ESE 150 – Lab 04: The Discrete Fourier Transform (DFT)

LAB 04

In this lab we will do the following:

1. Use Matlab to perform the Fourier Transform on sampled data in the time domain, converting it to the frequency domain
2. Add two sinewaves together of differing frequency using a summing OpAmp circuit
3. Use Arduino A2D to sample and quantize output of summing OpAmp

Bring headphones to lab.

Background:

In lecture we studied the Fourier Series and the Fourier Transform. The Fourier series is representation of a periodic function using a summation of sines and cosines. It offers a way to represent a time-based periodic signal in the frequency domain. The Fourier Transform is an extension of the Fourier Series that allows us to represent non-periodic functions using a summation of complex sinusoids. It allows us to take signals in the "time domain" and see their breakdown or "frequency domain" components. The Discrete Fourier Transform (DFT) is a variation of the Fourier Transform that applies when our function is discrete. This version of the Fourier Transform becomes very useful in computer engineering, where we have "digitized" incoming analog signals, taking them from a continuous form to a discrete form. In our case, we've sampled "music" using our Arduino A2Ds and as a result, we have a set of discrete sampled data. In this lab, we'll apply the DFT to our discrete sampled data to transform it from the time domain to the frequency domain and look more carefully at its frequency components.

In previous labs we've only sampled simple waveforms: sine wave, square wave, triangle wave. We could "convert" them to the frequency domain without doing the Fourier transform, they simply have 1 frequency, so only 1 sine wave could represent them. We need to create a more interesting waveform. So we'll begin the lab by using two function generators, each producing a separate sine-wave. Then we'll use a special adding/summing circuit to combine them together electronically. Next we'll sample this newly sampled data, import it into Matlab and apply the discrete Fourier Transform to it, so we can see the frequency components of our sampled signal, as opposed to its time-domain only representation.

ESE 150 – Lab 04: The Discrete Fourier Transform (DFT)

Prelab: Applying the Discrete Fourier Transform in Matlab

- In this section, we'll take the data you've collected in previous labs, convert it from the time domain to the frequency domain using the DFT
- We'll use a built-in function in Matlab to help us apply the DFT, called FFT()

Recall from lecture, the formula for DFT

$$F(n) = \sum_{k=0}^{N-1} x(k) e^{\frac{-j2\pi kn}{N}}$$

1. If you take ESE224, you will implement this formula in MATLAB by hand. However, MATLAB provides an implementation of this formula, so you don't have to worry about it for this class! (This is one of the reasons why many people use MATLAB). The implementation is called a FFT, or Fast Fourier Transform, because of the efficient algorithm for computation. You can read about the FFT in MATLAB here: http://www.mathworks.com/help/matlab/ref/fft.html
2. The basic idea is that it takes in N samples from the time domain, and determines the sine/cosine components at various frequencies: k

3. Begin by importing your sine wave data into Matlab as you did in Lab 3; create a 800x1 matrix called: sine_sampled_time [You don't need to sample this again; use the 5000Hz sampled data you captured from Lab 1.]

4. Using a conversion factor, convert the samples to their voltage values.

5. Plot the data against the appropriate time axis (you must turn in this plot), label all axis (voltage vs. time and the units, and title) – *zoom in so we can see 4 cycles.*

6. Because you applied an "offset" to the sine-wave on the function generator, before you sampled it, your plot ranges from 0-2V. Subtract the "offset" from your sampled data to show it going between –1 V to + 1 V.

7. Now, re-plot the data against the appropriate time axis (you must turn in this plot too), label all axis (voltage vs. time and the units, and title) - *zoom in so we can see 4 cycles.*

8. Now, convert the sine wave from the time domain to the frequency domain using Matlab's built in DFT by typing the following (*Make the sine data you are converting is in Volts!)*

```
sine_sampled_frequency = fft(sine_sampled_time)
```

What you will notice is 800 complex #'s are produced (see the real # + imaginary #'s). Why complex #'s? Recall Euler's identity, where $e^{i\vartheta}=\cos\vartheta+I\sin\vartheta$.

