

Circuit Analysis with Matlab, Maple, and PSpice

Hemanshu Roy Pota
h-pota@adfa.edu.au

July 8, 2003

1 Introduction

Circuit analysis can be thought of as a two step process; the first step is the conceptual one and the second step is a mechanical one. The conceptual step consists of converting a circuit description into a set of mathematical equations which relates the currents and voltages in different parts of the circuit. The mechanical step consists in solving the derived set of equations. This is illustrated using a sketch in Figure 1.

My experience has been that students give excessive importance to the mechanical step, i.e., in solving the mathematical equations, and consequently they lack confidence in the concepts of circuit theory. A good example is the phasor analysis. Students learn phasor analysis just when they are learning about complex numbers, so very soon they are oblivious of the theory behind phasor analysis and all their energy goes into mastering the complex number calculations.

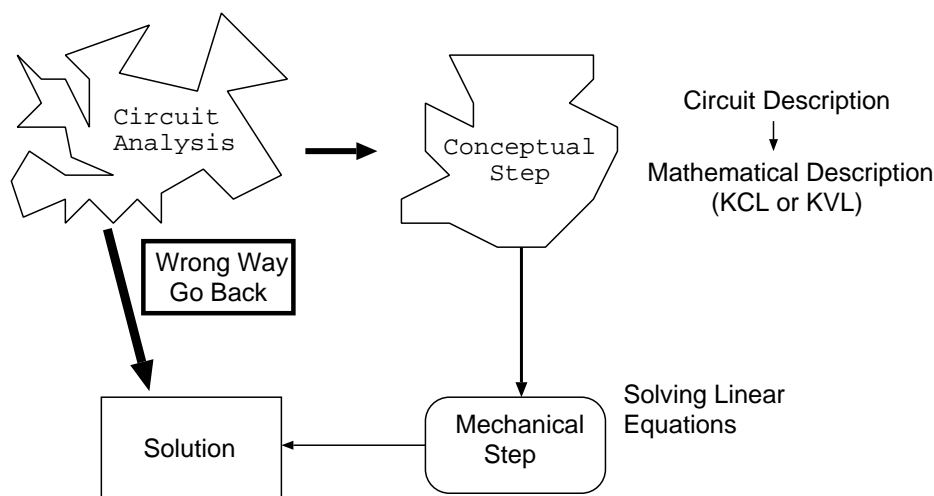


Figure 1: A Pictorial Guide to Circuit Analysis

With a desire to remedy the situation I encourage you to use computers as much as possible to carry out the (second) mechanical step of the circuit analysis. Concentrate all your energies on the conceptual step and very soon circuit analysis will be that very fun you were promised when you opted for a degree in Electrical Engineering!

The conceptual step consists in using KCL (node equations) or KVL (loop equations) to mathematically describe a circuit. You need to master only, I repeat, *only* two basic principles. Even out of these two principles you can afford to concentrate on just the KCL and writing node equations and you will be set to analyse electrical circuits for the rest of your professional life.

It is my intention in this course to firstly show you how circuit analysis can be always broken into the above two steps and secondly to get you to use computers to ease your burden in carrying out the mechanical step. In the following we discuss three general purpose packages you are encouraged to use to help you with the mechanical manipulations in the circuit analysis. After a brief introduction to the three general purpose packages we will see how we analyse two circuits with the assistance of these three packages.

Matlab

Matlab is a 'calculator' on a computer. To start the Matlab program you either type 'matlab' or double-click as the case may be. Once the program is running it prompts for an input; the most common prompt is '>'. When you want to calculate $2*2$ on a calculator you press the following key sequence: $2 * 2 =$ and the answer is displayed on the screen. A Matlab session to do the same looks like:

```
> 2*2 ENTER  
ans=4
```

Most calculators have a one line display but Matlab has almost an endless display and ability to recall previous calculations. Moreover the output of calculations can be stored in variables and used later. For example the above session can be:

```
> c=2*2 ENTER  
c=4
```

A big difference between a calculator and Matlab is that most calculators can only store real numbers while Matlab variables can be more general like complex numbers and matrices. This means that after giving a numerical value to a matrix variable it can be used just like any other variable. For example try:

```
> a=[1,0;0,1] ENTER  
> b=[2,0;0,2] ENTER  
> c=a*b ENTER
```

Matlab also has hundreds of predefined functions which make it possible to write short scripts to perform complex calculations.

