5ELEN018W - Tutorial 4 2026 Solutions

[ ]:

```
import math
import numpy as np
from scipy import linalg, optimize
import matplotlib.pyplot as plt
from spatialmath import *
from spatialmath.base import *
from spatialmath.base import sym
from spatialgeometry import *
from roboticstoolbox import *
```

Exercise 1

[ ]:

```
from math import *

q1 = radians(90)
tr1 = np.array([[cos(q1),  -sin(q1), 0, 0],
                [sin(q1),  cos(q1),  0, 0],
                [0,         0,       1, 0],
                [0,         0,       0, 1]])
tr2 = np.array([[cos(q1),  0, sin(q1), 0],
                [0,        1, 0,       0],
                [-sin(q1), 0, cos(q1), 0],
                [0,        0, 0,       1]])
tr3 = np.array([[1, 0, 0, 4],
                [0, 1, 0, -3],
                [0, 0, 1, 7],
                [0, 0, 0, 1]])
P = np.array([7, 3, 1, 1])

tr3@tr2@tr1@P
```

Exercise 2

[ ]:
```python
from sympy import *

theta, r, d, alpha = symbols('theta r d alpha')

A = [[cos(theta), -sin(theta), 0, 0], [sin(theta), cos(theta), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]
# convert to sympy matrix
A = Matrix(A)

B = [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, d], [0, 0, 0, 1]]
B = Matrix(B)

C = [[1, 0, 0, r], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]
C = Matrix(C)

D = [[1, 0, 0, 0], [0, cos(alpha), -sin(alpha), 0], [0, sin(alpha), cos(alpha), 0], [0, 0, 0, 1]]
D = Matrix(D)

res = simplify(A@B@C@D)
res
```

0.1 Exercise 3

[ ]:
```python
from sympy import *

theta1, r1, d1, alpha1 = symbols('theta1 r1 d1 alpha1')
theta2, r2, d2, alpha2 = symbols('theta2 r2 d2 alpha2')

# let's use the robotics toolbox in this exercise
t1 = trotz(theta1)
t2 = transl(0, 0, d1)
t3 = transl(r1, 0, 0)
t4 = trotx(alpha1)

# DH for joint 1
J1 = t1@t2@t3@t4

t5 = trotz(theta2)
t6 = transl(0, 0, d2)
t7 = transl(r2, 0, 0)
t8 = trotx(alpha2)

# DH for joint 2
J2 = t5@t6@t7@t8
```

```
# Overall DH
simplify(Matrix(J1@J2))
```

Exercise 4

[ ]:

```
from sympy import *
from math import *

theta1, r1, theta2, theta3, d3 = symbols('theta1, r1, theta2, theta3, d3')

J1 = trotz(theta1)@transl(r1, 0, 0)

J2 = trotz(theta2 + math.pi/2)@trotx(math.pi/2)

J3 = trotz(theta3)@transl(0, 0, d3)

Matrix(simplify(J1@J2@J3))
```

Exercise 5

[ ]:

```
def DH(theta, d, r, alpha):
    t1 = trotz(theta)
    t2 = transl(0, 0, d)
    t3 = transl(r, 0, 0)
    t4 = trotx(alpha)

    return t1@t2@t3@t4
```

Exercise 6

[ ]:

```
theta1, theta2, theta3, theta4, theta5, theta6, r2, r3, d3, d4 =␣
  ↪symbols('theta1, theta2, theta3, theta4, theta5, theta6, r2, r3, d3, d4')

table = [[theta1, 0, 0, 0],
         [theta2, 0, 0, -math.pi/2],
         [theta3, r2, d3, 0],
         [theta4, r3, d4, -math.pi/2],
         [theta5, 0, 0, math.pi/2],
         [theta6, 0, 0, -math.pi/2]]

total_dh = np.identity(4)  # The identity array I is a square array with ones␣
  ↪on the main diagonal.
for row in table:
    theta = row[0]
    d = row[1]
    r = row[2]
    alpha = row[3]

    total_dh = total_dh@DH(theta, d, r, alpha)

print(total_dh)
```