Asra Mehrolhassani [810800024]

▪ Problem 1

A) Compute the **range R** of the particle

- **With aerodynamic drag**

  The required background|derivation of the equations of motion:

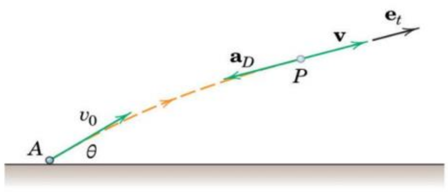


Figure Prob. 1

-y Direction | $a_{D_y} \downarrow \downarrow g$; $a_{Dy} = (-Kv^2)\left(\frac{v_y}{\sqrt{v_x^2+v_y^2}}\right)$

$\Rightarrow a_{Dy} = -Kv^2\left(\frac{v_y}{v}\right) = -Kvv_y$

Motion Equation : $a_y = \frac{dv_y}{dt} \Rightarrow -Kvv_y - g = \frac{dv_y}{dt}$

-x Direction | $\xleftarrow{a_{Drag}} \xrightarrow{v_x}$ $a_{xDrag} = \left(-Kv^2\left(\frac{v_x}{\sqrt{v_x^2+v_y^2}}\right)\right)$

$\Rightarrow a_{xDrag} = -Kvv_x$

Motion Equation : $a_x = -Kvv_x = \frac{dv_x}{dt}$

without Drag | $a_y = -g = \frac{dv_y}{dt} \Rightarrow -gt = v_y - v_0 \Rightarrow v_y = v_0 - gt \Rightarrow \frac{v_0}{g} = t$

$a_x = 0 \Rightarrow v_{x_{in}} t = X - X_0$

$\Rightarrow 2v_{0x}\left(\frac{v_{0y}}{g}\right) = X - X_0^{\,0}$

$\Rightarrow \text{Range} = \frac{2v_{0x}v_y}{g}$

Since both motion equations are dependent on each other, the mathematical solution is fairly complicated(if one exists:)), we solve the problem in both y and x direction by using **numerical methods (ode45 algorithm)**. Note that there's a downward (negative) gravitational acceleration which must be included in the calculations.

```
%Defining initial conditions matrix required for solving the differential equation via ode45
[t,m] = ode45(@f,[0 11], [0 vx_in 0  vy_in]);


%finding where y = 0, y[0] is the initial height which is zero, y[1]'s corresponding x is the range_Note that ode45 provides estimated values we spot the zero values by the error of 2, the last one is the desired value

h_index = find(abs(m(:,3) - 0)<2);
h_index(end);
Range_with_drag = m(h_index(end),1)
```

```
function dydt = f(t,m)
     v = sqrt(m(2)^2 + m(4)^2);
     g = -9.81;
     k = 4.0 * 10^(-3);
     dydt = [m(2);-1*m(2)*v*k; m(4); -1*m(4)*v*k + g];
end
```

Output:

```
Range_with_drag = 217.8889
```

- **Without aerodynamic drag**
  Required Background:
  Since the acceleration in y direction is equal to –g, by determining the instant at which y is zero(hits the ground) we can obtain the range by substituting the value in x-motion equation : x2 – x1 = V0x(t).

trajectory without air drag: solving the differential equation via ode function

```
%solving the differential equation ay = g = dy2/dt
syms y(t) t y_nodrag(t)
Dy = diff(y,t);
ode = diff(y,t,2) == G;
cond1 = y(0) == 0;
cond2 = Dy(0) == vy_in;
conds = [cond1 cond2];
y_nodrag(t) = simplify(dsolve(ode, conds))
```

Determining when y = 0(hits the ground), note that y_initial is zero and is not the desired value

```
eqn = y_nodrag(t) == 0;
t2 = eval(solve(eqn, t));
t2 = t2(2);
```

defining the x-motion equation and substituting the t value obtained above in the relation

```
% motion equation for x direction
syms x_nodrag(t) t
x_nodrag(t) = vx_in * t;
Range_noDrag = eval(x_nodrag(t2))
```

