PHYS2600: Homework 2*

Due on Thursday, Feb, 15th 2018

Yangrui Hu

*This Latex template is from http://www.latextemplates.com

Contents

Problem 1

Problem: Use the both the Euler and RK4 methods to calculate the trajectory of a projectile ignoring the effect of drag. Compare your results with the exact solution. For this problem you must implement your own RK4 routine.

Solution: In this problem, I set the mass is 1.0 kg, the gravitational acceleration is 9.8 m/s^2, and the initial velocity of the projectile is 100 m/s. Because all of these four problems in this homework are to study the trajectories of a projectile in different conditions, I write a general code which I can use to solve all these four problems. The code is shown as Listing 1 at the last section of my report.
I set three different initial angles with respect to the horizontal axis: 70°, 45°, and 30°. Because in the second problem, the projectile should be launched to high altitudes, I set the initial position is $x = 0\,m,\ y = 10000\,m$. I use three different methods to do calculation: Euler method, RK4 method, and the analytical solution. The analytical solution for this problem is:

$$\begin{cases} x(t) = v_0 \cos\theta_0 t, \\ y(t) = y_0 + v_0 \sin\theta_0 t - \dfrac{1}{2} g t^2, \end{cases} \tag{1}$$

Figure 1 shows my results. Comparing three different calculation methods, they give the same results. So both Euler method and RK4 method can give the exact solution.

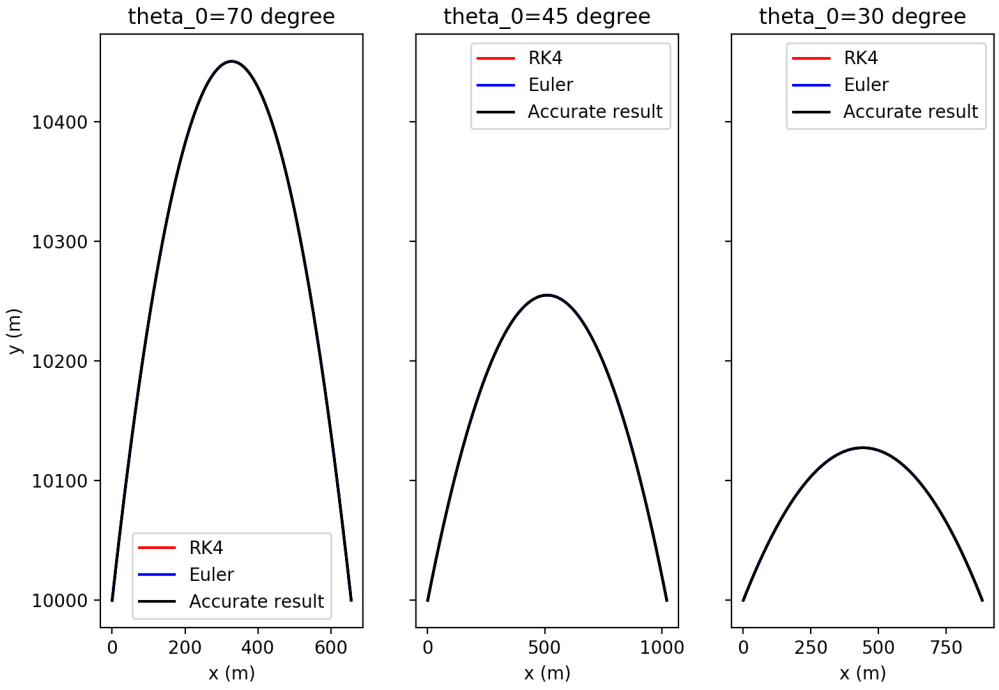

Figure 1: Trajectories of a projectile without drag force

Problem 2

Problem: Assuming that the air in the atmosphere is a poor conductor of heat and that convection is very slow, we can use the adiabatic approximation to find the dependence of density on altitude,

