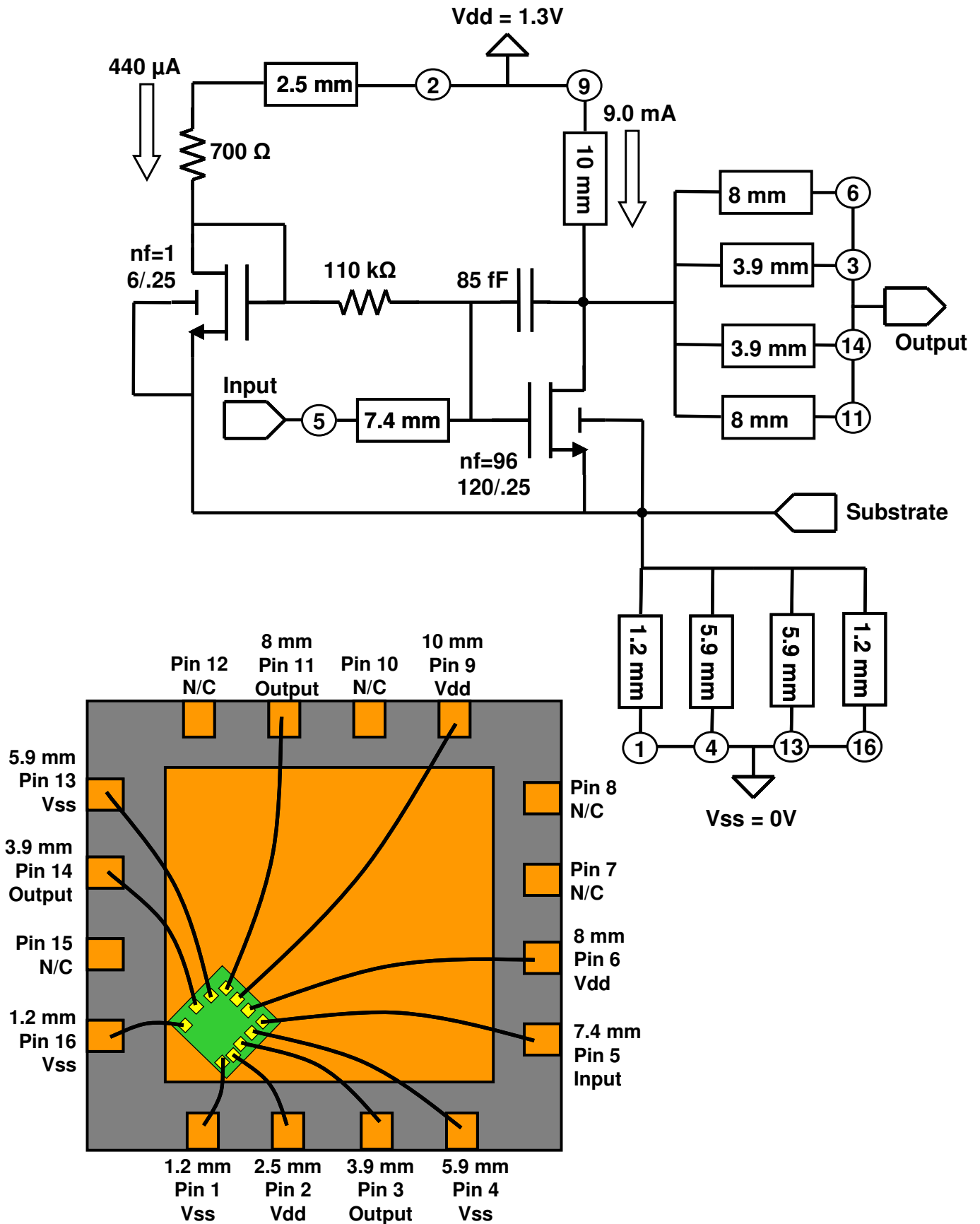


	Requirement	Our design
Power	< 13 mW	12.3 mW
Vdd	≤ 2.5 V	1.3 V
Noise Figure @ 2GHz	minimize	.70 dB
S11 @ 2GHz	< -10 dB	-18.5 dB
S21 @ 2GHz	> 10 dB	11.5 dB
IIP3 @ 2GHz	> -5 dBm	21.7 dBm
Number of Pins	≤ 12	11
On-chip Capacitance	≤ 200 pF	85 fF
On-chip Resistance	≤ 120 kΩ	110.7 kΩ
Off-chip components	≤ 1	0



```

.subckt lna in out vdd vss

*** Input pin ***
Xp5  in  ng1  sub      pin  np=1  nb=1  lb=7.4m

*** Main transistor ***
XM1  nd1  ng1  sub sub  nnmos  w=120u  l=0.25u  nf=96

*** Extra Cgd capacitance ***
XCgd  ng1  nd1  sub      mncap          cm=85f

*** Substrate connection pins (Ls) ***
Xp1  vss  sub  sub      pin  np=1  nb=1  lb=1.2m
Xp4  vss  sub  sub      pin  np=1  nb=1  lb=5.9m
Xp13 vss  sub  sub      pin  np=1  nb=1  lb=5.9m
Xp16 vss  sub  sub      pin  np=1  nb=1  lb=1.2m

*** Output pins ***
Xp6  out  nd1  sub      pin  np=1  nb=1  lb=8m
Xp3  out  nd1  sub      pin  np=1  nb=1  lb=3.9m
Xp14 out  nd1  sub      pin  np=1  nb=1  lb=3.9m
Xp11 out  nd1  sub      pin  np=1  nb=1  lb=8m

*** Drain inductance pin ***
Xp9  vdd  nd1  sub      pin  np=1  nb=1  lb=10m

*** Bias network ***
XM2  ng2  ng2  sub sub  nnmos  w=6u    l=0.25u
Rbs1  nbs  ng2          700
Rbs2  ng2  ng1          110k
Xp2  vdd  nbs  sub      pin  np=1  nb=1  lb=2.5m

.ends lna

