Table of Contents

Gagandeep Thapar; AERO 560 HW3 1
Problem 1 1
Problem 2 2
run sim 2
unpack data: A 2
unpack data: B 2
plot data: A 2
plot data: B 3
plot misc 4

Gagandeep Thapar; AERO 560 HW3

Problem 1

```
Orbital Parameters Calculated:
 Radius of Orbit [km]: 7078.000
 Velocity of SC [km/s]: 7.504
 Orbital Period [min]: 98.770

Using the Atmospheric Model from MSISE-90 Atmosphere Model...
 Assuming Cd [~]: 2.2
 Density @ 700 km [kg/m3]: 3.580e-14
 Drag Force [N]: 6.653e-7

Drag Calcs:
 Drag Momentum across Orbit Period [mNms]: 3.943

Market RWs:
 To overcome 3.94 mNms, the Blue Canyon RWP015 is sufficient with 15.00 mNms.
 It will need to be destaurated after 3.80 orbits.
```

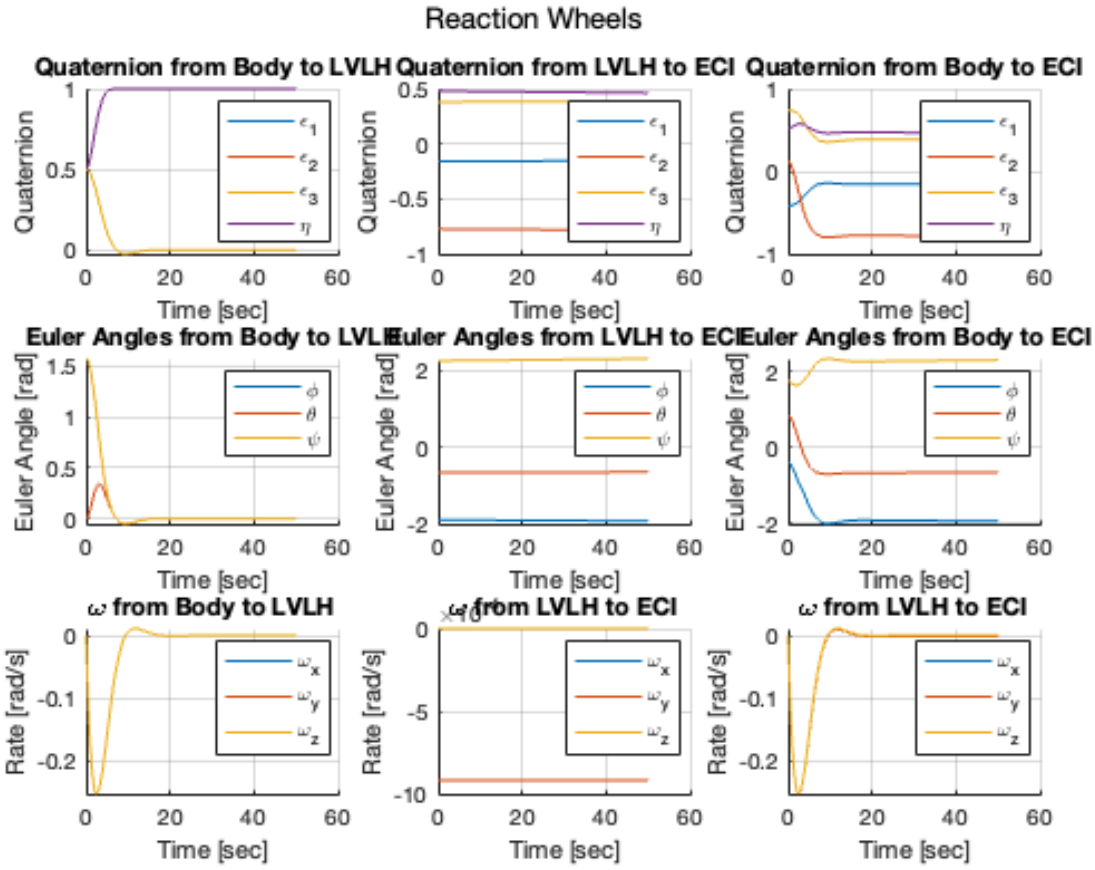
Problem 2

run sim

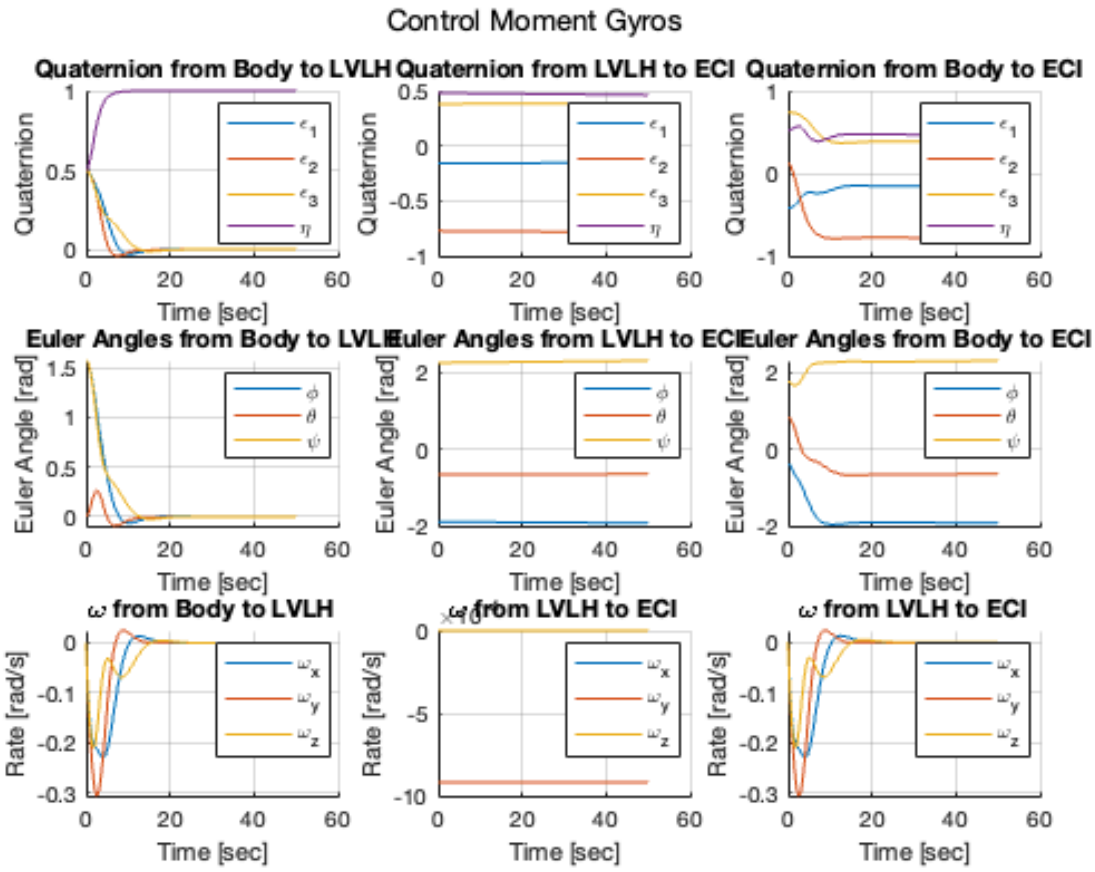
unpack data: A

unpack data: B

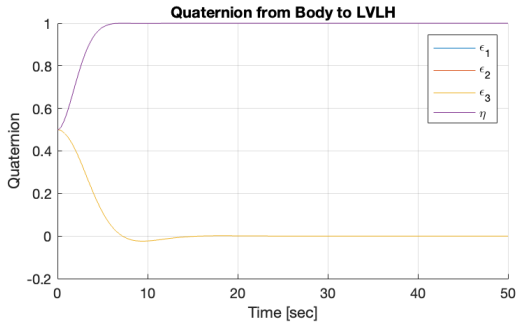
plot data: A

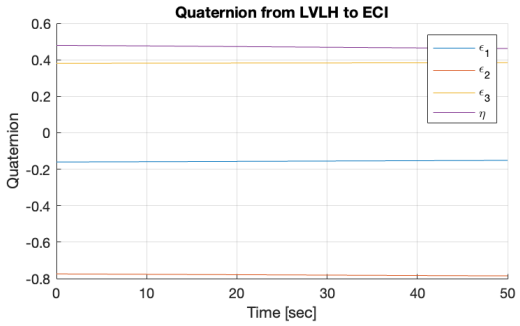


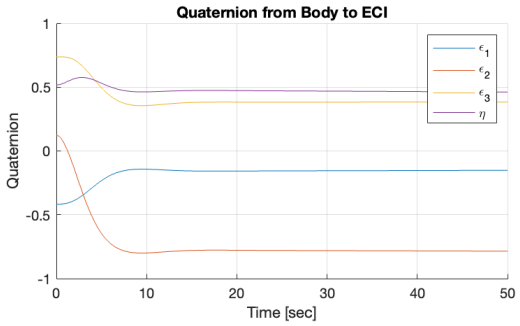
plot data: B

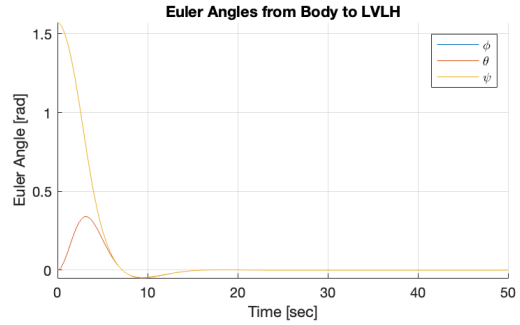

Control Moment Gyros
Quaternion from Body to LVLH
Quaternion from LVLH to ECI
Quaternion from Body to ECI
Quaternion
Time [sec]
Euler Angles from Body to LVLH
Euler Angles from LVLH to ECI
Euler Angles from Body to ECI
Euler Angle [rad]
ω from Body to LVLH
ω from LVLH to ECI
ω from LVLH to ECI
Rate [rad/s]

