Table of Contents


```
%ASEN 3128 Lab 1 Simulate Equations of Motion EOM
%Cameron Mitchell, Tyler Gaston, Quinn Lewis, Eric Tate
```

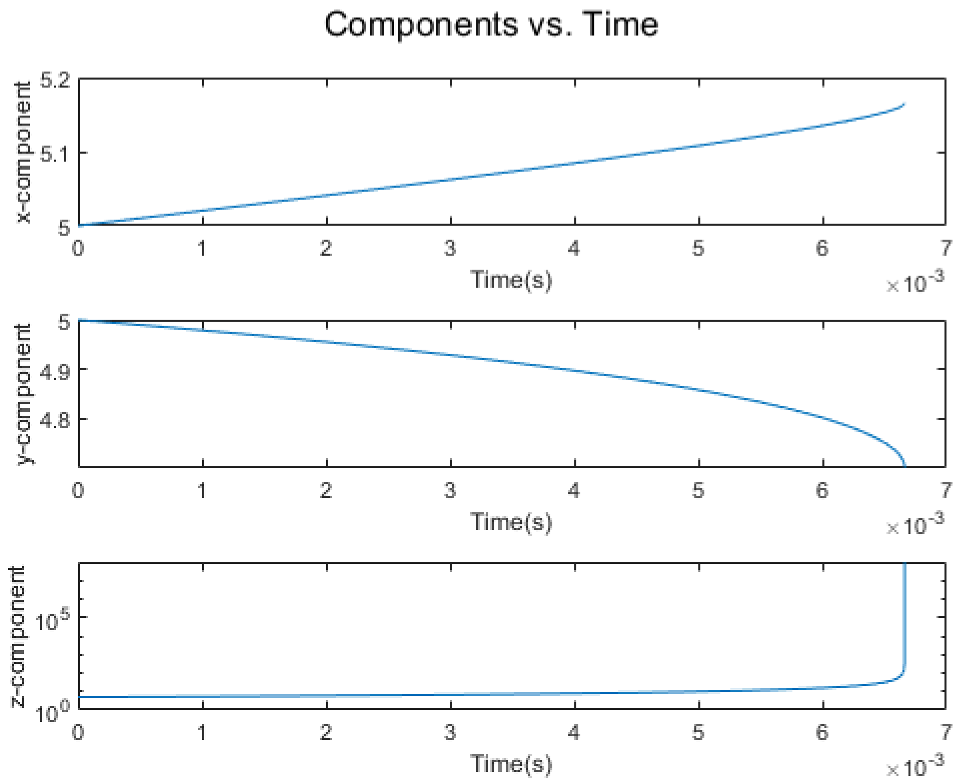
Housekeeping:

```
clear all; close all; clc;
```

Problem 1

Simulate the following set of equations with non-zero initial conditions and duration

```
x0_1=[5;5;5]; %initial conditions for x, y, and z
tSpan1=[0 1]; %time span [s]

[Tout1, state1] = ode45(@eom, tSpan1, x0_1);  %ODE45 function call

%Below is creating the subplot of the x y and z graphs with respect to
 time
figure(1)
subplot(3,1,1)
    plot(Tout1, state1(:,1))
    xlabel('Time(s)')
    ylabel('x-component')
subplot(3,1,2)
    plot(Tout1, state1(:,2))
    xlabel('Time(s)')
    ylabel('y-component')
subplot(3,1,3)
    semilogy(Tout1, state1(:,3))
    xlabel('Time(s)')
    ylabel('z-component')
sgtitle('Components vs. Time')      %title of group of subplots
```



