Forward and Inverse Kinematics

Trajectories

Robotic Applications

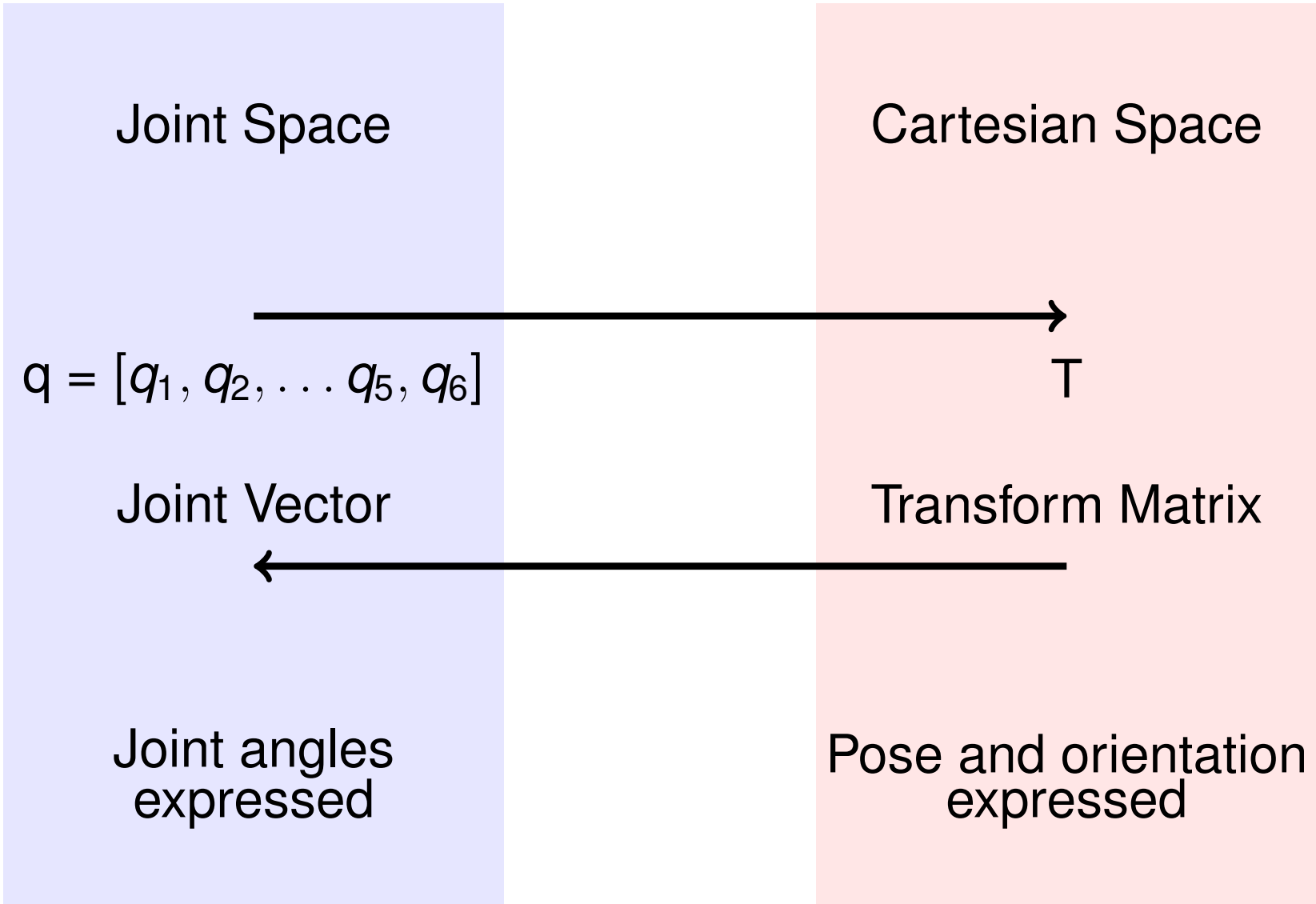
Steven Dirven, Massey University, 2015

Reflection on Forward and Inverse Kinematics

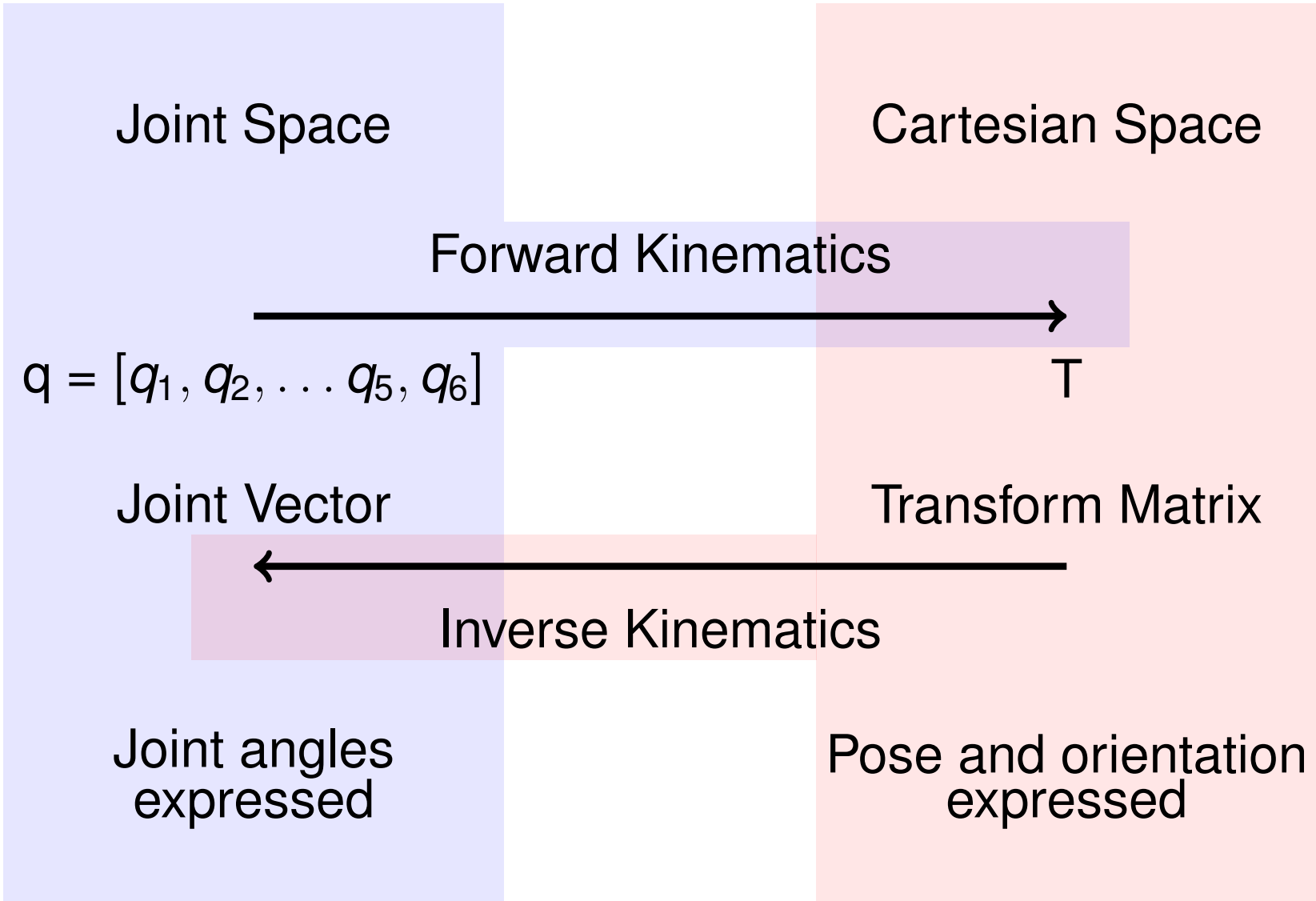
So far, the following concepts have been discussed:

1. Assignment of Denavit-Hartenberg frames, and recording of the parameters $\theta_j, d_j, a_j, \alpha_j$ into a table for each joint
2. Forward Kinematics is the process of finding a robot's pose (location and orientation in space) based on the specified configuration of each joint
3. Inverse Kinematics is the process of finding the joint configurations (angles for revolute, length for prismatic) based a robot's specified pose

The Relationship between Joint and Cartesian Spaces



The Relationship between Forward and Inverse Kinematics



Trajectory Generation : Techniques

Trajectory : A path that can be followed to get from one location to another.

Suppose that two poses in Cartesian space are known, and the robot needs to move from one to the other

Joint angles can be calculated by inverse kinematics for the two poses. How can the robot move between these points?

Trajectory Generation : Techniques

Trajectory : A path that can be followed to get from one location to another.

Suppose that two poses in Cartesian space are known, and the robot needs to move from one to the other

Joint angles can be calculated by inverse kinematics for the two poses. How can the robot move between these points?

- Joint space trajectories eg. specified rate of rotation
- Cartesian trajectories eg. straight-line motion in 3D

Trajectory Generation : PUMA Example

Define two poses that the robot can be moved between. Will do this in 2D such that the relationships between the paths can be observed.

Defining 2 poses for the PUMA 560 robot

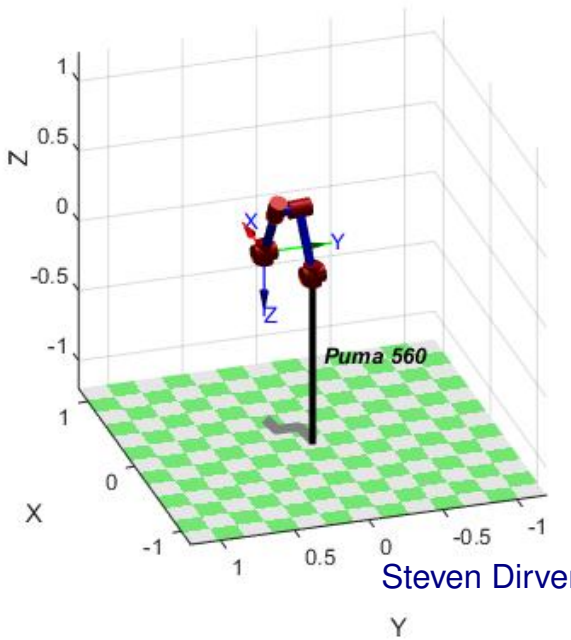
Pose 1: Wrist at (0.4,0.2,0) rotated about x-axis by *pi*
Pose 2: Wrist at (0.4,-0.2,0) rotated about x-axis by $pi/2$

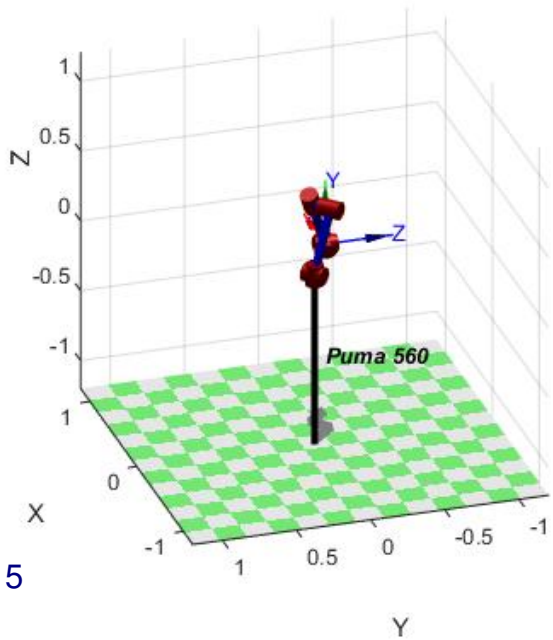
These can be specified in MATLAB, where the initial joint angles can be calculated by the closed form inverse kinematics method

Trajectory Generation : MATLAB PCT PUMA Example

Define the 2 poses in MATLAB PCT

```
mdl_puma560
T1 = transl(0.4,0.2,0)*trotx(pi)
T2 = transl(0.4,-0.2,0)*trotx(pi/2)
q1 = p560.ikine6s(T1)
q2 = p560.ikine6s(T2)
```





Trajectory Generation : Joint Space Technique

What is known?

- Starting and finishing pose.
- By the inverse kinematics method, the initial and final joint angles can be calculated

How to make a path between these points in the **joint space**?

- Decide on a number of time points to investigate
- Decide on a joint space method for moving between these points. Eg. Smooth Acceleration and Deceleration.

