18-100: Intro to Electrical and Computer Engineering

LAB09: DSP Lab

Writeup Due: Thursday, April 21st, 2022 at 10 PM

Name: Eunice Lee

Andrew ID: eunicel

How to submit labs:

Download from this file from *Canvas* and edit it with whatever PDF editor you're most comfortable with. Some recommendations from other students and courses that use Gradescope include:

pdfescape.com	A web-based PDF editor that works on most, if not all, devices.
Preview	Pre-installed default MacOS PDF Editor.
iAnnotate	A cross-platform editor for mobile devices (iOS/Android).

If you have difficulties inserting your image into the PDF, simply append them as an extra page to the END of your lab packet and mark the given box. ***Do NOT insert between pages.***

If you'd prefer not to edit a PDF, you can print the document, write your answers in neatly and scan it as a PDF. *(Note: We do not recommend this as unreadable lab reports will not be graded!).* Once you've completed the lab, upload and submit it to *Gradescope*.

Note that while you may work with other students on completing the lab, this writeup is to be completed alone. Do not exchange or copy measurements, plots, code, calculations, or answer in the lab writeup.

Your lab grade will consist of two components:

1. Answers to all lab questions in your lab handout. The questions consist of measurements taken during the lab activities, calculations on those measurements and questions on the lab material.
2. A demonstration of your working lab circuits and conceptual understanding of the material. These demos are scheduled on an individual basis with your group TA.

Question:	1	2	3	Total
Points:	15	10	15	40
Score:				

Lab Outline

This lab aims to familiarize students with the concept of Analog-to-Digital Conversion (ADC) and demonstrate how the resolution of the ADC can improve reconstruction of the original analog signal.

1. Introduction to Raspberry Pi Pico
2. Quantization
3. Sampling
4. IIR Low Pass Filter

Small Group Check-off Circuits

☐ None

Equipment Required

- 1x RPi Pico
- 1x MCP4725 12-bit DAC
- 2x USB A - USB B Micro Cable
- 1x ADALM2000 {`Oscilloscope`, `Signal Generator`}
- 1x Wire Strippers
- 1x Diagonal Cutters

Introduction to Raspberry Pi Pico

RPi Pico ADC & DAC

The RPi Pico has a built-in 16-bit ADC. On the RPi Pico, pins `31` through `35` support reading values from an analog signal source. We will also be using the MCP4725, a 12-bit DAC with an I2C interface. The MCP4725 DAC can output a true analog signal, not just a PWM signal.

Net	Use
AGND	The ground value to compare ADC# to
ADC#	The signal being measured
ADC_VREF	The maximum voltage possible

Table 1: Analog Reading Pin Descriptions

Reading Analog Values

To read an analog value, we simply need to connect the input signal according to Table 1 and then use `adc.value` to store the read value in a variable or pass it to a function.

Writing Analog Values

In order to write data to the MCP4725 DAC, we will need to send it commands via I2C. We will first send the write DAC Command `0x40` and then send the 12-bit value representing the intended output voltage. An example of how to do this is in the file `adc-dac-quantization.py`.

The RPi Pico communicates with the computer via a micro-USB cable. The USB connection also supplies 3.3V power to the board.
Warning: Unlike most boards, the RPi PIco runs on 3.3V logic. Do not send more than 3.3V to the microcontroller pins. Applying voltages higher than 3.3V to any I/O pin could damage the board.

The following diagram shows the pin-out of the RPi Pico board. Pay close attention to the right-side group that contains analog pins from 31 to 35.

