ADCS Project Part 2

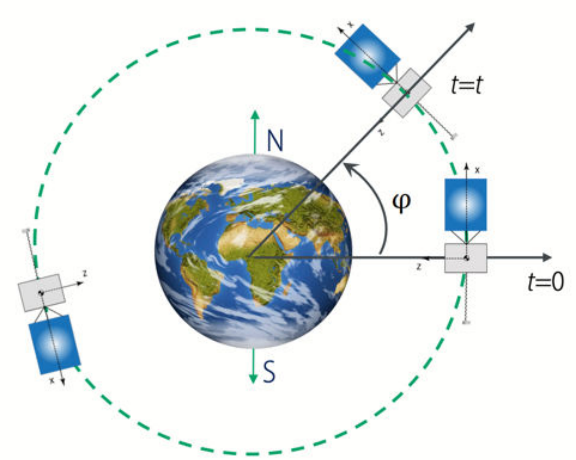
Dillon Allen

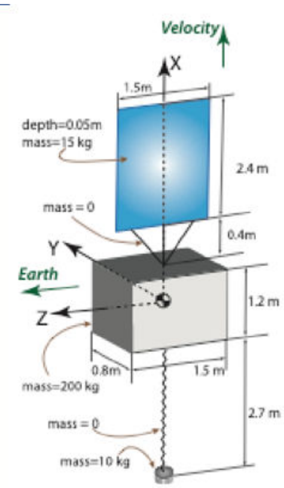
Important Data from last time

```
clear, clc, close all;
orbital_rate       = 0.0010; % rad/s
wheel.maxTorque    = 0.025; % N-m
wheel.maxMomentum = 0.5; % N-m-s
wheel.maxSpeed     = 6000; % rpm
period = 6000;
rE = 6378.14;
r_orbit = 750;
a = rE + r_orbit;
```

Problem 13

Show spacecraft moment of inertias along the three axes are 51.0, 253.0, and 223.4 kg-m^2 respectively.





3

Calculating the inertias and applying parallel axis theorem, we have

```
% following the layout for the solution to pset 4
masses = [200 15 10];
dPanel = (1.2/2) + 0.4 + (2.4/2);
dPM = (1.2/2) + 2.7;
r_to_bc = [0 0 0; ...
           dPanel 0 0];
sizes = [1.2 0.8 1.5; ...
         2.4 0.05 1.5]
```

```
sizes = 2×3
    1.2000    0.8000    1.5000
    2.4000    0.0500    1.5000
```

```
I_pm = masses(3).*[0 0 0; 0 dPM^2 0; 0 0 dPM^2];

I_syst = zeros(3,3);
```

```
for ii = 1:2
    Irect(:,:,ii) = [masses(ii)/12*(sizes(ii,2)^2 + sizes(ii,3)^2) 0 0;...
                     0 masses(ii)/12*(sizes(ii,3)^2 + sizes(ii,1)^2) 0;...
                     0 0 masses(ii)/12*(sizes(ii,1)^2 + sizes(ii,2)^2)];
    % Parallel Axis Theorem
    Irect(:,:,ii) = Irect(:,:,ii) + ...
        masses(ii)*[r_to_bc(ii,2)^2+r_to_bc(ii,3)^2 0 0;...
                    0 r_to_bc(ii,1)^2+r_to_bc(ii,3)^2 0;...
                    0 0 r_to_bc(ii,1)^2+r_to_bc(ii,2)^2];
    I_syst = Irect(:,:,ii) + I_syst;
end

I_syst = I_syst + I_pm;
I_syst
```

```
I_syst = 3x3
   50.9823         0         0
         0  253.0125         0
         0         0  223.3698
```

```
sc.Ix = I_syst(1,1);
sc.Iy = I_syst(2,2);
sc.Iz = I_syst(3,3);
fprintf("(Ix, Iy, Iz): (%f, %f, %f)", sc.Ix, sc.Iy, sc.Iz);
```

```
(Ix, Iy, Iz): (50.982292, 253.012500, 223.369792)
```

Problem 14

Calculate the inertia of each wheel

We can calculate the inertia at max momenta, i.e

$$h_{max} = I_w \omega_{max}$$

```
wheel.maxSpeedrps = wheel.maxSpeed*(2*pi/1)*(1/60);
wheel.inertia = wheel.maxMomentum / wheel.maxSpeedrps;
fprintf("Wheel ineratia is: %f {kg-m^2}", wheel.inertia);
```

```
Wheel ineratia is: 0.000796 {kg-m^2}
```