```

The design of a high performance LNA is a challenging endeavor, even for experienced engineers. Despite the luxury of several unrealistic assumptions that experienced designers may not enjoy, our “simplified” design nonetheless proved to be a formidable challenge. From reading the course textbook, one may be led to believe that a straightforward, prescribed LNA design recipe could yield a good design, perhaps only one or two hspice-tweaks away from optimal. We were not so lucky with the recipe, but in analyzing the discrepancies, we were able to develop better hand calculation models. Of even greater value, we developed an excellent automated strategy that allowed us to achieve sophisticated optimizations of the design and to gain powerful insights into its complicated nature. The result is a solid design in which we have a high degree of confidence, even over component variations.

Initially, we tried to faithfully stick to the recipe. We chose a cascode architecture so that output loading would not be a problem (mostly to bolster the recipe’s assumption that ignoring C_{gd} is okay). We even cleverly incorporated g_m/I_d -based techniques into the recipe in order to achieve reliable estimates for device width and capacitance. The original script is shown in Figure 1 and is worthy of a few comments. Assuming an I_d that is fixed by power constraints, f_T (and thus C_{gs}) is determined by g_m/I_d . This makes Q a simple function of g_m/I_d . Therefore, we can tweak g_m in order to obtain an acceptable Q . Yet even this simple script reveals an immediate difficulty. Already L_s is smaller than we can achieve in this package, and Q is slightly too low. To increase Q , we must increase f_T , which requires that we lower L_s even further in order to keep a 50 Ohm input. This specific tradeoff would persist throughout the design process. Initially, however, we had much bigger problems.

Our first simulations looked nothing at all like we predicted. Instead of a manageable second order system, we found multiple resonances trampling our passband. Why? The circuit is surrounded by leads (read: inductors), all in the range of a few nH. Concurrently, the parasitic capacitances of the transistors (from any terminal) are on the order of a few hundred fF. This unfortunate state of affairs leads to resonances of comparable frequency to our desired passband. Furthermore, attempts to find each of these resonances and move them proved extremely difficult. We immediately scaled back to a much simpler circuit.

Our new design was not cascoded. This decision reflects the fact that the 50 Ohm load we must drive is actually similar to the $1/g_m$ (~30 Ohm) load provided by the cascode. We also modified the recipe to take C_{gd} into account explicitly in the hand calculations. The lecture notes suggest replacing C_{gd} by an equivalent capacitance to ground using Miller’s theorem, but we obtained much more accurate predictions by analyzing it in the context of a feedback system. A series of circuit transformations, as shown in Figure 2, yields a simple feedback system to which we can apply Blackman’s Impedance formula. It is composed of our familiar model for input impedance (denoted Z_g), which is modified by the feedback action of C_{gd} . Figure 3 shows the final circuit under analysis, and Figure 4 shows a full derivation of the formulas.

Using this strategy, we were able to do a much better job of predicting the input impedance of the circuit. The primary result is that feedback reduces the input impedance by $[1 + \text{Loop Gain}]$, roughly a 3X drop in our case. For reference, Figure 5 plots the real part of Z_{in} using our final circuit parameters. Three approaches are compared: A) ignoring C_{gd} as prescribed in the recipe, B) replacing C_{gd} with an equivalent Miller impedance as described in the textbook, and C) using Blackman’s Impedance formula. Comparing this to Figure 6, which is the actual simulated circuit performance, we see clearly that feedback analysis does the best job of accommodating C_{gd} .

This analysis also provided us with an opportunity to further improve noise by adding an explicit C_{gd} capacitor to the design. In reality, there is no fundamental reason this should decrease noise figure. However, there is a secondary reason, which is linked to this equation for noise figure.

$$F_{\min} := 10 \cdot \log \left[1 + \frac{2}{\sqrt{5}} \cdot \frac{\omega_0}{2 \cdot \pi f_T} \cdot \sqrt{\gamma \cdot \delta \cdot (1 - c^2)} \right]$$

One way to decrease the minimum possible noise figure is to increase f_T , though we know that we cannot do this without decreasing L_s (otherwise, we will not have an input match). Here, C_{gd} can help because it decreases the input impedance without affecting f_T (the formula for F_{\min} is defined to be exact specifically when f_T excludes C_{gd} , so we aren't just fooling ourselves). The result is that we can use a larger f_T as long as we also increase C_{gd} enough to maintain an input match. In this way, C_{gd} provides a new degree of freedom with which to tweak the circuit that can be set independently of transistor sizing. In our final design, we achieved a minimum NF of 0.70dB, which is very close to the minimum of 0.78dB that is predicted by this equation in Figure 7. In fairness, we must also mention that extra feedback decreases gain, which in a real design would place greater noise burden on downstream stages – a sacrifice probably not worth the incremental noise figure improvement. However “maximizing gain” was not a constraint that we were given, so we were happy with this tradeoff. Figure 10 demonstrates the noise improvement we obtained by adding an explicit C_{gd} capacitor, as well as other tradeoffs that result, such as decreased gain.

In order to manage these various degrees of freedom, we employed a systematic method for sweeping virtually any parameter in simulation. By incorporating MATLAB into our design flow (see Appendix B for MATLAB and hspice scripts), we could cycle through thousands of parameter combinations and compare the results of multiple hspice simulations on the same plots. With this automated procedure, we even found it fairly simple to sweep multiple parameters according to complex relationships. After the initial time investment required to write the scripts, our test cycle time was reduced to ~15 seconds to compare any 5 sets of parameter combinations. This tiny loop time was essential for optimizing our circuit's noise figure, which could not be accurately predicted with our hand calculations.

