XMUT203 Analogue Electronics and Systems

Demo 4: First- and Second-Order Circuits

1. Introduction

In this demo, we will cover two important topics in the circuit analysis and design e.g. frequency response simulation and component tolerance simulation in LTspice.

2. Introduction to Frequency Response

Frequency response of a device or a circuit describes its operation over a specified range of signal frequencies by showing how its gain, or the amount of signal it lets through changes with frequency.

Generally, the frequency response analysis of a circuit or system is shown by plotting its gain, that is the size of its output signal to its input signal, output/input against a frequency scale over which the circuit or system is expected to operate. Then by knowing the circuits gain, (or loss) at each frequency point helps us to understand how well (or badly) the circuit can distinguish between signals of different frequencies.

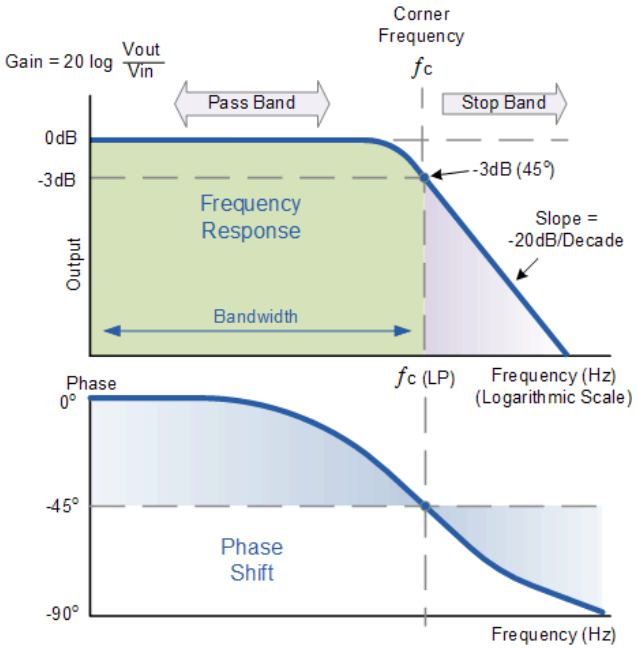

Figure 1: Frequency response (Bode) plot of a low-pass filter

The frequency response of a given frequency dependent circuit can be displayed as a graphical sketch of magnitude (gain) against frequency (ƒ). The horizontal frequency axis is usually plotted on a logarithmic scale while the vertical axis representing the voltage output or gain, is usually drawn as a linear scale in decimal divisions. Since a systems gain can be both positive or negative, the y-axis can therefore have both positive and negative values.

In electronics, the logarithm, or "log" for short is defined as the power to which the base number must be raised to get that number. Then, on a frequency response plot, the logarithmic x-axis scale is graduated in log10 divisions, so every decade of frequency (e.g. 0.01, 0.1, 1, 10, 100, 1000, etc.) is equally spaced onto the x-axis. The opposite of the logarithm is the antilogarithm or "antilog".

2.1. Bode Plot

Graphical representations of frequency response curves are commonly called Bode Plots and as such plots are generally said to be a semi-logarithmic graph because one scale (x-axis) is logarithmic and the other (y-axis) is linear (e.g. log-lin plot) as shown in the figure below.

Bode plots are graphical representations of the circuit frequency response characteristics and as such can be used in solving design problems. Generally, the circuits gain magnitude and phase functions are shown on separate graphs using logarithmic frequency scale along the x-axis.

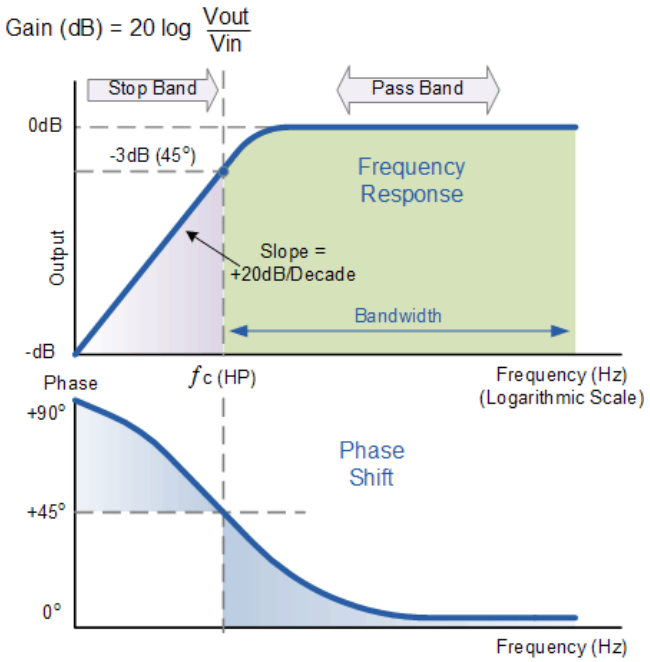

Figure 2: Frequency response (Bode) plot of a high-pass filter

2.2. Bandwidth

Bandwidth is the range of frequencies that a circuit operates at in between its upper and lower cut-off frequency points.

$$\text{Bandwidth } (BW) = f_H - f_L$$

Most amplifiers and filters have a flat frequency response characteristic in which the bandwidth or passband section of the circuit is flat and constant over a wide range of frequencies.

