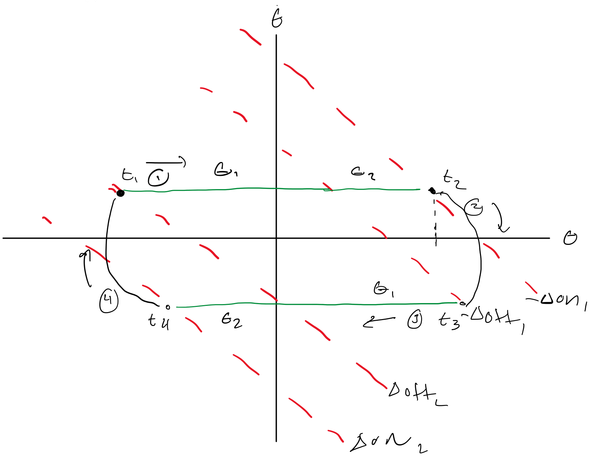
Homework 5

Sunday, November 13, 2022 8:34 PM

Gagandeep Thapar

1) a) Duty cycle of switch = 100% (never turns off)

b) Schmitt Trigger



$\Delta on_1 \sim e + \tau\dot{e} = \theta + \tau\dot{\theta}$

$\Delta off_1 \sim e + \tau\dot{e} = \theta - \tau\dot{\theta}$

$\Delta on_2 \sim e + \tau\dot{e} = -\theta - \tau\dot{\theta} = -\Delta on$

$\Delta off_2 \sim e + \tau\dot{e} = -\theta + \tau\dot{\theta} = -\Delta off$

recall: $T = J\ddot{\theta} \quad \therefore \ddot{\theta} = \frac{T}{J}$ where $T = m$ $\quad \ddot{\theta} = \frac{m}{J}$

notice: $\Delta on + \Delta off = \theta + \tau\dot{\theta} + \theta - \tau\dot{\theta}$

$2\theta = \Delta on + \Delta off$

$\theta = \frac{\Delta on + \Delta off}{2}$

$\Delta on - \Delta off = \theta + \tau\dot{\theta} - \theta + \tau\dot{\theta}$

$2\tau\dot{\theta} = \Delta on - \Delta off$

$\dot{\theta} = \frac{\Delta on - \Delta off}{2\tau}$

notice: $t_{rest} = t_1 + t_3$ and $t_1 = t_3$

$t_{thrust} = t_2 + t_4$ and $t_2 = t_4$

$\therefore t_{rest} = 2t_1$

$t_{thrust} = 2t_2$

$DC = \frac{t_{thrust}}{t_{rest} + t_{thrust}} = \frac{2t_2}{2t_1 + 2t_2} = \frac{t_2}{t_1 + t_2}$

notice: $\theta(t) = \theta_0 + \dot{\theta}(t)$

$\theta(t_1) = -\theta + \dot{\theta}t_1 + \frac{1}{2}\ddot{\theta}t_1^2$

$\dot{\theta}t_1 = 2\theta$

$t_1 = \frac{-2\theta}{\dot{\theta}}$

$= \frac{2\left(\frac{\Delta on + \Delta off}{2}\right)}{\frac{\Delta on - \Delta off}{2\tau}} = \frac{2\tau(\Delta on - \Delta off)}{\Delta on - \Delta off} = t_1$

notice: $\dot{\theta}(t) = \dot{\theta}_0 + \ddot{\theta}t$

$\dot{\theta}(t_2) = \dot{\theta} + \ddot{\theta}t_2 = -\dot{\theta}$

$\ddot{\theta}t_2 = -2\dot{\theta}$

$t_2 = \frac{-2\dot{\theta}}{\ddot{\theta}}$

$= \frac{-2\left(\frac{\Delta on - \Delta off}{2\tau}\right)}{\frac{m}{J}} = \frac{-(\Delta on - \Delta off)}{\tau\frac{m}{J}} = t_2$

$DC = \frac{t_2}{t_1 + t_2}$

$= \frac{\Delta on - \Delta off}{\tau\frac{m}{J}} \cdot \left(\frac{\Delta on - \Delta off}{\tau\frac{m}{J}} + \frac{2\tau(\Delta on + \Delta off)}{\Delta on - \Delta off}\right)^{-1}$

$= \frac{\Delta on - \Delta off}{\tau\frac{m}{J}} \cdot \left(\frac{(\Delta on - \Delta off)^2 + 2\tau^2\frac{m}{J}(\Delta on + \Delta off)}{(\Delta on - \Delta off)(\tau\frac{m}{J})}\right)^{-1}$

$= \frac{(\Delta on - \Delta off)\cdot(\Delta on - \Delta off)(\tau\frac{m}{J})}{(\tau\frac{m}{J})(\Delta on - \Delta off)^2 + 2\tau^2\frac{m}{J}(\Delta on + \Delta off)}$

$= \frac{(\Delta on - \Delta off)^2}{(\Delta on - \Delta off)^2 + 2\tau^2\frac{m}{J}(\Delta on + \Delta off)}$

Gagandeep Thapar

Table of Contents

AERO 560; HW 4

Housekeeping

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

Givens and Setup

```
% problem 2
trigger.delOff = 0.1;
trigger.delOn = 0.2;
trigger.mag = 1;

trigger2.delOff = 0.001;
trigger2.delOn = 0.002;

sat.J = 100;

sat.p1.command = 0;
sat.p1.initial = 5;
sat.p1.tau = 5;



% problem 3
sat.mass = 750;
sat.side = 1;
sat.dx = sat.side/2;
sat.dy = sat.side/2;
sat.dz = sat.side/2;

sat.p3.J = sat.mass * sat.side^2 / 6 * eye(3);


sat.zeta = 0.65;    % [~] Dampening Coefficient
```

```
sat.ts = 30;    % [sec] settling time
sat.wn = log(0.02*sqrt(1 - sat.zeta^2))/(-1*sat.zeta*sat.ts);
sat.zeta = sat.zeta*eye(3);
sat.wn = sat.wn*eye(3);

sat.Kd = 2*sat.p3.J*sat.zeta*sat.wn;
sat.Kp = 2*(sat.p3.J)*(sat.wn^2);

sat.T = 50;
sat.HF = sat.T * [-1 0 0 -1 0 0 1 0 0 1 0 0;
          0 1 0 0 -1 0 0 1 0 0 -1 0;
          0 0 1 0 0 -1 0 0 -1 0 0 1];
sat.HM = [0 sat.dz -sat.dy 0 sat.dz -sat.dy 0 -sat.dz sat.dy 0 -sat.dz sat.dy;
          sat.dz 0 -sat.dx -sat.dz 0 sat.dx sat.dz 0 -sat.dx -sat.dz 0 sat.dx;
          -sat.dy sat.dx 0 sat.dy -sat.dx 0 sat.dy -sat.dx 0 -sat.dy sat.dx
 0];

sat.H = [sat.HF;sat.HM];

sat.p3.psi0 = 23;
sat.p3.theta0 = -16;
sat.p3.phi0 = 42;

sat.p3.euls0_lvlh = [sat.p3.psi0;sat.p3.theta0;sat.p3.phi0];
sat.p3.w0_lvlh = [0;0;0];

sat.p3.quat0 = eul_quat(sat.p3.phi0, sat.p3.theta0, sat.p3.psi0);
sat.p3.des_quat = [0;0;0;1];
```

Simulation

```
p1 = sim('HW4Model.slx', 1000);
p3_sim = sim('HW4ThrusterModel.slx', 100);
```