An especially interesting trend that we identified using this method is the fact that performance is only a very weak function of supply voltage. Figure 8 shows the effect of sweeping the supply voltage while keeping *power* constant. We did choose a low V_{dd} due to a slight noise improvement, but if system constraints mandated a higher supply voltage, this figure suggests that there would be very little penalty. In such a situation, this result obviates the need to create a special supply voltage for the LNA, which is certainly a valuable thing to know.

In conclusion, we achieved a good design, but much of the credit for this goes to the strength of our automated procedure. Figure 9 shows the final performance including resistor variations. In truth, the final results we obtained are only distantly correlated with the original design recipe we were given. Discrepancies were laboriously resolved, with great difficulty and yet with limited scope. I believe this is the difficulty of working near the device limits. Because they operate in the “every parasitic matters” regime, RF circuits are inherently complex, inherently high order, and therefore, inherently difficult to analyze by hand. With proper assumptions, some hand analysis can be effective within a very narrow range of operation, but powerful simulation tools are essential for final optimization. In this regime, even hspice results are not above speculation and one must always understand the human limitations to its effectiveness. In the end, only operational silicon is proof that you have a working design.

Appendix A: Figures and Equations

$I_d := 10 \text{ mA}$	$g_m := 30 \frac{\text{mA}}{\text{V}}$	
$f_T := \text{nft} (0.25 \mu\text{m}, g_m I_d)$		
$C_{gs} := \frac{g_m}{f_T \cdot 2 \cdot \pi}$		$C_{gs} = \blacksquare \text{ pF}$
$Q := \frac{1}{\omega_0 \cdot C_{gs} \cdot 2 \cdot R_s}$		$Q = \blacksquare$
$L_s := \frac{50 \Omega}{2 \cdot \pi \cdot f_T}$		$L_s = \blacksquare \text{ nH}$
$Z_g(f) := \frac{1}{j \cdot 2 \cdot \pi f \cdot C_{gs}} + j \cdot 2 \cdot \pi f \cdot L_s + 2 \cdot \pi f \cdot L_s$		
$L_g := \frac{1}{\omega_0^2} \left(\frac{1}{C_{gs}} \right) - L_s$		$L_g = \blacksquare \text{ nH}$
$\text{idw}_m := \text{nidw} (L_{m1}, g_m I_d)$	$W_m := \frac{I_d}{\text{idw}_m}$	$W_m = \blacksquare \mu\text{m}$

Figure 1, original design script

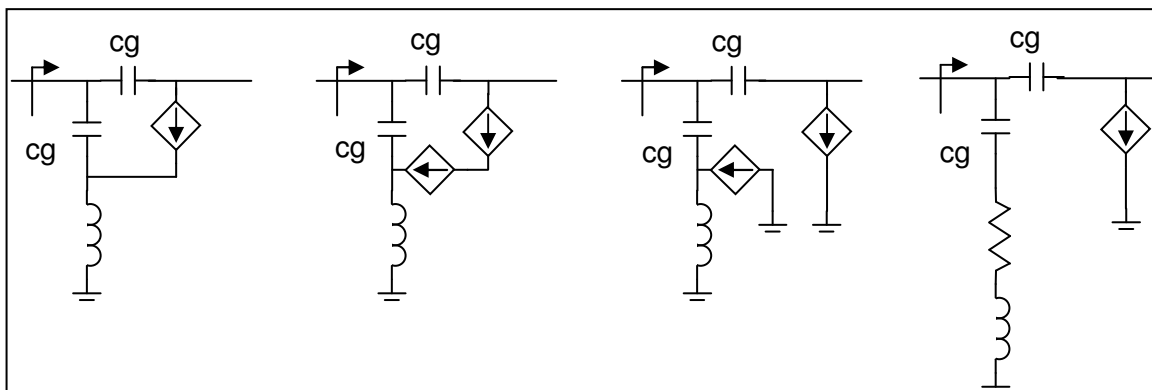


Figure 2, circuit transformation sequence to enable input impedance calculations

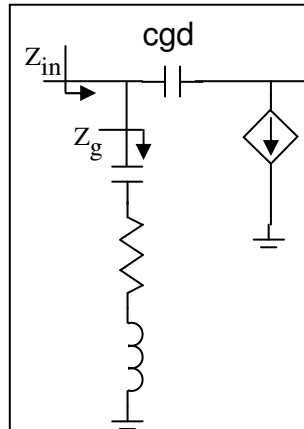


Figure 3, transformed circuit for applying feedback analysis

$$Z_g(f) := \frac{1}{s(f) \cdot C_{gs_eff}} + s(f) \cdot L_s + \omega_T \cdot L_s$$

$$Z_{inOL}(f) := \left[Z_g(f)^{-1} + \left(\frac{1}{s(f) \cdot C_{gd_eff}} + Z_{LR}(f) \right)^{-1} \right]^{-1}$$

$$\beta(f) := \frac{\frac{1}{s(f) \cdot C_{gs_eff}}}{\frac{1}{s(f) \cdot C_{gd_eff}} + Z_g(f)}$$

$$Z_L(f) := \left[\frac{1}{Z_{LR}(f)} + \left(\frac{1}{s(f) \cdot C_{gd_eff}} + Z_g(f) \right)^{-1} \right]^{-1}$$

