep (power sweep), then press <ENTER> to accept the default options. From the display, as illustrated in Fig. 13.6, you can clearly see a substantial mismatch between the simulated and measured responses before optimization.

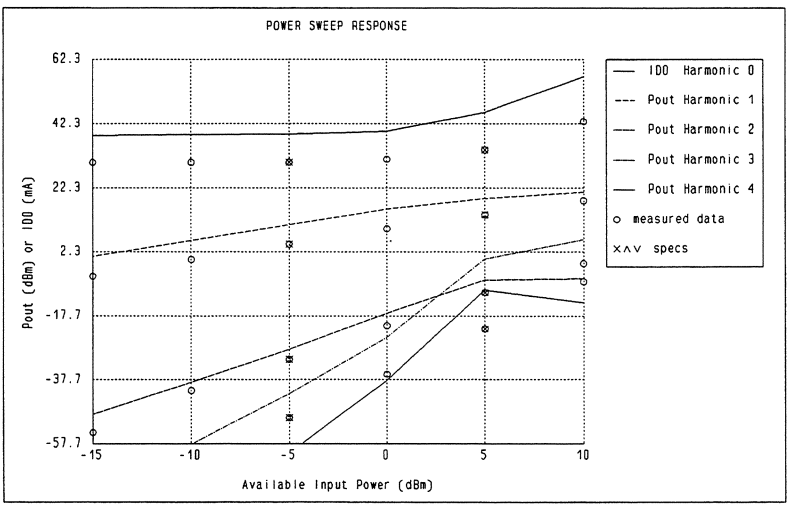


Fig. 13.6 Match between simulated and measured power spectra before optimization.

3 Press <ESC> to return to the main menu and press "O" for HarPE.Optimize. Press <ENTER> to start the optimization. The progress of optimization is reported on the screen.

4 After the optimization is completed, press "F" to return to the editor. You can see that the variables in the circuit file are updated with the optimized values. The updated file is only a copy in the editor's memory buffer. If you wish to save it to the disk, press <Ctrl-S> and specify a new file name. The optimized solution for this example is already saved in the file demo02_o.ckt.

5 Press <F7> to exit from the editor and press "D" for HarPE.Display. Wait for the simulation to complete, then press "P" for Psweep and press <ENTER>. The display of the simulated and measured power spectra is illustrated in Fig. 13.7. Clearly, the match is much improved in comparison with Fig. 13.6.

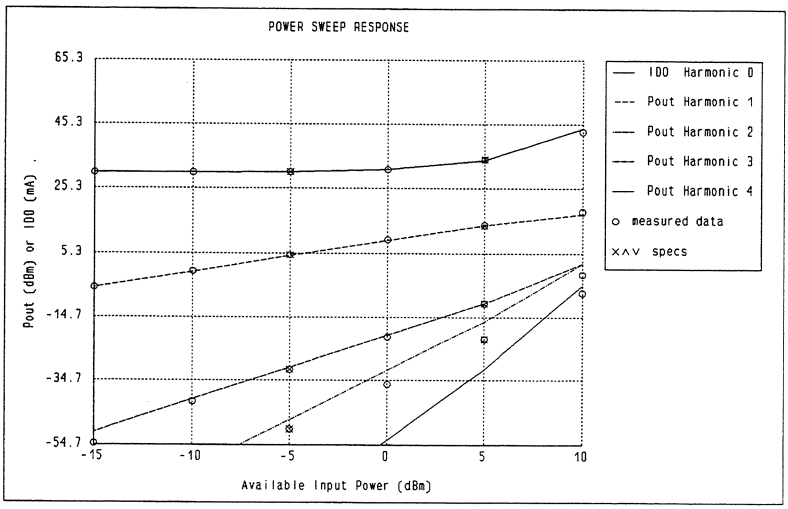


Fig. 13.7 Match between simulated and measured power spectra after optimization.

6 Press <ESC> to exit from the HarPE.Display> menu. You can press "F" to go back to the file editor, then press <Ctrl-I> to read another example. Or, you can press <ESC> to exit from HarPE and return to the operating system.

Appendix
Diagnostic Messages

Appendix
Diagnostic Messages

Overview

This appendix lists the diagnostic messages that HarPE may produce at run time. The probable cause of each error is discussed and a possible solution is suggested if available.

Diagnostic message:

identifier: *message*

where *identifier* is a three-character code which OSA can use to trace the error.

Internal Errors

Internal error message:

identifier: `Internal Error` *xxxx*

where *xxxx* is an error index which OSA uses to identify the internal error.

If you encounter an internal error message, please contact Optimization Systems Associates Inc. for technical assistance:

Phone 905 628 8228
Fax 905 628 8225

CFP: A Constant Value Or Constant Label Is Expected

In the Control block, some options such as Accuracy must be assigned with constant values, check the requirement for the corresponding options you used.

