Shreyas Rane
Prof. Jiantao Kong
Intro to Scientific Computing
May 4, 2020

# Final Project- 2-D Projectile Motion (with air resistance)

## Problem:

The goal is to simulate a cannonball's projectile motion and how air drag can affect it. In my code I have plot two paths one without any air drag and another one with air drag.

### Literature:

The first obvious step was to think about projectile motion without air drag. Without accounting for the drag the only forces acting on the projectile were w=mg and the components of acceleration were simply ax=0 and ay=-g.
The drag force acts both in the opposite direction to the projectile and in the direction perpendicular to it. There were are going to consider +x-axis for horizontal and +y-axis for vertical components.
We get the first equation for drag force from Newton's Law's
$f = Dv^2$ where $v^2 = vx^2 + vy^2$
F being a vector the equation for x and y-axis would be
$f = -DVVx$ and $f = -DVVy$

The constant D can be given by:
$D = (\rho CA)/2$
Where $\rho$ is the density of air and C is the drag coefficient of that object dependant on its shape
Now if we want to calculate drag for both x and v components while taking into account the velocity it will be given as follows:

a) $-Dx = ((\rho CA)/M).V.Vx$ (negative sign indicates that drag acts in opposite direction.
b) $-Dy = -G - ((\rho CA)/M).V.Vy$ (-G means acceleration due to gravity acts in the direction opposite to y)

### Basic Equations:

The basic idea of the numerical calculations is as follows:

The acceleration components ax and ay are constantly changing as the velocity changes over Δt. During the course of Δt acceleration becomes a=Δv/Δt
So we can have the following equations for Δv for both its x and y components.

$vx + \Delta v = vx + ax\Delta t$ for x components
$vy + \Delta v = vy + ay\Delta t$ for y components

While this happens the co-ordinates are actually moving.
Therefore we can have the following equation for x-velocity.

$\Delta x = (vx + \Delta v/2)/\Delta t = vx\Delta t + 1/2\ ax\Delta t^2$
Therefore the coordinates for both x and y components are:
$x + \Delta x = x + vx + 1/2(ax\Delta t^{2)}$
$y + \Delta y = y + vy + 1/2(ay\Delta t^{2)}$

# Plan and Algorithm:

The implement this simulation I have to consider some base conditions for the cannonball. For the sake of reference I have used the cannonball measurements of the civil war era(more on this later).

## Definitions, variables, and calculations:

Initial Conditions:
m= mass of the object(kg)
All the civil war era cannonballs weighed 5.4kg with a radius of 4 inches

A=cross sectional area of the cannonball
Calulation:
Cr=4; %radius of the cannonball (in)
CAx= (pi*Cr^4)/4; %cross sectional area of x component of cannonball
CAy= (pi*Cr^4)/4; %cross sectional area of y component of cannonball
Ca=CAx+CAy; %x and y components added for total area
A= Ca*0.00064516;% conversion from in^2 to m^2

V= combination of Vx and Vy components of velocities. Comes from the equation $v^2 = vx^2 + vy^2$ (listed above)

G= acceleration due to gravity(m/s^2)
D= Drag coefficient of the cannonball
x= initial x position
y= initial y position
xf= final x position
yf=final y position

t= initial time
dt= Δt (I have set this at 0.5s, it can be changed)
Air_Density= air density at sea level.

## Algorithm:

2) Initialize all the variables listed above.
3) While y is greater than - Δt
    a) Increment t with Δt
    b) Increase i (iteration-initialized at 1)
    c) Calculate x position as it changes with time $\Delta x = vx\Delta t + 1/2\ ax\Delta t^2$
    d) Calculate drag for the x component given by $-Dx = ((\rho CA)/M).V.Vx$
    e) Calculate drag for the x component given by $-Dy = -G - ((\rho CA)/M).V.Vy$ (since G acts in opposite y-direction
    f) Increment Vx w.r.t $Dx$ and $\Delta t$
    g) Increment Vy w.r.t $Dy$ and $\Delta t$
    h) Calculate $x + \Delta x = x + vx + 1/2(ax\Delta t^{2)}$
    i) Calculate $y + \Delta y = y + vy + 1/2(ay\Delta t^{2)}$
    j) Allocate the calculations to Vx and Vy respectively

Time Complexity: The Algorithm runs in O(n) time.

# Input:

This is the part where the role of the cannon operator comes in. The cannon operator can choose the horizontal velocity and vertical velocity i.e Vx and Vy

This is given in the beginning of the code:
```
Vx = input('Awaiting your command Captain.Cannon is ready to fire!What would like the horizontal velocity to be,Captain?: ');
Vy = input('And Vertical?:');
```

At this point I have restricted the ceiling for velocities to 50 since I realized the path descent starts to straighten beyond this point. Feel free to comment on them if statement and increase the velocity.