Unpack data

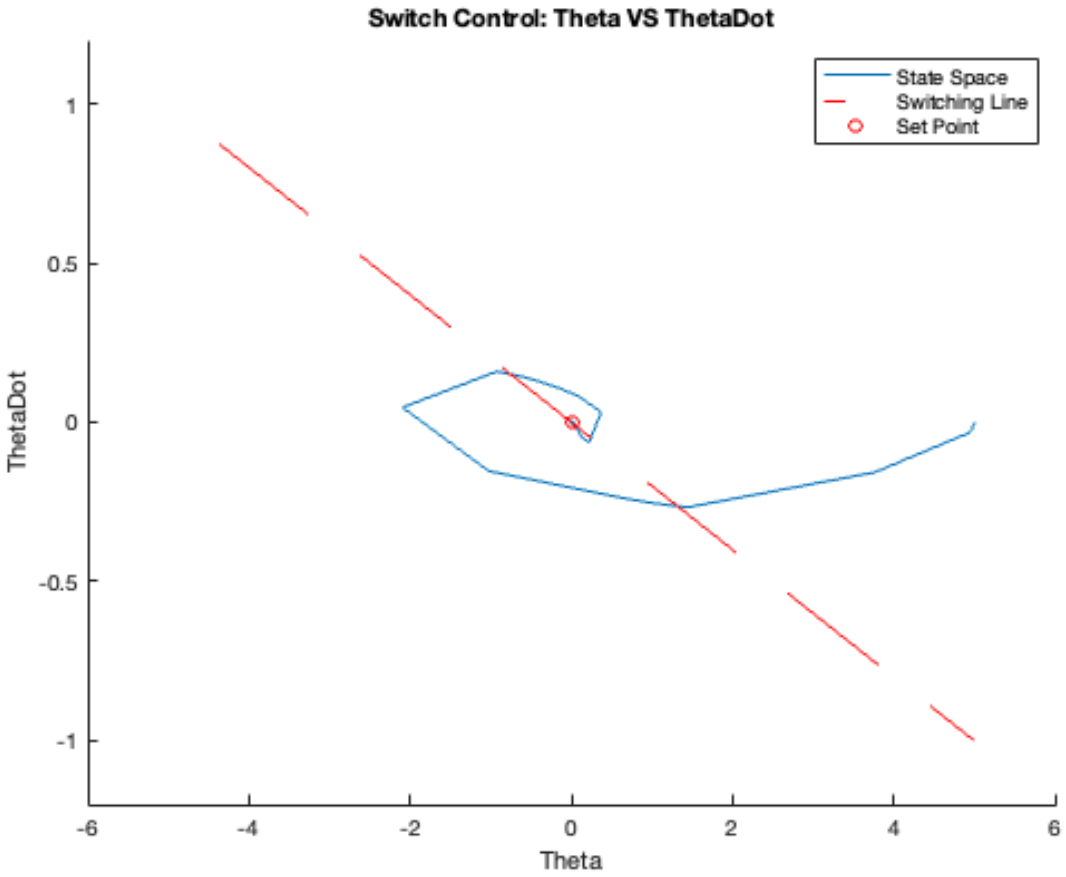
P2

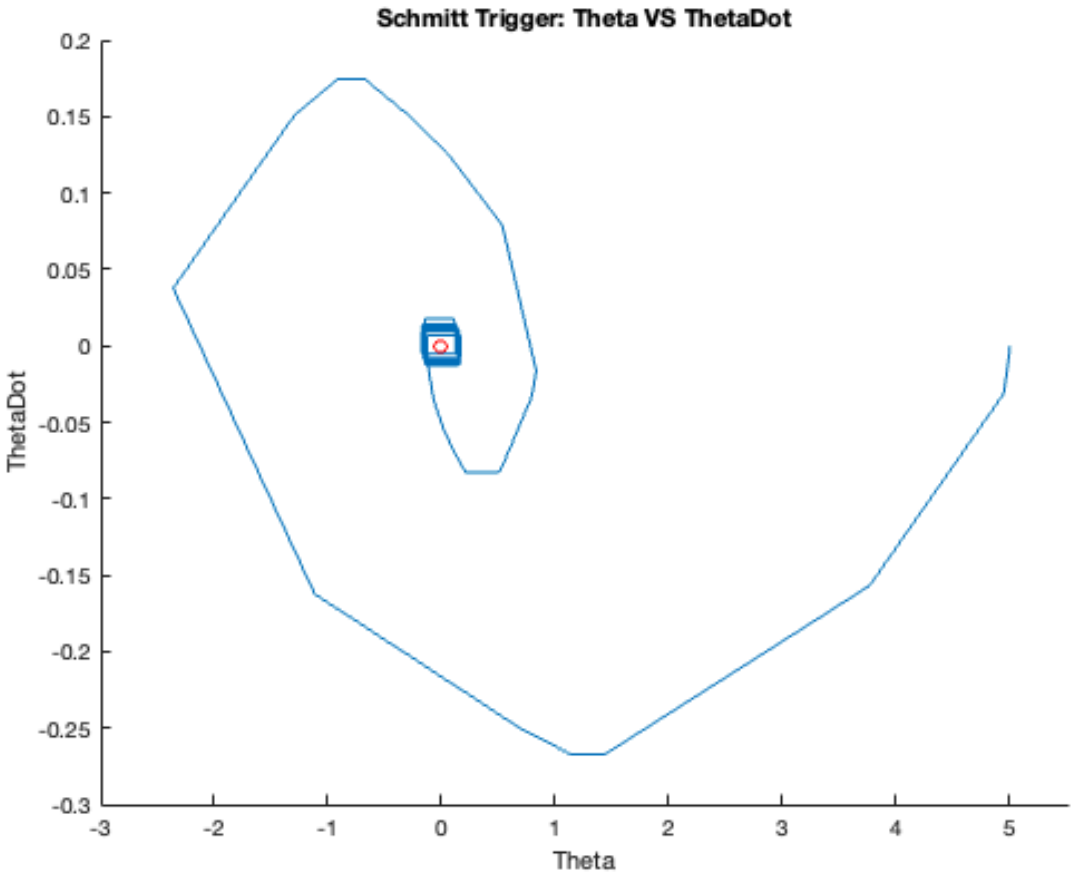
```
time = p1.tout;

sat.switch.theta = p1.switch_theta;
sat.switch.thetaDot = p1.switch_thetaDot;

sat.schmitt.theta = p1.schmitt_theta;
sat.schmitt.thetaDot = p1.schmitt_thetaDot;

% P3
p3.time = squeeze(p3_sim.tout)';
p3.req_torque = squeeze(p3_sim.command_torque)';
p3.sat.w = squeeze(p3_sim.w_Body_LVLH)';
p3.sat.eul = squeeze(p3_sim.eul_Body_LVLH)';
p3.sat.q = squeeze(p3_sim.q_Body_LVLH)';
```

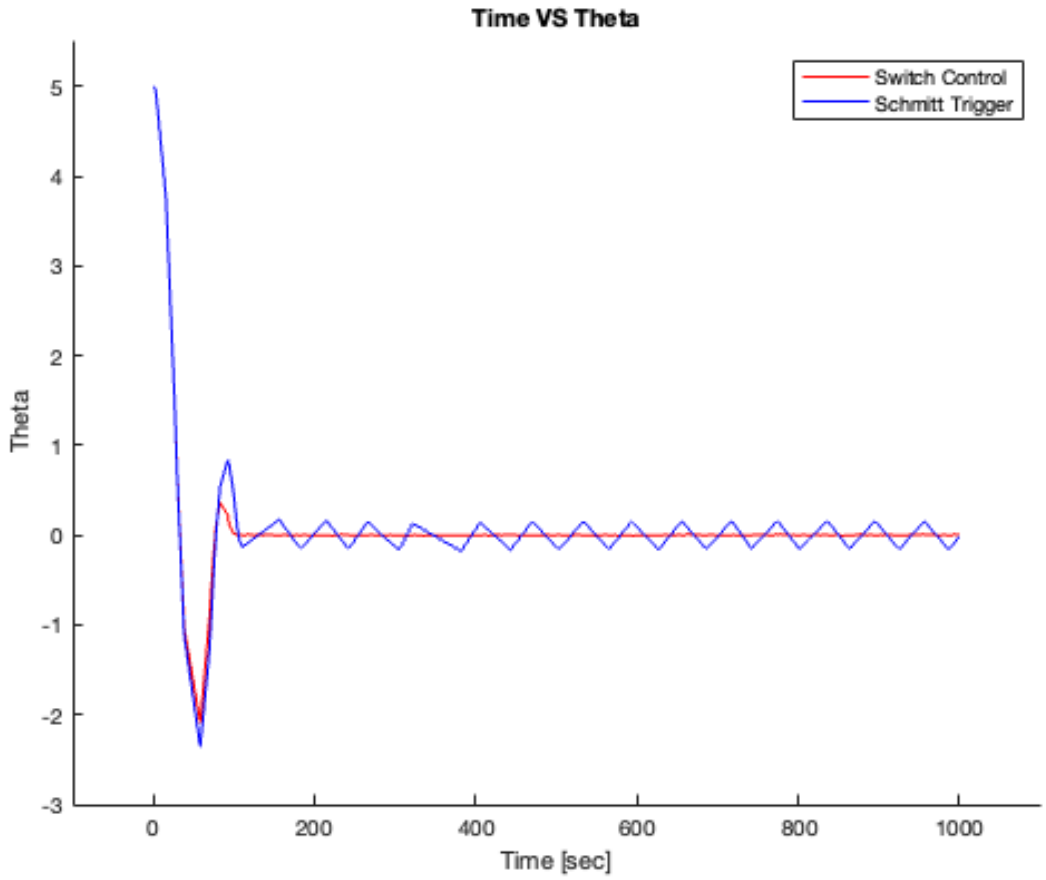
```
p3.allocation = squeeze(p3_sim.thrust_alloc)';
p3.thrust_sol = squeeze(p3_sim.thrust_sol)';
```

Plot Data

```
% switch switching lines
x = [-5 5];
tau_line = -1/sat.p1.tau * x;

figure
hold on
plot(sat.switch.theta, sat.switch.thetaDot);
plot(x, tau_line, 'r--')
scatter(0,0,'ro')
hold off
xlabel('Theta')
ylabel('ThetaDot')
title('Switch Control: Theta VS ThetaDot')
legend('State Space', 'Switching Line', 'Set Point')
axis padded


figure
hold on
plot(sat.schmitt.theta, sat.schmitt.thetaDot);
% plot(x, tau_line, 'r--')
scatter(0,0,'ro')
hold off
xlabel('Theta')
ylabel('ThetaDot')
title('Schmitt Trigger: Theta VS ThetaDot')
axis padded
% legend('State Space', 'Switching Line', 'Set Point')

figure
hold on
plot(time, sat.switch.theta, 'r')
plot(time, sat.schmitt.theta, 'b')
hold off
xlabel('Time [sec]')
ylabel('Theta')
title('Time VS Theta')
axis padded
legend('Switch Control', 'Schmitt Trigger')
```


Switch Control: Theta VS ThetaDot
State Space
Switching Line
Set Point
ThetaDot
Theta
1
0.5
0
-0.5
-1
-6
-4
-2
0
2
4
6


Schmitt Trigger: Theta VS ThetaDot
ThetaDot
Theta
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-3
-2
-1
0
1
2
3
4
5

