# hw2_exA

February 6, 2020

[2]:
```python
# -*- coding: utf-8 -*-
"""
Created on Tue Feb  4 20:30:05 2020

@author: akswa
"""

# python 3 version
import matplotlib.pyplot as plt
import numpy as np
from exD import interpf_mod # Import interpolation function from last assignment

# balle - Program to compute the trajectory of a baseball
#         using the Euler method.


def balle(theta = 45,tau = .01, plot_trajectory = False,
          airFlag = True, verbose = False, magnus = True, dimples = True):


    # Set default initial conditions
    y1 = 0.0
    speed = 70.0
    #theta = 45.0




    r1 = np.array([0.0, y1]);     # Initial vector position
    v1 = np.array([[speed*np.cos(theta*np.pi/180)], [speed*np.sin(theta*np.pi/
 ↪180)]]) # Initial velocity
    r = np.copy(r1)
    v = np.copy(v1)  # Set initial position and velocity, best to copy to avoid␣
 ↪overwrites

    #* Set physical parameters (mass, Cd, etc.)
```

```
    radius = 2.213e-2
    area = np.pi*radius**2  # Cross-sectional area of projectile (m^2)
    grav = 9.81    # Gravitational acceleration (m/s^2)
    mass = 0.04593   # Mass of projectile (kg)
    w = np.array([[0,0,1]]) # Relative spin of ball (x,y,z) Rotation is constrained to z-axis so that we only have a 2-D force
    magnus_eff = .2 # S0w/m i.e. the normed magnus "multiplier"
    S_0 = magnus_eff*mass/(np.linalg.norm(w))


    if not airFlag:
        rho = 0      # No air resistance
    else:
        rho = 1.2    # Density of air (kg/m^3)

    # If the ball has dimples, then the drag coef decrease inversely to velocity above 14 m/s
    if dimples:
        air_const = lambda v: -0.5*.5*rho*area/mass if v <= 14  else -0.5*(7/v)*rho*area/mass  # Air resistance constant with variable Cd
    else: # If no dimples, then the drag coeff is constant
        Cd = .5
        air_const = lambda v: -0.5*Cd*rho*area/mass



    #* Loop until ball hits ground or max steps completed
    maxstep = 100000   # Maximum number of steps
    for istep in range(0,maxstep):
        #* Record position (computed and theoretical) for plotting
        t = (istep)*tau     # Current time
        if(istep ==0):
            xplot = [r[0]]   # Record trajectory for plot
            yplot = [r[1]]
            xNoAir = [r[0]]
            yNoAir = [r[1]]
            time = [t]
            velocity = np.array(v)
        else:
            xplot.append(r[0,0])   # Record trajectory for plot
            yplot.append(r[0,1])
            xNoAir.append(r1[0] + v1[0]*t)   # Record trajectory for plot
            yNoAir.append(r1[1] + v1[1]*t - 0.5*grav*t**2)

        #* Calculate the acceleration of the ball
        accel = air_const(np.linalg.norm(v))*np.linalg.norm(v)*v   # Air resistance
```

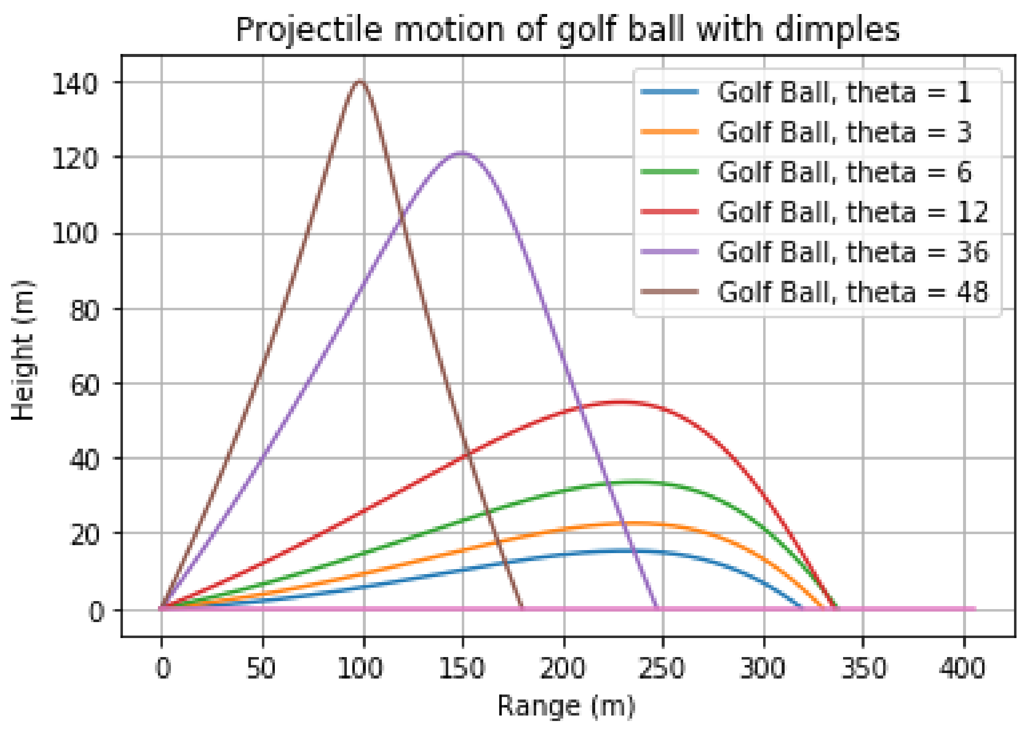
```
        accel[1] = accel[1]-grav      # Gravity
        if magnus: # add the effect of the magnus force
            magnus_accel = np.cross((S_0/mass)*w,v.T)
            accel[0] += magnus_accel[0][0]
            accel[1] += magnus_accel[0][1]

        # Use midpoint method by default
        v_new = v + tau*accel # Midpoint method
        r = r + (tau/2)*(v+v_new).T  #Midpoint method
        v = v_new #Midpoitn method

        time.append(t)
        velocity = np.concatenate((velocity,v),axis=1)
        #* If ball reaches ground (y<0), break out of the loop
        if( r[0,1] < 0 ):
            xplot = np.append(xplot,r[0,0]);   # Record trajectory for plot