$$T(f) := g_m \cdot Z_L(f) \cdot \beta(f)$$

$$Z_{inFB}(f) := \frac{Z_{inOL}(f)}{1 + T(f)}$$

Figure 4, input analysis using Blackman's Impedance formula

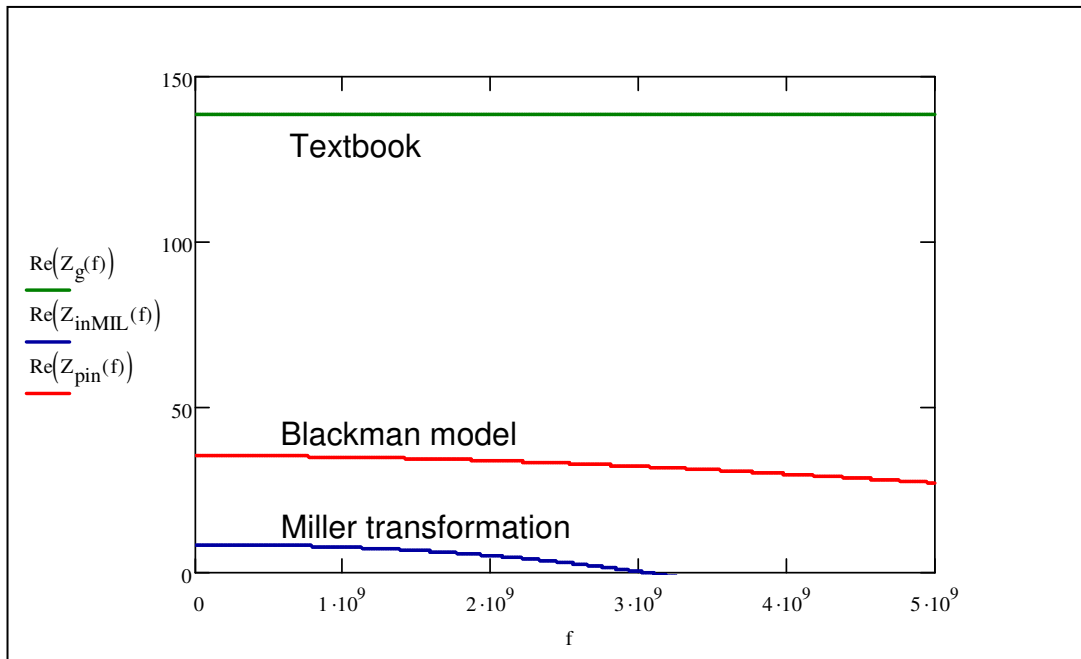


Figure 5, comparison of three different input impedance models

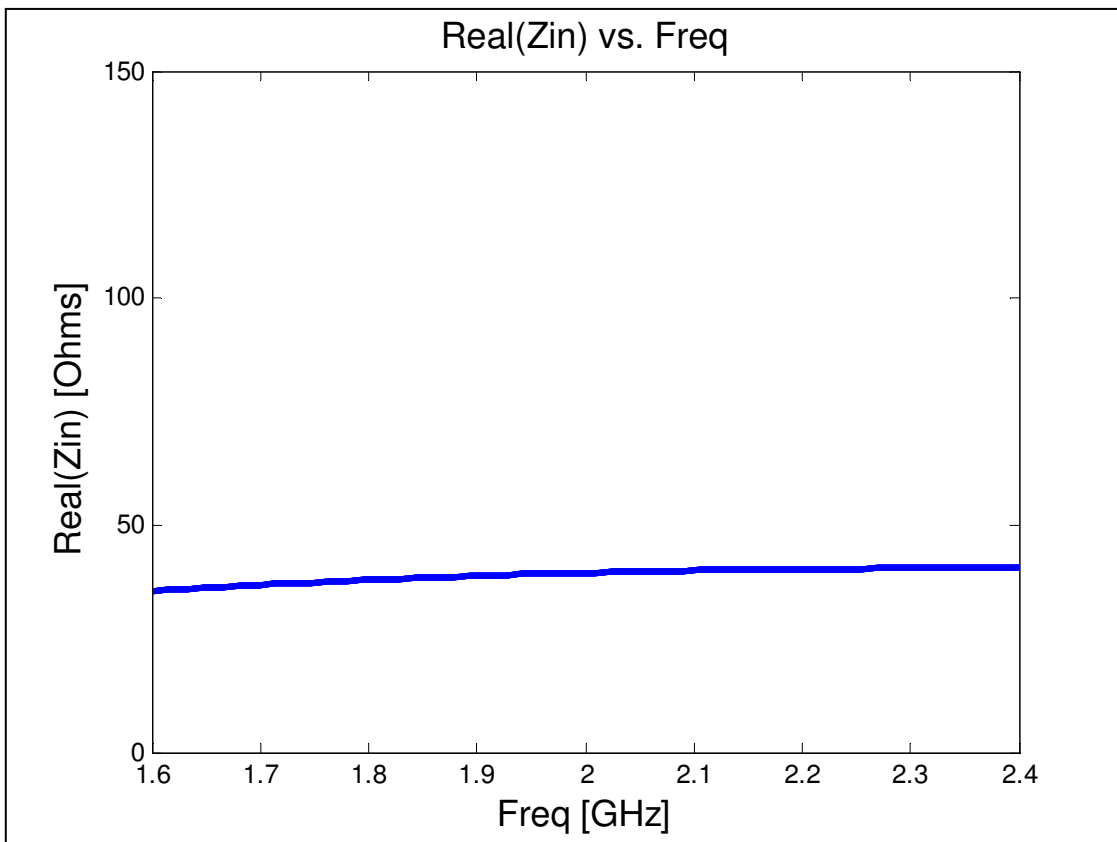


Figure 6, simulated input impedance

$$\begin{array}{l}
 \alpha := 1 \qquad \gamma := 1.2 \cdot \frac{2}{3} \qquad \delta := 4 \qquad c := 0 \\
 F_{\min} := 10 \cdot \log \left[1 + \frac{2}{\sqrt{5}} \cdot \frac{\omega_0}{2 \cdot \pi f_T} \cdot \sqrt{\gamma \cdot \delta \cdot (1 - c^2)} \right] \qquad F_{\min} = \blacksquare
 \end{array}$$