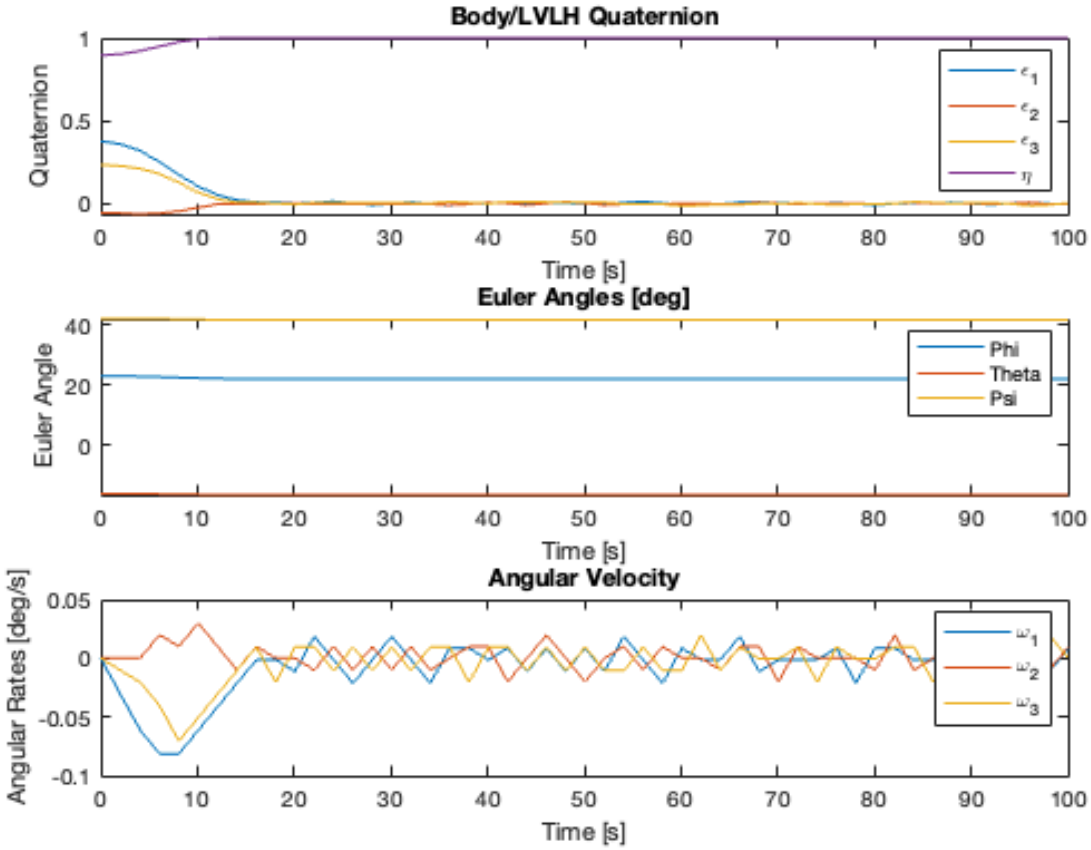


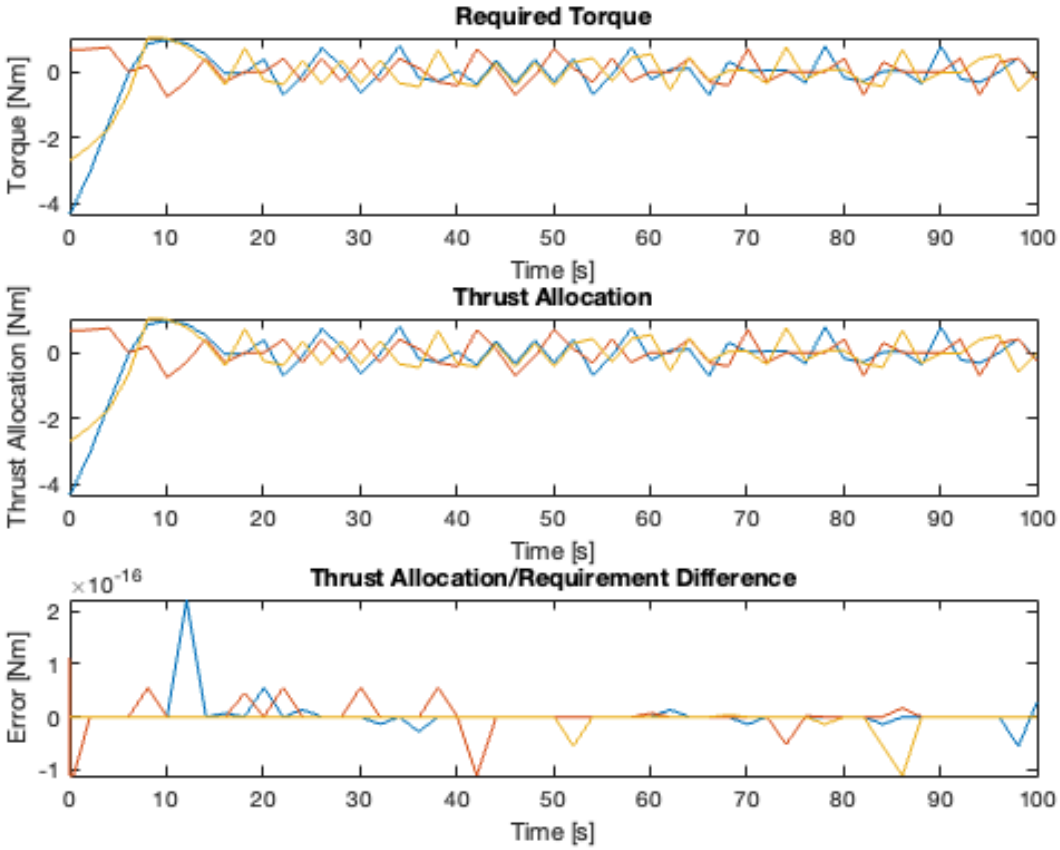
Plot Data (P3)

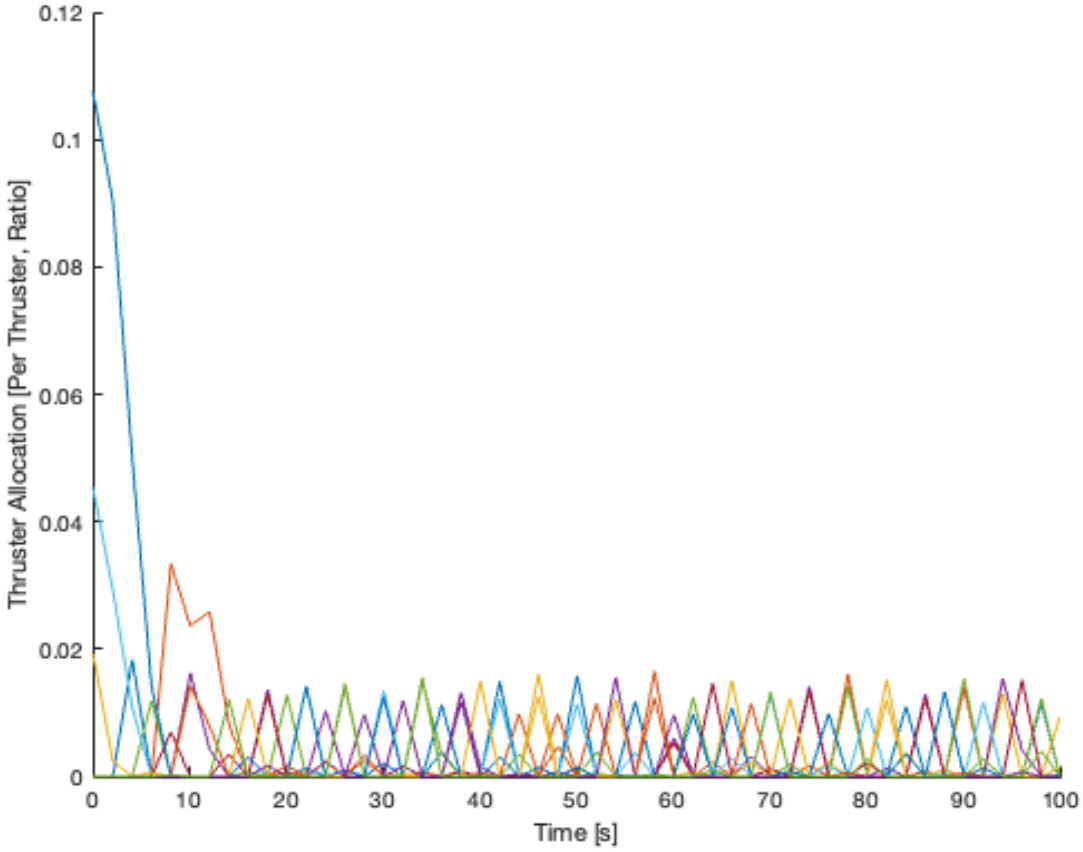
close all;

```
figure
subplot(3,1,1)
plot(p3.time, p3.sat.q);
xlabel('Time [s]')
ylabel('Quaternion')
title('Body/LVLH Quaternion')
legend('\epsilon_1','\epsilon_2','\epsilon_3','\eta')

subplot(3,1,2)
plot(p3.time, p3.sat.eul);
xlabel('Time [s]')
ylabel('Euler Angle')
title('Euler Angles [deg]')
legend('Phi', 'Theta', 'Psi')

subplot(3,1,3)
plot(p3.time, p3.sat.w);
xlabel('Time [s]')
ylabel('Angular Rates [deg/s]')
```

```
title('Angular Velocity')
legend('\omega_1', '\omega_2', '\omega_3')

figure
subplot(3,1,1)
plot(p3.time, p3.req_torque);
xlabel('Time [s]')
ylabel('Torque [Nm]')
title('Required Torque')

subplot(3,1,2)
plot(p3.time, p3.thrust_sol);
xlabel('Time [s]')
ylabel('Thrust Allocation [Nm]')
title('Thrust Allocation')

subplot(3,1,3)
plot(p3.time, p3.req_torque-p3.thrust_sol);
xlabel('Time [s]')
ylabel('Error [Nm]')
title('Thrust Allocation/Requirement Difference')

figure
[~,a] = size(p3.allocation);
hold on
for i = 1:a
p3.allocation(:,i) = p3.allocation(:,i)/sat.T;
plot(p3.time, p3.allocation(:,i));
end
hold off
xlabel('Time [s]')
ylabel('Thruster Allocation [Per Thruster, Ratio]')
```


