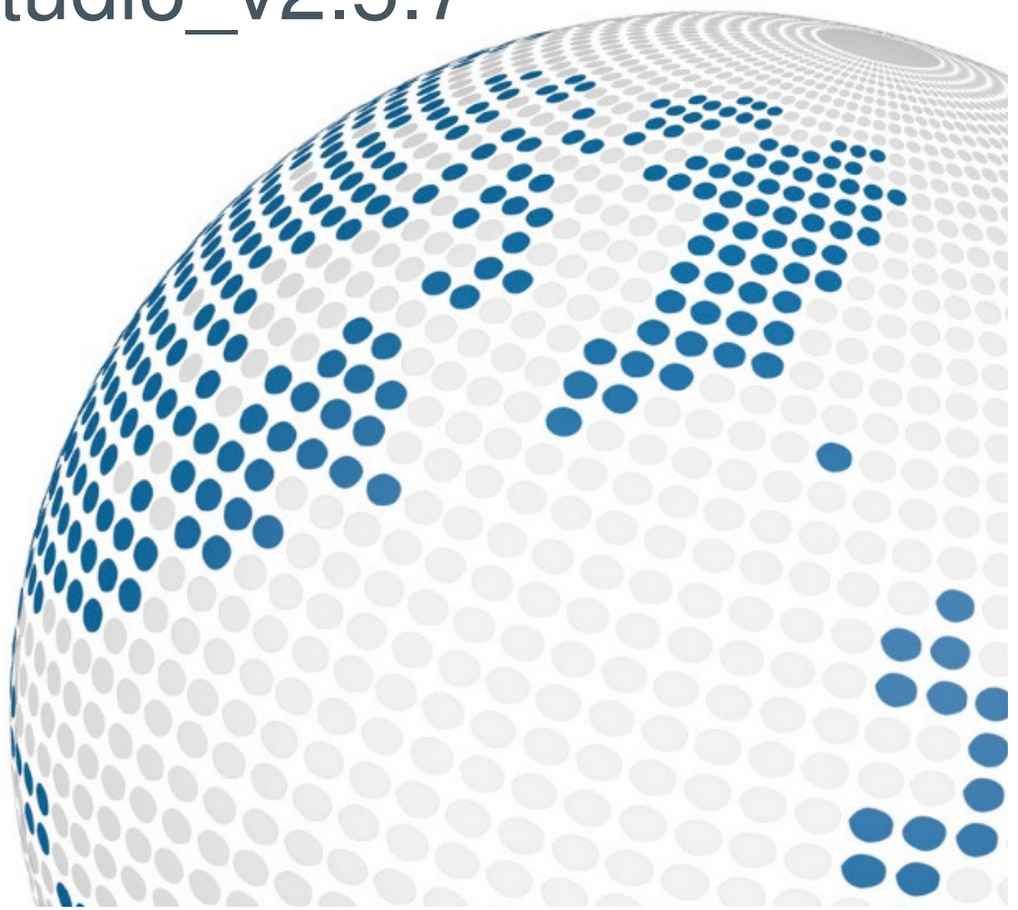
Parameter Extraction with QucsStudio_v2.5.7

32th BipAK Workshop at STMictorelectronics, Crolles, France, November 14&15 2019

Letter Session



Zoltan Huszka

31. October 2019

Parameter extraction with QucsStudio_v257

Introduction

A brief tutorial on optimization was released as of 13-Apr-2019. The `model.sch` schematic suggested a diode parameter extraction using Fig. 1.

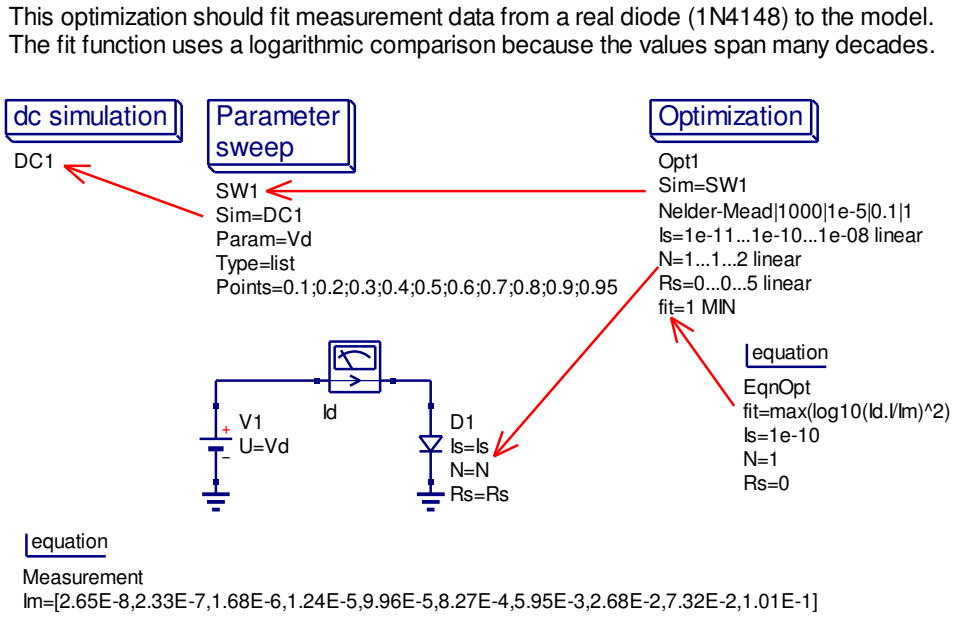


Fig. 1 Optimization scheme

ASCO Manual: *„ASCO is designed to be an encapsulation to a SPICE simulator with the purpose of presenting only a numeric cost to the optimizer."*
The control is performed by the Optimization block. It triggers the SW1 sweep block which sequentially scans the sweep list elements. A DC simulation is invoked at every `Vd` point. When the list is exhausted the `fit` scalar is computed in EqnOpt and it is passed to the optimizer. Using the `fit` scalar, it generates a new `Is, N, Rs` parameter triplet and triggers the next cycle with SW1. A new `Id.I` vector is built which provides a new `fit` value in comparision to the measurement vector `Im` found in an equation block. The process resumes when the optimizer has found an extrema. The `Is, N, Rs` values set in EqnOpt are not obligatory since the Optimization block supplies them in each cycle including the first one.
Internally, the circuit is described by a netlist which `qucssim.exe` is started with by the optimizer. Its only output is the scalar goal quantity `fit` returned to Optimization.

Challenges to be answered next:

- QucsStudio does not have a built-in regression feature
- there is no built-in mechanism given for the insertion of the external measurement vector in the `equation` block
- filling complicated system of model equations one-by-one in `equations` is lengthy and occupies a huge space on the schematic
- possible bugs during the execution

zoltan.huszka@t-online.hu

1. Regression

The detailed derivation and the QucsStudio implementation of a 3-variable linear regression scheme is shown in Appendix I.
Working with regression alone needs a simple dummy circuit like Fig. 2.

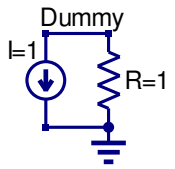


Fig. 2 Dummy circuit for working merely with regression.

Otherwise the system delivers the error meassage „`Circuit is empty!`".
When a joint optimization and regression is performed a restriction shall be observed. Parameters supplied by the regression engine are *not* advised to pass to the encapsulated device (diode here). It can not be granted that the optimized and regressed parameters arrive together to the simulator at the timepoint it is started. If this rule is broken the simulation time increases and the result may become unpredictable even if the system does not deliver error messsages.

1.1 Classical capacitance with full optimization

$$C_j = C_{j0}\left(1-\frac{V_d}{v_D}\right)^{-z_j} \tag{1}$$

The extraction range is restricted to $-2V \le vd \le 0.5V$ by the built-in `range()` function. The measured `cjcm` included in the `cjc_bs122a50` equation block prepared by the Octave routine of Appendix II. is an independent vector. To make it suitable for range selection it has to be given a V_d dependence. It is achieved by multiplying it with the unit vector $U = 1 + vd - vd$.
Only the `cbcpar` parasitic value is used this case from the `Misc` box.