9. Now, let's prepare to plot the converted data in the frequency domain. Type the following:

```
samp_period = 0.0002+0.000125    % sampling period + analogRead()'s delay
samp_freq = 1/samp_period        % sampling frequency
samples = 800                    % # of samples

sine_sampled_frequency = abs(sine_sampled_frequency / samples)
```

- Notice in the above code, that our sampling period is not just 200uS; there is a delay associated with analogRead(). So our sampling frequency is actually a little less than 5000 Hz.
- ***What is that actual sampling frequency when you include this additional delay?***
- Also, notice the last line; it takes the "absolute" value of our frequency data. When one takes the absolute value of a complex #, we get its magnitude (like polar magnitude).

10. Let's plot the data…
    a. Recall, there are 800 elements in the sine_sampled_frequency matrix. These represent the SPIKES or magnitudes of the sine-waves at various frequencies.
    b. But what are the x-axis values? They will now be frequency!
    c. What is our range? Let's say 0 Hz to start with, but what about the upper bound?

    Use the actual samping frequency you determined in Step 9.

    d. Create a vector called: freqs = (0:799).
    e. Scale it so that the highest frequency is "samp_freq" (again, determined in Step 9).
    f. Plot your data by using `plot(freqs, sine_sampled_frequency)`

11. Interpreting the plot…
    a. You will see two spikes, one at approximately 300 Hz, and one ~2800Hz. Why is the high one false? [**Hint:** What's your actual sampling frequency from Step 9?]
    b. Change your "sine_sampled_frequency" matrix and cut off false frequencies.
    c. Also, notice the amplitude is ½ of what it should be? Double the amplitudes in the sine_sampled_frequency matrix; this is a byproduct of the absolute value function.
    d. Lastly, replot your data…does it line up with what you expected?
        i. You could adjust that value of "+0.000125" to figure out analogRead()'s exact delay.
    e. Make sure there is a title and axis labels with units.

12. Make your own Matlab FUNCTIONs plot_time() and plot_dft() that takes in "800 time domain samples" and plots them in the time domain (subtracting the offset voltage offset appropriately) and in the frequency domain, respectively.
    a. This is the Matlab code that must be turned in for this section.
    b. You will also need to use this function in lab (and in Lab 6).
    c. Your two functions should take in arguments as specified below:
        i. plot_time(***time domain samples, figure number, start time, end time***)

            Specifically, ***start time*** and ***end time*** let you plot within a time range, so that you don't have to zoom in manually as you did in Step 5. But you need to do some calculations here according to the sampling frequency to get the correct entries in your time domain samples.

        ii. plot_dft(***time domain samples, figure number, sampling frequency***)

            This function should generate the frequency domain plot.

    Note: ***figure number*** should be used as "figure(***figure number***)" when you open up a figure window. This argument will be useful in Lab 6. Your functions should work for any size time domain samples vector, not just 800. Use the MATLAB length function to determine the number of samples in the input vector.

13. Perform a dot product in MATLAB to extract specific frequencies.
    While we used the FFT in MATLAB above, it is effectively computing the dot product between each of the frequencies and the sampled data. To see this, do the following:
    a. Create a vector that contains the time-sample coefficients for a 300 Hz sine sampled at the sample frequencies identified in Step 9 (possibly refined in Step 11d.i).
        i. Use your equation developed from Lab 2 postlab to create this vector in Matlab. **But make sure to leave it in "voltage" with offset 0 and 800 data points.**
        ii. To do this, first create a vector t = (0:799). Then, scale this vector by the sample period you calculated in step 9 above. To apply a function to t, you can do $y = sin(w * t)$ (substitute this with your own equation). y would be the vector with time-sample coefficients.

    b. Perform a dot product between that vector and the sine_sampled_time vector.
        i. To perform dot product of two vectors v1 and v2, use **dot(v1, v2)**.
        ii. What result do you get?
    c. Repeat a and b for a 300Hz cosine.
    d. Repeat a and b for a 200Hz sine and cosine.
    e. Repeat a and b for a 400Hz sine and cosine.
    f. Report all results and relate to what you know of the input signal and the FFT plot.

