
Table of Contents

Project 2 - Team 4	1
Question 3.1	1
Question 3.2	3
Question 4	6
Question 5	6
Question 5.3	13
Functions	20

Project 2 - Team 4

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

Question 3.1

```
Iw = 0.1; %kg*m^2
initial = [5;0;0;0;0;0]./60.*2.*pi; % initial tumbling angular velocity
(rad/s)
opts = odeset('MaxStep',2.0); % Sets max time step for ode45
ts = 5000; % wheel spin-up time (sec)
hw = 55; % nominal angular momentum (kg*m^2/s)
Iww = [0,0,0;0,0,0;0,0,Iw]; % inertia matrix for wheels (kg*m^2)
J = [500,0,0;0,400,-7;0,-7,440]; % inertia matrix for full satellite system
(kg*m^2)

% Calculate the satellite behavior during the wheel spin up
[t,y] = ode45(@angular_rates(t,w,J,hw,Iw,ts),[0,5000],initial,opts);

% Save data in a vector
y2 = y;

% Calculate the nutation angle at each time step
nutation_angle = nutation(J,Iww,y,[0,0,1]);

% Plot angular velocities of the satellite and the nutation angle
figure(1);
tcl = tiledlayout(4,1);
title(tcl,'Time Histories of Angular Velocities and Nutation Angle')
nexttile(tcl)
hold on;
plot(t,y(:,1));
ylabel('W1 (Rad/s)', 'Interpreter','latex','FontWeight','Bold')
nexttile(tcl)
hold on;
plot(t,y(:,2));
ylabel('W2 (Rad/s)', 'Interpreter','latex','FontWeight','Bold')
nexttile(tcl)
hold on;
plot(t,y(:,3));
```

```

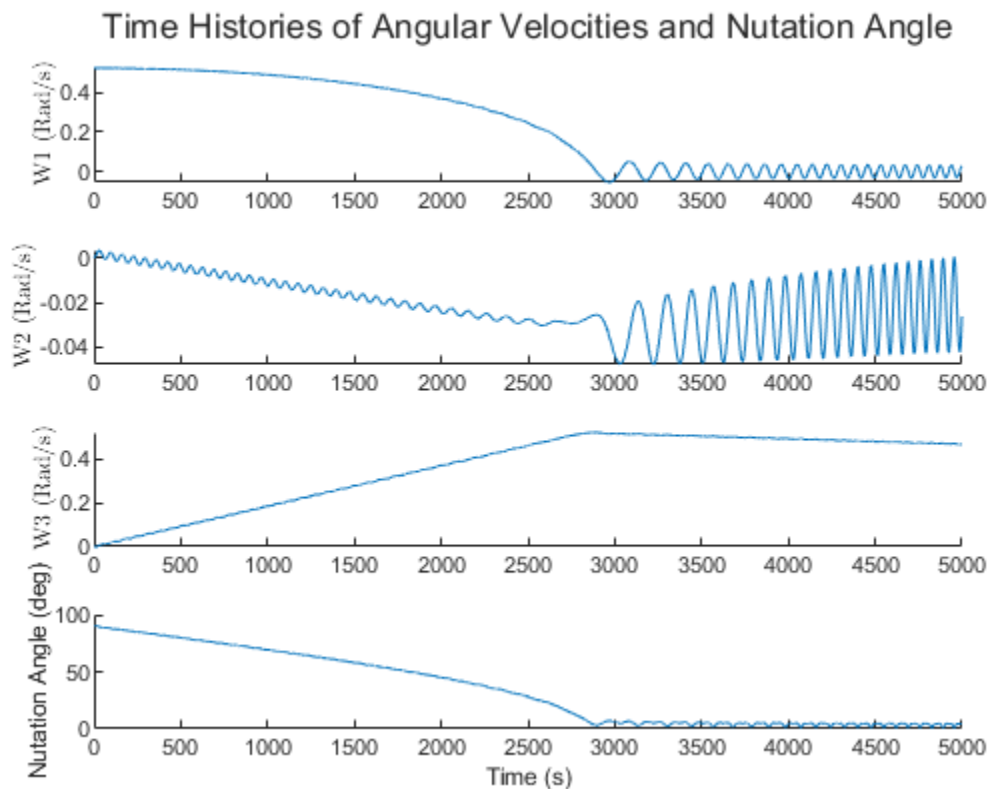
ylabel('W3 (Rad/s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(t, nutation_angle);
ylabel('Nutation Angle (deg)')
xlabel('Time (s)')

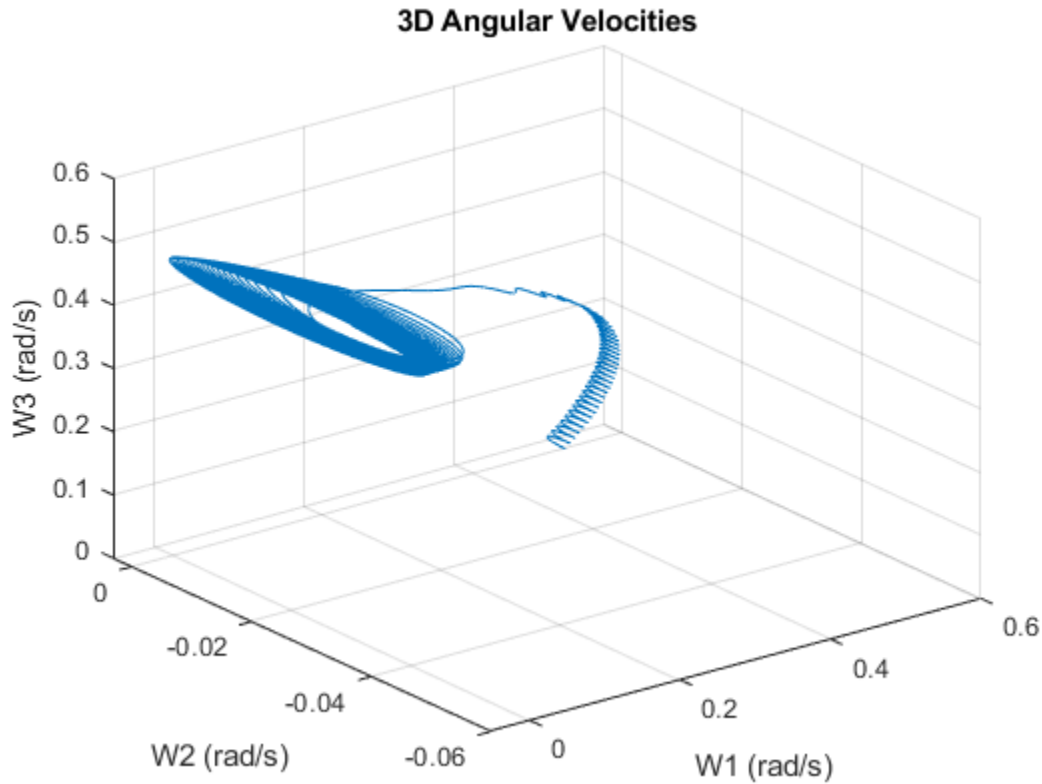
% Plot 3D angular velocities
figure(2)
plot3(y(:,1), y(:,2), y(:,3));
title('3D Angular Velocities')
xlabel('W1 (rad/s)'); ylabel('W2 (rad/s)'); zlabel('W3 (rad/s)')
grid on;

% Check initial and final angular momentum to ensure conservation
wi = [5;0;0]./60.*2.*pi;
wf = [y(end,1); y(end,2); y(end,3)];
wwf = [y(end,4); y(end,5); y(end,6)];
H_i = norm(J*wi);
H_f = norm(J*wf+ Iww*wwf);
if abs(H_i-H_f) < 1e-3
    disp('3.1 passes angular momentum conservation check')
else
    disp('3.1 fails angular momentum conservation check')
end

