ASEN 3128 Lab 01

Table of Contents

Group Members: -Caleb Bristol -Tess Brodsky -Kyle Bowen -Quihan Cai

Date: 1/19/22

Housekeeping

```
clc
clear
close all;
```

Problem 1

Code taken from MATLAB autograder

Because the purpose of this problem is to utilize ode45, this should be sufficient "proof of concept" considering it passed the autograder

```
tspan_1 = [0 3];
x_0 = [0.1;0.1;0.1];

opts = odeset('RelTol',1e-9,'AbsTol',1e-9);
[t_1,x] = ode45(@(t,x) odefunc(t,x),tspan_1, x_0 ,opts);
```

Plotting

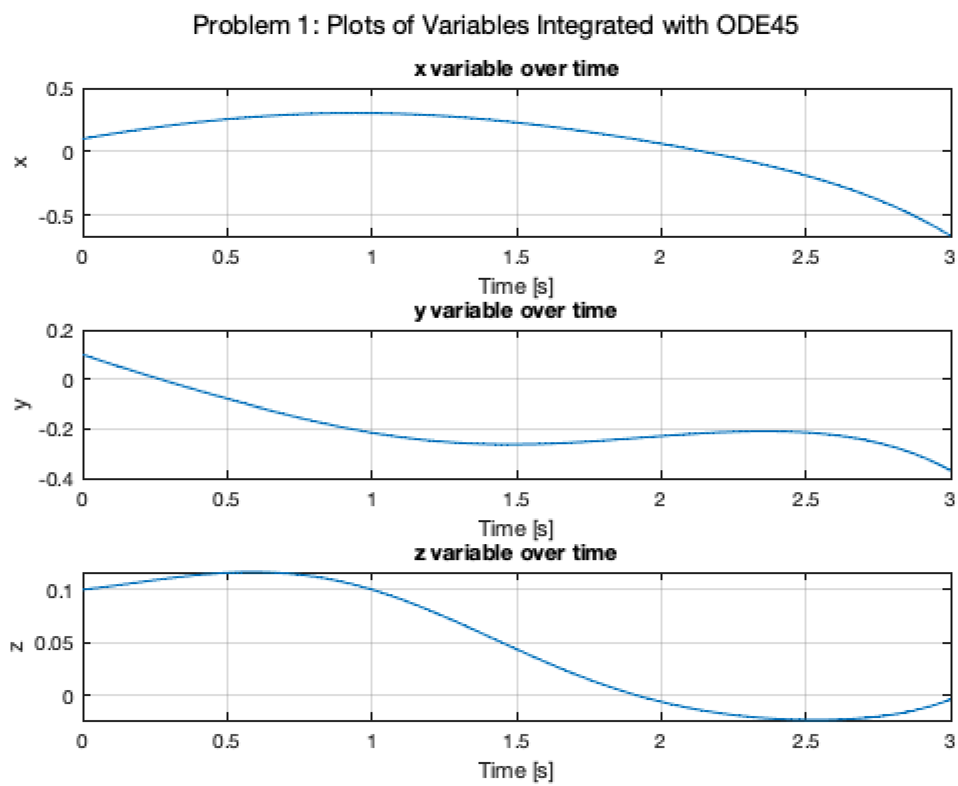
```
figure()
hold on
sgtitle('Problem 1: Plots of Variables Integrated with ODE45')
subplot(3,1,1)
plot(t_1,x(:,1))
xlabel('Time [s]')
ylabel('x')
grid on
```

```
title('x variable over time')

subplot(3,1,2)
plot(t_1,x(:,2))
xlabel('Time [s]')
ylabel('y')
grid on
title('y variable over time')

subplot(3,1,3)
plot(t_1,x(:,3))
xlabel('Time [s]')
ylabel('z')
grid on
title('z variable over time')

hold off
```