CFP: A Constant Integer Is Expected

In the Control block, some options such as N_Iterations for optimization must be assigned with constant integers, check the requirement for the corresponding options you used.

CFP: A Node Number Or Name Expected

A node (numerical or string) is expected (see Chapter 5).

CFP: A Positive Integer Expected

A positive integer is expected to follow certain keywords.

CFP: All Input Keywords Must Precede Output Keyword

In the definition of sweep ranges in the Sweep block, input keywords such as PIN and FREQ must precede any response keywords such as POUT1.

CFP: Character String Entry Expected

A character string entry is expected following a string label definition or certain keywords such as NAME.

CFP: Circuit File Block Name Expected

Entries in the circuit file must be between a block name and an End. Check Chapter 3 for the list of block names.

CFP: Correlation Matrix Diagonal Entry Must Be 1.0

A diagonal entry of the correlation matrix represents the "self-correlation" of a statistical variable, and hence it must be 1.0 by definition.

CFP: Correlation Matrix Dimension Is Undefined

In the Statistics block, the dimension of each correlation matrix must be explicitly defined.

CFP: Correlation Matrix Dimension Must Be > 1

The function of correlation matrices is to describe the correlations between statistical parameters. Obviously two or more statistical parameters must be involved. Therefore, the dimension of correlation matrices must be greater than one.

CFP: Correlation Matrix Must Be Symmetrical

Correlation matrices must be symmetrical by definition.

CFP: Correlation Name Expected

In the definition of statistical parameters, following the keyword CORRELATION you must specify the name of a correlation matrix defined in the Statistics block.

CFP: Data FORMAT Statement Missing

A FORMAT statement is required for each data set in the Data block of the circuit file. See Chapter 9.

CFP: Different Types Of Output Cannot Be Mixed

When creating user-defined responses using expressions involving the built-in responses (predefined labels), all the responses in one expression must be of the same simulation type (DC, small-signal or large-signal). For example, you cannot mix POUT1 (a large-signal response) with MS21 (a small-signal response) in the same expression.

CFP: Duplicate Definition Of Circuit File Block

Each circuit file block can only be defined once in the circuit file.

CFP: Duplicate Definition Of Keyword

Duplicate definition of certain keywords (device names, parameter names, etc.) is not allowed.

CFP: Duplicate Definition Of Label Name

When creating a user-defined label, you must give it a unique name. You cannot use names which have already been assigned to other labels. Also, check Chapter 4 for the list of predefined label names which cannot be duplicated.

`CFP: End-Of-Record Symbol Semicolon (;) Expected`

The current statement is expected to be completed and terminated with a semicolon ";". Instead, extraneous entries are encountered which appear to be out of context.

`CFP: Equal Sign (=) Or Colon (:) Missing`

Certain keywords in the circuit file are expected to be followed by an equal sign "=" or a colon ":" and then followed by a user entry.

`CFP: Expression Operand Expected`

Typically in an expression the operands (numerical constants, labels and math function names) alternate with the operators (+, -, *, /, etc.). It does not make sense to have operators next to one another.

`CFP: Expression Operator Expected`

Typically in an expression the operands (numerical constants, labels and math function names) alternate with the operators (+, -, *, /, etc.). It does not make sense to have operands next to one another.

`CFP: Expression Too Long`

The length of an expression exceeds the limit. You can break up the long expression into smaller ones. Also, make sure that each expression is terminated by a semicolon ";".

`CFP: Extrinsic Circuit Definition Expected For Preprocess`

For analytically extracting the parasitic parameters of the extrinsic circuits of FETs or HEMTs from the cold measurements, the user must define an EXTRINSIC circuit model in the Model block (see Chapter 9).

`CFP: File Name Expected`

A file name is expected to follow certain keywords, such as "#include".

`CFP: Illegal Data Type`

The offending entry in the circuit file has out-of-context attribute(s). For instance, a real value is supplied where an integer is expected (such as for a node number).

`CFP: Illegal Harmonic Index`

The harmonic index exceeds the limit (see Chapter 1 for the size limits). Also, a harmonic index must not be negative.

`CFP: Illegal Index (Not An Integer Or Exceeds Limit)`

An index (such as a port number) must be specified by a positive integer. Either the entry encountered is not an integer, or it has a negative value, or its value exceeds the limit.

`CFP: Illegal Number Of Ports`

The number of ports defined in the circuit file must be either one or two.

`CFP: Illegal Operator Or Reserved Symbol`

See Chapter 4 for the list of reserved symbols.

`CFP: Illegal Port Definition`

Improper definition of a port in the circuit file. Most likely you have assigned the same node number to both nodes of the port, thus creating a short-circuit.