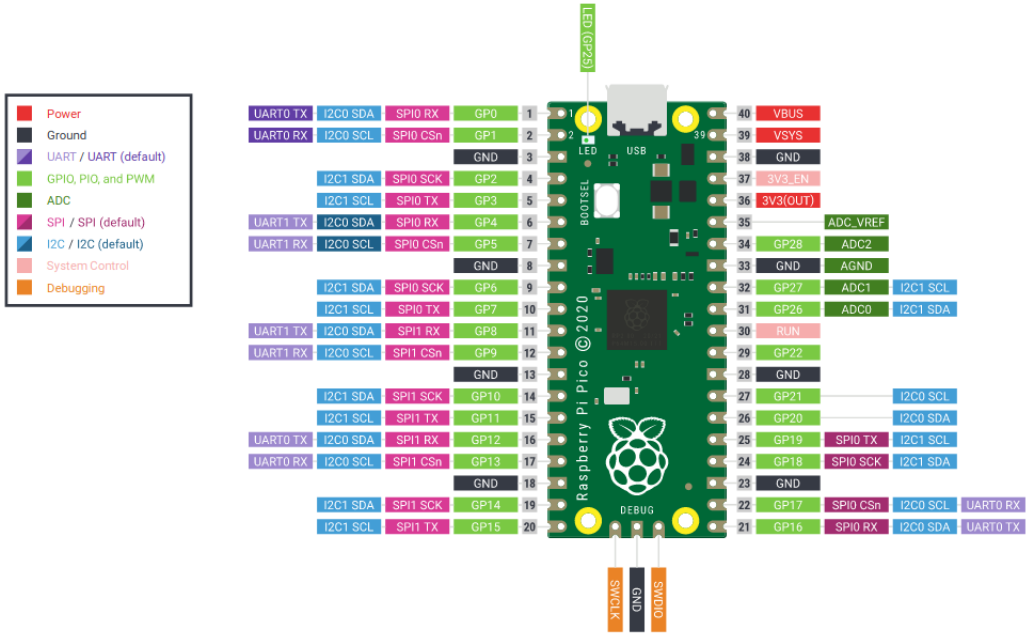


Figure 1: RPi Pico Pinout

1. ADCs, DACs, and Quantization Noise

Construct the following setup. Set your signal generator output W1 to a 20Hz $2V_{pp}$ sine wave with a 1V offset. Hook up the Signal generator to output a signal into `ADC0` which you will measure with `Ch1`. Then measure the output of the DAC using `Ch2`. **Double check that your voltage values are correct, as anything too high could damage your RPi Pico.**