Body/LVLH Quaternion
Quaternion
Time [s]
Euler Angles [deg]
Euler Angle
Phi
Theta
Psi
Time [s]
Angular Velocity
Angular Rates [deg/s]
Time [s]


Required Torque
Torque [Nm]
Time [s]
Thrust Allocation
Thrust Allocation [Nm]
Time [s]
Thrust Allocation/Requirement Difference
×10^-16
Error [Nm]
Time [s]

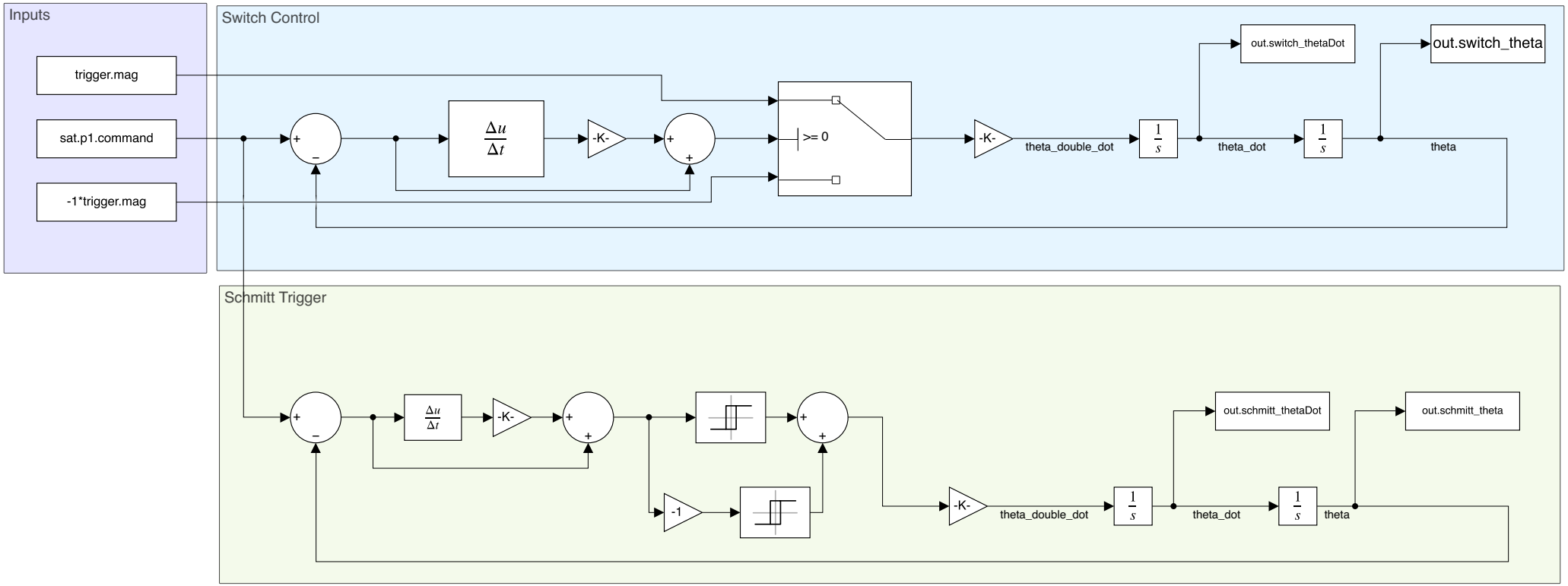


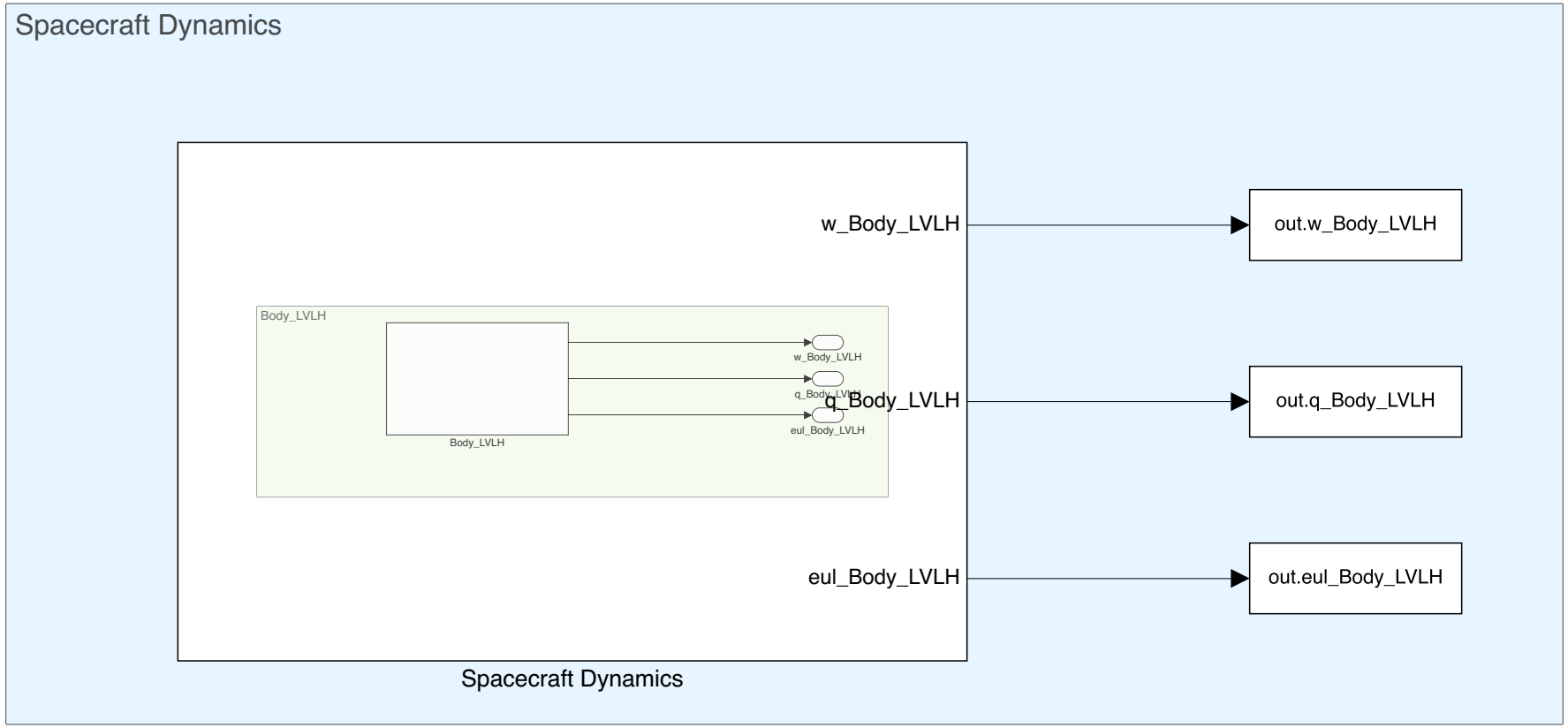
functions

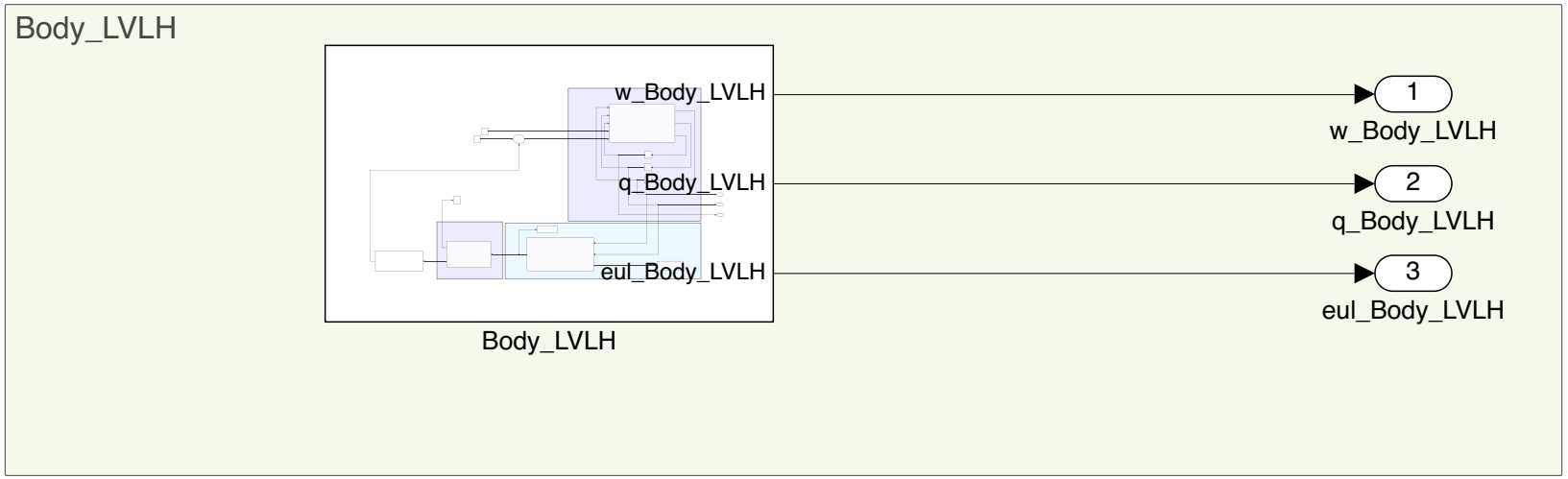
```
function eul = quat_eul(q)

    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 quat = eul_quat(phi, theta, psi)

    function Q = Qx(theta)
        Q = [1 0 0;
            0 cosd(theta) sind(theta);
```

