**Lab-4: Frequency domain analysis of RC filters**

In this lab, we shift away from time-domain analysis into frequency domain analysis of the RC circuits. Objectives are:

1. Investigate low-pass and high-pass RC filters.
2. Learn about the frequency representation of periodic signals.

**Design:**

**Step.1: How does an RC circuit behave at different frequencies?**

Consider the following RC circuit. Assume the input voltage is a sinusoidal wave that can be written as V2 = 5 sin(ωt).

1. Using the impedance method, find an expression (Vc = A(ω) sin(ωt + φ)) for voltage across the capacitor. Plot the transfer function A(ω) and the phase φ as a function of input frequency (complete the following plots).
2. Denote the break-frequency and slope of the transfer function.
3. What type of filter is this, and why? (low pass, bandpass, high pass, or band stop?)
4. What if we replace the position of the resistor and the capacitor? What type of filter then this would be?

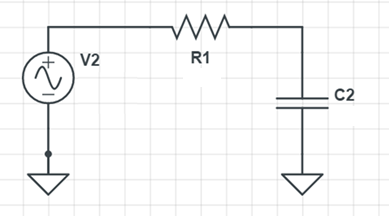


Figure.1. Schematics of an RC circuit

**Step.2: Circuits are linear…**

Let's get specific. Assume V2 = 5 sin(2π\*1t) + 3 sin(2π\*100t) + 1.5 sin(2π\*1000t). Knowing that the equations that govern these circuits are linear (superposition rule), you can easily find the system's total response using plots you derived in the last step. Note that our new input signal (V2) in the frequency domain can also be represented in its Fourier components, as seen in Fig.2.

1. Using these tips and the result of Step.1, derive the expression for the capacitor's voltage given the new input.

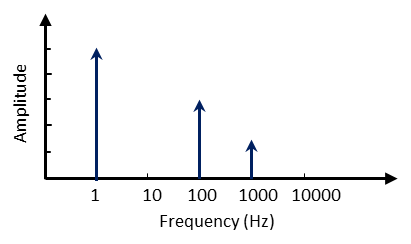


Figure.2. Frequency space representation of the input signal

**Step.3: The sinusoidal nature of square waves.**

Recall from the Fourier theorem that a square wave can be decomposed to a series of sinusoidal waves. (See this [video](https://www.youtube.com/watch?v=k8FXF1KjzY0&ab_channel=BrekMartin) for more intuition)

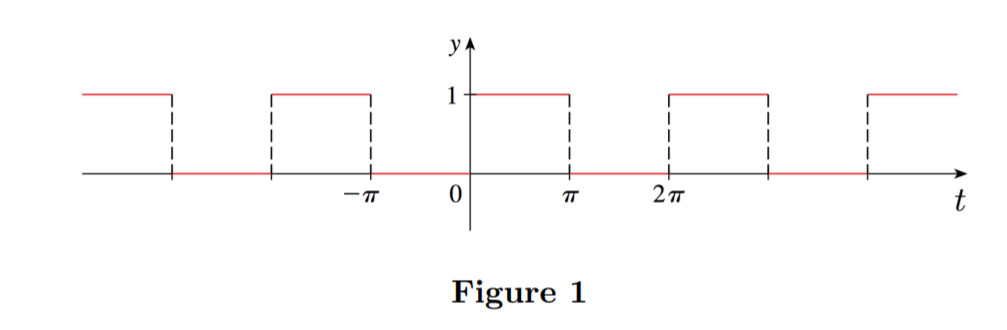
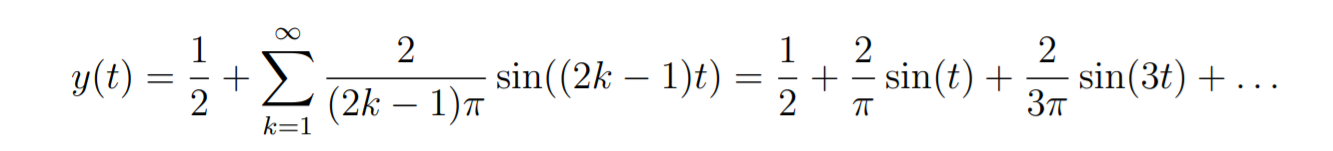


