Technical Report #9402

FIT–S, A SIMULATION–BASED DATA–DRIVEN PARAMETER EXTRACTION PROGRAM

by

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

[†] Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada A1C-5S7

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

May 1994

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

The Natural Sciences and Engineering Research Council of Canada
partially supported this research through Research Grant A8222
and International Colaborative Research Grant ICR0121191.

FIT–S, A SIMULATION–BASED DATA–DRIVEN PARAMETER EXTRACTION PROGRAM [1]

A b s t r a c t

FIT–S is an interactive program for extraction of device 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. The extractor supports numerical as well as symbolic simulation so repeated analyses of linearized circuit (for frequency domain analyses) can be performed very efficiently using the symbolic functions generated from the Coates flowgraph representation of the circuit. 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–S capabilities.

R é s u m é

FIT–S est un programme interactif d'extraction de paramètres des composants 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. L'extracteur effectue aussi bien l'analyse numérique que symbolique, ce qui fait que des analyses répétées du circuit linéarisé (dans le domaine fréquentiel) peuvent être réalisées très efficacement en utilisant les fonctions obtenues à partir de la répresentation du circuit par le graphe de fluence de Coates. Plusieurs méthodes d'optimisation sont implantées dans le programme pour obtenir un fit des caractéristiques 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.

Acknowledgements

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] This report replaces Technical Report #9111 which described an earlier version of the FIT program.

1. INTRODUCTION

Reliable simulation of electronic circuits cannot be obtained without accurate 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 the 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 that use gradient optimization techniques to fit electrical models to measurement data [DoSc,CCLL,YaCh]. 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 five main sections. Section 2 briefly describes general organization of a data–driven program. Section 3 formulates parameter extraction as an (data–driven) optimization problem and also contains a brief discussion of the optimization methods used in FIT–S; it also presents an illustration of the performance of parameter extraction depending upon the optimization method used. Section 4 presents an enhancement of the original extractor in which self–heating effects of devices are taken into account by a combined DC transfer curve and thermal analyses. A brief description of integrated numerical– symbolic simulation is given in Section 5, and illustrated by a comparison of performance results. Section 6 discusses generalizations of parameter extraction in which optimization is performed with respect to non–electrical parameters, usually technological and/or geometric ones. Section 7 concludes the report; it also indicates a number of topics that need further research.

A description of input data for the FIT–S program is given in Appendix 1, while Appendix 2 describes the command language in greater detail.

2. DATA–DRIVEN ORGANIZATION

The organization of data for a simulation–based extractor is partially implied by its circuit simulation tool. FIT–S uses three different types of input data:

- the circuit description file,
- the definition of variables file,
- the measurement data file.

The circuit description file contains the (SPICE–style) description of a circuit that corresponds to the measurement environment (including all parasitics). The same circuit file may contain circuits corresponding to several measurement environments.

The second file defines all variables (e.g., all extracted transistor model parameters as well as parasitics) with their lower and upper bounds (the bounds are used as optimization constraints); it also contains the nominal values of parameters (which are used as the starting point) and the actual values of all variables. Initially, the actual values are equal to nominal values, but during the extraction process, the actual values are being replaced by those results of optimizations which improve the fit.

Measurement data normally include DC measurements, frequency–domain (AC) and/or time–domain (TR) measurements (used for large–signal analysis of the periodic steady–state), but harmonic and noise measurements can also be handled by the extractor. Measurements of the same type (e.g., AC for a given bias point, steady–state time–domain for a given frequency, etc.) form a data "group". It is assumed that each group is composed of a header that identifies the group and describes its contents, and a rectangular table of numerical (measurement) data. The table of numerical data is composed of values of one independent variable and a number of (dependent) measurements; the values of independent variables are voltages or currents for DC data, frequencies for frequency–domain data, and timepoints for time–domain results. A more detailed description of data groups is given in Appendix 1.

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 provide a 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).

These three different types of data (circuit description, specification of variables and measurement data) are handled by different modules of FIT–S. Fig.1 shows a general organization of the FIT–S extractor. Its main components are:

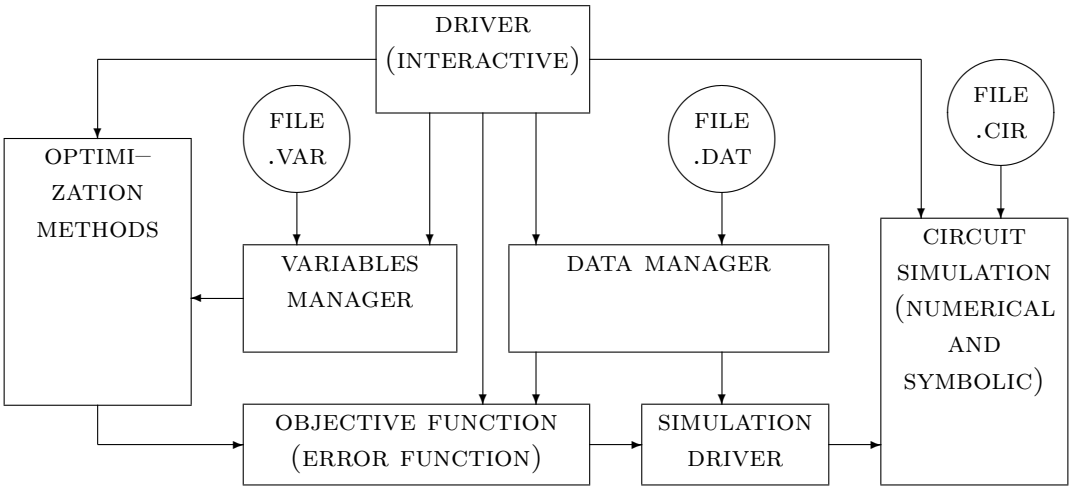

Fig.1. General organization of the FIT–S program.

- General driver which coordinates all remaining parts of the program and performs interaction with the user using an interpreter of a simple command language that describes consecutive steps of the extraction process.

- Variables manager which controls the set of optimization variables. All optimization variables, with their lower and upper bounds as well as nominal and actual values, are entered from the variables file during the initial phase of the program. The variables manager performs all run–time modifications of these variables, it selects subsets of variables for subsequent optimizations, and updates the corresponding 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 extractions of data for any subset of groups and any subset of columns within a group, and it stores the results of circuit simulation with corresponding measurement data.

- Optimizer which selects the optimization method and adjusts optimization parameters accordingly; it also selects the starting point for the optimization.

- Circuit simulator which performs the analyses required by the evaluation of the objective function, e.i., analyses that correspond to consecutive data groups selected for fitting. The simulator is composed of a SPICE–like (numerical) simulation package SPICE–PAC [Zub2] integrated with a symbolic package SYBILIN [KMGB].

A general scheme of data–driven extraction can be illustrated by the following outline of a typical extraction step:

```
select(variables);
select(data_groups);
continue_optimization := true;
```

```
while continue_optimization do
   update_the_values_of_variables;
   error := 0;
   for each selected_data_group do
      adjust_circuit_parameters;
      find_simulation_results(results);
      error := error + differences(data,results)
   endfor;
   if error < accuracy then continue_optimization := false
   else get_new_values_of_variables(error) endif
endwhile;
display_the_results;
```

where the `select` operations and the optimization are performed by interactive commands, the results of optimizations are presented to the user in some convenient form (e.g., a graphical plot), and the optimization loop can also be terminated if no progress is detected, or one of the optimization limits is reached.

The organization of data for FIT–S is described in greater detail in Appendix 1, and the command language in Appendix 2.

3. EXTRACTION THROUGH OPTIMIZATION

Extraction of device parameters can be formulated as an optimization problem [DoSc, Garw, MMD] in which a nonlinear objective function $\mathcal{E}$ is minimized with respect to a set of device parameters $\mathcal{P}$ subject to a set of constraints $\mathcal{C}$. The objective function $\mathcal{E}$ 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:

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

For the set of measurement data composed of K data groups, with each data group being a rectangular table of N_i rows and L_i columns of numerical values, $1 \le i \le K$, the (nonlinear) objective function $\mathcal{E}(\mathcal{D},\mathcal{S})$ used in FIT–S is

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

where:

$D[i,j,k]$ is a measured 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 "built–in" 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.,

$W_{i,j}$ 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 have a different error function associated with it. 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.

