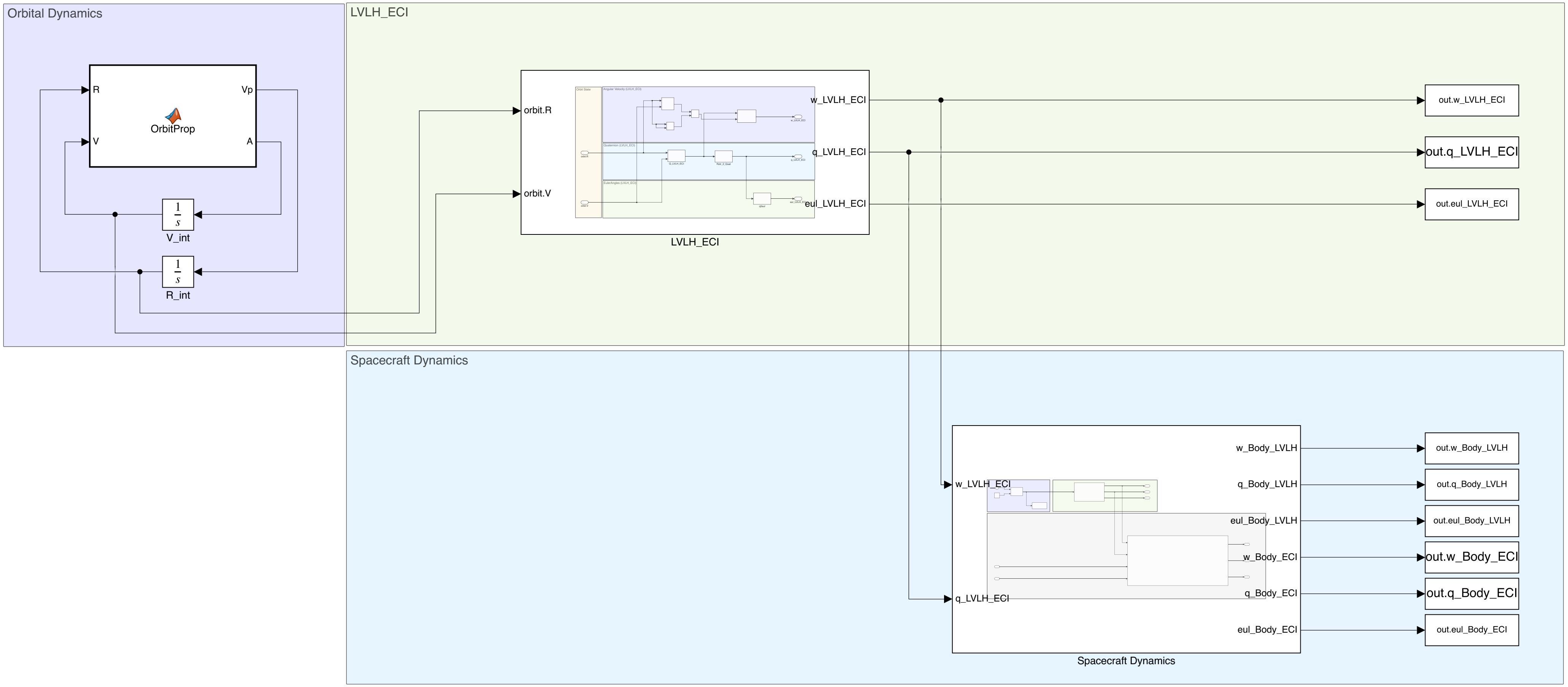
