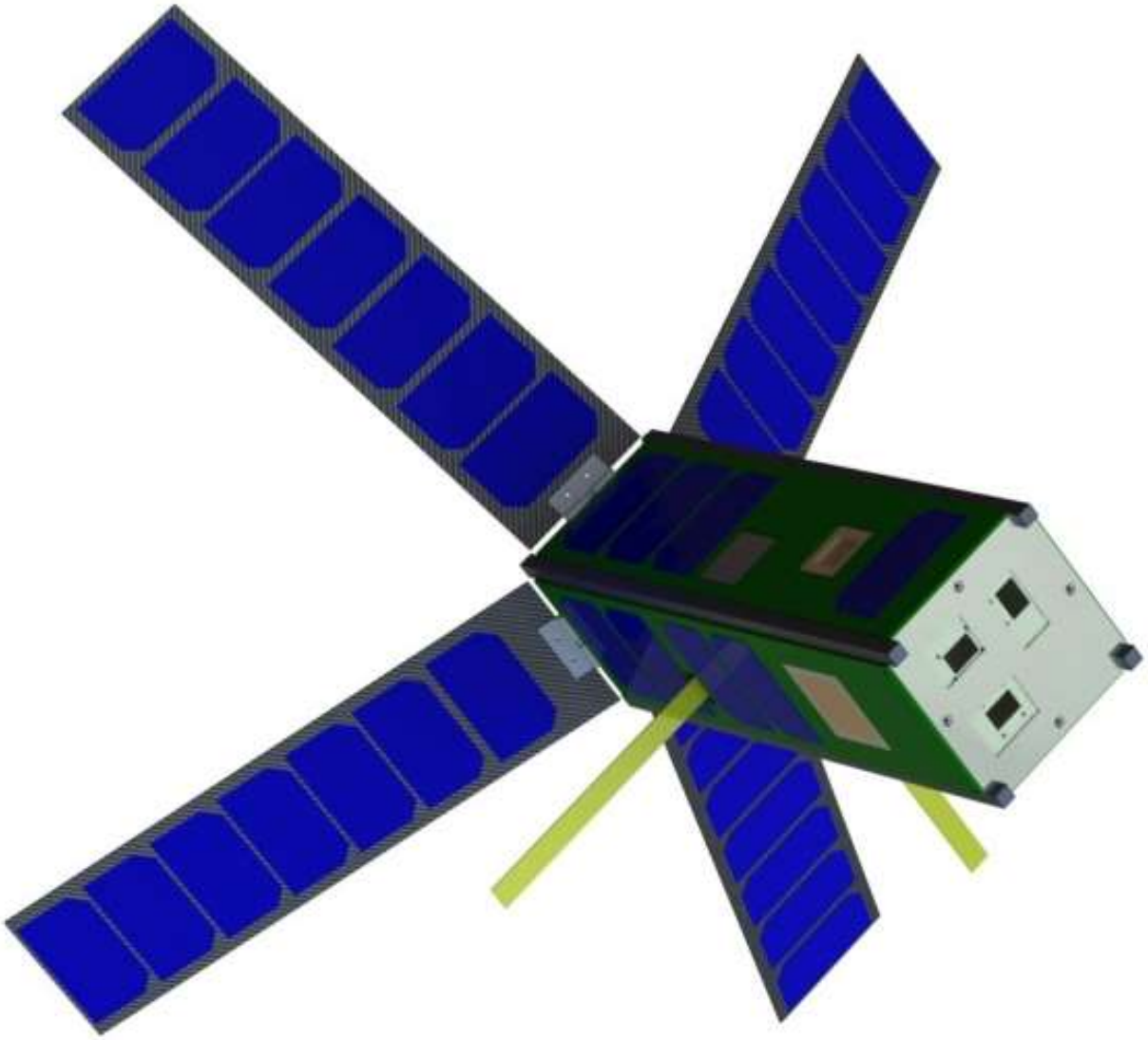
CADRE: Disturbance Model



DUNCAN MILLER

Advised By: Chas Galey

Winter 2012

BACKGROUND

The CubeSat Investigating Atmospheric Density Response to Extreme Driving (CADRE) is the next spacecraft mission under development at the University of Michigan. The overarching science mission is to characterize the thermosphere to better predict orbital trajectories of free bodies. CADRE's primary payload, WINCS, is a sensitive instrument that can measure solar winds in the ionosphere using a collection of mass spectrometers. In order for its data to be reliable it needs to maintain a 1 degree of pointing accuracy (2 degree cone) with a 0.1 degree determination. These high tolerances demand a very robust ADCS architecture.

COORDINATE DEFINITONS

Most CADRE documentation follows the body fixed coordinate system in Figure 1 (left) to comply with P-Pod requirements. However, in the scope of this analysis, I use the coordinate system shown in Figure 1 (right). This defines +Z as the direction of motion, +X away from the Earth, and +Y completes the right hand rule.