Trajectory Generation : Joint Space Technique

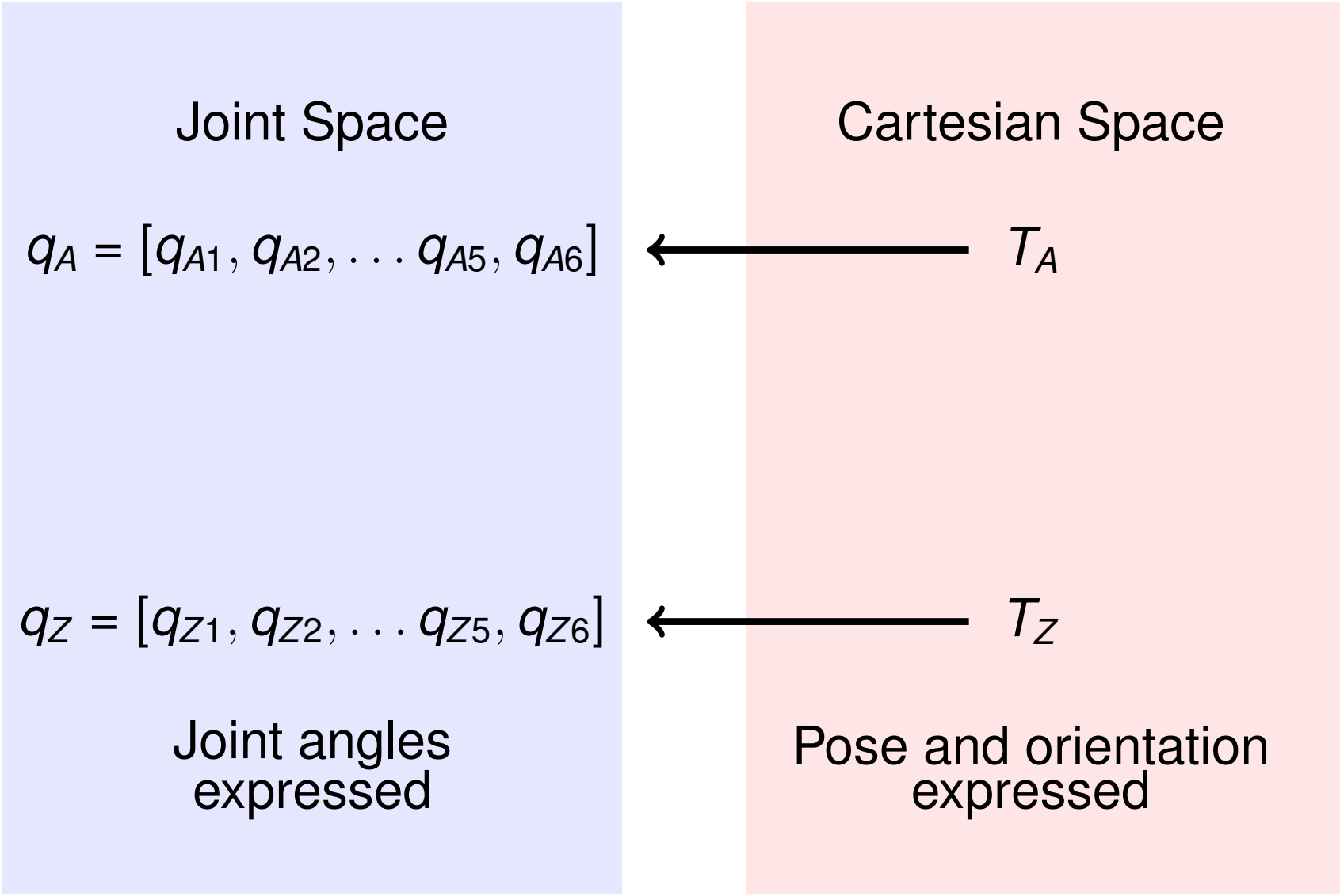
The Joint space method consists of the following steps:

Joint Space Method

1. Specify the initial and final pose
2. Find the joint angles at these points by inverse kinematics
3. Specify the number of time steps for which the trajectory should be calculated
4. Generate a trajectory in the joint space that smoothly moves from the initial to final pose
5. Find the robot pose by using forward kinematics on each set of joint angles in the trajectory

Trajectory Generation : Joint Space Technique

Step 1 & 2 : Specify poses. Inv. kine. to **joint space**.



Trajectory Generation : Joint Space Technique

Step 3 & 4 : Interpolate in **joint space**

Joint Space

$q_A = [q_{A1}, q_{A2}, \ldots q_{A5}, q_{A6}]$

Interpolate between q_A and q_Z

$q_Z = [q_{Z1}, q_{Z2}, \ldots q_{Z5}, q_{Z6}]$

Joint angles expressed

Cartesian Space

T_A

T_Z

Pose and orientation expressed

Trajectory Generation : Joint Space Technique

Step 3 & 4 : Interpolate in **joint space**

Joint Space

$$q_A = [q_{A1}, q_{A2}, \ldots q_{A5}, q_{A6}]$$
$$q_B = [q_{B1}, q_{B2}, \ldots q_{B5}, q_{B6}]$$
$$q_C = [q_{C1}, q_{C2}, \ldots q_{C5}, q_{C6}]$$

etc.

$$q_Y = [q_{Y1}, q_{Y2}, \ldots q_{Y5}, q_{Y6}]$$
$$q_Z = [q_{Z1}, q_{Z2}, \ldots q_{Z5}, q_{Z6}]$$

Joint angles expressed

Cartesian Space

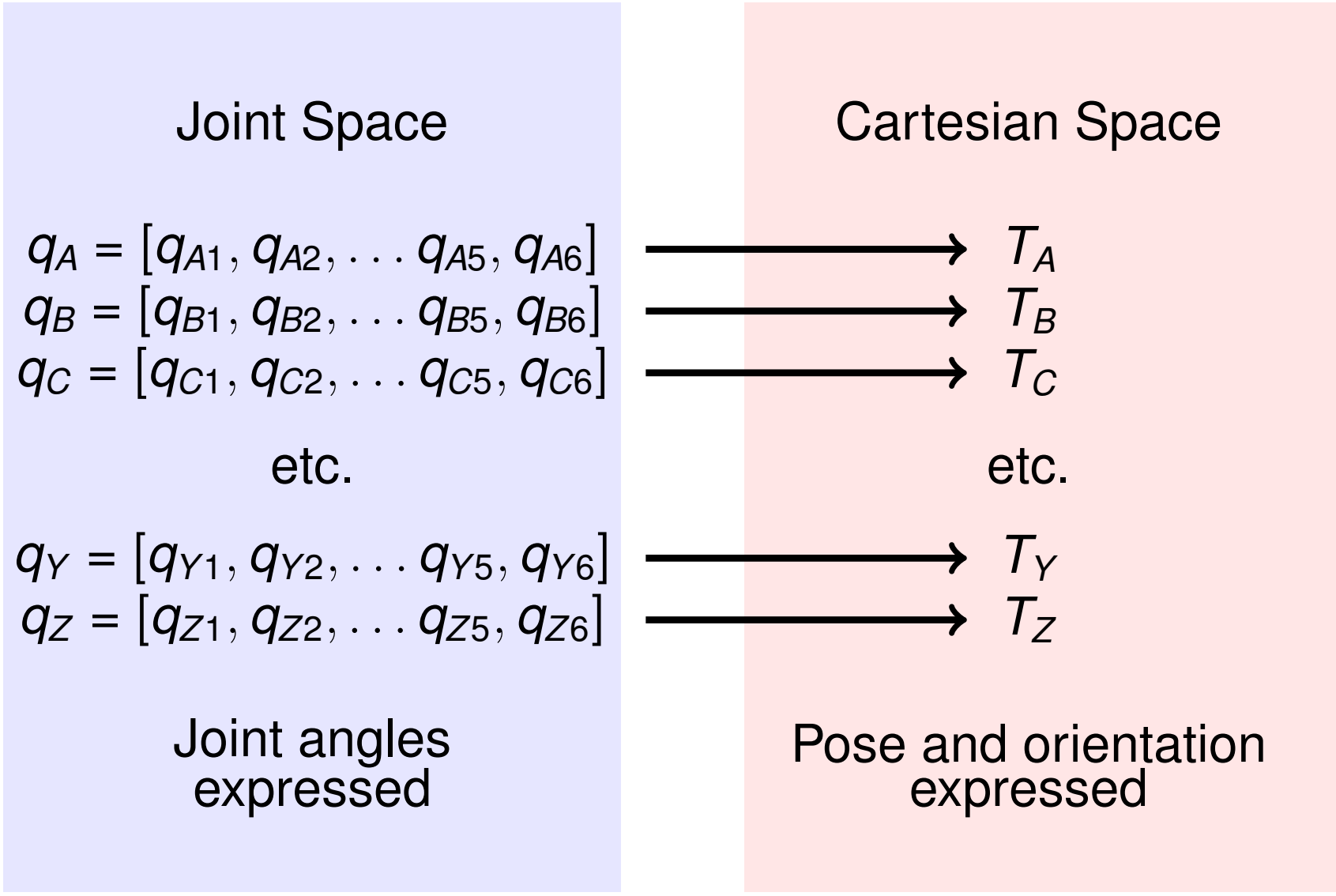
$$T_A$$

$$T_Z$$

Pose and orientation expressed

Trajectory Generation : Joint Space Technique

Step 5 : Forward kinematics for **cartesian space**



Trajectory Generation : Joint Space MATLAB PCT PUMA Example

Joint Space MATLAB PCT PUMA Example

Suppose that the robot should move through the motion between the following poses in 2 seconds, with an update period of 50ms.
Pose 1: Wrist at (0.4,0.2,0) rotated about x-axis by *pi*
Pose 2: Wrist at (0.4,-0.2,0) rotated about x-axis by $pi/2$

Need to create a time vector, and find the poses at each instance.

Joint Space MATLAB PCT PUMA : Time Vector

```
t = [0:0.05:2]' % 2 Seconds, 50 ms period
% The ' causes the vector to be transposed
```

Trajectory Generation : Joint Space MATLAB PCT PUMA Example

Desire is to move between the poses smoothly in the joint space. Create the joint space trajectory.

Joint Space MATLAB PCT PUMA : Trajectory generation

```
Syntax:  traj = jtraj(pose1,pose2,time)

mdl_puma560
T1 = transl(0.4,0.2,0)*trotx(pi)
T2 = transl(0.4,-0.2,0)*trotx(pi/2)
t = [0:0.05:2]' % 2 Seconds, 50 ms period
q = p560.jtraj(T1,T2,t)
p560.plot(q)
```

Trajectory Generation : Joint Space MATLAB PCT PUMA Example

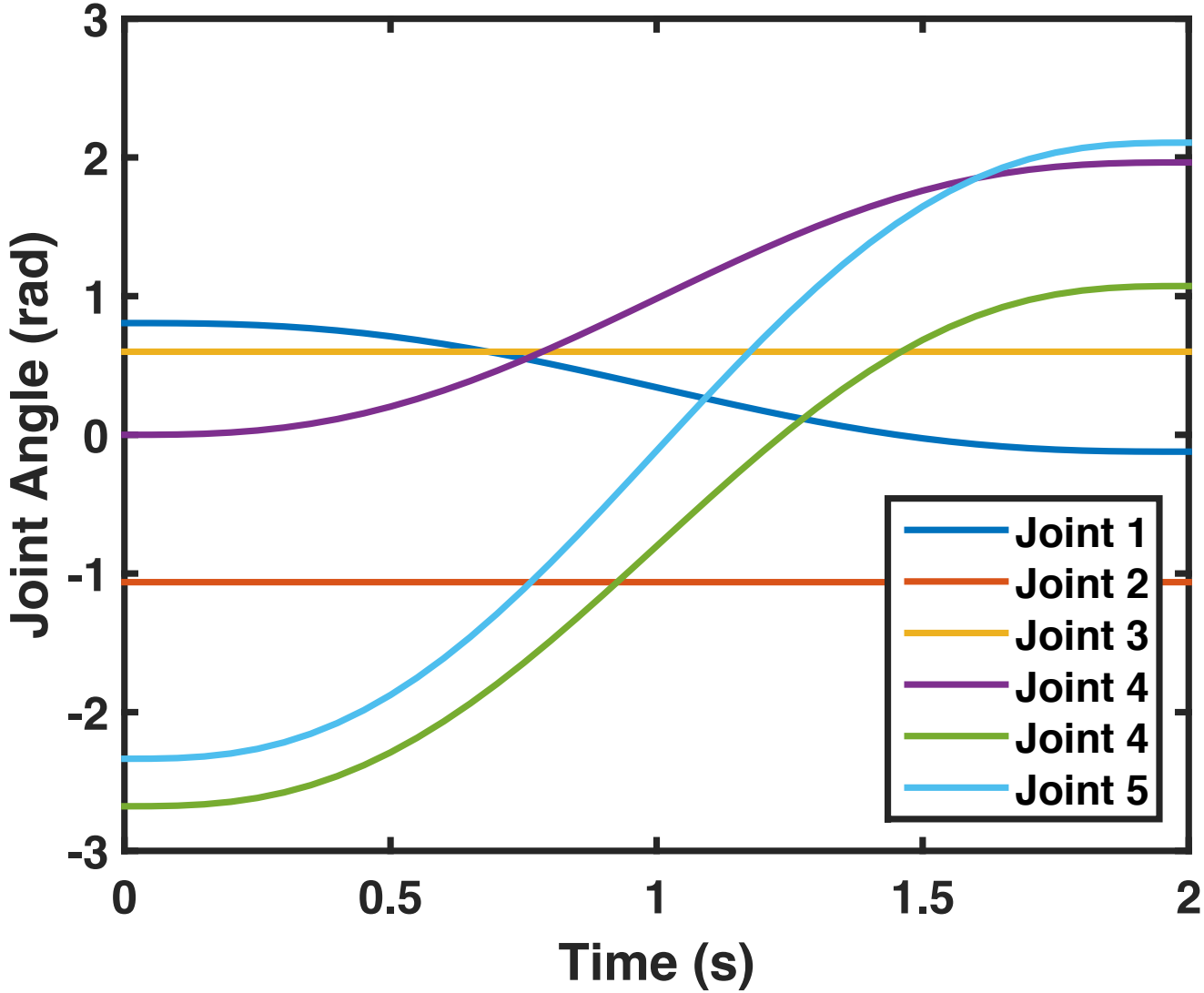
This results in 'q' containing 50 x poses consisting of 6 joint angles

```
q =

    0.8058   -1.0625    0.5992    0.0000   -2.6783   -2.3358
    0.8057   -1.0625    0.5992    0.0003   -2.6777   -2.3351
    0.8047   -1.0625    0.5992    0.0023   -2.6739   -2.3306
    0.8023   -1.0625    0.5992    0.0074   -2.6642   -2.3191
    0.7979   -1.0625    0.5992    0.0168   -2.6462   -2.2978
    0.7909   -1.0625    0.5992    0.0315   -2.6181   -2.2645
    0.7811   -1.0625    0.5992    0.0523   -2.5785   -2.2176
    0.7682   -1.0625    0.5992    0.0796   -2.5263   -2.1558
    0.7521   -1.0625    0.5992    0.1138   -2.4610   -2.0785
    0.7326   -1.0625    0.5992    0.1550   -2.3822   -1.9852
    0.7098   -1.0625    0.5992    0.2033   -2.2900   -1.8759
    0.6838   -1.0625    0.5992    0.2585   -2.1846   -1.7511
    0.6546   -1.0625    0.5992    0.3203   -2.0666   -1.6113
```