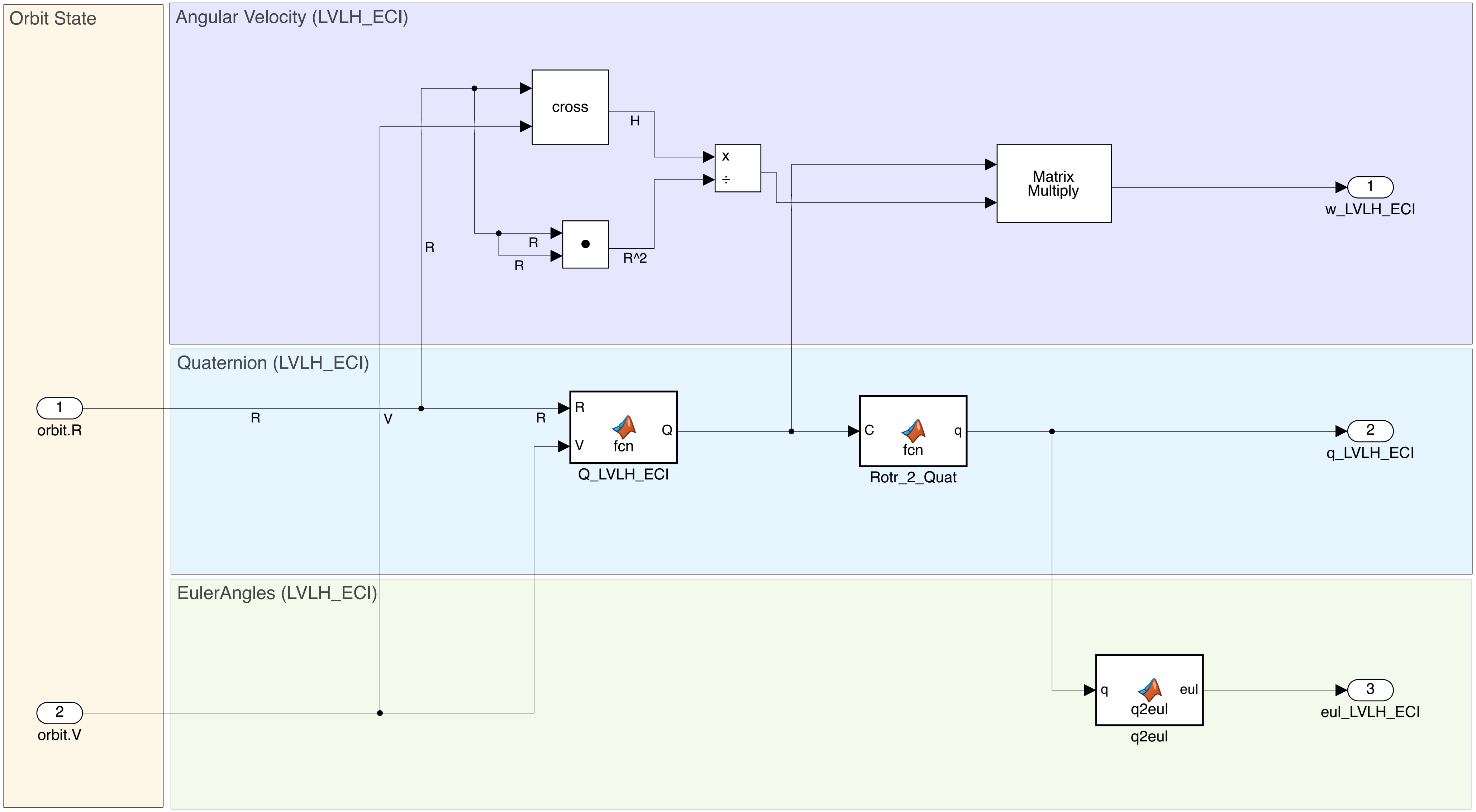