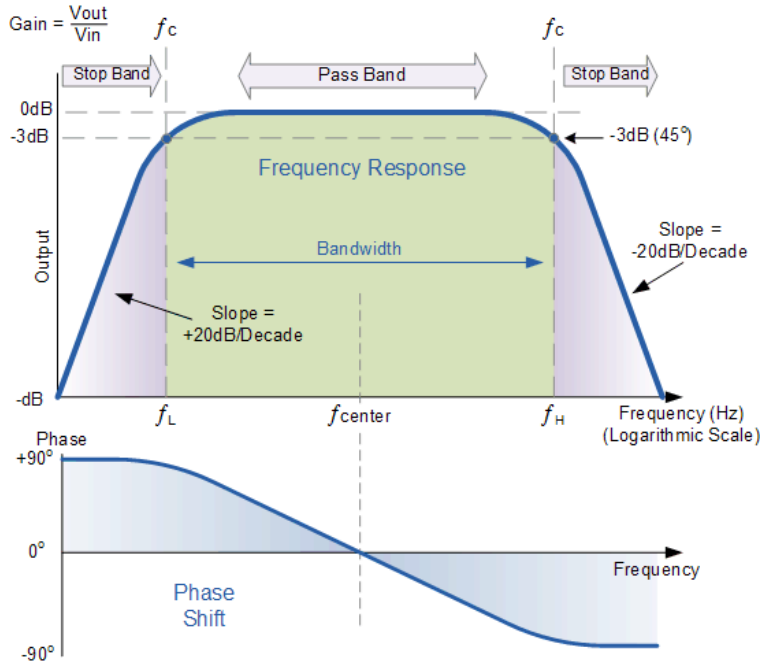

Figure 3: Frequency response (Bode) plot of a band-pass filter

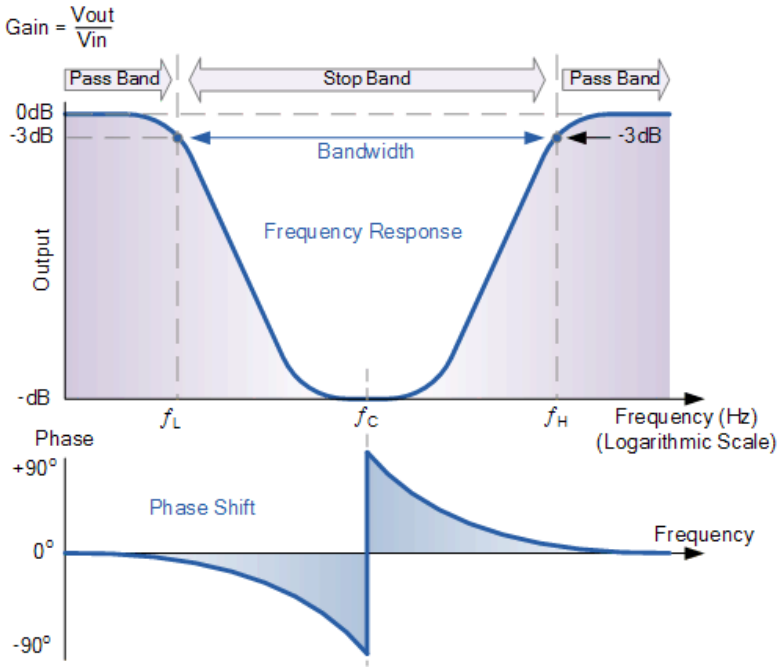

Figure 4: Frequency response (Bode) plot of a band-stop filter

2.3. Q-Factor

Resonant circuits are designed to pass a range of frequencies and block others. They are constructed using resistors, inductors, and capacitors whose reactance vary with the frequency. As shown below, their frequency response curves can look like a sharp rise or point as their bandwidth is affected by resonance which depends on the Q-factor of the circuit, as a higher Q-factor provides a narrower bandwidth and sharper cut-off.

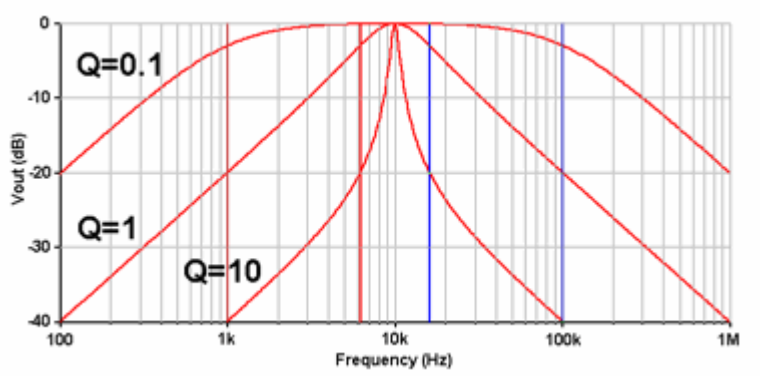

Figure 5: Q-factor of the band-pass filter circuit

2.4. Half-Power Points

These cut-off or corner frequency points indicate the frequencies at which the power associated with the output falls to half its maximum value. These half power points correspond to a fall in gain of 3 dB (0.7071) relative to its maximum dB value.

$$-3\ \text{dB} = 20\log(0.707)$$

The -3 dB point is also known as the half-power points since the output power at this corner frequencies will be half that of its maximum 0 dB value as shown in the figure below.

$$P = \frac{V^2}{R} = I^2R$$

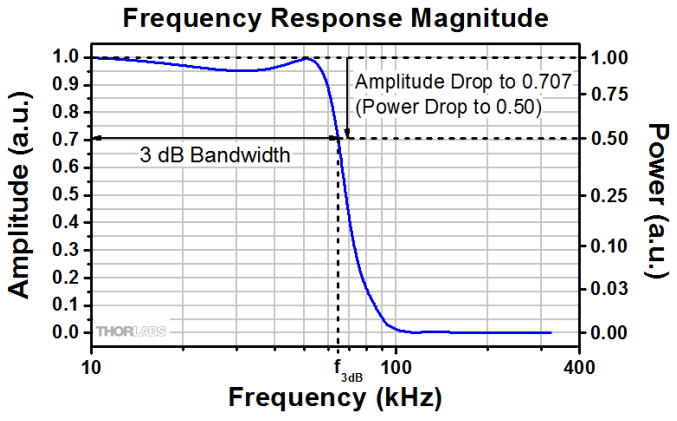

Figure 6: Half-power point in the frequency response

At lower cut-off frequency (f_L) or upper cut-off frequency (f_H), V or I = 70.7% of maximum V or I respectively. If R = 1, then the power is:

$$P = \frac{(0.707 \times V)^2}{1} \quad \text{or} \quad (0.707 \times I)^2 \times 1$$

As a result, the power is:

$$P = 0.5 \times V \quad \text{or} \quad 0.5 \times I$$

Therefore, the amount of output power delivered to the load is effectively "halved" at the cut-off frequency and as such the bandwidth (BW) of the frequency response curve can also be defined as the range of frequencies between these two half-power points.

