Linear Opamp 03 : Non Inverting Amplifier

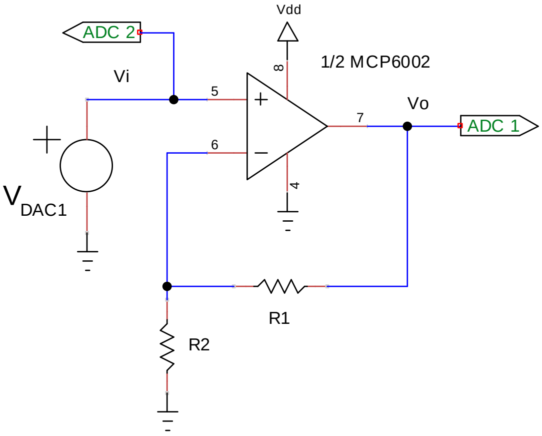
This project deals with one of the most common opamp topologies: the non inverting amplifier. Using this circuit the basics of linear opamp circuits bandwidth will be explained.

BOM	1x Dual Opamp MCP6002 2x 1 kΩ, 1x 2,2 kΩ and 2x 180 kΩ resistors 1x 1 µF capacitor

Index

The non inverting topology

The following circuit is a non inverting amplifier that generates an output voltage that is greater that the input. In the extreme case of choosing R2 infinite, the circuit defaults to our beloved follower.



We can analyze this circuit using the ideal zero order model. If you assume that it does not saturate, the **virtual short circuit** $V_{(+)} = V_{(-)}$ can be used.

In order to measure the circuit we will choose R1 = 2,2 kΩ and R2 = 1kΩ.

1

Determine the Vo(Vi) relationship as function of R1 and R2 for the above circuit assuming virtual short circuit.

Obtain the relationship for the selected resistor values.

Measuring the circuit

Now that we have calculated the circuit, let's measure it.

We will first obtain the Vo(Vi) curve for the circuit. Start the Python interpreter and issue the commands needed to connect with the hardware board:

```
>>> import slab
>>> slab.connect()
```

Now perform a Vo(Vi) curve between 0V an 3.2V in 0.1V steps:

```
>>> import slab_dc as dc
>>> dc.curveVV(0,3.2,0.1)
```

We can ease the gain calculation. From the figure we know that we are out of the saturation region for voltages below 1V. We can obtain the Vo value for two known Vi values to calculate the gain:

```
>>> slab.setVoltage(1,0.7)
>>> va = slab.readVoltage(1)
>>> slab.setVoltage(1,0.2)
>>> vb = slab.readVoltage(1)
>>> Gain = (va - vb) / 0.5
>>> Gain
```

2

Mount the proposed circuit and perform the proposed measurements.

Obtain the DC gain both from the graph, remember that you can pan and zoom, and using the two measurement method.

Do you obtain the expected value?

We can also test the circuit against a time varying signal. The SLab system can generate different waveforms. We will first select a sine waveform. Our amplifier features a gain below 4, so a signal less than 800 mV in the high peak is adequate. We will select a high peak of 800 mV and a low peak of 400 mV.

Waveforms are generated sending a set of samples. In our case we want a pretty detailed sine son we will use 100 samples for one cycle.

```
>>> slab.waveSine(0.4,0.8,100)
```

It will respond with something like:

```
100 point wave loaded
Wave frequency must be between 0.000100 and 400.00 Hz
Current frequency is 10.0 Hz
```

The maximum frequency at which the F303RE board, with the current firmware, can generate the DAC samples is 40 kHz. So, if you use 100 samples for one waveform, the maximum frequency of this waveform is 400 Hz. When the system starts the default sample rate is 1 kHz so, if you send the 100 samples at a sample frequency of 1 kHz, you get a 10 Hz sinewave.

We don't want a 10 Hz frequency so we will change it to 100 Hz:

```
>>> slab.setWaveFrequency(100)
```

The last step before performing the measurement is selecting how much data we want to obtain. As 100 samples get one wave, we will select 500 samples to get 5 full waves. We also want to obtain data from 2 ADCs (ADC 1 and ADC 2).

```
    >>> slab.tranStore(500,2)
```

Now we can finally obtain the wave response. The ***wavePlot*** command generates the wave during the time indicated in the previous ***setTransientStorage*** command. You can see the command details in the SLab Python reference document.

```
    >>> slab.wavePlot()
```

That should give you a nice presentation of the input signal V_i (ADC 2) and the output signal V_O (ADC 1). Both signals should be in phase as you are well below the band pass limit of the circuit.