Problem 2

This problem is the more involved dynamic modeling of a golfball

The design constants are either provided in the lab document, or assumed or be std atmosphere at sea level

```
% Design Constants
tspan = [0 10]; %[s]: this is arbitrarily picked
```

```
rho = 1.225; %[kg/m^3]
Cd = 0.6; %[unitless]
A = pi * (0.03/2)^2; %[m^2]
m = 0.03; %[kg]
g = 9.8; %[m/s^2]
```

Part a)

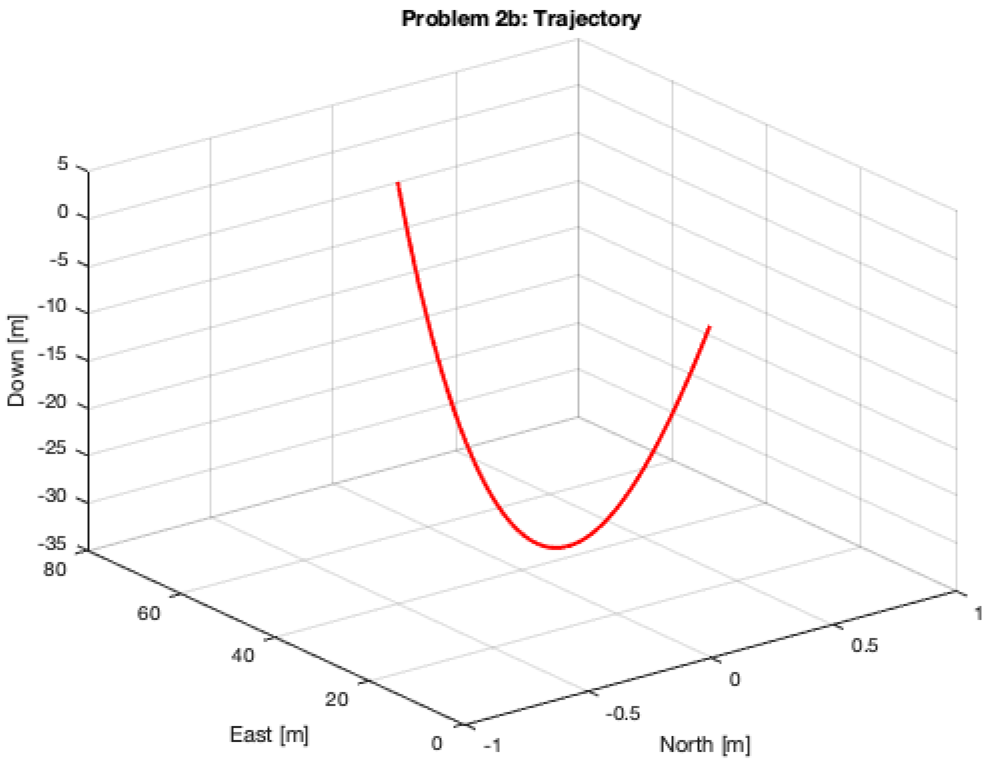
See functions section at bottom

Part b)

Simulate the inertial frame and verify that results make sense

```
    % Initial Conditions
    x_i = [0;0;0;0;20;-30];
    wind_b = [0;0;0];
    [t,x_b] = ode45(@(t,x_b) objectEOM(t,x_b,rho,Cd,A,m,g,wind_b),
 tspan, x_i, opts);

    % Plot part b
    figure()
    plot3(x_b(:,1),x_b(:,2),x_b(:,3),'r','LineWidth',2); hold on
    xlabel('North [m]')
    ylabel('East [m]')
    zlabel('Down [m]')
    title('Problem 2b: Trajectory')
    grid on
    hold off
```



Part c)

