PHYS 216: Homework 2

1. Analytically found final values:
   1. Mathematics:

For an initial x velocity of 60\*cos(π/6) and initial z velocity of 60\*sin(π/6), this gives us a final distance of 318.13m.

* 1. Explanation:

The mathematics contained within this basic equation is simply algebraic manipulation. Using the equation for the z distance of an object in ballistic motion with only gravity, we can find the final time by setting the equation to zero. Once this is received, it can be plugged into the equation for the x distance to get the final x distance.

1. Write fixed time step Euler integrator:
2. Code:

**def EulerMethod(func, prevt, finalt, prevy, step):**

**nexty = prevy + step\*np.array(func(prevy, prevt))**

**if nexty[3]<=0:**

**return np.array([nexty])**

**elif prevt <= finalt+step:**

**return np.append(np.array([nexty]),**

**EulerMethod(func, prevt+step, finalt, nexty, step),**

**axis=0)**

**else:**

**return np.array([nexty])**

**def projectileMotionDifferential(vector, time):**

**xvelocity = vector[0]**

**yvelocity = vector[1]**

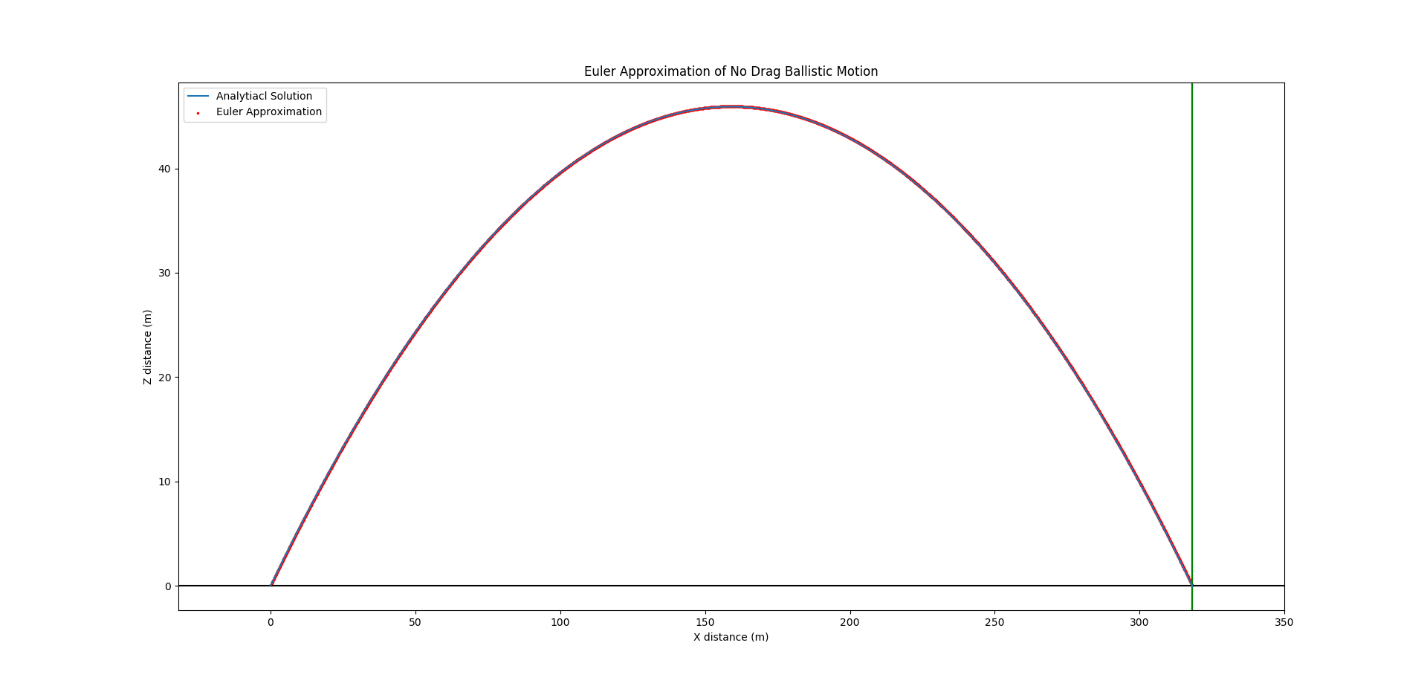
**return [0, GRAVITY, xvelocity, yvelocity]**

1. Explanation:

The code above describes two separate methods, one that performs a Euler method and another that describes the differential equation. The Euler method takes a differential equation func, initial time prevt, final time finalt, initial condition prevy, and step step and calculates the points using the approximation. It first calculates the next vector in the series before checking the conditions describing if it needs to stop. This particular Euler method stops either when the value of the z-coordinate drops below zero or when the final time step has been reached.