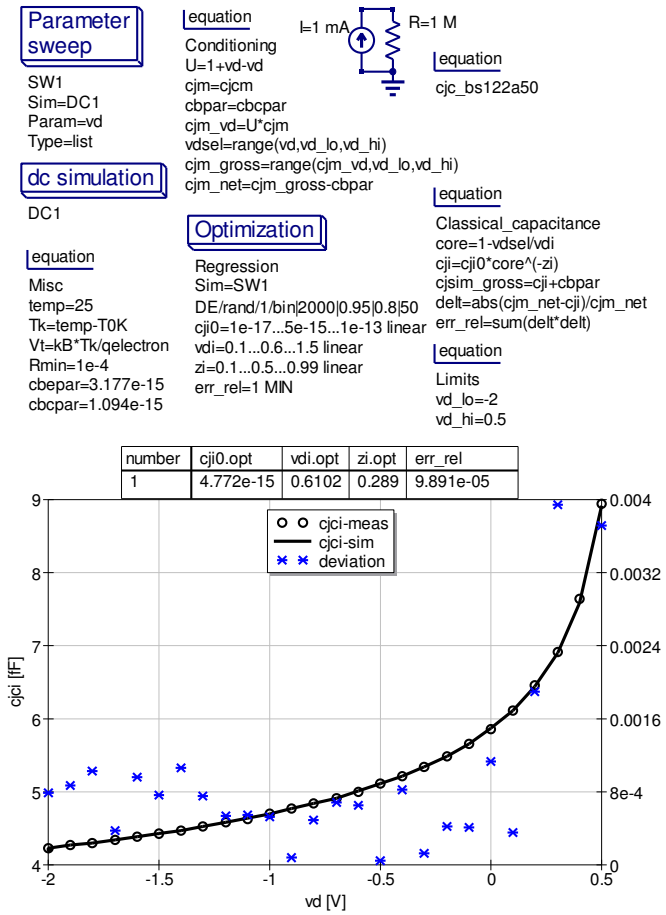


Fig. 3 Collector capacitance extraction by full optimization

1.2 Classical capacitance with 1 optimized and 2 regressed parameter

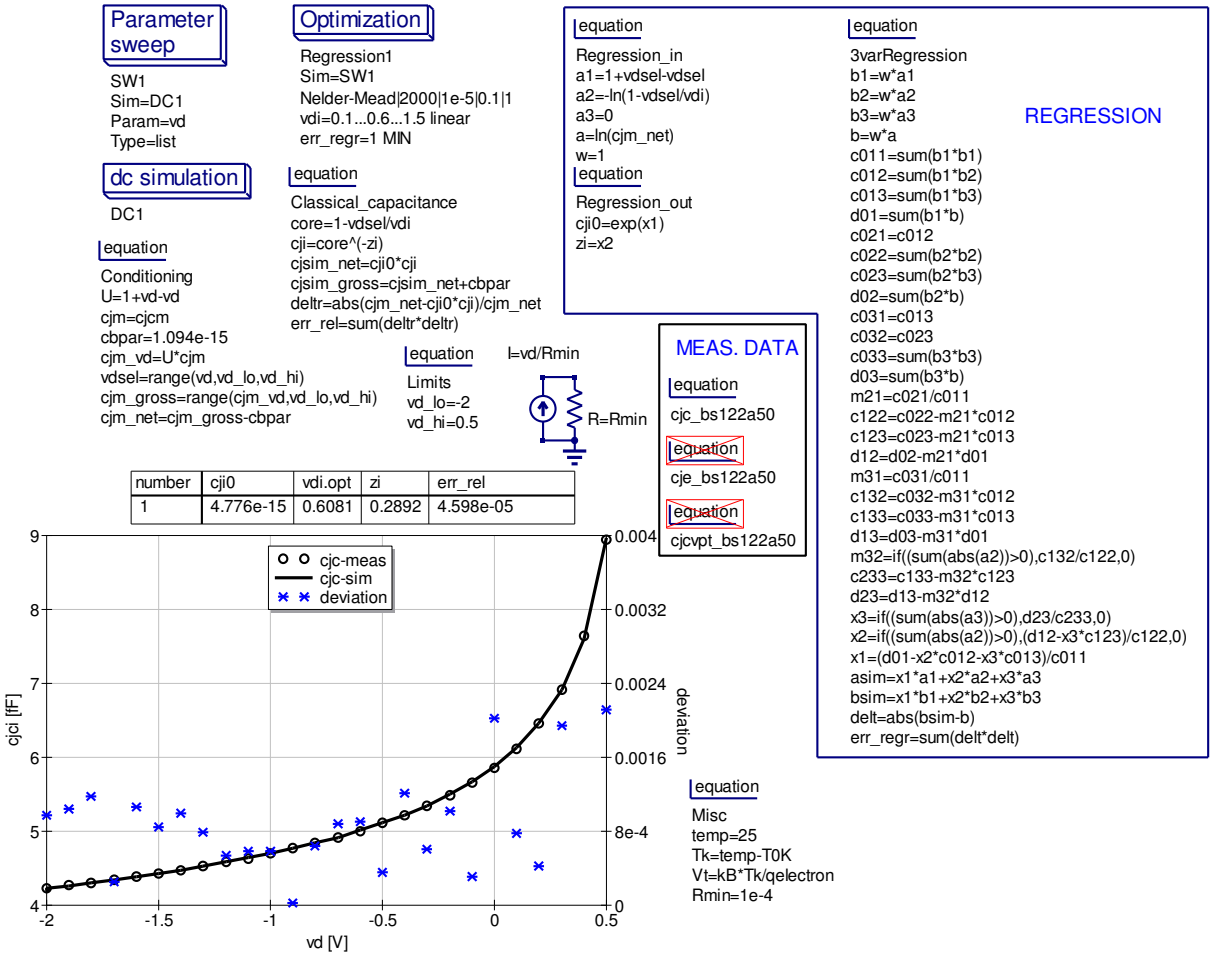


Fig. 4 Collector capacitance extraction by two regressed and one optimized parameter

$$\ln(C_j) = \ln(C_{j0}) - z_j \ln\left(1 - \frac{V_d}{v_D}\right) \tag{2}$$

The logarithmic form (2) is the basis of the last process. Only the v_D nonlinear parameter has to be optimized, $\ln(C_{j0})$ and z_j being linear parameters, can be obtained by regression.

As predicted by theory in Appendix I. the error is less than half of that achieved by full optimization. „Refinement" by optimization is not necessary because the result could not surpass the outcome of the 2 variable regression.

The open circuit voltage of the adopted dummy circuit in this schematic is vd. This will be utilized in the further examples to come.

1.3 Extraction of the Hicum collector capacitance, large VPT

$$C_{jCi} = C_{jCi0}\left[\frac{1}{\left(1 - \frac{v_j}{V_{DCi}}\right)^{z_{Ci}}} \cdot \frac{dv_j}{dv_{B'C'}} + a_{jCi}\left(1 - \frac{dv_j}{dv_{B'C'}}\right)\right] \quad a_{jCi} = 2.4 \tag{3}$$

The component includes a complicated set of nonlinear equations. This formula is used when the punch-through voltage is large, $v_{PTCi} \geq 100$. For smaller punch-through values an even more complex equation describes the capacitance-voltage characteristics. The switch between the two cases is perforemed by the `hicjq` macro in Hicum.

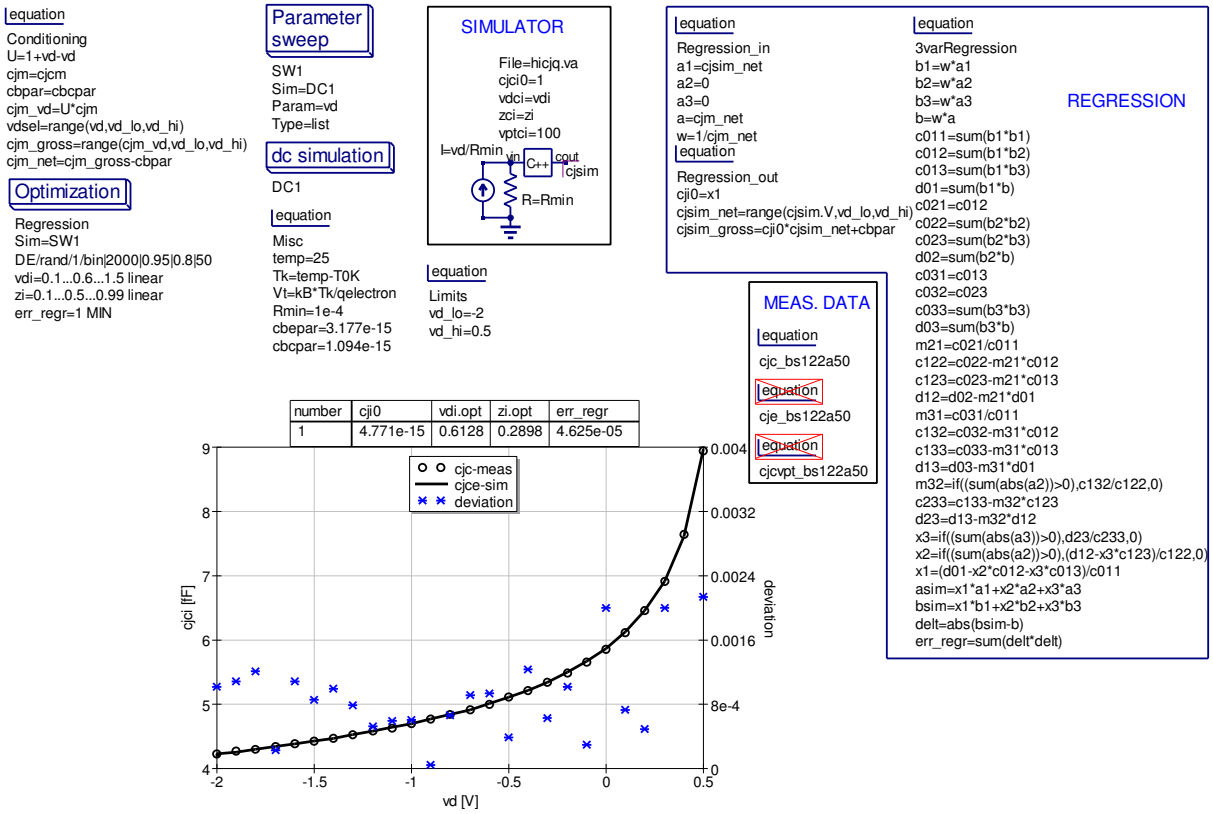


Fig. 5 Hicum collector capacitance extraction, large `vptci`

It would be impractical to implement such a complicated set of equations in the GUI. Instead, the relevant macros have been copied in a Verilog-A code `hicjq.va`, see Appendix III. Finally, the C++ block obtained by the „turn-key" compilation process is used for the computations.

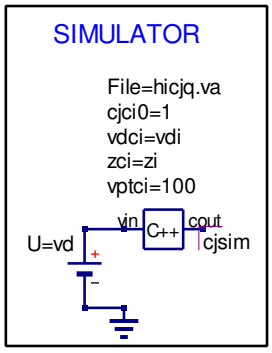


Fig. 6 Voltage source drive

When an ideal voltage source is inserted for driving the `hicjq` simulator on Fig. 6 an error message of Fig. 7 appears and the process aborts. Since the VA code is formulated as a system simulation in terms of voltages the Norton equivalent is not loaded and provides the exact `vd` driving voltage to the capacitance simulator.

The simulation error of `4.625e-15` is somewhat larger than that for the classical capacitance. This is due to the distortion of the Hicum-type capacitance limiting with the parameter a_{jCi}.

