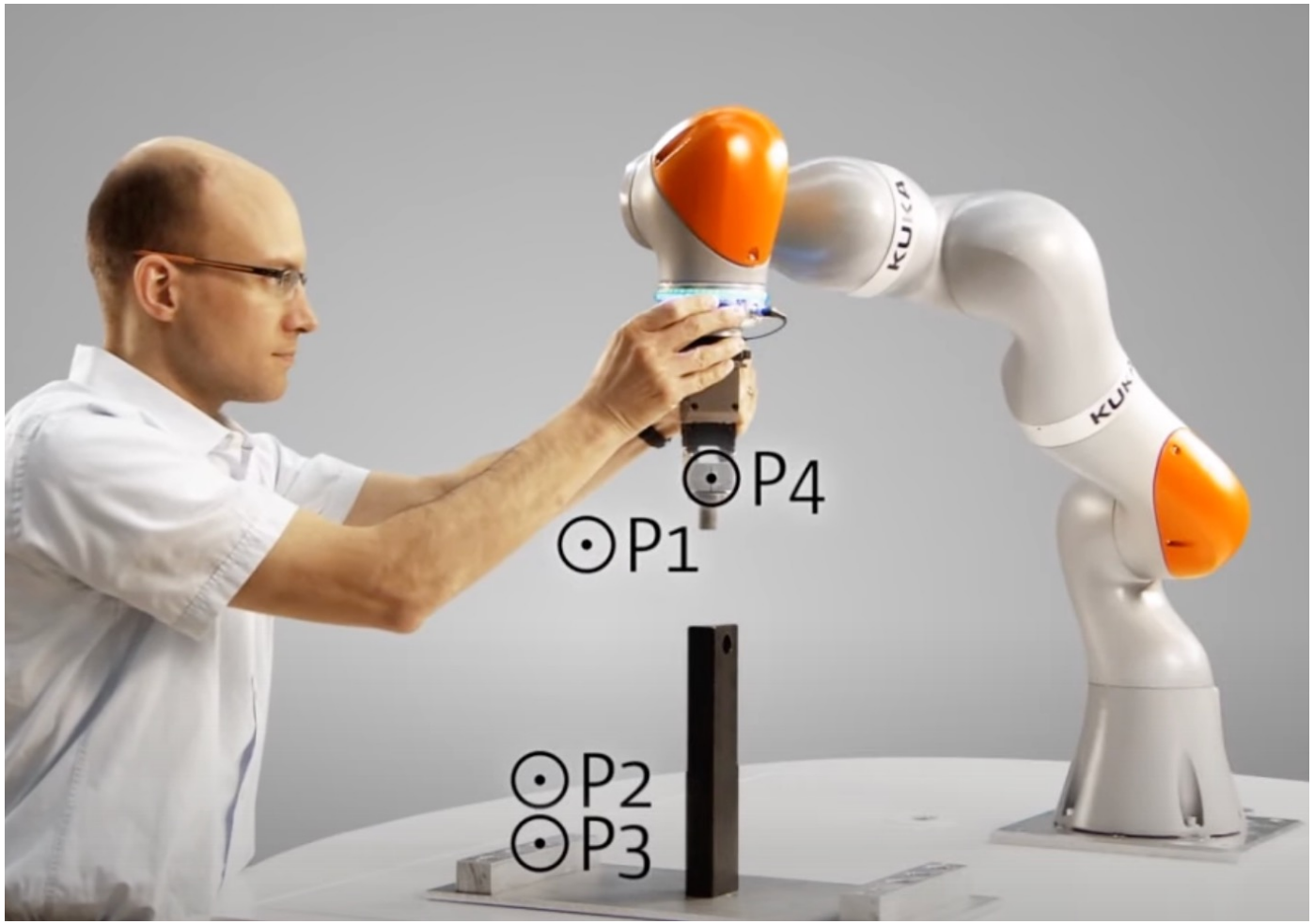
WPI

Lecture 11

Trajectory generation, Joint-Space and Cartesian Motion

Alexandros Lioulemes, PhD

Path planning - Teaching by Demonstration



https://www.youtube.com/watch?v=r7gU74Yv9Es

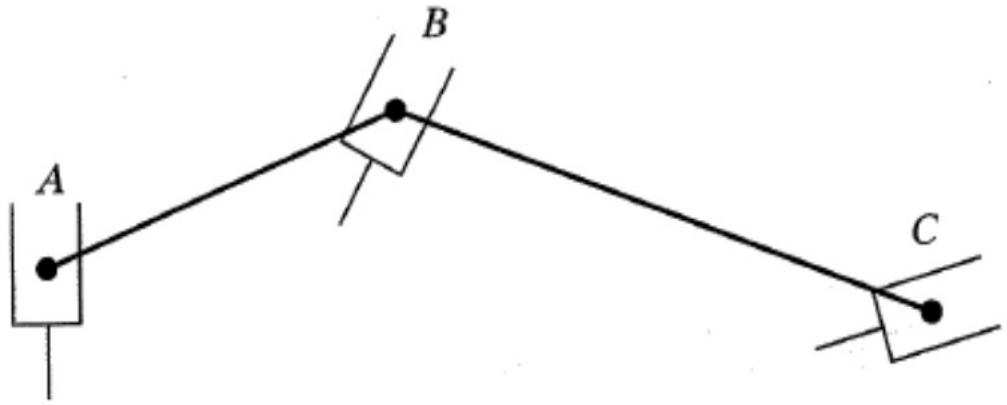
Path planning - Teaching by Demonstration

https://www.youtube.com/watch?v=r7gU74Yv9Es

Path vs. Trajectory

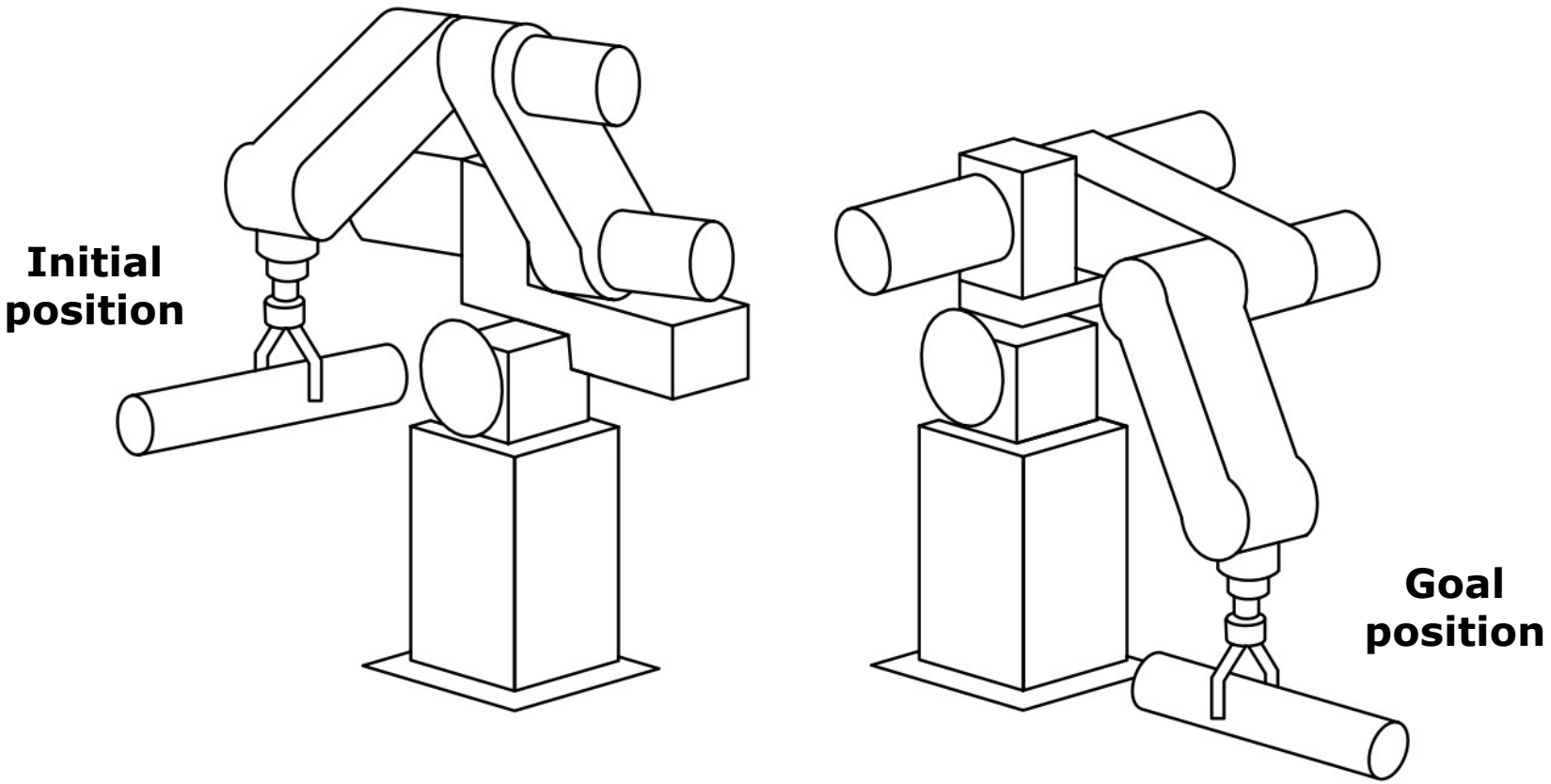
- Path is a sequence of robot configurations in a particular order that define points/positions "*p*" without regard to the timing of these configurations.

