Lecture 5 - Revision

https://kahoot.it/

✓ Inverse Kinematics

- Kinematic decoupling
- Algebraic approach (solving trigonometry problems)

✓ The Jacobian

$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q}_i$$

- Use DH parameters to calculate the Jacobian

✓ Singularities

- $\det(J(\mathbf{q}))=0$
- Common singularities: Elbow, Shoulder, Wrist singularity

MTRN4230

Robotics

Lecture 7

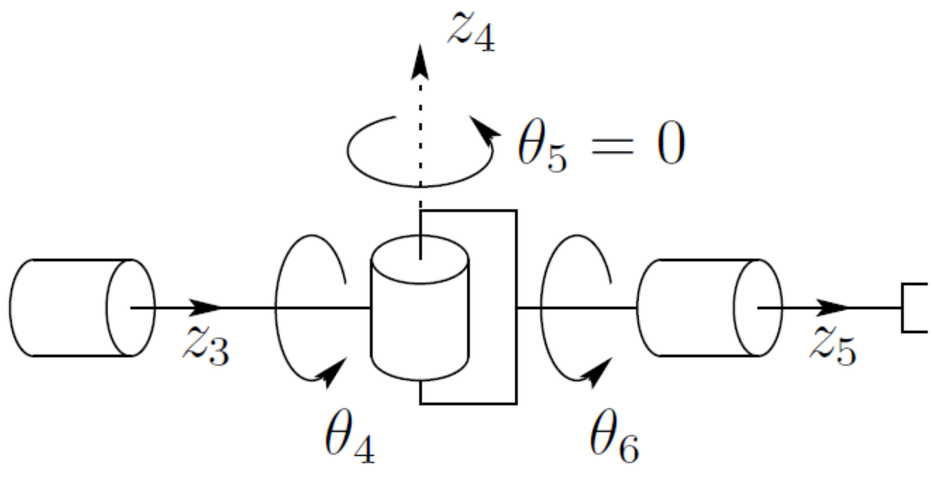
Robot Trajectories

James Stevens– T2 2023

Further explanation on singularity (Lecture 5)

- A certain configuration $\mathbf{q}$ is said to be singular if $\det(J(\mathbf{q})) = 0$
  - The robot may move very fast
  - OR it may lose some DOFs

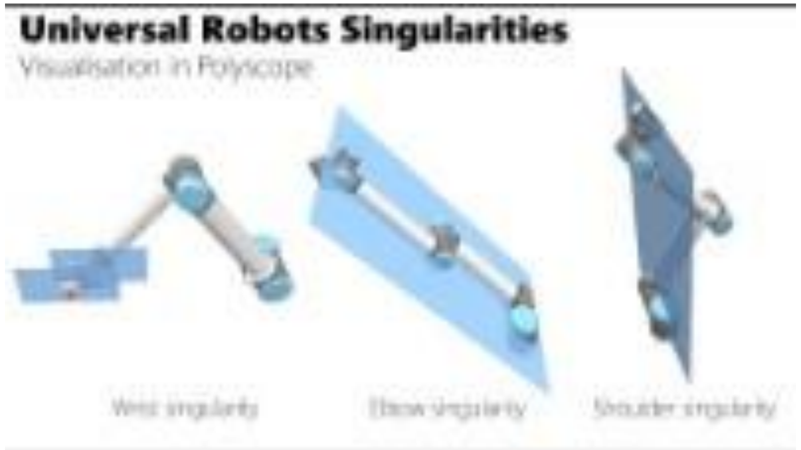
- Wrist singularity
  - Lose 1 DOF
  - Equal and opposite rotation about Z_3 and Z_5 results in no net motion of the end-effector.



$$J_{22} = [z_3 \ z_4 \ z_5] \qquad \det J_{22} = 0$$

Z_3 and Z_5 are linearly dependent

Example: UR Series Singularities



Learning Objectives

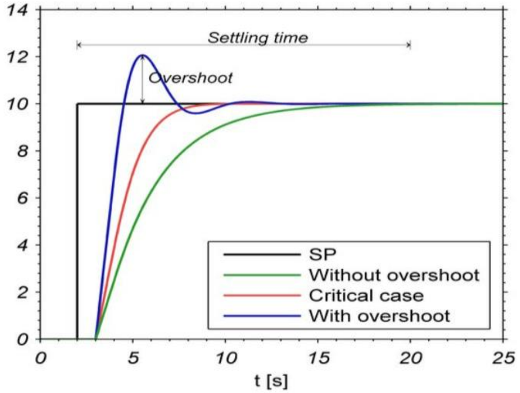
- ❑ Trajectory Generation (Joint-based method)
- ❑ Types of Trajectories
- ❑ Controllability
- ❑ Accuracy and Repeatability

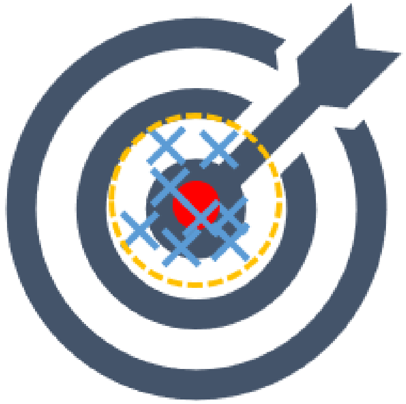
Motion Control

Aims of Robot Motion Control

- ❑ Maximise response speed and performance
- ❑ Improve Accuracy and repeatability
- ❑ Disturbance and noise rejection
- ❑ Minimise undershoot and overshoot
- ❑ Generate smooth motion
- ❑ Minimise wear on the mechanisms

We have hardware, now what?



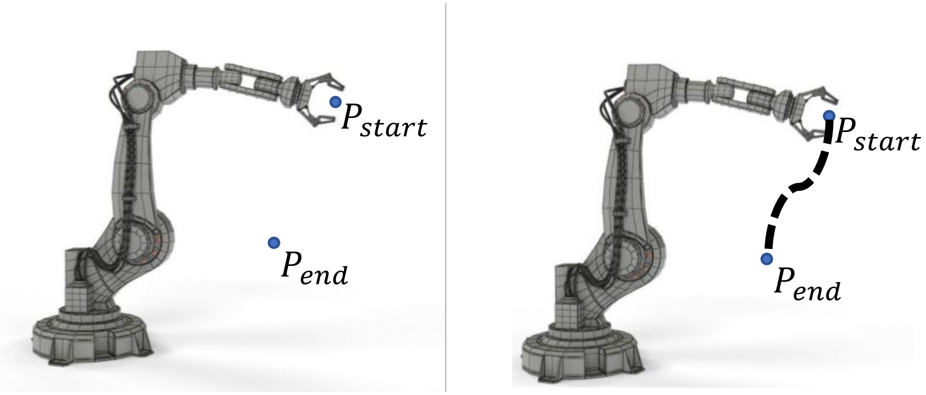


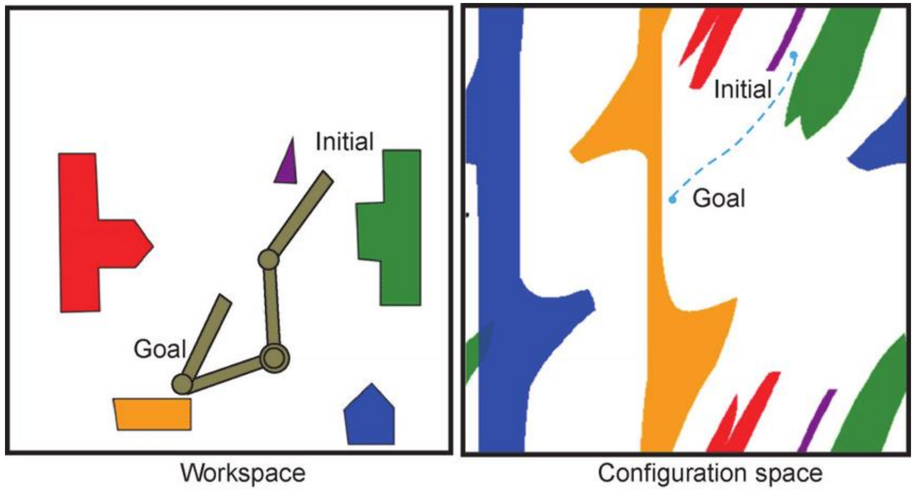
Trajectory Generation and Path Planning

Main aspects of Trajectory/Path Planning

❑ Type of Motion
- Point-to-point
- Continuous Path

❑ Trajectory Generation
- Joint-based
- Cartesian-based

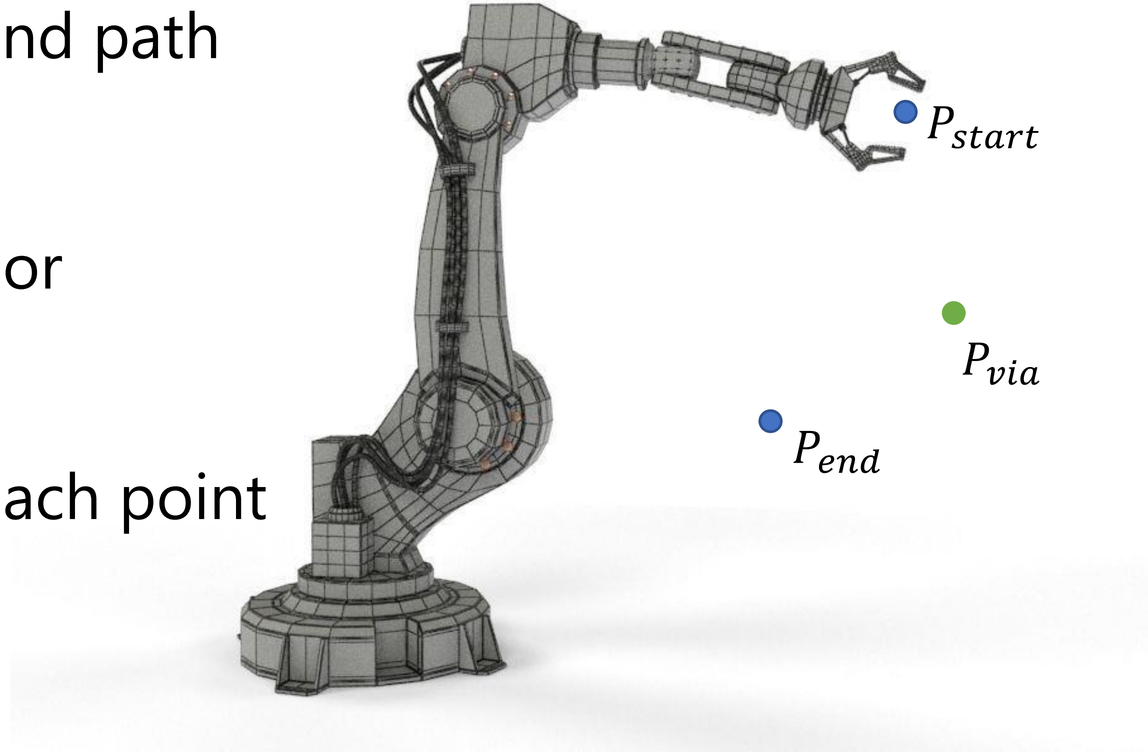




Types of Motion

❑ Point-to-point Control (PTP)

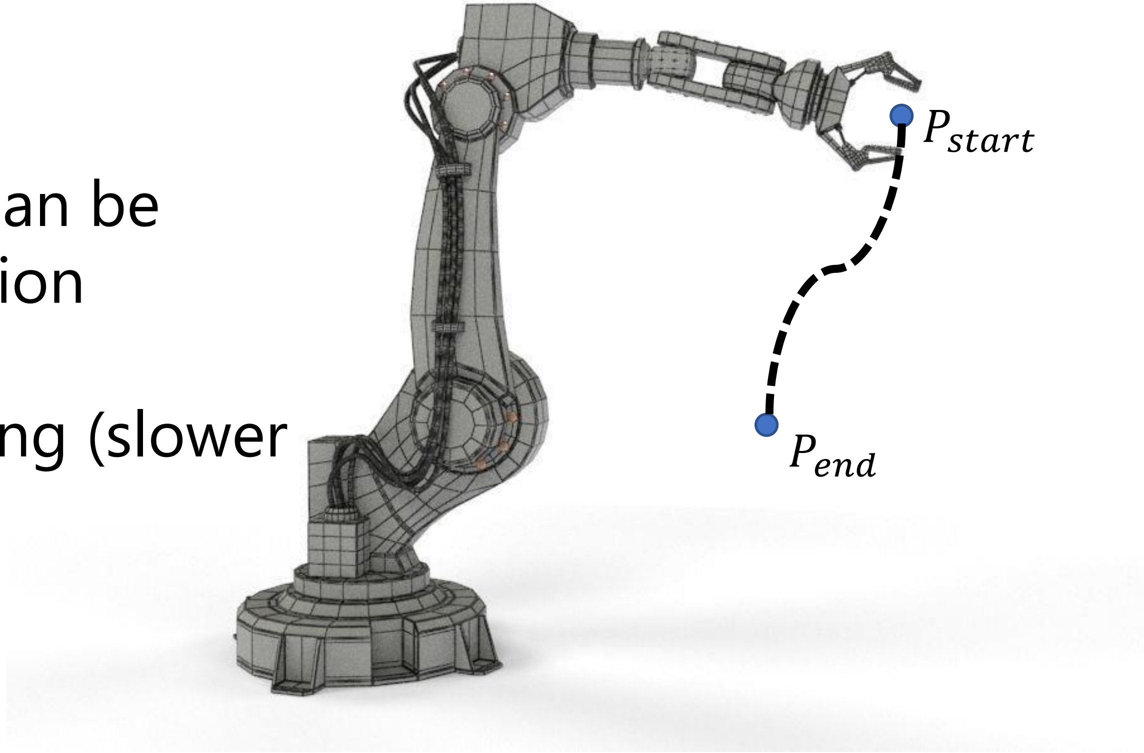
- Only start and stop position programmed and path between points not defined
- Via-points are taught with a teach pendant or programmed
- Advantage: easy control, shortest time to reach point
- Disadvantage: uncontrolled motion



