OSA

NONLINEAR CIRCUIT OPTIMIZATION IN A FRIENDLY ENVIRONMENT

OSA-92-OS-12-V

September 30, 1992

Optimization Systems Associates Inc.

Dundas, Ontario, Canada

NONLINEAR CIRCUIT OPTIMIZATION

IN A FRIENDLY ENVIRONMENT

OSA-92-OS-12-V

September 30, 1992

NONLINEAR CIRCUIT OPTIMIZATION IN A FRIENDLY ENVIRONMENT

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

OSA90/hope™ Version 2.0

general nonlinear circuit simulation and optimization
customization oriented optimization shell
analytically unified
- large-signal harmonic balance analysis
- small-signal AC analysis
- nonlinear DC analysis

interacts with external simulators
- Empipe™
- Spicepipe™
- user's simulators

statistical analysis and yield optimization

HarPE™ Version 1.6

parameter extraction and device characterization
single-device circuit simulation and optimization
statistical modeling
Monte Carlo analysis

OSA90™ (OSA90/hope™ Engine)

OSA90/hope™ Optimization

state-of-the-art gradient-based optimizers with a proven track record in electrical circuit and system optimization

- L1
- least squares (L2)
- minimax
- quasi-Newton
- conjugate gradient
- simplex
- random
- yield (one-sided L1) for statistical centering

exact or approximate gradient

specification and goal definition

quadratic modeling of functions and gradients

sensitivity displays help the user to select the most crucial variables for optimization

OSA90/hope™ Circuit Features

general nonlinear circuit simulation and optimization
physics-based yield optimization

analytically unified simulation:
DC/small-signal/large-signal harmonic balance

arbitrary topology
multitone, multisource excitations
nonlinear sources controlled by arbitrary voltages

symbolic subcircuit definition: linear and nonlinear
voltage labels (probes)

comprehensive device library
microstrip models
user-definable models

OSA90/hope™ Math Features

variable definition

expressions including conditional *if else* structures

extensive math library

vector and matrix operations

- multiplication
- transposition
- inverse
- LU factorization
- solving linear equations

vector and matrix elements fully optimizable

built-in transformations, including DFT

OSA90/hope™ Graphics

continuous, point, bar plots

parametric and multiple plots

histograms and run charts

sensitivity displays

waveforms

Smith chart and polar plots

user-defined legends, colors, views

graphics zoom

graphics hardcopies (PCL printers and HPGL files)

Built-in Device Models

Diode

FET:	Curtice
	Materka
	Statz
	Trew (physics)
	modified Trew (physics)
	KTL (physics)
Bipolar:	Gummel-Poon

Preprogrammed User Models

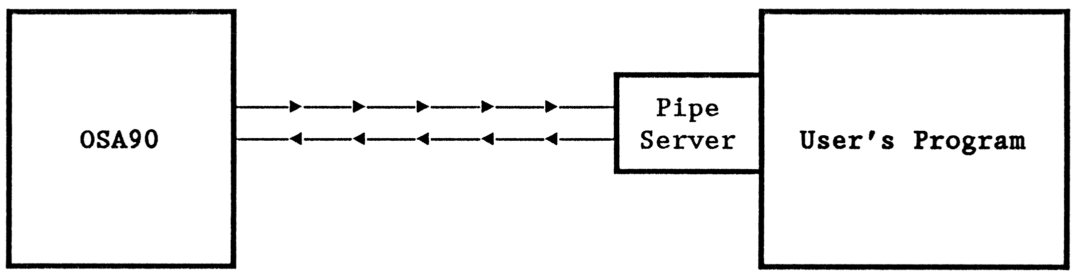
FET:	Curtice
	KTL (physics)
	Materka
	Plessey
	TOM

Custom Designed Models

OSA90/hope™ Datapipe™

Datapipe™: predefined protocols for UNIX pipes

ready-to-use to facilitate high-speed data connections to and from the user's software



typical READ and WRITE statements are used to receive and send data

a small pipe server (about 350 lines) establishes the protocols; OSA provides the source code of the pipe server

maintains complete security of user's software: OSA does not need access to the user's source code

