AME 556 HW2

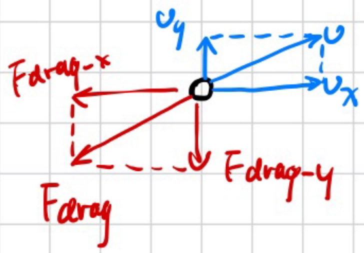
Name: Shuai Zhao Uscid: 7722927131

Problem 1

a. Derive the dynamics of the system.

By Newton's Law:

$$m\ddot{\vec{p}} = m\vec{g} + \vec{F_{rag}}$$



Divide it into x, y:

Since $\vec{F_{drag}} = -|cv|\vec{v}$,

$$|F_{drag\_x}| = cv^2 \cdot \frac{v_x}{|\vec{v}|} = c|\vec{v}|\cdot v_x$$

$$|F_{drag\_y}| = cv^2 \cdot \frac{v_y}{|\vec{v}|} = c|\vec{v}|\cdot v_y$$

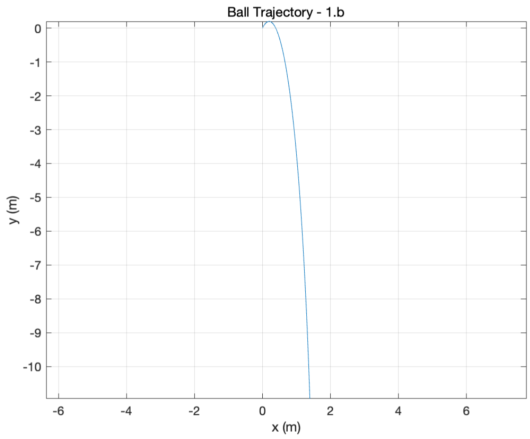
Thus, the dynamic:

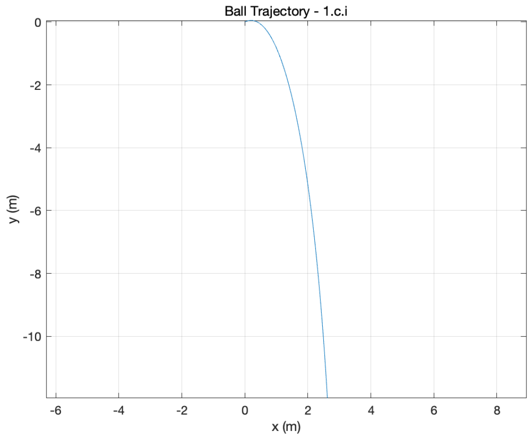
$$\begin{cases} m\dot{v}_x = -c|\vec{v}|\cdot v_x = -c\sqrt{v_x^2+v_y^2}\cdot v_x \\ m\dot{v}_y = -mg - c|\vec{v}|\cdot v = -mg - c\sqrt{v_x^2+v_y^2}\cdot v_y \end{cases}$$

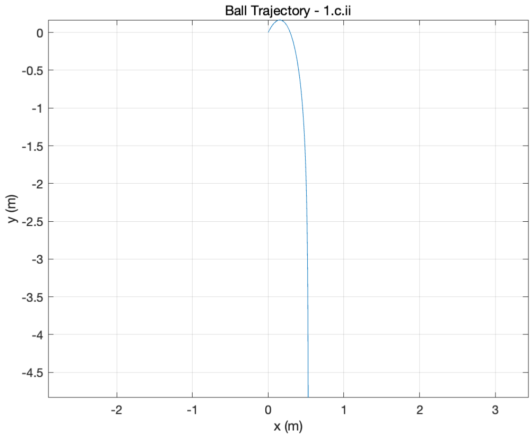
$$\Rightarrow \begin{cases} \dot{v}_x = -\frac{c}{m}\sqrt{v_x^2+v_y^2}\cdot v_x \\ \dot{v}_y = -g - \frac{c}{m}\sqrt{v_x^2+v_y^2}\cdot v_y \end{cases}$$

b. & c.

