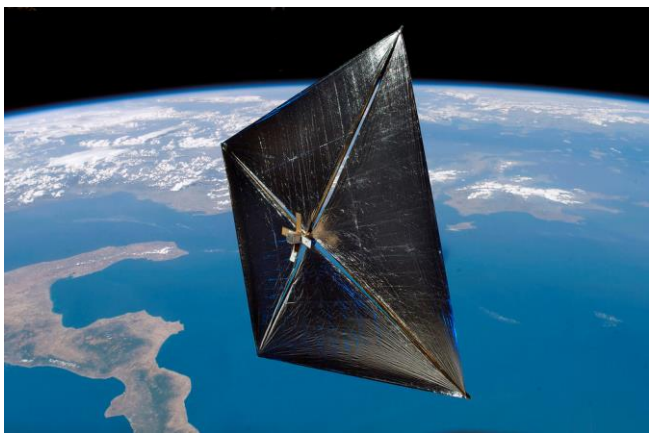


Homework Assignment II

UUM 532E - Spacecraft Control Systems / CRN: 22670 Dr. Demet

ABSTRACT

This homework involves simulating the dynamics and control of a sail craft with a gimballed sail mechanism. The primary objective is to achieve the desired attitudes using quaternion-based control and propagate the spacecraft's attitude over time.

Khady sarah Sall

922310129

I. Introduction

In this exercise, we aim to orient and control a solar sail in two specific directions (case 1 and case 2). To achieve our objectives, we have followed a specific control law that involves determining the following elements:

Error Quaternion: Calculate the error quaternion by comparing the current orientation of the solar sail with the desired orientation.
Calculate Control Torque: Determine the control torques required based on the error quaternion and angular velocities.
Calculate Gimbal Angles: Compute the necessary gimbal angles to correctly orient the solar sail.
Calculate New Spacecraft Center of Mass: Update the spacecraft's center of mass based on the new gimbal configurations and components.
Calculate Actual Control Torques Produced: Determine the forces produced by the sail and the actual control torques applied to the spacecraft.
Calculate New Moment of Inertia Matrix and Its Inverse: Update the spacecraft's inertia matrix based on the new configurations and calculate its inverse.
Simulate Spacecraft Dynamics (Euler's Equation): Propagate the spacecraft states using Euler's equation to simulate its dynamics.
Calculate New Sun Vector in the Body Frame: Update the sun vector in the body frame based on the new orientations of the solar sail.

The complete dynamics of our spacecraft, along with all necessary parameters and expressions, have been provided, including initial conditions. All values and data used originate from the exercise itself, with no external data employed except for clarification or comparison purposes.
This process includes a time-stepping loop where the spacecraft states are propagated, quaternions are normalized, and results are stored for analysis. Plots of Euler angles, angular velocities, and control torques are generated to visualize the performance and dynamics of the control system.

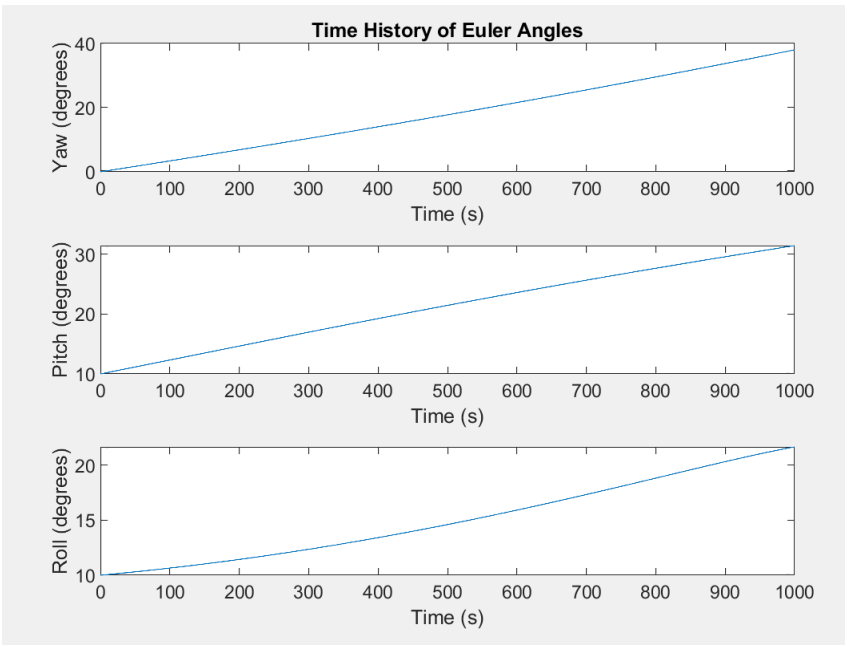
*For both cases, **we calculated the initial quaternion from the given set of initial Euler angles**. We also **normalized the quaternions** and ensured they were **real** whenever necessary, given the numerous error messages we received regarding the calculation of the sun vector. The calculations of Gimbal Angles, New Spacecraft Center of Mass, New Moment of Inertia Matrix and Its Inverse, and New Sun Vector in the Body Frame can be viewed in the **workspace** for each case. Their size prevented us from including them in the report, especially since they do not constitute the main subject of this study, although they were essential for calculating the required and necessary parameters. We considered a time step of **0.1** second.*
For each case, we plotted the Euler angles, angular velocities (omega), the desired control torque, and the actual control torque.