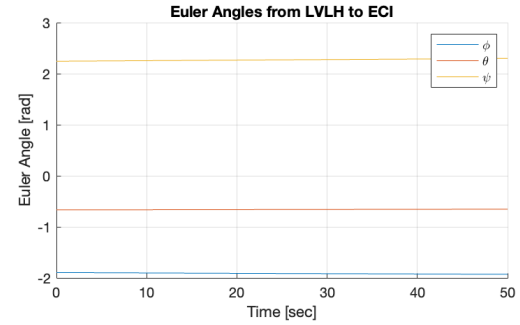
Reaction Wheels

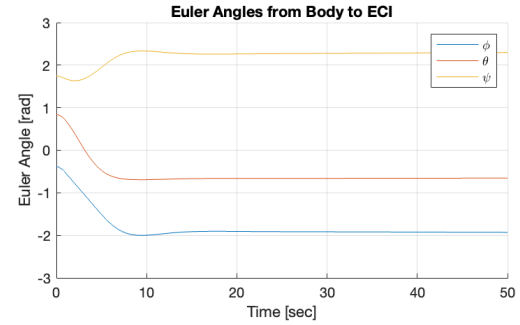


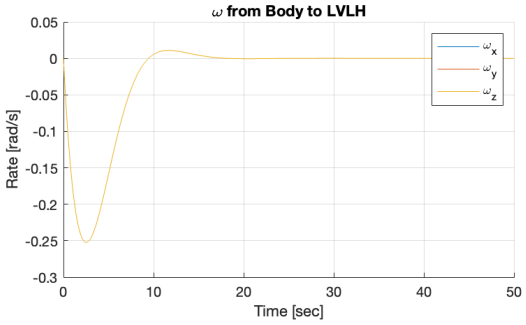


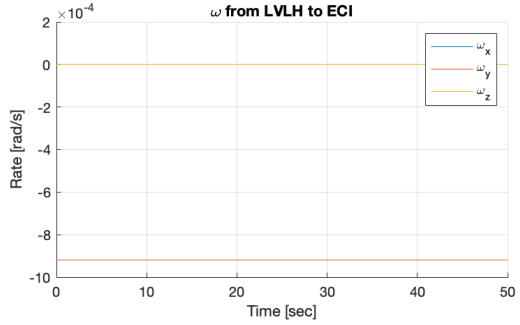


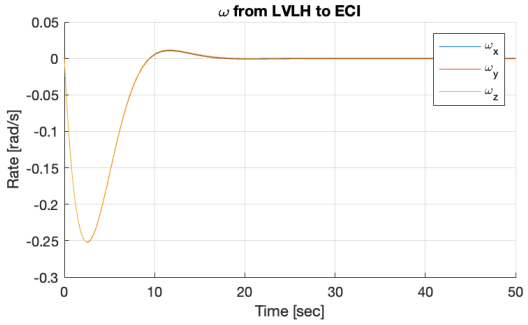




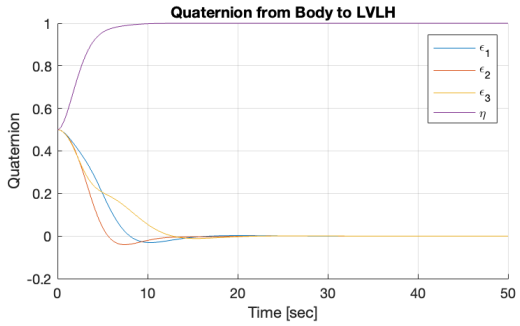


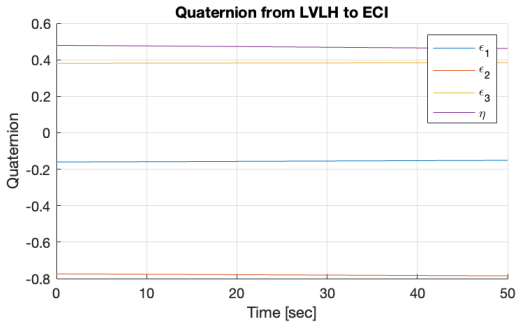


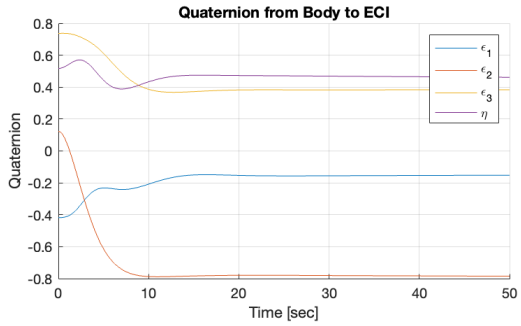


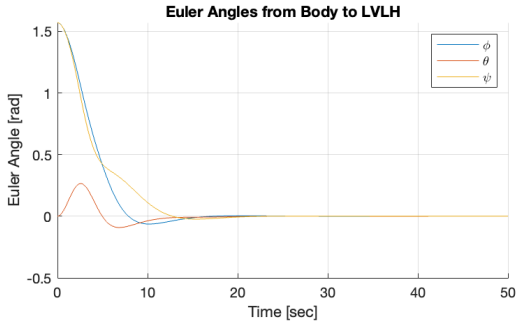


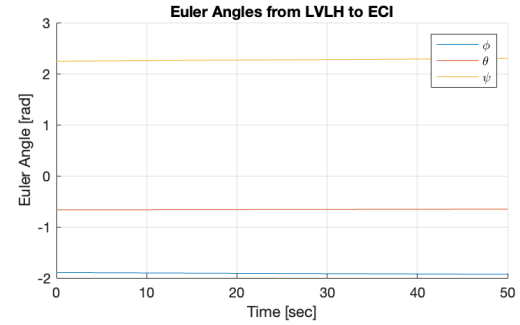
Control Moment Gyros

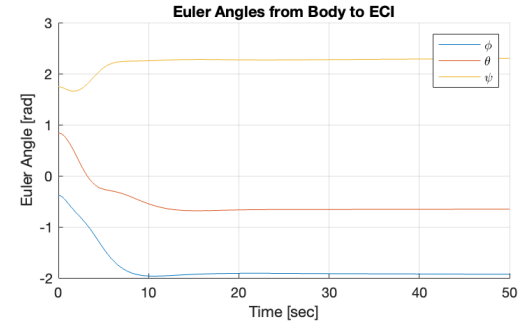


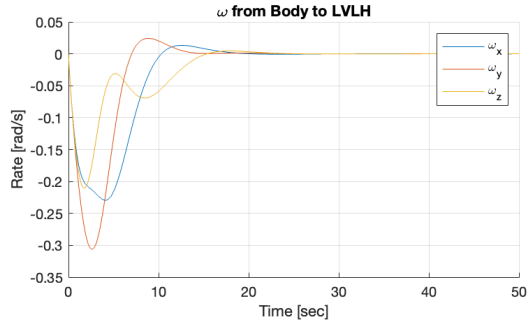


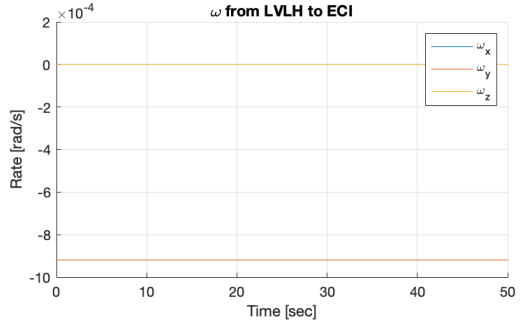


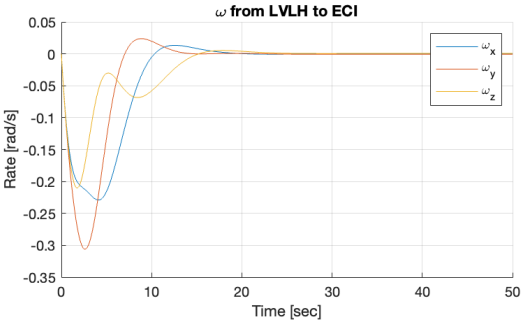




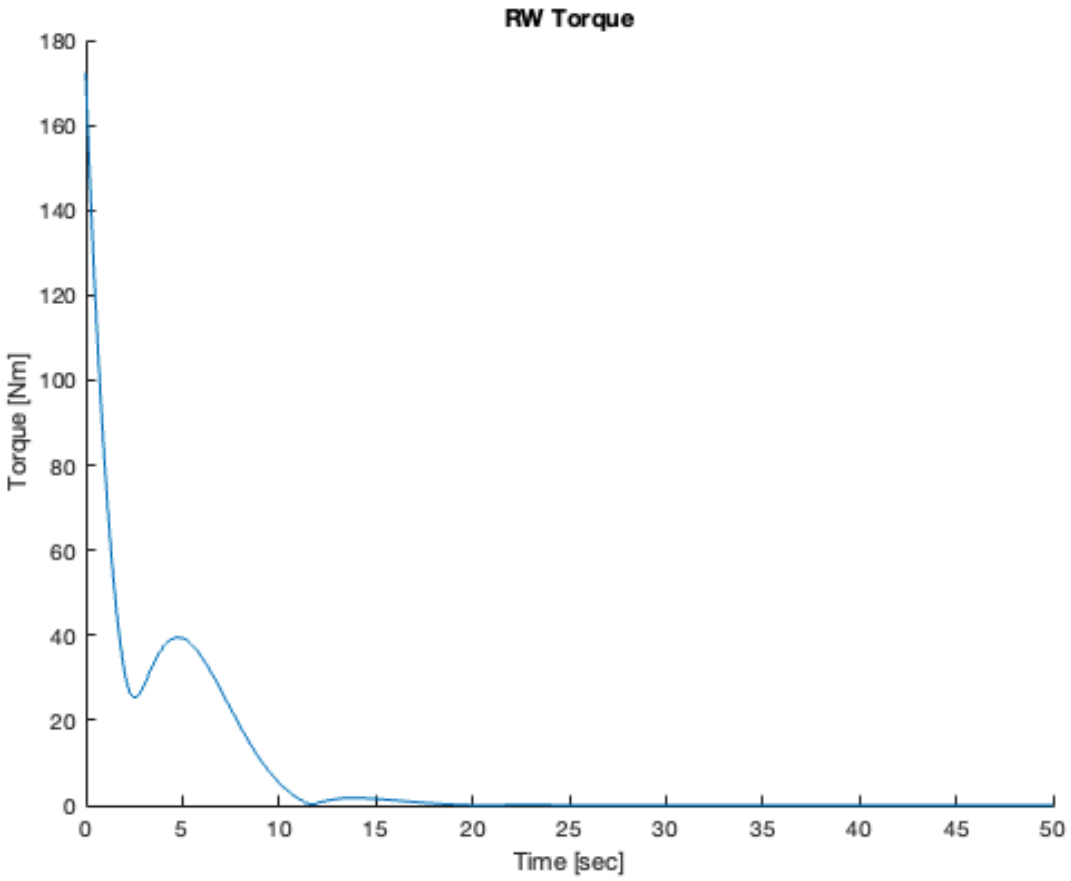


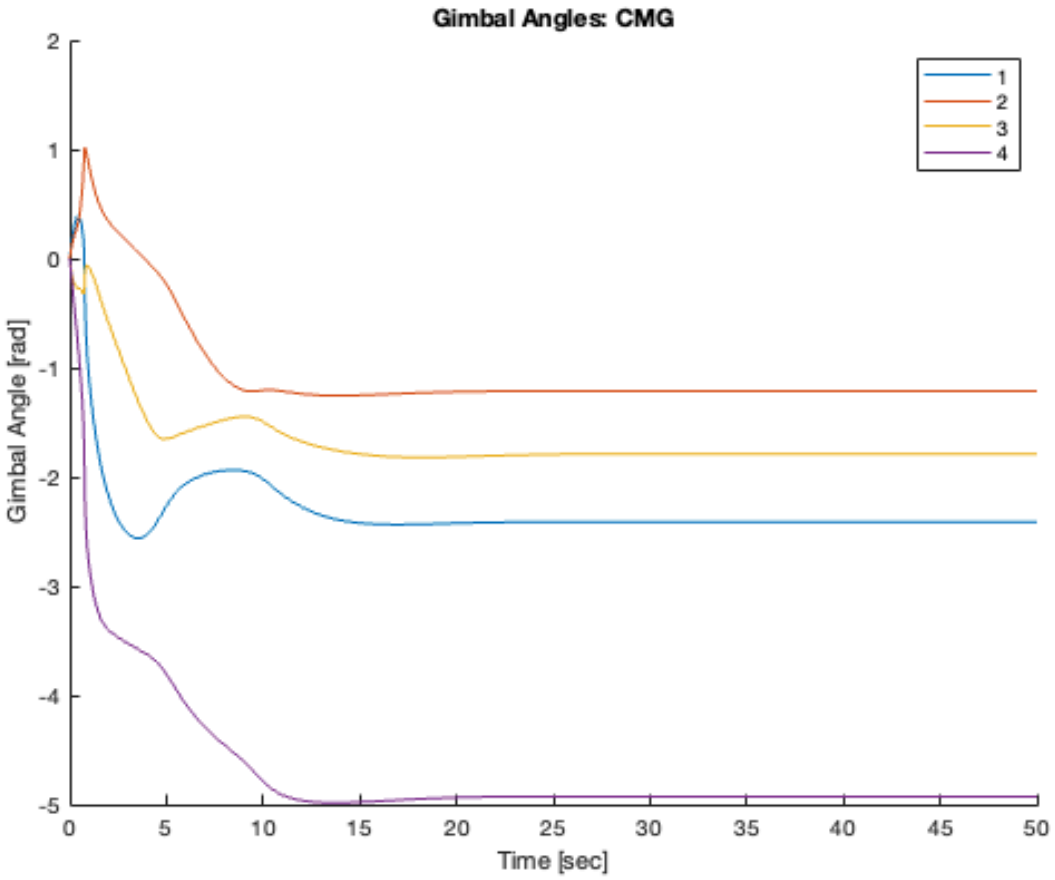






plot misc


RW Torque
Torque [Nm]
Time [sec]
180
160
140
120
100
80
60
40
20
0
0
5
10
15
20
25
30
35
40
45
50



Published with MATLAB® R2022b

Table of Contents

Gagandeep Thapar; AERO 560 HW3 1
Problem 1 1
Problem 2 2
run sim 3
unpack data: A 4
unpack data: B 4
plot data: A 4
plot data: B 7
plot misc 9

Gagandeep Thapar; AERO 560 HW3

```
% housekeeping
clc;
% clearvars;
% close all;
```

Problem 1

```
orbit.rad = 6378;   % [km] rad of earth
orbit.mu = 398600;  % [km3/s2] grav parameter of earth

% givens
z = 700;
sat.A = 3;
sat.del_CP = 1;
sat.Cd = 2.2;

% work
fprintf('\nOrbital Parameters Calculated:\n')

v = sqrt(orbit.mu/(orbit.rad+z));
T = 2*pi*(orbit.rad + z)^1.5 / sqrt(orbit.mu);

fprintf('\tRadius of Orbit [km]: %.3f\n', orbit.rad+z);
fprintf('\tVelocity of SC [km/s]: %.3f\n', v);
fprintf('\tOrbital Period [min]: %.3f\n', T/60);

fprintf('\nUsing the Atmospheric Model from MSISE-90 Atmosphere Model...\n')

rho = 3.58e-15;
F = 0.5 * sat.Cd * rho * sat.A * (v*1000)^2;

fprintf('\tAssuming Cd [~]: 2.2\n')
fprintf('\tDensity @ %d km [kg/m3]: %.3fe-14\n', z, rho*1e15);
fprintf('\tDrag Force [N]: %.3fe-7\n',F*1e7);

M = F*sat.del_CP * T;
```

```
fprintf('\nDrag Calcs:\n')
fprintf('\tDrag Momentum across Orbit Period [mNms]: %.3f\n', M*1e3);

rwp = 0.015;

fprintf('\nMarket RWs:\n')
fprintf('\tTo overcome %.2f mNms, the Blue Canyon RWP015 is sufficient with
 %.2f mNms. It will need to be destaurated after %.2f orbits.\n', M*1E3,
 rwp*1e3, (rwp/M))
```

Problem 2

```
% givens
orbit.h = 55759;    % [kg/m2] angular momentum
orbit.ecc = 0.001;  % [~] eccentricity
orbit.raan = 10;    % [deg] raan
orbit.inc = 42;     % [deg] inc
orbit.omega = 22;   % [deg] arg of perigee
orbit.theta = 0;    % [deg] true anomaly
orbit.mu = 398600;

sat.mass = 500;   % [kg]
sat.l = 1.5;    % [m] length dimension
sat.w = 1.5;    % [m] width dimension
sat.h = 3;    % [m] height dimension

sat.J = sat.mass/12 * [sat.l^2 + sat.h^2, 0, 0;
                       0, sat.w^2 + sat.h^2, 0;
                       0, 0, sat.l^2 + sat.w^2];    % principal inertial
 matrix

sat.des_e_lvlh = [0;0;0];
sat.des_n_lvlh = 1;
