## Problem 1. Develop a very simple representation of the Hubble telescope.

```
########################################################################################
#                                         Hubble                                       #
#     ^V2                                                                              #
#     |                          +----------------------+                              #
#  <--+                          |                      |                              #
#  V1                            |         SA1          |                              #
#                                |                      |                              #
#                                +-----------+----------+                              #
#                                            |                                         #
#                                            |    +-------+                            #
#                             +--------------++---+       |                            #
#                             |               |   |       |                            #
#                             |       1       | 2 |   3   |                            #
#                             |               |   |       |                            #
#                             |               |   |       |                            #
#                             +--------------++---+       |                            #
#                                            |    +-------+                            #
#                                            |                                         #
#                                +-----------+----------+                              #
#                                |                      |                              #
#                                |         SA1          |                              #
#                                |                      |                              #
#                                +----------------------+                              #
#                                                                                      #
########################################################################################
#                                                                                      #
#     ^V3                                                                              #
#     |                                                                                #
#  <--+                                           +-------+                            #
#  V1                         +---------------+---+       |                            #
#                             |               |   |       |                            #
#                             |       1       | 2 |   3   |                            #
#                             |  +-----------+----------+ |                            #
#                             |               |   |       |                            #
#                             +---------------+---+       |                            #
#                                                 +-------+                            #
#                                                                                      #
#                                                                                      #
#                                                                                      #
########################################################################################
```

Diagram 1: Hubble ASCII Diagram. Solar panel distance from HST is exaggerated. One character is ≈ 20 inches.

We model both the body (3 sections) and the solar panels (2 sections) of the Hubble Space Telescope (HST). The body sections are connected as such: Section 1 is connected to Section 2, and Section 2 is connected to Section 3. The solar arrays are connected on Section 1, along the centerline, 20.75 inches $V_1$ away from the connection point with Section 2, and the near edge of the SA is 129 inches from center of Section 1. These sections are modeled with thin walled cylinders (TWC), solid cylinders (SC), and flat plates (FP). The rough