$$\rho(y) = \rho_0(1 - \frac{ay}{T_0})^{\alpha}, \tag{2}$$

where $a \sim 6.5 \times 10^3 K/m$ is determined empirically. T_0 is the temperature at sea level in K and $\alpha = 2.5$ for air.
The drag force acting on the projectile is, $F_D = 0.5 C_D \rho(y) A v \vec{v}$, $C_d = 0.5$ is a constant, and A is the cross-sectional area of the projectile. Use the adiabatic model of the air density to calculate the trajectory of a projectile launched to high altitudes. Compare with results found in question 1. How much effect will the adiabatic model have on the maximum range and the launch angle to achieve it?

Solution: In this problem, I use the same parameter as problem 1. Figure 2 shows my calculation results with considering the drag force. Compare with results found in problem 1 (Figure 1), I find:

- The maximum range of the projectile is much smaller than without drag force. For $\theta_0 = 70°$, the ratio of amplitudes is approximately 3%; for $\theta_0 = 45°$, the ratio is approximately 4%; for $\theta_0 = 30°$, the ratio is approximately 5%.
- Trajectories of the projectile with drag force are not symmetric about a line parallel to y axis, while trajectories without drag force are symmetric.
- Trajectories with drag force tilt towards x-direction.

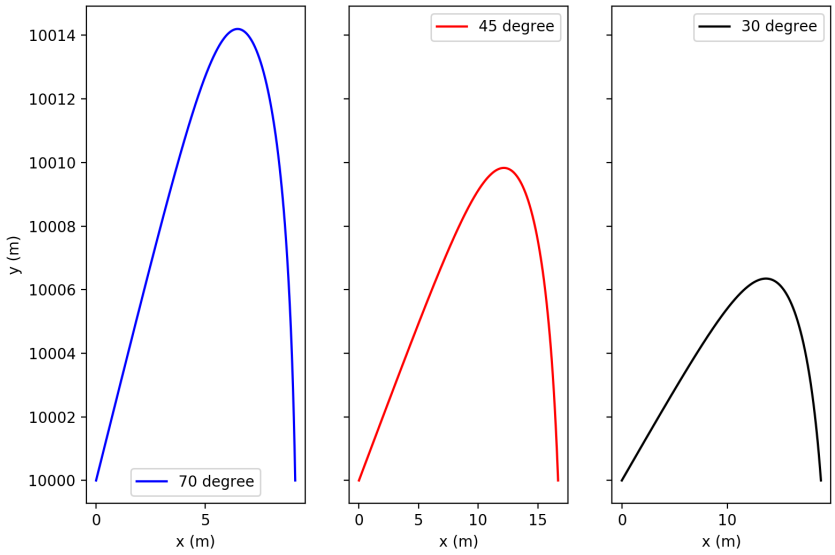

Figure 2: Trajectories of a projectile with drag force

Problem 3

Problem: Up until this point, we have assumed that the acceleration due to gravity, g, is a constant. Of course g will depend on altitude. Add this to your projectile model and determine how much it affects the range.

Solution: Consider the effect of the altitude, the modified gravitational acceleration is

$$g_h = g_0(\frac{r_e}{r_e + h})^2 \tag{3}$$

where h is the height above sea level, $r_e = 6371000\ m$ is the Earth's mean radius, and $g_0 = 9.8\ m/s^2$ is the standard gravitational acceleration (Data source: Wikipedia). Use this modified formula to simulate trajectories and Figure 3 is my result. Compare Figure 3 and Figure 2, we cannot distinguish the effect of changed g. It is also not difficult to understand: $r_e/(r_e + h) = 1 - h/(r_e + h)$ is so small for $h \sim 10000 \ll r_e$ that $r_e/(r_e + h) \sim 1$.