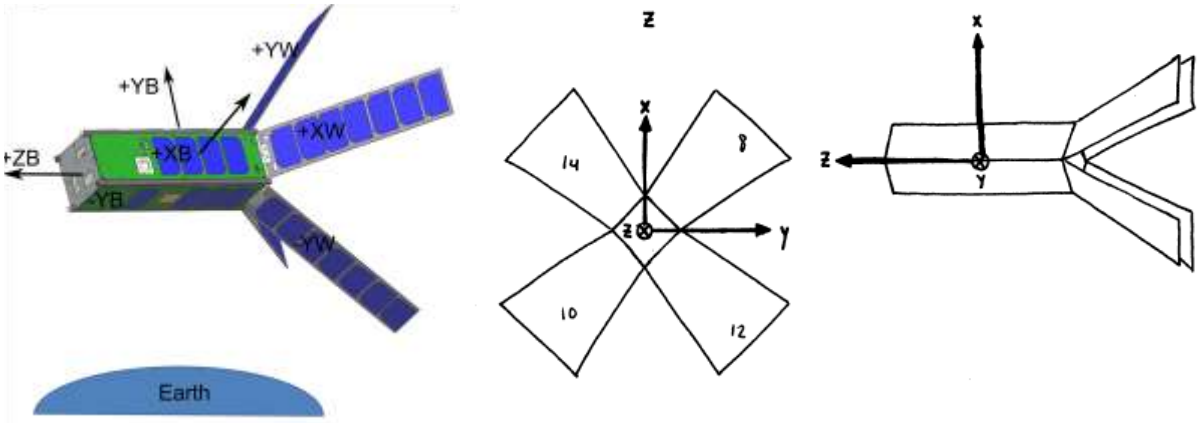


Figure 1: Body Fixed Coordinate Systems for P-Pod (left) and for orbit analysis (right)

The distinction is important, because CADRE flies (unusually) at a 45 degree angle with respect to the radial direction. This orientation is advantageous because we can downlink from both patch antennas to the ground without the use of a slew maneuver. The drawback to this is that Structures has had to mount the momentum wheels at a 45 degree angle within the spacecraft, which is necessary to save power.

Using the second (hand drawn) coordinate system, mass moments of inertia were calculated in my SolidWorks model. As body fixed quantities, they are constant in the body fixed frame (BFF). Since the equations of motion for rotational dynamics are derived in the BFF, the mass moments of inertia are constant in this frame.

OUTER LOOP CONTROL

WINCS collects measurements of the ion winds, which show greatest fluctuations over the poles. For this reason, CADRE is expected to fly in a 500 km altitude orbit with an inclination between 45 degrees and 90 degrees. I have written a code that solves the equations of motion (acted on

purely by gravity) of the center of mass and resolves the position of CADRE over the course of 10 orbits, at 90 degrees inclination. Graphically, Figure 2 shows the flight path of CADRE. CADRE's final orbital characteristics will only be determined after we have been matched with a specific rocket.

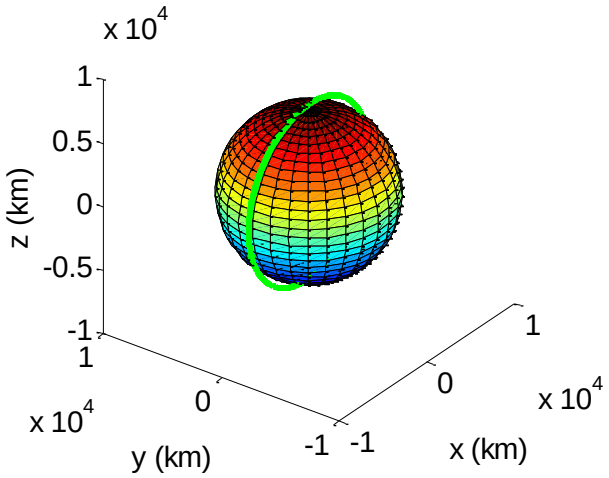


Figure 2: Nominal orbit about Earth

INNER LOOP STABILIZATION

Although the center of mass is expected to follow the path shown in Figure 2, the orientation of CADRE must be perfect if the science measurements from WINCS can be used. There is a ±1° pointing accuracy required (2° cone) with a very high tolerance 0.1° determination. This ambitious requirement drives the need for precise three axis control. Stabilization will come from three forces: magnetorquers (active), reaction wheels (active), drag panels (passive). Position location comes from both Two Line Elements (TLEs), GPS data, and a startracker. The orientation is determined from two sunsensors, gyroscopes (integrated over time), and the star tracker.

Thus, the rotational equations of motion are critical in resolving the coupled behavior of the CADRE spacecraft when disturbed from equilibrium. These rotation dyamics are defined as follows.

$$\dot{p}I_{xx} + qr\left(I_{zz} - I_{yy}\right) - (\dot{r} + pq)I_{xz} = L_A + L_T$$
$$\dot{q}I_{yy} - pr(I_{zz} - I_{xx}) + (p^2 - r^2)I_{xz} = M_A + M_T$$
$$r\dot{I}_{zz} - pr\left(I_{yy} - I_{xx}\right) + (qr - \dot{p})I_{xz} = N_A + N_T$$

Disturbance from this equilibrium must be first measured, and then countered using on board controllers. In general, the roll rotation rates can be linearized about the equilibrium condition and solved for small disturbances. The first step in solving Euler's equation, however, is to quantify the disturbance torques acting on CADRE (L, M, N).