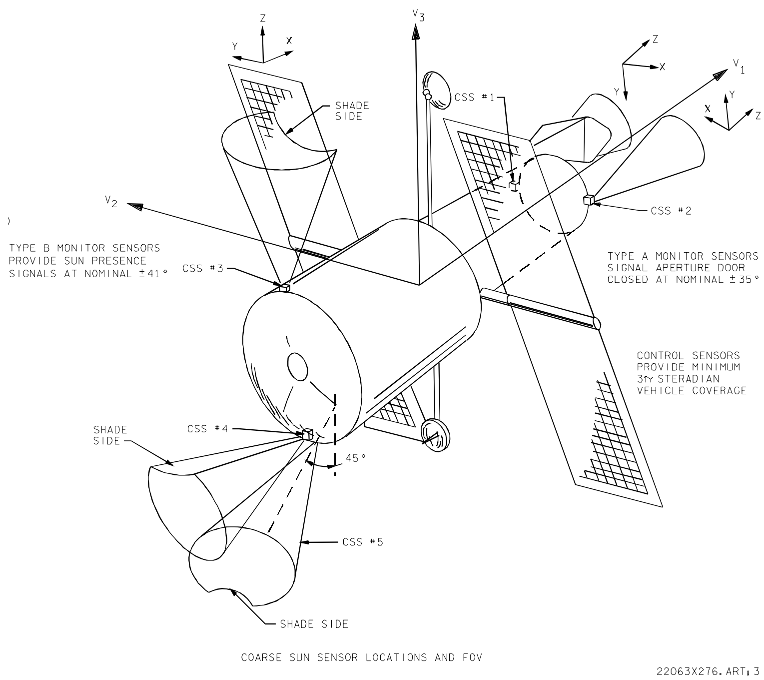


Figure 1: HST Axes Definition for $V_1, V_2, V_3$, CG is located at axis origin

| Section | Model | $V_1$ (in) | $V_2$ (in) | Weight (lb) |
|---|---|---|---|---|
| *Section 1* | | | | |
| Light Shield (LS) | - | 153.2 | 120 | - |
| Forward Shell (FS) (except OTA) | - | 156.05 | 121.2 | - |
| Total | TWC | 270.75 | 121.2 | 2796 |
| *Section 2* | | | | |
| OTA Equipment Section | TWC | 38.5 | 121.2 | 9033 |
| *Section 3* | | | | |
| SSM Equipment Section (SSM-ES) | - | 61.25 | 168.16 | 10594 |
| Aft Shroud (AS) | - | 138.00 | 168.16 | 569 |
| Total | SC | 199.25 | 168.16 | 11163 |
| *Section 4* | | | | |
| Solar Arrays (SA) | FP | 476.8[1] | 113.5 | 735[2] |

Table 1: [1]: This length can be fully rotated into $V_3$. [2]: Weight of both solar arrays. $V_1$ and $V_2$ indicate the measurements of the parts. All lengths taken from Hubble technical drawings [2]; all masses from [3].

layout of the sections is shown on the previous page, and the mass and length properties of each section are listed in Table 1.

## Problem 2. Use this model as a basis to write a function(s) to determine the Mass Center and Inertia Matrix for any location.

Using the radial-center of the farthest tip of Section 1 as our zero point , the center of mass is located at $V = [327, 0, 0]$ inches. This makes since, as the model is symmetric about the $V_2$ and $V_3$ axes, and this value is on the boundary between Section 2 and Section 3, which is very close to where the reaction wheels are located.

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

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

The Python script in Appendix gives the result of:

$$I = \begin{bmatrix} 35914 & 0 & 0 \\ 0 & 88215 & 0 \\ 0 & 0 & 113393 \end{bmatrix} kg \cdot m^2,$$

which has a relative error of:

$$I = \begin{bmatrix} 0 & 100 & 100 \\ 100 & -2 & 100 \\ 100 & 100 & -21 \end{bmatrix} \%.$$

Note that while our simple, 5 part model does a very good job of predicting the $I_{V_1}$ and $I_{V_2}$ components, the $I_{V_3}$ component is not represented very well. This is likely due to leaving out the antenna booms, which should have the largest effect in the $V_3$ directions. It should also be noted that, due to the symmetric nature of our model, all of the off-axis terms are missing. For the rest of the analysis, the true values for the inertia matrix will be used. See Figure 2 for the effects of rotating the solar arrays on the principle axes.

## Problem 3. Write a function to find the current angular momentum relative to the mass center.

$H$ increases linearly with $\omega$. See Appendix for the code used to create Figure 3.

## Problem 4. Choose the optimal location for a torque producing system and explain why you think is the best location.

The optimal location for a torque producing system is usually on the centerline, as near to the center of mass of the spacecraft as possible. Placing the torque producing system close the the spacecraft's center of mass minimizes the amount of undesirable resultant torque when the system is activated. In HST's case, the optimal location for a torque producing system is in the SSM-ES, see Fig. 4. The two wheels of a single SSM bay are canted off the