sat.des_q_lvlh = [sat.des_e_lvlh;sat.des_n_lvlh];
sat.zeta = 0.7*eye(3);    % [~] Dampening Coefficient
sat.wn = 0.5*eye(3);

wheel.I = 1.2;
wheel.theta = 57;
wheel.As = [sind(wheel.theta), 0, -sind(wheel.theta), 0;
            0, sind(wheel.theta), 0, -sind(wheel.theta);
            cosd(wheel.theta), cosd(wheel.theta), cosd(wheel.theta),
 cosd(wheel.theta)];

cmg.r = 0.2;
cmg.h = 0.05;
cmg.mass = 4.5;
cmg.omega = 800*[1;1;1;1];
cmg.theta = 57;
cmg.gamma = 0*[1;1;1;1];

cmg.J = cmg.mass/12 * [cmg.h^2 + 3*cmg.r^2, 0, 0;
                       0, cmg.h^2 + 3*cmg.r^2, 0;
```

```
                    0, 0, 6*cmg.r^2];    % principal inertial matrix]


% initial conditions
[orbit.R0, orbit.V0] = COES2STATE(orbit.h, orbit.ecc, orbit.inc, orbit.raan,
 orbit.omega, orbit.theta, orbit.mu);
orbit.R = orbit.R0;
orbit.V = orbit.V0;
orbit.T = 2*pi*norm(orbit.R0)^(1.5)/sqrt(orbit.mu);
orbit.n = 2*pi/orbit.T;

orbit.state0 = [orbit.R;orbit.V];

sat.w0_lvlh = [0;0;0]; % initial ang vel
sat.e0_lvlh = [0.5;0.5;0.5];   % initial quat
sat.n0_lvlh = 0.5;
sat.q0_lvlh = [sat.e0_lvlh;sat.n0_lvlh];
sat.euls0_lvlh = quat_eul([sat.e0_lvlh;sat.n0_lvlh]); % initial euler

sat.state0_lvlh = [sat.w0_lvlh;sat.e0_lvlh;sat.n0_lvlh;sat.euls0_lvlh];  %
 initial state

sat.Kd = 2*sat.J*sat.zeta*sat.wn;
sat.Kp = 2*(sat.J)*(sat.wn^2);

st = sind(cmg.theta);
ct = cosd(cmg.theta);

cmg.Ag = [st 0 -st 0;
            0 st 0 -st;
            ct ct ct ct];

cmg.At = [-ct 0 ct 0;
            0 -ct 0 ct;
            st st st st];

cmg.As = [0 -1 0 1;
            1 0 -1 0;
            0 0 0 0];

cmg.Iws = cmg.J(3,3)*eye(4);
cmg.Iwg = cmg.J(1,1)*eye(4);
cmg.Iwt = cmg.J(2,2)*eye(4);
```

run sim

```
t_max = 50;    % [sec] sim time
out_A = sim('hw3_rw', t_max);
out_B = sim('hw3_cmg', t_max);
```

unpack data: A

```
A_t = squeeze(out_A.tout)';

A_orbit.w_lvlh = squeeze(out_A.w_LVLH_ECI)';
A_orbit.q_lvlh = squeeze(out_A.q_LVLH_ECI)';
A_orbit.eul_lvlh = squeeze(out_A.eul_LVLH_ECI)';

A_orbit.dist_torque_lvlh = squeeze(out_A.dist_torque)';

A_sat.w_lvlh = squeeze(out_A.w_Body_LVLH)';
A_sat.q_lvlh = squeeze(out_A.q_Body_LVLH)';
A_sat.eul_lvlh = squeeze(out_A.eul_Body_LVLH)';

A_sat.w_eci = squeeze(out_A.w_Body_ECI)';
A_sat.q_eci = squeeze(out_A.q_Body_ECI)';
A_sat.eul_eci = squeeze(out_A.eul_Body_ECI)';

A_command_tq = squeeze(out_A.command_torque)';
```

unpack data: B

```
B_t = squeeze(out_B.tout)';

B_orbit.w_lvlh = squeeze(out_B.w_LVLH_ECI)';
B_orbit.q_lvlh = squeeze(out_B.q_LVLH_ECI)';
B_orbit.eul_lvlh = squeeze(out_B.eul_LVLH_ECI)';

B_orbit.dist_torque_lvlh = squeeze(out_B.dist_torque)';

B_sat.w_lvlh = squeeze(out_B.w_Body_LVLH)';
B_sat.q_lvlh = squeeze(out_B.q_Body_LVLH)';
B_sat.eul_lvlh = squeeze(out_B.eul_Body_LVLH)';

B_sat.w_eci = squeeze(out_B.w_Body_ECI)';
B_sat.q_eci = squeeze(out_B.q_Body_ECI)';
B_sat.eul_eci = squeeze(out_B.eul_Body_ECI)';

B_command_gimbal = squeeze(out_B.gamma)';
```