- A trajectory is a path on which a time law is specified
  - The trajectory incorporates time information as a function of time "*p(t)*".



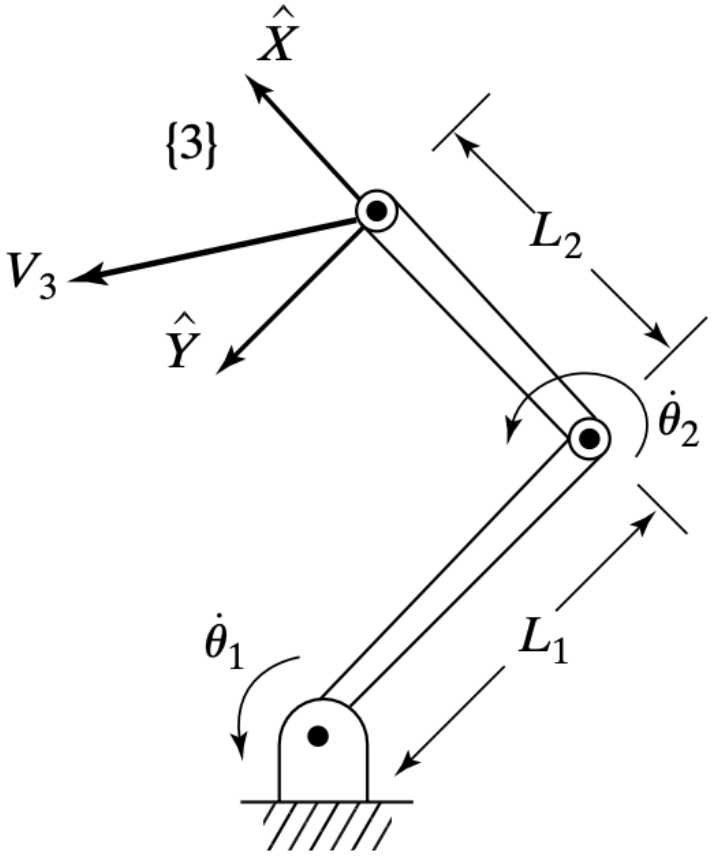
A sequential robot movements in a path

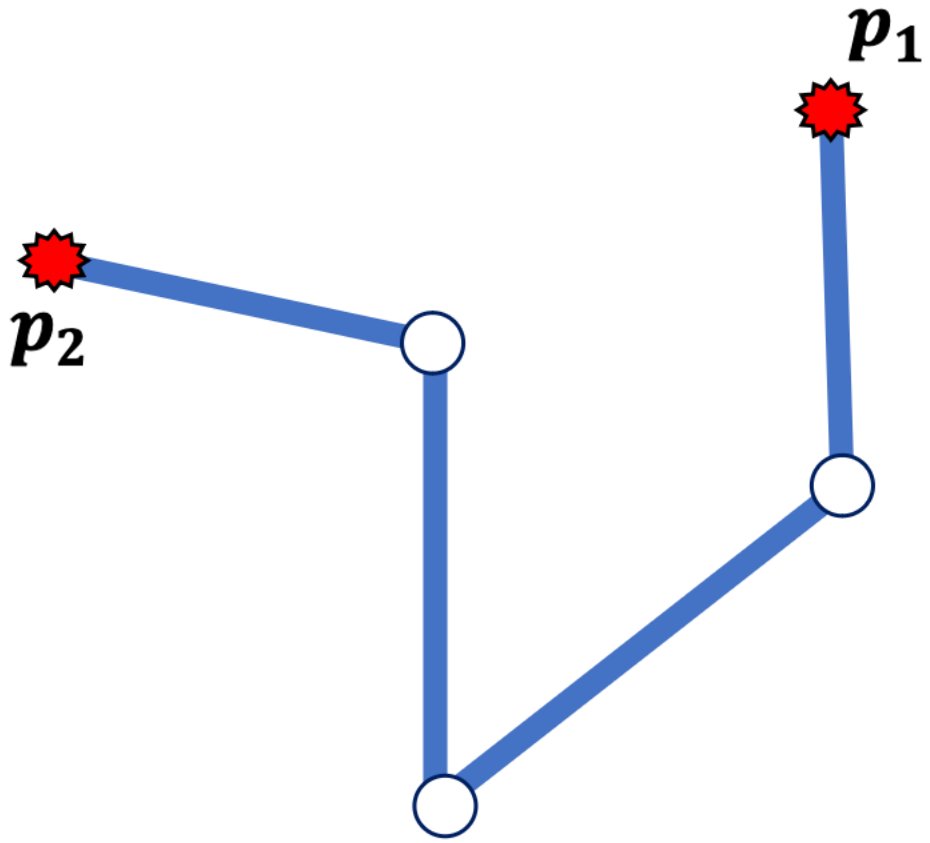
Trajectory execution



Trajectory planning

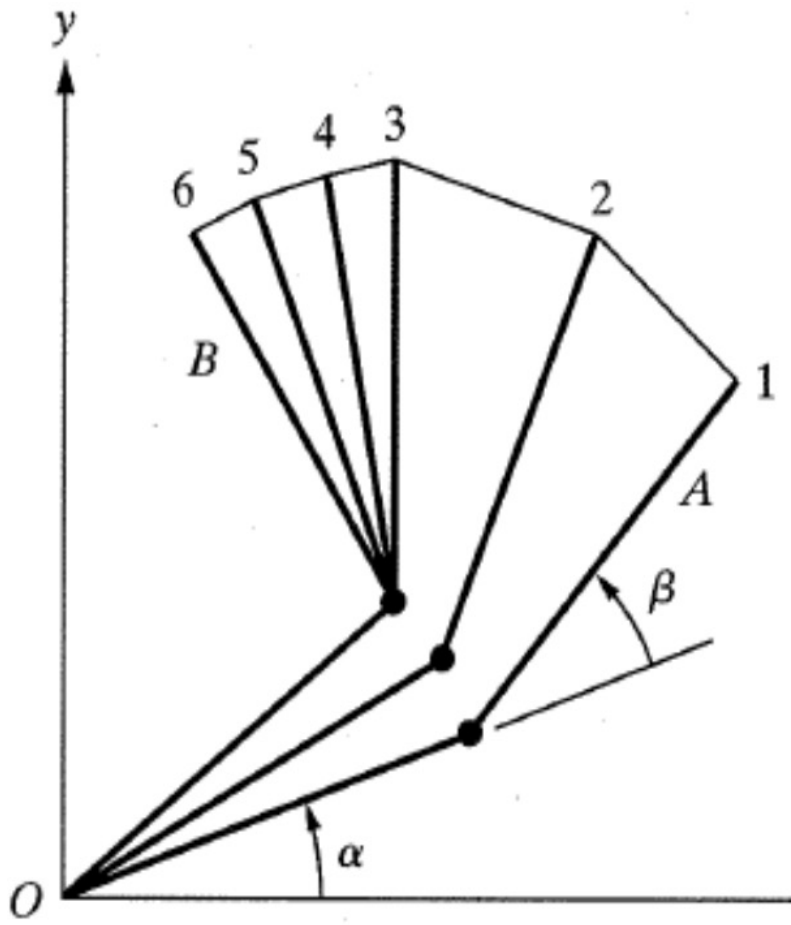
- Assume that you want to move from point $\boldsymbol{p_1}$ to $\boldsymbol{p_2}$





Trajectory planning - Joint space

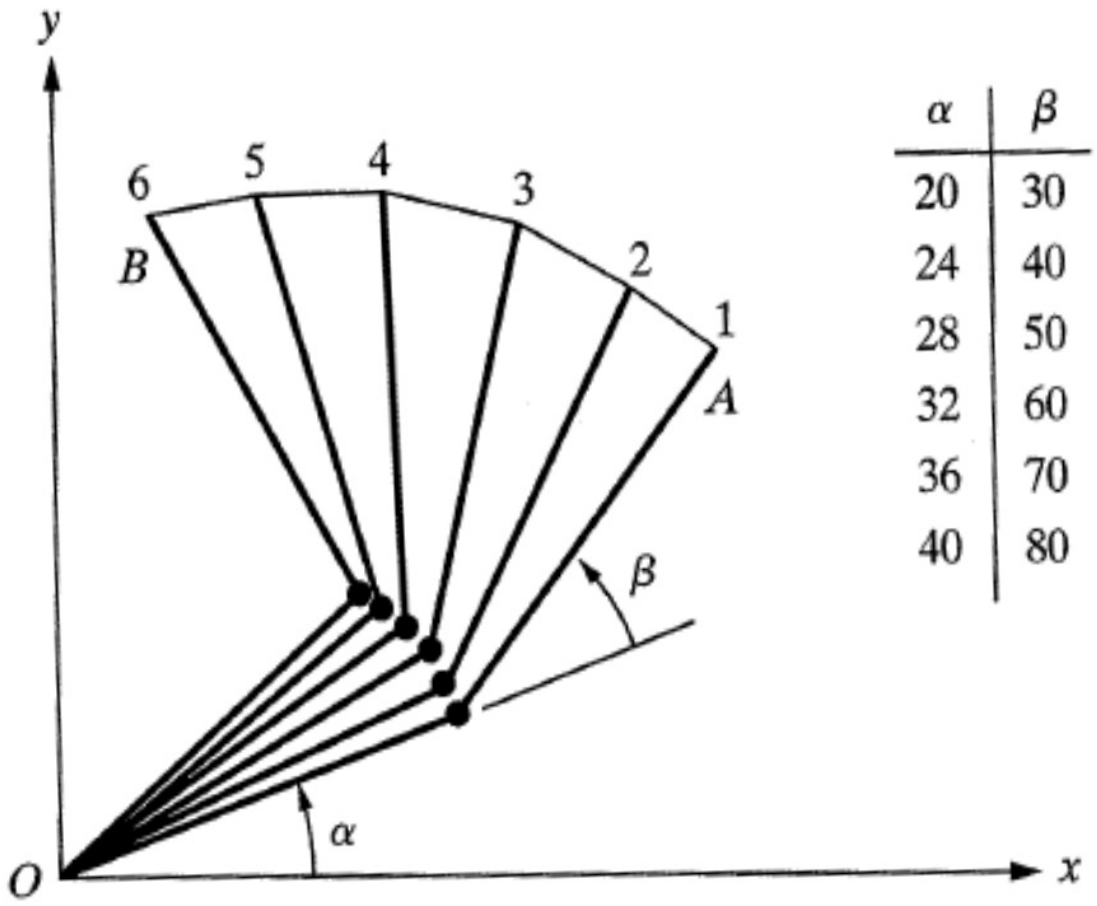
- Both joints move at the maximum rate of 10/sec