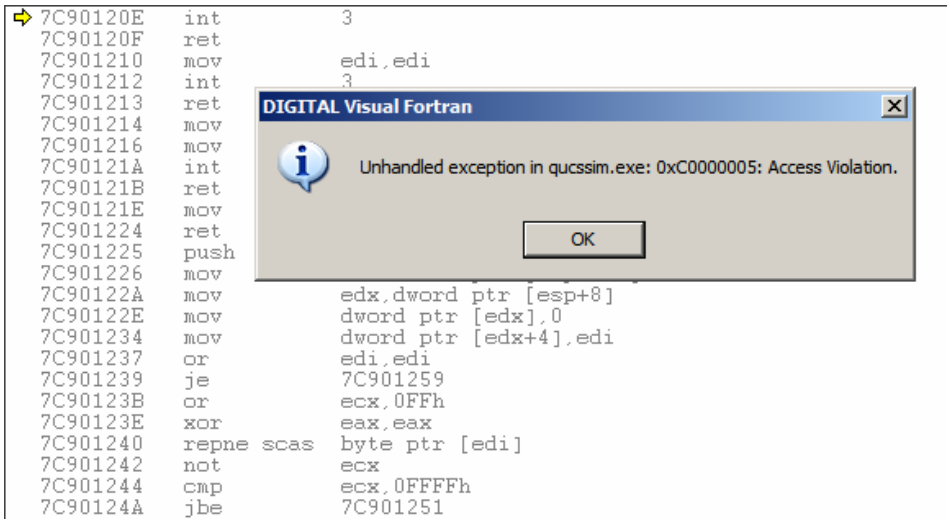


Fig. 7 Error message in response to using the voltage source drive of Fig. 6.

1.4 Extraction of the Hicum collector capacitance, low VPT

	vdci	cjci0	zci	vptci
synthetic	4.776e-15	0.6081	0.2892	5
extracted	3.713e-15	0.6931	0.4469	5.57

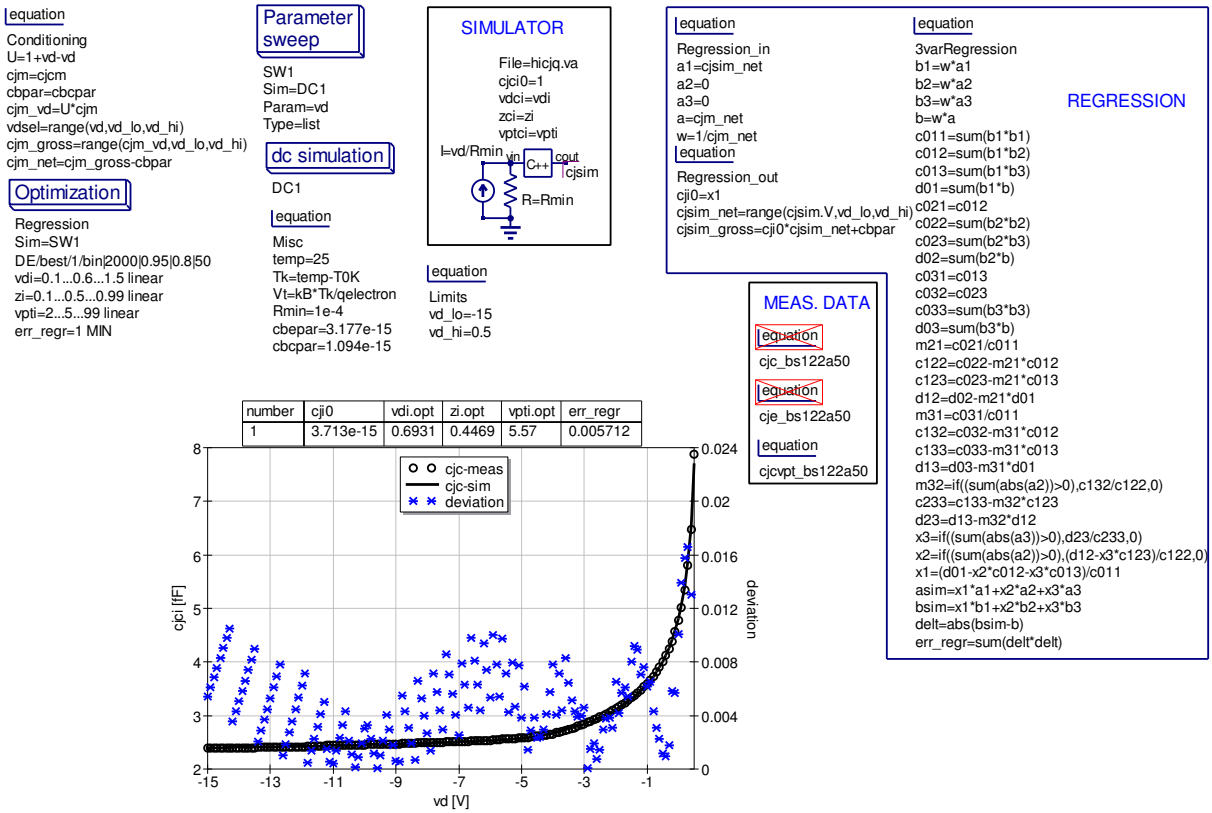


Fig. 8 Extraction of synthetic `cjci` data with small `vptci`

The synthetic parameters can not be returned with reasonable accuracy by the built-in ASCO optimizers. However, apart from the positive top `vd` region the deviations are acceptable. The capacitance appears to be not strictly defined: several parameter combinations may result in the same fit quality.

1.5 Extraction of the Hicum base current

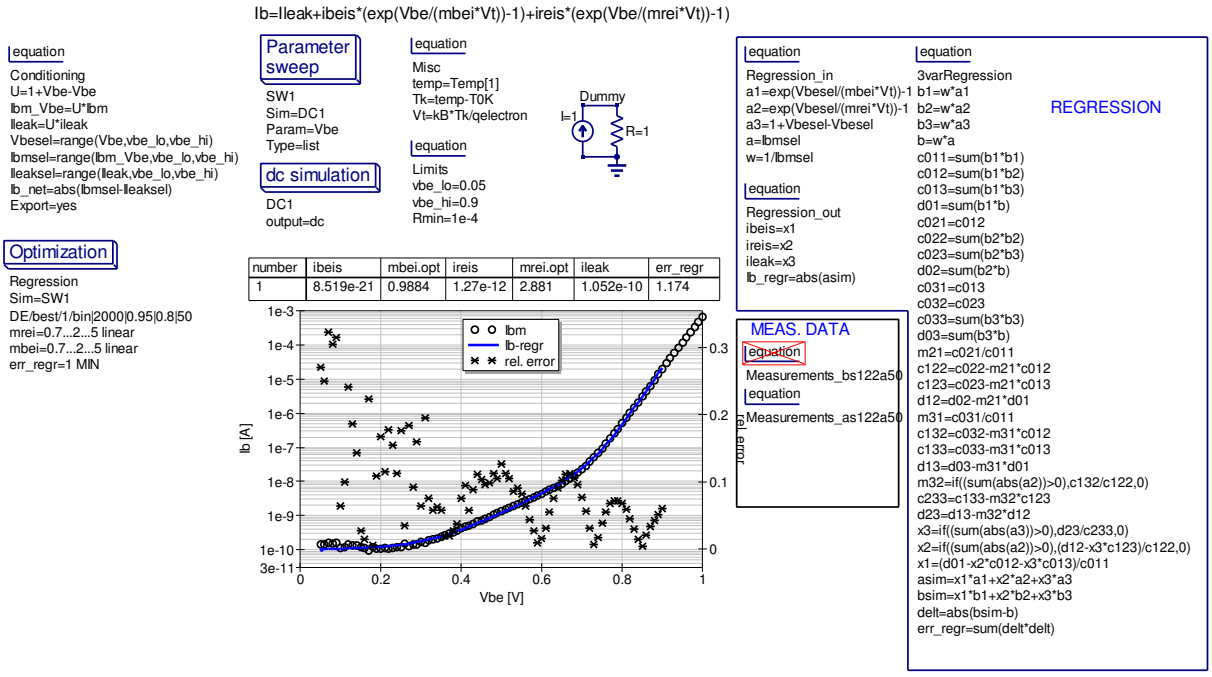


Fig. 9 Extraction of the Hicum base current (see equation in the top line)

This is the most complex task demonstrated here for the proposed 3-variable regression technique. A 5 parameter optimization – not shown – provides practically unusable results.

2. Double sweep device optimization

The following example demonstrates a case when the low bias FT has been properly modeled but there are deviations in the post-peak values on the falloff part. There are several parameters influencing the cutoff region moreover self heating also strongly affects the high current behaviour. Standard regression techniques can not be adopted therefore further refinements can only be obtained by optimization.
It is not the aim to give detailed explanations on the active parameters. The main target is to identify the specialities using the present QucsStudio and suggest workarounds for the possible issues.

2.1 Optimization of FT

The measurement data is saved in the equation `ftdc_st1008` a truncated part of which is shown below.

```
equation
ftdc_st1008
Vbe=[0.7,0.71,0.72,0.7,0.71,0.72,0.7,0.71,0.72]
Vcb=[-0.5,-0.5,-0.5,0,0,0,0.5,0.5,0.5]
Vbc=[0.5,0.5,0.5,0,0,0,-0.5,-0.5,-0.5]
Temp=[27,27,27,27,27,27,27,27,27]
Ibm=[1.212e-008,1.7e-008,2.487e-008,1.135e-008,1.687e-008,2.468e-008,1.068e-008,1.589e-008,2.328e-008]
Icm=[1.287e-005,1.822e-005,2.582e-005,1.306e-005,1.847e-005,2.617e-005,1.321e-005,1.864e-005,2.633e-005]
Ftm=[2.5983e+009,3.3576e+009,4.4924e+009,2.7663e+009,3.6564e+009,4.9674e+009,2.8552e+009,3.8075e+009,5.197e+009]
```

Excerpt 1. Truncated (demo) FT data equation block

The files were written by a slightly modified version of the one shown in Appendix II. It is seen that this is a double sweep or 2D data: `length(Vbe)=31`, `length(Vcb)=3`, consequently `length(Ibm)=31x3` etc. The data in the vectors are arranged in blocks of the inner (`Vbe`) sweep which are tiled along the outermost sweep (`Vcb`).

Basically all data in QucsStudio is arranged in vector format indicating possible higher dimensions by linking dependency markers. It can be seen in QUCS which outputs the data in text format

```
<indep vbe 81>
  +4.00000000000000022204e-001
  +4.10000000000000031086e-001
  +4.20000000000000039968e-001

</indep>
<indep vcb 3>
  -5.00000000000000000000e-001
  +0.00000000000000000000e+000
  +5.00000000000000000000e-001
</indep>

<dep Trise.V vbe vcb>
  +2.93129764124375153406e-006
  +2.56435517220686956381e-006
  +2.20415585881756315734e-006

</indep>
```

The dependent variable `Trise.V` is a vector of `31x3=93` elements. The data is tiled in blocks of `31` elements belonging to the three `vcb` values each. An independent data can be converted to a dependent vector by multiplication with a unit vector `U` of the same length as shown on the sheets. This is an important feature to observe at plotting or determining differences of two quantities.