Datapipe™ Pre- and Post-Processing

the user can define constants, variables and expressions in the input file; Datapipe handles all the parsing

the inputs received by the user's software can be both pure numerical values and character strings; any of the variables to be passed to the user's program(s) can be preprocessed

the outputs from the user's program can be labelled and further postprocessed using standard mathematical functions and operations

any of the output values returned from the user's program(s) can be passed as input to another program

the user can define his/her own responses for graphical display, numerical output, optimization or statistical analysis

an in-house program, e.g., a circuit simulator, can be turned into a powerful CAE system, enhanced by an elegant user interface and quality graphics

SIM Protocol

outputs from the user's program are individually labelled

syntax:

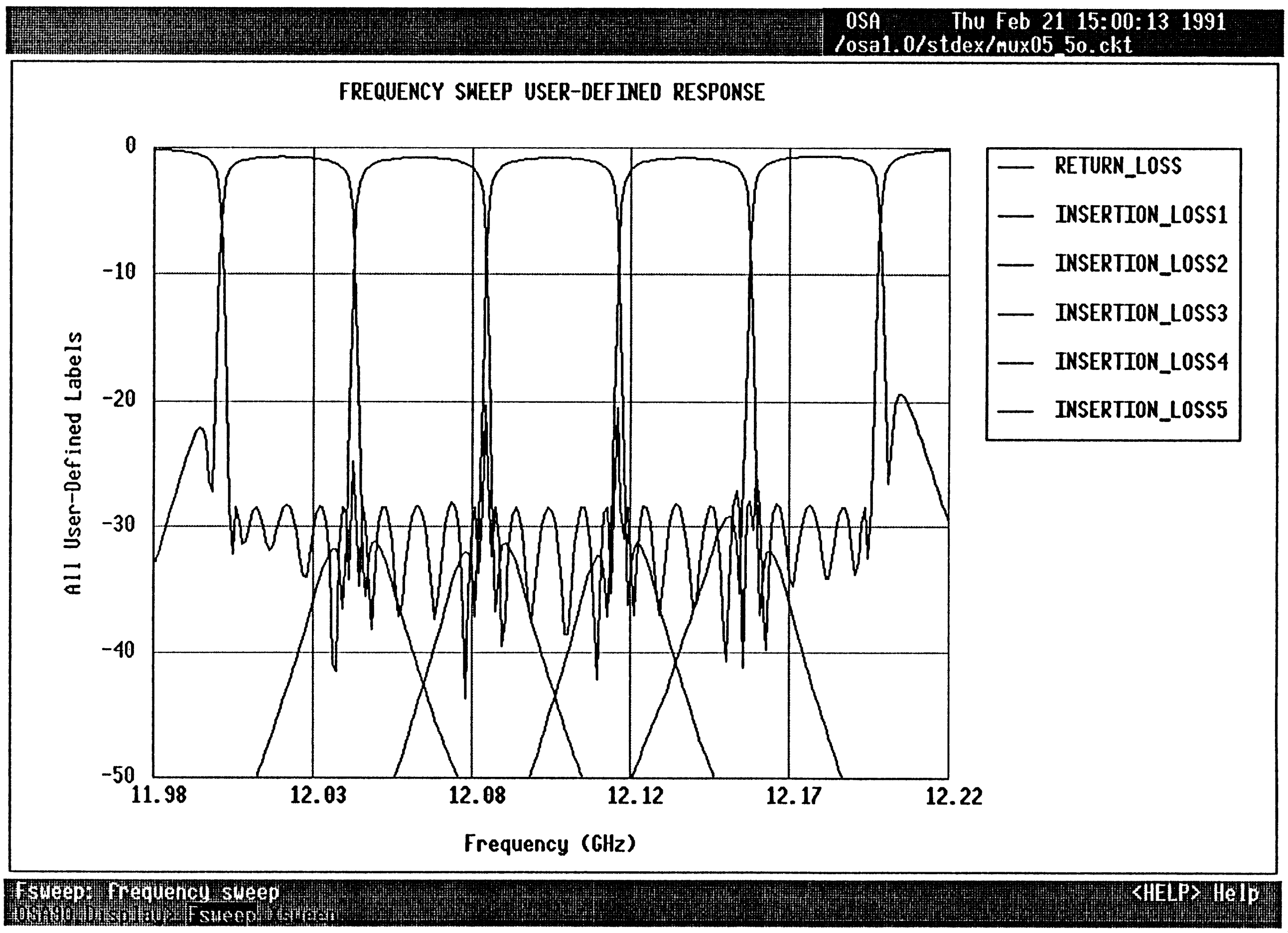
```
Datapipe:  SIM  FILE = filename
                N_INPUT = n        INPUT = (x1, ..., xn)
                N_OUTPUT = m       OUTPUT = (y1, ..., ym);
```