α	β
20	30
30	40
40	50
40	60
40	70
40	80

Trajectory planning - Joint space

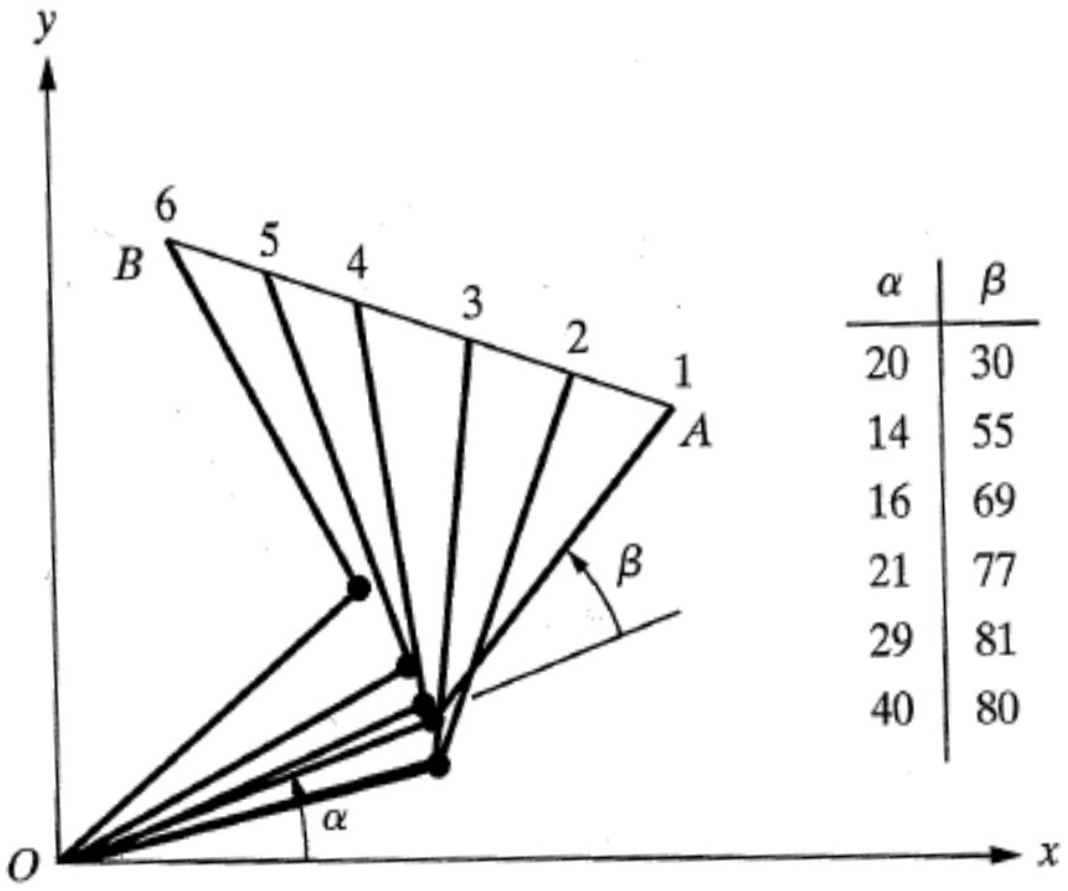
- Joints start and stop their motion simultaneously
- Both joints move at different speeds



α	β
20	30
24	40
28	50
32	60
36	70
40	80

Trajectory planning - Cartesian space

- Known path between A and B, e.g., straight line
- Joint angles are not uniformly changing



α	β
20	30
14	55
16	69
21	77
29	81
40	80

How do we plan a trajectory?

1. Points p_1 and p_2 are in the TASK SPACE!

2. We can execute motion, ONLY in the JOINT SPACE (Joints/Actuators)

3. Use IK to map from task space to joint space!

4. For each p_i you will get a specific configuration i.e. set of (vector of) joint values $\boldsymbol{q}_i$!

5. Do the trajectory planning (interpolation) in the joint space by planning a trajectory for each joint motion! (joint-by-joint)

How do we plan a trajectory?

1. Points p_1 and p_2 are in the TASK SPACE!

2. We can execute motion, ONLY in the JOINT SPACE (Joints/Actuators)

3. Use IK to

4. For each
i.e. set of

5. Do the trajectory planning (interpolation) in the joint space by planning a trajectory for each joint motion! (joint-by-joint)

All set! Just one little question: How to interpolate/plan a trajectory?!

How do we plan a trajectory?

Trajectory – a function of time **q**(t) such that

$$\mathbf{q}(t_0) = \mathbf{q}_0 \text{ and } \mathbf{q}(t_f) = \mathbf{q}_f$$

where $(t_f - t_0)$ is the time taken to execute the trajectory

T_0 : Initial Time
T_f : Final Time
q_0 : Initial Joint Position
q_f : Final Joint Position

Using Boundary Conditions, one simple solution for q(t) is a polynomial equation:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + \ldots$$

What are the Boundary Conditions (constrains)?

Boundary conditions are usually 3 things!

1. Initial and Final Positions ($q(t_0)$ and $q(t_f)$): Always must be given!
2. Initial and Final Velocities →
3. Initial and Final Acceleration →

- Each of the above adds two constraints (initial and final) to the problem!

What are the Boundary Conditions (constrains)?

1. Initial and Final Positions ($q(t_0)$ and $q(t_f)$): Always must be given!
2. Initial and Final Velocities →
3. Initial and Final Acceleration →

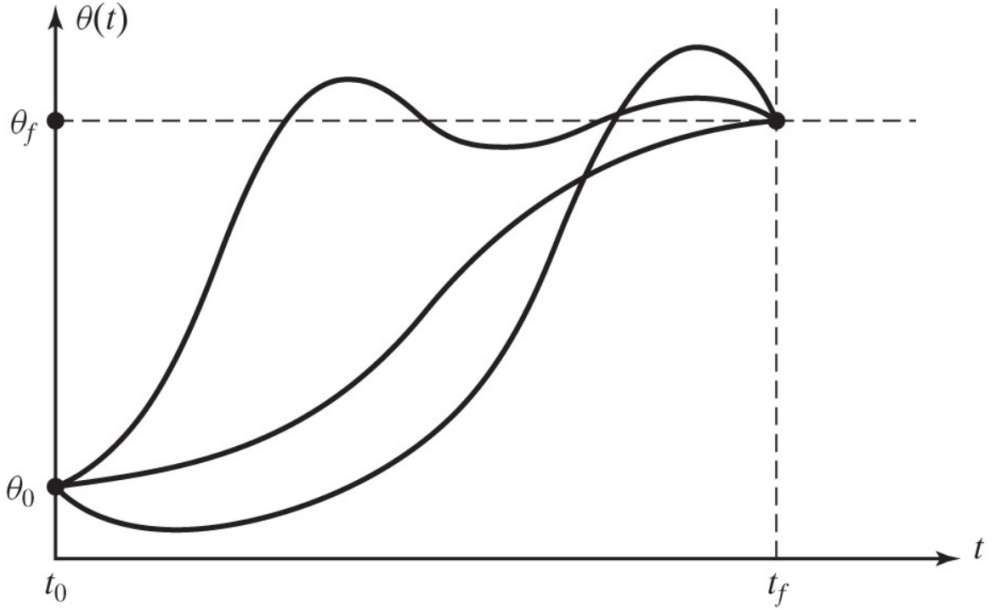
$[q_0, q_f]$
$[\dot{q}_0, \dot{q}_f]$
$[\ddot{q}_0, \ddot{q}_f]$

- Depending on the number of constraints, we can determine the order of the polynomial for the trajectory! How:
- Given 1 (total of 2 constraints) → Linear Polynomial (line)
- Given 1 & 2 (total of 4 constraints) → Cubic Polynomial
- Given 1 & 2 & 3 (total of 6 constraints) → Quintic Polynomial

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

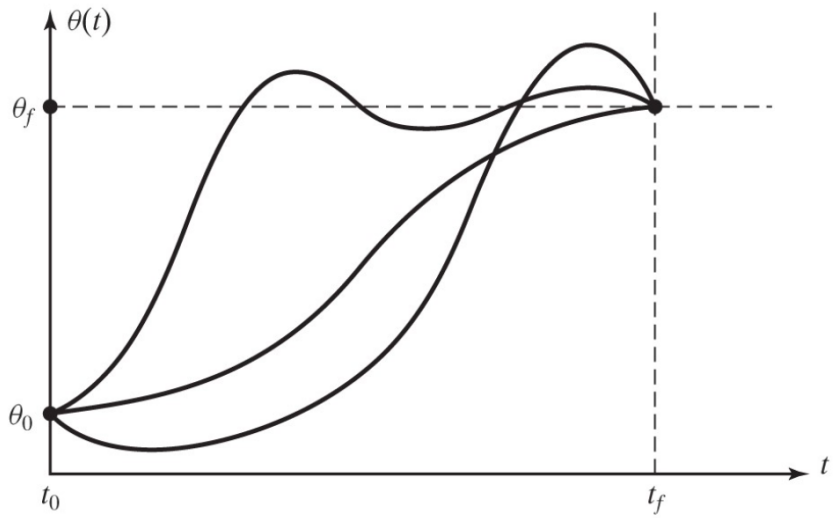
Joint-Space schemes

- **Problem:** Moving the tool from its initial pose to a goal pose in a certain amount of time.
- The initial pose of the tool is known (Joint angles)
- The goal pose can be calculated by the inverse kinematics.



