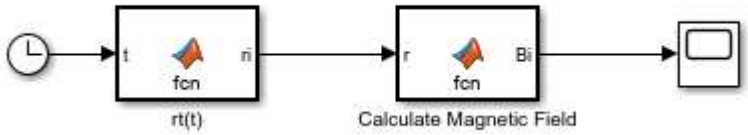
Izaac Facundo

Imf339

HW#8

PROBLEM 1

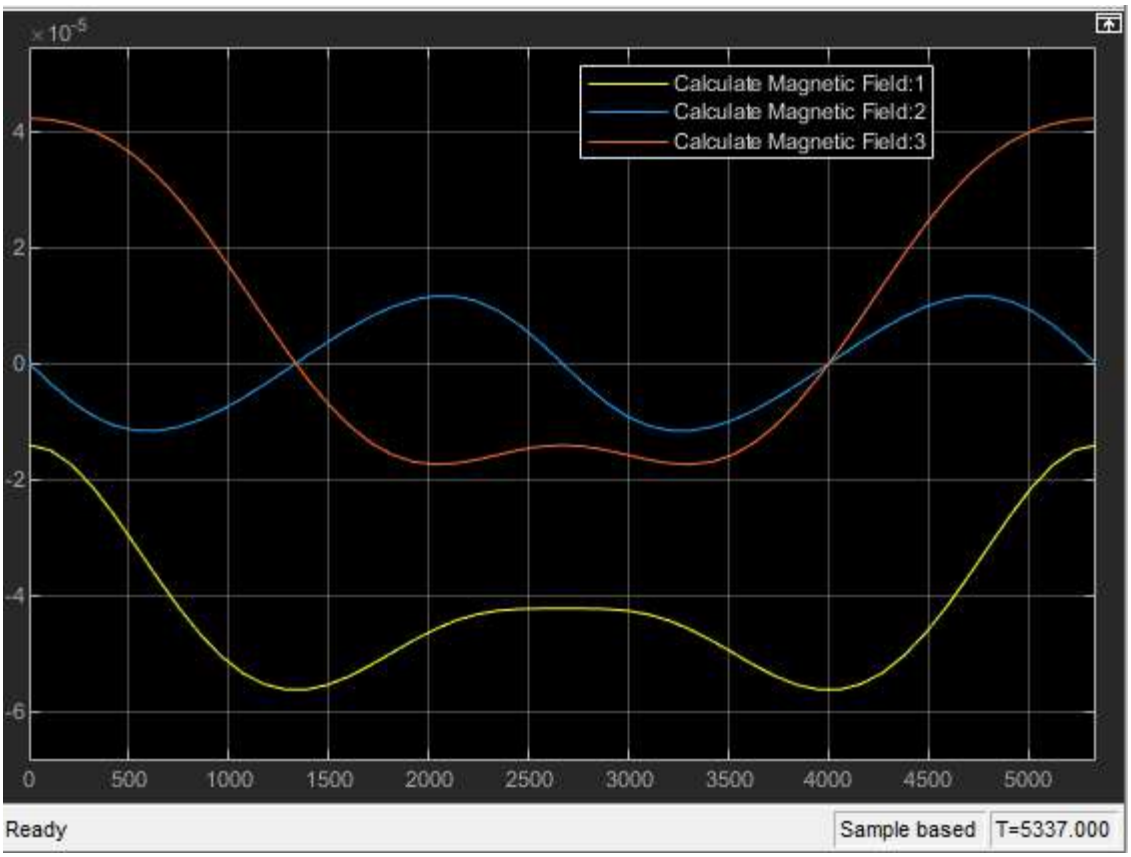


ADHW8p1 ▸ rt(t)

```
1   function ri = fcn(t)
2
3   p = 6600;
4   mu = 398600;
5   w = sqrt(mu / (p^3));
6
7   ri = (p/sqrt(2)) * [-cos(w*t); sqrt(2)*sin(w*t); cos(w*t)];
8
9
```