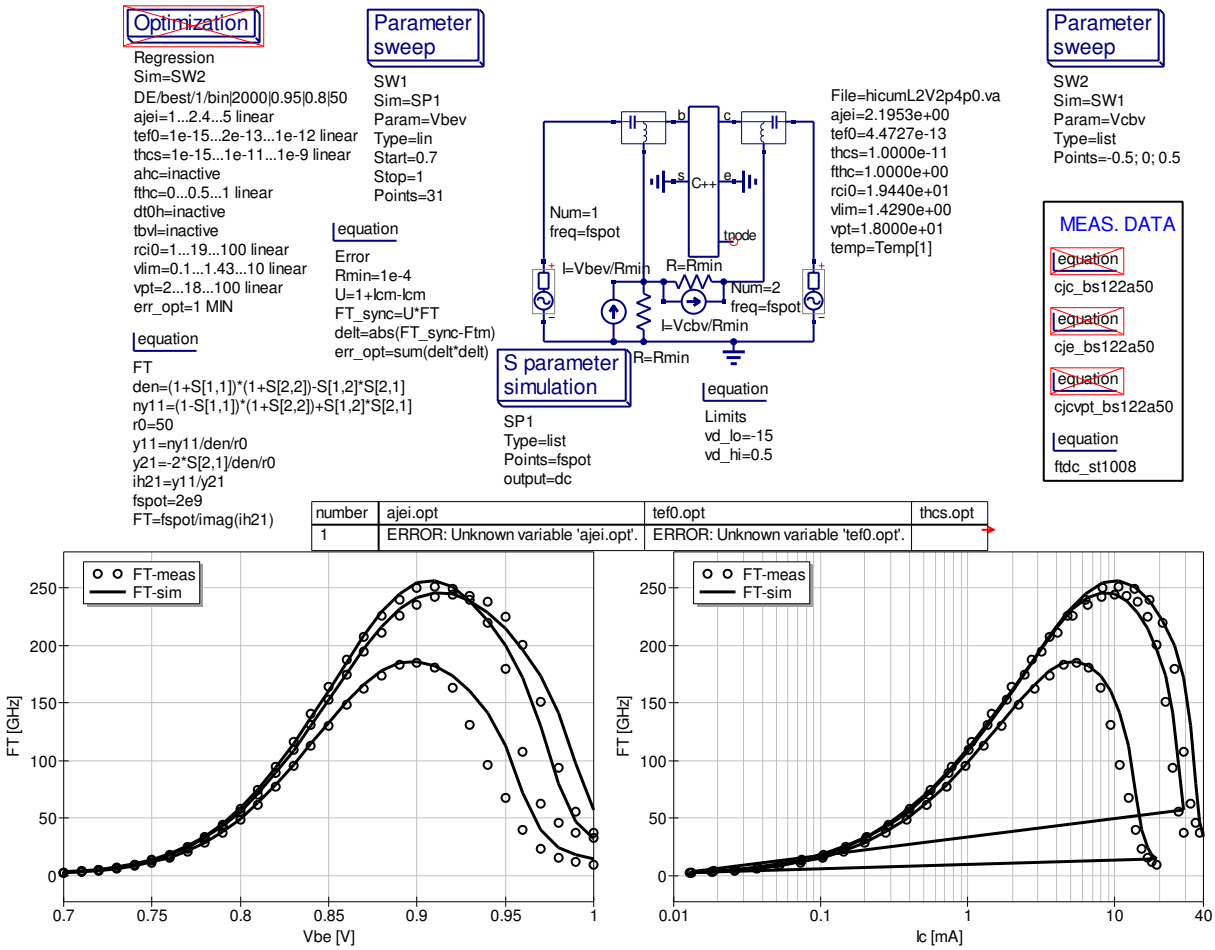
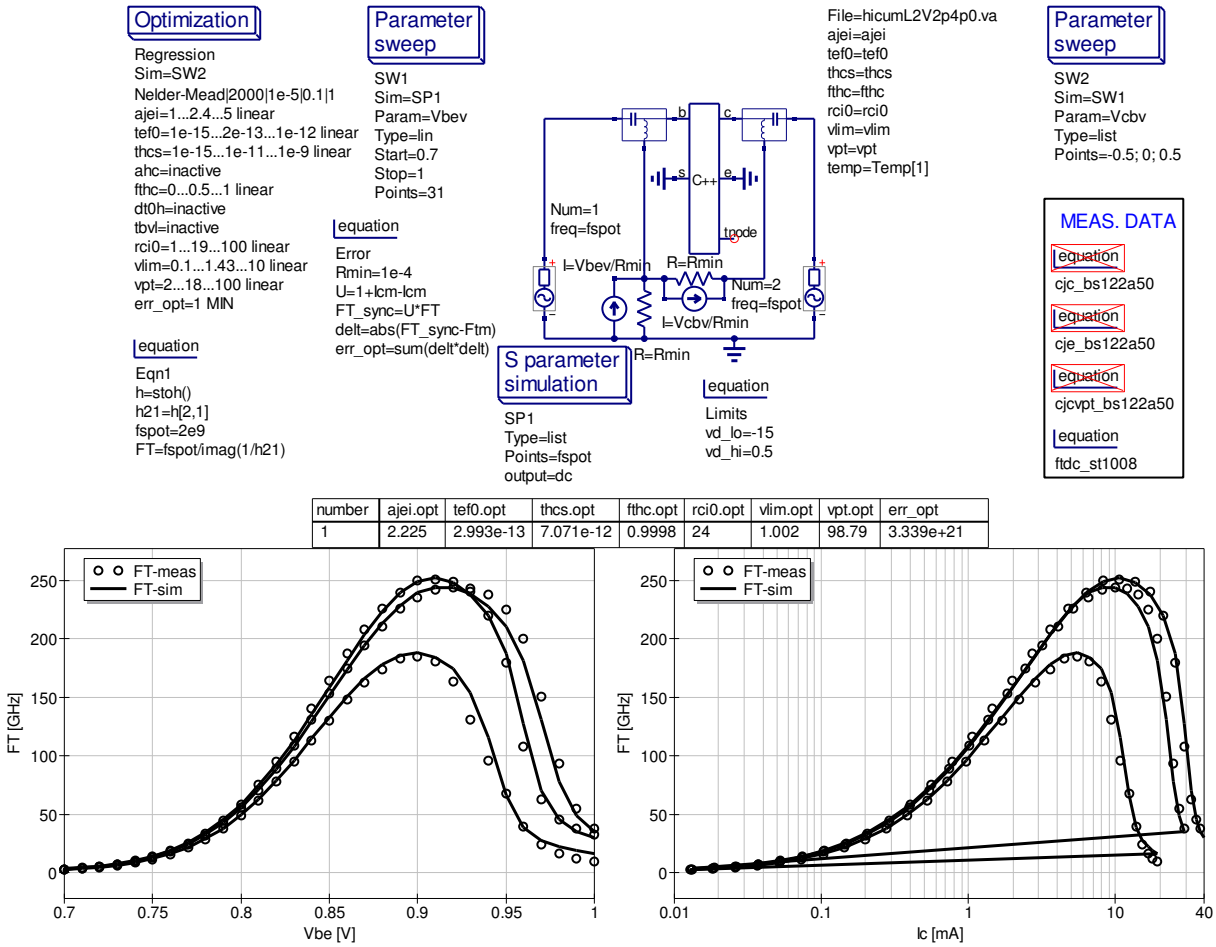

Optimization
Regression
Sim=SW2
DE/best/1/bin|2000|0.95|0.8|50
ajei=1...2.4...5 linear
tef0=1e-15...2e-13...1e-12 linear
thcs=1e-15...1e-11...1e-9 linear
ahc=inactive
fthc=0...0.5...1 linear
dt0h=inactive
tbvl=inactive
rci0=1...19...100 linear
vlim=0.1...1.43...10 linear
vpt=2...18...100 linear
err_opt=1 MIN
equation
FT
den=(1+S[1,1])*(1+S[2,2])-S[1,2]*S[2,1]
ny11=(1-S[1,1])*(1+S[2,2])+S[1,2]*S[2,1]
r0=50
y11=ny11/den/r0
y21=-2*S[2,1]/den/r0
ih21=y11/y21
fspot=2e9
FT=fspot/imag(ih21)
Parameter sweep
SW1
Sim=SP1
Param=Vbev
Type=lin
Start=0.7
Stop=1
Points=31
equation
Error
Rmin=1e-4
U=1+Icm-Icm
FT_sync=U*FT
delt=abs(FT_sync-Ftm)
err_opt=sum(delt*delt)
S parameter simulation
SP1
Type=list
Points=fspot
output=dc
equation
Limits
vd_lo=-15
vd_hi=0.5
File=hicumL2V2p4p0.va
ajei=2.1953e+00
tef0=4.4727e-13
thcs=1.0000e-11
fthc=1.0000e+00
rci0=1.9440e+01
vlim=1.4290e+00
vpt=1.8000e+01
temp=Temp[1]
Num=1
freq=fspot
I=Vbev/Rmin
R=Rmin
Num=2
freq=fspot
I=Vcbv/Rmin
R=Rmin
b
c
s
C++
e
tnode
Parameter sweep
SW2
Sim=SW1
Param=Vcbv
Type=list
Points=-0.5; 0; 0.5
MEAS. DATA
equation
cjc_bs122a50
equation
cje_bs122a50
equation
cjcvpt_bs122a50
equation
ftdc_st1008
number
ajei.opt
tef0.opt
thcs.opt
1
ERROR: Unknown variable 'ajei.opt'.
ERROR: Unknown variable 'tef0.opt'.
FT-meas
FT-sim
FT [GHz]
Vbe [V]
Ic [mA]

Fig. 10 Raw FT



number	ajei.opt	tef0.opt	thcs.opt	fthc.opt	rci0.opt	vlim.opt	vpt.opt	err_opt
1	2.225	2.993e-13	7.071e-12	0.9998	24	1.002	98.79	3.339e+21

Fig. 11 Optimized FT

The improvement of the FT fit is well seen between Fig. 10 (raw data) and Fig. 11 (optimized data). The 7-variable regression runs in a robust way with gradient type optimizers. Since the deviations on Fig. 10 appear mostly at large FT values using absolute instead of relative errors was more appropriate. This gave more significance to the large FT part.