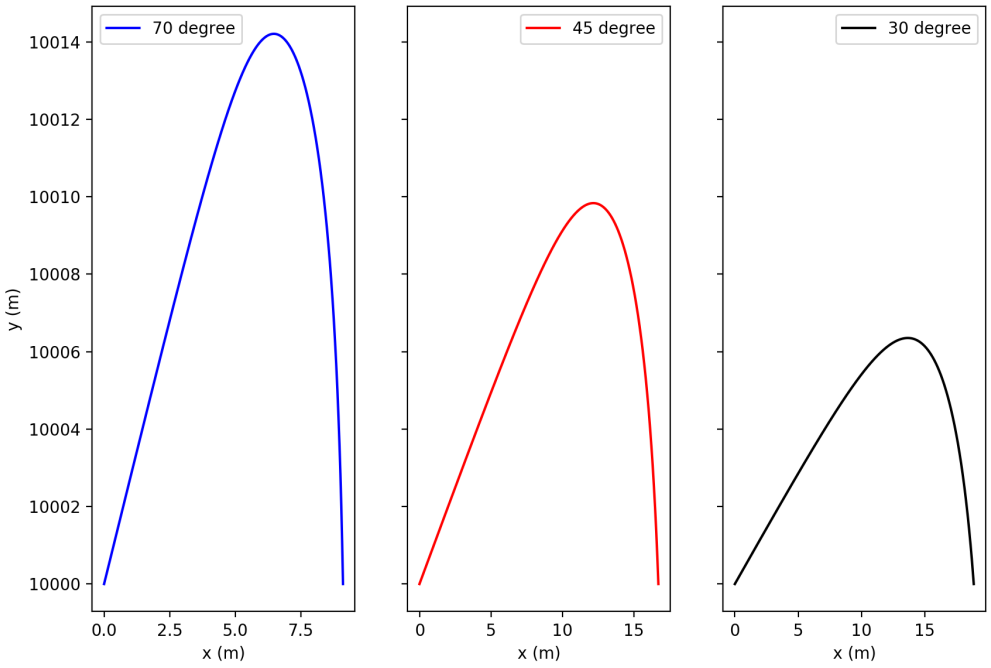

Figure 3: Trajectories of a projectile with drag force and variable g

Problem 4

Problem: (a) The drag coefficient, C_D for a baseball can be modeled with the relation

$$C_d = a + \frac{b}{1 + exp(\chi)} - c \times \begin{cases} \exp(-\chi^2), \ \chi < 0, \\ \exp(-\chi^2/4), \ \chi > 0, \end{cases} \tag{4}$$

where $\chi = (vv_c)/4$, $v_c = 34m/s$, $a = 0.36$, $b = 0.14$, and $c = 0.27$. v_c is the critical speed.
It has also been empirically determined that the lift coefficient C_L depends on the spin parameter, $S = r\omega/v$, in the following way

$$C_L = 0.6 \times S^{0.7}. \tag{5}$$

The Magnus force is $F_M = 0.5C_L\rho Ar/S\vec{\omega} \times \vec{v}$.Investigate the combined effects of quadratic drag and backspin on a fastball. How much does an angular velocity of 1000 rpm affect the trajectory? You can take g, ρ, and T to be a constant for this problem.
(b) Is it reasonable to assume the angular velocity is constant? If the total torque opposing the rotational motion is proportional to the angular velocity, how should the angular velocity depend on time? Explore the effect of a variable angular velocity in your model.

Solution: Consider the trajectory of a baseball. Using the given model, I calculate trajectories with the initial velocity 30 m/s. The mass is 0.1 kg, $g = 9.8\ m/s^2$, $\rho = 1.225\ kg/m^3$, and $r = 0.06\ m$. If the angular velocity is the constant 1000 rpm, calculation results with different initial angle are shown as Figure 4. In order to study the effect of angular velocity, Figure 5 shows results without Magus force. Compare Figure 4 and Figure 5, I find:

- Magus force makes the maximum range of y larger and the maximum range of x smaller because the direction of Magus force is $\vec{\omega} \times \vec{v}$. The x-component is negative and y-component is positive.
- Effect of Magus force on the shape of trajectory is larger as θ_0 larger, because x-component of Magus force is larger as θ_0 larger which influences the shape a lot.