Orbital Dynamics
R
V
Vp
A
OrbitProp
1/s
V_int
1/s
R_int
LVLH_ECI
orbit.R
orbit.V
w_LVLH_ECI
q_LVLH_ECI
eul_LVLH_ECI
LVLH_ECI
out.w_LVLH_ECI
out.q_LVLH_ECI
out.eul_LVLH_ECI
Spacecraft Dynamics
w_LVLH_ECI
q_LVLH_ECI
w_Body_LVLH
q_Body_LVLH
eul_Body_LVLH
w_Body_ECI
q_Body_ECI
eul_Body_ECI
Spacecraft Dynamics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

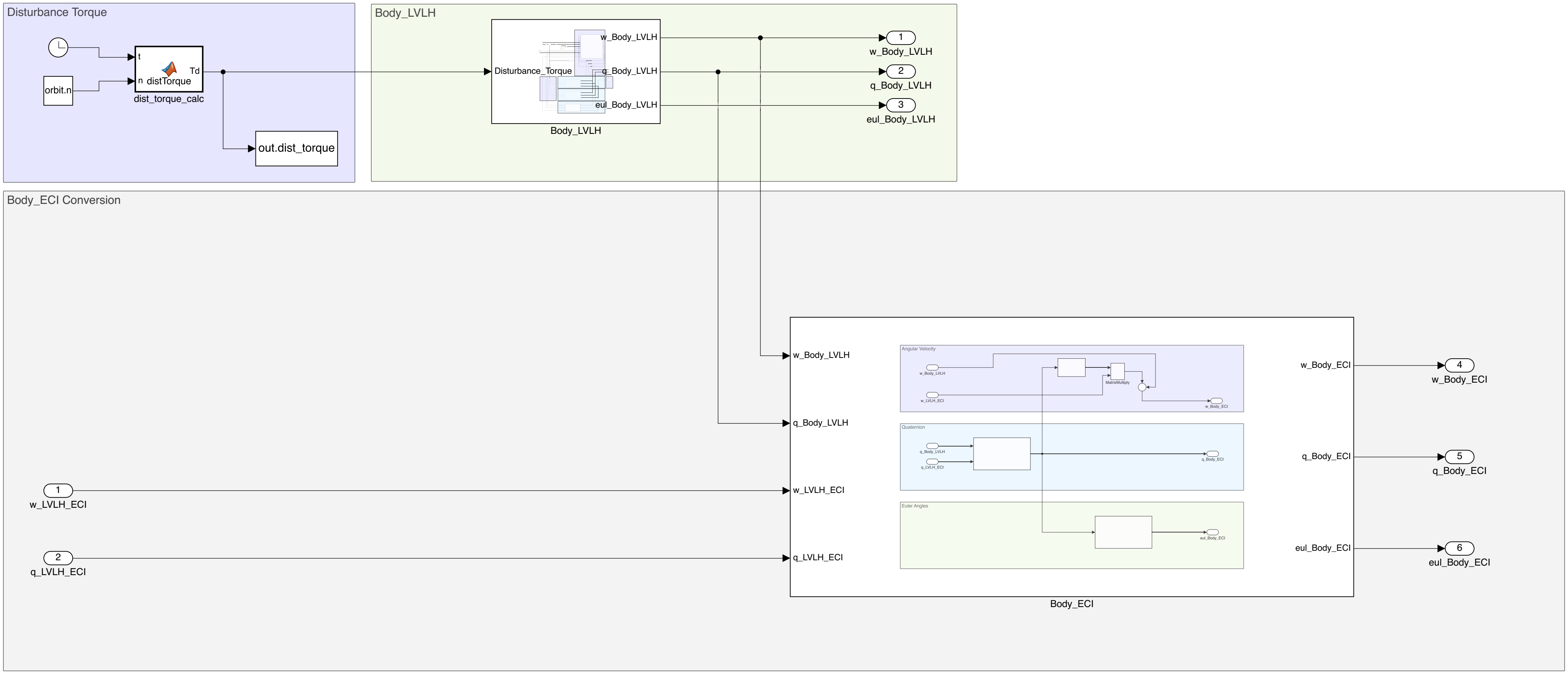