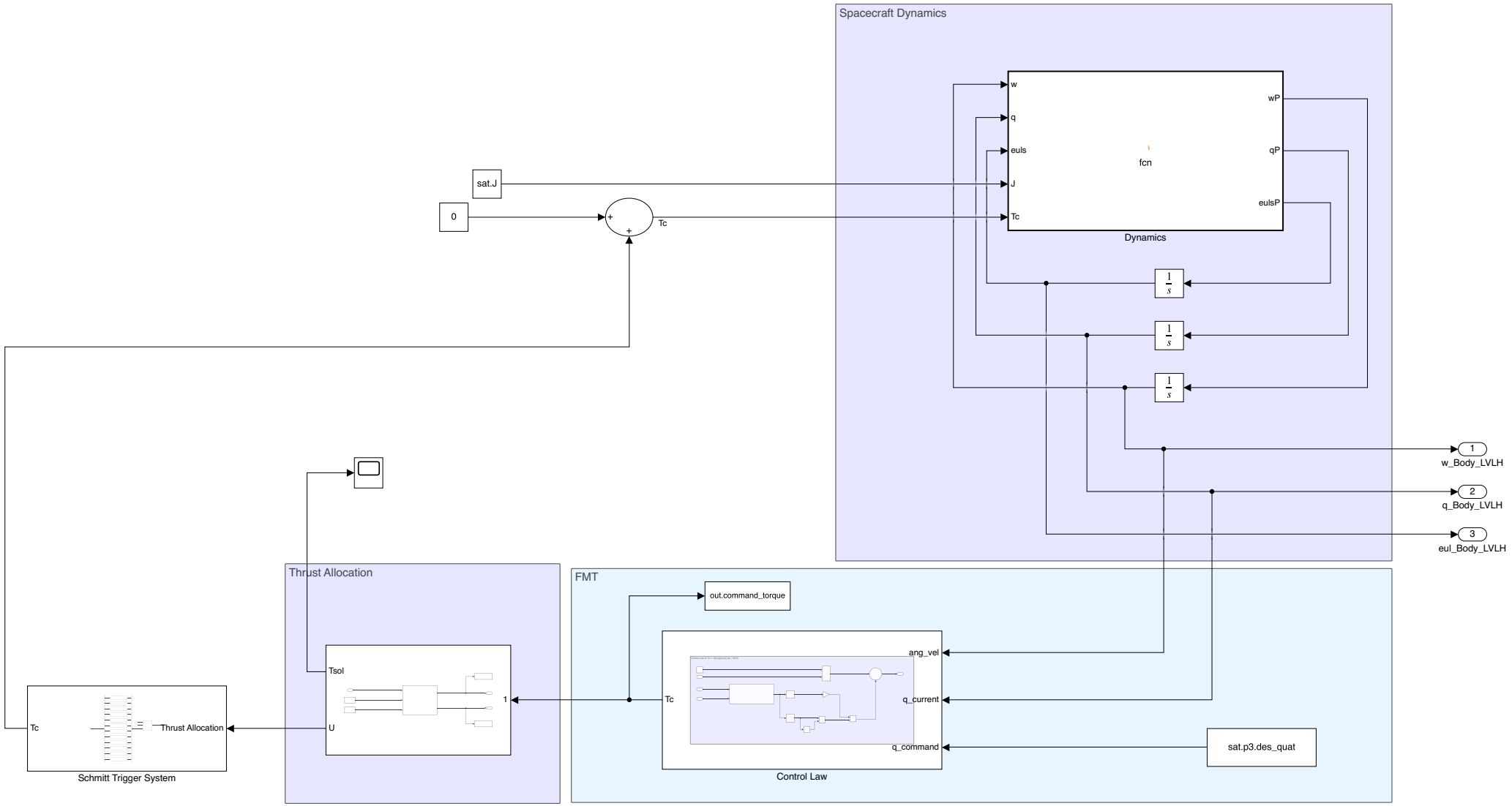
```
            0 -sind(theta) cosd(theta)];
    end

    function Q = Qy(theta)
        Q = [cosd(theta) 0 -sind(theta);
                0 1 0;
                sind(theta) 0 cosd(theta)];
    end

    function Q = Qz(theta)
        Q = [cosd(theta) sind(theta) 0;
                -sind(theta) cosd(theta) 0;
                0 0 1];
    end

    C = Qx(phi)*Qy(theta)*Qz(psi);

    e = zeros(3,1);

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

    quat = [e;eta];

end
```

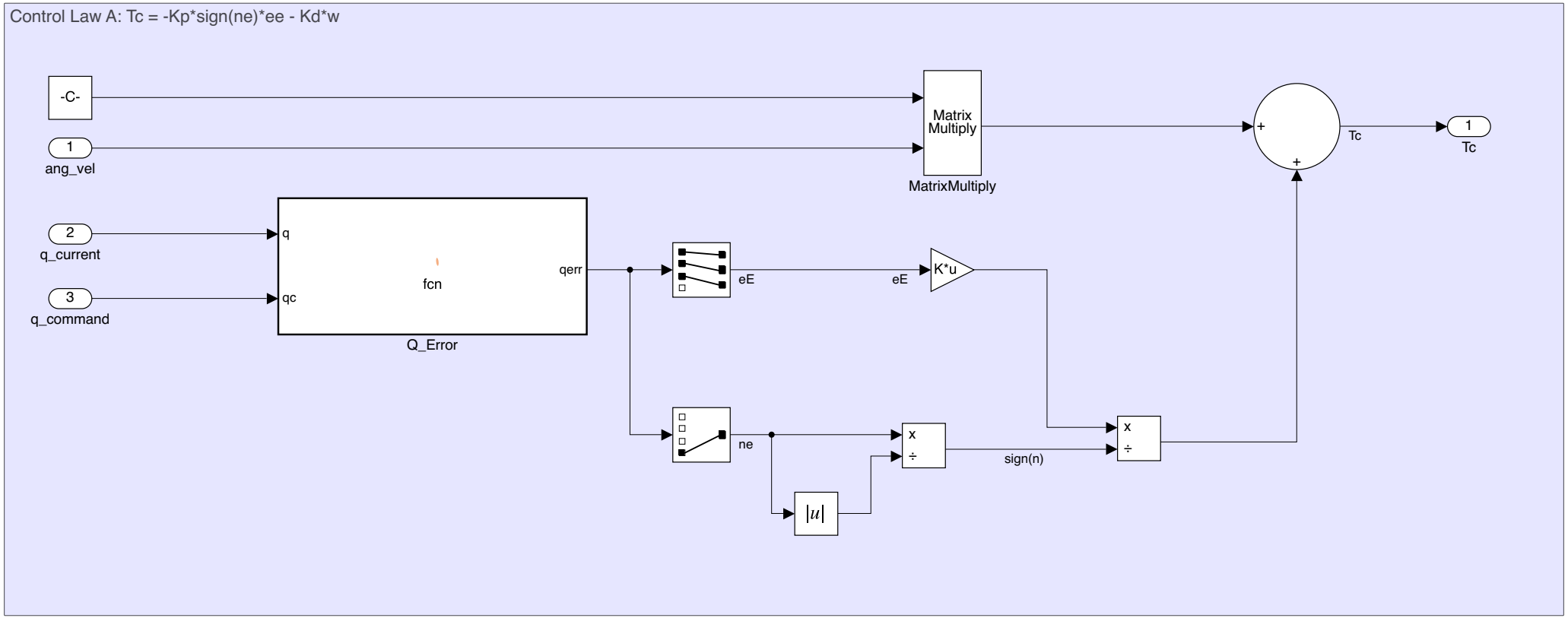
Published with MATLAB® R2022b


Inputs
trigger.mag
sat.p1.command
-1*trigger.mag
Switch Control
+
−
Δu/Δt
-K-
+
+
>= 0
-K-
theta_double_dot
1/s
theta_dot
1/s
theta
out.switch_thetaDot
out.switch_theta
Schmitt Trigger
+
−
Δu/Δt
-K-
+
+
+
+
-1
-K-
theta_double_dot
1/s
theta_dot
1/s
theta
out.schmitt_thetaDot
out.schmitt_theta


Spacecraft Dynamics
w_Body_LVLH
q_Body_LVLH
eul_Body_LVLH
Body_LVLH
w_Body_LVLH
q_Body_LVLH
eul_Body_LVLH
Body_LVLH
out.w_Body_LVLH
out.q_Body_LVLH
out.eul_Body_LVLH
Spacecraft Dynamics

