Interactive Numerical and Symbolic Analysis: A New Paradigm for Teaching Electronics

Jean-Claude Thomassian
SUNY Maritime College/Engineering, Throggs Neck, New York

***Abstract*—Analog Insydes, Mathematica's symbolic circuit analysis toolbox, uses modern algorithms of expression simplification depending on comparisons with a numerical reference solution of the circuit under investigation. Some insight is offered on how the complexity of an expression barrier is overcome followed by two classical examples, a BJT emitter follower and a MOSFET common-gate amplifier stage to illustrate the proposed method at work. A concluding section discusses that time spent teaching introductory electronics by computer-aided circuit analysis, interactive numerical and symbolic, is a worthwhile investment.**

***Index Terms*—Analog Insydes, computer algebra, electrical engineering education, expression simplification, symbolic techniques.**

I. INTRODUCTION

As is well known, too little time is made available in modern four year electrical and computer engineering curricula to teach introductory electronics in the traditional manner. One way to improve the outcome is to use computer-aided techniques including the newly developed symbolic circuit simulators such as Analog Insydes 2.1 [1].

The computer revolution has acquired two complementary but distinct faces: pure numerical solution techniques and symbolic solution techniques, about which more shortly. Before attempting to discuss how the computer is best to be employed in our endeavor, let us digress briefly to discuss a philosophical but relevant matter. When a competent, experienced engineer engaged in design, optimization, revision, or whatever considers a circuit or system, we often say that the engineer understands the entity in question. What exactly do we mean by the term "understands?" No answer could be totally unequivocal, but we might be able to agree with the following: he understands, at least qualitatively, a significant portion of the cause and effect relationships that exist within the circuit or system, and he understands in workable detail the analyses or computer simulations that could be and would need be done to provide quantitative answers to any questions that could reasonably be asked about the circuit or system in question. If we further agree that our educational objectives are . . . to give the student some level of understanding via simple examples of cause and effect . . . to survey and codify the questions, it is possible and useful to ask . . . and to teach how the answers to the questions can be obtained via computer, then we can explore the question . . . how the computer can be most expediently employed.

We are proposing to begin using the computer immediately to solve problems yes, both symbolically and numerically; however, we are also proposing to begin immediately teaching the student how to use both the computer and selected software to solve the underlying mathematical problem. As example: a mature user of SPICE [2], say, readily understands the mathematical problem he is posing to the computer software combo even though he may have little or no understanding of the actual algorithmic processes involved in obtaining the solution. We are proposing that students be taught that the analysis of electric circuits can be codified. When confronted by a specific example, the student must be able to recognize the essential type of underlying mathematical problem. In the particular arena of circuits, it is not difficult to do because there are only a few types and they are readily recognizable for what they are. Next, the student must be able to formulate his example in a manner acceptable to the hardware software combo used; this will invariably require system-specific formatting but the process is essentially generic. The basic type of mathematical problem being solved may well include adjustable parameters, variable ranges, or other user specified inputs. The student must understand the mathematical nature of his example well enough to guarantee that the information he is seeking will be included in the output from the computer. The final step, by no means trivial, requires that the student be able to recognize and interpret the information produced.

Analog Insydes simplifies expressions the same way a human analyst would. It selectively restricts the frequency range of interest to that specified by one or more design points and discards relatively insignificant terms of the expression as determined by an allowable error bound and the reference solution. The program was developed primarily as a serious tool for professional analog designers; however it seemed to us to have a strong pedagogic potential as well [3], [4].

II. SYMBOLIC SYMPLIFICATION METHODOLOGY

It has long been a goal of researchers to develop methods of symbolic analysis of electric circuits. Historically, two major problems have blocked progress. Symbolic circuit analysis, as one major application of computer algebra, has labored under traditional shortcomings: it required, relatively speaking, large amounts of computer resources, memory and CPU time. These requirements are largely met by modern machines, but a problem of an essentially mathematical nature also needed a solution. Formal solutions of even simple circuits yield expressions of enormous complexity—so complex as to effectively defy human understanding and thereby render them useless. The number of terms in a formal solution tends to increase near exponentially as a function of the number of components in the circuit. This effect is sometimes referred to as the "complexity of expression barrier." Over the past several years, researchers have developed at least two essentially different approaches to simplification routines. The first developed has been named Simplification After Term Generation (SAG) [5], [6]; the second is named Simplification Before Term Generation (SBG) [7], [8].

SAG or solution-based approximation starts by calculating the complete symbolic expression followed by a simplification step where product terms in the coefficients of the numerator and denominator polynomials of the transfer function are deleted. This approach can stop being useful when the number of terms is large because it alone cannot reduce the expression to a comprehensible level of complexity. The newest approaches to simplification, simplification during and before generation were developed about the same time [9], [10]. The rationale underlying SBG or equation-based approximation is to emulate the technique of a seasoned circuit analyst who makes approximations even as he is describing the circuit; e.g., if he is interested only in low frequency behavior where small capacitors have no effect, he will simply leave them out. It may also turn out that the computational burden of solving an approximated system of equations is dramatically lower than solving the full system of non-approximated equations followed by an SAG routine. The next two subsections describe the implementation of these routines as built into Analog Insydes.

A. *Simplification Before Generation (SBG)*

ApproximateMatrixEquation[dae, var, errspec] approximates a symbolic matrix equation, dae, with respect to a designated output variable, var, while observing the error specification, errspec. The above operation approximates a linear symbolic matrix equation by discarding insignificant matrix entries before the system is solved (SBG). The significance of a symbolic matrix element is determined by calculating its numerical influence on the magnitude of the transfer function at one or several frequency sampling points. The approximation process generates a term ranking obtained by computing the large-change sensitivities of the output variable with respect to all matrix entries and sorting the resulting list in order of decreasing influence. The algorithm deletes all possible matrix entries subject to the constraint that their cumulative influence is less than the user-specified error bound. For a given errspec, the algorithm assures that the relative deviation of the magnitude of the output variable, var, measured at the frequency values fvar=freqi is less than maxerri.