`CFP: Illegal Range Definition`

Improper definition of "From *x1* to *x2* step=*x3*" or "From *x1* to *x2* N=*n*" (see Chapters 8, 9 and 10). Perhaps one of the keywords is missing or misspelled, or the number of steps resulting from the definition exceeds the limit.

`CFP: Illegal Specification On Waveform Response`

Waveform responses can only be specified for modeling optimization, i.e., to match the simulated and measured waveforms.

`CFP: Illegal Use Of Output Label`

Expressions which involve responses (such as POUT*k* and MS*ij*) can only be used as user-defined responses. They cannot be used to define circuit model parameters.

`CFP: Import Data Incomplete: Whole Matrix Expected`

To use imported *S*, *Y* or *Z* data as linear "black-box" subcircuits, the imported data must supply the complete *S*, *Y* or *Z* matrix. For example, for a two-port *S* matrix, all of the eight parameters, namely MS11, PS11, ..., MS22 and PS22, must be provided.

CFP: Import Data Must Contain At Least 3 Frequencies

Each set of import data must contain at least three different frequencies so that the necessary interpolation can be performed (see Chapter 5).

CFP: Improper Device For Preprocess

For analytically extracting the parasitic parameters of the extrinsic circuits of FETs or HEMTs from the cold measurements, the user must define a FET or HEMT model in the Model block (see Chapter 9).

CFP: Inadequate Data

Ambiguous PARAMETER or FORMAT statement in the Data block of the circuit file. Likely, no response keyword is specified (see Chapter 9).

CFP: Include File Or Macro Cannot Contain Variables

Include files and text macros are handled by the preprocessor. The contents of an include file is merged into the circuit file at parsing time. Text macros are expanded into the appropriate substitution text in the circuit file. Include files and text macros cannot contain optimization variables, because such variables cannot be properly updated after optimization.

CFP: Inconsistent Bounds

Inconsistent bounds defined for an optimization variable: either the lower bound is larger than the nominal value or the nominal value is larger than the upper bound.

CFP: Inconsistent Cold Measurements For Preprocess

The cold measurements (unbiased and pinched-off) must have the same number of data points for analytically extracting the parasitic parameters of the extrinsic circuits of FETs or HEMTs (see Chapter 9).

CFP: Incorrect Number Of Entries

The number of entries is different from what is expected. For instance, an optimization variable can be defined as "?*x*?" or "?*x1 x2 x3*?", but not as "?*x1 x2*?".

CFP: Insufficient Cold Measurements For Preprocess

For analytically extracting the parasitic parameters of the extrinsic circuits of FETs or HEMTs from the cold measurements, the user must supply both the unbiased (VD = 0, VG = 0) and pinched-off (VD = 0, VG < VP) measurement data in the Data block. Each set of measurements must consist of at least 2 data points (see Chapter 9).

`CFP: Keyword DATA = Expected For Import Data`

The keyword DATA is required for the element *n*PORDATA (see Chapters 5 and 7).

`CFP: Keyword Illegal For Measured Data`

The keyword used is not applicable for measured data.

`CFP: Keyword Incompatible With Those Already Specified`

Keywords in the same data set, sweep set or specification set have incompatible simulation types. For example, the keyword PIN which represents the large-signal input power is incompatible with the keyword MS11 which represents a small-signal response.

`CFP: Keyword Is Applicable To 2-Ports Only`

Certain keywords, such as MS22, SK and GMAX, are not available for one-port circuits.

`CFP: Keyword Reserved For Optimization Goals`

Certain keywords, such as SK and GMAX, are to be used only for specifying optimization goals. They cannot be used to in the Data block.

`CFP: Length Of Line Exceeds Limit`

The length of a statement or a text line in the circuit file exceeds the limit. Most likely a statement is not terminated with a semicolon ";".

`CFP: Name Of Import 1POR Data Expected`

A port terminating impedance can be specified by imported one-port data. The keyword Z must be followed by the name of a set of imported one-port data (see Chapter 5).

`CFP: Name Of Import nPOR Data Expected`

Imported *S*, *Y* or *Z* data can be used to represent "black-box" linear subcircuits using the *n*PORDATA element. The keyword DATA must be followed by the name of a set of imported data (see Chapters 5 and 7).

`CFP: Name Of The Source Of The Include File Expected`

The keyword source can be used to indicate that an include file supplies data in an external format. It must be followed by a valid option name (see Chapters 9 and 11).

CFP: Nested Include Files Are Not Allowed

The "#include" statement can only be used in the main circuit file. It cannot be used in an include file.

CFP: No DC Definition Available

DC data and/or sweep set required for the intended operation.

CFP: No Large-Signal Data Available

Large-signal data supplied in the Data block of the circuit file is required for the intended operation.