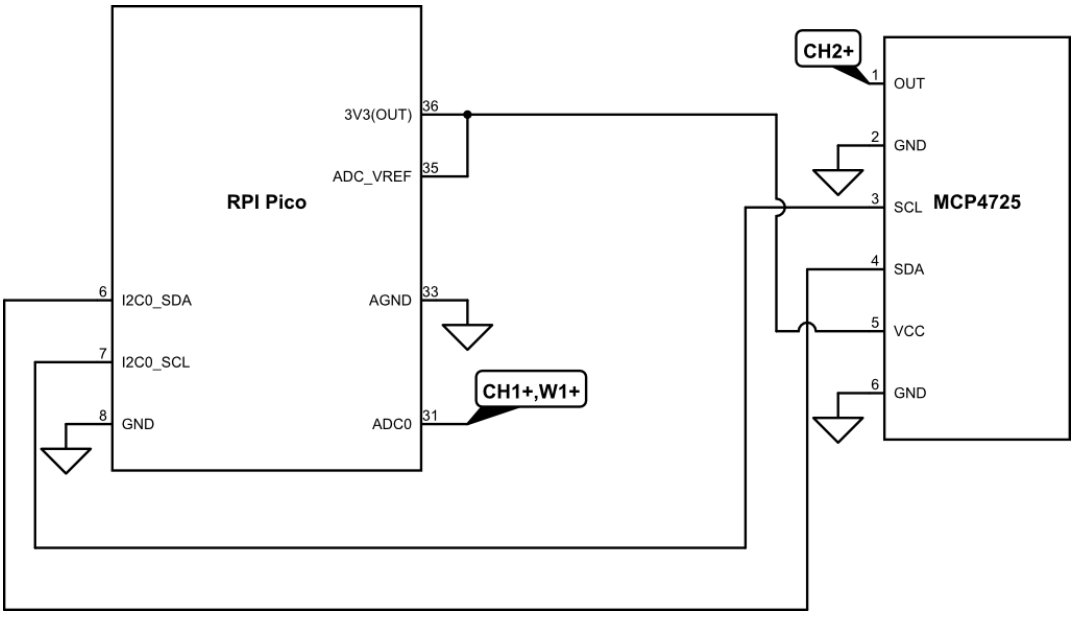


Figure 2: RPi Pico Measurement Setup

Code

Open up the file `adc-dac-quantization.py`, containing the following code snippet, then copy paste it into the `code.py` file on your RPi Pico

```
DAC_ADDR = 0x60
DAC_CMD_REGISTER = 0x40
RESOLUTION = 3
i2c = busio.I2C(scl=board.GP5, sda=board.GP4, frequency=400000)
adc = analogio.AnalogIn(board.GP26)

# @brief write voltage to the DAC to output
# @param[in] voltage voltage to output (12bit resolution)
def writeDac(voltage):
    output = voltage
    data = bytearray([DAC_CMD_REGISTER, (output & 0xFF0) >> 4,\
    (output & 0x0F) << 4])
    while not i2c.try_lock():
        time.sleep(0.01)
    i2c.writeto(DAC_ADDR, data)
    i2c.unlock()

while True:
    val = adc.value >> (16 - RESOLUTION)
    writeDac(val << (12 - RESOLUTION))
```

LSB Step Size

The size of the least significant bit of an ADC or DAC will determine how much information it is able to capture. The smaller the LSB step size, the more finely your ADC/DAC can represent a signal. The formula for calculating LSB step size if given as:

$$\text{LSB Step Size [V]} = \frac{\text{Full Scale Resolution [V]}}{2^N}$$

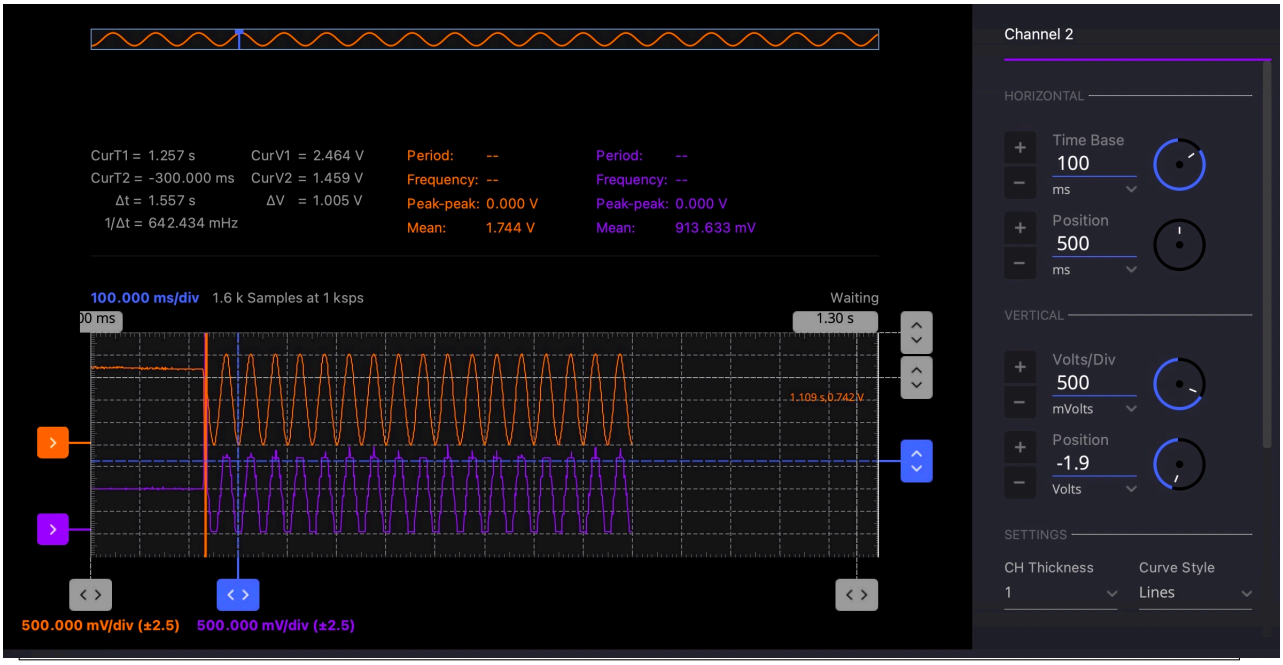
Where the Full Scale Resolution of the PiPico is 3.3V and N is the number of resolution bits.

2 pts **1.1** What is the LSB step size of the ADC when `RESOLUTION` is set to 3 bits?