Cubic Polynomials

- *Two constraints from the initial and final joint angles:*
  - $\theta(0) = \theta_0$
  - $\theta(t_f) = \theta_f$
- *Two constraints from the initial and final velocities:*
  - $\dot{\theta}(0) = 0$
  - $\dot{\theta}(t_f) = 0$



Cubic Polynomials

$$\theta(0) = \theta_0,$$
$$\theta(t_f) = \theta_f.$$
$$\dot{\theta}(0) = 0,$$
$$\dot{\theta}(t_f) = 0.$$

- We need a function *θ(t)* → $q(t)$
- Polynomial of at least third degree
- Cubic polynomial: it has four coefficients
  - $\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$

Cubic Polynomials

$$\theta(0) = \theta_0, \quad \theta(t_f) = \theta_f, \quad \dot{\theta}(0) = 0, \quad \dot{\theta}(t_f) = 0.$$

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3,$$

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2,$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t.$$

$$\theta_0 = a_0,$$

$$\theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3,$$

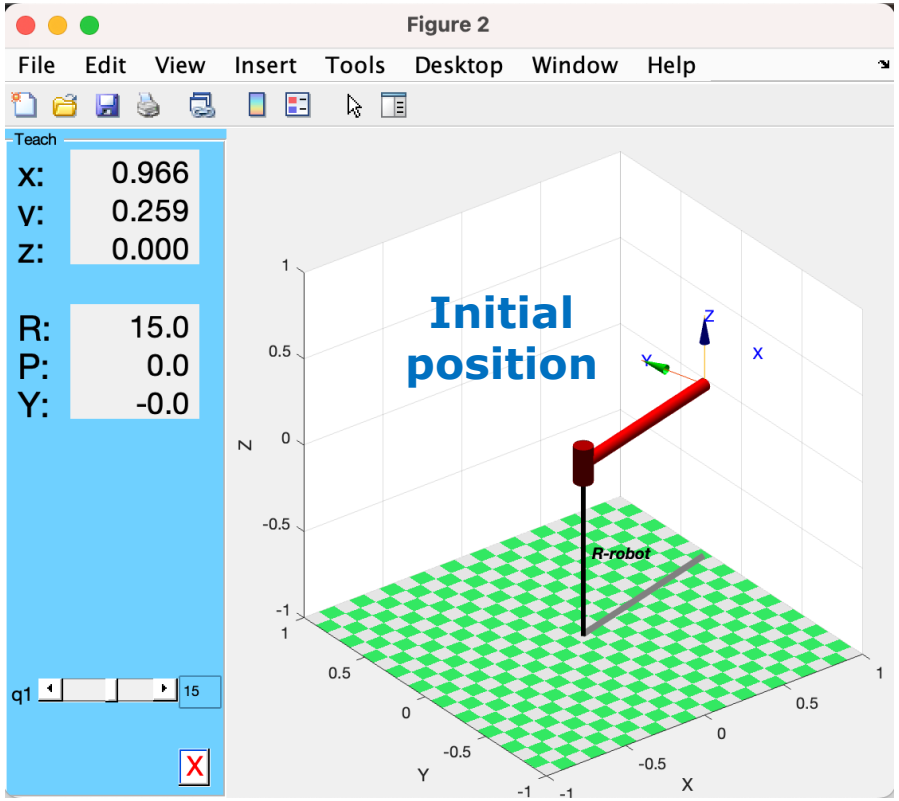
$$0 = a_1,$$

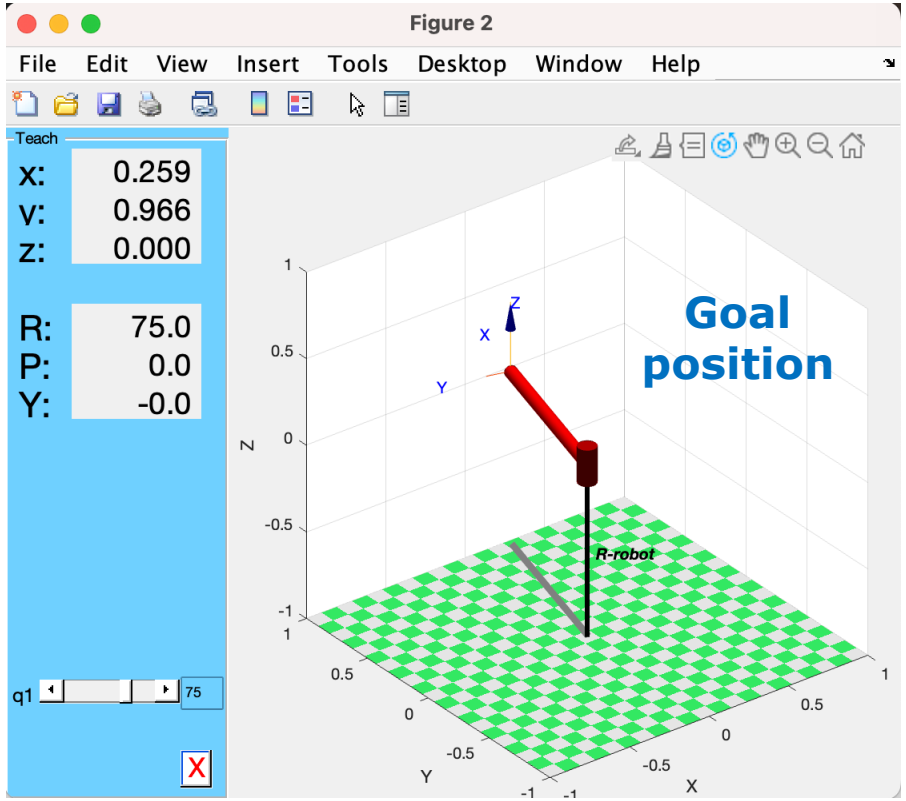
$$0 = a_1 + 2a_2 t_f + 3a_3 t_f^2.$$

$$a_0 = \theta_0, \quad a_1 = 0, \quad a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0), \quad a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0).$$

Example 1 - Cubic Polynomials

- A single-link robot with a rotary joint is motionless at θ = 15 degrees. It is desired to move the joint in a smooth manner to θ = 75 degrees in 3 seconds.
- ***Find the coefficients*** of a cubic polynomial that accomplishes this motion?



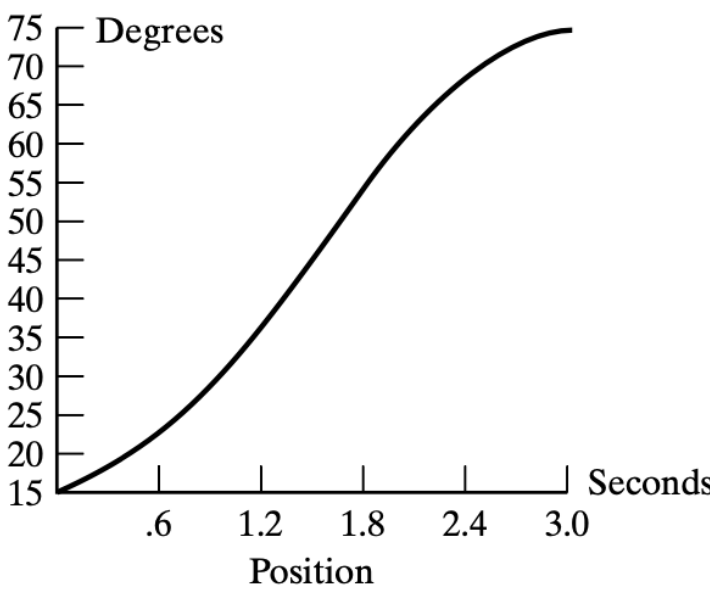


Example 1 - Cubic Polynomials

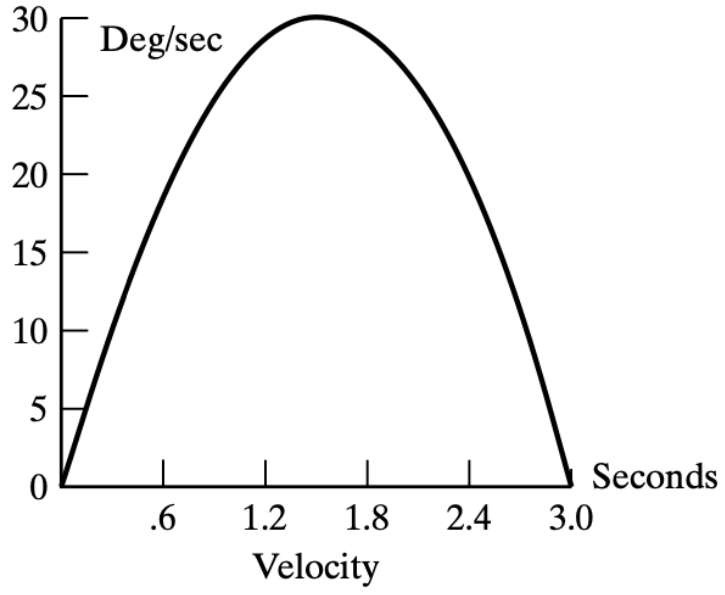
$$a_0 = \theta_0, \qquad a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0),$$
$$a_1 = 0, \qquad a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0).$$