Measure Sensitivity in response to northern wind speed

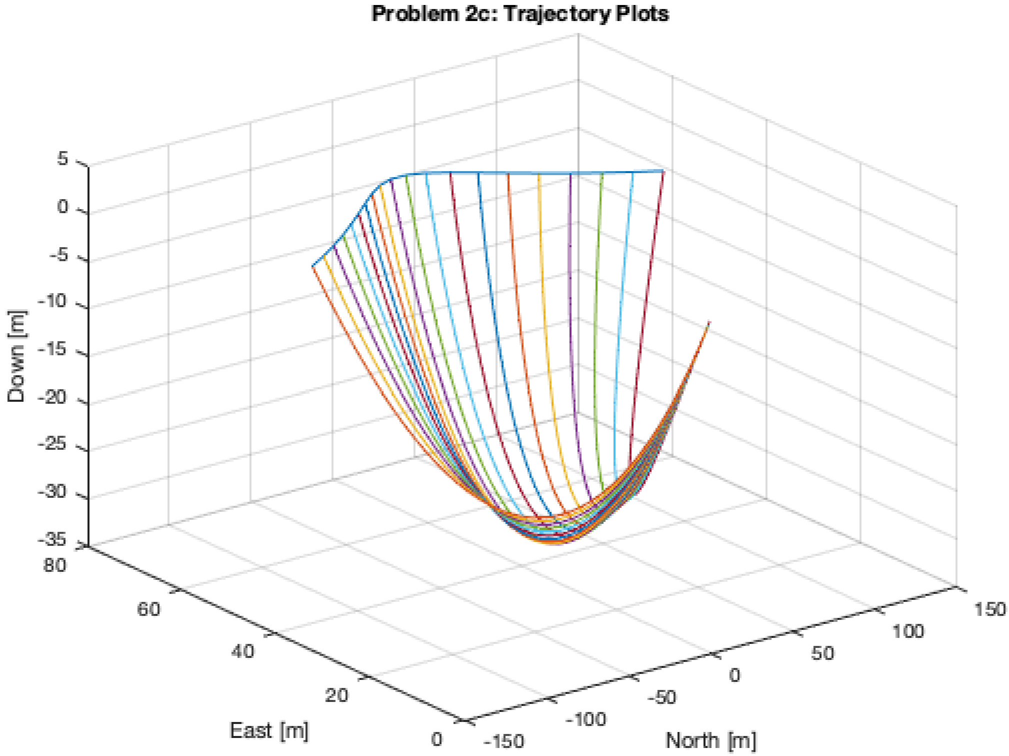
```
% Define wind vectors
%
% All windspeed in north direction, from -50 to 50 m/s
wind_c = zeros(2,20);
wind_c = [linspace(-50,50,20);wind_c];

% Preallocate Landing Location and state vectors over time in cell
landinglocation = zeros(3,20);
x_c_cell = cell(1,20);

% Run ODE45 for all possible wind vectors, save data
%
% The landing location is saved as the final state
% The trajectories are saved in x_c_cell for plotting
for i = 1:20
    [t,x_c] = ode45(@(t,x_c)
objectEOM(t,x_c,rho,Cd,A,m,g,wind_c(:,i)), tspan, x_i, opts);
    landinglocation(:,i) = x_c(end,1:3);
    x_c_cell(i) = {x_c};
end

% Plot Trajectories of Golfballs with Different Windspeeds
```

```
    %
    % Windspeed varies linearly from -50 to 50 m/s
    figure()
    plot3(landinglocation(1,:),landinglocation(2,:),zeros(1,20));
hold on
    for i = 1:20
        plot3(x_c_cell{1,i}(:,1),x_c_cell{1,i}(:,2),x_c_cell{1,i}
(:,3))
    end
    xlabel('North [m]')
    ylabel('East [m]')
    zlabel('Down [m]')
    title('Problem 2c: Trajectory Plots')
    grid on
    hold off
```