8p1 ▸ Calculate Magnetic Field

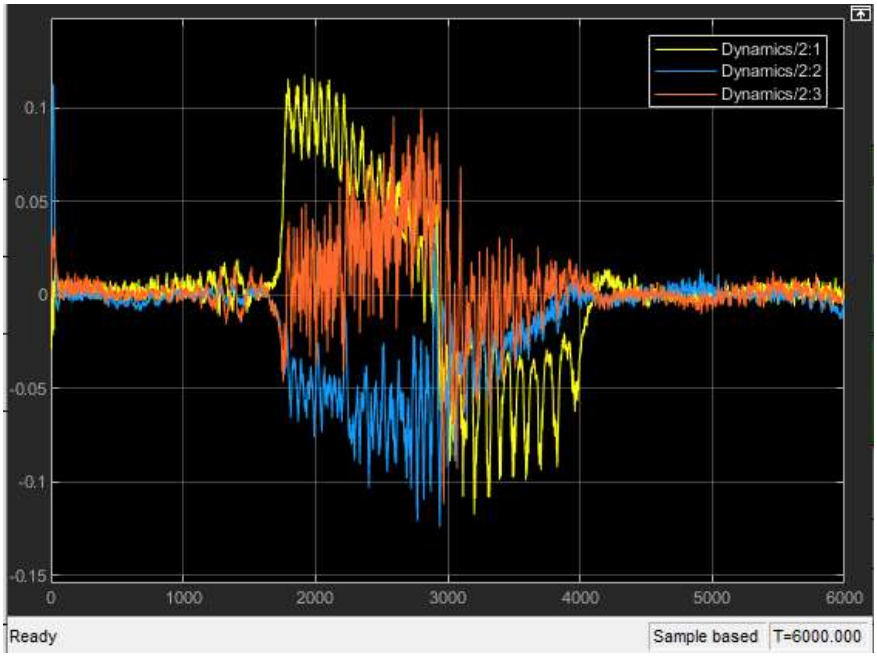
```
function Bi = fcn(r)

Re = 6378; % km
B0 = 3.12e-5; % T

nz = -r./norm(r);
ny = cross(nz, [0; 0; 1]) ./ norm(cross(nz, [0; 0; 1]));
nx = cross(ny,nz);
Ti2n = [nx'; ny'; nz'];

lat = acos(r(3)/norm(r));

Bn = (B0*(Re/norm(r))^3) * [cos(lat); 0; -2*sin(lat)];

Bi = Ti2n*Bn;
```


×10^-5
Calculate Magnetic Field:1
Calculate Magnetic Field:2
Calculate Magnetic Field:3
4
2
0
-2
-4
-6
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
Ready
Sample based
T=5337.000