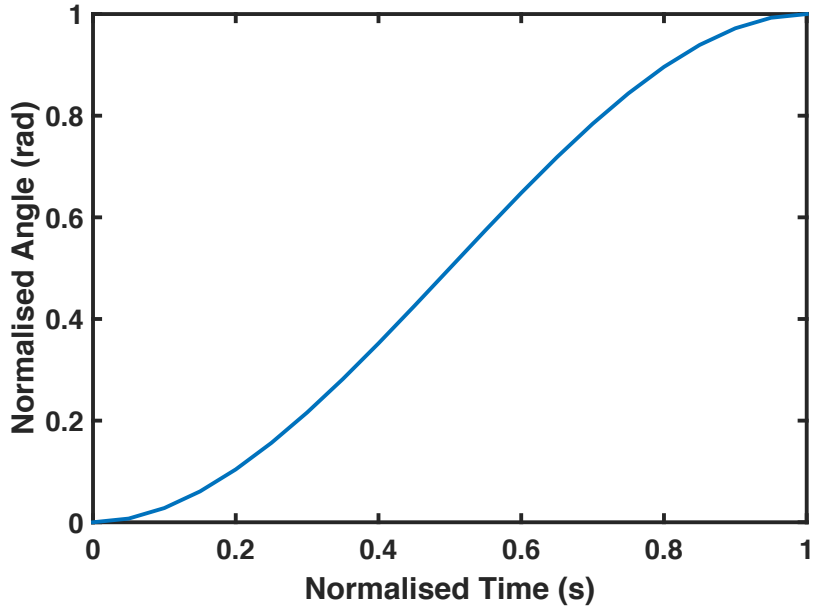
Etc . . .

Trajectory Generation : Joint Space MATLAB PCT PUMA Example



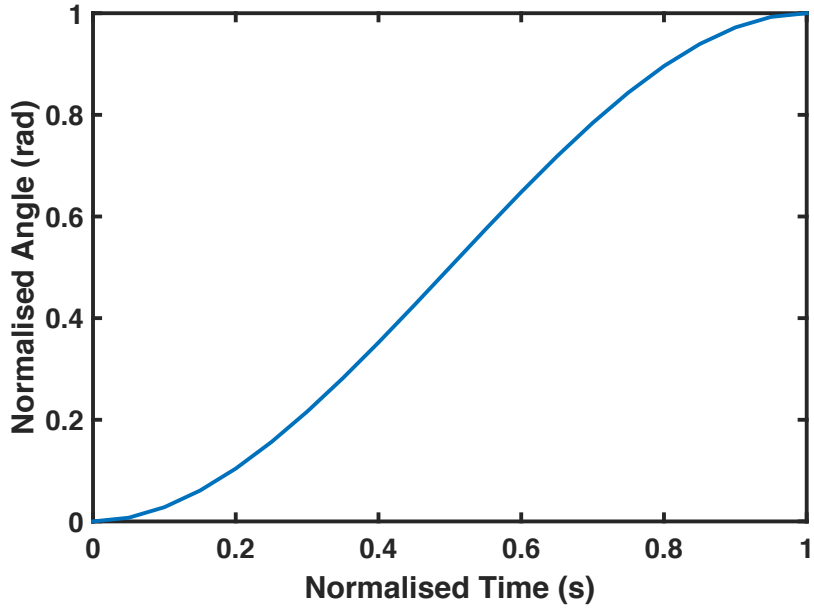
Trajectory Generation : Joint Space MATLAB PCT PUMA Example

Notice that all of the joint angle trajectories start and end with zero gradient. This means that there is no angular velocity at this point. Acceleration is a linear function.



A third order polynomial is one method of creating the interpolant points

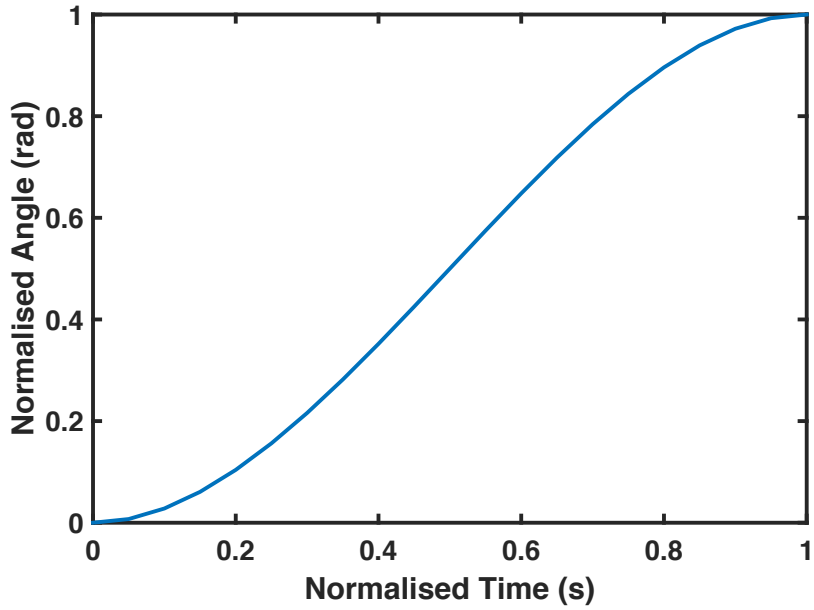
Trajectory Generation : Joint Space MATLAB PCT PUMA Example



$$\theta(t) = a_0 t^3 + a_1 t^2 + a_2 t + a_3 \quad \theta(0) = 0 \;;\; \theta(1) = 1$$

$$\frac{d\theta}{dt}(t) = 3a_0 t^2 + 2a_1 t + a_2 \qquad \frac{d\theta}{dt}(0) = \frac{d\theta}{dt}(1) = 0$$

Trajectory Generation : Joint Space MATLAB PCT PUMA Example



$$\theta(t) = a_0 t^3 + a_1 t^2 + a_2 t + a_3 \quad \theta(0) = 0 \;;\; \theta(1) = 1$$

$$\therefore a_3 = 0$$

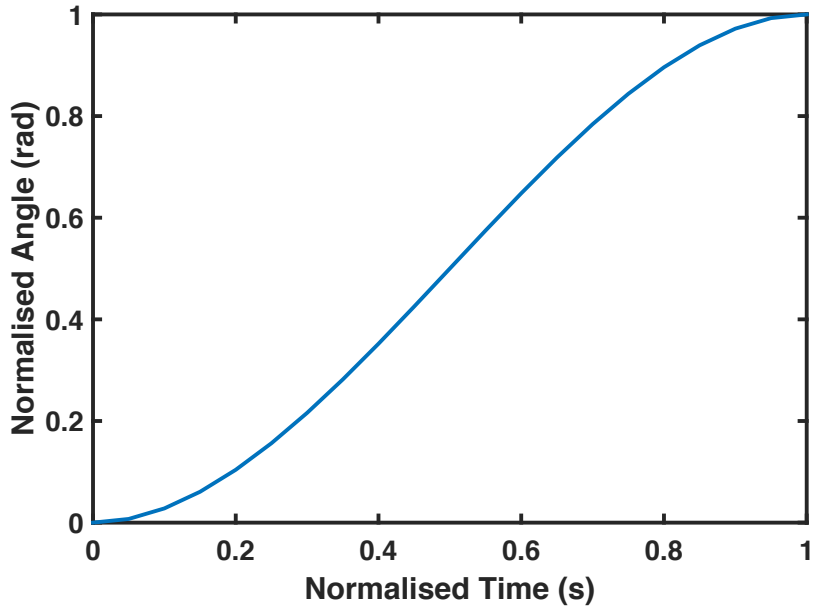
$$\frac{d\theta}{dt}(t) = 3a_0 t^2 + 2a_1 t + a_2 \quad \frac{d\theta}{dt}(0) = \frac{d\theta}{dt}(1) = 0$$

$$\therefore a_2 = 0$$

$$\therefore a_0 + a_1 = 1$$

$$\therefore 3a_0 + 2a_1 = 0$$

Trajectory Generation : Joint Space MATLAB PCT PUMA Example

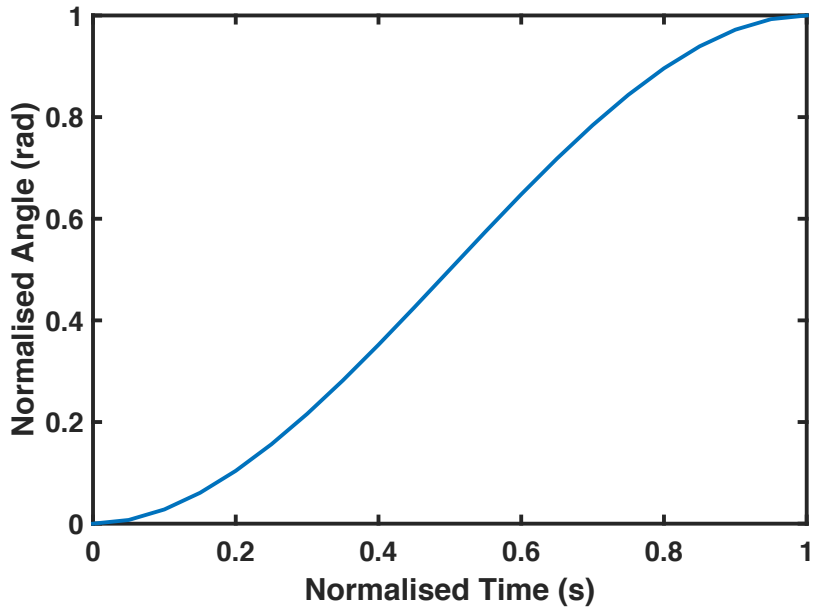


$$\theta(t) = -2t^3 + 3t^2$$

$$\frac{d\theta}{dt}(t) = -6t^2 + 6t$$

How can this technique be generalised to accomodate different angular offsets and displacements or time periods?

Trajectory Generation : Joint Space MATLAB PCT PUMA Example



$$\theta(t) = \theta_{displacement}\left[-2\left(\frac{t}{t_{period}}\right)^3 + 3\left(\frac{t}{t_{period}}\right)^2\right] + \theta_{offset}$$

Trajectory Generation : Joint Space MATLAB PCT PUMA Example

From these joint configurations it is interesting to evaluate the forward kinematics to find out the location and orientation of the end effector.