$$a_0 = 15.0,$$
$$a_1 = 0.0,$$
$$a_2 = 20.0,$$
$$a_3 = -4.44.$$

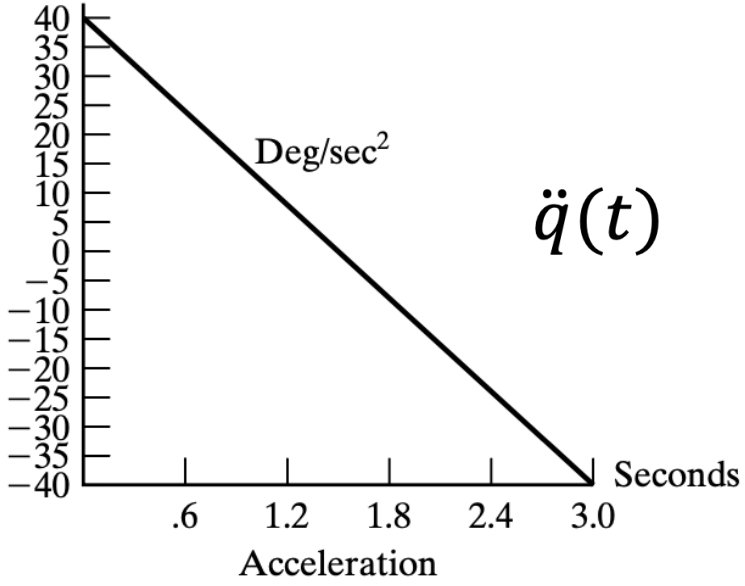
$$\theta(t) = 15.0 + 20.0t^2 - 4.44t^3,$$
$$\dot{\theta}(t) = 40.0t - 13.33t^2,$$
$$\ddot{\theta}(t) = 40.0 - 26.66t.$$



$q(t)$



$\dot{q}(t)$



$\ddot{q}(t)$

Example 1 - Cubic Polynomials - in MATLAB

```
% Cubic Polynomials
syms a0 a1 a2 a3 tf th0 thf t thff;
a0  = th0;
a1 = 0;
a2 = (3/tf^2) * (thf - th0);
a3 = (-2/tf^3) * (thf - th0);
thff = a0 + a1*t + a2*t^2 + a3*t^3;

% Set variables
th0 = 15;
thf = 75;
tf = 3;

% Evaluation
a0 = eval(a0);
%a1 = eval(a1);
a2 = eval(a2);
a3 = eval(a3);

eval(thff)
first = diff(thff, t);
eval(first)
second = diff(first, t);
eval(second)
```

$$a_0 = 15.0,$$
$$a_1 = 0.0,$$
$$a_2 = 20.0,$$
$$a_3 = -4.44.$$

$$a_0 = \theta_0, \qquad a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0),$$
$$a_1 = 0, \qquad a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0).$$

$$\theta(t) = 15.0 + 20.0t^2 - 4.44t^3,$$
$$\dot{\theta}(t) = 40.0t - 13.33t^2,$$
$$\ddot{\theta}(t) = 40.0 - 26.66t.$$

```
ans =

- 4.4444*t^3 + 20*t^2 + 15


ans =

- 13.3333*t^2 + 40*t


ans =

40 - 26.6667*t
```

Cubic polynomials for a path with via points

- Usually, we will to be able to pass through a via point without stopping
- We need to generalize the way which we fit cubic to the path constrains.

Cubic polynomials for a path with via points

- Usually we wish to be able to pass through a via point without stopping.
- *Two constraints from the initial and final joint angles for a via point:*
  - $\theta(0) = \theta_0$
  - $\theta(t_f) = \theta_f$
- *Two constraints from the initial and final velocities for a via point:*
  - $\dot{\theta}(0) = \dot{\theta}_0$
  - $\dot{\theta}(t_f) = \dot{\theta}_f$

Cubic polynomials for a path with via points

$$\dot{\theta}(0) = \dot{\theta}_0,$$
$$\dot{\theta}(t_f) = \dot{\theta}_f.$$

$$\theta_0 = a_0,$$
$$\theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3,$$
$$\dot{\theta}_0 = a_1,$$
$$\dot{\theta}_f = a_1 + 2a_2 t_f + 3a_3 t_f^2.$$

$$a_0 = \theta_0,$$
$$a_1 = \dot{\theta}_0,$$
$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) - \frac{2}{t_f}\dot{\theta}_0 - \frac{1}{t_f}\dot{\theta}_f,$$
$$a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0) + \frac{1}{t_f^2}(\dot{\theta}_f + \dot{\theta}_0).$$

Higher-order polynomials

- Higher-order polynomials are sometimes used for path segments
- Specify the position, velocity, and acceleration at the beginning and end of a path segment

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

Higher-order polynomials

$$\theta_0 = a_0,$$
$$\theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5,$$

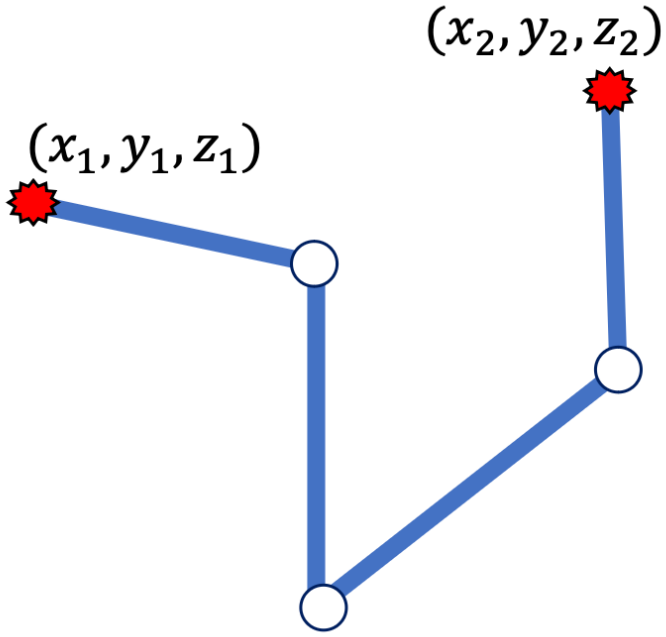
$$\dot{\theta}_0 = a_1,$$
$$\dot{\theta}_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4,$$
$$\ddot{\theta}_0 = 2a_2,$$
$$\ddot{\theta}_f = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3.$$

$$a_0 = \theta_0,$$
$$a_1 = \dot{\theta}_0,$$
$$a_2 = \frac{\ddot{\theta}_0}{2},$$
$$a_3 = \frac{20\theta_f - 20\theta_0 - (8\dot{\theta}_f + 12\dot{\theta}_0)t_f - (3\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2}{2t_f^3},$$
$$a_4 = \frac{30\theta_0 - 30\theta_f + (14\dot{\theta}_f + 16\dot{\theta}_0)t_f + (3\ddot{\theta}_0 - 2\ddot{\theta}_f)t_f^2}{2t_f^4},$$
$$a_5 = \frac{12\theta_f - 12\theta_0 - (6\dot{\theta}_f + 6\dot{\theta}_0)t_f - (\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2}{2t_f^5}.$$

Cartesian straight-line motion

Everything mentioned for trajectory planning in the joint space, i.e. $q(t)$, remains the same except replace $q(t)$ with $x(t)$ where x refers to x, y, and z in the task space.

- Points $p_1\ (x_1, y_1, z_1)$ and $p_2\ (x_2, y_2, z_2)$ are in the TASK SPACE!
- Do the trajectory planning (interpolation) in the ~~joint~~ task space by planning a trajectory for each ~~joint~~ coordinate/axis (x, y, z) motion! ~~(joint-by-joint)~~
- Use either Cubic Polynomial or Quintic Polynomial depending on the given constraints.
- Once done, you will have polynomials for $x(t)$, $y(t), z(t)$. Therefore, given a time, calculate point (x,y,z) and use IK for joint commands.