Figure 7, derivation of theoretical minimum noise

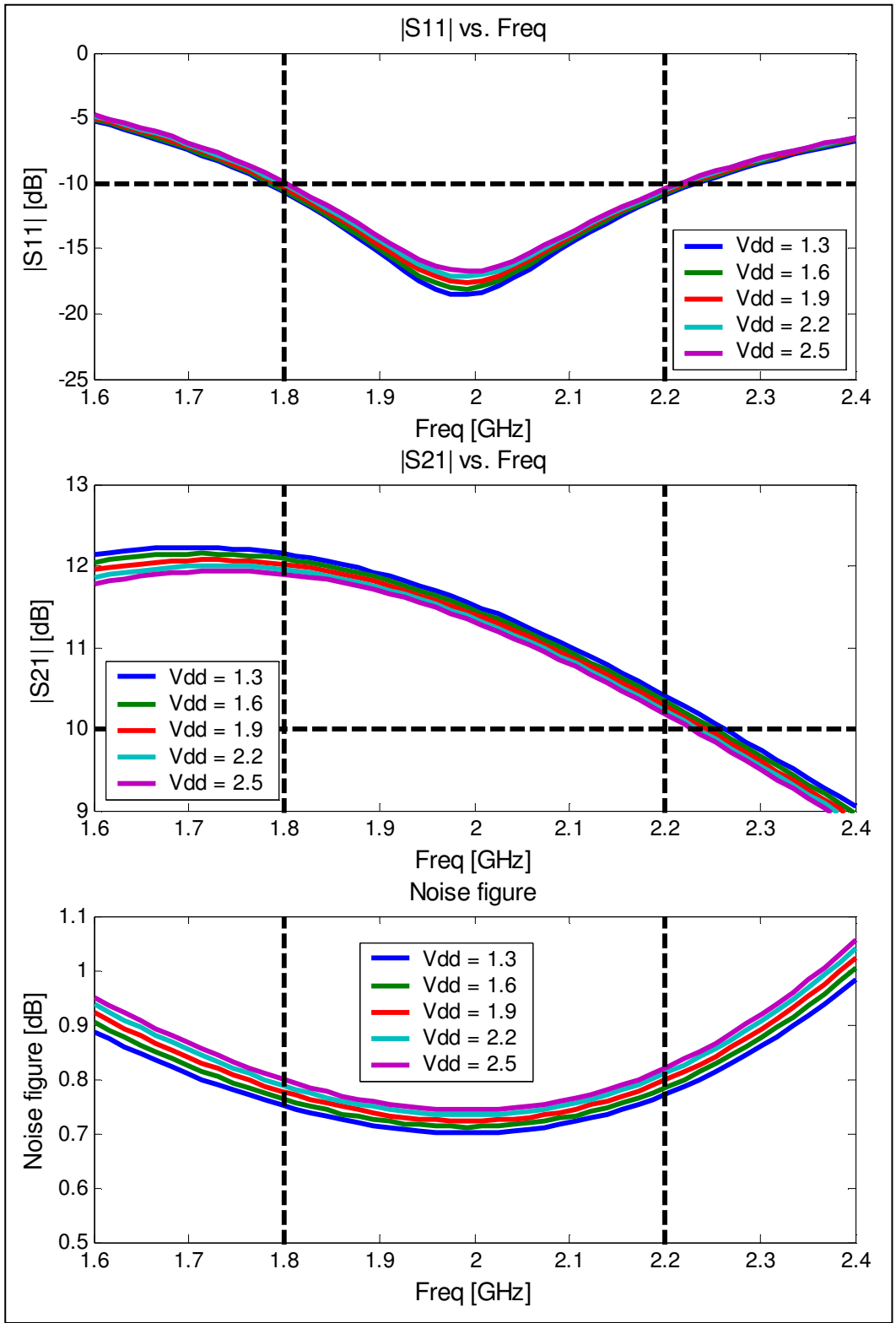


Figure 8, effect of Vdd on performance

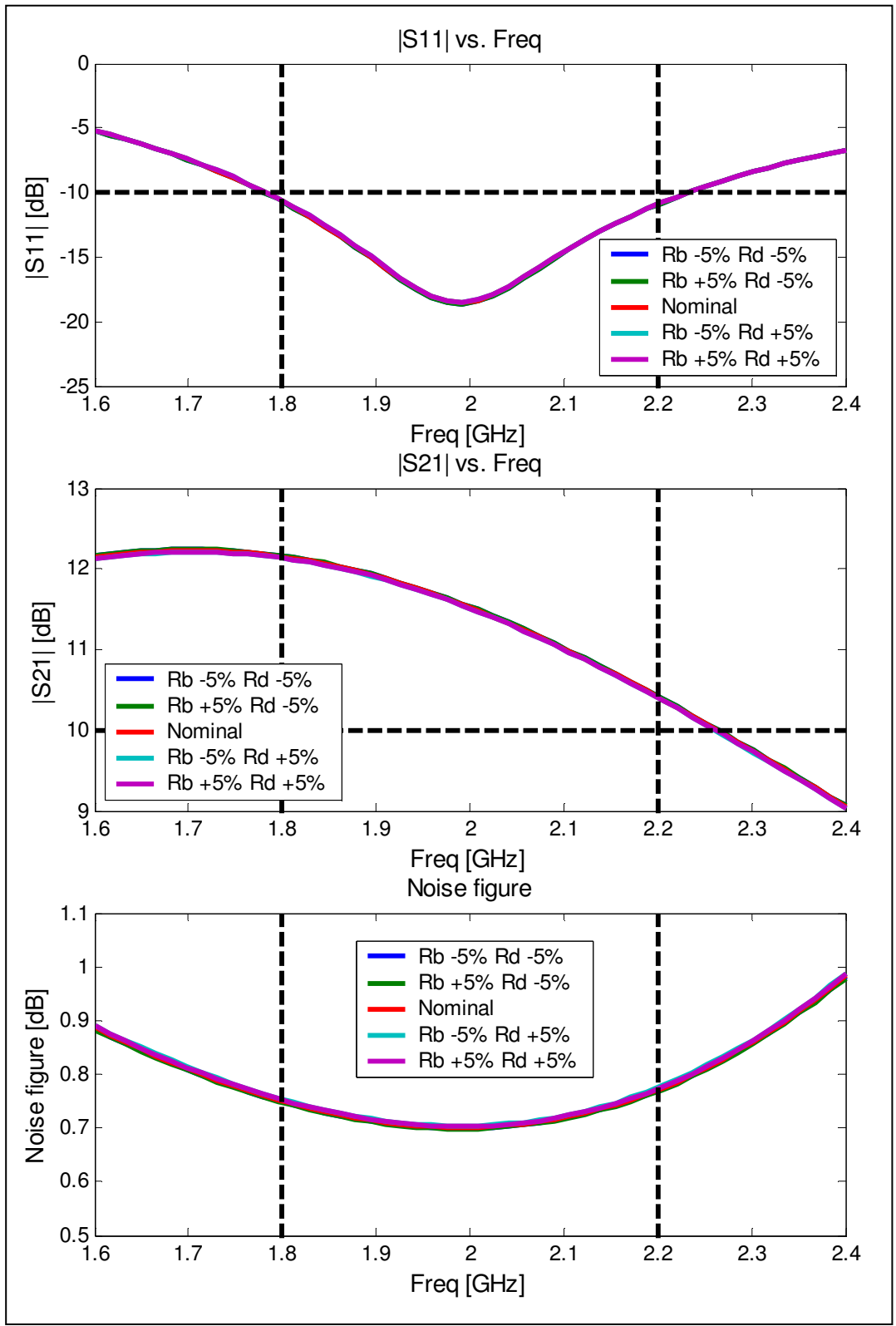


Figure 9, effect of resistor variation on performance

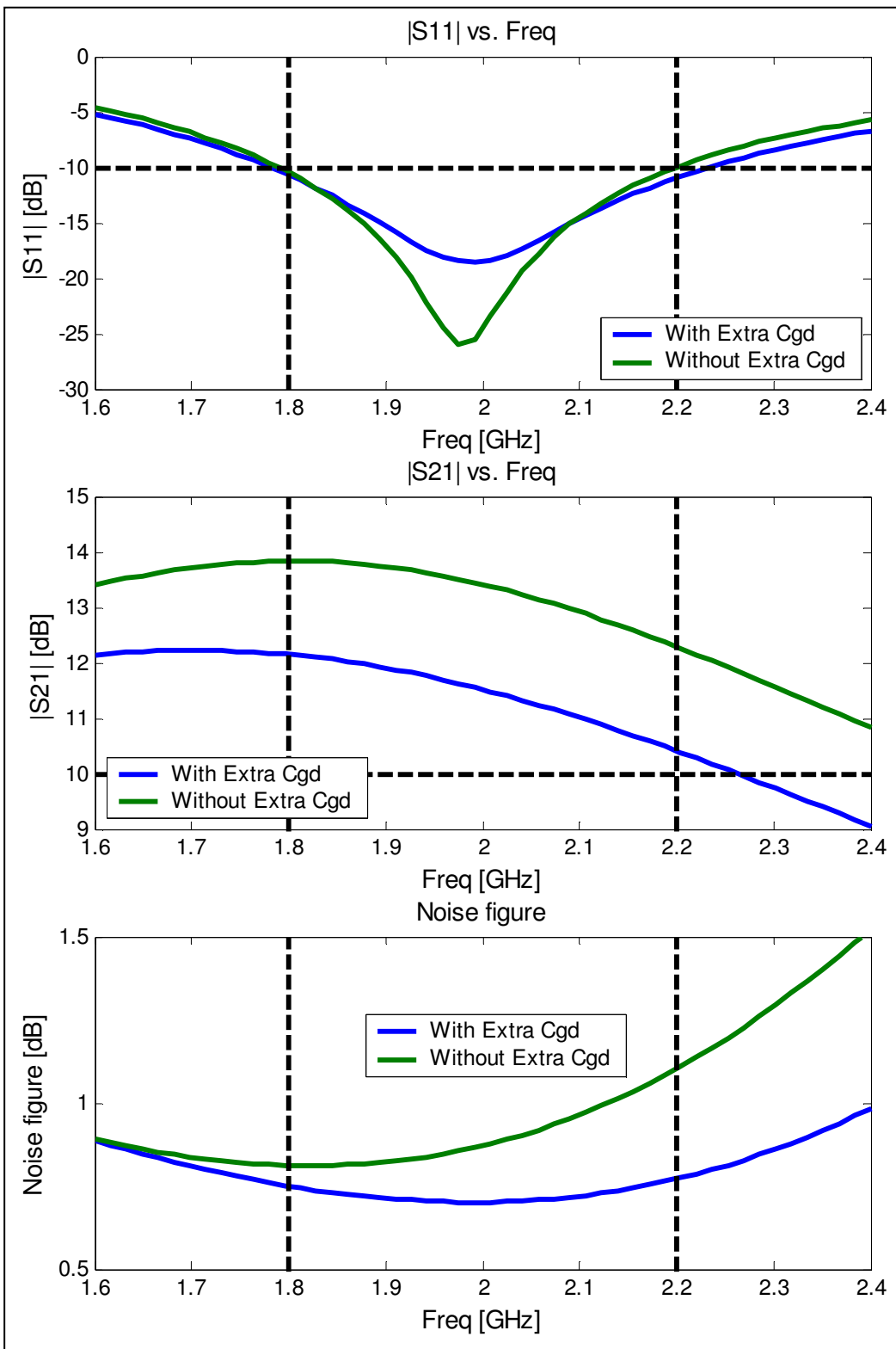


Figure 10, effect of extra Cgd on performance

Appendix B: MATLAB and hspice code

Writing parameters files:

```
project_directory='Z:\EE314\hspice\Project\';
parms_filename='params.txt';
full_parms_filename=cat(2,project_directory,parms_filename);

n = 1:5;
vdd= [1.3];
Rd = [.7]*1e3;
W1 = [120]*1e-6;
Wbias = W1/20;
Len = .25e-6;
Rb = 110e3;
Cgd2 = [85 .1 .1 .1 .1]*1e-15;
nf=[96];
nLs = 1;
lLs1 = [1.2 1.2 1.2 1 1]*1e-3;
lLs2 = [5.9 5.9 5.9 2.5 2.5]*1e-3;
nLg = 1;
lLg = [7.4 7.4 12 7.4 12]*1e-3;
nLd = 1;
lLd = [10]*1e-3;
nRl = [1];
lRl1 = [8]*1e-3;
lRl2 = [3.9]*1e-3;
lBias = [2.5]*1e-3;

for n=1:5;

    vddn = loopassign(vdd,n);
    Rbn = loopassign(Rb,n);
    Rdn = loopassign(Rd,n);