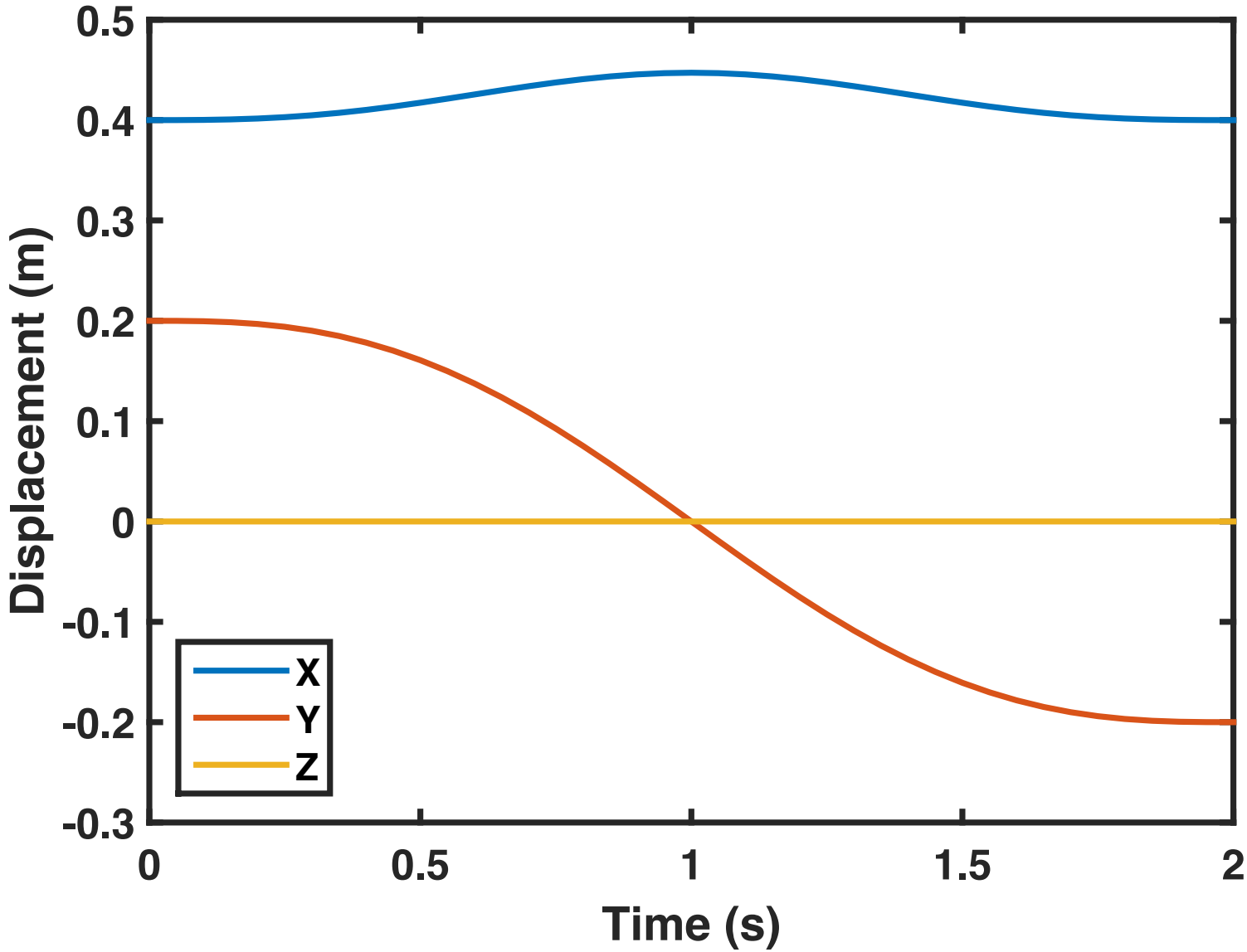
Find the translational component:

Joint Space MATLAB PCT PUMA : Forward kinematics to find translation component of cartesian location

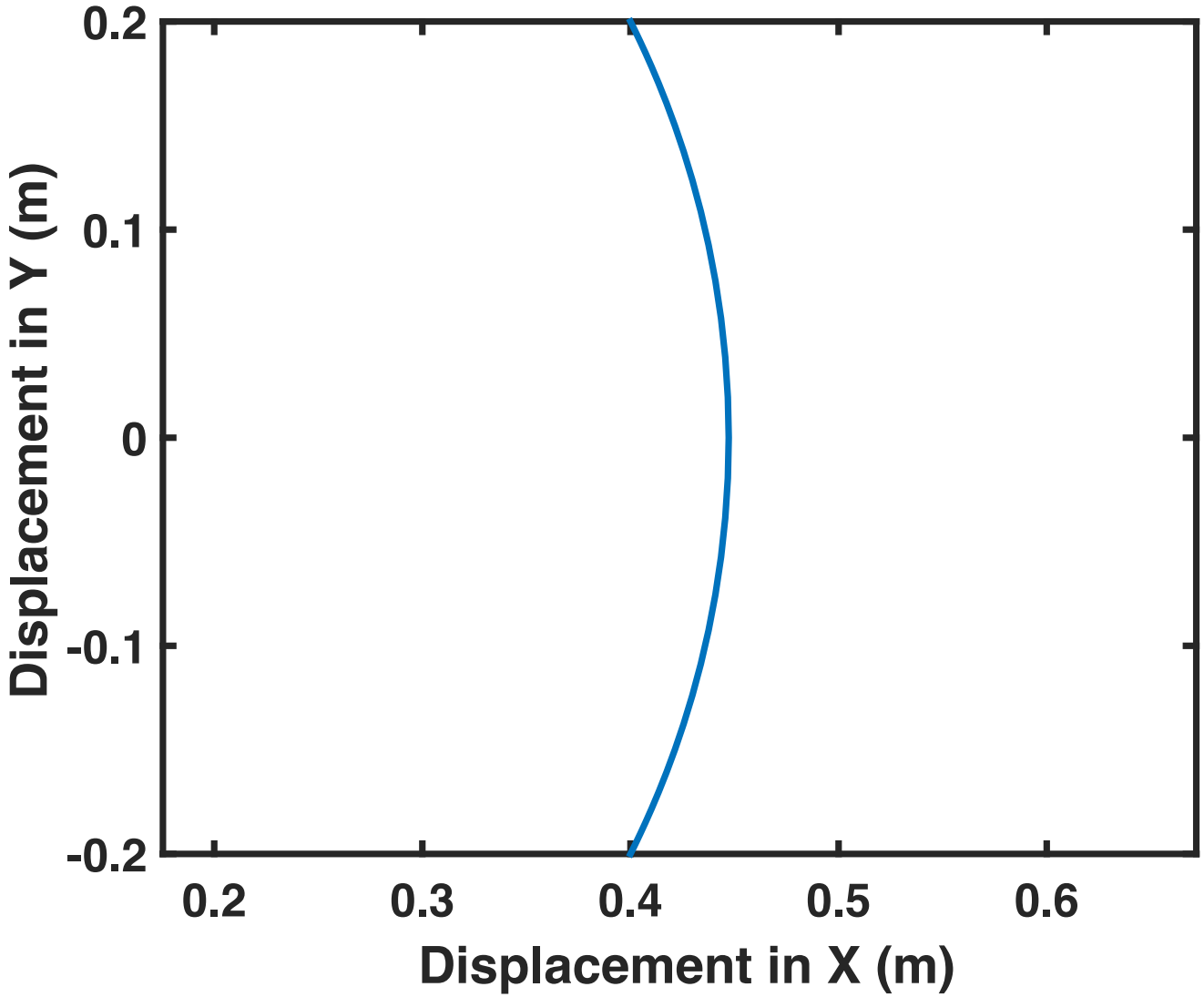
```
Syntax:  Transf = p560.fkine(joint angles)
Syntax:  Transl = transl(Transf)

T = p560.fkine(q)
p = transl(T)
plot(p(:,1),p(:,2)) % Plot X vs Y
```

Trajectory Generation : Joint Space MATLAB PCT PUMA Example



Trajectory Generation : Joint Space MATLAB PCT PUMA Example



Trajectory Generation : Joint Space MATLAB PCT PUMA Example

What about the orientation of the end effector as it goes through this process?

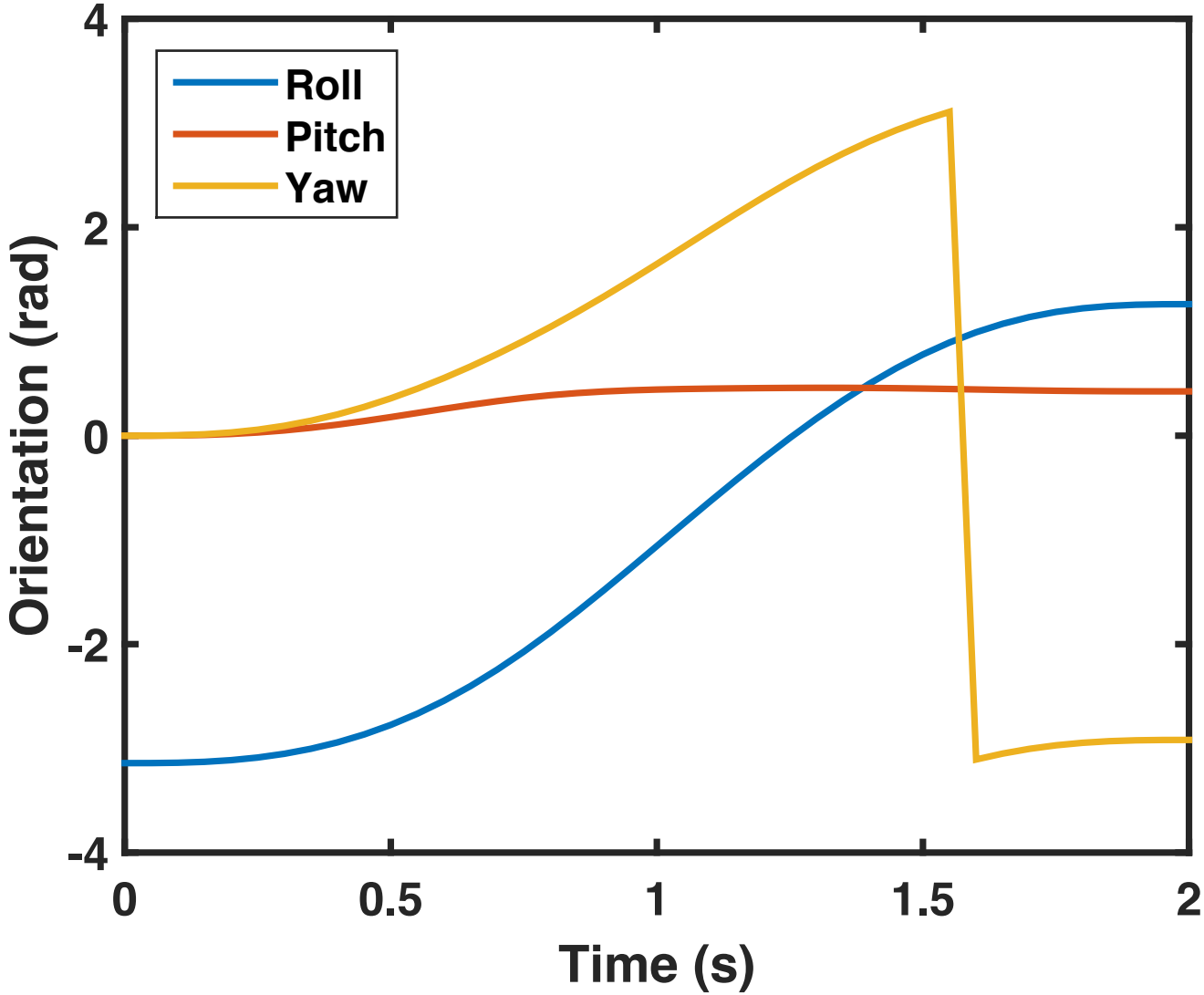
Find the orientation component:

Joint Space MATLAB PCT PUMA : Forward kinematics to find orientation component of cartesian location in Roll Pitch and Yaw

```
Syntax:  plot(time,tr2rpy(Transforms))

plot(t,tr2rpy(T))
```

Trajectory Generation : Joint Space MATLAB PCT PUMA Example



Trajectory Generation : Joint Space Summary

The **joint space** trajectory generation technique results in:

- A smooth propagation in joint angles with smooth acceleration and deceleration
- Mathematically described joint angles with respect to time
- Pose, in terms of location and orientation are dependant on the joint angles (uncontrollable). The result is calculated by forward kinematics

Advantage: Smooth accelerations of links
Disadvantage: Location and Orientation along the trajectory are dependant

Trajectory Generation : Cartesian Space Technique

The Cartesian space method consists of the following steps:

Cartesian Space Method

1. Specify the initial and final pose
2. Specify the number of time steps for which the trajectory should be calculated
3. Generate a trajectory in the cartesian space that smoothly moves from the initial to final pose (eg. straight line)
4. Find the joint angles by using inverse kinematics on each set of joint angles in the trajectory

Trajectory Generation : Joint Space Technique

Step 1 : Specify poses. Already in **cartesian space**.