While for voltage gain, we use 20log10(Av), and for current gain 20log10(Ai), for power gain we use 10log10(Ap). Note that the multiplying factor of 20 does not mean that it is twice as much as 10 as the decibel is a unit of the power ratio and not a measure of the actual power level. Also, gain in dB can be either positive or negative with a positive value indicating gain and a negative value attenuation.

Exercise 1 – Frequency Response

SPICE type programs like LTspice excels at frequency response analysis of passive networks containing resistors, capacitors, inductors, and crystals. For example, consider a ladder-like passive low-pass filter circuit as shown in the schematic given in the figure below.

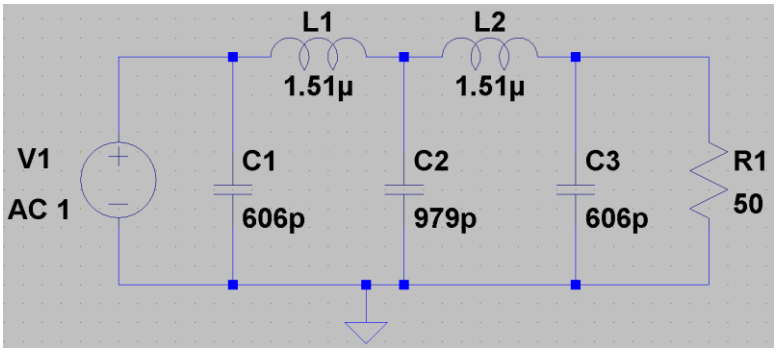

Figure 7: Ladder-like passive low-pass filter circuit

This exercise has a slightly less detailed step-by-step procedure for creating and simulation the circuit in LTspice, since you have done in the previous introduction exercise and should have learned a few things.

We are drawing a ladder type filter circuit schematic, typically used as the output of a transmitter. Two horizontal inductors are the "backbone" and three capacitors are "legs" extending vertically down from the backbone to ground. A source is on the left and a load resistor on the right.

1. Press 'L' for inductor, then control-R to rotate, and place two inductors in a line, about two or so grid dots apart.

2. Press ‘C’ and place a capacitor with its top terminal just below the left terminal of the left inductor, a second capacitor with top terminal just below the area between the two inductors, and a third with top terminal just below the right-hand inductor.
3. Press ‘R’ and put a resistor just to the right of the right-most capacitor.
4. Press F2, choose ‘voltage’, and put a voltage source left of the circuit with top terminal near the left terminal of the inductor.
5. Press ‘G’ and put ground terminals below the voltage source, the three capacitors, and the resistor.
6. Press F3 and wire left to right, source-to-inductor, inductor-to-inductor, and inductor-to-resistor. Wire the center capacitor to the junction of the two inductors and the end capacitors to the ends of the inductors just above them. Connect each ground to the device above it. Press Escape.
7. Assign values to each component by putting the cursor over it and right clicking, as follows: Resistor, 50. Outside capacitors 606p, inner capacitor 979p, inductors 1.51u. Voltage source, click advanced, then in the small signal AC analysis blocks, put AC in amplitude and 1 in phase.
8. Label the output node. Press F4, type ‘out’ in the box and OK. Then put the dot on the wire at the top of the right-hand resistor. Press Escape.
9. Go to the Menu, Simulation, and Edit Simulation. Select the AC analysis tab. Leave at Octave type simulation. Enter 20 points per octave, 700E3 start frequency, 70E6 stop frequency, and OK. Click on the page to drop the simulate command.
10. Press Run (running man icon).
11. Now double-click on V(out) in the box. Maximize the plot if it is not already full size. Now you see the plot of the filter response, which is fairly flat at 0 dB attenuation from 700 kHz to around 7.05 MHz and then slopes down to -93 dB attenuation at 70 MHz.

