The Octave Program.

Introduction

If you're serious about performing Radio-related numerical calculations, particularly on networks, I recommended downloading the `Octave` program in the August/September "Morseman" column in *Break-In*. It's almost completely compatible with the `Matlab` program used widely in industry and University. However, `Matlab` starts at $2450 (US), while Octave is completely free. It has built-in support for complex numbers, matrices, plotting, string handling, and can be run either in "immediate" mode as a sophisticated calculator, or run from source files, which can call user-written functions. While commands are interpreted, the built-in functions are compiled and run very fast.

But it doesn't run in a nice "Windows-like" environment, the user interface looks more like `DOS`. This is a small price to pay for the immense computational power that you get.

Download `Octave` from

`http://www.gnu.org/software/octave/download.html`

You'll get an installation file, about 71 Mb. Run it, and the program will install in the default folder. At writing, the current version was 3.2.4.

Wikipedia gives a history of the program with a short instruction summary here:

`http://en.wikipedia.org/wiki/GNU_Octave`

A complete list of all the built-in functions is here:

`http://octave.sourceforge.net/octave/overview.html`

Short tutorials are available here:

`http://en.wikibooks.org/wiki/Octave_Programming_Tutorial/Getting_started`
`http://volga.eng.yale.edu/sohrab/matlab_tutorial.html`

A complete 575 page on-line `html` manual is downloadable here:
`http://www.gnu.org/software/octave/docs.html`

However, you don't need to download this, since

- Entering `help command <enter>` on the command line shows abbreviated help on any command,
- Extensive help files are included in the distribution you installed. If you installed in the default folder, you'll find links to both `html` and `PDF` versions if you navigate here:

`C:\Octave\3.2.4_gcc-4.4.0\doc\octave`

Starting Octave

1. Navigate to the folder

   `C:\Octave\3.2.4_gcc-4.4.0\bin`

   and find the file `octave-3.2.4.exe`. It's convenient to make a shortcut to this on the desktop.

2. If you're running `Windows 7`, you'll need to run in "`XP-compatibility`" mode. Select this by right-clicking on the `Octave` icon and following instructions.

3. Double-click either the program name or the shortcut. `Octave` should start in a "`DOS`-like" command window.

4. Change the working folder in this window using the standard "`cd 'my working folder'` command, where the path specifies some sensible new folder you've chosen - you don't want to work in its home folder. The `Octave` command window is rather primitive. You can't (in this version at least) cut and paste into it with standard "`ctl-x`" and "`ctl-v`" editing commands. But right-click on the top (blue) bar and move down to "`edit`", then "`paste`". Now anything you've previously copied will be pasted into the window.

Running the Simple Program Supplied.

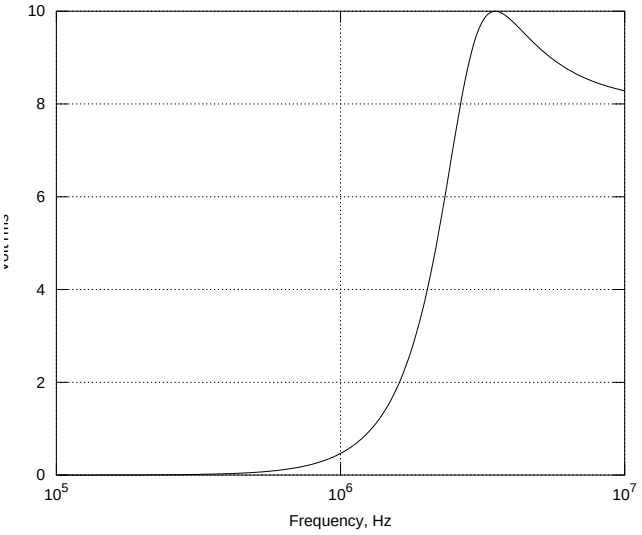
The sample program that I gave in the column is shown below.

