Bagby Assignment 3

Foundations of Problem

This assignment we were required to calculate the trajectory of a projectile, with and without air friction. We then made air friction depend on velocity^2, and finally we added the magnus force.

I used the Euler method to calculate the trajectory of the projectile over time. Using the Euler method you can calculate the velocity as v(i+1) = v(i) + a(i)*dt, and use that velocity to calculate the position doing x(i+1)= x(i) + v(i)*dt. I used a 'while' loop in MATLAB to loop until the projectiles y position became less than zero. I used a deltaTime variable to use in my position calculations each loop.

For trajectory, all you must do is keep track of the projectiles position, changing it by Velocity*deltaTime each loop. If there is an acceleration involved, you can use the acceleration to determine the velocity by adding to the velocity acceleration*deltaTime each loop. With trajectory alone, gravity is the only acceleration considered.

Adding air resistance that depends on velocity, I used Ai = Vi(i-1) *(-b/m) with b being a constant. This calculates the air resistance depending on its velocity from the last calculation.

For air resistance depending on velocity^2, I did v(i+1) = a(i)*deltaTime + v(i).

Finally, for magnus force, you must do MagnusCoefficient * (Omega X Velocity) where Omega is an angular velocity vector and Velocity is the velocity vector. Using the force you can calculate the acceleration by simply dividing by mass.

Text from Program

Constants

To calculate the trajectory of a projectile in the x,y, and z axis you must first decide an initial velocity and initial projectile angle. I chose the X axis as the axis headed away from home plate, the Y axis as the sky, and the Z axis as the sideways axis. We also must choose constants for the mass, air friction coefficient, magnus force coefficient, and angular velocity or omega.

```
%constants
gravity = -9.8;
velocity = 100;
angle = deg2rad(45);
m = 1; %mass
b = 0.1; %air friction
userAirFriction = 1;
magnusCoefficient = 1e-2;
omega = [0,5,0];
```

Since my initial velocity is a magnitude I then separated it into its 3 dimensions using the initial launch angle. This Vx, Vy, and Vz will be used each iteration of our loop to keep track of the projectiles current velocity.

```
%velocity
Vx = velocity * cos(angle);
Vy = velocity * sin(angle);
Vz = 0;
```

I then initialized 3 variables to keep track of our projectiles each loop as well, and arrays to keep track of all the calculated points that we can use for plotting later.

```
%var to hold position each loop
x=0;
y=0;
z=0;
%array to keep track of position
pointsX = [];
pointsY = [];
pointsZ = [];
```

The Loop

This while loop will continue until the position is less then or equal to 0

```
while y >= 0
```

Each iteration we update the projectiles Y velocity by (Gravity * deltaTime)

```
    %add gravity
    Vy = Vy + (gravity * step);
```

I then calculate the air resistance using the equation mentioned above.

Here is air resistance dependent on just velocity

```
    %calculate air resistance
    Ax = ((-b/m)*Vx);
    Ay = ((-b/m)*Vy);
    Az = ((-b/m)*Vz);
```

And, here is the equation to calculate air resistance dependent on velocity^2, where velocity is the magnitude of the velocity vector from the last iteration

```
    %calculate air resistance
    Ax = ((-b/m)*Vx * velocity);
    Ay = ((-b/m)*Vy * velocity);
```

```
    Az = ((-b/m)*Vz * velocity);
```

Next you must add the air resistance velocity to the projectiles current velocity.

Note: ‘useAirFriction’ was a variable I have in the program which can be set to 0 or 1 and allows air friction to easily be disabled

```
    %add air resistance
    Vy = Vy + ((Ay/m) * step * userAirFriction);
    Vx = Vx + ((Ax/m) * step * userAirFriction);
    Vz = Vz + ((Az/m) * step * userAirFriction);
```

To calculate the magnus force we must do the cross product of omega and the velocity vector, as discussed above. Then the result is multiplied by the magnus coefficient.

```
    vVector = [Vx,Vy,Vz];
    crossP = cross(omega,vVector);
    magnus = magnusCoefficient * crossP;
```