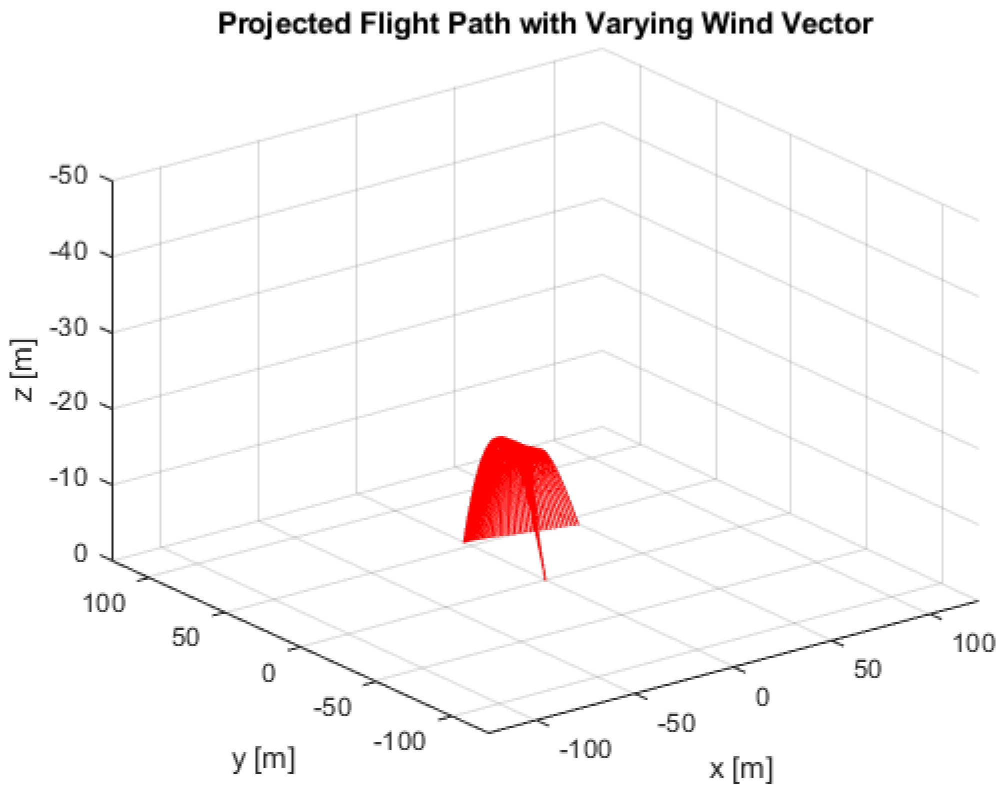
Problem 2

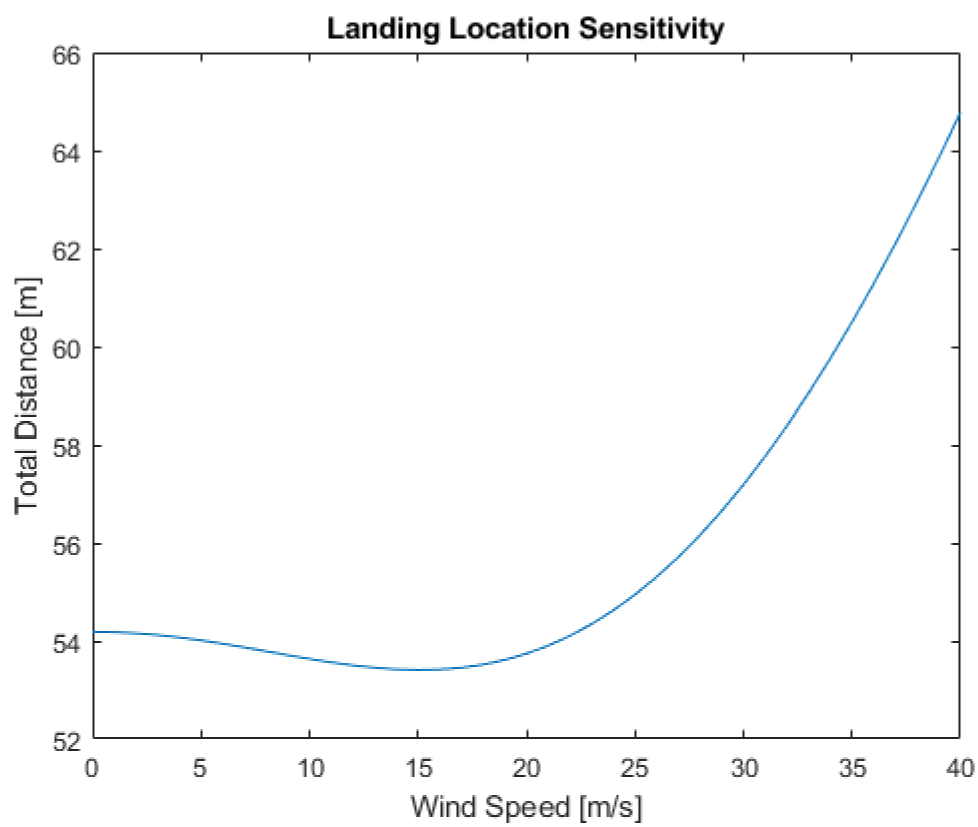
Construct simulation of the translational dynamics of a ball through air, where the forces on the body are not a function of the body attitude, but include drag(acts opposite inertial velocity vecotr) and gravity.