Problem 15

Calculate the maximum spacecraft acceleration ($\dot{\omega}$) due to the wheel along each axis

To calculate this, we will use

$$I_x \dot{\omega}_x = T_{max}$$

```
sc.accelX = wheel.maxTorque / sc.Ix;
sc.accelY = wheel.maxTorque / sc.Iy;
sc.accelZ = wheel.maxTorque / sc.Iz;
fprintf("Spacecraft accelerations: (%f, %f, %f) {rad/s^2}", ...
```

```
    sc.accelX, sc.accelY, sc.accelZ);
```

```
Spacecraft accelerations: (0.000490, 0.000099, 0.000112) {rad/s^2}
```

Problem 16

What is the momentum needed to maintain nadir pointing?

Since nadir pointing is along the -Y axis, we will use the orbital rate times the inertia in the Y axis to see the momenta needed.

```
sc.nadirMomentum = sc.Iy * orbital_rate;
fprintf("Momentum needed for Nadir pointing: %f {N-m-s}", sc.nadirMomentum);
```

```
Momentum needed for Nadir pointing: 0.253013 {N-m-s}
```

Problem 17

What wheel speed would that level of momentum correspond to?

```
wheel.nadirSpeedrps = sc.nadirMomentum / wheel.inertia; % rad/s
wheel.nadirSpeed    = wheel.nadirSpeedrps * (1/(2*pi))*60;
fprintf("Wheel speed for nadir pointing: %f {rad/s}, %f {rpm}", ...
    wheel.nadirSpeedrps, wheel.nadirSpeed)
```

```
Wheel speed for nadir pointing: 317.944885 {rad/s}, 3036.150000 {rpm}
```

Problem 18

Find the PID gains for each axis, given parameters of $\omega_n = 0.5$ (rad/s), $\zeta = 0.7$, $T = \dfrac{10}{\zeta \omega_n}$ (s)

```
% Control parameters
K.omega_n = 0.5; % rad/s
K.zeta    = 0.7;
K.T       = 10/(K.omega_n * K.zeta); % seconds

% Gain Terms
p_term = K.omega_n^2 + (2*K.omega_n*K.zeta)/K.T;
i_term = K.omega_n^2 / K.T;
d_term = 2*K.omega_n*K.zeta + 1/K.T;

% Axis Gains
% X-AXIS
K.px = sc.Ix * p_term;
K.ix = sc.Ix * i_term;
K.dx = sc.Ix * d_term;

% Y-AXIS
K.py = sc.Iy * p_term;
K.iy = sc.Iy * i_term;
K.dy = sc.Iy * d_term;
```

```
% Z-AXIS
K.pz = sc.Iz * p_term;
K.iz = sc.Iz * i_term;
K.dz = sc.Iz * d_term;

fprintf("X-Axis gains")
```

```
X-Axis gains
```

```
fprintf("Kp: %f", K.px);
```

```
Kp: 13.994639
```

```
fprintf("Ki: %f", K.ix);
```

```
Ki: 0.446095
```

```
fprintf("Kd: %f", K.dx);
```

```
Kd: 37.471984
```

```
fprintf("Y-Axis gains")
```

```
Y-Axis gains
```

```
fprintf("Kp: %f", K.py);
```

```
Kp: 69.451931
```

```
fprintf("Ki: %f", K.iy);
```

```
Ki: 2.213859
```

```
fprintf("Kd: %f", K.dy);
```

```
Kd: 185.964188
```

```
fprintf("Z-Axis gains")
```

```
Z-Axis gains
```

```
fprintf("Kp: %f", K.pz);
```

```
Kp: 61.315008
```

```
fprintf("Ki: %f", K.iz);
```

```
Ki: 1.954486
```

```
fprintf("Kd: %f", K.dz);
```

```
Kd: 164.176797
```

Problem 19

Derive the state space equations:

The linearized equations of motion can be taken from the notes, and is seen as

$$
\begin{aligned}
T_{dx}+T_{cx} &= I_x\ddot{\phi}+4\omega_o^2(I_y-I_z)\phi+\omega_o(I_y-I_z-I_x)\dot{\psi}+\dot{h}_{wx}-\omega_o h_{wz}\\
&\quad -\dot{\psi}h_{wyo}-\phi\omega_o h_{wyo}-I_{xy}\ddot{\theta}-I_{xz}\ddot{\psi}-I_{xz}\omega_o^2\psi+2I_{yz}\omega_o\dot{\theta},\\
T_{dy}+T_{cy} &= I_y\ddot{\theta}+3\omega_o^2(I_x-I_z)\theta+\dot{h}_{wy}\\
&\quad -I_{xy}(\ddot{\phi}-2\omega_o\dot{\psi}-\omega_o^2\phi)+I_{yz}(-\ddot{\psi}-2\omega_o\dot{\phi}+\omega_o^2\psi),\\
T_{dz}+T_{cz} &= I_z\ddot{\psi}+\omega_o(I_z+I_x-I_y)\dot{\phi}+\omega_o^2(I_y-I_x)\psi+\dot{h}_{wz}+\omega_o h_{wx}\\
&\quad +\dot{\phi}h_{wyo}-\psi\omega_o h_{wyo}-I_{yz}\ddot{\theta}-I_{xz}\ddot{\phi}-2\omega_o I_{xy}\dot{\theta}-\omega_o^2 I_{xz}\phi.
\end{aligned}
$$

We will take a small angular rate (approximately zero), and we know that our cross term MOIs are zero, since we are working with principal MOIs only. With small angles, all products of angular velocities are zero, so we can eliminate those as well. Deleting all the zero value terms, we are left with the equations

$$T_{c,i}+T_{d,i}=I_i\frac{d}{dt}(\omega_i),\ \ i=x,y,z$$

$$T_{c,i}=I_{wh}\frac{d}{dt}(\Omega_i),\ \ i=x,y,z$$

and the control law is written as (y-axis as an example)

$$T_{cy}=K_{py}\theta+K_{iy}\int\theta+K_{dy}\dot{\theta}$$

Write the state space equations for each axis

placing the equations above into matrix form, we have

```
% X - Axis
Ax = [0 1 0 0; ...
      0 0 1 0;...
     -K.ix/sc.Ix         -K.px/sc.Ix         -K.dx/sc.Ix          0; ...
      K.ix/wheel.inertia K.px/wheel.inertia K.dx/wheel.inertia  0];
Bx = [0; 0; 1/sc.Ix; 0];
Cx = [0 180/pi 0       0; ...
      0   0    0      30/pi; ...
      0   0    0   wheel.inertia];
Dx = zeros(3,1);

% Y - Axis
Ay = [0 1 0 0; ...
      0 0 1 0;...
     -K.iy/sc.Iy         -K.py/sc.Iy         -K.dy/sc.Iy          0; ...
      K.iy/wheel.inertia K.py/wheel.inertia K.dy/wheel.inertia  0];
By = [0; 0; 1/sc.Iy; 0];
Cy = [0 180/pi 0       0; ...
      0   0    0      30/pi; ...
      0   0    0   wheel.inertia];
Dy = zeros(3,1);

% Z - Axis
Az = [0 1 0 0; ...
```

```
      0 0 1 0;...
     -K.iz/sc.Iz         -K.pz/sc.Iz         -K.dz/sc.Iz          0; ...
      K.iz/wheel.inertia K.pz/wheel.inertia K.dz/wheel.inertia  0];
Bz = [0; 0; 1/sc.Iz; 0];
Cz = [0 180/pi 0        0; ...
      0   0    0       30/pi; ...
      0   0    0   wheel.inertia];
Dz = zeros(3,1);
```