CFP: No Measured Data Available

If an optimization goal refers to a response keyword or label, then measured data of that response at all the sweep points must be provided. Without the measured response, the goal cannot be calculated. See Chapter 9. Multi-device measurements must be supplied for statistical display of the data and statistical modeling. See Chapters 10 and 11.

CFP: No Optimization Goals Specified

Each specification set in the Specification block must contain at least one optimization goal (explicitly or implicitly, see Chapter 9).

CFP: No Output Keyword Specified

The FORMAT statement of each data set in the Data or ImportData block must contain the appropriate data keywords, such as RY11 and IY11. When HarPE is used as a child of OSA90/hope for simulation the user-defined output labels must be given in the Sweep block for nominal simulation or in the MonteCarlo block for statistical analysis.

CFP: No S-parameter Definition Available

S-parameter data and/or sweep set required for the intended operation.

CFP: No Statistical Variables Have Been Defined

In order to invoke the Monte Carlo analysis at least one statistical parameter must be defined in the circuit file.

`CFP: No Sweep Definition Available`

Certain display options, such as Psweep (power sweep), requires at least one sweep set with the appropriate simulation type and/or sweep range be defined in the Sweep block of the circuit file. See Chapter 8.

`CFP: No Yield Specification Defined In MONTECARLO Block`

For the yield estimated by Monte Carlo analysis to be available, the MonteCarlo block must contain at least one yield specification (see Chapter 10).

`CFP: Not A Valid Numerical Unit`

See Chapter 3 for the list of valid physical units.

`CFP: Number Of Arguments Inconsistently Specified`

The number of arguments specified in a text macro reference is inconsistent with the macro definition.

`CFP: Number Of Optimization Variables Exceeds Limit`

Too many optimization variables have been defined. See Chapter 1 for the size limit.

`CFP: Number Of Ports Mismatched Between Data And Usage`

The number of ports specified by the *n*PORDATA element in the Model block must be consistent with the corresponding *S*, *Y* or *Z* data supplied in the Importdata block.

`CFP: Number Of Statistical Variables Exceeds Limit`

Too many statistical variables are defined in the circuit file. Reduce the number of statistical variables. See Chapter 1 for the size limit.

`CFP: Numerical Entry Expected`

A character string is encountered where a numeric entry is expected.

`CFP: Only One Bias Port Is Defined: VD/IG Not Allowed`

If the circuit model contains only one bias port, then only VG and ID are available. VD and IG are available only if two bias ports are defined. See Chapter 5.

CFP: Only One Sweep Label Allowed Per MonteCarlo Set

Each Monte Carlo sweep set in the MonteCarlo block can contain only one sweep label (bias voltage, frequency or input power) which is defined with multiple sweep points. If you wish to define multidimensional sweeps for Monte Carlo analysis, you have to create multiple sweep sets (see Chapter 10).

CFP: Only One X Sweep Label Allowed Per Sweep Set

Each sweep set in the Sweep block may contain only one user-defined sweep label (in addition to the built-in sweep labels). See Chapter 8.

CFP: Output Keyword Cannot Be A PARAMETER, Use FORMAT

Responses such as MS*ij* and POUT*k* cannot be defined in the PARAMETER statement in the Data block of the circuit file. They must be defined by the FORMAT statement. See Chapter 9.

CFP: Output Keywords Incompatible With Input Keywords

The input and output keywords in the same sweep set, specification set, or data set must be consistent. You cannot, for example, mix PIN (large-signal input keyword) with MS11 (small-signal output keyword).

CFP: Port Definition Must Be Last In MODEL Block

Port definition is required. It must be the last statement in the Model block and it must be followed by the End statement.

CFP: Range Definition Incomplete: Input Keyword Missing

All the input keywords required to completely define a sweep set, specification set or data set must be specified. For instance, for a two-port circuit, a DC sweep set must contain both VG and VD keywords.

CFP: Reference Impedance RREF Must Be Positive

The reference impedance for *S* parameters must be a positive constant.

CFP: Required Circuit File Block Missing

See Chapter 3 for the list of circuit file blocks.

CFP: Sample Size Of Statistical Measurements Too Large

The sample size of the multi-device measurements for statistical modeling is too large. See Chapter 1 for the size limit.

CFP: Size Of File Exceeds Limit

The size of a circuit file or data file exceeds the limit. See Chapter 1 for the size limit. This limit applies to the size of a single file, hence you can break up a large file into several smaller files and use the "#include" preprocessor directive to link them together.

CFP: The Last File Block Not Closed With END

The last block in the circuit file is not properly closed with an End keyword.

CFP: The Node Referenced Has Not Been Defined

Reference to an undefined node is encountered in the Model block.

CFP: Tolerance Undefined For Statistical Variable

When defining a statistical parameter, you must specify the tolerance or standard deviation. See Chapter 10.