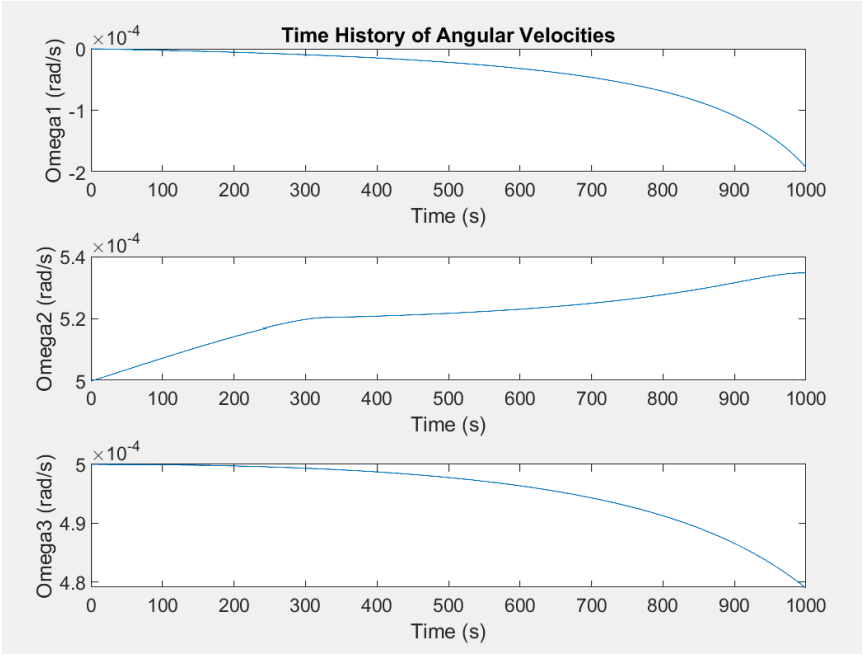
II. Case 1

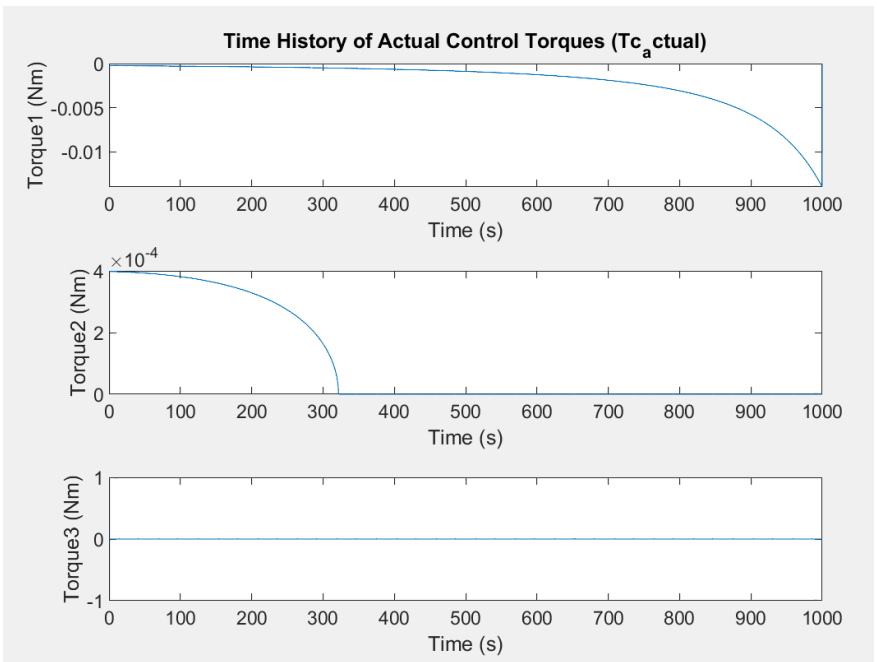
Zero Desired Attitude ($\boldsymbol{\theta}$ = [0∘, 0∘, 0∘])

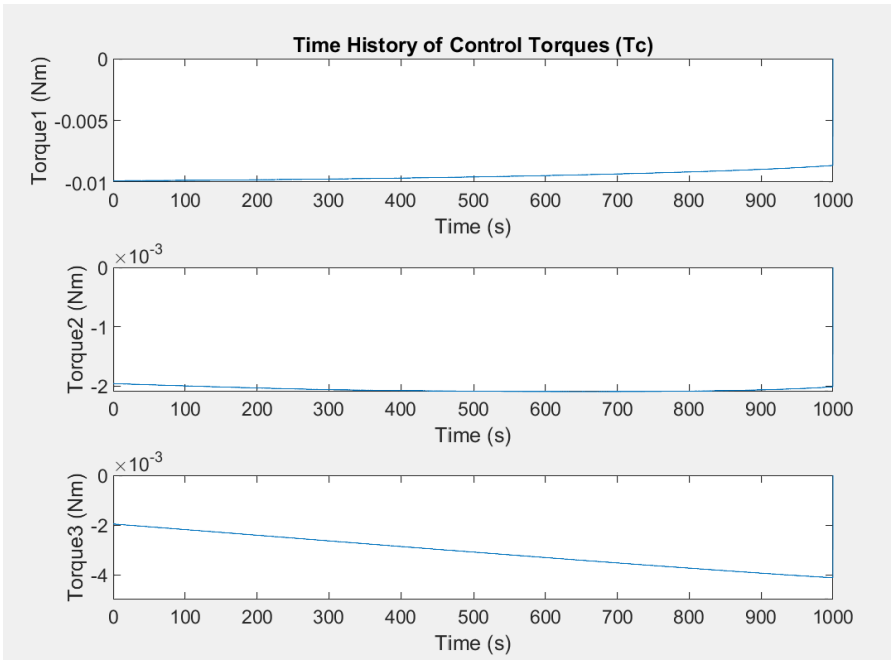
• Initial attitude of $\boldsymbol{\theta}$ = [θ1, θ2, θ3] = [0∘, 10∘, 10∘]
• Initial angular velocities of [ω1, ω2, ω3] = [0, 5, 5] × 10-4 rad/s