the user specifies:

filename	-	the name of user's executable program
n	-	the number of numerical values to be passed to the user's program
x1, x2,..	-	individual labels and/or values to be passed to the user's program
m	-	the number of numerical values that will be returned from the user's program
y1, y2,..	-	individual labels for the returned values

Optimization of 5-Channel Multiplexer

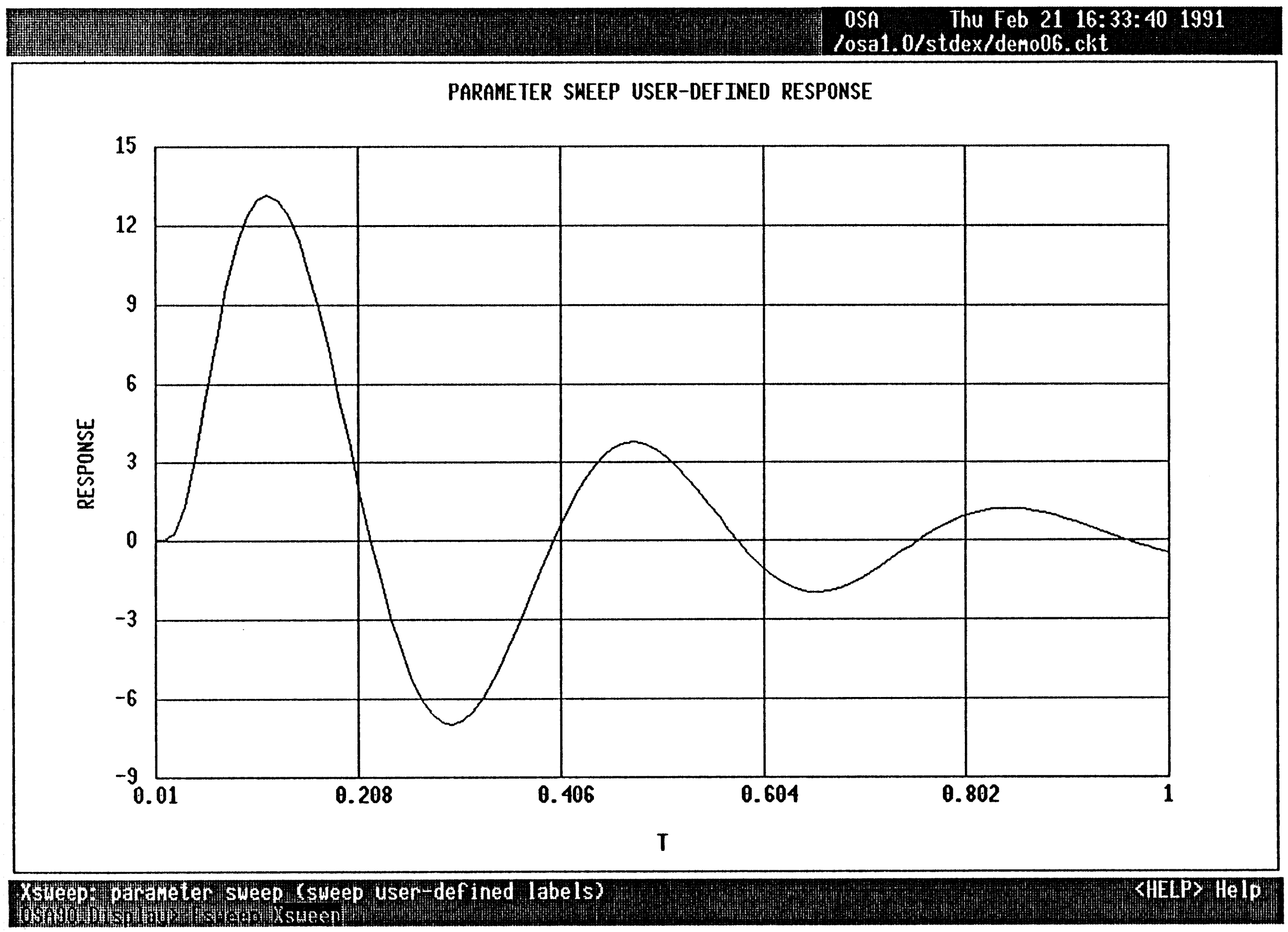
common port return loss and individual channel insertion loss responses after optimization:



