Introduction to Matlab, HSPICE and SUE

Introduction
The primary objective of this lab is to familiarize you with three tools that come in handy in circuit design and analysis. In this lab, we will use Matlab to plot the transfer function of a system and explore its use in examining results from HSPICE simulations. HSPICE is a powerful circuit simulation tool that can run several different simulations on a circuit specified by a netlist. Simulations we will run are DC analysis, AC analysis, and transient analysis. SUE is a graphical program that we will use to create the netlist for use in HSPICE. This laboratory is intended to give you enough information so you will be comfortable using these tools on future homework assignments and the final project.

Getting Started
Log on to one of the Linux machines in B123 with the username (same as your fas account) and password you should have received in your email. Change to the es154 directory ($ cd es154). All of your files and programs should always be run from that directory.

Task 1: Bode plots in Matlab
The `bode` command in Matlab is useful for plotting the transfer function. Type `help bode` at the Matlab prompt for information on how to use this command. Let’s take a simple example to see how to use Matlab in plotting the transfer function. Consider the following circuit:

We can find the transfer function of this circuit to be \( H(s) = \frac{(R/L)s}{s^2 + (R/L)s + \frac{1}{LC}} \).

Here is an example Matlab script that will generate the Bode magnitude and phase plots:

```matlab
R = 11;  \% resistance in ohms
L = 100e-3;  \% inductance in henries
C = 10e-3;  \% capacitance in farads
w = logspace(0,4,200);  \% creates an array of logarithmically spaced
\% 200 points of numbers from 1e0 to 1e4
num = [R/L 0];  \% [bl b0] of equation b1*s^1 + b0*s^0
den = [1 R/L 1/(L*C)];  \% [a2 a1 a0] of equation a2*s^2 + a1*s + a0
[mag, phase] = bode(num,den,w);  \% uses the bode command to create the
\% magnitude and phase plots
```
figure % initializes a new figure window
semilogx(w,20*log10(mag)) % plots the mag in dB w.r.t. log (w)
xlabel('w')
ylabel('Magnitude (dB)')
title('Bode magnitude plot of example circuit')

figure
semilogx(w,phase)
xlabel('w')
ylabel('Phase (degrees)')
title('Bode phase plot of example circuit')

Some things to note in Matlab:
- Start up Matlab by typing `matlab` in a terminal window
- Generally it is easier to create an M-file (e.g., `<file>.m`) with all the commands scripted, rather than typing the commands in over and over.
- The enter a row vector in Matlab elements are separated by a space (e.g. if you want a vector x to have elements 1,2,and 3 type `x = [1 2 3]`)
- Putting a semicolon at the end of the line suppresses the output, if you leave it off the output will print to the screen
- The `bode` command can generate the plots automatically, but often it is useful to do it as in the example since you can control the scale of your axes and their labels.
- `Logspace(p1,p2,N)` creates a vector of N data points between $10^{p1}$ and $10^{p2}$

Using the example above as a guide, answer the following questions. What happens to the frequency response of the circuit if the value of L is raised and lowered? What if C is raised or lowered? How about R?

Now let’s look at a system with two complex poles. We will look at how $\zeta$ affects the magnitude response. Create a Matlab script to plot the magnitude response of

$$H(s) = \frac{1}{1 + 2\zeta \left( \frac{s}{\omega_n} \right) + \left( \frac{s}{\omega_n} \right)^2}$$

for $\zeta$ values of 0.05, 0.25, 0.5, 0.707, and 1 all on the same graph, where $\omega_n = 1$. Make sure your plot is over the frequency range of $10^{-1}$ and 10 and that you use enough data points to capture what is happening. You will need to use the `hold on` command to plot all the function on the same graph. You may wish to do this with a `for` loop (type `help for` for details).

After these plots are created, add in the straight-line approximation. Here is the code to do this:

```matlab
x=[.1 1 10];
y=[0 0 -40];
semilogx(x,y)
```

How does the value of $\zeta$ affect the magnitude response of the system? For what values of $\zeta$ is the straight-line approximation more accurate than a system with two real poles at $\omega_n$?
Find the transfer function to explain the bode data below. Also create bode plots to show how closely your transfer function matches the data.

A useful function may be `conv(a,b)` which performs polynomial multiplication, where `a` and `b` are vectors of the coefficients in descending powers (see `help conv`).
SUE and HSPICE

Before coming to lab, it will be helpful to read over the handout ES 154 – Using SUE and Appendix C of Sedra and Smith, An Introduction to SPICE. This will allow you to make better use of the lab time. When we make a simulation, there are four steps to follow.

1. Create the circuit in SUE and generate the netlist.
2. Create a SPICE control deck using your favorite word processing program.
3. Run the simulation using HSPICE.
4. View the results in MATLAB and verify the results.

There are four types of simulations that we will use: Operating point, DC analysis, AC analysis, and transient analysis.