    nLsn = loopassign(nLs,n);
    lLs1n = loopassign(lLs1,n);
    lLs2n = loopassign(lLs2,n);
    nfn = loopassign(nf,n);
    Cgd2n = loopassign(Cgd2,n);
    nLgn = loopassign(nLg,n);
    lLgn = loopassign(lLg,n);
    lBiasn = loopassign(lBias,n);
    nLdn = loopassign(nLd,n);
    lLdn = loopassign(lLd,n);
    nRln = loopassign(nRl,n);
    lRl1n = loopassign(lRl1,n);
    lRl2n = loopassign(lRl2,n);
    W1n = loopassign(W1,n);
    Wbiasn = loopassign(Wbias,n);
    Lenn = loopassign(Len,n);

    parms_filename=['params',num2str(n),'.txt'];
    full_parms_filename=cat(2,project_directory,parms_filename);
    disp(['Writing parameters file params',num2str(n),'.txt']);
    fid=fopen(full_parms_filename,'w');
    fprintf(fid,'%Parameters file for LNA\n');
```

```

fprintf(fid, '.param vdd=\'\'%6.2f\'\'\'\'n', vddn);
fprintf(fid, '.param _Rb=\'\'%6.2fk\'\'\'\'n', Rbn/1e3);
fprintf(fid, '.param _Rd=\'\'%6.3fk\'\'\'\'n', Rdn/1e3);