```

3.1 passes angular momentum conservation check





Question 3.2

```
Iw = 0.1; %kg*m^2
initial = [5;0;0;0;0;0]./60.*2.*pi; % initial tumbling angular velocity
(rad/s)
opts = odeset('MaxStep',2.0); % Sets max time step for ode45
ts = 5000; % wheel spin-up time (sec)
hw = 55; % nominal angular momentum (kg*m^2/s)
Iww = [0,0,0;0,0,0;0,0,Iw]; % inertia matrix for wheels (kg*m^2)
J = [500,0,0;0,400,0;0,0,440]; % inertia matrix for full satellite system
(kg*m^2)

% Calculate the satellite behavior during the wheel spin up
[t,y] = ode45(@(t,w) angular_rates(t,w,J,hw,Iw,ts), [0,5000],initial,opts);

% Calculate the nutation angle at each time step
nutation_angle = nutation(J,Iww,y,[0,0,1]);

% Plot angular velocities of the satellite and the nutation angle
figure(3);
tcl = tiledlayout(4,1);
title(tcl, 'Time Histories of Angular Velocities and Nutation Angle')
nexttile(tcl)
hold on;
plot(t,y(:,1));
```

```

ylabel('W1 (Rad/s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(t, y(:, 2));
ylabel('W2 (Rad/s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(t, y(:, 3));
ylabel('W3 (Rad/s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(t, nutation_angle);
ylabel('Nutation Angle (deg)')
xlabel('Time (s)')

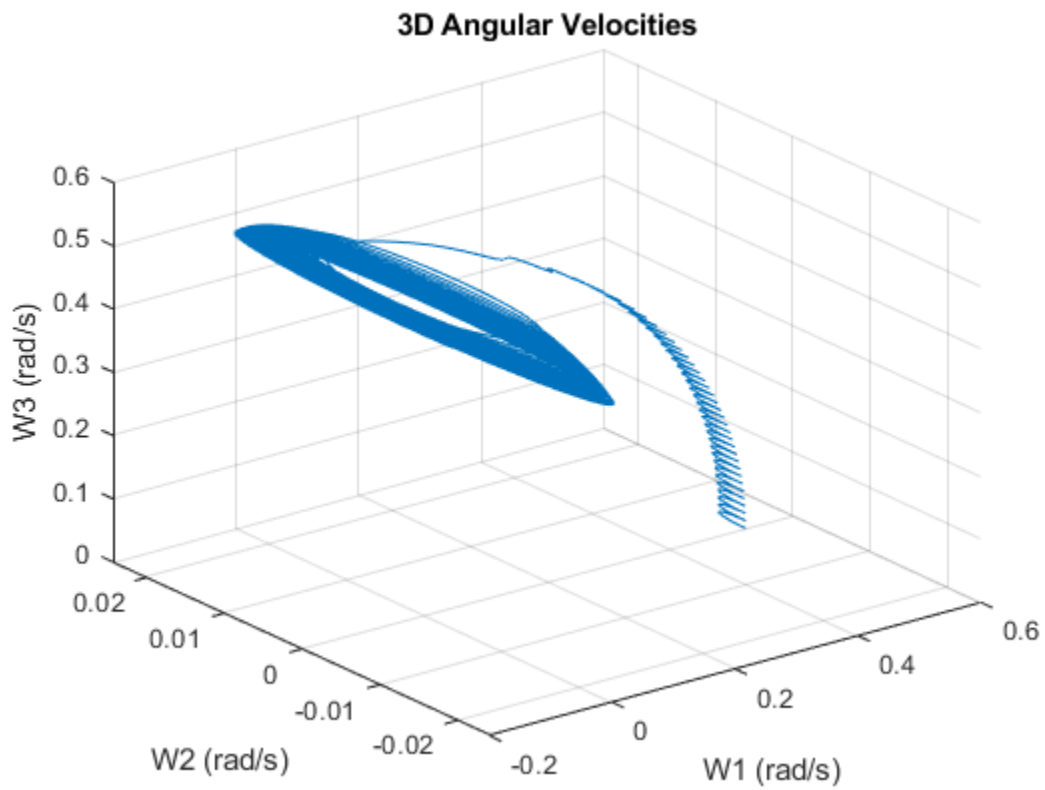
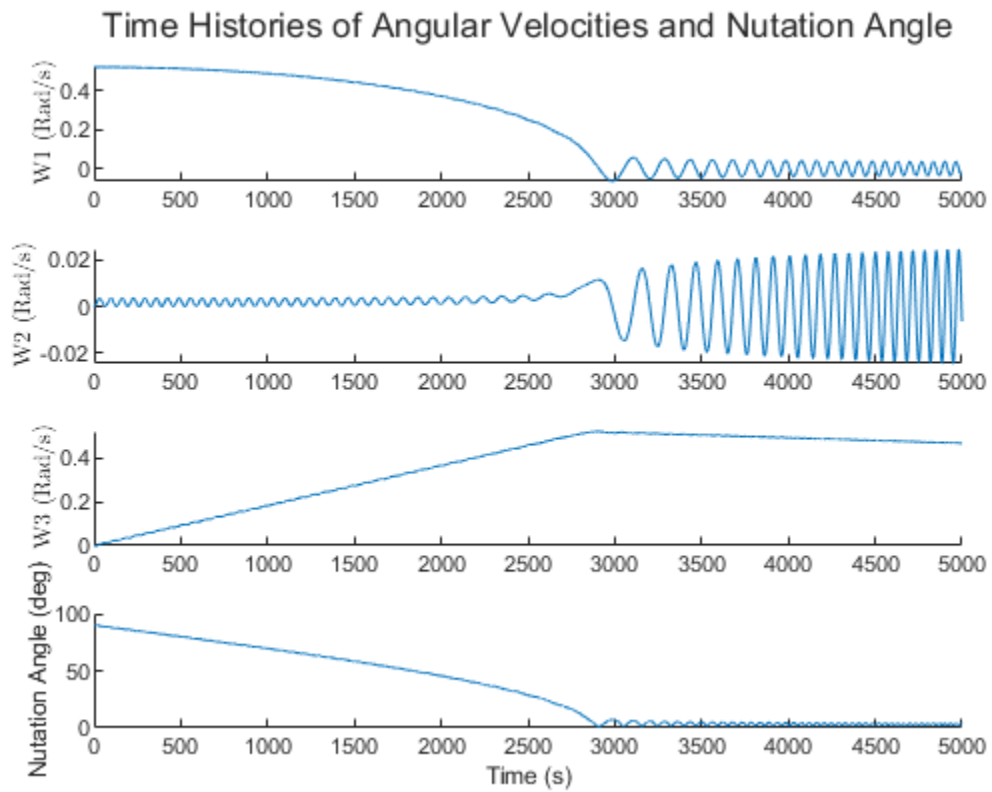
% Plot 3D angular velocities
figure(4)
plot3(y(:, 1), y(:, 2), y(:, 3));
title('3D Angular Velocities')
xlabel('W1 (rad/s)'); ylabel('W2 (rad/s)'); zlabel('W3 (rad/s)')
grid on;

% Check initial and final angular momentum to ensure conservation
wi = [5; 0; 0] ./ 60 .* 2 .* pi;
wf = [y(end, 1); y(end, 2); y(end, 3)];
wwf = [y(end, 4); y(end, 5); y(end, 6)];
H_i = norm(J * wi);
H_f = norm(J * wf + Iww * wwf);

if abs(H_i - H_f) < 1e-3
    disp('3.2 passes angular momentum conservation check')
else
    disp('3.2 fails angular momentum conservation check')
end

3.2 passes angular momentum conservation check

```



Question 4

```
% Initialize values for calculation
Bv1 = [-0.3;-0.1;0.9]; % star tracker observation vector
Bv2 = [0.8;-0.5;0.2]; % sun sensor observation vector
Nv1 = [0;0;1]; % star tracker reference vector
Nv2 = [1;0;0]; % sun sensor reference vector

% Calculate attitude DCM using TRIAD
C_BN_triad = TRIAD(Bv1,Bv2,Nv1,Nv2);

disp('DCM from TRIAD: ')
disp(C_BN_triad)

% Normalize vectors for QUEST use
Bv1 = Bv1./norm(Bv1);
Bv2 = Bv2./norm(Bv2);
Nv1 = Nv1./norm(Nv1);
Nv2 = Nv2./norm(Nv2);
Bv = [Bv1,Bv2];
Nv = [Nv1,Nv2];
w = [2,1]; % weight array, with star tracker twice as accurate

% Calculate attitude DCM using QUEST
C_BN_quest = QUEST(Bv,Nv,w);

disp('DCM from QUEST: ')
disp(C_BN_quest)

% Calculate the principle rotation angle between the two DCMs
angle_diff = PR(C_BN_quest*C_BN_triad');

disp('Principle rotation angle between DCMs: ')
disp(round(angle_diff,4))

DCM from TRIAD:
    0.8262    0.4674   -0.3145
   -0.5196    0.8479   -0.1048
    0.2177    0.2500    0.9435

DCM from QUEST:
    0.8273    0.4674   -0.3115
   -0.5193    0.8479   -0.1067
    0.2142    0.2500    0.9442

Principle rotation angle between DCMs:
    0.2076
```

