??2

October 16, 2019

0.1 Problem 5

[1]:
```
import numpy as np

# Define one-liner R(t) function
R = lambda t: np.matrix([[np.cos(t), -np.sin(t), 0],
                         [np.sin(t),  np.cos(t), 0],
                         [0,          0,         1]])

# Define print func
def rotation_property(t):
    print('R(t) for t = ', t)
    r = R(t)
    print('r.T: \n', r.T)
    print('r.I: \n', r.I)
    print('r inverse: \n', np.linalg.inv(r))
    print('r determinat: \n', np.linalg.det(r))
    n = r[:,0]
    s = r[:,1]
    a = r[:,2]
    print('norm of n: \n', np.linalg.norm(n))
    print('norm of s: \n', np.linalg.norm(s))
    print('norm of a: \n', np.linalg.norm(a))
    print('n * s: \n', np.dot(n.T,s))
    print('s * a: \n', np.dot(s.T,a))
    print('s * n: \n', np.dot(s.T,n))
```

[2]:
```
rotation_property(np.pi/6)
```

```
R(t) for t =  0.5235987755982988
r.T:
 [[ 0.8660254  0.5        0.       ]
 [-0.5        0.8660254  0.       ]
 [ 0.         0.         1.       ]]
r.I:
 [[ 0.8660254  0.5        0.       ]
 [-0.5        0.8660254  0.       ]
 [ 0.         0.         1.       ]]
```

```
r inverse:
 [[ 0.8660254  0.5        0.       ]
 [-0.5        0.8660254  0.       ]
 [ 0.         0.         1.       ]]
r determinat:
 1.0
norm of n:
 1.0
norm of s:
 1.0
norm of a:
 1.0
n * s:
 [[0.]]
s * a:
 [[0.]]
s * n:
 [[0.]]
```

[3]:
```
rotation_property(np.pi)
```

```
R(t) for t =  3.141592653589793
r.T:
 [[-1.0000000e+00  1.2246468e-16  0.0000000e+00]
 [-1.2246468e-16 -1.0000000e+00  0.0000000e+00]
 [ 0.0000000e+00  0.0000000e+00  1.0000000e+00]]
r.I:
 [[-1.0000000e+00  1.2246468e-16 -0.0000000e+00]
 [-1.2246468e-16 -1.0000000e+00 -0.0000000e+00]
 [ 0.0000000e+00  0.0000000e+00  1.0000000e+00]]
r inverse:
 [[-1.0000000e+00  1.2246468e-16 -0.0000000e+00]
 [-1.2246468e-16 -1.0000000e+00 -0.0000000e+00]
 [ 0.0000000e+00  0.0000000e+00  1.0000000e+00]]
r determinat:
 1.0
norm of n:
 1.0
norm of s:
 1.0
norm of a:
 1.0
n * s:
 [[0.]]
s * a:
 [[0.]]
s * n:
 [[0.]]
```

[4]:
```
rotation_property(np.pi/4)
```

```
R(t) for t =  0.7853981633974483
r.T:
 [[ 0.70710678  0.70710678  0.        ]
 [-0.70710678  0.70710678  0.        ]
 [ 0.          0.          1.        ]]
r.I:
 [[ 0.70710678  0.70710678  0.        ]
 [-0.70710678  0.70710678  0.        ]
 [ 0.          0.          1.        ]]
r inverse:
 [[ 0.70710678  0.70710678  0.        ]
 [-0.70710678  0.70710678  0.        ]
 [ 0.          0.          1.        ]]
r determinat:
 1.0
norm of n:
 1.0
norm of s:
 1.0
norm of a:
 1.0
n * s:
 [[0.]]
s * a:
 [[0.]]
s * n:
 [[0.]]
```

0.2 Problem 6

[5]:
```
skew = lambda v : np.matrix([[0,    -v[2], v[1]],
                             [v[2], 0,    -v[0]],
                             [-v[1], v[0],    0]])

def verify_skew(v1, v2):
    print('v1 x v2 = ', np.cross(v1, v2))
    print('S(v1)v2 = ', skew(v1).dot(v2))
    print('v1 x v2 - S(v1)v2 = ', np.cross(v1, v2) - skew(v1).dot(v2))
```

[6]:
```
verify_skew(np.array([1, 0, 0]), np.array([0, 1, 0]))
```

```
v1 x v2 =  [0 0 1]
S(v1)v2 =  [[0 0 1]]
v1 x v2 - S(v1)v2 =  [[0 0 0]]
```

[7]:
```
verify_skew(np.array([1, 1, 1]), np.array([-1, -1, -1]))
```

```
v1 x v2 =  [0 0 0]
S(v1)v2 =  [[0 0 0]]
v1 x v2 - S(v1)v2 =  [[0 0 0]]
```