Lab Procedure:

Lab – Section 1: Adding two sinewaves together on the scope

- In this section we'll use two function generators to produce two sine waves of different frequencies
- We'll view the sine waves on the oscilloscope

1. We will set up the function generator we used in previous labs:
    a. Select a 300Hz sine wave with 4Vpp and 2V offset.
    b. Set the output to high-Z.
    c. Turn on the output, attach a BNC to BNC cable, and plug it into Channel 1 on the oscilloscope.
2. For the second wave, we will use the OLD looking function generator on your lab station, called "Hewlett Packard 33120A".
    a. Turn the function generator on.
    b. To set high z:
        i. Enter the menu: shift, then Enter buttons
        ii. Use the right arrow to navigate to D: SYS MENU.
        iii. Select by hitting the down button, then hit the down button again on "Out Term".
        iv. Use the left arrow to select high z, then press Enter.
    c. Set the generator to a 600Hz sine wave with 2Vpp and 1V offset.
        i. Press the buttons corresponding to wave parameters, and use the arrows or dial to change the number.
    d. The output should be automatically on. Attach a BNC to BNC cable, and plug it into Channel 2 on the oscilloscope.
3. Turn on the oscilloscope and view your waves!
    a. Adjust the scaling (try pressing "autoscale") so that you see the two waves.
    b. Save this image using Excel.
4. Now we will use the oscilloscope to visualize the sum of these waves!
    a. On the right side of the menu, press the Math button.
    b. Select the + operator and add Source 1 and Source 2 on the oscilloscope screen.
    c. Manually rescale the three displayed waves, and take a screenshot.
5. Keep your function generators on for the next part of the lab!

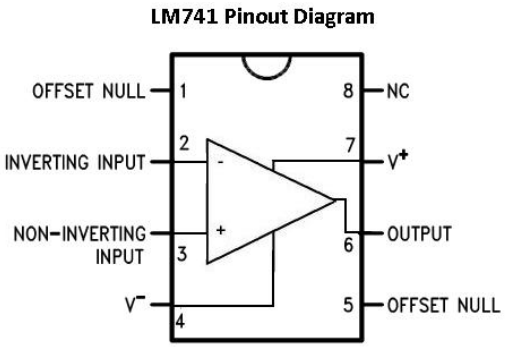
Lab – Section2: Adding two sinewaves together with a circuit

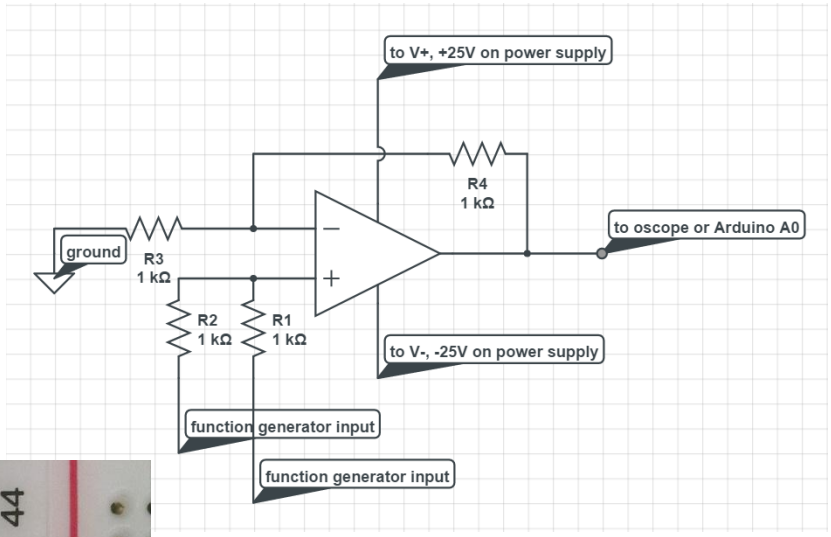
- In this section, you'll build a "summing amplifier" circuit to add the two sine waves together
- Finally, you'll sample the resulting "summed" sine wave using your A2D