Types of Motion

❑ Continuous Path Control (CPC)

- End points are programmed (usually)
- Path, velocity and acceleration in between can be controlled (e.g. linear, circular) by interpolation
- Similar to CNC controller, real time processing (slower than PTP)



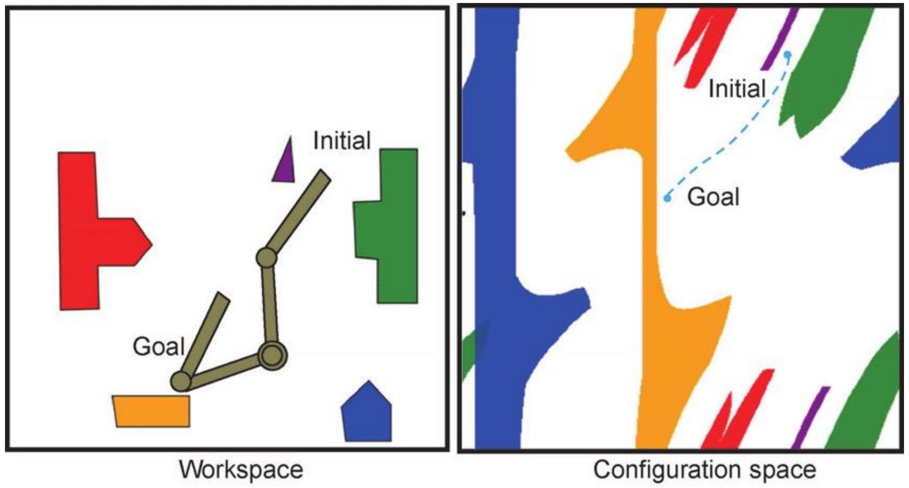
Joint-based Trajectory Generation

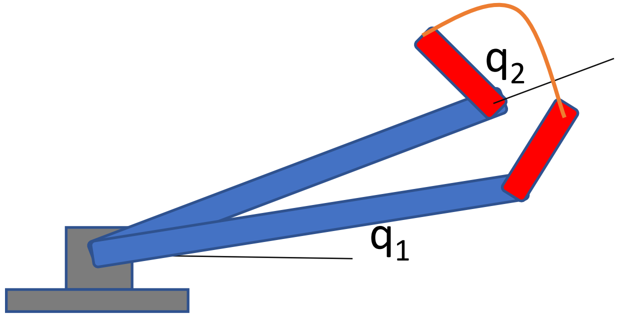
- Desired trajectory available in terms of time histories of **joint** position, velocity and acceleration.

- Errors expressed in joint space are incorporated into the feedback system.

- Computationally efficient

- High sampling frequency





Time-history of joint variables

(q_1,q_2) Start → (q_1,q_2) Goal

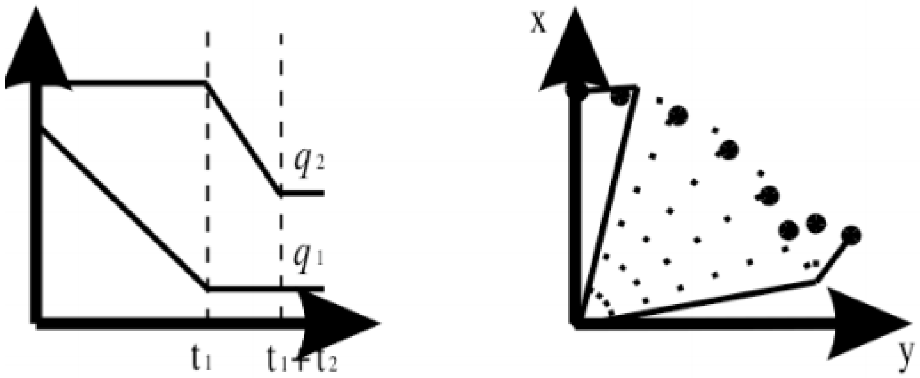
Joint-based Trajectory Generation

(q_1,q_2) Start

↓

(q_1,q_2) Goal

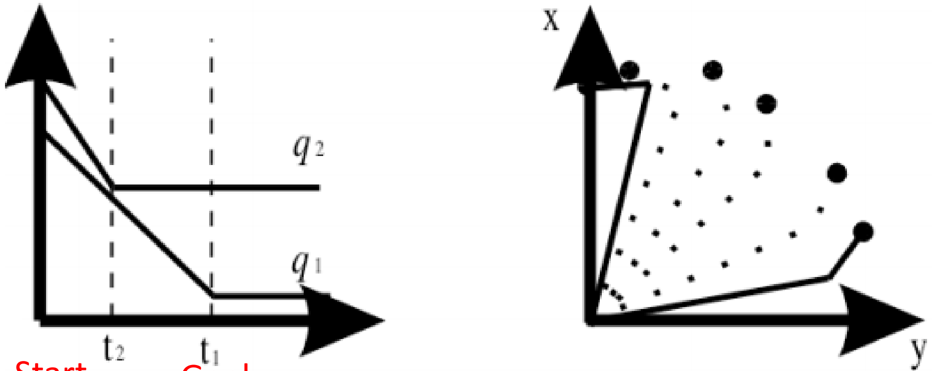
Axis to axis movement



Start Goal

- Move axis 1 first: q1 reaches its goal position
- Then move link 2: q2 reaches its goal position

Simultaneous movement on axes



Start Goal

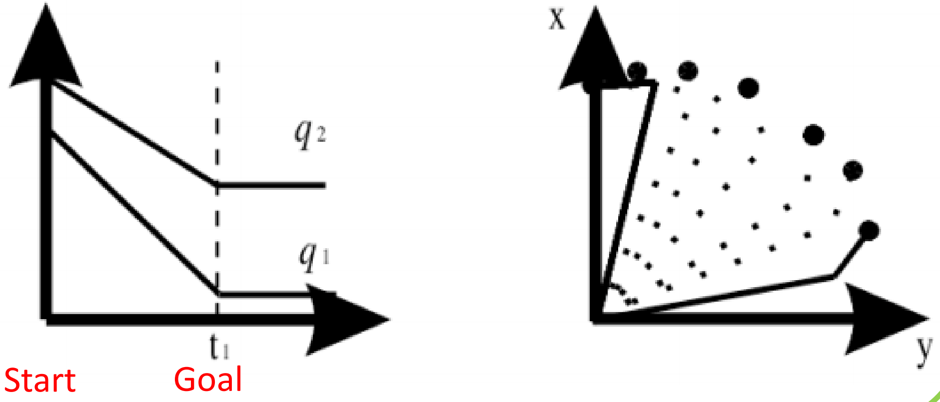
- Move two links at their maximum allowable speeds to reach their goal positions

Joint-based Trajectory Generation

(q1,q2) Start

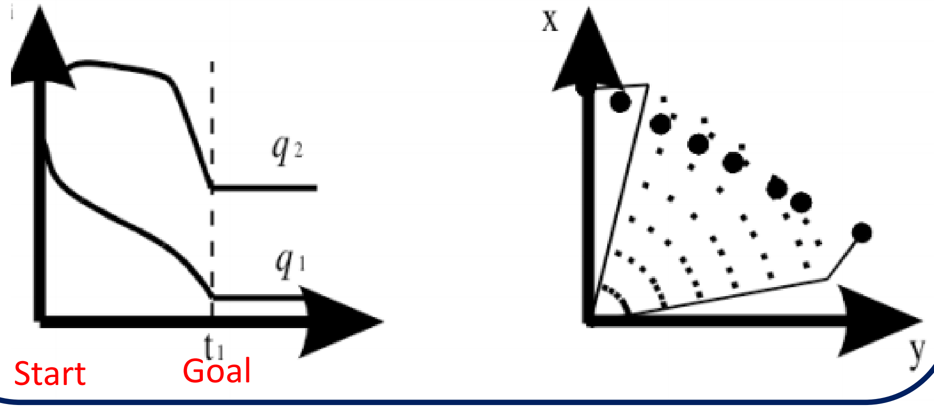
(q1,q2) Goal

Coordinated path



- Calculate the time required for one of the joint angles to reach its goal position (e.g., joint 1)
- Coordinate the other joint's velocity (joint 2) to reach its final position at the same time

Continuous straight path

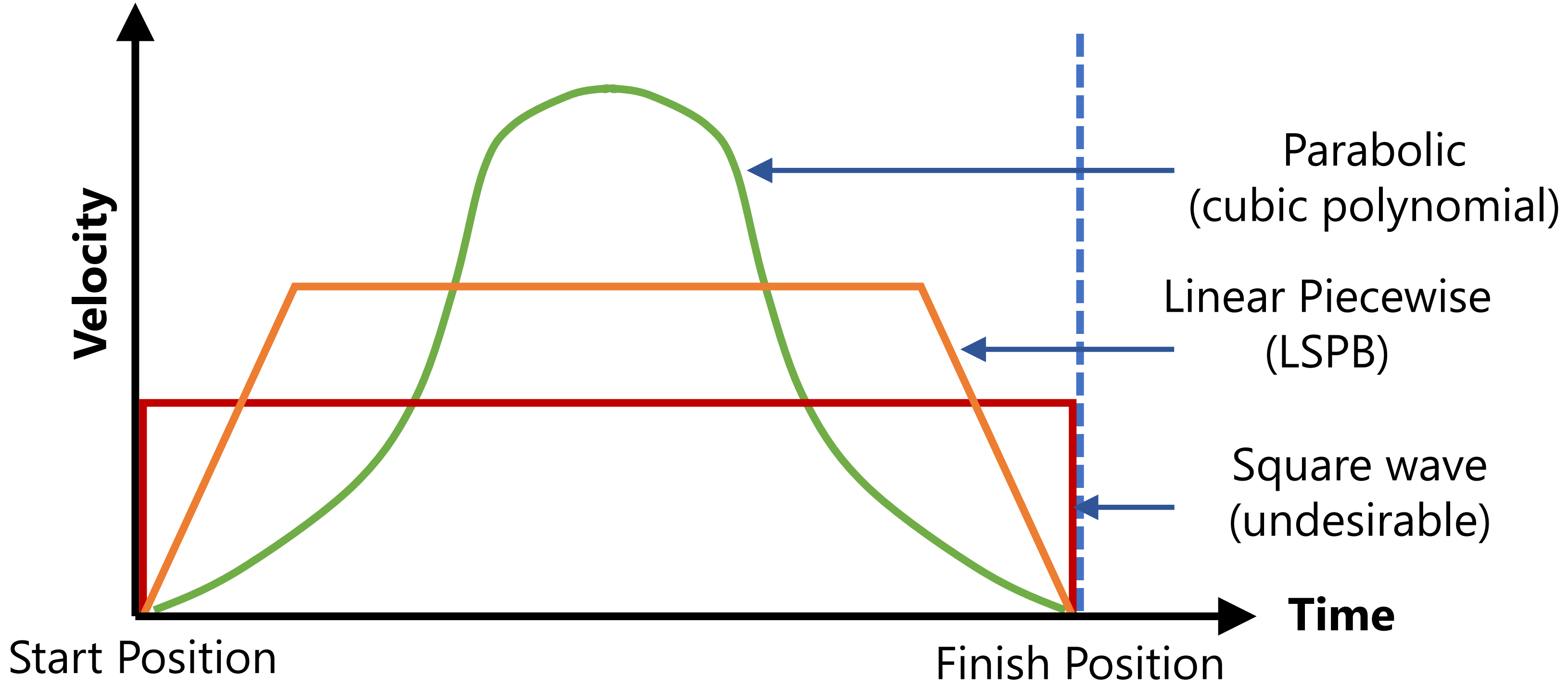


- **moveL**
- Calculate q1 and q2 so that the movement of the end effector is linear

