AERO 424 Homework 1

By Jonathan Jermstad (8/31/2024)

```
import sympy as sy
from sympy.physics.vector import ReferenceFrame,dynamicsymbols
from sympy.physics.vector import Point

import IPython
from IPython.display import display

import matplotlib.pyplot as plt

import numpy as np

import scipy
```

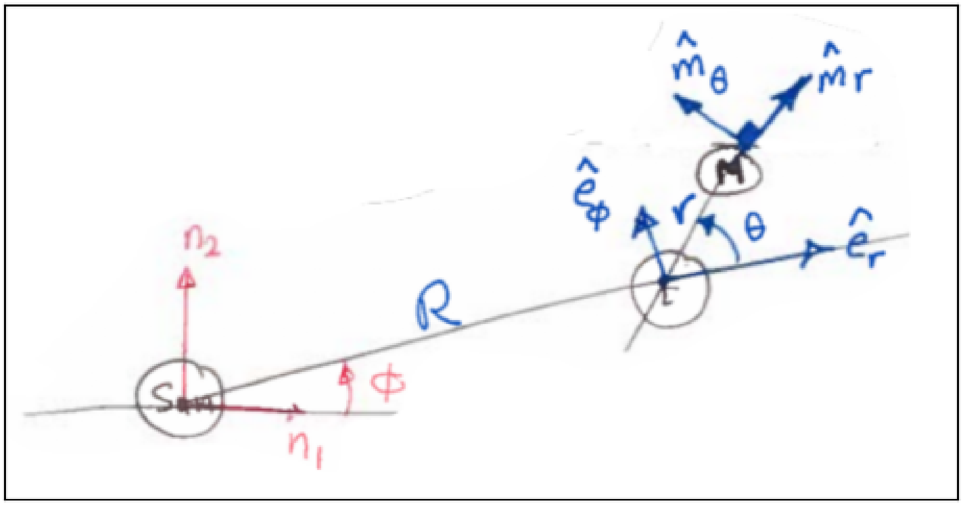
```
def displayH(a1,a2='', a3='', a4='', a5='', a6='', a7='',):
    latex_a1 = sy.latex(a1)
    latex_a2 = sy.latex(a2)
    latex_a3 = sy.latex(a3)
    latex_a4 = sy.latex(a4)
    latex_a5 = sy.latex(a5)
    latex_a6 = sy.latex(a6)
    latex_a7 = sy.latex(a7)
    display( IPython.core.display.Math(latex_a1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

1) Transport Theorem:

Consider a simplified planetary system shown in the figure. The Earth is in a circular orbit of radius R around the Sun, and is orbiting at a constant rate $\dot{\phi}$. The moon is orbiting Earth also in a circular orbit of radius r at a constant rate $\dot{\theta}$. Assume the Sun is inertially fixed with body-frame $N = \{\hat{n_1}, \hat{n_2}, \hat{n_3}\}$. Let Earth body-fixed frame be $E = \{\hat{e_r}, \hat{e_\theta}, \hat{e_3}\}$ Let moon body-fixed frame be $M = \{\hat{m_r}, \hat{m_\theta}, \hat{m_3}\}$. Find the inertial velocity and acceleration of the Moon.

```
img = plt.imread("Homework_1_1_Problem.PNG")
plt.imshow(img)
plt.tick_params(axis='x',which='both',bottom=False,top=False,labelbottom=False)
plt.tick_params(axis='y',which='both',left=False,right=False,labelleft=False)
plt.title("Problem 1")
plt.show()
```

Problem 1



```
phi = dynamicsymbols("phi")
theta = dynamicsymbols("theta")
R = sy.Symbol("R")
r = sy.Symbol("r")
t = sy.Symbol("t")

# Define inrtl reference frame (sun)
N = ReferenceFrame('N')
```

```
O = Point('O')
# Define intermediate reference frame (earth) rotated about N.z by phi
E = N.orientnew('E','Axis',[phi,N.z])
# Define final reference frame (moon) rotated about E.z by theta
M = E.orientnew('M','Axis',[theta,E.z])

displayH("N =",N.x+N.y+N.z)
displayH("E =",E.x+E.y+E.z)
displayH("M =",M.x+M.y+M.z)
```

N $=\hat{\mathbf{n}}_{\mathbf{x}} + \hat{\mathbf{n}}_{\mathbf{y}} + \hat{\mathbf{n}}_{\mathbf{z}}$

E $=\hat{\mathbf{e}}_{\mathbf{x}} + \hat{\mathbf{e}}_{\mathbf{y}} + \hat{\mathbf{e}}_{\mathbf{z}}$

M $=\hat{\mathbf{m}}_{\mathbf{x}} + \hat{\mathbf{m}}_{\mathbf{y}} + \hat{\mathbf{m}}_{\mathbf{z}}$

Solution:

1. Define position vector of the moon.

$$\vec{r}_{moon} = R\hat{e}_r + r\hat{m}_r$$