CFP: Too Few Distinct Nodes Specified Than Expected

Each library element requires a predefined number of distinct nodes. Check Chapters 6 and 7 for the required and optional nodes.

CFP: Too Few Time Samples For Waveform Data

The waveform measurement contains too few time samples for a reasonably accurate conversion to frequency-domain power spectrum data.

CFP: Too Many Goals In A Single Specification Set

Too many goals specified in a single specification set in the Specification block. You can define a number of specification sets and distribute the responses and goals among the different specification sets. See Chapter 9.

CFP: Too Many Nodes Specified Than Expected

The proper numbers of nodes for the library elements are listed in Chapters 6 and 7.

CFP: Too Many Steps In A Single Range Definition

The number of steps in a single sweep range exceeds the limit. See Chapter 1 for the size limit.

CFP: Too Many User-Defined Output Labels

The number of user-defined responses in a single sweep set or specification set exceeds the limit. You can define a number of sweep or specification sets and distribute the user-defined responses among the different sets.

CFP: Trailing Index Expected For This Keyword

Certain keywords require trailing index or indices. For instance, MS*ij* requires two indices (*i* and *j*). MS or MS1 is incorrect usage.

CFP: Unbalanced Or Missing Parenthesis

Parentheses "()" must appear in pairs in the circuit file. Check for a matching number of left and right parentheses.

CFP: Undefined Symbol Referenced

Reference to an undefined label is encountered in the circuit file.

CFP: Unrecognizable Entry

Unrecognizable keywords or entries. Check the syntax.

CFP: Unsuitable Optimizer

The Minimax optimizer is not suitable for multi-device parameter extraction in indirect statistical modeling, check the Control block for the default optimizer you specified.

CFP: User-Defined Goal Cannot Involve Variables

In the Specification block, the responses can be defined as functions of optimization variables, but the goals must be constants or measured data. See Chapter 9.

CFP: User Entry Expected Following A Keyword

An appropriate user entry is expected to follow the keyword.

`CFP: Weighting Factors Cannot Be Negative`

Weights in the Specification block of the circuit file cannot be negative.

`CFP: Value Exceeds Limit: Too Large Or Too Small`

The value specified exceeds the limits.

`CFP: X Sweep Must Be A Constant Or Optimizable Label`

User-defined sweep label in the Sweep block must first be defined in the Expression block to represent a numerical constant or an optimization variable. In other words, labels which represent formulas cannot be used as sweep labels.

`DCM: Too Many Bias-Frequency Combinations`

Too many data points specified in the Specification block for optimization. Select a subset of data to reduce the total number of points.

`DCM: Too Many Bias-Frequency-Power Combinations`

The total number of bias/frequency/input power combinations defined in the Specification block for optimization exceeds the limit.

`FPE: Floating Point Exception`

Floating-point operation results in an error.

`FPE: Floating Point Overflow`

Floating-point operation results in overflow.

`FPE: Math Function Argument Domain Invalid`

Floating-point error: argument for math library function is out of its domain of definition. For instance, reference to log(x) when x is negative.

`FPE: Math Function Argument Singular`

Floating-point error: argument for math library function is "singular". For instance, reference to tan(x) when x is PI/2.

GOO: Cannot Find File Directory Path

Cannot find the specified path (directory). Check the path name.

GOO: Cannot Open File

Cannot open a disk file for reading or writing because the file/path name is illegal, or the file/path does not exist, or the file/path is protected from the intended operation.

GOO: Error Writing To Disk File

Error when writing to a disk file due to file protection or insufficient disk space.

GOO: Exceed Demo Limits On Number Of Nodes

The copy of HarPE you are using is a demo copy with a restricted number of circuit nodes.

GOO: Exceed Demo Limits On Optimization Variables

The copy of HarPE you are using is a demo copy with a restricted number of optimization variables.

GOO: Exceed Demo Limits On Statistical Variables

The copy of HarPE you are using is a demo copy with a restricted number of statistical parameters.

GOO: File Not Found

Cannot find the specified file. Check the file/path name.

GOO: Invalid Color Map Entry Encountered

An unrecognizable entry is found in the specified color map file. See Chapter 12 for color map file syntax.

GOO: No Circuit File Has Been Read

A valid circuit file must be loaded into HarPE before other operations. Invoke the menu option HarPE.File and read a circuit file in the editor.

`GOO: Not Enough Memory For The Intended Operation`

Cannot allocate sufficient memory for the intended operation. Reduce the number of variables, sweep steps and/or data points.

`GOO: Size Of File Exceeds Limit`

The size of a circuit file or data file exceeds the limit. See Chapter 1 for the size limit. This limit applies to the size of a single file, hence you can break up a large file into several smaller files and use the "#include" preprocessor directive to link them together.