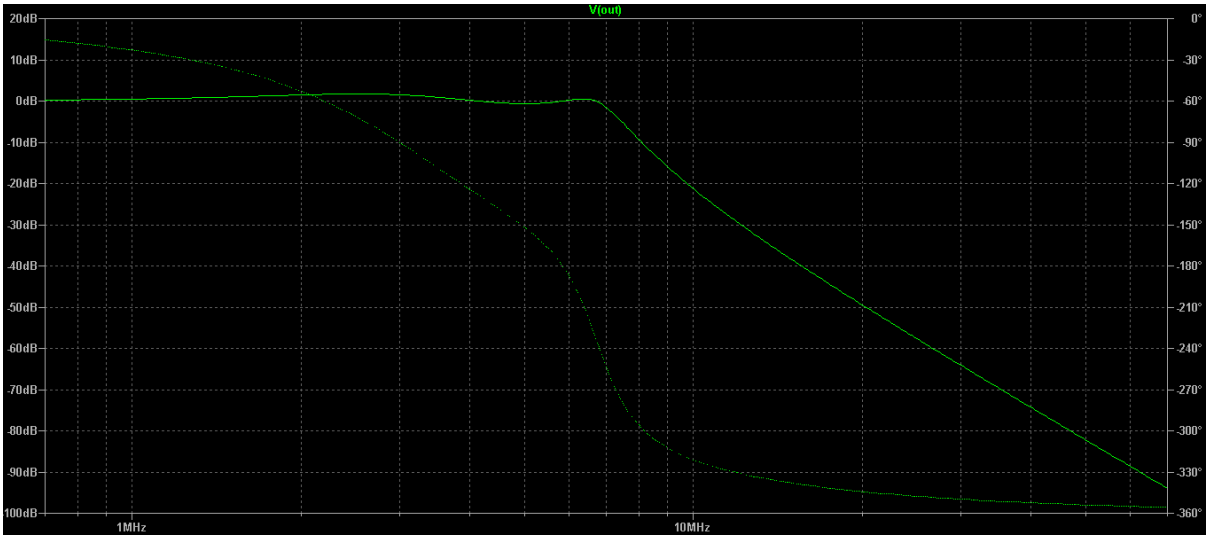

Figure 8: Frequency response of the low-pass filter circuit

12. An experiment. You would like to add more attenuation at 21 MHz by putting a parallel resonant capacitor across one of the inductors. You would like to know if this would cause problems in the passband or elsewhere. Put a 37 pF capacitor across the left inductor and run the simulation again.

If you save your circuit while a plot is still open, the plot data will also be saved. Some of the data files can get very large, so consider this when saving and close the plot window before saving if you wish.

2. Introduction to Component Tolerance

LTspice is a handy tool for evaluating circuits without having to use breadboard and through its "directives". It provides a powerful method for analyzing how a circuit might perform with components exhibiting real-world tolerances.

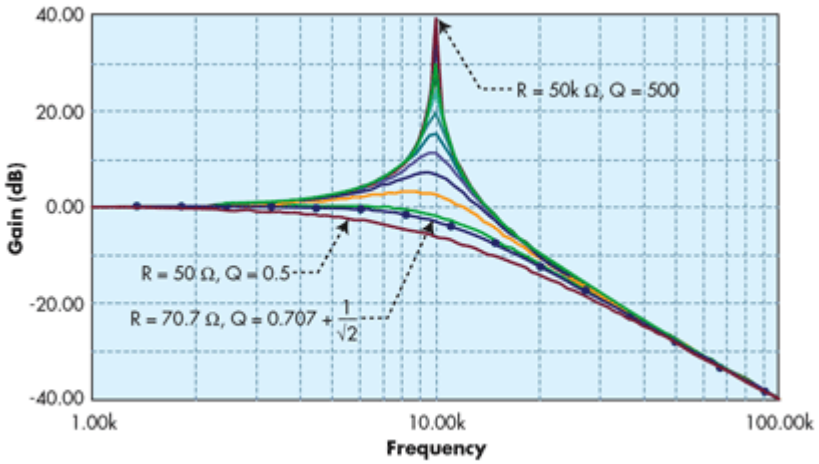

Figure 9: Effects of variation of the component values on frequency response profile of a circuit

As shown in the diagram given above, the frequency response of a given filter circuit changes as the value of the component i.e. the resistor, R in the circuit is varied.

Notice that for low values of resistor (i.e. 50 Ω and 70.7 Ω) the type of frequency response of the system is overdamped, but at high value (i.e. 50 kΩ) the frequency response of the circuit at the cut-off frequency has a peak indicating that it becomes underdamped.

Notice also different values of resistor, R in the circuit change the quality factor (Q -factor) of the filter circuit (i.e. from 0.5 at 50 Ω, 0.707 at 70.7 Ω to 500 at 50 kΩ).

2.1. Monte Carlo Analysis in LTspice

One such method of "real-world" analysis is Monte Carlo analysis, which, with each new analysis run, randomly varies parameters (within their user-defined limits) to give the user a useful picture of actual circuit performance.

However, as a circuit designer, often it is required to evaluate the worst-case performance of the devices or circuits. That is, we want to know how a circuit performs at the extremes of component values, to ensure that we meet whatever design specification we are designing to.