Orbit State
Angular Velocity (LVLH_ECI)
cross
H
x
÷
Matrix Multiply
1
w_LVLH_ECI
R
R
R
R^2
Quaternion (LVLH_ECI)
1
orbit.R
R
V
R
R
V
Q
fcn
Q_LVLH_ECI
C
q
fcn
Rotr_2_Quat
2
q_LVLH_ECI
EulerAngles (LVLH_ECI)
q
eul
q2eul
q2eul
3
eul_LVLH_ECI
2
orbit.V

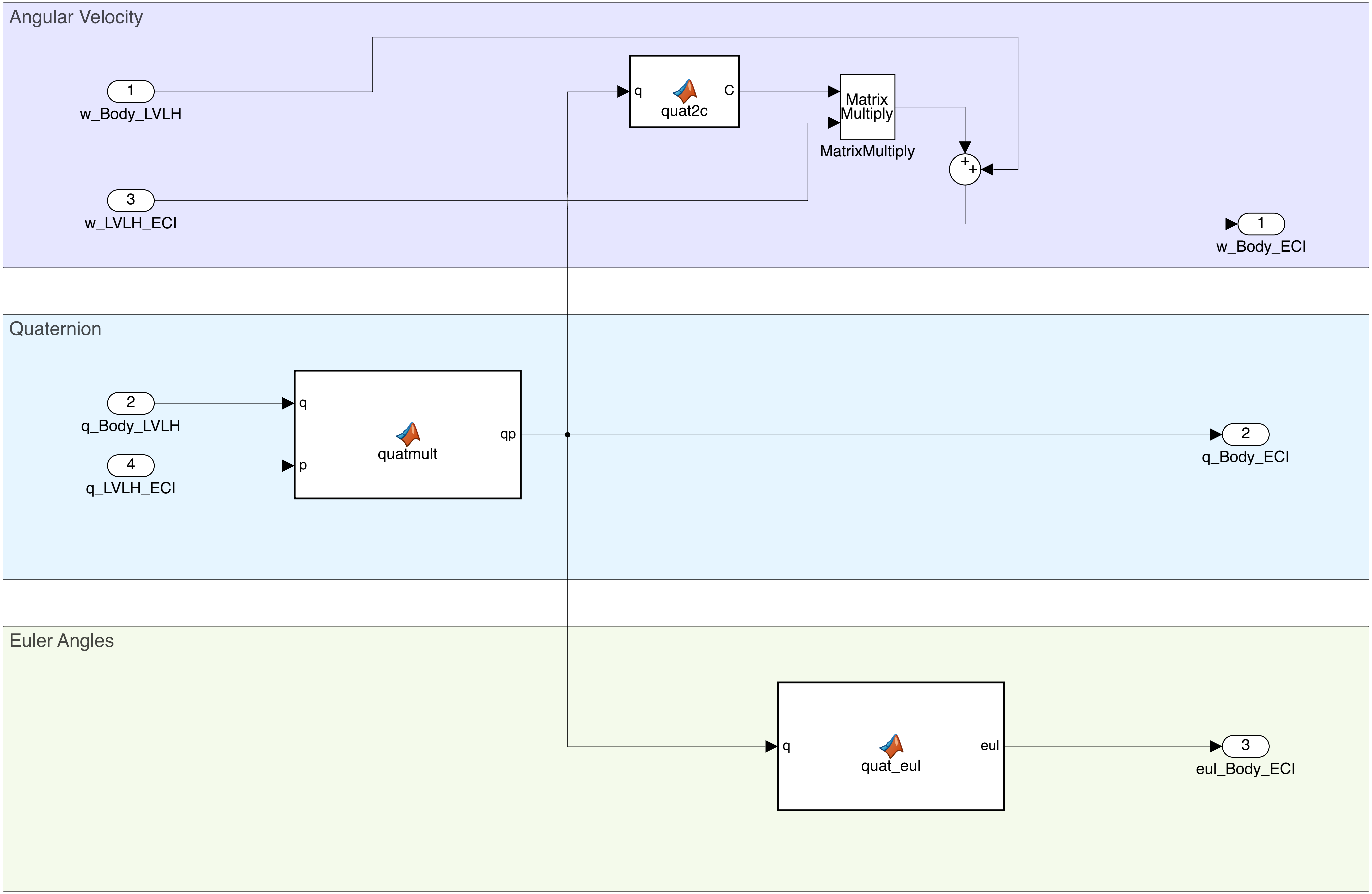
```
function Q = fcn(R,V)


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
```