Problem 20

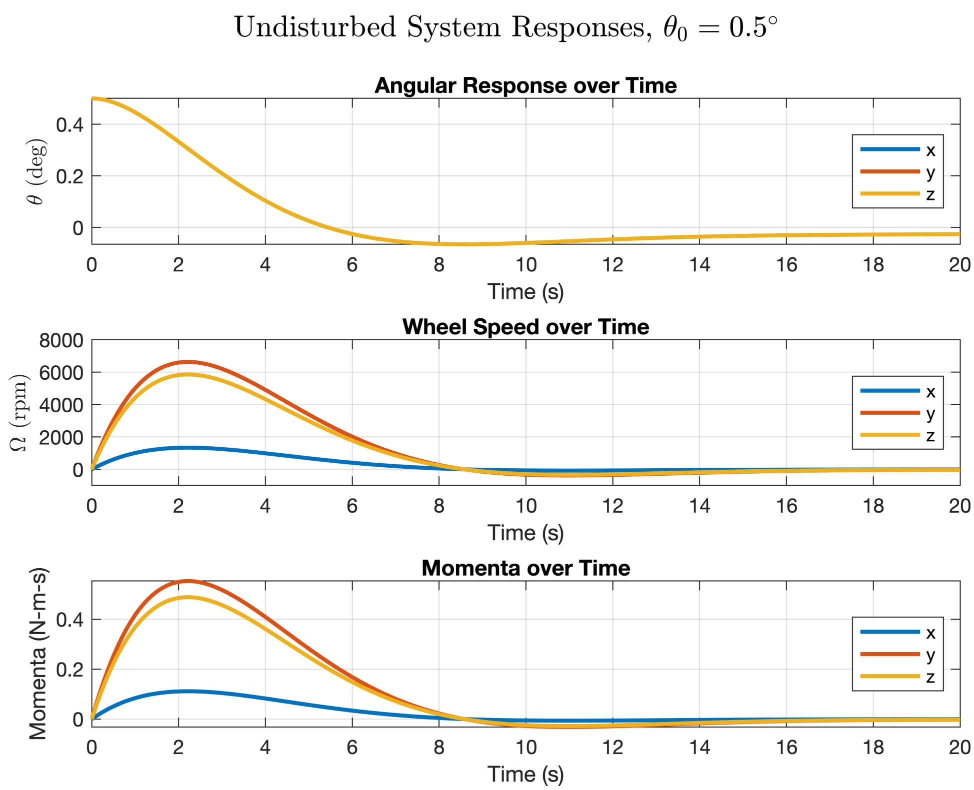
Create a state-space transfer function for each axis

```
Gx = ss(Ax, Bx, Cx, Dx);
Gy = ss(Ay, By, Cy, Dy);
Gz = ss(Az, Bz, Cz, Dz);
```

Problem 21

Use lsim to plot the response along each axis

```
t = 0:0.1:20;
x0 = [0; deg2rad(0.5); 0; 0];
u = zeros(1,length(t));
x = lsim(Gx, u, t, x0);
y = lsim(Gy, u, t, x0);
z = lsim(Gz, u, t, x0);
responses = {x y z};
figure();
for i = 1:3
    subplot(3,1,i);
    for j = 1:3
        hold on
        plot(t, responses{j}(:,i), 'LineWidth',2);
        xlabel("Time (s)");
        if i == 1
            ylabel("$\theta$ (deg)",'Interpreter','latex');
            title("Angular Response over Time")
        elseif i == 2
            ylabel("$\Omega$ (rpm)", 'Interpreter','latex');
            title("Wheel Speed over Time");
            ylim([-1000, 8000])
        else
            ylabel("Momenta (N-m-s)");
            title("Momenta over Time");
        end
    end
    legend(["x", "y","z"], 'Location','best');
    grid on; box on;
    hold off
end
sgtitle("Undisturbed System Responses, $\theta_0 = 0.5^{\circ}$",...
        'Interpreter','latex')
```