```
# Coordinates of final reference frame (moon) translated byR in E.x and r in M.x
r_moon = O.locatenew("pos_moon",R*E.x+r*M.x)

displayH(sy.Symbol(r"\vec{r}_{moon} ="),r_moon.pos_from(O))
```

$\vec{r}_{moon} = R\hat{\mathbf{e}}_{\mathbf{x}} + r\hat{\mathbf{m}}_{\mathbf{x}}$

2. Define position vector in body frame.

$$\hat{e}_r = cos(\theta)\hat{m}_r - sin(\theta)\hat{m}_\theta$$

$$\vec{r}_{moon} = \left(Rcos(\theta) + r\right)\hat{m}_r - Rsin(\theta)\hat{m}_{theta}$$

```
displayH(sy.Symbol(r"\vec{r}_{moon} ="),r_moon.pos_from(O).express(M))
```

$\vec{r}_{moon} = (R\cos(\theta(t)) + r)\hat{\mathbf{m}}_{\mathbf{x}} - R\sin(\theta(t))\hat{\mathbf{m}}_{\mathbf{y}}$

3. Define angular velocity vector ($\vec{\omega}$).

$$\vec{\omega} = \dot{\phi}\hat{n}_3 + \dot{\theta}\hat{e}_3 = (p\dot{h}i + \dot{\theta})\hat{m}_3$$

```
omega = M.ang_vel_in(N)
displayH(sy.Symbol(r"\vec{\omega} ="),omega,"=",omega.express(M))
```

$\vec{\omega} = \frac{d}{dt}\theta(t)\hat{\mathbf{e}}_{\mathbf{z}} + \frac{d}{dt}\phi(t)\hat{\mathbf{n}}_{\mathbf{z}} {=} (\frac{d}{dt}\phi(t) + \frac{d}{dt}\theta(t))\hat{\mathbf{m}}_{\mathbf{z}}$

4. Take the inertial derivative (N) of the position vector.

$$\frac{{}^{N}d\vec{r}_{moon}}{dt} = \frac{{}^{M}d\vec{r}_{moon}}{dt} + \vec{\omega} \times \vec{r}_{moon}$$

$$\frac{{}^{M}d\vec{r}_{moon}}{dt} = -R\dot{\theta}sin(\theta)\hat{m}_r - R\dot{\theta}cos(\theta)\hat{m}_\theta$$

$$\vec{\omega} \times \vec{r}_{moon} = (\dot{\theta} + \dot{\phi})\left[Rsin(\theta)\hat{m}_r + (Rcos(\theta) + r)\hat{m}_\theta\right]$$

$$\boxed{\begin{aligned} {}^{N}\vec{v}_{moon} = \frac{{}^{N}d\vec{r}_{moon}}{dt} &= \left[R\dot{\phi}sin(\theta)\right]\hat{m}_r \\ &+ \left[R\dot{\phi}cos(\theta) + (\dot{\phi} + \dot{\theta})r\right]\hat{m}_\theta \end{aligned}}$$

```
displayH(sy.Symbol(r"\frac{^{M}d \vec{r}_{moon}}{dt} ="),sy.diff(r_moon.pos_from(O),t,M))
displayH(sy.Symbol(r"\vec{\omega}\times\vec{r}_{moon} ="),omega.cross(r_moon.pos_from(O)).
v_moon = sy.diff(r_moon.pos_from(O),t,N)
displayH(sy.Symbol(r"\frac{^{N}d \vec{r}_{moon}}{dt} ="),v_moon.express(M).simplify())
```

$\frac{{}^{M}d\vec{r}_{moon}}{dt} = -R\sin(\theta(t))\frac{d}{dt}\theta(t)\hat{\mathbf{m}}_{\mathbf{x}} - R\cos(\theta(t))\frac{d}{dt}\theta(t)\hat{\mathbf{m}}_{\mathbf{y}}$

$\vec{\omega} \times \vec{r}_{moon} = R\left(\frac{d}{dt}\phi(t) + \frac{d}{dt}\theta(t)\right)\sin(\theta(t))\mathbf{\hat{m}_x} + (R\left(\frac{d}{dt}\phi(t) + \frac{d}{dt}\theta(t)\right)\cos(\theta(t)) + r\left(\frac{d}{dt}\phi(t) + \frac{d}{dt}\theta(t)\right))\mathbf{\hat{m}_y}$

$\frac{^{N}d\vec{r}_{moon}}{dt} = R\sin(\theta(t))\frac{d}{dt}\phi(t)\mathbf{\hat{m}_x} + (R\cos(\theta(t))\frac{d}{dt}\phi(t) + r\frac{d}{dt}\phi(t) + r\frac{d}{dt}\theta(t))\mathbf{\hat{m}_y}$

5. Take the inertial derivative (N) of the velocity vector.

$$\frac{^{N}d\vec{v}_{moon}}{dt} = \frac{^{M}d\vec{v}_{moon}}{dt} + \vec{\omega} \times \vec{v}_{moon}$$

$$\frac{^{M}d\vec{v}_{moon}}{dt} = \left[R(\ddot{\phi}sin(\theta) + \dot{\theta}\dot{\phi}cos(\theta))\right]\hat{m}_r + \left[R(\ddot{\phi}cos(\theta) - \dot{\theta}\dot{\phi}sin(\theta)) + (\ddot{\theta} + \ddot{\phi})r\right]\hat{m}_\theta$$

$$\vec{\omega} \times \vec{v}_{moon} = (\dot{\phi} + \dot{\theta})\left[-\left(R\dot{\phi}cos(\theta) + (\dot{\phi} + \dot{\theta})r\right)\hat{m}_r + R\dot{\phi}sin(\theta)\hat{m}_\theta\right]$$

$$\boxed{\begin{aligned} {}^{N}\vec{a}_{moon} = \frac{^{N}d\vec{v}_{moon}}{dt} &= \left[R(\ddot{\phi}sin(\theta) - \dot{\phi}^2cos(\theta)) - r(2\dot{\theta}\dot{\phi} + \dot{\theta}^2 + \dot{\phi}^2)\right]\hat{m}_r \\ &+ \left[R(\dot{\phi}^2sin(\theta) + \ddot{\phi}cos(\theta)) + r(\ddot{\theta} + \ddot{\phi})\right]\hat{m}_\theta \end{aligned}}$$