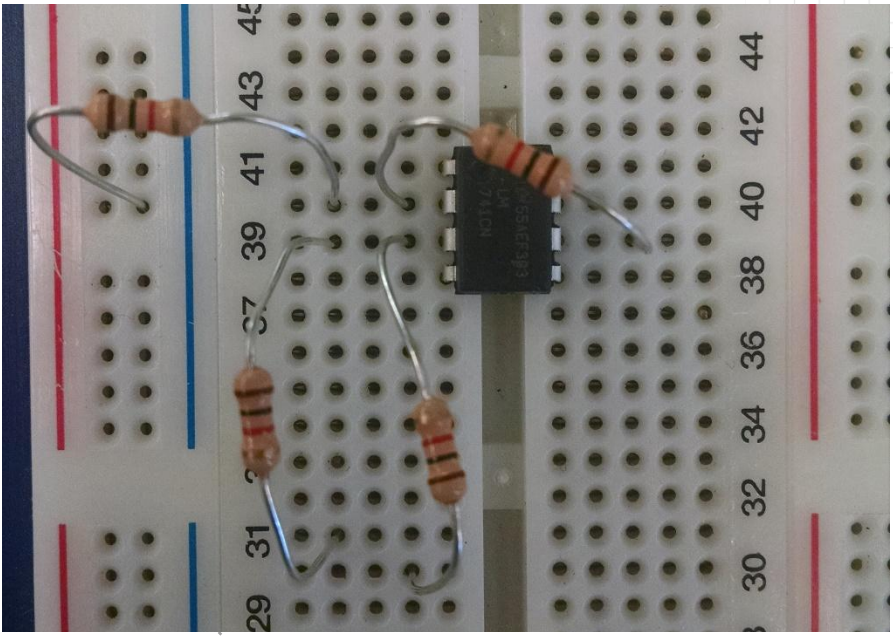
1. Obtain an LM741 OpAmp Circuit from the equipment supply (or your TA).
    a. Also obtain 4 resistors of size: 1 kΩ

Note: For more information on LM741 OpAmp please read the Appendix (Optional).

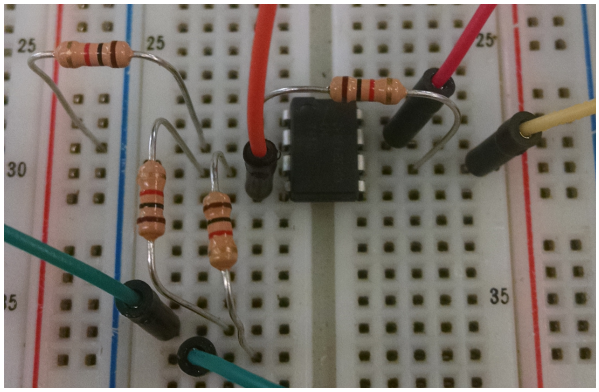
2. On a breadboard, attach the OpAmp circuit across a divet with the indentation facing upwards (pay attention to orientation). Attach the resistors to the OpAmp as seen in the circuit diagram below. Use the LM731 diagram to match the pins to the op amp inputs and outputs. Consult the photo below to make sure the resistors are set up correctly.
    a. Make sure the orientation of your OpAmp chip is correct!
    b. Note that the resistors connecting from the non-inverting terminal are not connected to the same point of the breadboard. These will be connected to different inputs.
    c. Don't worry about wires yet, we'll do that in the next step!



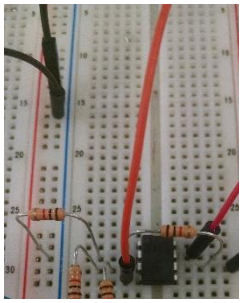




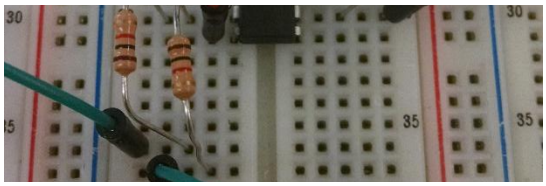
3. Now we must provider "POWER" to the OpAmp. We'll need to use the power supply on the lab bench. Turn it on and set the 25+ to 12V output, and set the 25- to -12V output.
    a. Turn on the power supply and output.
        i. Set the 25+ output to 12V, and the 25- output to -12V.
    b. Connect the power supply to your circuit.
        i. Use banana grabber cables to connect the 25+ output to a wire, and plug it into the V+ on your op amp (see circuit diagram above).
        ii. Connect the 25- output to V- on the OpAmp.
        iii. You should have the 25+ going into where the red cable is the photo below, and the 25- where the orange cable is in the photo below. The RED cable is on the left in the diagram.