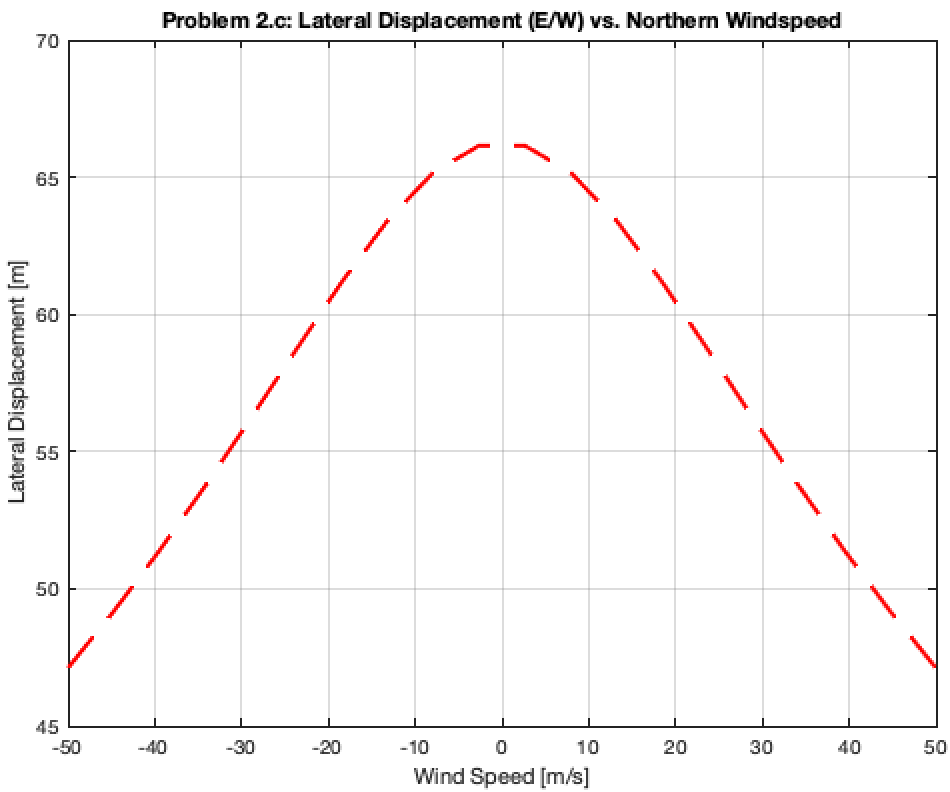


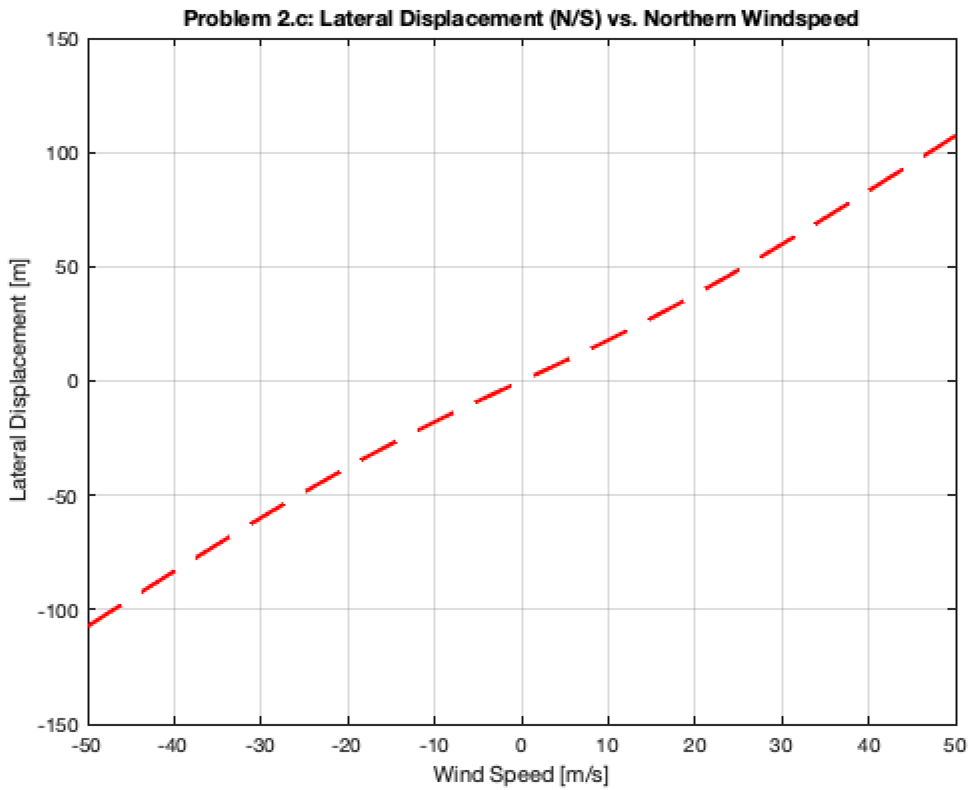
Lateral Displacement vs. Windspeed

```
        % Plot Lateral Displacement (East/West) against Windspeed
        figure()
        plot(wind_c(1,:),landinglocation(2,:),'--r','LineWidth',2);
hold on
        xlabel('Wind Speed [m/s]')
        ylabel('Lateral Displacement [m]')
        title('Problem 2.c: Lateral Displacement (E/W) vs. Northern
Windspeed')
```

```
        grid on
        hold off

        % Plot Lateral Displacement (North/South) against Windspeed
        figure()
        plot(wind_c(1,:),landinglocation(1,:),'--r','LineWidth',2);
hold on
        xlabel('Wind Speed [m/s]')
        ylabel('Lateral Displacement [m]')
        title('Problem 2.c: Lateral Displacement (N/S) vs. Northern
Windspeed')
        grid on
        hold off
```