Joint Space

Cartesian Space

T_A

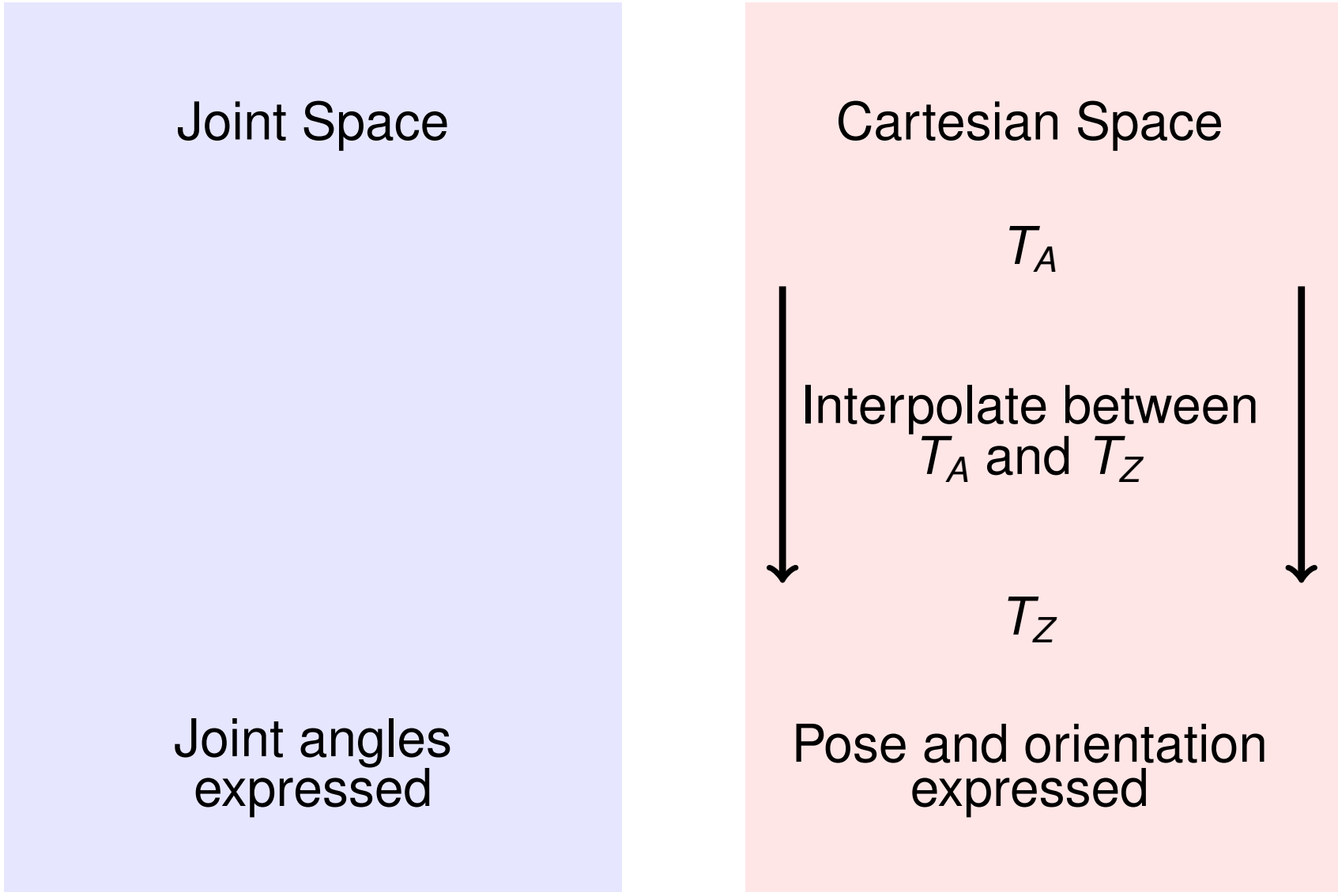
T_Z

Joint angles expressed

Pose and orientation expressed

Trajectory Generation : Joint Space Technique

Step 3 & 4 : Interpolate in **cartesian space**



Trajectory Generation : Joint Space Technique

Step 3 & 4 : Interpolate in **cartesian space**

Joint Space

Joint angles expressed

Cartesian Space

T_A
T_B
T_C

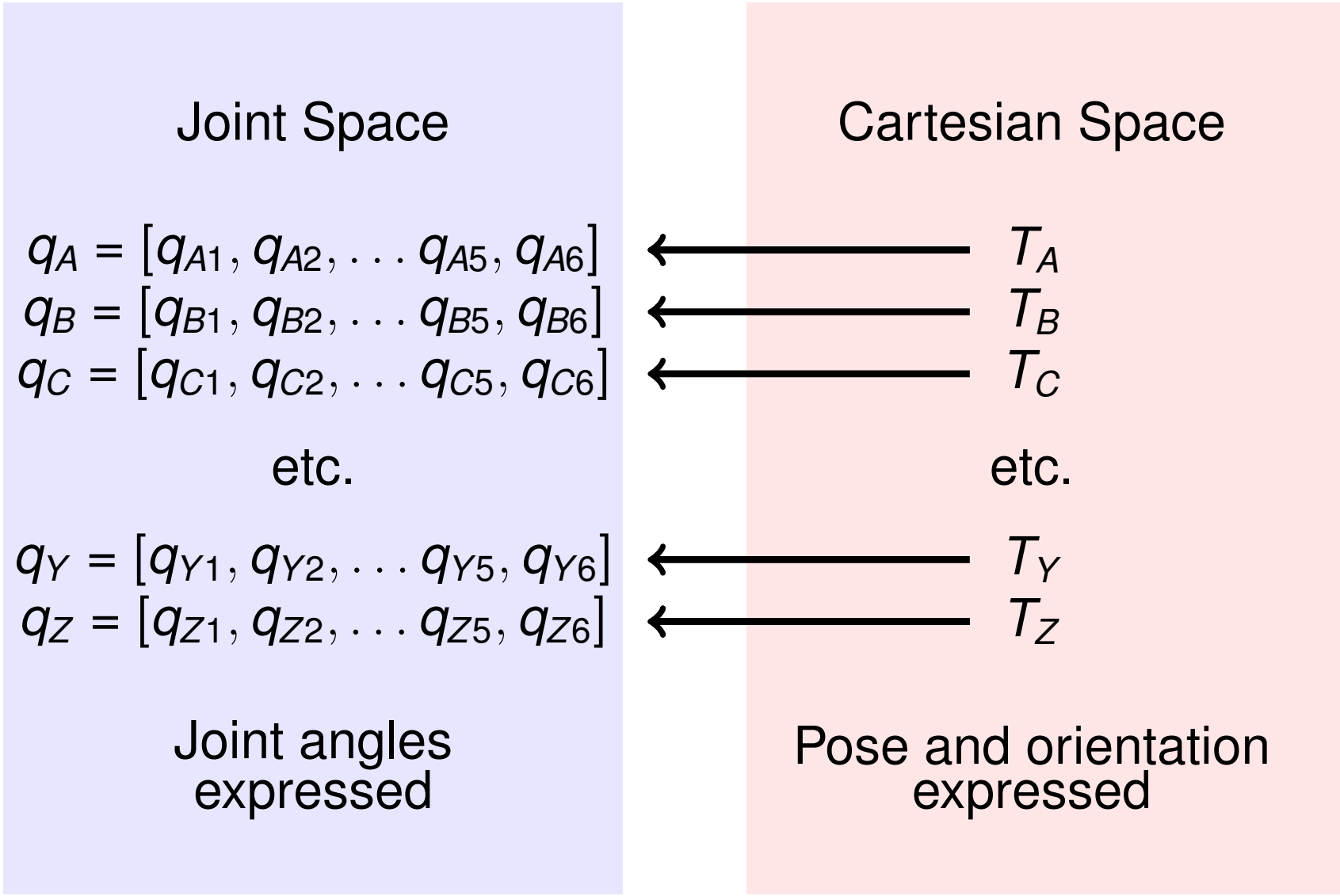
etc.

T_Y
T_Z

Pose and orientation expressed

Trajectory Generation : Joint Space Technique

Step 5 : Inverse kinematics for **joint space**



Trajectory Generation : Cartesian Space MATLAB PCT PUMA Example

Joint Space MATLAB PCT PUMA Example

Suppose that the robot should move through the motion between the following poses in 2 seconds, with an update period of 50ms.
Pose 1: Wrist at (0.4,0.2,0) rotated about x-axis by *pi*
Pose 2: Wrist at (0.4,-0.2,0) rotated about x-axis by $pi/2$

Need to create a time vector, and find the poses at each instance.

Cartesian Space MATLAB PCT PUMA : Time Vector

```
t = [0:0.05:2]' % 2 Seconds, 50 ms period
% The ' causes the vector to be transposed
```

Trajectory Generation : Cartesian Space MATLAB PCT PUMA Example

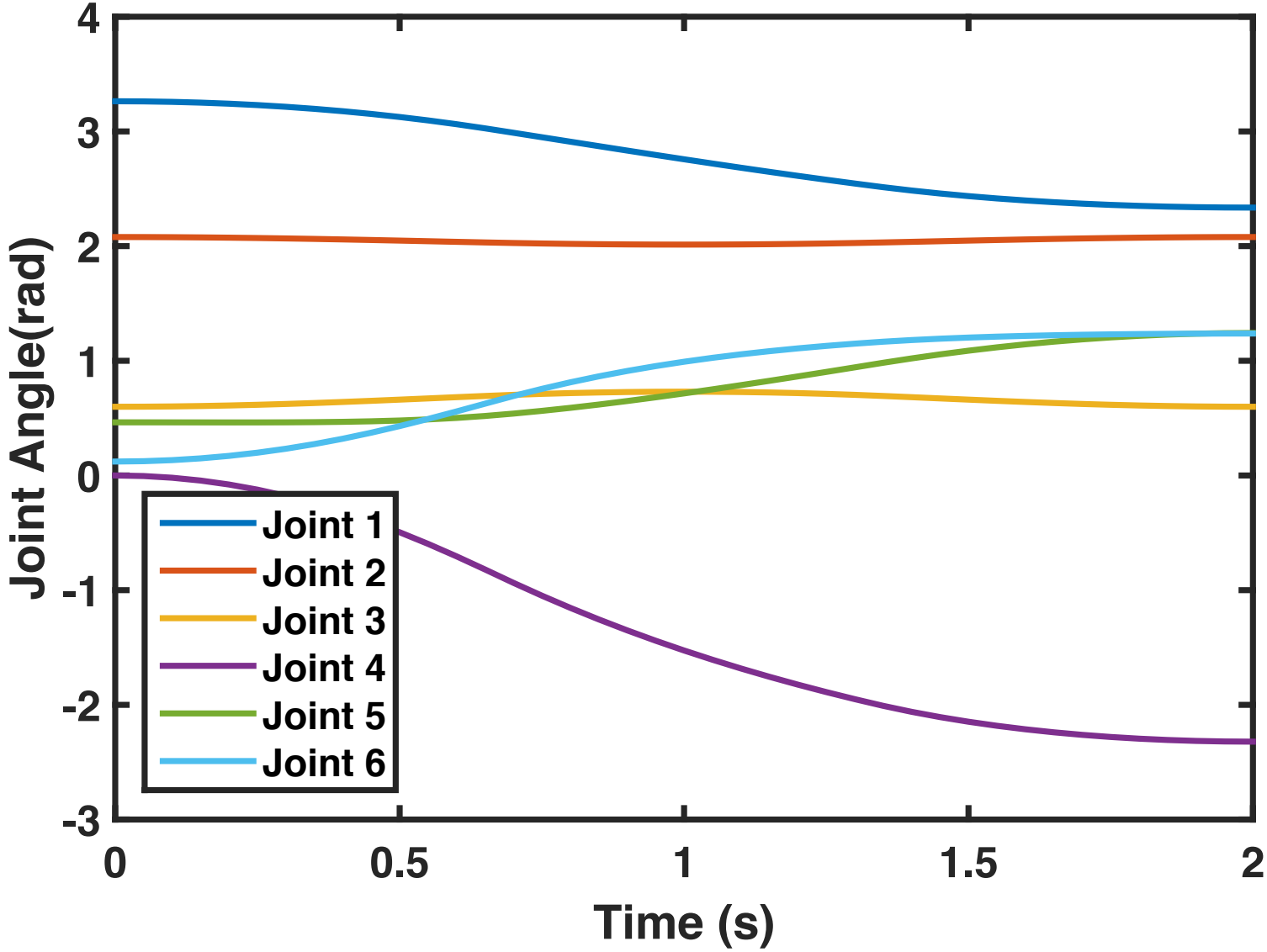
Desire is to move between the poses smoothly in the cartesian space. Create the cartesian space trajectory. (Observe that it is ctraj, not jtraj as before)

Cartesian Space MATLAB PCT PUMA : Trajectory generation

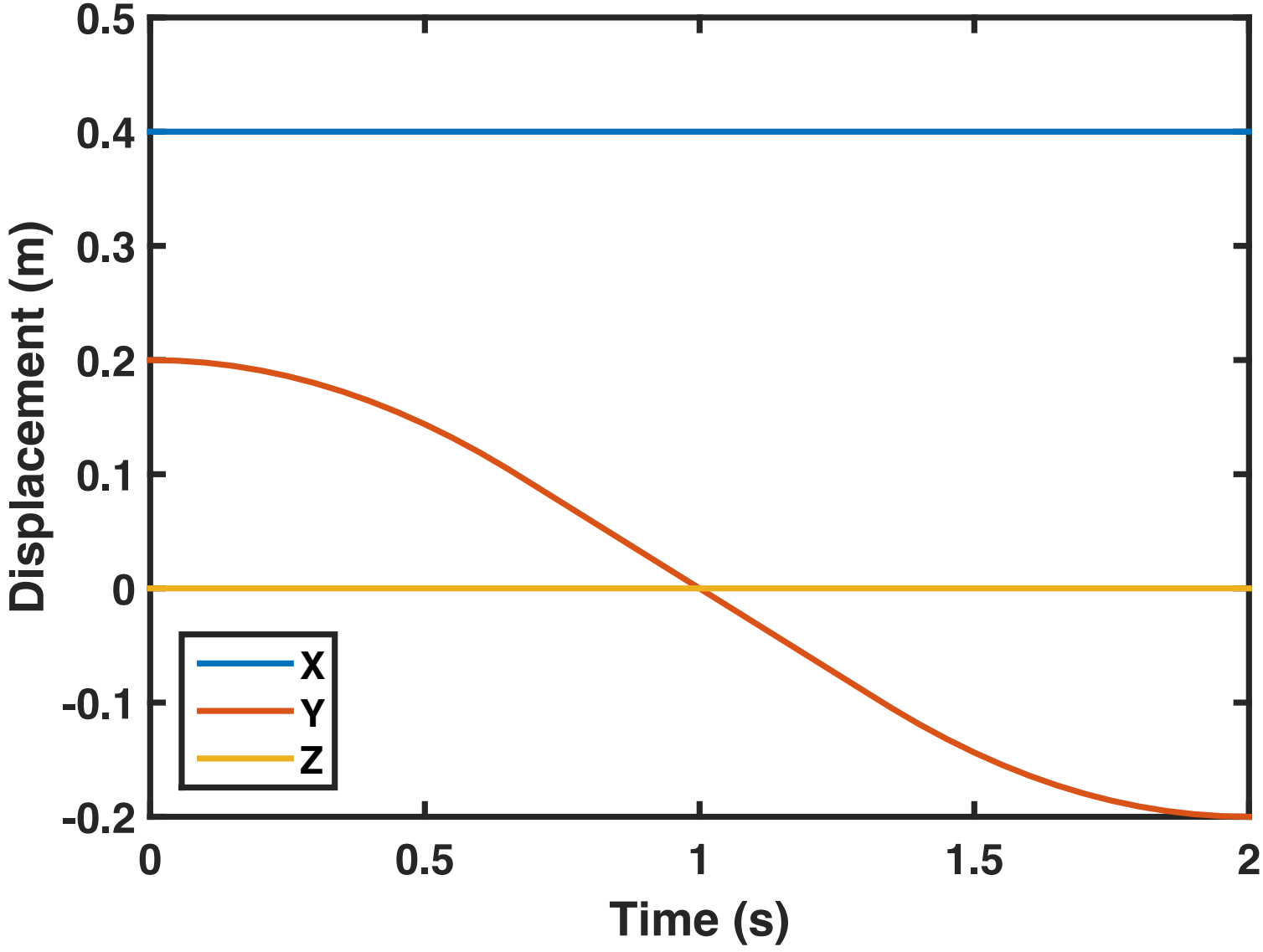
```
Syntax:  traj = ctraj(pose1,pose2,#steps)


mdl_puma560
T1 = transl(0.4,0.2,0)*trotx(pi)
T2 = transl(0.4,-0.2,0)*trotx(pi/2)
t = [0:0.05:2]' % 2 Seconds, 50 ms period
T = ctraj(T1,T2,41)
q = p560.ikine6s(T)
p560.plot(q)
```