The "built–in" 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, $e_{10}(x,y) = max_{1 \le k \le N_i}(abs(ln(abs(y/x))))$.

Simple examples of parameter extraction for heterojunction bipolar transistors are shown in Fig.2 to Fig.7. Fig.2, Fig.4 and Fig.6 show DC common–base (Gummel plot), DC common–emitter and AC measurement data (all four S–parameters for the frequencies from 0.2 to 17 GHz) together with the simulated characteristics obtained for the initial values of transistor parameters (the initial values of parameters are obtained by simple prediction formulas). It can be observed that these initial values of parameters do not provide a reasonably good fit of transistor characteristics against the measurement data.

Fig.3, Fig.5 and Fig.7 show the same groups of measurement data but with extracted values of transistor parameters (using weights equal to 1 and the log norm_1, relative norm_2 and norm_2, respectively). It can be observed that simulated characteristics fit very closely to the measurement data.

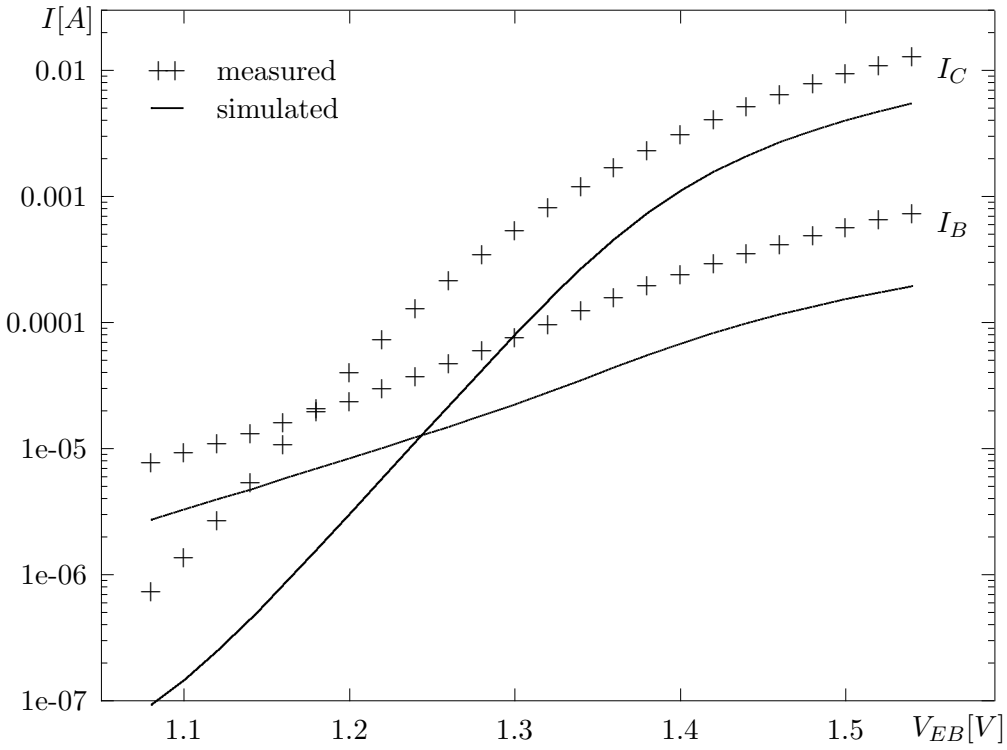

Fig.2. HBT common–base DC characteristics; initial values of parameters.

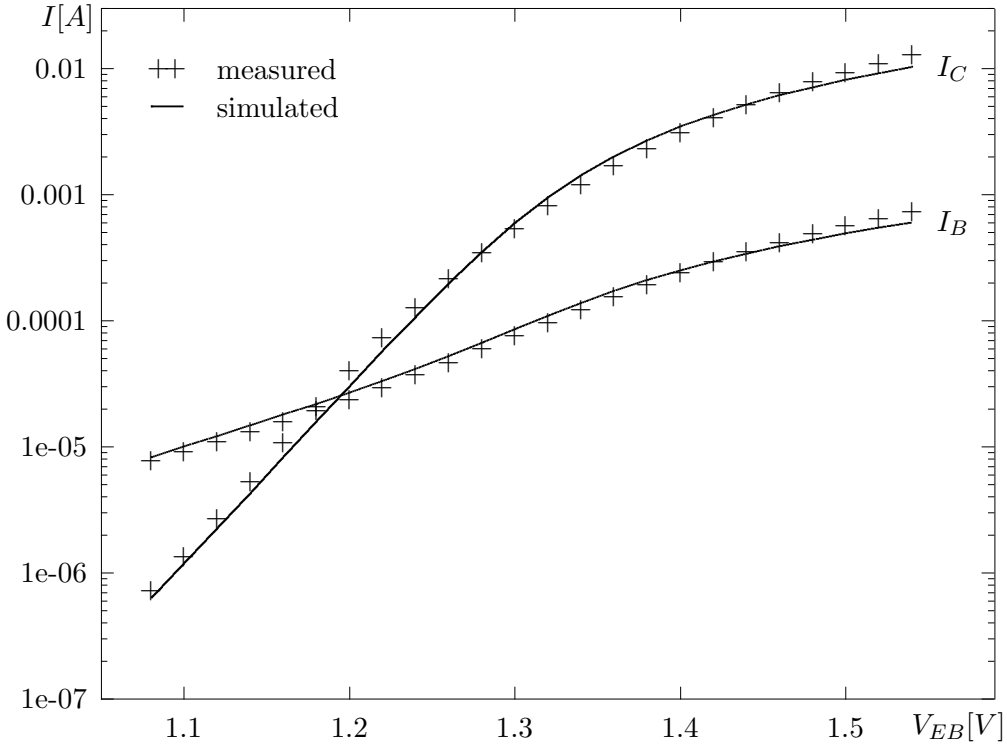

Fig.3. HBT common–base DC characteristics; extracted values of parameters.

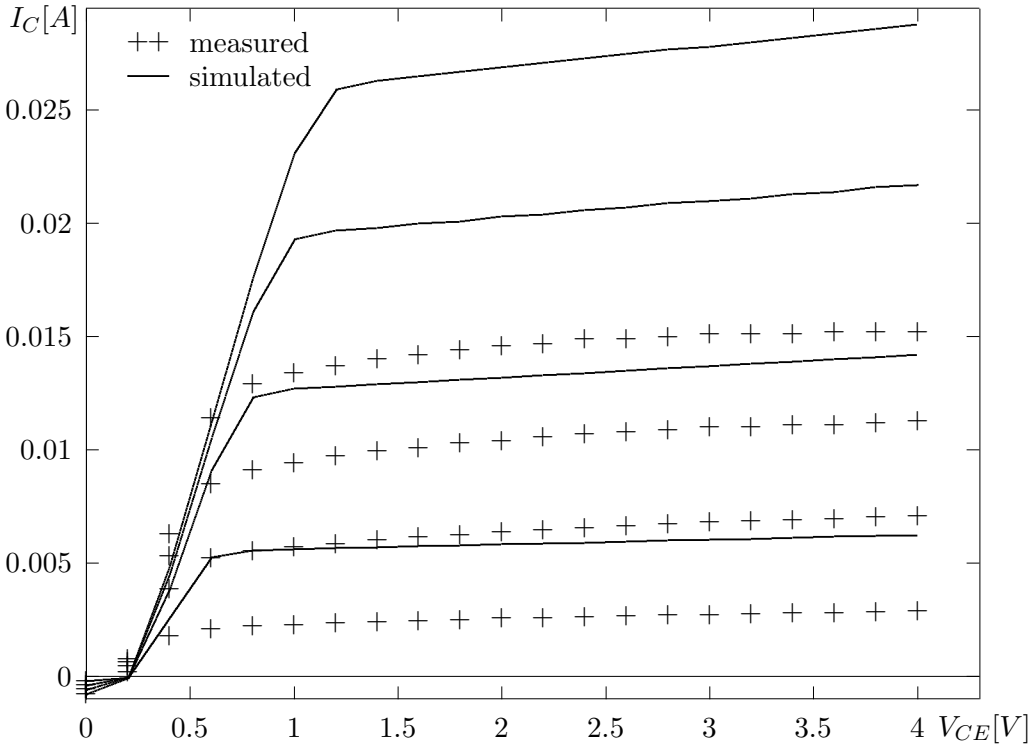

Fig.4. HBT common–emitter DC characteristics; initial values of parameters.