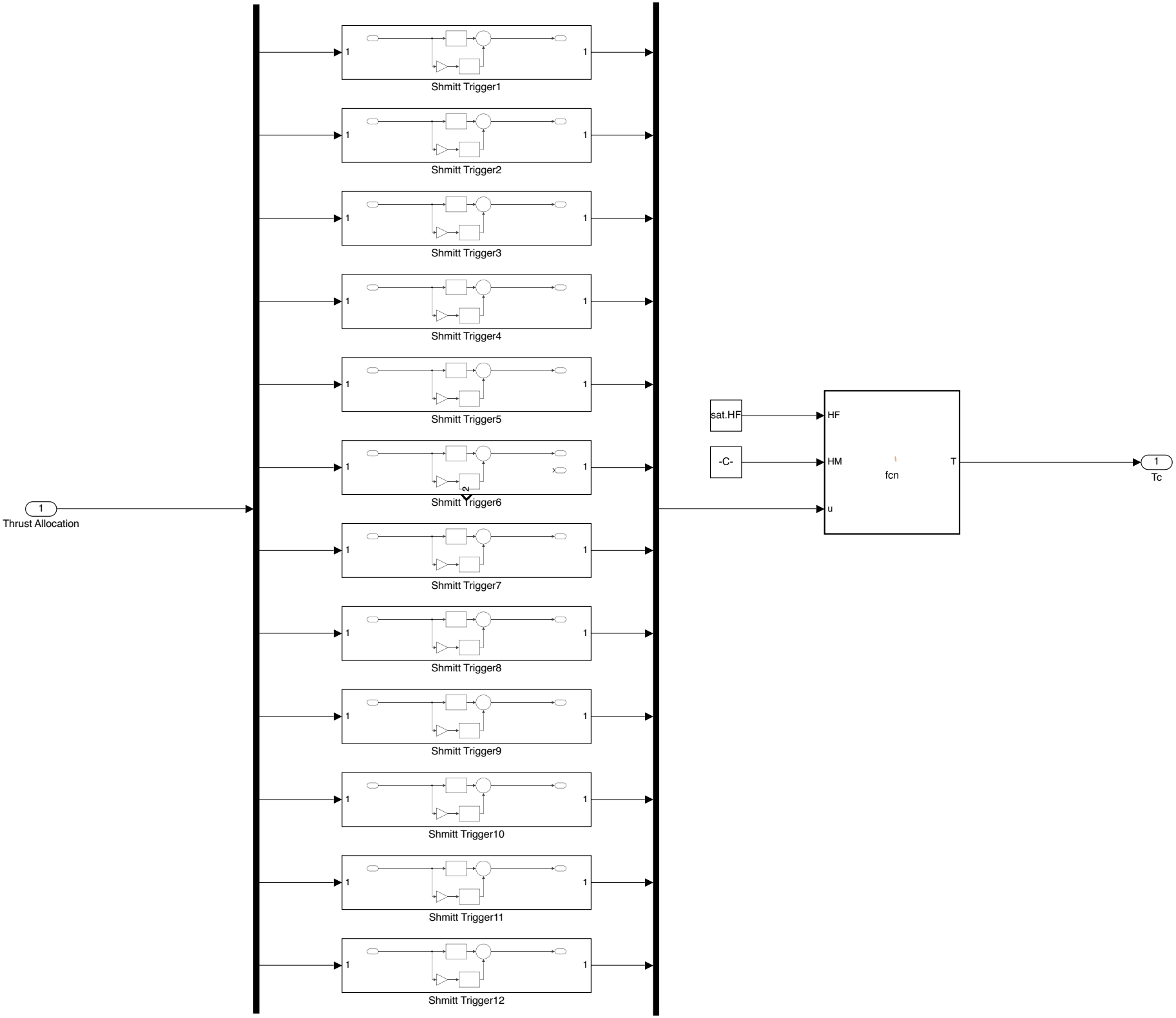

Body_LVLH
w_Body_LVLH
q_Body_LVLH
eul_Body_LVLH
Body_LVLH
1
w_Body_LVLH
2
q_Body_LVLH
3
eul_Body_LVLH


Spacecraft Dynamics
w
q
euls
J
Tc
wP
qP
eulsP
fcn
Dynamics
sat.J
0
Tc
1
s
1
s
1
s
1
w_Body_LVLH
2
q_Body_LVLH
3
eul_Body_LVLH
Thrust Allocation
Tsol
U
1
FMT
out.command_torque
Tc
ang_vel
q_current
q_command
Control Law
sat.p3.des_quat
Tc
Thrust Allocation
Schmitt Trigger System

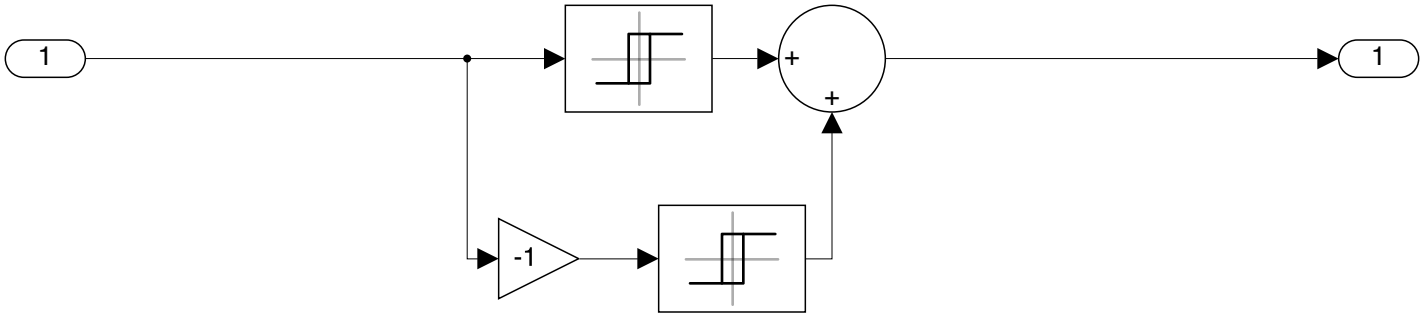

Control Law A: Tc = -Kp*sign(ne)*ee - Kd*w
-C-
1
ang_vel
Matrix Multiply
MatrixMultiply
+
+
Tc
1
Tc
2
q_current
3
q_command
q
qc
fcn
qerr
Q_Error
eE
eE
K*u
ne
x
÷
sign(n)
x
÷
|u|

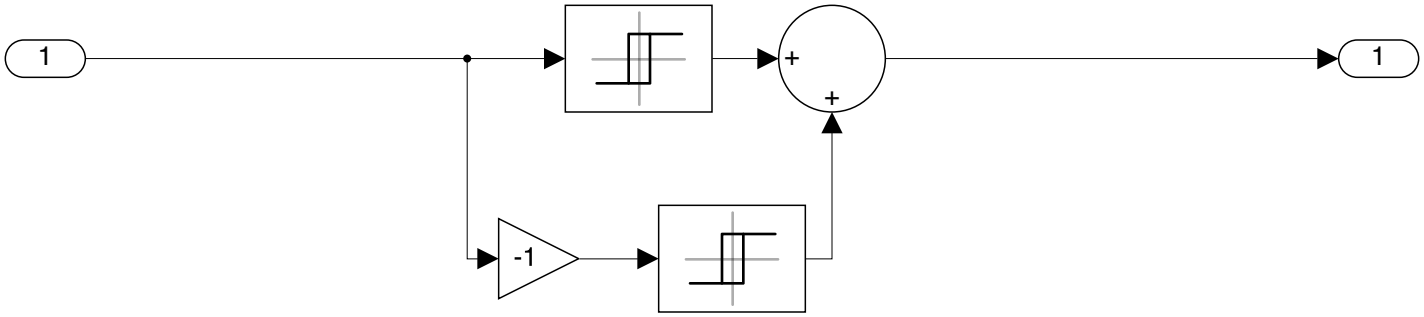
```
function qerr = fcn(q, qc)

%     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
```

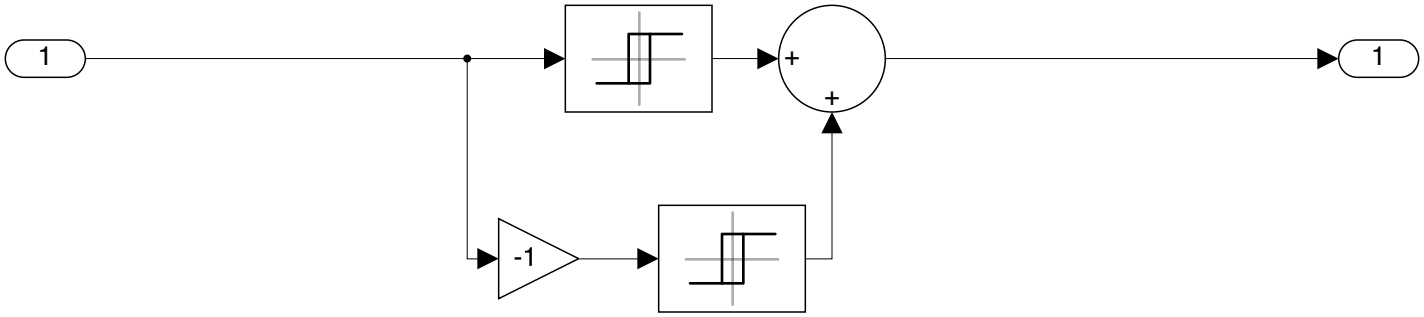
```
function [wP, qP, eulsP] = fcn(w, q, euls, J, Tc)

    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)];

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

    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
```