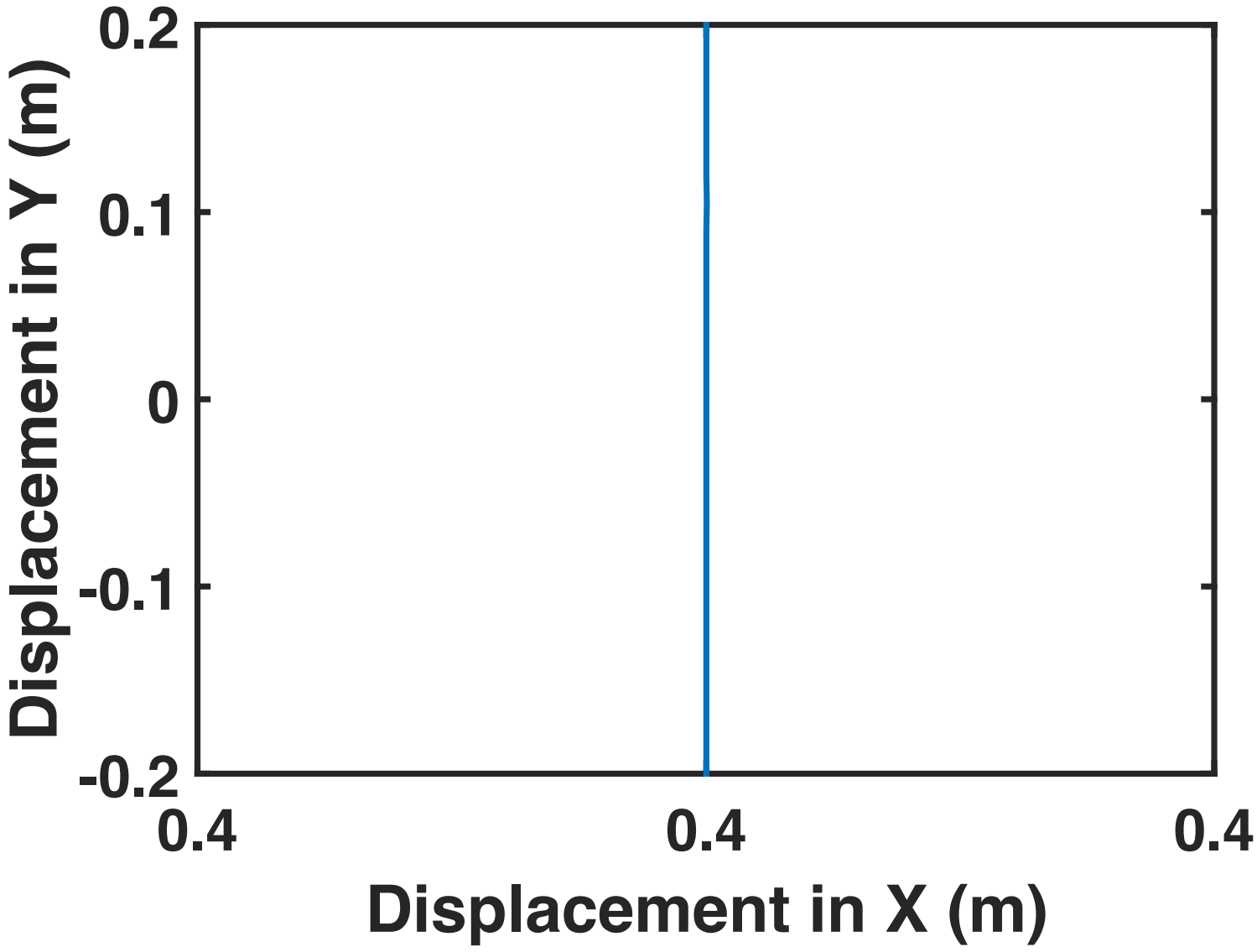
Trajectory Generation : Cartesian Space MATLAB PCT PUMA Example



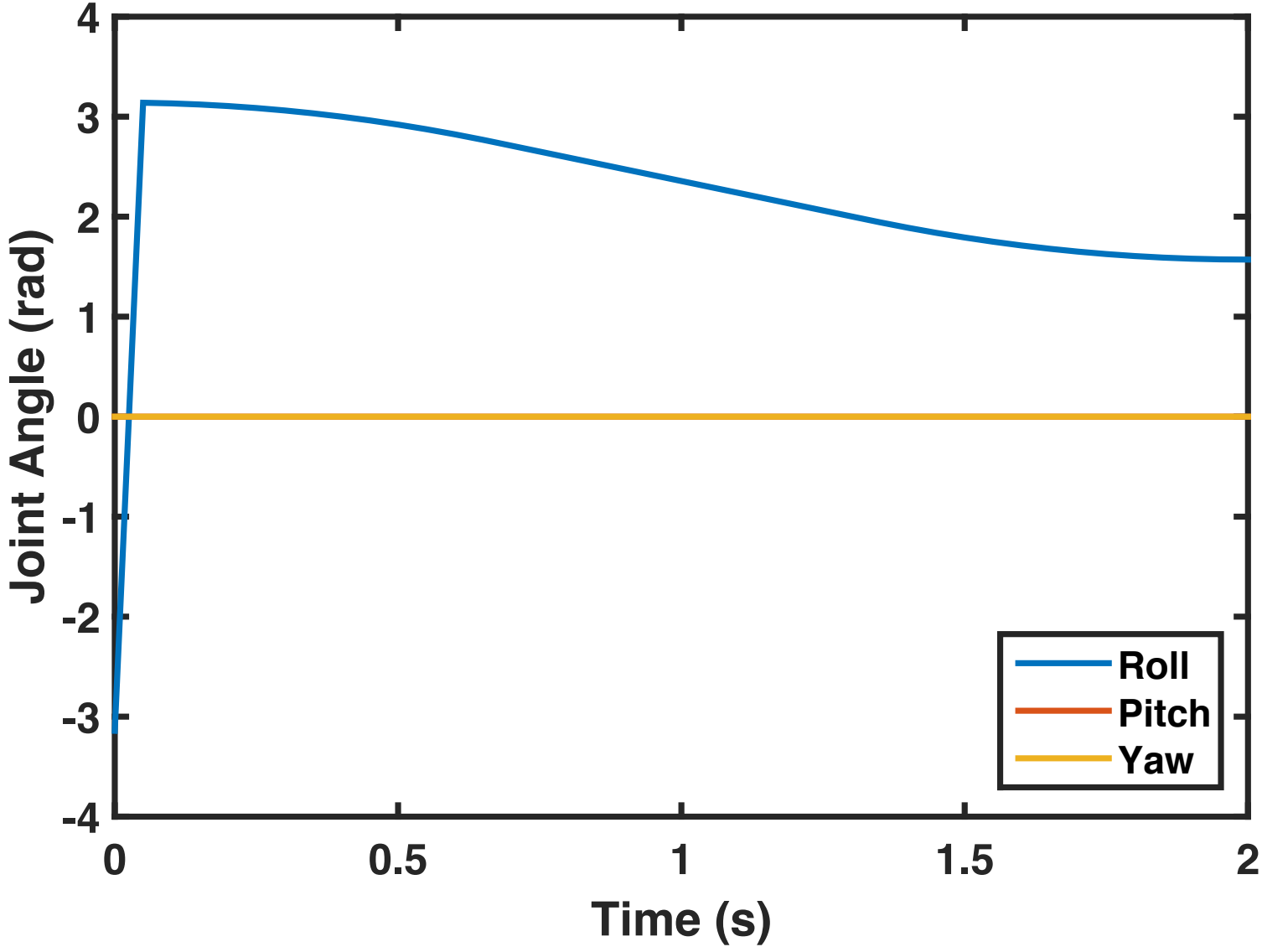
Trajectory Generation : Cartesian Space MATLAB PCT PUMA Example



Trajectory Generation : Cartesian Space MATLAB PCT PUMA Example



Trajectory Generation : Cartesian Space MATLAB PCT PUMA Example



Trajectory Generation : Cartesian Space Summary

The **cartesian space** trajectory generation technique results in:

- A smooth propagation along a cartesian path with smooth acceleration and deceleration
- Mathematically described positoin and orientation angles with respect to time
- Joint angles are dependant (uncontrollable). The joint angles are found by inverse kinematics

Advantage: Good position and pose control
Disadvantage: Can require large torques from rotational joints

Introduction to Singularities

A singularity in robotics occurs when one or more joints no longer represent independent controlling variables (eg. they become aligned, which eliminates a degree of freedom)

What does this mean for evaluating kinematics?

Introduction to Singularities

A singularity in robotics occurs when one or more joints no longer represent independent controlling variables (eg. they become aligned, which eliminates a degree of freedom)

What does this mean for evaluating kinematics?

- The relationship between the output of two joints act within the same plane. Their summed output results in only a single degree of freedom
- Moving through (or close to) a singularity in cartesian motion can result in excessive joint motion.

Trajectory Generation : Motion Through a Singularity MATLAB PCT PUMA Example

Motion near a singularity can be modelled by moving the PUMA560 robot through the following motion

Cartesian Motion Near a Singularity Example

Suppose that the robot should move through the motion between the following poses in 2 seconds, with an update period of 50ms.

Pose 1: Wrist at (0.5,0.3,0.44) rotated about y by $pi/2$
Pose 2: Wrist at (0.5,-0.3,0.44) rotated about y by $pi/2$