- Plots







- Videos

  https://drive.google.com/drive/folders/1am3pdWizw8bbAHBCgThMnKUuKT9bJgz0?usp=share_link

Code

```
clc;
clear;
```

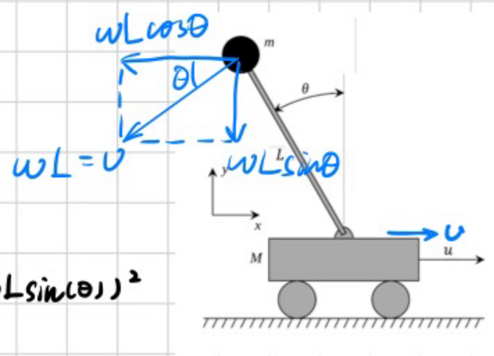
```
close all;

%% parameters
m = 0.5;
g = 9.81;
c = [0.05, 0.05, 0.5];
tspan = [0, 2];

% initial_conditions
X0s = [
    0, 0, 1, 2;    % 1.b
    0, 0, 2, 1;    % 1.c.1
    0, 0, 1, 2     % 1.c.2
];

% labels on the plots and videos
labels = {'1.b', '1.c.i', '1.c.ii'};

%% main
for i = 1:length(c)
    % parameters
    current_c = c(i);
    X0 = X0s(i, :);

    % compute ode
    [t, X] = ode45(@(t, X) dynamics(t, X, m, g, current_c), tspan, X0);

    % Extract states from X
    x = X(:, 1);
    y = X(:, 2);
    vx = X(:, 3);
    vy = X(:, 4);

    plot_trajectory(x, y, labels{i});
    generate_video(t, x, y, labels{i});
end

function X_dot = dynamics(~, X, m, g, c)
    % |v|
    vx = X(3);
    vy = X(4);
    v = sqrt(vx^2 + vy^2);

    % a
    ax = -c / m * v * vx;
    ay = -g - c / m * v * vy;

    % q_dot
    X_dot = [vx; vy; ax; ay];
end

%% plot function
```

```
function plot_trajectory(x, y, label)
    if ~exist('plots', 'dir')
        mkdir('plots');
    end

    % plot
    figure;
    plot(x, y);
    title(['Ball Trajectory - ' label]);
    xlabel('x (m)');
    ylabel('y (m)');
    axis equal;
    grid on;

    % save plots
    saveas(gcf, fullfile('plots', ['trajectory_' label '.png']));
end

%% video function
function generate_video(t, x, y, label)
    if ~exist('videos', 'dir')
        mkdir('videos');
    end

    % video
    video = VideoWriter(fullfile('videos', ['animation_' label '.mp4']), 'MPEG-4');
    video.FrameRate = 30;
    open(video);

    % plot
    figure;
    hold on;
    plot(x, y, 'r--');
    ball = plot(x(1), y(1), 'bo', 'MarkerSize', 10, 'MarkerFaceColor', 'b');
    axis equal;
    title(['Ball Motion Animation - ' label]);
    xlabel('x (m)');
    ylabel('y (m)');
    grid on;

    time_text = text(0.1, 0.9, '', 'Units', 'normalized', 'FontSize', 12, 'Color',
'black');

    for i = 1:length(t)
        set(ball, 'XData', x(i), 'YData', y(i));

        % timer
        set(time_text, 'String', sprintf('Time: %.2f s', t(i)));

        frame = getframe(gcf);
        writeVideo(video, frame);
    end
```

```
    hold off;
    close(video);
end
```

Problem 2

a. Derive the dynamic of the robot.



a. Let $q = \begin{bmatrix} x \\ \theta \end{bmatrix}$, $\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{\theta} \end{bmatrix}$