However, it is not reasonable to assume the angular velocity is constant, because frictional force will make ω smaller and smaller. If consider the total torque opposing the rotational motion which is proportional to the angular velocity, the angular velocity will depend on time:

$$\omega(t) = \omega(0) \exp(-t/\tau) \tag{6}$$

where τ is characteristic time and I set $\tau = 1.0$ in my simulations. Figure 6 shows the simulation result. Compare Figure 6, Figure 4, and Figure 5, and we can see that decaying ω will weaken the effect of Magus force because Magus force is proportional to ω.

Problem 4 continued on next page...

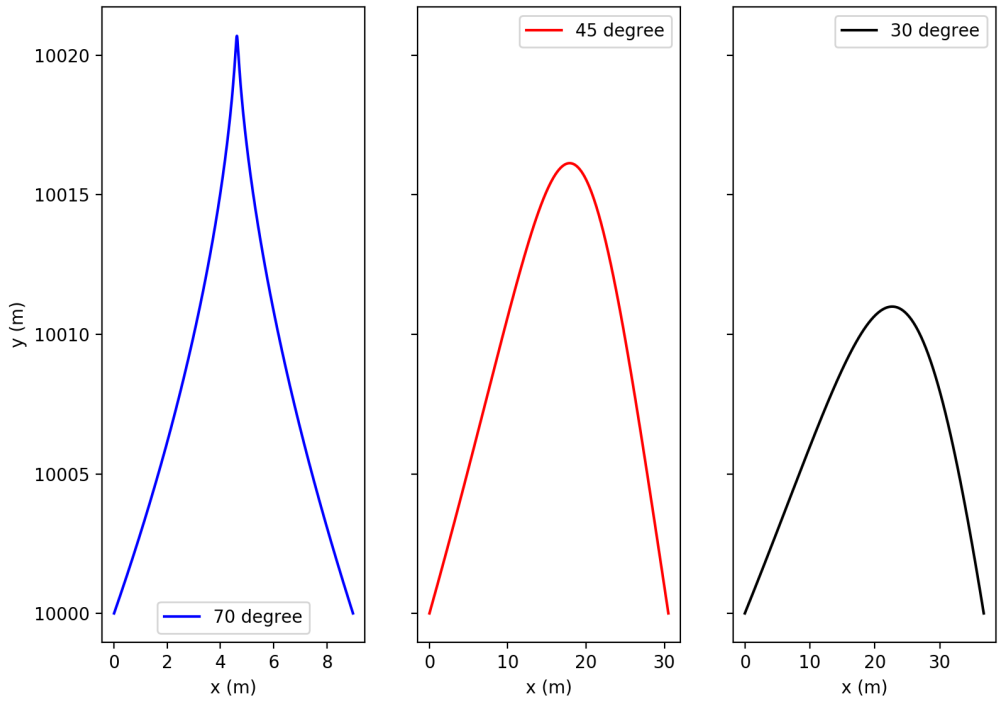

Figure 4: Trajectories of a baseball with drag force and Magus force

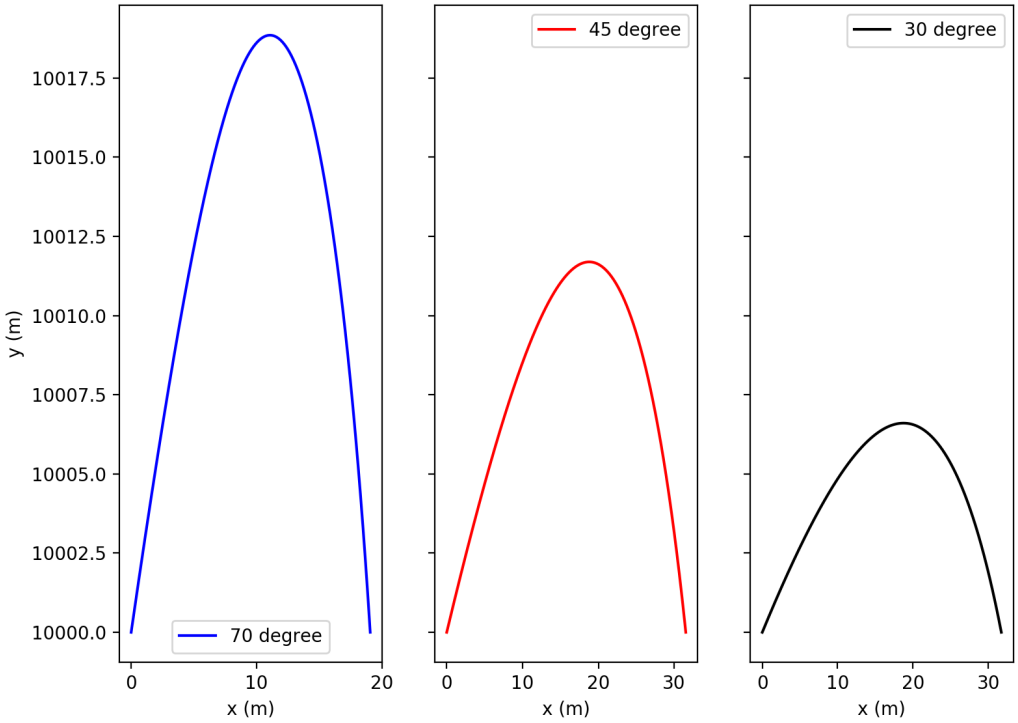

Figure 5: Trajectories of a baseball with drag force only

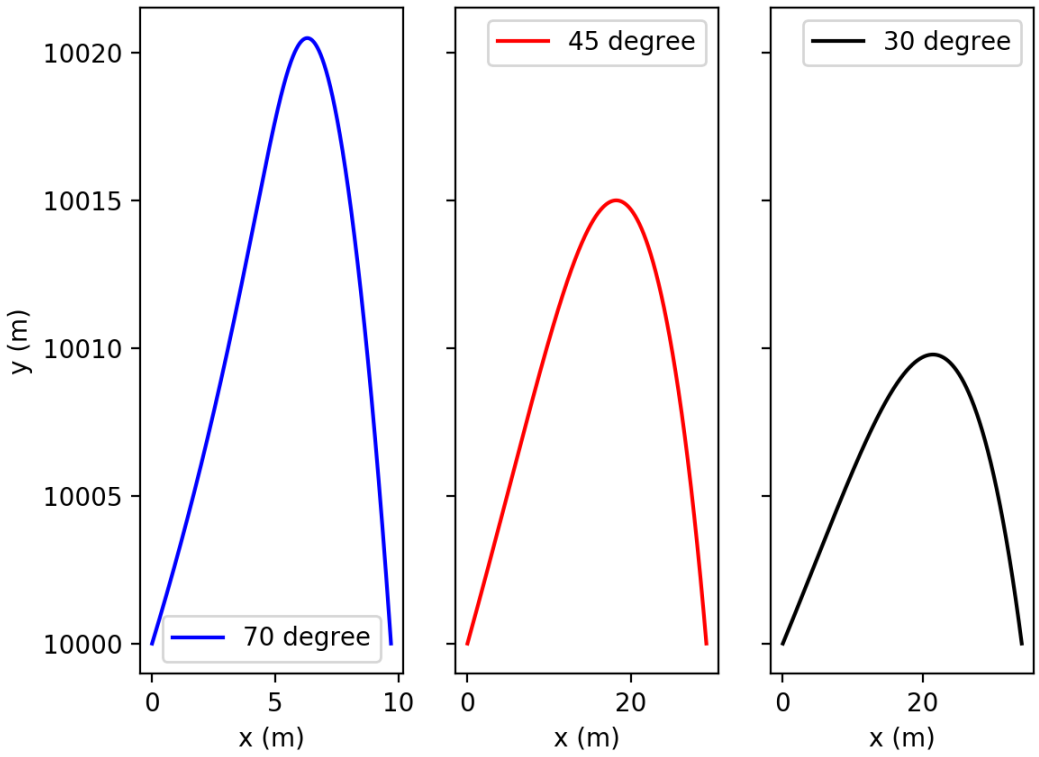

Figure 6: Trajectories of a baseball with drag force, Magus force and torque

Listing 1: Original script for all of 4 questions in this homework

```
import numpy as np
import matplotlib.pyplot as plt

class Projectile(object):

    def __init__(self):

        self.x1 = []
        self.y1 = []
        self.x2 = []
        self.y2 = []
        self.x3 = []
        self.y3 = []


    def drag(self, R, v, t):
        g_variable = 1
        baseball = 1
        d = 1 #if d=0, no drag force; if d=1, add drag force

        if (baseball == 0):
            r_e = 6371000
            if (g_variable == 1):
                g = 9.807*(r_e/(r_e+R[1]))**2
            if (g_variable == 0):
                g = 9.8
            A = 1.0
            rho0 = 1.225
            a = 6.5*0.001
            T0 = 300.0
            alpha = 2.5
