Problem 1. Part 1 Attitude Disturbance Analysis

Do calculations for the disturbances present for equatorial circular orbits at 200km and 350km. Consider a cube with two different halves. One half has twice the density (and therefore twice the mass) as the other. We will only be concerned with rotations about the axis out of the page.

I define $\hat{x}$ along the velocity/sun vector, and $\hat{y}$ in the opposite direction of the Earth. $\hat{z}$ completes the right hand rule. See the attached code for derivations of the following values.

Part (a)

1. Drag Find the disturbance torque on the cube about the mass center (use CD = 2.2)

$$\underline{T} = \sum \underline{r_i} \times \underline{F_i}$$
$$\underline{F_i} = \frac{1}{2}\rho V^2 C_D \left(\hat{n}_i \hat{V}\right) A_i \left(-\hat{V}\right)$$

$T = 5.45e - 06\hat{z}$ Nm at altitude 200km. $T = 5.18e - 07\hat{z}$ Nm at altitude 350km

Part (b)

2. Solar Pressure Find the disturbance torque on the cube about the mass center. Assume that the entire surface is covered in solar panels (fs≈0.21, fd≈0.1)

$$\underline{T} = \sum \underline{r_i} \times \underline{F_i}$$
$$\underline{F_i} = a_i \hat{s} + b_i \hat{n}_i$$
$$a_i = -PA_i \left(1 - f_{\mathrm{s,i}}\right) \cos\theta_i$$
$$b_i = -2PA_i \left(f_{\mathrm{s,i}} \cos\theta_i + \frac{1}{3} f_{\mathrm{d,i}}\right) \cos\theta_i$$

$T = 5.34e - 10\hat{z}$ Nm at altitude 200km. $T = 5.34e - 10\hat{z}$ Nm at altitude 350km.

Part (c)

3. Gravity Gradient Find the disturbance torque on the cube about the mass center

$$\underline{T} = \sum \underline{r_i} \times \underline{F_i}$$
$$\underline{F_i} = \frac{\mu m_i}{r_i^2} \left(-\hat{r}\right)$$

$T = -0.031\hat{z}$ Nm at altitude 200km. $T = -0.031\hat{z}$ Nm at altitude 350km.

Part (d)

4. Magnetic Dipole Find the disturbance torque on the cube about the mass center for two configurations

$$\underline{T} = \sum \underline{M_i} \times \underline{B}$$
$$\underline{M_i} = nIA\hat{c}$$

Magnetic Dipole Torque $T = 5.36e - 6\hat{x} - 9.29e - 4\hat{y}$ Nm at altitude 200km. $T = 5.36e - 6\hat{x} - 9.29e - 4\hat{y}$ Nm at altitude 350km.

Problem 2. Part 2 Reaction Wheel and Thruster Analysis

Assume a maximum constant on orbit disturbance torque of 1x10^{-5} Nm on a single axis. Size a reaction wheel to keep a craft pointed despite this disturbance. Assume a saturation speed of 6000rpm for the wheel. It should be capable of eliminating the maximum on orbit disturbance without desaturating for two weeks.

Given,

$$H_{storage} = \int \frac{dH}{dt} = \int T_{disturbance,max} dt = T_{disturbance,max} t$$
$$\int \frac{dH}{dt} = \int I_{wheel} \alpha dt = I_{wheel} \omega_{saturated}$$

$$\begin{aligned} I_{wheel}\omega_{saturated} &= \int I_{wheel}\alpha dt = \int \frac{dH}{dt} \\ &= \int T_{thrusters} dt = T_{thrusters} t \end{aligned}$$

as such,

$$T_{disturbance,max} t = I_{wheel}\omega_{saturated}$$
$$\text{or}$$
$$I_{wheel} = \frac{T_{disturbance,max} t}{\omega_{saturated}}$$

selecting T =1x10^{-5} Nm, $t = 2 * 7 * 24 * 60 * 60 = 1209600$ s, and $\omega = 6000$ rpm, gives:

$$\begin{aligned} I_{wheel} &= \frac{T_{disturbance,max} t}{\omega_{saturated}} \\ &= \frac{1E - 5 \text{ Nm} * 1209600 \text{ s}}{6000 \text{ rpm} \cdot \dfrac{0.1047\text{rad/s}}{1 \text{ rpm}}} \\ &= 0.01925 \text{ kg m}^2 \end{aligned}$$

To size the reaction wheel, we have:

$$
\begin{aligned}
I_{wheel} &= \frac{mr^2}{2} \\
&= \frac{\rho \pi r^4 h}{2}
\end{aligned}
$$

where the reaction wheel is modelled as a simple disc-shape. Choosing $h = r$ and $\rho = 8000$ kg m^{-3} (316 stainless steel) leads to

$$
\begin{aligned}
r &= (\frac{2I_{wheel}}{\pi \rho})^{1/5} \\
&= (\frac{2 \cdot 0.01925}{\pi 8000})^{1/5} \\
&= 0.069 \text{ m}
\end{aligned}
$$

