ELEC 341 – Graded Assignments

Assignment A-1

20 Marks

Learning Objectives

- Modeling White Box System
- Impulse Response
- Natural Response
- Digital Noise & Filter Design

- Matlab
  - minreal()
  - lsim()
  - impulse()
  - step()
  - plot()
  - legend()
- Simulink
  - Approximate Impulse
  - Scope Formatting

When you design a circuit from scratch, you have full information about what's inside and can compute the system model analytically. This is called a "White Box" system.

An "Integrated Circuit" is a complex circuit represented by an elementary symbol, for the sake of simplicity. Many ICs have a response time that is much faster than the rest of the circuit and may be treated as "ideal". Since you are only modeling part of the system (the dominant part), this is called a "Grey Box" system.

Calc 1 2 mark(s) CCT Analysis

Compute the transfer function of the Voltage Amplifier circuit.
Treat the Op-Amp as an ideal component.
Use **minreal()** to cancel any common factors.

- C1_tf = Transfer Function (V/V)

Calc 2 2 mark(s) Envelope

Use Matlab to plot the Impulse response.
Find the EXPONENTIAL envelope by computing the Inverse Laplace Xform, or estimate it by trial & error.
Your envelope should have the form K exp(-a* t)

- C2_K = scalar constant term
- C2_a = exponential term

Fig 1 2 mark(s) Matlab Impulse Response

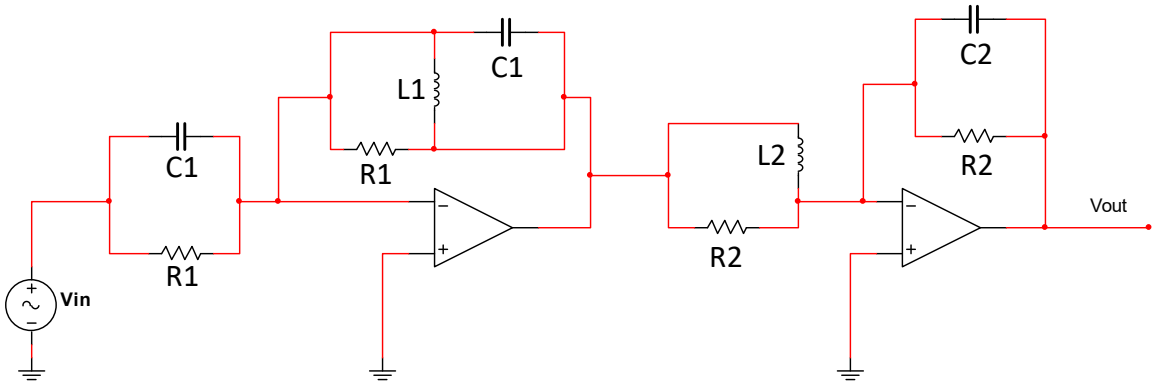
Plot the Impulse response, final value and envelope.

- Time scale in msec
- LineWidth = 3 (all of the following)
- Curve: Solid red
- Final Value: Dashed black
- Envelope: Dotted blue

Voltage Amplifier Circuit

$R1 = 5 \times \#A\ \Omega$
$L1 = \#B\ mH$
$C1 = \#C\ \mu F$

$R2 = 5 \times \#D\ \Omega$
$L2 = \#E\ mH$
$C2 = \#F\ \mu F$



In Matlab, the impulse() function is used to generate an impulse response. But Simulink does not have an "Impulse" block to do the same. It has a "Discrete Impulse" block, but you already discovered in the practice homework that it isn't the same thing. Since an impulse is IMPOSSIBLE to create in practice (even inside a computer), even Matlab only approximates it. The Step block may be used to do something similar in Simulink.

Calc 3 2 mark(s) Simulink Impulse Response

Use Simulink to approximate the Impulse Response of the Voltage Amplifier.

Use the **Sources / Step** block to create an approximate impulses. Start with a pulse width of 1 sec and reduce it by a factor of 10 until further reduction makes no noticeable difference. Make sure the area under the curve is always correct.

- C3_pw = largest practical pulse width that approximates an impulse (sec)

Fig 2 2 mark(s) Simulink Impulse Response

Adjust the y-axis of the Simulink Scope window to match the Matlab plot.

Use **View/Configuration Properties/Display/Y-limits**.

If you don't do this, the auto-scale feature may zoom out too much to see the interesting parts.

- LineWidth = 3 (all of the following)

Aside from the missing envelope, you should see little difference between the Matlab & Simulink plots.

Another way to get an impulse response in Simulink is to use the derivative of the step function. Of course, there is no derivative block, but as long as the order of the denominator of your transfer function is higher than the order of the numerator, you can include the derivative in the transfer function by adding a zero at zero. It's mathematically equivalent but you do have to remember that you modified it in case you decide to use it for some other purpose in the future.

Fig 3 2 mark(s) Alternative Impulse

Add to your Simulink system a copy of the same transfer function, but add a zero at zero in the numerator.

Apply a step input to the modified transfer function.

Plot the original and new "Impulse" responses in the same Scope window.

- Use 2 very different trace colours to make both easily visible.
- Use **View/Style** to increase all trace widths to 3.

Did this approach work ??? Did you get what you expected ???

What are the physical units of your new transfer function ???

Do these units make physical sense ???

To simulate an Impulse in Simulink, you used a pulse with a high voltage that you could never produce from a micro-controller & amplifier circuit. Adding a "derivative" is not easy to do a physical system either.

*The only reason we use an impulse is because it is "Mathematically Equivalent" to raising a system to an initial condition, removing the driving function, and seeing how the system "**Naturally**" responds. So why not just do that!*

Calc 4 1 mark(s) Scaled Envelope

Use Matlab to generate a signal that starts high (1V) at t=0 sec, goes low (0V) at t=1 sec, and stays low for **20 msec**. Use **lsim()** to apply it to the Voltage Amplifier transfer function. Plot the response.

Adjust the envelope to enclose your natural response.

The exponential term should not change if your natural response is right.

- C4_K = scalar constant term

Fig 4 2 mark(s) Natural Response

Plot the response and envelope.

Start the time axis at 1 sec or you won't see any detail. You don't care what happens before then anyway. That's just the time spent raising the system to an initial condition.

- Time axis begins at t=1 sec
- Same format as Fig 1

Are the voltages reasonable ??? Could a micro-controller generate this curve ???

Any control system uses a sensor to close its control loop. A common source of noise is the discretization of the sensor signal which is either done by the sensor itself, or by the ADC of the micro-controller. This particular form of noise is a real nuisance because it makes computing an exact derivative impossible.

You can use a filter to smooth the corners but the smoother the signal (GOOD), the greater the delay (BAD). Designing a filter means finding the best compromise.

Calc 5 2 mark(s) Filter Design

Use **A1_ExpDataPlot.p** to generate experimental data (**CF = 200**). The digital data is also returned. Design a **UNITY-GAIN**, **FIRST-ORDER** filter to smooth out the corners without introducing too much delay or losing too much detail.

- C5_p = filter pole
- C5_tf = filter transfer function

Fig 5 3 mark(s) Filtered Signal

Plot the digital data, the filtered data, and the error (absolute value) between them.

- LineWidth = 3 (all of the following)
- Digital Data: Solid blue
- Filtered Data: Solid red
- Error: Dashed green
- Legend with labels “Digital Data”, “Filtered Data” and “Error”

How much is too much ???

Notice how a filter trades noise for delay ???