To turn this into velocity we divide it by the mass and multiply it by deltaTime.

```
%calculate magnus velocity. Acceleration * Dt
    MVx = (magnus(1) * step)/m;
    MVy = (magnus(2) * step)/m;
    MVz = (magnus(3) * step)/m;
```

Finally we can add this velocity to our current velocity

```
    %add magnus velocity
    Vz = (Vz + MVz);
    Vy = Vy + MVy;
    Vx = Vx + MVx;
```

Finally, after we have calculated our current velocity, we can use this velocity to update our position using Position(i) = Position(i-1) * deltaTime

```
    %calculate new position
    x = x + Vx * step;
    y = y + Vy * step;
    z = z + Vz * step;
    pointsX(index) = x;
    pointsY(index) = y;
    pointsZ(index) = z;
```

And to get ready for the next loop we can add deltaTime to total time, increment the index, and calculate the magnitude of our current velocity.

```
    t = t + step;

    index = index + 1;

    %calculate current velocity magnitude
    velocity = sqrt((Vx^2) + (Vy^2) * (Vz^2));
```

Plotting

Plotting was simply done by making two figures, one for the Y vs X position and one for the Z vs X position. I let the Z vs X position plot calculate its own bounds.

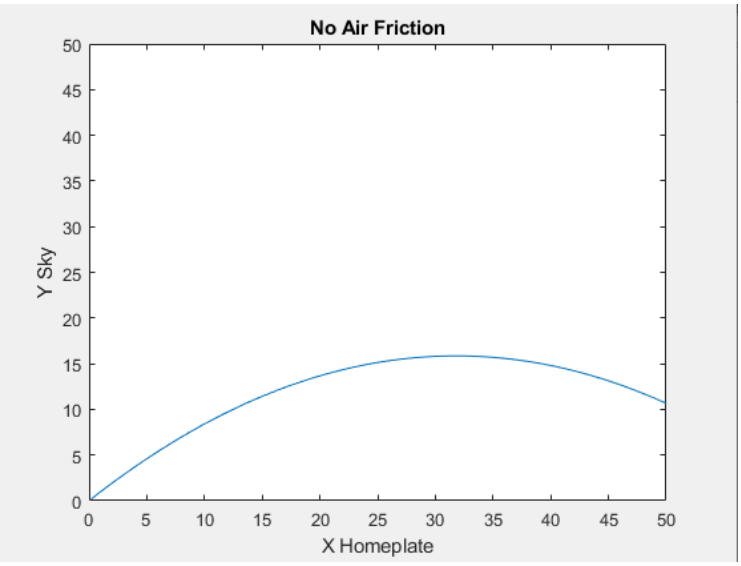
```
%PLOT for x and y
plot(pointsX,pointsY);
xlabel('X Homeplate')
ylabel('Y Sky')
title("Baseball");
axis([0, 50, 0, 50]);

%PLOT for x and z
figure(2);
plot(pointsX,pointsZ);
xlabel('X Homeplate')
ylabel('Z Sideways')
title("Baseball Magnus");
```

Outputs

No air friction

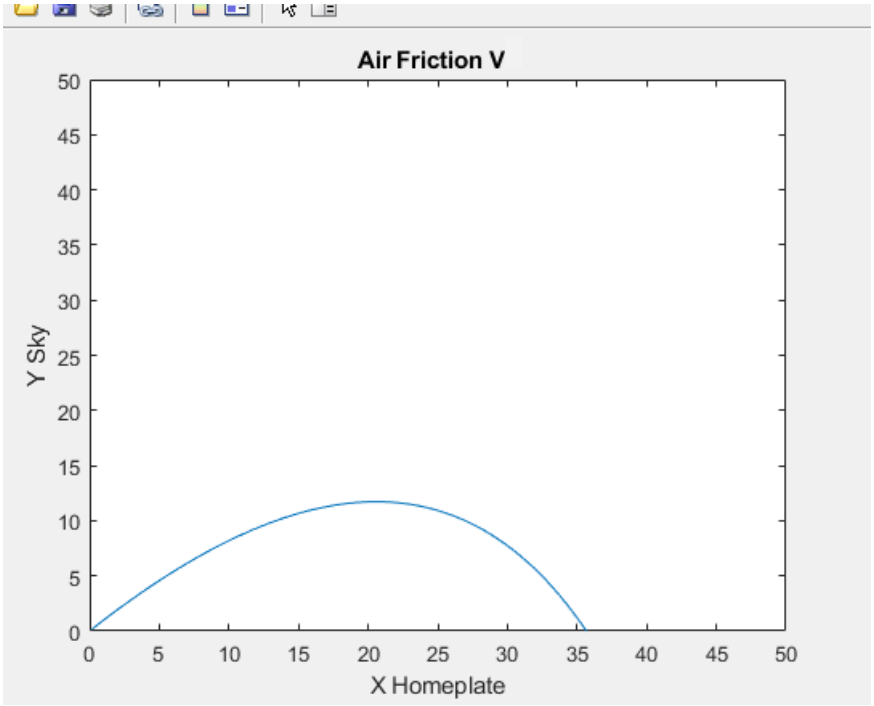
Initial velocity 25 with an angle of 45 degrees.