plot data: A

```
figure

subplot(3,3,1)
hold on
plot(A_t, A_sat.q_lvlh(:,1))
plot(A_t, A_sat.q_lvlh(:,2))
plot(A_t, A_sat.q_lvlh(:,3))
plot(A_t, A_sat.q_lvlh(:,4))
hold off
grid on
```

```
title('Quaternion from Body to LVLH')
xlabel('Time [sec]')
ylabel('Quaternion')
legend('\epsilon_1','\epsilon_2','\epsilon_3','\eta')

subplot(3,3,2)
hold on
plot(A_t, A_orbit.q_lvlh(:,1))
plot(A_t, A_orbit.q_lvlh(:,2))
plot(A_t, A_orbit.q_lvlh(:,3))
plot(A_t, A_orbit.q_lvlh(:,4))
hold off
grid on
title('Quaternion from LVLH to ECI')
xlabel('Time [sec]')
ylabel('Quaternion')
legend('\epsilon_1','\epsilon_2','\epsilon_3','\eta')

subplot(3,3,3)
hold on
plot(A_t, A_sat.q_eci(:,1))
plot(A_t, A_sat.q_eci(:,2))
plot(A_t, A_sat.q_eci(:,3))
plot(A_t, A_sat.q_eci(:,4))
hold off
grid on
title('Quaternion from Body to ECI')
xlabel('Time [sec]')
ylabel('Quaternion')
legend('\epsilon_1','\epsilon_2','\epsilon_3','\eta')

subplot(3,3,4)
hold on
plot(A_t, A_sat.eul_lvlh(:,1))
plot(A_t, A_sat.eul_lvlh(:,2))
plot(A_t, A_sat.eul_lvlh(:,3))
hold off
grid on
title('Euler Angles from Body to LVLH')
xlabel('Time [sec]')
ylabel('Euler Angle [rad]')
legend('\phi', '\theta', '\psi')

subplot(3,3,5)
hold on
plot(A_t, A_orbit.eul_lvlh(:,1))
plot(A_t, A_orbit.eul_lvlh(:,2))
plot(A_t, A_orbit.eul_lvlh(:,3))
hold off
grid on
title('Euler Angles from LVLH to ECI')
xlabel('Time [sec]')
ylabel('Euler Angle [rad]')
legend('\phi', '\theta', '\psi')
```

```
subplot(3,3,6)
hold on
plot(A_t, A_sat.eul_eci(:,1))
plot(A_t, A_sat.eul_eci(:,2))
plot(A_t, A_sat.eul_eci(:,3))
hold off
grid on
title('Euler Angles from Body to ECI')
xlabel('Time [sec]')
ylabel('Euler Angle [rad]')
legend('\phi', '\theta', '\psi')


subplot(3,3,7)
hold on
plot(A_t, A_sat.w_lvlh(:,1))
plot(A_t, A_sat.w_lvlh(:,2))
plot(A_t, A_sat.w_lvlh(:,3))
hold off
grid on
title('\omega from Body to LVLH')
xlabel('Time [sec]')
ylabel('Rate [rad/s]')
legend('\omega_x', '\omega_y', '\omega_z')

subplot(3,3,8)
hold on
plot(A_t, A_orbit.w_lvlh(:,1))
plot(A_t, A_orbit.w_lvlh(:,2))
plot(A_t, A_orbit.w_lvlh(:,3))
hold off
grid on
title('\omega from LVLH to ECI')
xlabel('Time [sec]')
ylabel('Rate [rad/s]')
legend('\omega_x', '\omega_y', '\omega_z')

subplot(3,3,9)
hold on
plot(A_t, A_sat.w_eci(:,1))
plot(A_t, A_sat.w_eci(:,2))
plot(A_t, A_sat.w_eci(:,3))
hold off
grid on
title('\omega from LVLH to ECI')
xlabel('Time [sec]')
ylabel('Rate [rad/s]')
legend('\omega_x', '\omega_y', '\omega_z')

sgtitle('Reaction Wheels')
```

plot data: B

```
figure

subplot(3,3,1)
hold on
plot(B_t, B_sat.q_lvlh(:,1))
plot(B_t, B_sat.q_lvlh(:,2))
plot(B_t, B_sat.q_lvlh(:,3))
plot(B_t, B_sat.q_lvlh(:,4))
hold off
grid on
title('Quaternion from Body to LVLH')
xlabel('Time [sec]')
ylabel('Quaternion')
legend('\epsilon_1','\epsilon_2','\epsilon_3','\eta')

subplot(3,3,2)
hold on
plot(B_t, B_orbit.q_lvlh(:,1))
plot(B_t, B_orbit.q_lvlh(:,2))
plot(B_t, B_orbit.q_lvlh(:,3))
plot(B_t, B_orbit.q_lvlh(:,4))
hold off
grid on
title('Quaternion from LVLH to ECI')
xlabel('Time [sec]')
ylabel('Quaternion')
legend('\epsilon_1','\epsilon_2','\epsilon_3','\eta')

subplot(3,3,3)
hold on
plot(B_t, B_sat.q_eci(:,1))
plot(B_t, B_sat.q_eci(:,2))
plot(B_t, B_sat.q_eci(:,3))
plot(B_t, B_sat.q_eci(:,4))
hold off
grid on
title('Quaternion from Body to ECI')
xlabel('Time [sec]')
ylabel('Quaternion')
legend('\epsilon_1','\epsilon_2','\epsilon_3','\eta')

subplot(3,3,4)
hold on
plot(B_t, B_sat.eul_lvlh(:,1))
plot(B_t, B_sat.eul_lvlh(:,2))
plot(B_t, B_sat.eul_lvlh(:,3))
hold off
grid on
title('Euler Angles from Body to LVLH')
xlabel('Time [sec]')
ylabel('Euler Angle [rad]')
```

```
legend('\phi', '\theta', '\psi')

subplot(3,3,5)
hold on
plot(B_t, B_orbit.eul_lvlh(:,1))
plot(B_t, B_orbit.eul_lvlh(:,2))
plot(B_t, B_orbit.eul_lvlh(:,3))
hold off
grid on
title('Euler Angles from LVLH to ECI')
xlabel('Time [sec]')
ylabel('Euler Angle [rad]')
legend('\phi', '\theta', '\psi')

subplot(3,3,6)
hold on
plot(B_t, B_sat.eul_eci(:,1))
plot(B_t, B_sat.eul_eci(:,2))
plot(B_t, B_sat.eul_eci(:,3))
hold off
grid on
title('Euler Angles from Body to ECI')
xlabel('Time [sec]')
ylabel('Euler Angle [rad]')
legend('\phi', '\theta', '\psi')


subplot(3,3,7)
hold on
plot(B_t, B_sat.w_lvlh(:,1))
plot(B_t, B_sat.w_lvlh(:,2))
plot(B_t, B_sat.w_lvlh(:,3))
hold off
grid on
title('\omega from Body to LVLH')
xlabel('Time [sec]')
ylabel('Rate [rad/s]')
legend('\omega_x', '\omega_y', '\omega_z')

subplot(3,3,8)
hold on
plot(B_t, B_orbit.w_lvlh(:,1))
plot(B_t, B_orbit.w_lvlh(:,2))
plot(B_t, B_orbit.w_lvlh(:,3))
hold off
grid on
title('\omega from LVLH to ECI')
xlabel('Time [sec]')
ylabel('Rate [rad/s]')
legend('\omega_x', '\omega_y', '\omega_z')

subplot(3,3,9)
hold on
plot(B_t, B_sat.w_eci(:,1))
```

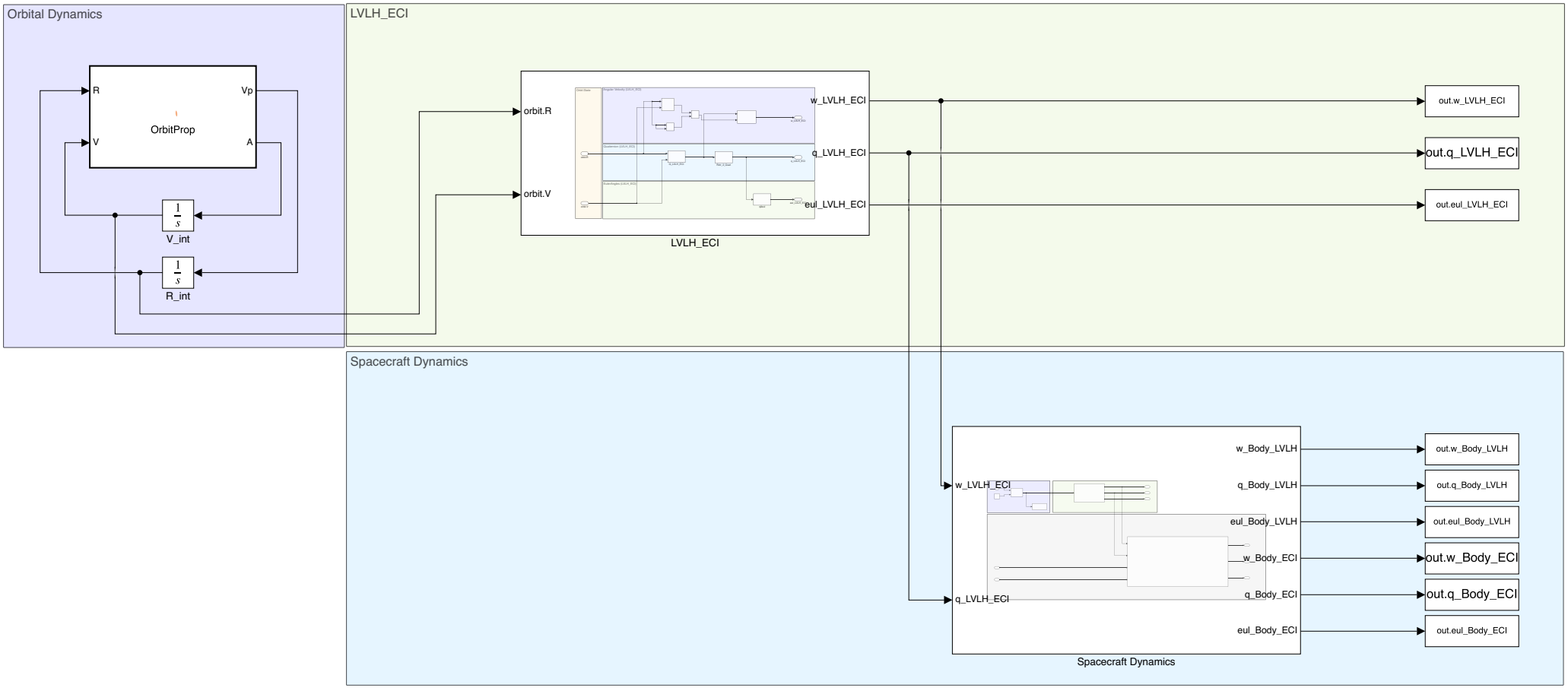
```
plot(B_t, B_sat.w_eci(:,2))
plot(B_t, B_sat.w_eci(:,3))
hold off
grid on
title('\omega from LVLH to ECI')
xlabel('Time [sec]')
ylabel('Rate [rad/s]')
legend('\omega_x', '\omega_y', '\omega_z')

sgtitle('Control Moment Gyros')
```