`GOO: Too Many Macro Keys In A Single Macro File`

The size of a macro file exceeds the limit. See Chapter 1 for the size limit. You can break up a large macro file into several smaller ones and chain them together (see Chapter 12).

`GOO: Unknown Run-Time Option`

Unrecognizable entry or entries are encountered on the command line when you start HarPE. See Chapter 12 for the set of valid run-time options.

`GOO: Unrecognizable Macro Key Name In Macro File`

An unrecognizable entry is found in the specified macro command file. See Chapter 12 for macro command file syntax.

`GRF: No Data Range for Plot`

A graphical plot requires at least two distinct points for the X-axis. Perhaps you have specified too closely spaced sweep points.

`HAR: Floating Node Encountered In MODEL`

Every node defined in the Model block of the circuit file must be referenced at least twice (i.e., at least two elements must be connected to a node). Floating node, i.e., a node defined for an element but not connected to any other element or port, is not permitted.

You can use the OPEN element (see Chapter 7) for intentionally introduced floating nodes.

`HAR: Inconsistent Extrinsic And Port Configuration`

The nodes specified for the input and output ports differ from the input and output nodes specified for the extrinsic super-component (EXTRINSIC*i*) in the Model block of the circuit file. See Chapters 5 and 7. This message is also displayed if no extrinsic element is defined in the Model block.

HAR: Invalid Intrinsic And Extrinsic Connection

The nodes specified as the gate (base), drain (collector) and source (emitter) of the intrinsic device differ from the corresponding nodes specified for the extrinsic super-component (EXTRINSIC*i*) in the Model block of the circuit file. See Chapters 5 and 7.

HAR: No Circuit Components Defined

No circuit element has been defined in the Model block of the circuit file.

HEF: DC Simulation Difficult To Converge

Difficulties in solving the nonlinear DC circuit model. It could be due to wild model parameter values. Try a different set of parameters.

HEF: Harmonic Simulation Difficult To Converge

Difficulties in nonlinear simulation. Increase the number of harmonics (HARM) specified in the Sweep block in the circuit file. It could also be due to wild model parameter values. If you continue to experience this problem, please contact Optimization Systems Associates Inc. at 416 628 8228.

HEF: Matrix To Be Factorized Is Singular

A singular matrix encountered during simulation. Check your circuit for possible reasons such as open or short circuited paths, etc. Try to redefine the circuit or sweep parameters such as frequency, or modify circuit parameters, if possible.

HEF: Simulation Did Not Converge At Starting Point

Difficulties in nonlinear simulation at the starting point of optimization. Try a different starting point.

HEF: Too Many Harmonics: Not Enough Memory

The number of harmonics specified by the HARM keyword exceeds the limit.

LSM: Complex Value Encountered For DC Calculations

A non-zero imaginary part of a complex number has been encountered at DC. Most likely this error is caused by an inappropriate angle of the controlling coefficient in a controlled source.

LSM: Diode And Extrinsic Connection Not Allowed

The extrinsic super-components EXTRINSIC1, EXTRINSIC2, EXTRINSIC3 and EXTRINSIC4 are intended for transistors only. They cannot be used *alone* to describe the linear environment of diode circuits.

LSM: Invalid External Or Bias Port Configuration

The definition of input, output and/or bias ports in the Model block of the circuit file violates the topological requirements of HarPE.

LSM: Linear Subcircuit Singular

The linear subcircuit defined is singular.

LSM: Negative Or Zero Element Parameter Encountered

A zero or negative value of a parameter has been encountered during circuit simulation. Check the circuit file for erroneous entry. Consult Chapters 6 and 7 for the proper ranges of the parameter values. Also check user-defined labels that may evaluate to zero or a negative value.

LSM: Port Termination Singular: Open Circuit

The value of the termination impedance of an external or bias port is singular (infinity). An open circuit port is not allowed.

LSM: Relative Dielectric Constant Less Than One

A less than one value of the relative or effective dielectric constant has been encountered during circuit simulation. Check the circuit file for erroneous entry. Also, if applicable, check user-defined labels involved in evaluation of that parameter.

LSM: Transmission Line Attenuation Must Be Non-negative

A negative value of the transmission line attenuation has been encountered during circuit simulation. Check the circuit file for erroneous entry. Also, if applicable, check user-defined labels involved in evaluation of that parameter.

LSM: Zero Characteristic Impedance

A zero or negative value of the transmission line characteristic impedance has been encountered during circuit simulation. Check the circuit file for erroneous entry. Also, if applicable, check user-defined labels involved in evaluation of that parameter.

LSM: Zero Or Negative Reference Frequency For Scaling

A zero or negative value of the reference frequency for scaling the transmission line electrical length or attenuation has been encountered during circuit simulation. Check the circuit file for erroneous entry. Also, if applicable, check user-defined labels involved in evaluation of that parameter.

