Tutorial 9 - Robotics Toolbox for Python

Installing the toolbox

Assuming you have already installed pythons, in the command line execute:

`pip install roboticstoolbox-python`

You might need to run `pip3` instead of `pip` above, depending on your system.

Trying the Toolbox

You will be better if you install and use `jupyterlab` to type your code below. This is similar to the Matlab live scripts we have seen.

In [ ]:
```
import roboticstoolbox as rtb
```

In [ ]:
```
from spatialmath import *
```

Supports different transformations such as rotations, raw-pitch-yaw angles, etc.

In [ ]:
```
# from spatialmath.base import
# T = transl(0.5, 0.0, 0.0) @ rpy2tr(0.1, 0.2, 0.3, order="xyz") @ trotx(-90, 'deg')
```

In [ ]:
```
#print(T)
```

In [ ]:
```
T = SE3(0.5, 0.0, 0.0) * SE3.RPY([0.1, 0.2, 0.3], order='xyz') * SE3.Rx(-90, unit='deg')
```

In [ ]:
```
print(T)
```

In [ ]:
```
T.R
```

In [ ]:
```
T.t
```

In [ ]:
```
T.plot(color='red')
```

Support for *inverse* operations:

In [ ]:
```
T.inv()
```

Creating Robot Models using DH notation

In [ ]:
```
from roboticstoolbox.robot import *
from math import *

robot = DHRobot(
    [RevoluteDH(alpha=pi/2), # not assigned values to parameters correspond to 0 values
     RevoluteDH(a=0.4318),  # a corresponds to r as we named it in the slides
     RevoluteDH(d=0.15005, a=0.0203, alpha=-pi/2),
     RevoluteDH(d=0.4318, alpha=pi/2),
     RevoluteDH(alpha=-pi/2),
     RevoluteDH()
    ], name="Puma560")
```

In [ ]:
```
print(robot)
```

Perform kinematic operations. Could use predefined or your own models. Forward kinematics:

In [ ]:
```
puma = rtb.models.DH.Puma560()
```

In [ ]:
```
print(puma)
```

Various configurations are predefined:

qz = (0, 0, 0, 0, 0, 0) # zero angle qr = (0, pi/2, −pi/2, 0, 0, 0) # read the arm is straight and vertical qs = (0, 0, −pi/2, 0, 0, 0) # stretch the arm is straighrt and horizontal qn = 0, pi/4, −pi, 0, pi, 0) # nominal, the arm is in a dextrous working pose

In [ ]:
```
print(puma.qr)
```

In [ ]:
```
T = puma.fkine([0.1, 0.2, 0.3, 0.4, 0.5, 0.6])  # forward kinematics
```

In [ ]:
```
print(T)
```

In [ ]:
```
print(T)
```

In [ ]:
```
sol = puma.ikine_LM(T)
```

In [ ]:
```
print(sol)
```

In [ ]:
```
puma.plot(sol.q)
```

For some robot configurations the inverse kinematics problem can be solved analytically:

In [ ]:
```
puma = rtb.models.DH.Puma560()       # instantiate robot model
```

In [ ]:
```
T = puma.fkine([0.1, 0.2, 0.3, 0.4, 0.5, 0.6])
```

In [ ]:
```
puma.ikine_a(T, config="lun")
```

This toolbox support URDF (Unified Robot Description Format) format as well, which is a widely used XML-based format for the description of robo models.

In [ ]:
```
panda = rtb.models.URDF.Panda()
```

In [ ]:
```
print(panda)
```

In [ ]:
```
T = panda.fkine(panda.qz, end='panda_hand')
```

In [ ]:
```
print(T)
```

In [ ]:
```
panda.plot(panda.qz, backend="swift")
```