Cartesian space vs. Joint Space trajectories

	Task Space	Joint Space
Pros	• Motion is predictable (interpolation in task space) • Better handling of obstacles and collisions	• Faster execution (IK at waypoints only) • Actuator motion is smooth and easier to validate
Cons	• Slower execution (IK every time step) • Actuator motion is not necessarily smooth and more difficult to validate	• Intermediate points not guaranteed to respect joint limits or be collision-free

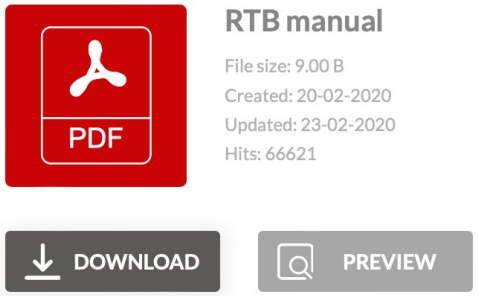
Trajectories_Puma560_Joint_space_motion.m

Joint space trajectory – Quintic Polynomial

jtraj

Compute a joint space trajectory

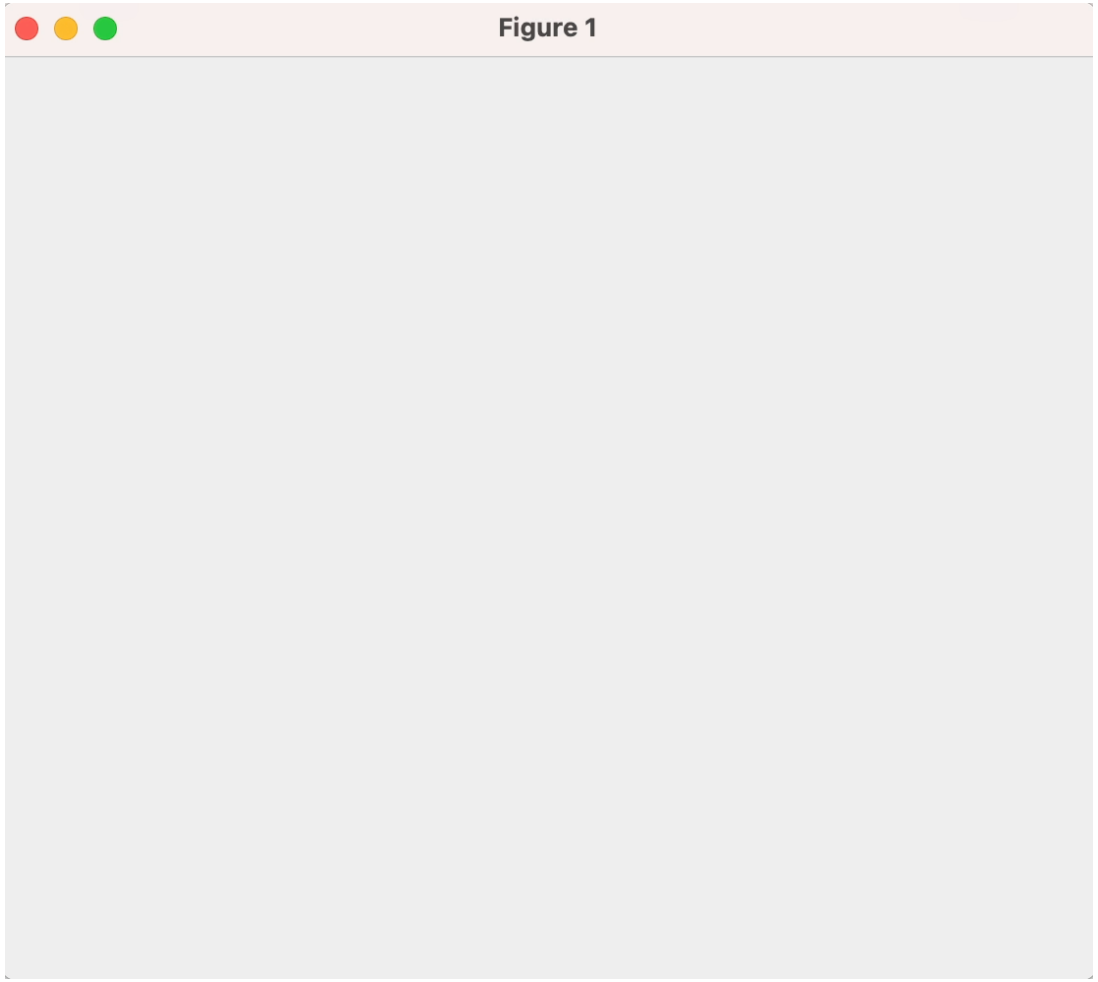
`[Q,QD,QDD] = JTRAJ(Q0, QF, M)` is a joint space trajectory `Q` ($M \times N$) where the joint coordinates vary from `Q0` ($1 \times N$) to `QF` ($1 \times N$). A quintic (5th order) polynomial is used with default zero boundary conditions for velocity and acceleration. Time is assumed to vary from 0 to 1 in `M` steps. Joint velocity and acceleration can be optionally returned as `QD` ($M \times N$) and `QDD` ($M \times N$) respectively. The trajectory `Q`, `QD` and `QDD` are $M \times N$ matrices, with one row per time step, and one column per joint.

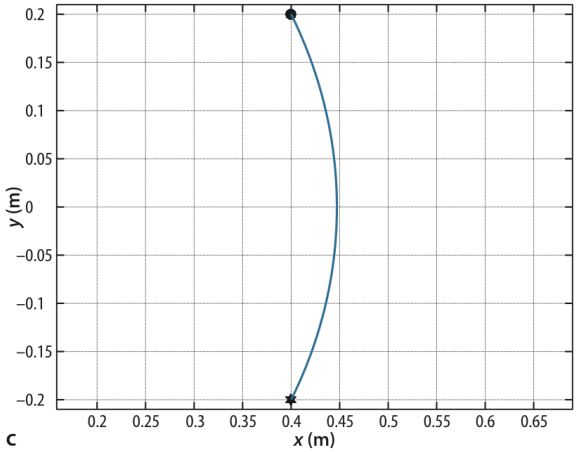


https://petercorke.com/download/27/rtb/1050/rtb-manual.pdf

Trajectories_Puma560_Joint_space_motion.m

Joint-Space Motion - PUMA 560

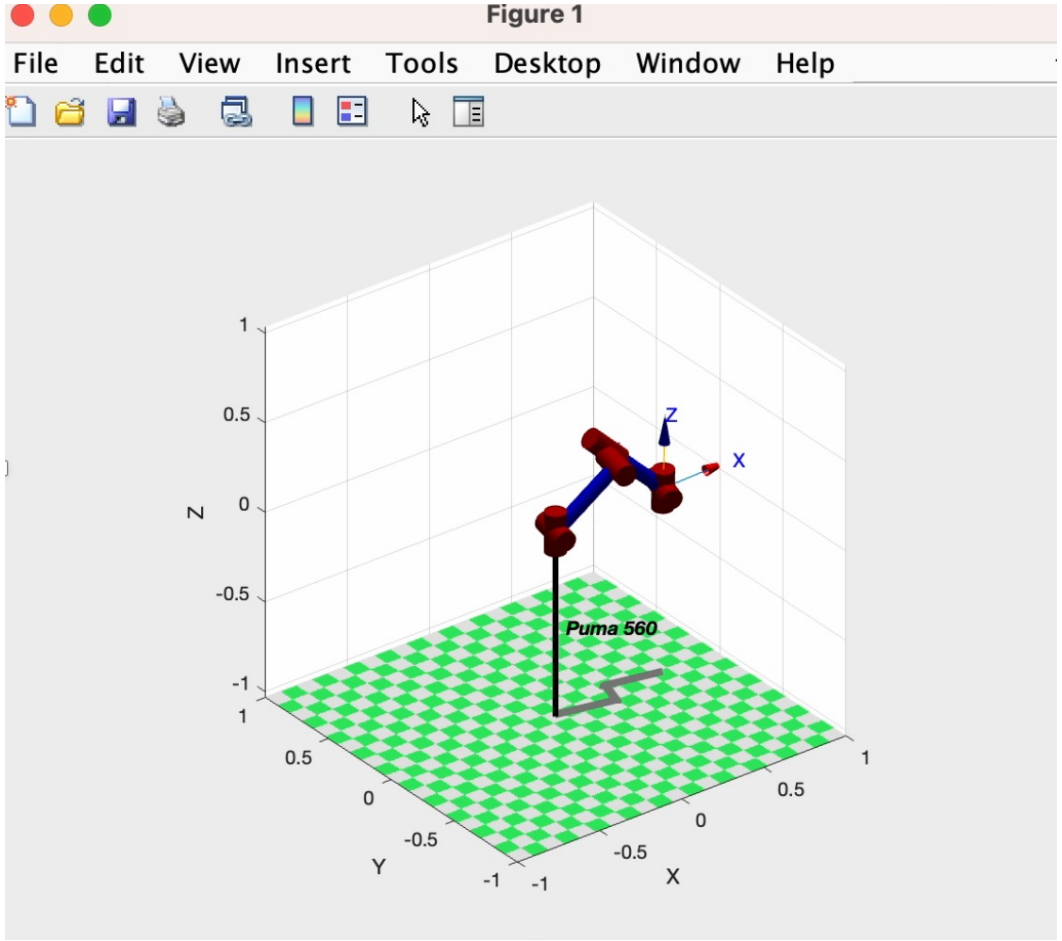


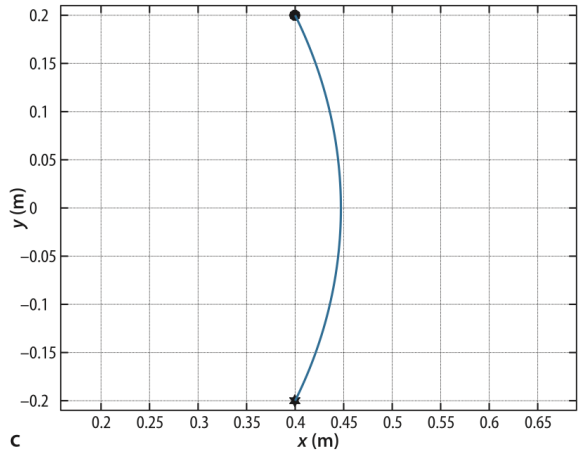


https://drive.google.com/file/d/1FovuGTJZIN8d82_tAiTYH3irZMRj2slp/view?usp=sharing