OSA90 and a multiplexer simulator interacted for 3724 iterations

Time-Domain Simulation of a Feedback Network

transient response to a pulse excitation:



OSA90 and a time-domain simulator interacted 100 times

COM Protocol

similar to SIM, but additionally allows strings to be passed to or returned from the user's program

syntax:

```
char x2[] = "string to be passed to the user's program";
Datapipe:  COM FILE = filename
                N_INPUT = n   INPUT = (x1, x2, ...)
                N_OUTPUT = m  OUTPUT = (y1, char y2[15], ...);
```

the user specifies:

filename	-	the name of user's executable program
n	-	the number of arguments to be passed to the user's program
x2,..	-	labelled string to be passed to the user's program
m	-	the number of arguments that will be returned from the user's program
y2,..	-	labelled strings to be returned from the user's programs

Direct Optimization through Datapipe™

the user supplies or defines individual error functions (FUN protocol)

the objective function is formulated and its minimization executed entirely by OSA90/hope

OSA90/hope can take advantage of user-supplied gradients (FDF protocol)

alternatively, OSA90/hope will generate and use approximate gradients

FUN Protocol

the user's program provides directly the error functions for the optimizer; OSA90/hope generates gradients

syntax:

```
Datapipe:  FUN  FILE = filename       NAME = id_name
                N_INPUT = n           INPUT = (x1, ..., xn)
                N_OUTPUT = m;
```

the user specifies:

filename	-	the name of executable user's program
n	-	the number of numerical values to be passed to the user's program
x1, x2,..	-	individual labels and/or values to be passed to the user's program
m	-	the number of error values that will be returned from the user's program
id_name	-	name for reference elsewhere in the input file

FDF Protocol

the user's program provides directly for the optimizer both the error functions and gradients

the only difference w.r.t. the FUN type is that $m(n + 1)$ values will be returned from the user's program, i.e., for each of m error functions the error value and n partial derivatives will be returned

syntax:

```
Datapipe:  FDF  FILE = filename        NAME = id_name
                N_INPUT = n            INPUT = (x1, ..., xn)
                N_OUTPUT = m;
```

OSA90/hope™ Connections

Empipe™ merges *em*™

from Sonnet Software, Inc.

for direct field-level optimizable microstrip designs
under circuit-level linear/nonlinear analysis

Spicepipe™ merges SPICE-PAC

for time-domain simulation, noise analysis, etc.
nominal optimization
yield optimization

OSA90/hope™ Input File Overview

Model/Expression
...
End

Sweep
...
End

Specification
...
End

MonteCarlo
...
End

ImportData
...
End

Control
...
End

Statistics
...
End

OSA90/hope™ Input File Expression Block

```
Expression

  x1: ? -0.5 ?;
  x2: ? 0.5 ?;
  z1: x1 * x1 + x2 * x2;
  z2: x1 - x2;

  Datapipe: SIM  FILE = user_program_name
                 N_INPUT = 4      INPUT = (freq, x1, z1, z2)
                 N_OUTPUT = 2     OUTPUT = (gain, sk);

End
```

OSA90/hope™ Input File Model/Expression Block

Model

expressions

datapipes

circuit elements

VLABEL ...