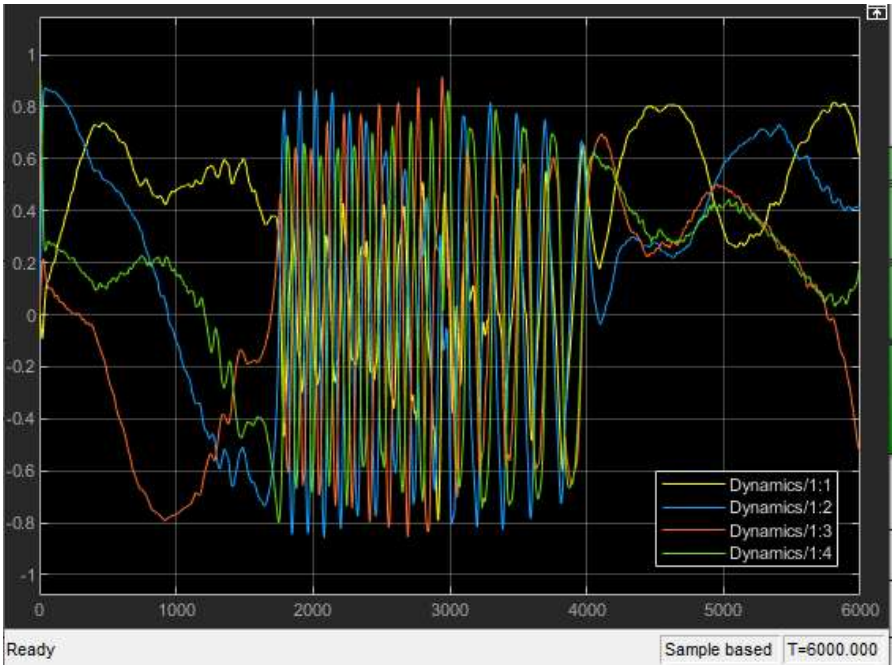
Problem 2

Graphs first, then code

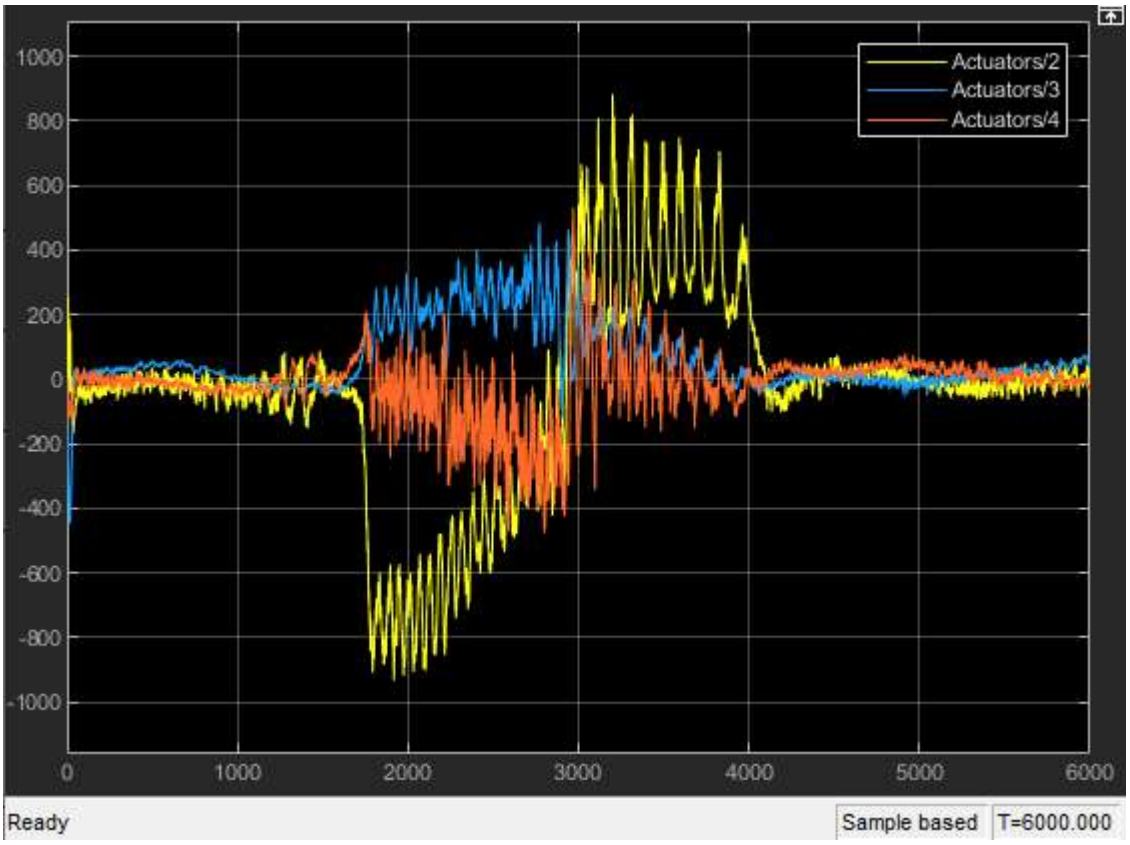
Angular Velocity in body frame v. time