Point-to-point Trajectory

Point-to-point Trajectory

- Used in …. (You tell me!)
  - Predictable environment – absence of obstacles or known location
  - Teach and playback mode (teach pendant)
    - No need for calculation of the forward or inverse kinematics
    - Position recorded as a set of joint angles (e.g. encoder values)



Point-to-point Trajectory

- The trajectory connects an initial to a **final configuration** while satisfying other specified constraints at the endpoints

- Initial constraints (at time t_0), the i^{th} joint variable satisfies

$$\boxed{\begin{aligned} q_i(t_0) &= q_0 \\ \dot{q}_i(t_0) &= \dot{q}_0 \end{aligned}}$$

- Endpoint constrains (at time t_f),

$$\boxed{\begin{aligned} q_i(t_f) &= q_f \\ \dot{q}_i(t_f) &= \dot{q}_f \end{aligned}}$$

This is a problem involving four constraints

Cubic Polynomial Trajectory

- A polynomial with four independent coefficients can satisfy these four constrains

- A cubic trajectory has four independent coefficients

$$q_i = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

- The time derivative results in,

$$\dot{q}_i = a_1 + 2a_2 t + 3a_3 t^2$$

Initial constrains

$$q_0 = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3$$
$$\dot{q}_0 = a_1 + 2a_2 t_0 + 3a_3 t_0^2$$

Final constrains

$$q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3$$
$$\dot{q}_f = a_1 + 2a_2 t_f + 3a_3 t_f^2$$

Cubic Polynomial Trajectory

$$\begin{cases} q_0 &= a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \\ \dot{q}_0 &= a_1 + 2a_2 t_0 + 3a_3 t_0^2 \\ q_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ \dot{q}_f &= a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{cases}$$

- We can form the following matrix equations

$$\begin{pmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} q_0 \\ \dot{q}_0 \\ q_1 \\ \dot{q}_1 \end{pmatrix}$$

- Assume t_0=0,

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} q_0 \\ \dot{q}_0 \\ q_1 \\ \dot{q}_1 \end{pmatrix}$$

Cubic Polynomial Trajectory

- Assume $t_0 = 0$ results in,

$$a_0 = q_0;$$
$$a_1 = \dot{q}_0;$$
$$a_2 = \frac{3(q_1 - q_0) - (2\dot{q}_0 + \dot{q}_1)t_f}{t_f^2};$$
$$a_3 = \frac{2(q_0 - q_1) - (\dot{q}_0 + \dot{q}_1)t_f}{t_f^3}$$

- If $t_0 \neq 0$ then we simply shift the time axis to the left by t_0
- $t = t - t_0$ and $t_f = t_f - t_0$

Example 1 - Cubic Polynomial Trajectory

- Suppose $t_0 = 0$ and $t_f = 1s$, with $\dot{q}_0 = 0$ and $\dot{q}_f = 0$. Find the corresponding cubic trajectory.
In other words, we want to move from q_0 to q_f, and we start and end with zero velocity (Just use the previous slide, no need for MATLAB)

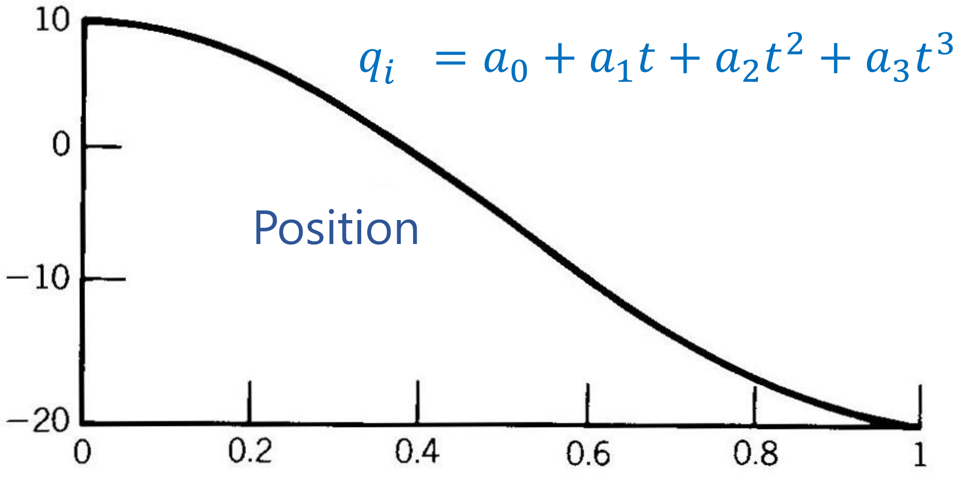
<u>Answer</u>

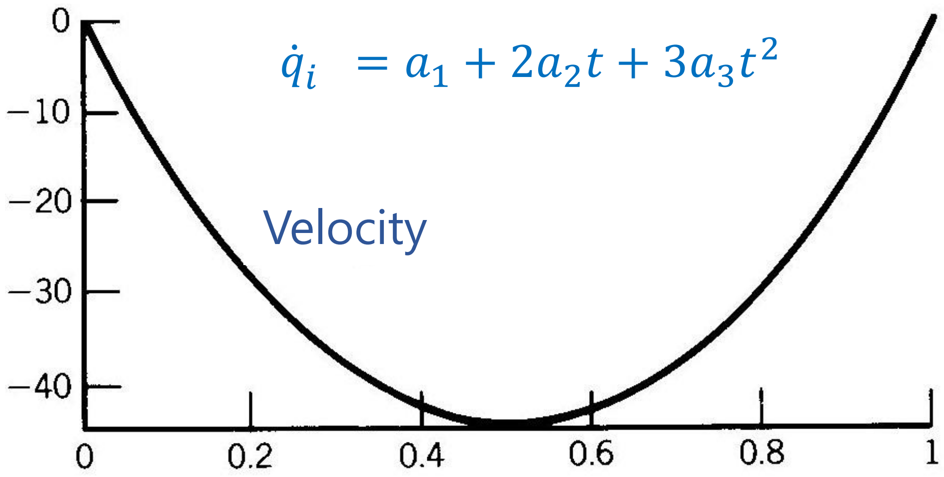
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} q_0 \\ 0 \\ q_f \\ 0 \end{pmatrix} \Rightarrow \begin{aligned} a_0 &= q_0 \\ a_1 &= 0 \\ a_2 &= 3(q_f - q_0) \\ a_3 &= -2(q_f - q_0) \end{aligned}$$

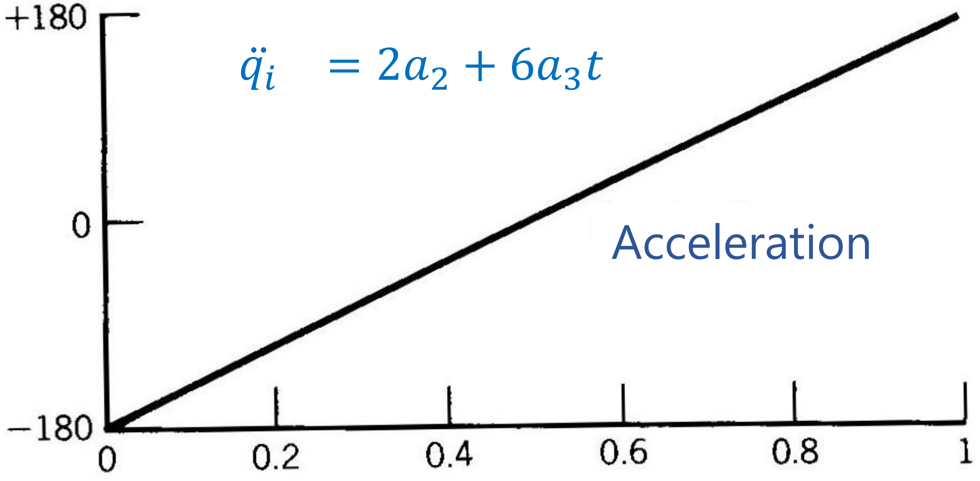
- And the required cubic polynomial is,

$$q_i(t) = q_0 + 3(q_f - q_0)t^2 - 2(q_f - q_0)t^3$$

Cubic Polynomial Trajectory







- Discontinuities in the acceleration at the start and the end point
- The derivative of acceleration is called the jerk

Can anyone propose a solution?

-> Use Quintic polynomials

(II) Quintic polynomial Trajectories

- ❑ Use Quintic polynomial for six constrains (positions, velocity, acceleration at initial and end point)

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

⇩

Constrain

$$\begin{aligned}
q_0 &= a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5 \\
v_0 &= a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4 \\
\alpha_0 &= 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3 \\
q_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \\
v_f &= a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \\
\alpha_f &= 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3
\end{aligned}$$

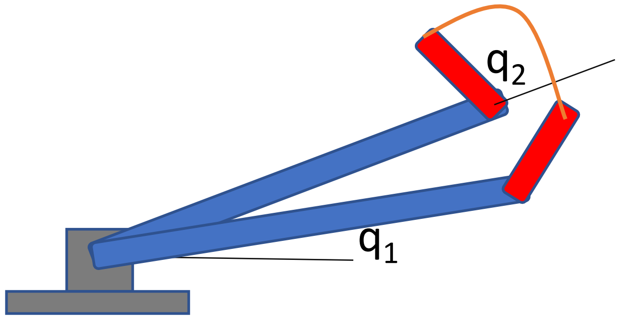
(II) Quintic polynomial Trajectories

- ❑ Use Matlab to calculate the 6 coefficients

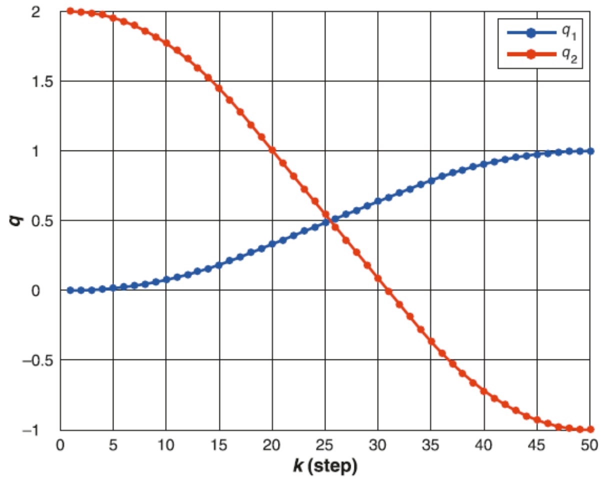
$$\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} q_0 \\ v_0 \\ \alpha_0 \\ q_f \\ v_f \\ \alpha_f \end{bmatrix}$$

(II) Quintic polynomial Trajectories

- RVC Toolbox
  - The Toolbox function tpoly generates a quintic polynomial trajectory
  - [trajectory, velocity, acceleration] = tpoly(q_i, q_f, time) %%% for a single joint
  - For multiple-joint robot: mtraj
  - E.g., for a two revolute-joint robot: mtraj(@tpoly, [0 2], [1 -1], 50);

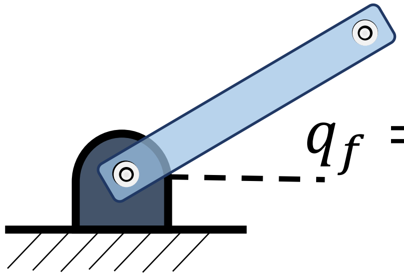


More general form: tpoly(q_i, q_f, time, V_i , V_f)

