6elen018w_tutorial2_2025_code

October 31, 2024

6ELEN018W - Tutorial 2 2025 Solutions

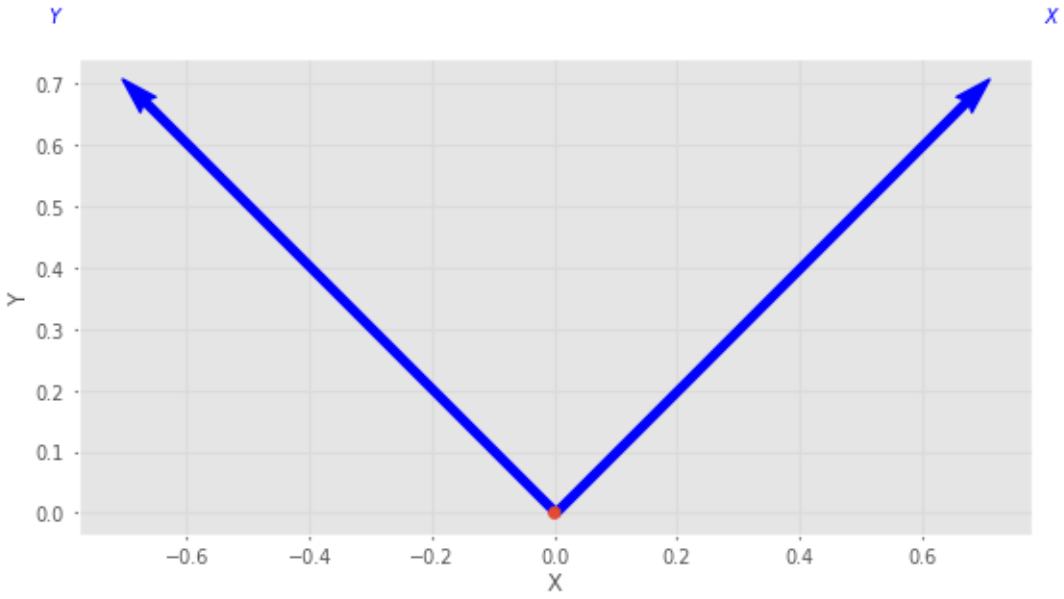
[52]:
```
from sympy import *
from roboticstoolbox import *
from spatialmath.base import *
import math
import numpy as np
```

Exercise 1

[53]:
```
e1 = trot2(math.pi/4)
trplot2(e1, color='b')

P = [3, 5, 1]

P2 = np.array(e1)@np.array(P)
print(f'P2: {P2}')

e2 = np.linalg.inv(e1)  # inverse the transformation
np.array(e2)@P2  # we get back the original P
```

```
P2: [-1.414  5.657  1.   ]
```

[53]:
```
array([3., 5., 1.])
```



Exercise 2

[54]:
```
import math

def ex2(theta, units):
    if units == 'deg':
        theta = math.radians(theta)
    s = [[cos(theta), -sin(theta)],
         [sin(theta), cos(theta)]]

    return s

print(ex2(90, 'deg'))

rot2(math.pi/2)
```

```
[[6.12323399573677e-17, -1.00000000000000], [1.00000000000000,
6.12323399573677e-17]]
```

[54]:
```
array([[ 0., -1.],
       [ 1.,  0.]])
```

Exercise 3

[55]:
```
def ex3(theta, units):
    if units == 'deg':
        theta = math.radians(theta)
    s = np.array([[cos(theta), -sin(theta)],
         [sin(theta), cos(theta)]])

    return s

print(ex2(90, 'deg'))
```

```
[[6.12323399573677e-17, -1.00000000000000], [1.00000000000000,
6.12323399573677e-17]]
```