B. *Simplification After Generation (SAG)*

ApproximateTransferFunction[expr, fvar, dp, error] approximates a symbolic transfer function, expr, by discarding insignificant terms where fvar, dp, and error denote the complex frequency variable [i.e. the Laplace variable s], the design point [here understood to mean the numerical values of the components and device parameters appearing in the reference solution], and the bound on the coefficient error. The foregoing operation implements SAG. The significance or insignificance of a term is assessed on the basis of numerical reference values for the symbols, i.e., the design point. For each coefficient, the algorithm removes the numerically least significant terms until the maximum coefficient error is reached. The fourth argument, noted above, denotes the maximum coefficient error; it does not constitute a bound on the absolute error of the transfer function. The absolute error may be larger or smaller than the maximum coefficient error. This operation does not discriminate frequency, and the result is valid for whatever frequency range was used in the reference solution.

III. INTERACTIVE NUMERICAL AND SYMBOLIC SIMULATION EXAMPLES

In the examples to follow, two very simple and very well-known one-transistor circuits are presented; a BJT emitter follower (EF) and a MOSFET common-gate (CG) amplifier stage. The examples presented employ PSpice and Analog Insydes as a toolbox of Mathematica. Tight interaction between symbolic and numeric computations helps to ensure continuous error control and verification of the results.

A. *Common Collector Example*

We will analyze the circuit of Fig. 1, which is a common collector (emitter follower EF) amplifier. The thing to note is that a load, if present, would be connected to the emitter. The collector is connected to VCC, and the output is in phase with the input. The EF amp has a voltage gain close to one. With this kind of gain, we may ask what is the point? The point is that an EF has a very low output resistance and a relatively high input resistance. It is therefore useful as a buffer placed between a high resistance source and a low resistance load, thereby providing the necessary current to the load without undue voltage drop.

The DC bias for a stand alone EF amp is similar to the CE amp, but since the gain is near one, we would establish a quiescent or DC bias point around VB = VCC/2, VC = VCC and VE = VB-0.7 V. The 0.7 V reflects a diode drop across the base-emitter junction.

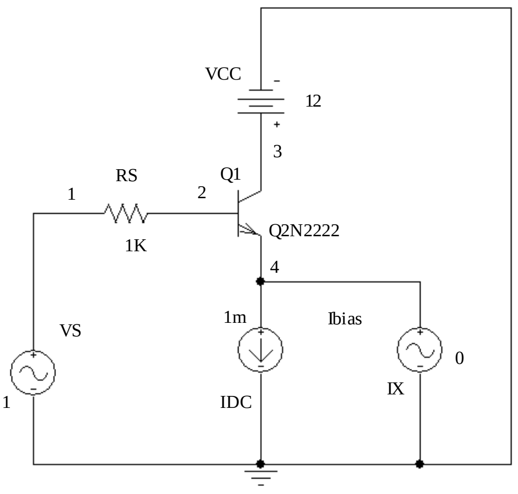
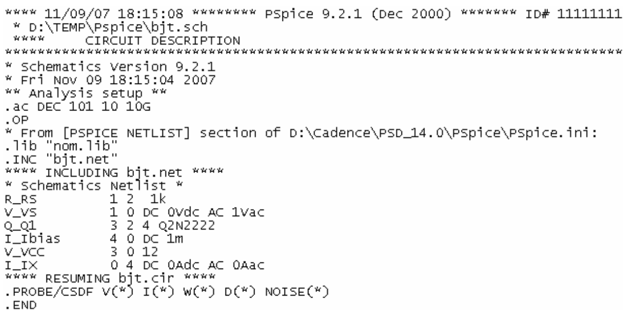


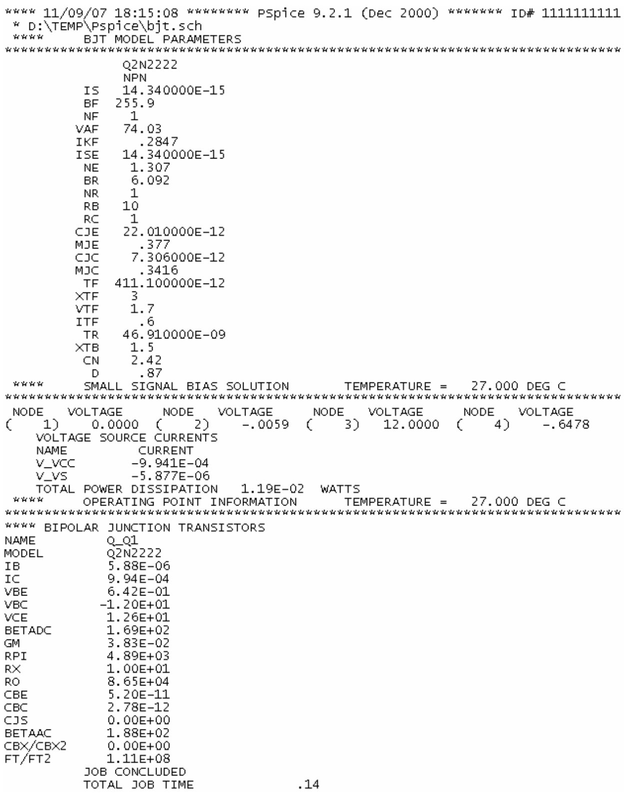
Figure 1. Emitter follower amplifier schematic – no load.