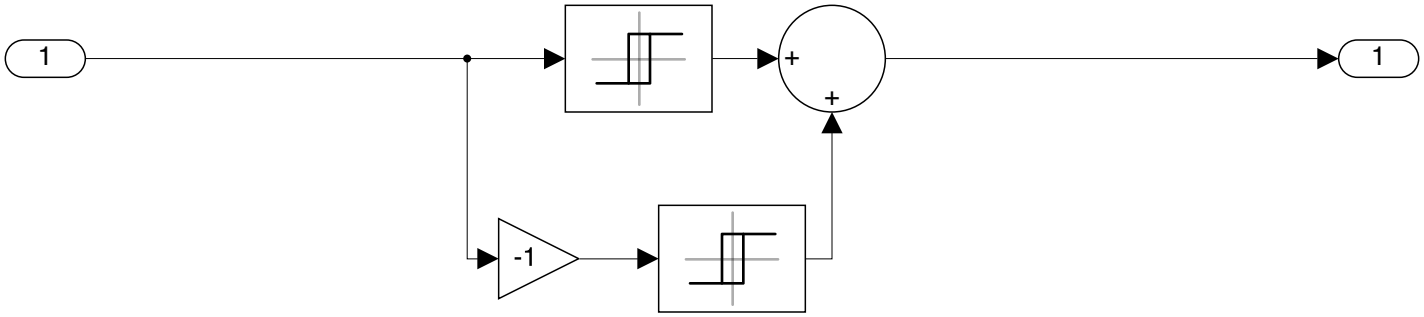

1
Thrust Allocation
Shmitt Trigger1
Shmitt Trigger2
Shmitt Trigger3
Shmitt Trigger4
Shmitt Trigger5
Shmitt Trigger6
Shmitt Trigger7
Shmitt Trigger8
Shmitt Trigger9
Shmitt Trigger10
Shmitt Trigger11
Shmitt Trigger12
sat.HF
-C-
HF
HM
u
fcn
T
1
Tc

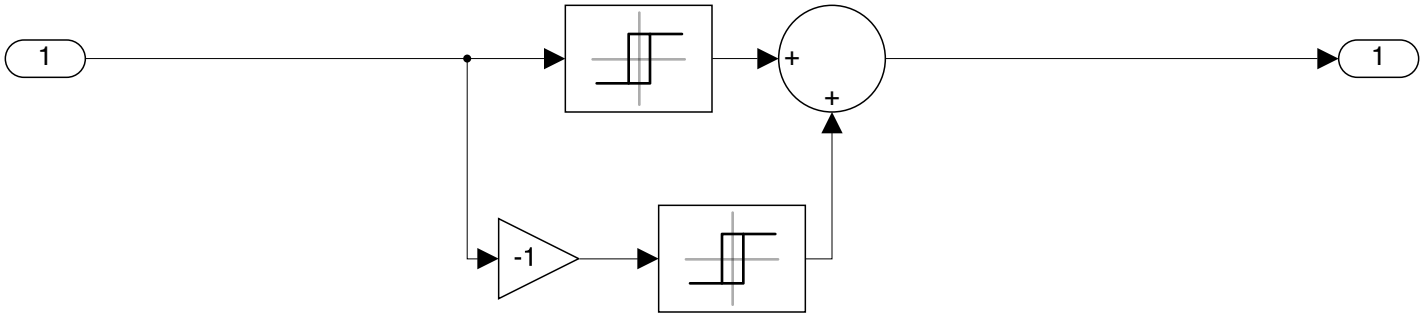
```
function T = fcn(HF, HM, u)

H = [HF;HM];
T = HM*u;
```