```
displayH(sy.Symbol(r"\frac{^{M}d \vec{v}_{moon}}{dt} ="),sy.diff(v_moon,t,M).simplify())
displayH(sy.Symbol(r"\vec{\omega}\times\vec{v}_{moon} ="),omega.cross(v_moon).express(M).s
a_moon = sy.diff(v_moon,t,N)
displayH(sy.Symbol(r"\frac{^{N}d \vec{v}_{moon}}{dt} ="),a_moon.express(M).simplify())
```

$\frac{^{M}d\vec{v}_{moon}}{dt} = R\left(\sin(\theta(t))\frac{d^2}{dt^2}\phi(t) + \cos(\theta(t))\frac{d}{dt}\phi(t)\frac{d}{dt}\theta(t)\right)\mathbf{\hat{m}_x} + (-R\sin(\theta(t))\frac{d}{dt}\phi(t)\frac{d}{dt}\theta(t) + R\cos(\theta(t))\frac{d^2}{dt^2}\phi(t) + r\frac{d^2}{dt^2}\phi(t) + r\frac{d^2}{dt^2}\theta(t))\mathbf{\hat{m}_y}$

$\vec{\omega}\times\vec{v}_{moon} = \left(-\frac{d}{dt}\phi(t) - \frac{d}{dt}\theta(t)\right)\left(R\cos(\theta(t))\frac{d}{dt}\phi(t) + r\frac{d}{dt}\phi(t) + r\frac{d}{dt}\theta(t)\right)\mathbf{\hat{m}_x} + R\left(\frac{d}{dt}\phi(t) + \frac{d}{dt}\theta(t)\right)\sin(\theta(t))$

$\frac{^{N}d\vec{v}_{moon}}{dt} = (R\sin(\theta(t))\frac{d^2}{dt^2}\phi(t) - R\cos(\theta(t))\left(\frac{d}{dt}\phi(t)\right)^2 - r\left(\frac{d}{dt}\phi(t)\right)^2 - 2r\frac{d}{dt}\phi(t)\frac{d}{dt}\theta(t) - r\left(\frac{d}{dt}\theta(t)\right)^2)\mathbf{\hat{m}_x} + (R\sin(\theta(t))\left(\frac{d}{dt}\phi(t)\right)^2 + R\cos(\theta(t))\frac{d^2}{dt^2}\phi(t) + r\frac{d^2}{dt^2}\phi(t) + r\frac{d^2}{dt^2}\theta(t))\mathbf{\hat{m}_y}$

2. ODEs and Numerical Integration:

Consider the following differential equation for the Simple Harmonic Motion (SHM):

$$m\ddot{x} + kx = 0$$

This second-order differential equation has a closed form analytic solution given by,

$$x(t) = Asin(\omega t + \phi)$$

Where frequency $\omega = \sqrt{\frac{k}{m}}$, and the constants A and are amplitude and phase angle, respectively, and are deter- mined using initial conditions.

Given: * m = 1 kg. * k = 10 N/m. * Initial conditions: x(t0) = 3 m and x(t0) = 0 m/s.

Perform the following tasks:

(a) Find the exact values of x(t) over the time interval t = (0, 0.1, 0.2, . . . 10) secs using the closed form analytic solution. Plot these values.

Solution:

1. Find the characteristic equation and roots of the differential equation.

$$m\ddot{x} + kx => ms^2 + k = 0$$
$$s = \sqrt{\frac{-k}{m}} = i\sqrt{\frac{k}{m}}$$
$$\omega = imag(s)$$

2. Write the general solution to the differential equation.

$$x(t) = Asin(\omega t) + Bcos(\omega t)$$
$$\dot{x}(t) = A\omega cos(\omega t) - B\omega sin(\omega t)$$

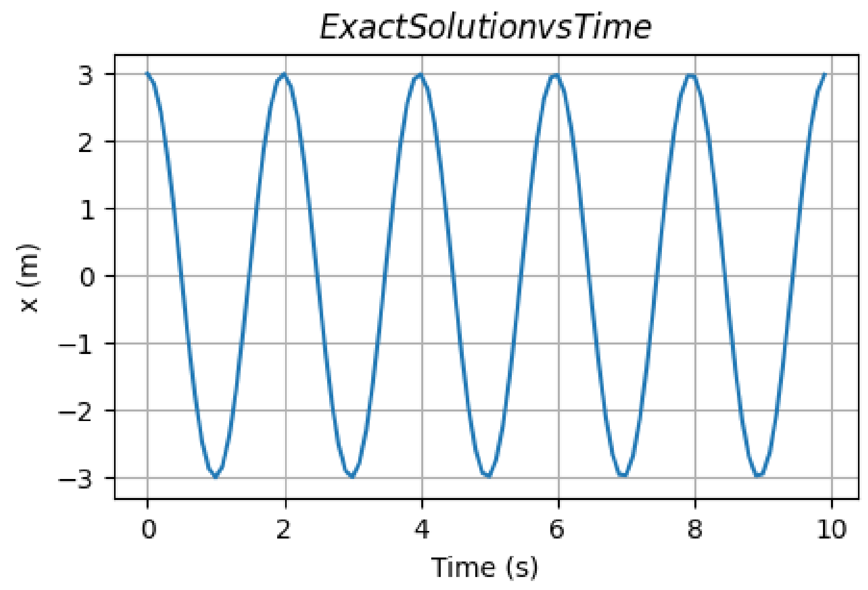
3. Plug in known quantities to solve for coefficients, knowing that sin(0) = 0 and cos(0) = 1.

$$x(0) = 3 = Asin(\sqrt{10}(0)) + Bcos(\sqrt{10}(0)) = B$$
$$\dot{x}(0) = 0 = A\sqrt{10}cos(\sqrt{10}t) - B\sqrt{10}sin(\sqrt{10}(0)) = A$$

4. This gives us the final solution to the differential equation.

$$\boxed{x(t) = 3cos\left(\sqrt{10}t\right) = 3sin\left(\sqrt{10}t + \frac{\pi}{2}\right)}$$