Output

```
Range_noDrag = 469.3674
```

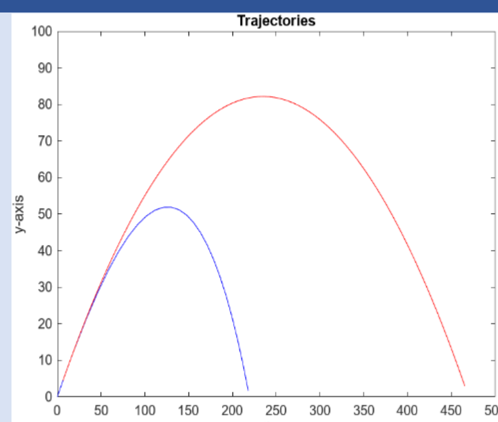
B) Plot the trajectories

```
%y and x ( with air drag):
y = m(:,3);
x = m(:,1);


%y and x (without air drag)
time = linspace(0,11,100);
x_nd = x_nodrag(time);
y_nd = y_nodrag(time);


%ploting
plot(x(y>0), y(y>0),'b', x_nd(y_nd>0), y_nd(y_nd>0), 'r');
axis([0 500 0 100]);
title ("Trajectory(with air drag)");
xlabel("time");
ylabel("y-axis");
```

Output:

Red: With aerodynamic drag

Blue: Without aerodynamic drag



C) The Game: Determine the best range provided that the ball avoids all the obstacles and hits the goal

By plotting trajectories for different angles, is it perceived that **beta = 20 satisfies the game's condition**. Since ode45 provides estimated values, the exact result cannot be processed. However it can be **visually justified**. The program is designed such that it predicts the outcome by plotting the trajectory by red (successful) or green (unsuccessful), although as mentioned before, it is **not accurate**. **Beta = 23** seems to provide the **best range**.

beta < 20 will never reach the height 20 and beta >= 24 won't is unrealistic since the ball is thrown after the first obstacle.

```
%b is the angle : beta
b=24;

%determinig the initial velocity in x and y direction
vxandy = V0(b, v_in);
vx0 =vxandy(1);
vy0 =vxandy(2);

%calling the below function
plot_trajectory(vx0, vy0);
```

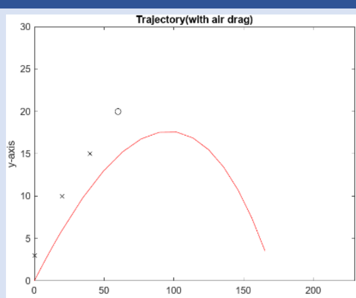
below comes the code of the two functions: plot_trajectory and game_rules. Notice how the plot_trajectory calls the game_rules function which returns **1 if the conditions are satisfied and returns 0 otherwise**. Then based on the on the provided output, plot_trajectory() determines the color of the figure. **(Emphasis on how the process is fairly inaccurate)**

```
unction dydt = f(t,m)
     v = sqrt(m(2)^2 + m(4)^2);
     g = -9.81;
     k = 4.0 * 10^(-3);
     dydt = [m(2);-1*m(2)*v*k; m(4); -1*m(4)*v*k + g];
end


function v = V0(b, vin)
    vx0 = vin * cos(b * pi / 180);
    vy0 = vin * sin(b * pi / 180);
    v = [vx0;vy0];
end


function hit = game_rules(y_60, y_40, y_20)
    %goal at x = 80     obstacle 3 at gx - 20   obstacle 2 at gx - 40   obstacle 1 at
gx - 60
    obs3 = 15;
    obs2 = 10;
    obs1 = 3;


    hit = 1;
    if (abs(y_60 - obs3) > 10)
        hit = 0;
    elseif (abs(y_40 - obs2) > 10)
        hit = 0;
    elseif (abs(y_20 - obs1) > 10)
        hit = 0;
    end
```