$v_M = \dot{x}$, $\vec{v_m} = \omega L$,

$v_{mx} = -\omega L \cos\theta$, $v_{my} = -\omega L \sin\theta$

$$K = \frac{1}{2}Mv^2 + \frac{1}{2}m(v - \omega L\cos(\theta))^2 + \frac{1}{2}m(-\omega L \sin(\theta))^2$$

$$P = mgh_m = mgL\cos(\theta)$$

Then, $L = K - P$

$$= \frac{1}{2}Mv^2 + \frac{1}{2}m(v - \omega L\cos(\theta))^2 + \frac{1}{2}m(\omega L \sin(\theta))^2 - mgL\cos(\theta)$$

$$= \frac{1}{2}M\dot{x}^2 + \frac{1}{2}m(\dot{x} - \dot{\theta}L\cos(\theta))^2 + \frac{1}{2}m(\dot{\theta}L\sin(\theta))^2 - mgL\cos(\theta)$$

So, $$\frac{\partial L}{\partial \dot{q}} = \begin{bmatrix} \frac{\partial L}{\partial \dot{x}} \\ \frac{\partial L}{\partial \dot{\theta}} \end{bmatrix} = \begin{bmatrix} M\dot{x} + m\dot{x} - m\dot{\theta}L\cos(\theta) \\ -m\dot{x}\cdot L\cos(\theta) + m\dot{\theta}L^2 \end{bmatrix}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) = \begin{bmatrix} M\ddot{x} + m\ddot{x} - mL\left[\ddot{\theta}\cos(\theta) - \dot{\theta}^2\sin(\theta)\right] \\ -m\ddot{x}L\cos(\theta) + m\dot{x}L\sin(\theta)\cdot\dot{\theta} + m\ddot{\theta}L^2 \end{bmatrix}$$

$$\frac{\partial L}{\partial q} = \begin{bmatrix} 0 \\ -m\dot{x}\dot{\theta}L\sin(\theta) + mgL\sin(\theta) \end{bmatrix}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \begin{bmatrix} M\ddot{x} + m\ddot{x} - mL\left[\ddot{\theta}\cos(\theta) - \dot{\theta}^2\sin(\theta)\right] \\ -m\ddot{x}L\cos(\theta) + m\ddot{\theta}L^2 - mgL\sin(\theta) \end{bmatrix} = \tau = \begin{bmatrix} u \\ 0 \end{bmatrix}$$

b. derive symbolic functions of $D(q), N(q, \dot{q})$:

(1) Result

```
D(q):
[          M + m, -L*m*cos(theta)]
[-L*m*cos(theta),           L^2*m]

N(q, q_dot):
L*m*theta_dot^2*sin(theta)
         -L*g*m*sin(theta)

>>
```