Figure.3. Square wave in time domain

For instance, in the above picture (with a frequency of w = 1 rad/s), the square wave can be written as: a sum of series of different sinusoidal waves:



This series of sinusoidal waves can be represented in the frequency domain (blue plot on the right) as visualized below:

Note that in the frequency domain, the x-axis represents the frequency of the sinusoidal component. The y-axis represents the amplitude of that component in the overall Fourier series.

Up until this point in our labs, we have been working with square waves generated from Arduino. However, we need to find a way to create sinusoidal waves to perform our future labs. As you just saw, the square waves can be decomposed to a series of sinusoidal waves. What if there was a way to filter out one of these frequency components? This is the core motivation behind the 10CL. Filter circuits designed by RLC elements can be used to isolate specific frequency components in any given signal. You have already seen a glimpse of this in Step.1, where you can see from the transfer function that your circuit allows low-frequency component (f < corner frequency) to pass while heavily suppressing the frequency components higher than the break frequency.

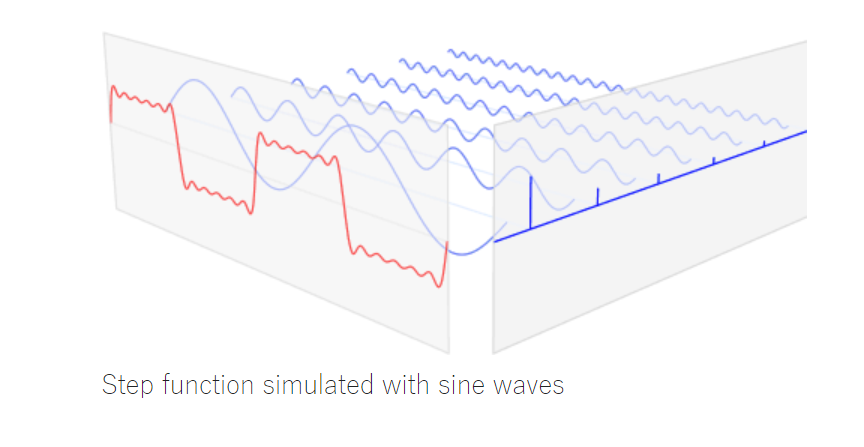


Figure.4. Square wave translated to frequency domain

Let's think about using the low-pass filter we analyzed in step-1 in a meaningful way. Consider an arbitrary square wave similar to the left-hand side of Fig.3 that translates to the following frequency components.

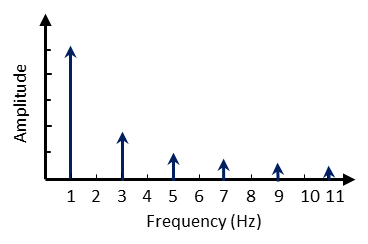


Figure.5. Frequency space representation of a 1Hz square wave

1. Draw the gain function of an **ideal** low a pass filter that isolates only a single sinusoidal component out of a square wave on the same plot above.
2. Let's move away from the ideal case. Given the transfer function of the RC low-pass filter that you analyzed in step-1. Where would you set the break frequency of this circuit to isolate a single sinusoidal component?

**Experiments:**

**Step.4:**

Setup the circuit shown in Fig.1a (R=1k, C=10uF). Use a square-wave signal, as discussed in previous labs. Measure the full swing (peak-to-peak amplitude) of the output signal across the capacitor as a function of the following input frequencies. Use these values to calculate A(ω) = Vpp\_cap/5v or gain function.