Figures 2 and 3 are what the PSpice input and output files look like for the simulation of the emitter follower amplifier we are going to examine



```
**** 11/09/07 18:15:08 ******** PSpice 9.2.1 (Dec 2000) ******* ID# 11111111
 * D:\TEMP\Pspice\bjt.sch
 ****     CIRCUIT DESCRIPTION
******************************************************************************
* Schematics Version 9.2.1
* Fri Nov 09 18:15:04 2007
** Analysis setup **
.ac DEC 101 10 10G
.OP
* From [PSPICE NETLIST] section of D:\Cadence\PSD_14.0\PSpice\PSpice.ini:
.lib "nom.lib"
.INC "bjt.net"
**** INCLUDING bjt.net ****
* Schematics Netlist *
R_RS          1 2  1k
V_VS          1 0 DC 0Vdc AC 1Vac
Q_Q1          3 2 4 Q2N2222
I_Ibias       4 0 DC 1m
V_VCC         3 0 12
I_IX          0 4 DC 0Adc AC 0Aac
**** RESUMING bjt.cir ****
.PROBE/CSDF V(*) I(*) W(*) D(*) NOISE(*)
.END
```

Figure 2. PSpice emitter follower amplifier input file.



```
**** 11/09/07 18:15:08 ******** PSpice 9.2.1 (Dec 2000) ******* ID# 1111111111
 * D:\TEMP\Pspice\bjt.sch
 ****     BJT MODEL PARAMETERS
******************************************************************************
               Q2N2222
               NPN
          IS   14.340000E-15
          BF   255.9
          NF    1
         VAF   74.03
         IKF     .2847
         ISE   14.340000E-15
          NE    1.307
          BR    6.092
          NR    1
          RB   10
          RC    1
         CJE   22.010000E-12
         MJE     .377
         CJC    7.306000E-12
         MJC     .3416
          TF  411.100000E-12
         XTF    3
         VTF    1.7
         ITF     .6
          TR   46.910000E-09
         XTB    1.5
          CN    2.42
           D     .87
 ****     SMALL SIGNAL BIAS SOLUTION       TEMPERATURE =   27.000 DEG C
******************************************************************************
 NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE
(    1)    0.0000  (    2)    -.0059  (    3)   12.0000  (    4)    -.6478
    VOLTAGE SOURCE CURRENTS
    NAME         CURRENT
    V_VCC       -9.941E-04
    V_VS        -5.877E-06
    TOTAL POWER DISSIPATION   1.19E-02  WATTS
 ****     OPERATING POINT INFORMATION      TEMPERATURE =   27.000 DEG C
******************************************************************************
**** BIPOLAR JUNCTION TRANSISTORS
NAME          Q_Q1
MODEL         Q2N2222
IB            5.88E-06
IC            9.94E-04
VBE           6.42E-01
VBC          -1.20E+01
VCE           1.26E+01
BETADC        1.69E+02
GM            3.83E-02
RPI           4.89E+03
RX            1.00E+01
RO            8.65E+04
CBE           5.20E-11
CBC           2.78E-12
CJS           0.00E+00
BETAAC        1.88E+02
CBX/CBX2      0.00E+00
FT/FT2        1.11E+08
          JOB CONCLUDED
          TOTAL JOB TIME            .14
```

Figure 3. PSpice emitter follower amplifier output file.

Figure 4 is a Probe plot for the simulation results of the gain magnitude and phase. From the plot we can read the low frequency gain to be approximately 1V/V and the cut off frequency to be about 2 MHz.

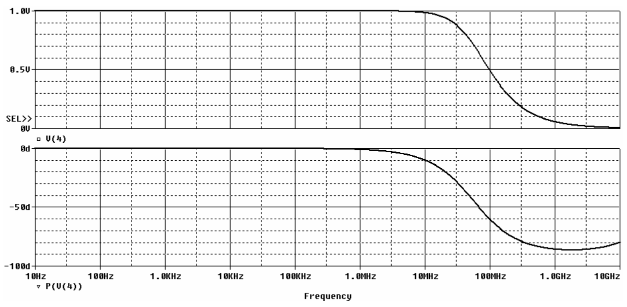
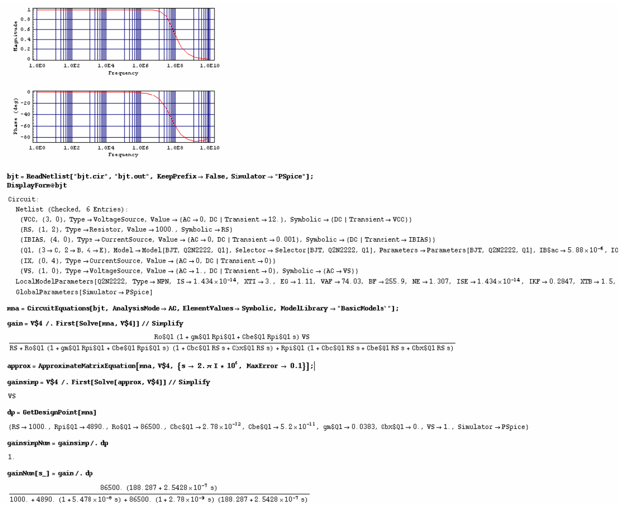


Figure 4. Emitter follower amplifier gain magnitude and phase.

Figure 5, is the Analog Insydes simulation result of analyzing the emitter follower circuit under investigation. Included in the figure are the emitter follower gain magnitude and phase angle, the imported PSpice netlist, and the symbolic expression for the transfer function in fully symbolic form and using numbers in the low frequency approximation. The numerical expression calculated uses both the design point information and the component values specified.

Here we have solved the modified nodal analysis matrix for the output voltage at node V$4 using the basic BJT model, which produced the fully symbolic result for gain as shown in the figure. Then we have applied a simplification before generation command ApproximateMatrixEquation and obtained an approximate symbolic result of gainsimp which in our case turns out to be VS (gain = 1).