LSM: Zero Or Negative Transmission Line Length

A zero or negative value of the transmission line length has been encountered during circuit simulation. Check the circuit file for erroneous entry. Also, if applicable, check user-defined labels involved in evaluation of that parameter.

MSL: Dielectric Constant Value Too Large

The dielectric constant of a microstrip element is found to exceed the upper limit of the model validity range. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Dielectric Constant Value Too Small

The dielectric constant of a microstrip element is found to exceed the lower limit of the model validity range. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Frequency Too High For Microstrip Components

The simulation frequency for a microstrip element is found to exceed the upper limit of the model validity range.

MSL: Improper Dielectric Constant

The dielectric constant of a microstrip element is found to be invalid during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Microstrip Shielding Height < Substrate Thickness

The shielding height of a microstrip element is found to be smaller than the substrate thickness. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Microstrip Slit Too Deep

The depth of a microstrip slit is found to exceed the upper limit of the model validity range. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Microstrip Slit Too Wide

The width of a microstrip slit is found to exceed the upper limit of the model validity range. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Microstrip Step Has Same Widths (W1 = W2)

The widths of a microstrip step cannot be defined with the same value.

MSL: Microstrip Width-Height Ratio Too Large

The width-height ratio of a microstrip element is found to exceed the upper limit of the model validity range. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Microstrip Width-Height Ratio Too Small

The width-height ratio of a microstrip element is found to exceed the lower limit of the model validity range. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Negative Microstrip Conductor Resistivity

The conductor resistivity of a microstrip element is found to have a negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Negative Microstrip Conductor Thickness

The conductor thickness of a microstrip element is found to have a negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Negative Microstrip Surface Roughness

The surface roughness of a microstrip element is found to have a negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Negative Substrate Loss Tangent

The loss tangent of a microstrip substrate is found to have a negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Zero Or Negative Microstrip Gap Spacing

The spacing of a microstrip gap is found to have a zero or negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Zero Or Negative Microstrip Line Length

The length of a microstrip line is found to have a zero or negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Zero Or Negative Microstrip Line Spacing

The spacing of a microstrip line is found to have a zero or negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Zero Or Negative Microstrip Line Width

The width of a microstrip line is found to have a zero or negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Zero Or Negative Microstrip Shielding Height

The height of a microstrip shielding is found to have a zero or negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Zero Or Negative Microstrip Slit Depth

The depth of a microstrip slit is found to have a zero or negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Zero Or Negative Microstrip Slit Length

The length of a microstrip slit is found to have a zero or negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MSL: Zero Or Negative Microstrip Substrate Height

The height of a microstrip substrate is found to have a zero or negative value during circuit simulation or optimization. If the parameter is assigned by a label, make sure that the label evaluates to a proper value.

MTH: Bad Reference Impedance Rref For S-Pars

The reference impedance for *S* parameters (RREF) defined in the Sweep or Data block is either too large, zero, or leads to a divide-by-zero. Redefining RREF should overcome the problem.

MTH: Divide By Zero In Insertion Loss Calculation

An attempt to divide by zero has been made in the calculation of the insertion loss. Check the *S* parameters. Note that actual terminations, whether default or specified, are involved in the calculations.

MTH: Insufficient Data For Spline Interpolation

Cubic spline interpolation employed by HarPE requires a minimum of three supplied data points. Please provide more data points to avoid this error.

MTH: Linear Y Matrix Singular

The *Y* matrix of the linear subcircuit defined is found to be singular in matrix factorization.

MTH: Relevant Hybrid Matrix Does Not Exist

The hybrid matrix needed in simulation of your current-driven circuit does not exist, so the simulation cannot be completed. Check your circuit for possible reasons such as open or short circuited paths, etc. Try to redefine the circuit or sweep parameters such as frequency, or modify circuit parameters, if possible.

MTH: Zero Reference Impedance Rref For S-Pars

The reference impedance for *S* parameters (RREF) defined in the Sweep or Data block must be greater than zero.

`OPT: No Error Function Defined`

No specification has been defined for optimization. See Chapter 9.

`OPT: No Variable Defined`

In order to perform optimization, at least one variable must be defined in the circuit file. See Chapters 4 and 5.

`SED: Cursor Already At End Of File`

Screen editor warning: the cursor can not move beyond the end of file.

`SED: Cursor Already At Top Of File`

Screen editor warning: the cursor can not move beyond the beginning of file.

`SED: File Buffer Full`

Screen editor warning: the size of file exceeds the limit. See Chapter 1 for the size limit. This limit applies to the size of a single file, hence you can break up a large file into several smaller files and use the "#include" feature to link them together.

`SED: File Is Absolute Read-Only: Cannot Be Edited`

Screen editor warning: the current file is read-only and therefore it is protected from any modifications.