- Simulation time 1000s

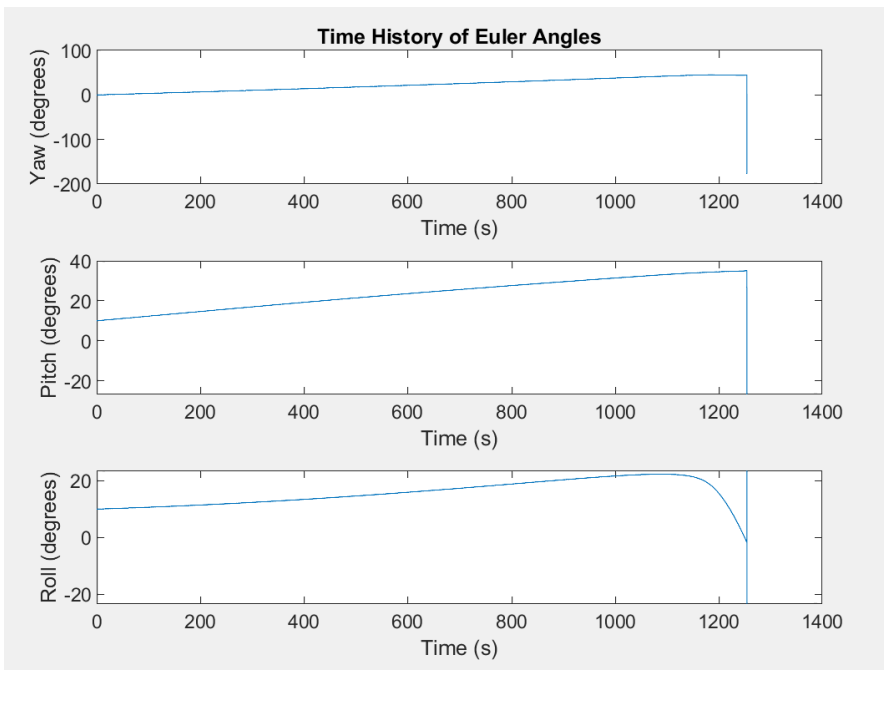


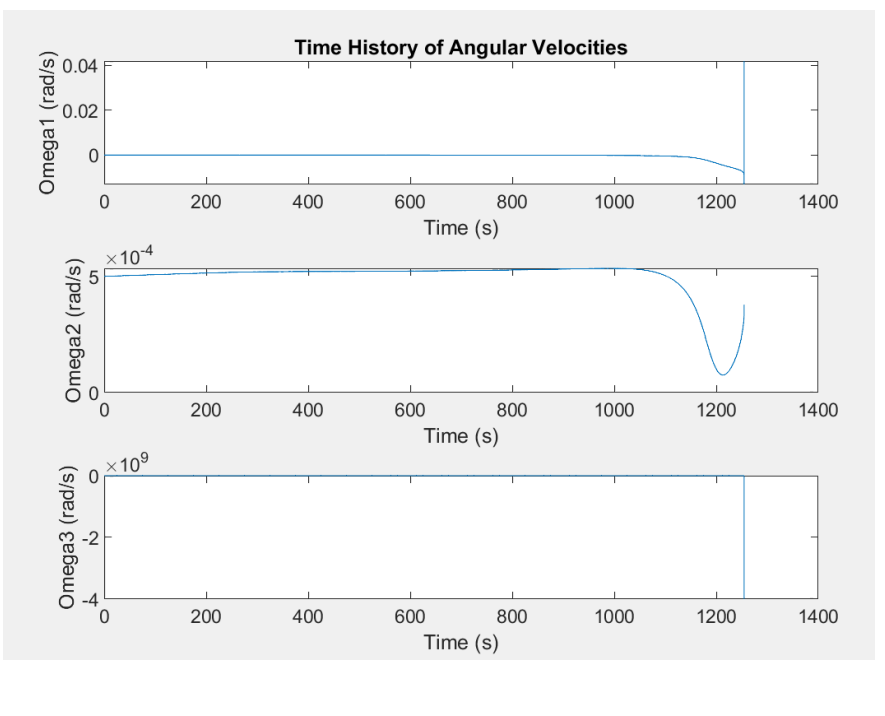


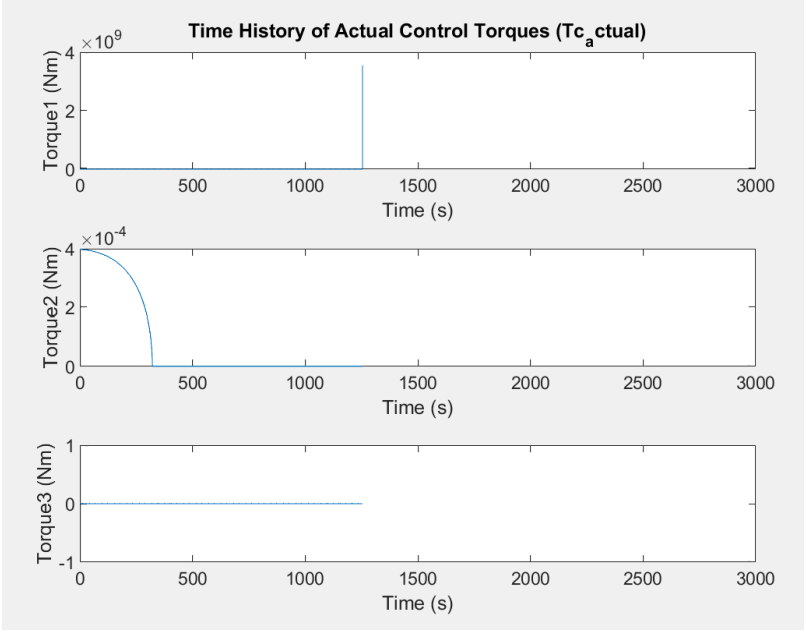


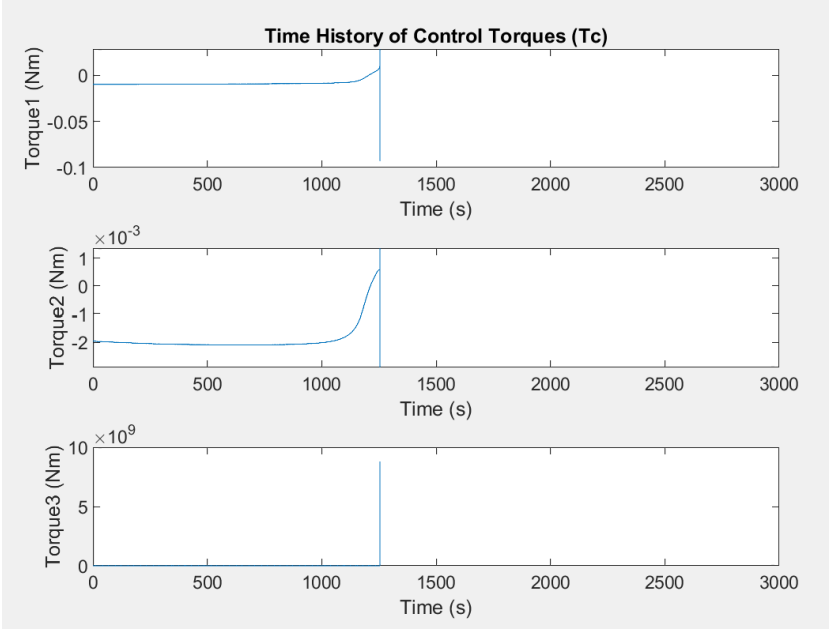


- Simulation time 3000s

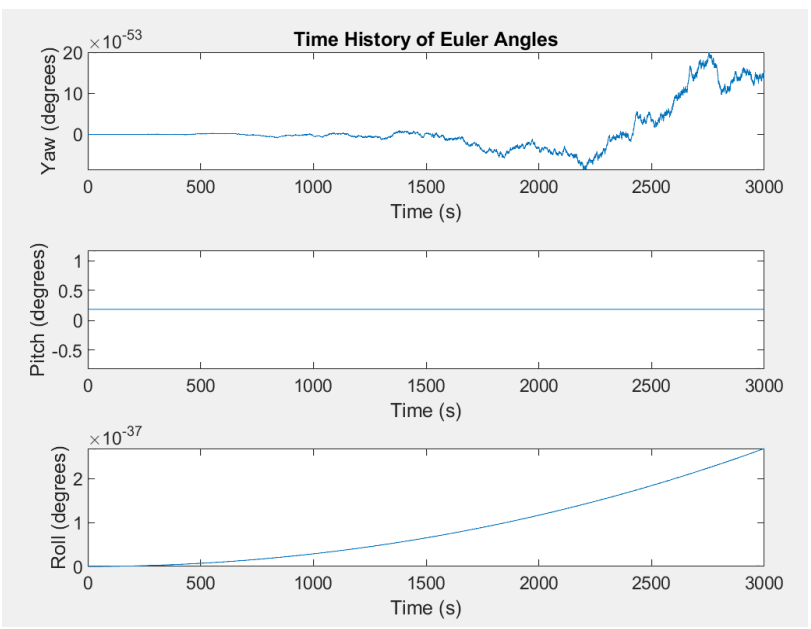


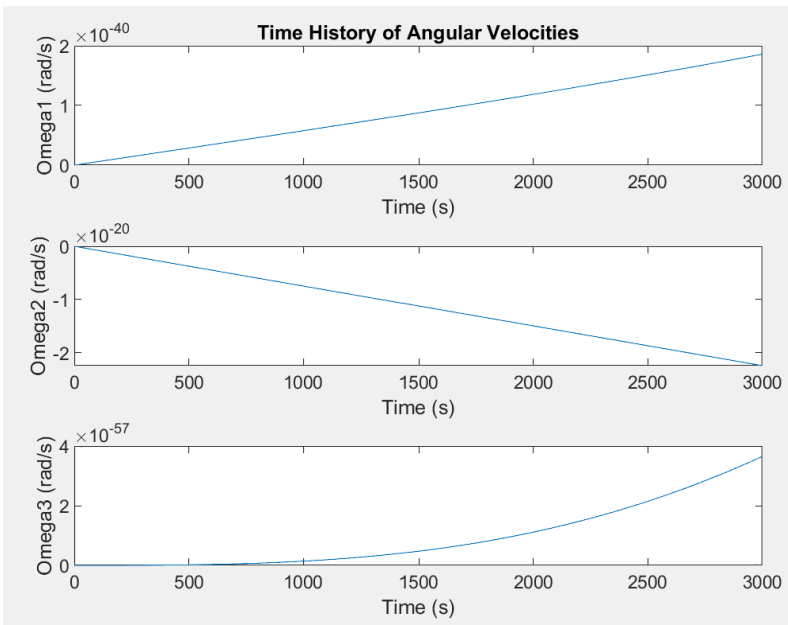


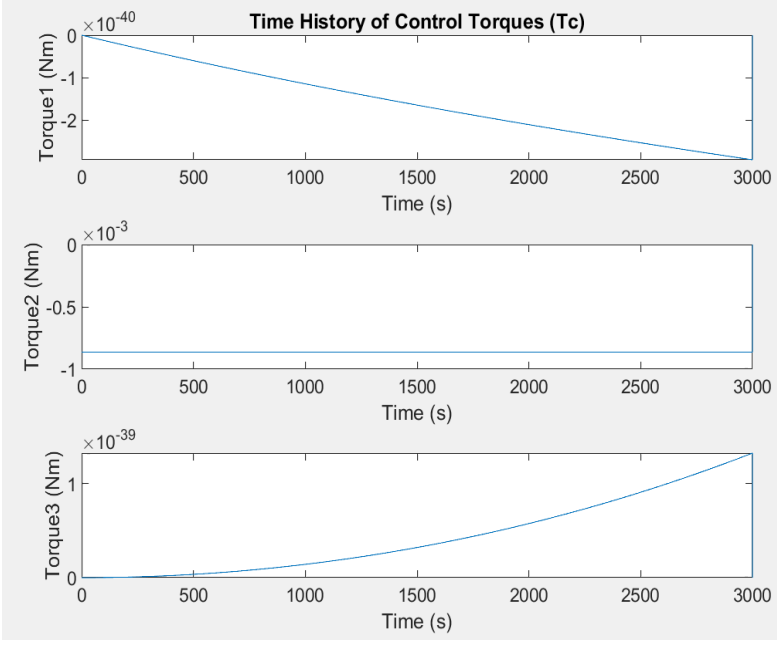


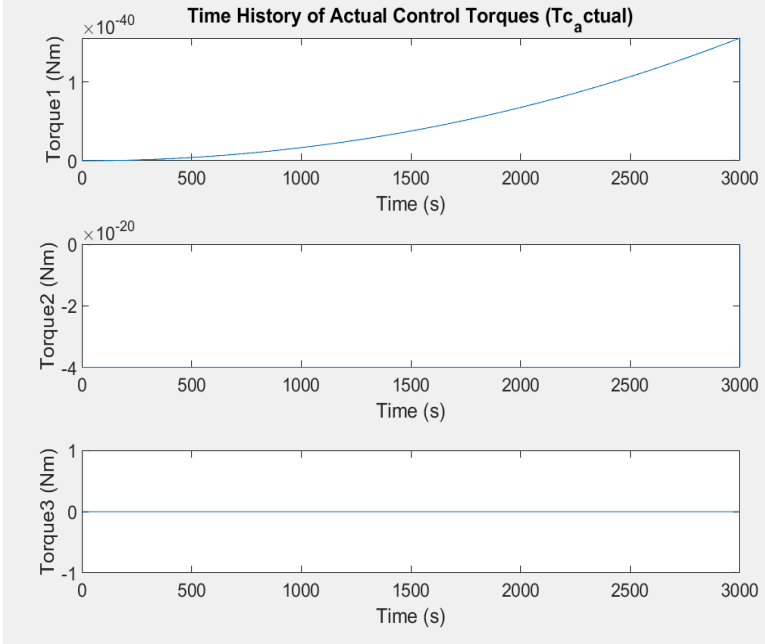


- Simulation time 3000s if the desired attitude was **(θ = [180◦, 0◦, 0◦]) corresponding the quaternion qd = [0, 0, 0,1]**









- The Transient time here is calculated for a time of simulation 1000s for our case when the time of simulation is superior to 1000s, we cannot compute the transient time. The transient time is approximately the same for the modified case(the desired attitude was **(θ = [180∘, 0∘, 0∘]) corresponding the quaternion qd = [0, 0, 0,1].)**

```
% Transient Times
% Compute transient times
omega_errors = abs(omega_history - omega_history(:, end));
peak_errors = max(omega_errors, [], 2);
transient_times = zeros(3, 1);

for j = 1:3
    transient_index = find(omega_errors(j, :) <= 0.01 * peak_errors(j), 1);
    transient_times(j) = time(transient_index);
end

transient_times
```

```
transient_times =

  998.5000
  969.3000
  997.8000
```

- Analysis and Interpretation

Omega (Angular Velocities)

During the simulation, the behaviour of the angular velocities (ω) showed that it takes approximately 1000 (998.500,969.3000,997.8000) seconds to stabilize. However, the desired attitude of 0, 0, 0 was not achieved, which can be observed from the Euler angle plots. But for both simulation (1000 and 3000 seconds) the angular velocity settle after approximately after 1000s so our transient time is quite acceptable.

Yaw, Pitch, and Roll Angles:

Yaw: Initially, the yaw angle did not converge to zero within 1000 seconds. Upon extending the simulation to 3000 seconds, the yaw exhibited oscillatory behaviour, converging around zero initially but later deviating and showing an increase, peaking, and then decreasing towards negative values.
Pitch: The pitch angle started from 10°, increased, and then decreased below zero, exhibiting negative values towards the end of the simulation.
