Table of Contents


```
% Ian Faber, Ashton Miner, Teegan Oatley, Chaney Sullivan
% ASEN 3128-011
% ASEN3128Lab1Problem1.m
% Created: 8/23/22
```

Housekeeping

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

ODE45 Setup

```
% Initial conditions
X0 = [1; 1; 0.01];

% Integration time span
tspan = [0 .7];

% Tolerance settings
setup = odeset('RelTol', 1e-12, 'AbsTol', 1e-12);
```

ODE45 Call

```
[t, state] = ode45(@(t, state)EOM(t, state), tspan, X0, setup);
```

Plotting

```
figure(1)
sgtitle("1.a. ODE45 Integration Output")
subplot(3,1,1)
hold on
title("X vs Time")
xlabel("Time")
ylabel("X")
plot(t, state(:,1))
hold off

subplot(3,1,2)
hold on
title("Y vs Time")
xlabel("Time")
```

```
ylabel("Y")
plot(t, state(:,2))
hold off

subplot(3,1,3)
hold on
title("Z vs Time")
xlabel("Time")
ylabel("Z")
plot(t, state(:,3))
hold off
```

Published with MATLAB® R2022a

```
% Ian Faber, Ashton Miner, Teegan Oatley, Chaney Sullivan
% ASEN 3128-011
% EOM.m
% Created: 8/23/22

function [dX] = EOM(t,X)
%
% Inputs:   t = Time vector
%           X = State vector
%
% Outputs:  dX = change of state vector
%
% Methodology: Rate of change equations to be used in ode45 call

x = X(1);
y = X(2);
z = X(3);

xdot = x + 2*y + z;
ydot = x - 5*z;
zdot = x*y - y^2 + 3*(z^3);

dX = [xdot;ydot;zdot];

end
```

Published with MATLAB® R2022a

Table of Contents


```
% Ian Faber, Ashton Miner, Teegan Oatley, Chaney Sullivan
% ASEN 3128-011
% ASEN3128Lab1Problem2.m
% Created: 8/23/22
```

Housekeeping

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

Setup

```
mass = 0.03; % kg
diam = 0.03; % m
Cd = 0.6;
rho = 1.275; % kg/m^3
g = [0;0;9.81]; % m/s^2

A = pi*(diam/2)^2; % m^2

X0 = [0;0;0;0;20;-20]; % Initial conditions
tspan = [0 10]; % Integration time span
wind = [0;0;0]; % Wind vector

options = odeset('Events', @detectGround);
```

Trajectory Calculation and plottinh, problem 2 part b

```
[t, state] = ode45(@(t, state)ObjectEOM(t, state, rho, Cd, A, mass, g, wind),
 tspan, X0, options);

figure(1)
sgtitle("2.b. ODE45 Output")
subplot(3,1,1)
hold on
title("North Distance vs. Time")
xlabel("Time (sec)")
ylabel("North Distance (m)")
```

```
plot(t, state(:,1));
hold off

subplot(3,1,2)
hold on
title("East Distance vs. Time")
xlabel("Time (sec)")
ylabel("East Distance (m)")
plot(t, state(:,2));
hold off

subplot(3,1,3)
hold on
title("Down Distance vs. Time")
xlabel("Time (sec)")
ylabel("Down Distance (m)")
set(gca, 'YDir', 'reverse')
plot(t, state(:,3));
hold off


figure(2)
hold on
title("2.b. Ball Trajectory with No Wind")
xlabel("North")
ylabel("East")
zlabel("Down")
set(gca, 'ZDir', 'reverse')
view([30 35])
plot3(state(:,1), state(:,2), state(:,3))
hold off

% Calculate distance from the origin with no wind for later comparison
distance0 = norm([state(end,1), state(end,2)]);
```

Wind Sensitivity Analysis, problem 2 part c

```
% Test vector of north windspeeds
testWind = -40:1:40;

% Plot label index initialization
kk = 0;

figure(3)
hold on
title("2.c. Trajectory with Varying North Windspeed")
xlabel("North")
ylabel("East")
zlabel("Down")
set(gca, 'ZDir', 'reverse')
view([30 35])

for k = 1:length(testWind)
```

```
    % Update wind vector with values from test vector
    wind = [testWind(k); 0; 0];

    % Calculate trajectory with updated wind vector
    [t, state] = ode45(@(t, state)ObjectEOM(t, state, rho, Cd, A, mass, g,
 wind), tspan, X0, options);

    % Calculate distance from origin
    distance(k) = norm([state(end, 1), state(end, 2)]);

    % Plot every 20th trajectory (increments of 20 m/s, starting at -40)
    % with the updated wind vector
    if mod(k-1,20) == 0
        kk = kk + 1;
        plot3(state(:,1), state(:,2), state(:,3))
        % Create label vector for dynamic legend
        label(kk) = sprintf("North windspeed %.0f m/s", testWind(k));
    end

end
% Create legend
legend(label, 'Location', 'best')
hold off

figure(4)
hold on
title("2.c. Distance vs North Windspeed")
xlabel("North wind speed (m/s)")
ylabel("Distance (m)")
plot(testWind, distance)
hold off

% Calculate North axis deflection vector
deflection = distance - distance0;

% figure(5)
% hold on
% title("Deflection vs North Windspeed")
% xlabel("North wind speed (m/s)")
% ylabel("Deflection in North (m)")
% plot(testWind, deflection)
% hold off
```