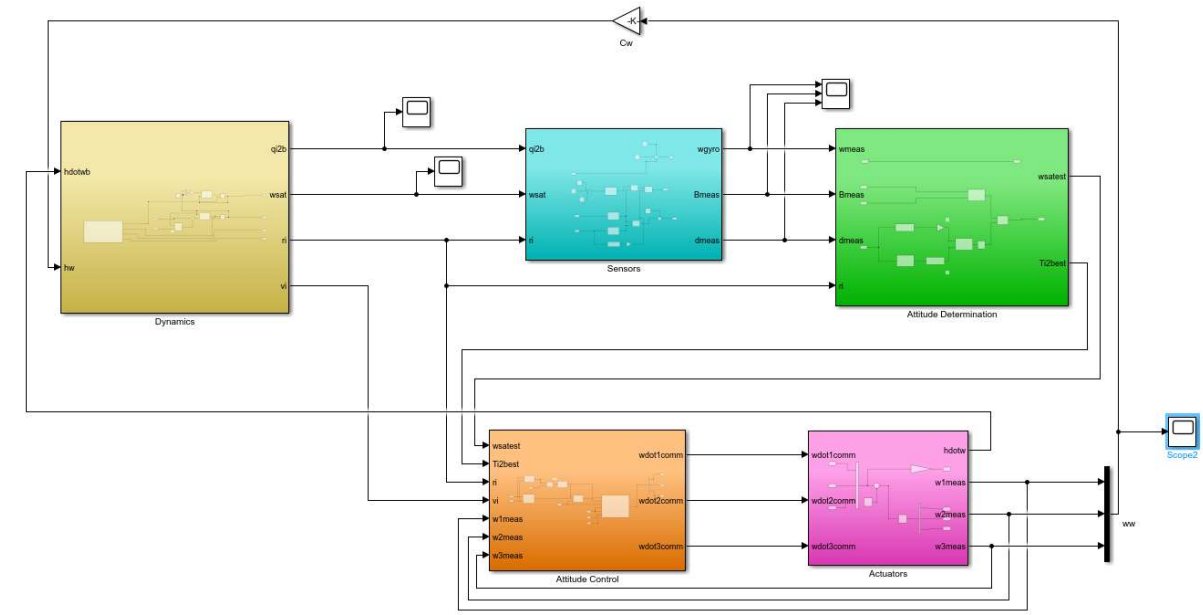
Inertial to body Quaternion



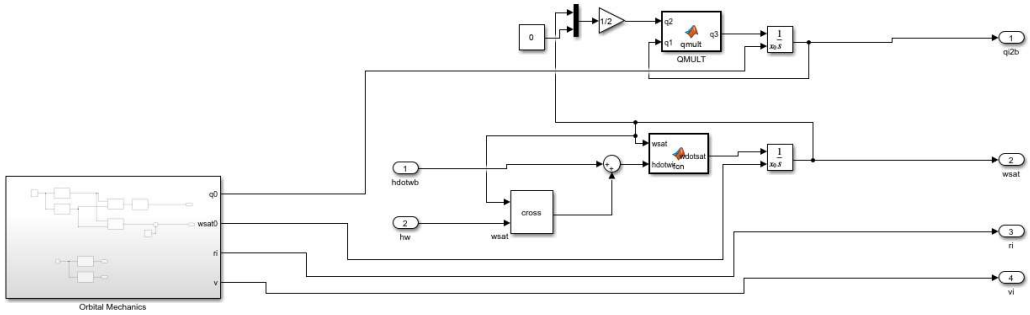
Angular velocities of the wheels (wheel 1 yellow, etc.)