$\frac{3.3}{2^3} = 0.4125$

$V_{LSB} =$ 412.5 mV

3 pts **1.2** Paste a screenshot of your oscilloscope plot showing both the input to the ADC and the output of the DAC. It is up to you to size the window and choose the correct settings. Any window in which both your input and output waveforms are clearly visible will receive full credit.



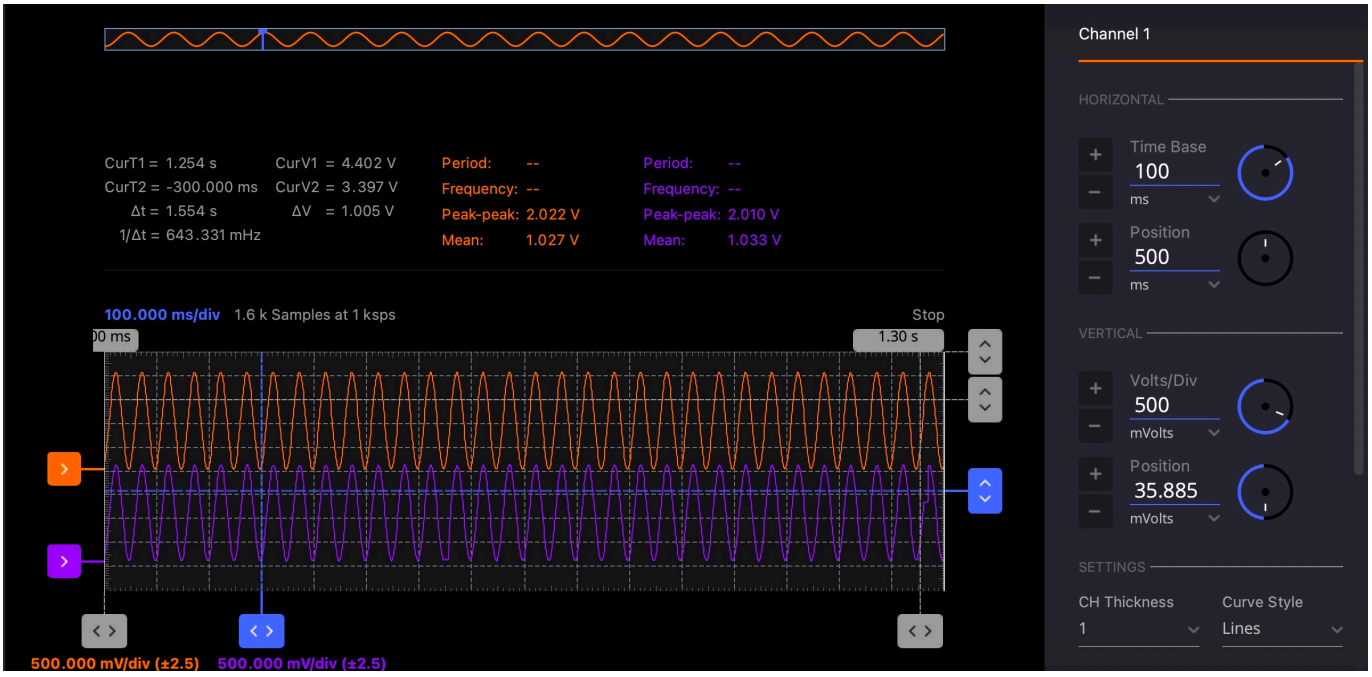
2 pts

1.3 Describe the differences between the input and output waveforms. Why is the output waveform shaped the way it is?

While the input waveform is a smooth sine wave, the output wave is rigid and bumpy. The output looks like a sine wave but has steps. However, they look similar because they have the same offset and period, but because the resolution is low, the output wave is not smooth.

3 pts

1.4 Now set the value of `RESOLUTION` to 10. Paste a screenshot of your oscilloscope plot showing both the input to the ADC and the output of the DAC. It is up to you to size the window and choose the correct settings. Any window in which both your input and output waveforms are clearly visible will receive full credit. (Note: *If you have difficulties inserting your image into the PDF, simply append them as an extra page to the END of your lab packet.* ***Do NOT insert between pages.***)



2 pts **1.5** Describe the differences between the input and output waveforms. What changed from the previous section? Why?