Task 2: DC Analysis – I-V curve generation

DC analysis can be used to generate IV curves of different transistors. The following example will walk you through the steps of the simulation to create the IV curves. By adjusting this template, you should be able to create a simulation for any circuit.

First, run SUE and start a new schematic. On the right of your screen you should see three boxes. The top box is a list of open schematics, the bottom two are for different parts that can be added to your circuit. In the middle window, click on the menu bar and a choice box should appear. Choose the devices option. This should put some generic parts like input, output, global, etc. into the window. The bottom window should already contain electronic parts like transistors, resistors, etc. Create the following circuit.

In SUE, it will look like the figure on the right. If you double click on a part, you can set the properties for that part. In the case of the transistor, we can set the length, width, etc. These can be specified directly here, or you can choose to set them to a variable name and specify the value in the control deck. This allows you to run multiple simulations on the circuit without having to recreate the netlist in SUE. In this example, the width parameter has been set to wn. Under the run tab in the menu, there is a create netlist option. When you choose this, SUE creates <filename>.sp and <filename>.spi files in your current working directory. We will be using the <filename>.spi file which explicitly specifies the source and drain area and perimeter sizes for the transistors. Now we are ready to create a control deck to run the simulation.
The following is a template for the control deck.

* template.hsp

* include a models file which can be found in the following path
  * /export/cad/es154/spice/models/
  * and choices are:
  *   bjt.models - has models for q2n3904 and ideal pnp and npn
  *   tsmc0p25.models - has models for TSMC 0.25um CMOS
  *   tsmc0p35.models - has models for TSMC 0.35um CMOS
  * usage: .include '/export/cad/es154/spice/models/{file}.models'
  *   AMI0p5.lib - has corner models for AMI 0.5um CMOS
  *   use one of the following corners: TT,FF,SS,FS,SF
  * usage: .library '/export/cad/es154/spice/models/AMI0p5.lib' TT

  * must scale with respect to lambda which depends on process technology
  *.option scale=0.25u (for AMI0p5)
  *.option scale=0.13u (for tsmc0p25)
  *.option scale=0.18u (for tsmc0p35)

* include the schematic netlist
  *.include 'schematic.spi'

* specify parameters
  .param
    + wn = 10

* power supply
  *Vdd vdd gnd 5
  *Vcc vcc gnd 12
  *vee vee gnd -12

* input stimulus - need to drive the inputs with sources
  *Vsine in gnd sin(Vdc_offset Vamplitude freqi phase)
  *Vac in gnd dc=# ac=#
  *Vdc in gnd #
  *Vpulse in gnd pulse (v1 v2 tdelay trise tfall tduty tpd)
  *VPWL in gnd PWL (t1,v1 t2,v2 t3,v3 etc)

  * whatever run you want -- .ac or .dc or .tran

* tell hspice to dump out all nodes
  .option post

* need to tell hspice the end of the file
  .end

NOTE: a line beginning with a ‘*’ is a comment to HSPICE.
Here is the control deck for the example circuit (the actual deck must exclude the numbers in the beginning of the lines – they are there for explaining the deck):

* task2a.hsp

*.include '/export/cad/es154/spice/models/tsmc0p35.models'

1) .include '/export/cad/es154/spice/models/tsmc0p35.models'

2).option scale=0.18u (for tsmc0p35)

* include the schematic netlist

3).include 'test.spi'

* specify parameters

4).param
    + wn = 10

* power supply

Vdd vdd gnd 3.3

*input stimulus

5)Vgate vgate gnd 1

6)Vdrain vdrain gnd 1

7).option post

8).dc vdrain 0 3.3 0.1 sweep vgate 0 3.5 0.5

9).end

Explanation of the control deck:

1) includes the HSPICE model for the transistor

2) sets the scale for determining the length and width of the transistor

   a. All widths and lengths specified in units of lambda are multiplied by this
      scaling factor – see 4) for definition of lambda

3) include the netlist created by SUE

4) set the parameters not specified in SUE

   a. NOTE: transistor dimensions are in units of lambda, which is defined to
      be equal to \(L_{\text{min}}/2\) (\(L_{\text{min}}\) = minimum channel length specified by the
      technology). This limits the granularity of transistor sizes.

5,6) sets the input stimuli

7) tells HSPICE to print all outputs

8) gives the type of analysis – in this case, we will take vdrain and increase it from

   0V to 3.3V in 0.1V steps and we will iterate multiple times for vgate of 0V to

   3.5V in 0.5 volt steps

9) tell HSPICE to end the simulation

After creating this file <file>.hsp, we can simulate it in HSPICE as follows:

$> hspice <file>.hsp

You can also pipe the output to a file instead of the window as follows:

$> hspice <file>.hsp >> <output_file_name>
If the simulation is successful HSPICE will tell you so and you should see and output file such as <file>.st0. This is the file that we will read into Matlab to view the results.