If we plot gainNum[s_] and gainsimpNum and compare to the SPICE Bode plot we will find that gainNum[s_] constitutes a global description of the frequency response of the amplifier and closely approximates the original PSpice plot while gainsimpNum is just the SBG solution and is valid only in the low to midband region and does not reflect the behavior the amplifier exhibits at high frequencies. Note that such a local approximation was precisely what we asked for, because we specified only one reference point on the frequency axis at 1 MHz.

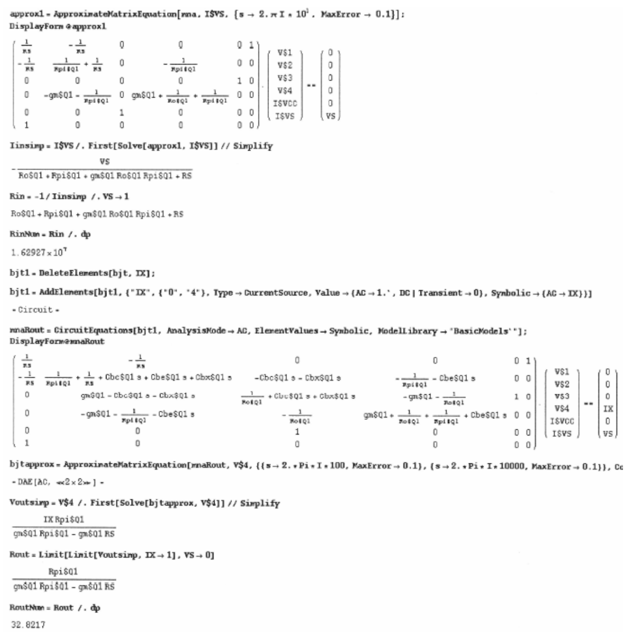


```
bjt = ReadNetlist["bjt.cir", "bjt.out", KeepPrefix → False, Simulator → "PSpice"];
DisplayForm@bjt

Circuit:
  Netlist (Checked, 6 Entries):
   {VCC, {3, 0}, Type → VoltageSource, Value → {AC → 0, DC | Transient → 12.}, Symbolic → {DC | Transient → VCC}}
   {RS, {1, 2}, Type → Resistor, Value → 1000., Symbolic → RS}
   {IBIAS, {4, 0}, Type → CurrentSource, Value → {AC → 0, DC | Transient → 0.001}, Symbolic → {DC | Transient → IBIAS}}
   {Q1, {3 → C, 2 → B, 4 → E}, Model → Model[BJT, Q2N2222, Q1], Selector → Selector[BJT, Q2N2222, Q1], Parameters → Parameters[BJT, Q2N2222, Q1], IB$ac → 5.88×10^-6, IC
   {IX, {0, 4}, Type → CurrentSource, Value → {AC → 0, DC | Transient → 0}}
   {VS, {1, 0}, Type → VoltageSource, Value → {AC → 1., DC | Transient → 0}, Symbolic → {AC → VS}}
  LocalModelParameters[Q2N2222, Type → NPN, IS → 1.434×10^-14, XTI → 3., EG → 1.11, VAF → 74.03, BF → 255.9, NE → 1.307, ISE → 1.434×10^-14, IKF → 0.2847, XTB → 1.5,
  GlobalParameters[Simulator → PSpice]

mna = CircuitEquations[bjt, AnalysisMode → AC, ElementValues → Symbolic, ModelLibrary → "BasicModels`"];

gain = V$4 /. First[Solve[mna, V$4]] // Simplify

                           Ro$Q1 (1 + gm$Q1 Rpi$Q1 + Cbe$Q1 Rpi$Q1 s) VS
------------------------------------------------------------------------------------------------------------------------
RS + Ro$Q1 (1 + gm$Q1 Rpi$Q1 + Cbe$Q1 Rpi$Q1 s) (1 + Cbc$Q1 RS s + Cbx$Q1 RS s) + Rpi$Q1 (1 + Cbc$Q1 RS s + Cbe$Q1 RS s + Cbx$Q1 RS s)

approx = ApproximateMatrixEquation[mna, V$4, {s → 2. π I * 10^6, MaxError → 0.1}];

gainsimp = V$4 /. First[Solve[approx, V$4]] // Simplify

VS

dp = GetDesignPoint[mna]

{RS → 1000., Rpi$Q1 → 4890., Ro$Q1 → 86500., Cbc$Q1 → 2.78×10^-13, Cbe$Q1 → 5.2×10^-11, gm$Q1 → 0.0383, Cbx$Q1 → 0., VS → 1., Simulator → PSpice}

gainsimpNum = gainsimp /. dp

1.

gainNum[s_] = gain /. dp

                     86500. (188.287 + 2.5428×10^-7 s)
---------------------------------------------------------------------------------
1000. + 4890. (1 + 5.478×10^-8 s) + 86500. (1 + 2.78×10^-9 s) (188.287 + 2.5428×10^-7 s)
```

Figure 5. Analog Insydes Emitter Follower amplifier Gain.

It may also be noted that by setting VS to 1, we can avoid computing the gain as (V$4/V$1); instead it is given by solving for V$4 only.