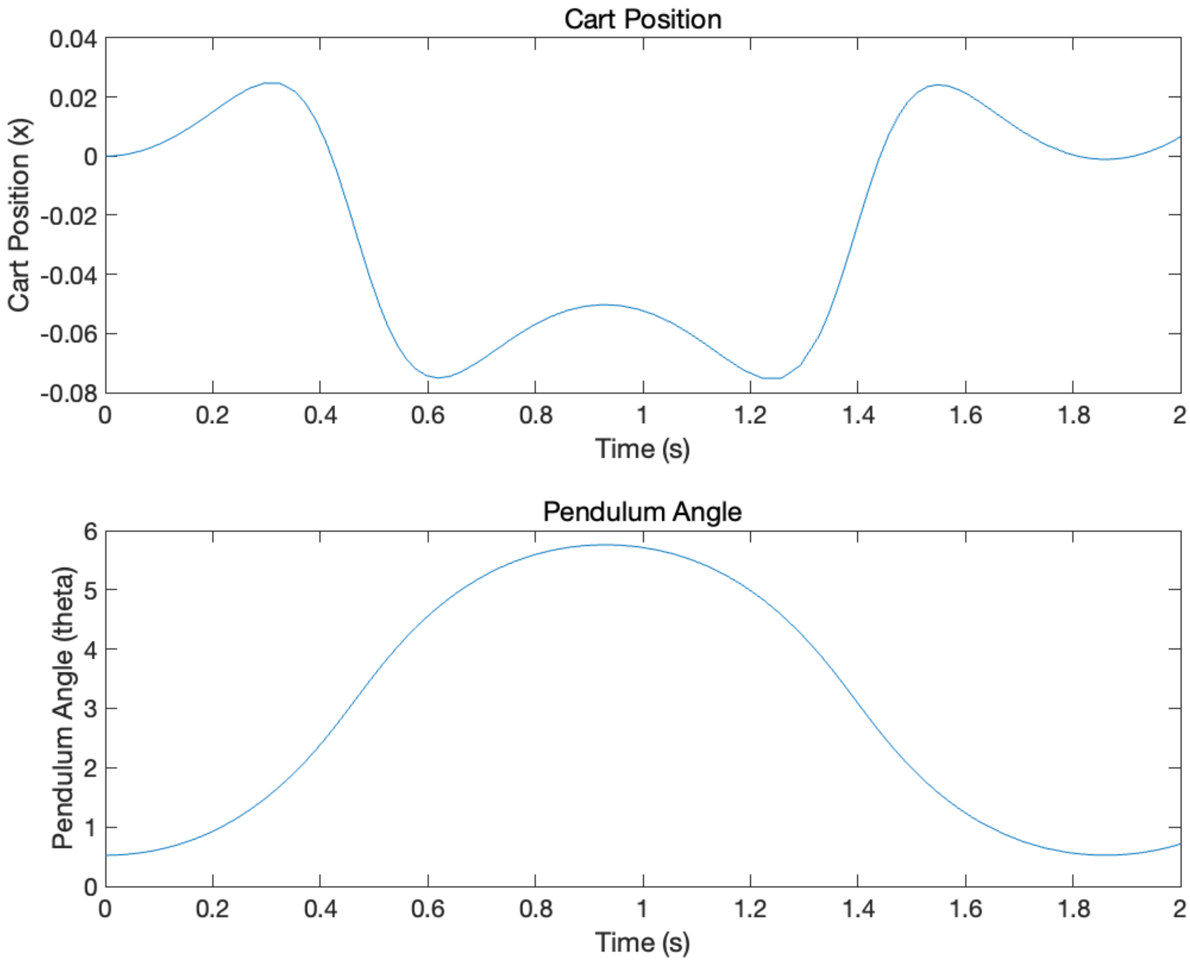
(2) Code

```
clc;
clear;

%% parameters
syms x theta x_dot theta_dot x_ddot theta_ddot real
syms M m L g real

% q
q = [x; theta];
q_dot = [x_dot; theta_dot];

%% compute La = K - P
K_cart = 0.5 * M * x_dot^2;
K_pole = 0.5 * m * (x_dot - theta_dot * L * cos(theta))^2 + 0.5 * m * (-theta_dot * L *
sin(theta))^2;
K = K_cart + K_pole;

P = m * g * L * cos(theta);

La = K - P;
La = simplify(La);

%% compute D, N
% D(q)
D = jacobian(jacobian(La, q_dot), q_dot);
D = simplify(D);

% C(q,q_dot)*q_dot
f = D * q_dot;
Cq_dot = jacobian(f,q) * q_dot - jacobian(K,q)';

% G(q)
G = jacobian(P, q)';
```

```
% N(q,q_dot)
N = Cq_dot + G;
N = simplify(N);

%% answer
disp('D(q):');
disp(D);
disp('N(q, q_dot):');
disp(N);
```

c. simulate the system in 2 seconds

(1) Plot



(2) Video

https://drive.google.com/drive/folders/1am3pdWizw8bbAHBCgThMnKUuKT9bJgz0?usp=share_link