At the `Vbe` dependent plot the simulated FT had the proper dependence to the `Vbev` sweep variable. The measured data were plot between two independent vectors – `Vbe` and `Ftm` - from the `ftdc_st1008` equation.

On the `Ic` dependent plot the measurements are displayed between the independent `Icm` and `Ftm` vectors. However the simulated FT had to be synchronized to `Icm` by the unit vector `U` to yield the proper FT_sync vector for error computation and plot.

Note that the regular parameter sweep blocks were adopted to generate the 2D sweep. Suffix v in `Vbev` and `Vcbv` were added to distinguish them from the full 2D measurement vectors `Vbe` and `Vcb` in the `ftdc_st1008` data container.

As in the former cases the sweeps had to be supplied to the transistor by the Norton VS replacements. The use of the ideal voltage sources will be made possible in a future release. Anyway, the proposed workaround does no imply any appreciable error moreover it spares the current node of the ideal VSs.

3. Summary

- whenever it is possible use pure regression alternatively, regression combined with optimization acting on the smallest possible number of parameters
- the proposed 3-variable regression code is suitable to solve most of the necessary regression tasks. The routine can be simply reduced to perform 2 and 1 variable regressions. The provided weight entry allows a selection between absoulute and weighted – typically relative - errors.
- in mixed regression & optimization mode parameter passing to the encapsulated device is allowed only from the optimizer. Attempts to pass regressed parameters to the device leads to race problems resulting in unexpected results (e.g. false and/or noisy curves) and long simulation times
- whenever it is inquired check the `Export=yes` or `output=DC` radio buttons/roll-down menus for saving the quantities in the QucsStudio dataset
- enclose complicated model equations copy-and-pasted from the device model to local Verilog-A blocks for avoiding coding errors. Since no internal nodes are involved system simulation with voltages as signal flow provides a fast, robust solution.
- do not use ideal voltage sources in the optimization schematic for avoiding an existing bug
- lossy Norton equivalents of non-ideal VSs give robust replacements
- convert the instrument generated mesurement data to CSV files with headers as an interface to the QucsStudio data storage (`equation`) format
- prepare a multi-use data writer in Octave, Matlab, optionally in Fortran for creating the equation components for the measurement data from the source CSVs
- carefully observe the proper data dependencies and convert multi-dimensional internal QucsStudio data to the required single variable vector of selected dependency
- range selection on the primary sweep can be performed by the built-in `range()` function
- at 2D data the equation data containers must be re-created when new secondary sweep ranges are needed
- collection of the optimized parameters is possible by inspection from the data visualization Table(s).
- importing parameters like `ajei=loadQucsVariable('ftopt.dat','ajei.opt')` is an alternative for further processing in Octave. This form is also suitable to invoke simulated vectors to the Octave workspace.

The built-in optimizer of QucsStudio was found to be a useful alternative or extension to the existing processes in the Foundries' extraction flow. When the auxiliary data exchange functions have been constructed parameter extraction either from dedicated measurements or using the whole model of the device is relatively easy to set up. The results can be immediately displayed by the built-in QucsStudio plotting functions.
If a standard CSV file system could be approved, standard writers were possible to prepare and distribute. Until that time each User has to create his own auxiliary functions to work with. The most convenient and straightforward way is to use Octave routines which can be directly started by double-click from schematics.

zoltan.huszka@t-online.hu

Appendix I.
Regression

A1. Derivation of the regression equations

In the characterization practice a large amount of parameters can be extracted - at least partially – by regression. The general solution by matrix-vector technique available e.g. in Matlab and Octave can not be fully realized in QucsStudio. As a minimalist approach the 3-variable linear regression will be discussed and implemented in the tool. This is not a severe restriction because experiments justify that higher order techniques are mostly unstable.
For clarity the Hicum base current will be selected as an example

$$Ib = ibeis\left[\exp\left(\frac{Vbe}{mbei\cdot V_T}\right)-1\right]+ireis\left[\exp\left(\frac{Vbe}{mrei\cdot V_T}\right)-1\right]+ileak \tag{1}$$

Current *ileak* is the possible leakage of the measurement instrument at low biases. Though it is not a model parameter it must be taken into account for not to distort the physical parameter values.
The expression contains three linear parameters *ibeis*, *ireis* and *ileak* which have to be determined so as to provide the best fit to the measured base current. For a given pair of nonlinear parameters *mbei, mrei* the V_{be} dependent terms are known and can be collected in column vectors

$$\mathbf{a}_1(k)=\exp\left(\frac{V_{be}(k)}{mbei\cdot V_T}\right)-1 \quad \mathbf{a}_2(k)=\exp\left(\frac{V_{be}(k)}{mrei\cdot V_T}\right)-1 \quad \mathbf{a}_3(k)=1 \quad \mathbf{a}(k)=I_b(k) \tag{2}$$

With the matrix

$$\mathbf{A}=\left[\mathbf{a}_1 \quad \mathbf{a}_2 \quad \mathbf{a}_3\right] \tag{3}$$

and the vector of unknowns

$$\mathbf{x}=\left[ibeis, ireis, ileak\right]^T \tag{4}$$

the problem to solve reads

$$\mathbf{A}\cdot\mathbf{x}=\mathbf{a} \tag{5}$$

For a given **x** the vector of deviations in the measurement points read

$$\boldsymbol{\delta}=\mathbf{A}\cdot\mathbf{x}-\mathbf{a} \tag{6}$$

We want to minimize the sum of squared deviations

$$S=\boldsymbol{\delta}^T\cdot\boldsymbol{\delta}=(\mathbf{A}\cdot\mathbf{x}-\mathbf{a})^T\cdot(\mathbf{A}\cdot\mathbf{x}-\mathbf{a}) \tag{7}$$

This is minimum when the gradient of *S* w.r.t. **x** is zero

$$\mathbf{grad}(S)=2\mathbf{A}^T\cdot(\mathbf{A}\cdot\mathbf{x}-\mathbf{a})=\mathbf{0} \tag{8}$$

yielding the linear equation

$$(\mathbf{A}^T\cdot\mathbf{A})\cdot\mathbf{x}=\mathbf{A}^T\cdot\mathbf{a} \tag{9}$$

Thus the solution of the least-square problem results in

$$\mathbf{x}=(\mathbf{A}^T\cdot\mathbf{A})^{-1}\cdot\mathbf{A}^T\cdot\mathbf{a} \tag{10}$$

providing the simulated result

$$\mathbf{a}_{sim}=\mathbf{A}\cdot\mathbf{x}=\mathbf{A}\cdot(\mathbf{A}^T\cdot\mathbf{A})^{-1}\cdot\mathbf{A}^T\cdot\mathbf{a} \tag{11}$$

The so called "hat" matrix **H** is specified by

$$\mathbf{H}=\mathbf{A}\cdot(\mathbf{A}^T\cdot\mathbf{A})^{-1}\cdot\mathbf{A}^T \tag{12}$$

yielding

$$\mathbf{a}_{sim}=\mathbf{H}\cdot\mathbf{a} \tag{13}$$

The error vector (6) with the unit matrix **E** of dimension **H** reads

$$\boldsymbol{\delta}=\mathbf{a}_{sim}-\mathbf{a}=\left[\mathbf{H}-\mathbf{E}\right]\cdot\mathbf{a} \tag{14}$$

providing the total (scalar) error (7)

$$S = \mathbf{a}^T \cdot [\mathbf{H} - \mathbf{E}]^T \cdot [\mathbf{H} - \mathbf{E}] \cdot \mathbf{a} \tag{15}$$

The error $S(mbei, mrei)$ depends only on the non-ideality factors which can be minimized by an optimizer in terms of the two parameters. Than, the optimal value of $\mathbf{x}$ is given by (10). Note however that the error vector $\boldsymbol{\delta}$ and the scalar error S do not depend on the solution $\mathbf{x}$.
That makes it clear why it is – generally - not possible to obtain a physical solution to this task by a *five-variable optimization*. The optimizer blindly allocates new values in each cycle not only for *mbei, mrei* but also for *ibeis*, *ireis* and *ileak* too because it does not know anything about the internal link (correlation) among these variables. The process ends up either at a local minimum or stops at a nearby point. It can be concluded that

The result of a (semi) linear regression can not be surpassed by optimization

Hence, often performed post-optimizations using the regression result as initials is not necessary or even destructive.
The absolute deviation (6) can be used when the variation of the data is moderate over the interval. Otherwise a weighting scheme must be applied for allocating prescribed significance to the data points. The weights shall be collected in a diagonal weight matrix $\mathbf{W}$ yielding

$$\boldsymbol{\delta} = \mathbf{W}(\mathbf{A} \cdot \mathbf{x} - \mathbf{a}) \tag{16}$$

Most of the cases *relative weights* are applied. In the present example it is equivalent to taking

$$\mathbf{W} = diag(1/I_b) \tag{17}$$

what means $\mathbf{W}(k,k) = 1/I_b(k)$ $\mathbf{W}(i,k) = 0$ $\mathsf{k} = 1,2,...\mathsf{N}; \mathsf{i} \neq \mathsf{k}$. The deviations in point k become $\delta_k = \dfrac{I_b(k) - I_{bsim}(k)}{I_b(k)}$.
Observing (16) the substitution

$$\begin{aligned} \mathbf{B} &= \mathbf{W} \cdot \mathbf{A} \\ \mathbf{b} &= \mathbf{W} \cdot \mathbf{a} \end{aligned} \tag{18}$$

results in

$$\mathbf{B} \cdot \mathbf{x} = \mathbf{b} \tag{19}$$

which can be solved the same way as in the case of using absolute errors. The regressed result however must be computed by (6) since the weighting has its influence only on the solution vector $\mathbf{x}$.
(9) can be detailed as

$$x_1\mathbf{a}_1^T\mathbf{a}_1 + x_2\mathbf{a}_1^T\mathbf{a}_2 + x_3\mathbf{a}_1^T\mathbf{a}_3 = \mathbf{a}_1^T\mathbf{a}$$
$$x_1\mathbf{a}_2^T\mathbf{a}_1 + x_2\mathbf{a}_2^T\mathbf{a}_2 + x_3\mathbf{a}_2^T\mathbf{a}_3 = \mathbf{a}_2^T\mathbf{a}$$
$$x_1\mathbf{a}_3^T\mathbf{a}_1 + x_2\mathbf{a}_3^T\mathbf{a}_2 + x_3\mathbf{a}_3^T\mathbf{a}_3 = \mathbf{a}_3^T\mathbf{a}$$