To extend the analysis, Fig. 6 shows how we calculate the input and output resistance for this simple emitter follower circuit. Input resistance is calculated by taking the input voltage divided by input current. The output resistance is calculated by adding a current source IX at the output, node 4 in our case, and then the output resistance is obtained as the voltage at node 4 by setting VS=0 and IX=1 as shown in the figure. The numerically approximated results agree closely with PSpice probe results.



```
approx1 = ApproximateMatrixEquation[mna, I$VS, {s → 2. π I * 10^3, MaxError → 0.1}];
DisplayForm@approx1
```

$$\begin{pmatrix} \frac{1}{RS} & -\frac{1}{RS} & 0 & 0 & 0 & 1 \\ -\frac{1}{RS} & \frac{1}{Rpi\$Q1}+\frac{1}{RS} & 0 & -\frac{1}{Rpi\$Q1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -gm\$Q1-\frac{1}{Rpi\$Q1} & 0 & gm\$Q1+\frac{1}{Ro\$Q1}+\frac{1}{Rpi\$Q1} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} V\$1 \\ V\$2 \\ V\$3 \\ V\$4 \\ I\$VCC \\ I\$VS \end{pmatrix} == \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ VS \end{pmatrix}$$

```
Iinsimp = I$VS /. First[Solve[approx1, I$VS]] // Simplify
```

$$-\frac{VS}{Ro\$Q1+Rpi\$Q1+gm\$Q1\,Ro\$Q1\,Rpi\$Q1+RS}$$

```
Rin = -1/Iinsimp /. VS → 1
```

$$Ro\$Q1+Rpi\$Q1+gm\$Q1\,Ro\$Q1\,Rpi\$Q1+RS$$

```
RinNum = Rin /. dp
```

$$1.62927\times10^{7}$$

```
bjt1 = DeleteElements[bjt, IX];
bjt1 = AddElements[bjt1, {"IX", {"0", "4"}, Type → CurrentSource, Value → {AC → 1.`, DC | Transient → 0}, Symbolic → {AC → IX}}]
- Circuit -
mnaRout = CircuitEquations[bjt1, AnalysisMode → AC, ElementValues → Symbolic, ModelLibrary → "BasicModels`"];
DisplayForm@mnaRout
```

$$\begin{pmatrix} \frac{1}{RS} & -\frac{1}{RS} & 0 & 0 & 0 & 1 \\ -\frac{1}{RS} & \frac{1}{Rpi\$Q1}+\frac{1}{RS}+Cbc\$Q1\,s+Cbe\$Q1\,s+Cbx\$Q1\,s & -Cbc\$Q1\,s-Cbx\$Q1\,s & -\frac{1}{Rpi\$Q1}-Cbe\$Q1\,s & 0 & 0 \\ 0 & gm\$Q1-Cbc\$Q1\,s-Cbx\$Q1\,s & \frac{1}{Ro\$Q1}+Cbc\$Q1\,s+Cbx\$Q1\,s & -gm\$Q1-\frac{1}{Ro\$Q1} & 1 & 0 \\ 0 & -gm\$Q1-\frac{1}{Rpi\$Q1}-Cbe\$Q1\,s & -\frac{1}{Ro\$Q1} & gm\$Q1+\frac{1}{Ro\$Q1}+\frac{1}{Rpi\$Q1}+Cbe\$Q1\,s & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} V\$1 \\ V\$2 \\ V\$3 \\ V\$4 \\ I\$VCC \\ I\$VS \end{pmatrix} == \begin{pmatrix} 0 \\ 0 \\ 0 \\ IX \\ 0 \\ VS \end{pmatrix}$$

```
bjtapprox = ApproximateMatrixEquation[mnaRout, V$4, {{s → 2. * Pi * I * 100, MaxError → 0.1}, {s → 2. * Pi * I * 10000, MaxError → 0.1}}, Co
- DAE[AC, <<2 × 2>>] -
Voutsimp = V$4 /. First[Solve[bjtapprox, V$4]] // Simplify
```

$$\frac{IX\,Rpi\$Q1}{gm\$Q1\,Rpi\$Q1-gm\$Q1\,RS}$$

```
Rout = Limit[Limit[Voutsimp, IX → 1], VS → 0]
```

$$\frac{Rpi\$Q1}{gm\$Q1\,Rpi\$Q1-gm\$Q1\,RS}$$

```
RoutNum = Rout /. dp
```

$$32.8217$$

Figure 6. Analog Insydes emitter follower input and output resistances.

B. *Common Gate Example*

In this subsection we will analyze the common gate (CG) amplifier circuit of Fig. 7 where the gate is grounded, input is applied at the source, and the output is taken from the drain. The Q-point of the transistor is set by DC biasing. It is obvious that the current gain for this amplifier must be unity, since the gate current for a MOSFET is zero. The CG is a good current buffer; it takes a current at the input that may come from a source with a relatively small Norton resistance and replicates it at the output port, which approximates a strong source due to the relatively high output resistance.

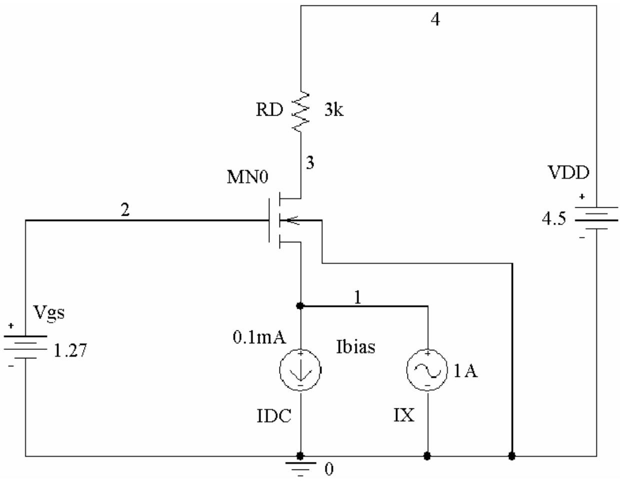
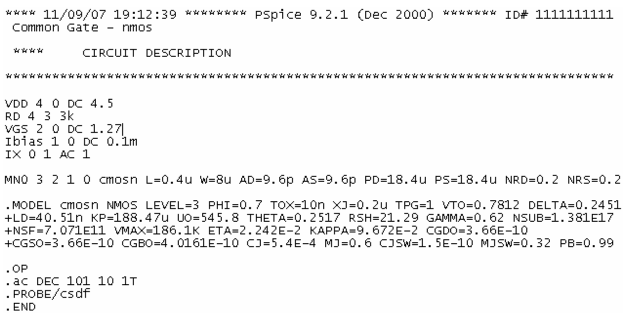


Figure 7. Common Gate "CG" amplifier.

