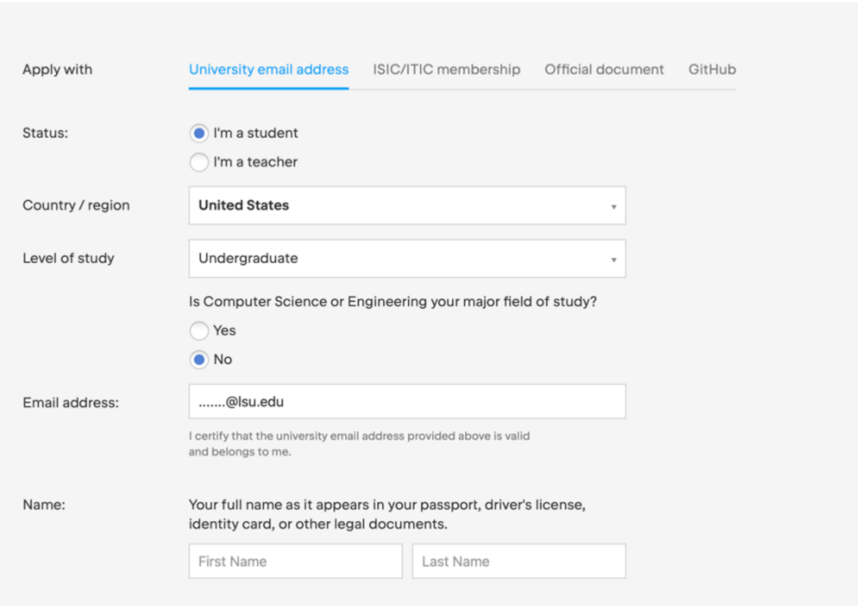
EE 4162 – LAB 1 A– GETTING STARTED

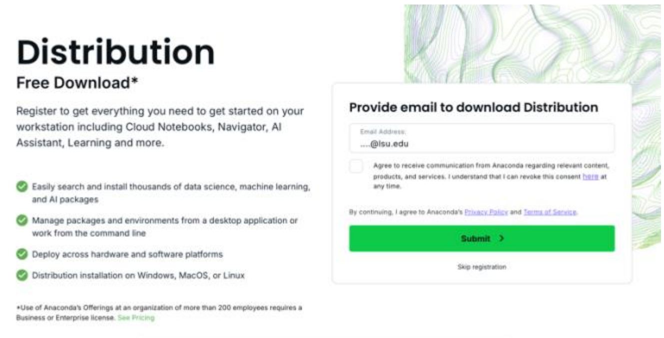
This lab's purpose is to help set up and familiarize yourself with PyCharm and Python. You will run code in Python (recommended) or MATLAB, if you desire, for this lab alone, to explore some basic concepts in Digital Signal Processing.

Installing PyCharm and Anaconda

PyCharm and Ananconda are recommended for this lab. On your personal computer, install PyCharm and apply for the Educational license with your LSU credentials. Applying for the license will direct you to this form you will have to complete.



After completing this, install the Anaconda distribution using your LSU email.



Setting up PyCharm

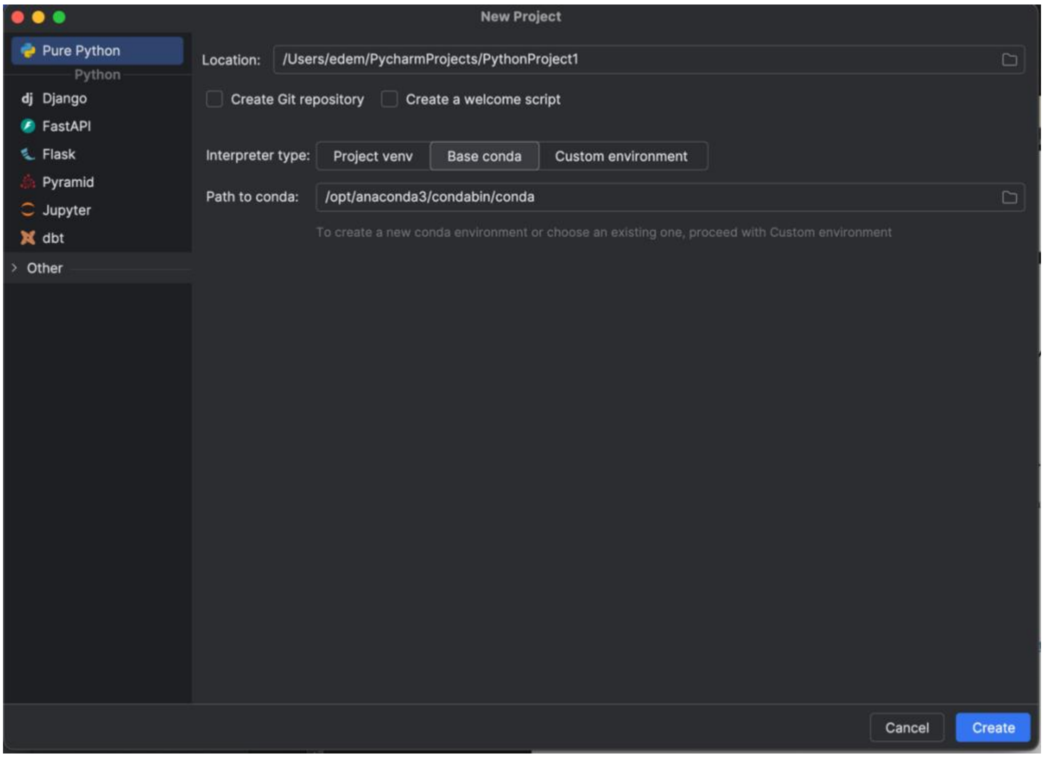
After downloading and installing both Anaconda and PyCharm;