(3) Code

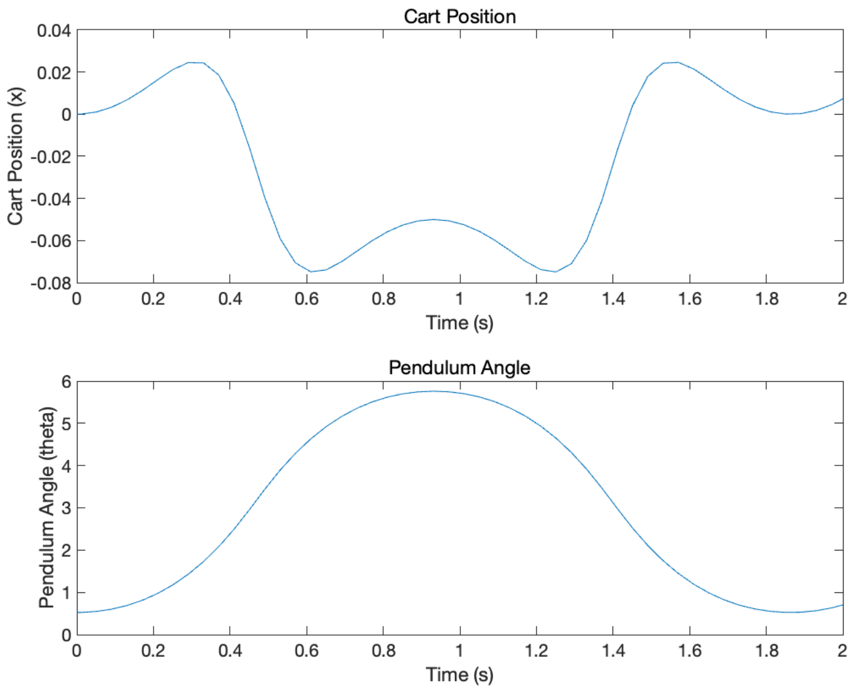
```
clc;
clear;
close all;

% parameters
M = 1;
m = 0.2;
L = 0.3;
g = 9.81;

% I.C.
x0 = 0;
x_dot0 = 0;
theta0 = pi/6;
theta_dot0 = 0;

q0 = [x0; x_dot0; theta0; theta_dot0];

%% main
% ODE
tspan = [0 2];
[t, q] = ode45(@(t, q) dynamics(t, q, M, m, L, g), tspan, q0);

% extract x & theta
x = q(:, 1);
theta = q(:, 3);

% plot and video
plot_x_theta(t, x, theta);
generate_video(t, x, theta, L);

%% dynamics
function q_dot = dynamics(~, q, M, m, L, g)
    x = q(1);
    x_dot = q(2);
    theta = q(3);
    theta_dot = q(4);

    % D & N
    D = [M + m, -L * m * cos(theta);
         -L * m * cos(theta), L^2 * m];
    N = [L * m * theta_dot^2 * sin(theta);
         -L * g * m * sin(theta)];

    a = D \ (-N);    % tau = 0

    % q_dot
    q_dot = [x_dot; a(1); theta_dot; a(2)];
end
```

```matlab
%% plot x(t) & theta(t)
function plot_x_theta(t, x, theta)
    if ~exist('plots', 'dir')
        mkdir('plots');
    end

    figure;
    subplot(2, 1, 1);
    plot(t, x);
    xlabel('Time (s)');
    ylabel('Cart Position (x)');
    title('Cart Position');

    subplot(2, 1, 2);
    plot(t, theta);
    xlabel('Time (s)');
    ylabel('Pendulum Angle (theta)');
    title('Pendulum Angle');
    saveas(gcf, fullfile('plots', '2.c.png'));
end

%% video
function generate_video(t, x, theta, L)
    if ~exist('videos', 'dir')
        mkdir('videos');
    end

    figure;
    cart_width = 0.2;
    cart_height = 0.1;
    pendulum_radius = 0.03;

    axis equal;
    set(gca, 'XLim', [-0.8, 0.5]);
    set(gca, 'YLim', [-0.4, 0.5]);
    xlabel('X Position');
    ylabel('Y Position');
    title('Cart-Pole System Animation');

    video = VideoWriter(fullfile('videos', 'animation_2_c.mp4'), 'MPEG-4');
    open(video);

    for i = 1:length(t)
        cla;

        hold on;
        % Draw cart
        rectangle('Position', [x(i) - cart_width / 2, -cart_height, cart_width,
cart_height], ...
                  'FaceColor', [0 0 1], 'EdgeColor', [0 0 1]);
```