Part d)

Compare Landing Distance for different masses given the constraint of kinetic energy

```
% Calculate Kinetic Energy Constraint from Original Mass
KE = 0.5*m*norm(x_i)^2;

% Create values for variable mass and speed based on KE constraint
m_d = linspace(0.01,1,50);
v_d = sqrt(2.*KE./m_d);

% Preallocate vector/cell for landing distance and trajectory
x_d_cell = cell(1,50);
landingdistance_d = zeros(50,1);

% Run ODE45 for various masses
for i = 1:50
    % Create Initial conditions vector with KE constrained speed
    x_i_d = x_i ./ norm(x_i) .* v_d(i);

    % Run ODE45
    [t,x_d] = ode45(@(t,x_d)
objectEOM(t,x_d,rho,Cd,A,m_d(i),g,wind_b), tspan, x_i_d, opts);
    landingdistance_d(i) = norm(x_d(end,1:3));
    x_d_cell(i) = {x_d};
end
```

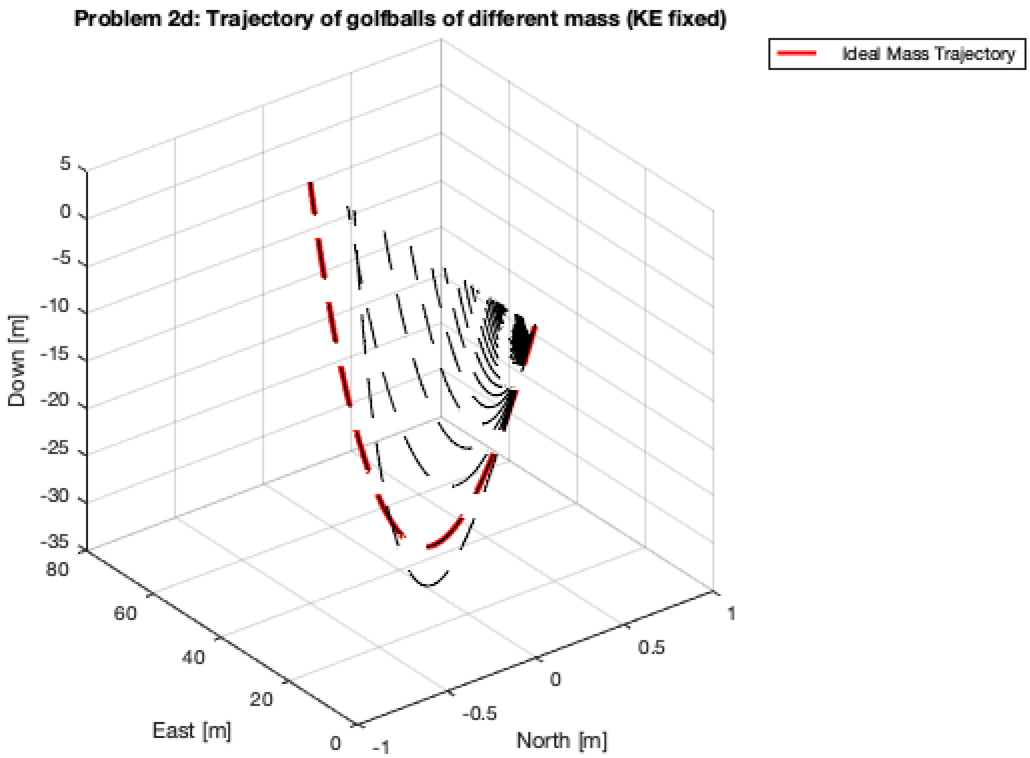
Best Optimized Mass

```
        % Calculate longest distance travelled
        [dist,i_d] = max(landingdistance_d);

        % Index ideal mass
        m_best = m_d(i_d);
        fprintf("The Most Optimized Mass for Distance was: %.3f [kg]
 \n",m_best)

        % Plot Trajectory for different mass values
        figure()
        % Plot and label ideal mass trajectory
        best = plot3(x_d_cell{1,i_d}(:,1),x_d_cell{1,i_d}
(:,2),x_d_cell{1,i_d}(:,3),'--r','LineWidth',2); hold on
        for i = 1:50
            plot3(x_d_cell{1,i}(:,1),x_d_cell{1,i}(:,2),x_d_cell{1,i}
(:,3),'--k')
        end
        legend(best,"Ideal Mass Trajectory")
        xlabel('North [m]')