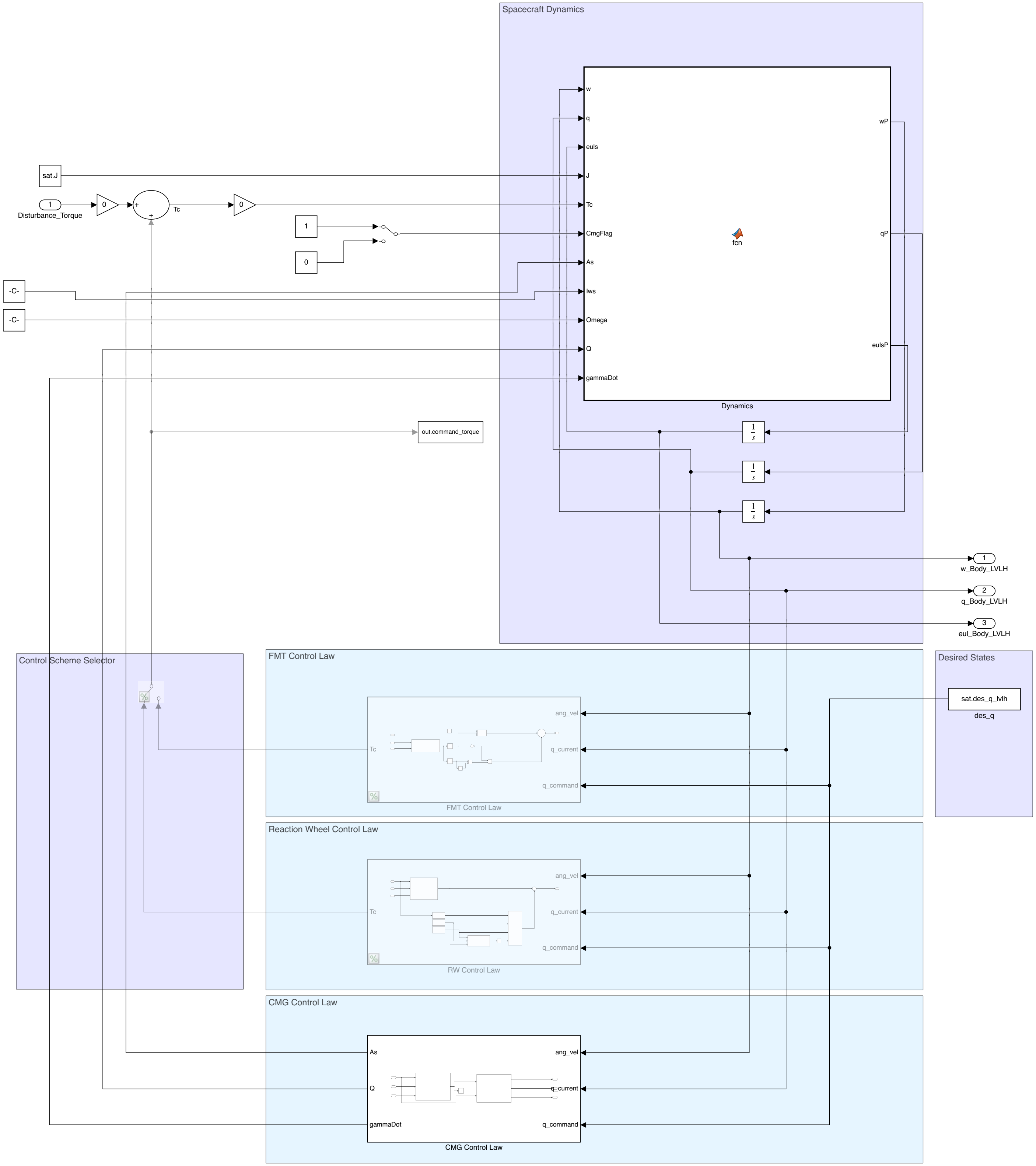

Disturbance Torque
t
n
distTorque
Td
orbit.n
dist_torque_calc
out.dist_torque
Body_LVLH
Disturbance_Torque
w_Body_LVLH
q_Body_LVLH
eul_Body_LVLH
Body_LVLH
1
w_Body_LVLH
2
q_Body_LVLH
3
eul_Body_LVLH
Body_ECI Conversion
w_Body_LVLH
q_Body_LVLH
w_LVLH_ECI
q_LVLH_ECI
Angular Velocity
w_Body_LVLH
w_LVLH_ECI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

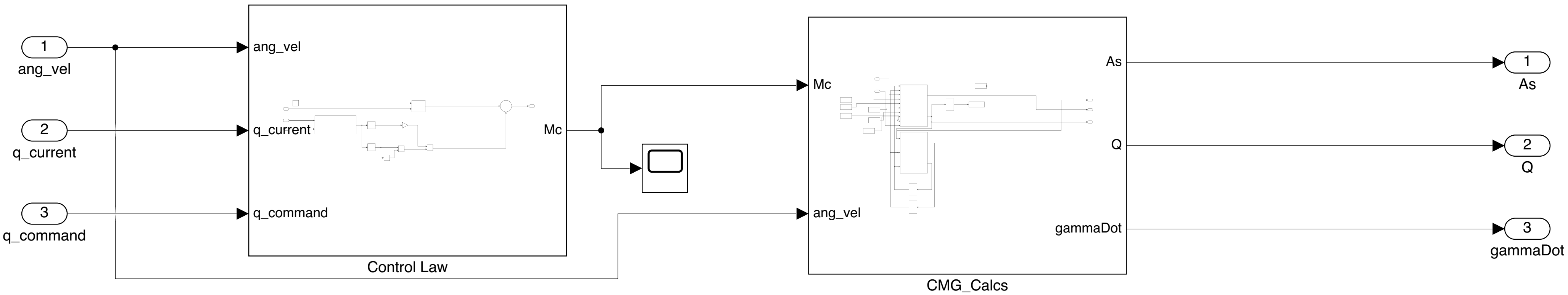

Angular Velocity
1
w_Body_LVLH
3
w_LVLH_ECI
q
quat2c
C
Matrix Multiply
MatrixMultiply
+
+
1
w_Body_ECI
Quaternion
2
q_Body_LVLH
4
q_LVLH_ECI
q
p
quatmult
qp
2
q_Body_ECI
Euler Angles
q
quat_eul
eul
3
eul_Body_ECI