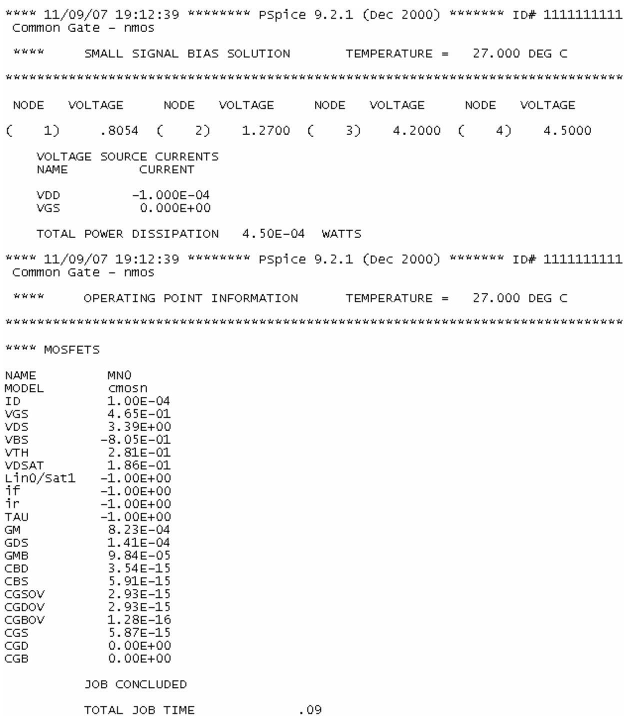
Figures 8 and 9 are the PSpice input and output files for the simulation of the common gate amplifier we are examining.



```
**** 11/09/07 19:12:39 ******** PSpice 9.2.1 (Dec 2000) ******* ID# 1111111111
 Common Gate - nmos

 ****     CIRCUIT DESCRIPTION

******************************************************************************

VDD 4 0 DC 4.5
RD 4 3 3k
VGS 2 0 DC 1.27|
Ibias 1 0 DC 0.1m
IX 0 1 AC 1

MN0 3 2 1 0 cmosn L=0.4u W=8u AD=9.6p AS=9.6p PD=18.4u PS=18.4u NRD=0.2 NRS=0.2

.MODEL cmosn NMOS LEVEL=3 PHI=0.7 TOX=10n XJ=0.2u TPG=1 VTO=0.7812 DELTA=0.2451
+LD=40.51n KP=188.47u UO=545.8 THETA=0.2517 RSH=21.29 GAMMA=0.62 NSUB=1.381E17
+NSF=7.071E11 VMAX=186.1K ETA=2.242E-2 KAPPA=9.672E-2 CGDO=3.66E-10
+CGSO=3.66E-10 CGBO=4.0161E-10 CJ=5.4E-4 MJ=0.6 CJSW=1.5E-10 MJSW=0.32 PB=0.99

.OP
.ac DEC 101 10 1T
.PROBE/csdf
.END
```

Figure 8. PSpice common gate amplifier input file.

Here we have used a Level 3 model for a 0.4 um technology to analyze this circuit, we have biased the NMOS to produce a drain current of 0.1 mA as shown in the following figure for the operating point information.



```
**** 11/09/07 19:12:39 ******** PSpice 9.2.1 (Dec 2000) ******* ID# 1111111111
 Common Gate - nmos

 ****     SMALL SIGNAL BIAS SOLUTION       TEMPERATURE =   27.000 DEG C

******************************************************************************

 NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE

(    1)     .8054  (    2)    1.2700  (    3)    4.2000  (    4)    4.5000

    VOLTAGE SOURCE CURRENTS
    NAME         CURRENT

    VDD         -1.000E-04
    VGS          0.000E+00

    TOTAL POWER DISSIPATION   4.50E-04  WATTS
**** 11/09/07 19:12:39 ******** PSpice 9.2.1 (Dec 2000) ******* ID# 1111111111
 Common Gate - nmos

 ****     OPERATING POINT INFORMATION      TEMPERATURE =   27.000 DEG C

******************************************************************************

**** MOSFETS

NAME         MN0
MODEL        cmosn
ID           1.00E-04
VGS          4.65E-01
VDS          3.39E+00
VBS         -8.05E-01
VTH          2.81E-01
VDSAT        1.86E-01
Lin0/Sat1   -1.00E+00
if          -1.00E+00
ir          -1.00E+00
TAU         -1.00E+00
GM           8.23E-04
GDS          1.41E-04
GMB          9.84E-05
CBD          3.54E-15
CBS          5.91E-15
CGSOV        2.93E-15
CGDOV        2.93E-15
CGBOV        1.28E-16
CGS          5.87E-15
CGD          0.00E+00
CGB          0.00E+00

          JOB CONCLUDED

          TOTAL JOB TIME            .09
```

Figure 9. PSpice common gate amplifier output file.

Figure 10 is the Probe plots for the gain magnitude and phase of the common gate; here the gain is calculated to be 2.232 V/V. We are using an NMOS with a width of 8 um, applied Vgs of 1.27 V, and a supply voltage of 4.5 V.