You can measure both signals amplitudes to compute the gain, but there is an easier way by using the ***analize*** command.
We can instruct the SLab module to return the data of the ***wavePlot*** by setting its optional parameter ***returnData*** to True. This way, you will get a list returned after the plot. The list will contain in the 0 position a vector with the independent X axis data. From positions 1 onwards you will get all Y axis curves. In our case the ***wavePlot*** command draws two curves so we will get a list of three elements: DAC 1, ADC 1 and ADC 2. You can see the data contents just issuing its name.

```
    >>> data = slab.wavePlot(returnData=True)
    >>> data
```

Now we can analyze the data contents using the **analyze** command. This command is part of the **meas** module, so we will need to import it.
This command will give a lot of information, including peak to peak values of information contained in the data list.

```
    >>> import slab_meas as meas
    >>> meas.analyze(data)
```

From the peak to peak information of each signal it is easy to compute the gain.

	3	Perform the proposed measurement and obtain the gain from the wave amplitudes. Do you obtain the same result as in the Vo(Vi) DC curve?

It could be interesting to measure the bandwidth of this circuit. Unfortunately it is too high for the SLab system.

Amplifier Bandwidth

We now will use circuit analysis in the "s" domain. It is recommended to have knowledge on this subject to continue.

The model of the amplifier, based on **virtual shortcircuit**, predicts an amplification that is independent on the frequency of the input signal. It is not reasonable to expect the amplifier to work the same with any signal at any frequency.
To go beyond the **virtual shortcircuit** model we need to use a better opamp model We will use the first order model for the opamp with a DC gain A_O and a dominant pole p1.

$$V_O = A(s)\left(V_{(+)} - V_{(-)}\right) \qquad\qquad A(s) = \frac{A_O}{1 + \frac{s}{p1}}$$

We also have the topology of the circuit, so we can start doing serious work.

📋 **4**

Obtain the frequency response H1(s) = Vo/Vi for the previous circuit using the first order opamp parameters p1 and Ao and the resistor values R1, R2 as unknowns. As usual, you can assume Ao much greater than 1.

How is the zero frequency gain H1(0) related to R1 and R2?

Do this value coincide with the result in 📋1 ?

What is the bandwidth?

Is the GBW conserved compared with the open loop opamp?

There should be only one pole and it should be in the negative region of the "s" plane.

As we will implement the circuit using the MCP6002 opamp, we can perform some calculations specific to this device. Remember the data for this opamp:

Ao = 112 dB GBW = 1MHz.

For now we will also select R1 = 2.2 kΩ and R2 = 1 kΩ.

📋 **3**

Give values to the DC frequency gain and total bandwidth of the circuit for considered case of R1 and R2 .

Check that the DC gain is 3,2 and the bandwidth is about 312 kHz

Of course, the H1(0) value has been previously calculated in a much easier way by applying virtual short circuit on the (+) and (-) inputs of the opamp. From that, if the GBW product is conserved, then we can just apply:

$$BW = \frac{GBW}{H(0)}$$

The fast virtual short circuit method gives the same results but they are not guaranteed. We needed to perform the real calculations so we can be sure that:

① The final closed loop system pole is in the negative "s" plane.

② The GBW product really is conserved.

The **virtual shortcircuit** method can give the good result but it can also give a wrong result, try to change V(+) for V(-) if you are not convinced. As you see the problem is that this method does not give any way to check if we are getting the good result.

It could be interesting to measure the circuit to obtain the bandwidth. Unfortunately, the SLab system is quite limited in its frequency measurements. We need to increase the gain so that the bandwidth is low enough to be measured.

Increasing the gain

Let’s move on to bigger gains.
Leave R2 at 1 kΩ and change R1 with a 180 kΩ resistor.