`SED: File Is Read-Only`

Screen editor warning: the current file is read-only. You can change the mode from read-only to editing (see Chapter 2).

`SED: Invalid File Directory Path Name`

Screen editor warning: the specified path name is not valid.

`SED: No File Found In The Current Working Directory`

Screen editor warning: no file found.

SED: No Match String Found

Screen editor warning: no match is found for the specified string for Search and/or Replace operation.

SED: No Printer Connected Or Accessible

Screen editor warning: no printer connected for printing.

SED: No Trailing Blank Encountered

This is just to inform you that the file did not contain any trailing blanks, and therefore no trimming has taken place.

STM: Correlation Matrix Is Not Positive Semi-Definite

By definition, correlation matrices must be positive semidefinite.

STM: Duplicate Entries In SAMPLE FORMAT

Duplicate names found in the FORMAT statement in the Sample block of the circuit file.

STM: FORMAT Missing In The SAMPLE Block

The Sample block of the circuit file must contain a FORMAT statement which identifies the statistical data contained in the block. See Chapters 10 and 11.

STM: Illegal FORMAT In The SAMPLE Block

An unrecognizable item found in the FORMAT statement in the Sample block of the circuit file.

STM: Inaccurate Factorization Of Correlation Matrix

Numerical inaccuracy encountered in the factorization of the correlation matrix.

STM: Insufficient Measurement For Display

Multi-device measurements must be supplied for graphical display of histograms or cumulative probability distributions of the measured data. See Chapters 10 and 11.

STM: Insufficient Measurement For Statistical Modeling

Multi-device measurements must be supplied for statistical modeling (see Chapter 11).

STM: MonteCarlo Block Required

The MonteCarlo block is required for the Monte Carlo simulation and statistical match options. See Chapters 10 and 11.

STM: SAMPLE Block Required For Postprocessing

The Sample block is required for the statistical postprocessing options. See Chapter 11.

STM: Sample Size Too Large For Postprocessing

The sample size of the data contained in the Sample block of the circuit file exceeds the limit (i.e., too many outcomes). See Chapter 1 for the size limit.

STM: Sample Size Too Small For Postprocessing

The sample size of the data contained in the Sample block of the circuit file is too small (i.e., too few outcomes).

STM: Statistical Model Already Exists

The current circuit file already contains a statistical model. Statistical modeling will not proceed.

TLP: A Text Macro Argument Name Expected

Commas are used to separate the arguments of macro functions. This message may indicate that an extraneous comma is encountered which is not followed by an argument name or that an expected argument is missing.

TLP: Illegal Operator Or Reserved Symbol

See Chapter 4 for the list of reserved symbols (characters).

TLP: Illegal Or Missing Text Macro Name

The preprocessor directive "#define" must be followed by a text macro name.

TLP: Macro Cannot Define Explicit Variable: Use Label

Text macro is a preprocessor feature for the user's convenience. They are expanded into the corresponding substitution text by the file parser. A text macro cannot contain explicit definition of optimization variables, for it cannot be properly updated after optimization, and it may cause ambiguity when it is referenced more than once.

`TLP: Nested Definition Of Text Macros Is Illegal`

The definition of a text macro may contain references to other macros which are already defined, but it cannot embed another macro definition.

`TLP: Not A Valid Number`

Most likely, a field (token) in the circuit file is encountered which begins with a digit but cannot be parsed as a valid numerical value.

`TLP: Not A Valid Numerical Unit`

See Chapter 3 for the list of valid physical units.

`TLP: Overflow: Numerical Value Exceeds Limit`

The value of a numeric entry in the circuit file exceeds the limit.

`TLP: Too Many Arguments For A Text Macro`

The number of arguments for a text macro function is limited to 512.

`TLP: Too Many Nested Levels Of Text`

The number of levels of nested text macro references (i.e., references to other macros contained within the definition of a text macro) is limited to 8.

`TLP: Too Many Tokens (Fields) In A Single Record`

Too many tokens (fields) in a single statement in the circuit file. Most likely you forgot to terminate a statement with a semicolon ";".

`TLP: Unbalanced Quotation Mark (Not Closed)`

Quotation marks must be used in pairs. The closing quotation mark is missing.

`XLT: Device Model Not In Libra`

The device model defined in the circuit file is not supported by Libra.

`XLT: Device Model Not In Microwave Harmonica`

The device model defined in the circuit file is not supported by Microwave Harmonica.

XLT: Expressions Cannot Be Translated

One or more of the device model parameters are defined by expressions unique to HarPE and therefore cannot be translated into Libra or Microwave Harmonica format.

XLT: No Nonlinear Device Model Defined In Circuit File

The Model block in the circuit file does not contain any nonlinear device model for translation.

Index

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

OSA

Dundas, Ontario, Canada