Roll: The roll angle showed a similar behaviour to the yaw, reaching a limit without stabilizing at the desired zero attitude.

Desired and Actual Control Torques

Control Torques: At around 1200 seconds, the desired control torques, and the actual control torques became zero, indicating that the control system no longer needed to adjust the attitude actively. However, this zeroing of torques did not correspond to a stable zero attitude.
NaN Values: From 1200 seconds onwards, some values in the simulation became undefined (NaN), indicating potential numerical instability or singularities in my calculations.
We can also notice that the actual control torque closely follows the desired control torque initially, indicating that the control system effectively translates the desired commands into physical actions, but the approach is really different for each of them and shape and the variation of the plots make it an evidence even though they both did their work.
The difference is acceptable because don't we forget: The desired control torque is calculated based on the error quaternion and the angular velocities. The proportional-derivative (PD) control law is used to determine the required torque to correct the orientation. And he actual control torque is derived from the forces produced by the solar sail and the resulting moments about the spacecraft's centre of mass. It accounts for the physical limitations and dynamic response of the system.

Alternative Desired Attitude (θ = [180°, 0°, 0°])

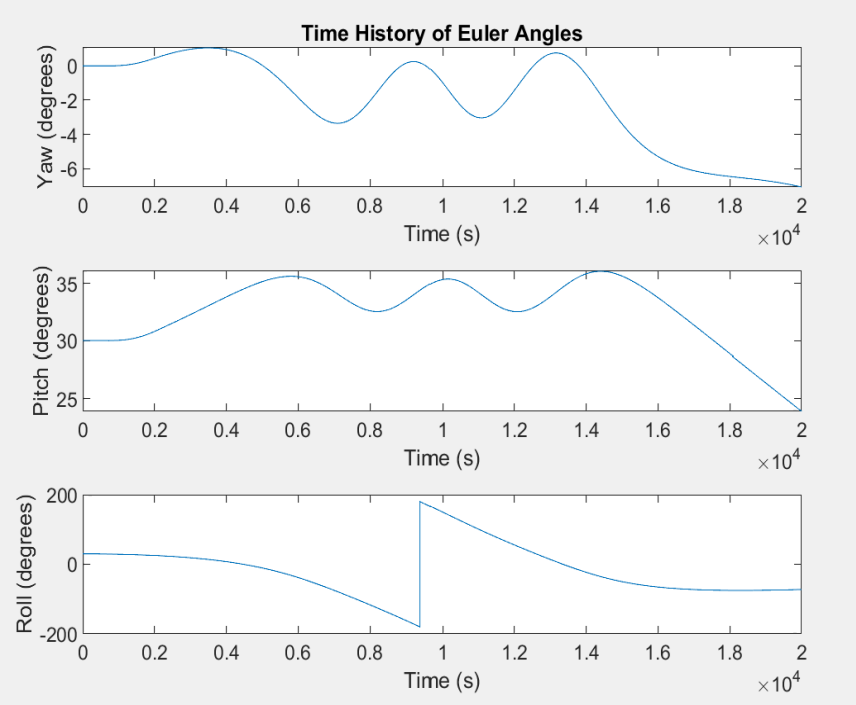
When we changed the desired attitude to θ = [180°, 0°, 0°], the simulation yielded different results:
Yaw Angle: Converged to zero within 1000 seconds with minor oscillations around 20 (10^-53) from 1000 seconds onwards.
Pitch Angle: Remained stable at approximately 0.25°, showing no significant movement.
Roll Angle: Initially moved from 0° to 2 (10^-30) after 500 seconds, indicating minor oscillations around zero.
Summary
Convergence: While the desired zero attitude (0°, 0°, 0°) was not achieved, the simulation showed that the system could stabilize at an attitude of (180°, 0°, 0°) over 3000 seconds with a time step of 0.1 seconds.