            Cd = 0.5
            rho = rho0 * np.power((1.0-a*R[1]/T0), alpha)
            #return [ax,ay]
            FD = 0.5*Cd*rho*A*np.sqrt(v[0]*v[0]+v[1]*v[1])
            return np.array([-d * FD * v[0],-g - d * FD*v[1]])

        if (baseball == 1):
            m = 0.1
            g = 9.8
            rho = 1.225

            r = 0.06
            A = np.pi * r * r

            v_ab = np.sqrt(v[0]*v[0]+v[1]*v[1])
            v_c = 34
            chi = (v_ab - v_c)/4

            if (chi < 0):
                Cd = 0.36 + 0.14/(1+np.exp(chi))-0.27*np.exp(-chi**2)
            if (chi > 0):
```

```
                Cd = 0.36 + 0.14/(1+np.exp(chi))-0.27*np.exp(-chi**2/4.0)

            #drag force:
            FD = -0.5*Cd*rho*A*v_ab

            #Magnus force:
            #omega = 1000*2*np.pi/60.0
            omega = 1000*2*np.pi/60.0*np.exp(-t)
            S = r * omega / v_ab
            CL = 0.6 * np.power(S,0.7)
            #FM = 0
            FM = 0.5*CL*rho*A*r/S
            return np.array([(FD * v[0] - omega * v[1] * FM)/m,
                -g + (FD * v[1] + omega * v[0] * FM)/m])


    def Euler_trajectory(self, v0 = 100, theta0 = 45):

        vx = v0 * np.cos(theta0/180.0*np.pi)
        vy = v0 * np.sin(theta0/180.0*np.pi)
        R = [0.0,10000.0] #[x,y]
        v = [vx, vy]
        t = 0.0
        dt = 0.0001
        while ( R[1] >= 10000.0 ):
            self.x1.append(R[0])
            self.y1.append(R[1])
            v[0] += self.drag(R, v, t)[0] * dt
            v[1] += self.drag(R, v, t)[1] * dt
            R[0] += v[0] * dt
            R[1] += v[1] * dt
            t += dt

    def RK4_trajectory(self, v0 = 100, theta0 = 45):

        vx = v0 * np.cos(theta0/180.0*np.pi)