This report identifies the external forces acting on CADRE. The sum of the torques will be used as a worst case scenario requirement, which will drive actuator selection. I have performed a comprehensive trade study that selects the recommended components for this mission.

By following the recommended SMAD analysis for Attitude, Determination, and Control, I have identified the 5 primary external forces acting on CADRE during the mission. The details are summarized in Table 1.

Parameter	Definition	Parameters
Atmospheric Drag Force	$N \cdot D = \|N\|\|D\|\cos(\theta)$ $F_d = 0.5\rho v^2 C_d A(N \cdot D)$	N Normal vector of body face D Drag vector ρ Atmospheric density C_d Coefficient of drag A Cross sectional area v velocity relative to atmosphere
Atmospheric Torque about cg	$\tau = P\ x\ F_d$	P lever arm, distance between center of aerodynamic pressure, and cg
Gravity Gradient	$T_g = \frac{3\mu}{2R^3}\|I_z - I_y\|\sin(2\theta)$	$\mu = 3.986e14\ m^3/s^2$ $I_y, I_z\ Mass\ moments\ of\ inertia$ $\theta = \max deviation\ from\ vertical$
Solar Radiation Pressure	$F_s = \frac{\phi}{c^2}A(1+Q)(N \cdot S)$	ϕ, universal solar constant c, speed of light A, surface area Q, panel reflectance S, sun vector (gives angle of incidence)
Magnetic Disturbance Torque	$\tau_b = DB$ $B = \frac{\mu_0 M}{4\pi r^3} sqrt(1+\sin(\lambda)^2)$	$\frac{\mu_0 M}{4\pi}$magnetic dipole moment R, distance to center of Earth λ,magnetic lattitude

Table 1: Disturbance torque definitions

The next step was to model these forces over the course of an entire orbit. To do this, I began by referencing an Excel spread sheet compiled by Professor Ridley. It defines normal vectors of all 6 body faces and all 8 deployable panel faces. It also calculates the sun vector relative to the Body Fixed Frame at all points in the orbit. The orbit was discretized into 60 second steps and has so far been primarily used in thermal analyses (calculating maximum temperature in full sun and eclipse). It has now been applied to attitude determination and control. Appendix C summarizes many of the constants used in the Matlab analysis.

From the results of my analysis, I have calculated the net reaction torque on CADRE at every time of a single orbit, assuming small disturbance. Figure 3 shows the instantaneous torque over the course of a single orbit.

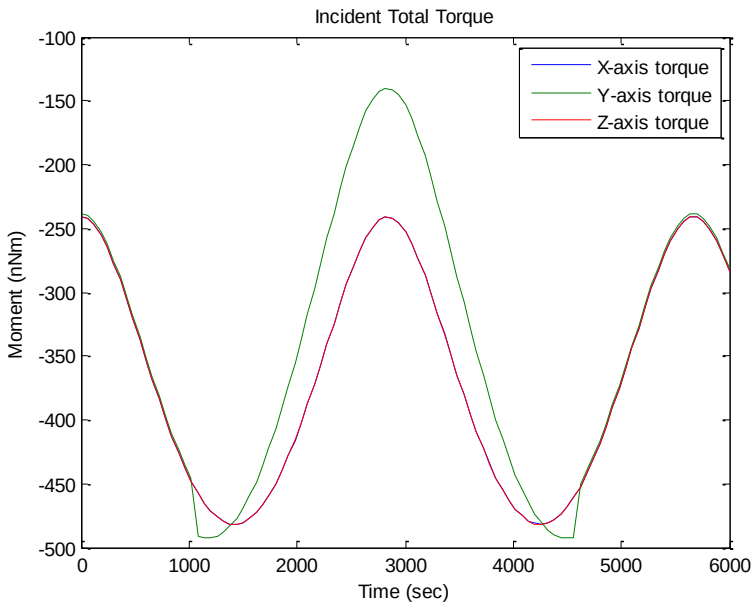


Figure 3: Net total torque acting on CADRE during one orbit

Two criteria that drive momentum wheel selection are maximum torque and maximum momentum storage. In order to maintain the equilibrium orientation, the three momentum wheels in three perpendicular directions speed up or slow down in order to counteract the external torques. Conservation of angular momentum allows us to do this—spinning a wheel faster (or slower) in one direction, spins CADRE in the opposite direction. This change in momentum is the counteracting torque. By orienting the wheels at a 45 degree angle inside of the body, we only need to power a single wheel at once for pitch stability (this helps our power budget).

The second principle concept is that the wheels have a limited maximum rotation speed. If the wheels are continuously changing momentum to counter balance the external torques, then the spin rate will increase until it has reached the maximum rotation speed. This known as saturation and the spacecraft will begin to deviate from equilibrium.

Desaturation is done by actuating the magnetorquers and at the same time letting the wheels de-spin. Typically, this is done only once or twice per orbit (when the magnetic field is a maximum), so the wheels must be able to store up to the maximum integrated torque (momentum) according to Figure 4.