The weights are collected in a column vector $\mathbf{w}$. Typically $\mathbf{w} = 1/\mathbf{a}$ for relative weights. By (18)

$$\mathbf{b}_1 = \mathbf{w} \otimes \mathbf{a}_1; \quad \mathbf{b}_2 = \mathbf{w} \otimes \mathbf{a}_2; \quad \mathbf{b}_3 = \mathbf{w} \otimes \mathbf{a}_3; \quad \mathbf{b} = \mathbf{w} \otimes \mathbf{a};$$

where $\otimes$ stands for the element-wise or dot product. The equation to solve becomes

$$x_1\mathbf{b}_1^T\mathbf{b}_1 + x_2\mathbf{b}_1^T\mathbf{b}_2 + x_3\mathbf{b}_1^T\mathbf{b}_3 = \mathbf{b}_1^T\mathbf{b}$$
$$x_1\mathbf{b}_2^T\mathbf{b}_1 + x_2\mathbf{b}_2^T\mathbf{b}_2 + x_3\mathbf{b}_2^T\mathbf{b}_3 = \mathbf{a}_2^T\mathbf{b}$$
$$x_1\mathbf{b}_3^T\mathbf{b}_1 + x_2\mathbf{b}_3^T\mathbf{b}_2 + x_3\mathbf{b}_3^T\mathbf{b}_3 = \mathbf{a}_3^T\mathbf{b}$$

This is of the form

$$\begin{bmatrix} c_{011} & c_{012} & c_{013} \\ c_{021} & c_{022} & c_{023} \\ c_{031} & c_{032} & c_{033} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} d_{01} \\ d_{02} \\ d_{03} \end{bmatrix}$$

The third order linear equation can be solved the easiest way by Gaussian elimination.

Step 1
Make elements in column 1 under the main diagonal zero. This is achieved by deducing *row1* multiplied by factors m_{21} and m_{31} from *row2* and *row3* repectively.

$$m_{21} = c_{021} / c_{011} \quad m_{31} = c_{031} / c_{011}$$
$$c_{122} = c_{022} - m_{21} c_{012} \quad c_{123} = c_{023} - m_{21} c_{013} \quad d_{12} = d_{02} - m_{21} d_{01}$$
$$c_{132} = c_{032} - m_{31} c_{012} \quad c_{133} = c_{033} - m_{31} c_{013} \quad d_{13} = d_{03} - m_{31} d_{01}$$
$$\begin{bmatrix} c_{011} & c_{012} & c_{013} \\ 0 & c_{122} & c_{123} \\ 0 & c_{132} & c_{133} \end{bmatrix} \begin{bmatrix} d_{01} \\ d_{12} \\ d_{13} \end{bmatrix}$$

Step 2
Make the 2nd element of *row3* zero.

$$m_{32} = c_{132} / c_{122}$$
$$c_{233} = c_{133} - m_{32} c_{123} \quad d_{23} = d_{13} - m_{32} d_{12}$$
$$\begin{bmatrix} c_{011} & c_{012} & c_{013} \\ 0 & c_{122} & c_{123} \\ 0 & 0 & c_{233} \end{bmatrix} \begin{bmatrix} d_{01} \\ d_{12} \\ d_{23} \end{bmatrix}$$

The solution results in by back substitution

$$x_3 = \frac{d_{23}}{c_{233}} \quad x_2 = \frac{d_{12} - x_3 c_{123}}{c_{122}} \quad x_1 = \frac{d_{01} - x_3 c_{013} - x_2 c_{012}}{c_{011}}$$

yielding the simulated result

$$\mathbf{a}_{sim} = \mathbf{A} \cdot \mathbf{x} = x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + x_3 \mathbf{a}_3$$

and the deviations and total errors from

$$\mathbf{b}_{sim} = x_1 \mathbf{b}_1 + x_2 \mathbf{b}_2 + x_3 \mathbf{b}_3$$

as

$$\boldsymbol{\delta} = abs(\mathbf{b}_{sim} - \mathbf{b})$$
$$err\_regr = sum(\boldsymbol{\delta} \otimes \boldsymbol{\delta})$$

A2. QucsStudio implementation

The `if` statements in the code are switches to the simpler regression forms.

$a3 = 0$	two-variable regression
$a2 = a3 = 0$	single variable regression

The following script can be directly pasted in a created archive, say `R3.sch` file:

```
<Eqn 3varRegression 1 770 110 -29 16 0 0 "b1=w*a1" 1 "b2=w*a2" 1 "b3=w*a3"
1 "b=w*a" 1 "c011=sum(b1*b1)" 1 "c012=sum(b1*b2)" 1 "c013=sum(b1*b3)" 1
"d01=sum(b1*b)" 1 "c021=c012" 1 "c022=sum(b2*b2)" 1 "c023=sum(b2*b3)" 1
"d02=sum(b2*b)" 1 "c031=c013" 1 "c032=c023" 1 "c033=sum(b3*b3)" 1
"d03=sum(b3*b)" 1 "m21=c021/c011" 1 "c122=c022-m21*c012" 1 "c123=c023-
m21*c013" 1 "d12=d02-m21*d01" 1 "m31=c031/c011" 1 "c132=c032-m31*c012" 1
"c133=c033-m31*c013" 1 "d13=d03-m31*d01" 1
"m32=if((sum(abs(a2))>0),c132/c122,0)" 1 "c233=c133-m32*c123" 1 "d23=d13-
m32*d12" 1 "x3=if((sum(abs(a3))>0),d23/c233,0)" 1
"x2=if((sum(abs(a2))>0),(d12-x3*c123)/c122,0)" 1 "x1=(d01-x2*c012-
x3*c013)/c011" 1 "asim=x1*a1+x2*a2+x3*a3" 1 "bsim=x1*b1+x2*b2+x3*b3" 1
"delt=abs(bsim-b)" 1 "err_abs=sum(delt*delt)" 1 "yes" 0>
```

Its GUI picture shown below can be copied from there to the required schematic.

equation

```
3varRegression
b1=w*a1
b2=w*a2
b3=w*a3
b=w*a
c011=sum(b1*b1)
c012=sum(b1*b2)
c013=sum(b1*b3)
d01=sum(b1*b)
c021=c012
c022=sum(b2*b2)
c023=sum(b2*b3)
d02=sum(b2*b)
c031=c013
c032=c023
c033=sum(b3*b3)
d03=sum(b3*b)
m21=c021/c011
c122=c022-m21*c012
c123=c023-m21*c013
d12=d02-m21*d01
m31=c031/c011
c132=c032-m31*c012
c133=c033-m31*c013
d13=d03-m31*d01
m32=if((sum(abs(a2))>0),c132/c122,0)
c233=c133-m32*c123
d23=d13-m32*d12
x3=if((sum(abs(a3))>0),d23/c233,0)
x2=if((sum(abs(a2))>0),(d12-x3*c123)/c122,0)
x1=(d01-x2*c012-x3*c013)/c011
asim=x1*a1+x2*a2+x3*a3
bsim=x1*b1+x2*b2+x3*b3
delt=abs(bsim-b)
err_regr=sum(delt*delt)
```

A2. Starting and evaluating the result of a regression

The most widely used regressions are performed in terms of two unknown variables t_1 and t_2

$$t_1 \cdot p_1(u,v) + t_2 \cdot p_2(u,v) = p(u,v) \tag{20}$$

with *u, v,...* as independent variables. Very often, this standard equation is reformulated as

$$q_2(u,v) = \frac{p_2(u,v)}{p_1(u,v)}; \quad q(u,v) = \frac{p(u,v)}{p_1(u,v)}$$
$$t_1 + t_2 \cdot q_2(u,v) = q(u,v) \tag{21}$$

which is the familiar *slope-intercept* scheme.
For the derived 3-variable regression (20) is started with the input code block

```
Regression_in
      a1 =p1
      a2 =p2
      a3 =0
      a  =p
      w  =1
```

and the results are obtained by

```
Regression_out
      t1 =x1
      t2 =x2
      psim  = asim
      error = err_regr
```

Scheme (21) is implemented as

```
Regression_in
      a1 =U
      a2 =q2
      a3 =0
      a  =q
      w  = 1
Regression_out
      t1 =x1
      t2 =x2
      psim  = asim
      error = err_regr
```

The unit vector must be of the same length as the other varables, e.g. `U=1+u-u` or `U=u/u`.
Alternatively, the input for (21) can be formulated as for the standard equation (20) but applying a weight function 1/p_1

```
Regression_in
      a1 =p1
      a2 =p2
      a3 =0
      a  =p
      w  =1/p1
```

It is seen that the *slope-intercept* scheme is obtained by an unvoluntary weighting with `1/p1`. The results are different from those of the unweighted case. The scheme overweights the region around the origin and the computation can even stop if `p1` has a zero. Generally it is recommended to use the standard form (20) with the use of a clearly intended weight. Relative errors are assigned to each point with the weight `w =1/p`.
The starting and evaluating blocks are problem specific. The frequently used cases such as *is* or TC extraction blocks are recommended to save in a separate QucsStudio schematic say in `stock.sch`.

Appendix II.
Sweep and Measurement blocks

The `Parameter Sweep` „SW1” and the `equation` „Measurement” blocks on Fig. AII_1 are application specific. They must be „hardware” modified by re-writing the schematic files. Select a project to work in. All schematics and datafiles specified here must be stored therein.

Step 1.