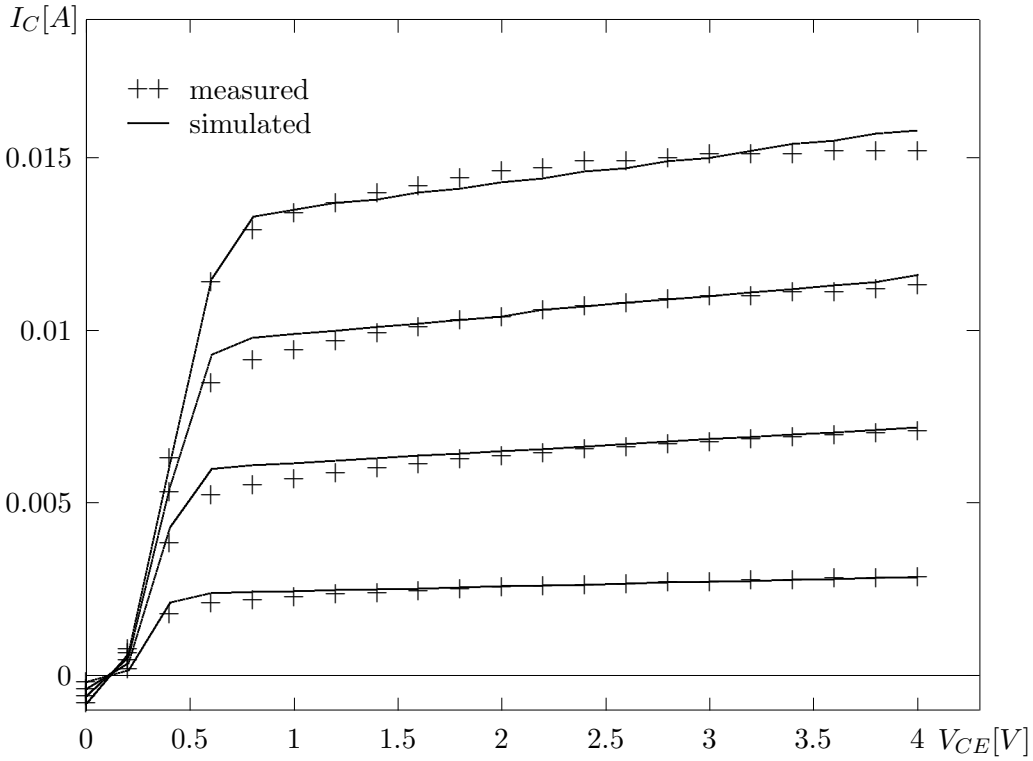

Fig.5. HBT common–emitter DC characteristics; extracted values of parameters.

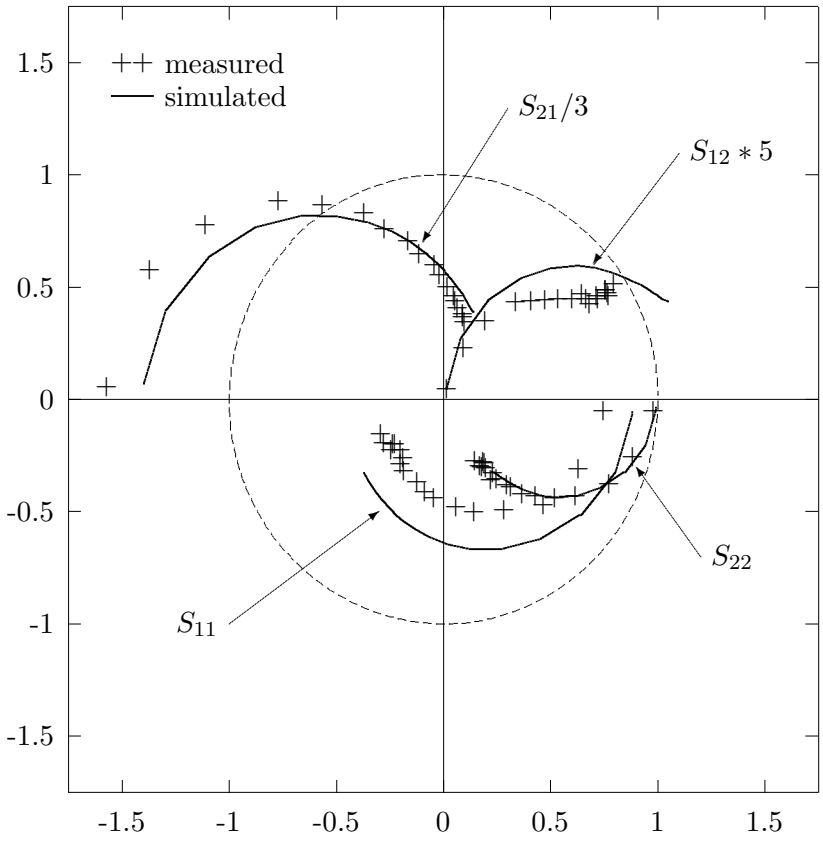

Fig.6. HBT S–parameters as functions of frequency; initial values of parameters.

Presently, the FIT–S extractor contains two different optimization techniques:

- Quite robust but rather slow direct search method of Nelder and Mead [NeMe] (as in [Garw]) which requires only function evaluations, not derivatives (so it is 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 method 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 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.

- A comprehensive quasi–Newton algorithm (the E04JBF routine from the NAG library [NAG]) 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

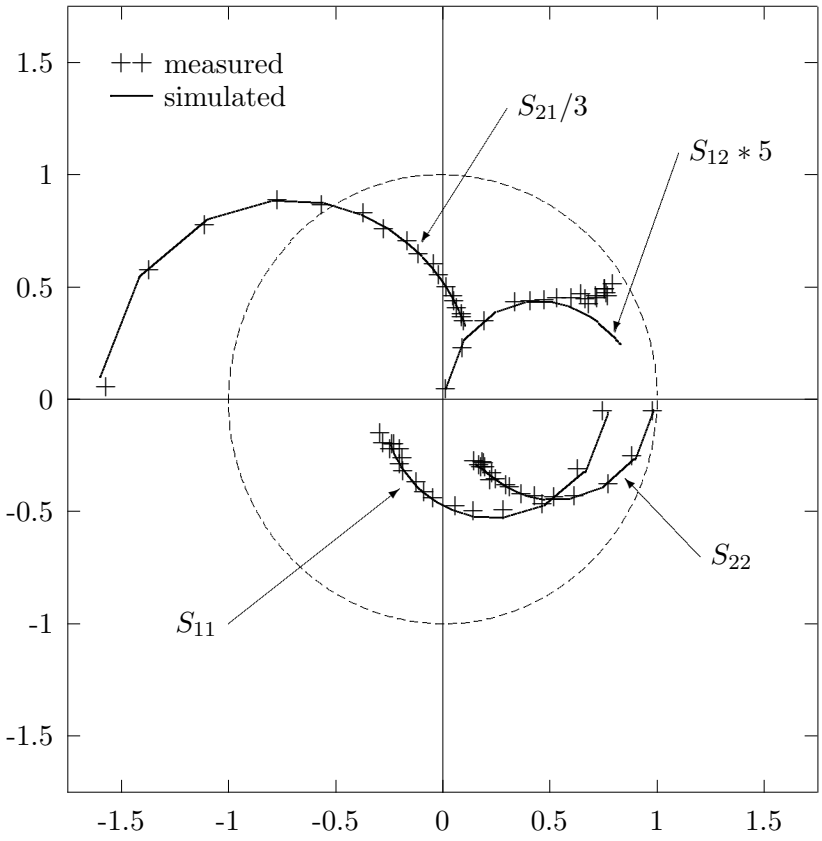

Fig.7. HBT S–parameters as functions of frequency; extracted values of parameters.