[Side note 1]: you can easily read the max and min of the voltage from the serial monitor. Remember to measure the minimum as well as the maximum of the signal.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *Frequency* | 2Hz | 10Hz | 20 Hz | 40Hz | 200 Hz | 2000Hz |
| *Swing* |  |  |  |  |  |  |
| *Gain* |  |  |  |  |  |  |

* Does the result of step.4 follows the step.1?

**Step.5: Let's try to make a bad sinusoidal wave.**

Following the last step, set the input frequency of your square-wave signal back to 10Hz. Note that a 10Hz square wave with a 5V swing centered around 2.5V can be decomposed to its frequency components as follows:

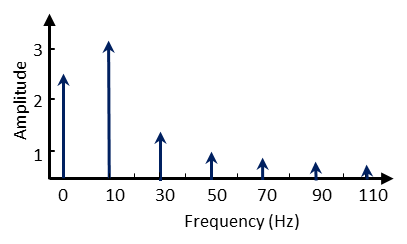


Figure.6. Frequency space representation of a 1Hz square wave

1. Scope and plot the output signal on the capacitor.
2. Calculate the break-frequency of the low-pass filter you configured in step.4.
3. Why is the output not perfectly sinusoidal?

**Step.6: Let's try to make a good sinusoidal signal.**

In later labs, we will study different circuit topologies that can isolate frequency components more accurately. However, for now, there is a simple way to generate a sinusoidal signal. Arduino is also capable of generating SPWM waves instead of simple square-waves. In an SPWM wave, the duty cycle of square waves varies according to a sine wave's amplitude. SPWM waves combined with RC passives allow a simple way to create almost pure sine waves. Let's try this method first. Integrate the following code to your Arduino template:

#include <PWM.h> //PWM library for controlling freq. of PWM signal

double angle = 0;

double increment = 0.02;

void setup() {

// put your setup code here, to run once:

InitTimersSafe();

}

void loop() {

double sineValue = sin(angle);

sineValue \*= 255;

int plot = map(sineValue, -255, +255, 0, 255);

analogWrite(5,plot);

angle += increment;

The code above generates a 2 hz SPWM wave on pin-5 of your Arduino ( This may not work on other pins). Run the code and Plot the SPWM signal. Input the SPWM circuit to the RC filter of step-4 and observe the voltage over the capacitor.

**Step.7:**

Now that we have been able to produce a pure sine wave from SPWM method. Let's use this signal to investigate other properties of RC filters. Construct the following circuit. (R1 = 1k, C1= 10uF, C2= 10uf, R\_pot = use potentiometer)