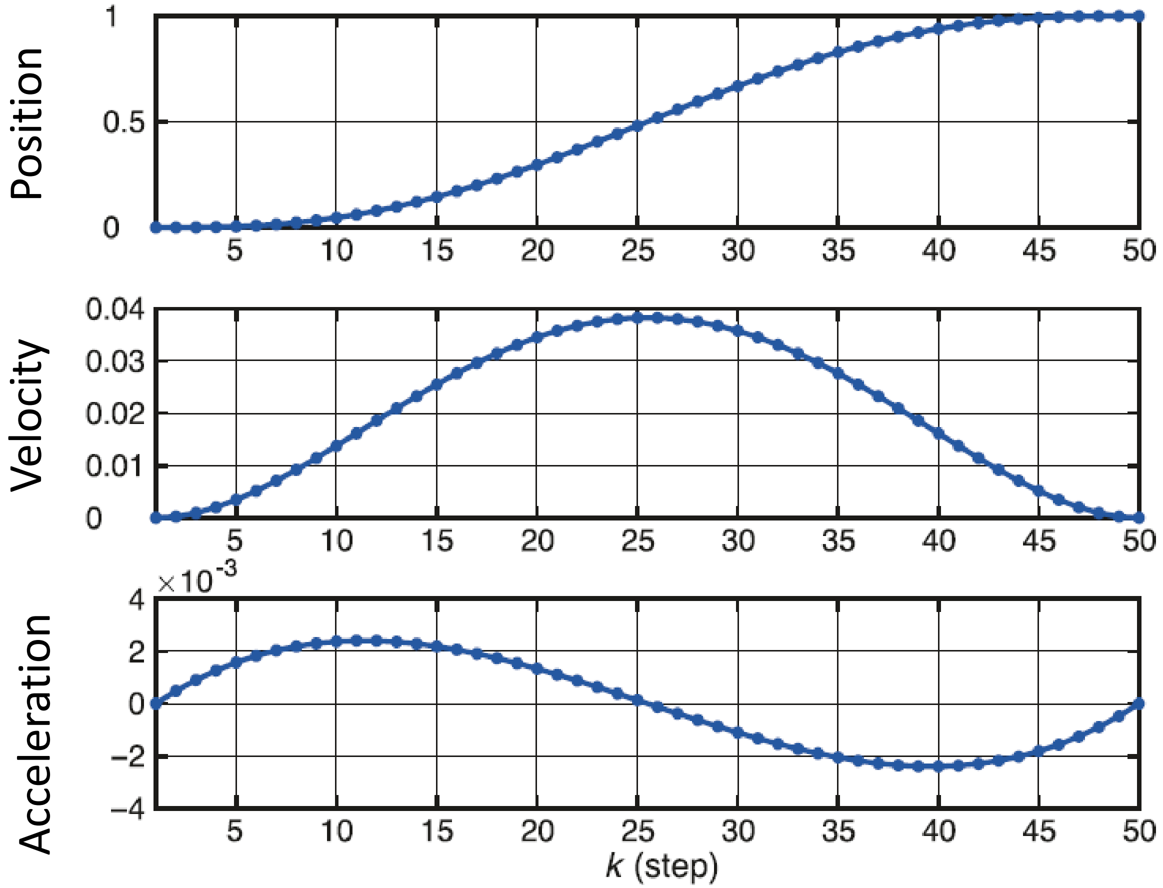


Example 2: Quintic polynomial Trajectories

- Find the trajectory for the joint q to move from 0 (rad) to 1(rad) in 5 (s) with zero velocity boundary conditions

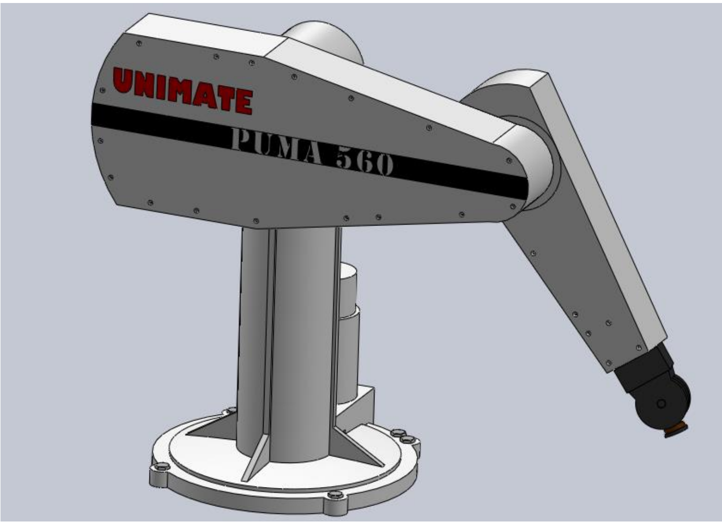


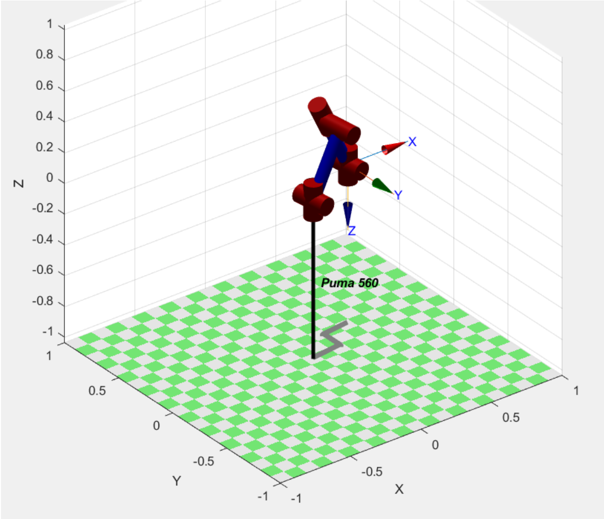
$$q_f = 1\ (rad)$$



Example 3: Quintic polynomial Trajectories for 6DOF robot

- ❑ PUMA560





Transformation matrix

T1 =

$$\begin{matrix} 1.0000 & 0 & 0 & 0.4000 \\ 0 & -1.0000 & 0 & 0.2000 \\ 0 & 0 & -1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \end{matrix}$$

T2 =

$$\begin{matrix} 1.0000 & 0 & 0 & 0.4000 \\ 0 & 0 & -1.0000 & -0.2000 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 \end{matrix}$$

Pose 1: Translate by (0.4, 0.2, 0) and rotate by π about x-axis
Pose 2: Translate by (0.4, -0.2, 0) and rotate by π/2 about x-axis
Generate a quintic trajectory for the end effector to travel from pose 1 to pose 2 in 5(s)

Example 3: Quintic polynomial Trajectories for 6DOF robot

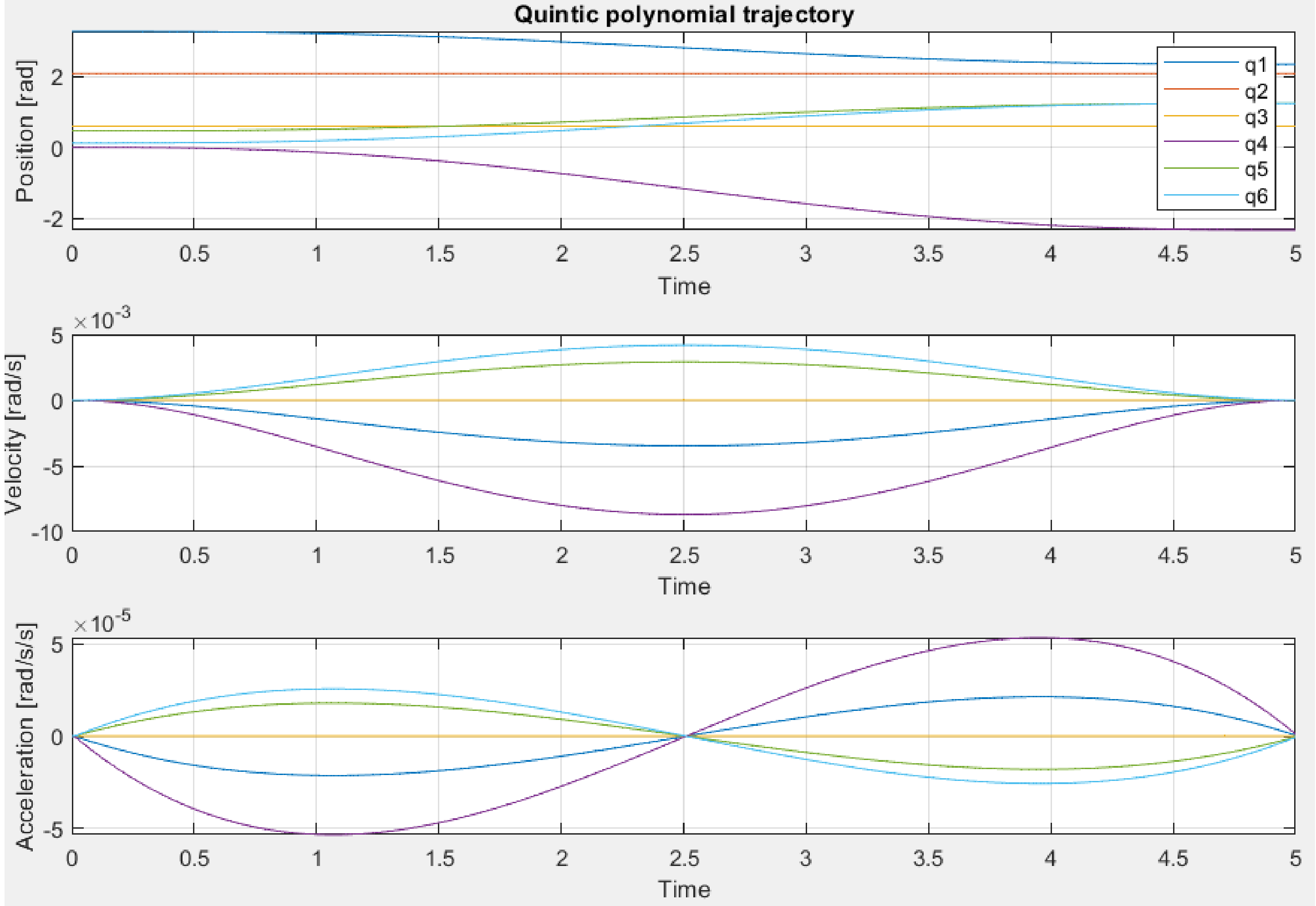
❑PUMA560

- Step 1: Convert Poses 1 and 2 into joint variable $(q_1, q_2, q_3, q_4, q_5, q_6)$: Inverse kinematic (ikine)

- Step 2: Generate the quintic polynomial trajectory for all joints using

  s= mtraj(@tpoly, pose1, pose2, t)

Example 3: Quintic polynomial Trajectories for 6DOF robot



Convert these joint space variables into workspace configurations

T1 =

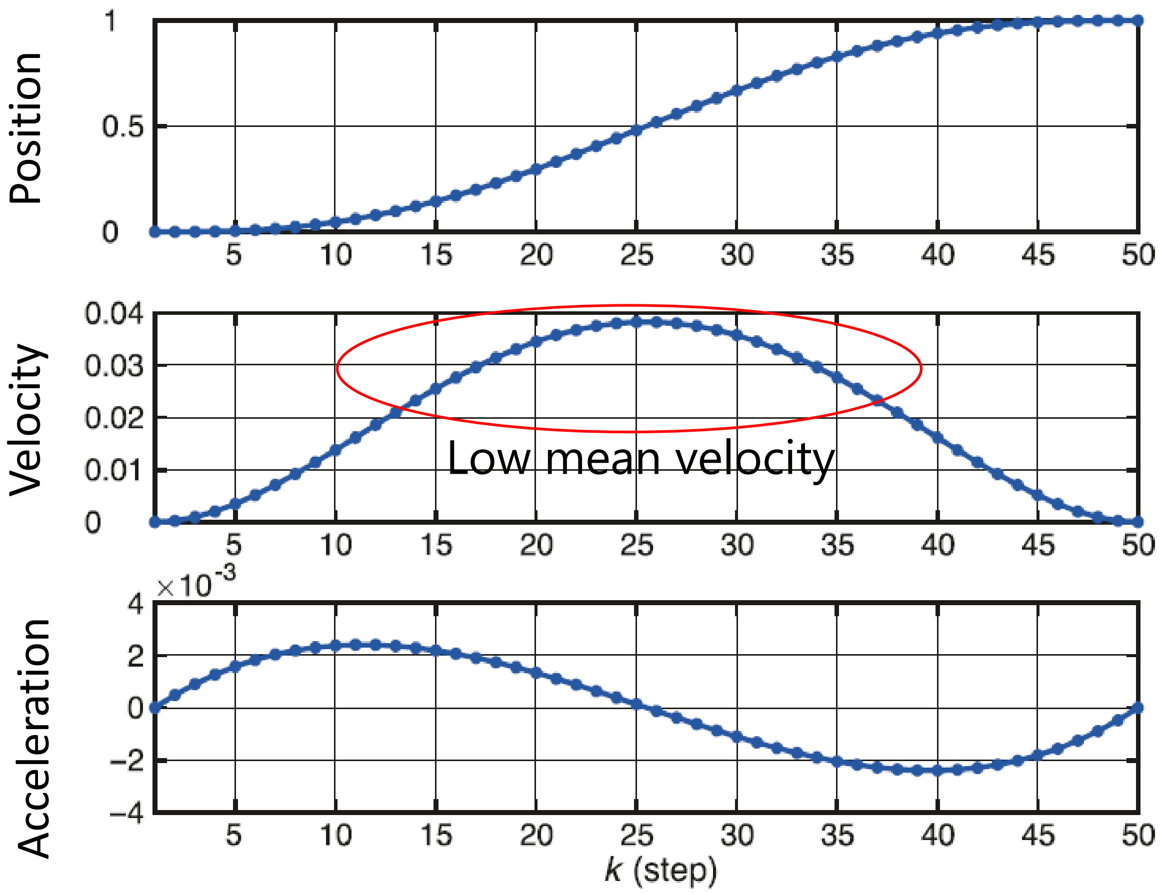
$$\begin{matrix} 1.0000 & 0 & 0 & 0.4000 \\ 0 & -1.0000 & 0 & 0.2000 \\ 0 & 0 & -1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \end{matrix}$$