Question 5

```
Iw = 0.1; %kg*m^2
% Calculate current attitude in MRPs
```

```

mrp = DCM2MRPs(C_BN_quest);
% Set initial conditions for integration
initial = [mrp;y2(end,:)]';
opts = odeset('MaxStep',2); % Sets max time step for ode45
J = [500,0,0;0,400,-7;0,-7,440]; % inertia matrix for full satellite system
(kg*m^2)

states = cell(1,3);
times = cell(1,3);
K = [1000,1000,1000,50,700,1000];
d = [50,150,700,50,50,50];
% Integrate the system utilizing different gains
for i = 1:6
    D = diag([d(i),d(i),d(i)]);
    mrp = [0;0;0];
    [t,s] = ode45(@(t,w) full_dyn(w,J,Iw,D,K(i),mrp),[0,50],initial,opts);
    times{i} = t;
    states{i} = s;
end

% Plot the comparison of D gains (MRP history)
figure(5);
tcl = tiledlayout(4,1);
title(tcl,'Time Histories of MRPs, Comparing D Gains')
nexttile(tcl)
hold on;
plot(times{1},states{1}(:,1));
plot(times{2},states{2}(:,1));
plot(times{3},states{3}(:,1));
ylabel('$\sigma_1$', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(times{1},states{1}(:,2));
plot(times{2},states{2}(:,2));
plot(times{3},states{3}(:,2));
ylabel('$\sigma_2$', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(times{1},states{1}(:,3));
plot(times{2},states{2}(:,3));
plot(times{3},states{3}(:,3));
ylabel('$\sigma_3$', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
Lgnd = legend('K=1000,D=50', 'K=1000,D=150', 'K=1000,D=700');
Lgnd.Layout.Tile = 4;

% Plot the comparison of K gains (MRP history)
figure(50);
tcl = tiledlayout(4,1);
title(tcl,'Time Histories of MRPs, Comparing K Gains')
nexttile(tcl)
hold on;

```

```

plot(times{4},states{4}(:,1));
plot(times{5},states{5}(:,1));
plot(times{6},states{6}(:,1));
ylabel('$\sigma_1$', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(times{4},states{4}(:,2));
plot(times{5},states{5}(:,2));
plot(times{6},states{6}(:,2));
ylabel('$\sigma_2$', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(times{4},states{4}(:,3));
plot(times{5},states{5}(:,3));
plot(times{6},states{6}(:,3));
ylabel('$\sigma_3$', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
Lgnd = legend('K=50,D=50', 'K=700,D=50', 'K=1000,D=50');
Lgnd.Layout.Tile = 4;

% Plot the angular velocity history for the chosen gains
figure(6);
tcl = tiledlayout(3,1);
title(tcl, 'Time Histories of Angular Velocities')
nexttile(tcl)
hold on;
plot(times{3},states{3}(:,4));
ylabel('W1 (Rad/s)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(times{3},states{3}(:,5));
ylabel('W2 (Rad/s)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(times{3},states{3}(:,6));
ylabel('W3 (Rad/s)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")

% Plot the angular velocity of the RWs for the chosen gains
figure(16);
tcl = tiledlayout(3,1);
title(tcl, 'Time Histories of Reaction Wheel Speed')
nexttile(tcl)
hold on;
plot(times{3},states{3}(:,7).*60./2./pi);
ylabel('$W_{w1}$ (RPM)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(times{3},states{3}(:,8).*60./2./pi);