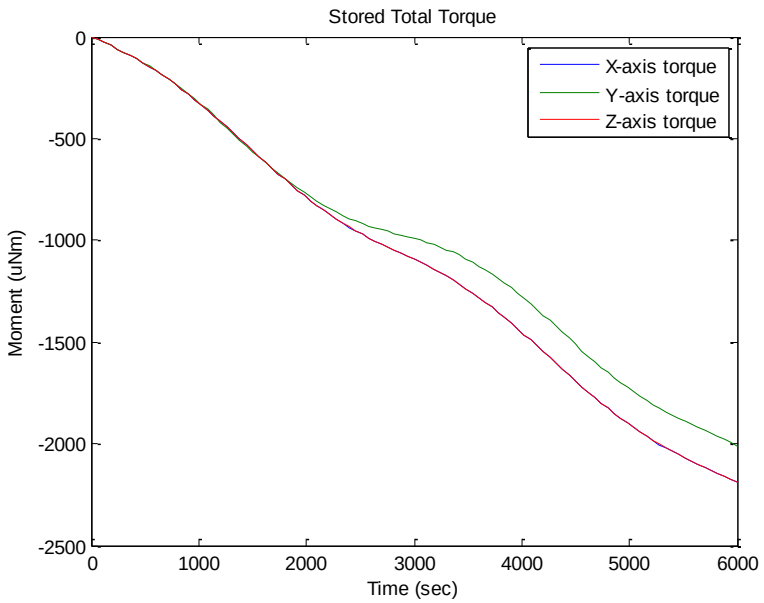


Figure 4: Net stored torque (momentum) required by reaction wheels

This model gages the necessary actuators (momentum wheels), which in turn drive our magnetorquers. To ensure that we meet the system control requirement (with 0.1 degree determination), I have compiled a comprehensive trade study of available ADCS packages that I could find for purchase online. It includes the relevant specifications (mass, power, strength, accuracy, heritage) where available and will allow the ADCS to select the best components for our mission.

First, the driver for momentum wheels is momentum storage—all of those options identified exceed the 500nN-m expected torque. Thus, I recommend the Maryland Aerospace Institute 300 series ADCS system, which can store 7.6 mN-m-s per wheel, above that predicted by Figure 4. Second, I will recommend the ISIS magnetorquer board due to its heritage on past Pumpkin missions.

The determination requirement is even finer than control, because accurate pointing *knowledge* can compensate (to a degree) for poor pointing direction. We can effectively back out accurate data if we have previously characterized the system behavior in ground testing. Thus, I have traded a number of available sun sensors, IMUs and star trackers to

First, I recommend the Sinclair Interplanetary S3S star tracker because Boeing is providing this (>$100,000) instrument to us for no cost. It allows us to measure the orientation of CADRE to within a 0.01 degree envelope of accuracy (with a known error margin). Second, I recommend two ISIS miniaturized Analog Fine Sun Sensors as redundant corroboration with the star tracker. Moreover, the field of view is the widest that I could find, which will help coarsely gage our orientation before the star tracker takes over. Thirdly, I recommend the CubeSat shop magnetometer due to its widespread use and availability. Finally, the gyroscopes (housed within our IMU box) that should be selected are the Analog Devices ADIS16405. They were used on RAX, so in addition to past heritage, Michigan students have experience integrating them.

APPENDIX A : Orbit Simulation

```
%Duncan Miller
%CADRE Nominal Orbit

clear all
close all
clc

alt=500 %km
mu=398600;
Re      = 6378;      % Earth Radius (km)
v0=sqrt(mu/(Re+alt));
R0=[Re+alt, 0,0];
V0=[0,0,v0];
X0 = [R0; V0];

t0     = 0;
tstep  = 20;        % sec
tfinal = 30*pi/sqrt(mu/(Re+alt)^3); % approximately 1 orbit
tspan    = t0:tstep:tfinal;
options = odeset('AbsTol',1e-6,'RelTol',1e-8);
[Tp, Xp] = ode45(@newton,tspan,X0, options);

figure(1);hold off;
plot3(Xp(:,1),Xp(:,2),Xp(:,3),'g-','linewidth',3)
xlabel('x (km)','fontsize',16);
ylabel('y (km)','fontsize',16);
zlabel('z (km)','fontsize',16)
set(gca,'fontsize',16)


% plot the Earth
[XS, YS, ZS] = sphere(30);
figure(1);
hold on;
surf(XS*Re, YS*Re, ZS*Re);
axis([-1e4, 1e4, -1e4, 1e4, -1e4, 1e4])
set(gca,'fontsize',16)
```