VSOURCE ...

PORT ...

CIRCUIT ... (response labels created here)

expressions post-processing responses

End

Example of OSA90/hope™ Model/Expression Block

```
Model        ! a diode mixer

        SRL  1  0  R=0.01  L=1.8UH;
        RES  1  2  R=10.0;  !  resistance in diode model
        CAP  3  0  C=2.8PF;
        SRL  3  4  R=0.01  L=.1UH;
        SRL  1  5  R=10KOH  L=20UH;

! use built-in nonlinear diode model

        DIODE     2  3  IS: 1.e-14  N: 1  TEMP: 295  VJ: 0.9
                  CJ0: 0.001PF;

! define bias source

        VSOURCE 5 0  NAME=BIAS VDC=0.5V;

! define user labels

        Power_LO: -5DBM;
        Power_RF: -20DBM;

! define input (two-tone excitation) and output ports

        PORT 1 0  NAME=in R=1KOH  P=Power_LO  P2=Power_RF;
        PORT 4 0  NAME=out R=1KOH;

        CIRCUIT;
```

Example of OSA90/hope™ Model/Expression Block

```
! use built-in functions for polar-rectangular transformation;
! after "CIRCUIT" statement the labels MVout and PVout are
! automatically generated by the program

    MP2RI(MVout, PVout, RVout[0:13], IVout[0:13]);

! define more user labels

    IF_freq: 7MHZ;
    T1: 1 / IF_freq;
    Time: 0;
    K: 0;

! use built-in function for frequency-to-time transformation

    DFT_FT(RVout, IVout, SPECTRAL_FREQ, Time, Vout_T);

End
```

OSA90/hope™ Symbolic Subcircuit Definition

```
! define a symbolic subcircuit

#define trewmodel (gate, drain, cr1, cr2, cr3, cr4, $) {

SRL  @gi$  gate       R: P_Rg  L: P_Lg;
SRL  @di$  drain      R: P_Rd  L: P_Ld;
SRL  @si$  @ground    R: P_Rs  L: P_Ls;

FETT1    @gi$ @di$ @si$
  L:  gate_L   {NORMAL SIGMA=3.5% CORRELATION=FET[cr2]}
  A:  gate_a   {NORMAL SIGMA=3.5% CORRELATION=FET[cr1]}
  ND: doping   {NORMAL SIGMA=7.0% CORRELATION=FET[cr4]}
  W:  gate_W   {NORMAL SIGMA=2.0% CORRELATION=FET[cr3]}
  EPSR: Eps    EC:   Ecv     VS:   Vsv
  U0:   U0v    VBI:  biv     D:    diffu
  LAMBDA:  Lamdav    ALPHA: alpha    TAU:   Tauv ;

}

Model

! use the symbolic subcircuit

      trewmodel(@gate1, @drain1, 1, 2, 3, 4, FET1);
      trewmodel(@gate2, @drain2, 5, 6, 7, 8, FET2);

End
```

OSA90/hope™ Input File Sweep and Specification Blocks

```
Sweep

    X1: from -1.0 to 1.0 step 0.1  F1 F2;
    X2: from -1.0 to 1.0 step 0.1  F1 F2;

    FREQ:   from 0.01GHZ to 1GHZ step 0.01GHZ
            from 1.1GHZ to 1.4GHZ step=0.05GHZ
            Gain, Insertion_loss;
    C6:     from 0.1NF to 0.5NF step 0.025NF  FREQ: 1ghz
            Gain, Insertion_loss;

End

Specification

    F1 < 1.0,  F1 > 3.0,  F2 < -3.0;

    FREQ:   from 0.02GHZ to 1GHZ step 0.02GHZ
            Insertion_loss < 0.53;
    FREQ:   1.3GHZ
            Insertion_loss > 52;

    x1:     from 0 to 10 step 1
            z1 < 10.3;

End
```