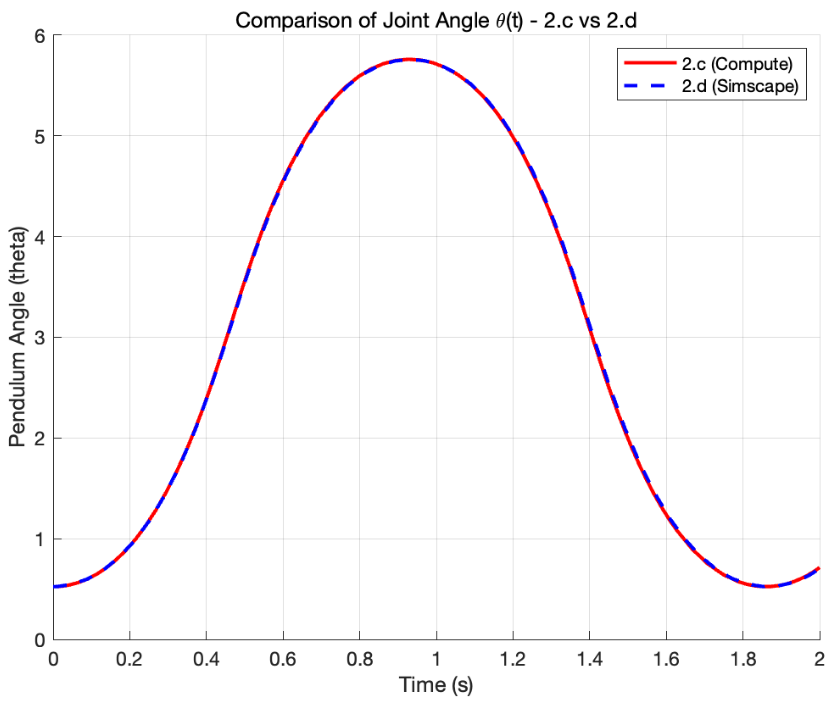
```
        % Compute pendulum position
        pendulum_x = x(i) - L * sin(theta(i));
        pendulum_y = L * cos(theta(i));

        % Draw pendulum
        rectangle('Position', [pendulum_x - pendulum_radius, pendulum_y - pendulum_radius,
...
                  2 * pendulum_radius, 2 * pendulum_radius], ...
                  'Curvature', [1, 1], 'FaceColor', [1 0 0], 'EdgeColor', [1 0 0]);

        % Draw rod
        plot([x(i), pendulum_x], [0, pendulum_y], 'r-', 'LineWidth', 2);

        % Display information
        theta_deg = rad2deg(theta(i));
        text(-0.6, 0.4, sprintf('Time: %.2f s', t(i)), 'FontSize', 10, 'Color', 'k');
        text(-0.6, 0.35, sprintf('x: %.2f m', x(i)), 'FontSize', 10, 'Color', 'k');
        text(-0.6, 0.3, sprintf('\\theta: %.2f rad (%.2f°)', theta(i), theta_deg),
'FontSize', 10, 'Color', 'k');

        drawnow;

        % Capture and write frame
        frame = getframe(gcf);
        writeVideo(video, frame);
    end

    close(video);
end
```

d. Use MATLAB Simscape to solve problem 2.c.

(1) Plot for 2.d.



(2) Compare with 2.c.