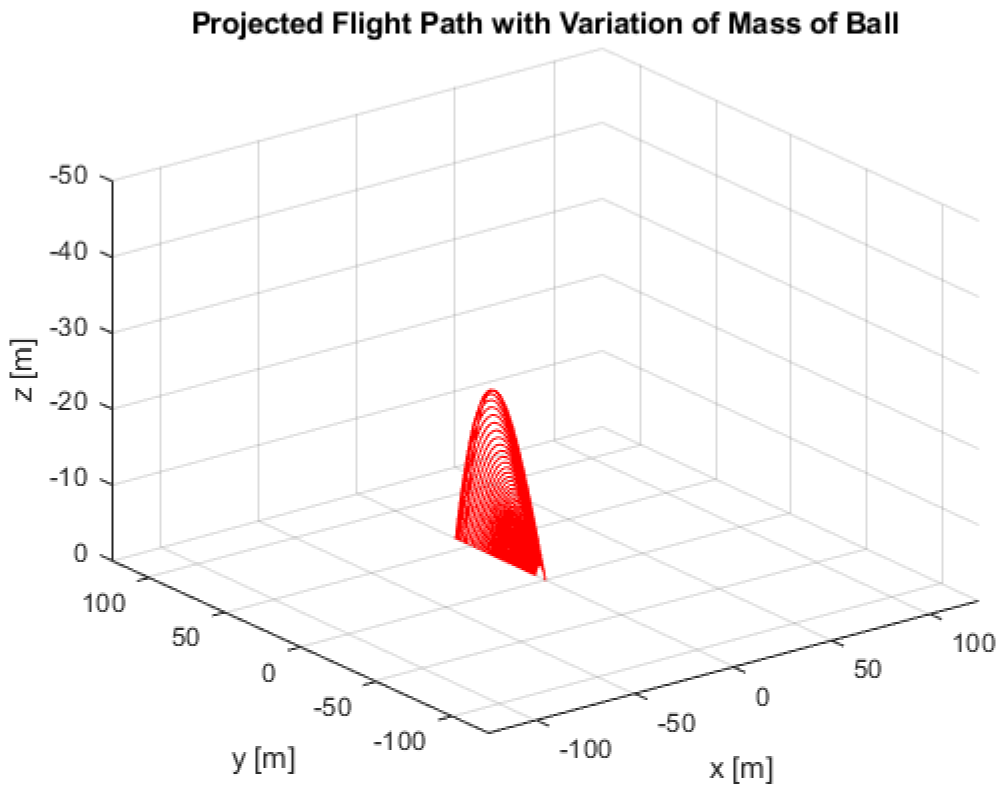
```
tSpan2 = [0 5];         %time span [s]
m=.03;                 %mass of ball in [kg]
d =.03;                 %diameter of ball in [m]
Cd=0.6;
rho = 1.225;              %typical air density in Boulder, CO [kg/m^3]
A = pi*(.5*d)^2;         %cross sectional area of ball [m^2]
g = 9.8;                 %acceleration of gravity on Earth [m/s^2]
W = [0;0;0];            %Wind velocity in intertial coordinates [m/s]
p_EE = [0;0;0];          %Initial intertial position in inertial
 coordinates vector of golf ball [x;y;z] in [m]
v_EE = [0;20;-20];        %Initial intertial velocity vector [u;v;w] in
 [m/s]

S0=[p_EE, v_EE];          %Initial state vector of inertial velocy and
 position in inertial coordinates [x,u; y,v; z,w]

L=0;                     %Wind index
maxWindNorth=40;
while L <= maxWindNorth
    W(1) = L;            %set wind in north direction [m/s]
```