fprintf(fid, '.param _nLs=\'\'%6.0f\'\'\'\'n', nLsn);
fprintf(fid, '.param _lLs1=\'\'%6.2fm\'\'\'\'n', lLs1n/1e-3);
fprintf(fid, '.param _lLs2=\'\'%6.2fm\'\'\'\'n', lLs2n/1e-3);
fprintf(fid, '.param _nf=\'\'%6.2f\'\'\'\'n', nfn);
fprintf(fid, '.param _Cgd2=\'\'%6.2ff\'\'\'\'n', Cgd2n/1e-15);
fprintf(fid, '.param _nLg=\'\'%6.0f\'\'\'\'n', nLgn);
fprintf(fid, '.param _lLg=\'\'%6.2fm\'\'\'\'n', lLgn/1e-3);
fprintf(fid, '.param _nLd=\'\'%6.0f\'\'\'\'n', nLdn);
fprintf(fid, '.param _lLd=\'\'%6.2fm\'\'\'\'n', lLdn/1e-3);
fprintf(fid, '.param _lBias=\'\'%6.2fm\'\'\'\'n', lBiasn/1e-3);
fprintf(fid, '.param _nRl=\'\'%6.0f\'\'\'\'n', nRln);
fprintf(fid, '.param _lRl1=\'\'%6.2fm\'\'\'\'n', lRl1n/1e-3);
fprintf(fid, '.param _lRl2=\'\'%6.2fm\'\'\'\'n', lRl2n/1e-3);
fprintf(fid, '.param _W1=\'\'%6.2fu\'\'\'\'n', W1n/1e-6);
fprintf(fid, '.param _Wbias=\'\'%6.2fu\'\'\'\'n', Wbiasn/1e-6);
fprintf(fid, '.param _L=\'\'%6.2fu\'\'\'\'n', Lenn/1e-6);
fclose(fid);
end

disp('Done.');
```