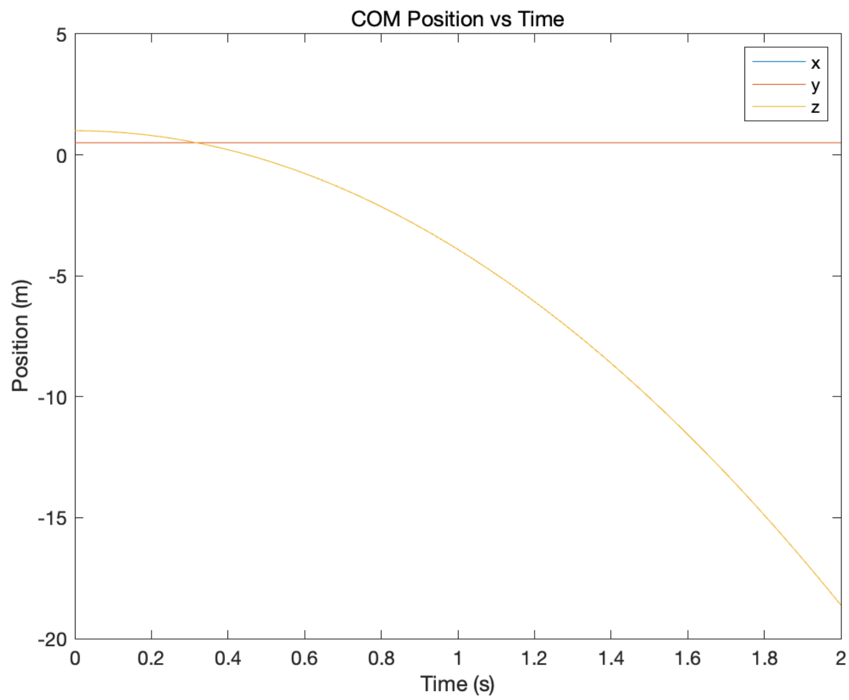
(3) Video

https://drive.google.com/drive/folders/1am3pdWizw8bbAHBCgThMnKUuKT9bJgz0?usp=share_link

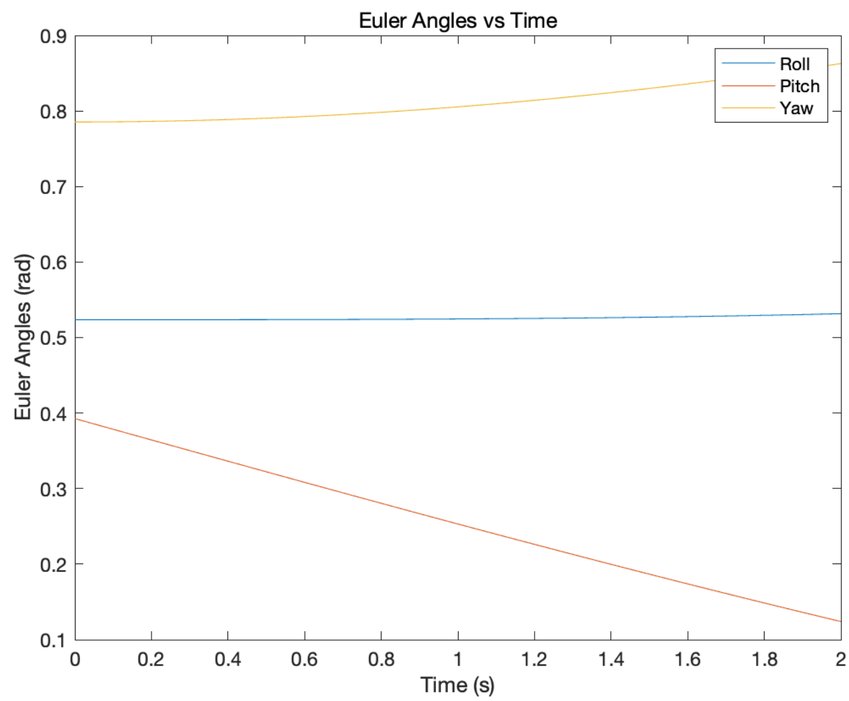
Problem 3

(1) Plot

- COM's positions



- Body orientation's Euler angles



(2) Video

https://drive.google.com/drive/folders/1am3pdWizw8bbAHBCgThMnKUuKT9bJgz0?usp=share_link