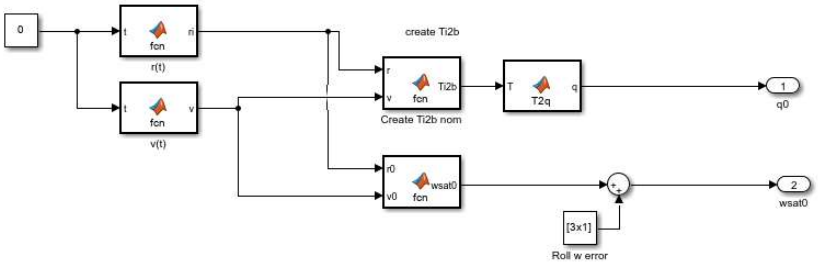
CODE

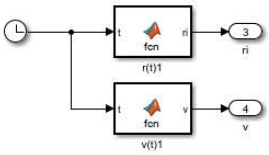

-K-
Cw
qi2b
wsat
ri
vi
hdotwb
hw
Dynamics
qi2b
wsat
ri
wgyro
Bmeas
dmeas
Sensors
wmeas
Bmeas
dmeas
ri
wsatest
Ti2best
Attitude Determination
wsatest
Ti2best
ri
vi
w1meas
w2meas
w3meas
wdot1comm
wdot2comm
wdot3comm
Attitude Control
wdot1comm
wdot2comm
wdot3comm
hdotw
w1meas
w2meas
w3meas
Actuators
ww
Scope2

Dynamics ▸


0
1/2
q2
q1
s3
qmult
QMULT
1
qi2b
wsat
hdotwb
hdotsat
2
wsat
1
hdotwb
2
hw
cross
wsat
3
ri
4
vi
q0
wsat0
ri
v
Orbital Mechanics

Orbital Mechanics





Dynamics ▸ Orbital Mechanics ▸ r(t)

```
function ri = fcn(t)

p = 6600;
mu = 398600;
w = sqrt(mu / (p^3));

ri = (p/sqrt(2)) * [-cos(w*t); sqrt(2)*sin(w*t); cos(w*t)];
```

Dynamics ▸ Orbital Mechanics ▸ v(t)

```
function v = fcn(t)

p = 6600;
mu = 398600;
w = sqrt(mu / (p^3));

v = (w*p/sqrt(2)) * [sin(w*t); sqrt(2)*cos(w*t); -sin(w*t)];
```

Create Ti2b nom

3p2 ▸ Dynamics ▸ Orbital Mechanics ▸ Create Ti2b nom

```
function Ti2b = fcn(r,v)

xib = -r / norm(r);
yib = v / norm(v);
zib = cross(xib,yib);

Ti2b = [xib'; yib'; zib'];
```

MATLAB Function1

p2 ▸ Dynamics ▸ Orbital Mechanics ▸ MATLAB Function1

```
function q = T2q(T)

q = zeros(4,1);

q(4) = sqrt(.25*(1 + T(1,1) + T(2,2) + T(3,3)));

q = (1/(4*q(4))) .* [T(2,3) - T(3,2); T(3,1) + T(1,3); T(1,2) + T(2,1); 4 * q(4)^2 ];

end
```

MATLAB Function

p2 ▸ Dynamics ▸ Orbital Mechanics ▸ MATLAB Function

```
function wsat0= fcn(r0,v0)
mu = 398600.0;
rho = 6600;
wmag = sqrt(mu / rho^3);

h_orbit = cross(r0,v0); % this is the axis our satellite is orbiting about
axis = h_orbit / norm(h_orbit);

wsat0 = wmag .* axis;
```