Matlab is available on the department and the computer centre unix network. It is a good idea to purchase the Student Edition of Matlab (available both for a PC and a Mac) from the local bookstore. Student's edition costs between \$70 – \$80 and is well worth the money. Start using it as early as possible as a general computation tool; it will pay rich dividends later. We will discuss more about Matlab along with the two examples later.

Maple

Maple will do most of the things Matlab does and it will do more; Maple will perform symbolic computations for you. Once you learn to use Maple it can replace the traditional pencil and paper note taking. Many times while taking notes or solving a problem you would want to keep some variables in symbolic form and not assign any numerical value to them. With programs like Matlab which only do numerical computation it is not possible to keep anything in a symbolic form and hence a computer can't be turned into a note book. With Maple you can turn your computer into a computing machine and a traditional note book. Maple is available on the department and the computer centre unix network. If you need to get a copy to load on your own micro computer (PC or Mac) request a set of disks from the computer centre or download it from the computer centre server. There are a few books in the library which give a good introduction to Maple and you can also borrow manuals from the computer centre. I strongly encourage you to pick up this important tool of the Engineering trade. We will see more of Maple while solving the circuits later.

PSpice

PSpice is a port of the legendary SPICE (simulation program with integrated circuit emphasis) package to micro computers. This program performs all the circuit analysis you will need to perform and much more. The text-book has a good introduction to PSpice. After learning to use this program the only question that remains to be answered is: Given programs like PSpice why should anyone learn circuit analysis at all? We will attempt to answer this question during the course. Evaluation version of PSpice is available for Mac
`/luciano/users/ugrads/Share/hrp/pspice6.0-mac.`

The PC version of PSpice is available from our ftp site:

`ftp://evans.adfa.oz.au/pub/staff/hrp/c80dlabe.exe.`

Component models of various devices to be used with SPICE are available in `/pub/spice`. Web browsers

can use <http://www.ee.ualberta.ca/ftp.html>. MicroSim, the company which markets PSpice has its own ftp site and also a www site (<http://www.microsim.com>). You can ftp the latest PSpice evaluation version from their site: [ftp ftp.netcom.com](ftp://ftp.netcom.com), login anonymous, cd /pub/mi/microsim/Eval_Versions, get the files winevals.txt and README.TXT, read them, and then download what you want. Have a look at other directories and you will find many other interesting things.

2 A Simple Problem

In this example we find the voltages at nodes 1 and 2 in the circuit shown in Figure 2. Node numbers are shown next to the nodes.

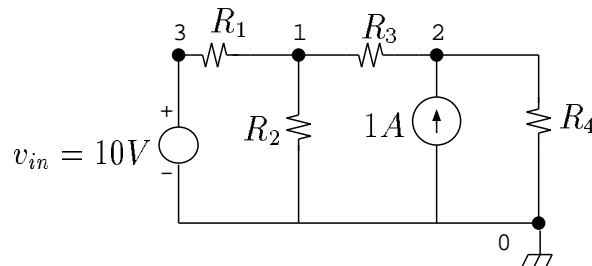


Figure 2: Circuit Example 1

The Conceptual Step

To analyse the circuit in Figure 2 the first step is to use KCL and write node voltage equations.

$$\frac{V_1 - 10}{R_1} + \frac{V_1}{R_2} + \frac{V_1 - V_2}{R_3} = 0 \quad (\text{KCL @ node 1}) \quad (1)$$

$$\frac{V_2 - V_1}{R_3} + \frac{V_2}{R_4} = 1 \quad (\text{KCL @ node 2}) \quad (2)$$

The most important part of the solution is to write node voltage equations (1) and (2); rest is mechanical. To get the values of V_1 and V_2 we have to solve equations (1) and (2). Next we use Matlab and Maple to solve these two equations for V_1 and V_2 . Later PSpice is also used to solve the circuit.