In the previous section, the output wave was bumpy but now it appears smooth, extremely similar to the original input waveform. Because we increased the resolution, the steps decreased in size. The step size is small so the wave looks like no steps have been taken, making it look smooth.

3 pts **1.6** What is the new LSB step size of the ADC? Compare it to the LSB step size of the previous ADC. How does LSB step size impact the output?

$\frac{3.3}{2^{10}} = 0.0032$ V

$V_{LSB} =$ 3.2 mV

previous LSB step size was larger and the V_{LSB} was 412.5 mV, which is a lot larger than the current 3.2 mV, where the step size is much smaller. This makes the output appear smooth, making it more similar to the input.

2. Sampling and Aliasing

Sampling is the process of converting a continuous-time signal to a discrete-time signal. The rate of sampling is based on the number of times per second a signal is read. For example, the sampling rate of audio CDs is 44100 samples per second, which is equivalent to 44.1kHz. This sample rate then affects the highest audio frequency that can be reproduced.

Nyquist-Shannon Sampling Theorem

The Nyquist-Shannon Sampling Theorem is used in digital signal processing to allow discrete-time signals to be properly sampled from continuous-time signals. The theorem states that if a continuous-time function $x(t)$ has frequencies up to f_{max} Hertz, then a sample rate of $2 * f_{max}$ samples/second (Nyquist Rate) or higher is sufficient. This can be easily represented by the following:

$$f_s \geq 2 * f_{max}, \text{where } f_{max} \text{ is the Nyquist Frequency}$$

When this criterion is not met, aliasing occurs. Aliasing causes signals to become different and unrecognizable, and refers to distortion or artifacts being present in reconstructed signals.

Keep the signal generator in the same setup as the previous part. Set V_{in} to a 10Hz $2V_{pp}$ sine wave with a 1V offset. **Make sure that you use the offset so that your signal is always positive! The ADC is expecting a signal from 0 to 3.3V** Double check that your voltage values are correct, as anything too high could damage your RPi Pico.

Open up the file `sampling-aliasing.py`, containing the code snippet below

```
1
2 adc = analogio.AnalogIn(board.GP26)
3
4 while True:
5     for x in range(100):
6         print((adc.value,))
7         time.sleep(0.01)
8     input()
```

Note that the line `time.sleep(0.01)` creates a 10 millisecond delay after each loop iteration in the program. This makes the period of our program about 10ms, making the sampling frequency ~100Hz. Open the Serial Plotter to see the waveform being plotted continuously.

The Serial Plotter continuously plots new samples, and it can be difficult to read. To make things a little easier on the eyes, this program measures 100 samples (1 full second) and then pauses, waiting for the user to press ENTER on the serial console before moving on. This will momentarily freeze the program and the plotter, pausing the waveform for inspection.

3 pts **2.1** Slowly increase the frequency of the signal generator from 1 Hz to 100 Hz, going in increments of 1-5 Hz. What do you notice start to happen to the frequency of the plotted signal in the 45-55 Hz range?

In the 45-55 Hz range, the plotted signal looks much more consistent, where the amplitude increases and decreases, forming a pattern. At 45 Hz and at 55 Hz, the wave is still a little sporadic/irregular, but near 50 Hz, the plotted signal is more precise.

3 pts **2.2** Assuming the sampling rate of the ADC is 100 Hz, calculate the Nyquist frequency of the circuit. How does this relate to what you observed in the previous question?