Loading multiple simulations:

```
function sweep_array = LoadSignals(dir_name,sp_file)

% Jim Salvia, October 11, 2005
%
% function sweep_array = LoadSignals(dir_name,sp_file)
%
% sweep_array: a cell array containing the variables from each sweep.
%     sweep_array{k} contains the variables from sweep #k. Notice that
%     MATLAB uses indexes starting at 1, while Spice indexes from zero.
%     Therefore, the variables from sweep.sw0 will be stored in
%     sweep_array{1}
% dir_name: the name of the directory with the spice output. Use '.' for
%     the present directory
% sp_file: the name of the spice output files that you want to read, without
%     including the number. For example, if you want to read all of the
%     DC sweep files (.sw0) and your spice input file was 'test.sp',
%     sp_name would be 'test.sw'. If you wanted the AC sweep files,
%     sp_name would be 'test.ac'
%
% This function will only work for a maximum of 10 sweeps because I wasn't
% sure how Spice named the 11th sweep (.sw11 ?). It could easily be
% expanded to include 100 sweeps if you know the naming convention.
%
% Example:
% Ran spice DC sweep with 3 .alter statements (4 total sweeps).
% Spice file was named test.sp . Ran in current directory.
% gm_Id was a .probe variable in the spice file.
%
% array = LoadSignals('.', 'test.sw')
% for k=1:length(array)
%     gm_over_Id(:,k) = evalsig(array{k}, 'gm_Id');
% end
% plot(gm_over_Id)
%
% This should generate a plot with 4 lines, each one representing gm_id from
% one of the spice DC sweeps

files = dir(dir_name);

for k = 1:length(files)
    file_name = files(k).name;
    len = length(file_name);
    if(len>3)
        if(len==length(sp_file)+1)
            if(prod(double(file_name(1:len-1))==sp_file))
                x = str2num(file_name(len));
                sweep_array{x+1} = loadsig(file_name);
            end
        end
    end
end
end
```

Plotting simulation results:

```
acout_cells = LoadSignals('.', 'Sparam_test.ac');
for n=1:length(acout_cells)
    acvout(:,n) = evalsig(acout_cells{n}, 'output');
    S21dB(:,n) = evalsig(acout_cells{n}, 's21_db');
    S11dB(:,n) = evalsig(acout_cells{n}, 's11_db');
    S22dB(:,n) = evalsig(acout_cells{n}, 's22_db');
    Zin(:,n) = evalsig(acout_cells{n}, 'zin_real') +
evalsig(acout_cells{n}, 'zin_imag')*j;
    Zout(:,n) = evalsig(acout_cells{n}, 'zout_real') +
evalsig(acout_cells{n}, 'zout_imag')*j;
    freq(:,n) = evalsig(acout_cells{n}, 'HERTZ');
end

dB = 20*log10(abs(acvout));

figure(101)
subplot(2,2,1)
plot(freq,S11dB,[freq(1),freq(end)],[-10,-10],'k--',[1.8e9,1.8e9],[-30 0],'k--',
',[2.2e9,2.2e9],[-30 0],'k--','linewidth',2)
title('S11 vs. Freq')
ylabel('S11 [dB] ')
xlabel('Freq [Hz]')

subplot(2,2,3)
plot(freq,S21dB,[freq(1),freq(end)], [10,10], 'k--', [1.8e9,1.8e9], [8 13], 'k--',
',[2.2e9,2.2e9], [8 13], 'k--', 'linewidth',2)
title('S21 vs. Freq')
ylabel('S21 [dB] ')
xlabel('Freq [Hz]')

subplot(2,2,2)
plot(freq,real(Zin),freq,imag(Zin),...
    [freq(1),freq(end)], [0,0], 'k--', [1.8e9,1.8e9], [-100 100], 'k--',
    ', [2.2e9,2.2e9], [-100 100], 'k--', 'linewidth',2)
title('Zin')
xlabel('Freq [Hz]')
ylabel('Zin [Ohms]')

nout = LoadSignals('.', 'nfse.ac');

for n=1:length(nout)
    nf(:,n) = evalsig(nout{n}, 'nf');
    freq(:,n) = evalsig(nout{n}, 'HERTZ');
end

subplot(2,2,4)
plot(freq,nf,[freq(1),freq(end)], [0,0], 'k--', [1.8e9,1.8e9], [0 5], 'k--',
',[2.2e9,2.2e9], [0 5], 'k--', 'linewidth',2)
title('Noise figure')
xlabel('Freq [Hz]')
ylabel('Noise figure [dB]')
axis([freq(1) freq(end) .5 1.5])
```

Hspice deck that incorporates MATLAB parameter files

```
* EE314 Midterm Project
* Jim Salvia & Clay Daigle
* February 20, 2006

**** Library ****
.include '~/EE314/hspice/ee314models.txt'
.include '~/EE314/hspice/Project/params1.txt'
.include '~/EE314/hspice/Project/cdaigle_LNA2.sp'
*****

.param rsrc=50

VDD  Vdd      Gnd  dc=vdd
Vg   Gnd      0    dc=0
Vs   Antenna Gnd  dc=0 ac=1
Cin  Antenna Input 100p
Xlna Input Output Vdd 0 lna
Cout output      output1 100p
Rout output1     Gnd      10g

.param fmin = 1.6G
.param fmax = 2.4G

.temp 17
.ac lin 50 fmin fmax
.net v(output,0) vs Rin=rsrc Rout=rsrc
.probe ac S21(dB) S11(dB) S22(dB) Zin(R) Zin(I) Zout(R) Zout(I)
.op
.options post brief method=gear accurate dccap

.alter
.include '~/EE314/hspice/Project/params2.txt'

.alter
.include '~/EE314/hspice/Project/params3.txt'

.alter
.include '~/EE314/hspice/Project/params4.txt'

.alter
.include '~/EE314/hspice/Project/params5.txt'

.end
```