        ylabel('East [m]')
        zlabel('Down [m]')
        title('Problem 2d: Trajectory of golfballs of different mass
 (KE fixed)')
        grid on
        hold off
```

The Most Optimized Mass for Distance was: 0.030 [kg]



Functions

Problem 1

```
function dx = odefunc(t,x)

    % Define derivatives as specified in lab document
    x_dot = x(1) + 2*x(2) + x(3);
    y_dot = x(1) - 5*x(3);
    z_dot = x(1)*x(2) - x(2)^2 + 3*x(3)^2;

    % Return state vector
    dx = [x_dot;y_dot;z_dot];
end
```

Problem 2

```
function xdot = objectEOM(t,x,rho,Cd,A,m,g,wind)

    % Define first order derivatives as pulled from state vector
    v_x = x(4);
    v_y = x(5);
    v_z = x(6);
```

```
    % Create inertial velocity vector
    v = [v_x;v_y;v_z];

    % Define air relative velocity vector
    v_a = v - wind;

    % Calculate drag and drag vector, opposite air relative velocity
    drag = 0.5 * rho * norm(v_a)^2 * Cd * A;
    a_d = (drag / m) * -v_a / norm(v_a);

    % Calculate 2nd order derivatives
    a_x = a_d(1);
    a_y = a_d(2);
    a_z = g + a_d(3);

    % Return state vector
    xdot = [v_x;v_y;v_z;a_x;a_y;a_z];

    % IF statement to prevent passing through the ground
    if x(3) > 0
        xdot = [0;0;0;0;0;0];
    end


end
```

Published with MATLAB® R2020a

1. Participation Table:

Name	Plan	Model	Experiment	Results	Report	Code	ACK
Caleb Bristol	X	X	X	X	2	2	X
Kyle Bowen	X	X	X	X	2	1	X
Qihan Cai	X	X	X	X	2	1	X
Tess Brodsky	X	X	X	X	2	2	X