Problem 22

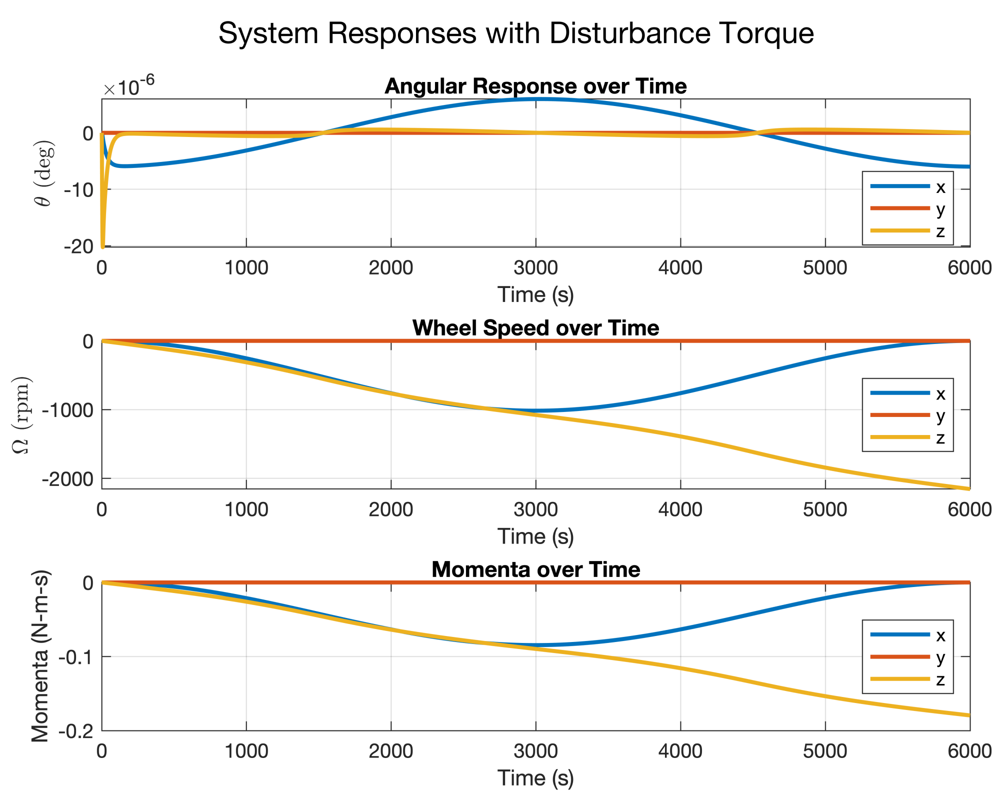
Model the system response with the disturbances calculated earlier

```
% Recreate the disturbance forces
% solar disturbance from part1 sol
solarDisturbance = [0 0 -4.455e-05];
M = csvread("Satellite3_LLA_Position_1Period_1s.csv",1,1);
distTime = 0:1:length(M)-1;
solarMat = repmat(solarDisturbance, length(distTime),1);
lat = M(:,1);
B0 = 31200E-9; % Teslas
m_dipole = 1; % A-T-m^2
m_dipole_vec = m_dipole .* [0 -1 0];
R = a/rE;
B = B0/ R^3;
Bz = -2*B*sind(lat);
By = zeros(size(lat));
Bx = B*cosd(lat);
Bfield = [Bx By -Bz];
rep_mag = repmat(m_dipole_vec,length(lat),1);
magField = cross(rep_mag, Bfield );

disturbanceMat = magField + solarMat;

ux = disturbanceMat(:,1)';
uy = disturbanceMat(:,2)';
uz = disturbanceMat(:,3)';
```

```
xDist = lsim(Gx, ux, distTime);
yDist = lsim(Gy, uy, distTime);
zDist = lsim(Gz, uz, distTime);
distResponses = {xDist, yDist, zDist};
figure();
for i = 1:3
    subplot(3,1,i);
    for j = 1:3
        hold on
        plot(distTime, distResponses{j}(:,i), 'LineWidth',2);
        xlabel("Time (s)");
        if i == 1
            ylabel("$\theta$ (deg)",'Interpreter','latex');
            title("Angular Response over Time")
        elseif i == 2
            ylabel("$\Omega$ (rpm)", 'Interpreter','latex');
            title("Wheel Speed over Time");
        else
            ylabel("Momenta (N-m-s)");
            title("Momenta over Time");
        end
    end
    legend(["x", "y","z"], 'Location','best')
    grid on; box on;
    hold off
end
sgtitle("System Responses with Disturbance Torque");
```



Problem 23

How long does it take to slew?

```
wheel.cappedSpeed = 4000 * (pi/30); % rad/s
t1 = (wheel.inertia * (-wheel.cappedSpeed - 0))/(-wheel.maxTorque);
fprintf("Time to get to -4000 rpm: %f {sec}", t1);
```

```
Time to get to -4000 rpm: 13.333333 {sec}
```

```
thetaT1Slew = rad2deg(wheel.maxTorque/(2*sc.Iy)*t1^2);
dTheta = 90 - thetaT1Slew;
t2 = sqrt(2*sc.Iy*deg2rad(dTheta)/wheel.maxTorque);
totalTime = 2*(t1 + t2);
fprintf("Total time to slew 180 degrees: %f {sec}",totalTime);
```

```
Total time to slew 180 degrees: 382.288436 {sec}
```