APS106 – Lab #2

Preamble

This week you will practice writing functions. In the first part you will write two functions – `magnitude(x,y)` and `phase(x,y)` - that convert a 2-dimensional vector into a magnitude and phase angle, respectively. In the second part, you will write a function to approximate the vertical position of a charged particle in an electrostatic precipitator[1].

By completing this lab, you will develop, utilize, and practice the following skills:

- *setup, debug, and run a program*
- *interpret real world parameters as variables*
- *utilize math functions*
- *utilize built-in functions*
- *develop effective test cases*
- *read Python documentation*

Use appropriate variable names and place comments throughout your program.

Deliverables

For this lab, you must submit all functions within a single file named 'lab2.py' to MarkUs by the posted deadline.

Five test cases are provided on MarkUs to help you prepare your solution. **Passing all these test cases does not guarantee your code is correct.** You will need to develop your own test cases to verify your solution works correctly (see part 1.1 for more details). Your programs will be graded using ten secret test cases. These test cases will be released after the assignment deadline.

IMPORTANT:

- Do not change the file name or function names
- Do not use input() inside your program

[1] Don't worry if you haven't heard of an electrostatic precipitator before. In these labs we will use example of engineering technologies to motivate problems, but we will always provide you all information and equations you need to write the code.

Part 1 - Problem and Motivation

For many problems in engineering and sciences[2], analysis can be simplified by converting between different coordinate systems. A common transformation is to convert 2-dimensional cartesian coordinates (*x* and *y* components) to polar coordinates (magnitude and phase angle). Figure 1 below illustrates how a vector can be defined using either coordinate system.

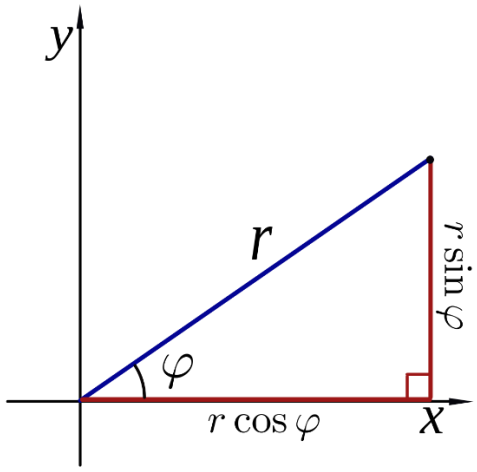

Figure 1. Example of vector defined by both cartesian and polar coordinate systems where *r* represents the magnitude and *φ* represents the phase angle.

We can calculate the magnitude, *r,* and phase angle, *φ,* with the following equations:

$$r = \sqrt{x^2 + y^2} \tag{1}$$

$$\varphi = \tan^{-1}\left(\frac{y}{x}\right) \tag{2}$$

where *x* and *y* represent the two components of a vector in cartesian coordinates.

Part 1.1 Calculate Vector Magnitude

Download the file lab2.py from quercus.

[2] Examples include analysis of AC circuits, mechatronic control systems, ionic flow in fluids, and quantum mechanics

For the first part of this lab, you will complete the function `magnitude(x,y)` which accepts `x` and `y` as floating point input parameters and returns the magnitude as calculated using equation 1.

Sample Test Cases

Note: each week we will provide some example test cases that you can use to design and evaluate your programs. **Passing these sample test cases does not guarantee your code is correct.** You will need to consider your own additional test cases to ensure your program works correctly. An important part of programming is designing test cases to ensure your program functions correctly with different input combinations. The example test cases will be given with brief descriptions to give you insight into the scenarios being tested and how you might design further test cases.

Inputs		**Expected Output (to six decimal places)**	**Description**
x	**y**	**magnitude**	
0	0	0.0	Zero vector
10.2	63.2	64.017810	Quadrant 1: x and y both positive
0	45.0	45.0	Border between quadrant 1 and 2
-11.3	-3.9	11.954079	Quadrant 3: x and y both negative

Hint: Notice that not all quadrants are tested in the above examples.

Part 1.2 – Calculate Vector Phase Angle

For the second part of this lab, you will complete the function `phase` which accepts two inputs, `x` and `y`, as floating-point input parameters and returns the phase angle in **radians** as calculated using equation 2.

To do this, you will need to use an inverse tangent function from Python's math module. The available trigonometric functions can be found here[3]. Note that there are two different inverse tangent or arctangent functions. Read the descriptions and determine which you believe should be used in your function. If you are not sure, your TA will be happy to discuss with you during your PRA session.

[3] https://docs.python.org/3.7/library/math.html#trigonometric-functions

Sample Test Cases

Inputs		Expected Output (to six decimal places)	Description
x	**y**	**phase**	
0	0	0.0	Zero vector
10.2	63.2	1.410784	Quadrant 1: x and y both positive
0	45.0	1.570796	Border between quadrant 1 and 2
-11.3	-3.9	-2.809260	Quadrant 3: x and y both negative

Part 2 – Problem and Motivation

In this part of the lab, you will write a function to calculate the vertical position of a charged particle in a simple electrostatic precipitator.