QMULT

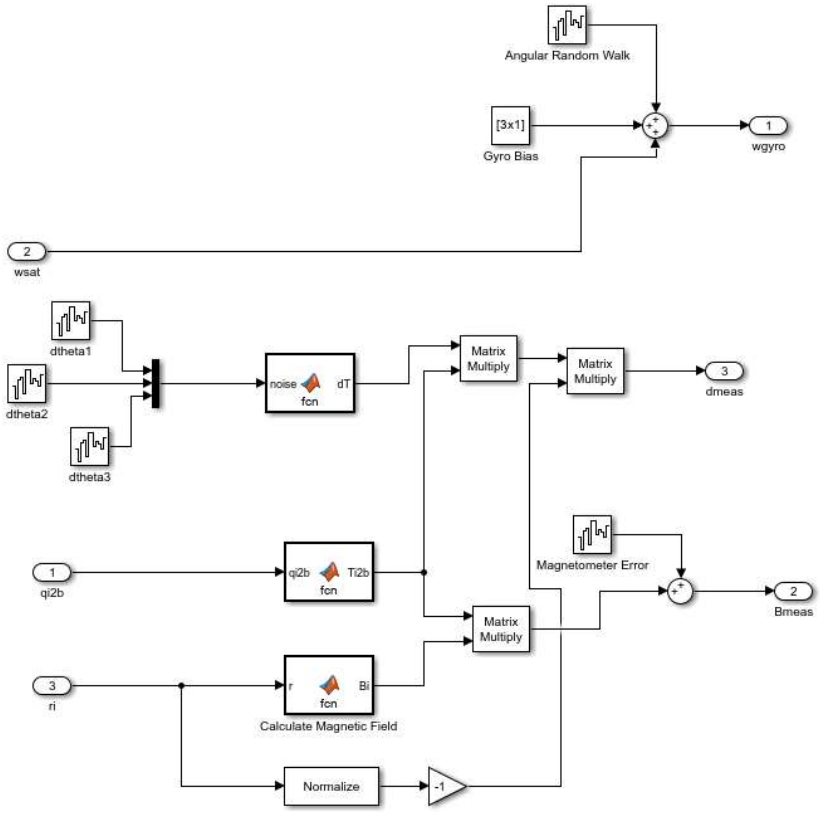
3p2 ▸ Dynamics ▸ QMULT

```
function q3 = qmult(q2,q1)

q3 = [q2(4).*q1(1:3) + q1(4).*q2(1:3) - cross(q2(1:3),q1(1:3));
    q2(4)*q1(4) - dot(q2(1:3),q1(1:3))];

end
```

p2 ▸ Dynamics ▸ MATLAB Function

```
function wdotsat = fcn(wsat,hdotwi)

J = [1000 0 0;
     0 500 0;
     0 0 600];

wdotsat = inv(J) * (cross(-wsat,(J*wsat)) - hdotwi); %#ok<MINV>
end
```

Sensors ▸



p2 ▸ Sensors ▸ MATLAB Function1

```
function dT = fcn(noise)

angle = norm(noise);

axis = noise/norm(noise);

dT = eye(3) - sin(angle)*skewsym(axis) + (1 - cos(angle)) * skewsym(axis)^2;

end

function skewed = skewsym(v)
skewed = [0 -v(3) v(2);
          v(3) 0 -v(1);
          -v(2) v(1) 0];
end
```

