Technical Report #9111

FIT–2, A SIMULATION–BASED PARAMETER EXTRACTION PROGRAM

by

W.M. Zuberek[†] and A. Konczykowska[‡]

[†] Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada A1B 3X5

[‡] Centre National d'Etudes des Télécommunications
Laboratoire de Bagneux
92220 Bagneux, France

September 1991

Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada A1B 3X5
tel: (709) 737–8627
fax: (709) 737–2009

An extended version of this report is available as a NOTE TECHNIQUE,
Centre National d'Etudes des Télécommunications, Laboratoire de Bagneux.

Copyright © 1991 by W.M. Zuberek and A. Konczykowska.
All rights reserved.

The Natural Sciences and Engineering Research Council of Canada
partially supported this research through Research Grant A8222.

FIT–2, A SIMULATION–BASED PARAMETER EXTRACTION PROGRAM [1]

A b s t r a c t

Accurate and reliable simulation of circuit behavior cannot be obtained without adequate device models. FIT–2 is an interactive program for extraction of transistor parameters for SPICE–like circuit simulators. It is based on a circuit simulator rather than an explicit set of model equations. Basic advantages of the proposed approach include: (1) explicit model equations need not be known as they are provided by the circuit simulation tool used, (2) fitting can be performed not only for single devices but for functional blocks or whole circuits as well, and (3) the same extractor can be used for a variety of devices and/or device models. Several optimization methods are built into the program to provide robust as well as efficient fitting of device characteristics. Flexibility of the approach is obtained by specification of extraction details in the data sets rather than the extraction procedure. Parameter extraction for heterojunction bipolar transistors (HBT) is used as an illustration of FIT-2 capabilities. The effects of optimization methods on extraction performance are presented. Several directions for further research are identified.

R é s u m é

La simulation précise et fiable du comportement des circuits nécessite des modèles adéquats des composants. FIT-2 est un programme interactif d'extraction de paramètres des transistors pour les simulateurs de type SPICE. Le programme est basé sur le simulateur de circuits et non sur une formulation explicite des équations du modèle. Les avantages essentiels de l'approche proposée sont: (1) les équations explicites du modèle ne sont pas nécessaires parce qu'elles sont fournies par le simulateur, (2) l'extraction peut être effectuée non seulement pour des composants mais aussi pour des blocs fonctionnels ou même des circuits entiers, (3) le même extracteur peut être utilisé pour une grande variété de composants et/ou modèles de composants. Plusieurs méthodes d'optimisation sont implantées dans le programme pour obtenir un fit des aractéristiques aussi robuste qu'efficace. La flexibilité de l'approche est obtenue par la spécification de détails d'extraction dans les fichiers de données plutôt que dans la procédure d'extraction. L'exemple du Transistor Bipolaire à Héterojonction (TBH) est donné pour illustrer les divers aspects de la procédure d'extraction. Nous présentons l'impact, sur les performances, de la méthode d'optimisation utilisée. Plusieurs directions pour des recherches futures sont indiquées.

A c k n o w l e d g e m e n t s

Collaboration with François Durbin of Commissariat à l'Energie Atomique, Bruyères–le–Châtel, France, and with Michel Bon of Centre National d'Etudes des Télécommunications, Laboratoire de Bagneux, Bagneux, France, is gratefully acknowledged.

[1] Early versions of some parts of this report have been presented at the 34th Midwest Symp. on Circuits and Systems, Monterey, CA, May 1991, and European Conf. on Circuit Theory and Design, Copenhagen, Denmark, September 1991.

1. INTRODUCTION

Accurate simulation of electronic circuits cannot be obtained without adequate specification of circuit elements and device models. Passive elements, such as resistors or capacitors, can easily be characterized by a few parameters, values of which can usually be obtained by simple measurements. In the case of semiconductor devices that are characterized by highly nonlinear models with large sets of of parameters and complicated relationships between them, a proper selection of values of parameters is a nontrivial task which, if performed inadequately, can significantly distort simulation results. Usually these model parameters cannot be determined by direct measurements because of device nonlinearities; popular parameter extraction methods use thus iterative techniques to minimize the differences between measurement data and model's behavior in the full range of operating conditions.

Several different approaches to parameter extraction have been proposed; some characteristic features of these approaches include:

- extraction methods can be general or specialized; specialized methods extract some subsets of model parameters only, for example, model resistances, or capacitances [CFG], or DC parameters only [IbGr], while general methods determine all parameters of the model;

- parameter extraction can be direct or iterative; direct methods approximate model equations by linear functions and determine the values of parameters graphically or by solving linearized equations; iterative methods fit the model responses to a set of measured characteristics by minimizing an objective function that quantitatively characterizes the fit [DoSc,CCLL,Garw]; sometimes a mixed approach is used in which some parameters are extracted using the direct methods, and remaining by an iterative procedure [DaJa,IbGr];

- iterative methods can be equation–based or simulation–based; equation–based methods use a set of model equations to obtain device responses that correspond to measurement data [DoSc,EGMT]; in simulation–based approach, a circuit simulator (or its part that handles devices and their models) is used to provide circuit responses; simulation–based approach eliminates potential inconsistencies between model equations used by the extractor and equations implemented in simulation tools as the same simulation tool can be used for both extraction and simulation,

- extraction methods can be program–driven or data–driven; in program–driven approach the structure of the data as well as the sequence of processing steps are determined by the extracting software; data–driven approach is more flexible to use but also more difficult to implement as the extraction "strategy" is specified together with the measurement data, and the extracting program mainly recognizes and executes extraction directives formulated in some sort of "high–level language".

The approach presented in this report is iterative, simulation–based and data–driven; it uses general optimization methods and an "open" circuit simulation tool rather than traditional set of model equations. The data-driven capability allows integrated parameter

extraction [BCYZ] as well as selective extraction, performed on subsets of measurement data and subsets of parameters. Different extraction strategies can thus be developed for different types of devices and/or their models in order to perform the extraction of parameters efficiently.

Several extraction programs have been reported [DoSc,CCLL,YaCh] that use gradient optimization techniques to fit electrical models to measurement data. These programs have successfully demonstrated the general principle of applicability of optimizations methods, nonetheless they suffered from significant convergence problems. The convergence properties of these methods depend upon properties of the error functions; typically, it is required that error function have no singularities, be unimodal, and approximately quadratic in the region of a minimum. These conditions are not always met by popular error functions [CCLL,BST] especially in the absence of good initial estimates of parameters. To achieve convergence, strategies must be developed which perform "partial" extractions using subsets of parameters and subsets of measurement data. Also, less efficient but robust methods are being used [CCLL,Garw] in order to avoid convergence problems of gradient techniques. The approach presented in this report uses two optimization algorithms; the initial optimization is performed by very robust direct search method of Nelder and Mead [NeMe], while a more efficient gradient–based method (from the NAG library [Phil]) is used in a neighborhood of the solution.

The proposed approach is simulation–based, i.e., it uses a general circuit simulation tool rather than a set of model equations. Basic advantages of such an approach include:

- explicit model equations need not be known as the required circuit responses are provided by a general circuit simulation tool,
- the same extractor can be used for a variety of devices and/or device models; the actual limitations are imposed by the tool used for circuit simulation rather than by the extractor,
- fitting can be performed not only for single devices (as is the case for equation–based extractors) but for any (sub)circuits as well; consequently, all packaging, mounting and fixture parasitics [EGMT] can easily be taken into account.