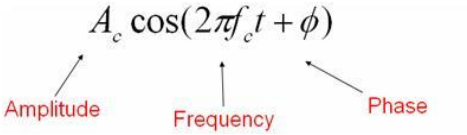
        iv. Connect the a ground wire to the black "COM" output on the power supply, and wire to the "blue" column of your breadboard (where the black wires are displayed below).



4. Next, we will attach the function generator inputs:
    a. Replace the BNC to BNC connections from the last section with a BNC to grabber cable on both function generators.
    b. Attach the black grabbers to the ground column (where the black wires are displayed in the image above). Make sure this goes in before the next step.
    c. Attach each of the red grabbers to one of the input resistors by placing a wire in the same *row* as the resistor (where the green wires are in the below reference image). It does not matter which output goes to which resistor!



5. Let's check our output on the oscilloscope:
    a. Remove the BNC cables from the previous section.
    b. Attach a BNC to grabber table to channel 1, and attach the grabber to the output of your summing circuit (where the yellow wire is in the image above).
    c. Hopefully your circuit is has successfully added two voltages together! (Be proud -- usually it takes more than a month of a circuit's class (ESE 215) to build a circuit like this!)
    d. Adjust the scale if necessary and see your wave! Ask a TA if you are unsure that it is correct
6. Finally, you will explore the idea of "phase." In class, we talked a lot about a signal's frequency, and the idea of a frequency domain. However, recall that the formula for an arbitrary wave includes a *Phase* term.



    a. On the Agilent function generator, change the phase parameter, and see how the shape of the wave depends partially on phase
7. Lastly, we will listen to the output here to see that the sound does not depend on phase
    a. Use your headphones for this section.
    b. The output from the op amp (pin 6) is connected to either side of the audio jack which is shown below. The center pin is grounded.

c. Now change the phase and check the sound each time. Make a note of your observation.

Lab – Section 3: Capturing, Importing to MATLAB, and Plotting in the Time Domain

- In this section you'll use your A2D to capture the output of the summing amplifier
- Afterwards, you'll import the data into MATLAB

1. Use your Arduino to sample the "mixed" 300 Hz + 600 Hz sinusoid (see Lab 1 for help!)
    a. Take the output from the output of your summing amplifier.
2. Import the 800 samples into Matlab:
    a. Use your Matlab functions: plot_time() and plot_dft() to plot the signal in time and in frequency.
    b. Properly label and turn in the plots.
3. Show your time and frequency plots to your TA and answer a few questions. This is the Lab Exit Check-off.
4. Make sure both partners have access to the data collected before leaving lab. We recommend setting up a shared folder on Google drive.
5. Time permitting: experiment with other frequency pairs.
    a. What waveforms can you create? Make sure you set your sample rates to avoid aliasing.
    b. Deliberately set one of the frequencies so that aliasing occurs. Note the alias frequency that shows up in your frequency plot.
6. Cleanup your lab station, leaving everything as you found it when you arrived.

ESE 150 – Lab 04: The Discrete Fourier Transform (DFT)

Postlab

First, download the three "unknown/mystery" signals from the link on the syllabus and import the 3 different sets of samples into Matlab.

1. Create plots of the provided data in the time domain, just like you have done with your own sampled data in labs:
   - Note, these samples are already scaled to voltage values
   - the sampling period for this data was 0.00002s
   - please plot only the first 200 samples
   - make sure to label axes and title your plot

2. Create plots of the provided data in the frequency domain. Use the MATLAB code provided in the lab, with a few changes
   - change the sampling period to 0.00002s
   - the length of the provided data is no longer 800. If you correctly coded your plot_time() and plot_dft() functions, they should work for the longer data. If not, this is a chance to test and refine them so that they do.
   - change the title for each mystery save file.