Exercise 4

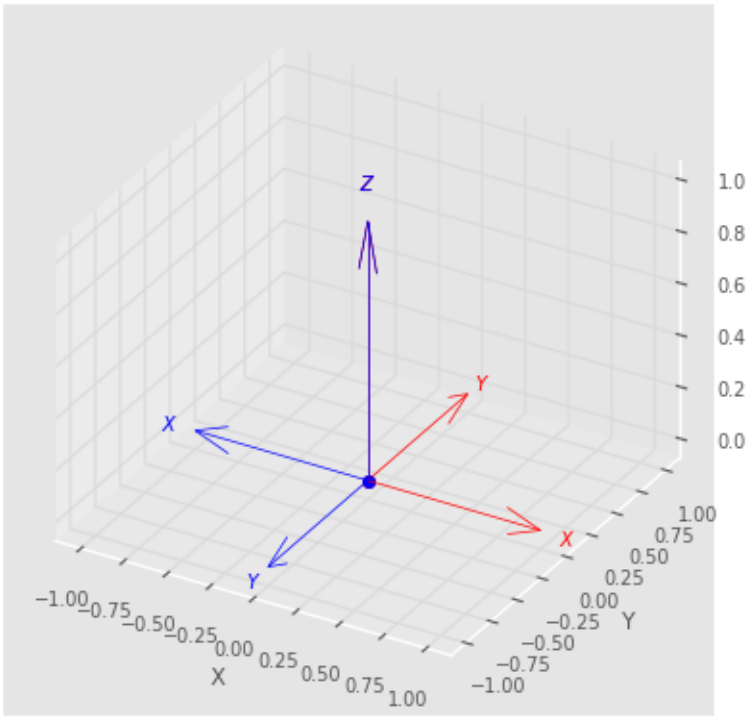
[56]:
```
def ex4(theta, tr_list):
    s = [[cos(theta), -sin(theta), tr_list[0]],
         [sin(theta), cos(theta),  tr_list[1]],
         [0,              0,          1]]

    return np.array(s)

print(ex4(math.pi, [1, 2]))

# equivalent with the robotics toolbox
print()
print(transl2(1,2) @ trot2(math.pi))
```

```
[[-1.00000000000000 -1.22464679914735e-16 1]
 [1.22464679914735e-16 -1.00000000000000 2]
 [0 0 1]]

[[-1. -0.  1.]
 [ 0. -1.  2.]
 [ 0.  0.  1.]]
```

Exercise 5

[57]:
```
# original frame
R_orig = rotz(0)
trplot(R_orig, color = 'r')

R = rotz(math.pi)
trplot(R)  # plot in blue (default colour)
```

[57]:
```
<Axes3D: xlabel='X', ylabel='Y', zlabel='Z'>
```



Exercise 6

[61]:
```
def ex6(theta, axis):
    if axis == 'x':
        R = [[1, 0, 0, 0],
             [0, cos(theta), -sin(theta), 0],
             [0, sin(theta), cos(theta), 0],
             [0, 0, 0, 1]]
    elif axis == 'y':
        R = [[cos(theta), 0, sin(theta), 0],
             [0, 1, 0, 0],
              [-sin(theta), 0, cos(theta), 0],
             [0, 0, 0, 1]]
    else:  # default is 'z'
        R = [[cos(theta), -sin(theta), 0, 0],
             [sin(theta), cos(theta), 0, 0],
             [0, 0, 1, 0],
             [0, 0, 0, 1]]

    return np.array(R)

# suppress scientific notation for numpy
np.set_printoptions(suppress=True, precision=3)
```

```
# rotate about 'x'
print(f'ex6: {ex6(math.pi, "x")}')
print(f'Toobox: {trotx(math.pi)}')

# rotate about 'y'
print(f'\nex6: {ex6(math.pi, "y")}')
print(f'Toobox: {troty(math.pi)}')

# rotate about 'z'
print(f'\nex6: {ex6(math.pi, "z")}')
print(f'Toobox: {trotz(math.pi)}')
```

```
ex6: [[1 0 0 0]
 [0 -1.00000000000000 -1.22464679914735e-16 0]
 [0 1.22464679914735e-16 -1.00000000000000 0]
 [0 0 0 1]]
Toobox: [[ 1.  0.  0.  0.]
 [ 0. -1. -0.  0.]
 [ 0.  0. -1.  0.]
 [ 0.  0.  0.  1.]]

ex6: [[-1.00000000000000 0 1.22464679914735e-16 0]
 [0 1 0 0]
 [-1.22464679914735e-16 0 -1.00000000000000 0]
 [0 0 0 1]]
Toobox: [[-1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [-0.  0. -1.  0.]
 [ 0.  0.  0.  1.]]

ex6: [[-1.00000000000000 -1.22464679914735e-16 0 0]
 [1.22464679914735e-16 -1.00000000000000 0 0]
 [0 0 1 0]
 [0 0 0 1]]
Toobox: [[-1. -0.  0.  0.]
 [ 0. -1.  0.  0.]
 [ 0.  0.  1.  0.]
 [ 0.  0.  0.  1.]]
```

[62]:
```
a = np.array([[0.123456, 0.123456],
              [0.123456, 0.123456]])

print(type(ex6(math.pi, "z")))
```

```
<class 'numpy.ndarray'>
```

Exercise 7

[64]:
```
def ex7(P, T):
    return T@P

# the vector
p1 = np.array([3, 5])
# the angle
theta = math.pi/2
# the rotation transformation
t1 = np.array([[cos(theta), -sin(theta)],
     [sin(theta), cos(theta)]])

# call the function
res1 = ex7(p1, t1)
print(res1)

# robotics toolbox check
rot2(theta)@p1
```

```
[-5.00000000000000 3.00000000000000]
```

[64]:
```
array([-5.,  3.])
```