The report is composed of seven main sections. Section 2 describes the organization of data. Section 3 formulates parameter extraction as an (data–driven) optimization problem. Some simple examples of parameter extraction are shown in Section 5. General structure of the FIT–2 extraction program is presented in Section 5, and while Section 6 describes the command language used for interactive extraction. Section 7 contains a brief discussion of the optimization methods used by FIT–2 as well as an illustration of performance of parameter extraction depending upon the optimization method used. Section 8 discusses generalizations of parameter extraction in which optimization is performed with respect to non–electrical parameters, usually technological and/or geometric ones. Section 9 concludes the report; it also indicates a number of topics that need further research.

2. INPUT DATA

Simulation–based parameter extraction uses three different types of input information; it appears that it is convenient to store these three different types of data in three different files:

- the circuit file (file `.CKT`),
- the variables file (file `.VAR`), and
- the measurement data file (file `.DAT`).

The circuit file contains the (SPICE–style) description of a circuit that corresponds to the measurement environment (including all parasitics). Usually the same circuit file is used for fitting all available data obtained in the same measurement environment. If extraction is performed on a (sub)circuit level rather than for a single device (e.g., the transfer curve of an inverter is fitted rather than characteristics of transistors), the circuit file must describe the corresponding (sub)circuit as well as any measurement related components.

Shown below is the circuit description file for extraction of (electrical) parameters of heterojunction bipolar transistors, for which the measurement data include DC measurements in common–base and common–emitter configurations, in forward and reverse modes, and AC measurements of all four S–parameters; the independent voltage and current sources are used to control the bias conditions and to sense the corresponding currents – the initial values of these voltage and current sources are immaterial because they are (dynamically) adjusted during the extraction process to the values used during the measurements:

```
**** subcircuit TRPAR : transistor with parasitics
.SUBCKT TRPAR 1 2 3
QA   1 5 3 HBT
RBN  5 2 50
RFCB 1 2 1E8
RFCE 1 3 1E8
RFBE 2 3 1E8
CFCB 1 2 1E-16
CFCE 1 3 1E-16
CFBE 2 3 1E-16
.ENDS
.MODEL HBT NPN (IS=6.79D-24 BF=96.4 NF=1.057 VAF=229 IKF=5.53
+  ISE=1.44D-17 NE=1.67 BR=0.28 NR=1.060 VAR=104 IKR=1.00D-06
+  ISC=1.38D-11 NC=2.21 RB=27.7 IRB=6.29D-3 RBM=15.30 RE=9.29
+  RC=26.5 CJE=1.58D-13 VJE=1.80 MJE=0.50 TF=3.76D-12 XTF=0.0
+  VTF=100 ITF=0 PTF=0 CJC=1.09D-13 VJC=1.4 MJC=0.5 XCJC=0.52
+  TR=3.76D-12 CJS=0 VJS=0.75 MJS=0 XTB=0 EG=1.4 XTI=3.0 KF=0
+  AF=1.0 FC=0.5)
**** CB forward and reverse
XCB 101 102 103 TRPAR
VE  0 103  0
VB  0 102  0
VC  0 101  0
**** CE forward and reverse; parameter: IB'DC
XCE  201 202 0  TRPAR
```

```
ECE  201 203 0 203  2.0
VCE  0   203  DC=4
IB   209 202  DC=2E-4
VIB  0   209  0
**** AC parameters
.SUBCKT TRPOL 1 2 3 4
CD  2 5 1.0
CS  1 6 1.0
XAC 6 5 3 TRPAR
FIB 3 5 VB0 1.0
LD  4 6 1.0
.ENDS
**** reference sources
VC0 99 0  DC=3
RC0 99 0  1.0
IB0 0  98 DC=1.3E-4
VB0 98 97 0
RB0 97 0  1.0
**** parameters S11 (V(11)) and S21 (V(21))
I1  1  0  AC -20M
RS  1  0  50.0
E11 10 0  1 0 2.0
V11 10 11 AC 1.0
R11 11 0  1.0
X1  2  1  0 99 TRPOL
RL  2  0  50.0
R21 21 0  1.0
E21 21 0  2 0 2.0
**** parameters S12 (V(12)) and S22 (V(22))
RSB 31 0  50.0
E12 12 0  31 0 2.0
R12 12 0  1.0
X2  32 31 0 99 TRPOL
RLB 32 0  50.0
I2  32 0  AC -20M
V22 20 22 AC 1.0
E22 20 0  32 0 2.0
R22 22 0  1.0
.END
```

The second file describes all variables and their lower and upper bounds; it also contains the "nominal" values (which are used as starting points) and the actual values of variables; initially, actual values are equal to nominal ones, but during extraction they are replaced by the (partial) results of fitting.

For the circuit description shown above, the variables file contains all transistor model parameters as well as parasitics defined in the subcircuit `TRPAR`:

```
* var         min        nom        max        act
HBT'IS      1.0D-24  6.79000D-24  1.0D-23  6.79000D-24
HBT'BF      1.0D+01  9.64000D+01  1.5D+02  9.64000D+01
HBT'NF      1.0D+00  1.05700D+00  1.0D+00  1.05700D+00
HBT'VAF     1.0D+01  2.29000D+02  3.0D+02  2.29000D+02
HBT'IKF     1.0D+00  5.53000D+00  1.0D+01  5.53000D+00
HBT'ISE     1.0D-30  1.44672D-17  1.0D-16  1.44672D-17
HBT'NE      1.6D+00  1.66752D+00  1.8D+02  1.66752D+00
```

```
.......     .......  ...........  .......  ...........
TRPAR.RBN   1.0D-03  5.00000D+01  1.0D+02  5.00000D+01
TRPAR.RFCB  1.0D+05  1.00000D+08  1.0D+10  1.00000D+08
TRPAR.RFCE  1.0D+05  1.00000D+08  1.0D+10  1.00000D+08
TRPAR.RFBE  1.0D+05  1.00000D+08  1.0D+10  1.00000D+08
TRPAR.CFCB  1.0D-16  2.56330D-15  8.0D-15  2.56330D-15
TRPAR.CFCE  1.0D-16  2.35640D-15  8.0D-15  2.35640D-15
TRPAR.CFBE  1.0D-16  5.61994D-15  1.0D-14  5.61994D-15
```

During the extraction process it is possible to store any (partial) results in a file, and subsequently use such stored results as a "new" variables file; the extraction process can thus be suspended at any stage, its (partial) results saved, and then restored when needed. This capability is the main reason of storing the circuit description and the specification of variables in two independent files.

Measurement data normally include DC measurements, frequency–domain (AC) and/or time–domain (TR) measurements. Measurements of the same type (e.g., DC for a particular configuration, AC for a given bias point, etc.) form a data "group". It is assumed that each data group is composed of:

- a series of "comment" lines identified by the exclamation point as the leading character; one of the comment lines must be the group identification line (the `!IDENT` line),
- the header line that describes the contents of consecutive columns of data,
- a rectangular table of numerical values, which is composed of values of one independent variable (the first column of data) and a number of (dependent) measurements (the remaining columns); the values of independent variables are voltages or currents for DC data, frequencies for frequency–domain data, and timepoints for time–domain results,
- a blank line that terminates the group.

The identification line contains:

- an optional (unique) name of the group that is not longer than 8 characters and that is followed by the equality sign,
- the type of data which can be DC, AC or TR,
- an optional group parameterization (enclosed in square brackets), and
- an optional list of group parameters enclosed in parentheses; these parameters specify the bias point for AC data, the selected value of the base current for an I–V characteristic of a bipolar transistor, etc.