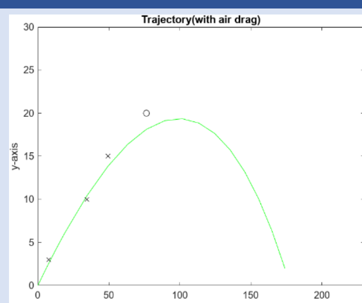
```
end


function plot_trajectory(vx0, vy0)
    [t,m] = ode45(@f,[0 11], [0 vx0 0  vy0]);

    %obs1
    h_goal = find(abs(m(:,3) - 20) < 2);
    if (size(h_goal) > 0)
        goalx = m(h_goal(1),1);
    else
        goalx = 60;
    end


    if(goalx > 61)
        h_obs3 = find(abs(20 - goalx + m(:,1)) < 10);
        h_obs2 = find(abs(40 - goalx + m(:,1)) < 10);
        h_obs1 = find(abs(60 - goalx + m(:,1)) < 10);


        obs1x = m(h_obs1(1),1);
        obs2x = m(h_obs2(1),1);
        obs3x = m(h_obs3(1),1);
        if (game_rules( m(h_obs3(1),3), m(h_obs2(1),3), m(h_obs1(1),3)) )
            color = 'g';
        else
            color = 'r';
        end
    else
            color = 'r';
            obs1x = goalx - 60;
            obs2x = goalx - 40;
            obs3x = goalx - 20;
    end
    %y and x ( with air drag):
    y = m(:,3);
    x = m(:,1);

    %ploting
    plot(x(y>0), y(y>0),color, obs1x, 3 , 'black -x',obs2x, 10 , 'black -x',obs3x, 15 ,
'black -x',goalx, 20 , 'black -o');
    axis([-20 230 0 30]);
    title ("Trajectory(with air drag)");
    xlabel("time");
    ylabel("y-axis");
end
```

```
function hit = game_rules(y_80, y_60, y_40, y_20)
    %goal at x = 80     obstacle 3 at gx - 20   obstacle 2 at gx - 40   obstacle 1 at
gx - 60
    goal = 20;
    obs3 = 15;
    obs2 = 10;
    obs1 = 3;

    hit = 1;
    if (abs(y_80 - goal) > 10)
        hit = 0;
    elseif (abs(y_60 - obs3) > 10)
        hit = 0;
    elseif (abs(y_40 - obs2) > 10)
        hit = 0;
    elseif (abs(y_20 - obs1) > 10)
        hit = 0;
    end
end
```

output:

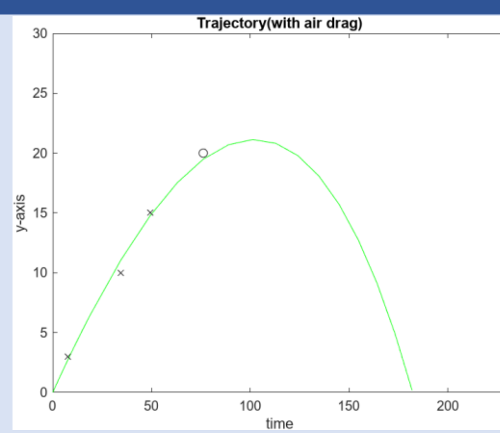


beta = 18 lost x

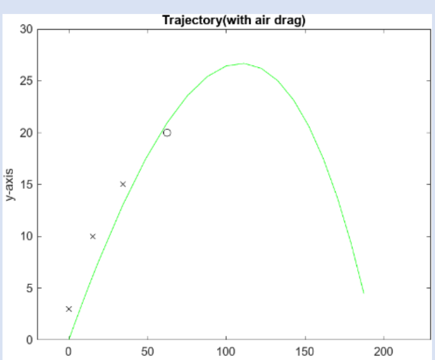


beta = 19 lost x

notice the error

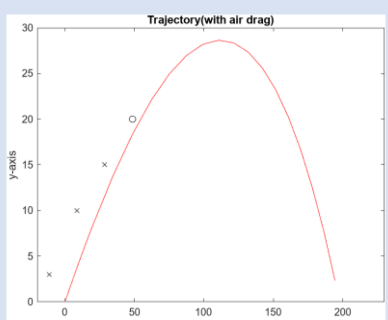


beta = 20 won ☺



beta = 23 best range

not exactly clear though



beta = 24 lost x

theta>= 24 there is no place for the first obstacle.