```

```

ylabel('$W_{w2}$ (RPM)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(times{3}, states{3}(:, 9) .* 60 ./ 2 ./ pi);
ylabel('$W_{w3}$ (RPM)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")

% Calculate the norm errors for the both the attitude and angular
% velocities, and the torque at each step
goal = [0; 0; 0];
s_e = [];
w_e = [];
u = []; K = 1000; D = diag([700, 700, 700]);
J = [500, 0, 0; 0, 400, -7; 0, -7, 440];
for i = 1:length(states{3}(:, 1))
    sig = states{3}(i, 1:3)';
    sig_e = ((1 - norm(sig)^2) * goal - (1 - norm(goal)^2) * sig +
cross(2 * goal, sig)) ...
        / (1 + norm(sig)^2 * norm(goal)^2 - 2 * dot(sig, goal));
    s_e(i) = norm(sig_e);
    w_e(i) = norm(states{3}(i, 4:6));
    w = states{3}(i, 4:6)';
    u(i, :) = (-K * sig_e + D * w + skew(w) * J * w)';
end

% Plot the angular velocity history for the chosen gains
figure(15);
tcl = tiledlayout(2, 1);
title(tcl, 'Time Histories of Orientation Errors')
nexttile(tcl)
semilogy(times{3}, s_e); hold on;
yline(10^-5);
ylabel('MRP Error Norm')
xlabel('Time (s)')
nexttile(tcl)
semilogy(times{3}, w_e); hold on;
yline(10^-5);
ylabel('Angular Velocity Error Norm')
xlabel('Time (s)')

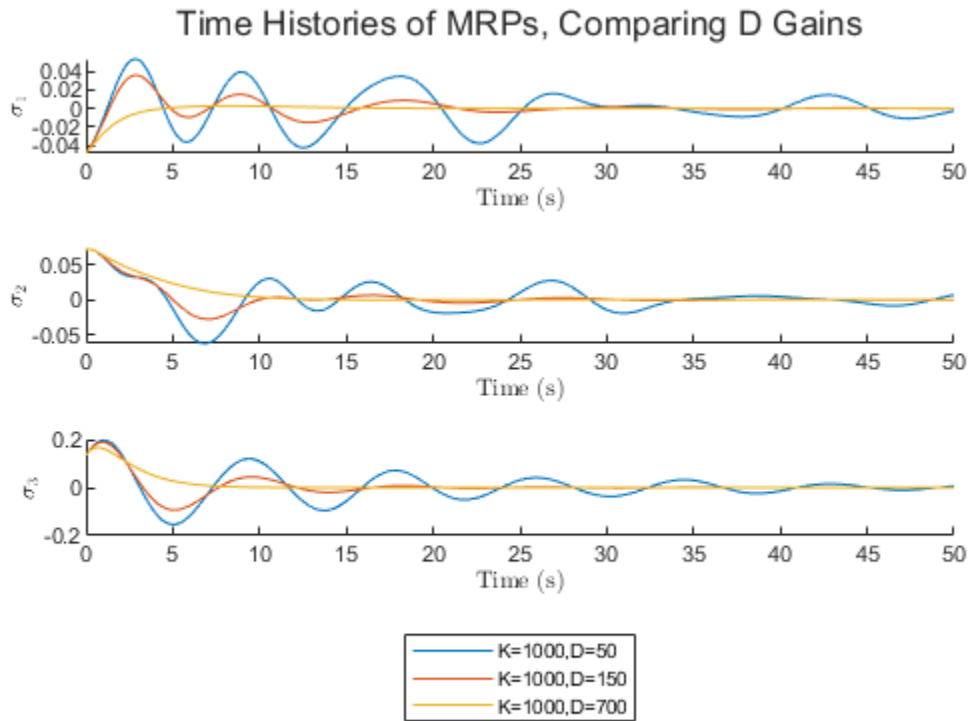
% Plot the angular velocity of the RWs for the chosen gains
figure(17);
tcl = tiledlayout(3, 1);
title(tcl, 'Time Histories of Reaction Wheel Torque')
nexttile(tcl)
hold on;
plot(times{3}, u(:, 1));
ylabel('U1 (Nm)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(times{3}, u(:, 2));
ylabel('U2 (Nm)', "Interpreter", "latex", "FontWeight", "Bold")

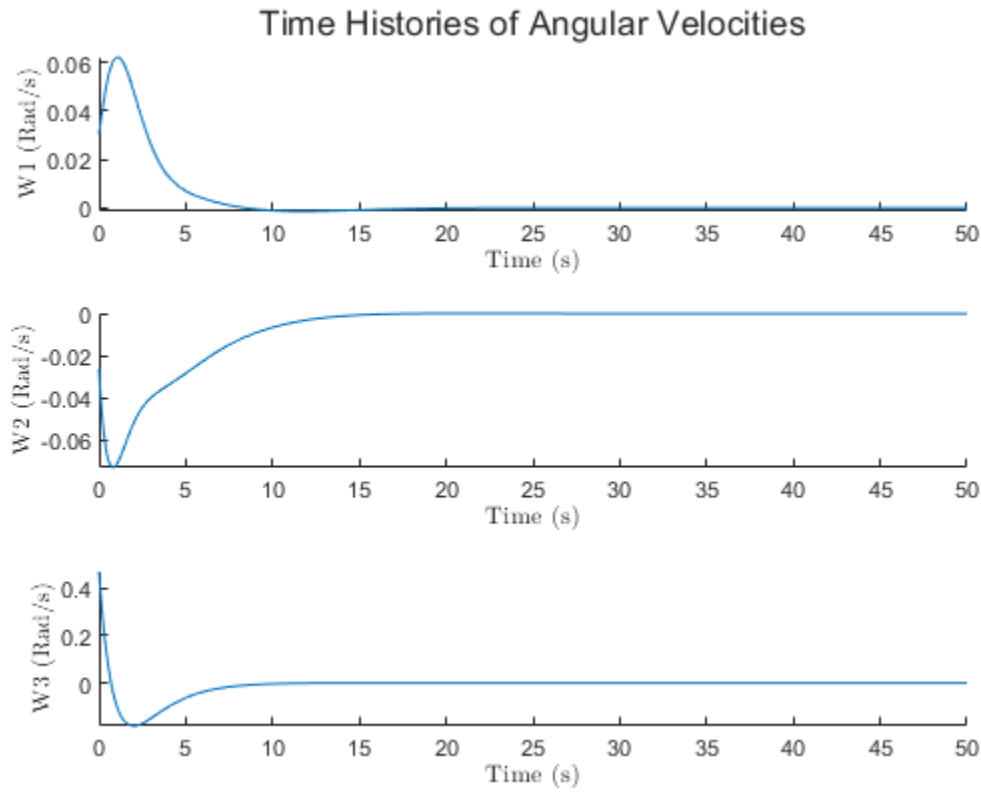
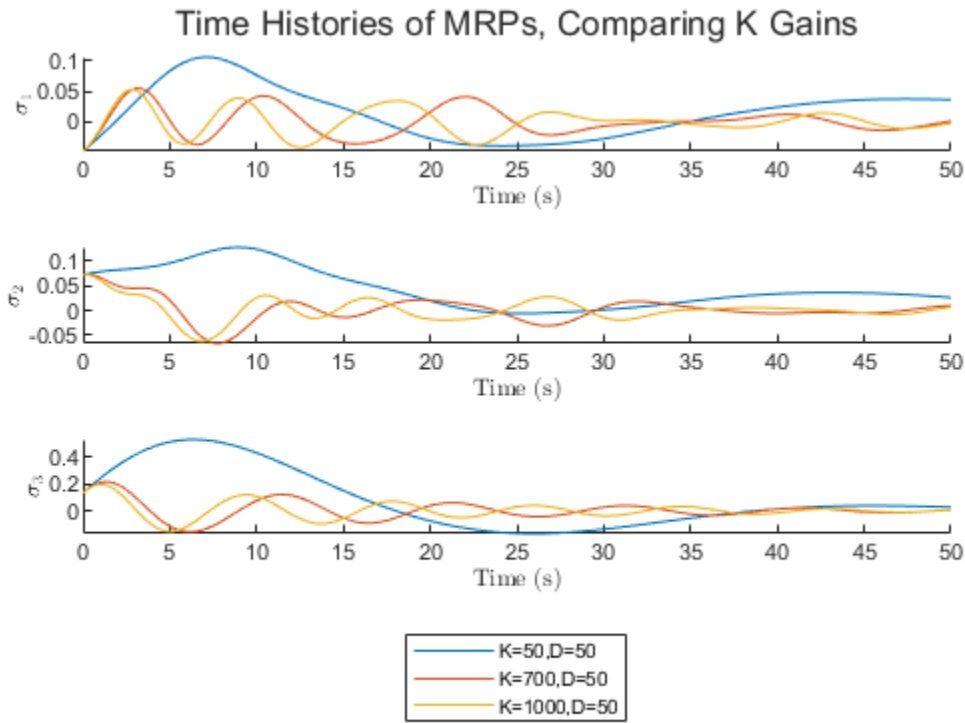
```