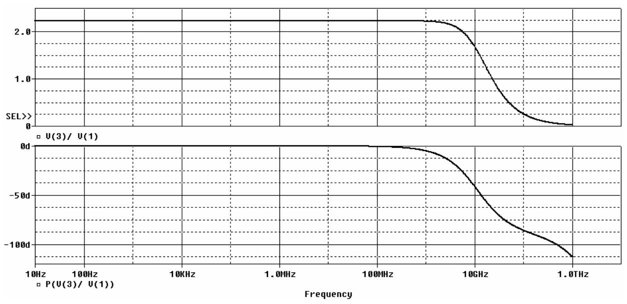
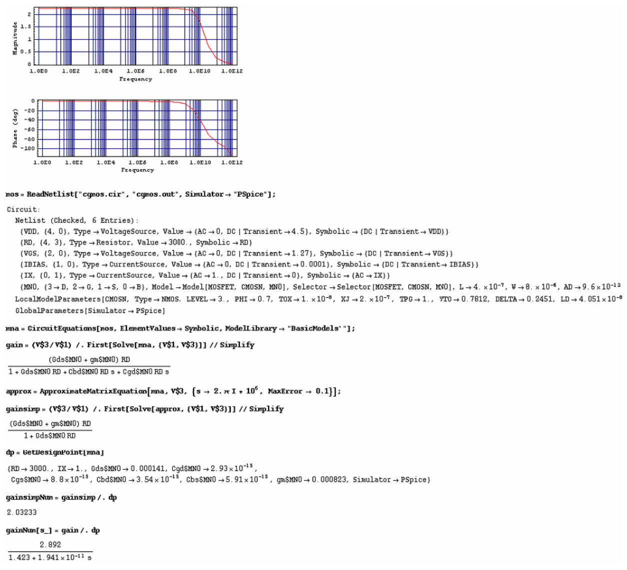


Figure 10. Common gate amplifier gain magnitude and phase.

The following Fig. 11 is the Analog Insydes simulation result excerpted from analyzing the common gate circuit under investigation. Included in the figure are the common gate gain magnitude and phase angle, the imported Pspice netlist, and the expressions for the transfer function in fully symbolic form and in the low frequency numerical approximation. The numerical expression calculated uses the design point information and component values given.

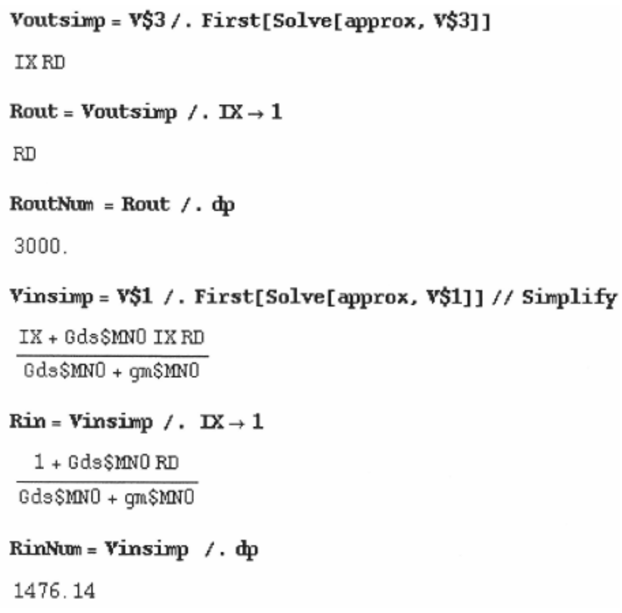
Here we have solved the modified nodal analysis matrix for the gain using the basic MOS model, which produced the fully symbolic result for gain as shown in the figure. Then we have applied a simplification before generation command ApproximateMatrixEquation and obtained an approximate symbolic result of gainsimp. The command ApproximateMatrixEquation is restricted to control only a single output variable. To control several variables, we need to repeat the approximation. When computing a small-signal voltage gain, we usually apply a constant voltage source to the input. The value of the input variable is a constant. Thus, in our application, it is sufficient to carry out the approximation for the output voltage only and then solve the equations as set forth in the figure below.



```
mos = ReadNetlist["cgmos.cir", "cgmos.out", Simulator → "PSpice"];

Circuit:
  Netlist (Checked, 6 Entries):
   {VDD, {4, 0}, Type → VoltageSource, Value → {AC → 0, DC | Transient → 4.5}, Symbolic → {DC | Transient → VDD}}
   {RD, {4, 3}, Type → Resistor, Value → 3000., Symbolic → RD}
   {VGS, {2, 0}, Type → VoltageSource, Value → {AC → 0, DC | Transient → 1.27}, Symbolic → {DC | Transient → VGS}}
   {IBIAS, {1, 0}, Type → CurrentSource, Value → {AC → 0, DC | Transient → 0.0001}, Symbolic → {DC | Transient → IBIAS}}
   {IX, {0, 1}, Type → CurrentSource, Value → {AC → 1., DC | Transient → 0}, Symbolic → {AC → IX}}
   {MN0, {3 → D, 2 → G, 1 → S, 0 → B}, Model → Model[MOSFET, CMOSN, MN0], Selector → Selector[MOSFET, CMOSN, MN0], L → 4. × 10^-7, W → 8. × 10^-6, AD → 9.6 × 10^-12
  LocalModelParameters[CMOSN, Type → NMOS. LEVEL → 3., PHI → 0.7, TOX → 1. × 10^-8, XJ → 2. × 10^-7, TPG → 1., VTO → 0.7812, DELTA → 0.2451, LD → 4.051 × 10^-8
  GlobalParameters[Simulator → PSpice]

mna = CircuitEquations[mos, ElementValues → Symbolic, ModelLibrary → "BasicModels`"];

gain = (V$3 / V$1) /. First[Solve[mna, {V$1, V$3}]] // Simplify

            (Gds$MN0 + gm$MN0) RD
 ------------------------------------------
 1 + Gds$MN0 RD + Cbd$MN0 RD s + Cgd$MN0 RD s

approx = ApproximateMatrixEquation[mna, V$3, {s → 2. π I * 10^6, MaxError → 0.1}];

gainsimp = (V$3 / V$1) /. First[Solve[approx, {V$1, V$3}]] // Simplify

 (Gds$MN0 + gm$MN0) RD
 ---------------------
    1 + Gds$MN0 RD

dp = GetDesignPoint[mna]

{RD → 3000., IX → 1., Gds$MN0 → 0.000141, Cgd$MN0 → 2.93 × 10^-15,
 Cgs$MN0 → 8.8 × 10^-15, Cbd$MN0 → 3.54 × 10^-15, Cbs$MN0 → 5.91 × 10^-15, gm$MN0 → 0.000823, Simulator → PSpice}

gainsimpNum = gainsimp /. dp

2.03233

gainNum[s_] = gain /. dp

        2.892
 ----------------------
 1.423 + 1.941 × 10^-11 s
```

Figure 11. Analog Insydes common gate amplifier output file.