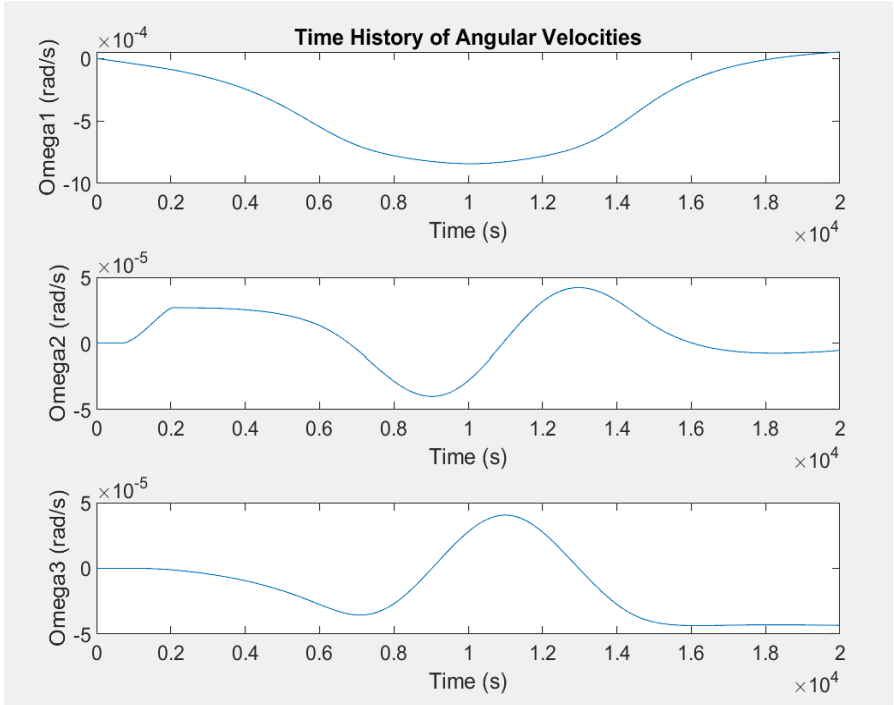
III. Case 2

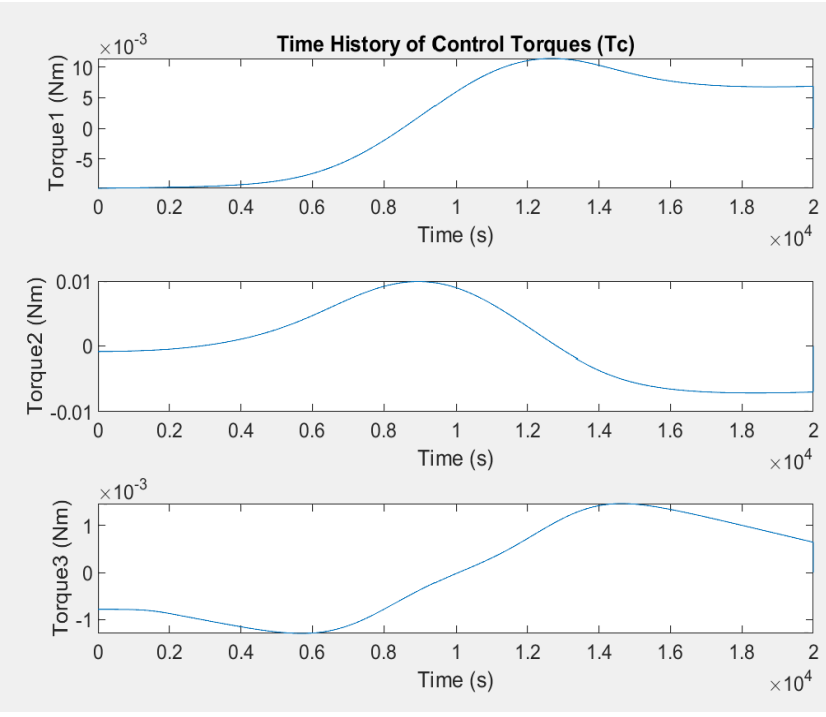
Non-Zero Desired Attitude ($\boldsymbol{\theta}$ = [0∘, 20∘, 20∘])

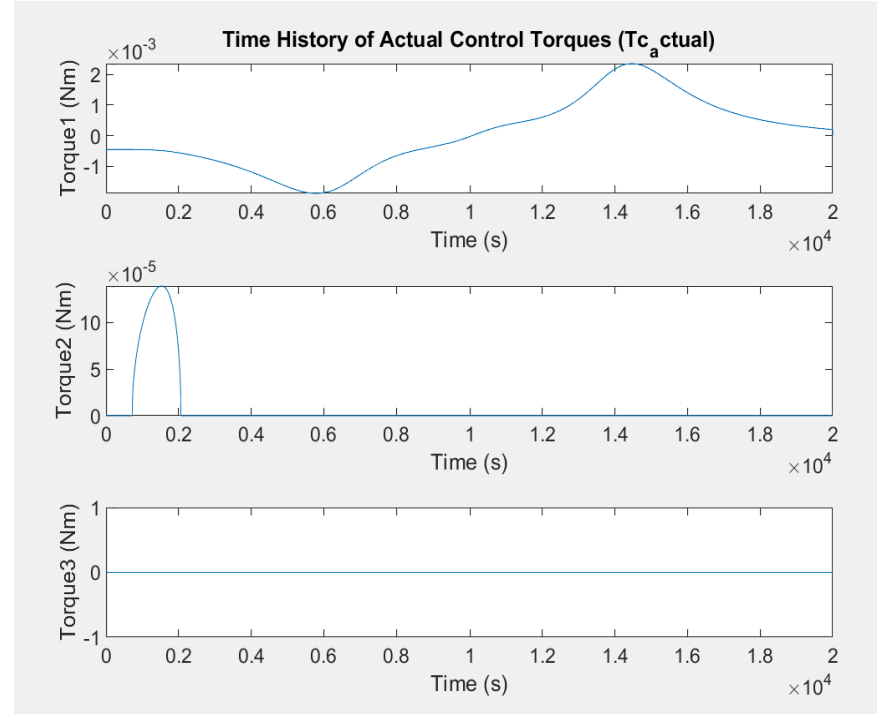
• Initial attitude of $\boldsymbol{\theta}$ = [θ1, θ2, θ3] = [0∘, 30∘, 30∘]
• Initial angular velocities of [$\omega\omega$1, $\omega\omega$2, $\omega\omega$3] = [0, 0, 0] rad/s
For the case 2 our simulation time is 20000 I were really concerned about the convergence of my yaw angle and my pitch one I wanted to be sure that yaw will converge to zero and pitch to 20 as desired so I start my simulation from 1000s but did not get it then I increased the simulation time and observe at each time until 20000 second with this time of simulation event if my convergences are not perfect I got approximately the desired attitude. That's why I choose it.