For example, the following group describes the DC collector and base current characteristics of a heterojunction bipolar transistor (in common–base configuration):

```
!IDENT:TBH_CB_1=DC
!ERRNUM=10
!' PLAQUE: 129_19 DISPO: BIP, TYPE: T1C, POSIT: 8_5_1'
!' DATE  : 11 Dec 1989 AT 22:11:17'
    VE          I(VC)         I(VB)
  0.80     3.028E-08    6.419E-08
  0.84     3.225E-08    7.246E-08
  0.88     3.589E-08    8.801E-08
  0.92     4.696E-08    1.210E-07
  0.96     9.328E-08    1.986E-07
  1.00     3.017E-07    3.894E-07
  1.08     5.321E-06    2.122E-06
  1.16     6.808E-05    1.174E-05
  1.24     3.506E-04    3.795E-05
  1.32     9.349E-04    7.726E-05
  1.40     1.856E-03    1.262E-04
  1.44     2.463E-03    1.540E-04
  1.48     3.178E-03    1.836E-04
  1.52     4.015E-03    2.152E-04
  1.56     4.976E-03    2.483E-04
  1.60     6.069E-03    2.828E-04
```

The identification line in this data group contains only the group name (`TBH_CB_1`) and the data type (`DC`). The header line indicates `VE` as the independent variable (`VE` must be an independent voltage source controlling the collector–base voltage of the transistor; see the circuit description), while `I(VC)` and `I(VB)` determine the collector and base currents, respectively (i.e., currents flowing through the independent voltage sources `VC` and `VB`).

The values of `VE` (in the `VE` column) can be distributed quite arbitrarily as SPICE–PAC performs "data–driven" circuit analyses in which a table of explicit data values determines the analysis points (for DC Transfer Curve as well as time–domain and frequency–domain analyses).

AC data groups (indicated by `AC` on the `!IDENT` line) specify the bias point (parameters `VC0` and `IB0` in the data group shown below) as the group parameters; again, `VC0` and `IB0` must be the names of independent sources which are used (in the circuit description) to control the collector voltage and the base current, respectively. The circuit description is specified in such a way that the nodes 11, 12, 21 and 22 represent the values of the corresponding S–parameters, i.e., VM(11) and VP(11) represent the magnitude and phase of S_{11}, VM(12) and VP(12) represent S_{12}, etc.:

```
!IDENT:TBH_S_31=AC(VC0=3,IB0=1.3E-4)
!' PLAQUE: 129_19, DISPO: BIP, TYPE: T1C, POSIT: 8_5_1'
!' DATE  : 11 Dec 1989 AT 22:11:17'
  FREQ  VM(11)  VP(11)  VM(12)  VP(12)  VM(21)  VP(21)  VM(22)  VP(22)
 0.1E9 .907900 -1.9024 .005604 68.7193 1.92283 176.782 .992218 -1.3091
 0.5E9 .885851 -9.2346 .024712 74.4871 1.96212 164.147 .977838 -6.3971
 1.0E9 .841972 -18.392 .045578 68.0622 1.93063 152.206 .932817 -12.136
 1.5E9 .762174 -28.541 .063145 61.8087 2.06325 140.790 .889376 -16.343
 2.0E9 .706567 -35.935 .076629 56.7939 1.87871 131.411 .846614 -19.134
 3.0E9 .582672 -48.811 .096839 49.1194 1.61879 114.934 .754855 -23.171
 5.0E9 .385694 -67.021 .121180 44.1407 1.31416 93.5795 .672503 -26.436
 7.0E9 .272881 -77.497 .143552 41.8148 1.07349 78.5958 .625978 -29.892
10.0E9 .186544 -85.502 .175547 38.1709 .837550 61.8570 .568874 -35.373
```

```
15.0E9 .125009 -95.021 .224270 31.3440 .657729 42.2776 .521664 -47.433
20.0E9 .066596 -93.885 .251645 19.4876 .548006 25.0648 .471522 -56.130
25.0E9 .069205 -80.422 .261124 14.4976 .487924 15.8900 .437293 -63.089
```

Families of similar characteristics (e.g., a family of a bipolar transistor's collector current characteristics for different values of the base current) can be described by a "parameterized" group in which the same values of the independent variable (the collector–emitter voltage in this case) correspond to several columns of data (e.g., different collector current characteristics); for a parameterized group, the name of the "column" parameter as well as the specification of output (which is the same for all columns) are given in the identification line as the "group parameterization" (within square brackets). The following DC group describing typical common–emitter characteristics of a bipolar transistor is an example of a parameterized group:

```
!IDENT:TBH_CE_2=DC[IB/I(VCE)]
!' PLAQUE: 129_19, DISPO: BIP, TYPE: T1C, POSIT: 8_5_1'
!' DATE  : 11 Dec 1989 AT 22:11:17'
  VCE      5.E-5     1.E-4    1.5E-4     2.E-4    2.5E-4
 0.00  -4.993E-5 -9.990E-5 -1.499E-4 -1.999E-4 -2.499E-4
 0.20  -4.899E-5 -9.798E-5 -1.468E-4 -1.954E-4 -2.438E-4
 0.40   1.299E-4  1.963E-4  2.351E-4  2.570E-4  2.705E-4
 0.60   4.936E-4  1.140E-3  1.556E-3  1.785E-3  1.934E-3
 0.80   5.024E-4  1.314E-3  2.323E-3  3.394E-3  4.165E-3
 1.00   5.039E-4  1.318E-3  2.339E-3  3.555E-3  4.961E-3
 1.20   5.048E-4  1.320E-3  2.343E-3  3.561E-3  4.972E-3
 1.40   5.054E-4  1.321E-3  2.346E-3  3.563E-3  4.974E-3
 1.60   5.060E-4  1.323E-3  2.348E-3  3.564E-3  4.976E-3
 1.80   5.068E-4  1.324E-3  2.348E-3  3.566E-3  4.976E-3
 2.00   5.072E-4  1.325E-3  2.350E-3  3.566E-3  4.973E-3
 2.40   5.082E-4  1.327E-3  2.352E-3  3.567E-3  4.970E-3
 2.80   5.091E-4  1.328E-3  2.354E-3  3.567E-3  4.968E-3
 3.20   5.096E-4  1.329E-3  2.355E-3  3.567E-3  4.963E-3
 3.60   5.106E-4  1.331E-3  2.356E-3  3.567E-3  4.959E-3
 4.00   5.115E-4  1.332E-3  2.358E-3  3.567E-3  4.956E-3
```

`IB` (base current) is the column parameter, and it must be the name of an independent current source (see the circuit description) used (by the extractor) to control the base current during DC circuit simulation. Similarly, the first element of the header line (`VCE`) must be the name of an independent voltage source that is used for the voltage sweep in the DC Transfer Curve analysis. The remaining values of the header line are values of `IB` that correspond to collector current (`I(VCE)`) characteristics.

Usually the column parameter is the name of an independent current or voltage source; the associated output must be an "output variable" in the sense of SPICE; both must be valid with respect to the circuit description from the circuit file.

For parameterized groups, the simulation results are obtained in a number of analyses, one analysis for each column, i.e., for each value of the column parameters that is indicated in the header line.

The file of measurement data contains a sequence of data groups. There is no limit imposed on the number or composition of data groups; in fact, a section of one data group can be repeated (with more data points) as another data group to obtain better fit in regions

which are believed to be more important or more difficult for fitting (e.g., initial parts of characteristics or highly nonlinear regions).