As such, our reaction wheels are made of 316 stainless steel, and have a radius and a height of 0.069 meters. To calculate the burn time to desaturate the thrusters, we have

$$
I_{wheel}\omega_{saturated} = T_{thrusters} t
$$

or

$$
t = \frac{I_{wheel}\omega_{saturated}}{T_{thrusters}}
$$

selecting $T = 1$ Nm, $\omega = 6000$ rpm, and $I = 0.01925$ kg m^2, gives:

$$
\begin{aligned}
t &= \frac{I_{wheel}\omega_{saturated}}{T_{thrusters}} \\
&= \frac{0.01925 \text{ kg m}^2 \cdot 6000 \text{ rpm} \cdot \dfrac{0.1047 \text{ rad/s}}{1 \text{ rpm}}}{1 \text{ Nm}} \\
&= 12.25 \text{ s}
\end{aligned}
$$

Problem 3. Part 3 Hubble Slew Problem

Either use published inertia values for Hubble or the estimates you made in the dynamics assignment. Assume torquers are aligned with the principle axis.

$$
\begin{aligned}
I_{xx}\dot{\Omega}_x + H_{spin}\Omega_y &= T_x \\
I_{yy}\dot{\Omega}_y + H_{spin}\Omega_x &= T_y \\
H_{spin} &= I_{wheel}\omega_{wheel}
\end{aligned}
$$

Part (a)

1. Two axis torques Find the required variable torques (x and y) to constantly accelerate the angular precession by 0.1 deg/s^2 on the x axis while trying to keep the angular procession on the y-axis at a constant 0.1 deg/s.

The HST inertia matrix [1], was at one point measured as:

$$
I = \begin{bmatrix} 36046 & -706 & 1491 \\ -706 & 86868 & 449 \\ 1491 & 449 & 93848 \end{bmatrix} kg \cdot m^2.
$$

Plugging in the given values of $I_{wheel} = 1$ kg m^2, $\omega = 1500$ rpm, $I_{xx} = 36046$ kg m^2 $I_{yy} = 86868$ kg m^2, $\dot{\Omega}_x = 0.01$ deg s^{-2}, $\dot{\Omega}_y = 0$ deg s^{-2}, $\Omega_y = 0.1$ deg s^{-1} gives

$$
\begin{aligned}
H_{spin} &= 1 \cdot 1500 \cdot 0.1047 = 157.05 \\
T_x &= 36046 \cdot 0.0001745 + 157.05 \cdot 0.0017453 = 6.56 \\
T_y &= 86868 \cdot 0 + 157.05 \cdot \Omega_x \\
&= 157.05 \cdot \Omega_x
\end{aligned}
$$

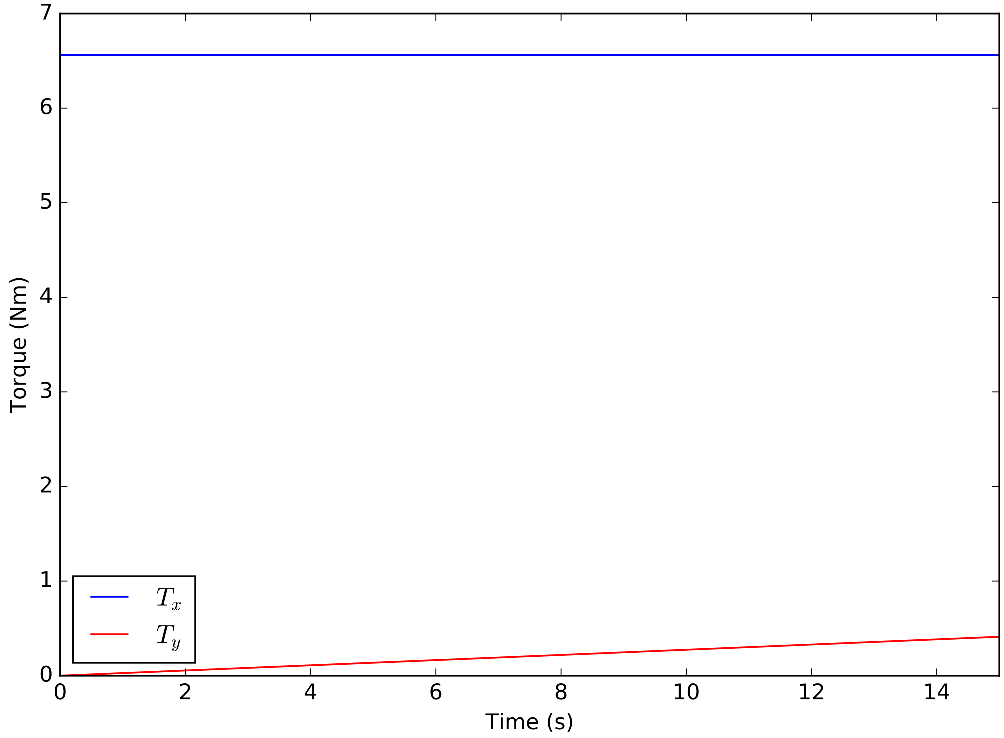
Part (b)

