AERO 424 Homework 6

```
import sympy as sy

import IPython
from IPython.display import display

import matplotlib.pyplot as plt

import numpy as np
import pandas as pd

from scipy.integrate import odeint

def displayH(a1,a2='', a3='', a4='', a5='', a6='', a7='',):
    latex_a1 = sy.latex(a1)
    latex_a2 = sy.latex(a2)
    latex_a3 = sy.latex(a3)
    latex_a4 = sy.latex(a4)
    latex_a5 = sy.latex(a5)
    latex_a6 = sy.latex(a6)
    latex_a7 = sy.latex(a7)
    display( IPython.core.display.Math(latex_a1 + latex_a2 + latex_a3 + latex_a4 + latex_a
```

1. TRIAD Algorithm

A spacecraft equipped with a star tracker measures the following two vectors in its body frame:

$$\mathbf{b}_1 = \begin{bmatrix} 0.2500 \\ 0.5177 \\ 0.8311 \end{bmatrix}$$

$$\mathbf{b}_2 = \begin{bmatrix} 0.8479 \\ 0.5040 \\ 0.2018 \end{bmatrix}$$

An onboard star catalog provides the following unit vectors representing the positions of the tracked stars in the Earth-Centered Inertial (ECI) frame:

$$\mathbf{r}_1 = \begin{bmatrix} 0.5637 \\ 0.3054 \\ 0.7674 \end{bmatrix}$$

$$\mathbf{r}_2 = \begin{bmatrix} 0.2569 \\ 0.9337 \\ 0.2495 \end{bmatrix}$$

Using the TRIAD algorithm, determine the attitude matrix/DCM that transforms vectors from the ECI frame to the spacecraft body frame.

```
b1 = np.array([0.2500, 0.5177, 0.8311])
b2 = np.array([0.8479, 0.5040, 0.2018])

r1 = np.array([0.5637, 0.3054, 0.7674])
r2 = np.array([0.2569, 0.9337, 0.2495])

def Triad_Estimate(b1,b2,r1,r2):

    bx = np.cross(b1,b2)/np.linalg.norm(np.cross(b1,b2))
    rx = np.cross(r1,r2)/np.linalg.norm(np.cross(r1,r2))

    return np.outer(b1,r1)+np.outer(bx,rx)+np.outer(np.cross(b1,bx),np.cross(r1,rx))

T_ECI_to_B = Triad_Estimate(b1,b2,r1,r2)

displayH(sy.Symbol(r"T_{ECI\rightarrow B}="),sy.Matrix(T_ECI_to_B))
```

$$T_{ECI \to B} = \begin{bmatrix} 0.359545290467058 & 0.895380710736912 & -0.294686665096864 \\ -0.41225566685024 & 0.437678951238699 & 0.803212049507664 \\ 0.841308754616298 & -0.166700423583808 & 0.531282334188543 \end{bmatrix}$$

2. Quaternion Solution of Wahba's Problem - Davenport's q Method

A spacecraft is equipped with a solid-state star tracker capable of simultaneously tracking multiple stars. It records vector measurements to 30 stars in the spacecraft body frame. These measurements are provided in the Excel file "bvec_Meas.xls", which contains a matrix of size 3×30. Each column of the matrix corresponds to a vector $\mathbf{b}_i$, where $i = 1, 2, \ldots, 30$.

An onboard star catalog provides the unit vectors representing the positions of the tracked stars in the Earth-Centered Inertial (ECI) frame. These vectors are provided in the Excel file "rvec_Refs.xls", which also contains a 3×30 matrix. The columns of this matrix represent the individual vectors $\mathbf{v}_i$, corresponding to $\mathbf{b}_i$ for $i = 1, 2, \ldots, 30$.

Assuming that all measurements are weighted equally, with $a_i = 0.01$, $i = 1, 2, \ldots, 30$.

Use the Davenport's q method, to determine the following:

(a) The estimated optimal quaternion: $\hat{\mathbf{q}}$

(b) The corresponding attitude matrix/DCM that transforms vectors from the ECI frame to the spacecraft body-frame: $\mathbf{A}(\hat{\mathbf{q}})$

The following equations may be useful in the implementation of the Davenport's q method:

$$\mathbf{A}(\mathbf{q}) = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \quad \text{... Attitude matrix/DCM}$$

$$\mathbf{B} = \sum_{i=1}^{N} a_i \mathbf{b}_i \mathbf{r}_i^T \quad \text{... Attitude profile matrix}$$

$$\mathbf{z} = \begin{bmatrix} B_{23} - B_{32} \\ B_{31} - B_{13} \\ B_{12} - B_{21} \end{bmatrix}$$

$$\mathbf{K}(\mathbf{B}) = \begin{bmatrix} \mathbf{B} + \mathbf{B}^T - (\operatorname{tr}\mathbf{B})\mathbf{I}_3 & \mathbf{z} \\ \mathbf{z}^T & \operatorname{tr}\mathbf{B} \end{bmatrix}$$

```
def Quat_to_DCM(q):
    q_1 = q[0]
    q_2 = q[1]
    q_3 = q[2]
    q_4 = q[3]
```

```
    return np.array([[q_1**2 - q_2**2 - q_3**2 + q_4**2, 2*(q_1*q_2 + q_3*q_4), 2*(q_1*q_3
                     [2*(q_1*q_2 - q_3*q_4), -q_1**2 + q_2**2 - q_3**2 + q_4**2, 2*(q_2*q_
                     [2*(q_1*q_3 + q_2*q_4), 2*(q_2*q_3 - q_1*q_4), -q_1**2 - q_2**2 + q_3
```

```
bvec_Meas = pd.read_csv("bvec_Meas.csv",header=None).T.to_numpy()
rvec_Refs = pd.read_csv("rvec_Refs.csv",header=None).T.to_numpy()
weights = len(rvec_Refs)*[0.01]

# Attitude Profile Matrix
B = sum([a*np.outer(b,r.T) for a,b,r in zip(weights,bvec_Meas,rvec_Refs)])

z = np.array([B[1,2]-B[2,1],
              B[2,0]-B[0,2],
              B[0,1]-B[1,0]])

K = np.zeros([4,4])
K[:3,:3] = B+B.T-np.trace(B)*np.eye(3)
K[:3,3] = z
K[3,:3] = z
K[3,3] = np.trace(B)
e,v = np.linalg.eig(K)
v_max = v[:,np.argmax(e)]
q = v_max/np.linalg.norm(v_max)
A = Quat_to_DCM(q)

displayH("(a)")
displayH(sy.Symbol(r"\hat{q}="),sy.Matrix(q))

displayH("(b)")
displayH(sy.Symbol(r"A(\hat{q})="),sy.Matrix(A))
```

(a)

$$\hat{q} = \begin{bmatrix} -0.365829310955887 \\ 0.449588224137211 \\ -0.783515167387612 \\ -0.223927056061237 \end{bmatrix}$$

(b)

$$A(\hat{q}) = \begin{bmatrix} -0.632051177618577 & 0.0219553891248767 & 0.774615562559484 \\ -0.679845590124864 & -0.495454204561794 & -0.540680224134207 \\ 0.371915692676103 & -0.868356546627257 & 0.32807868792538 \end{bmatrix}$$