Matlab



Matlab understands the language of matrices. Equations (1) and (2) have to be rewritten in a matrix form before we can use Matlab to solve them. A matrix equivalent of equations (1) and (2) is:

$$\begin{bmatrix} \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} & -\frac{1}{R_3} \\ -\frac{1}{R_3} & \frac{1}{R_3} + \frac{1}{R_4} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} \frac{10}{R_1} \\ 1 \end{bmatrix}$$

The above relationship is solved using Matlab as shown below. At the command prompt (bash\$ for the EE computers) type 'matlab'. Wait for the Matlab interactive session to start. Type each line as shown below, terminating them with ENTER.

```
%Example 1 (14-02-96)
R1=2; R2=2; R3=3; R4=4;
a=[1/R1+1/R2+1/R3, -1/R3; -1/R3, 1/R3+1/R4]
vnode=inv(a)*[10/R1;1]
v1=vnode(1)
v2=vnode(2)
```

Everything following a % sign on a line is ignored by Matlab; in other words comments should be preceded by %. The second line above assigns numerical values to the resistors. The third line defines 'a' as the required matrix. In Matlab, matrices are enclosed in square brackets '[']; row 1 is input first with entries separated by a coma ',' or a space and then row 2 is entered and so on. Rows are separated by either a semicolon ';' or a new-line character. The fourth line in the script above solves for the vector with voltages V_1 and V_2 ; inv is a built-in Matlab function. To see what it does type help inv and Matlab tells you the entire story. Incidentally to learn more about any function 'funname' just say help funname and Matlab will tell you what it's all about. If you want to know if Matlab has a built-in function for what you want to do just type lookfor funIwant. For example if you wanted to know if there is a function which will perform matrix inversion type lookfor inverse. To learn more about Matlab just type expo or help and follow the directions.

Instead of typing all the instruction in an interactive Matlab session one by one you can store them in a file and get Matlab to execute it all in one go. Use your favourite editor or emacs to create a file 'example1.m'. Type in the entire sequence of instructions as you see above in the file and save it. Start a Matlab session from the same directory where you created the file and when Matlab prompts you for a command, type example1. There is nothing special about the name 'example1.m' you can use any name; remember that the file should have an extension .m; if you don't want the output of a command to be displayed on the screen, terminate the command with a semi-colon, ';'. Try help diary. Matlab keeps a command history of all the commands entered in the session; to scan forward and backward in the command history, type  for backward, and  for forward.

Maple

To solve the circuit in Example 1 the first step was a conceptual one leading to node voltage equations (1) and (2). To use Matlab in solving these equations we had to convert the equations in a matrix form. With Maple we can avoid even that step; problems can be presented to Maple just the way you formulate it. Start a Maple session by typing 'xmaple&' on a unix machine or double-click as the case may be. Once a Maple session starts, type the following lines one at a time and terminate them by pressing ENTER.

```
#Example 1 (14-02-96)
eqn1:=(v1-10)/R1+v1/R2+(v1-v2)/R3=0;
eqn2:=(v2-v1)/R3+v2/R4=1;
sol:=solve({eqn1,eqn2},{v1,v2});
simplify(subs(sol,{eqn1,eqn2}));
assign(sol);
R1:=2; R2:=2; R3:=3; R4:=4;
v1;
v2;
```

The first line in the script is a comment line; comments should be preceded by a '#' character. Second and third lines tell Maple the equations describing the circuit. The next line is a command to solve both these equations for the variables 'v1' and 'v2'. The next statement verifies that Maple has indeed generated a correct solution; it substitutes the solution 'sol' in the equations to make sure that there has been no error in the computation. The line following it tells Maple to assign values to the variables 'v1' and 'v2' as computed by the solve command. The last part of the script assigns numerical values to these parameters.

This interactive Maple session should give you a feel for what symbolic computation is; explore further by going through the tutorial or the help features. To get more information about any command, you need to type ?commandIneedInfoAbout; question mark, '?', should precede the command about which you want more information. Try ?solve. An example of the command use can be obtained by example(commandIneedInfoAbout). Try example(solve).

Every Maple command should end with a semi-colon, ';' and if you want Maple not to display the result on the terminal, terminate the command by a colon ':'.

Maple also has an option to read in script files. Use your favourite editor or emacs and create a file 'example1maple'. Type in all of the Maple script shown above. Start a new Maple session from within the

directory where the file 'example1maple' lives. At the prompt, type `read example1maple;` and note what Maple does.

It's a good idea not to have any file extensions like .m or .maple with Maple script files. The file name can be anything and when you want to use an extension please read the information in `?read` beforehand.