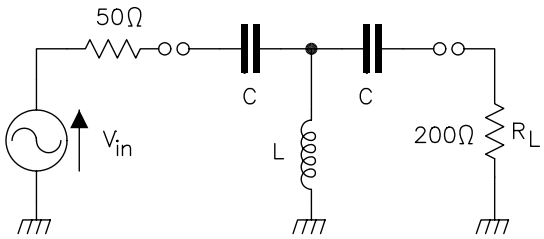
```
% zin designs, plots response
% of a Tee matcher.

vin = 10; R = 200; % Load resistance
X = sqrt(50 * R);
L = X/(7*pi*1e6); C = 1/(7*pi*X*1e6);
f = logspace(5,7,200);
zc = 1./(2*j*pi*f*C);
zL = 2*pi*j*f*L;
z1 = 50 + zc;
vt = vin * zL./(z1 + zL);
zt = zc + (zL.*z1)./(zL + z1);
vL = R * vt ./(R + zt);
semilogx(f,abs(vL),'k');
xlabel('Frequency, Hz');
ylabel('Volt rms'); grid;
```

Normally you (see later) enter programs with `Octave`'s editor. You can either do this with the program above, cut and paste it into a text file from this listing, or download it as `zin.m` from my web-page. Put it in your working folder.

Having done this, a standard `Dos` "`dir`" command should show you the program `zin.m` in your working folder.

Now enter "`zin`" (the "`m`" is unnecessary) followed by "`<enter>`". This should run the program and produce the plot below left.





This is a plot of the magnitude of the output voltage across the load resistance R_L in the circuit at right above, where $v_{in} = 10\,\mathrm{V}$ *rms* and the capacitors and inductors have been chosen to give a maximum power match at 3.5 MHz.

This sample program is very condensed, but column space required succinctness. For an expanded program with comments, using any load value and matching frequency, with power loss estimation, email me[1]. (The column said I'd upload this too, but I decided to wait to see if anyone wants it.)

You can export the plot in several forms. enter "`help print`" to see the options. For example, you can capture it as "`figure.jpg`" with the command

"`print -djpg figure.jpg`".

The file should appear in your working folder. You can now paste it into `Word` or other documents in the standard way. You'll see that you can also produce `gif`, `Postscript` files, and a variety of others. Experiment to find what suits you and your word processor.

Some Plotting Hints.

- You may find font sizes on the plot too small. You can change the axis label font by inserting `set (0, "defaultaxesfontsize", 18);` before the plotting call.
- Change the label font sizes by including them in the "`label`" call:
  `xlabel('Frequency, Hz','fontsize',18);` Similarly for the `y` axis label.
- You can change to a finer plot grid by replacing the `grid` call with `grid("minor");`
- You can re-size the plot by dragging a corner or a side.

Explanation of the Program.

- The first two lines are comments. Anything after a "`%`" symbol is treated as a comment.
- See the circuit diagram at right on the previous page: Line 3 sets the input source voltage as $10\,\mathrm{V}$ *rms*, and the load resistance as $200\,\Omega$.

[1]My email address is on the NZART website, or you can send directly to `garyzl1an@gmail.com`

- I've used the "equal reactance" design algorithm. Line 4 finds "`X`", the magnitude of all three reactances. But remember that capacitive reactance is *negative*! We allow for this in line 7.

- Line 5 finds values of the inductor and capacitors, rearranging the standard equations $|X_L| = 2\pi\omega L$ and $|X_c| = 1/2\pi\omega C$.
  The factor `7*pi*1e6` means $7 \times \pi \times 10^6$. `Octave` knows that "`pi`" in an expression means π. This stands for $2\pi fC$, where $f = 3.5 \times 10^6$ Hz, the desired matching frequency. Expressed smybolically,

$$z_c = \frac{1}{2\pi jfC} \qquad z_L = j2\pi jfL$$

- Line 6 sets the frequency range of the calculation. It generates a vector `f`, 200 values long, with values logarithmically spaced between 10^5 Hz and 10^7 Hz.

- Lines 7 and 8 generate values of capacitor impedance, `zc`, and inductor impedance, `zL`, at *each* of the 200 frequencies specified in line 6. No looping construct is needed, since `Octave` generates a value corresponding to each element in `f`! It also knows that the symbol "`j`" means "the imaginary operator", so that `zL` has a value in quadrature with the real axis, and `zc` in negative quadrature. This "`j`" operator converts the reactance values into impedance values.

