

The simulation works by first taking in the object's initial acceleration, velocity, displacement, mass, cross-sectional area, and drag coefficient, as shown in **Code Segment 1**. External resources were used to find the drag coefficient of a sphere.

#### Code segment 1:

```
steelBall = object(a0 = [0, -9.81],  
                  v0=createUnitVector([4.878, 45]),  
                  s0 = [0, 0.146 + 0.908], m=13.7/1000,  
                  crossArea=(math.pi*((0.75/100)**2)),  
                  dragCoefficient=0.47)  
  
steelBall.runSimulation()
```

There are multiple steps that occur when the simulation first runs. To better visualize this, the following table, **Table 1**, will contain the data simulated by the program.

Table 1 - Headers of the data simulated by program						
Time (s)	X Displacement (m)	Y Displacement (m)	X Velocity (m/s)	Y Velocity (m/s)	X Acceleration (m/s <sup>2</sup> )	Y Acceleration (m/s <sup>2</sup> )

The initial parameters are then inserted into the simulated data table, shown in **Table 2**

Table 2 - Initial displacement, velocity, and acceleration data inserted						
Time (s)	X Displacement (m)	Y Displacement (m)	X Velocity (m/s)	Y Velocity (m/s)	X Acceleration (m/s <sup>2</sup> )	Y Acceleration (m/s <sup>2</sup> )
0	0	1.054	3.45	3.45	0	-9.81

**Table 2** visualizes the initialization of each data value.

The program will then start to calculate the forces upon the marble for the next 0.001 second time interval. These calculations are done by both **code segments 2 & 3**:

### code segment 2:

```
def getAirResistance(self, velocityVector):
    airResistance = []
    for dimensionalVelocity in velocityVector:
        directionalAirResistance =
-0.5*1.293*self.dragCoefficient*self.crossArea*(dimensionalVelocity**2)
        if dimensionalVelocity < 0:
            directionalAirResistance *= -1
        airResistance.append(directionalAirResistance)

    return airResistance
```

Code segment 2 uses Equation 1 to find drag force in each direction to create one vector:

### Equation 1:

$$F = \frac{1}{2} C_d \rho_{air} A v^2$$

$$F = \frac{1}{2} * 1.293 * .47 * (.018) * 3.45^2$$

$$F = 0.006 \text{ N}$$

### Code segment 3:

```
def getForceOfGravity(self):
    return [0, -9.81 * self.mass]
```

Code segment 3 finds the force of gravity acting upon the marble, using Equation 2:

### Equation 2:

$$F = mg$$

These forces are then added to a table, visualized by Table 3

Table 3 - Forces		
	Force in X direction (N)	Force in Y direction (N)
Force of gravity	0	-0.134

Force of air resistance	-0.006	-0.006
-------------------------	--------	--------

**Table 3** shows the forces that are exerted upon the object. The calculations for those values are done in **code segments 2 & 3**.

The forces are then summed by their directions and added to **Table 4**:

Table 4 - Net Force in each Direction	
Force in X direction	Force in Y direction
-0.006	-0.140

**Table 4** includes the net force in each direction. The accelerations in both the x and y directions can be calculated by dividing each value by the mass of the object.

**Equation 3:**

$$F/m = a$$

$$\frac{0.140}{0.0137} = 10.21 \text{ m/s}^2$$

Using Equation 3, **Table 5** can be created.

Table 5 - Acceleration in each direction	
X Acceleration (m/s <sup>2</sup> )	Force in Y direction
-0.43	-10.21

This data can now be appended to **Table 2**, creating **Table 6**.

Table 6 - Data from program						
Time (s)	X Displacement (m)	Y Displacement (m)	X Velocity (m/s)	Y Velocity (m/s)	X Acceleration (m/s <sup>2</sup> )	Y Acceleration (m/s <sup>2</sup> )
0	0	1.054	3.45	3.45	0	-9.81
0.001					-0.43	-10.21

**Table 6** shows **Table 2**, updated with acceleration values from **code segment 3**.

#### Code Segment 4:

```
def getVelocity(self, timeInterval, v0, a0):

    velocityX = a0[0] * timeInterval + v0[0]
    velocityY = a0[1] * timeInterval + v0[1]

    return [velocityX, velocityY]
```

Code Segment 4 gets the new velocity using **Equation 3**, using the previously calculated acceleration value and the time interval of 0.001 seconds.