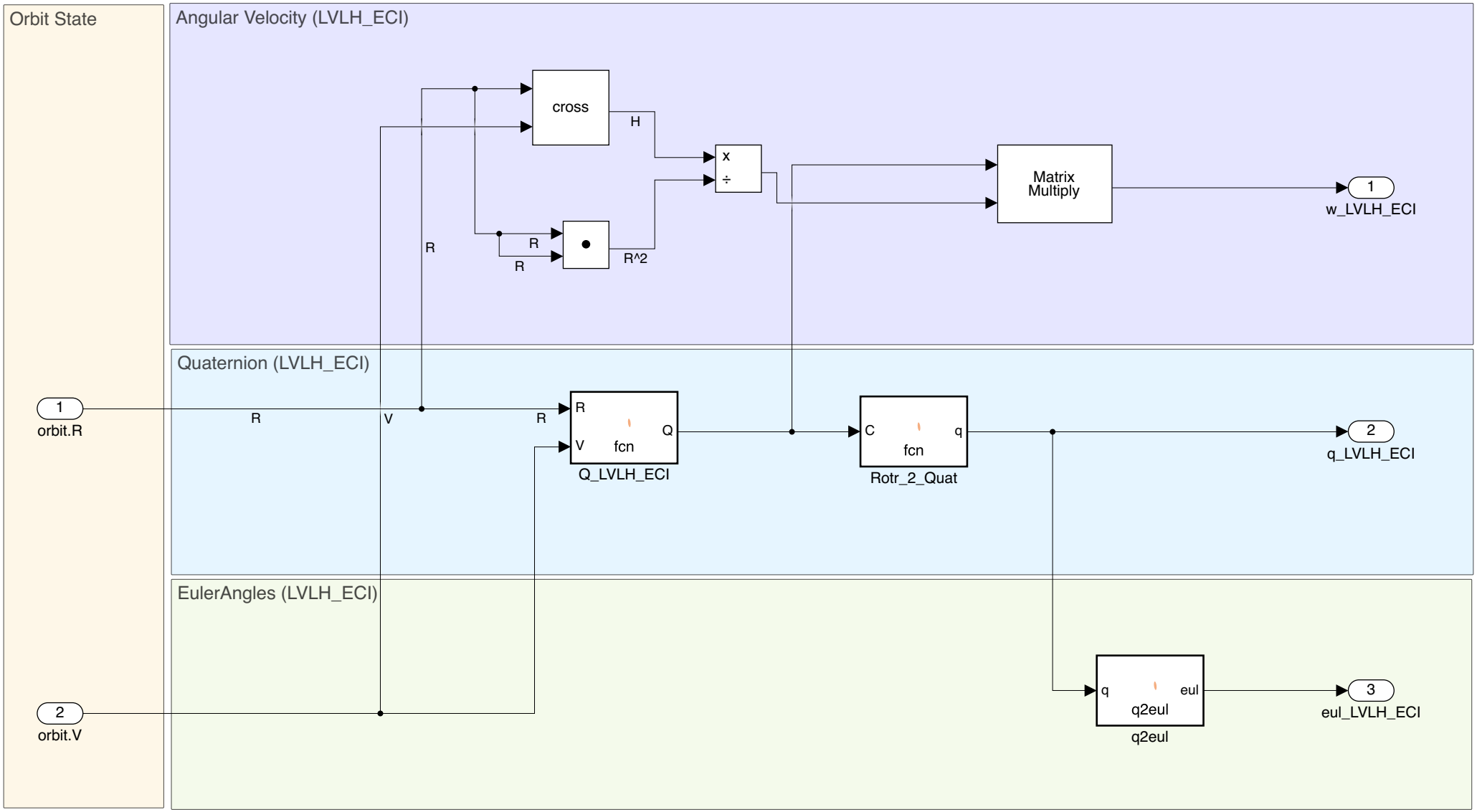
plot misc

```
figure
hold on
plot(A_t, vecnorm(A_command_tq, 2, 2))
hold off
title('RW Torque')
xlabel('Time [sec]')
ylabel('Torque [Nm]')


figure
hold on
plot(B_t, B_command_gimbal(:,1))
plot(B_t, B_command_gimbal(:,2))
plot(B_t, B_command_gimbal(:,3))
plot(B_t, B_command_gimbal(:,4))
hold off
title('Gimbal Angles: CMG')
xlabel('Time [sec]')
ylabel('Gimbal Angle [rad]')
legend('1', '2', '3', '4')



function eul = quat_eul(q)

    n = q(4);
    e = q(1:3);

    q = [n, e(1), e(2), e(3)];

    phi = atan2(2*(q(1)*q(2) + q(3)*q(4)), 1 - 2*(q(2)^2 + q(3)^2));
    theta = asin(2*(q(1)*q(3) - q(4)*q(2)));
    psi = atan2(2*(q(1)*q(4) + q(2)*q(3)), 1-2*(q(3)^2 + q(4)^2));

    eul = [phi; theta; psi];

end

function wx = skewSymmetric(w)
        wx = [0, -1*w(3), w(2);
```

```
            w(3), 0, -1*w(1);
            -1*w(2), w(1), 0];
end

function Ad = diagA(A)
    [~,x] = size(A);
    Ad = zeros(x, 3*x);
    for i = 1:x
        Ad(i,3*(i-1)+1:3*(i-1)+3) = A(:,1)';
    end
end
```

Published with MATLAB® R2022b


Orbital Dynamics
R
Vp
OrbitProp
V
A
1/s
V_int
1/s
R_int
LVLH_ECI
orbit.R
orbit.V
w_LVLH_ECI
q_LVLH_ECI
eul_LVLH_ECI
LVLH_ECI
out.w_LVLH_ECI
out.q_LVLH_ECI
out.eul_LVLH_ECI
Spacecraft Dynamics
w_LVLH_ECI
q_LVLH_ECI
w_Body_LVLH
q_Body_LVLH
eul_Body_LVLH
w_Body_ECI
q_Body_ECI
eul_Body_ECI
Spacecraft Dynamics
out.w_Body_LVLH
out.q_Body_LVLH
out.eul_Body_LVLH
out.w_Body_ECI
out.q_Body_ECI
out.eul_Body_ECI

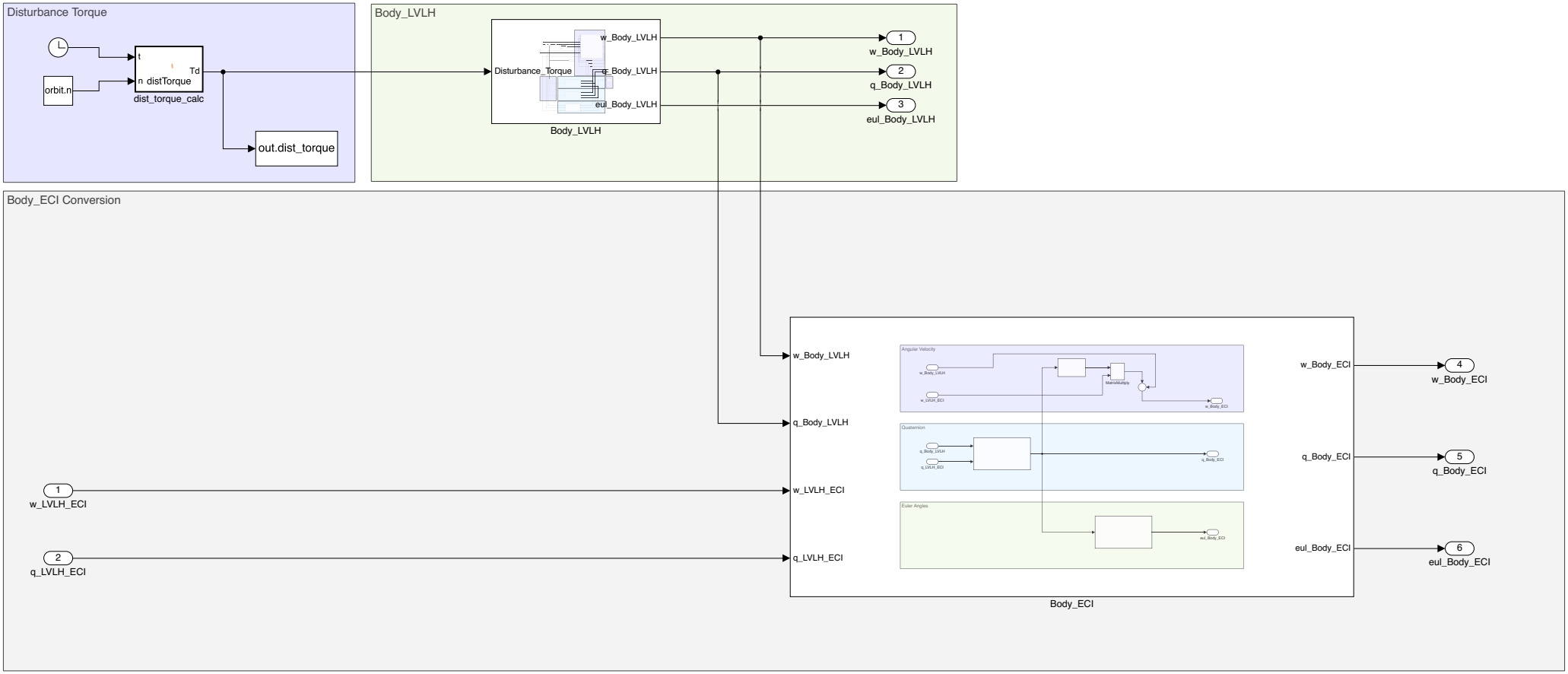

Orbit State
Angular Velocity (LVLH_ECI)
cross
H
x
÷
R
R
R
R^2
Matrix
Multiply
1
w_LVLH_ECI
Quaternion (LVLH_ECI)
1
orbit.R
R
V
R
R
V
fcn
Q
Q_LVLH_ECI
C
fcn
q
Rotr_2_Quat
2
q_LVLH_ECI
EulerAngles (LVLH_ECI)
2
orbit.V
q
q2eul
eul
q2eul
3
eul_LVLH_ECI