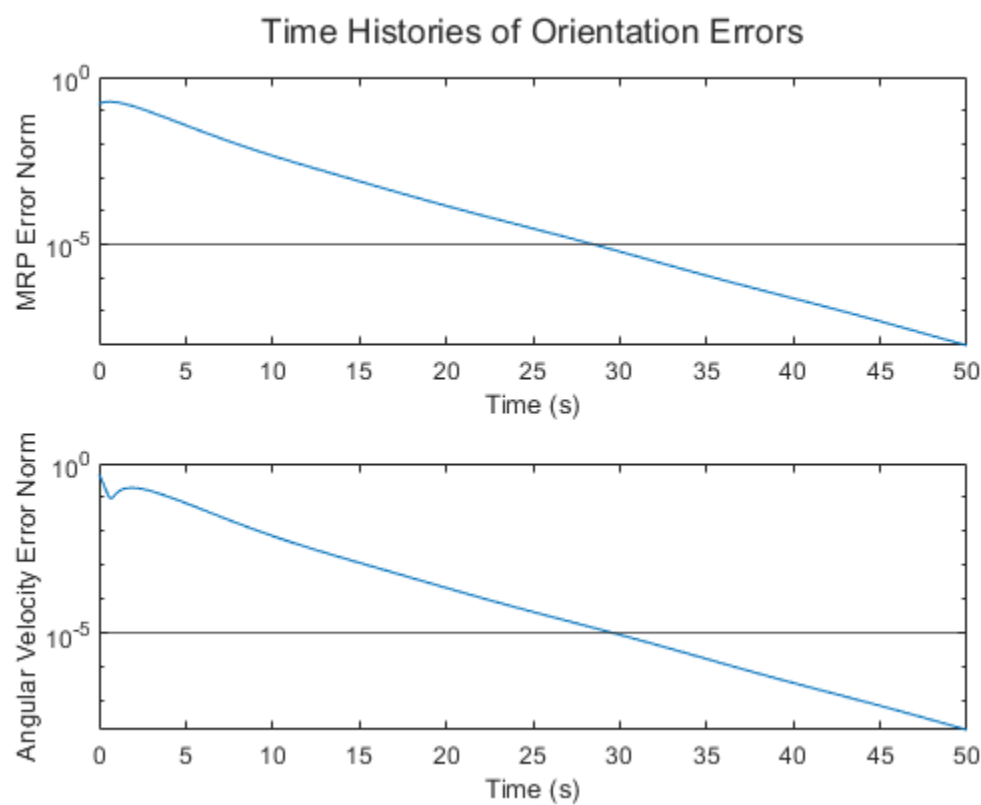
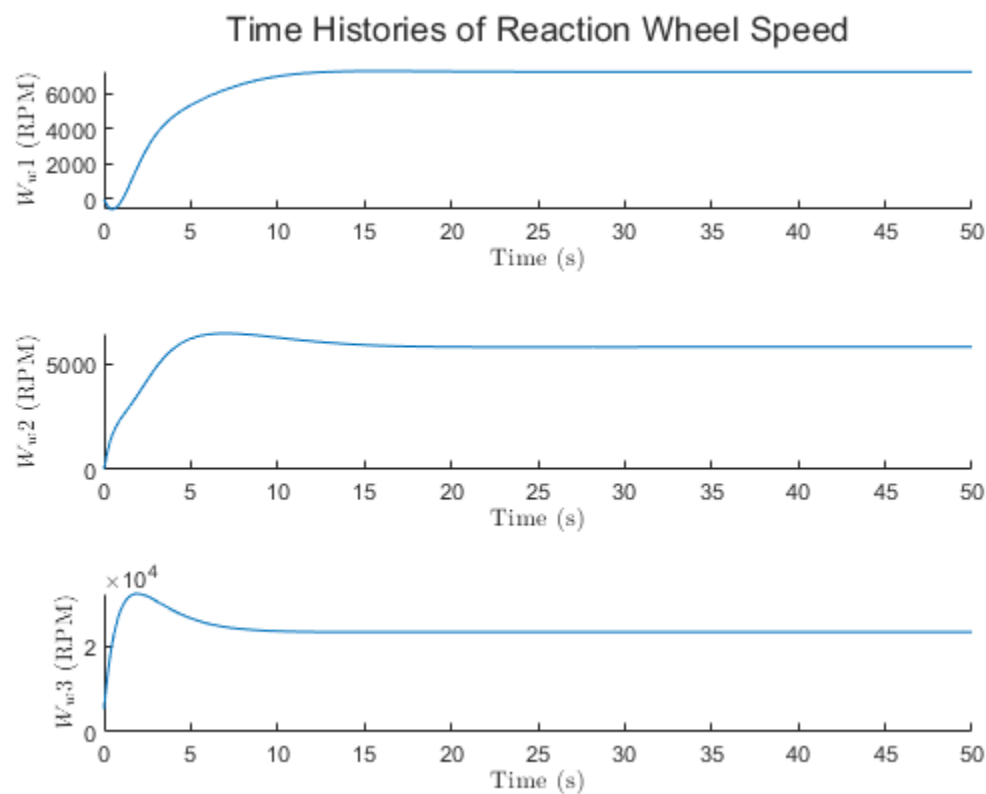
```

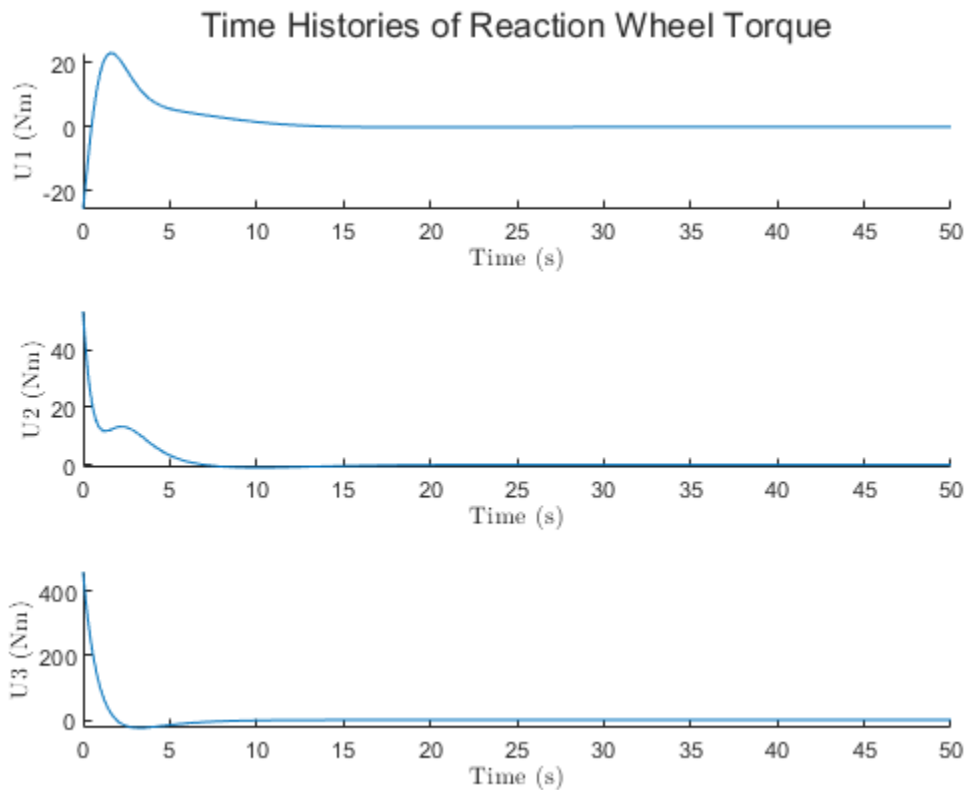
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(times{3},u(:,3));
ylabel('U3 (Nm)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")

```









Question 5.3

```
Iw = 0.1; %kg*m^2
mrp = DCM2MRPs(C_BN_quest);
initial = [mrp;y2(end,:)'];
opts = odeset('MaxStep',2);
J = [500,0,0;0,400,-7;0,-7,440];
mrp = [0;0;0];
[t,s] = ode45(@(t,w) lin_dyn(w,J,Iw,mrp),[0,50],initial,opts);

figure(7);
tcl = tiledlayout(4,1);
title(tcl,'Time Histories of MRPs')
nexttile(tcl)
hold on;
plot(t,s(:,1));
plot(times{3},states{3}(:,1));
ylabel('$\sigma_1$', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(t,s(:,2));
plot(times{3},states{3}(:,2));
ylabel('$\sigma_2$', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
```

```

nexttile(tcl)
hold on;
plot(t,s(:,3));
plot(times{3},states{3}(:,3));
ylabel('$\sigma_3$', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
Lgnd = legend('Adaptive LQR', 'Lyapunov');
Lgnd.Layout.Tile = 4;

figure(8);
tcl = tiledlayout(4,1);
title(tcl, 'Time Histories of Angular Velocities')
nexttile(tcl)
hold on;
plot(t,s(:,4));
plot(times{3},states{3}(:,4));
ylabel('W1 (Rad/s)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(t,s(:,5));
plot(times{3},states{3}(:,5));
ylabel('W2 (Rad/s)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(t,s(:,6));
plot(times{3},states{3}(:,6));
ylabel('W3 (Rad/s)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
Lgnd = legend('Adaptive LQR', 'Lyapunov');
Lgnd.Layout.Tile = 4;

% Plot the angular velocity of the RWs for the chosen gains
figure(20);
tcl = tiledlayout(4,1);
title(tcl, 'Time Histories of Reaction Wheel Speed')
nexttile(tcl)
hold on;
plot(t,s(:,7).*60./2./pi);
plot(times{3},states{3}(:,7).*60./2./pi);
ylabel('$W_{w1}$ (RPM)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(t,s(:,8).*60./2./pi);
plot(times{3},states{3}(:,8).*60./2./pi);
ylabel('$W_{w2}$ (RPM)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(t,s(:,9).*60./2./pi);
plot(times{3},states{3}(:,9).*60./2./pi);