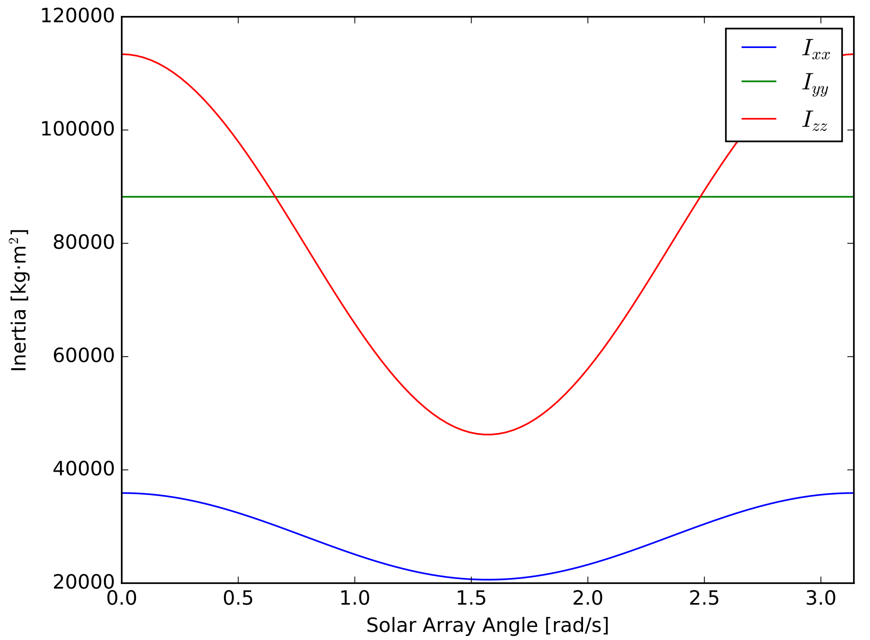


Figure 2: Inertias for principal axes for different solar array configurations

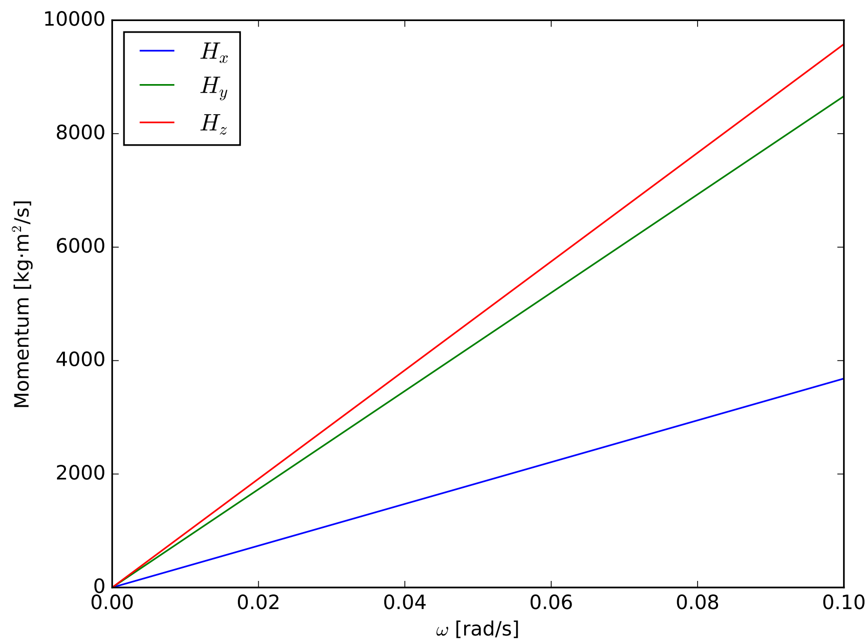


Figure 3: Effects of spin on each axis

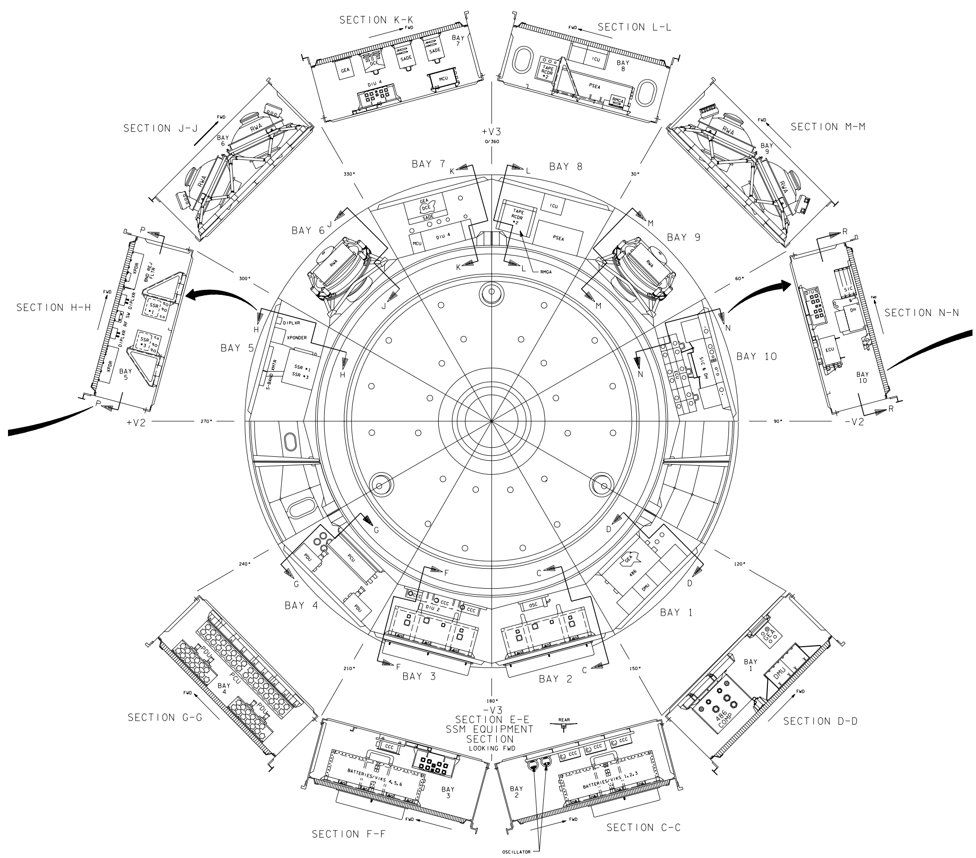

Figure 4: HST's reaction wheels are located in the SSM-ES in bays 6 and 9. The pairs were installed in at a $90^o$ offset to protect against failures [2].

$V_2$-$V_3$ plane by $20^o$, one toward the $+V_1$ and one toward the -$V_1$. I think that this is likely the best location to place the system as it's the one that was actually used, and because the engineers that placed them the reaction wheels there had much greater access to information about HST than I do.

As we do not have to worry about reaction wheels failing for the purposes of this homework, I would instead simplify the system to a single location of three reaction wheels, one of which is aligned with each axis. "Nominal dynamic torque range is 0.003 to 0.605 ft-lb (0.004 to 0.7 N·m), with a maximum wheel speed of ±3000 rpm" for each of the four reaction wheels [2].

## Problem 5. Using your previous functions, write a program to find the resulting angular acceleration produced from a given torque.