```
k = 10
m = 1
x0 = [3,0]

A = 0
B = 3
w = (k/m)**.5

def exact_x(t):
    return A*np.sin(w*t)+B*np.cos(w*t)

times = np.arange(0,10,0.1)
plt.figure(figsize=(5,3))
plt.plot(times,exact_x(times))
plt.xlabel("Time (s)")
plt.ylabel("x (m)")
plt.title(r"$Exact Solution vs Time$")
plt.grid()
plt.show()
```



(b) For the same time interval, numerically integrate the SHM differential equation using MATLAB's "ode45" function. Further, perform the integration with four different relative and absolute error tolerances: 1. tol1 = 103 2. tol2 = 106 3. tol3 = 109 4. tol4 = 1012 This will provide you with four numerically integrated approximate solutions for x(t).

```
import scipy.integrate


tolerances = [1e-3,1e-6,1e-9,1e-12]

def dx(x,t):
    return [x[1],-k*x[0]/m]

x_list = []
for tol in tolerances:
    x_list.append(scipy.integrate.odeint(dx,x0,times,rtol=tol,atol=tol)[:,0])
```

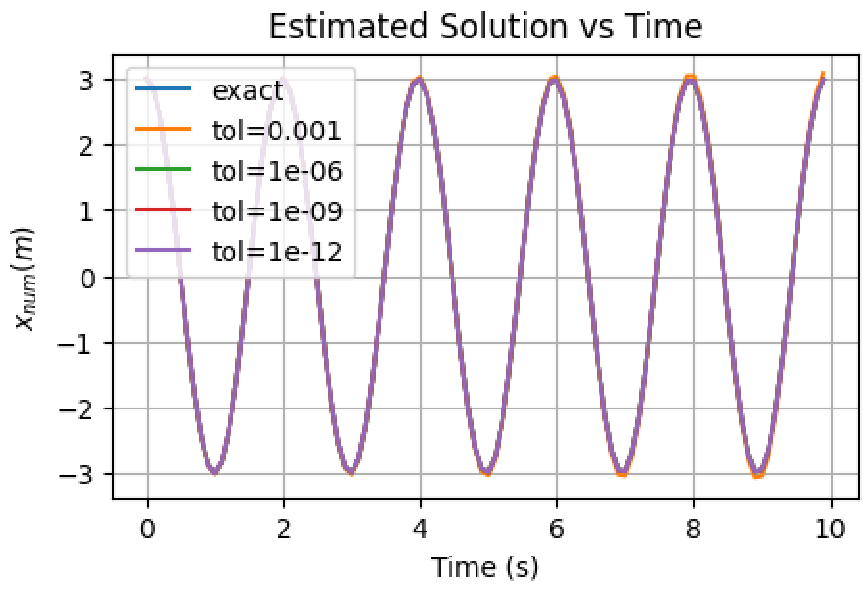
(c) For each of the four solutions, compute the absolute error between the exact solution and the numerically integrated solutions as follows: $err(t) = |x_{num} x_{exact}|$ **This will result in four sets of error values corresponding to the four numerical solutions.**

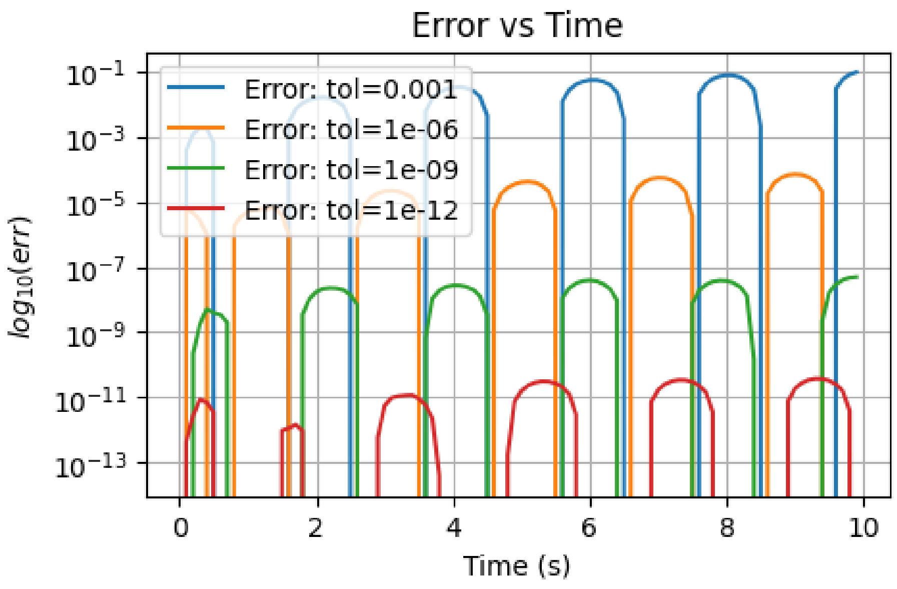
```
error_list = [[x[idx]-exact_x(times[idx]) for idx in range(len(x))] for x in x_list]
```

(d) Plot the four error curves on the same graph and comment on your observations. Make sure to provide your computer program along with your solution. Hint: Since the errors between the exact and approximate solutions will be small, consider using a logarithmic scale for the y-axis on the error plot for better visualization.

Note: You may use software packages other than MATLAB for this problem, as long as they support numerical integration with adjustable tolerances.