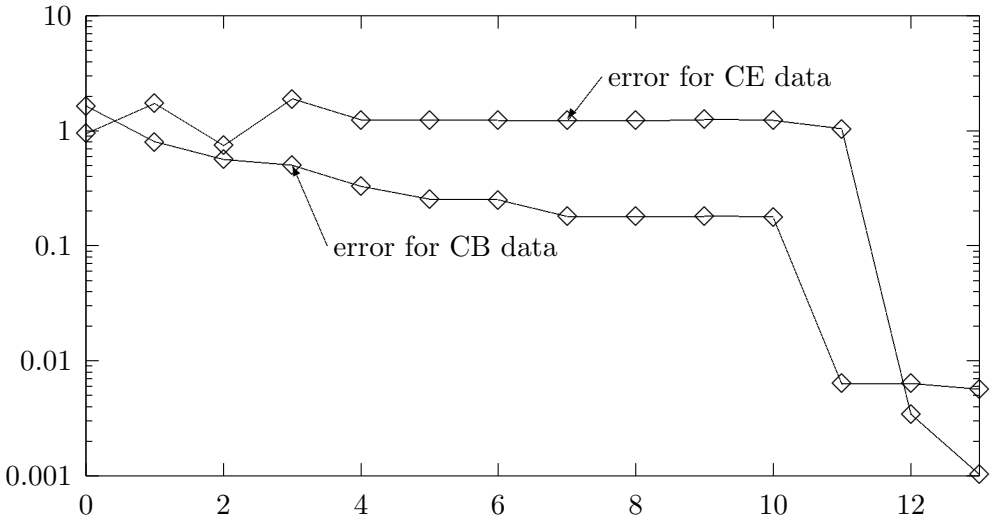

Fig.8. The values of error functions for consecutive steps of optimization.

which have continuos first and second derivatives, it usually works well even if the derivatives have occasional discontinuities. Explicit 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 direct search method of Nelder and Mead, but it also is more sensitive to large changes of function values.

Normally, the initial optimization is performed by the direct search method which 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 direct search method. Fig.8 shows the changes of the value of the objective function in consecutive steps of the optimization.

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 corresponding values of the error functions at the solution:

optimization method	CPU time (in min)	objective function at solution
direct search	178	7.5E-3
quasi–Newton	82	2.4E-3
direct search + quasi–Newton	70	2.0E-3

In the first two cases, the optimizations were restarted until no improvements was obtained. In the third case (i.e., direct search + quasi–Newton), the quasi–Newton method was used to refine the fit produced by the direct search method.

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

4. SELF–HEATING EFFECTS

For heterojunction bipolar transistors (HBTs), the most apparent difference with respect to silicon BJTs is the HBT's negative slope of I–V curves. This negative slope can be observed in most of the published HBT's I–V curves, however, it cannot be obtained from the Gummel–Poon model. Experimental results have shown that this negative slope can be explained by the transistor's self–heating effect [GrOk].

In order to take into account specific properties of GaAs/GaAlAs semiconductor devices, such as the high base doping effect, the GaAs material properties and the heterojunction effect, a new thermal–electrical model based on comparative analysis of HBTs and silicon BJTs has been developed [WAK]. The model is based on the following two assumptions: (i) the non–ideal base current is dominated by Shockley–Read–Hall recombination, and (ii) the ideal current gain is determined mainly by the base transport factor. In this model, the temperature dependence of the current gain β is proportional to $e^{E_\infty/kT}$ where E_∞ is the activation energy, k is the Boltzman constant, and T is the absolute temperature (in degrees K). The temperature dependence of the collector saturation current J_s is the same as for silicon BJTs, and a good approximation of GaAs/GaAlAs HBT's behavior can be obtained if the recombination saturation current has the same temperature variation as $J_s/n\beta$, where n

is the ideal factor of recombination currents. Consequently, the only modification needed to represent the self–heating effect in HBTs is the current gain temperature dependence, which affects not only the current gain itself, but also the recombination current. Implementation of this modification in FIT–S (and in fact, in the SPICE–PAC simulation tool) was relatively straightforward because of SPICE–PAC's direct access to model parameters.

In order to simulate self–heating effects of semiconductor devices, an enhancement of the DC Transfer Curve analysis has been developed which provides an independent "external" iteration for each point of the DC Transfer Curve. This "external" iteration is used in FIT–S for finding, at each sweep point of the transfer curve, the increase of the temperature ΔT which is due to the power dissipated in the device P_{diss} using the concept of "thermal resistance" R_{the}:

$$\Delta T \;=\; R_{the} P_{diss}$$

In effect, the enhancement implements mixed–domain simulation which combines DC Transfer Curve analysis with temperature variation and is quite independent of specific properties of simulated devices.

The following outline of the modified implementation the DC Transfer Curve analysis uses parameters `DCstrt`, `DCstop`, `DCincr` and `SOURCE` to denote the initial value of the voltage (or current) sweep, the final value of the sweep, the increment and the independent source used for the sweep, respectively; the logical condition `enhanced` is TRUE only if the external enhancement is used:

```
initialize;
if enhanced then call enhancement(initial_parameters) endif;
value:=DCstrt;
while (DCincr>0 and value<DCstop) or
      (DCincr<0 and value>DCstop) do
   update(SOURCE,value);
   repeat := true;
   while repeat do
      repeat := false;
      solve_the_system_of_circuit_equations;
      if nonconvergent then stop_analysis endif;
      store_results;
      if enhanced then
         call enhancement(parameters);
         if solution_is_unsatisfactory then
            delete_last_results;
            repeat := true
         endif
      endif
   endwhile;
   value:=value+DCincr
endwhile;
```

The `enhancement` procedure is performed after the DC solution for each sweep point is found. This procedure implements, for each sweep point, an additional (temperature) iteration which adjusts the device temperature due to self–heating effects. Its outline is as follows:

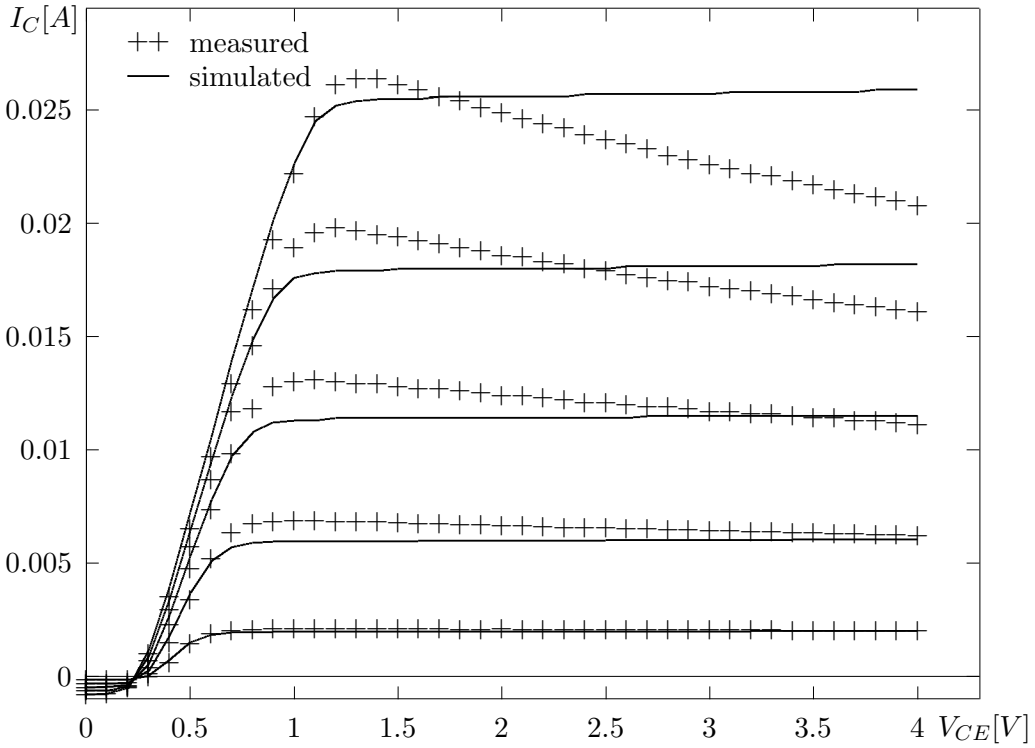

Fig.9. HBT common–emitter DC characteristics; extraction without self–heating effects.