The equation is defined in the projectileMotionDifferential function. The function takes a vector in the form (x-velocity, z-velocity, x-coordinate, z-coordinate) and outputs the result of the system of differential equations in the form (x-accleration, z-acceleration, x-velocity, z-velocity).

1. Graph:

The following is a graph of the projectile motion differential equation plotted with both the analytical solution and the Euler approximation:

The blue line depicts the analytical solution while the red dots behind it shows the many points plotted with the Euler approximation. The green/blue line on the right-hand side of the graph shows the predicted final value of x distance that we calculated above. The above plot converges in 3271 points.

1. Use Euler integrator to solve for trajectory with gas drag included:
   1. Code:

**def dragMotionDifferential(vector, time):**

**xvelocity = vector[0]**

**zvelocity = vector[1]**

**velocity = math.sqrt(xvelocity\*\*2+zvelocity\*\*2)**

**return [-1/2\*DRAG\*DENSITY\*SPHEREAREA/SPHEREMASS\*velocity\*xvelocity,**

**GRAVITY-1/2\*DRAG\*DENSITY\*SPHEREAREA/SPHEREMASS\*velocity\*zvelocity,**

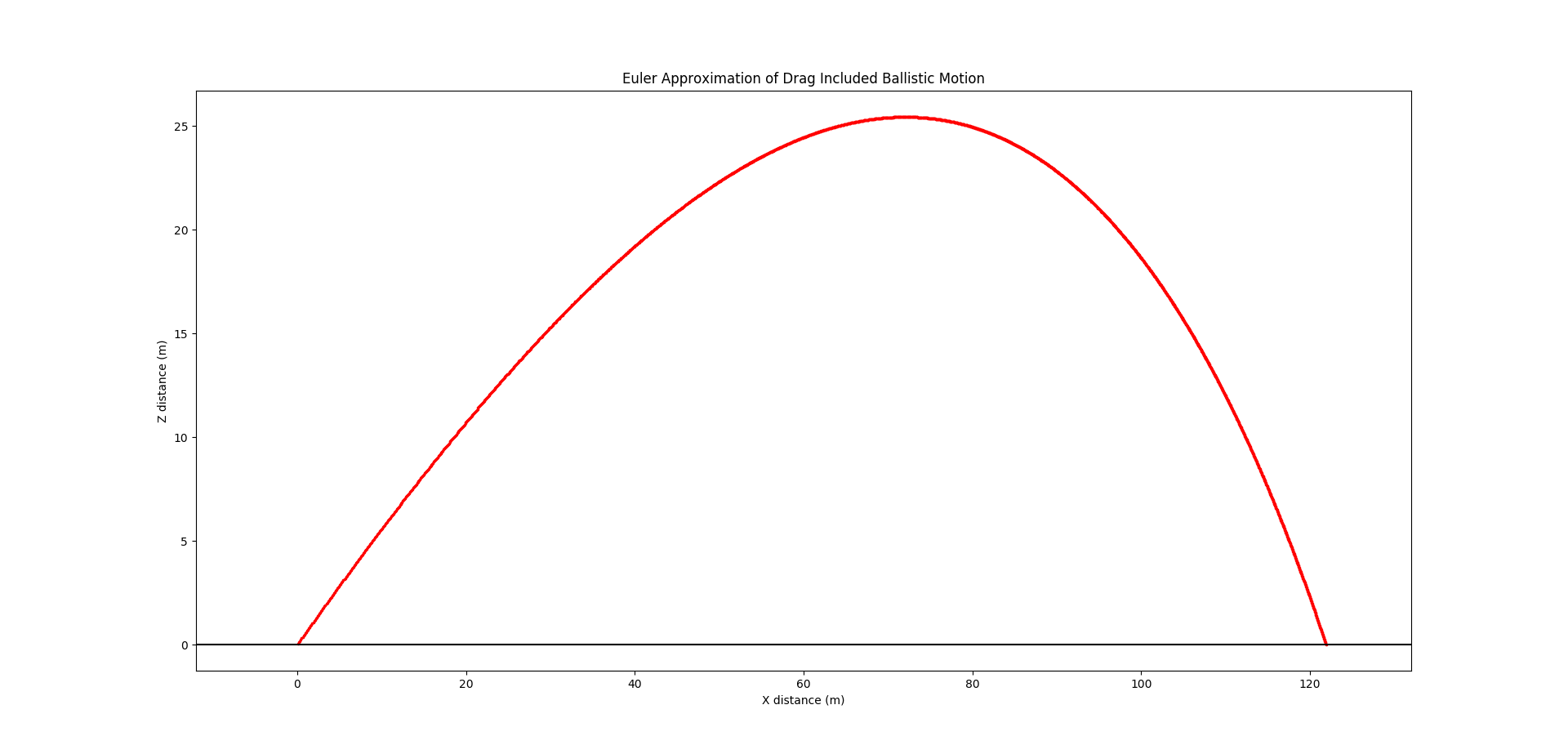
**xvelocity, zvelocity]**

* 1. Explanation:

The code above is the function used to describe the system of differential equations of ballistic motion with a gas drag component. The constants DRAG, DENSITY, SPHEREAREA, SPHEREMASS, and GRAVITY are defined elsewhere in the code as their respective values given in the problem. This is meant to streamline the process of changing the situation that is being plotted without compromising the equation in the meantime. The function works very similarly to the projectile motion equation from the last problem. It takes in a vector in the form (x-velocity, z-velocity, x-coordinate, z-coordinate) and outputs the vector (x-acceleration, z-acceleration, x-velocity, z-velocity) that is the result of the given system of differential equations.

* 1. Graphs:

The following graph depicts the resulting trajectory of the drag equation when evaluated with the Euler method of approximation:

 The above computation converges in 2202 points. The red points are the results of the method while the black line shows the zero point of the z component. The final x and z velocities are 15.377 m/s and -18.916 m/s, respectively. The final calculated x distance and z distance is 121.896m and 0.009m, respectively.

1. Write and use predictor-corrector integrator:
   1. Code:

**def RungeKuttaMethod(func, prevt, finalt, prevy, step):**

**k1 = np.array(func(prevy, prevt))**

**k2 = np.array(func(prevy+step\*k1/2, prevt+step/2))**

**k3 = np.array(func(prevy+step\*k2/2, prevt+step/2))**

**k4 = np.array(func(prevy+step\*k3, prevt+step))**

**nexty = prevy+step\*(k1+2\*k2+2\*k3+k4)/6**

**if nexty[3]<=0:**

**return np.array([nexty])**

**elif prevt <= finalt+step:**

**return np.append(np.array([nexty]),**

**RungeKuttaMethod(func, prevt+step, finalt, nexty,**

**step),**

**axis=0)**

**else:**

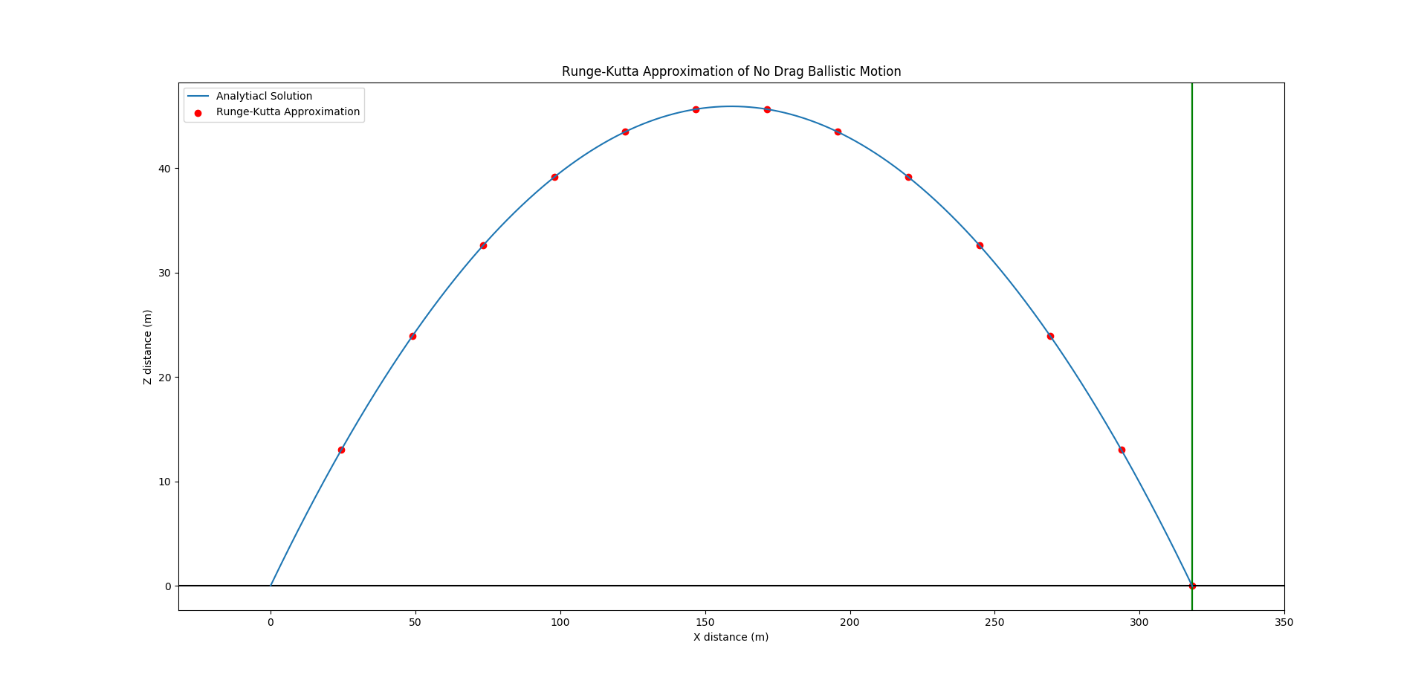
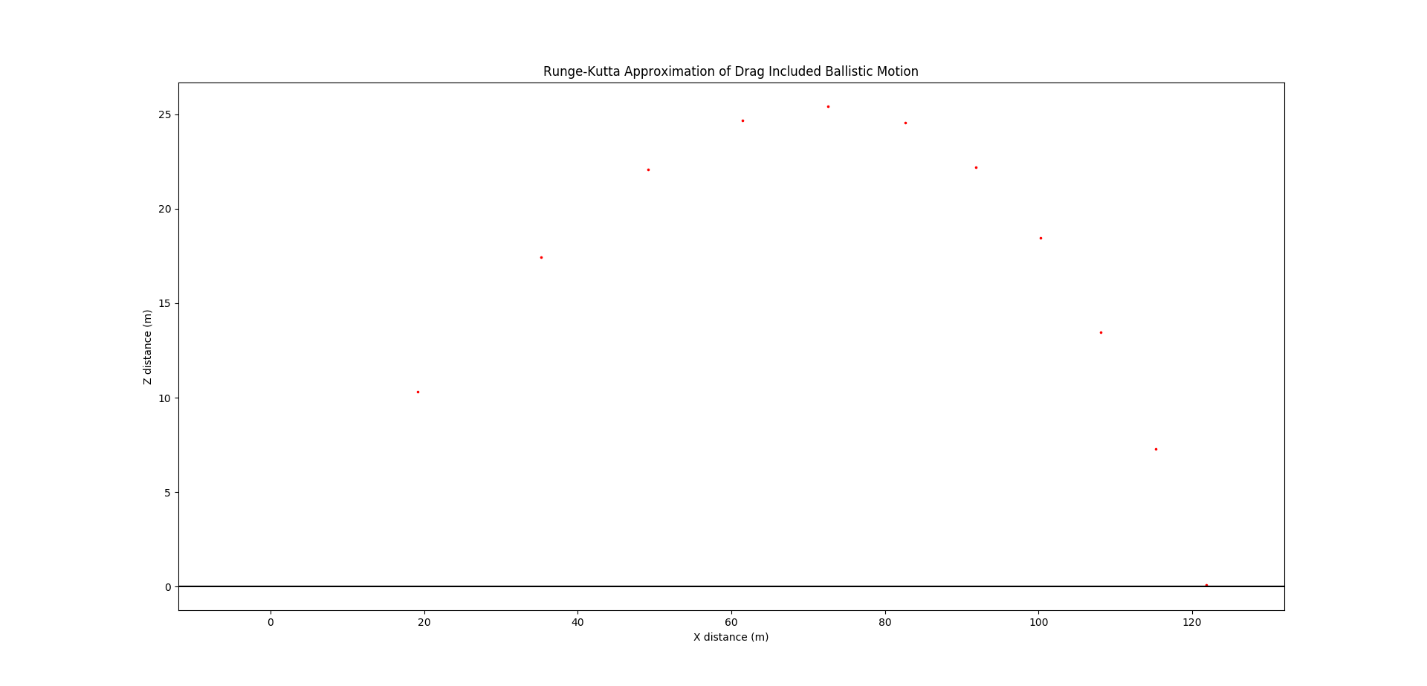
**return np.array([nexty])**

* 1. Explanation:

The above method is a four-stage predictor corrector method also known as the Runge-Kutta method. The Runge-Kutta method contains two additional prediction/correction terms instead of the simple two that is included in Heun’s method or the Improved Euler method. This equation works similarly to the Euler method. It takes in a differential function, initial time prevt, final time finalt, initial condition prevy, and step step. Then it outputs the array full of all the vectors for the approximation.

* 1. Graphs:

The first graph shows the Runge-Kutta approximation of the no drag equation:

The Runge-Kutta approximation reaches convergence at 13 points. This gives the following numbers: final x velocity of 51.962 m/s, final z velocity of -30 m/s, final x distance of 318.132 m, final z distance of 0 m. The next plot displays the Runge-Kutta approximation of the drag function:

This approximation converges at 22 points. It has a final x velocity of 15.397 m/s, final y velocity of -18.884 m/s, final x distance of 121.834 m, and final z distance of 0.074 m.