Air friction dependent on V

Velocity 25

Angle 45 degrees

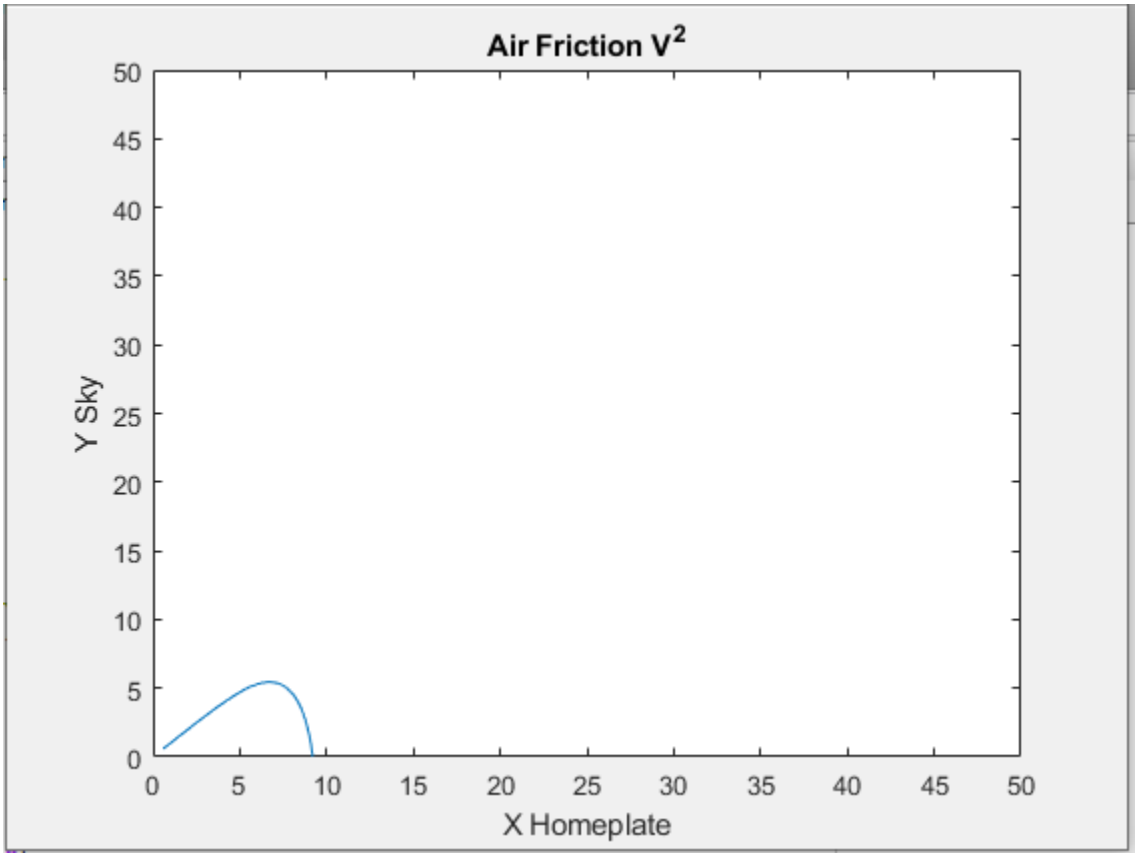


Air friction made the distance shorter with the projectile curving into the ground.