```

```

ylabel('$W_{w3}$ (RPM)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
Lgnd = legend('Adaptive LQR', 'Lyapunov');
Lgnd.Layout.Tile = 4;

% Calculate the norm errors for the both the attitude and angular
% velocities, and the torque at each step
goal = [0;0;0];
s_e2 = [];
w_e2 = [];
u2 = []; K = 1000; D = diag([700,700,700]);
J = [500,0,0;0,400,-7;0,-7,440];
for i = 1:length(s(:,1))
    sig = s(i,1:3)';
    sig_e = ((1-norm(sig)^2)*goal - (1-norm(goal)^2)*sig +
cross(2*goal,sig)) ...
        /(1+norm(sig)^2*norm(goal)^2 - 2*dot(sig,goal));
    s_e2(i) = norm(sig_e);
    w_e2(i) = norm(s(i,4:6));
    w = s(i,4:6)';
    [~,temp_u] = lin_dyn([sig;w;s(i,7:9)'],J,Iw,[0;0;0]);
    u2(i,:) = temp_u';
end

% Plot the angular velocity history for the chosen gains
figure(21);
tcl = tiledlayout(3,1);
title(tcl,'Time Histories of Orientation Errors')
nexttile(tcl)
semilogy(t,s_e2); hold on;
semilogy(times{3},s_e);
yline(10^-5);
ylabel('MRP Error Norm')
xlabel('Time (s)')
nexttile(tcl)
semilogy(t,w_e2); hold on;
semilogy(times{3},w_e);
yline(10^-5);
ylabel('Angular Velocity Error Norm')
xlabel('Time (s)')
Lgnd = legend('Adaptive LQR', 'Lyapunov');
Lgnd.Layout.Tile = 3;

% Plot the angular velocity of the RWs for the chosen gains
figure(22);
tcl = tiledlayout(4,1);
title(tcl,'Time Histories of Reaction Wheel Torque')
nexttile(tcl)
plot(t,u2(:,1)); hold on;
plot(times{3},u(:,1));
ylabel('U1 (Nm)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;

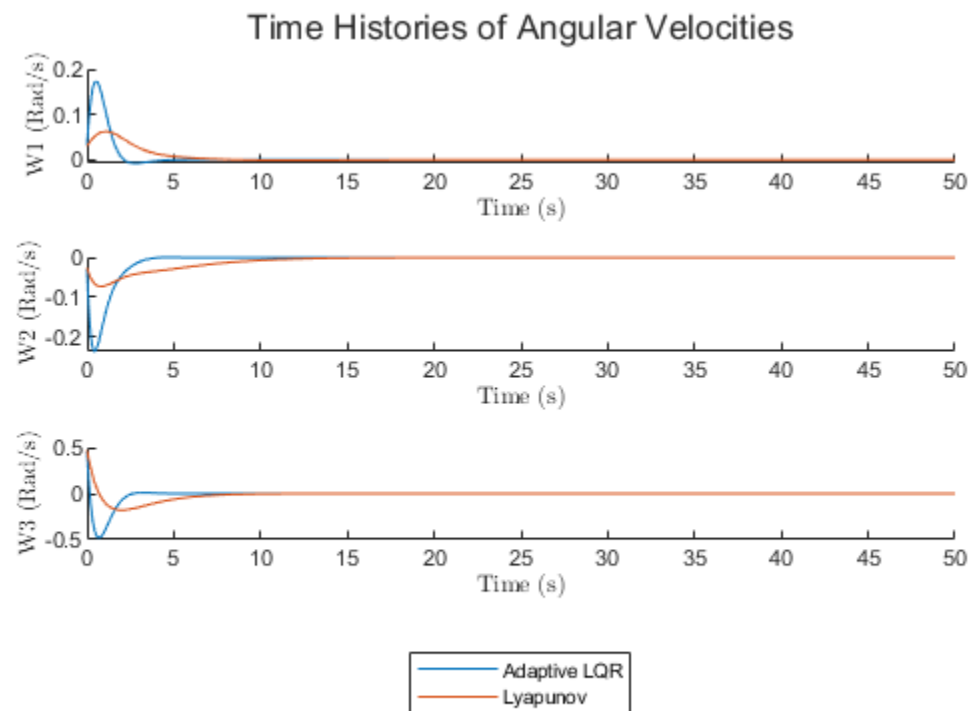
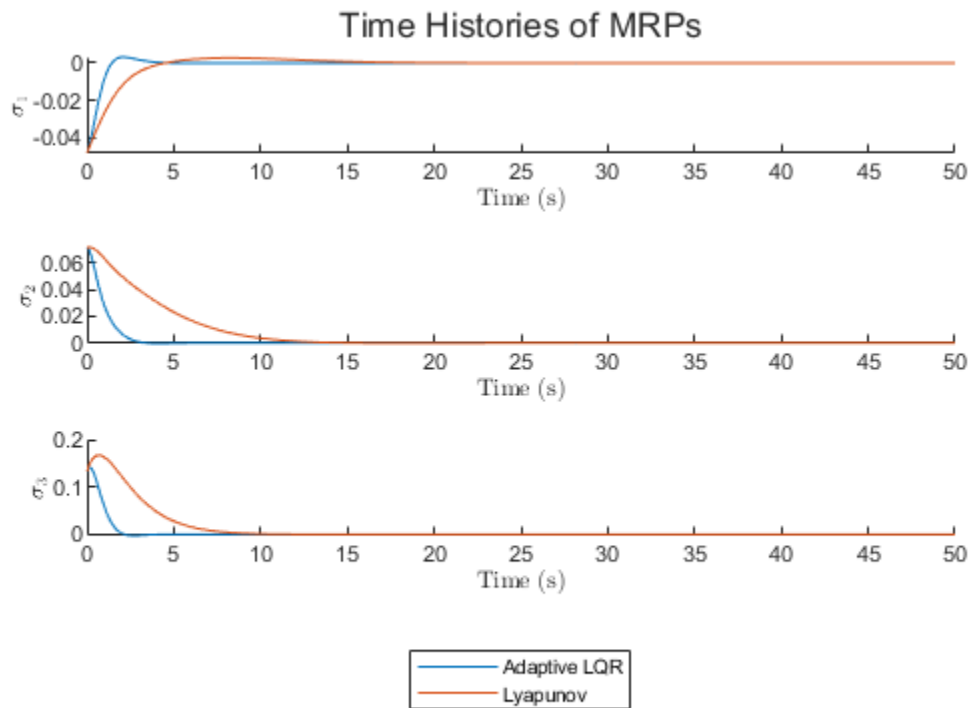
```