```
plt.figure(figsize=(5,3))
plt.plot(times,exact_x(times),label="exact")
for tol,x in zip(tolerances,x_list):
    plt.plot(times,x,label=f"tol={tol}")
plt.xlabel("Time (s)")
plt.ylabel(r"$x_{num} (m)$")
plt.title("Estimated Solution vs Time")
plt.legend(loc=2)
plt.grid()
plt.show()

plt.figure(figsize=(5,3))
for tol,errors in zip(tolerances,error_list):
    plt.plot(times,errors,label=f"Error: tol={tol}")
plt.xlabel("Time (s)")
plt.ylabel(r"$log_{10}(err)$")
plt.title("Error vs Time")
plt.yscale("log")
plt.legend(loc=2)
plt.grid()
plt.show()
```





As shown in the plot above, we see that error is sinusoidal and increases by accumulation over time. We can also see that as the tolerance decreases, the magnitude of the error decreases.

From this, we can see that error is a big deal, and must be dealt with, specifically, for a spacecraft one must periodically use sensor data to correct errors in the state vector.

3. Attitude Description:

The initial (3-2-1) Euler angles are given as $(\psi_0, \theta_0, \phi_0) = (40, 30, 80)$.

Assume the angular velocity in the body-fixed frame is,

$$\vec{\omega}(t) = 20^{s^{-1}} \begin{bmatrix} sin(0.1t) \\ 0.01 \\ cos(0.1t) \end{bmatrix}$$

```
def omega(t):
    return 20/180*np.pi*np.array([np.sin(0.1*t),0.01,np.cos(0.1*t)])

rtol = 1e-12
atol = 1e-12

times = np.arange(0,60,0.1)

EA0 = np.radians(np.array([40,30,80]))

psi = EA0[0]
theta = EA0[1]
phi = EA0[2]
```

Perform the following tasks:

(a) Convert the initial (3-2-1) Euler angles to the corresponding quaternion vector.

Solution:

1. Convert Euler angles to the associated 321 Rotation Matrix (T)

$$T_{321} = \begin{bmatrix} c(\theta)c(\psi) & s(\theta) & c(\theta)s(\psi) \\ -c(\phi)s(\psi)+s(\phi)s(\theta)c(\psi) & s(\phi)c(\theta) & c(\phi)c(\psi)+s(\phi)s(\theta)s(\psi) \\ s(\phi)s(\psi)+c(\phi)s(\theta)c(\psi) & c(\phi)c(\theta) & -s(\phi)c(\psi)+c(\phi)s(\theta)s(\psi) \end{bmatrix}$$