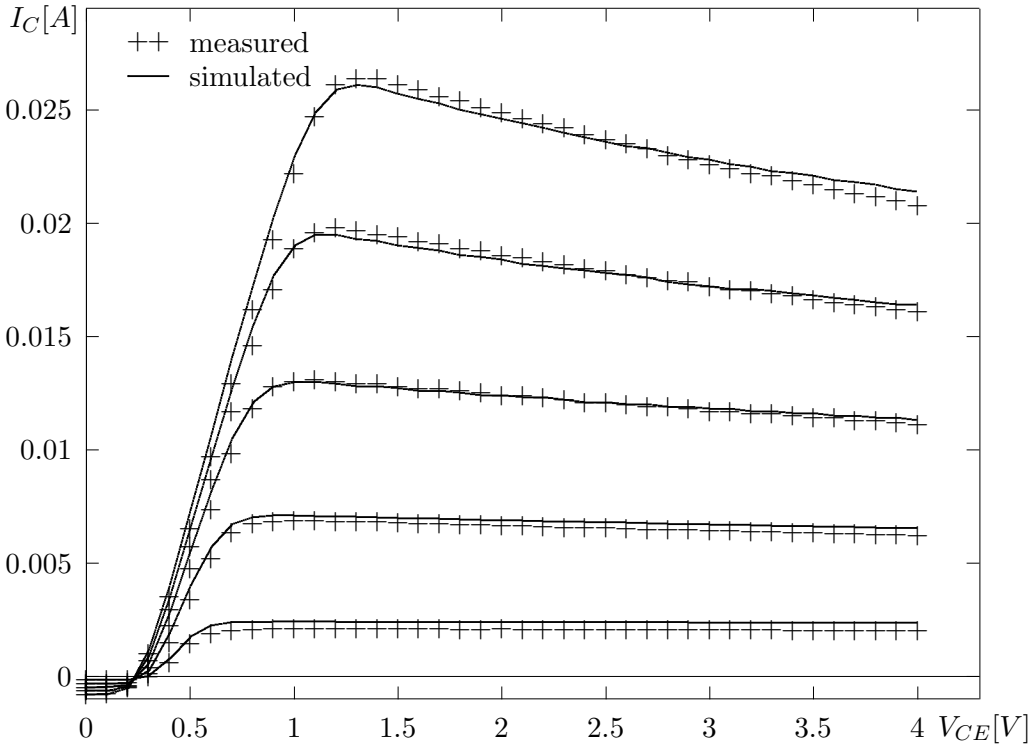

Fig.10. HBT common–emitter DC characteristics; extraction with self–heating effects.

```
if initial_invocation then
   Tref := the_reference_temperature_from_the_circuit;
   Told:=Tref;
   get_reference_values_of_temperature_dependent_parameters_from_the_circuit
endif;
find_temperature_increase_due_to_released_power(Tdelta);
Tact:=Tref+Tdelta;
if abs(Told-Tact) > accuracy then
   update_the_ambient_temperature_of_the_circuit(Tact);
   find_values_of_temperature_dependent_parameters(Tref,Told,Tact);
   Told:=Tact;
   set_return_flag(solution_unsatisfactory)
else
   Told:=Tref;
   set_return_flag(continue)
endif;
```

Fig.9 and Fig.10 compare the measurement data with simulated I–V characteristics without and with self–heating effects taken into account, respectively (it should be noted that characteristics shown in Fig.2 to 7 correspond to a different device for which the self–heatings effects were negligible). Fig.10 clearly shows the negative slopes of simulated results that match very closely the measurement data.

These results indicate that the self–heating effects in HBTs can be represented using only two parameters, the thermal resistance of a transistor R_{the}, and the activation energy E_∞ (the results in Fig.10 were obtained for R_{the}=800 deg $/W$, and E_∞=70 meV).

The dependence of the energy gap E_g and thermal resistance R_{the} of temperature was not considered here, however, for large temperature ranges, these additional dependencies should also be taken into account.

The implementation of mixed–domain analysis is reasonably efficient. The total numbers of iteration steps for DC Transfer Curve analysis without and with self–heating effects (and with temperature accuracy of 0.1 degrees Celsius), are as follows:

base current I_B	without self–heating	with self–heating
160 μA	62	113
320 μA	63	112
480 μA	62	154
640 μA	62	176
800 μA	70	215

The self–heating effects increase the total number of Newton–Raphson iteration steps on average from 2 to 3 times (for this accuracy), and this average value grows as the self–heating effects become more significant.

5. SYMBOLIC SIMULATION

For simulation–based parameter extraction in general, but especially in the case of microwave applications, a significant part of simulation is performed for the small–signal, linear behavior of the circuit. These linear analyses can conveniently be done using symbolic

methods rather then numerical ones; the analyzed circuits are typically very small, and the number of symbols can be significantly reduced by eliminating, as soon as possible, all those symbols whose values cannot be modified during analyses.

The principle of symbolic simulation is to derive analytic (or symbolic) functions describing the circuit responses, using symbols representing (some) circuit parameters (rather than their numerical values). Evaluation of these symbolic functions for specific values of symbols provides the required responses of the circuit.

Circuit functions which are used in parameter extraction are driving–point immitances and various matrices (Y, Z, A, S, etc.) for two–port circuit representation. All these circuit functions can easily be obtained from characteristic polynomials of the analyzed circuit [Chen].

For linear, lumped and stationary circuits, the transfer functions $\mathcal{H}(s)$ of a two–port network are in the form of rational functions of the complex frequency s:

$$\mathcal{H}(s) \;=\; \frac{\mathcal{F}_j(s)}{\mathcal{F}_k(s)}$$

in which the numerator $\mathcal{F}_j(s)$ and the denominator $\mathcal{F}_k(s)$ are characteristic polynomials of the two–port:

$$\mathcal{F}_i(s) \;=\; \sum_{\ell=0}^{n_i} s^\ell \mathcal{P}_{i\ell}(x_1, ..., x_m)$$

and the coefficients $\mathcal{P}_{i\ell}(x_1, ..., x_m)$ are (nested or expanded) polynomial functions of symbolic elements $x_1, ..., x_m$. In the fully expanded form (used in FIT–S) the coefficients are in the sum–of–product form:

$$\mathcal{P}(x_1, ..., x_m) \;=\; \sum_{k=1}^{p} C_k \prod_{\ell=1}^{r} x_{k\ell}$$

where C_k are real numbers, $x_{k\ell}$ are circuit symbols, and p and r depend upon the topology of the circuit.

An equivalent representation of the characteristic polynomials (obtained by extracting common factors and rearranging the terms) is:

$$\mathcal{F}_i \;=\; s^{k_i} \mathcal{T}_i \sum_{j=0}^{n_i} s^j \mathcal{R}_{ij}$$

where each $\mathcal{T}_i$ is a product of a constant C_i and (some) symbols x_{ik}, $k = 1, ..., m_i$

$$\mathcal{T}_i \;=\; C_i \prod_{k=1}^{m_i} x_{ik}$$

and each $\mathcal{R}_{ij}$, $j = 0, 1, ..., n_i$, is a sum of products

$$\mathcal{R}_{ij} \;=\; \sum_{k=1}^{\ell_{ij}} C_{ijk} \prod_{\ell=1}^{m_{ijk}} x_{ijk\ell}$$

Since all frequency–domain analyses are performed for the circuit with the same topology, the generation of symbolic functions can be done only once. Furthermore, the number of of symbols which can change their values during one optimization (or one "partial extraction") is rather small, and includes a subset of extracted parameters (updated in the optimization process) and all those symbols which depend upon the operating point solution (parameters of small–signal transistor models). All such symbols are called *variable* symbols while the remaining symbols are called *fixed* symbols. It should be observed that all *fixed* symbols can be replaced by their numerical values during the generation of the symbolic functions, reducing the functions and simplifying their subsequent evaluations. The values of all *variable* symbols are retrieved after each operating point solution, and are used for evaluation of all $\mathcal{T}_i$ and $\mathcal{R}_{ij}$ products. This results in a reduced form of symbolic functions

$$\mathcal{F}_i^{(r)} \;=\; s^{k_i} A_i \sum_{j=0}^{n_i} s^j A_{ij}$$

where all A_i and A_{ij}, $j = 0, 1, ..., n_i$, are constants provided that no frequency–dependent elements are used. Only this very simple polynomial form needs to be evaluated during the frequency–domain analysis.