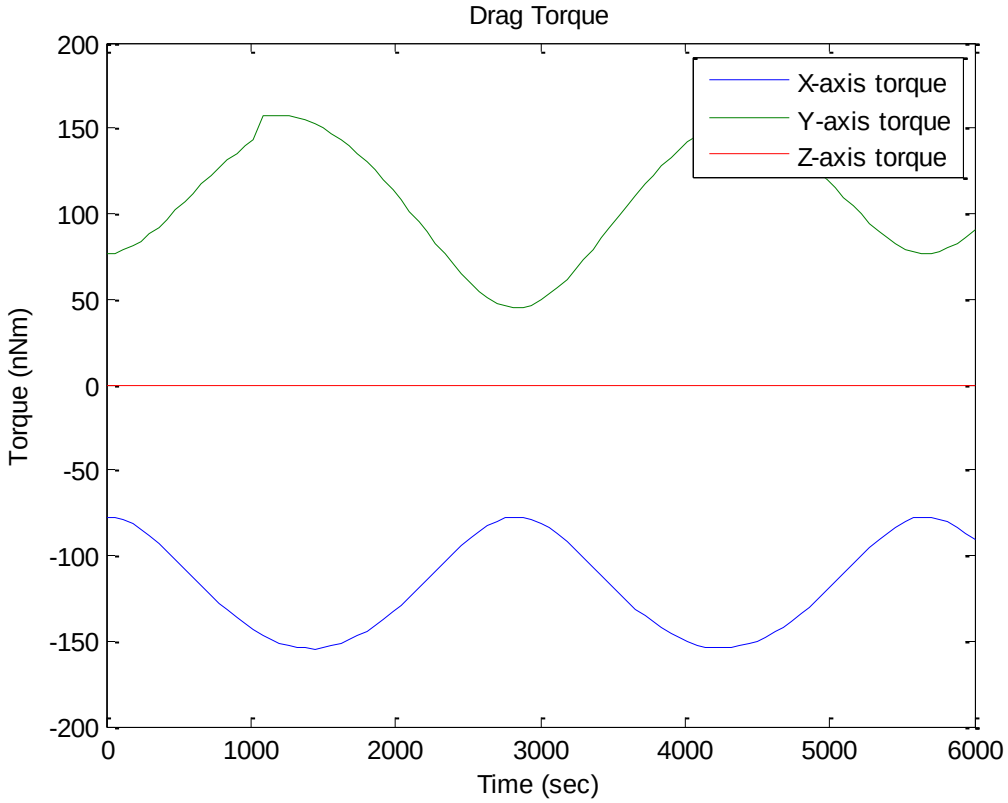
```
% Right hand side of Newton's 2BP equations (zero perturbation force)

function Xdot = newton(t,X)
 mu = 398600.4405;     % gravitational constant for Earth (km^3/s^2)
 r  = X(1:3);          % position (km)
 v  = X(4:6);          % velocity (km/sec)
Xdot(1:3)  = v;
rn         = norm(r);
Xdot(4:6)  = -mu/rn^3*r;

Xdot       = Xdot(:); % convert to vector colum

return;
```