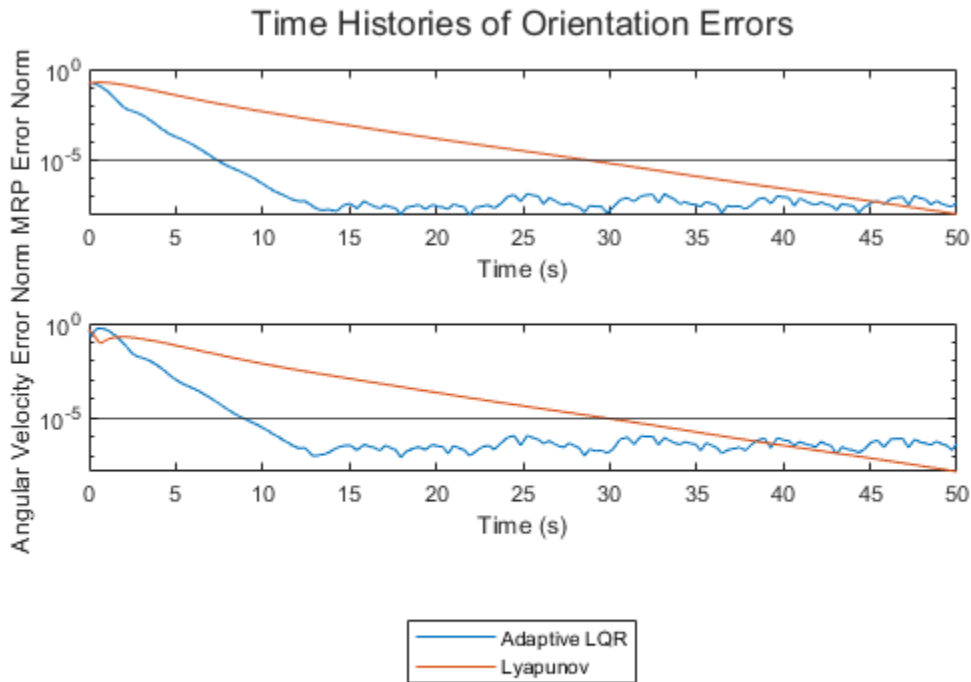
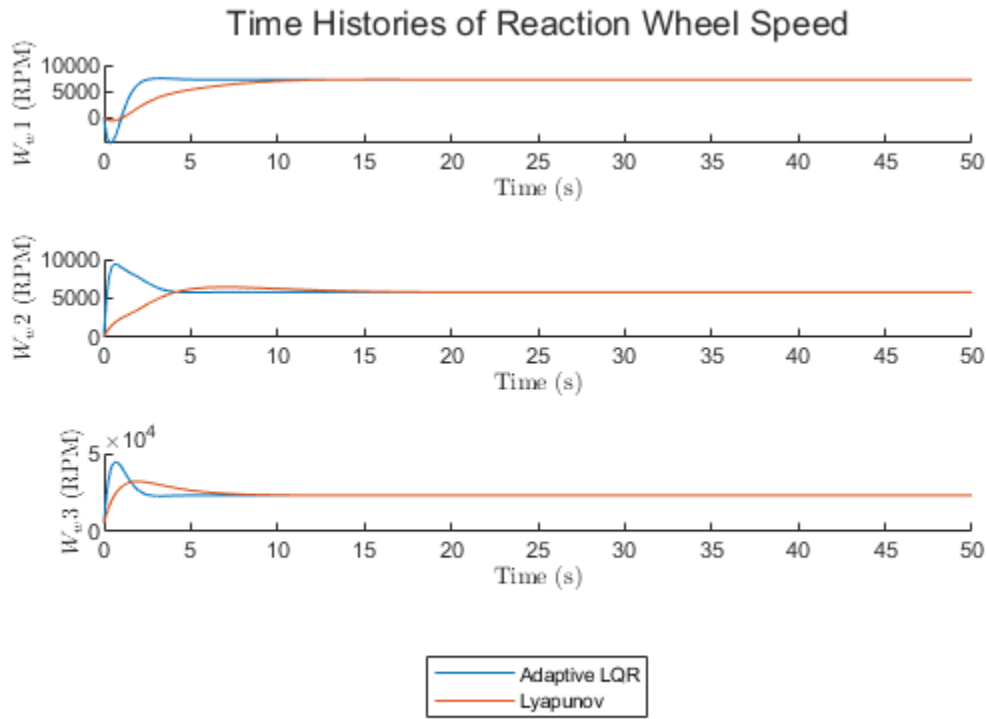
```

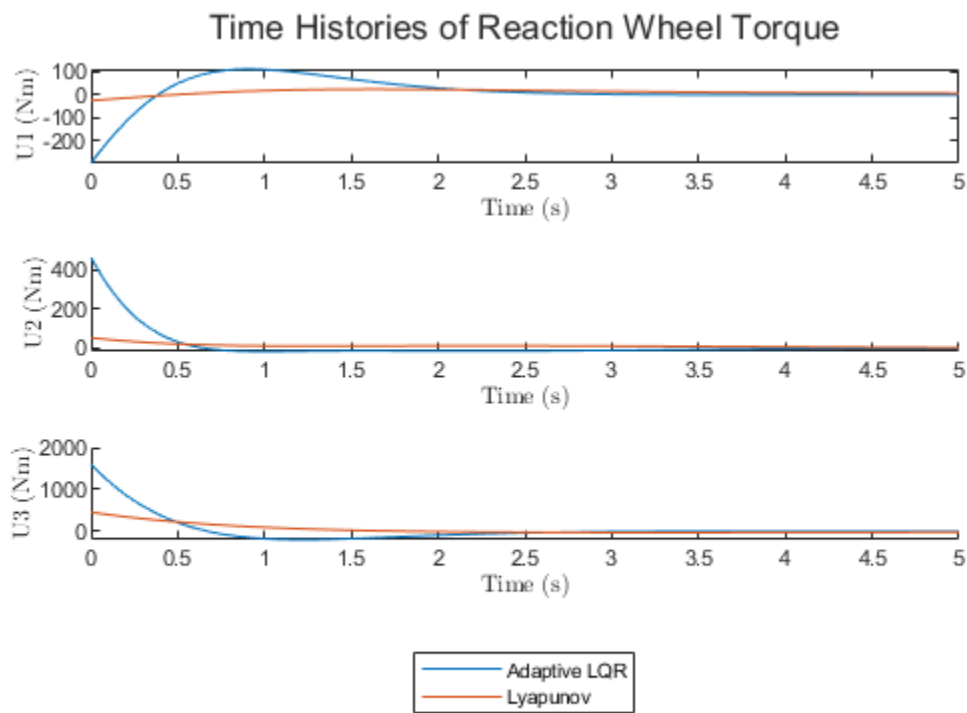
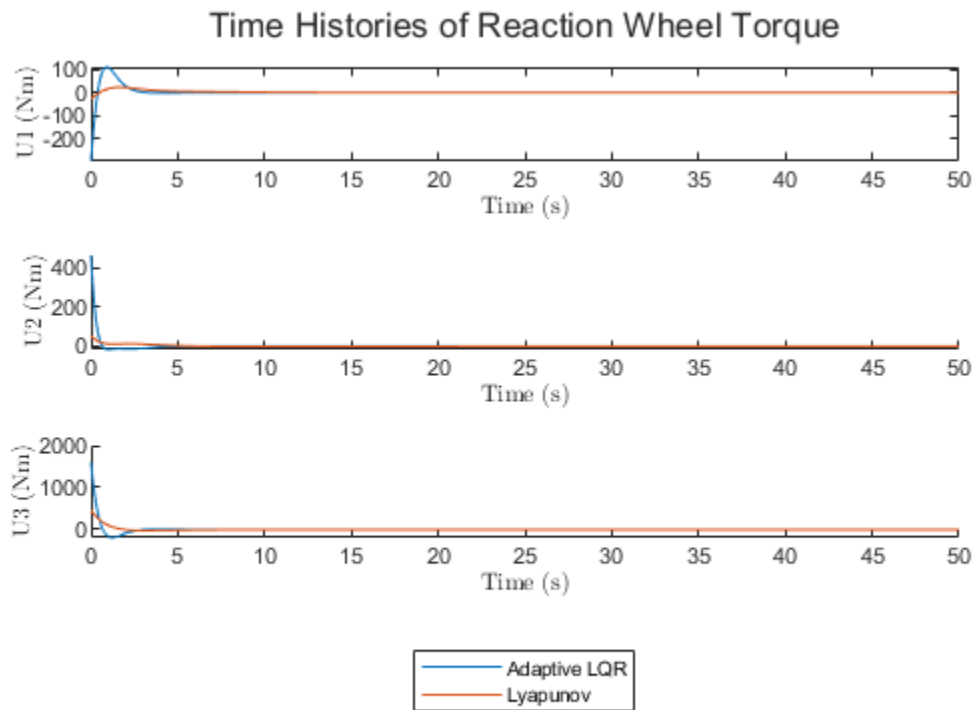
plot(t,u2(:,2)); hold on;
plot(times{3},u(:,2));
ylabel('U2 (Nm)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
nexttile(tcl)
hold on;
plot(t,u2(:,3)); hold on;
plot(times{3},u(:,3));
ylabel('U3 (Nm)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
Lgnd = legend('Adaptive LQR', 'Lyapunov');
Lgnd.Layout.Tile = 4;

% Plot the angular velocity of the RWs for the chosen gains
figure(23);
tcl = tiledlayout(4,1);
title(tcl, 'Time Histories of Reaction Wheel Torque')
nexttile(tcl)
plot(t,u2(:,1)); hold on;
plot(times{3},u(:,1));
ylabel('U1 (Nm)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
xlim([0,5])
nexttile(tcl)
hold on;
plot(t,u2(:,2)); hold on;
plot(times{3},u(:,2));
ylabel('U2 (Nm)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
xlim([0,5])
nexttile(tcl)
hold on;
plot(t,u2(:,3)); hold on;
plot(times{3},u(:,3));
ylabel('U3 (Nm)', "Interpreter", "latex", "FontWeight", "Bold")
xlabel('Time (s)', "Interpreter", "latex", "FontWeight", "Bold")
xlim([0,5])
Lgnd = legend('Adaptive LQR', 'Lyapunov');
Lgnd.Layout.Tile = 4;