T2 =

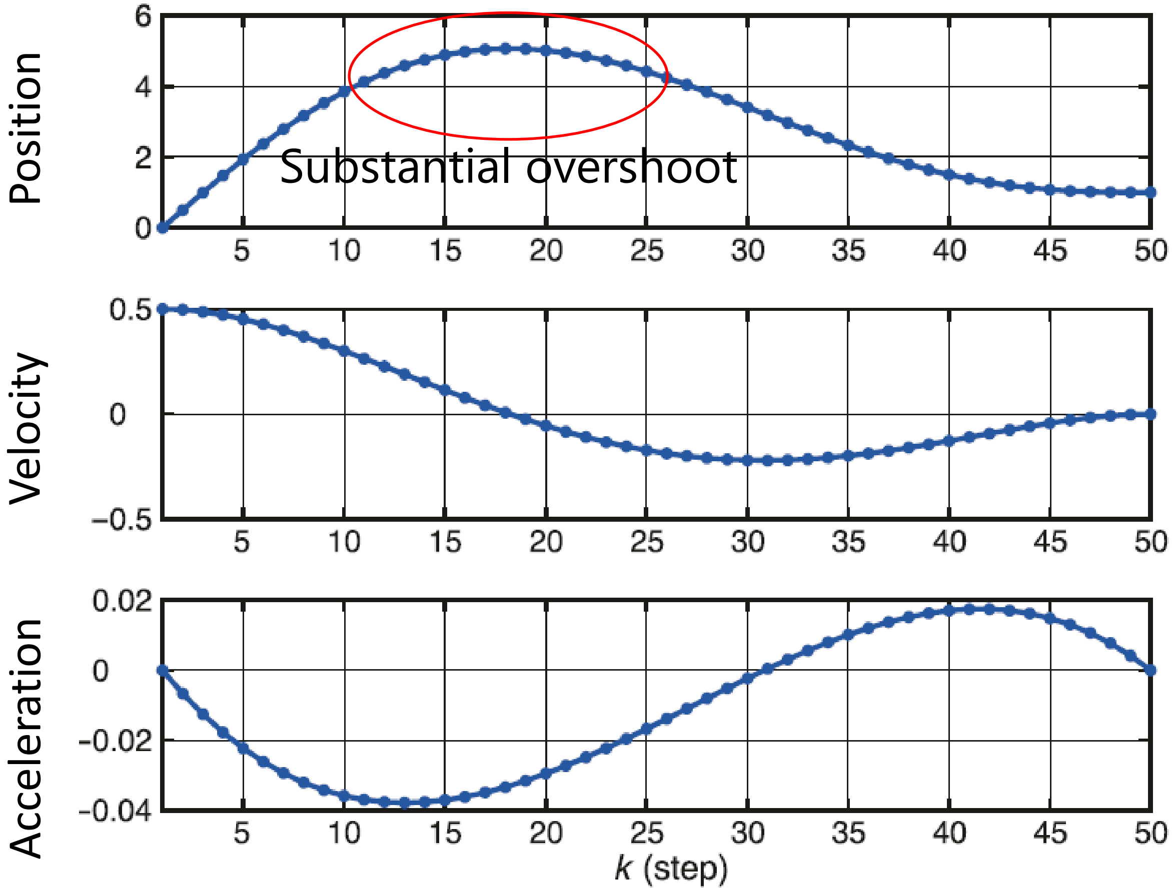
$$\begin{matrix} 1.0000 & 0 & 0 & 0.4000 \\ 0 & 0 & -1.0000 & -0.2000 \\ 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 \end{matrix}$$

Problems with Quintic

- Trajectory from 0 to 1 with zero velocity boundary conditions

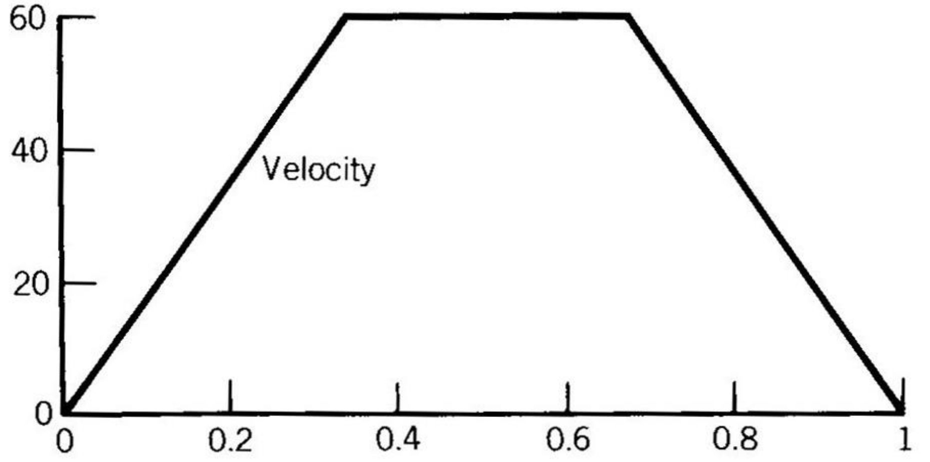


- Trajectory from 0 to 1 with an initial velocity of 5(rad/s) and a final velocity of 0



(III) Linear Segments with Parabolic Blends LSPB

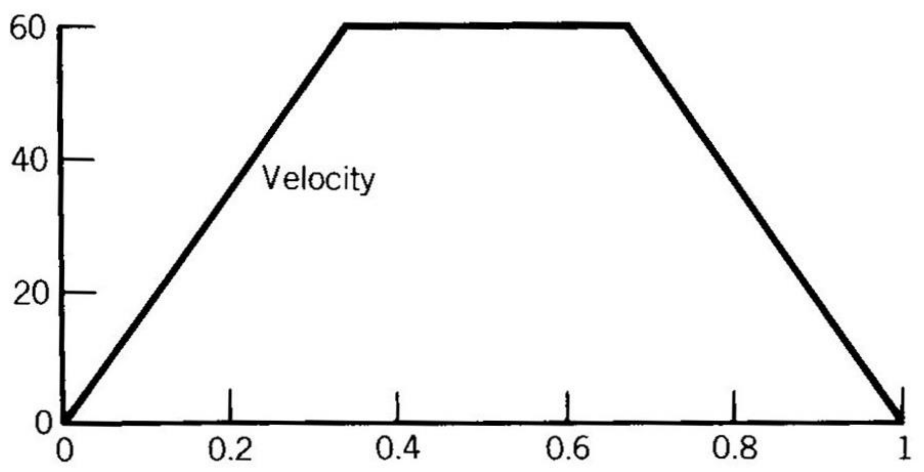
- Revision: What is a parabola?
- Combine linear and parabolic behavior in the robot trajectory
- Often we would like a **constant joint velocity** over **a portion of the trajectory**.
  - Velocity is "ramped up" to its constant set point and then "ramped down" at the final position.
  - This results in a **trapezoidal velocity** profile

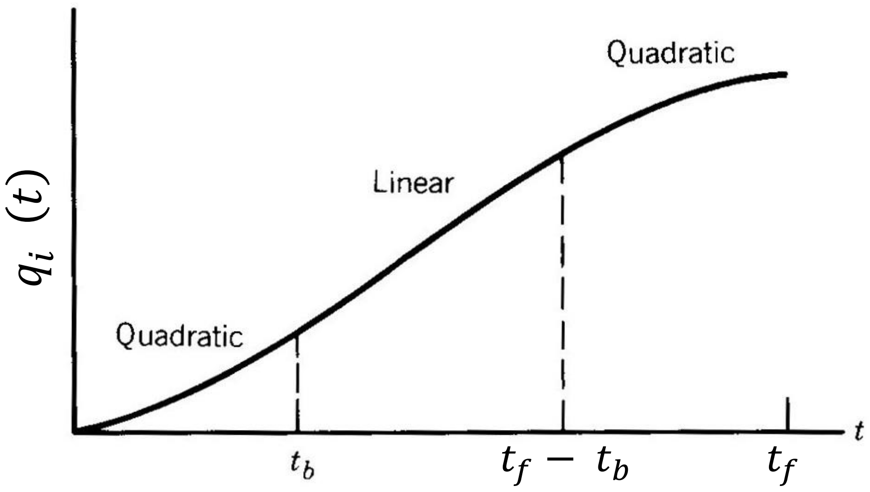


(III) Linear Segments with Parabolic Blends LSPB

❑ Solution: **Linear Segments with Parabolic Blends (LSPB)**

- For the period t_0 to t_b and $t_f - t_b$ to t_f we apply a quadratic polynomial
- In between the velocity profile is linear constant
- Given: q_0, q_f, t_0, t_f, V

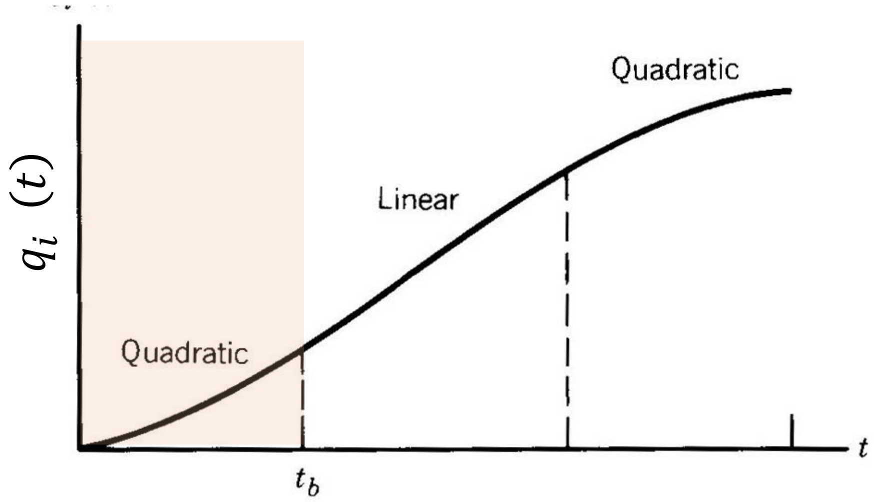




t_b is the blend time

LSPB Formulation

- For convenience, we suppose $t_0 = 0$ and $\dot{q}_0 = 0$
- Between times 0 and t_b (RED FOR UNKNOWN)



$$q_i\ (t) = a_0 + a_1 t + a_2 t^2$$
$$\dot{q}_i\ (t) = a_1 + 2a_2 t$$

- Applying $\dot{q}_0 = 0$, we get,

$$a_0 = q_0$$
$$a_1 = 0$$

- At time t_b, we want the velocity to be equal to a given constant V:

$$\dot{q}_i\ (t) = 2a_2 t_b = V \Rightarrow a_2 = V/2t_b$$

LSPB Formulation

- Hence the required trajectory between 0 and t_b becomes,

  - Position

$$q_i\ (t) = q_0 + \frac{V}{2t_b}\ t^2$$

  - Velocity

$$\dot{q}_i\ (t) = \frac{Vt}{\mathrm{t_b}}$$

  - Acceleration

$$\ddot{q}_i\ (t) = \frac{V}{t_b}$$

LSPB Formulation

- Between time t_b and $t_f - t_b$, the trajectory is a linear segment (i.e., constant velocity)

