Introduction to Robotic Toolbox for Kinematics of Manipulators - IRB 140

The Robotics Toolbox for MATLAB and Python is a collection of functions and classes that provides tools for modeling, simulating, and analyzing robotic systems. The toolbox supports forward and inverse kinematics, dynamics, and trajectory generation among other functionalities.

```
import roboticstoolbox as rtb
from spatialmath import SE3
from math import pi
```

Importing Libraries:

1) roboticstoolbox : which provides tools for robot modeling, simulation, and analysis. 2) SE3: used to represent 3D transformations (translation and rotation).

```
irb140 = rtb.models.DH.IRB140()
irb140
```

DHRobot: IRB 140 (by ABB), 6 joints (RRRRRR), dynamics, geometry, standard DH parameters

θ_j	d_j	a_j	α_j	q^-	q^+
q1	0.352	0.07	-90.0°	-180.0°	180.0°
q2	0	0.36	0.0°	-100.0°	100.0°
q3	0	0	-90.0°	-220.0°	60.0°
q4	0.38	0	90.0°	-200.0°	200.0°
q5	0	0	-90.0°	-120.0°	120.0°
q6	0.065	0	0.0°	-400.0°	400.0°

name	q0	q1	q2	q3	q4	q5
qr	0°	-90°	90°	0°	90°	-90°
qz	0°	0°	0°	0°	0°	0°
qd	0°	-90°	180°	0°	0°	-90°

Creates an instance of the ABB IRB 140 robot using the built-in Denavit-Hartenberg model provided by the Robotics Toolbox. This model includes the kinematic parameters (link lengths, offsets, etc.) for the IRB 140.

```
q = [0, -pi/2, pi/2, 0, pi/2, 0]

T = irb140.fkine(q)
print(f'Forward Kinematics:\n{T}')
```

```
Forward Kinematics:
   0         0        -1         0.005
   0        -1         0         0
  -1         0         0         0.332
   0         0         0         1
```

Forward Kinematics:

1) `Specify the joint angles for each of the six joints of the robot in radians. These angles will be used for forward kinematics.` 2) `Compute the forward kinematics of the robot, calculating the pose (position and orientation) of the end-effector given the joint angles q.` 3) `Outputs the resulting transformation matrix T, which represents the end-effector pose.`

```
target_pose = SE3(0.4, 0.2, 0.5) * SE3.RPY([pi/2, 0, pi/2])

solution = irb140.ikine_LM(target_pose)
print(f'Inverse Kinematics:\nJoint Angles: {solution.q}')
```

```
Inverse Kinematics:
Joint Angles: [ 0.53822157  0.69152005  2.5631033   2.60054922 -
1.47380021  1.62891947]
```

Inverse Kinematics:

1) `Create a target end-effector pose using the SE3 class.` 2) `Computes the inverse kinematics using the Levenberg-Marquardt method to find the joint angles that achieve the specified target pose.` 3) `Outputs the resulting joint angles that place the end-effector at the target pose.`