Trajectories_Puma560_Joint_space_motion.m

Joint-Space Motion - PUMA 560



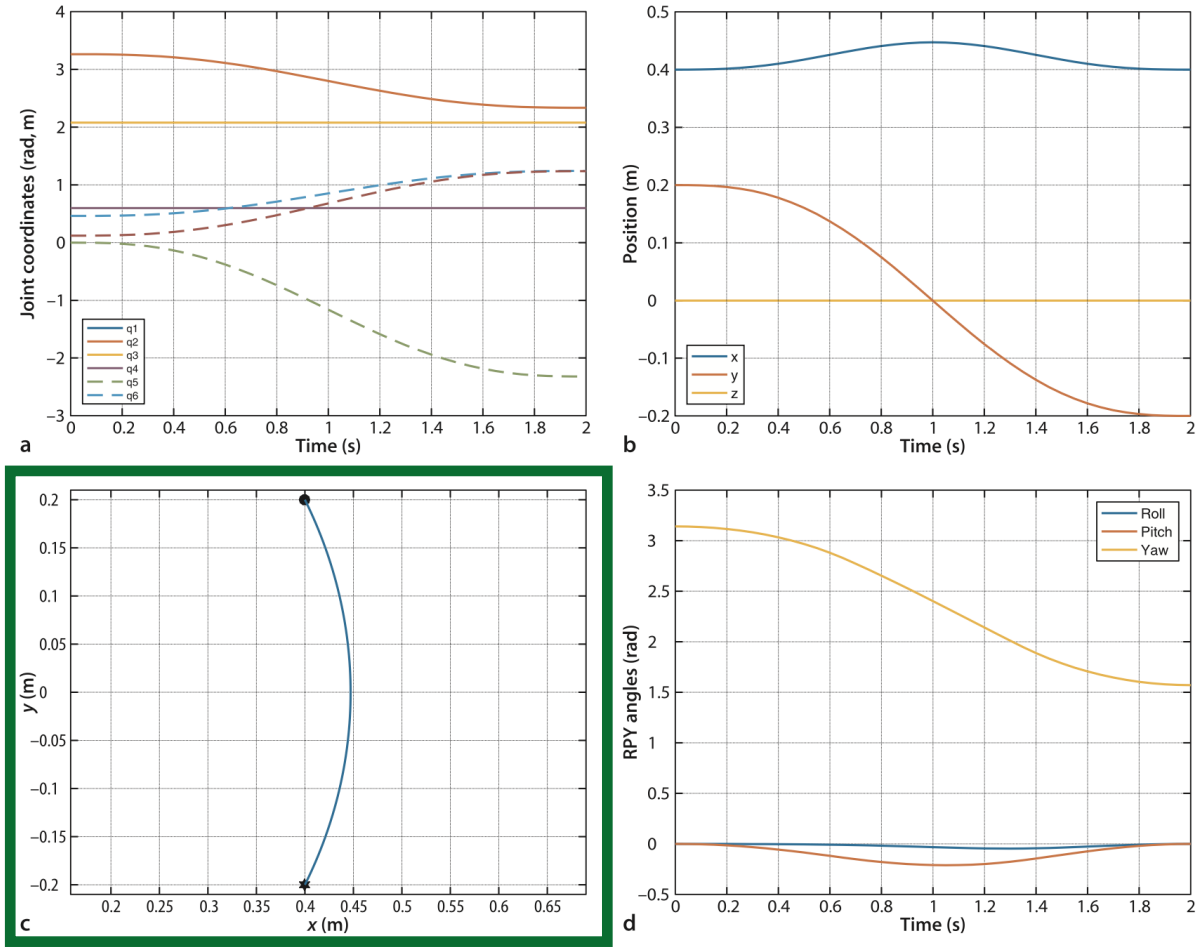


c

https://drive.google.com/file/d/1FovuGTJZIN8d82_tAiTYH3irZMRj2slp/view?usp=sharing

Trajectories_Puma560_Joint_space_motion.m

Joint-Space Motion - PUMA 560



Trajectories_Puma560_Cartesian_space_motion.m

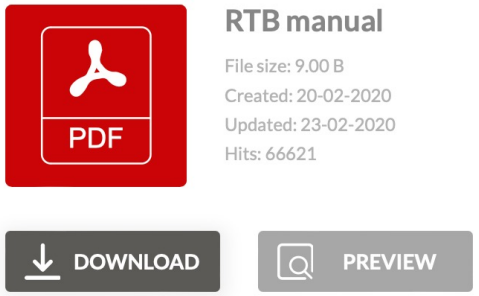
Cartesian trajectory – Quintic Polynomial

ctraj

Cartesian trajectory between two poses

`TC = CTRAJ(T0, T1, N)` is a Cartesian trajectory ($4 \times 4 \times$ `N`) from pose `T0` to `T1` with `N` points that follow a trapezoidal velocity profile along the path. The Cartesian trajectory is a homogeneous transform sequence and the last subscript being the point index, that is, T(:,:,i) is the i'th point along the path.

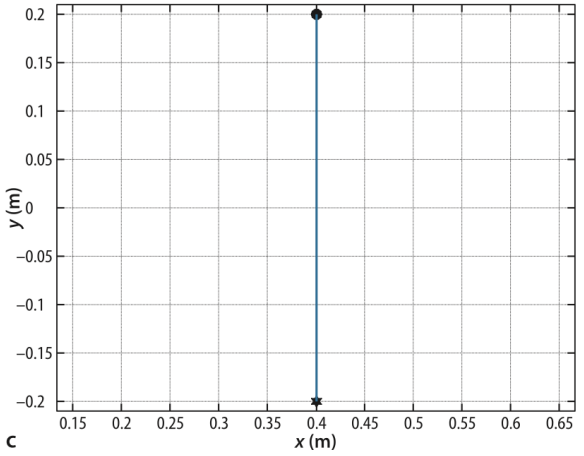
`TC = CTRAJ(T0, T1, S)` as above but the elements of `S` (`N` $\times$ 1) specify the fractional distance along the path, and these values are in the range [0 1]. The i'th point corresponds to a distance `S`(i) along the path.