3. Write down the functions that sum up to make each mystery wave. They will all be in the form:
   $A\sin(2\pi ft)$
   You should find A and f from the plots created in step two.

Report the A and f values for the functions that make up each of the mystery waves.

Turn in all plots created **and** your resulting functions.

ESE 150 – Lab 04: The Discrete Fourier Transform (DFT)

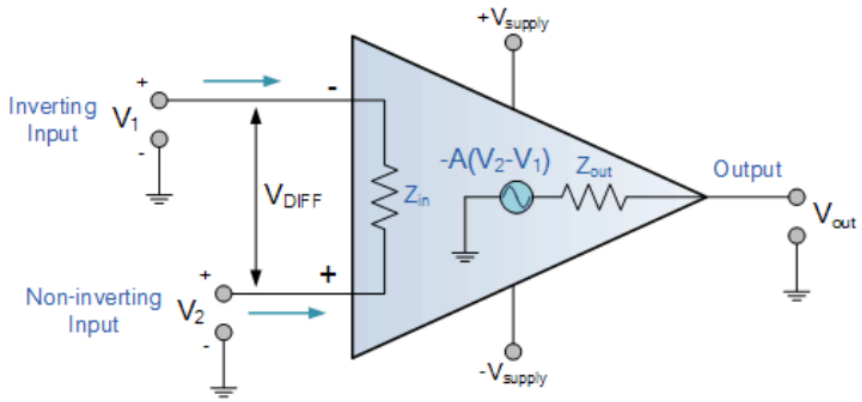
HOW TO TURN IN THE LAB

- Upload a PDF document to canvas containing:
  - All Plots with axis and labels and titles!
  - All Matlab code – just the function you created in prelab is sufficient
  - Saved oscilloscope screenshots
  - Answers to all questions in the lab
  - Plots and answers to postlab

APPENDIX

Op-amps :

An **Operational Amplifier**, or op-amp for short, is fundamentally a voltage amplifying device designed to be used with external feedback components such as resistors and capacitors between its output and input terminals. These feedback components determine the resulting function or "operation" of the amplifier and by virtue of the different feedback configurations whether resistive, capacitive or both, the amplifier can perform a variety of different operations, giving rise to its name of "Operational Amplifier". Hence, op-amps are used to perform mathematical operations such as addition, subtraction, integration and differentiation



An *Operational Amplifier* is basically a three-terminal device which consists of two high impedance inputs, one called the **Inverting Input**, marked with a negative or "minus" sign, (–) and the other one called the **Non-inverting Input**, marked with a positive or "plus" sign (+).

The third terminal represents the operational amplifiers output port which can both sink and source either a voltage or a current. In a linear operational amplifier, the output signal is the amplification factor, known as the amplifiers gain (A) multiplied by the value of the input signal and depending on the nature of these input and output signals, there can be four different classifications of operational amplifier gain.

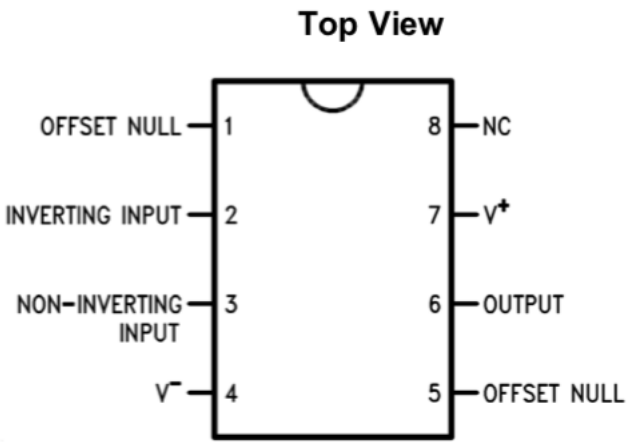
- Voltage – Voltage "in" and Voltage "out"
- Current – Current "in" and Current "out"
- Transconductance – Voltage "in" and Current "out"
- Transresistance – Current "in" and Voltage "out"

In this lab we are dealing with voltage amplifiers, that is, Vin and Vout.

$$\text{Voltage Gain, (A)} = \frac{V_{out}}{V_{in}}$$

LM741 Op-amp :

The LM741 devices are general-purpose operational amplifiers which feature improved performance. It is intended for a wide range of analog applications. The high gain and wide range of operating voltage provide superior performance in integrator, summing amplifier, and general feedback applications.