#### Equation 3:

$$v = v_0 + at$$

$$v = 3.45 - 10.21 * 0.001$$

$$v = 3.45 - 10.21 * 0.001$$

$$v = 3.44 \text{ m/s}$$

The simulated data is updated again, and this update is represented in **Table 7**.

Table 7 - Data from program						
Time (s)	X Displacement (m)	Y Displacement (m)	X Velocity (m/s)	Y Velocity (m/s)	X Acceleration (m/s <sup>2</sup> )	Y Acceleration (m/s <sup>2</sup> )
0	0	1.054	3.45	3.45	0	-9.81
0.001			3.45	3.44	-0.43	-10.21

**Table 7** is a representation of the data simulated by the program, updated with calculated velocity values.

To calculate displacement, **Equation 4** can be used:

$$x = 0.5at^2 + v_0t + x_0$$

$$x = 0.5(-9.81)(.001)^2 + (3.44)(.001) + (1.054)$$

$$x = 1.057 \text{ m}$$

**Equation 4** is implemented in **Code Segment 5**:

### Code Segment 5:

```
def getDisplacement(self, timeInterval, x0, v0, a0):  
  
    displacementX = 0.5 * a0[0] * timeInterval ** 2 + v0[0] *  
timeInterval + x0[0]  
  
    displacementY = 0.5 * a0[1] * timeInterval ** 2 + v0[1] *  
timeInterval + x0[1]  
  
    return [displacementX, displacementY]
```

**Code segment 5** will return the x displacement and y displacement as a vector. The simulated data is updated, shown in Table 8

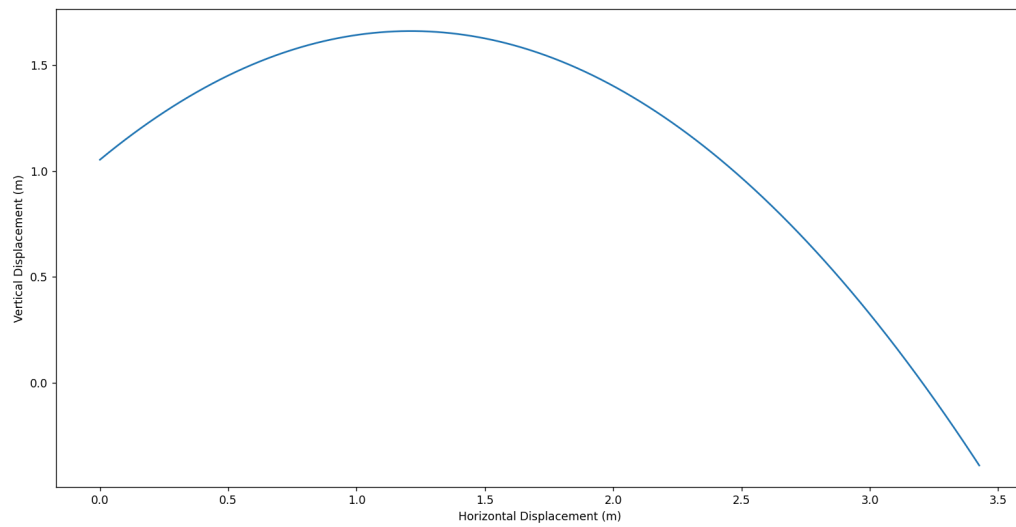
Table 8 - Data simulated from program						
Time (s)	X Displacement (m)	Y Displacement (m)	X Velocity (m/s)	Y Velocity (m/s)	X Acceleration (m/s <sup>2</sup> )	Y Acceleration (m/s <sup>2</sup> )
0	0	1.054	3.45	3.45	0	-9.81
0.001	0.003	1.057	3.45	3.44	-0.43	-10.21


**Table 8** is a representation of the data simulated by the program, updated with calculated displacement values.

**Code Segments 2-5** repeat until one second of projectile motion data is simulated. When graphed, the data appears as below:

### Graph 1 - Horizontal Displacement vs. Vertical Displacement

Figure 1



 (x, y) = (3.577, 1.742)

**Graph 1** is a simulated trajectory of the marble. According to the graph, the marble, when launched from the floor, would reach the floor at a horizontal displacement of 2.55 m.

When launched from a table, it will reach a horizontal displacement of 3.2 m.