2. Two axis torques Find the required variable torque (x) to constantly accelerate the angular precession by 0.1 deg/s^2 on the x axis while applying no torque to the y axis.

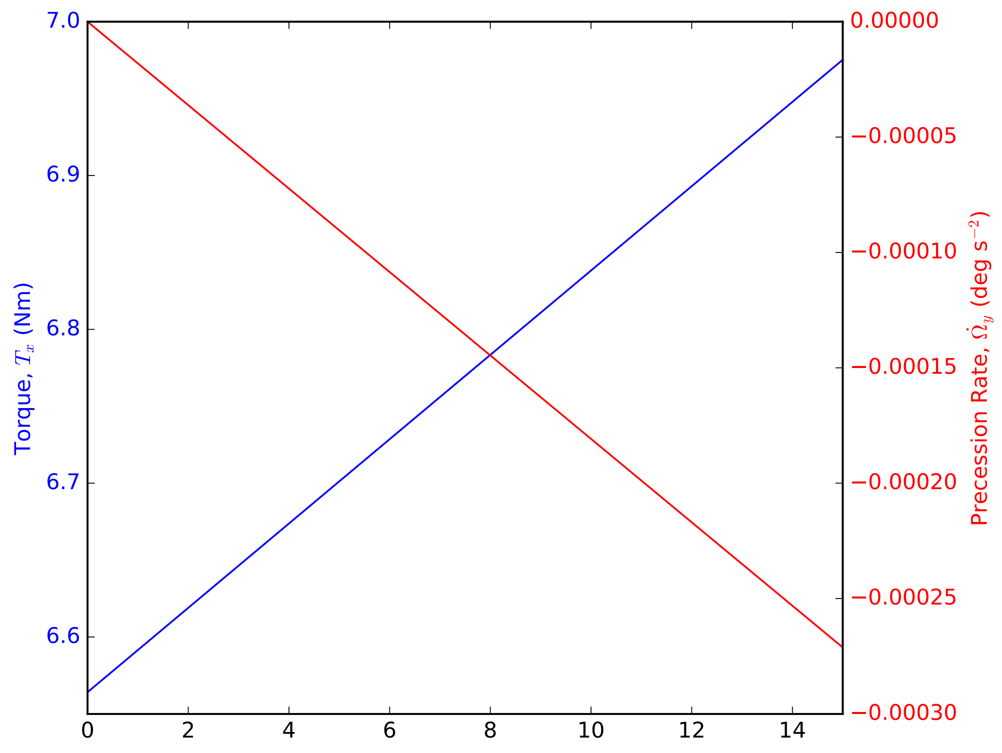
Plugging in the given values of $I_{wheel} = 1$ kg m^2, $\omega = 1500$ rpm, $I_{xx} = 36046$ kg m^2 $I_{yy} = 86868$ kg m^2, $\dot{\Omega}_x = 0.01$ deg s^{-2}, $\Omega_y = 0.1$ deg s^{-1} gives

$$
\begin{aligned}
H_{spin} &= 1 \cdot 1500 \cdot 0.1047 = 157.05 \\
T_x &= 36046 \cdot 0.0001745 + 157.05 \cdot \Omega_y \\
T_y &= 86868 \cdot \dot{\Omega}_y + 157.05 \cdot \Omega_x \\
\dot{\Omega}_y &= -\frac{157.05 \cdot \Omega_x}{86868}
\end{aligned}
$$

[1] Queen, S., "HRV GNC Peer Review, Flight Performance Analysis," Tech. rep., NASA Goddard Space Flight Center, 2004.