Pin Functions

PIN		I/O	DESCRIPTION
NAME	NO.		
INVERTING INPUT	2	I	Inverting signal input
NC	8	N/A	No Connect, should be left floating
NONINVERTING INPUT	3	I	Noninverting signal input
OFFSET NULL	1, 5	I	Offset null pin used to eliminate the offset voltage and balance the input voltages.
OFFSET NULL			
OUTPUT	6	O	Amplified signal output
V+	7	I	Positive supply voltage
V–	4	I	Negative supply voltage

As a Non inverting Amplifier:

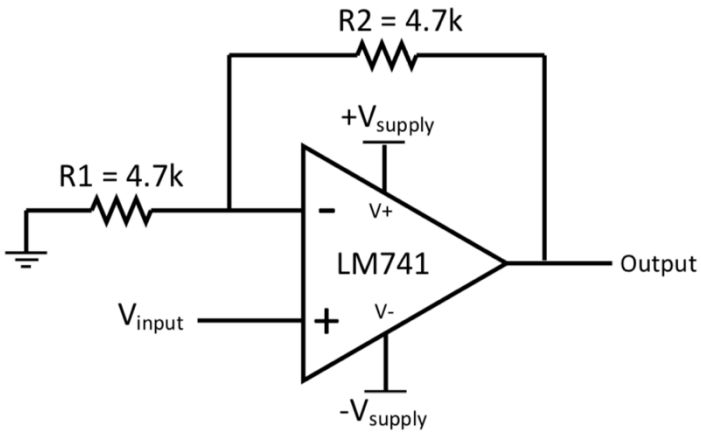
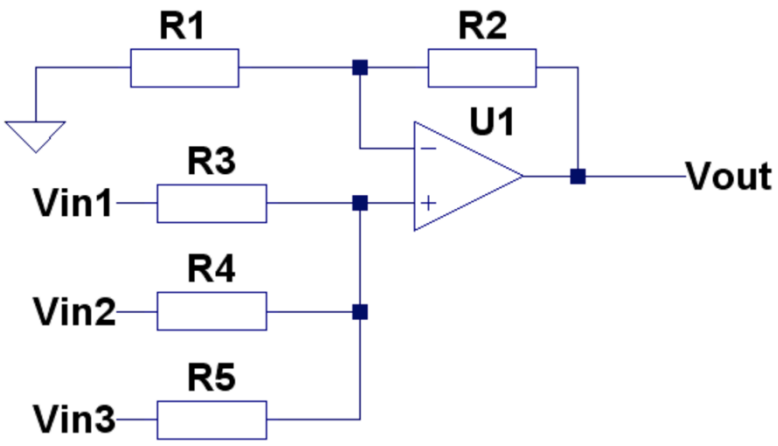


Figure 1. LM741 Noninverting Amplifier Circuit

The LM741 is a general-purpose amplifier than can be used in a variety of applications and configurations. One common configuration is in a noninverting amplifier configuration. In this configuration, the output signal is in phase with the input, the input impedance of the amplifier is high, and the output impedance is low. The characteristics of the input and output impedance is beneficial for applications that require isolation between the input and output. No significant loading will occur from the previous stage before the amplifier. The gain of the system is set accordingly so the output signal is a factor larger than the input signal.

As a summer circuit:



We saw previously in the non-inverting operational amplifier that the non-inverting amplifier has a single input voltage, (Vin) applied to the non-inverting input terminal. If we add more input resistors with more inputs, we end up with another operational amplifier circuit called a **Summing Amplifier**, "*summing inverter*" or even a "*voltage adder*" circuit as shown above.