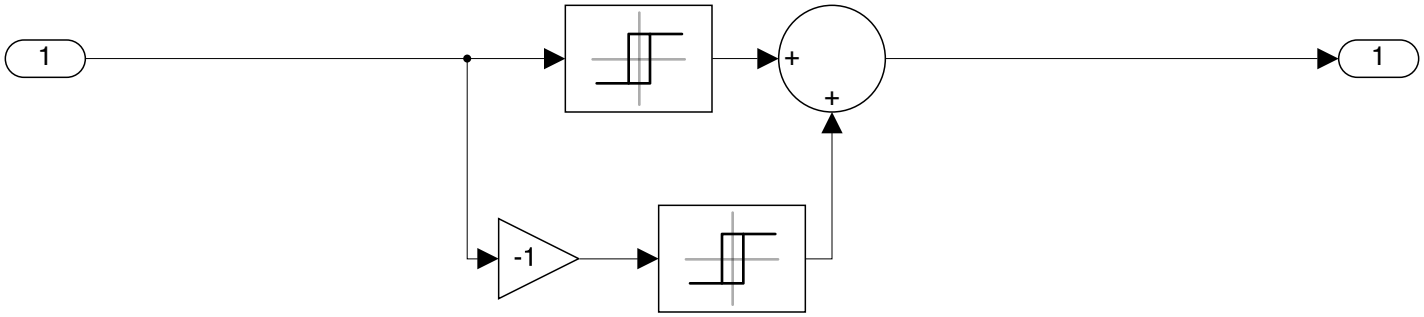

1
+
+
1
-1

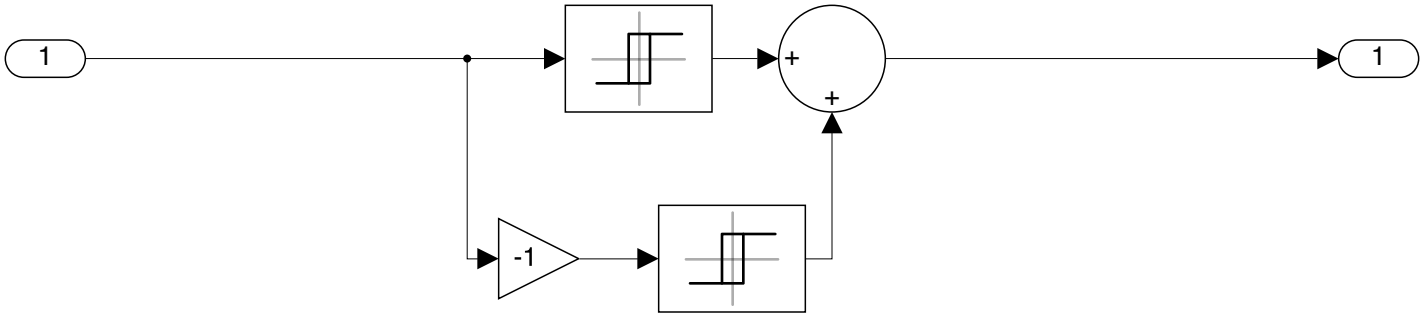

1
+
+
1
-1

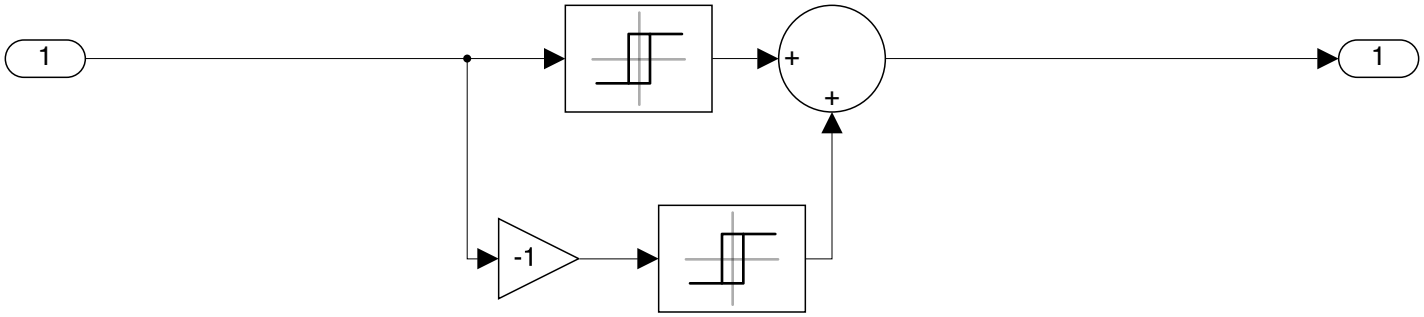

1
+
+
1
-1

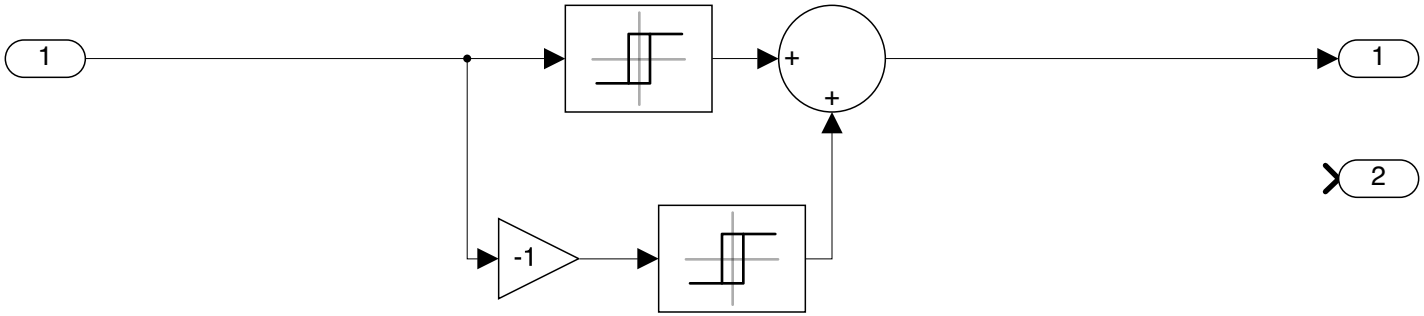

1
+
+
1
-1

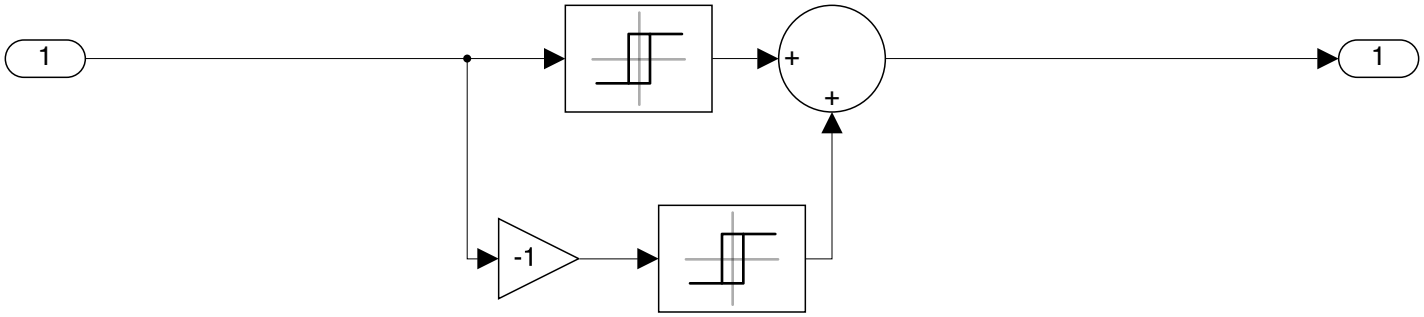

1
+
+
1
-1

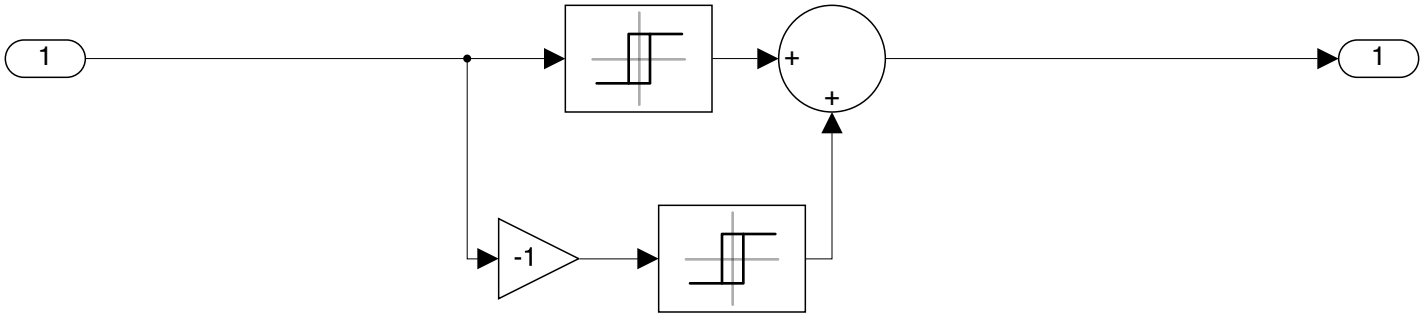

1
+
+
1
-1

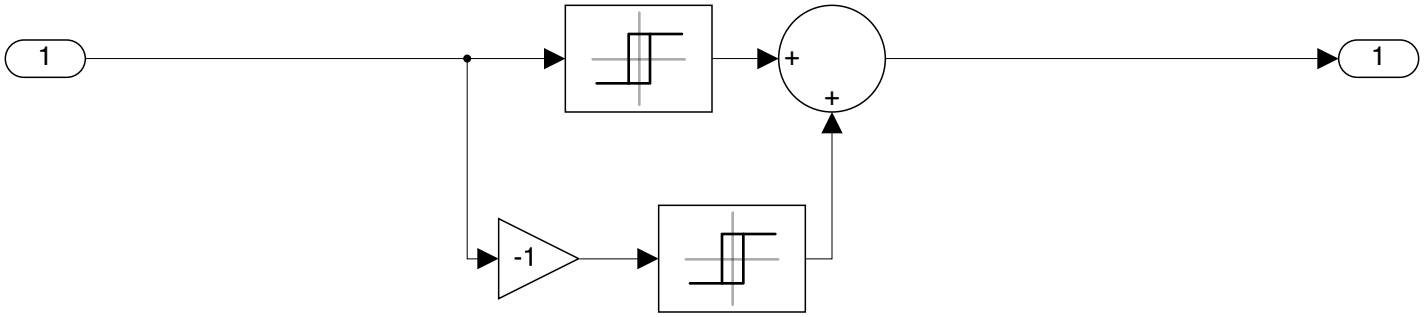

1
+
+
1
-1

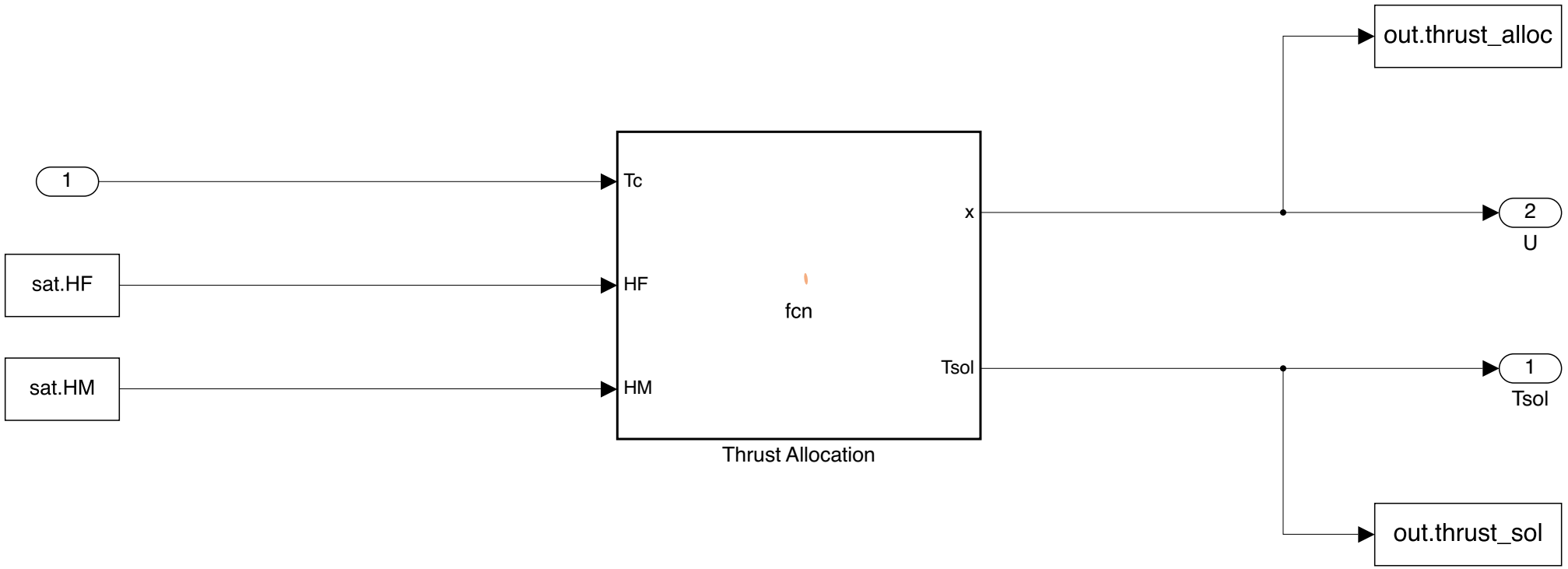

1
+
+
1
-1


1
+
+
1
2
-1


1
+
+
1
-1


1
+
+
1
-1


1
+
+
1
-1


1
sat.HF
sat.HM
Tc
HF
HM
fcn
x
Tsol
Thrust Allocation
out.thrust_alloc
2
U
1
Tsol
out.thrust_sol

```
function [x, Tsol] = fcn(Tc, HF, HM)

coder.extrinsic('linprog');
coder.extrinsic('intlinprog');
coder.extrinsic('optimoptions')

intlinprogflag = 0;

x = zeros(12,1);
lb = zeros(12,1);   % lower bound
ub = 50 * ones(12,1);   % upper bound
f = ones(1,12); % cost function to minimize is the sum of all thrusters

translate_sol = zeros(3,1); % want 0 translation
moment_sol = Tc;    % need to reach ideal Tc using thruster alloc

H = [HF;HM];    % combine translate/moment matrices

%Aeq = H;    % allocation matrix
Aeq = HM;

%Beq = [translate_sol;moment_sol]; % solution matrix
Beq = moment_sol;



if intlinprogflag == 1
    intcon = 1:12;
    options = optimoptions('intlinprog', 'Display', 'off');
    x = intlinprog(f, intcon, [], [], Aeq, Beq,  lb, ub, [], options);   % solves linprog problem

else
    options = optimoptions('linprog', 'Display', 'off');
    x = linprog(f, [], [], Aeq, Beq, lb, ub, options);
end

Tsol = Aeq * x;









Tsol = HM*x;    % gets torque from thruster allocation
```