OSA90/hope™ Input File Sweep Block

```
Sweep

! small-signal simulation

AC:   FREQ: from 6GHZ to 15GHZ step 0.1GHZ
      VG: -1.8  VD: 6  gain VSWR;

! harmonic balance simulation

HB:   IF_freq: 7MHZ 9MHZ
      FREQ=900MHZ  FREQ2=(FREQ + IF_freq)
      Time: from 0 to T1 N=1000
      Vout_T, Vin_T, Iin_T_mA
      Title="Mixer Analysis At Two Different IF Frequencies";

HB:   IF_freq: 7MHZ 9MHZ
      FREQ=900MHZ  FREQ2=(FREQ + IF_freq)
      K: from 0 to N_SPECTRA step=1
      SPECTRAL_FREQ[K]  MVout[K]  MVin[K]  MIin[K],
      Title="Mixer Analysis: Frequency Spectrum";

End
```

OSA90/hope™ Input File Specification Block

```
Specification

! small-signal specifications

AC:   FREQ: from 8GHZ to 12GHZ step=1GHZ
      VG: -1.8  VD: 6 gain > 12 gain < 16  VSWR < 2;
AC:   FREQ: 6GHZ  VG: -1.8  VD: 6 gain < 2;
AC:   FREQ: 15GHZ VG: -1.8  VD: 6 gain < 2;

End
```

OSA90/hope™ Input File MonteCarlo Block

```
MonteCarlo

AC:   N_OUTCOMES=100
      FREQ: from 6GHZ to 15GHZ step 0.1GHZ
      VG: -1.8  VD: 6  gain VSWR;
AC:   FREQ: from 8GHZ to 12GHZ step=1GHZ
      VG: -1.8  VD: 6 gain > 12 gain < 16  VSWR < 2;
AC:   FREQ: 6GHZ  VG: -1.8  VD: 6 gain < 2;
AC:   FREQ: 15GHZ VG: -1.8  VD: 6 gain < 2;

End
```

OSA90/hope™ Input File Statistics Block

```
Statistics

    Correlation: FET  dimension=12  format=full;

!A1   L1   W1   Nd1   A2   L2   W2   Nd2   A3   L3   W3   Nd3

1.00 0.00 0.00 -0.25 0.80 0.00 0.00 -0.20 0.78 0.00 0.00 -0.10
0.00 1.00 0.00 -0.10 0.00 0.80 0.00 -0.05 0.00 0.78 0.00 -0.05
...
...

End
```

Monte Carlo Analysis

variables which are subject to statistical variations are identified in the input file