As the gain will be higher, in order to obtain the DC Vo(Vi) curve we need to use a DC sweep step low enough to have more than two points in the linear region. We will use a 2mV step. Check that it is adequate. It makes no sense to sweep all the input range from 0 V to 3.3 V as at such high gains the output will be saturated for most of the range.

We can obtain the curve for input values lower than 100 mV:

```
>>> dc.curveVV(0,0.1,0.002)
```

You should get a high gain curve.
Use two non saturated points to obtain the gain.
Don't worry if the curve doesn't go straight to the (0,0) position. The DAC is not guaranteed to operate without errors at very low voltages.

🛠	6	Change the R2 value and measure the circuit. Do you obtain the expected gain?

Now, we want to see how the circuit performs in AC. As the gain is near 200, we would need to generate a signal with about 5 mV amplitude (10 mV peak to peak) to prevent the opamp saturation. That will push the capabilities of the SLab system.

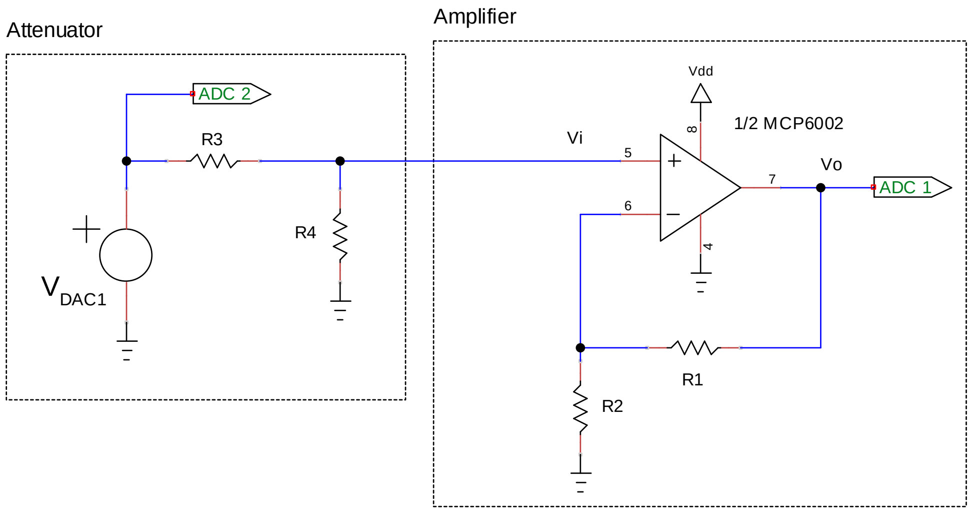
In the case of the F303RE board, DAC 1 has 12 bits and a 3.3 V maximum output voltage. If you have another board, check its characteristics. The quantization step is related to the DAC Vdd reference voltage and the number of bits n:

$$\Delta V_{Step} = \frac{V_{dd}}{2^n}$$

7 Determine the number of quantization steps of a 10mV peak to peak waveform in your SLab system.

In general it is not a god idea to generate signals that are close to the quantization limits of the DAC. The best way to generate the input signal is using an attenuator.

The following circuit implements an attenuator of the DAC 1 output based on the use of the resistor divider composed of R3 and R4. The attenuator is not affected by any loading effect because the opamp does not draw any current in its $V_{(+)}$ terminal.



Remember that the DC response of the non inverting amplifier was:

$$H(0) = \left.\frac{V_O}{V_i}\right|_{DC} = \frac{R1 + R2}{R2}$$

The response of the attenuator is:

$$\frac{V_i}{V_{DAC\ 1}} = \frac{R4}{R3 + R4}$$

If we chose R3 = R1 = 180 kΩ and R4 = R2 = 1 kΩ , the amplifier compensates the attenuation so that:

$$V_O = \frac{R1 + R2}{R2} \cdot \frac{R4}{R3 + R4} V_i \Big|_{\substack{R1=R3 \\ R2=R4}} = V_i$$

We can test the full chain that includes the attenuator and the amplifier:

```
    >>> dc.curveVV(0,3.2,0.1)
```

It should give a line with a unity slope. Perhaps there is some offset and the line doesn't pass through the (0,0) origin but this is not important.

🛠	8	Test the full chain DC $V_O(V_{DAC\ 1})$ response.