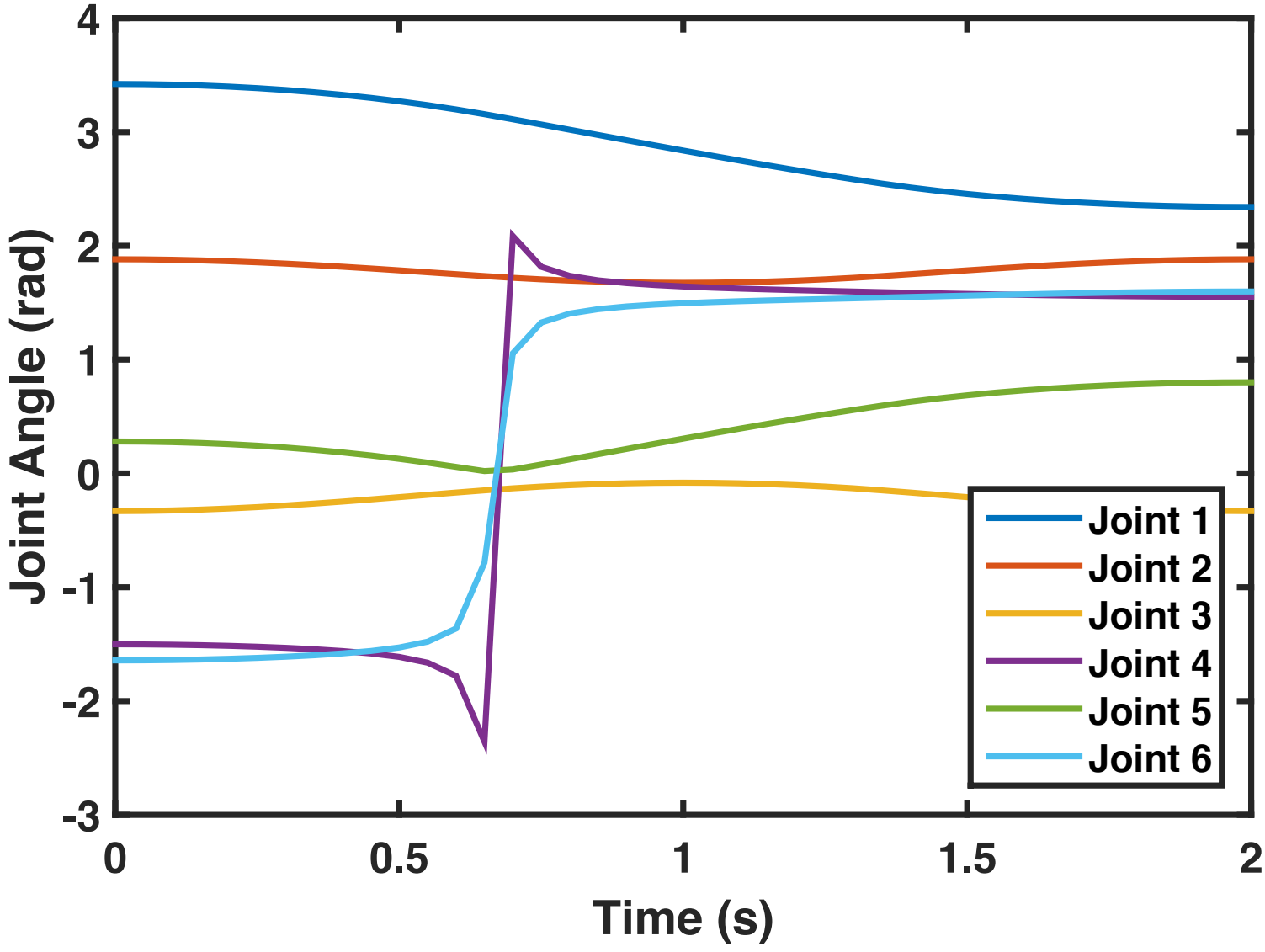
Moving in the direction Y, with constant X and Z, with the end effector z-axis pointing in the global Y direction.

Trajectory Generation : Motion Through a Singularity
MATLAB PCT PUMA Example (Cartesian Trajectory)

MATLAB PCT Cartesian Motion Near a Singularity Example

```
mdl_puma560
T1 = transl(0.5,0.3,0.44)*troty(pi/2)
T2 = transl(0.5,-0.3,0.44)*troty(pi/2)
t = [0:0.05:2]' % 2 Seconds, 50 ms period
T = ctraj(T1,T2,length(t))
q = p560.ikine6s(T)
plot(t,q)
p560.plot(q)
```

Trajectory Generation : Motion Through a Singularity
MATLAB PCT PUMA Example (Cartesian Trajectory)

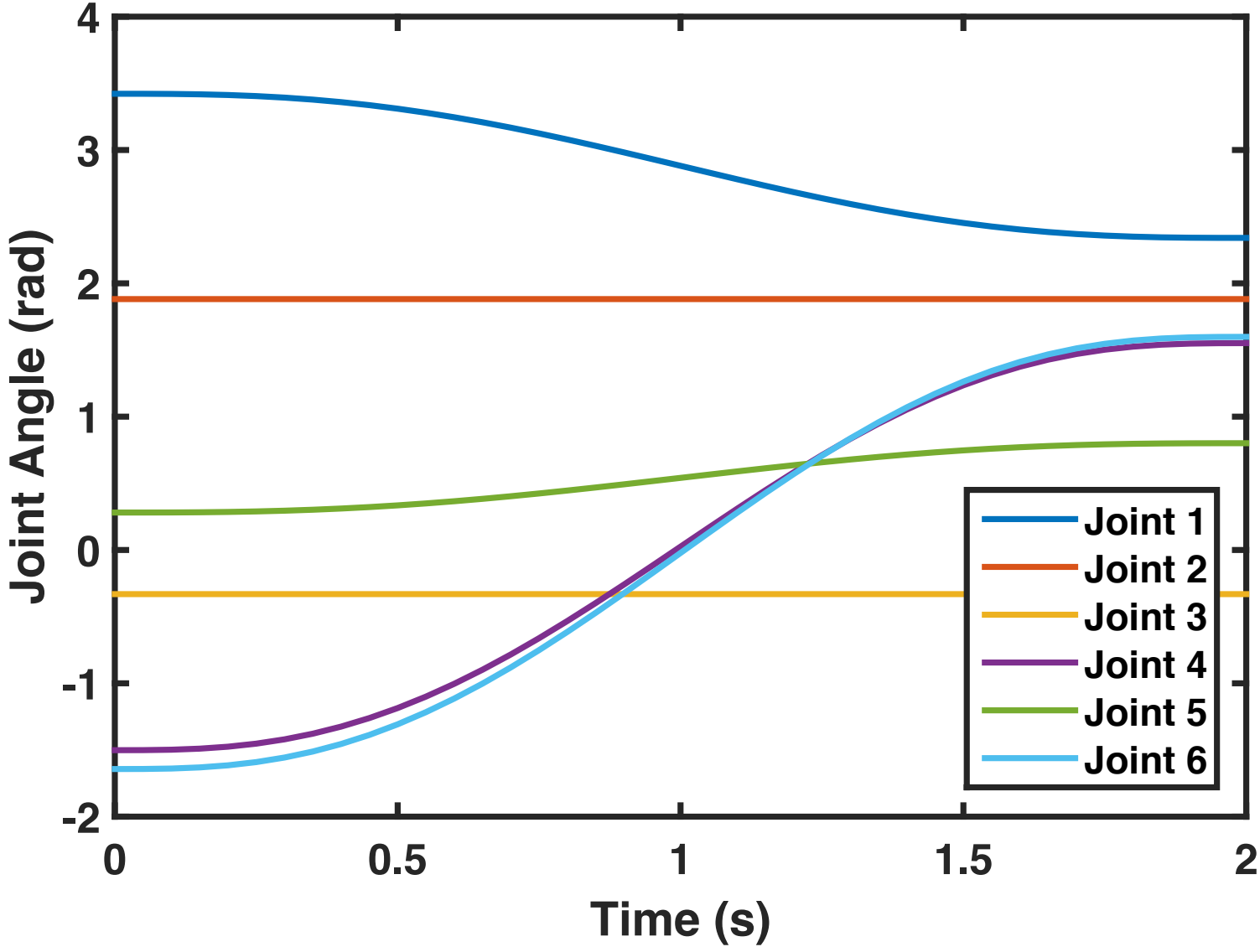


Trajectory Generation : Motion Through a Singularity
MATLAB PCT PUMA Example (Joint Space Trajectory)

MATLAB PCT Joint Motion Near a Singularity Example

```
mdl_puma560
T1 = transl(0.5,0.3,0.44)*troty(pi/2)
T2 = transl(0.5,-0.3,0.44)*troty(pi/2)
q1 = p560.ikine6s(T1)
q2 = p560.ikine6s(T2)
t = [0:0.05:2]' % 2 Seconds, 50 ms period
q = mtraj(@tpoly,q1,q2,t)
plot(t,q)
p560.plot(q)
```

Trajectory Generation : Motion Through a Singularity
MATLAB PCT PUMA Example (Joint Space Trajectory)



Trajectory Generation : Motion Through a Singularity Summary

Cartesian space trajectories near singularities result in very high acceleration demand of each robotic joint.

However, **joint space** trajectory generation is immune to this effect as the trajectories are based on the joint angle changing smoothly.

Remember, the **joint space** trajectory generation does not allow the end effector orientation to be controlled along the path, only at the two ends.

Introduction to Configuration Change

Configurations have been alluded to, the wrist pose, left or right handed pose, and up or down elbow pose, but how can the robot move between them?

What is known, what trajectory generation tools are available and how can they be used?

- Can a Joint Space trajectory be used?

- Can a Cartesian space trajectory be used?

Introduction to Configuration Change

Configurations have been alluded to, the wrist pose, left or right handed pose, and up or down elbow pose, but how can the robot move between them?

What is known, what trajectory generation tools are available and how can they be used?

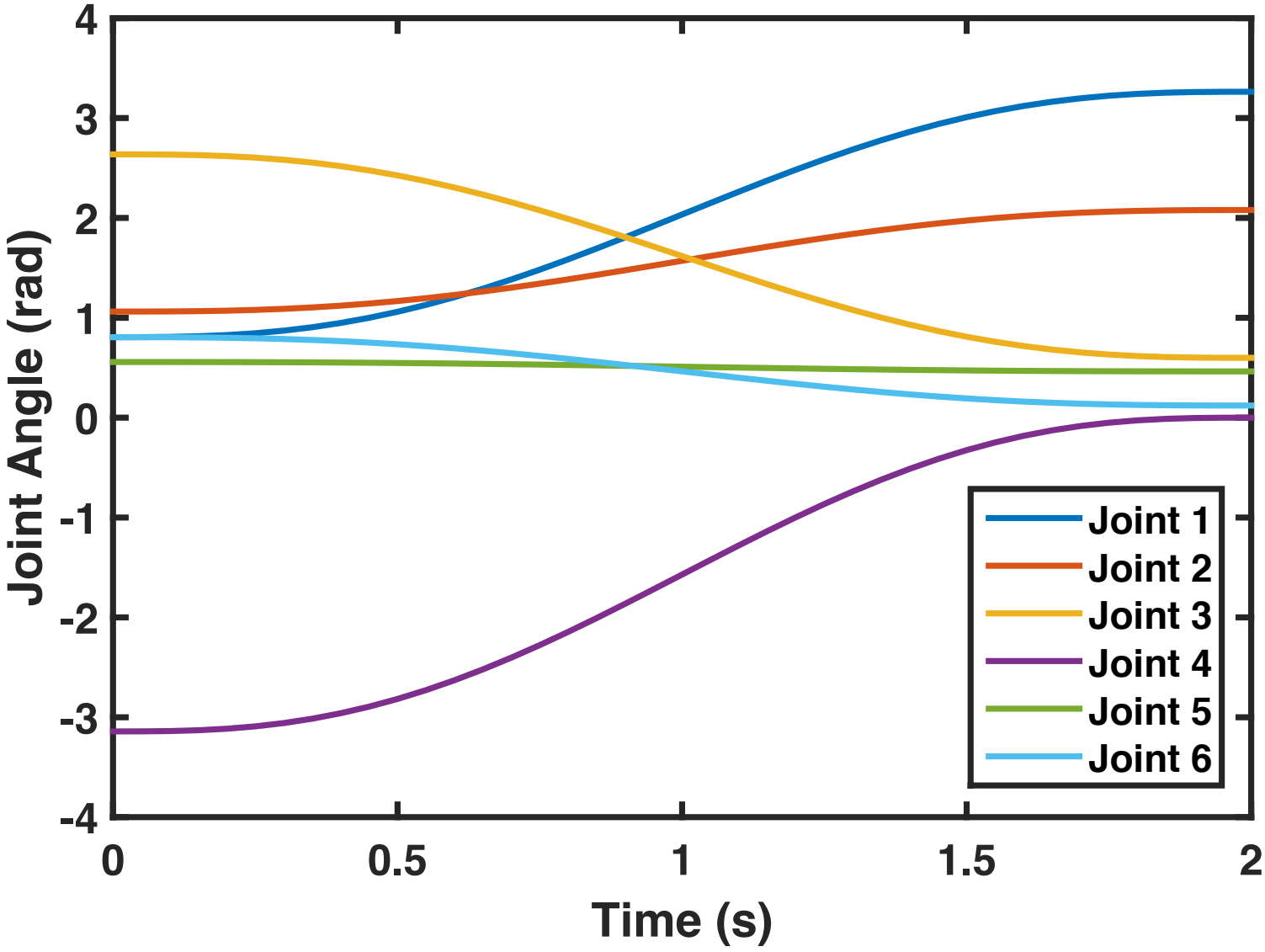
- Can a Joint Space trajectory be used?
  - Yes, the location and orientation may be uncontrollable, but each joint can smoothly change to its new configuration. Need to find the joint angles
- Can a Cartesian space trajectory be used?
  - No, because the initial and final poses are the same this trajectory generation technique will not move the robot. It believes that it is at the final solution.

Trajectory Generation : Configuration Change MATLAB PCT PUMA Example (Joint Space Trajectory)

MATLAB PCT Joint Motion Near a Singularity Example

```
mdl_puma560
T = transl(0.4,0.2,0)*trotx(pi)
% Right handed Elbow up
q1 = p560.ikine6s(T,'ru')
% Left handed Elbow up
q2 = p560.ikine6s(T,'lu')
t = [0:0.05:2]' % 2 Seconds, 50 ms period
q = jtraj(q1,q2,t)
plot(t,q)
p560.plot(q)
```

Trajectory Generation : Configuration Change MATLAB PCT PUMA Example (Joint Space Trajectory)



Trajectory Generation : Summary Joint Space Configuration Change

Joint space trajectories are a simple method to move between two configurations. This is because:

- The initial and final joint angles are unique and can be found.
- Smooth trajectories can be mathematically described between these configurations

Remember, the **joint space** trajectory generation does not allow the end effector orientation to be controlled along the path, only at the two ends.

Applications of Trajectory Generation

Trajectory Generation can be applied to more general cases, where the robot follows prescribed paths.