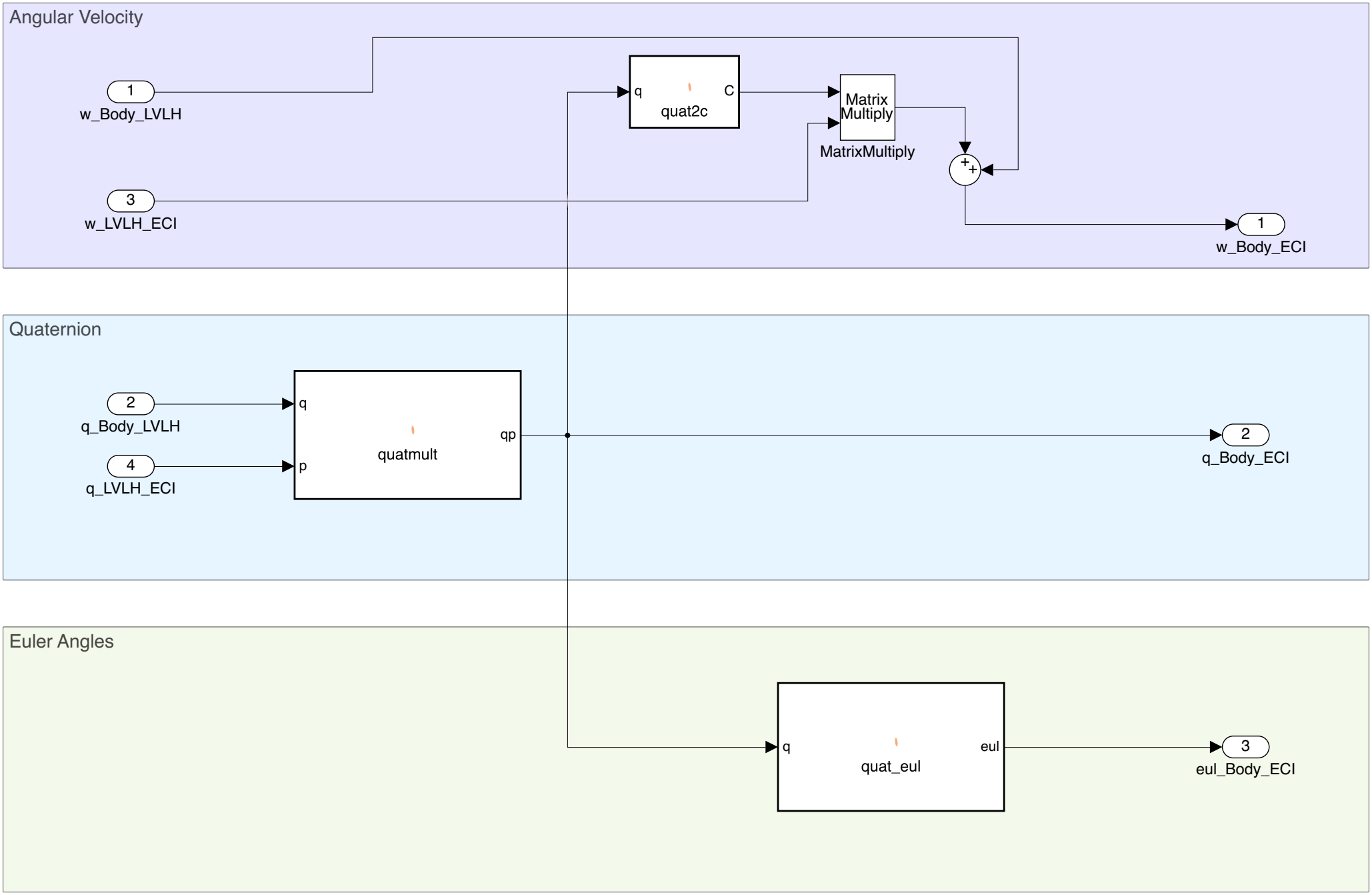
```
function Q = fcn(R,V)


z = -1*R / norm(R);
y = -1 * cross(R,V) / norm(cross(R,V));

x = cross(y,z);

Q = [x,y,z]';
```

```
function q = fcn(C)

        e = zeros(3,1);
        n = 0.5 * sqrt(1 + trace(C));
        e(1) = 0.25 * (C(2,3) - C(3,2))/n;
        e(2) = 0.25 * (C(3,1) - C(1,3))/n;
        e(3) = 0.25 * (C(1,2) - C(2,1))/n;

        q = [e;n];
```

```
function eul = q2eul(q)

    n = q(4);
    e = q(1:3);

    q = [n, e(1), e(2), e(3)];

    phi = atan2(2*(q(1)*q(2) + q(3)*q(4)), 1 - 2*(q(2)^2 + q(3)^2));
    theta = asin(2*(q(1)*q(3) - q(4)*q(2)));
    psi = atan2(2*(q(1)*q(4) + q(2)*q(3)), 1-2*(q(3)^2 + q(4)^2));

    eul = [phi; theta; psi];
```

```
function [Vp,A] = OrbitProp(R,V)

    mu = 398600;

    rad = norm(R);
    rx = R(1);
    ry = R(2);
    rz = R(3);

    ax = -mu*rx/rad^3;
    ay = -mu*ry/rad^3;
    az = -mu*rz/rad^3;

    Vp = [V(1);V(2);V(3)];
    A = [ax;ay;az];

end
```

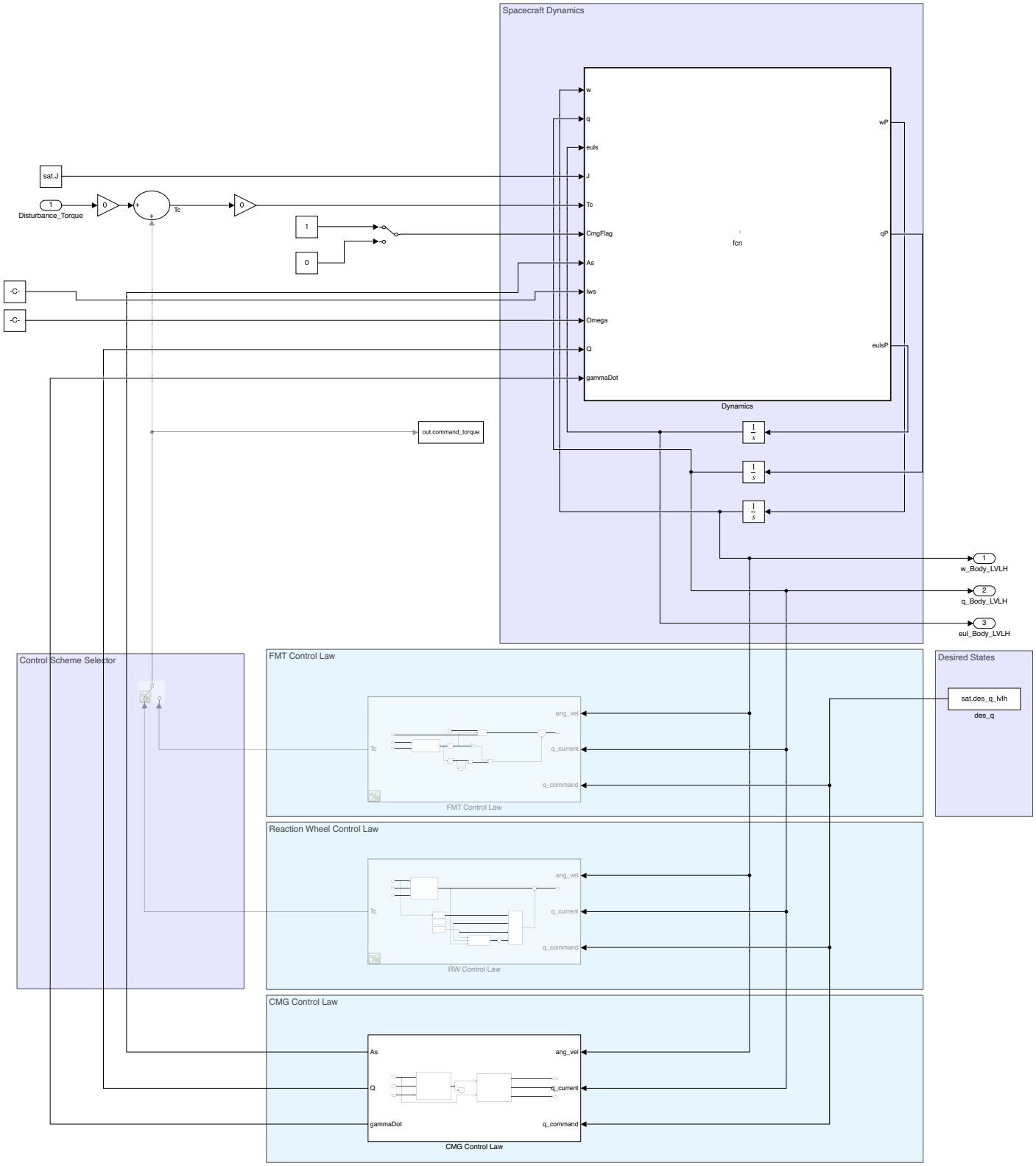

Disturbance Torque
t
orbit.n
n
distTorque
Td
dist_torque_calc
out.dist_torque
Body_LVLH
Disturbance_Torque
w_Body_LVLH
q_Body_LVLH
eul_Body_LVLH
Body_LVLH
1
w_Body_LVLH
2
q_Body_LVLH
3
eul_Body_LVLH
Body_ECI Conversion
1
w_LVLH_ECI
2
q_LVLH_ECI
w_Body_LVLH
q_Body_LVLH
w_LVLH_ECI
q_LVLH_ECI
Angular Velocity
Quaternion
Euler Angles
w_Body_ECI
q_Body_ECI
eul_Body_ECI
Body_ECI
4
w_Body_ECI
5
q_Body_ECI
6
eul_Body_ECI

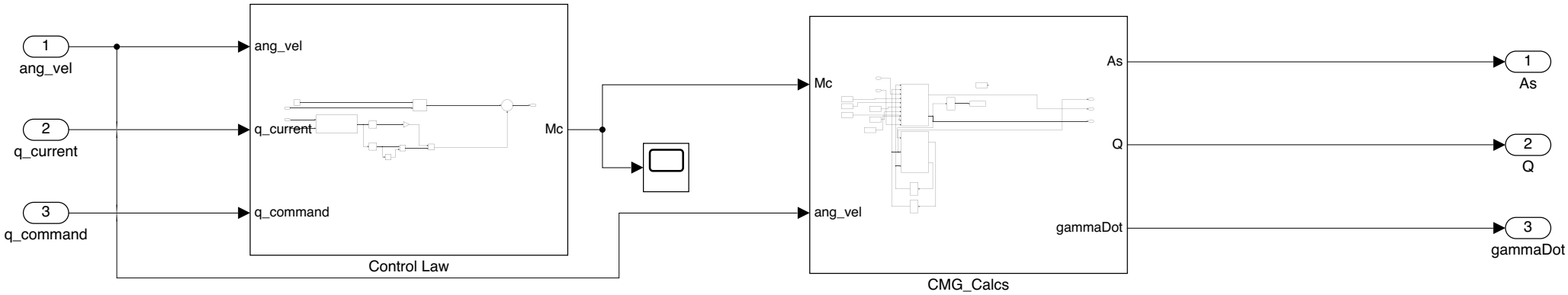

Angular Velocity
1
w_Body_LVLH
q
quat2c
C
Matrix
Multiply
MatrixMultiply
3
w_LVLH_ECI
1
w_Body_ECI
Quaternion
2
q_Body_LVLH
4
q_LVLH_ECI
q
p
quatmult
qp
2
q_Body_ECI
Euler Angles
q
quat_eul
eul
3
eul_Body_ECI