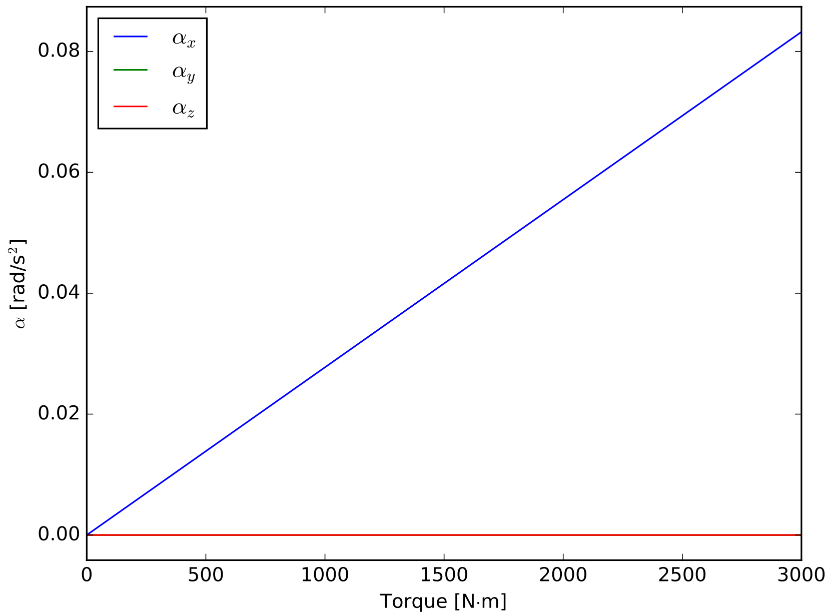


Figure 5: Angular acceleration response along all three axis from a single reaction wheel producing $\tau_x$, from initial conditions $\omega = [0, 0, 0]$ rad/s.

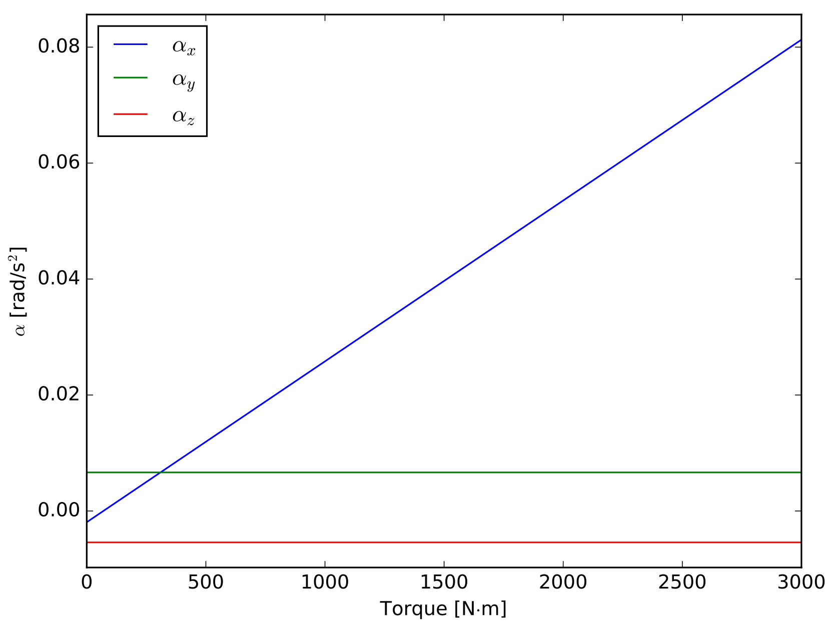


Figure 6: Angular acceleration response along all three axis from a single reaction wheel producing $\tau_x$, from initial conditions $\omega = [0.1, 0.1, 0.1]$ rad/s.

## Problem 6. Write what next steps you would take to develop a controller that keeps the craft pointed in a specific direction.

To develop a controller, we would need to:

(a) Determine where the craft should be pointed (science requirements)

(b) Determine where the craft is currently pointed (using sensors)

(c) Find the difference between these two (the error)

(d) Determine which reaction wheels need to be activated to minimize the error

(e) Activate the reaction wheels

Once we had sensors and the ability to activate our reaction wheels, the majority of the time developing the controller would involve tuning it to perform to some specification (minimizing overshoot, minimizing response time, etc.).

# Bibliography

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

[2] NASA, "Cargo Systems Manual (CSM): Hubble Space Telescope," February 13, 2002

[3] Mattice, J., "Hubble Space Telescope Systems Engineering Case Study."