Create a `skeleton.sch` file with the content

```
<QucsStudio Schematic 2.5.7>
<Properties>
  <View=0,-104,970,379,1,0,0>
  <Grid=10,10,1>
  <DataSet=skeleton.dat>
  <DataDisplay=skeleton.dpl>
  <OpenDisplay=1>
  <showFrame=0>
  <FrameText0=Title>
  <FrameText1=Drawn By:>
  <FrameText2=Date:>
  <FrameText3=Revision:>
</Properties>
<Symbol>
</Symbol>
<Components>
  <.SW SW1 1 120 -60 0 61 0 0 "DC1" 1 "Vbe" 1 "list" 1 "5 Ohm" 0 "50 Ohm" 0 "5; 10"
1>
  <Eqn Eqn1 1 370 150 -29 16 0 0 "data=[1; 10]" 1 "yes" 0>
</Components>
<Wires>
</Wires>
<Diagrams>
</Diagrams>
<Paintings>
</Paintings>
```

It looks in the GUI like

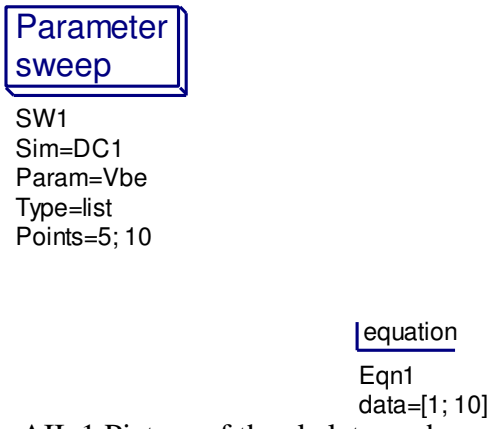


Fig. AII_1 Picture of the skeleton schematic

Step2.

The User is expected to have prepared the data file in `*.csv` format. An example is shown below.

```
"vc";"vb";"ve";"vs";"ic";"ib";"is";"temp";"identv"
 -0.5;0;  0.9;  0.9;  -0.0073428; -0.00019257;0;25;102
 -0.5;0; 0.89; 0.89;  -0.0070648; -0.00018402;0;25;103
 -0.5;0; 0.88; 0.88;  -0.0067911; -0.00017545;0;25;115
 -0.5;0; 0.87; 0.87;  -0.0065211; -0.00016697;0;25; 32
 -0.5;0; 0.86; 0.86;  -0.0062583; -0.00015874;0;25; 32
 -0.5;0; 0.85; 0.85;  -0.0060008;  -0.0001507;0;25; 32
 -0.5;0; 0.84; 0.84;  -0.0057492; -0.00014282;0;25; -3
 -0.5;0; 0.83; 0.83;  -0.0055032; -0.00013516;0;25;191
 -0.5;0; 0.82; 0.82;  -0.0052627;  -0.0001277;0;25;  1
 -0.5;0; 0.81; 0.81;  -0.0050285; -0.00012048;0;25;  5
 -0.5;0;  0.8;  0.8;  -0.0047993; -0.00011346;0;25;  0
 -0.5;0; 0.79; 0.79;  -0.0045761; -0.00010663;0;25;  0
 -0.5;0; 0.78; 0.78;  -0.0043574; -0.00010002;0;25;  0
```

Excerpt A1. CSV data file

The header is written in QucsStudio format so that it can be directly imported by the tool if needed. The voltages are specified by SPICE syntax as node potentials then the currents and temperature follow. The last column is information data. It is optional but helps a lot when matrix reshaping operations are performed. The first six characters are ASCII codes determing the type of the measurement. This case it reads '`fgs  `' that is, forward Gummel with E-S shorted implying zero measured substrate current. Next comes the node the first sweep is adopted to and below, the length of the first sweep follows. The number `3` refers to the `ve` colunm, the negative sign indicates that negative potentials shall be applied `ve=-0.9, -0.89...V` for a positive `vbe=vb-ve=0.9, 0.89,...V`. The following two entries specify the node of the second sweep and its length. In the present case it is `vc=1` or `vcb=vc-vb=-0.5` etc. alltogether 5 different values. The rest of the entries are zero.
The format is suitable to describe multidimensional measurements. The selected one is a single temperature (`temp=25C`) data of dimension `193x5`. At multi-temperature measurements the temperature blocks are sequentially tiled.
The `fillmeas_oct.m` Octave routine shown later converts the CSV data to the one requested by QucsStudio.

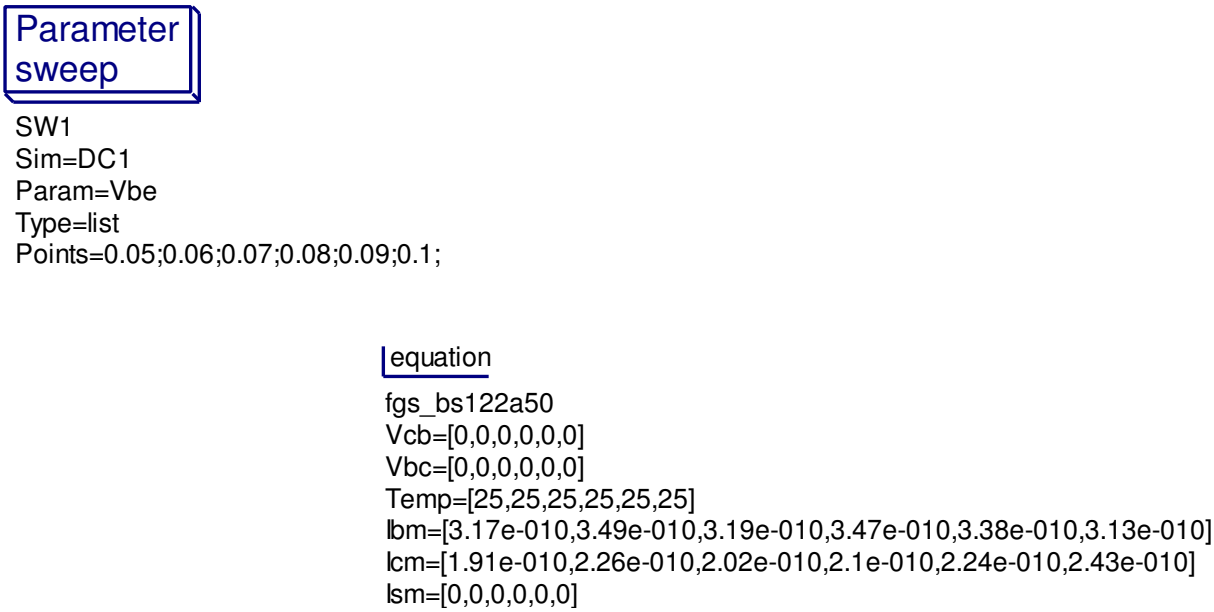


Fig. AII_2 Picture of the generated sweep and measurement data
(strongly truncated for visibility)

The blocks can be copy&pasted to the actual schematic, alternatively they can be directly put in the working scheme. The routine places them to the schematic `filled.sch` by default. The names can be changed at request e.g. `Vbe` to `Vd` in case of capacitance extractions in the routine or on the schematic.
The following *m*-file need to be copied in a selected project. It will appear in the Octave segment of the Content view. Open by double-click and simulate by F2 or by the wheel to perform. Users must adjust the routine to their specific CSV data formats.

```
fillmeas_oct.m
function []=fillmeas_oct();
%
%multisweep range selection
%
%Octave routine
%
% @(#) fillmeas_oct.m  1.0 (C) Z. Huszka 19-Aug-2019
%
%
%Warning: fill in the range and environment specifications next before use!
%
%============== specify ranges ===================
vbe_lo=0.05;
vbe_hi=1;
vcb_lo=0.0;
vcb_hi=0.0;
temp_lo=25;
temp_hi=25;
%============== specify ranges ===================
%=========== specify environment =================
device='bs122a50';
meas_name='fgs';
datafile=[meas_name,'_',device];
qucs_proj='Optim_regr0_prj';
skeleton='skeleton.sch';
target='filled.sch';
delim=';';
%=========== specify environment =================
%
%CSV data with QucsStudio compatible header
%"vc";"vb";"ve";"vs";"ic";"ib";"is";"temp";"identv"';
%  1    2    3    4    5    6    7     8      9

parent=getenv('qucsprj');
dataf=fullfile(parent,qucs_proj,[datafile,'.csv']);
skeleton=fullfile(parent,qucs_proj,skeleton);
target=fullfile(parent,qucs_proj,target);

[vc,vb,ve,vs,ic,ib,is,temp,identv]=...
  textread(dataf,'%f   %f   %f   %f   %f   %f   %f   %f   %f','headerlines',1,'delimiter',delim,'whitespace','
\b\r\n\t=*');

mesh_depth_node=identv(7);
mesh_depth=identv(8);
mesh_width_node=identv(9);
mesh_width=identv(10);
vcmat=reshape(vc,mesh_depth,[]);
vbmat=reshape(vb,mesh_depth,[]);
vemat=reshape(ve,mesh_depth,[]);
vsmat=reshape(vs,mesh_depth,[]);
icmat=reshape(ic,mesh_depth,[]);
ibmat=reshape(ib,mesh_depth,[]);
ismat=reshape(is,mesh_depth,[]);
tempmat=reshape(temp,mesh_depth,[]);
vbemat=vbmat-vemat;
vbe_1D=vbemat(:,1);
vcbmat=vcmat-vbmat;
vcb_1D=vcbmat(1,:);
temp_1D=tempmat(1,:);
ind_vbe=find(vbe_1D>=vbe_lo & vbe_1D<=vbe_hi);
ind_vcb=find(vcb_1D>=vcb_lo & vcb_1D<=vcb_hi);
ind_temp=find(temp_1D>=temp_lo & temp_1D<=temp_hi);
vbe_1Dsel=vbe_1D(ind_vbe);                lenvbev=length(ind_vbe);
vcb_1Dsel=vcb_1D(ind_vcb);                lenvcbv=length(ind_vcb);
temp_1Dsel=temp_1D(ind_temp);             lentempv=length(ind_temp);
ibsel=ibmat(ind_vbe,ind_vcb);             ibsel=ibsel(:);
icsel=icmat(ind_vbe,ind_vcb);             icsel=icsel(:);
issel=ismat(ind_vbe,ind_vcb);             issel=issel(:);
vbe_rect=vbemat(ind_vbe,ind_vcb);         vbeswp=vbe_rect(:);
vcb_rect=vcbmat(ind_vbe,ind_vcb);         vcbswp=vcb_rect(:);
temp_rect=tempmat(ind_vbe,ind_vcb);       tempswp=temp_rect(:);
```

```
%build the single primary sweep
swp1=[];
for k=1:length(vbeswp)
  swp1=[swp1,num2str(vbeswp(k)),';'];
end

%build equation lines for the rest of the control and dependent variables
eq1=[];
eq2=[];
eq3=[];
eq4=[];
eq5=[];
eq6=[];
for k=1:length(vcbswp)
  eq1=[eq1,num2str(vcbswp(k)),','];
  eq2=[eq2,num2str(0-vcbswp(k)),','];
  eq3=[eq3,num2str(tempswp(k)),','];
  eq4=[eq4,num2str(ibsel(k)),','];
  eq5=[eq5,num2str(icsel(k)),','];
  eq6=[eq6,num2str(issel(k)),','];
end

%build datalines in a single equation box
%<Eqn Eqn1 1 870 420 -29 16 0 0 "Vcb=[0,0,...]
termin='"yes" 0>';
eqline=[];
eqline=[eqline,'"Vcb=[',eq1(1:end-1),']" 1 '];
eqline=[eqline,'"Vbc=[',eq2(1:end-1),']" 1 '];
eqline=[eqline,'"Temp=[',eq3(1:end-1),']" 1 '];
eqline=[eqline,'"Ibm=[',eq4(1:end-1),']" 1 '];
eqline=[eqline,'"Icm=[',eq5(1:end-1),']" 1 '];
eqline=[eqline,'"Ism=[',eq6(1:end-1),']" 1 ',termin];

fidr=fopen(skeleton,'r');
fidw=fopen(target,'w');

%cut out and fill <components> section in a cell array
comp='';
sor=fgetl(fidr);
while 1
  if findstr(sor,'<Components>')
    break;
  else
    fprintf(fidw,'%s\n',sor);
  end
  sor=fgetl(fidr);
end
k=1;
while 1
  sor=fgetl(fidr);
  if isempty(findstr(sor,'</Components>'))
    comp{k}=sor;
    k=k+1;
  else
    break;
  end
end

%write rest of file as it is
while 1
  sor=fgetl(fidr);
  if feof(fidr)
    if ischar(sor)
      fprintf(fidw,'%s\n',sor);
    end
    break
  else
    fprintf(fidw,'%s\n',sor);
  end
end

%modify content with confined data
for k=1:length(comp)
  str=comp{k};
  if strfind(str,'.SW SW1')
    found=findstr(str,'SW1');
    str(found+4)='1'; %activate block
    %str=strrep(str,'indep','Vbe');
    found=findstr(str,'"');
    comp{k}=[str(1:found(end-1)),swp1,str(found(end):end)];
  elseif strfind(str,'Eqn Eqn1')
    found=findstr(str,'Eqn1');
    str(found+5)='1'; %activate block
    str=strrep(str,'Eqn1',datafile);
    found1=findstr(str,'"');
    comp{k}=[str(1:found1(1)-1),eqline];
  end
end
```

```
%write data to end of .sch
fprintf(fidw,'%s\n','<Components>');
for k=1:length(comp)
  str=comp{k};
  fprintf(fidw,'%s\n',str);
end
fprintf(fidw,'%s\n','</Components>');

fclose(fidr);
fclose(fidw);

return
```