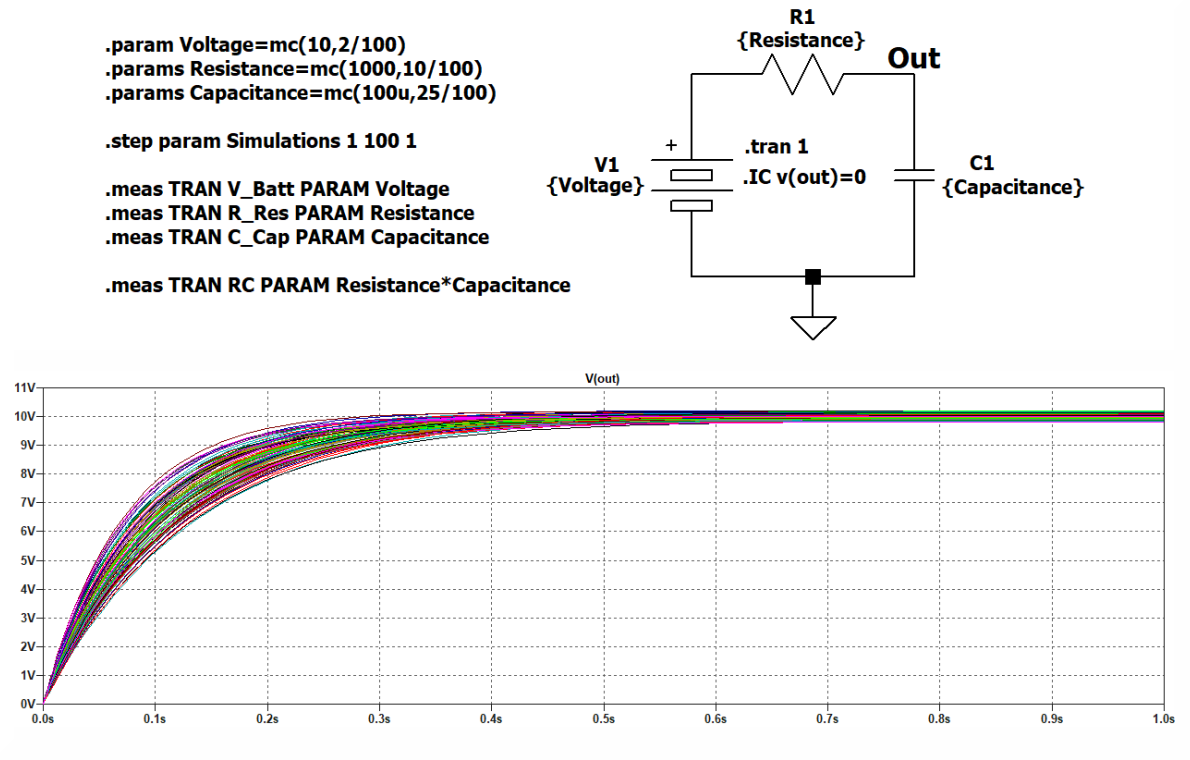

Figure 10: Monte Carlo analysis in LTspice

Although Monte Carlo analysis can tell us what the performance is at these limits, if the circuit contains many components, it can take quite a lot of runs before its random selection of parameter values happens to simultaneously correspond to the worst-case limits of all the components. It is quite possible that we never see the true worst-case limits i.e. after all it is a matter of chance.

Nevertheless, LTspice does indeed have a pre-defined Monte Carlo function. This is the "`mc`" function, and a search of the Help Topics for "`mc`" will point to the `.param` topic, and under this heading we find the function `mc(x,y)`, which, when invoked, returns a "random number between `x*(1+y)` and `x*(1-y)` with uniform distribution."

To use this function, rather than define a resistor's value as, say, 10K, we define it as:

```
{mc(10K,0.05)}
```

Where: `10K` is its nominal value, and `0.05` is its tolerance (5%). An example will follow, after the next section.

2.2. Worst Case Analysis in LTspice

To truly evaluate performance at a circuit's worst-case limits, we can perform a "worst-case" analysis in lieu of a Monte Carlo analysis. This analysis has an added advantage, too, in that not as many runs are required to ensure that we have truly evaluated all the components over all possible variations in

their tolerances. After all, we do not need to measure *between* the tolerance limits (which is what Monte Carlo analysis does).

For example, if one of the components is a 10 kΩ resistor with a 5% tolerance, worst-case analysis will only use resistance values of 10.5 kΩ and 9.5 kΩ for this component and not any other value between these limits, whereas Monte Carlo analysis would randomly use any value between, and including, these limits.

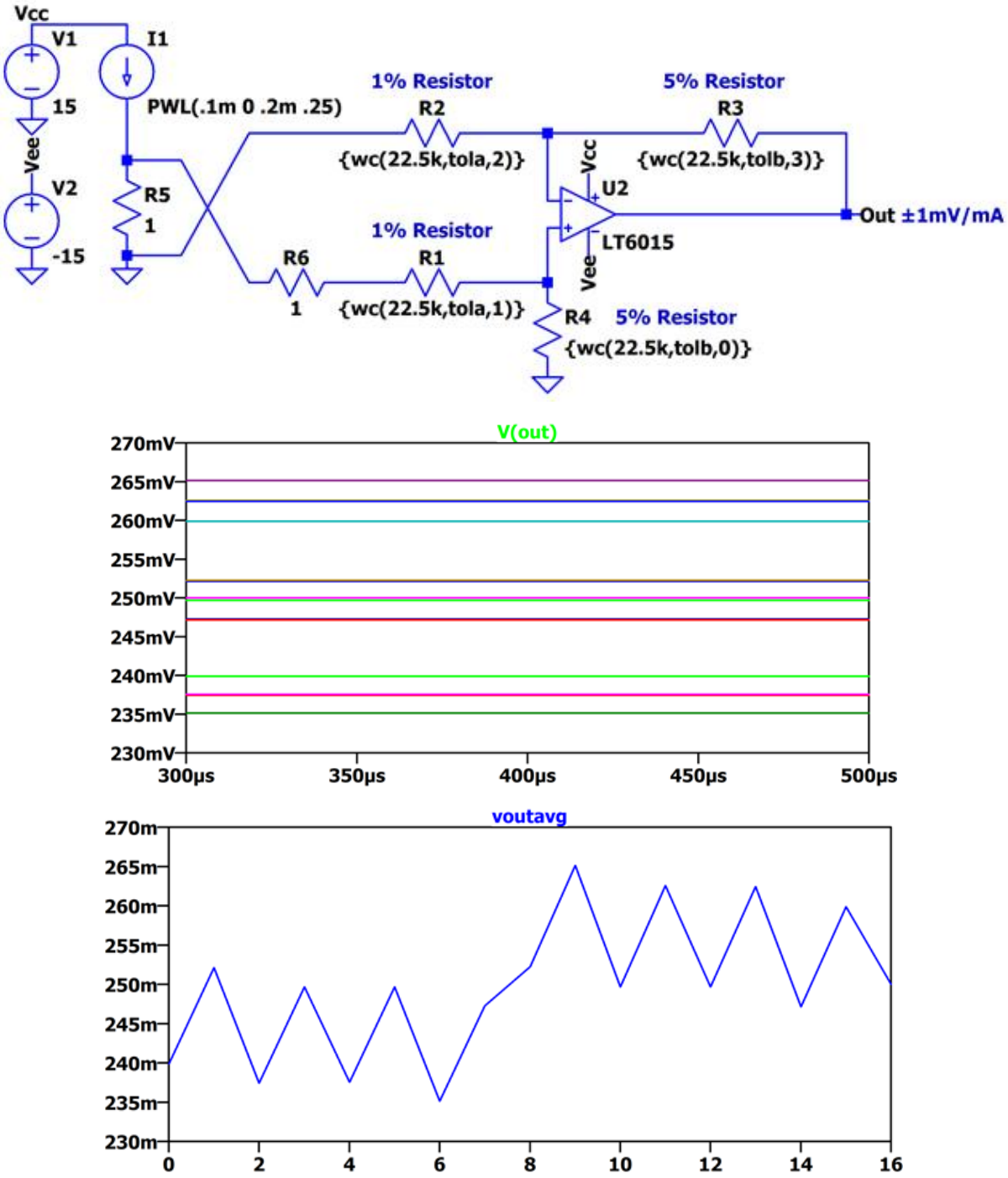

Figure 11: Worst-case analysis of an op amp-based amplifier circuit in LTspice (schematic diagram – top; transient response – middle; worst-case value simulation results - bottom)