            yplot = np.append(yplot,r[0,1]);
            time = np.array(time)
            break;                   # Break out of the for loop

    # Once the ball reaches the ground, interpolate the last 3 points to find␣
 ↪accurate endpoints
    x_end = interpf_mod(0,yplot[-3:],xplot[-3:]) # Note use interpf
    t_end = interpf_mod(0,yplot[-3:],time[-3:])

    if verbose:
        # Print maximum range and time of flight
        print("\nFor theta: %s" % theta)
        print('\tMaximum range is ',x_end,' meters');
        print('\tTime of flight is ',t_end,' seconds');

    return velocity,x_end,t_end,xplot,yplot

def plot_trajectories(xplot,yplot,theta,dimples):
        #if plot_trajectory:
        # Graph the trajectory of the baseball
        plt.figure(0)
        # Mark the location of the ground by a straight line
        xground = np.array([0, max(map(max, xplot_list))*1.2])
        yground = np.array([0, 0])
        legend_list = []

        # Plot the computed trajectories
        for i in range(len(xplot)):
            plt.plot(xplot[i],yplot[i],'-')
            legend_list.append('Golf Ball, theta = %s' % theta[i])
```

```python
        plt.xlabel('Range (m)')
        plt.ylabel('Height (m)')
        if dimples == 1:
            plt.title('Projectile motion of golf ball with dimples')
            plt.legend(legend_list)
        elif dimples == -1:
            plt.title('Projectile motion of smooth vs dimpled golf ball')
            plt.legend(['Dimpled ball','Smooth Ball']) # For comparison, we␣
 ↪manually set legend since we know the input

        elif dimples == 0:
            plt.title('Projectile motion of smooth golf ball')
            plt.legend(legend_list)


        plt.plot(xground,yground,'-')
        #axis equal; shg; # reset the aspect ratio, bring the plot to the front
        plt.grid(True)
        plt.show()
        return

def find_theta(verbose,dimple):
    range_list = []

    theta_range = np.linspace(1,40,100)
    for i in theta_range:
        v,r,t,_,_ = balle(theta = i,dimples=dimple)
        range_list.append(r)
    idx = np.where(range_list == max(range_list))[0][0]
    if verbose:
        print(max(range_list))
        print("Index:",idx,"\nOptimal Angle:", theta_range[idx], "\nRange at␣
 ↪Optimal Angle:", range_list[idx])

    return idx, theta_range[idx],range_list[idx]



if __name__ == '__main__':

    # Part a

    theta_list = [1,3,6,12,36,48]
    xplot_list = []
    yplot_list = []

    print("For a golf ball with dimples:")
```

```
    for i in theta_list:
        v,r,t,xplot,yplot = balle(theta = i,verbose = True,dimples=True)
        xplot_list.append(xplot)
        yplot_list.append(yplot)

    plot_trajectories(xplot_list, yplot_list, theta_list, dimples = 1)

    xplot_list = []
    yplot_list = []

    print("_____________________________\nFor a golf ball with a smooth surface:
↪")
    for i in theta_list:
        v,r,t,xplot,yplot = balle(theta = i,verbose = True,dimples=False)
        xplot_list.append(xplot)
        yplot_list.append(yplot)

    plot_trajectories(xplot_list, yplot_list, theta_list, dimples = 0)

    # Part b
    print("\n\n__________________\nPart b")
    print('Dimpled Ball:')
    i,theta_dimpled,distance_dimpled = find_theta(True,True)
    print("Smooth Ball:")
    i,theta_smooth,distance_smooth = find_theta(True,False)
    print("""