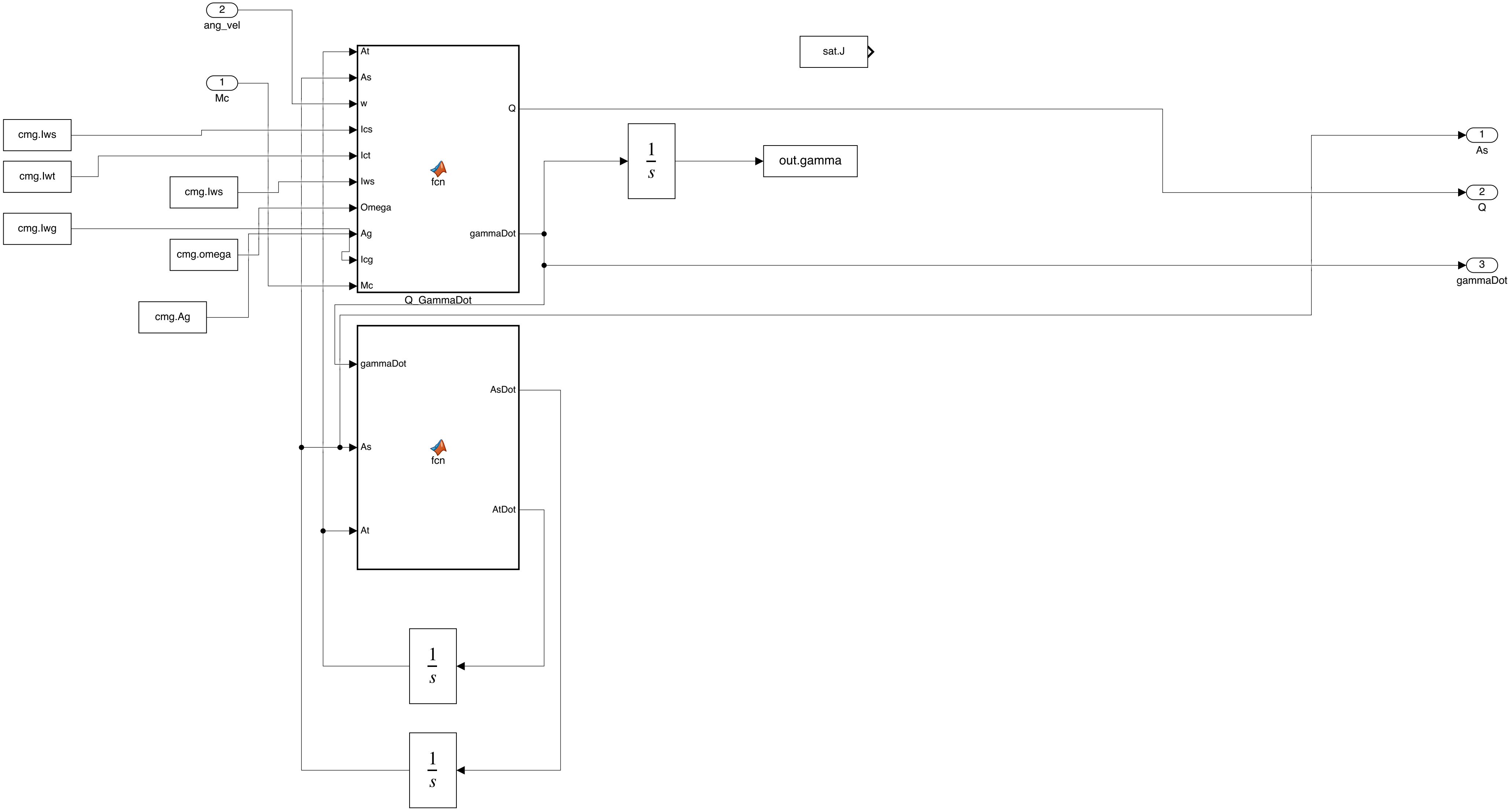
```
function qp = quatmult(q, p)

        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)]');
```

