CS410P

ASSIGNMENT 8P

Purpose

The purpose of this assignment is to have you work with NumPy arrays and matplotlib

Part A)

Scenario

When an object or particle (called a projectile) is thrown near the earth's surface, *it moves along a curved path under the action of gravity only*. The only force of significance that acts on the object is gravity, which acts downward to cause a downward acceleration. There are no horizontal forces needed to maintain the horizontal motion – consistent with the concept of inertia.

Write a Python program that initiates projectiles at various angles with a certain initial velocity and computes the X and Y displacements of the projectile at different time intervals. Plot the path of these projectiles as shown below.

For starters, I'm interested in seeing angles between 25 and 60 in degrees (excluding 60) in steps of 5, all with an initial velocity (v_0) of 30.

Here's what's available to you:

At any time t, the projectile's horizontal and vertical displacement can be approximated as follows where

$$x = v_0 t \cos(\theta),$$
$$y = v_0 t \sin(\theta) - \frac{1}{2} g t^2.$$

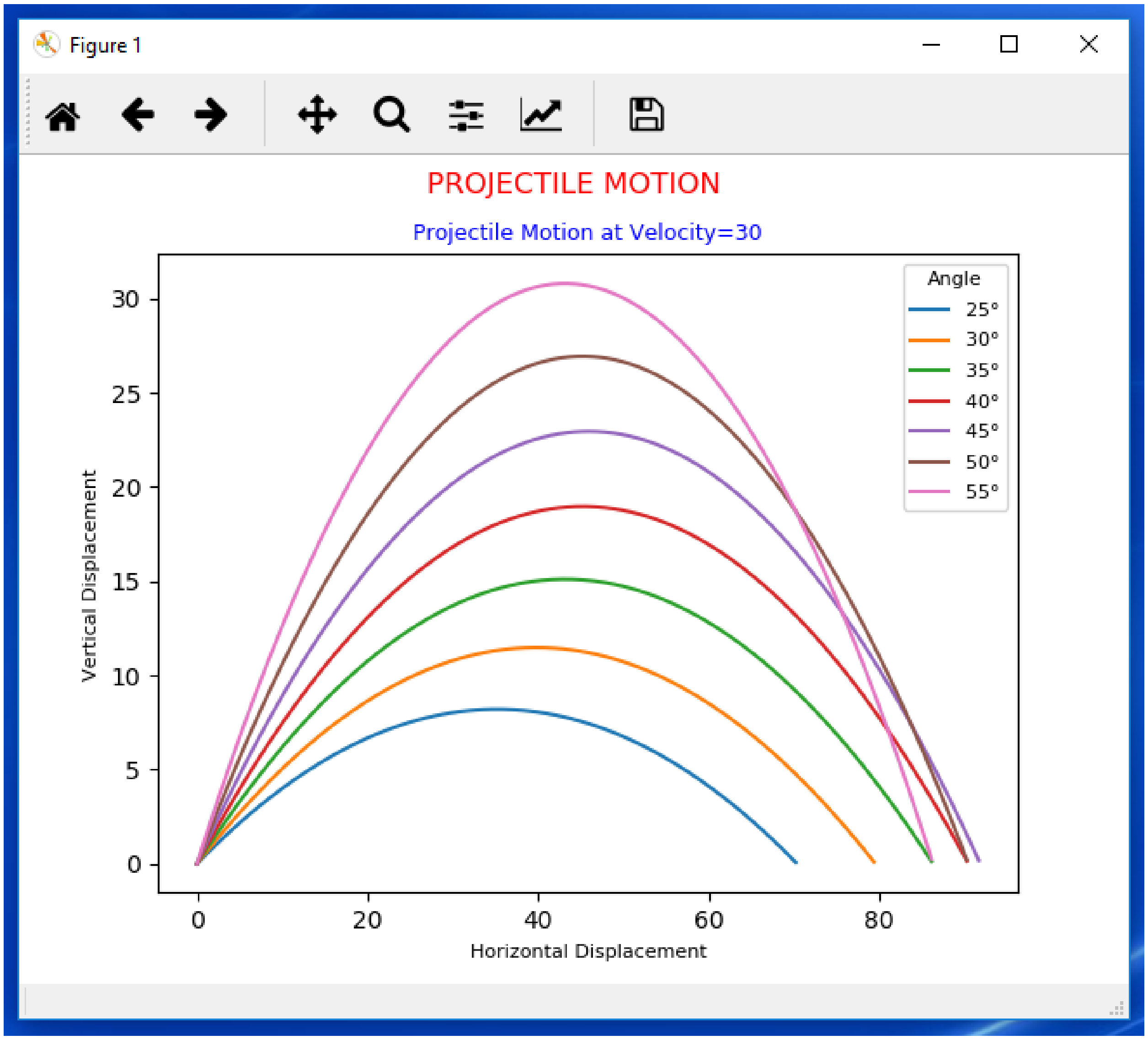
The total time that the projectile remains in the air is given by (neglecting air resistance):