```
T = np.array([[np.cos(theta)*np.cos(psi),np.cos(theta)*np.sin(psi),-np.sin(theta)],
              [-np.cos(phi)*np.sin(psi)+np.sin(phi)*np.sin(theta)*np.cos(psi),np.cos(phi)*
              [np.sin(phi)*np.sin(psi)+np.cos(phi)*np.sin(theta)*np.cos(psi),-np.sin(phi)*

displayH(sy.Symbol(r"T_{321} ="),sy.Matrix(T))
```

$$T_{321} = \begin{bmatrix} 0.663413948168938 & 0.556670399226419 & -0.5 \\ 0.265584356318795 & 0.449533332339233 & 0.852868531952443 \\ 0.699533332339233 & -0.698597058211014 & 0.150383733180435 \end{bmatrix}$$

2. Find Principal Rotation Vector ($\hat{e}$) and Angle (Ψ)

$$\Psi = cos^{-1}\left(\frac{trace(T)-1}{2}\right)$$

$$\hat{e} = \frac{1}{2sin(\Psi)}\begin{bmatrix} T_{1,2}-T_{2,1} \\ T_{2,0}-T_{0,2} \\ T_{0,1}-T_{1,0} \end{bmatrix}$$

```
PSI = np.arccos(0.5*(np.trace(T)-1))
e = 0.5/np.sin(PSI)*np.array([T[1,2]-T[2,1],
                              T[2,0]-T[0,2],
                              T[0,1]-T[1,0]])

displayH(sy.Symbol(r"\Psi ="),np.degrees(PSI),sy.Symbol(r"\degree"))
displayH(sy.Symbol(r"\hat{e} ="),sy.Matrix(e))
```

$\Psi = 82.4341538041605$

$$\hat{e} = \begin{bmatrix} 0.782545478181913 \\ 0.605033969881118 \\ 0.146821217359535 \end{bmatrix}$$

3. Find corresponding quaternion

$$\boxed{\vec{q} = \begin{bmatrix} c(\Psi/2) \\ e_0 s(\Psi/2) \\ e_1 s(\Psi/2) \\ e_2 s(\Psi/2) \end{bmatrix}}$$

```
Q0 = np.array([np.cos(PSI/2),
               e[0]*np.sin(PSI/2),
               e[1]*np.sin(PSI/2),
               e[2]*np.sin(PSI/2)])