[8]:
```
verify_skew(np.array([1, 2, 3]), np.array([3, 2, 1]))
```

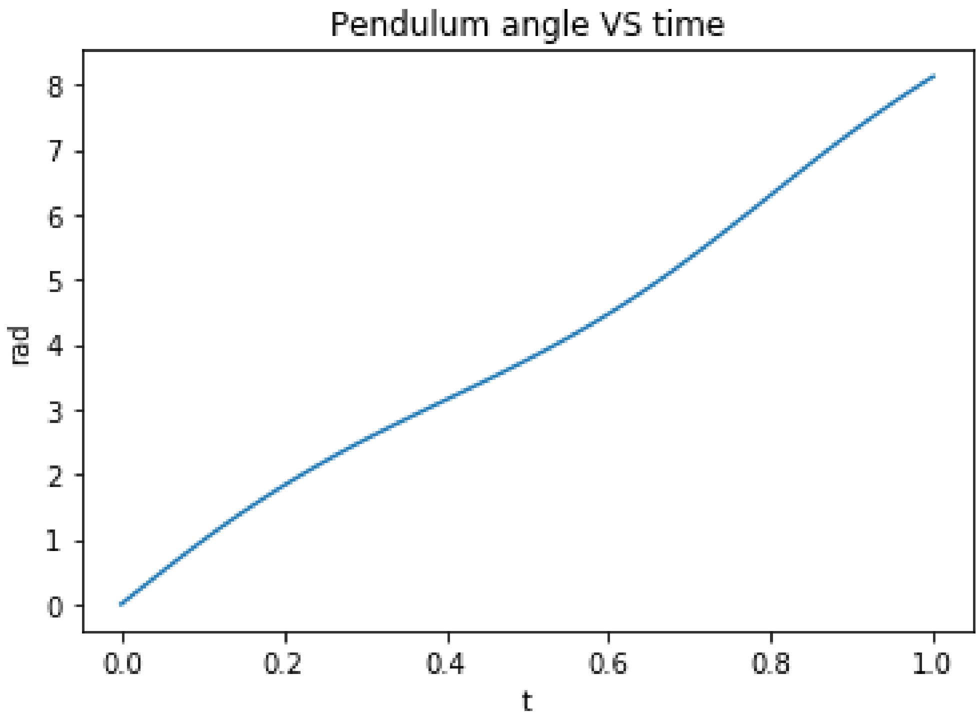
```
v1 x v2 =  [-4  8 -4]
S(v1)v2 =  [[-4  8 -4]]
v1 x v2 - S(v1)v2 =  [[0 0 0]]
```

0.3 Problem 7

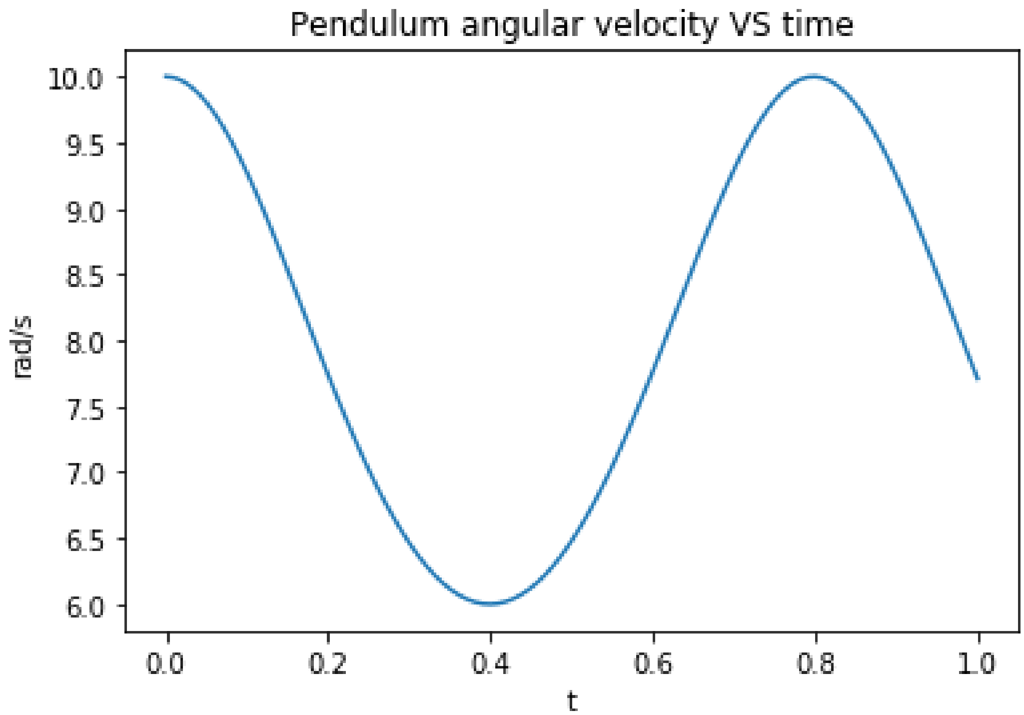
[9]:
```
from scipy.integrate import odeint
import matplotlib.pyplot as plt

def derfunc(y, t):
    g = 32 # ft/s^2
    L = 2  # ft
    ydot = [0, 0]
    ydot[0] = y[1]
    ydot[1] = -(g / L) * np.sin(y[0])
    return ydot
```

[10]:
```
y0 = [0, 10]
t = np.linspace(0, 1, 101)
sol = odeint(derfunc, y0, t)
```

[11]:
```
plt.plot(t, sol[:, 0], label = 'pendulum angle')
plt.xlabel('t')
plt.ylabel('rad')
plt.title('Pendulum angle VS time')
plt.show()
```



[12]:
```
plt.plot(t, sol[:, 1], label = 'pendulum angular velocity')
plt.xlabel('t')
plt.ylabel('rad/s')
plt.title('Pendulum angular velocity VS time')
plt.show()
```


Pendulum angular velocity VS time
10.0
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
rad/s
0.0
0.2
0.4
0.6
0.8
1.0
t