https://petercorke.com/download/27/rtb/1050/rtb-manual.pdf

Trajectories_Puma560_Cartesian_space_motion.m

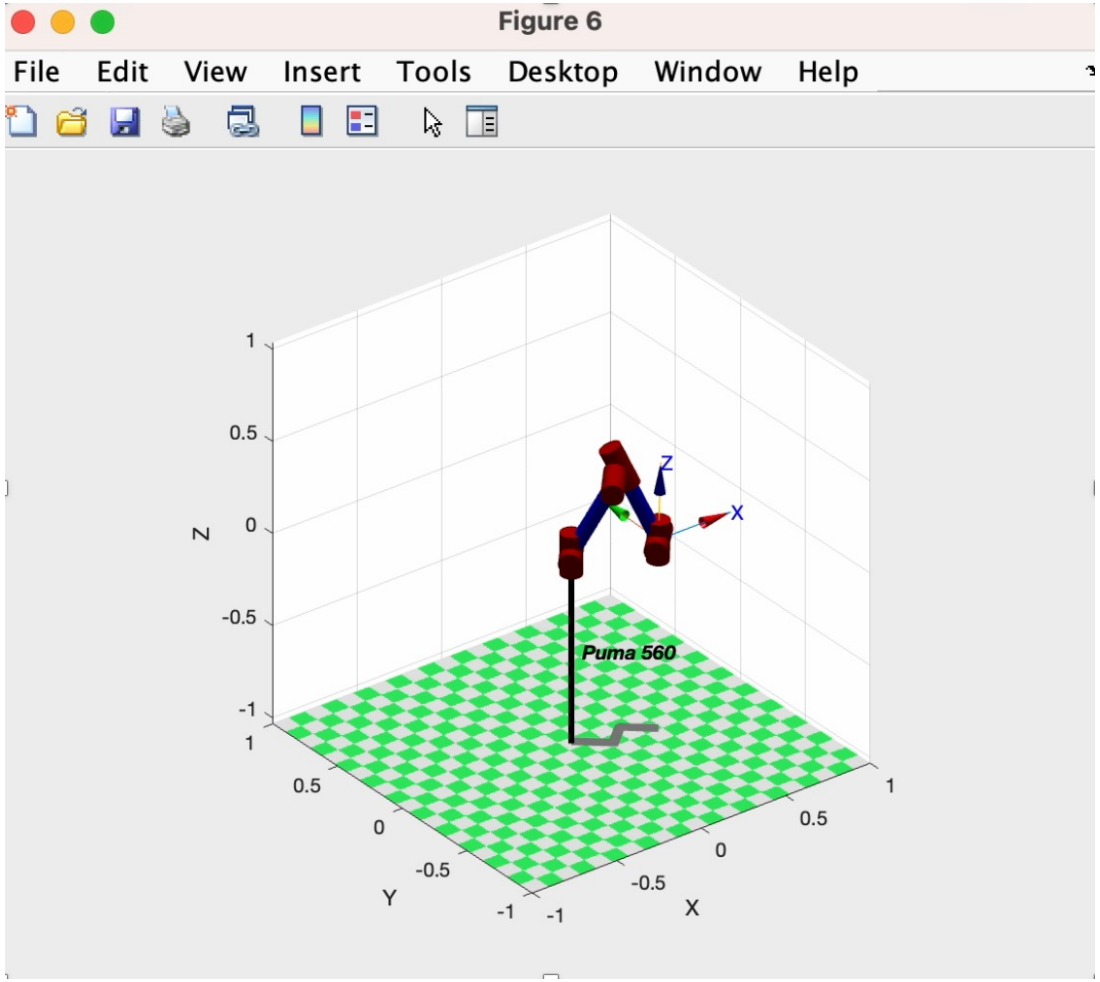
Cartesian Motion - PUMA 560

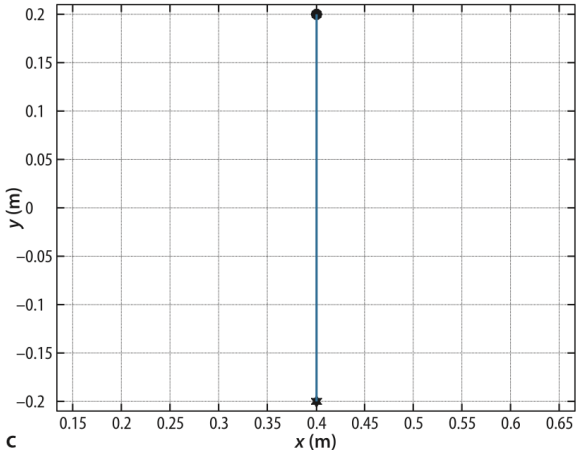


https://drive.google.com/file/d/1CZ4PsjWIk26j8qzjISQMlbgaMgZwF0aH/view?usp=sharing

Trajectories_Puma560_Cartesian_space_motion.m

Cartesian Motion - PUMA 560

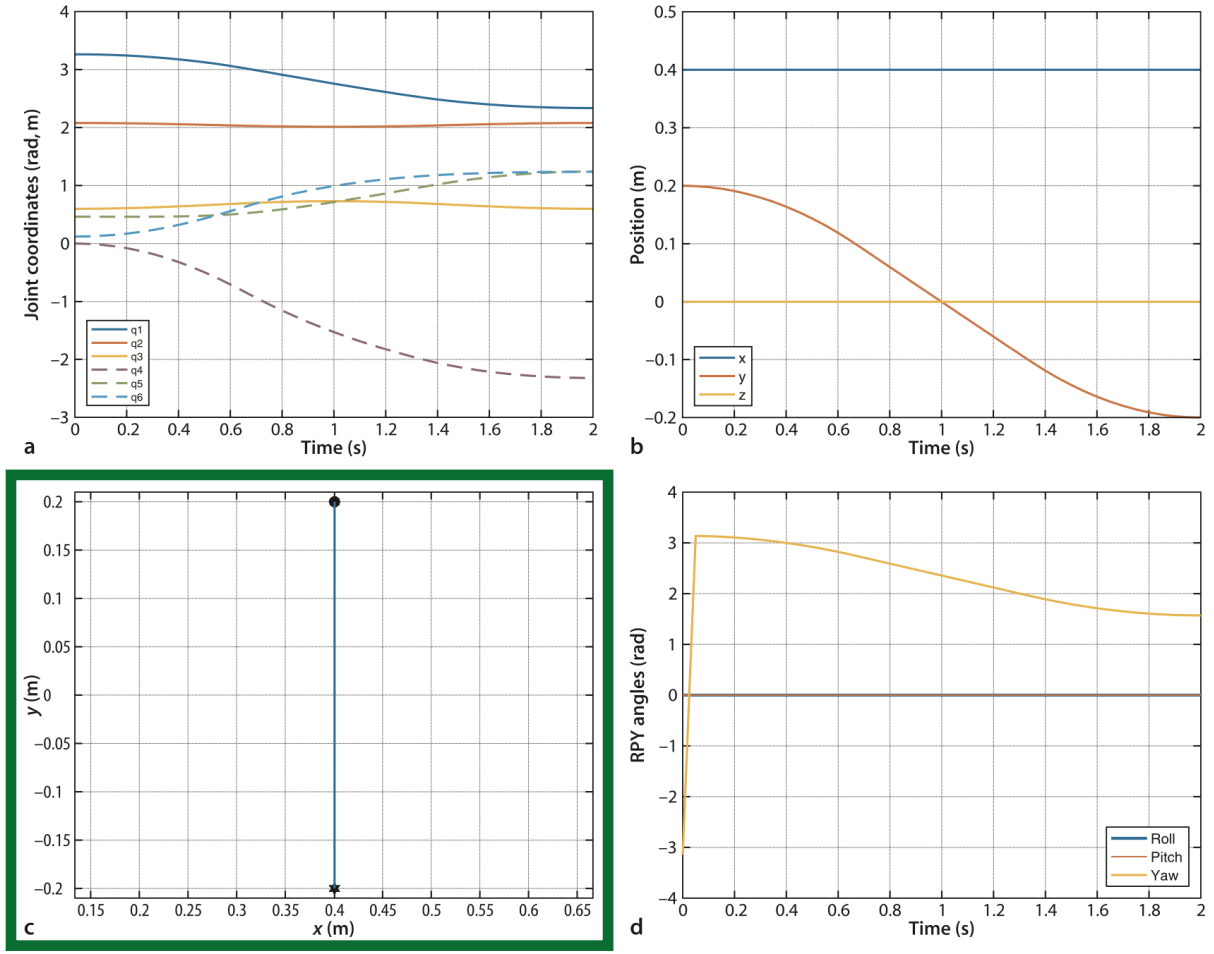




https://drive.google.com/file/d/1CZ4PsjWIk26j8qzjISQMlbgaMgZwF0aH/view?usp=sharing

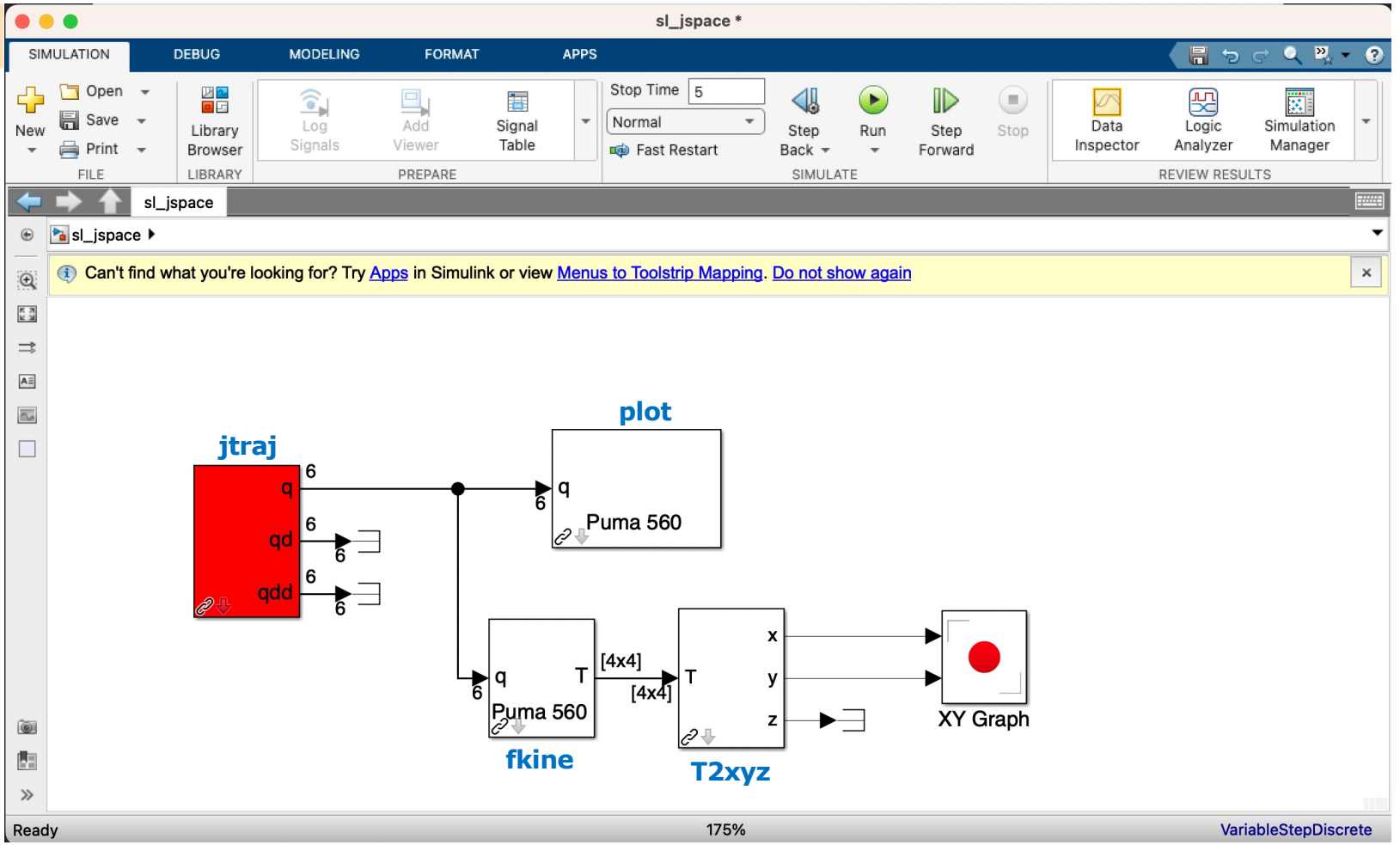
Trajectories_Puma560_Cartesian_space_motion.m

Cartesian Motion - PUMA 560



Kinematics in Simulink - MATLAB

```
>> sl_jspace
```



… end of Lecture 11