# Output:

The output of the code is a 2d plot that traces the paths of the cannonball. Blue path is the one with air resistance and red is the one without air resistance.

Implementation:
```
plot(x,y,'b'), hold on;%blue-plots the Projectile Motion with air resistance
```

```
plot(xf,y,'r'), hold off;%red-plots the Projectile Motion without air resistance
xlabel('Horizontal Distance (m)');
ylabel('Vertical Distance (m)');
title('Cannonball paths with and without resistance');
 legend({'blue = with resistance','red = without resistance'},'Location','northeast')
```

## Code:

```
clear % this will help you clear the values from the previous run
 close

 % Section 1
 %Initial display, this acts as a cannon operator enabling you to decide
 %the horizontal and vertical velocity
 Vx = input('Awaiting your command Captain.Cannon is ready to fire!What would like the
horizontal velocity to be,Captain?: ');
 Vy = input('And Vertical?:');
 %since I chose measurements for the cannonball from the civil war era I have
 %restricted the velocity to 50. Feel free to stretch it until the plot
 %turns from a curve to edges
 if Vx > 50 || Vy>50
    disp('Sorry sir, our civil war era cannons are not equipped for the mentioned velocities.
Perhaps try less than 50, Captain?')
    Vx = input('Cannon ready to fire, what you like the horizontal velocity to be,Captain?: ');
    Vy = input('And Vertical?:');
 end

 %Section 2
 %All of the initial information for our calculations islisted here.
  M = 5.4;%mass of a cannonball (kg)
  Cr=4; %radius of the cannon ball (in)
  CAx= (pi*Cr^4)/4; %cross sectional area of x component of cannonball
  CAy= (pi*Cr^4)/4; %cross sectional area of y component of cannonball
  Ca=CAx+CAy; %x and y components added for total area
  A= Ca*0.00064516;% conversion from in^2 to m^2
  V = sqrt(Vx^2 + Vy^2);% Combines horizontal aand vertical velocities
  G = 9.8;% Acceleration due to Gravity(m/s^2)
  Drag_coeff = 0.47;   %Drag coefficient
  x = 0; %xi
  y = 0; %yi
  xf = 0; %declaring xf variable for x final
  yf=0;
  t = 0; %time
  delta_t = 0.05;%s set the intervals at which time will be evalutated
```

```
Air_Density = 1.225;%Assuming we are at sea level(kg/m^3)
rho=Air_Density;


%Section 3
%While loop calculates projectile motion iteratively
i = 1;
  while min(y)> -0.05
    t = t + delta_t;
    i = i + 1;
    xf(i) = xf(i-1)+ Vx.*delta_t;
    X_Drag = - ( rho*Drag_coeff*A/2 / M ) * V * Vx;
    Y_Drag = -G - ( rho*Drag_coeff*A/2 / M ) * V * Vy;
    Vx_delta = Vx + X_Drag * delta_t;
    Vy_delta = Vy + Y_Drag * delta_t;
    x(i) = x(i-1) + Vx_delta * delta_t + 1/2 * X_Drag * delta_t^2;
    y(i) = y(i-1) + Vy_delta * delta_t + 1/2 * Y_Drag * delta_t^2;
    Vx = Vx_delta;
    Vy = Vy_delta;
  end

%Section 4
%Plotting the projectile motion
plot(x,y,'b'), hold on;%blue-plots the Projectile Motion with air resistance
plot(xf,y,'r'), hold off;%red-plots the Projectile Motion without air resistance
xlabel('Horizontal Distance (m)');
ylabel('Vertical Distance (m)');
title('Projectile Motion 2-D');
legend({'blue = with resistance','red = without resistance'},'Location','northeast')
```

## Problems and Improvement:

There are various changes which I would like to make to the existing code but could not due to limited time:

1) Implement 3D version of the code. This would need another component Vz.
2) Allow the cannon operation to input just the distance before firing the cannon. This would be like a real-life cannon fire where the target is set based on its distance.
3) Use Runge-Kutta 4th order for projectile implementation in 1st order variant.

Works Cited:

Khan, Salman. “What is projectile motion?”. Khanacademy.org
https://www.khanacademy.org/science/physics/two-dimensional-motion/two-dimensional-projectile-mot/a/what-is-2d-projectile-motion

Elert, Glenn. “The Physics Factbook.” Mass of Cannonball.2009
https://hypertextbook.com/facts/2009/JenniferChung.shtml

Misc. “MATLAB Documentation.” Mathworks.
https://www.mathworks.com/help/matlab/