Torque (Nm)
Time (s)
T_x
T_y
0
1
2
3
4
5
6
7
0
2
4
6
8
10
12
14



Torque, T_x (Nm)
Precession Rate, $\dot{\Omega}_y$ (deg s^{-2})
7.0
6.9
6.8
6.7
6.6
0.00000
−0.00005
−0.00010
−0.00015
−0.00020
−0.00025
−0.00030
0
2
4
6
8
10
12
14

.1 Python Code

```
import numpy as np
import matplotlib.pyplot as plt

################################################################################
# Problem 1
# Part 1
a = 10 * 1/100  # m
mu = 1/1000 * (100)**3  # kg/m^3
A = a**2  # m^2
CD = 2.2
theta = np.deg2rad(30)  # rad

n1 = np.array([np.cos(theta), np.sin(theta), 0])  # red face
n2 = np.array([np.sin(theta), np.cos(theta), 0])  # top face
V = np.array([1, 0, 0])

low = {'v': 7789,
       'rho': 8.28E-10,
       'alt': '200km'}
high = {'v': 7702,
        'rho': 8.05E-11,
        'alt': '350km'}

m1 = mu * (a * a * (a/2))
m2 = (mu/2) * (a * a * (a/2))
r1 = np.array([0, a/4, 0])
r2 = np.array([0, -a/4, 0])
Rz = np.array([[np.cos(theta), -np.sin(theta), 0],
               [np.sin(theta),  np.cos(theta), 0],
               [            0,              0, 1]])
r1_rot = np.dot(r1, Rz)
r2_rot = np.dot(r2, Rz)

x = (m1 * r1_rot[0] + m2 * r2_rot[0])/(m1 + m2)
y = (m1 * r1_rot[1] + m2 * r2_rot[1])/(m1 + m2)
r = np.array([x, y, 0])

print('\nDrag Torque')
for alt in [low, high]:
    T = np.array([0., 0., 0.])
    for n in [n1, n2]:
        F = 0.5 * alt['rho'] * alt['v']**2 * CD * np.dot(n, V) * A * (-V)
        T += np.cross(r, F)
    print(T, 'at altitude {}'.format(alt['alt']))

################################################################################
# Part 2
P = 4.67E-6
s = np.array([1, 0, 0])
a_i = -P * A * (1 - .21) * np.cos(theta)
b_i = -2 * P * A * (.21 * np.cos(theta) + (1/3) * 0.1) * np.cos(theta)
```

```python

print('\nSolar Pressure Torque')
for alt in [low, high]:
    T = np.array([0., 0., 0.])
    for n in [n1, n2]:
        F = a_i * s + b_i * n
        T += np.cross(r, F)
    print(T, 'at altitude {}'.format(alt['alt']))

################################################################################
# Part 3
mu = 0.3986E15
r_earth = np.array([0., 1., 0.])

print('\nGravity Gradient Torque')
for alt in [low, high]:
    T = np.array([0., 0., 0.])
    for m, r_rot in zip([m1, m2], [r1_rot, r2_rot]):
        rr = r_rot + np.array([0, -6371E3, 0])
        F = ((mu * m) / np.linalg.norm(rr)**2) * (-r_earth)
        T += np.cross(r, F)
    print(T, 'at altitude {}'.format(alt['alt']))

################################################################################
# Part 4
n = 25
I = 0.1
A = 25 * np.pi * (1/100)**2
c = np.array([np.cos(theta), np.sin(theta), 0])
B = np.array([0., 0., 1.])

print('\nMagnetic Dipole Torque')
for alt in [low, high]:
    T = np.array([0., 0., 0.])
    for n in [n1, n2]:
        M = n * I * A * c
        T += np.cross(M, B)
    print(T, 'at altitude {}'.format(alt['alt']))

################################################################################
# Problem 3
# Part a
t = np.linspace(0, 15)
f, ax = plt.subplots()

ax.plot(t, 6.56 * np.ones_like(t), 'b-', label='$T_x$')
ax.plot(t, 157.05 * np.deg2rad(t * 1/100), 'r-', label='$T_y$')
plt.xlim(0, 15)
plt.legend(loc='best')
plt.xlabel('Time (s)')
plt.ylabel('Torque (Nm)')
plt.tight_layout()
plt.savefig('p3_a.pdf')
plt.close()
```

```python

# Part b
t = np.linspace(0, 15)
f, ax = plt.subplots()

tx = 36046 * 0.0001745 + 157.05 * np.deg2rad(t * 1/100 + .1)
omega_dot_y = - np.rad2deg(np.deg2rad(157.05 * t * 1/100)/86868)
ax.plot(t, tx, 'b-', label='$T_x$')
ax.set_ylabel('Torque, $T_x$ (Nm)', color='b')
for tl in ax.get_yticklabels():
    tl.set_color('b')

r_ax = ax.twinx()
r_ax.plot(t, omega_dot_y, 'r-', label='$\dot{\Omega}_y$')
r_ax.set_ylabel('Precession Rate, $\dot{\Omega}_y$ (deg s$^{-2}$)', color='r')
for tl in r_ax.get_yticklabels():
    tl.set_color('r')

plt.xlim(0, 15)
plt.xlabel('Time (s)')
plt.tight_layout()
plt.savefig('p3_b.pdf')
plt.close()
```