displayH(sy.Symbol(r"\vec{q}_{0} ="),sy.Matrix(Q0))
```

$$\vec{q}_0 = \begin{bmatrix} 0.752218554292668 \\ 0.515629926073262 \\ 0.398665163699394 \\ 0.0967425096225328 \end{bmatrix}$$

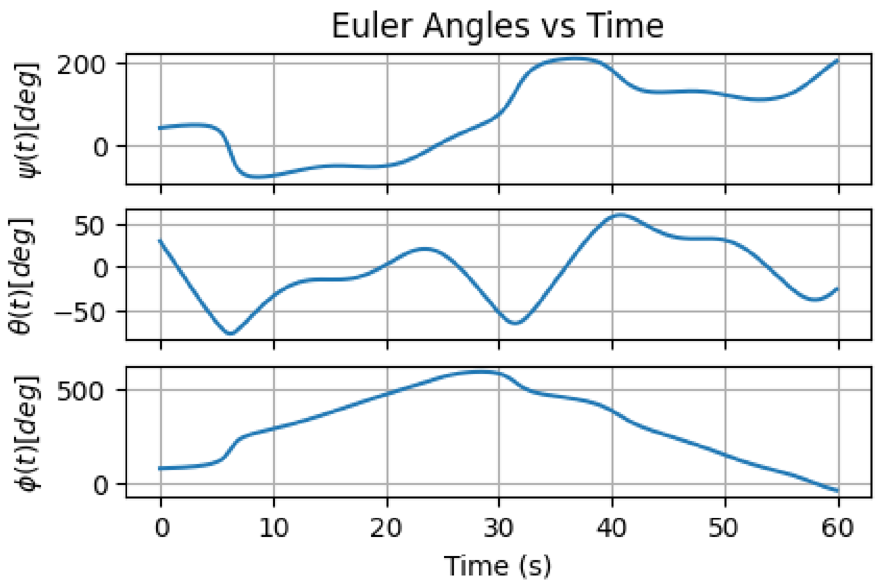
(b) Write a program to numerically integrate both the (3-2-1) Euler angles and the quaternion vector over a simulation time of 1 minute. Plot the time histories.

Solution:

1. Write the KDE for the (321) Euler Angles.

$$\begin{bmatrix} \dot{\psi} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \\ 0 & c(\phi) & -s(\phi) \\ 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \end{bmatrix} \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \end{bmatrix}$$

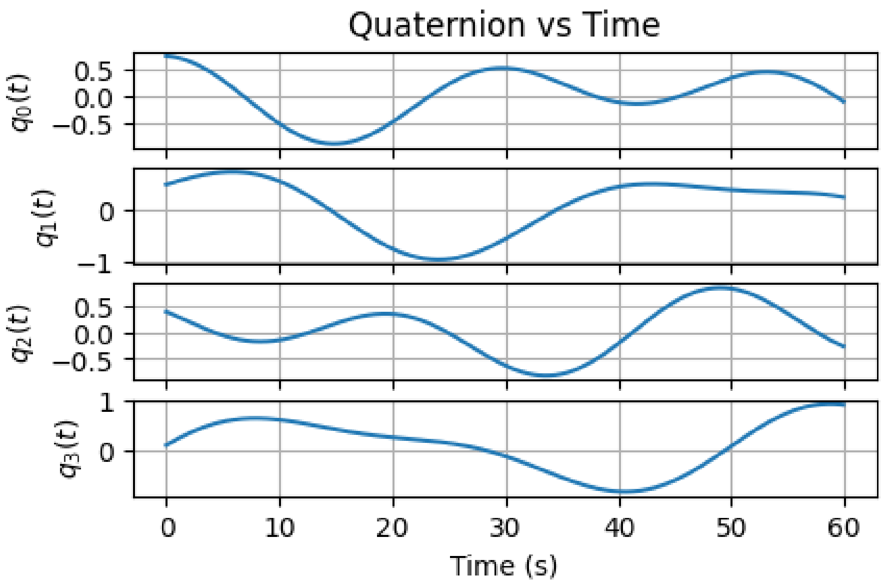
```
def d_EA(EA,time):
    c = np.cos;s = np.sin;t = np.tan
    psi = EA[0];theta = EA[1];phi = EA[2]
    return np.array([[0,s(phi)/c(theta),c(phi)/c(theta)],
                     [0,c(phi),-s(phi)],
                     [1,s(phi)*t(theta),c(phi)*t(theta)]])@omega(time)
EA = scipy.integrate.odeint(d_EA,EA0,times,rtol=rtol,atol=atol)
fig,axes = plt.subplots(3,1,sharex=True,figsize=(5,3))
axes[0].set_title("Euler Angles vs Time")
for idx in range(3):axes[idx].plot(times,np.degrees(EA[:,idx]));axes[idx].grid()
axes[0].set_ylabel(r'$\psi (t) [deg]$');axes[1].set_ylabel(r'$\theta (t) [deg]$');axes[2].
plt.xlabel("Time (s)")
plt.show()
```



2. Write the KDE for the Quaternion.

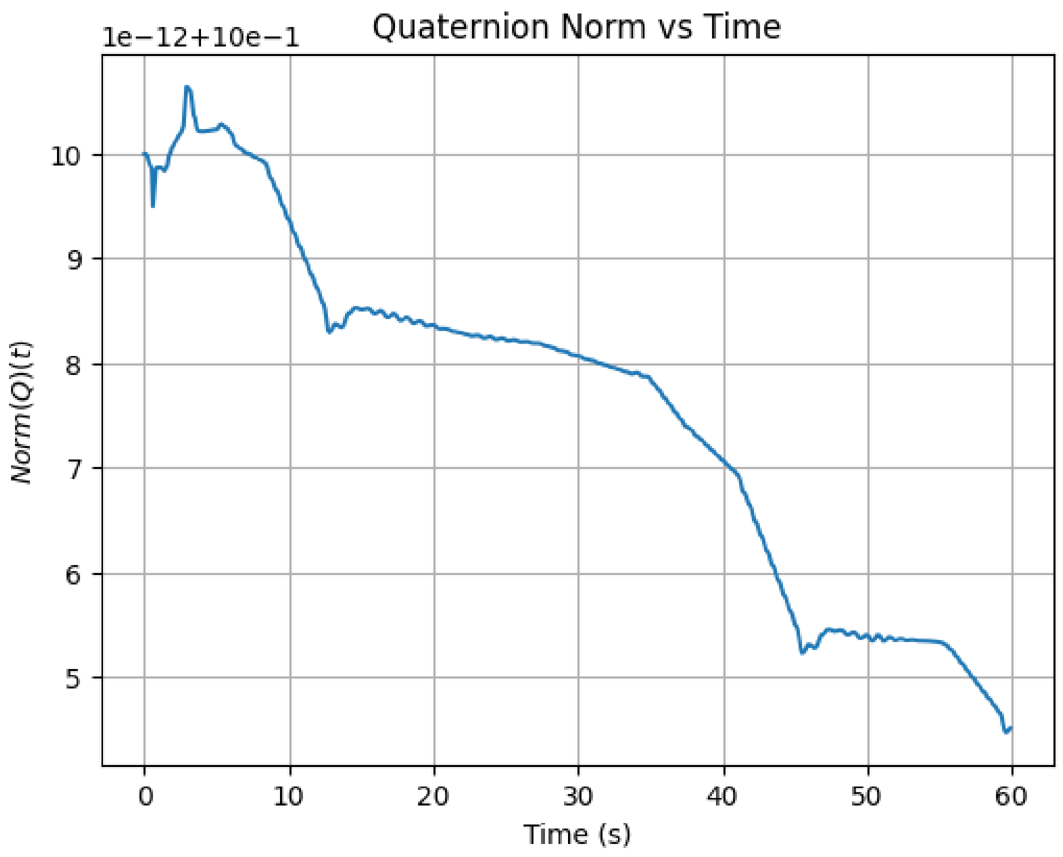
$$\underline{\dot{q}} = \frac{1}{2}\begin{bmatrix} q_0 & -q_1 \\ -q_2 & -q_3 \\ q_1 & q_0 \\ -q_3 & q_2 \\ q_2 & q_3 \\ q_0 & -q_1 \\ q_3 & -q_2 \\ q_1 & q_0 \end{bmatrix}\begin{bmatrix} 0 \\ \omega_0 \\ \omega_1 \\ \omega_2 \end{bmatrix}$$