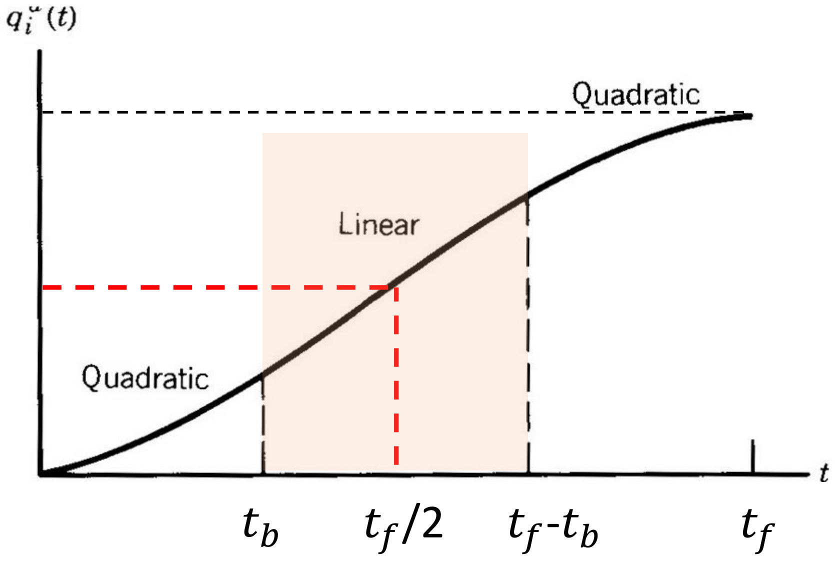
$$q_i\ (t) = b_0 + Vt$$

- Since symmetry:

$$\boxed{q_i\ (t_f/2) = \ (q_0 + q_f)/2}$$

Sub with $\frac{t_f}{2}$

$$\boxed{b_0 + Vt_f/2 = (q_0 + q_f)/2}$$



- Compare two equations:

$$b_0 = \frac{q_0 + q_f - Vt_f}{2}$$

$$t_b = ?$$

LSPB Formulation

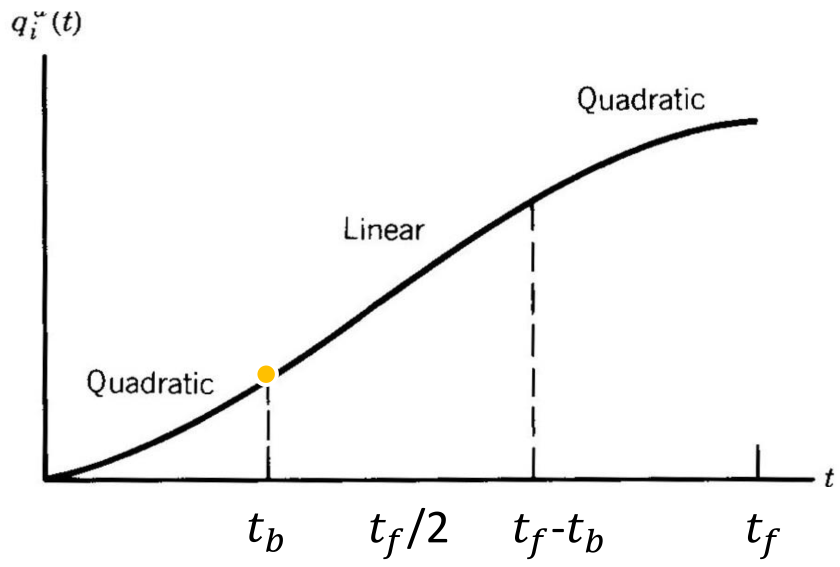
- Since the two segments must blend at time t_b:

$$q_0 + Vt_b/2 = \frac{q_0+q_f-Vt_f}{2} + Vt_b$$

- Solving for the blend time t_b:

$$\boxed{t_b = \frac{q_0-q_f+Vt_f}{V}} \text{ and } 0 < t_b \le t_f/2$$

Rearranging: $\boxed{\frac{q_f-q_0}{t_f} < V \le \frac{2(q_f-q_0)}{t_f}}$



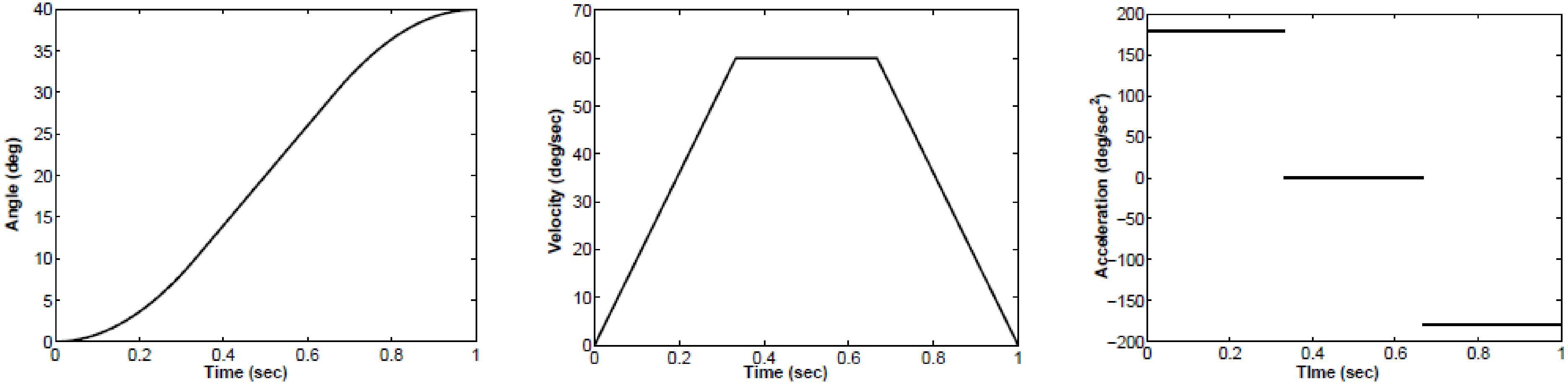
The specified velocity must lie within these limits or motion is not possible.

LSPB Formulation

- The complete LSPB trajectory is given by:

$$q_i\ (t) = \begin{cases} q_0 + \dfrac{at^2}{2} & ;\ 0\ \leq t\ \leq t_b \\ \dfrac{q_f + q_0 - Vt_f}{2} + Vt & ;\ t_b < t \leq t_f - t_b \\ q_f - \dfrac{at_f^2}{2} + at_f t - \dfrac{at^2}{2} & ;\ t_f - t_b < t\ \leq t_f \end{cases}$$

Where $a = \dfrac{V}{t_b}$



Example 4: Trapezoidal trajectory

- The initial and end positions of a joint are 0 (rad) and 1 (rad), respectively. Assume that the initial and end velocity are 0 (rad/s) and the travelling time is 18 (s).

a) Generate a trapezoidal trajectory where t_b =6 (s)
b) Generate a trapezoidal trajectory V = 0.06 (rad/s)
c) Repeat (b) with V= 0.02 (rad/s)
d) Repeat (b) with V= 0.11 (rad/s)

Answer

a) lspb(start, goal, time); blend at t_b = time/3 (by default)

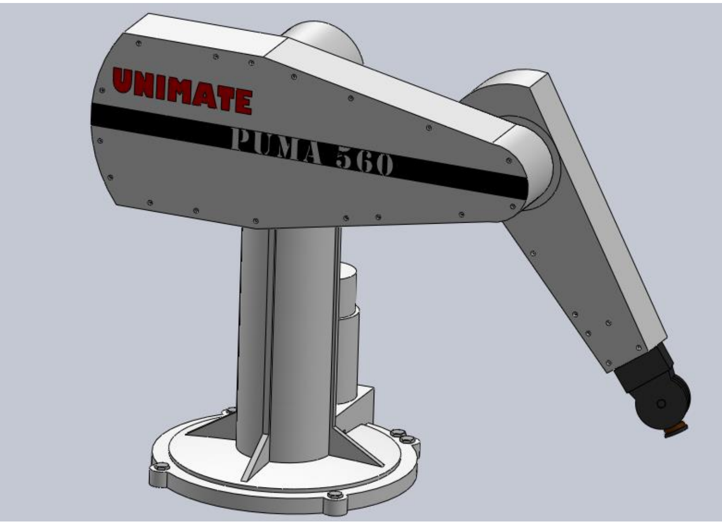
b) c) d) lspb(start, goal, time, V)

If you specify V, you can't specify t_b

$$\frac{q_f - q_0}{t_f} < V \leq \frac{2(q_f - q_0)}{t_f}$$

Homework: Trapezoidal Trajectories for 6DOF robot

❑ PUMA560



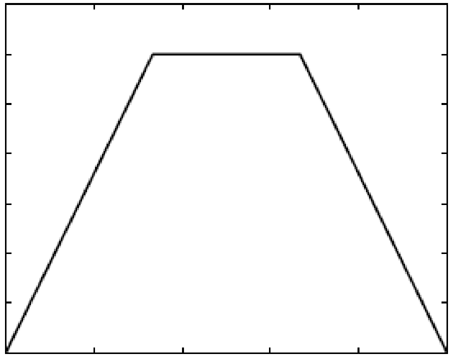
Pose 1: Translate by (0.4, 0.2, 0) and rotate by π about x-axis
Pose 2: Translate by (0.4, -0.2, 0) and rotate by $\pi/2$ about x-axis
Generate a trapezoidal trajectory for the end effector to travel from pose 1 to pose 2 in 5 (s) and the blend time t_b = 5/3 (s).

(IV) Minimum Time Trajectory – Bang-Bang trajectory

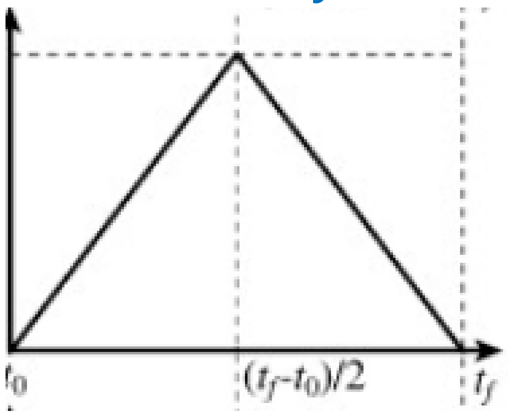
- What would "minimum time" look like?

- Fastest trajectory between q_0 and q_f with a given constant (maximum) acceleration a

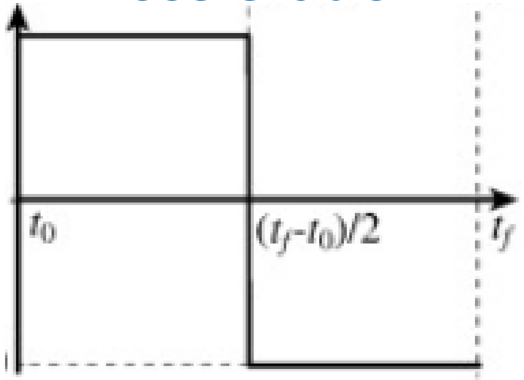
- The switching time t_s at which time acceleration switches to $-a$ (maximum deceleration) is $t_s = t_f/2$



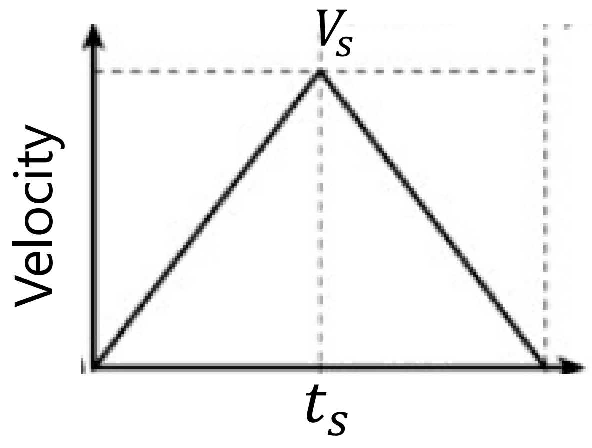
Velocity



Acceleration



Bang-Bang Trajectory



- Acceleration is constant

$$\ddot{q}_i = a \qquad 0 \le t \le t_s$$

- Therefore

$$\dot{q}_i = at + \cancel{\dot{q}_0} \qquad 0 \le t \le t_s$$

- Therefore

$$q_i = \frac{1}{2}at^2 + q_0 \qquad 0 \le t \le t_s$$

- We know that at t_s, $q_i = q_0 + \frac{q_f - q_0}{2}$. Therefore:

$$q_0 + \frac{q_f - q_0}{2} = \frac{1}{2}at_s^2 + q_0$$

$$\boxed{t_s = \sqrt{\frac{q_f - q_0}{a}}}$$

Trajectories for Paths Specified by Via Points

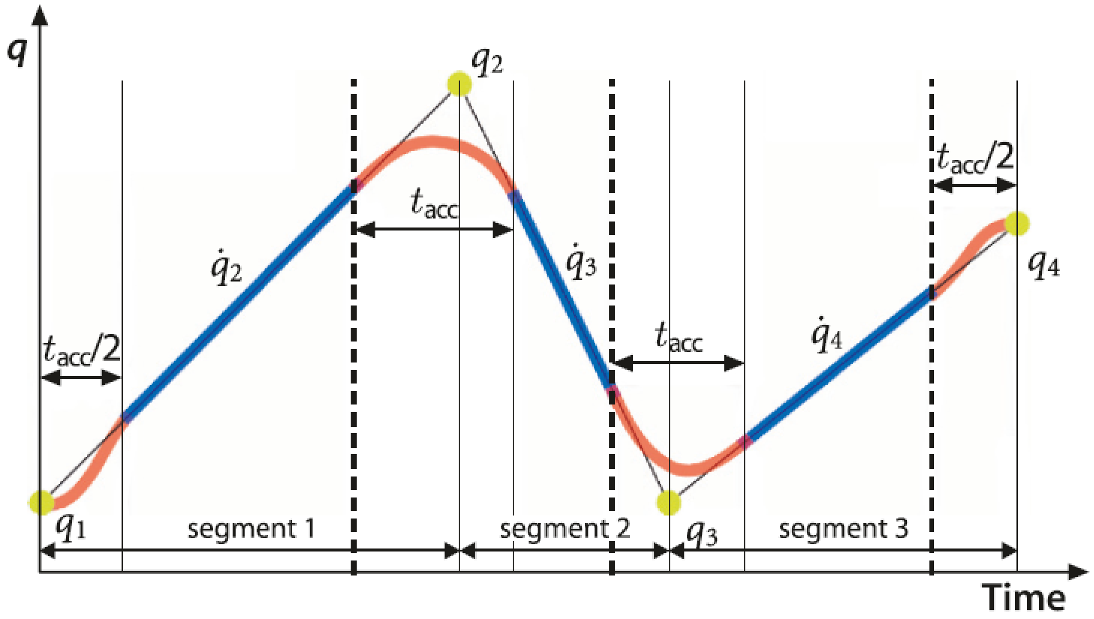
Trajectory with Via-points

- ❑ Compute the cubics that connect the via point values in a smooth way (instead of stopping and starting at each via-point)