Remember that once you have assigned values to variables using `assign` command then it doesn't make any sense to use them in equations again. Maple will complain if you try to do so; if you want to repeatedly run a script (after making certain changes) which has `assign` commands in it, precede the `read` command with a `restart` command. For our example, type `restart; read example1maple;`.

PSpice

PSpice is a wonderful program but it isn't an interactive one. The evaluation version for PCs has an interactive interface but the Macintosh version doesn't. Here we will look at the Macintosh version; it's easier to use the PC version.

Start an editor, may be MS Word, type in the script as you see below and save the file as `example1.cir`. Make sure you save the file as 'text' and not as 'normal'. If at times when you are sure that the file is there and PSpice can't see it, open the file and go to 'save as' under the 'file' menu; where it says 'Save File as Type' make it 'text' and then save the file.

After creating and saving the file, double-click on the PSpice application. Then go to the 'open' under 'file' menu; select the file 'example1.cir' and PSpice will perform the simulation. If there are any errors then an error description can be found in the file `example1.out`; correct them in the original file 'example1.cir' and then go to PSpice, open the file, etc.

PSpice comes with another program Probe which is used to display and plot the results from PSpice. Double-click on Probe icon and again go to 'open' in the 'file' menu and open the file `example1.dat` and follow the steps.

```
*Example 1 (14-02-96)
v1 3 0 DC 10V
R1 3 1 2
R2 1 0 2
R3 1 2 3
R4 2 0 4
I1 0 2 1
.DC v1 0 20 2
.PRINT DC I(R1) V(1) V(2) I(R4)
.PROBE
.END
```

Now a few words about the above script. The first line of every PSpice script has to be a comment line. I have a '*' (the comment indicator in PSpice scripts) as the first character but even without the '*' the first line is taken as the comment or title line. Every circuit should have a node '0', the reference node. Rest of the nodes can have any arbitrary number but the reference node has to be numbered '0'. The second line declares that the circuit has a voltage source between the nodes 3 and 0 of 10V DC. The following four lines tell resistor values. Voltage sources have to have 'v' as the first letter, resistors have to have 'R' as the first letter etc. PSpice is case insensitive. All commands are preceded by a dot, '.'. The command `.DC v1 0 20 2` tells PSpice to calculate various voltages and currents in the circuit when `v1` varies between 0 and 20V in the steps of 2V; `.DC` means perform a DC analysis; more about this later. The `.PRINT` command prints the values of all the parameters listed in a file `.out`; `I(R1)` means current through the resistor R_1 and `V(2)` means the voltage at node 2, etc. The command `.PROBE` tells PSpice to create a `.dat` file suitable to be read by Probe. All PSpice scripts must terminate with a `.END` line.

The PC version, known as Design Centre, has a schematic capture program; use this and save the circuit in a file `.sch`. The 'analysis' menu of the schematic capture program has commands to simulate the circuit etc.; a `.cir` file is automatically created from `.sch` to be fed into PSpice. Alternatively a `.cir` file can be created using an editor as discussed above and then read into PSpice. On completing the simulation Probe program is automatically fired. The Design Centre also has good help features.

3 A More Difficult Problem

The simple example in the last section was meant to get you familiar with three very useful packages. In this difficult example we will do something meaningful from the circuit analysis point of view and consolidate on what we have already learnt about the three packages. The problem here is to find the steady-state output voltage for the circuit shown in Figure 3 for a 1000 Hz sinusoidal input.

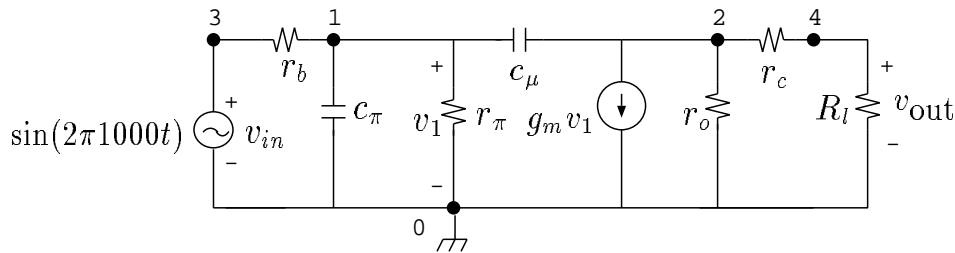


Figure 3: Circuit Example 2

The Conceptual Step

The problem asks for a steady-state output to a circuit with a sinusoidal input and this is where we use phasor analysis. Remember that in phasor analysis all the elements are replaced by their equivalent impedances. After replacing all the capacitors by their impedances, the following two nodal equations describing the circuit can be written.