The predefined `mc` function handles Monte Carlo analysis, but there is no pre-defined "worst case" function. We need to create this function ourselves, but it is not too difficult to do.

This can be done using SPICE's "`.function`" directive. The following code is an example of a function for worst-case analysis (we will use this later):

```
.function wc_a(nom,tola) if (run == 1, nom, if(flat(1)>0, nom*(1+tola),
nom*(1-tola)))
```

In this definition:

- `.function` is the SPICE directive for defining a function.
- `wc_a` is the name we give this instance of worst-case function.
- `nom` is the "nominal" value (e.g. 10K for a 10K resistor).
- `tola` is the tolerance (defined elsewhere with a `.param` directive (e.g. 0.1 for a 10% tolerance)).
- `run` is a variable (defined elsewhere in the `.step` directive) identifying the current run count).
- `flat(1)` is a Spice function that returns a random number between -1 and 1.

  Note that the definition of `.function` and `flat` can both be found in LTspice Help.
- So how do we use this function that we have just defined?

Take as an example a 10 kΩ resistor. Normally, we label its value in the LTspice schematic as "`10K`". However, to vary its value between its worst-case values, we instead use a more complex label for its value. Rather than entering "`10K`", in this case we enter the following into the component's value field (note the use of the curlicue brackets):

```
{wc_a(10K,tola)}
```

So, what happens when we run the analysis?

If this is the first analysis run (`run` is defined elsewhere in the `.step` directive and simply is the current run being performed), then, because `run=1`, the function returns the value assigned to `nom`, in this case, 10K.

But for each new run after the first run, and for each component defined with a `wc_a` function, the function `flat(1)` is re-evaluated for that component.

If the random result returned from `flat(1)` is greater than 0, then the tolerance is added to the component's nominal value (e.g. for a resistor whose nominal (`nom`) value is 10 kΩ, if `tola` is 0.1 (10% tolerance), the resistor's value is set to 11 kΩ).

Otherwise, the tolerance is subtracted from the component's nominal value (e.g. the 10 kΩ resistor is set to 9 kΩ).

Exercise 2 - Tolerance of Components

1. The diagram below shows a voltage divider with parallel capacitor circuit used for performing Monte Carlo and Worst Case analysis.

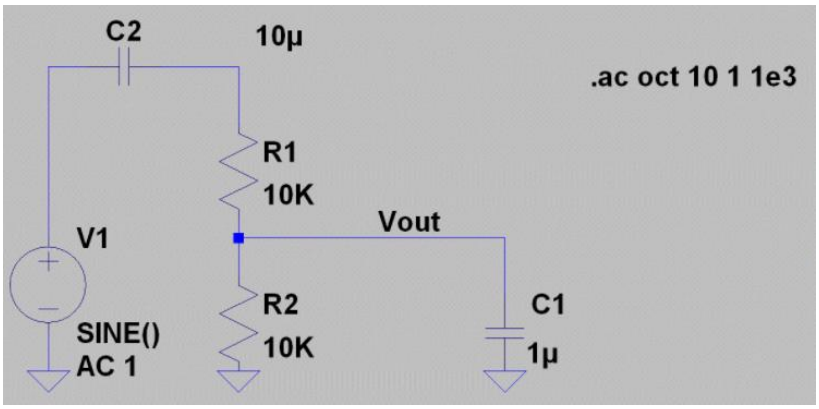

Figure 12: Voltage divider with parallel capacitor circuit

2. Given the component values in the circuit above, a frequency sweep of the input from 1 Hz to 1 kHz shows that the circuit has the following gain and phase transfer function when measured at its Vout node as shown in the plot below.