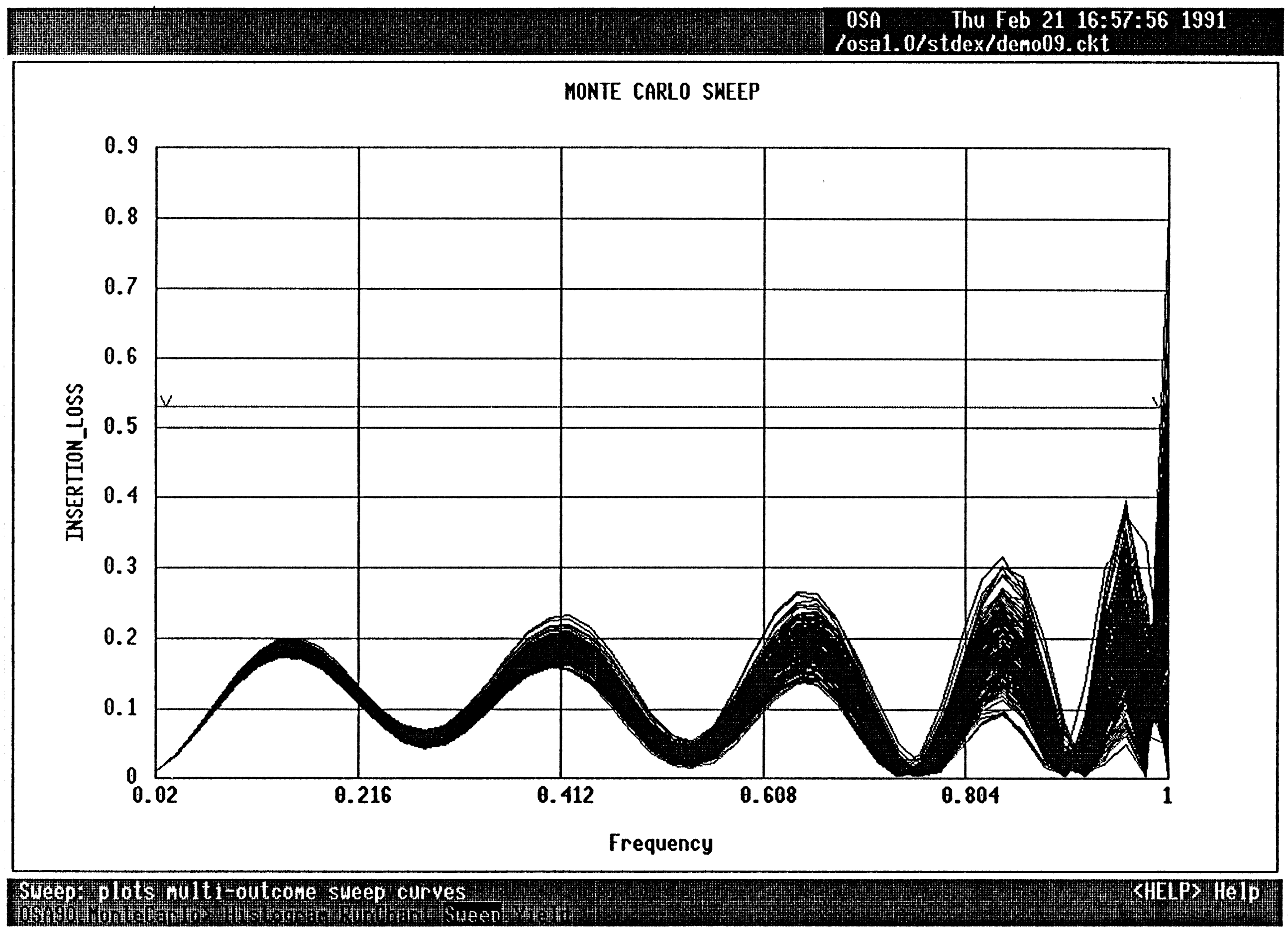
uniform, normal, exponential and lognormal distributions are available; variables can be correlated

OSA90/hope generates the specified number of statistical outcomes, simulates the circuit and/or calls the user's functions, checks design specifications, and evaluates yield

after Monte Carlo analysis is completed the user can examine histograms, run charts and sweep responses

Monte Carlo Analysis of 11-Element LC Filter

statistical insertion loss response:



200 statistical circuit outcomes

OSA90 and a filter simulator interacted 10,000 times

HarPE^{TM} Version 1.6

characterize devices

extract parameters

simulation-driven
data-driven

build equivalent circuit models

build physics models

simulate and optimize

single device circuits at DC, small-signal and large-signal harmonic balance

statistically model devices

estimate Monte Carlo yield

HarPE™ Measurements

power spectra

waveforms

S parameters

DC data

Cascade Microtech

MDIF

HarPE™ Modeling and Design

alter built-in models

define your own models

create functions and goals

use formulas and expressions

optimize linear and nonlinear parameters

model compatibility with OSA90/hope

HarPE™ Statistics

multi-device parameter extraction

statistical estimation

discrete approximation

graphics

- histograms
- scatter plots
- run charts
- zoom
- hardcopies

consolidated statistical models

- built-in or user-defined

Monte Carlo yield