Kinetic Energy Limitation Analysis, problem 2 part d

```
% Extract initial velocity vector and calculate the corresponding limited
% kinetic energy
V = X0(4:6);
speed = norm(V);
unitV = V/speed;
kineticE = 0.5*mass*speed^2
```

```
% Set up ball mass vector in kg, reset wind vector and label index
ballMass = (10:1:50)/1000; % g to kg
wind = [0; 0; 0];
kk = 0;

figure(6)
hold on
title("2.d. Trajectory with Varying Ball Mass")
xlabel("North")
ylabel("East")
zlabel("Down")
set(gca, 'ZDir', 'reverse')
view([30 35])

for k = 1:length(ballMass)
    % Calculate initial ball speed based on available kinetic energy and
    % a mass from the mass vector
    ballSpeed = sqrt((2*kineticE)/ballMass(k));

    % Calculate initial velocity vector
    ballV = ballSpeed*unitV;

    % Update initial conditions vector
    X0 = [0;0;0;ballV];

    % Calculate trajectory with updated ball mass and initial velocity
    [t, state] = ode45(@(t, state)ObjectEOM(t, state, rho, Cd, A, ballMass(k),
 g, wind), tspan, X0, options);

    % Calculate distance from origin for each mass in the mass vector
    distanceMass(k) = norm([state(end, 1), state(end, 2)]);

    % Plot every 10th trajectory (increments of 10 g, starting at 10) with
    % the updated ball mass
    if mod(k-1, 10) == 0
        kk = kk + 1;
        plot3(state(:,1), state(:,2), state(:,3));
        % Create label vector for dynamic legend
        label(kk) = sprintf("Ball mass %.0f g", ballMass(k)*1000);
    end
end

% Create legend
legend(label, 'Location', 'best')
hold off

figure(7)
hold on
title("2.d. Distance vs. Ball Mass")
xlabel("Ball mass (kg)")
ylabel("Distance (m)")
plot(ballMass, distanceMass)
hold off
```

Published with MATLAB® R2022a

```
% Ian Faber, Ashton Miner, Teegan Oatley, Chaney Sullivan
% ASEN 3128-011
% ObjectEOM.m
% Created: 8/23/22

function xdot = ObjectEOM(t,x,rho,Cd,A,m,g,wind)
%
% Inputs:   t = Time vector (sec)
%           x = State vector (m, m/s)
%             = [x; y; z; vx; vy; vz]
%           rho = Air density (kg/m^3)
%           Cd = Coefficient of drag
%           A = Cross-sectional area of golf ball relative to motion (m^2)
%           m = Mass of golf ball (kg)
%           g = Freefall acceleration due to gravity (m/s^2)
%           wind = Wind vector (m/s)
%                = [windx; windy; windz]
%
% Outputs: xdot = rate of change of state vector (m/s, m/s^2)
%               = [vx; vy; vz; ax; ay; az]
%
% Methodology: Function used by ode45 to calculate golf ball trajectory,
%              problem 2 part a

% Extract inertial velocity and calculate wind-relative velocity
Ve = x(4:6);
V = Ve - wind;

% Position rate of change is inertial velocity
pdot = Ve;

% Calculate speed and the wind-relative velocity unit vector
mag = norm(V);
unitV = V/mag;

% Calculate drag and weight forces
Fdrag = (-0.5*rho*(mag^2)*Cd*A) * unitV;
Fgrav = m*g;

% Combine forces into total force vector
F = Fdrag + Fgrav;

% Velocity rate of change is a = F/m (F = ma)
vdot = F/m;

% Format output vector
xdot = [pdot; vdot];

end
```

Published with MATLAB® R2022a

```matlab
% Ian Faber, Ashton Miner, Teegan Oatley, Chaney Sullivan
% ASEN 3128-011
% detectGround.m
% Created: 8/30/22

function [value, isterminal, direction] = detectGround(t, X)
%
%   Inputs:      t = Time vector
%                X = State vector
%                  = [x; y; z; vx; vy; vz]
%
%   Outputs:     value = Value of state vector z component
%                isterminal = Boolean used to stop or continue integration
%                direction = Indicator for detecting when a value occurs
%
%   Methodology: Function passed to odeset that detects when to stop
%                integration based on the value of a specified state vector
%                component

% Interested in detecting when z goes to 0
z = X(3);

value = z; % Look at the value of z
isterminal = 1; % 1: stop integration once z hits 0
direction = 1; % 1: Trigger when value goes to 0 from negative to positive
end
```

Published with MATLAB® R2022a