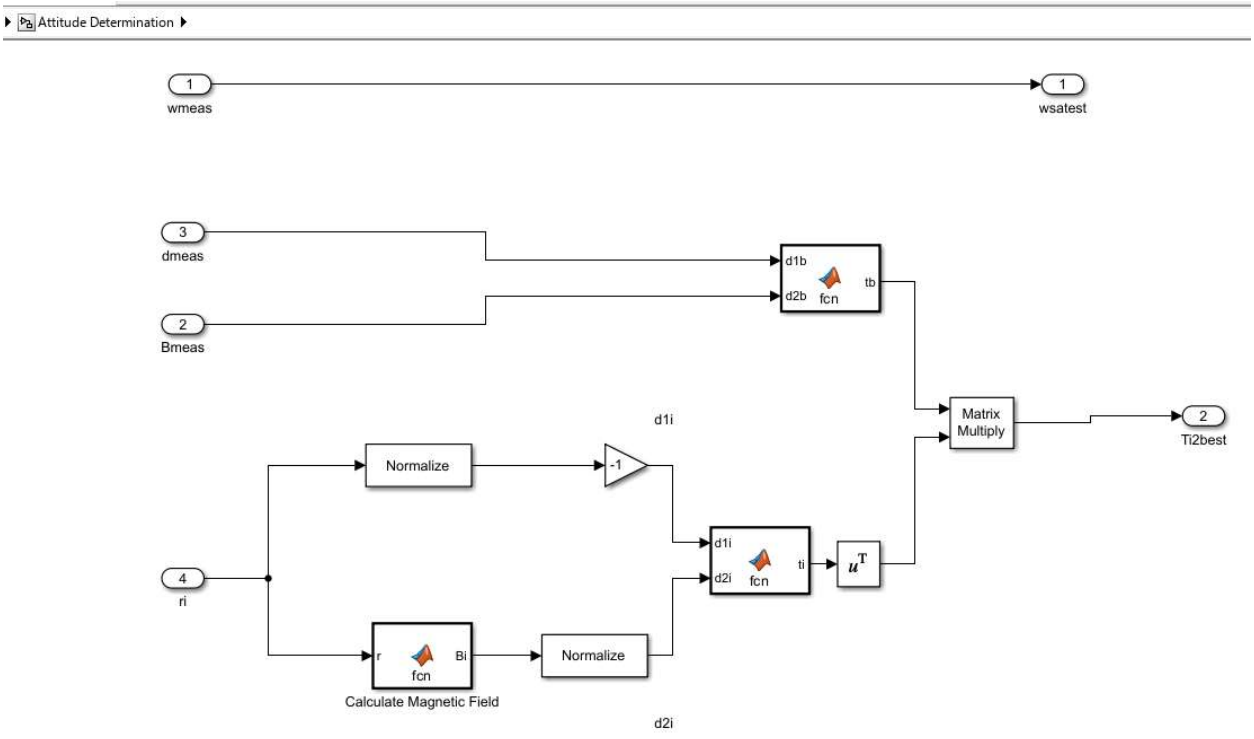
p2 ▸ Sensors ▸ MATLAB Function

```
function Ti2b = fcn(qi2b)

I = eye(3);
qs = qi2b(4);
qv = [qi2b(1:3)];

Ti2b = I - 2*qs*skewsym(qv) + 2*(skewsym(qv))^2;

end

function skewed = skewsym(v)
skewed = [0 -v(3) v(2);
          v(3) 0 -v(1);
          -v(2) v(1) 0];
end
```

Calculate Magnetic Field

8p2 ▸ Sensors ▸ Calculate Magnetic Field

```
function Bi = fcn(r)

Re = 6378; % km
B0 = 3.12e-5; % T

nz = -r./norm(r);
ny = cross(nz, [0; 0; 1]) ./ norm(cross(nz, [0; 0; 1]));
nx = cross(ny,nz);
Ti2n = [nx'; ny'; nz'];

lat = acos(r(3)/norm(r));

Bn = (B0*(Re/norm(r))^3) * [cos(lat); 0; -2*sin(lat)];

Bi = Ti2n*Bn;
```



'8p2 ▸ Attitude Determination ▸ MATLAB Function

```
function tb = fcn(d1b,d2b)

tbx = d1b;

tbz = cross(d1b,d2b) / norm(cross(d1b,d2b));

tby = cross(tbz,tbx);

tb = [tbx tby tbz];

end
```