3. EXTRACTION THROUGH OPTIMIZATION

Extraction of transistor parameters can be formulated as an optimization problem [CCLL,DoSc,Garw,MMD,YaCh] in which a nonlinear objective function $\mathcal{F}$ is minimized with respect to the set of transistor parameters $\mathcal{P}$ subject to a set of constraints $\mathcal{C}$. The objective function $\mathcal{F}$ describes the fit of simulated device responses $\mathcal{S}$ against a set of measurement data $\mathcal{D}$. The results of optimization determine such a set of parameter values which minimizes the differences between the measurement and simulated data:

$$\begin{array}{c} \text{minimize} \quad (\mathcal{F}(\mathcal{D},\mathcal{S})) \\ \mathcal{P} \\ \text{subject to } \mathcal{C} \end{array}$$

The set of measurement data can be regarded as a sequence of K data groups, and each data group is a rectangular table of numerical results composed of N_i rows and L_i columns, $1 \le i \le K$. The objective function $\mathcal{F}(\mathcal{D},\mathcal{S})$ is given as

$$\mathcal{F}(\mathcal{D},\mathcal{S}) = \frac{1}{K} \sum_{1\le i\le K} \frac{1}{L_i} \sum_{1\le j\le L_i} W_{ij}\ f_i \left(\frac{1}{N_i} \sum_{1\le k\le N_i} e_i(D[i,j,k], S[i,j,k]) \right)$$

where:

$D[i,j,k]$ is a measured data value (in the i-th group, j-th column and k-th row),

$S[i,j,k]$ is the corresponding simulated result,

e_i is one of the "standard" error functions such as the absolute value of the difference between $D[i,j,k]$ and $R[i,j,k]$ (norm_1), the square of this difference (norm_2), the square of the relative difference (relative norm_2), the logarithm of the absolute value of the difference between $D[i,j,k]$, etc.,

f_i is a function that is "complementary" to e_i, e.g., if e_i is the square function, f_i is the square root function, etc.,

W_{ij} is a weight factor associated with the j-th column of the i-th data group; weight factors are introduced in order to "scale" averaged error values of different columns and different groups, i.e., to make them comparable in magnitude.

Error functions e_i are thus associated with data groups, and each data group can be associated with a different error function. The "standard" error functions ($e_i(x,y)$) include:

- norm_1, $e_1(x,y) = abs(x-y)$,
- relative norm_1, $e_2(x,y) = abs((x-y)/x)$,

- norm$_2$, $e_3(x,y) = (x-y)^2$,
- relative norm$_2$, $e_4(x,y) = ((x-y)/x)^2$,
- norm$_4$, $e_5(x,y) = (x-y)^4$,
- relative norm$_4$, $e_6(x,y) = ((x-y)/x)^4$,
- norm$_8$, $e_7(x,y) = (x-y)^8$,
- relative norm$_8$, $e_8(x,y) = ((x-y)/x)^8$,
- logarithmic norm, $e_9(x,y) = abs(ln(abs(y/x)))$,
- logarithmic maximum, $e_{10}(x,y) = max_{1 \le k \le N_i}(abs(ln(abs(y/x))))$.

The formulation of the objective function "averages" the error values in order to make them independent of the number of data points. Furthermore, the error functions e_i (and f_i) are usually selected in such a way that the errors for different data groups are within the same range of magnitude (e.g., common–base characteristics of bipolar transistors are usually fitted with logarithmic error functions); selection of error functions and their associations with data groups can be specified in the input data (by the `!ERRFUN=i` lines).

4. SIMPLE EXAMPLES

Simple illustrations of parameter extraction are shown for heterojunction bipolar transistors.

Fig.4.1, 4.3 and 4.5 show three groups of measurement data (indicated by the "+" markers) together with the simulated characteristics obtained for the initial values of transistor parameters (continuous lines) for DC measurements in CB configuration and forward mode

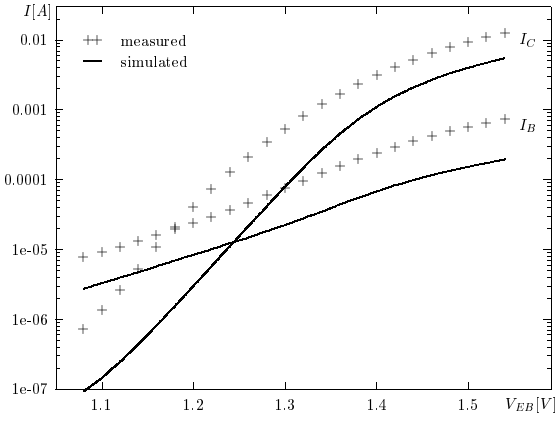

Fig.4.1. Initial DC CB data.

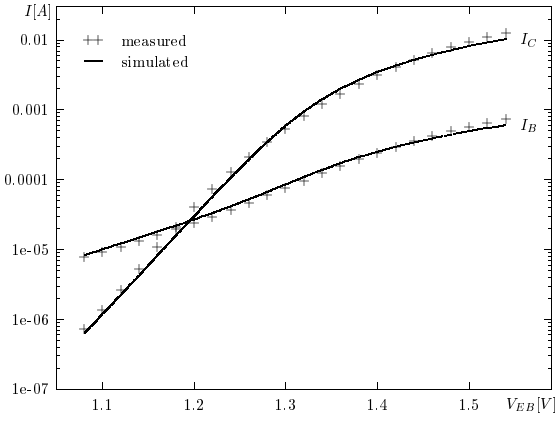

Fig.4.2. Fitted DC CB data.

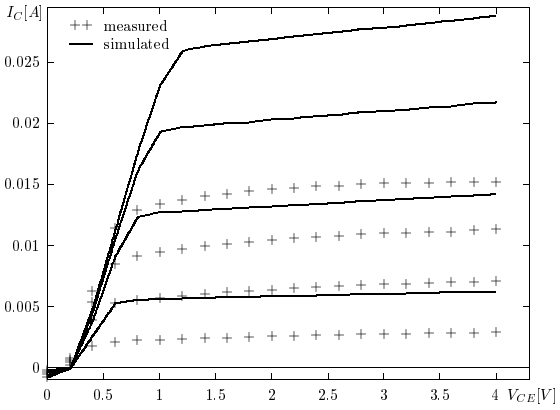

Fig.4.3. Initial DC CE data.

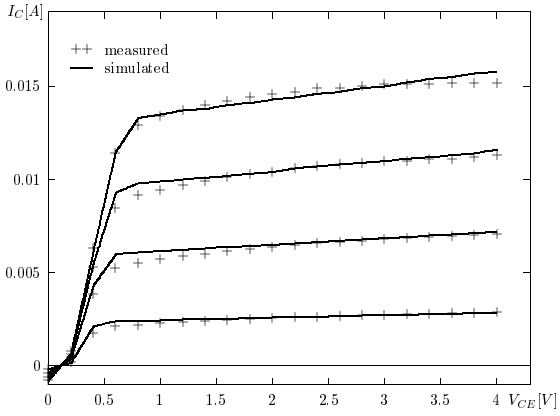

Fig.4.4. Fitted DC CE data.

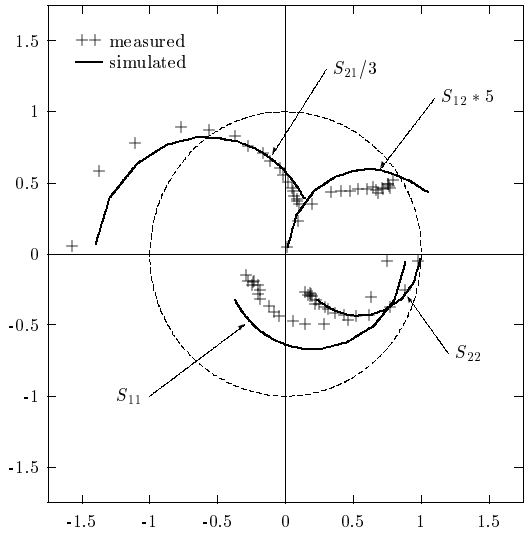

Fig.4.5. Initial AC data.