$f_s \geq 2 \cdot f_{max}$

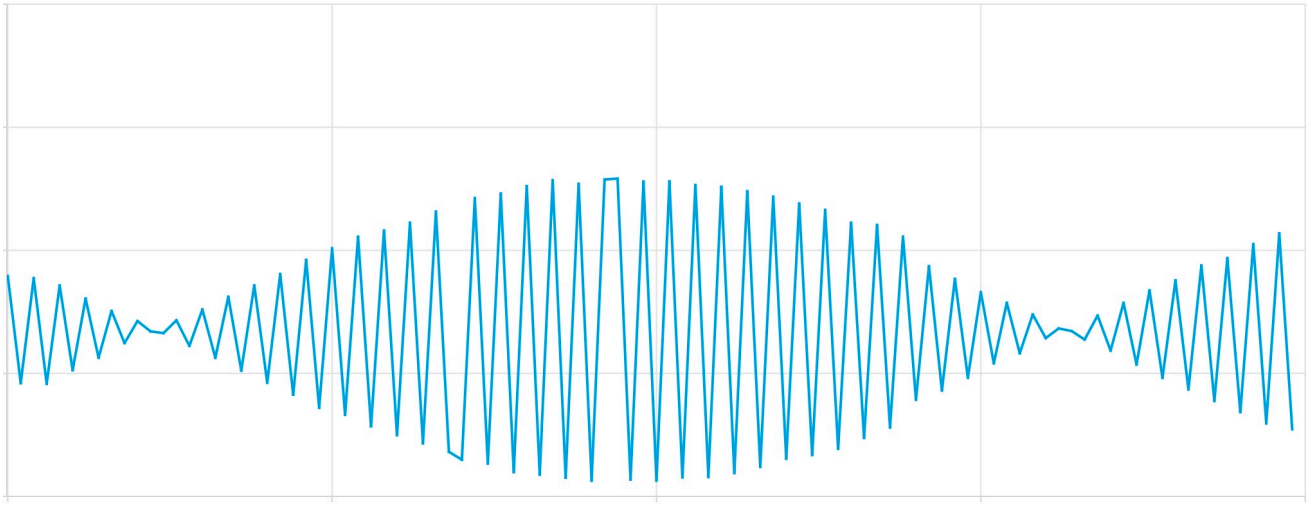
$\frac{100}{2} = 50$

$f_{nyquist} =$ 50 Hz

1 pts **2.3** Above which frequency can you see the output start to alias? Remember, you should see approximately 2 samples per signal period at this frequency, the **Nyquist frequency**. It might help to use 0.1-0.2 Hz increments as you get closer to the Nyquist frequency.

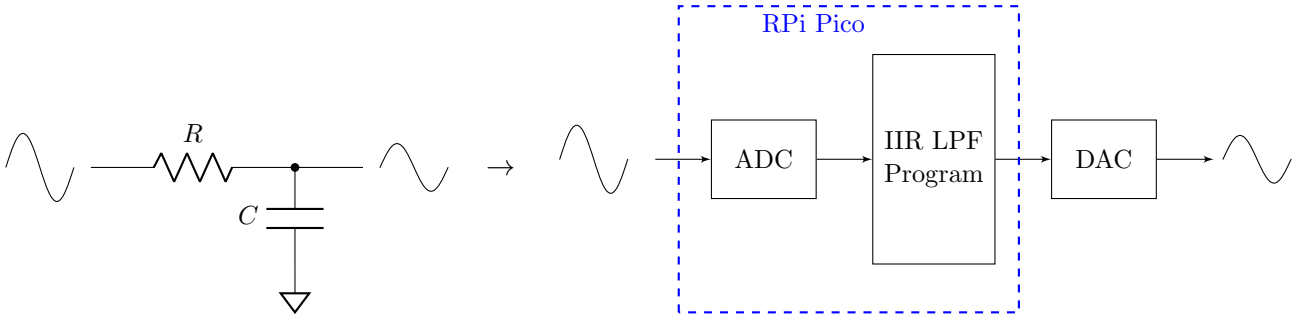
$f_{alias} =$ 50.5 Hz

3 pts **2.4** Attach a screenshot of the Serial Plotter with the output at the frequency from the previous question.



3. Digital IIR Filters

In previous labs, we've used passive components (resistors, capacitors, etc.) to make high/low-pass analog filters. However, filters can also exist is the digital space! In this lab, we will use the RPi Pico as a specific type of discrete-time (sampled) digital filter: an infinite impulse response (IIR) filter.



(a) Analog Low Pass Filter (b) Digital Low Pass Filter

Simple Low-Pass IIR Filters

In this lab you will implement a simple IIR low-pass filter digitally by writing some CircuitPython code. This filter has a gain of one at low frequency and a gain of zero at the Nyquist frequency.