Now we can repeat the AC measurement. If you have not disconnected from the task 🛠5 the following commands don't need to be repeated:

```
    >>> slab.waveSine(0.4,0.8,100)
    >>> slab.setWaveFrequency(100)
    >>> slab.tranStore (500,2)
```

You can increase the amplitude as the attenuator guarantees that you don't saturate. But this is not really necessary. You can now generate a plot and analyze the results.

```
    >>> data = slab.wavePlot(returnData=True)
    >>> meas.analyze(data)
```

The output of DAC 1, read at ADC 2, shall have its peaks at 0.4V and 0.8V. The output of the amplifier should have the same amplitude but can be displaced vertically. You can calculate the gain of the amplifier taking into account the attenuator as:

$$A = \frac{Amplitude\ ADC1}{Amplitude\ ADC2} \cdot \frac{R3 + R4}{R4}$$

🛠	9	Perform the requested measurements and compute the gain. Is it the same as the theoretical and measured DC values?

Bode plot

You can compute the expected bandwidth of the last circuit using the calculations performed in 📝2. Now we want to generate a Bode plot that shows the frequency response of the circuit. As we have included an attenuator, the plot will also include the response of the attenuator. This attenuator is a resistive network so it has no poles or zeros. We should expect to obtain a DC gain of 1 (0 dB) with the attenuator included. To obtain the response of the non inverting amplifier, without the attenuator, just add the attenuation as an offset.
Remember that multiplications in linear units correspond to additions in dB units.

$$H_{Amplifier}\ (f)\big|_{dB} = H_{Full\ circuit}\ (f)|_{dB} + Offset|_{dB}$$

$$Offset|_{dB} = 20 \cdot log_{10}\left(\frac{R3 + R4}{R4}\right)$$

📝 **10** Calculate the expected bandwidth of the amplifier circuit.

Calculate the offset from the above equation.

Now we can obtain the Bode plot using the ***bodeResponse*** command. This command is part of the **slab_ac** module, so we will need to import it.
We need to provide as parameters:

- The low peak of the input signal (1 V)
- The high peak of the input signal (2V)
- The minimum frequency (10 Hz)
- The maximum frequency (8 kHz)
- The number of points per decade (10)

We also want the returning of plot data. So, we will use the ***returnData*** parameter.

```
>>> import slab_ac as ac
>>> data = ac.bodeResponse(1,2,10,8000,returnData=True)
```

The bode response is calculated by obtaining the response of the circuit (at ADC 1) against different input frequency sine waves (at DAC 1). The plot is shown on magnitude and phase. In the phase plot it is possible that you get bad results at high frequencies. This is due to the SLab system limitations. For high frequencies we have too low number of samples at each wave cycle. Moreover, the attenuation makes those samples too noisy.

You can obtain the bandwidth from the plot where the curve crosses the -3 dB magnitude level.

One inconvenient in the plot is that it includes the attenuator so it does not represent the amplifier response but the combined response of the attenuator and the amplifier.

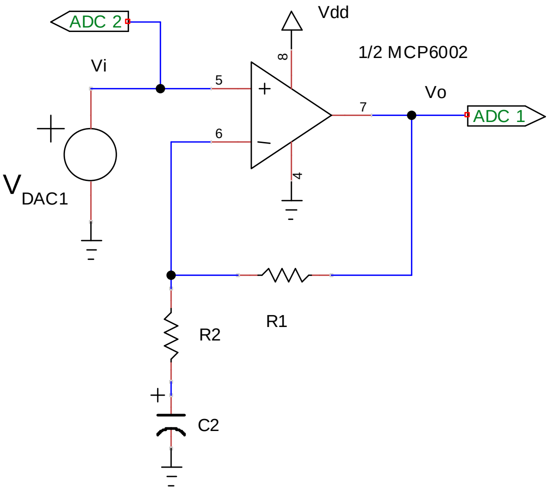
The **data** list obtained from the ***bodeResponse*** command contains two vectors. The first one **data[0]** contains the list of measured frequencies and the second one **data[1]** contains the complex gains of the circuit. We can compensate the effect of the attenuation and show the data for only the amplifier using the ***plotBode*** command.

```
>>> f = data[0]
>>> g = 181*data[1]
>>> ac.plotBode(f,g)
```

🛠	**11**	Obtain the bode response of the amplifier and measure the bandwidth. Is the bandwidth as expected?