Appendix III.
Verilog-A code of the Hicum collector capacitance

hicjq.va

```
/*12-Aug-2019
Return capacitance data using original Hicum macros
taken from hicumL2v2.40
@(#) hicjq.va 1.0 (C) Z. Huszka 12-Aug-2019
*/

`ifdef insideADMS
`define ATTR(txt) (*txt*)
`else
`define ATTR(txt)
`endif
`define DFa_fj 1.921812
`define VPT_thresh 1.0e2
`define Cexp_lim 80.0

`include "disciplines.vams"
`include "constants.vams"

// DEPLETION CHARGE CALCULATION
// Hyperbolic smoothing used; no punch-through
// INPUT:
// c_0 : zero-bias capacitance
// u_d : built-in voltage
// z : exponent coefficient
// a_j : control parameter for C peak value at high forward bias
// U_cap : voltage across junction
// IMPLICIT INPUT:
// VT : thermal voltage
// OUTPUT:
// Qz : depletion Charge
// C : depletion capacitance
`define QJMODF(c_0,u_d,z,a_j,U_cap,C,Qz)\
if(c_0 > 0.0) begin\
DFV_f = u_d*(1.0-exp(-ln(a_j)/z));\
DFv_e = (DFV_f-U_cap)/VT;\
DFs_q = sqrt(DFv_e*DFv_e+`DFa_fj);\
DFs_q2 = (DFv_e+DFs_q)*0.5;\
DFv_j = DFV_f-VT*DFs_q2;\
DFdvj_dv = DFs_q2/DFs_q;\
DFb = ln(1.0-DFv_j/u_d);\
DFC_j1 = c_0*exp(-z*DFb)*DFdvj_dv;\
C = DFC_j1+a_j*c_0*(1.0-DFdvj_dv);\
DFQ_j = c_0*u_d*(1.0-exp(DFb*(1.0-z)))/(1.0-z);\
Qz = DFQ_j+a_j*c_0*(U_cap-DFv_j);\
end else begin\
C = 0.0;\
Qz = 0.0;\
end

// DEPLETION CHARGE CALCULATION CONSIDERING PUNCH THROUGH
// smoothing of reverse bias region (punch-through)
// and limiting to a_j=Cj,max/Cj0 for forward bias.
// Important for base-collector and collector-substrate junction
// INPUT:
// c_0 : zero-bias capacitance
// u_d : built-in voltage
// z : exponent coefficient
// a_j : control parameter for C peak value at high forward bias
// v_pt : punch-through voltage (defined as qNw^2/2e)
// U_cap : voltage across junction
// IMPLICIT INPUT:
// VT : thermal voltage
// OUTPUT:
// Qz : depletion charge
// C : depletion capacitance
`define QJMOD(c_0,u_d,z,a_j,v_pt,U_cap,C,Qz)\
if(c_0 > 0.0) begin\
Dz_r = z/4.0;\
Dv_p = v_pt-u_d;\
DV_f = u_d*(1.0-exp(-ln(a_j)/z));\
DC_max = a_j*c_0;\
DC_c = c_0*exp((Dz_r-z)*ln(v_pt/u_d));\
Dv_e = (DV_f-U_cap)/VT;\
if(Dv_e < `Cexp_lim) begin\
De = exp(Dv_e);\
De_1 = De/(1.0+De);\
Dv_j1 = DV_f-VT*ln(1.0+De);\
end else begin\
De_1 = 1.0;\
Dv_j1 = U_cap;\
end\
```

```
Da = 0.1*Dv_p+4.0*VT;\
Dv_r = (Dv_p+Dv_j1)/Da;\
if(Dv_r < `Cexp_lim) begin\
De = exp(Dv_r);\
De_2 = De/(1.0+De);\
Dv_j2 = -Dv_p+Da*(ln(1.0+De)-exp(-(Dv_p+DV_f)/Da));\
end else begin\
De_2 = 1.0;\
Dv_j2 = Dv_j1;\
end\
Dv_j4 = U_cap-Dv_j1;\
DCln1 = ln(1.0-Dv_j1/u_d);\
DCln2 = ln(1.0-Dv_j2/u_d);\
Dz1 = 1.0-z;\
Dzr1 = 1.0-Dz_r;\
DC_j1 = c_0*exp(DCln2*(-z))*De_1*De_2;\
DC_j2 = DC_c*exp(DCln1*(-Dz_r))*(1.0-De_2);\
DC_j3 = DC_max*(1.0-De_1);\
C = DC_j1+DC_j2+DC_j3;\
DQ_j1 = c_0*(1.0-exp(DCln2*Dz1))/Dz1;\
DQ_j2 = DC_c*(1.0-exp(DCln1*Dzr1))/Dzr1;\
DQ_j3 = DC_c*(1.0-exp(DCln2*Dzr1))/Dzr1;\
Qz = (DQ_j1+DQ_j2-DQ_j3)*u_d+DC_max*Dv_j4;\
end else begin\
C = 0.0;\
Qz = 0.0;\
end
// DEPLETION CHARGE & CAPACITANCE CALCULATION SELECTOR
// Dependent on junction punch-through voltage
// Important for collector related junctions
`define HICJQ(c_0,u_d,z,v_pt,U_cap,C,Qz)\
if(v_pt < `VPT_thresh) begin\
`QJMOD(c_0,u_d,z,2.4,v_pt,U_cap,C,Qz)\
end else begin\
`QJMODF(c_0,u_d,z,2.4,U_cap,C,Qz)\
end

module hicjq(vin,cout);

//Node definitions
input vin;
output cout;
voltage vin,cout; //voltage is a signal flow

//branch definitions
branch (vin ) br_vin;
branch (cout ) br_c;

parameter real cjci0 = 4.77E-15 from [0:inf) `ATTR(info="Internal B-C zero-bias depletion capacitance"
unit="F");
parameter real vdci = 0.615 from (0:10] `ATTR(info="Internal B-C built-in potential" unit="V");
parameter real zci = 0.291 from (0:1) `ATTR(info="Internal B-C grading coefficient");
parameter real vptci = 5 from (0:100] `ATTR(info="Internal B-C punch-through voltage" unit="V");
`ifdef insideADMS
parameter real temp = 2.5000e+01 `ATTR(info="Circuit (ambient) temperature" unit="C");
`endif

real DFV_f,DFv_e,DFv_j,DFb,DFQ_j,DFs_q,DFs_q2,DFdvj_dv,DFC_j1;//QJMODF
real
Dz_r,Dv_p,DV_f,DC_max,DC_c,Da,Dv_e,De,De_1,Dv_j1,Dv_r,De_2,Dv_j2,Dv_j4,DQ_j1,DQ_j2,DQ_j3,DCln1,DCln2,Dz1,Dz
r1,DC_j1,DC_j2,DC_j3;//QJMOD
real Tamb,Tdev,VT,Vin,Cj,Qj;

analog begin

begin : Model_evaluation

`ifdef insideADMS
Tamb = temp+`P_CELSIUS0;
`else
Tamb = $temperature;
`endif
Tdev = Tamb;
VT = `P_K*Tdev /`P_Q;
Vin = V(br_vin);

`HICJQ(cjci0,vdci,zci,vptci,Vin,Cj,Qj)

end //of Model_evaluation
V(br_c) <+ Cj;

end //analog
endmodule
```

Thank you!

Please visit our website
www.ams.com