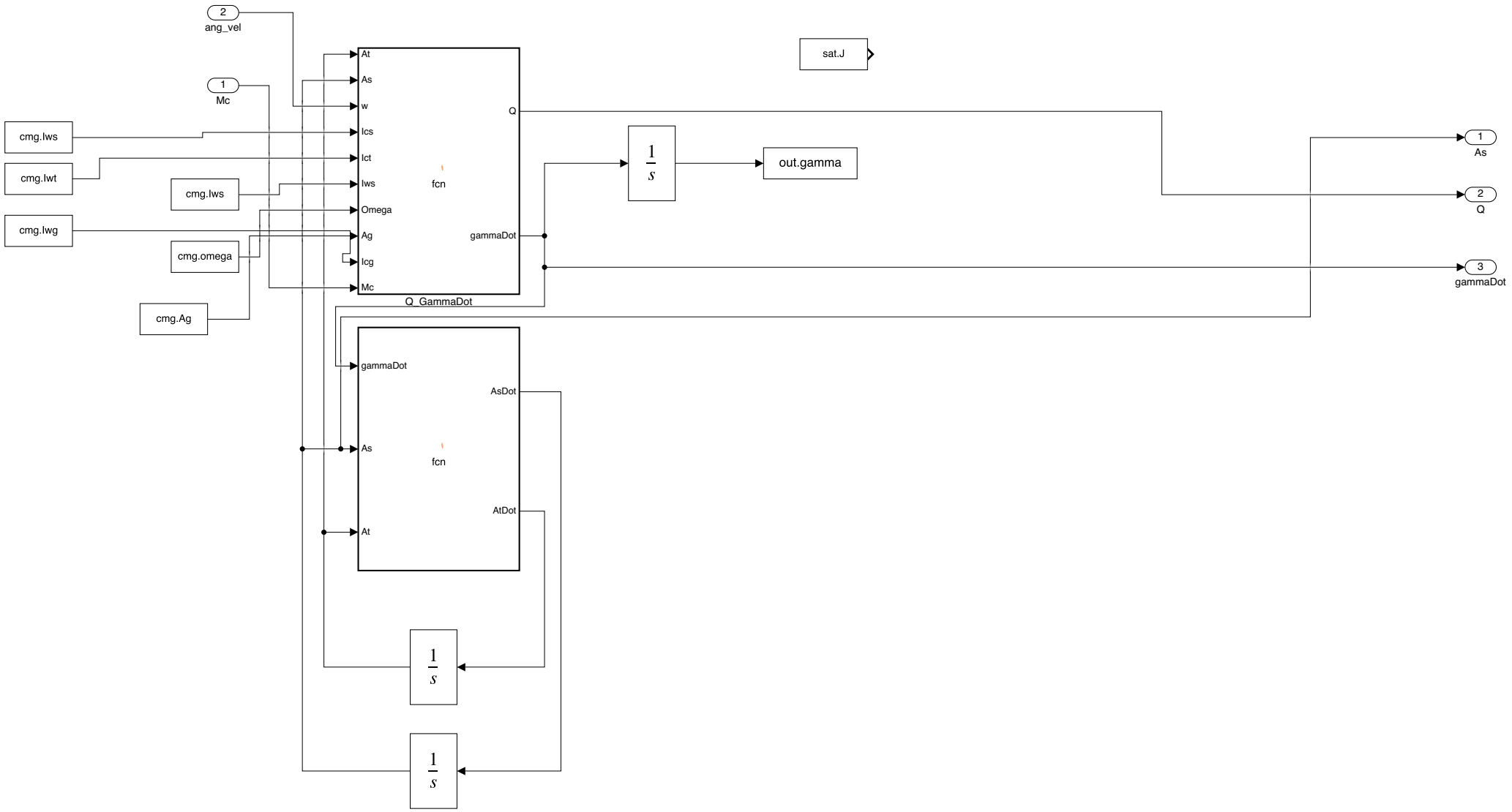
```
function qp = quatmult(q, p)

        function wx = skewSymmetric(w)
            wx = [0, -1*w(3), w(2);
                  w(3), 0, -1*w(1);
                  -1*w(2), w(1), 0];
        end

        qn = q(4);
        qe = q(1:3);

        pn = p(4);
        pe = p(1:3);

        n = pn * qn - pe'*qe;
        e = pn * qe + qn*pe + skewSymmetric(pe)*qe;

        qp = [e(1);e(2);e(3);n];

end
```

```
function eul = quat_eul(q)

    n = q(4);
    ex = q(1);
    ey = q(2);
    ez = q(3);

    a = 2*(n*ey - ez*ex);
    if a > 1
        a = 1;
    elseif a < -1
        a = -1;
    end

    phi = atan2(2*(n*ex + ey*ez), 1 - 2*(ex^2 + ey^2));
    theta = asin(a);
    psi = atan2(2*(n*ez + ex*ey), 1 - 2*(ey^2 + ez^2));

    eul = [phi;theta;psi];

end
```

```
function C = quat2c(q)
C = quat2rotm([q(4);q(1:3)]');
```

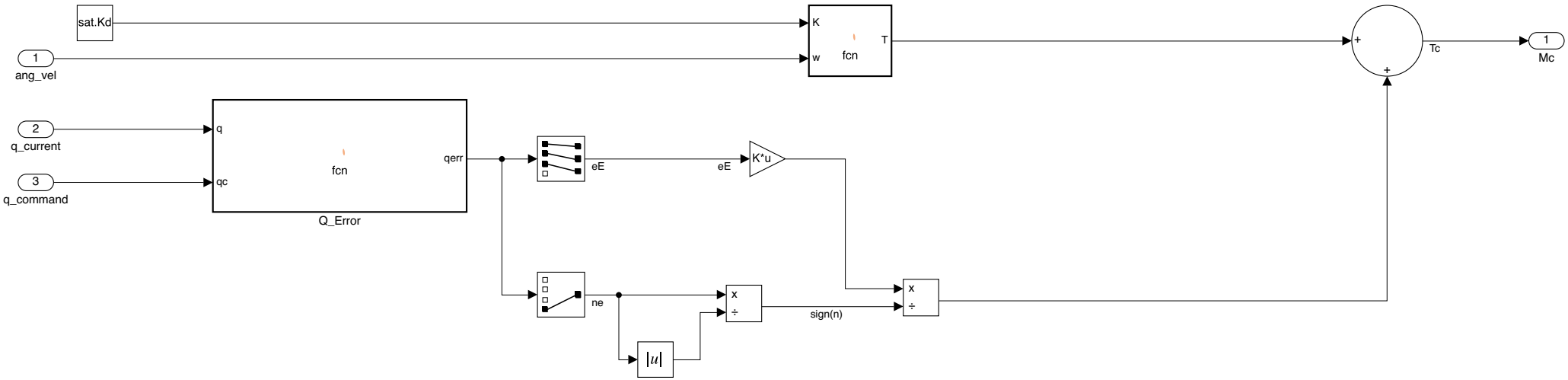

Spacecraft Dynamics
w
q
euls
J
Tc
CmgFlag
As
lws
Omega
Q
gammaDot
wP
qP
eulsP
fcn
Dynamics
sat.J
1
Disturbance_Torque
0
Tc
0
1
0
-C-
-C-
out.command_torque
1/s
1/s
1/s
1
w_Body_LVLH
2
q_Body_LVLH
3
eul_Body_LVLH
Control Scheme Selector
FMT Control Law
Tc
ang_vel
q_current
q_command
FMT Control Law
Desired States
sat.des_q_lvlh
des_q
Reaction Wheel Control Law
Tc
ang_vel
q_current
q_command
RW Control Law
CMG Control Law
As
Q
gammaDot
ang_vel
q_current
q_command
CMG Control Law


1
ang_vel
2
q_current
3
q_command
ang_vel
q_current
q_command
Mc
Control Law
Mc
ang_vel
As
Q
gammaDot
CMG_Calcs
1
As
2
Q
3
gammaDot


2
ang_vel
1
Mc
cmg.Ig
cmg.Iwt
cmg.Iws
cmg.Iwg
cmg.omega
cmg.Ag
At
As
w
Ics
Ict
Iws
Omega
Ag
Icg
Mc
Q
fcn
gammaDot
Q_GammaDot
sat.J
1
s
out.gamma
1
As
2
Q
3
gammaDot
gammaDot
AsDot
As
fcn
AtDot
At
1
s
1
s

```
function [AsDot,AtDot] = fcn(gammaDot,As,At)

    AtDot = -1*As*diag(gammaDot);
    AsDot = At * diag(gammaDot);
```

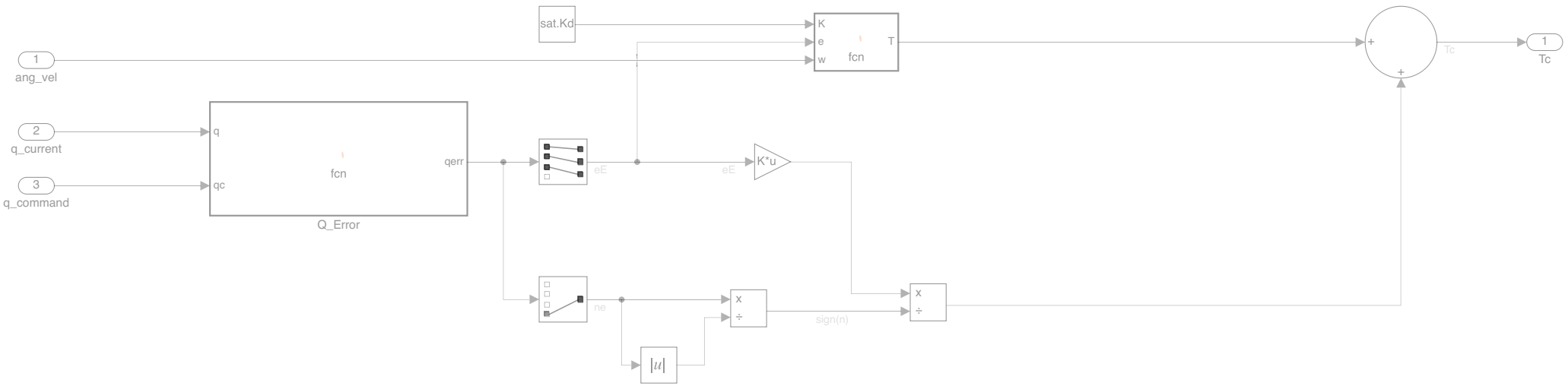
```
function [Q, gammaDot] = fcn(At, As, w, Ics, Ict, Iws, Omega, Ag, Icg, Mc)

    function Ad = a_diag(A)
        [x, ~] = size(A);
        X = zeros(x, 3*x);
        for i = 1:x
            X(i, 3*(i-1)+1:3*(i-1)+3) = A(i,:)';
        end
        Ad = X;
    end

    function wx = skewSymmetric(w)
        wx = [0, -1*w(3), w(2);
             w(3), 0, -1*w(1);
             -1*w(2), w(1), 0];
    end

Astd = a_diag(As'); % [As']^d
Attd = a_diag(At'); % [At']^d

a = At*Astd + As*Attd;
b = At * Iws * diag(Omega);
c = skewSymmetric(w)*Ag*Icg;

Q = (a * (kron(eye(4),w)) * (Ics - Ict)) + b + c;

%Q = At*Iws*diag(Omega);

singular = 0.01 * exp(-1*det(Q*Q'));

Qt = inv(Q*Q' + singular*eye(3));

gammaDot = Q'*Qt*Mc;

end
```


sat.Kd
1
ang_vel
K
w
fcn
T
+
+
Tc
1
Mc
2
q_current
3
q_command
q
qc
fcn
Q_Error
qerr
eE
eE
K*u
ne
x
÷
sign(n)
x
÷
|u|

```
function T = fcn(K, w)
T = -1*K*w;
```

```
function qerr = fcn(q, qc)

%     function wx = skewSymmetric(w)
%         wx = [0, -1*w(3), w(2);
%               w(3), 0, -1*w(1);
%               -1*w(2), w(1), 0];
%     end

    function qp = quatmult(q, p)

        function wx = skewSymmetric(w)
            wx = [0, -1*w(3), w(2);
                  w(3), 0, -1*w(1);
                  -1*w(2), w(1), 0];
        end

        qn = q(4);
        qe = q(1:3);

        pn = p(4);
        pe = p(1:3);

        n = pn * qn - pe'*qe;
        e = pn * qe + qn*pe + skewSymmetric(pe)*qe;

        qp = [e(1);e(2);e(3);n];

    end

qc(1:3) = -1*qc(1:3);
qerr = quatmult(qc, q);

end
```