DC Pass-through

Now that we have analyzed the basic non inverter amplifier, let's see one of its useful variations:



In this circuit we have a capacitor C2 in series with resistor R2.

What does C2 in our circuit? In short, it adds a pole and a zero. As the original amplifier featured one pole due to the GBW product, we end up with one zero and two poles. As the two poles are not related, all values are real. Nothing imaginary here.

Three ways to skin a cat

There are several ways to calculate the H2(s) transfer function of this circuit.

❶ First option is the brute force attack. Substitute the first order opamp model on the circuit and solve all the equations until you find the final solution:

$$H2(s) = \frac{V_o}{V_i}$$

This is a long way but it can be done.

❷ Second option is the feedback theory method. Separate the system in two transfer functions:

$$a(s) = \frac{V_o}{V_d}(s) \qquad f(s) = \frac{V_{(-)}}{V_0}(s)$$

Where a(s) is the first order opamp gain that depends on Ao and p1, and f(s) is the feedback function that depends on R1, R2 and C1 that relates the output voltage Vo with the $V_{(-)}$ input. Once we know a(s) and f(s), then we can do some calculations:

$$V_o = a(s)V_d = a(s)\left(V_{(+)} - V_{(-)}\right) = a(s)(V_i - f(s)V_o)$$

That gives:

$$H2(s) = \frac{V_o}{V_i} = \frac{a(s)}{1 + a(s)f(s)}$$

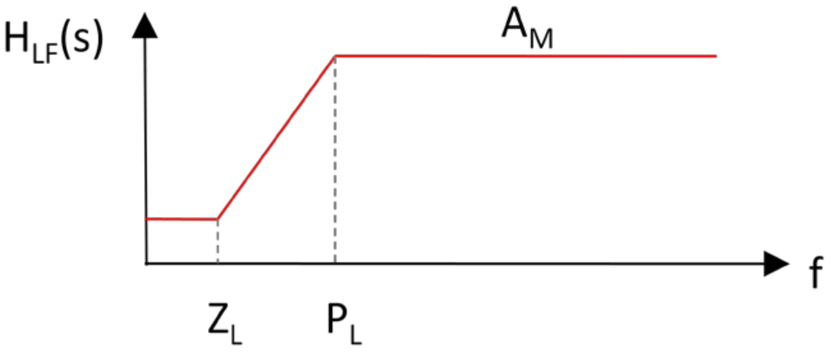
Substitute the known values of a(s) and f(s), operate and you are done.

❸ Third option is the divide and conquer lazy method.
We start using the virtual short circuit method that provides the low frequency response of the circuit. As this model doesn't include the opamp pole, it cannot predict the bandwidth of the circuit. That's fine for now.
From virtual short circuit, use the following equations to obtain $H_{LF}(s)$

$$V_{(+)} = V_{(-)} \qquad i = \frac{V_o - V_i}{R1} \qquad V_o = i\left(R1 + R2 + \frac{1}{C2 \cdot s}\right) \qquad H_{LF}(s) = \frac{V_o}{V_i}$$

The $H_{LF}(s)$ features a zero and a pole.

As explained, due to the use of virtual short circuit, the operational pole p1 is not considered. The expected behavior of $H_{LF}(s)$, as the zero is smaller than the pole, should be like that:

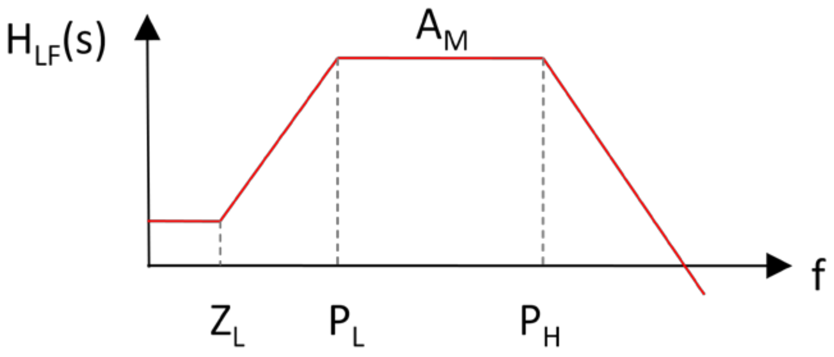


Where A_M is the medium frequency gain that should be the same as the H1(0) value we calculated in 📋1 . This makes sense: C2 is the component that produces Z_L and P_L. At frequencies much higher than P_L, C2 behaves like a short circuit, so the operation of the circuit is the same than the one of the original non inverting amplifier.