1) Open PyCharm and Go to PyCharm Settings -> Project.
2) Under Project Interpreter, navigate to the Anaconda python path on your computer: /anaconda/bin/python on Mac or C:/Users/YourUserName/anaconda/bin/python on Windows.

3) Click OK.

Create a new Project

1) Create a New Project from the PyCharm Welcome Screen or Go to -> New Project
2) Pick a Pure Python project and set the Interpreter to the Anaconda distribution.



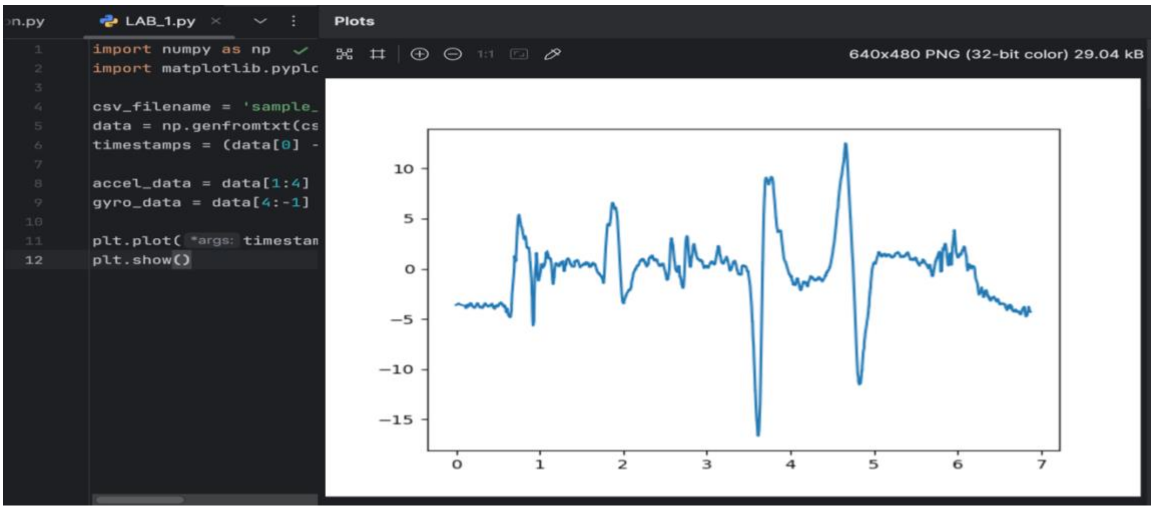
3) Name your project as LAB_1 and click OK.
4) Right-click the LAB_1 Navigation bar.
5) Choose New -> Python File and name it peak_detection.py, and click OK.

Experiment 1

Generated sample sensor data in the Comma Separated Values (CSV) file sample_sensor_data.csv which has been uploaded contains data samples collected from a pedometer.
Copy the file to your LAB_1 folder.
In the file peak_detection.py, type the following code as shown.

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3
4   csv_filename = 'sample_sensor_data.csv'
5   data = np.genfromtxt(csv_filename, delimiter=',').T
6   timestamps = (data[0] - data[0,0]) / 1000
7
8   accel_data = data[1:4]
9   gyro_data = data[4:-1]
10
11  plt.plot( *args: timestamps, accel_data[0])
12  plt.show()
```

1) The necessary libraries [numpy](#) and [matplolib](#) are imported using lines 1 and 2.
2) The np.genfromtxt function reads the CSV file and loads its data into a NumPy array. The delimiter argument specifies that the file is comma-separated, and .T transposes the array. (Why do we do this?)
3) The time vector is extracted in line 6 as timestamps. (data[0]-data[0,0]) extracts the first row and makes the first timestamp the reference point. We divide by 1000 to convert from milliseconds to seconds.
4) From the CSV file, the accelerometer and gyroscope data of the pedometer are extracted in lines 8 and 9 respectively.
5) Plot using plt.plot with the x-axis as the timestamps and the y-axis as the first component of accelerometer data.
6) We display the plot with plt.show()
7) Run the program by Right-clicking inside the code page and choosing Run 'peak_detection'
8) The plot should be as shown in the image.



Debugging code

1) Click on the space between the code and line number 11 to set a breakpoint which appears as a red circle.
2) Right-click inside the code area or click on the Debug button on the top right corner of PyCharm (the bug symbol). The execution will stop at line 11.
3) In the Debug console, you can view all the variables currently created. If it is a numpy array, you can click on View as Array (at the end of the row) to view it as a table.
4) While in the Debug mode, you can write code on the fly to test out your logic.
5) Switch to the Console tab-- an interactive console connected to the debugger.
6) Let's find the maximum value of the accelerometer data in the console by typing accel_data[0].max() in the console
7) Press Enter to see a value of 12.507.

Finding peaks

We define a function to find the peak in the signal generated from the plot. Type the following lines after line 2.