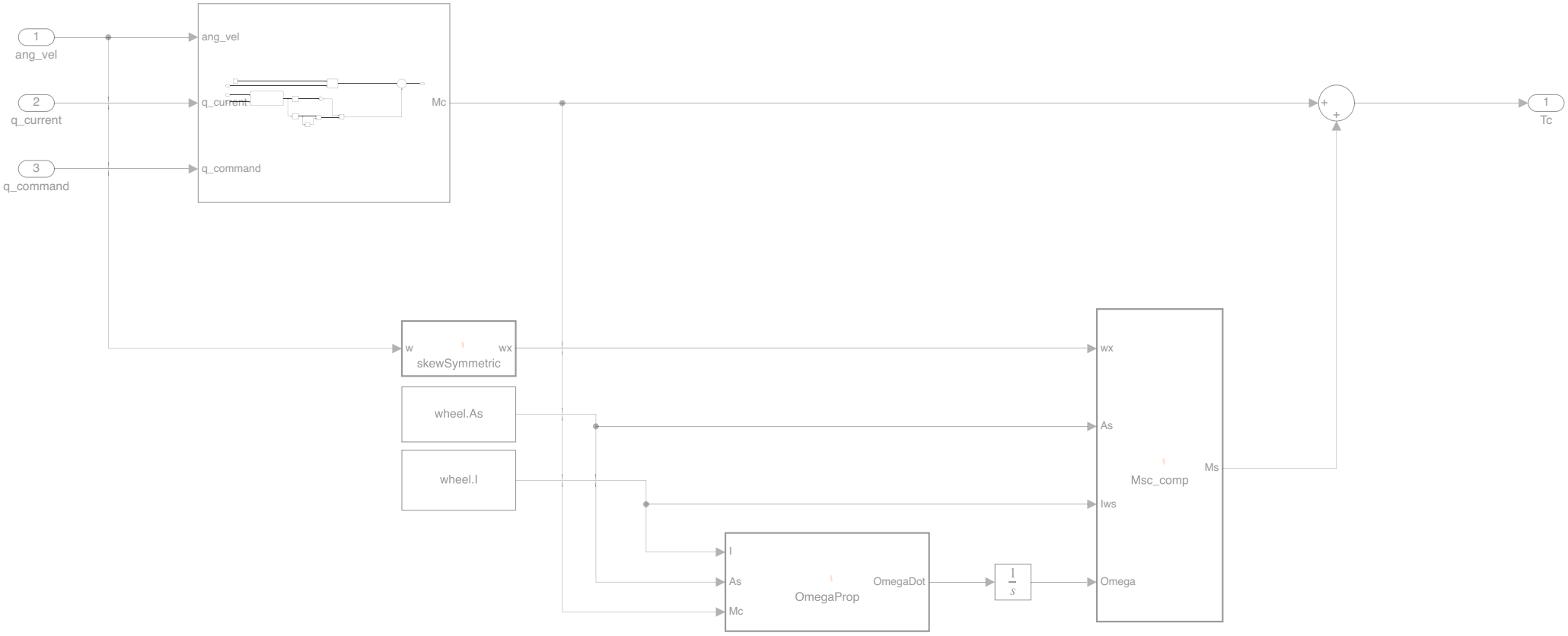
```
function [wP, qP, eulsP] = fcn(w, q, euls, J, Tc, CmgFlag, As, Iws, Omega, Q, gammaDot)

    function wx = skewSymmetric(w)
        wx = [0, -1*w(3), w(2);
             w(3), 0, -1*w(1);
             -1*w(2), w(1), 0];
    end

    T = [Tc(1);Tc(2);Tc(3)];

    if CmgFlag == 1

        rhs = skewSymmetric(w)*(J*w + As * Iws * Omega) - Q*gammaDot;
        wP = J\(T - rhs);

    else

        wP = J\(T - skewSymmetric(w)*J*w);

    end

    e = q(1:3);
    n = q(4);
    eP = 0.5*(n*eye(3) + skewSymmetric(e))*w;
    nP = -0.5*e'*w;
    qP = [eP;nP];

    phi = euls(1);
    theta = euls(2);
    psi = euls(3);
    eulsP = 1/(cos(theta)) * [cos(theta), sin(phi)*sin(theta), cos(phi)*sin(theta);
                              0, cos(phi)*cos(theta), -1*sin(phi)*cos(theta);
                              0, sin(phi), cos(phi)] * w;


end
```


sat.Kd
K
e
w
fcn
T
1
ang_vel
+
+
Tc
1
Tc
2
q_current
3
q_command
q
qc
fcn
qerr
Q_Error
eE
eE
K*u
ne
x
÷
sign(n)
x
÷
|u|

%

```
function T = fcn(K, e, w)
T = -1*K*(1 + e'*e)*w;
```

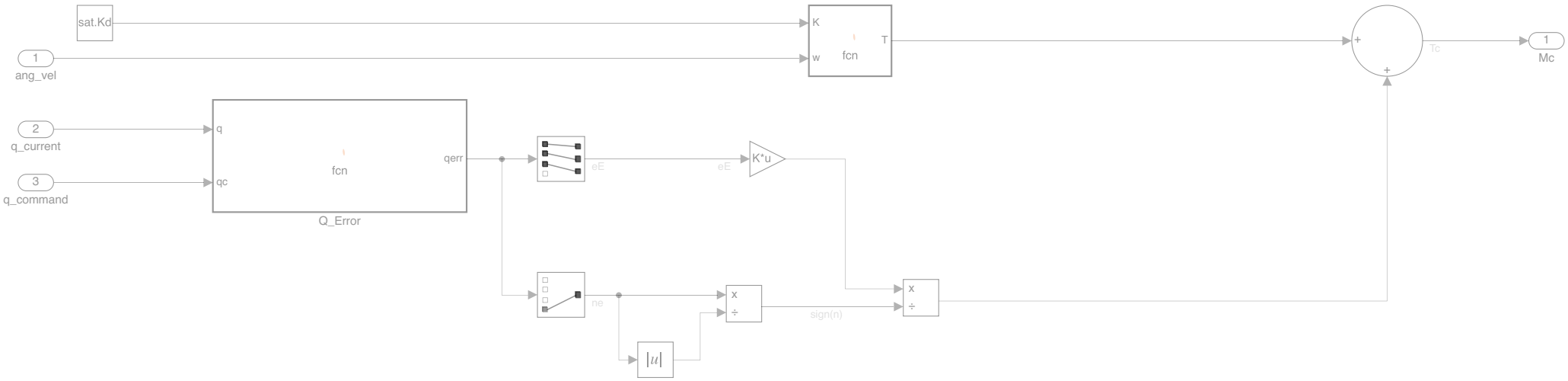
```
function qerr = fcn(q, qc)

%     function wx = skewSymmetric(w)
%         wx = [0, -1*w(3), w(2);
%               w(3), 0, -1*w(1);
%               -1*w(2), w(1), 0];
%     end

    function qp = quatmult(q, p)

        function wx = skewSymmetric(w)
            wx = [0, -1*w(3), w(2);
                  w(3), 0, -1*w(1);
                  -1*w(2), w(1), 0];
        end

        qn = q(4);
        qe = q(1:3);

        pn = p(4);
        pe = p(1:3);

        n = pn * qn - pe'*qe;
        e = pn * qe + qn*pe + skewSymmetric(pe)*qe;

        qp = [e(1);e(2);e(3);n];

    end

qc(1:3) = -1*qc(1:3);
qerr = quatmult(qc, q);

end
```


1
ang_vel
2
q_current
3
q_command
ang_vel
q_current
q_command
Mc
1
Tc
w
wx
skewSymmetric
wheel.As
wheel.I
wx
As
Ms
Msc_comp
Iws
I
As
Mc
OmegaProp
OmegaDot
1
s
Omega

```
function wx = skewSymmetric(w)
        wx = [0, -1*w(3), w(2);
             w(3), 0, -1*w(1);
             -1*w(2), w(1), 0];
end
```

```
function Ms = Msc_comp(wx, As, Iws, Omega)
Ms = wx*As*Iws*Omega;
end
```

```
function OmegaDot = OmegaProp(I, As, Mc)
OmegaDot = I\pinv(As)*Mc;
end
```


sat.Kd
1
ang_vel
K
w
fcn
T
+
+
Tc
1
Mc
2
q_current
3
q_command
q
qc
fcn
qerr
Q_Error
eE
eE
K*u
ne
x
÷
sign(n)
x
÷
|u|

```
function T = fcn(K, w)
T = -1*K*w;
```

```
function qerr = fcn(q, qc)

%     function wx = skewSymmetric(w)
%         wx = [0, -1*w(3), w(2);
%               w(3), 0, -1*w(1);
%               -1*w(2), w(1), 0];
%     end

    function qp = quatmult(q, p)

        function wx = skewSymmetric(w)
            wx = [0, -1*w(3), w(2);
                  w(3), 0, -1*w(1);
                  -1*w(2), w(1), 0];
        end

        qn = q(4);
        qe = q(1:3);

        pn = p(4);
        pe = p(1:3);

        n = pn * qn - pe'*qe;
        e = pn * qe + qn*pe + skewSymmetric(pe)*qe;

        qp = [e(1);e(2);e(3);n];

    end

qc(1:3) = -1*qc(1:3);
qerr = quatmult(qc, q);

end
```

```
function Td  = distTorque(t, n)

T = sin(3*n*t)*[0;0.5;0]*10^(-3);
Td = T;
end
```