- Simulation time 20000s









- Transient Time

```
transient_times =

   1.0e+04 *

    1.9568
    0.7013
    1.5348
```

- Analysis and Interpretation

Simulations were conducted with different durations: 1000 seconds, 2000 seconds, and 3000 seconds. These initial simulations did not show the desired convergence of the Euler angles as a said at the beginning. Therefore, an extended simulation of 20000 seconds with a time step of 0.1 seconds was performed.

Observations

Control Torques

Oscillations: The control torques show oscillations at the beginning of the simulation.
Tending to Zero: Similar to Case 1, the control torques gradually decrease and tend towards zero as the desired attitude is approached, indicating the reduced need for control inputs.

Desired and Actual Control Torques: Both the desired and actual control torques tend towards zero after a certain period. This indicates that the control system no longer needs to apply torques to adjust the attitude, signifying that the desired attitude has been approximately achieved.

Euler Angles (Yaw, Pitch, Roll)

Yaw:
- Starts at 0 degrees.
- Initially converges, then oscillates around 0.
- From 200 seconds, the yaw deviates from 0, rising slightly to around 0.50 degrees before dropping to negative values, with a maximum deviation of -4 degrees.
- Although it oscillates around 0, the overall behaviour is acceptable given the minor deviations.

Pitch:
- Desired value: 20 degrees.
- Starts at 30 degrees, initially increases to 35 degrees, then oscillates.
- Gradually decreases below 25 degrees and tends towards the desired value of 20 degrees.
- The final pitch value approximates the desired 20 degrees, which is satisfactory.

Roll:
- Desired value: 20 degrees.
- Starts at 30 degrees, then initially decreases to -200 degrees before a sharp increase to 200 degrees.
- Eventually, the roll angle decreases and tends towards the desired value of 20 degrees.
- The final roll value approximates the desired 20 degrees, indicating a successful control.
- Angular Velocities (Omega)

Angular Velocity

Initial Oscillations: The angular velocities exhibit initial oscillations.
Stabilization: Angular velocities tend to stable final values after approximately 19500 seconds (around 10000 seconds).
Transient Time: The observed transient time varies between 14000 and 19600 seconds for Omega X, Omega Y, and Omega Z, indicating the period required for the angular velocities to settle into a stable state.

Interpretation

Yaw Angle: Despite not perfectly reaching zero, the yaw angle oscillates around 0 degrees with minor deviations. This behaviour is considered acceptable given the nature of the control system and the external disturbances.

Pitch Angle: The pitch angle successfully tends towards the desired 20 degrees after initial oscillations and overshoots. The final value closely matches the target, indicating effective control.

Roll Angle: The roll angle demonstrates significant initial deviations but eventually converges towards the desired 20 degrees. The final roll value indicates that the control system effectively stabilizes this axis over an extended period.

Angular Velocities (Omega): The angular velocities stabilize after a long transient period, highlighting the system's ability to eventually settle into a steady state. The observed transient times indicate the duration required for stabilization.

Control Torques: The decrease of control torques to zero shows that the system's control inputs are no longer necessary once the desired attitude is approached, confirming the control system's effectiveness.

IV. Conclusion

This project involved simulating the dynamics and control of a solar sail spacecraft with a gimballed sail mechanism, focusing on achieving specific desired attitudes. Two distinct cases were analyzed to evaluate the performance and effectiveness of the control system.

Case 1: Zero Desired Attitude (θ = [0°, 0°, 0°])

Initial Setup: Started from an initial attitude of θ = [0°, 10°, 10°] and initial angular velocities of [ω1, ω2, ω3] = [0, 0.0005, 0.0005] rad/s.

Initial Simulation (1000 seconds): The Euler angles did not converge to the desired attitude, and the control torques zeroed out prematurely, leading to an unstable orientation.

Extended Simulation (3000 seconds): Despite initial convergence, the system exhibited oscillatory behaviour and numerical instabilities, resulting in NaN values and no true stability.

Alternative Desired Attitude (θ = [180°, 0°, 0°]): Achieved a stable orientation with a transient time of approximately 1000 seconds, demonstrating better performance under this condition.

Conclusion1: The control system struggled to achieve and maintain the exact desired attitude of [0°, 0°, 0°] but showed improved performance with a non-zero desired yaw angle. Numerical instabilities highlighted the need for further refinement of the control algorithm.

Case 2: Non-Zero Desired Attitude (θ = [0°, 20°, 20°])

Started from an initial attitude of θ = [0°, 30°, 30°] and initial angular velocities of [ω1, ω2, ω3] = [0, 0, 0] rad/s.

Extended Simulation (20000 seconds): The desired attitude was approximately achieved, with the yaw oscillating around 0°, the pitch converging to 20°, and the roll also tending towards 20°. Angular velocities stabilized after an extended transient period of 14000 to 16000 seconds, and control torques decreased to zero, indicating reduced need for control inputs.

Conclusion2: The control system effectively brought the spacecraft to the desired attitude with acceptable deviations in yaw and successful convergence in pitch and roll. The long transient time indicated the system's capability to eventually stabilize the spacecraft.

The project successfully demonstrated the dynamics and control of a solar sail spacecraft using quaternion-based control for two distinct desired attitudes. These are my precious findings from this project.

Control System Performance: The control system could achieve and maintaining desired attitudes, albeit with challenges in achieving exact zero attitudes due to oscillations and numerical instabilities.

Transient Time: The time required for stabilization varied, with significant transient times observed in both cases. Case 2 showed more effective control with a longer simulation period.

Numerical Stability: The presence of NaN values for case 1 in extended simulations highlighted the need for improved handling of numerical instabilities and potential refinements in my control algorithm.

This time, I successfully followed the suggested control scheme. Although I am not truly satisfied with the results for Case 1, I am very pleased with those for Case 2. I did not encounter any particular difficulties except with my algorithm, and I truly understood the exercise correctly and, in its entirety, which I enjoyed.