Symbolic simulation is included in the general extraction scheme in the following way:

```
retrieve_the_values_of_all_fixed_symbols;
generate_symbolic_products;
continue_optimization := true;
while continue_optimization do
   update_the_values_of_variables;
   error := 0;
   for each frequency_domain_data_group do
      update_op_point_voltages_and_currents;
      find_the_op_point_solution;
      retrieve_the_values_of_variable_symbols;
      evaluate_coefficients_of_reduced_functions;
      for each frequency do
         evaluate_reduced_functions(val);
         convert_to_circuit_responses(val,results);
         error := error + differences(data,results)
      endfor
   endfor;
   for each non_frequency_domain_data_group do
      find_circuit_responses(results);
      error := error + differences(data,results)
   endfor;
   if error < accuracy then continue_optimization := false
   else get_new_values_of_variables(error) endif
endwhile;
```

the step `generate_symbolic_products` generates the products $\mathcal{T}_i$ and $\mathcal{R}_{ij}$ above using an intermediate, partially processed representation of the Coates flowgraph [Chen], and the step `evaluate_coefficients_of_reduced_functions` calculates the values of A_i and A_{ij} using the retrieved values of variable symbols.

A comparison of numerical and symbolic simulation is given for parameter extraction of a submicron (0.25 μ) GaAs FET on InP substrate. The performance of symbolic and numerical analyses is compared by measuring the total execution times for typical extraction steps. These results are summarized in the following table in which the columns correspond to data groups with 10, 20, 50 and 100 frequency values (the execution times are in seconds, on a SPARCstation 2, for 100 optimization steps):

number of frequencies (in data groups)	10	20	50	100
execution time – symbolic analysis	3.30	3.62	4.32	5.61
execution time – numerical analysis	8.95	12.9	24.8	44.6
speedup	2.7	3.6	5.7	8.0

The number of variable symbols influences the execution time rather insignificantly, especially for data groups with large number of frequency values; the number of variable symbols affects the evaluation of the coefficients of reduced functions, but this evaluation is performed only once for each data group.

It should be noted that the circuit description must specify several 'special' parameters if symbolic analysis is to be used; Appendix 1 describes these parameters in greater detail.

6. 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 modeling, and once such a relationship is 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, which can be defined by users without detailed knowledge of the whole software, is an important aspect of extraction tools.

In order to support different sets of parameters, an interface has been incorporated into the FIT–S 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]); 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.

The incorporation of these routines into the structure of the FIT–S program (Fig.1) is shown in Fig.11

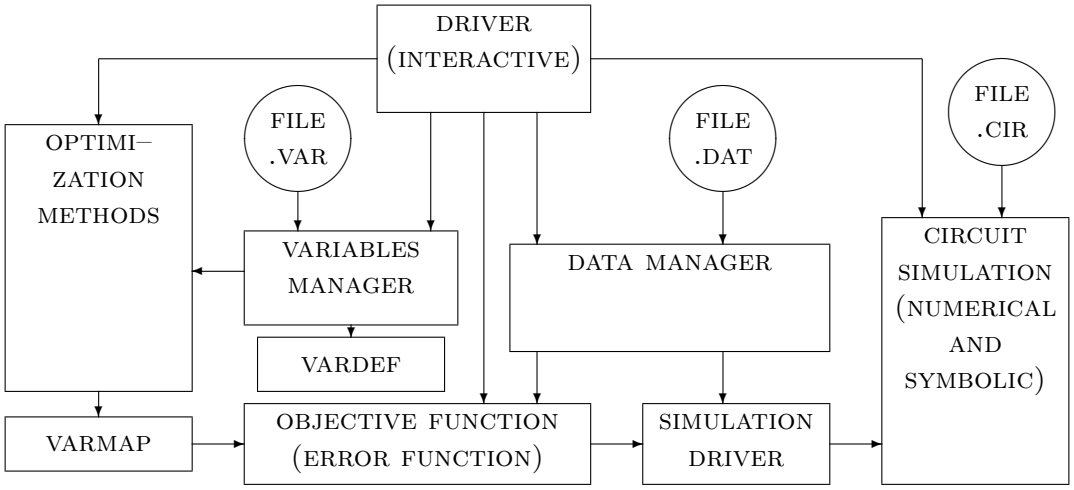

Fig.11. Parameter conversion in the FIT–S program.

For "nonstandard" applications, these two routines must be defined by the user and linked with the FIT–S 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., both routines 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
```

A potential application of parameter conversion can be illustrated by the following simple example which 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 typical 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.

7. 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. An integration with an existing symbolic simulator (SYBILIN) is implemented to speed up the extraction process by eliminating (numerical) analyses of the linearized circuit (for frequency–domain data). A mixed–domain analysis, combining the DC analysis with temperature variation, is provided to allow self–heating effects to be taken into account.

As expected, a strong correlation between the performance of the extractor and the sensitivity of the error function with respect to the optimization parameters has been observed; the fitting process is more efficient if the sensitive parameters are used as optimization variables. Consequently, the extraction process can be simplified considerably if optimizations are performed on small but relevant subsets of optimization variables. Mechanisms needed for such selective optimizations (or "partial extractions") are built into the FIT–S program. Furthermore, it is our experience that the extracted values of insensitive parameters are often rather inaccurate, so their usefulness for tuning technological processes or device designs may be limited. Parameter scaling might be considered to reduce the differences in sensitivities, however, scaling must be used carefully as it affects the objective function.

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–S; some further extensions of this formalisms should be included in future versions of the program.

Both optimization methods available in FIT–S 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, 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.

R e f e r e n c e s

[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.

[Chen] W.K. Chen, "Applied Graph Theory –Graphs and Electrical networks"; North–Holland 1976.

[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.

[KMGB] A. Konczykowska, V. Morin, J. Godin, M. Bon, "Symbolic analysis for CAD microwave circuits"; Symp. on Computer Aided Design of Microwave Circuits, London, England, 1985.

[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.

[WAK] H. Wang, C. Algani, A. Konczykowska, W.M. Zuberek, "Temperature dependence of DC currents in HBT"; Proc. IEEE Int. Microwave Symp., Albuquerque, NM, pp.731-734, 1992.

[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.

APPENDIX 1 – INPUT DATA

FIT–S uses three different types of input information stored in three different files:

- the circuit description file (file `.cir`),
- the variables file (file `.var`), and
- the measurement data file (file `.dat`).

Circuit description file

The circuit description file contains the (SPICE–style) description of a circuit that corresponds to the measurement environment (including all parasitics). 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 elements.

The following example shows 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) updated during the extraction process to the values used during the measurements:

```
.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)
**** 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
**** 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 analysis
.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
```

Note 1: Symbolic simulation can be used only if the circuit description file contains a directive `.ACSYMB`, a new directive introduced for integrated symbolic analysis, which specifies the subcircuit for the small–signal analysis, its input port and its output port. The subcircuit must be indicated by its expansion (an "X"–class name) or "*" if the whole circuit is to be analyzed. The ports are indicated by pairs of nodes, and if the second node is the "reference" node (i.e., "0"), it can be omitted.

For the previous example of circuit description (in which the subcircuit `TRPOL` is defined for AC analysis), the `.ACSYMB` directive can use either `X1` or `X2` as the `TRPOL` expansion; if `X1` is used, the corresponding ports are (1,0) as input and (2,0) as output:

```
.ACSYMB ckt(X1) input(1) output(2)
```

(the order of input, output and circuit sections is immaterial).

Note 2: Self–heating effects can be taken into account only if the circuit description contains a directive `.SELECT DC`, a special directive introduced for passing information from the DC iteration to the thermal iteration; its syntax is very similar to that of `.PRINT`:

```
.SELECT DC var1,var2,...
```

where each `var` is either a voltage between nodes `n1` and `n2` written as `V(n1,n2)` (or simply `V(n1)` if `n2` is zero), or the current flowing through an independent voltage source denoted by `I(Vsource)`.

For analysis of self–heating effects, the power dissipated in the device is determined from device voltages and currents which must be indicated for the thermal analysis in the `.SELECT DC` line; FIT–S assumes that the `.SELECT DC` directive indicates four variables for each analyzed device, describing two pairs of voltage and the corresponding current at some device terminal. In the case when self–heating effects of more than one device are analyzed, the `.SELECT DC` directive defines several 4–tuples of variables, and specific tuples are indicated, in the data groups, by positive values of their thermal analysis indicators (e.g., defined by the `!DCTEMP` lines).

Since the example circuit description (shown above) contains common-base and common-emitter transistor configurations, the corresponding `.SELECT` line should contain two 4–tuples of variables; assuming that the first tuple corresponds to the common–emitter configuration, the `.SELECT` line could be:

```
.SELECT DC V(201) I(VCI) V(202) I(VIB)  V(101) I(VC) V(103) I(VE)
```

and then self–heating effects for common–emitter data must use thermal analysis indicator equal to 1, and common–base data – thermal analysis indicator equal to 2.

Variables file

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` (the lines with "*" as the leading character are comment lines):

```
* 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
.......     .......  ...........  .......  ...........
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
```

Note 1: All identifiers of variables must be valid circuit variables in the sense of SPICE–PAC [Zub1] otherwise errors are reported and further processing of data is discontinued.

Note 2: The ordering of numerical attributes of variables is fixed as must follow the order indicated above, i.e., the minimal value, the nominal value, the maximal value and the actual value.

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 file

Measurement data normally include DC measurements, frequency–domain (AC) and/or time–domain (TR) measurements, but harmonic measurements and noise measurements can be included as well.

Measurements of the same type (e.g., DC for a particular configuration, AC for a given bias point, etc.) form a data "group". Each data group is composed of:

- a sequence 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, followed by
- 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 `!IDENT` contains:

- an optional (unique) name of the group which cannot be longer than 8 characters and which is followed by the equality sign,
- the type of data which can be DC, AC, TR, FO or NO for the DC, frequency–domain, time–domain, harmonic and noise measurements, respectively,
- an optional group parameterization (enclosed in square brackets), and
- an optional list of group parameters enclosed in parentheses; these group 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 **DC data** 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
```

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 the name of an independent voltage source controlling the collector–base voltage of the transistor; see the previous example of 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`; again see the previous example of circuit description).

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). On the other hand, the convergence properties of DC analysis are usually much better if the values of the independent variable are monotonously increasing or decreasing.

Note: The `!ERRNUM` line in the header defines the error function (as the "standard function # 10").

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 describes typical common–emitter characteristics of a bipolar transistor:

```
!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 the DC analysis. 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 consistent with the circuit description in the circuit description file.

AC data groups usually 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 type of data (on the `!IDENT` line) can be AC (which indicates general frequency–domain data) or `SPAR` for S–parameter data (as in the example below); type `SPAR` is required if data conversions are to be used (commands STOZ, STOY etc; Appendix 2). 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=SPAR(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
```

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.

Time–domain data groupd are indicated by `TR` on the `!IDENT` line; parameters of the time–dependent function of an independent source are typically specified as group parameters for this type of data:

```
!IDENT:T3=TR(VIN'#1=0.2)
!STEPMAX=1E-11
    TIME          V(2)
  4.00E-10    4.72996E-01
  5.00E-10    1.48589E-01
  6.00E-10   -4.56016E-01
  6.50E-10   -5.55942E-01
  7.00E-10   -4.41838E-01
  8.00E-10    1.75049E-01
  9.00E-10    4.72995E-01
```

The values of `TIME` can be distributed rather arbitrarily (but they must be monotonously increasing) as SPICE–PAC performs "data–driven" time–domain analysis in which a table of explicit values determines the timepoints.

Note 1: The `!STEPMAX` line in the group header defines the maximum value of the timestep for the time–domain analysis.

Note 2: The circuit description shown previouly cannot be used for time–domain analysis with the above data as the circuit does not contain the independent source `VIN`. In order to use time–domain measurements, another section in the circuit description file is needed, for example:

```
**** TR analysis
   VIN  51 0 SIN(0.0,0.5,2E9,0.0,0.0)
   RIN  51 0 50.0
   XEC  52 50 0 99 TRPOL
   ROUT 52 0 50.0
```

Harmonic data groups are indicated by `FO` on the `!IDENT` line; parameters of the time–dependent function of an independent source are typically specified as group parameters for this type of data:

```
!IDENT:F1=FO(VIN'#1=0.355)
!FFREQ=2E9
!TSTART=5E-10
!STEPMAX=1E-11
   HFREQ        V(2)
  2.0E+09    4.16304E-01
  4.0E+09    8.06085E-03
  6.0E+09    1.85168E-03
  8.0E+09    3.63230E-04
  1.0E+10    2.54212E-04
  0.0E+00   -1.18281E-02
```

The data lines indicate the values of harmonic frequencies and the corresponding measurements (there can be several measurement values for each harmonic frequency). The frequencies must be increasing, with the last value (zero frequency) corresponding to the

DC component of analysis. The value of the fundamental frequency must be specified by a `!FFREQ` line in the group header. Moreover, `!STEPMAX` and `!TSTART` lines in the header define the maximum timestep and the start time, respectively, both for the time–domain analysis performed before the proper harmonic analysis.

Note: The circuit description shown previously cannot be used for harmonic analysis with the above data as the circuit does not contain the independent source `VIN`. In order to use harmonic measurements, an addition similar to the time–domain data is needed.

Noise data groups are indicated by `NO` on the `!IDENT` line; noise data groups are usually parameterized, with column headers corresponding to SPICE–defined noise "outputs" (`INOISE` and `ONOISE` in this case); noise analysis is performed as an extension of frequency–domain analysis, so the values of independent variable are frequencies which can be specified in arbitrary order:

```
!IDENT:N2=NO[VIN/V(5)]
    FREQ       INOISE    ONOISE
  1.00E+03    6.79E-09  4.70E-07
  1.00E+04    6.79E-09  4.70E-07
  1.00E+05    6.79E-09  4.70E-07
  1.00E+06    6.80E-09  4.51E-07
  1.00E+07    6.92E-09  1.52E-07
  1.00E+08    1.61E-08  3.04E-08
```

Note: The circuit description shown previously cannot be used for harmonic analysis with the above data (the independent voltage source is undefined).

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).

"Special" comment lines

The following "special" comment lines are used in the measurement data files:

`!IDENT`	the identification line (all types of data)
`!ERRNUM`	the error function (all types of data)
`!TEMP`	the ambient temperature (all types of data)
`!DCTEMP`	the thermal analysis indicator for DC analysis
`!STEPMAX`	the max timestep for time–domain analyses
`!TSTART`	the start time for Fourier analysis
`!FFREQ`	the fundamental frequency for Fourier analysis

APPENDIX 2 – COMMAND LANGUAGE

FIT–S 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–S can also be used in a non–interactive mode, in which case a complete sequence of commands (sometimes 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 accepts incomplete commands prompting 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))
```

A special version of the command

```
data(*)
```

selects all available data groups.

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

VAR – specifies selection of variables (i.e., extraction parameters) as a sequence of indicators or names of variables or their subranges; 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 Appendix 1 – an equivalent variable selection is:

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

A special version of the command

```
var(*)
```

selects all defined variables.

Each VAR command overrides all previous selections of variables, so only the most recent command is used in subsequent extraction steps.

ACT – lists all variables with their lower and upper bounds and actual and nominal values; the ordering of values is as in the variables file (Appendix 1), i.e., the minimum values, the nominal value, the maximum values and the actual value.

SEL – lists all selected variables (selected by VAR) with their lower and upper bounds and actual and nominal values.

FIT, SFIT – selects the optimization algorithm and specifies its parameters; FIT and SFIT use 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)
```

FIT and SFIT differ for frequency–domain analyses; FIT always performs numerical analysis while SFIT uses symbolic approach for frequency–domain analysis; the two command are equivalent if no frequency–domain groups are selected.

STEP – defines the level of monitoring during the optimization iteration. The default level monitors all optimization iterations, displaying the iteration number and the corresponding value of the error function, which is equivalent to `step(1)`. `step(10)` results in monitoring every 10–th optimization iteration. `step(+)` causes monitoring of those iterations which improve the objective function, while `step(-)` turns off the monitoring process. `step` (without any argument) displays the present level of monitoring.