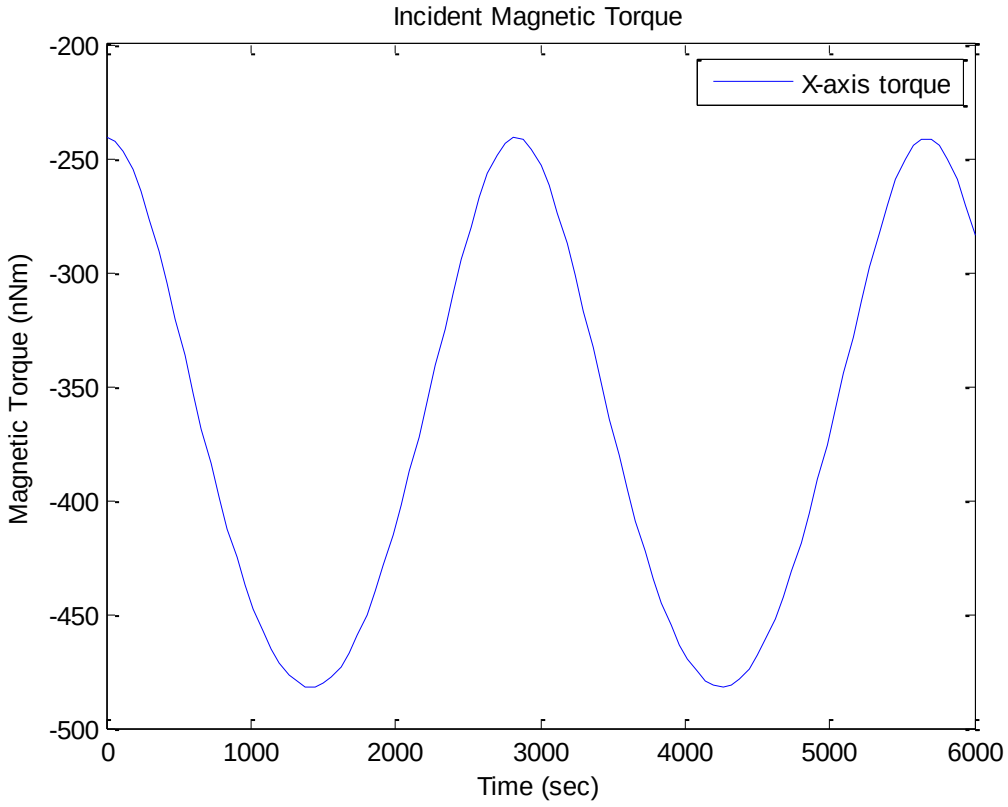
APPENDIX B: Sensors Trade Table

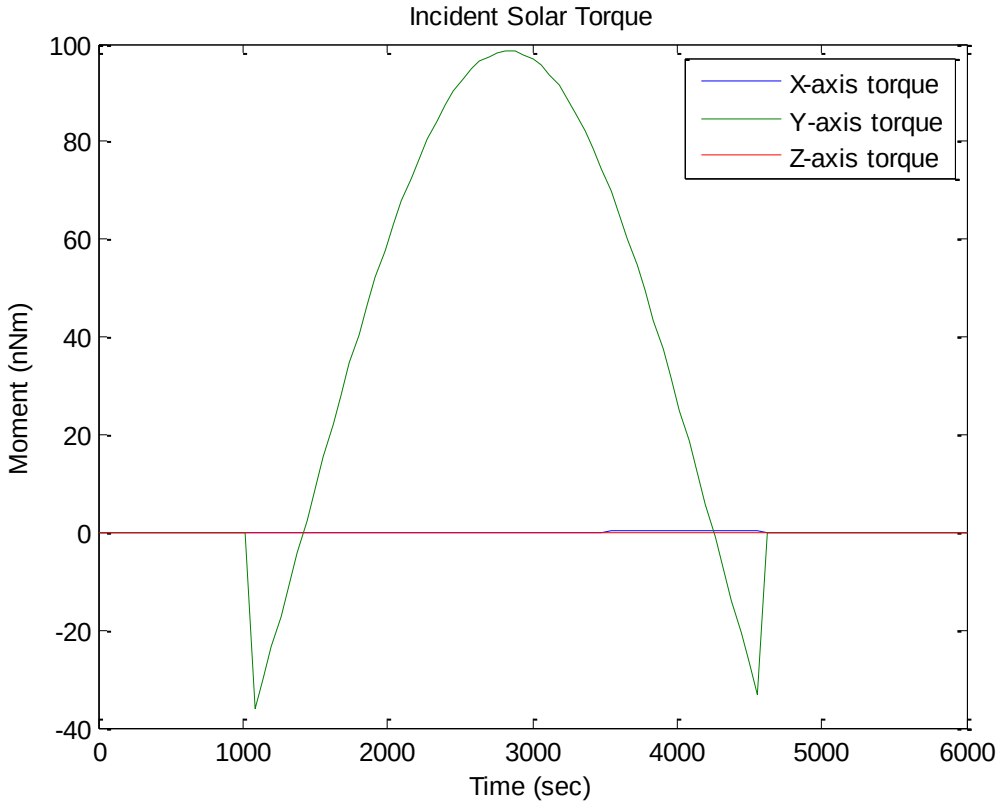
APPENDIX C

Constant	Value
Gravitational Constant (Earth)	$3.986x10^{14}$ m^3/s^2
Moments of Inertia, I_z I_y	0.047 kg-m^2, 0.012 kg-m^2
Drag Coefficient, C_d	2.0
Atm Pressure	10^{-14} Pa
Velocity	1.2 km/s
Panel Reflectance	0.6
Speed of Light	$3x10^8$ m/s
Universal Solar Constant, ϕ	1366 W/m^2
Residual Dipole, D	0.01 A-m^2
Magnetic Dipole moment (Earth)	$7.84x10^{15}$ Tesla-m^3
Radius of the Earth	6372000 m
Mass of Earth	$5.97x10^{24}$
Inclination	89.9 degrees
Altitude	500 km
Local Time	0
S/C Roll Angle	45 degrees

APPENDIX D: Particular Torque Disturbances






Incident Solar Torque
X-axis torque
Y-axis torque
Z-axis torque
Moment (nNm)
Time (sec)
100
80
60
40
20
0
-20
-40
0
1000
2000
3000
4000
5000
6000

APPENDIX E:

Disturbance Torque Code

```
%Aero 348 Run File

clear all
clc
close all
%% Inputs

d2r=pi()/180;
%Panel Angle
a=45*d2r;
%Panel Length
l=1;
%Spacecraft Orientation
r=45*d2r; %deg
%Center of Gravity shift
cg=-4; %cm
%Moments of Inertia
I_z=478836/1000/100^2; %From STR Subsystem
I_y=121134/1000/100^2;
%Orbit Altitude
alt=500; %km
%Residual Dipole
Md=[0;0;0.01]; %0.009 A-m^2 estimated by NRL Space Dart
                %This assumes that the dipole field of the space craft is
                %aligned along the Z axis meaning it will only produce
                %torques about the X and Y axis (Pitch and Yaw, no roll)
%Pointing Requirement
theta=1; %deg

%Spacecraft Mass
mass=4; %kg

%% Setup
[n_h,P,A,Q] = CADREsetup(a,l,r,cg);
P=P/100; %convert cm to m
x=linspace(0,0,101);
y=linspace(0,0,101);
i=1;
T=XLSREAD('Orbit_Power.xlsx','Sheet1','A43:A143'); %Discretized time (60
secs/step)
lat=XLSREAD('Orbit_Power.xlsx','Sheet1','D43:D143');
lat=abs(lat);
S=XLSREAD('Orbit_Power.xlsx','Sheet1','S43:U143')'*-1; %Sun vector at all
times
N=XLSREAD('Orbit_Power.xlsx','Sheet1','V43:V143');    %In Eclipse or not?
M_tot=zeros(3,101);

%% Calculate Moments

%Allocations
M_s=zeros(3,101);
M_b=zeros(3,101);
M=zeros(3,101);
```

```
M_tot=zeros(3,101);

h_t=zeros(3,101);

%Gravity Grad

M_g=[0;0;0]; %Allocation
M_g(2) = GravForce(alt,I_z,I_y,theta);


for i=1:101
%Magnetic
    [M_b(:,i)] = BForce(alt,lat(i),Md); %Calculate Magnetic Disturbance

%Solar
    if N(i) == 1 %Check for Eclipse
        M_s(:,i)=[0;0;0];
        M_s(:,i)=[0;0;0];
    else
    [M_s(:,i), M_ds] = sunForce(S(:,i),n_h,P,A,Q); %Calculate Moment
    end

%Drag
    M_tot(:,i)=[M_b(1,i)+M_s(1,i), M_g(2)+M_b(1,i)+M_s(2,i),
M_b(1,i)+M_s(3,i)]';

    D=dragVector(M_tot(:,i),cg,mass);
    [M(:,i)] = dragForce(D,n_h,P,A); %Calculate Drag force

    M_tot(:,i)=M_tot(:,i)+M(:,i);
end
%% Calculate Momentum storage

for i=1:100
h_t(:,i+1)=h_t(:,i)+(T(i+1)-T(i))*M_tot(:,i);
end


%% Plots
figure(1)
plot(T,M_s*10^9)
xlabel('Time (sec)')
ylabel('Moment (nNm)')
title('Incident Solar Torque')
legend('X-axis torque','Y-axis torque','Z-axis torque')
print figure(1) -djpeg

figure(2)
plot(T,M*10^9)
xlabel('Time (sec)')
ylabel('Torque (nNm)')
title('Drag Torque')
legend('X-axis torque','Y-axis torque','Z-axis torque')
print figure(2) -djpeg
```

```
figure(3)
plot(T,M_b(1,:)*10^9)
xlabel('Time (sec)')
ylabel('Magnetic Torque (nNm)')
title('Incident Magnetic Torque')
legend('X-axis torque','Y-axis torque','Z-axis torque')
print figure(3) -djpeg

figure(4)
plot(T,M_tot*10^9)
xlabel('Time (sec)')
ylabel('Moment (nNm)')
title('Incident Total Torque')
legend('X-axis torque','Y-axis torque','Z-axis torque')
print figure(4) -djpeg

figure(5)
plot(T,h_t*10^6)
xlabel('Time (sec)')
ylabel('Moment (uNm)')
title('Stored Total Torque')
legend('X-axis torque','Y-axis torque','Z-axis torque')
print figure(5) -djpeg
```

```
function [M_b, B] = BForce(alt,lat,Dm)
%Assuming cp_s is close to the panel cm
%Magnetic Force Calculation

B=7.84e15/(((alt+6378)*10^3)^3)*sqrt(1+3*sin(lat*pi()/180)^2);

B=[-B;B;0];

M_b=cross(Dm,B);
```

```
function [M_g] = GravForce(alt,I_z,I_y,theta)
%Assuming cp_s is close to the panel cm
%Magnetic Force Calculation

M_g=3*3.986e14/(2*((6378+alt)*10^3)^3)*abs(I_z-I_y)*sin(2*theta*pi()/180);
```

```
function [D] = dragVector(M_tot,cg,mass)

F=[M_tot(2)/(0.15+cg/100),M_tot(1)/(0.15+cg/100)];
a=F/mass;
s=1/2*a*60^2;
D=[s(1),s(2),0.15];
D=D/norm(D);
```

```
function [M, M_d, t] = dragForce(D,n_h,P,A)
%Assuming cp close to cm