We now calculate the high frequency behavior of the circuit. We know that we are limited by the opamp GBW, that way, the high frequency pole shall be:

$$P_H = \frac{GBW}{A_M}$$

If P_L is much lower than the calculated P_H, then we know that the GBW doesn't affect the low frequency behavior of the circuit and we can just add the new pole to the system:



12 Solve the circuit using, at least one of the three proposed methods. Obtain the zero value Z_L and the two pole values P_L and P_H.

Obtain also the DC gain H2(0).

If you want to learn more, feel free to use all three and compare the results.

The behavior of the system is as follows:

- At low frequencies, much below Z_L, capacitor behaves as an open circuit, so the circuit behaves like a follower.

- At medium, much higher than PL and much lower than PH, the capacitor behaves as a short circuit so the circuit behaves as a normal inverter in DC operation.

- At high frequencies the GBW product kicks as in the original amplifier.

We will select the low gain resistor values R1 = 2.2 kΩ and R2 = 1 kΩ. For C2 we will choose a 1 µF value.

13 Obtain H2(0) and the zero and poles frequencies for the selected components.

Now it is time to test the circuit.

The 1 µF capacitor is probably **electrolytic**. This kind of capacitor features a (-) marking and a shorter leg on the negative terminal and don't like to be reverse polarized. Be sure to connect the negative terminal to ground, not the other way.

Remember the command to get the bode plot. This time we don't aspire to reach the second pole as its frequency is too high for the SLab system, so we will limit ourselves to a frequency of 2 kHz. We will set the peak to peak amplitude to 0.2V. As the circuit blocks the DC component, we can set the DC value of the input signal wherever we want.

```
>>> ac.bodeResponse(1.4,1.6,5,8000)
```

14 Obtain the bode response of the amplifier and check the locations of the first zero and pole.

We can check that the gain is one (0 dB) at DC by obtaining a Vo(Vi) curve.

```
    >>> dc.curveVV(0,3.2,0.1)
```

You can do the same by using a sine wave if it is below the first zero. Gain should be one. We use a peak to peak amplitude of 1 V in this case.

```
    >>> slab.waveSine(1,2,100)
    >>> slab.setWaveFrequency(5)
    >>> slab.tranStore(500,2)
    >>> slab.wavePlot()
```

Over the first pole you get the gain defined by the resistor ratio. We use a peak to peak amplitude of 0.2 V to prevent the amplifier saturation. We also reduce the number of samples per wave because the maximum sample rate of the SLab system does not allow us to have 100 samples per wave for a wave frequency of 1 kHz.

```
    >>> slab.waveSine(1.4,1.6,20)
    >>> slab.setWaveFrequency(1000)
    >>> slab.tranStore(100,2)
    >>> slab.wavePlot()
```

Observe, also, that in this circuit, the DC is conserved, so the input and output signals have the same average value. Last measurement featured a 1.5 V DC value, now we test with a 2.5 V DC value.

```
    >>> slab.waveSine(2.4,2.6,20)
    >>> slab.wavePlot()
```

🛠	15	Perform the requested measurements.

Last comments

In this project we have seen different circuits based on the non inverting circuit topology.
We also have seen the conservation of the GBW product.
Keep note of that result because it will be an important point in the opamp inverting circuit.

References

SLab Python References

Those are the reference documents for the SLab Python modules. They describe the commands that can be carried out after importing each module.

They should be available in the **SLab/Doc** folder.

TinyCad

Circuit images on this document have been drawn using the free software TinyCad
https://sourceforge.net/projects/tinycad/

SciPy

All the functions plots have been generated using the Matplotlib SciPy package.
https://www.scipy.org/