  - Now the velocity constraints at end-points are non-zero
  - Velocities can be user specific based on some heuristics

- ❑ Multiple options for trajectories

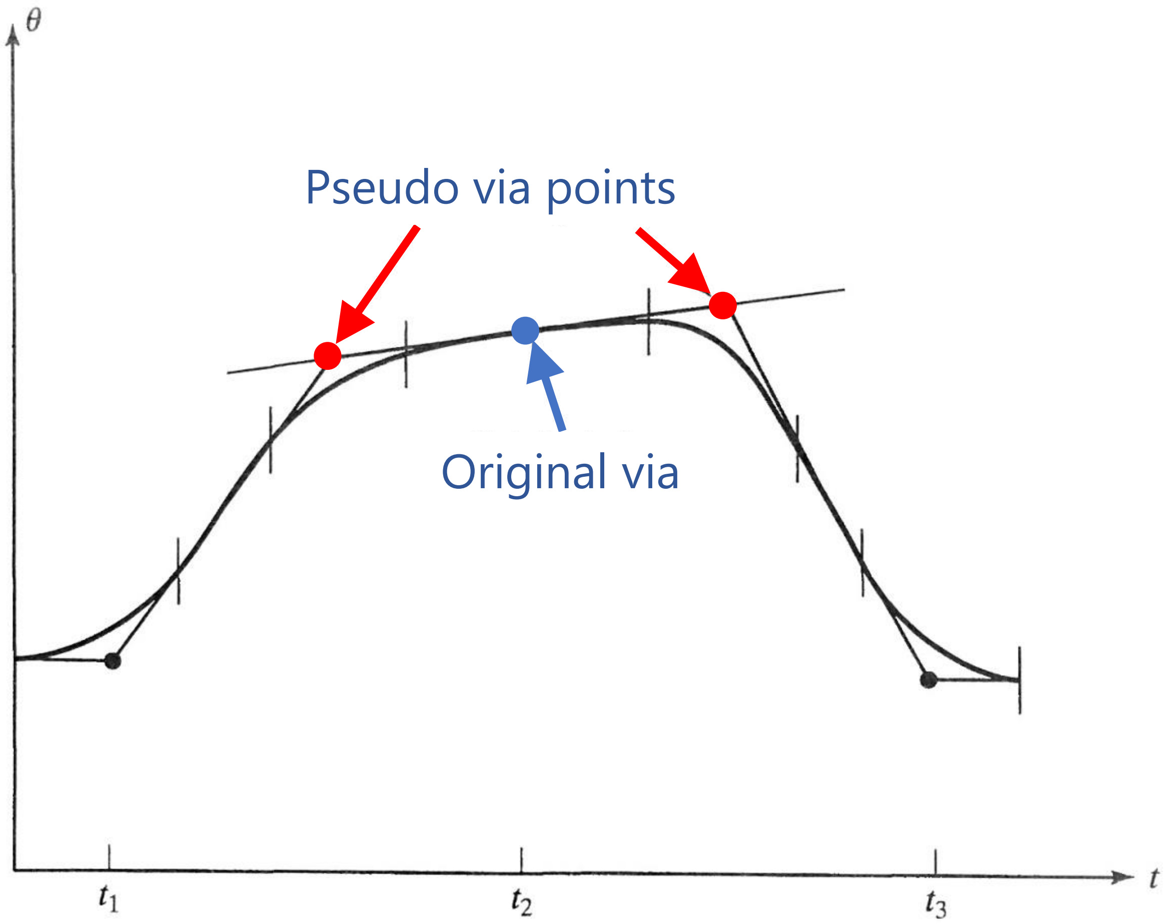
- ❑ Each via point may not actually be reached.



Pseudo Via-points

If the path is required to pass exactly through a via point, with a specific velocity, add two pseudo via-points so the original via-point is on the linear segment of the path.

The interested reader should research Bezier Curves



Comments on Trajectory Generation

- All of the above trajectory generation schemes ignore the actual dynamics of the system and provide a purely kinematic solution

- Limits on acceleration were included, but they must be chosen to be well below the physically obtainable values

- All of the above assumed that collision free paths exist

Manipulability

Well-conditioned Workplace

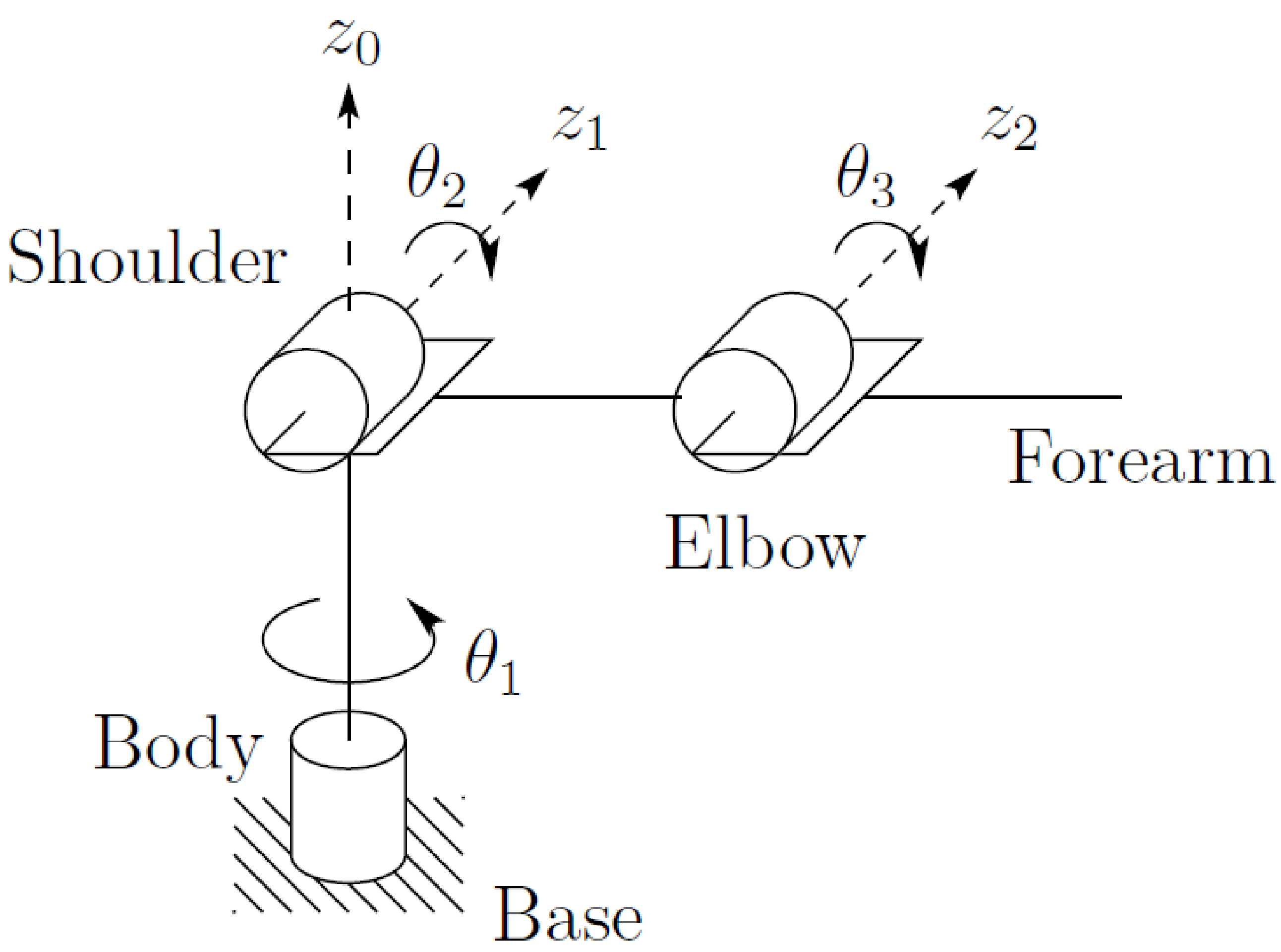
- Near singularities ($\det J\,(\mathbf{q}) = 0$), the workspace can be described as poorly conditioned

- Yoshikawa defined a measure of manipulability as,
  - $m = \sqrt{\det(J(\mathbf{q})J(\mathbf{q})^T)}$

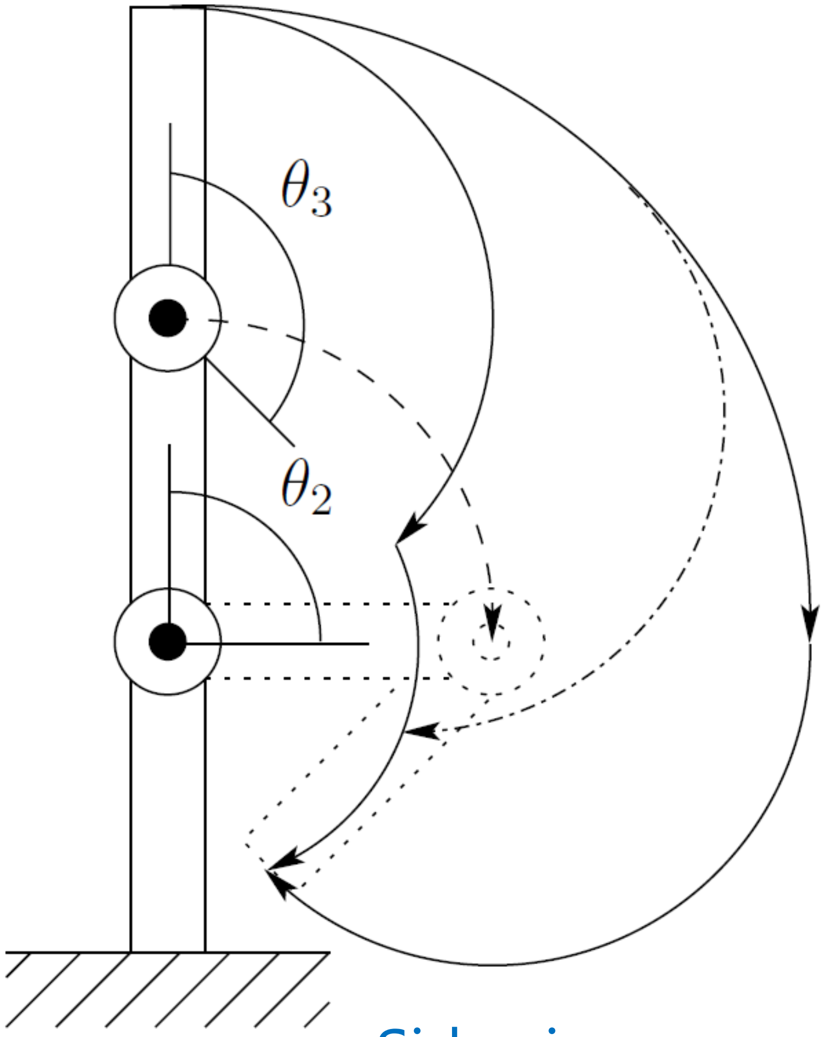
which for a non-redundant manipulator, reduces to,
  - $m = |\det(J(\mathbf{q}))|$

- Other measures of manipulability exist (eg. Asada)