$$t = \frac{2 v_0 \sin(\theta)}{g}$$

where g is the acceleration of gravity = 9.8

Dividing t into many intervals will give you a better curve, otherwise there are insufficient points to plot the motion as a continuous curve (You can assume a value for the number intervals t)

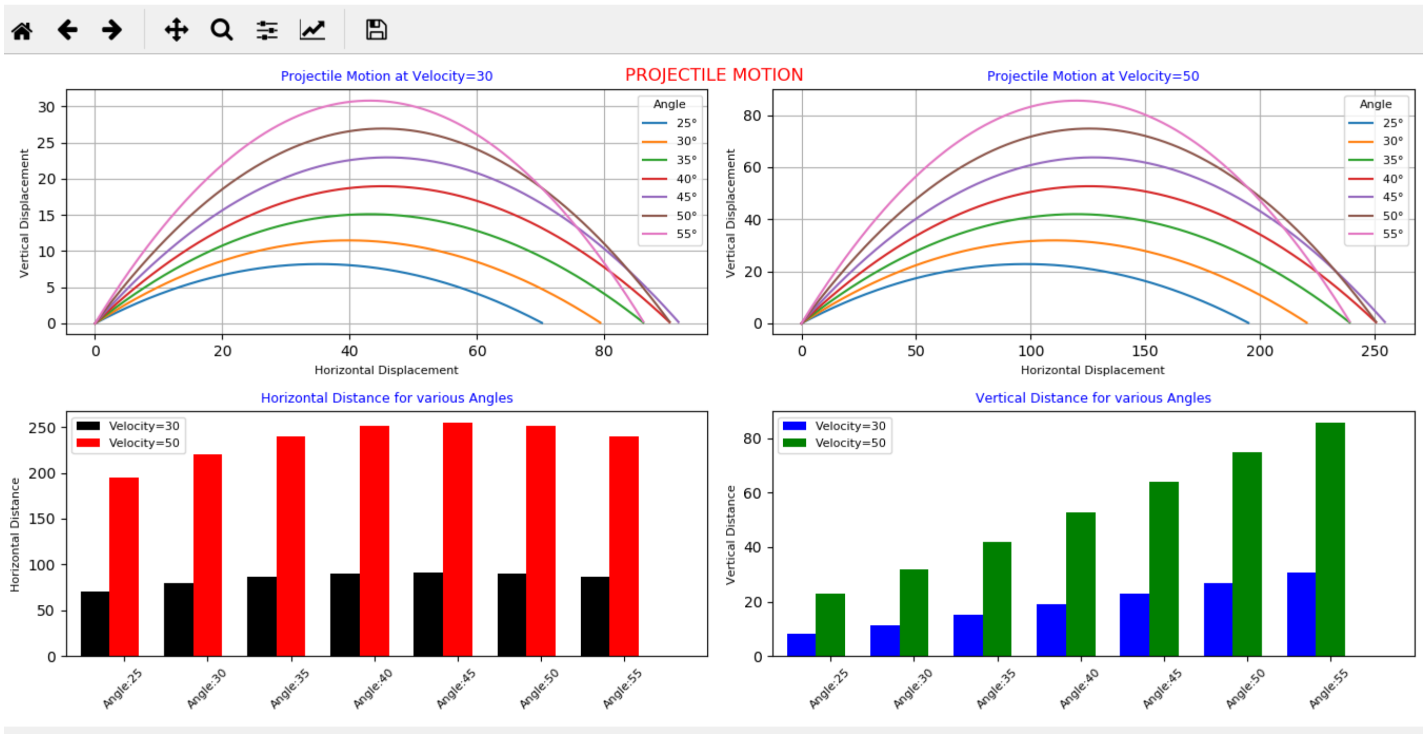
Output for the steps above:



After you get the above working, I need you to expand on your program to use one figure window (you don't need to retain the one above) to draw 4 sub-plots, one each for the following:

- Projectile motion for initial velocity of 30 for angles in degrees between 25-60 (as accomplished above)
- Projectile motion for initial velocity of 50 for angles in degrees between 25-60 (same as above except for change in initial velocity)
- Bar plot which compares maximum horizontal distance for projectiles with initial velocities of 30 and 50 (for varying angles in degrees between 25 and 60)
- Bar plot which compares maximum vertical distance for projectiles with initial velocities of 30 and 50 (for varying angles in degrees between 25 and 60)

Program Output:



Part B)

Scenario

Physics is the study of nature, in its most basic form. Mathematics is the language of Physics. We use mathematical equations to transcribe the physical laws that define the way our universe works. Often these equations are too complex for a direct solution, and so we turn to numerical methods.