- We now perform a Thevenin transformation on the network[2]. First, replace the source, left-hand capacitance, and inductor, with an equivalent single voltage source and single series impedance. Line 9 forms `z1`, the sum of the source resistance and the first capacitor's impedance. This sum is a complex quantity, but `Octave` understands complex numbers. Again, it computes 200 values of `z1`. Symbolically, putting $\omega = 2\pi f$,

$$z_1 = 50 + \frac{1}{j\omega C}$$

- Line 10 uses the voltage division theorem[3] to find the 200 values of the equivalent Thevenin source. Note the "dot-star" division operator in the second term on the eight-hand side. It means "divide each value of the numerator vector with the corresponding value of the denominator vector. So it does 200 complex divisions, forming another complex vector, `vt`.

$$\begin{aligned} v_t &= \text{Thevenin source voltage,} \\ v_t &= V_{in}\frac{z_L}{z_1 + z_L} \quad \text{where } z_1 = 50 + \frac{1}{j\omega C} \end{aligned}$$

- Line 11 Finds the equivalent Thevenin impedance, by adding the source resistance to the parallel impedance of `z1` and `zL`. This is the impedance seen looking "back" from the load resistance into the network, with the source voltage short-circuited. (Yes, my students have trouble applying Thevenin's theorem correctly too. See the *Wikipedia* reference again.) Symbolically,

$$z_t = z_c + \frac{z_L z_1}{z_L + z1}$$

- Line 12 uses the voltage division theorem again to find 200 values of the voltage across the load, using the Thevenin equivalent circuit we have just calculated.

$$v_L = v_t\frac{R}{R + z_t} \qquad \text{where } R = 50\,\Omega$$

[2]See the *Wikipedia* reference on "Thevenin's Theorem" if you don't know what this is.
[3]See the *Wikipedia* reference to "voltage division theorem"

- Line 13 plots the graph of load voltage versus frequency. "`semilogx`" means "plot with a logarithmic x axis and a linear y axis". The plot window will be automatically scaled to include all frequency and voltage values. Move the mouse across the plot. The co-ordinates of the mouse position, at bottom left, change as you move. Draw out a rectangular region with the right-hand mouse button down, and right-click again. The plot will zoom to show only the region selected.
- Lines 14 and 15 label the axes. See below for a more fully commented code, which allows arbitrary frequency, load resistance, and estimates the approximate power loss.

```
% zinexpanded designs, plots response
% of a Tee matcher.
% gejb 6 August 2010

% get the variables:
vin = input("Source voltage?  "); R = input("Load resistance?  ");
ff = input("Frequency in MHz?  ");
ff = ff * 1e6;      % convert to frequency in Hz
Q = input("Q of inductor?  "); Q
% perform the "equal reactance" calculation:
R1 = 50;            % Source resistance
X = sqrt(R1 * R);

% perform the approximate power loss calculation
R0 = R1 + R;        % characteristic impedance
Pin = vin^2/200;    % Input power to network
PL = Pin/Q * (sqrt(R0/R1 - 1) + sqrt(R0/R - 1));
printf("\n Power into network is %3.2f Watt",Pin);
printf("\n Approximate power lost is %3.2f Watt\n",PL);
printf("Percentage power lost is %3.2f\n",PL * 100/Pin);

% find the inductor and capacitor values:
L = X/(2*pi*ff); C = 1/(2*pi*ff*X);
printf("\n C = %4.2f pF",C*1e12);
printf("\n L = %4.2f uH\n",L*1e6);

f = logspace(5,7,200);  % create frequency vector
w = 2*pi*f;             % and angular frequency vector

% create impedance vectors
zc = 1./(j*w*C); zL = j*w*L;

% Make the Thevenin transformation
z1 = 50 + zc; vt = vin * zL./(z1 + zL);
zt = zc + (zL.*z1)./(zL + z1);
vL = R * vt ./(R + zt);

% plot the graph, with a finer grid.
semilogx(f,abs(vL),'k'); xlabel('Frequency, Hz');
ylabel('Volt rms'); grid("minor");
```