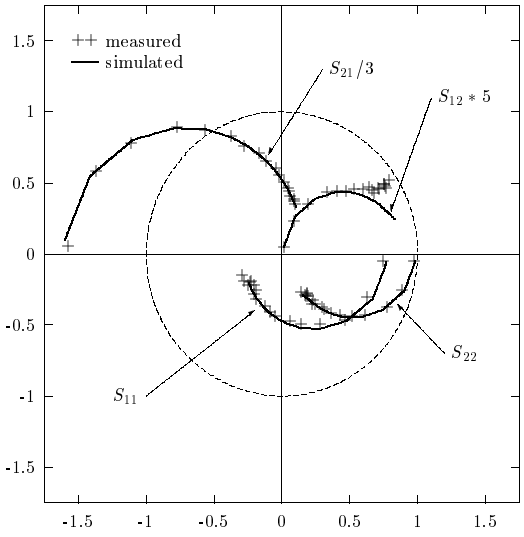

Fig.4.6. Fitted AC data.

(the collector and base currents), DC measurements in CE configuration and forward mode (4 different values of the base current), and AC measurements for all four S–parameters, respectively. It can be observed that the initial values of parameters do not provide a reasonably good fit of transistor characteristics against the measurement data.

Fig.4.2, 4.4 and 4.6 show the same three groups of measurement data with extracted values of transistor parameters. It can be observed that transistor characteristics fit very closely to the measurement data.

The error functions used in this extraction were logarithmic maximum error, relative norm_1 and relative norm_1, respectively, and all weight factors (W_{ij}) were equal to 1; the values of error functions were reduced approximately one order of magnitude, to 0.010, 0.018, and 0.108, respectively.

5. FIT–2 PROGRAM

FIT–2 is an interactive simulation–based program that extracts parameters for SPICE–like circuit simulators. It is based on the SPICE–PAC simulation package [Zub2]. General organization of the FIT–2 program in shown in Fig.5.1.

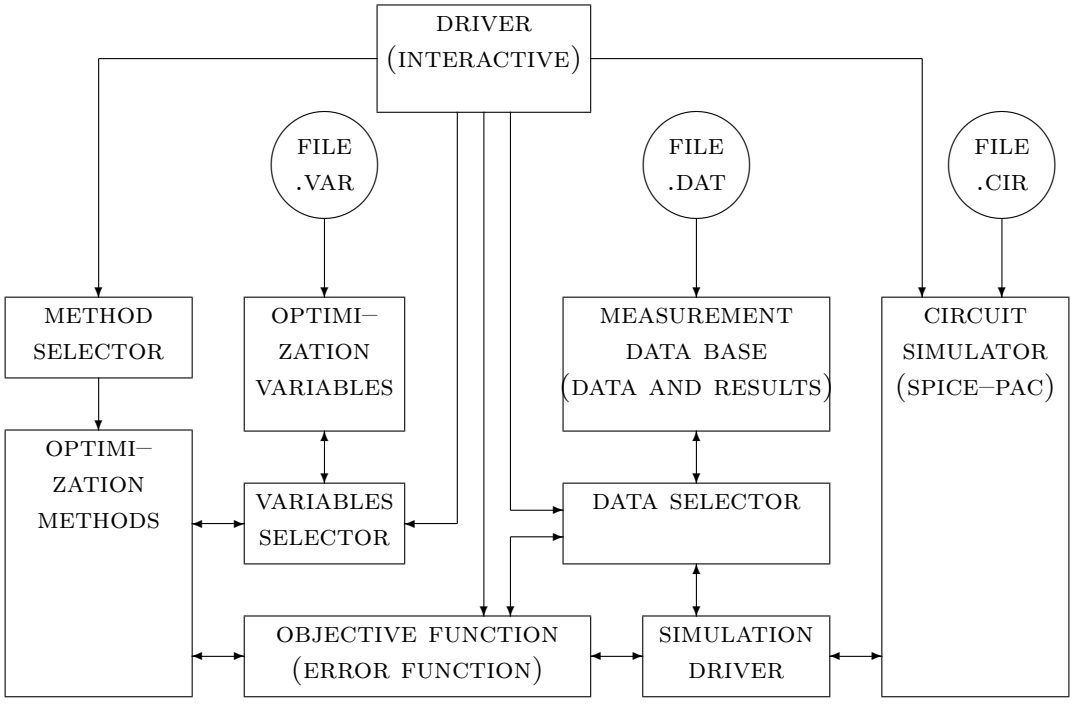

Fig.5.1. General organization of the FIT–2 program.

The program is composed of the following major parts:

- general driver which coordinates all remaining parts of the program; it also performs interaction with the user and contains an interpreter of a simple command language used for description of consecutive steps of the parameter extraction process; a number of "auxiliary" functions is supported by the driver, for example, loading and storing the values of optimization variables, printing and plotting the results, selecting data groups and subsets of variables, redefining error functions associated with data groups, modifying the weight coefficients assigned to data columns, etc.,

- variables manager which controls the set of optimization variables; all optimization variables, with their lower and upper bounds as well as starting and actual values, are defined in the variables file, and are loaded during the initialization phase of the program; the variables manager performs all run–time selections and adjustments of these variables, it updates the lower and upper bounds of variables as well as

their initial and actual values as indicated by appropriate interactive commands, it selects any subset of variables for subsequent optimization, as specified by the selection commands, and it updates the corresponding actual values after each optimization;

- data manager that maintains a collection of all measurement data and corresponding simulation results (for the initial and actual values of variables); it performs selective extraction of data for any subset of groups and any subset of columns within a group, as indicated by appropriate interactive commands, and it stores the results of circuit simulation with corresponding measurement data,

- optimizer which selects one of the optimization methods and adjusts optimization parameters accordingly, determines the starting point (using the nominal or actual values of selected variables) and invokes the optimization algorithm; for each evaluation of the objective function:

  - the values of optimization variables are used for updating corresponding parameters of the simulated circuit,
  - for each data group, group parameters (if any) are used for updating the control parameters of the simulated circuit (e.g., the bias point for AC simulations or the value of the base current for DC simulations), and then
  - the corresponding simulation is performed (DC, AC or time–domain) and its results are used for evaluation of the objective function;

- circuit simulator which performs analyses (of the circuit specified in the circuit description file) required by the evaluation of the objective function, e.i., analyses that correspond to consecutive data groups selected for optimization; circuit simulation is controlled by the simulation driver which recognizes data groups, updates circuit elements according to the measurement data (e.g., the bias point for AC analysis, the base current for DC analysis with CE configuration, etc.), initiates circuit analyses, and uses the results to update the value of the objective function:

```
for each data group do
   update group-dependent circuit parameters;
   define circuit outputs;
   set analysis parameters;
   if group is parameterized then
      for each data column do
         update column-dependent circuit parameters;
         perform circuit analysis;
         use results to update the value of the objective function
      endfor
   else
      perform analysis;
      for each data column
         use results to update the value of the objective function
   endif;
endfor;
```

It should be noted that the operations "`update circuit parameters`", "`define circuit outputs`", "`set analysis parameters`" and "`perform analysis`" correspond to the "main" operations of the SPICE–PAC package, so they directly correspond to invocations of the appropriate routines of the package [Zub1].

6. COMMAND LANGUAGE

FIT–2 is an interactive extractor which means that consecutive steps of the extraction process are usually determined by the user, who communicates with the program using a simple command language. FIT–2 can also be used in a non–interactive mode, in which case a complete sequence of commands (called a "script") is stored in a file from which the driver fetches consecutive commands and interprets them.

Most commands use some parameters. Command interpreter accepts commands with complete sets of parameters, but it also processes incomplete commands in which case it prompts the user for the missing parameters; quite often some auxiliary information is displayed that helps in selection of parameters.

