6ELEN018W - Tutorial 2 2026 Solutions

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

Exercise 1

[ ]:
```python
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
```

Exercise 2

[ ]:
```python
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)
```

Exercise 3

[ ]:
```
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'))
```

Exercise 4

[ ]:
```
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))
```

Exercise 5

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

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

Exercise 6

[ ]:
```
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)}')
```

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

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

Exercise 7

[ ]:
```
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
```

Exercise 8

[ ]:
```
T = transl(10, 35, 0)
R = trotz(math.radians(45))

T_total = T@R
pos_V = [5, 2, 38, 1]
pos_W = T_total@pos_V  ## (1)
print(pos_W)
# verify that if we do the inverse transform we get back the original␣
↪coordinates
print(f'original coordinates calculation: {np.linalg.inv(T_total)@pos_W}')
```

[ ]:
```
pos_W = [1, 30, 5, 1]  # pos in W

# (1) => pos_V = T_Total^-1 * pos_W
np.linalg.inv(T_total)@pos_W
```

[ ]:
```

```