An electrostatic precipitator is a device that is typically used to capture and remove harmful particles from the exhaust of industrial and/or chemical processes. The device works in two stages:

1. Particles in the exhaust are charged (either positively or negatively)
2. The particles pass through an air channel where high voltage electrodes attract the charged particles (see figure 2). The electrodes create an electric field that exerts a coulomb force on charged particles.

If we assume this force is constant, the vertical height of the particle can be approximated using the following equation:

$$h(t) = -\frac{1}{20000}\frac{qE}{m}t^2 + h_{init}$$

Variables:

- q is the particle charge in nanocoulombs (can be positive or negative)
- E is the electric field strength between the electrodes in kilonewtons per coulomb
- m is the mass of the particle in nanograms
- t is the time since the particle entered the air channel in microseconds
- h_{init} is the initial height of the particle in centimetres
- h(t) is the height of the particle in centimetres at time t

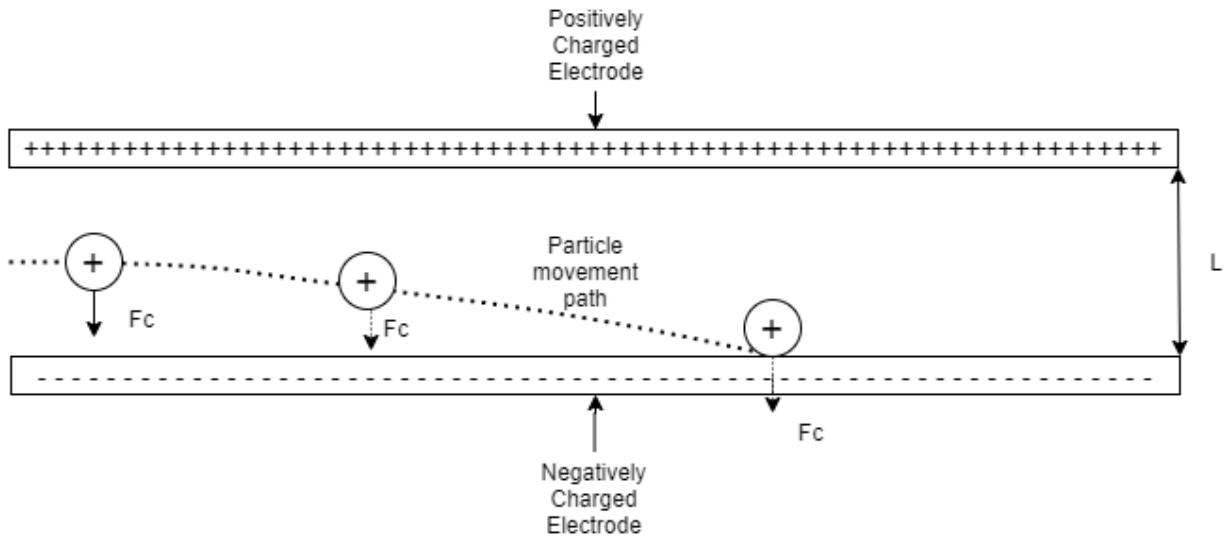

Figure 2. Example of charged particle movement between high voltage electrodes. Particles enter through the opening on the left and are attracted to the oppositely charged electrode. In this example, a positively charged particle enters in the middle of the two electrodes and is accelerated towards the negatively charged electrode by the coulomb force, Fc. When the particle reaches the electrode, it sticks to the surface and stop moving.

Your task is to complete the function `particle_height` which will calculate and **return the vertical height of a particle** within the electrostatic precipitator. The function accepts five input parameters:

1. `q` – The charge of the particle in nanocoulombs
2. `E` – The electric field strength in kilonewtons per coulomb
3. `m` – The mass of the particle in nanograms
4. `L` – The distance between the two electrodes in centimetres
5. `t` – The time since the particle entered the air channel in microseconds

For simplicity, we will assume the following:

1. The bottom electrode is at a height of 0 cm and the top electrode is at a height of L cm, L will be greater than zero
2. The top electrode will be positively charged and the bottom will be negatively charged
3. Particles enter the channel halfway between the two plates (i.e., h_{init} = L/2 cm)
4. Particles have zero vertical velocity when entering the air channel
5. **Particles immediately stop moving when contacting one of the electrodes**
6. All other forces (gravity, drag, etc.) are negligible

Sample Test Cases

Inputs					Expected Output (to six decimal places)	Description
q (nC)	**E (kN/C)**	**m (ng)**	**t (μs)**	**L (cm)**	**Height (cm)**	
0	150	9.2	3.6	5	2.5	Zero force scenario
2.3	150	9.2	26.8	5	1.1533	Positive Charge; has not yet contacted electrode
-2.3	160	9.2	36.8	5	5.0	Negative charge; attached to top electrode

Hint: This problem is designed to be completed without the use of conditional statements (i.e., if statements) as those will be introduced later in the course. For this lab, you may find the built-in Python functions `max` and `min` helpful. Use:

```
help(min)
```

```
help(max)
```

in the python interpreter to find out more about these functions.