Consider the simple phenomenon of a bouncing ball. Except for the interaction of the ball with the floor, the only physics involved is gravitational acceleration. The mathematical formula describing this motion is perhaps the (second) most famous equation: F=ma, where “a” is the (constant) acceleration due to gravity.

A common numerical approach to differential equations like F=ma is to rewrite the derivatives in terms of finite differences. In this case we first turn F=ma into two separate first-order differential equations, F=mdv/dt and v=dx/dt. Then the derivatives are written as:

$$\frac{dv}{dt} = F/m = -g \qquad \frac{dx}{dt} = v \qquad \longrightarrow \qquad \frac{v(t+\Delta t) - v(t)}{\Delta t} = -g \qquad \frac{x(t+\Delta t) - x(t)}{\Delta t} = v$$

These equations can then be rewritten to give the position and velocity at a time t+dt in terms of the position and velocity at a previous time t. By stepping forward in time in steps of dt, one can calculate the functions v(t), and x(t).

$$x(t + \Delta t) = x(t) + v\Delta t$$

$$v(t + \Delta t) = v(t) - g\Delta t$$

Write a Python program that executes the above numerical algorithm several times (Euler's method to compute the trajectory of a bouncing ball), stepping the trajectory of the bouncing ball through several bounces. Here are a few things you can assume:

- Perfect reflection at the surface
- Use of SI units (meters and seconds)
- End time = 3.0
- Acceleration of gravity (g) = 9.8

Create 1-dimensional arrays for time (t), position (x), velocity (v) for number of iterations needed, (each index of the array will hold the value for one iteration). The initial position can be assumed to be 1.

1) After executing the algorithm several times (several hundred? several thousand?), create a simple plot of t versus x.
2) Next, don't assume perfect reflection but say that you have 90% reflection, repeat the process i.e.

    - 90% reflection at the surface
    - End time = 3.0
    - Acceleration of gravity (g) = 9.8
    - Create a simple plot of t versus x

3) On a new figure window superimpose both plots created above. Add a third plot for 70% reflection on the same figure window and observe the results

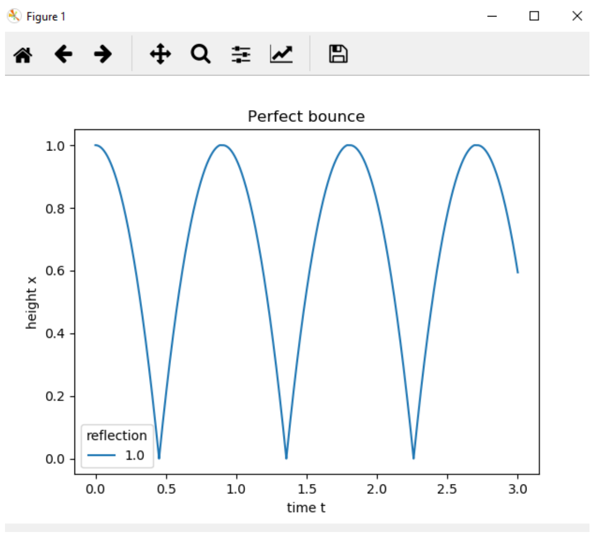
Modularity

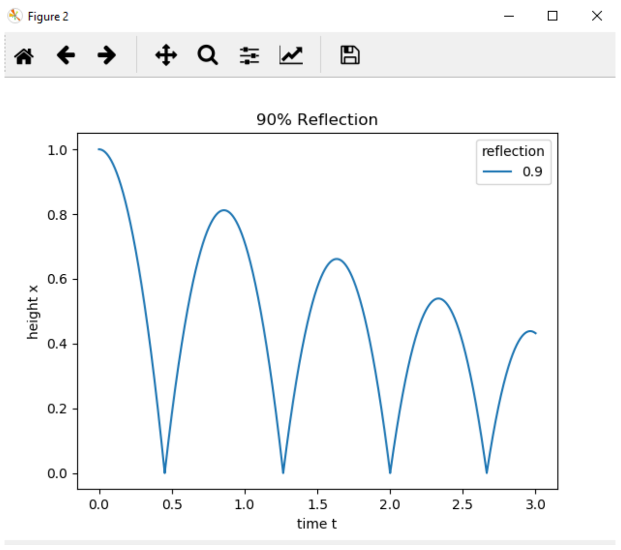
Create a function to compute and plot – call your function once each for figure windows 1 & 2, and three times for figure window 3.