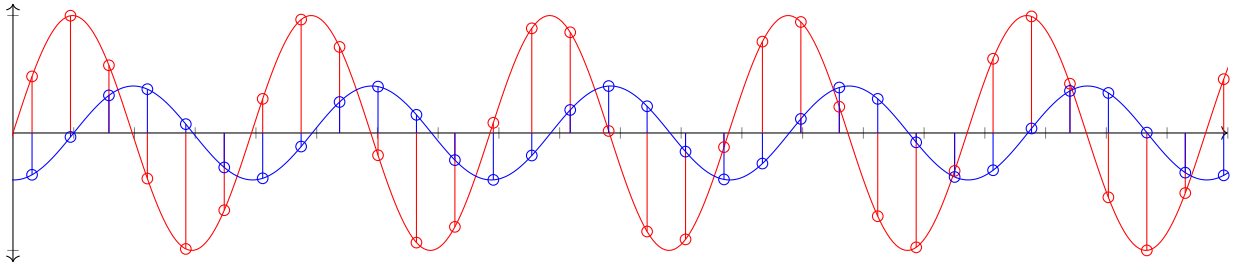


Figure 4: A Sine Wave

The filter has a single design parameter, which is the *decay value*, α ($0 < a < 1$). The following equation calculates the output, $y[n]$, of the filter:

$$y[n] = (1 - \alpha) * x[n] + \alpha * y[n - 1]$$

where $x[n]$ represents the current input sample and $y[n-1]$ is the last calculated sample. Substituting $b = 1 - \alpha$ in the given expression and rewriting it as a line of pseudo-code to get the expression:

`y += b * (x - y)` where $b = 1 - \alpha$

Relationship between Digital and Analog Filters

The simple IIR low-pass filter is completely analogous to the electronic low-pass filter that we implemented in previous labs with a single resistor R and a single capacitor C. The decay value α is related to the the *cutoff frequency* (-3dB point), f_c:

$$\alpha = \frac{1 - \sin(2\pi T_s f_c)}{\cos(2\pi T_s f_c)}$$

Where T_s is the sampling period (in this lab ~10 ms).

2 pts **3.1** What should the decay factor α be so that the cutoff frequency $f_c = 8$ Hz?

$$\frac{1 - \sin(2\pi(0.01)8)}{\cos(2\pi(0.01)8} = 0.5914$$

$\alpha =$ 0.5914

Using the starter code `simple_iir.py`, implement the simple digital IIR filter. There are a few blanks that need to be filled in with your own code. This task should be straightforward if you have carefully read the previous sections and the previous two CircuitPython files.

```
# TODO: fill in these expressions
alpha = _________
b = (1 - alpha)
adc = analogio.AnalogIn(board.GP26)

x = 0
y = 0
while True:
    for k in range(150):
        # TODO: fill in these expressions
        x = ____________
        y = ____________
        print((x,y))
        time.sleep(0.01)
    input()
```

5 pts

3.2 Implement a digital IIR filter by filling in the missing sections in the code. Make sure that it compiles! Paste your completed code here.

```
# input output packages
import analogio
import digitalio

led = digitalio.DigitalInOut(board.LED)
led.direction = digitalio.Direction.OUTPUT
led.value = True


alpha = 0.5914
b = (1 - alpha)
adc = analogio.AnalogIn(board.GP26)

x = 0
y = 0
while True:
    for k in range(150):
        x = adc.value # Read sample in from ADC
        y = y + (b*(x-y)) # Update the output y using the IIR filter
        print((x,y))
        time.sleep(0.01)
    input()
```

Save the code you wrote to the RPi Pico. Keep the signal generator in the same setup as the previous part. Set V_{in} to a $2V_{pp}$ sine wave with a 1V offset. Double check that your voltage values are correct, as anything too high could damage your RPi Pico. Set f to 1Hz, 2Hz, 4Hz, 8Hz, 16Hz, and 32Hz. Open up the `Serial Plotter` to observe the output.

3 pts **3.3** What happens to the filtered waveform as the frequency increases? Why?

As the frequency increases, the waveform has more waves than previously. In addition, the amplitude decreases because the filter is a low pass filter, filtering out higher frequencies. The amplitude, or gain, decreases.

1 pts **3.4** For which of these frequencies is the output amplitude closest to 0.707x ($\frac{1}{\sqrt{2}}$x or -3dB) of the input amplitude?

$f_c \approx$ 8 Hz

4 pts **3.5** Attach a screenshot of the Serial Plotter at this frequency (f_c).