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

ERROR, SERROR – outputs the result of evaluation of the error function for the selected data groups; ERROR and SERROR use one parameter which indicates the nominal or actual values of variables by `nom` or `act`, respectively. Similarly to FIT and SFIT, ERROR performs all required analyses numerically, while SERROR uses symbolic approach to frequency–domain analyses. ERR and SERR can be used as abbreviations of ERROR and SERROR.

PLOT – similar to ERROR but instead of evaluating the error function, it invokes the plotting facilities for measurement data and simulated results of selected data groups.

FTFM – evaluates the cut–off frequency (f_t) and the maximum oscillation frequency (f_{max}) for the measurement data and simulation results of all selected AC data groups (the H_{21} characteristic and the Mason gain curve are used for evaluation of f_t and f_{max}).

SAVE – stores the set of variables with their bounds and actual and nominal values in a file; the command may have one parameter which 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 string parameter which specifies the name of the new group followed by the equality sign and the name of an existing group with 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., rows 9, 12, 15, 18) and row 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`. A special symbol "`$`" can be used in row specification to denote the "last row", as in:

```
new(newgroup=oldgroup(1(3)$))
```

STOZ, STOY, STOH – create new groups converting existing S–parameter data groups into equivalent Z–parameter, Y–parameter and H–parameter data groups, respectively (so all these commands can be applied to AC data groups only; moreover, the AC data groups must use `SPAR` as the data type – Appendix 1, AC data groups). The groups to be converted must contain all four (complex) parameters. The commands use one string parameter, similar to that of NEW commands:

```
stoz(zgroup=sgroup(1,3,5(3)$))
```

DEL – deletes a group created by NEW or conversion commands; the group is the only argument of this command; only the most recently created group can be deleted (making the second recent 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 (default or indicated) 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.

MODTEMP – redefines the group indicators for thermal analysis (combined with DC Transfer Curve analysis); the default indicators are set to zero, indicating no thermal analysis; if self–heating effects are to be taken into account during DC analysis, the corresponding indicator must be nonzero, and typically is set to '1' (the value of the indicator must be consistent with the `.SELECT DC` line in the circuit description, as described in Appendix 1):

```
modtemp(1=1,3=0)
```

Thermal analysis indicators can also be defined in the data file by `!DCTEMP` lines.

COEF – defines or redefines weight coefficients W_{ij} associated with columns of data groups; default values of these coefficients are equal to '1.0'; 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.

TIME – outputs the value of the current "execution time" (or the processor's time).

MERGE – dynamically combines results of corresponding to several data groups and merges them for plotting; it has one string argument which specifies a sequence of results associated with subgroups; for example:

```
merge(1act(1-3),3nom(2,4,6))
```

combines the results of group '1' corresponding to actual values of variables and associated with columns 1 to 3, with the results of group '3' corresponding to nominal values of variables and associated with columns 2, 4 and 6.

DUMP – creates a new file which contains measurement data and simulation results associated with all data groups selected at the time of DUMP execution, and corresponding to actual or nominal values of variables which is indicated by a parameter; the values of all variables are also copied to this file:

```
dump:file.dmp(act)
```

creates a file `file.dmp` with measurement data and simulation results corresponding to actual values of variables (with some identification information and all variables):

```
number 1 of 2
ICVC-20 X.act  errfun(2)=1.486D-1
21 JAN 94 at 16:41:56  Temp=20 C s-h

Measurement data:

 0.00E+00,-1.60E-04,-3.20E-04,-4.80E-04,-6.40E-04,-8.00E-04,
 1.00E+00, 2.08E-03, 6.85E-03, 1.30E-02, 1.89E-02, 2.22E-02,
 2.00E+00, 2.07E-03, 6.65E-03, 1.24E-02, 1.86E-02, 2.49E-02,
 3.00E+00, 2.04E-03, 6.42E-03, 1.17E-02, 1.72E-02, 2.26E-02,
 4.00E+00, 2.02E-03, 6.22E-03, 1.11E-02, 1.61E-02, 2.08E-02,

Simulation results:

 0.00E+00,-1.60E-04,-3.20E-04,-4.80E-04,-6.40E-04,-7.99E-04,
 1.00E+00, 2.78E-03, 7.68E-03, 1.35E-02, 1.88E-02, 2.17E-02,
 2.00E+00, 2.76E-03, 7.43E-03, 1.29E-02, 1.87E-02, 2.46E-02,
 3.00E+00, 2.73E-03, 7.21E-03, 1.23E-02, 1.75E-02, 2.28E-02,
 4.00E+00, 2.71E-03, 7.02E-03, 1.18E-02, 1.66E-02, 2.15E-02,

number 2 of 2
n2      X.act  errfun(2)=1.052D-1
21 JAN 94 at 16:41:56  Temp=20 C s-h

Measurement data:

 1.30E+00, 2.07E-03, 6.82E-03, 1.29E-02, 1.97E-02, 2.65E-02,
 1.80E+00, 2.06E-03, 6.70E-03, 1.26E-02, 1.89E-02, 2.54E-02,
 2.30E+00, 2.06E-03, 6.58E-03, 1.22E-02, 1.82E-02, 2.42E-02,
 2.80E+00, 2.05E-03, 6.47E-03, 1.19E-02, 1.75E-02, 2.30E-02,
 3.30E+00, 2.04E-03, 6.37E-03, 1.16E-02, 1.69E-02, 2.21E-02,
 3.80E+00, 2.03E-03, 6.26E-03, 1.12E-02, 1.63E-02, 2.12E-02,

Simulation results:

 1.30E+00, 2.77E-03, 7.61E-03, 1.34E-02, 1.97E-02, 2.59E-02,
 1.80E+00, 2.76E-03, 7.48E-03, 1.30E-02, 1.89E-02, 2.50E-02,
 2.30E+00, 2.75E-03, 7.37E-03, 1.27E-02, 1.83E-02, 2.40E-02,
 2.80E+00, 2.73E-03, 7.26E-03, 1.24E-02, 1.77E-02, 2.31E-02,
 3.30E+00, 2.72E-03, 7.15E-03, 1.21E-02, 1.72E-02, 2.24E-02,
 3.80E+00, 2.71E-03, 7.06E-03, 1.19E-02, 1.68E-02, 2.17E-02,

* date : 21 JAN 94 at 16:41:56

TBH'IS         1.00000D-29  2.16500D-23  1.00000D-21  4.51890D-24
TBH'ISE        1.00000D-18  2.39700D-19  1.00000D-16  3.76642D-18
TBH'NE         1.20000D+00  1.42000D+00  2.10000D+00  1.57590D+00
TBH'BF         1.00000D+01  3.06000D+02  1.00000D+03  1.00000D+03
RTH            5.00000D+01  8.00000D+02  1.20000D+03  8.00000D+02
EIN            7.00000D-02  7.00000D-02  3.00000D-01  7.04232D-02
```

PRINT – outputs numerical values of measurement data or simulated results of group columns indicated in its string argument (this command introduced mainly for testing purposes):

```
print(1,1.2,1:2,1'2)
```

It should be noted that PRINT outputs the contents of areas reserved for the indicated data without checking if their values are consistent with other computations.

outputs the values of independent variables for group '1', then the values of measurement data in column '2' of group '1', then the values of simulated results corresponding to actual values of variables and associated with column '2' of group '1', and finally the values of simulated results corresponding to nominal values of variables and associated with column '2' of group '1'.

INPUT – redirects the source of input commands; normally, the commands are entered from an interactive device (e.g., a keyboard), however, a sequence of commands can be stored in a file (a "script") and executed from it. Often such a script is stored in the circuit description file, following the the `.END` line. The command `input` (without arguments) redirects the input to the circuit description file, so consecutive commands will be entered from there and executed, and when the "end–of–file" conditions is detected, the input is returned to the interactive device.

If, during reading commands from a file, `input(*)` is recognized, the input is redirected to the interactive device. Any subsequent `input` command redirects the input back to the file and continues execution of remaining commands.

Command scripts can include comment lines indicated by "#" or "*" as the leading character. Command scripts can also use multi–level "include" files indicated by commands `input(filename)`, where `filename` is the name of the "include" file.

OUTPUT – similar to INPUT but for redirecting the output.

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).