        vy = v0 * np.sin(theta0/180.0*np.pi)
        R = [0.0,10000.0] #[x,y]
        v = [vx, vy]
        t = 0.0
        dt = 0.0001
        vtemp = [0, 0]
        while ( R[1] >= 10000.0 ):
            self.x2.append(R[0])
            self.y2.append(R[1])

            K1x = dt * self.drag(R, v, t)[0]
            vtemp[0] = v[0] + K1x / 2.0
            vtemp[1] = v[1]
            K2x = dt * self.drag(R, vtemp,  t + dt / 2.0)[0]
            vtemp[0] = v[0] + K2x / 2.0
            vtemp[1] = v[1]
            K3x = dt * self.drag(R, vtemp, t + dt / 2.0)[0]
```

```
            vtemp[0] = v[0] + K3x
            vtemp[1] = v[1]
            K4x = dt * self.drag(R, vtemp, t + dt)[0]

            K1y = dt * self.drag(R, v, t)[1]
            vtemp[0] = v[0]
            vtemp[1] = v[1] + K1y / 2.0
            K2y = dt * self.drag(R, vtemp, t + dt / 2.0)[1]
            vtemp[0] = v[0]
            vtemp[1] = v[1] + K2y / 2.0
            K3y = dt * self.drag(R, vtemp, t + dt / 2.0)[1]
            vtemp[0] = v[0]
            vtemp[1] = v[1] + K3y
            K4y = dt * self.drag(R, vtemp, t + dt)[1]

            R[0] += v[0] * dt
            R[1] += v[1] * dt
            v[0] += (K1x + 2 * K2x + 2 * K3x + K4x)/6.0
            v[1] += (K1y + 2 * K2y + 2 * K3y + K4y)/6.0
            t += dt


    def Exact_trajectory(self, v0 = 100, theta0 = 90):

        vx0 = v0 * np.cos(theta0/180.0*np.pi)
        vy0 = v0 * np.sin(theta0/180.0*np.pi)
        t = 0.0
        dt = 0.01
        x = 0
        y = 10000
        while ( y >= 10000.0 ):
            self.x3.append(x)
            self.y3.append(y)
            x = vx0*t
            y = -0.5*9.8*t*t+vy0*t + 10000
            t += dt

    def clear(self):