    # Part c
    # Same as part b ??
    # Not really sure what the quesiton is asking here, it really doesn't make␣
↪sense.
    # Seriously, go read it.
    # maybe plot the maximum ranges to compare??? Might as well.""")
    print("\n\n__________________\nPart c")
    print("See plot")

    v,r,t,xplot1,yplot1 = balle(theta = theta_dimpled,verbose =␣
↪False,dimples=True)
    v,r,t,xplot2,yplot2 = balle(theta = theta_smooth,verbose =␣
↪False,dimples=False)

    plot_trajectories([xplot1,xplot2], [yplot1,yplot2],␣
↪[theta_dimpled,theta_smooth], dimples=-1)
```

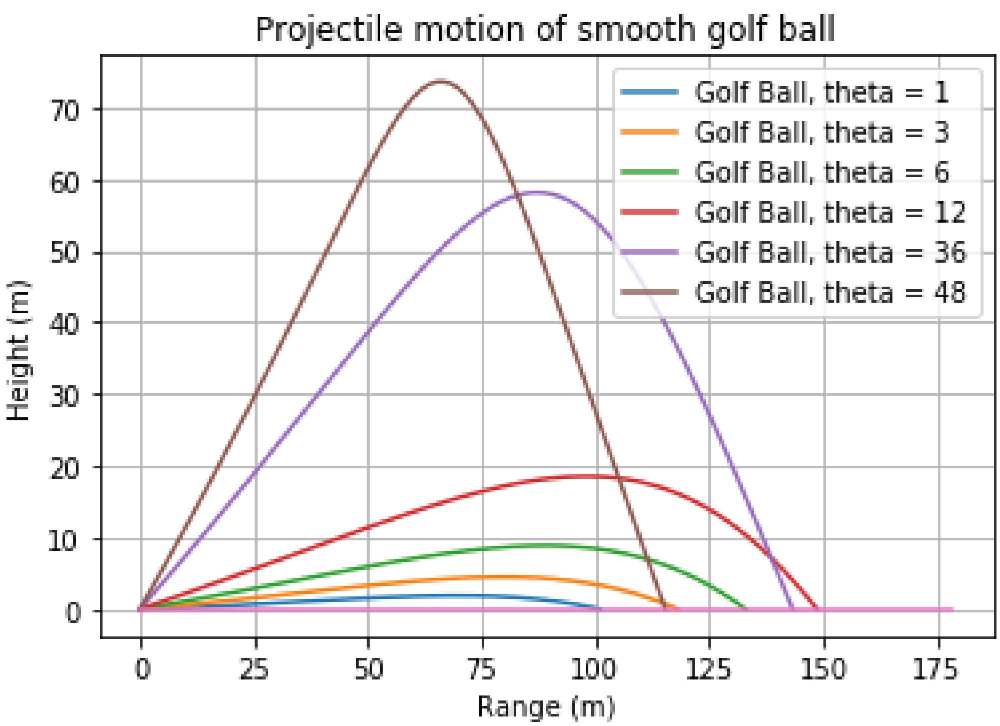
```
For a golf ball with dimples:

For theta: 1
        Maximum range is  319.7550647697863  meters
```

```
        Time of flight is  7.679327427889085  seconds

For theta: 3
        Maximum range is  330.32988820535104  meters
        Time of flight is  8.396363106398397  seconds

For theta: 6
        Maximum range is  337.5524638294372  meters
        Time of flight is  9.220562036896519  seconds

For theta: 12
        Maximum range is  335.90484394027516  meters
        Time of flight is  10.373963031719368  seconds

For theta: 36
        Maximum range is  247.29047235607422  meters
        Time of flight is  12.176738962681828  seconds

For theta: 48
        Maximum range is  180.04999493890847  meters
        Time of flight is  12.166075225541677  seconds
```