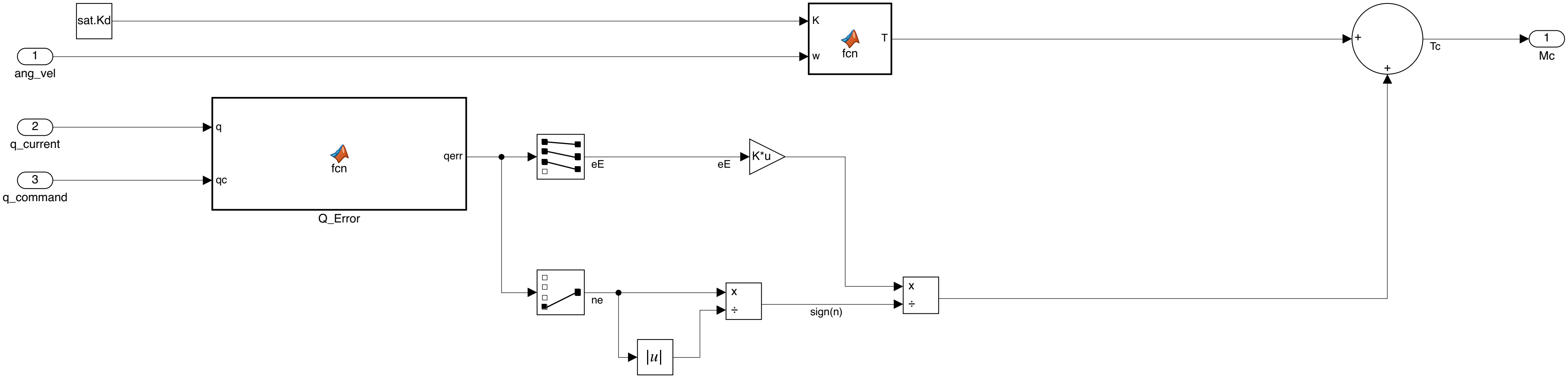

Spacecraft Dynamics
sat.J
1
Disturbance_Torque
0
+
+
+
Tc
0
1
0
-C-
-C-
w
q
euls
J
Tc
CmgFlag
As
Iws
Omega
Q
gammaDot
wP
qP
eulsP
fcn
Dynamics
out.command_torque
$\frac{1}{s}$
$\frac{1}{s}$
$\frac{1}{s}$
1
w_Body_LVLH
2
q_Body_LVLH
3
eul_Body_LVLH
Control Scheme Selector
FMT Control Law
ang_vel
Tc
q_current
q_command
FMT Control Law
Desired States
sat.des_q_lvlh
des_q
Reaction Wheel Control Law
ang_vel
Tc
q_current
q_command
RW Control Law
CMG Control Law
As
ang_vel
Q
q_current
gammaDot
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.Iws
cmg.Iwt
cmg.Iws
cmg.Iwg
cmg.omega
cmg.Ag
At
As
w
Ics
Ict
Iws
Omega
Ag
Icg
Mc
fcn
Q
gammaDot
Q_GammaDot
sat.J
1/s
out.gamma
1
As
2
Q
3
gammaDot
gammaDot
As
At
fcn
AsDot
AtDot
1/s
1/s

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

    AtDot = −1∗As∗diag(gammaDot);
    AsDot = At ∗ diag(gammaDot);
```

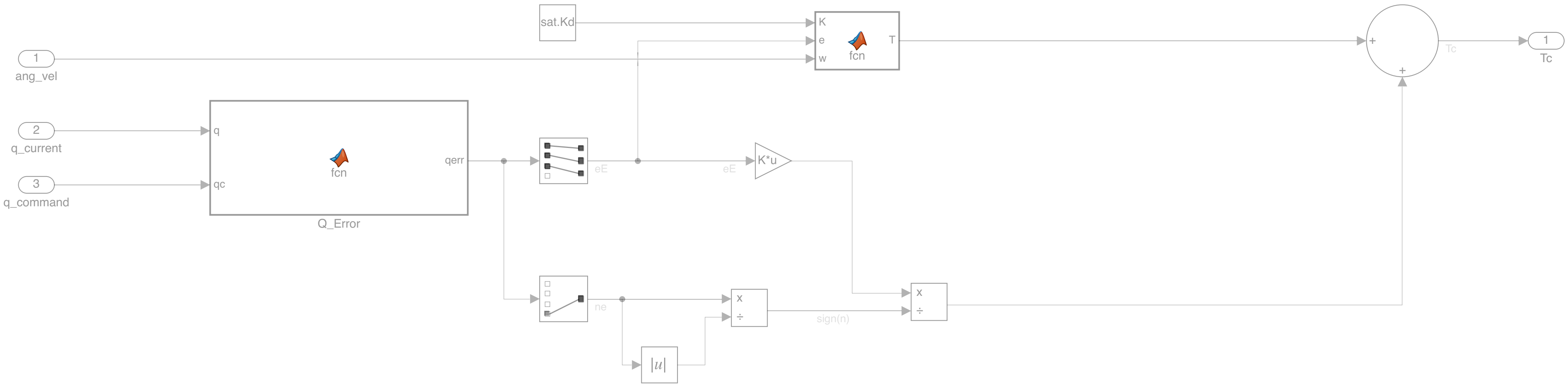
```
function [Q, gammaDot] = fcn(At, As, w, Ics, Ict, Iws, Omega, Ag, Icg, Mc)

    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
qerr
Q_Error
eE
eE
K*u
ne
x
÷
|u|
sign(n)
x
÷

```
function T = fcn(K, w)

T = -1*K*w;
```