        self.x1 = []
        self.y1 = []
        self.x2 = []
        self.y2 = []
        self.x3 = []
        self.y3 = []

    def plot2D(self):

        f, (ax1, ax2, ax3) = plt.subplots(1, 3, sharey=True)

        self.clear()
        #self.RK4_trajectory(v0 = 100, theta0 = 70)
        #ax1.plot(self.x2,self.y2,'red', label = "RK4")
        self.Euler_trajectory(v0 = 30, theta0 = 70)
```

```
        ax1.plot(self.x1,self.y1,'blue', label = "70 degree")
        #self.Exact_trajectory(v0 = 100, theta0 = 70)
        #ax1.plot(self.x3,self.y3,'k', label = "Accurate result")
        #ax1.set_title('theta_0=70 degree')
        ax1.set_ylabel('y (m)')
        ax1.set_xlabel('x (m)')
        ax1.legend()


        self.clear()
        #self.RK4_trajectory(v0 = 100, theta0 = 45)
        #ax2.plot(self.x2,self.y2,'red', label = "RK4")
        self.Euler_trajectory(v0 = 30, theta0 = 45)
        ax2.plot(self.x1,self.y1,'red', label = "45 degree")
        #self.Exact_trajectory(v0 = 100, theta0 = 45)
        #ax2.plot(self.x3,self.y3,'k', label = "Accurate result")
        #ax2.set_title('theta_0=45 degree')
        ax2.set_xlabel('x (m)')
        ax2.legend()

        self.clear()
        #self.RK4_trajectory(v0 = 100, theta0 = 30)
        #ax3.plot(self.x2,self.y2,'red', label = "RK4")
        self.Euler_trajectory(v0 = 30, theta0 = 30)
        ax3.plot(self.x1,self.y1,'k', label = "30 degree")
        #self.Exact_trajectory(v0 = 100, theta0 = 30)
        #ax3.plot(self.x3,self.y3,'k', label = "Accurate result")
        #ax3.set_title('theta_0=30 degree')
        ax3.set_xlabel('x (m)')
        ax3.legend()

        f.subplots_adjust(hspace=0.5)
        plt.show()

P1=Projectile()
P1.plot2D()
```