The present version of the program recognizes the following commands (the commands can be given in lower case or upper case but not in combination of both):

DATA – specifies selection of data as a sequence of groups or subgroups; for example:

```
data(1,3-5,6(1-3,5))
```

selects data groups 1, 3 to 5, and a subgroup of group 6 that is composed of columns 1 to 3 and 5. Data groups can also be indicated by their (unique) names, so assuming that `name1`, `name2`, etc., are the names of consecutive data groups, an equivalent data selection command is:

```
data(name1,name3-name5,name6(1-3,5))
```

Each DATA command overrides all previous data selections, so only the most recent command is used for data selections.

VAR – specifies selection of variables as a sequence of indicators or names of variables; for example:

```
var(1,3,5-7,12)
```

selects variables 1, 3, 5 to 7, and 12 (as defined in the variables file); the variables can also be identified by their names, so – in the context of the variables file shown in section 2 – an equivalent variable selection is:

```
var(HBT'IS,HBT'NF,HBT'IKF-HBT'NE,HBT'ISC)
```

Each VAR command overrides all previous selections of variables, so only the most recent command is used for selection of variables.

FIT – selects the optimization algorithm and specifies its parameters; FIT uses three parameters, the first parameter identifies the optimization method: `nag` indicates the E04JBF routine from the NAG library and `sim` the simplex direct search method of Nelder and Mead; the second parameter indicates the maximum number of iteration steps, and the third parameter determines the starting point as `act` or `nom` for actual or nominal values of optimization variables; for example:

```
fit(nag,25,act)
```

PLOT – invokes plotting of extraction or simulation results.

NOPLOT – turns off the plotting capabilities; during optimization (execution of the FIT command), plotting is automatically invoked when the objective function is reduced by at least 10 % unless plotting is turned off.

ERR – prints the result of evaluation of the error function for the selected data groups; `err` uses one parameter which indicates the nominal or actual values of variables by `nom` or `act`, respectively.

SAVE – stores the set of variables with their bounds and actual and nominal values in a file; the command may have one parameter that is the file name to be used for storing the variables; if SAVE is used without its parameter, the original variables file is used for storing.

LOAD – restores (saved) values of variables and their bounds; the command may have one parameter which is the file name to be used for loading the variables; if LOAD is used without its parameter, the original variables file is used for loading.

NEW – creates a new data group from an indicated subset of data rows of an existing group; it uses one parameter which specifies the name of the new group, the equality sign, the name of the existing group and the selection list enclosed in parentheses; for example:

```
new(newgroup=oldgroup(1,3-7,9(3)20,25))
```

creates a new group named `newgroup` that contains rows 1, 3 to 7, every third row from 9 to 20 (i.e., 9, 12, 15, 18) and 25 of the data group named `oldgroup`; all attributes of the created `newgroup` group, such as group and column parameters (if any), the number of data columns and their associated names, etc., are the same as for `oldgroup`.

DEL – deletes a group created by NEW; the group is the only argument of this command; only the most recently created group can be deleted (making the previous group the most recent one).

TEMP – defines or redefines the (ambient) temperature for data groups; the temperature can also be specified in the data groups by `!TEMP=value` lines; TEMP command specifies the temperatures (in degrees Celsius) by a sequence of pairs "group=value" where "group" identifies the data group, and "value" is the corresponding temperature; for example:

```
temp(1=75.0,3=50.0)
```

defines the ambient temperature of the data group "1" as 75 degrees, and that of group "3" as 50 degrees Celsius. The temperatures assigned to groups remain valid until they are redefined. The default values of the temperature are equal to the reference temperature in the circuit description.

MODERR – redefines error functions assigned to data groups; error functions can be assigned by defaults, or they can be specified in the data file by `!ERRFUN=number` lines; the MODERR command redefines error functions by a sequence of pairs "group=number" where "group" identifies the data group, and "number" is one of the standard error functions; for example:

```
moderr(1=9,3=2)
```

assigns the logarithmic norm function (9) to the data group "1" and the relative norm_1 function (2) to the data group "3". The assigned error functions remain valid until they are redefined.

MODACT – redefines actual values of variables indicated in a sequence of pairs "variable=value" where "variable" identifies one of optimization variables, and "value" is its new actual value; for example:

```
modact(1=5E-24,4=100.0)
```

or equivalently using the names of variables:

```
modact(HBT'IS=5E-24,HBT'VAF=25.0)
```

assigns the value 5E-5 to the first variable as its actual value, and 25.0 as the actual value of the third variable. The actual value of a variable must be defined within its lower and upper bounds.

MODNOM – as MODACT but for the nominal values of variables.

MODMIN – as MODACT but for lower bounds of variables; the lower bound of a variable cannot be defined greater than its actual value.

MODMAX – as MODACT but for upper bounds of variables; the upper bound of a variable cannot be defined smaller than its actual value.

COEF – defines or redefines weight coefficients W_{ij} associated with columns of data groups; default values of these coefficients are equal to 1; the COEF command specifies a sequence of pairs "column=value" or "group=value" where "column" identifies a column j of a data group i as `i.j`, and "value" is the value of the corresponding coefficient W_{ij}; the specification "group=value" assigns the "value" to all columns of the indicated data group; for example:

```
coef(1.2=0.5,2=0.75)
```

assigns the coefficient 0.5 to the second column of the first data group, and the coefficient 0.75 to all columns of the second data group. The assigned coefficients remain valid until they are redefined.

IF ... THEN ... ELSE ... END – a conditional structure for description of simple extraction strategies (for noninteractive applications); its condition is a logical expressions that is composed of simple relations and logical operators AND represented by "&" and OR represented by "|".

For example, the following script first selects data groups 1 to 4 and a subset of optimization variables 1 to 5, 10 and 12 from the set of variables specified in the variables file; then it performs at most 100 optimization steps using the simplex method and the actual values of circuit variables as the starting point (command `fit(sim,100,act)`); if the error value remains greater than 0.1 (`ERROR` is a global variable that stores the value of the error function), then a smaller subset of variables and smaller subset of data are selected and optimization is repeated using the simplex method; if the new error value becomes less than 0.5, previous subsets of variables and data are selected and optimization continues, otherwise some other actions are specified; the final step performs yet another optimization, but in this case the NAG routine is used under the assumption that a point close to a minimum has been reached (which may not be the case, actually):

```
data(1-4)
var(1-5,10,12)
fit(sim,100,act)
if ERROR>0.1 then
   data(1,2)
   var(1-3,5)
   fit(sim,50,act)
   if ERROR<0.5 then
      var(1-5,10,12)
      data(1-4)
      fit(sim,50,act)
   else
      ...
   endif
endif
fit(nag,25,act)
```

`ERROR` is a global variable that saves the value of the error function (evaluated by the FIT or ERR commands). `RETURN` is another such variable and it indicates the termination condition

of the FIT command (and in fact, the termination condition of the optimization routine used within FIT).

Command scripts can include comment lines indicated by "#" or "*" as the leading character. Command scripts can also use multi–level "include" files indicated by

```
<filename
```

lines, where `filename` is the name of the "include" file.

7. OPTIMIZATION METHODS

Presently FIT–2 contains two different optimization techniques:

- quite robust but rather slow simplex direct search method of Nelder and Mead [NeMe] (as in [CCLL]); the method requires only function evaluations, not derivatives; the method (also known as "downhill simplex") maintains a "simplex" composed of $N+1$ vertices (N is the number of dimensions), and in each step it replaces the vertex corresponding to the largest value of the objective function with a reflection, a reflection and extension, or a contraction, depending upon the value of the objective function in the new vertex; if no "better" vertex can be found in such a way, the whole simplex is reduced, and the procedure continues; the method is thus not very efficient in terms of the number of function evaluations, but it is quite insensitive to large changes and even discontinuities of the objective function;
- the E04JBF routine from the NAG library [NAG]; E04JBF implements a comprehensive quasi–Newton algorithm (that updates an approximation of the Hessian matrix using the BFGS formula) for finding a minimum of a function of several variables subject to fixed upper and lower bounds on the variables; although the method is intended for functions which have continuos first and second derivatives, it usually works even if the derivatives have occasional discontinuities; the derivatives are not required as they are approximated by finite differences; the method is much more efficient with respect to the number of required function evaluations than the downhill simplex, but it also is more sensitive to large changes of function values.