```





Functions

```
% Full dynamics including Lyapunov control law
function [prime] = full_dyn(x,J,Iw,D,K,goal)
    sig = x(1:3);
    sig2 = sig'*sig;
    % Calculate and switch to shadow MRP if necessary
    if sig2 > 1
        sig = -sig./sig2;
    end
    sig_e = ((1-norm(sig)^2)*goal - (1-norm(goal)^2)*sig +
cross(2*goal,sig)) ...
        /(1+norm(sig)^2*norm(goal)^2 - 2*dot(sig,goal));

    w = x(4:6);
    ww = x(7:9);

    % Inertia matrix for RWs
    Iww = [Iw,0,0;0,Iw,0;0,0,Iw];
    % Torque calculation using predefined gains
    u = -K*sig_e + D*w + skew(w)*J*w;
    % Angular acceleration of RWs
    ww_dot = inv(Iww)*(skew(ww)*Iww*ww+u);
    % Angular acceleration of the spacecraft
    wdot = J\(-skew(w)*J*w-skew(w)*Iww*ww-Iww*ww_dot);
    % Attitude acceleration of the spacecraft
    sig_dot = 0.25*((1-sig2)*eye(3) + 2*skew(sig) + 2*sig*sig')*w;
    % Propagate solution
    prime = [sig_dot;wdot;ww_dot];
end

% Full dynamics using linear control law
function [prime,u] = lin_dyn(x,J,Iw,goal)
    % Calculate gains using current state
    K = gain(x(1:6));
    sig = x(1:3);
    sig2 = sig'*sig;
    % Calculate and switch to shadow MRP if necessary
    if sig2 > 1
        sig = -sig./sig2;
    end
    sig_e = ((1-norm(sig)^2)*goal - (1-norm(goal)^2)*sig +
cross(2*goal,sig)) ...
        /(1+norm(sig)^2*norm(goal)^2 - 2*dot(sig,goal));

    w = x(4:6);
    ww = x(7:9);

    % Inertia matrix for RWs
    Iww = [Iw,0,0;0,Iw,0;0,0,Iw];
    % Torque calculation using LQR gains
    u = -K*x(1:6);
    % Angular acceleration of RWs
```

```

ww_dot = inv(Iww)*(skew(ww)*Iww*ww+u);
% Angular acceleration of the spacecraft
wdot = J\(-skew(w)*J*w-skew(w)*Iww*ww-Iww*ww_dot);
% Attitude acceleration of the spacecraft
sig_dot = 0.25*((1-sig2)*eye(3) + 2*skew(sig) + 2*sig*sig')*w;
% Propagate solution
prime = [sig_dot;wdot;ww_dot];

% Function to assemble skew matrix from a vector
function M = skew(s)
    M = [0,-s(3),s(2); s(3),0,-s(1); -s(2),s(1),0];
end
% Function to calculate gains using LQR
function K = gain(x)
    sig_ = x(1:3);
    % Calculate and switch to shadow MRP if necessary
    sig2_ = sig_'*sig_;
    if sig2_ > 1
        sig_ = -sig_/sig2_;
    end

    w_ = x(4:6);
    Iw_ = 0.1;
    J_ = [500,400,440];
    % Calculate linear coefficients for state variables
    sd = (1-sig2_)*eye(3) + 2*skew(sig_) + 2*sig_*sig_';
    w_var = diag(J_)\(-skew(w_)*diag(J_));
    A = [0,0,0,0.25*sd(1,1),0.25*sd(1,2),0.25*sd(1,3);
        0,0,0,0.25*sd(2,1),0.25*sd(2,2),0.25*sd(2,3);
        0,0,0,0.25*sd(3,1),0.25*sd(3,2),0.25*sd(3,3);
        0,0,0,w_var(1,1),w_var(1,2),w_var(1,3);
        0,0,0,w_var(2,1),w_var(2,2),w_var(2,3);
        0,0,0,w_var(3,1),w_var(3,2),w_var(3,3)];

    % Calculate linear coefficients for input variables
    B = zeros(6,3);
    B(4,1) = -J_(1)*Iw_*inv(Iw_);B(5,2) = -J_(2)*Iw_*inv(Iw_);B(6,3) =
-J_(3)*Iw_*inv(Iw_);

    % Weights for each state variable and input
    Q = diag([500,500,500,20,20,20]);
    R = diag([0.00001,0.00001,0.00001]);

    % Calculate gain matrix using LQR
    [K,~,~] = lqr(A,B,Q,R);

end
end

% Define the differential kinematic equation for 3-2-1 Euler angles
function [rotation_rates] = angular_rates(t,w,J,hw,Iw,ts)

    hw_dot = hw/ts;
    ww_dot = hw_dot/Iw;

```

```

ww_dot = [0;0;ww_dot];

Iww = [Iw,0,0;0,Iw,0;0,0,Iw];

wdot = J\(-skew(w(1:3))*J*w(1:3)-skew(w(1:3))*Iww*w(4:6)-Iww*ww_dot);

rotation_rates = [wdot;ww_dot];

end

% Function to assemble skew matrix from a vector
function M = skew(s)
    M = [0,-s(3),s(2); s(3),0,-s(1); -s(2),s(1),0];
end

% Calculate the nutation angle
function angle = nutation(J,Iw,w,control)
    for i = 1:length(w(:,1))
        H(i,:) = (J*w(i,1:3)' + Iw*w(i,4:6)')';
        angle(i) = acosd(dot(H(i,:),control)/(norm(H(i,:))*norm(control)));
    end
end

% Calculate attitude DCM using TRIAD
function C_BN = TRIAD(Bv1,Bv2,Nv1,Nv2)
    % Normalize all vectors
    Bv1 = Bv1./norm(Bv1);
    Bv2 = Bv2./norm(Bv2);
    Nv1 = Nv1./norm(Nv1);
    Nv2 = Nv2./norm(Nv2);
    % Calculate vectors in intermediate frame
    Bt1 = Bv1;
    Bt2 = cross(Bv1,Bv2)/norm(cross(Bv1,Bv2));
    Bt3 = cross(Bt1,Bt2);
    Nt1 = Nv1;
    Nt2 = cross(Nv1,Nv2)/norm(cross(Nv1,Nv2));
    Nt3 = cross(Nt1,Nt2);
    % Compute DCMs for both B and N frames
    C_BT = [Bt1,Bt2,Bt3];
    C_NT = [Nt1,Nt2,Nt3];
    % Compute attitude DCM
    C_BN = C_BT*C_NT';
end

% Calculate attitude DCM using QUEST
function C_BN = QUEST(Bv,Nv,w)
    % Calculate intermediate values
    B = zeros(3,3);
    for i = 1:length(w)
        B = B + w(i)*Bv(:,i)*Nv(:,i)';
    end
    S = B + B';
    sig = trace(B);
    Z = [B(2,3)-B(3,2);B(3,1)-B(1,3);B(1,2)-B(2,1)];

```

```

% Assemble K matrix
K = [sig,Z';Z,S-sig*eye(3,3)];
% Calculate max eigenvalue using Newton Raphson
eig_val = NR(sum(w),1e-10);

% Calculate Rodrigues parameters (eq. 3.238)
q = inv((eig_val+sig)*eye(3,3)-S)*Z;

% Assemble DCM from Rodrigues parameters
C_BN = (1/(1+q'*q))*[1+q(1)^2-q(2)^2-q(3)^2, 2*(q(1)*q(2)+q(3)),
2*(q(1)*q(3)-q(2)),
2*(q(1)*q(2)-q(3)), 1-q(1)^2+q(2)^2-q(3)^2,
2*(q(2)*q(3)+q(1)),
2*(q(1)*q(3)+q(2)), 2*(q(2)*q(3)-q(1)), 1-q(1)^2-
q(2)^2+q(3)^2];

% Eq 3.236 for Newton Raphson root solving
function y = f(s)
    y = det(K-s*eye(4,4));
end
% 1st derivative of Eq 3.236
function dy = fp(s)
    dy = -det(K-s*eye(4,4))*trace(inv(K-s*eye(4,4)));
end
% Root solving using Newton Raphson for the eigenvalue
function eig_val = NR(initial,tol)
    e = initial;
    err = f(e);
    while err > tol
        e = e - f(e)/fp(e);
        err = f(e);
    end
    eig_val = e;
end
end

% Find principle rotation angle of a DCM
function angle = PR(C)
    angle = acosd(0.5*(trace(C)-1));
end

% Convert DCM to MRP
function mrp = DCM2MRPs(DCM)
    c = sqrt(trace(DCM)+1);
    mrp = 1/(c*(c+2))*[DCM(2,3)-DCM(3,2);DCM(3,1)-DCM(1,3);DCM(1,2)-
DCM(2,1)];
end

```

Published with MATLAB® R2024b