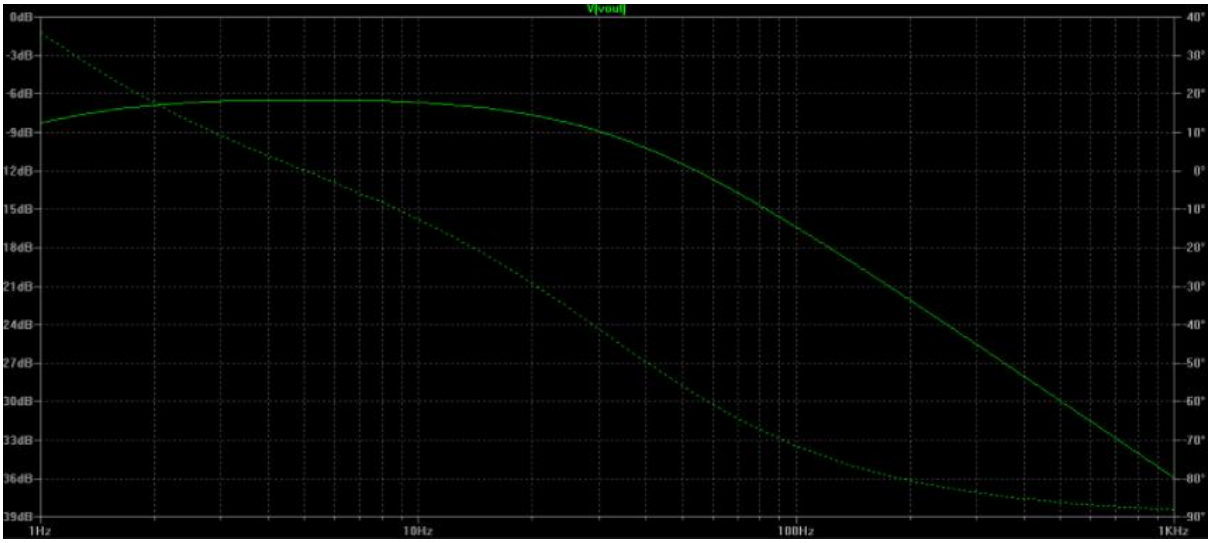

Figure 13: Frequency response of the voltage divider circuit

But what happens when the component values vary over their tolerance range? Suppose that the resistors have 10% tolerance, and the capacitors have 20% tolerance. We can perform a **Monte Carlo** analysis on this circuit, given these tolerance values.

3. The diagram below shows the same circuit, but now set up with its Monte Carlo functions and `.param` directives.

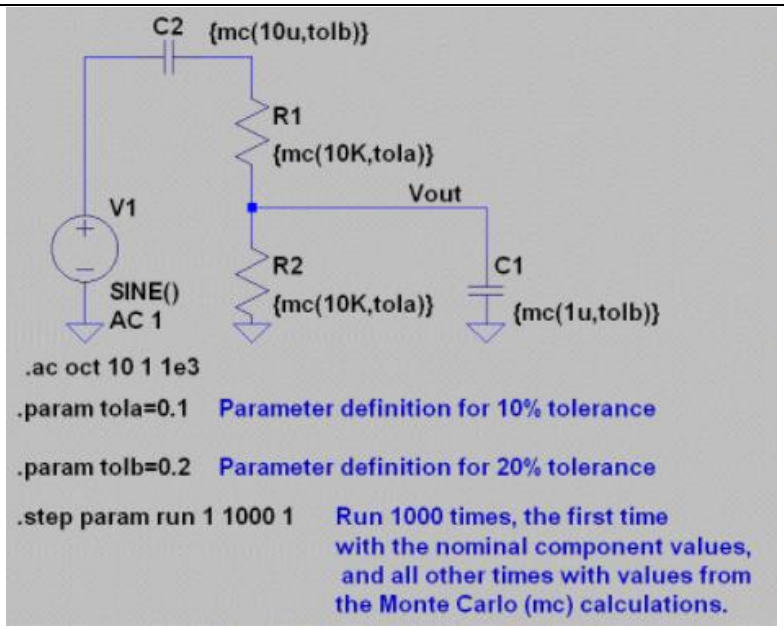

Figure 14: Schematic diagram of the voltage divider circuit for Monte Carlo analysis

Note that within the "`mc`" function, we are not setting the tolerance field to an actual number (although we could have done it this way, too). Instead, we are using a separate directive (`.param`) to define the tolerances (in this case, 10% and 20%) globally.

4. Running the Monte Carlo analysis 1000 times (via the directive "`.step param run 1 1000 1`") gives us the following spread of gain and phase plots:

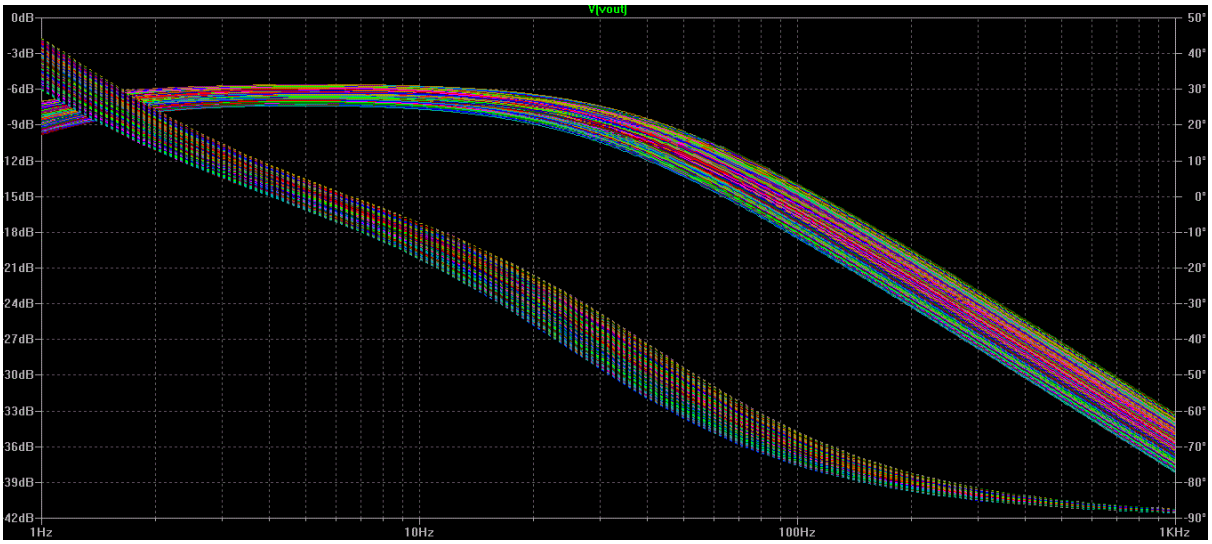

Figure 15: Frequency response of the voltage divider circuit with Monte Carlo analysis

But we cannot be sure that we have truly evaluated worst-case performance. So, we instead use our new "worst-case" function for a worst-case evaluation.

Because there are two component tolerances for the components in the schematic (the resistors are 10% and capacitors are 20%), we will define two worst-case functions: `wc_a` and `wc_b`.

- The function `wc_a` will vary the value of those components whose tolerance is labeled "`tola`" (in this example, `tola = 10%`).
- Similarly, the function `wc_b` will vary the value of those components whose tolerance is labeled "`tolb`" (in this example, `tolb = 20%`).