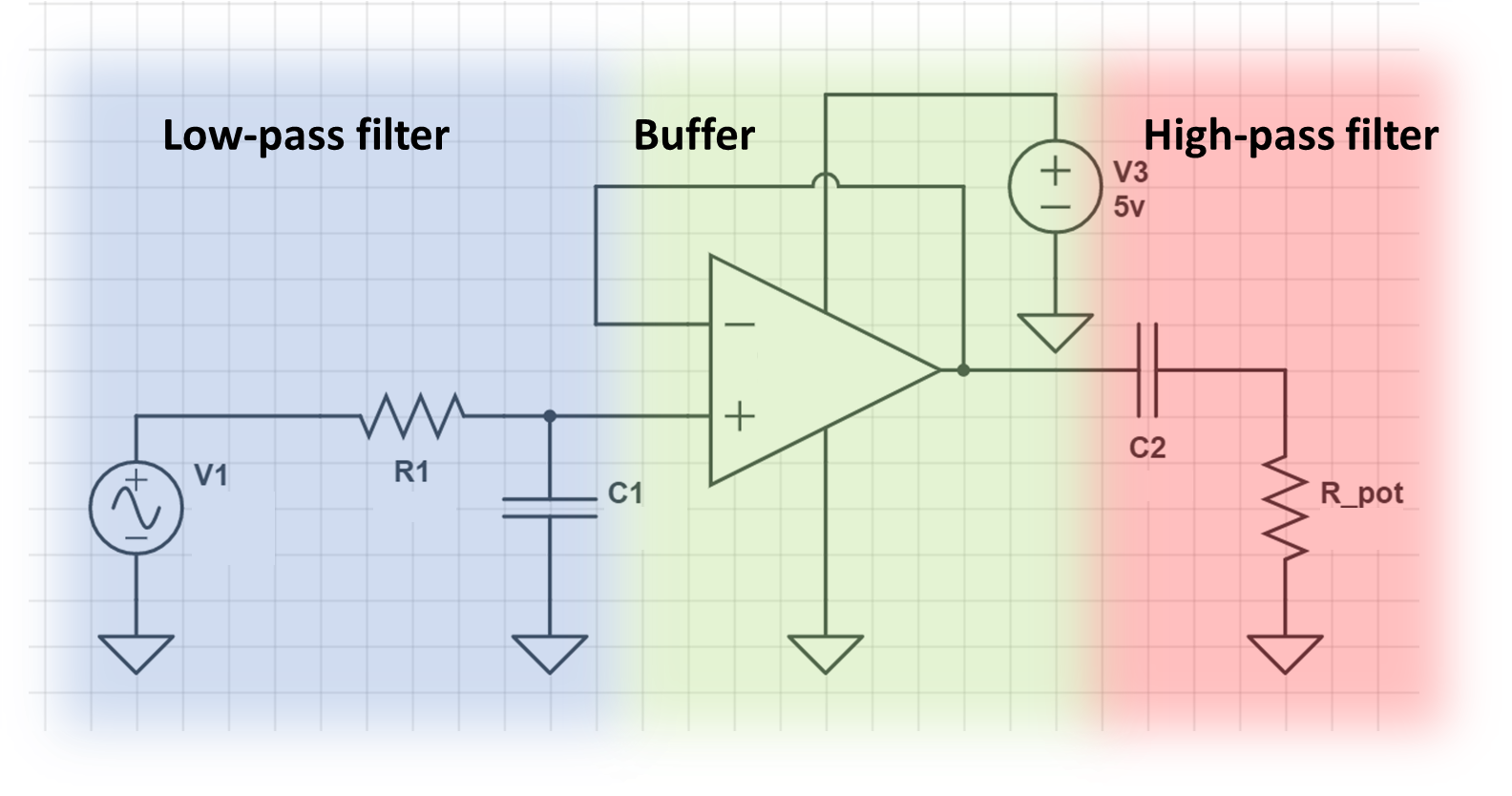


Figure.7. Low-pass RC + Buffer + High-pass RC setup.

You note that the first stage of this circuit (shown in blue) is responsible for generating a 2hz sine wave. This is the same as the setup in step.6. As you saw in lab-3, it is necessary to isolate the output/input of RC/RL/RLC filters with buffers to eliminate the loading effect of the next stage on the first stage's operation. Hence, we place a unity gain non-inverting buffer in the path of the first stage's signal. The final stage is an RC high pass filter (Vout is defined as voltage over the resistor). Note that for the resistor, here we will use a potentiometer in the same manner, which we utilized it in lab-3.

* Find the general expression for the transfer function of the RC high-pass filter.
* Scope and plot the output of the RC high-pass filter (voltage across the resistor) for an arbitrary value of R. Why is the sinusoidal wave centered around 0 and not 2.5v? [Side note]: Keep in mind that Arduino cannot scope negative outputs.
* Assuming that the frequency of your input signal is 2 Hz. Given the amplitude of the sinusoidal signal at the low-pass filter's output (first stage), tune the potentiometer such that the corner frequency of your RC high-pass filter is set at 2 Hz. [hint]: You need to find what is the gain of a RC high-pass filter at its corner frequency. Scope and plot the output signal of first (low-pass) and third (high-pass) when this condition is met.

* Given that the corner frequency of the high-pass stage is now set at 2Hz. Draw an intuitive transfer function of the overall circuit (Hint: think about what are the transfer functions of each stage independently). What type of filter is this (the overall circuit)?