Two applications will be investigated:

1. Drawing the Letter 'E' with a PUMA 560 robot
2. Calculating and demonstrating the motion of a walking robot

Trajectory Generation : Drawing 'E' MATLAB PCT PUMA Example

In order to draw the 'E' in 3D space, a trajectory is required to be generated.

Trajectory Generation Process

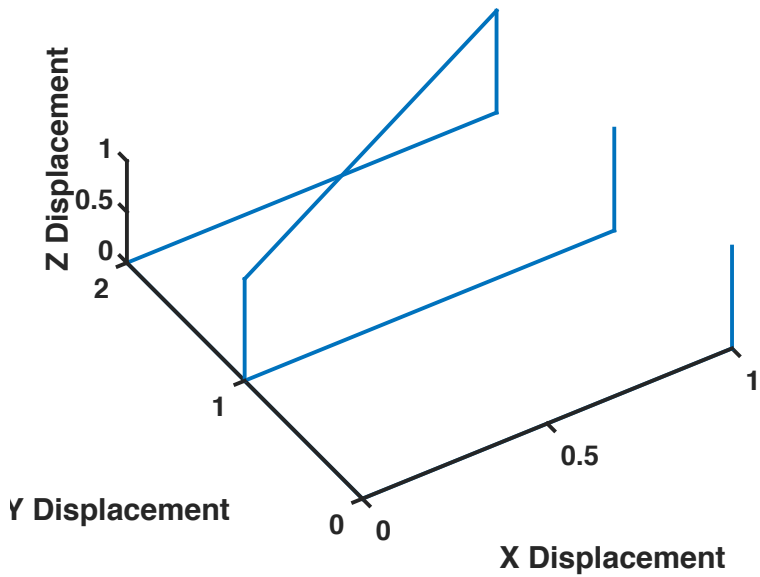
1. Specify the nodes through which the path should travel
2. Interpolate the path to create a trajectory
3. Use inverse kinematics to find the joint angles
4. Plot the trajectory to observe the behaviour

Trajectory Generation : Drawing 'E' MATLAB PCT PUMA Example

Trajectory Generation Example 'E'

Create the following path of X,Y,Z co-ordinates:

$$path = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 2 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$



Trajectory Generation : Drawing 'E' MATLAB PCT PUMA Example

The 'E' path is very large, can interpolate it and scale it and move the start to (.4 0 0) by performing the following:

MATLAB PCT Drawing 'E' with PUMA

```
Syntax :  trajectory = mstraj(path,[speedx
speedy speedz],[],[initx inity initz],
sampleperiod,acceltime)

p=mstraj(path,[.5 .5 .5],[],[2 2 2],0.02,
0.2)
% total_time = timesteps * step time
total_time = numrows(p)*0.02 T =
transl(0.1*p) % Scale to 10% size
T_Final = homtrans(transl(0.4,0,0),T)
```

Trajectory Generation : Drawing 'E' MATLAB PCT PUMA Example Video

Trajectory Generation : Drawing 'E' MATLAB PCT PUMA Example

However, from the previous example the 'E' is drawn upward (eg. on the underside of the desk) The orientation

can be flipped by rotating about the x axis by 180 degrees or π radians.

MATLAB PCT Drawing 'E' with PUMA on Table

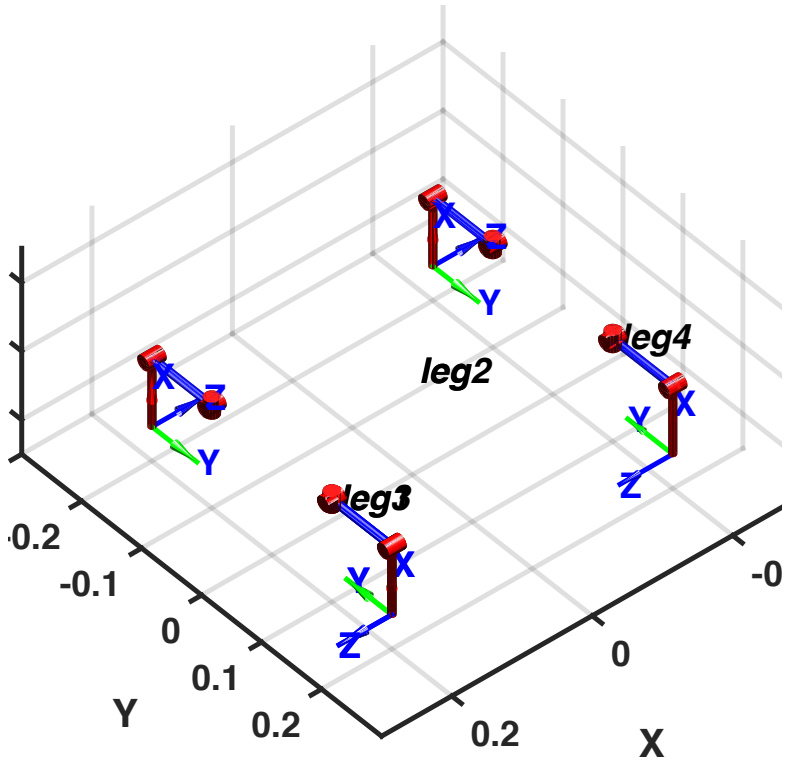
```
p560.tool = trotx(pi)
q = p560.ikine6s(T_Final)
p560.plot(q)
```

Trajectory Generation : Walking Robot MATLAB PCT Example

Suppose that a 4 legged robot needs to have trajectories developed for walking. How can this be implemented?

Walking Robot Example

Suppose that the robot has 4 legs, spaced 0.4 m apart front to back and 0.2 m from left to right. The legs have an RRR configuration with 2 revolute joints acting at the hip, and one at the knee.

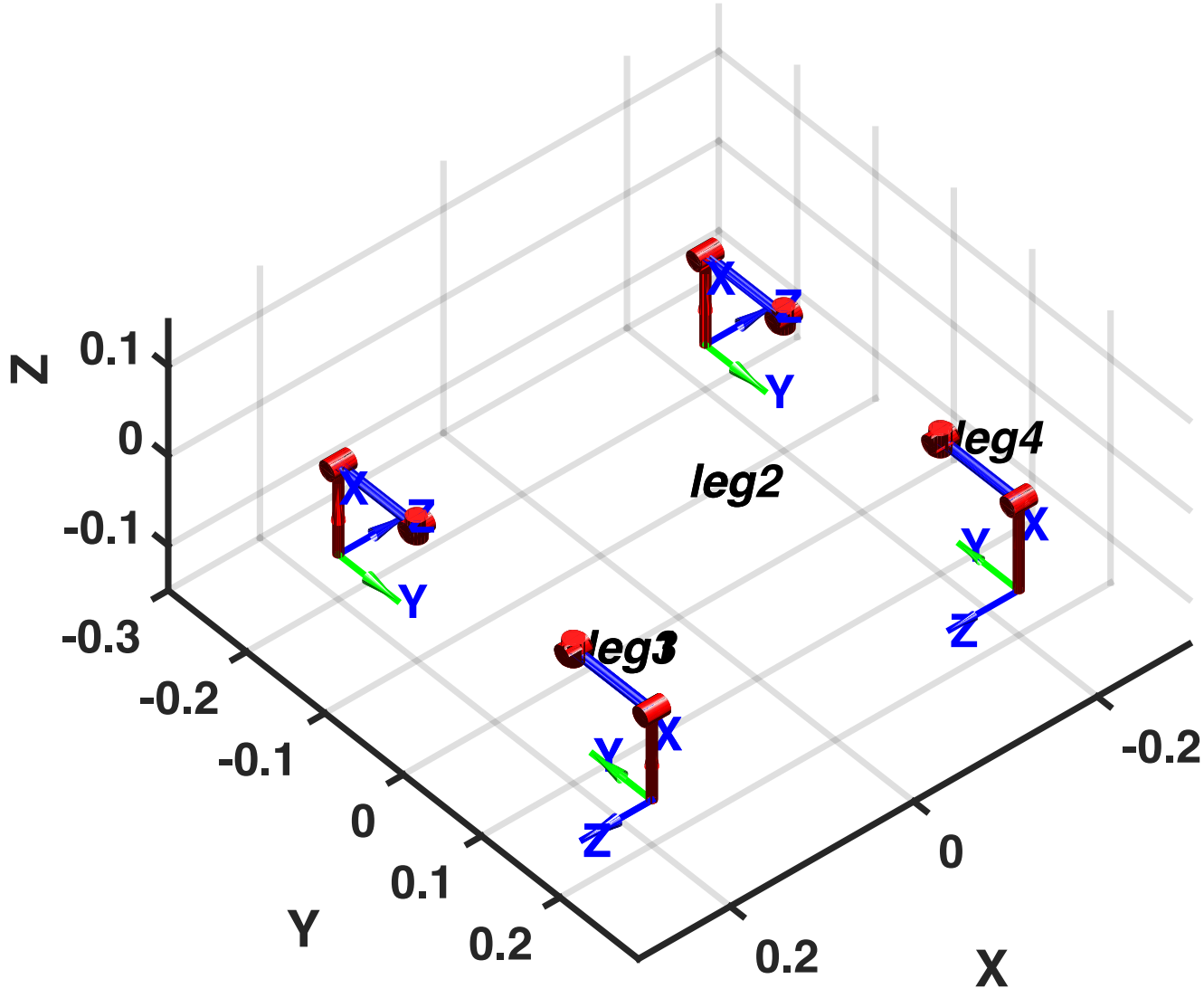


Trajectory Generation : Walking Robot MATLAB PCT Example

Walking Robot MATLAB PCT PUMA Example

```
L(1) = Link([0,0,0,pi/2])
L(2) = Link([0,0,.1,0])
L(3) = Link([0,0,-.1,0])
legs(1) = SerialLink(L,'name','leg1','offset',[pi/2,0,pi/2],
'base',transl(.2,.1,0))
legs(2) = SerialLink(L,'name','leg2','offset',[pi/2,0,pi/2],
'base',transl(-.2,.1,0))
legs(3) = SerialLink(L,'name','leg3','offset',[pi/2,0,pi/2],
'base',transl(.2,-.1,0)*trotz(pi))
legs(4) = SerialLink(L,'name','leg4','offset',[pi/2,0,pi/2],
'base',transl(-.2,-.1,0)*trotz(pi))
hold on
legs(1).plot([0 0 0],'nobase','noshadow')
legs(2).plot([0 0 0],'nobase','noshadow')
legs(3).plot([0 0 0],'nobase','noshadow')
legs(4).plot([0 0 0],'nobase','noshadow')
axis([-.3 .3 -.3 .3 -.15 .15])
```

Trajectory Generation : Walking Robot MATLAB PCT Example



Trajectory Generation : Walking Robot MATLAB PCT Example

Now that a 4-legged robot of appropriate dimensions has been created, how can the leg trajectories be created so that it walks?

Similar to creating the trajectory for the 'E'

Need to specify nodes, and interpolate between them. All feet on the ground need to move with the same linear speed, rather than accelerating between points along the path, a time for each segment will be specified, and the velocity will be constant across each vector.

Can only have 1 leg in the air at a time.

Trajectory Generation : Walking Robot MATLAB PCT PUMA Example

Require **Cartesian space** trajectory so that the foot positions follow the correct path.

Trajectory : Walking Robot MATLAB PCT Example

Decide on a ride height(platform off ground), step length, step height, and foot offset from the robot edge. For example:
ride height = rh = 0.05 m
step length = sl = 0.05 m
step height = sh = 0.01 m
foot offset = fo = 0.07 m

Trajectory : Walking Robot MATLAB PCT Example

```
rh = 0.05, sl = 0.05, sh = 0.01, fo = 0.07
```

Walking Robot MATLAB PCT Example

Walking Robot MATLAB PCT Example

```
path = [sl/2 fo -rh;-sl/2 fo -rh;-sl/2 fo -rh+sh;sl/2 fo -rh+sh]
p = mstraj(path,[],[0,3,.25,.5,.25],path(1,:),0.01,0)
qcycle = legs(1).ikine(transl(p),[],[1 1 1 0 0 0])
plot3([path(:,1);path(1,1)],[path(:,2);path(1,2)],[path(:,3);
path(1,3)])
axis([-.2 .2 -.2 .2 -.15 .15])
legs(1).plot(qcycle(i,:),'nobase','noshadow')
```

4 Legs all walking MATLAB PCT Example

```
qcycle = legs(1).ikine(transl(p),[],[1 1 1 0 0 0])
fig = figure('visible','on');
plot3([path(:,1);path(1,1)],[path(:,2);path(1,2)],[path(:,3);
path(1,3)])
axis([-.5 .5 -.5 .5 -.5 .5])
fps = 25;
n_samples = 5 * fps;
filename = 'thing.avi';
mov.frames = getframe(fig);
writerObj = VideoWriter('robotlegs.avi');
open(writerObj);
for i=1:375,
    legs(1).plot(gait(qcycle,i,0,0),'nobase','noshadow');
    legs(2).plot(gait(qcycle,i,100,0),'nobase','noshadow');
    legs(3).plot(gait(qcycle,i,200,1),'nobase','noshadow');
    legs(4).plot(gait(qcycle,i,300,1),'nobase','noshadow');
    drawnow;
    frames(i) = getframe(fig);
    writeVideo(writerObj,frame);
end
close(writerObj);
```

Trajectory Generation : Walking Robot MATLAB PCT Example Video