5. The schematic of the circuit, with its new SPICE directives and functions, now looks like shown in the diagram below.

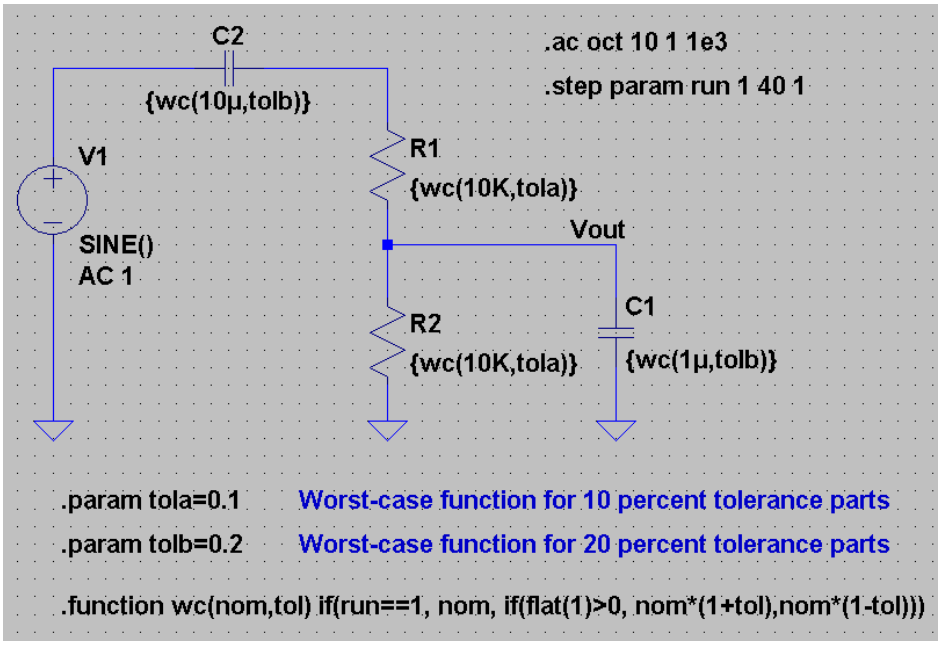

Figure 16: Schematic diagram of the voltage divider circuit for worst-case analysis

6. The analysis output, after 40 runs, looks like as shown in the plot below

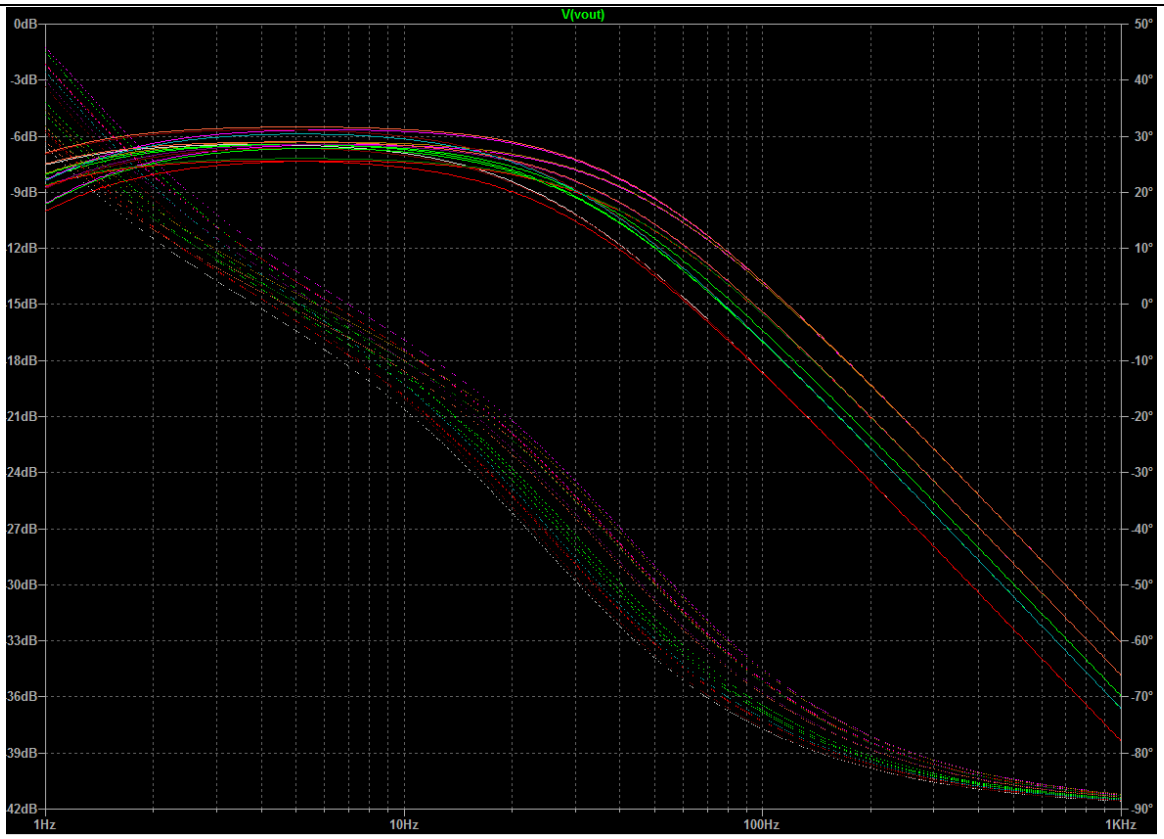

Figure 17: Frequency response of the voltage divider circuit for worst-case analysis

Note the discrete intervals between plots. This is because the worst-case analysis is only using component values that are at the +/- tolerance limits for each component, and not any intermediary values (except for the first of the 40 plots, which uses the nominal component values for its analysis).