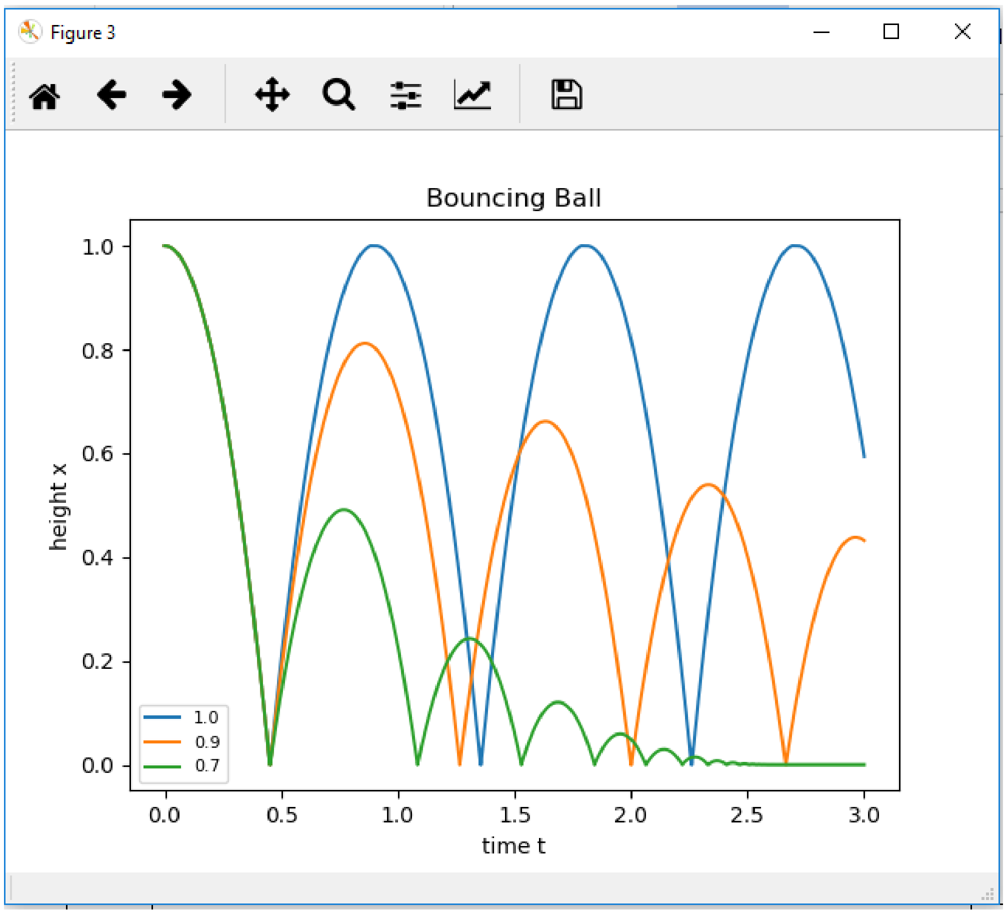
Input

None

Program Output







Grade Key:

A	Part A – Projectile (algorithm, number of time slices, accuracy of plot for different angles) for velocity=30	15
B	Part A – Projectile (Legend, Axes labels, Plot title) for velocity=30	6
C	Part A – Works correctly for velocity = 50 with same details as velocity=30	9
D	Part A – Bar plot for horizontal distance comparison, legend, axes, title	10
E	Part A – Bar plot for vertical distance comparison, legend, axes, title	10
F	Part A – Projectile (4 sub plots in 1 figure window)	10
G	Part B – Bouncing Ball (algorithm, number of time slices, accuracy of plot for different reflections): 8 points per reflection	24
H	Part B – Bouncing Ball (3 figure windows as specified, Legend, Axes labels, Plot title)	6
I	Part B – Bouncing Ball (Function created without unnecessary repetition of code for 3 different plots)	10
J	Late Penalty	