(3) Code

```
clc;
clear;
close all;

% parameters
m = 0.5;
I = diag([0.01, 0.01, 0.05]);
g = [0; 0; -9.81];

% I.C.
p0 = [0.5; 0.5; 1];
v0 = [0; 0; 0];
R0 = eul2rotm([pi/6, pi/8, pi/4]);
w0 = [0; -0.1; 0.1];

q0 = [p0; reshape(R0, [], 1); v0; w0];

% ode
tspan = [0, 2];
[t, q] = ode45(@(t, q) uav_dynamics(t, q, m, I, g), tspan, q0);

% extract from q
position = q(:, 1:3);
R = reshape(q(:, 4:12)', 3, 3, []);
euler = zeros(length(t), 3);
for i = 1:length(t)
    euler(i, :) = rotm2eul(R(:, :, i));
end

%% plot
if ~exist('plots', 'dir')
    mkdir('plots');
end

% pos
figure;
plot(t, position);
xlabel('Time (s)');
ylabel('Position (m)');
legend('x', 'y', 'z');
title('COM Position vs Time');
saveas(gcf, fullfile('plots', 'Position.png'));

% euler
figure;
plot(t, euler);
xlabel('Time (s)');
ylabel('Euler Angles (rad)');
legend('Roll', 'Pitch', 'Yaw');
```

```
title('Euler Angles vs Time');
saveas(gcf, fullfile('plots', 'Euler.png'));

%% animate
generate_animation(t, q);

%% dynamics
function dq = uav_dynamics(~, s, m, I, g)
    p = s(1:3);
    R = reshape(s(4:12), 3, 3);
    v = s(13:15);
    wb = s(16:18);

    dp = v;
    dR = R * skew_symmetric(wb);
    dv = g;

    tau = [0; 0; 0];
    dwb = I \ (tau - cross(wb, I * wb));

    dq = [dp; reshape(dR, [], 1); dv; dwb];
end

%% Compute S(omega)
function S = skew_symmetric(omega)
    S = [0, -omega(3), omega(2);
         omega(3), 0, -omega(1);
        -omega(2), omega(1), 0];
end

function generate_animation(t, state)
    position = state(:, 1:3);
    R_matrices = reshape(state(:, 4:12)', 3, 3, []);

    % video open
    v = VideoWriter('uav_simulation.mp4', 'MPEG-4');
    v.FrameRate = 30;
    open(v);
    figure;

    % arms
    d = 0.2;
    r = 0.05;
    arm_body = [
        d, 0, 0;
        -d, 0, 0;
        0, d, 0;
        0, -d, 0
    ]';
    colors = {'r', [0.6, 0.8, 1], 'g', 'b'};

    for i = 1:length(t)
```

```
        pos = position(i, :)';
        R = R_matrices(:, :, i);
        arm_world = R * arm_body + pos;

        % subplot for different view
        for subplot_index = 1:4
            subplot(2, 2, subplot_index);
            hold on;
            grid on;

            % arms
            for j = 1:4
                plot3([pos(1), arm_world(1, j)], ...
                      [pos(2), arm_world(2, j)], ...
                      [pos(3), arm_world(3, j)], ...
                      'Color', colors{j}, 'LineWidth', 2);
            end

            % circles
            for j = 1:4
                draw_circle(arm_world(:, j), r, R);
            end

            % set view
            switch subplot_index
                case 1
                    view(0, 0); % Front
                    title('Front View');
                case 2
                    view(-90, 0); % Left
                    title('Left View');
                case 3
                    view(0, 90); % Top
                    title('Top View');
                case 4
                    view(3); % 3D
                    title('3D View');
            end

            % change the xylim to keep uav at the center of the plot
            % or the uav would fall out from z-
            xlim([pos(1) - 0.5, pos(1) + 0.5]);
            ylim([pos(2) - 0.5, pos(2) + 0.5]);
            zlim([pos(3) - 0.5, pos(3) + 0.5]);
            xlabel('X'); ylabel('Y'); zlabel('Z');
            hold off;
        end

        drawnow;
        frame = getframe(gcf);
        writeVideo(v, frame);
```

```
        if i < length(t)
            clf;
        end
    end

    close(v);
end

function draw_circle(center, radius, R)
    theta = linspace(0, 2*pi, 50);
    circle_body = radius * [cos(theta); sin(theta); zeros(1, length(theta))];
    circle_world = R * circle_body + center;
    plot3(circle_world(1, :), circle_world(2, :), circle_world(3, :), 'k', 'LineWidth',
1);
end
```