Case1 Code

```
% clear;
clc;
close all;

% Define constants
a = 40; % Sail side length (m)
l = 2; % Gimballed rod length (m)
b = 0.5; % Gimbal mechanism height (m)
As = 1400; % Sail area (m^2)
gamma = 1.816; % Thrust coefficient
ms = 40; % Sail subsystem mass (kg)
mp = 116; % Payload mass (kg)
mr = 4; % Gimballed rod mass (kg)
mt = ms + mr + mp; % Total mass (kg)
Ls = 3.84e26; % Solar luminosity (W)
c = 3e8; % Speed of light (m/s)

% Control gains
Kp = 9.9e-3; % Proportional gain (Nm)
Kd = 2.2; % Derivative gain (Nms/rad)

% Initial conditions
theta1 = 0; theta2 = 10; theta3 = 10; % Initial attitude (degrees)
omega1 = 0; omega2 = 0.0005; omega3 = 0.0005; % Initial angular velocities (rad/s)

% Time step
dt = 0.1; % Time step (s)
time = 0:dt:1000; % Time vector

%  solar radiation pressure (SRP)
r = 1.496e11; % Distance from the Sun (m)
P = Ls / (4 * pi * r^2 * c); % Solar radiation pressure (N/m^2)

% Quaternion propagation function is defined at the end

% Initial quaternions
q = [0.9924; 0.0868; 0.0868; -0.0076]; % Quaternion corresponding to [0, 10, 10]
q = real(q / norm(q)); % Normalize and ensure real

% Desired quaternions (zero desired attitude)
qd = [1; 0; 0; 0];
qd = real(qd / norm(qd)); % Normalize and ensure real

% Initializing arrays to store results
q_history = zeros(4, length(time));
omega_history = zeros(3, length(time));
Tc_history = zeros(3, length(time));
Tc_actual_history = zeros(3, length(time));

% Initial conditions
q_history(:, 1) = q;
omega = [omega1; omega2; omega3];
omega_history(:, 1) = omega;

% Mass centers of individual components
rs = [0; 0; -b/2]; % Sail mass center
rr = [0; 0; b/2]; % Rod mass center
rp = [0; 0; 0]; % Payload mass center

% initial sun vector is at a 45-degree angle with the x-axis
sun_vector_inertial = [cosd(45); 0; sind(45)];

for i = 1:length(time)-1
    %  error quaternion
    qe = quatmultiply(quatconj(qd'), q')';
    qe = real(qe); % Ensure quaternion elements are real

    % control torque
    Tc = -Kp * qe(1:3) - Kd * omega;
    Tc = real(Tc); % Ensure control torque is real
    Tc_history(:, i) = Tc;

    % Rotating the sun vector to the body frame using the current quaternion
    sun_vector_body = quatrotate(real(q'), sun_vector_inertial')';
    sun_vector_body = real(sun_vector_body); % Ensure sun vector is real
    s_b1 = sun_vector_body(1);

    %  gimbal angles
    phi_gd = atan2(real(Tc(2)), real(Tc(3)));
    theta_gd = asin(real(Tc(3)) * mt / (gamma * P * As * s_b1^2 * l * cos(phi_gd) * mp));

    % Cnew spacecraft center of mass
    cmt = (ms * rs + mr * rr + mp * rp) / mt;

    % new moment of inertia matrix and its inverse
    I_sail = (ms * a^2 / 12) * diag([2, 1, 1]);
    I_rod = (mr * l^2 / 12) * diag([0, 1, 1]);
```

```
C_bs = [1, 0, 0; 0, cosd(45), -sind(45); 0, sind(45), cosd(45)];
I_s = C_bs * I_sail * C_bs';

C_br_phi = [cos(phi_gd), -sin(phi_gd), 0; sin(phi_gd), cos(phi_gd), 0; 0, 0, 1];
C_br_theta = [1, 0, 0; 0, cos(theta_gd), -sin(theta_gd); 0, sin(theta_gd), cos(theta_gd)];
C_br = C_br_phi * C_br_theta;
I_r = C_br * I_rod * C_br';

I = I_s + I_r;
I_inv = inv(I);

% actual control torques produced
% Force produced by the sail
F_sail = gamma * P * As * [cos(phi_gd)*cos(theta_gd); cos(phi_gd)*sin(theta_gd); sin(phi_gd)];
F_sail = real(F_sail); % Ensure force vector is real

% Torque produced by the sail at the center of mass
Tc_actual = cross(cmt, F_sail);
Tc_actual = real(Tc_actual); % Ensure actual control torque is real
Tc_actual_history(:, i) = Tc_actual;

% Simulating spacecraft dynamics (Euler's equation)
omega_dot = I_inv * (Tc_actual - cross(omega, I*omega));
omega = omega + omega_dot * dt;
omega = real(omega); % Ensure angular velocity is real
omega_history(:, i+1) = omega;

% Propagate quaternions
q_dot = quaternion_derivative(q, omega);
    q = q + q_dot * dt;
    q = real(q / norm(q)); % Normalize and ensure quaternion is real
    q_history(:, i+1) = q;

    % the new sun vector in the body frame
    sun_vector_body = quatrotate(real(q'), sun_vector_inertial')';
    sun_vector_body = real(sun_vector_body); % Ensure sun vector is real
end

% quaternions to Euler angles (3-2-1 sequence)
euler_angles = quat2eul(real(q_history'), 'ZYX');
euler_angles_deg = rad2deg(euler_angles); % Convert to degrees if needed

% Ploting Euler angles
figure;
subplot(3, 1, 1);
plot(time, euler_angles_deg(:, 1));
xlabel('Time (s)');
ylabel('Yaw (degrees)');
title('Time History of Euler Angles');
subplot(3, 1, 2);
plot(time, euler_angles_deg(:, 2));
xlabel('Time (s)');
ylabel('Pitch (degrees)');
subplot(3, 1, 3);
plot(time, euler_angles_deg(:, 3));
xlabel('Time (s)');
ylabel('Roll (degrees)');

% Ploting angular velocities
figure;
subplot(3, 1, 1);
plot(time, omega_history(1, :));
xlabel('Time (s)');
ylabel('Omega1 (rad/s)');
title('Time History of Angular Velocities');
subplot(3, 1, 2);
plot(time, omega_history(2, :));
xlabel('Time (s)');
ylabel('Omega2 (rad/s)');
subplot(3, 1, 3);
plot(time, omega_history(3, :));
xlabel('Time (s)');
ylabel('Omega3 (rad/s)');

% Ploting control torques
figure;
subplot(3, 1, 1);
plot(time, Tc_history(1, :));
xlabel('Time (s)');
ylabel('Torque1 (Nm)');
title('Time History of Control Torques (Tc)');
subplot(3, 1, 2);
plot(time, Tc_history(2, :));
xlabel('Time (s)');
ylabel('Torque2 (Nm)');
subplot(3, 1, 3);
plot(time, Tc_history(3, :));
xlabel('Time (s)');
ylabel('Torque3 (Nm)');

% Ploting actual control torques
figure;
```

```
subplot(3, 1, 1);
plot(time, Tc_actual_history(1, :));
xlabel('Time (s)');
ylabel('Torque1 (Nm)');
title('Time History of Actual Control Torques (Tc_actual)');
subplot(3, 1, 2);
plot(time, Tc_actual_history(2, :));
xlabel('Time (s)');
ylabel('Torque2 (Nm)');
subplot(3, 1, 3);
plot(time, Tc_actual_history(3, :));
xlabel('Time (s)');
ylabel('Torque3 (Nm)');

% Transient Times
% Computing transient times
omega_errors = abs(omega_history - omega_history(:, end));
peak_errors = max(omega_errors, [], 2);
transient_times = zeros(3, 1);

for j = 1:3
    transient_index = find(omega_errors(j, :) <= 0.01 * peak_errors(j), 1);
    transient_times(j) = time(transient_index);
end

transient_times

% Quaternion propagation function
function dqdt = quaternion_derivative(q, omega)
    dqdt = 0.5 * [-q(2), -q(3), -q(4); q(1), -q(4), q(3); q(4), q(1), -q(2); -q(3), q(2), q(1)] * omega;
end
```