Normally, the initial optimization is performed by the simplex method as it is less sensitive to changes of the objective function; the quasi–Newton method is most efficient in the neighborhood of the solution, so its typical application is in the second stage of optimization, when a neighborhood of a minimum has been reached by the simplex method.

The influence of the optimization method can be illustrated by the following comparison of the computational effort (i.e., the CPU time) required to find the best fit and the values of the error function at the solution:

optimization method	CPU time (in min)	error value
simplex	178	7.9E-3
E04JBF	82	3.0E-3
simplex + E04JBF	70	2.6E-3

The CPU time was measured on a microVAX 3600 machine running VMS.

8. PARAMETER CONVERSIONS

Although electrical parameters of semiconductor devices are very useful from circuit design perspective, they are quite inconvenient from manufacturing viewpoint as they cannot provide the required feedback for process analysis or device design optimization. A set of technological and geometric parameters is much more relevant to manufacturing processes than a set of electrical parameters.

Usually, the relationship between such two sets of parameters is provided by device modelling, and once these relationships are established and verified, the new set of parameters can be used for device characterization, optimization as well as design. Technological and geometric parameters are also much more convenient to impose technology constraints, and to analyze parameters deviations and correlations. Quite often a "mixed" set of parameters is used that includes electrical as well as technological and geometric parameters.

It should be noted that technological parameters and dependencies between technological and electrical parameters are closely related to manufacturing technology; as this technology evolves, both parameters and dependencies change. Therefore, a capability of (flexible) parameter conversions is an important aspect of extraction tools.

In order to support different sets of parameters, an interface has been incorporated into the FIT–2 extractor that accepts a user–defined conversion of parameters used as optimization variables (e.g., technological and geometric parameters) into electrical parameters (used in circuit simulation and called circuit variables). This interface is composed of two routines, VARDEF and VARMAP; VARDEF performs a mapping of the set of names of optimization variables into a corresponding set of names of circuit variables (names of circuit variables must be correct with respect to the circuit description file as required by SPICE–PAC [Zub2]), and VARMAP performs a mapping of values of optimization variables into the corresponding values of circuit variables assuming that the ordering of variables is the same as for VARDEF.

For "nonstandard" applications, these two routines must be user–defined and linked with the FIT–2 program. The routines must conform to the following FORTRAN headers:

```
SUBROUTINE VARDEF (NAMOPT,NOV,NAMCKT,LCV,NCV)
CHARACTER*16 NAMOPT(*),NAMCKT(*)
```

where:

`NAMOPT` is a vector of CHARACTER*16 names of optimization variables (as entered form the variables file),

`NOV` is the number of names in the vector `NAMOPT`,

`NAMCKT` is a vector that returns the CHARACTER*16 names of circuit variables,

`LCV` is the limit of circuit variables (i.e., the length of `NAMCKT`),

`NCV` is the number of defined circuit variables.

```
      SUBROUTINE VARMAP (VAROPT,NOV,VARCKT,LCV,NCV)
      DOUBLE PRECISION VAROPT(*),VARCKT(*)
```

where:

`VAROPT` is a vector of DOUBLE PRECISION values of optimization variables,

`NOV` is the number of optimization variables,

`VARCKT` is a vector that returns the DOUBLE PRECISION values of circuit variables,

`LCV` is the limit of circuit variables (i.e., the length of `VARCKT`),

`NCV` is the number of circuit variables.

The "standard" interfacing routines correspond to the case when the optimization variables are also circuit variables, i.e., they perform identity mappings:

```
      SUBROUTINE VARDEF (NAMOPT,NOV,NAMCKT,LCV,NCV)
      CHARACTER*16 NAMOPT(*),NAMCKT(*)
      NCV=0
      DO 10 I=1,NOV
        IF (NCV.LT.LCV) THEN
          NCV=NCV+1
          NAMCKT(I)=NAMOPT(I)
        ELSE
          WRITE(*,900)
  900     FORMAT(' ... vardef : too many circuit variables ...')
          RETURN
        ENDIF
   10 CONTINUE
      RETURN
      END

      SUBROUTINE VARMAP (VAROPT,NOV,VARCKT,LCV,NCV)
      DOUBLE PRECISION VAROPT(*),VARCKT(*)
      NCV=0
      DO 10 I=1,NOV
        IF (NCV.LT.LCV) THEN
          NCV=NCV+1
          VARCKT(I)=VAROPT(I)
        ELSE
          WRITE(*,900)
  900     FORMAT(' ... varmap : too many circuit variables ...')
          RETURN
        ENDIF
   10 CONTINUE
      RETURN
      END
```

The following simple example illustrates a potential application of parameter conversion. It assumes that many of the (electrical) Gummel–Poon parameters of a GaAs/GaAlAs transistor depend on a few other parameters:

ρ_b	base contact resistance
ρ_e	emitter contact resistance
ρ_c	collector contact resistance
L_t	transistor length

The first three parameters represent ohmic contacts (in Ω/cm^2) of the base, emitter and collector, respectively. Approximate values of these parameters can be obtained through measurements of special test devices, however, such test devices are usually much larger than other devices, so the measured values do not correspond to the non–test devices accurately. The last parameter, L_t, is very difficult to measure.

The relationships between these new parameters and the electrical ones are as follows:

RE	emitter resistance	$f_1(\rho_e, L_t)$
RC	collector resistance	$f_2(\rho_c, L_t)$
RB	zero bias base resistance	$f_3(\rho_b, L_t)$
RBM	minimum base resistance at high currents	$f_4(\rho_b, L_t)$
CJE	base–emitter zero bias depletion capacitance	$f_5(L_t)$
CJC	base–collector zero bias depletion capacitance	$f_6(L_t)$
IS	transport saturation current	$f_7(L_t)$
ISE	base–emitter leakage saturation current	$f_8(L_t)$
ISC	base–collector leakage saturation current	$f_9(L_t)$

Assuming that the set of optimization variables specifies the four variables indicated above, and that the set of electrical parameters (or circuit variables) includes only the nine listed parameters, the conversion routines VARDEF and VARMAP could be as follows:

```
      SUBROUTINE VARDEF (NAMOPT,NOV,NAMCKT,LCV,NCV)
      CHARACTER*16 NAMOPT(*),NAMCKT(*),NAMES1(4),NAMES2(9)
      DATA NAMES1 / 'RHOB','RHOE','RHOC','LT' /
      DATA NAMES2 / 'TBH:RE','TBH:RC','TBH:RB','TBH:RBM','TBH:CJE',
     +              'TBH:CJC','TBH:IS','TBH:ISE','TBH:ISC' /
      NCV=0
      DO 10 I=1,9
        IF (I.LE.4) THEN
          IF (NAMOPT(I).NE.NAMES1(I)) WRITE(*,910) NAMES1(I)
  910     FORMAT(' ... vardef : incorrect optimization variable : ',A)
        ENDIF
        IF (NCV.LT.LCV) THEN
          NCV=NCV+1
          NAMCKT(I)=NAMES2(I)
        ELSE
          WRITE(*,920) NAMES2(I)
  920     FORMAT(' ... vardef : too many circuit variables ... ',A)
        ENDIF
   10 CONTINUE
      RETURN
      END

      SUBROUTINE VARMAP (VAROPT,NOV,VARCKT,LCV,NCV)
      DOUBLE PRECISION VAROPT(*),VARCKT(*)
```

```
      IF (NOV.LT.4) THEN
        WRITE(*,910) NOV
  910   FORMAT(' ... varmap : incorrect optimization variables : ',I3)
      ELSE IF (LCV.LT.9) THEN
        WRITE(*,920) LCV
  920   FORMAT(' ... varmap : there are 9 circuit variables : ',I3)
      ELSE
        VARCKT(1)=f1(VAROPT(1),VAROPT(4))
        VARCKT(2)=f2(VAROPT(2),VAROPT(4))
        VARCKT(3)=f3(VAROPT(3),VAROPT(4))
        VARCKT(4)=f4(VAROPT(3),VAROPT(4))
        VARCKT(5)=f5(VAROPT(4))
        VARCKT(6)=f6(VAROPT(4))
        VARCKT(7)=f7(VAROPT(4))
        VARCKT(8)=f8(VAROPT(4))
        VARCKT(9)=f9(VAROPT(4))
        NCV=9
      ENDIF
      RETURN
      END
```