```
4   def peak_detection(t,sig):  1 usage
5       peaks = []
6       max_val = -np.Inf
7       N = len(sig)
8       for i in range(0,N):
9           if sig[i] > max_val:
10              max_val = sig[i]
11              position  = t[i]
12
13      peaks.append((position, max_val))
14      return np.array(peaks)
```

- The function accepts the time array `t` and the accelerometer data array `sig`. We define a Python list of peaks on line 5.
- This function is used to detect a single peak, the maximum value of our signal, using a for loop (line 8) to traverse through the array sig.
- Initially, max_val is set to minus infinity (Why?). Every time we encounter a greater value than max_val, this value and its time position are recorded in lines 10 and 11.
- The position and the maximum value are appended to the list of peaks in line 13. In line 14, we turn our list to a numpy array for plotting and return it to the calling function.
- Now that we define our function, we can call it and plot our peak. Add this line before the plot function.
  `max_peaks = peak_detection(timestamps, accel_data[0])`
- And this line after the plot function
  `plt.scatter(max_peaks[:,0], max_peaks[:,1], color = 'red')`

- Run the program and you should see a red dot indicating the peak

Assignment

1) a) Run the above experiment to display the maximum value of the provided accelerometer data.

b) Give your plot a title of "First axis of accelerometer data." Name your x-axis "Time" and your y-axis "Meters per second".

2) a) Modify the peak_detection function to detect other peaks in the signal. Add the new peaks and time positions to our peaks list. The function should accept a new input parameter, thresh (peaks should have a value greater than thresh).

b) Plot all the peaks you detected, title the figure, and label the axis accordingly.

3) Acceleration is the rate of change of velocity. By integrating acceleration over time, velocity can be obtained. Velocity, when integrated over time, yields displacement. Numerical integration can be performed using techniques like cumulative summation. You will need to use the following libraries for this problem; numpy, matplotlib, scipy.

   a) Open a new Python file.
   b) Load the CSV file and extract the timestamps and acceleration data as done earlier. However, use accel_data = data [1] for the acceleration data.
   c) Compute time intervals from the time data
      dt = np.diff(timestamps, prepend=timestamps[0]) # Time intervals
   d) Calculate the velocity by numerical integration
      velocity = np.cumsum(accel_data * dt) # Integrate acceleration to get velocity
   e) Calculate the displacement by numerical integration
   f) Create three subplots
      a. Acceleration vs Time
      b. Velocity vs Time
      c. Displacement vs Time

4) The Fourier Transform decomposes a time-domain signal into its constituent frequencies and is significant in signal processing to analyze the frequency spectrum of signals.
   a) In the same Python file from (3), apply the FFT to the acceleration data
   b) Compute the frequency bins and the magnitude of the FFT.

c)

```python
from scipy.fft import fft, fftfreq

# Apply Fourier Transform
N = len(accel_data)  # Number of data points
sampling_rate = 1 / np.mean(np.diff(timestamps))  # Sampling frequency (Hz)
freqs = fftfreq(N, d=1/sampling_rate)  # Frequency bins
fft_magnitude = np.abs(fft(accel_data))  # Magnitude of FFT

```

d) Plot the frequency spectrum (magnitude vs. frequency)

e)

```python
# Plot frequency spectrum
plt.figure(figsize=(8, 6))
plt.plot( freqs[:N // 2], fft_magnitude[:N // 2])  # Plot only positive frequencies
plt.title('Frequency Spectrum')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')
plt.grid()
plt.show()
```

i) How can the dominant frequencies in the signal be related to the motion of the object?

ii) What is the significance of only analyzing the positive half of the frequency spectrum?

<u>Links</u>

PyCharm - https://www.jetbrains.com/pycharm/
Educational License - https://www.jetbrains.com/community/education/#students
Anaconda - https://www.anaconda.com/download/
Numpy - https://numpy.org
Matplolib - https://matplotlib.org
Function - https://www.tutorialspoint.com/python/python_functions.htm
List - https://developers.google.com/edu/python/lists