Air friction dependent on V^2

Velocity 150

Angle 45 degrees



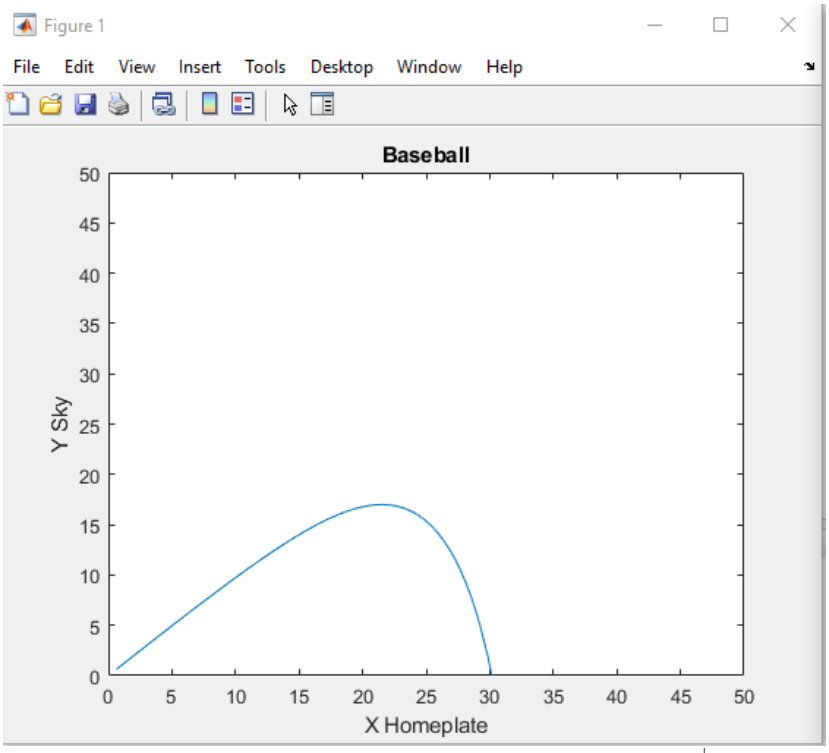
As you can see, V^2 air friction really cut down the projectiles distance with a harder curve into the ground.

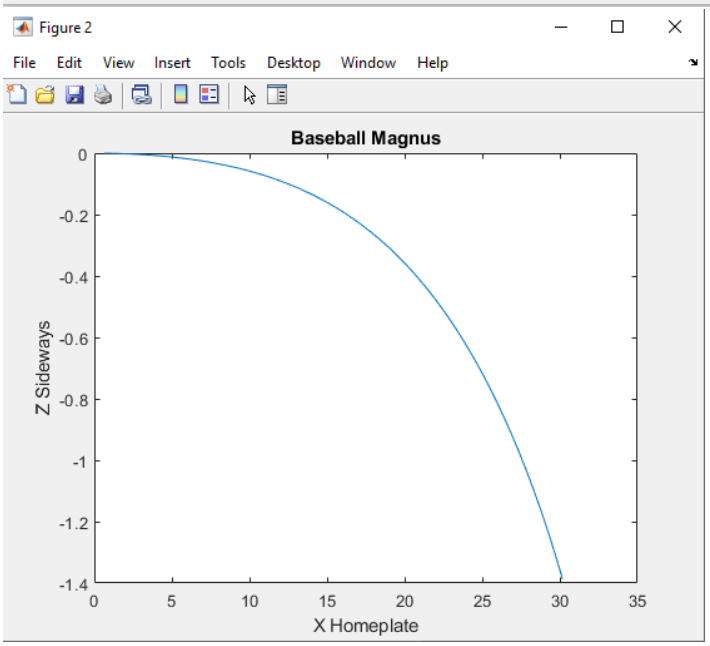
Air friction dependent on V^2 with Magnus force

Velocity 150

Angle 45 degrees

Decreased air friction coefficient so trajectory could be longer





As you can see the magnus force caused the baseball to curve in the z direction.