$$\frac{v_1 - v_{in}}{r_b} + \frac{v_1}{r_\pi} + j\omega c_\pi v_1 + j\omega c_\mu (v_1 - v_2) = 0 \quad (\text{KCL @ node 1}) \quad (3)$$

$$\frac{v_2}{r_o} + \frac{v_2}{(r_c + R_l)} + g_m v_1 + j\omega c_\mu (v_2 - v_1) = 0 \quad (\text{KCL @ node 2}) \quad (4)$$

In the following Matlab and Maple scripts the above two equations are solved for the output voltage, v_{out} .

Matlab

As an exercise rewrite the equations (3) and (4) in a matrix form and see that the following Matlab script correctly encodes the matrix. Either create a file 'example2.m' with the following script or execute this script interactively and note down the output voltage. What is the physical significance of the complex number you get as the output voltage?

```
%Example 2 (14-02-96)
rb=10; cpi=0.01*10^(-6); rpi=2600; cmu=1*10^(-12);
gm=40*10^(-3); ro=100*10^3; rc=100; w=2*pi*1000;
Rl=5000; vin=1;
a=[1/rb+1/rpi+i*w*cpi+i*w*cmu, -i*w*cmu; ...
   gm-i*w*cmu, 1/ro+1/(rc+Rl)+i*w*cmu]
vnode=inv(a)*[vin/rb;0]
Vout=vnode(2)*Rl/(rc+Rl)
```

Maple

The Maple script given below should by now begin to mean something to you. Either save the following script in a file 'example2maple' or interactively execute the commands and note the output voltage. The last command evalf in the following script is asking Maple to evaluate the variable in floating point arithmetic.

```
#Example 2 (14-02-96)
eqn1:=(v1-vin)/rb + v1/rpi + I*w*cpi*v1+(v1-v2)*I*w*cmu=0;
eqn2:=v2/r0+v2/(rc+Rl) + gm*v1 + (v2-v1)*I*w*cmu=0;
sol:=solve({eqn1,eqn2},{v1,v2});
simplify(subs(sol,{eqn1,eqn2}));
assign(sol);
rb:=10; cpi:=0.01*10^(-6); rpi:=2600; cmu:=1*10^(-12);
gm:=40*10^(-3); r0:=100*10^3; rc:=100; w:=2*Pi*1000;
Rl:=5000; vin:=1;
Vout:=v2*Rl/(rc+Rl);
evalf(Vout);
```

PSpice

Create and save the following PSpice script in a file called example2.cir. Use PSpice and Probe to simulate the circuit behaviour.

```
*Example 2 (14-02-96)
v1 3 0 AC 1V
rb 3 1 10
cpi 1 0 0.01uF
rpi 1 0 2600
cmu 1 2 1pF
r0 2 0 100K
rc 2 4 100
Rl 4 0 5000
G1 2 0 1 0 40m
.AC LIN 10 100 2000
.PROBE
.END
```

Most of the commands in the above script should be clear. The declaration `G1 2 0 1 0 40m` tells PSpice that this is a voltage dependent current source between the nodes 2–0, controlled by the voltage across the nodes 1–0, with a gain of 40×10^{-3} . The command `.AC LIN 10 100 2000` tells PSpice to perform phasor analysis for input sinusoids with frequencies between 100 and 2000 Hz, varying linearly in 10 Hz intervals. Finally use Probe to see the frequency response.

4 The Heart of the Matter

We analysed two circuits; one simple and another difficult. As far as the conceptual part is concerned it can be safely said that there is little difference between the two problems; the difference in the level of difficulty is in solving for the equations which describe the circuit. Writing down of equations (1) and (2) and equations (3) and (4) is a straightforward matter. My intention behind writing this note is to make you realise that once you master a simple thing like writing node equations you have indeed mastered circuit theory; rest is all mechanical and can be taken care of by programs like Matlab and Maple.