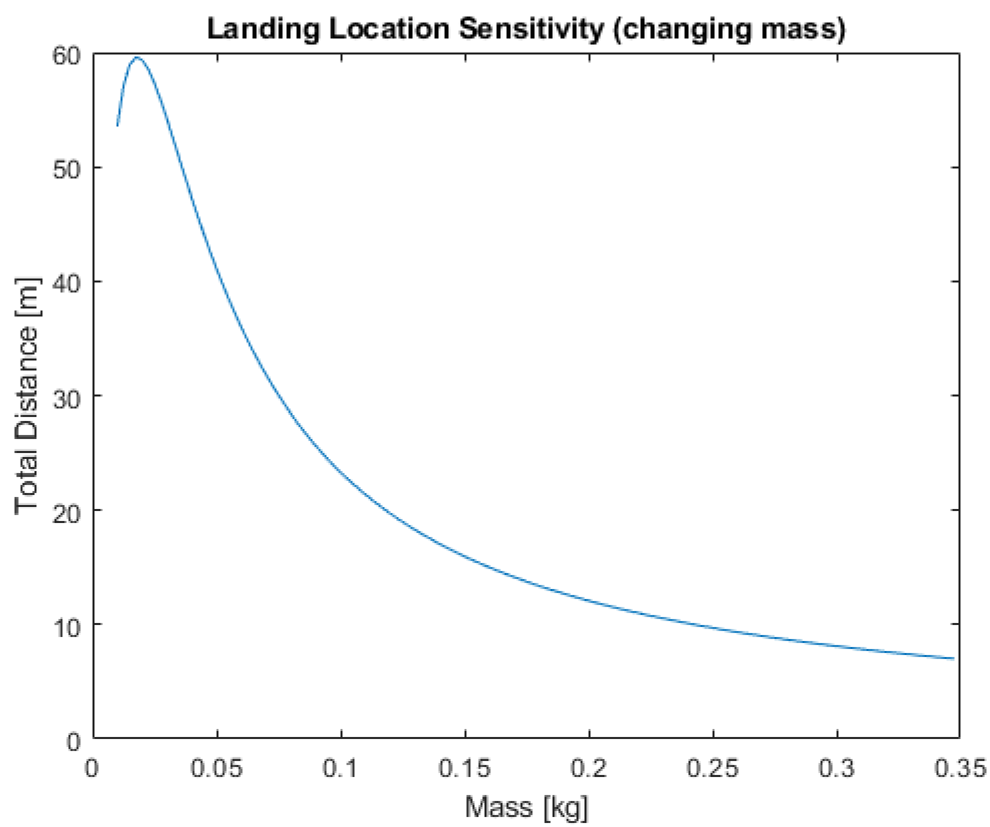
```
%ODE prop
    tol = 1e-3;
    opts = odeset('RelTol', tol, 'Events', @hitsGround);

    [Tout, state] = ode45(@(t, S) objectEOM(t, S, g, rho, m, Cd, A,
 W), tSpan2, S0, opts);
    i=L+1;
    windVariance(i) = L;
    totalDistance(i) = sqrt(state(end, 1)^2 + state(end, 2)^2);
  %total distance from start point[m]
%plot the trajectory
    figure(2)
    plot3(state(:,1), state(:, 2),state(:,3),'r');
        hold on
        set(gca,'Zdir','reverse')
        grid on
        xlim([-125 125])
        ylim([-125 125])
        zlim([-50 0])
        title('Projected Flight Path with Varying Wind Vector')
        xlabel('x [m]')
        ylabel('y [m]')
        zlabel('z [m]')

    L=L+1;   %index wind variance
end
hold off

figure(3)
plot(windVariance, totalDistance)%plot landing location delfection vs
 wind speed
    hold on
    title('Landing Location Sensitivity')
    xlabel('Wind Speed [m/s]')
    ylabel('Total Distance [m]')

 %Determine the distance with varying mass
 %Determine Kinetic Energy
 W = [0;0;0];                               %reset wind
 KE = .5 * m * norm(v_EE)^2;                %Determine kinetic energy
 possible from conservation of energy equation
 L=0.01;                                    %initial mass for changing
 mass trial
 i=0;
 maxMass = .35;                               %max mass for trial in [kg]

 %The below while loop runs until the set maximum mass is reached
 %this maximum mass value is entered by the user to depict how many
 many
 %different trials will be run
 while L <= maxMass
    m = L;                                  %set mass of ball [kg]
```

```
    v_EE = [0; sqrt(KE/m); -sqrt(KE/m)]; %This uses the assumption
 that the velocity in y and z direction are equal like in the given
 example
    S0=[p_EE, v_EE];

%ODE prop for varying mass
    tol = 1e-3;
    opts = odeset('RelTol', tol, 'Events', @hitsGround);

    [ToutM, stateM] = ode45(@(t, S) objectEOM(t, S, g, rho, m, Cd, A,
 W), tSpan2, S0, opts);
    i=i+1;
    massVariance(i) = L;
  %mass variance vector [kg]
    totalDistanceM(i) = sqrt(stateM(end, 1)^2 + stateM(end, 2)^2);
  %total distance from start point[m]
%plot the trajectory
    figure(4)
    plot3(stateM(:,1), stateM(:, 2),stateM(:,3),'r');
        hold on
        set(gca,'Zdir','reverse')
        grid on
        xlim([-125 125])
        ylim([-125 125])
        zlim([-50 0])
        title('Projected Flight Path with Variation of Mass of Ball')
        xlabel('x [m]')
        ylabel('y [m]')
        zlabel('z [m]')

    L=L+.0025;   %index mass variance step size [kg]
end
hold off

figure(5)
plot(massVariance, totalDistanceM)%plot landing location deflection vs
 wind speed
    hold on
    title('Landing Location Sensitivity (changing mass)')
    xlabel('Mass [kg]')
    ylabel('Total Distance [m]')
```