Manipulability

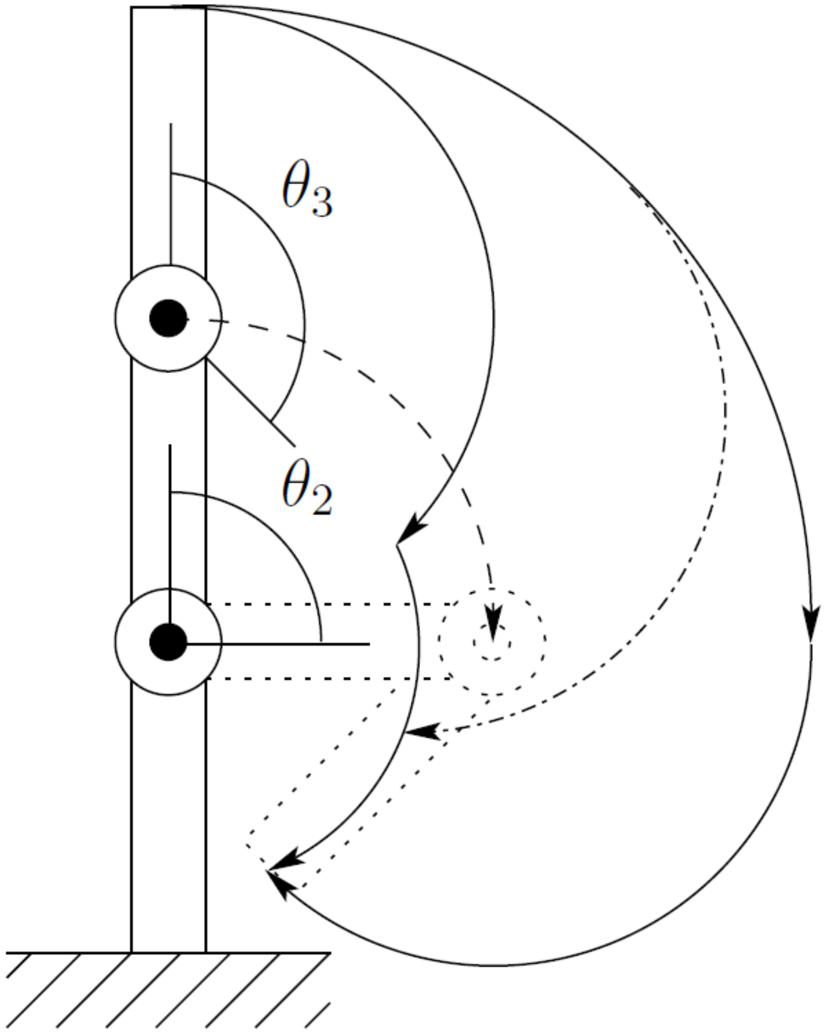


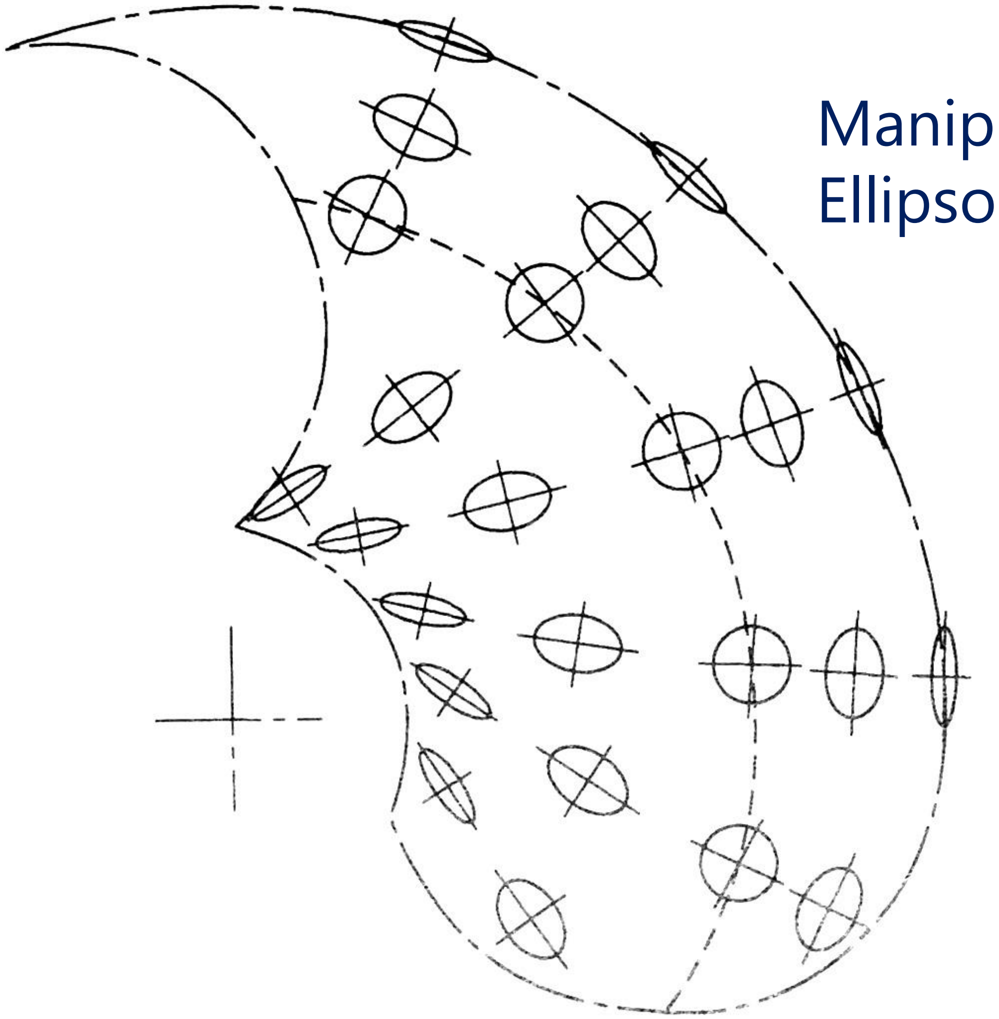
RRR (Elbow) manipulator



Side view

Manipulability





Manipulability Ellipsoid

Example 5 – Manipulability of two link planar arm

The link lengths of a two-link planar arm are a_1 and a_2, respectively. Find the joint variables θ_1 and θ_2 to achieve the maximum manipulability

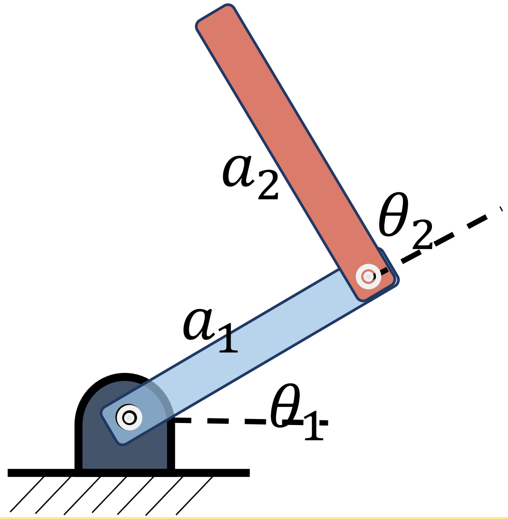
Answer

$$J = \begin{pmatrix} -a_1 sin\theta_1 - a_2\sin(\theta_1+\theta_2) & -a_2\sin(\theta_1+\theta_2) \\ a_1 cos\theta_1 + a_2\cos(\theta_1+\theta_2) & a_2\cos(\theta_1+\theta_2) \end{pmatrix}$$

Under-actuated Square it up by deleting some rows of the Jacobian (Lecture 5)

$$\det(J) = a_1 a_2 \sin(\theta_2)$$

$\max(\det(J))$ -> $\max(\sin(\theta_2))$; therefore $\theta_2 = \pm\pi/2$



Accuracy and Repeatability

Accuracy and Repeatability

❑ Spatial Resolution

- Robot joints work on increments (linear or rotational, Basic Length Unit)
- Total of all joint resolutions at end of arm

❑ Accuracy

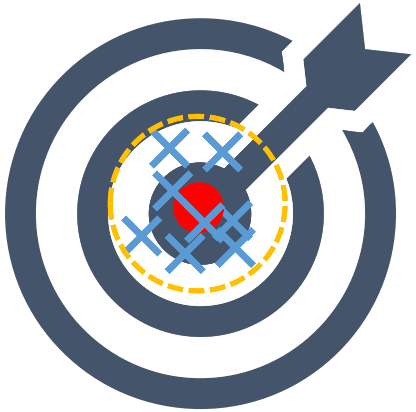
- Deviation between target point and real position
- A function of spatial resolution + control algorithm+ mechanical inaccuracies

❑ Repeatability

- Radius of smallest circle around all end-effector positions (Tool Center Point)
- Statistical term for robot ability to return to the same position repeatedly
- Resolution + all errors of total robot system

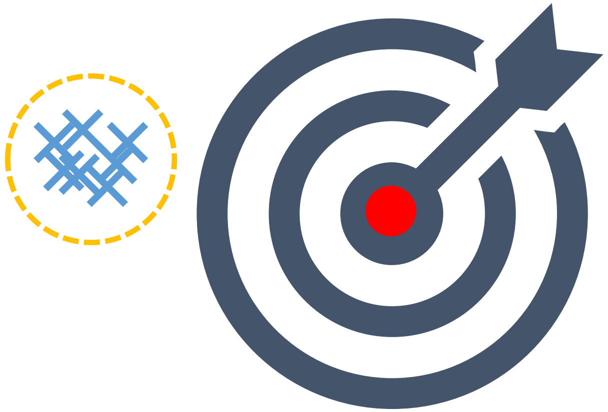
Movement	UR5e
Pose Repeatability	+/- 0.03 mm, with payload, per ISO 9283

Accuracy and Repeatability

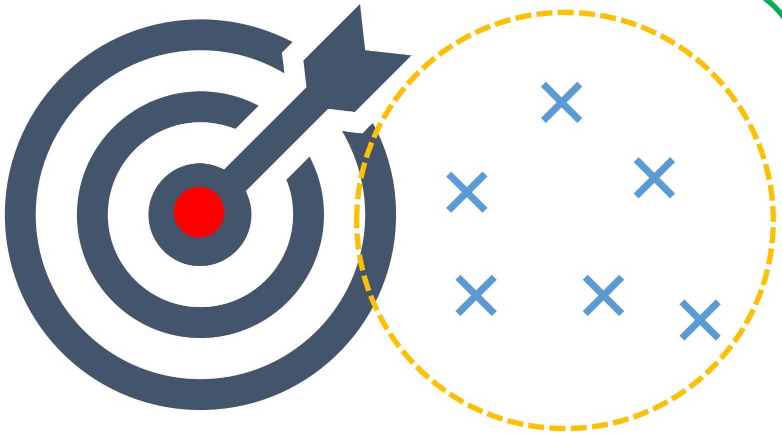


High accuracy + high repeatability

High accuracy + Low repeatability



Low accuracy + high repeatability



Low accuracy + Low repeatability

Machine Accuracy Considerations

❑ System Resolution (SR)

- Programming Resolution (PR)

PR = $L/(2^B-1)$, where L=length of prismatic axis, and B=number of control bits

- Basic Length Unit (BLU) (e.g., $\mathrm{BLU} \leq 0.01$ mm for typically for CNC tools)

❑ Control Resolution (CR)

- Smallest change in position that the feedback sensor can sense.

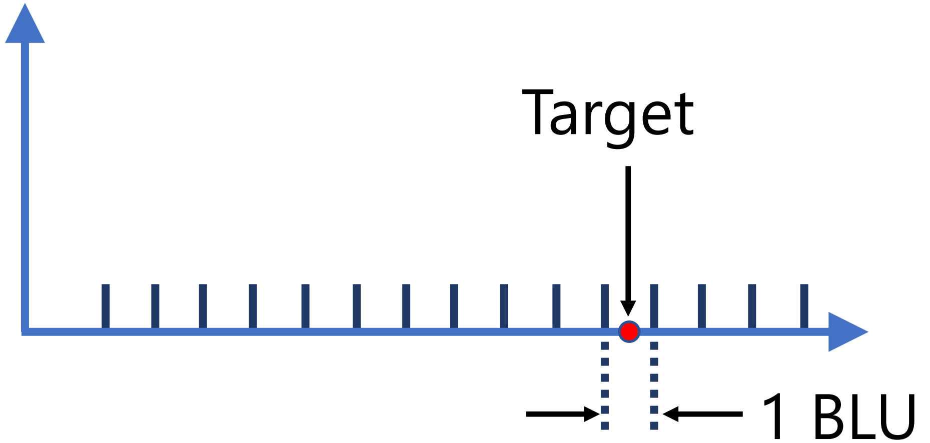
Example: Encoder with 1000 pulses/revolution + 10 mm pitch.
Leadscrew = 1 pulse for each 0.01 mm of linear displacement of axis.
0.01 mm is the control resolution.

For efficiency, **PR = CR = BLU = SR**

Machine Accuracy Considerations

❑ System Accuracy (SA)

- System inaccuracy due to control resolution = ½ BLU
- Displacements or movements < 1 BLU cannot be measured or programmed
- Inaccuracies associated with robot itself



$$\boxed{\text{System accuracy} = \tfrac{1}{2}\ \text{BLU} + \text{Robot accuracy}}$$

Aim: System resolution = system accuracy

Summary

- Trajectory generation involves the creation of time-varying sequences of poses
- Joint space trajectory: **polynomial, LSPB, bang-bang, etc**
- Accuracy vs repeatability: Bad accuracy can be corrected by reprogramming, bad repeatability is expensive to correct

Lecture 8 – Path planning

- ❑ Introduction to path planning – key definitions
- ❑ Map based method
  - Distance Transform
  - D* Method
  - Probabilistic Roadmap method (PRM)
- ❑ Artificial Potential Field (APF) method

Further information

- Repeatability of Practical Robots
  - $\pm$1 mm (medium duty work)
  - $\pm$ 0.05 – 0.1 (medium assembly work)
  - $\pm$ 0.02 mm (precision assembly work)