# .1 Python Code

```
import numpy as np
from collections import namedtuple
import pandas as pd
import matplotlib.pyplot as plt


# Origin is the tip of far center of the far end of Section1
Part = namedtuple('Part', ['mass', 'v1', 'v2', 'v3'])
Cylinder = namedtuple('Cylinder', Part._fields + ('length', 'radius'))
Plate = namedtuple('Plate', Part._fields + ('length', 'width'))

Section1 = Cylinder(mass=2796, v1=270.75/2, v2=0, v3=0, length=270.75, radius=121.2/2)
Section2 = Cylinder(mass=9033, v1=Section1.length + 38.5/2, v2=0, v3=0, length=38.5,
    radius=121.2/2)
Section3 = Cylinder(mass=11163, v1=Section1.length + Section2.length + 199.25/2, v2=0, v3
    =0, length=199.25, radius=168.16/2)
Solar1 = Plate(mass=735/2, v1=309.25 - 20.75, v2=129 + 113.5/2, v3=0, length=476.8, width
    =113.5)
Solar2 = Plate(mass=735/2, v1=309.25 - 20.75, v2=-129 - 113.5/2, v3=0, length=476.8,
    width=113.5)
```

```
parts = [Section1, Section2, Section3, Solar1, Solar2]
r_cm = ([part.mass * np.array([part.v1, part.v2, part.v3]) for part in parts] /
        np.sum([part.mass for part in parts])).sum(axis=0)
print(r_cm)


def parallel_axis(part, inertia, r=[0, 0, 0]):
    # The moment of inertia about the center of mass of the body with respect
    # to an orthogonal coordinate system.
    Ic = inertia(part)
    m = part.mass

    # The distances along the three ordinates that located the new point
    # relative to the center of mass of the body.
    d = np.array([part.v1, part.v2, part.v3])

    a = d[0] - r[0]
    b = d[1] - r[1]
    c = d[2] - r[2]
    dMat = np.zeros((3, 3), dtype=object)
    dMat[0] = np.array([b**2 + c**2, -a * b, -a * c])
    dMat[1] = np.array([-a * b, c**2 + a**2, -b * c])
    dMat[2] = np.array([-a * c, -b * c, a**2 + b**2])
    return Ic + m * dMat


def SolidCylinder(cylinder):
    m = cylinder.mass
    r = cylinder.radius
    h = cylinder.length
    I = np.array([[1/12 * m * (3 * r**2 + h**2), 0, 0],
                  [0, 1/12 * m * (3 * r**2 + h**2), 0],
                  [0, 0, 1/2 * m * r**2]])
    return I


def ThinWalledCylinder(cylinder):
    m = cylinder.mass
    r = cylinder.radius
    h = cylinder.length
    I = np.array([[1/12 * m * (3 * 2*(r**2) + h**2), 0, 0],
                  [0, 1/12 * m * (3 * 2*(r**2) + h**2), 0],
                  [0, 0, 1/2 * m * 2 * (r**2)]])
    return I


def FlatPlate(plate):
    m = plate.mass
    a = plate.length
    b = plate.width
    I = np.array([[1/12 * m * b**2, 0, 0],
                  [0, 1/12 * m * a**2, 0],
                  [0, 0, 1/2 * m * (a**2 + b**2)]])
```

```
    return I


def rotation_matrix(axis, theta):
    """
    Return the rotation matrix associated with counterclockwise rotation about
    the given axis by theta radians.
    """
    axis = np.asarray(axis)
    theta = np.asarray(theta)
    axis = axis/np.sqrt(np.dot(axis, axis))
    a = np.cos(theta)
    b, c, d = -axis*np.sin(theta)
    aa, bb, cc, dd = a*a, b*b, c*c, d*d
    bc, ad, ac, ab, bd, cd = b*c, a*d, a*c, a*b, b*d, c*d
    return np.array([[aa+bb-cc-dd, 2*(bc+ad), 2*(bd-ac)],
                     [2*(bc-ad), aa+cc-bb-dd, 2*(cd+ab)],