Projected Flight Path with Varying Wind Vector
-50
-40
-30
-20
-10
0
z [m]
100
50
0
-50
-100
y [m]
-100
-50
0
50
100
x [m]


Landing Location Sensitivity
66
64
62
60
58
56
54
52
Total Distance [m]
0
5
10
15
20
25
30
35
40
Wind Speed [m/s]


Projected Flight Path with Variation of Mass of Ball
-50
-40
-30
-20
-10
0
z [m]
100
50
0
-50
-100
y [m]
-100
-50
0
50
100
x [m]


Landing Location Sensitivity (changing mass)
60
50
40
30
20
10
0
Total Distance [m]
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
Mass [kg]

Functions

```
%Problem 1:
function stateDot = eom(~, state)
%Inputs:
%t = time [s]
%state = Vector containing x,y,z values [x;y;z]
%Outputs:
%stateDot =[xPrime, yPrime, zPrime]
%xPrime = x+2*y+z;
%yPrime = x-5*z;
%zPrime = x*y-y^2+3*z^3;
%Methodology: Simulate the given set of equations (xPrime, yPrime,
 zPrime)
%with non-zero initial conditions and duration by using numerical
%integration through ode45. This function simply takes in a state
 vector
%that contains the initial conditions for the function and then using
 those
%initial conditions the ode45 function numerically integrates over the
%inputted time span

x=state(1);
y=state(2);
z=state(3);
xPrime = x+2*y+z;
yPrime = x-5*z;
zPrime = x*y-y^2+3*z^3;
stateDot = [xPrime; yPrime; zPrime];

end

%Problem 2:
function xDot = objectEOM(~,S,g,rho,m,Cd,A, W)
%Inputs:
%t= time [s]
%x= state vector of inertial velocy and position in inertial
 coordinates [x,u; y,v; z,w]
%rho= density [kg/m^3]
%Cd= coefficient of drag
%A = cross sectional area of golf ball [m^2]
%mass = mass of golf ball [g]
%wind = wind velocity vector [wind north;wind east;wind down]
%Outputs
%xDot = change in x vector..

%Methodology:
%The function takes in the initial conditions of the ball which are
 the
%x,y, and z position and velocity and also takes in set values for
 area,
%gravity, density, mass, coefficient of drag, and wind vector. Then
 the
```

```
%force acting on the ball is summed to tell us the total force acting
 on
%the ball. We can then divide out the mass and get acceleration which
 ode45
%can integrate to give us velocity in return. To get position we can
 simply
%integrate the velocity of the ball which is only effected by the wind
%acting on the ball.

    VE = [S(4); S(5); S(6)];             % [m/s]
    V =  VE-W;                           % [m/s]

    dVE = ((-0.5*rho*norm(V)*A*Cd)/m)*V; % acceleration      drag*(v/|
v|)
    dVE(3) = dVE(3)+g;                   % acceleration in z direction
 always subtract gravity

    xDot = [VE;dVE];                       % [x0 y0 z0 dx0 dy0 dz0]


end

%This function is used to tell us when the ball hits the ground and
 can
%then be used as an acceptable function time span for ode45.
function[v, i, d] = hitsGround(~, S)
    v = S(3) - 0;
    i = 1;
    d= 0;
end
```

```
Warning: Failure at t=6.663734e-03.  Unable to meet integration
 tolerances
without reducing the step size below the smallest value allowed
 (1.387779e-17)
at time t.
```

Published with MATLAB® R2020a