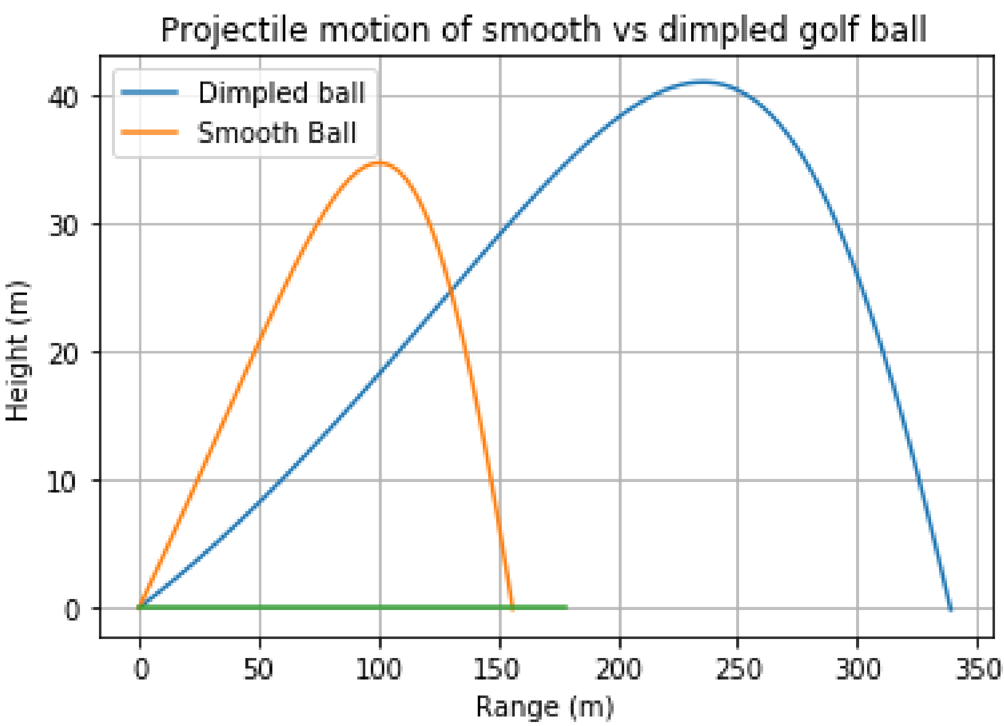



```
-----------------------------
For a golf ball with a smooth surface:
```

```
For theta: 1
        Maximum range is  100.88767311508516  meters
        Time of flight is  2.505014939833149  seconds

For theta: 3
        Maximum range is  117.99120965357253  meters
        Time of flight is  3.2721155107912843  seconds

For theta: 6
        Maximum range is  132.9679892557982  meters
        Time of flight is  4.110248008641759  seconds

For theta: 12
        Maximum range is  148.64636977634544  meters
        Time of flight is  5.338777137637475  seconds

For theta: 36
        Maximum range is  143.30109866645037  meters
        Time of flight is  8.099132635727633  seconds

For theta: 48
        Maximum range is  115.35854841574533  meters
        Time of flight is  8.687889791233069  seconds
```



```
-------------------
Part b
Dimpled Ball:
338.8315441949917
Index: 18
Optimal Angle: 8.090909090909
Range at Optimal Angle: 338.8315441949917
Smooth Ball:
156.01171553189633
Index: 52
Optimal Angle: 21.484848484848484
Range at Optimal Angle: 156.01171553189633

    # Part c
    # Same as part b ??
    # Not really sure what the quesiton is asking here, it really doesn't make
sense.
    # Seriously, go read it.
    # maybe plot the maximum ranges to compare??? Might as well.


-------------------
Part c
See plot
```



[ ]:

```
```