The example shows that parameter conversion can eventually reduce the set of optimization variables. Furthermore, because of nonlinear dependencies between these two sets of parameters, simple constraints of technological parameters correspond to nonlinear constraints of electrical parameters; such constraints may be difficult to take into account because more powerful optimization methods are needed to deal with general (nonlinear) constraints. However, the most important aspect of parameter conversion seems to be in a rather straightforward implementation of parameter dependencies that are introduced by functions f_i in the example, and which are ignored when optimization is performed with respect to electrical parameters only.

9. CONCLUDING REMARKS

A simulation–based parameter extraction program has been developed as an example of integrated computer–aided design tools; the program is based on an existing circuit simulation package instead of traditional model equations, and it uses general optimization algorithms for minimization of dissimilarities between measurement data and the model behavior. Conversion of parameters is provided for extraction with respect to technological and geometric parameters as well as for dealing with parameter dependencies which cannot be represented (at least not easily) by optimization constraints.

It appears that different types of (measurement and simulated) data are associated with subsets of transistor parameters. The extraction process can be simplified considerably if optimizations are performed on small but relevant subsets of optimization variables, designated by different types of available measurement data. Mechanisms needed for such selective optimizations (or "partial extraction") are built into the FIT–2 program.

Although a number of relationships between measurement data and extracted parameters are quite useful in parameter extraction [DaJa,IbGr], practical experiments seem to indicate that a general extraction strategy may be rather difficult to find. Therefore a

convenient formalism for higher–level specification of the extraction processes can be very helpful in automation of this process. Elements of such higher–level specification have been implemented in FIT–2; some further extensions of this formalisms should be included in future versions of the program.

Both optimization methods available in FIT–2 provide local optimization only, so in a case of numerous local minima, the starting point should be disturbed externally to cover as large part of the feasible space as seems reasonable. However, that local optimization algorithms are seldom satisfactory even when restarted from several randomly chosen initial points. Measurement error coupled with the large number of variables of a physically based circuit leads to an error function with many nonphysical local minima in addition to the global minimum [BST]. More general (and efficient) global optimization methods are needed but they are rather difficult to find. Simulated annealing [Rut] has recently been proposed as an alternative to gradient–descent methods. In simulated annealing, the actual values of variables are disturbed and the new error is calculated; if this error is smaller than the previous one, the new values replace the actual ones, as in descent methods. But, sometimes, in distinction to descent algorithms, the vector of variables with larger error may be accepted in accordance with a precise probabilistic criterion which becomes less tolerant of "bad" moves at late stages of the algorithm. The success of this algorithm depends on generating moves that are neither always accepted nor always rejected [Rut]. The method is very promising but further research is needed to make it generally applicable.

References

[BCYZ] J.W. Bandler, S.H. Chen, S. Ye, Q-J. Zhang, "Integrated model parameter extraction using large–scale optimization concepts"; IEEE Trans. on Microwave Theory and Techniques, vol.36, no.12, pp.1629-1638, 1988.

[BST] G.L. Bilbro, M.B. Steer, R.J. Trew, C-R Chang, S.G. Skaggs, "Extraction of the parameters of equivalent circuits of microwave transistors using tree annealing"; IEEE Trans. on Microwave Theory and Techniques, vol.38, no.11, pp.1711-1718, 1990.

[CCLL] P. Conway, C. Cahill, W.A. Lane, S.U. Lidholm, "Extraction of MOSFET parameters using the simplex direct search optimization method"; IEEE Trans. on Computer-Aided Design, vol.4, no.4, pp.694-698, 1985.

[CFG] W.M. Coughran Jr., W. Fichtner, E. Grosse, "Extracting transistor charges from device simulations by gradient fitting"; IEEE Trans. on Computer-Aided Design, vol.8, no.4, pp.380-394, 1989.

[DaJa] A. Davies, A.K. Jastrzebski, "Parameter extraction technique for nonlinear MESFET models"; Proc IEEE Conf. on Microwave Theory and Techniques, pp.747-750, 1990.

[DoSc] K. Doganis, D.L. Scharfetter, "General optimization and extraction of IC device model parameters"; IEEE Trans. on Electron Devices, vol.30, no.9, pp.1219-1228, 1983.

[EGMT] M. Eron, J. Gerber, L. Mah, W. Tompkins, "MESFET model extraction and verification techniques for nonlinear CAD applications"; Proc. 3–rd Asia–Pacific Microwave Conf., Tokyo, Japan, pp.321-324, 1990.

[Garw] K. Garwacki, "Extraction of BJT model parameters using optimization method"; IEEE Trans. on Computer-Aided Design, vol.7, no.8, pp.850-854, 1988.

[GrOk] P.C. Grossman, A. Oki, "A large signal DC model for GaAs/$Ga_{1-x}Al_xAs$ heterojunction bipolar transistors"; IEEE Proc. Bipolar Circuits Technology Mtg, Minneapolis, pp.258-261, 1989.

[IbGr] A. Ibarra, J. Gracia, "Strategy for DC parameter extraction in bipolar transistors"; IEE Proceedings part G, vol.137, no.1, pp.5-11, 1990.

[MMD] W. Maes, K.M. DeMeyer, L.H. Dupas, "SIMPAR: a versatile technology independent parameter extraction program using a new optimized fit strategy"; IEEE Trans. on Computer-Aided Design, vol.5, no.2, pp.320-325, 1986.

[NAG] NAG FORTRAN Library Manual Mark 9, vol.3; Numerical Algorithms Group, 1982.

[NeMe] J.A. Nelder, R. Mead, "A simplex method for function minimization"; Computer Journal, vol.7, pp.308-313, 1965.

[Phil] J. Phillips, "The NAG library: a beginner's guide"; Oxford University Press 1987.

[Rut] R.A. Rutenbar, "Simulated annealing algorithms: an overview"; IEEE Circuits and Devices Magazine, vol.5, no.1, pp.19-26, 1989.

[Vlad] A. Vladimirescu, K. Zhang, A.R. Newton, D.O. Pederson, A.L. Sangiovanni-Vincentelli, "SPICE Version 2G - User's Guide"; Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, 1981.

[YaCh] P. Yang, P.K. Chatterjee, "An optimal parameter extraction program for MOSFET models"; IEEE Trans. on Electron Devices, vol.30, no.9, pp.1214-1219, 1983.

[Zub1] W.M. Zuberek, "SPICE–PAC version 2G6c — user's guide"; Technical Report #8902, Department of Computer Science, Memorial University of Newfoundland, St. John's, Canada A1C-5S7, 1989.

[Zub2] W.M. Zuberek, "SPICE–PAC version 2G6c — an overview"; Technical Report #8903, Department of Computer Science, Memorial University of Newfoundland, St. John's, Canada A1C-5S7, 1989.