To view the results in Matlab we will be using a toolbox written by Michael Perrott at MIT. To use this toolbox you will need to add it to your path by typing:

```
>> addpath('/export/cad/es154/matlab/hspice_toolbox')
```

There is some documentation at

```
```

but the needed commands will be explained below.

The following script creates the IV characteristic for the transistor in our example.

```
%load the HSPICE output file
x = loadsig('test.st0'); % x can be any variable you assign the output

%list all the available signals
lssig(x)

%plot the IV curves (Id vs. Vds for each Vgs)
plotsig(x,'-I_vdrain')

title('IV curve for NMOS tsmc0p35')
```

Notice that this plots $I_d$ vs. $V_{ds}$ for the fixed $V_{gs}$ values we specified in the control deck.

Perform the following:

A) Create the IV curves for both nMOS and pMOS devices using the tsmc0p35 technology as specified above for the following width and length combinations for both nMOS and pMOS devices:

- $W = 10, L = 2$
- $W = 100, L = 2$
- $W = 100, L = 10$

Notice the pMOS setup is a little different

B) Create IV curves for a npn BJT found in the bjt.models models file, found in `/export/cad/es154/spice/models` directory, in the similar manner. You will need to create a similar SUE file and netlist for a BJT. Double click on the BJT to specify the model to be used – type in `q2n3904`.

Answer the following questions:

C) How does the length and width affect the IV characteristics of the transistor (Remember velocity saturation)?

D) What is the value of $V_A$ for the BJT and $1/\lambda$ for the MOSFETs (a useful command may be `axis([xmin xmax ymin ymax])`)?

E) How does the spacing between the curves of various $V_{gs}$ values in the MOSFET differ from the spacing of the curves of various $V_{be}$ for the BJT and what causes this difference?
Task 3: DC Operating Point

The DC operating point simulation results in all the DC node voltages and branch currents and all the power dissipation of all DC sources being written to the output file. Essentially, the DC sweep analysis that we performed in the previous task is just a bunch of operation point simulations all run back to back by varying the source voltage. This is a very simple, but useful simulation.

Find the DC operating point of the following circuit.

- Draw the above circuit in SUE.
- Double click on the BJTs and change the model from b2.4 to q2n3904 (this a BJT used in several examples in Appendix D of S&S).
- Use Vcc = Vee = 10V.
- To run a DC op simulation, use the .op command in your <file>.hsp input command file (i.e., instead of .dc used in Task 2)

Answer the following:
A) What is the magnitude of current I in the above figure?
B) What is the value of $\beta$ for the transistors?
C) What mode of operation is each transistor in?
Task 4: AC analysis

The .ac command performs a linear small-signal frequency response analysis. It calculates the DC operating point automatically and finds the small-signal parameters. Then it analyzes the linear small-signal circuit over the specified frequency range. You should perform the AC analysis on 4 circuits: common-emitter, common-base, common-collector, and common-source amplifiers. You should choose your analysis to have a frequency range so that when you view it in Matlab you can see the cutoff frequencies of the various amplifiers.

The AC analysis uses the following command line in your command file <file>.hsp.

Usage: .ac dec points_per_dec freq_start freq_stop

The circuits are shown below:

Common-emitter amplifier
Common-Base Amplifier

Common-Collector Amplifier
Answer the following questions:

A) What are the cutoff frequencies of each amplifier?
B) What is the Gain-BW product of the CE, CB and CC amplifiers?
C) How does the bandwidth of the CS amplifier compare with its BJT counterpart?

In class, we saw that we can increase the bandwidth of the CS amplifier by changing gm. Play around with the transistor size and see if you can increase the CS amplifier’s bandwidth.

D) Plot BW vs. wn.
   a. Is there a peak? If so, why?
Task 5: Transient Analysis

The transient analysis computes the circuit variables as a function of time over the time period specified. Using the previous netlists for the common-emitter and common-source amplifiers, perform a transient analysis.

For the common-emitter, sweep the magnitude of vs by editing the spice control deck as follows:

- You will need to specify a parameter for the voltage magnitude (i.e., `.param mag_v = 10e-3`).
- Set up the input stimulus Vs as a sine wave:
  \[ \text{Vs Vs gnd sin(<DC\_voltage> 'mag_v' <freq> 0 0 0))} \]
  where
  - `mag_v` is a parameter that sets the amplitude of the sinusoid
  - `<freq> = 10kHz`
  - Does the value of `<DC\_voltage>` matter?

For the analysis command use

\[ \text{.tran <t\_step> <t\_stop> sweep 'mag_v' <start_val> <end_val> <step_val>} \]

Run the simulation for 100ns (t\_stop) with a step size (t\_step) of 1ps. Sweep the magnitude of Vs from 10mV to 100mV in 10mV steps. Hspice will create an output file `<file>.tr0` which you should load into Matlab and plot your results.

For the common-source amplifier run a transient analysis over the same time interval, but vary the input frequency from 10kHz to 100kHz in 10kHz steps. Plot you results in Matlab.