                     [2*(bd+ac), 2*(cd-ab), aa+dd-bb-cc]])


def TotalI(r, theta=0):
    I_S1 = parallel_axis(Section1, ThinWalledCylinder, r=r)
    I_S2 = parallel_axis(Section2, ThinWalledCylinder, r=r)
    I_S3 = parallel_axis(Section3, SolidCylinder, r=r)
    I_SA1 = parallel_axis(Solar1, FlatPlate, r=r)
    I_SA2 = parallel_axis(Solar2, FlatPlate, r=r)

    # Rotate if necessary
    I_SA1 = np.dot(rotation_matrix(np.array([0, 1, 0]), theta), I_SA1)
    I_SA2 = np.dot(rotation_matrix(np.array([0, 1, 0]), theta), I_SA2)
    # Convert from lb * inches^2 to kg * m^2
    I = (I_S1 + I_S2 + I_S3 + I_SA1 + I_SA2) * 0.000292639653
    return I

truth = np.array([[36046,  -706,  1491],
                  [ -706, 86868,   449],
                  [ 1491,   449, 93848]])
print(truth)
print(TotalI(r_cm))


def plot1():
    plt.close('all')
    res = []
    for theta in np.linspace(0, np.pi, 100):
        res.append([theta, *TotalI(r=r_cm, theta=theta).diagonal()])
    pd.DataFrame(res).plot(x=0, y=[1, 2, 3])
    plt.legend(['$I_{xx}$', '$I_{yy}$', '$I_{zz}$'], loc='upper right')
    plt.ylabel('Inertia [kgm$^2$]')
    plt.xlabel('Solar Array Angle [rad/s]')
    plt.savefig('figure1.pdf')
plot1()
```

```python
def angularMomentum(w):
    ''' Using the true value for the inertia matrix.'''
    I = truth
    H = np.dot(I, w)
    return H


def plot2():
    plt.close('all')
    res = []
    for w in np.linspace(0, 0.1, 100):
        w *= np.ones(3)
        res.append([w[0], *angularMomentum(w)])
    pd.DataFrame(res).plot(x=0, y=[1, 2, 3])
    plt.legend(['$H_x$', '$H_y$', '$H_z$'], loc='upper left')
    plt.ylabel('Momentum [kgm$^2$/s]')
    plt.xlabel('$\omega$ [rad/s]')
    plt.savefig('figure2.pdf')
plot2()


def estimateAlpha(torque, omega=0, I=truth):
    ''' Using the true value for the inertia matrix.'''
    alpha = np.zeros(3)

    alpha[0] = (torque[0] + (I[1][1] - I[2][2]) * omega**2) / I[0][0]
    alpha[1] = (torque[1] + (I[2][2] - I[0][0]) * omega**2) / I[1][1]
    alpha[2] = (torque[2] + (I[0][0] - I[1][1]) * omega**2) / I[2][2]
    return alpha


def plot3():
    plt.close('all')
    res = []
    for torque in np.linspace(0, 3000, 100):
        torque *= np.array([1, 0, 0])
        res.append([torque[0], *estimateAlpha(torque, omega=0)])
    pd.DataFrame(res).plot(x=0, y=[1, 2, 3])
    #plt.ylim(0, .05)
    #plt.xlim(0, 5e-7)
    plt.legend(['$\\alpha_x$', '$\\alpha_y$', '$\\alpha_z$'], loc='upper left')
    plt.ylabel('$\\alpha$ [rad/s$^2$]')
    plt.xlabel('Torque [N$\cdot$m]')
    plt.margins(0.05)
    plt.savefig('figure3.pdf')
plot3()


def plot4():
    plt.close('all')
    res = []
    for torque in np.linspace(0, 3000, 100):
        torque *= np.array([1, 0, 0])
        res.append([torque[0], *estimateAlpha(torque, omega=0.1)])
```

```
    pd.DataFrame(res).plot(x=0, y=[1, 2, 3])
    #plt.ylim(0, .05)
    #plt.xlim(0, 5e-7)
    plt.legend(['$\\alpha_x$', '$\\alpha_y$', '$\\alpha_z$'], loc='upper left')
    plt.ylabel('$\\alpha$ [rad/s$^2$]')
    plt.xlabel('Torque [N$\cdot$m]')
    plt.margins(0.05)
    plt.savefig('figure4.pdf')
plot4()
```