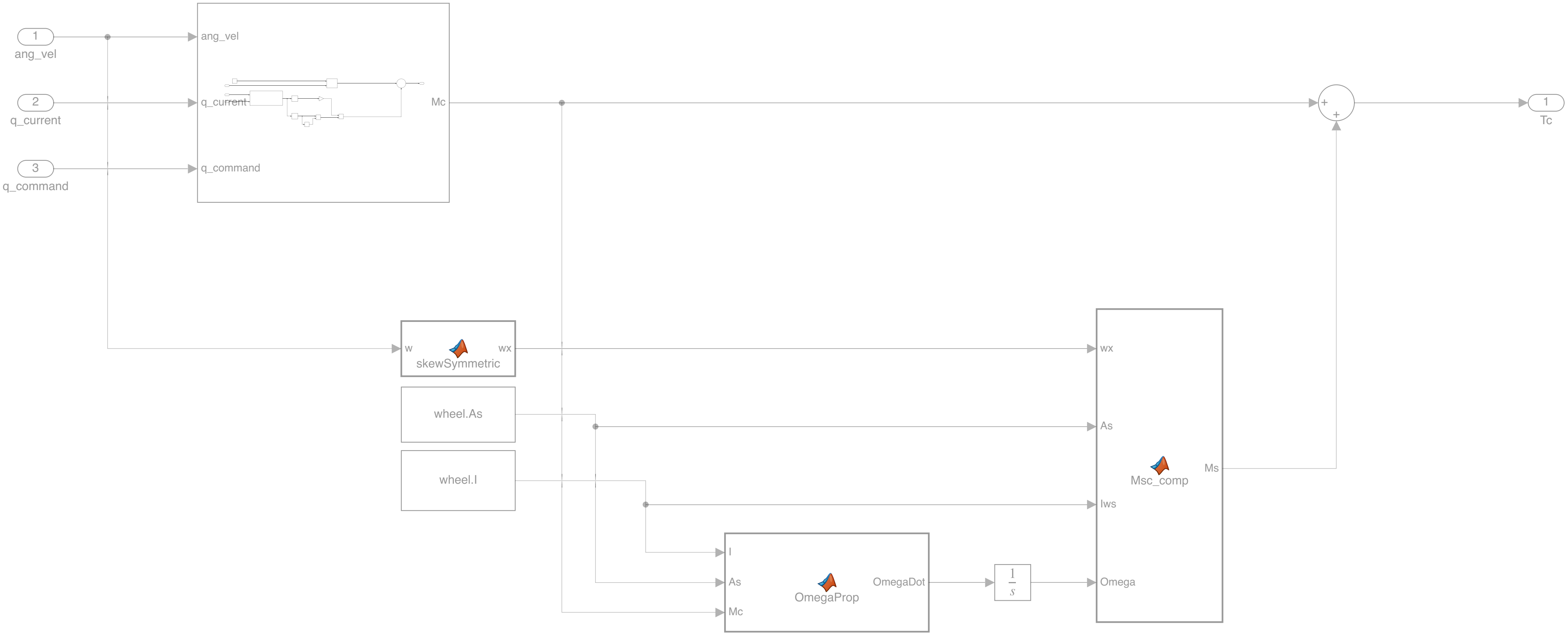
```
function qerr = fcn(q, qc)

%     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
```


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

```
function T = fcn(K, e, w)
T = −1∗K∗(1 + e'∗e)∗w;
```

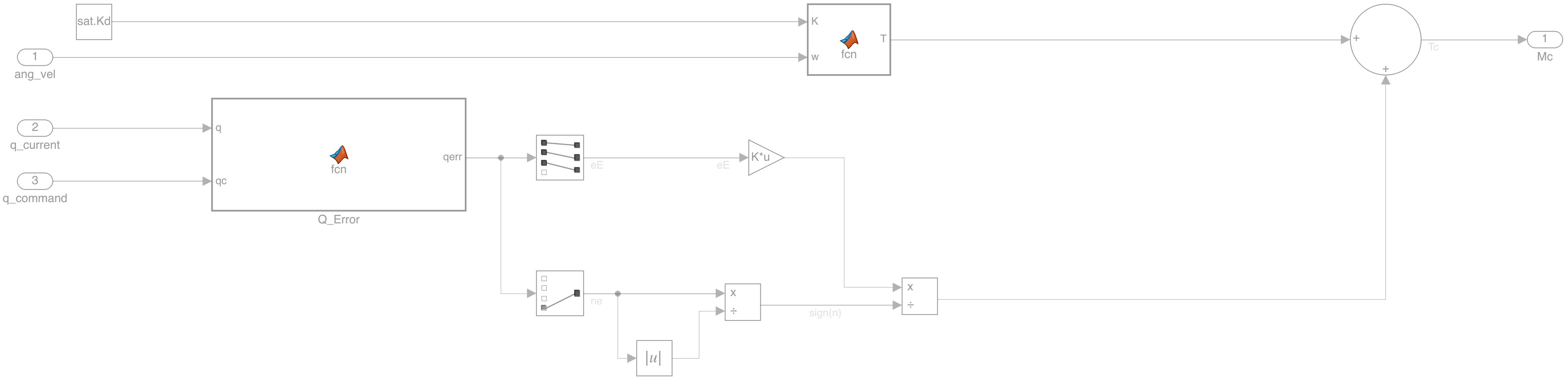
```
function qerr = fcn(q, qc)

%     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
Iws
Omega
Ms
Msc_comp
I
As
Mc
OmegaProp
OmegaDot
1/s

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

```
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
```