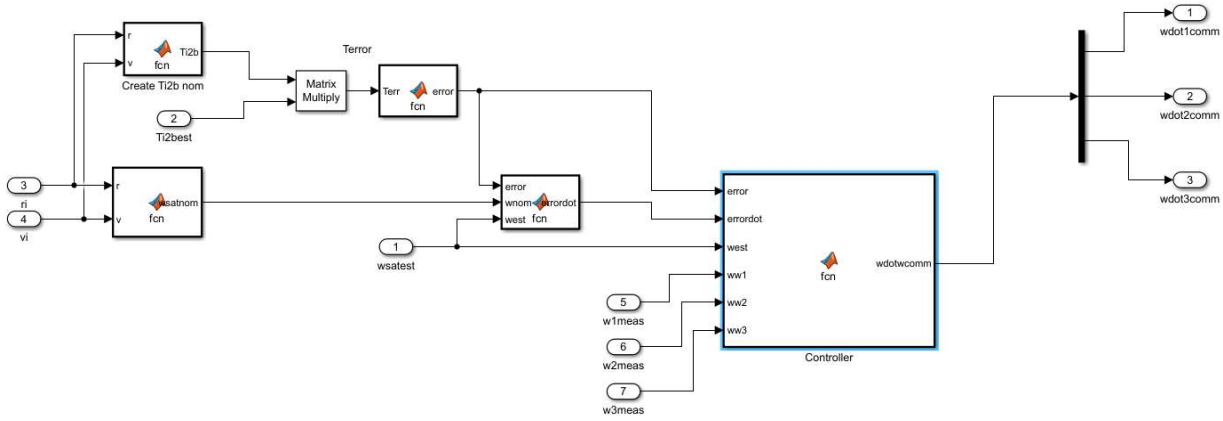
p2 ▸ Attitude Determination ▸ MATLAB Function1

```
function ti = fcn(d1i,d2i)

tix = d1i;

tiz = cross(d1i,d2i) / norm(cross(d1i,d2i));

tiy = cross(tiz,tix);

ti = [tix tiy tiz];
```

▸ Attitude Determination ▸ Calculate Magnetic Field

```
function Bi = fcn(r)

Re = 6378; % km
B0 = 3.12e-5; % T

nz = -r./norm(r);
ny = cross(nz, [0; 0; 1]) ./ norm(cross(nz, [0; 0; 1]));
nx = cross(ny,nz);
Ti2n = [nx'; ny'; nz'];

lat = acos(r(3)/norm(r));

Bn = (B0*(Re/norm(r))^3) * [cos(lat); 0; -2*sin(lat)];

Bi = Ti2n*Bn;
```

W8p2 ▸ Attitude Control ▸



8p2 ▸ Attitude Control ▸ Create Ti2b nom

```
function Ti2b = fcn(r,v)

xib = -r / norm(r);
yib = v / norm(v);
zib = cross(xib,yib);

Ti2b = [xib'; yib'; zib'];
```

3p2 ▸ Attitude Control ▸ MATLAB Function

```
function wsatnom = fcn(r,v)
mu = 398600.0;
rho = 6600;
wmag = sqrt(mu / rho^3);

h_orbit = cross(r,v); % this is the axis our satellite is orbiting about
axis = h_orbit / norm(h_orbit);

wsatnom = wmag .* axis;
```

p2 ▸ Attitude Control ▸ MATLAB Function1

```
function error = fcn(Terr)

error = -0.5 * [Terr(3,2) - Terr(2,3); Terr(1,3) - Terr(3,1); Terr(2,1) - Terr(1,2)];
```

o2 ▸ Attitude Control ▸ MATLAB Function2

```
function errordot = fcn(error,wnom,west)

errordot = -skewsym(wnom)*error + (wnom - west);

end
function skewed = skewsym(v)
skewed = [0 -v(3) v(2);
          v(3) 0 -v(1);
          -v(2) v(1) 0];
end
```

p2 ▸ Attitude Control ▸ Controller

```
function wdotwcomm = fcn(error,errordot,west,ww1,ww2,ww3)
Jsat = [1000 0 0; 0 500 0; 0 0 600];
Kd = -0.05*eye(3);
Kp = -0.2*eye(3);
Cw = .1295;
wdotwcomm = (Jsat*Kd ./ Cw) * error + (Jsat*Kp ./ Cw) * errordot - cross(west, [ww1; ww2; ww3]);
```

Actuators