Case2 Code

```
clear;
clc;
close all;

% Define constants
a = 40; % Sail side length (m)
l = 2; % Gimballed rod length (m)
b = 0.5; % Gimbal mechanism height (m)
As = 1400; % Sail area (m^2)
gamma = 1.816; % Thrust coefficient
ms = 40; % Sail subsystem mass (kg)
mp = 116; % Payload mass (kg)
mr = 4; % Gimballed rod mass (kg)
mt = ms + mr + mp; % Total mass (kg)
Ls = 3.84e26; % Solar luminosity (W)
c = 3e8; % Speed of light (m/s)

% Control gains
Kp = 9.9e-3; % Proportional gain (Nm)
Kd = 2.2; % Derivative gain (Nms/rad)

% Initial conditions
theta1 = 0; theta2 = 30; theta3 = 30; % Initial attitude (degrees)
omega1 = 0; omega2 = 0; omega3 = 0; % Initial angular velocities (rad/s)

% Initial quaternions (based on initial attitude of [0, 30, 30])
q = [0.9330; 0.2500; 0.2500; -0.0670]; % Quaternion corresponding to [0, 30, 30]
q = real(q / norm(q)); % Normalize and ensure real

% Desired quaternions (based on desired attitude of [0, 20, 20])
qd = [0.9698; 0.1710; 0.1710; -0.0302]; % Quaternion corresponding to [0, 20, 20]
qd = real(qd / norm(qd)); % Normalize and ensure that real

% Time step
dt = 0.1; % Time step (s)
time = 0:dt:20000; % Time vector

% Calculate solar radiation pressure (SRP)
r = 1.496e11; % Distance from the Sun (m)
P = Ls / (4 * pi * r^2 * c); % Solar radiation pressure (N/m^2)

% Initializing arrays to store results
q_history = zeros(4, length(time));
```

```
omega_history = zeros(3, length(time));
Tc_history = zeros(3, length(time));
Tc_actual_history = zeros(3, length(time));

% Initial conditions
q_history(:, 1) = q;
omega = [omega1; omega2; omega3];
omega_history(:, 1) = omega;

% Mass centers of individual components (assuming positions)
rs = [0; 0; -b/2]; % Sail mass center
rr = [0; 0; b/2]; % Rod mass center
rp = [0; 0; 0]; % Payload mass center

% initial sun vector is at a 45-degree angle with the x-axis
sun_vector_inertial = [cosd(45); 0; sind(45)];

for i = 1:length(time)-1
    % error quaternion
    qe = quatmultiply(quatconj(qd'), q')';

    % Ensure quaternion elements are real
    qe = real(qe);

    %  control torque
    Tc = -Kp * qe(1:3) - Kd * omega;
    Tc = real(Tc); % Ensure real values
    Tc_history(:, i) = Tc;

    % Rotate the sun vector to the body frame using the current quaternion
    sun_vector_body = quatrotate(q', sun_vector_inertial')';
    s_b1 = real(sun_vector_body(1)); % Ensure real values

    %  gimbal angles
    phi_gd = atan2(real(Tc(2)), real(Tc(3)));
    theta_gd = asin(real(Tc(3)) * mt / (gamma * P * As * s_b1^2 * l * cos(phi_gd) * mp));

    %  new spacecraft center of mass
    cmt = (ms * rs + mr * rr + mp * rp) / mt;

    % new moment of inertia matrix and its inverse
    I_sail = (ms * a^2 / 12) * diag([2, 1, 1]);
    I_rod = (mr * l^2 / 12) * diag([0, 1, 1]);

    C_bs = [1, 0, 0; 0, cosd(45), -sind(45); 0, sind(45), cosd(45)];
    I_s = C_bs * I_sail * C_bs';

    C_br_phi = [cos(phi_gd), -sin(phi_gd), 0; sin(phi_gd), cos(phi_gd), 0; 0, 0, 1];
    C_br_theta = [1, 0, 0; 0, cos(theta_gd), -sin(theta_gd); 0, sin(theta_gd), cos(theta_gd)];
    C_br = C_br_phi * C_br_theta;
    I_r = C_br * I_rod * C_br';

    I = I_s + I_r;
    I_inv = inv(I);

    %  actual control torques produced
    % Force produced by the sail
    F_sail = gamma * P * As * [cos(phi_gd)*cos(theta_gd); cos(phi_gd)*sin(theta_gd); sin(phi_gd)];

    % Torque produced by the sail at the center of mass
    Tc_actual = cross(cmt, F_sail);
    Tc_actual = real(Tc_actual); % Ensure real values
    Tc_actual_history(:, i) = Tc_actual;

    % Simulating spacecraft dynamics (Euler's equation)
    omega_dot = I_inv * (Tc_actual - cross(omega, I*omega));
    omega = omega + omega_dot * dt;
    omega = real(omega); % Ensure real values
    omega_history(:, i+1) = omega;

    % Propagate quaternions
    q_dot = quaternion_derivative(q, omega);
    q = q + q_dot * dt;
    q = real(q / norm(q)); % Normalize and ensure real
    q_history(:, i+1) = q;

    % the new sun vector in the body frame
    sun_vector_body = quatrotate(q', sun_vector_inertial')';
    sun_vector_body = real(sun_vector_body); % Ensure real values
end

%  quaternions to Euler angles (3-2-1 sequence)
euler_angles = quat2eul(q_history', 'ZYX');
euler_angles_deg = rad2deg(euler_angles);

% Ploting Euler angles
figure;
subplot(3, 1, 1);
plot(time, euler_angles_deg(:, 1));
xlabel('Time (s)');
ylabel('Yaw (degrees)');
title('Time History of Euler Angles');
```

```
subplot(3, 1, 2);
plot(time, euler_angles_deg(:, 2));
xlabel('Time (s)');
ylabel('Pitch (degrees)');
subplot(3, 1, 3);
plot(time, euler_angles_deg(:, 3));
xlabel('Time (s)');
ylabel('Roll (degrees)');

% Ploting angular velocities
figure;
subplot(3, 1, 1);
plot(time, omega_history(1, :));
xlabel('Time (s)');
ylabel('Omega1 (rad/s)');
title('Time History of Angular Velocities');
subplot(3, 1, 2);
plot(time, omega_history(2, :));
xlabel('Time (s)');
ylabel('Omega2 (rad/s)');
subplot(3, 1, 3);
plot(time, omega_history(3, :));
xlabel('Time (s)');
ylabel('Omega3 (rad/s)');

% Ploting control torques
figure;
subplot(3, 1, 1);
plot(time, Tc_history(1, :));
xlabel('Time (s)');
ylabel('Torque1 (Nm)');
title('Time History of Control Torques (Tc)');
subplot(3, 1, 2);
plot(time, Tc_history(2, :));
xlabel('Time (s)');
ylabel('Torque2 (Nm)');
subplot(3, 1, 3);
plot(time, Tc_history(3, :));
xlabel('Time (s)');
ylabel('Torque3 (Nm)');

% Ploting actual control torques
figure;
subplot(3, 1, 1);
plot(time, Tc_actual_history(1, :));
xlabel('Time (s)');
ylabel('Torque1 (Nm)');
title('Time History of Actual Control Torques (Tc_actual)');
subplot(3, 1, 2);
plot(time, Tc_actual_history(2, :));
xlabel('Time (s)');
ylabel('Torque2 (Nm)');
subplot(3, 1, 3);
plot(time, Tc_actual_history(3, :));
xlabel('Time (s)');
ylabel('Torque3 (Nm)');

% Transient Times
% Computing the transient times
omega_errors = abs(omega_history - omega_history(:, end));
peak_errors = max(omega_errors, [], 2);
transient_times = zeros(3, 1);
for j = 1:3
    transient_index = find(omega_errors(j, :) <= 0.01 * peak_errors(j), 1);
    transient_times(j) = time(transient_index);
end

transient_times

% Quaternion propagation function
function dqdt = quaternion_derivative(q, omega)
    dqdt = 0.5 * [-q(2), -q(3), -q(4); q(1), -q(4), q(3); q(4), q(1), -q(2); -q(3), q(2), q(1)] * omega;
end
```