The next excerpt shows how Analog Insydes calculated the input and output resistances. Again, the numerical values match well with PSpice values when compared. It is shown that the Thevenin resistance at node 3 is just RD as follows from the ideal current source drive.



```
Voutsimp = V$3 /. First[Solve[approx, V$3]]

IX RD

Rout = Voutsimp /. IX → 1

RD

RoutNum = Rout /. dp

3000.

Vinsimp = V$1 /. First[Solve[approx, V$1]] // Simplify

 IX + Gds$MN0 IX RD
 ------------------
 Gds$MN0 + gm$MN0

Rin = Vinsimp /. IX → 1

  1 + Gds$MN0 RD
 ----------------
 Gds$MN0 + gm$MN0

RinNum = Vinsimp /. dp

1476.14
```

Figure 12. Analog Insydes common gate amplifier input / output resistances.

IV. CONCLUSIONS

Symbolic methods or computer algebra has been around for many years; only recently, however, have powerful simplification algorithms been developed and implemented. This improvement has increased the value of symbolic methods enormously both for pedagogic and for serious design purposes. An example is that now it is possible to identify specific small-signal parameters of specific transistors as important contributors to dominant system poles and zeroes.

There is no going back to a simpler time. Modern problems are complex and demand computer simulation or computer-assisted analysis, synthesis, or optimization. Nothing is gained and much can be lost by postponing the introduction of computer methods or by introducing them in a partial, half hearted, adjunctive way. This is often done in the name of 'understanding.' We propose that if effectively instructed, a beginning student can 'understand' the analysis of a 50-node circuit just as readily as he can 'understand' that of a one node circuit.

By 'understand' we imply some knowledge of cause and effect, or as it is sometimes put, "simple formulas are the salt of understanding." Thus, traditionally, students are introduced to both classical problem formulations and simple solution techniques, but Hand analysis quickly runs up against the complexity barrier–one cannot use pencil and paper techniques to solve practical scale problems.

From Pspice output file there is no relevant information produced to give us any insight into the functional dependencies between the circuit elements and the output behavior. Symbolic analysis and computer algebra methods constitute a natural approach towards capturing the expertise of experienced designers. Symbolic analysis is a formal technique for calculating the behavior of circuits as closed-form mathematical expressions. The formulae provide insight into the functional dependencies between the circuit elements and the output behavior; for example, the voltage at a node is not just a number.

In our opinion, it would be better to support instruction in electronics on computer-aided circuit analysis for the same reason that a modern carpenter prefers to use electric skill saws and pneumatic nail drivers.

What are the challenges to the education community? Time, money and breaking the grip of habit and tradition. We look forward to learning about some real-world experiments, either as participants or as interested onlookers.

REFERENCES

[1] Analog Insydes, Intelligent Symbolic Design System, by Fraunhofer ITWM, Online at: http://www.analog-insydes.de/

[2] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits." Tech. Rep. UCB/ERL M520, University of California, Berkeley, 1975.

[3] J.-C. Thomassian and E. D. Smith, Interactive Numerical and Symbolic Simulation: A New Paradigm for Teaching Circuits, Proceedings of IEEE 34th Frontiers in Education Conference, pp. S2C 8-12, October 2004.

[4] J.-C. Thomassian, Symbolic Techniques: A New Tool for Teaching Circuits and Electronics, Proceedings of IEEE 37th Frontiers in Education Conference, pp. S2C 1-5, October 2007.

[5] H. Walscharts, G. Gielen and W. Sansen, "Symbolic Simulation of Analog Circuits in S- and Z-Domain," *IEEE ISCAS*, pp. 814-817, 1989.

[6] F. V. Fernandez, A. Rodriguez-Vazquez and J. L. Huertas, "A Tool for Symbolic Analysis of Analog Integrated Circuits Including Pole/Zero Extraction," in Proc. 10th ECCTD, pp. 752-761, Copenhagen (Denmark), July, 1991.

[7] J. J. Hsu and C. Sechen, "The Sifting Approach to Symbolic Analysis of Large Analog Integrated Circuits," in *Proc. 3rd Int. Workshop on Symbolic Methods and Applications in Circuit Design (SMACD)*, pp. 212-230, Sevilla (Spain), Oct. 1994.

[8] J. D. Rodriguez-Garcia, O. Guerra, E. Roca, F. V. Fernandez and A. Rodriguez-Vazquez, "A New Simplification Before and During Generation Algorithm," in *Proc. 5th Int. Workshop SMACD*, pp. 110-124, Kaiserslautern (Germany), Oct. 1998.

[9] E. Hennig, Symbolic Approximation and Modeling Techniques for Analysis and Design of Analog Circuits, Shaker Verlag, Aachen, Germany, 2000.

[10] W. Daems, G. Gielen and W. Sansen, "Circuit Simplification for Symbolic Analysis of Analog Integrated Circuits, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 21, no. 4, pp. 395-407, April 2002.

AUTHOR

J.-C. Thomassian is with the Department of Engineering, SUNY Maritime College, Throggs Neck, NY 10465 USA (e-mail: jthomassian@ieee.org)

Manuscript received 11 March 2008. Published as submitted by the author.