```
def d_Quat(Q,time):
    A = np.array([[Q[0],-Q[1],-Q[2],-Q[3]],
                  [Q[1],Q[0],-Q[3],Q[2]],
                  [Q[2],Q[3],Q[0],-Q[1]],
                  [Q[3],-Q[2],Q[1],Q[0]]])
    return 0.5*A@np.insert(omega(time),0,0)

Q = scipy.integrate.odeint(d_Quat,Q0,times,rtol=rtol,atol=atol)

fig,axes = plt.subplots(4,1,sharex=True,figsize=(5,3))
axes[0].set_title("Quaternion vs Time")
axes[0].plot(times,[q[0] for q in Q])
axes[0].grid()
for idx in range(1,4):
    axes[idx].plot(times,[q[idx] for q in Q])
    axes[idx].grid()
axes[0].set_ylabel(r'$q_{0} (t)$')
axes[1].set_ylabel(r'$q_{1} (t)$')
axes[2].set_ylabel(r'$q_{2} (t)$')
axes[3].set_ylabel(r'$q_{3} (t)$')
plt.xlabel("Time (s)")
plt.show()
```



(c) Additionally, plot the norm of the Quaternion vector at each time. Comment on these values.

```
plt.figure()
plt.title("Quaternion Norm vs Time")
plt.plot(times,[np.linalg.norm(q) for q in Q])
plt.grid()
plt.ylabel(r'$Norm(Q) (t)$')
plt.xlabel("Time (s)")
plt.show()
```



We can see that the norm of the quaternion is roughly equal to 1, which is expected. However, we can also see that error starts to accumulate over time, meaning that we need an additional sensor reading to correct the orientation every once in a while.

(d) Convert the quaternion vector at the final simulation time back to the (3-2-1) Euler angles. Compare these values to the final Euler angles obtained from numerically integrating the (3-2-1) Euler angles. Are these values the same?

Solution:

1. Convert the final quaternion to a dcm.

$$T_{123} = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_3-q_0q_2) & 2(q_1q_2+q_0q_3) \\ 2(q_1q_2-q_0q_3) & 2(q_2q_3+q_0q_1) & 1-2(q_1^2+q_3^2) \\ 2(q_1q_3+q_0q_2) & 1-2(q_2^2+q_2^2) & 2(q_2q_3-q_0q_1) \end{bmatrix}$$

```
w, x, y, z = Q[-1]
xx, xy, xz, xw = x * x, x * y, x * z, x * w
yy, yz, yw = y * y, y * z, y * w
zz, zw = z * z, z * w
T = np.array([
    [1 - 2 * (yy + zz),     2 * (xy + zw),     2 * (xz - yw)],
    [    2 * (xy - zw), 1 - 2 * (xx + zz),     2 * (yz + xw)],
    [    2 * (xz + yw),     2 * (yz - xw), 1 - 2 * (xx + yy)]
    ])
displayH(sy.Symbol(r"T_{321} ="),sy.Matrix(T))
```

$$T_{321} = \begin{bmatrix} -0.836012023004318 & -0.323658551596437 & 0.443090328635609 \\ 0.0440607960823473 & -0.844500726064536 & -0.533738859260372 \\ 0.546939250374761 & -0.42668919088075 & 0.720273413905361 \end{bmatrix}$$

2. Convert DCM to Euler Angles.

$$\begin{bmatrix} \psi \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} tan^{-1}\left(\frac{T_{0,1}}{T_{0,0}}\right) \\ -sin^{-1}(T_{0,2}) \\ tan^{-1}\left(\frac{T_{1,2}}{T_{2,2}}\right) \end{bmatrix}$$

```
EAf = np.array([np.arctan2(T[0,1],T[0,0]),-np.arcsin(T[0,2]),np.arctan2(T[1,2],T[2,2])])

displayH(sy.Symbol(r"\left[\begin{array}{ccc}\psi_{q} \\ \theta_{q} \\ \phi_{q} \end{array
displayH(sy.Symbol(r"\left[\begin{array}{ccc}\psi_{EA} \\ \theta_{EA} \\ \phi_{EA} \end{ar
```

$$\begin{bmatrix} \psi_q \\ \theta_q \\ \phi_q \end{bmatrix} = \begin{bmatrix} -2.7722163512284 \\ -0.459042940962261 \\ -0.637730754611225 \end{bmatrix} \xrightarrow{wrap(2\pi)} \begin{bmatrix} 3.51096895595119 \\ 5.82414236621733 \\ 5.64545455256836 \end{bmatrix}$$

$$\begin{bmatrix} \psi_{EA} \\ \theta_{EA} \\ \phi_{EA} \end{bmatrix} = \begin{bmatrix} 3.51096895594676 \\ -0.459042940983694 \\ -0.637730754641806 \end{bmatrix} \xrightarrow{wrap(2\pi)} \begin{bmatrix} 3.51096895594676 \\ 5.82414236619589 \\ 5.64545455253778 \end{bmatrix}$$

Comparing the difference in the euler angles, we see that they represent the same orientation.