%Force Calculation
a=500; %km
r=a+6378; %km
mu=396800;
rho=1.4103065e-13;
v=sqrt(mu/r); %m/s
c_d=2.0;
M_d=zeros(3,14);
i=1;
for i=1:14;
t(i)=dot(n_h(:,i)',D);
if t(i) < 0
    F_d(i)=1/2*rho*v^2*c_d*A(i)*t(i);
% elseif t(i) == -1
%     F_d(i)=1/2*rho*v^2*c_d*A(i);
elseif t(i) == 0
     F_d(i)=0;
else
    F_d(i)=0;
end
M_d(:,i)=cross(P(:,i)',F_d(i)*n_h(:,i)')';
end

M(1)=sum(M_d(1,:));
M(2)=sum(M_d(2,:));
M(3)=sum(M_d(3,:));
M=M';
```

```
function [M, M_d] = sunForce(S,n_h,P,A,Q)
%Assuming cp_s is close to the panel cm
%Force Calculation
phi=1366; %W/m^2
c=3*10^8; %m/s

M_d=zeros(3,14);
i=1;
for i=1:14;
t(i)=dot(n_h(:,i)',S);
%t(i)=acos(t(i)/(norm(S)))*180/pi();
if t(i) < 0
    F_s(i)=phi/c*A(i)*(1+Q(i))*t(i)/norm(S);
    M_d(:,i)=cross(P(:,i)',F_s(i)*n_h(:,i)')';
% elseif t(i) == -1
%     F_s(i)=phi/c*A(i)*t(i)*(1+Q(i));
elseif t(i) == 0
     F_s(i)=0;
     M_d(:,i)=[0,0,0]';
else
    F_s(i)=0;
```

```
    M_d(:,i)=[0,0,0]';
end

end

M(1)=sum(M_d(1,:));
M(2)=sum(M_d(2,:));
M(3)=sum(M_d(3,:));
M=M';
```

```
%THIS WAS TAKEN FROM PROF. RIDLEY'S CODE
%CADRE
%All values are in S/C coordinates
%units are in cm unless otherwise noted

function [n_h,P,A,Q] = CADREsetup(a,l,r,cg)

n=0.30; %m
m=0.10; %m
 %panel length (1=full panel 0.5=half etc)


a=a*pi()/180;


%\               +x
% \               /\
%  \_____________||__________
%  |             (.)+y        |--->+z
%  |_____________________________|m
%  /              n

% ____________________________________
%/             Earth                \

%Face Unit Normal vectors
n_h = zeros(3,14);
n_h(:,1) = [ 1   0   0]'; %+x
n_h(:,2) = [-1   0   0]'; %-x
n_h(:,3) = [ 0   1   0]'; %+y
n_h(:,4) = [ 0 -1   0]'; %-y
n_h(:,5) = [ 0   0   1]'; %+z
n_h(:,6) = [ 0   0 -1]'; %-z
n_h(:,7) = [cos(a)   0 sin(a)]'; %+x front
n_h(:,8) = [-cos(a)   0 -sin(a)]'; %+x back
n_h(:,9) = [-cos(a)   0 sin(a)]'; %-x front
n_h(:,10) = [cos(a)   0 -sin(a)]'; %-x back
n_h(:,11) = [ 0   cos(a) sin(a)]'; %+y front
n_h(:,12) = [ 0   -cos(a) -sin(a)]'; %+y back
n_h(:,13) = [ 0   -cos(a) sin(a)]'; %-y front
n_h(:,14) = [ 0   cos(a) -sin(a)]'; %-y back
```

```
P = zeros(3,14);
P(:,1) = [ 5  0  -cg]'; %+x
P(:,2) = [-5  0  -cg]'; %-x
P(:,3) = [ 0  5  -cg]'; %+y
P(:,4) = [ 0 -5  -cg]'; %-y
P(:,5) = [ 0  0  15-cg]'; %+z
P(:,6) = [ 0  0 -15-cg]'; %-z
P(:,7) = [5+15*sin(a)  0 -15-cg-15*cos(a)]'; %+x front
P(:,8) = [5+15*sin(a)  0 -15-cg-15*cos(a)]'; %+x back
P(:,9) = [-5-15*sin(a)  0 -15-cg-15*cos(a)]'; %-x front
P(:,10) = [-5-15*sin(a)  0 -15-cg-15*cos(a)]'; %-x back
P(:,11) = [0 5+15*sin(a) -15-cg-15*cos(a)]'; %+y front
P(:,12) = [0 5+15*sin(a) -15-cg-15*cos(a)]'; %+y back
P(:,13) = [0 -5-15*sin(a) -15-cg-15*cos(a)]'; %-y front
P(:,14) = [0 -5-15*sin(a) -15-cg-15*cos(a)]'; %-y back

T=[cos(r),sin(r),0;-sin(r),cos(r),0;0,0,1]^-1;

for i=1:14
    n_h(:,i)=T*n_h(:,i);
    P(:,i)=T*P(:,i);
end


A = zeros(1,14);
A(:,1) = n*m; %+x
A(:,2) = n*m; %-x
A(:,3) = n*m; %+y
A(:,4) = n*m; %-y
A(:,5) = m*m; %+z
A(:,6) = m*m; %-z
A(:,7) = n*m*l; %+x front
A(:,8) = n*m*l; %+x back
A(:,9) = n*m*l; %-x front
A(:,10) = n*m*l; %-x back
A(:,11) = n*m*l; %+y front
A(:,12) = n*m*l; %+y back
A(:,13) = n*m*l; %-y front
A(:,14) = n*m*l; %-y back

sp=0.5;
mli=0.5;
Q = zeros(1,14);
Q(:,1) = sp; %+x
Q(:,2) = mli; %-x
Q(:,3) = sp; %+y
Q(:,4) = mli; %-y
Q(:,5) = mli; %+z
Q(:,6) = mli; %-z
Q(:,7) = sp; %+x front
Q(:,8) = mli; %+x back
Q(:,9) = sp; %-x front
Q(:,10) = mli; %-x back
```

```
Q(:,11) = sp; %+y front
Q(:,12) = mli; %+y back
Q(:,13) = sp; %-y front
Q(:,14) = mli; %-y back
```