Trajectories

Planning

TECHNICAL UNIVERSITY OF CLUJ-NAPOCA ROMANIA

November 9, 2023

Agenda

- Why trajectories?
- Interpolation
- Joint trajectories
- End-effector position trajectories
- End-effector pose trajectories

Recap

Geometric Models

Forward kinematics

I want to know where will my end-effector be, if I give specific coordinates (values) to each joint

Recap

Geometric Models

Forward kinematics

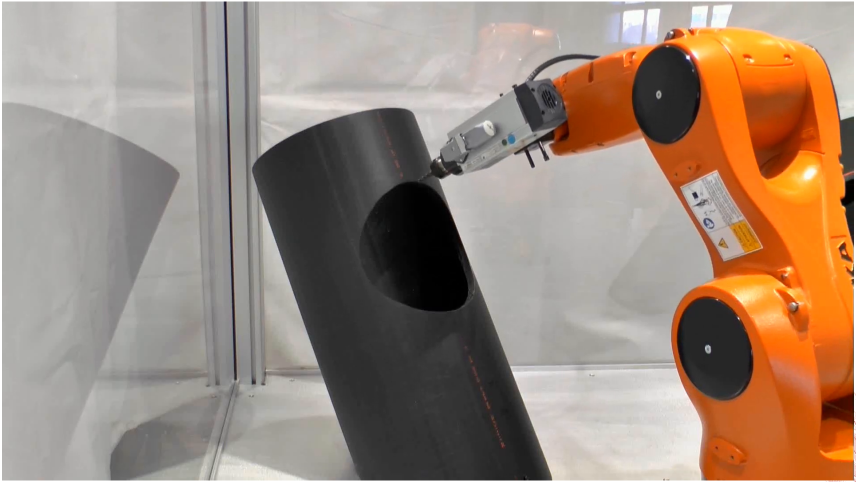
I want to know where will my end-effector be, if I give specific coordinates (values) to each joint

Inverse kinematics

I want to know what should the joint coordinates (values) be in order for my end-effector to reach a specific pose

Why trajectories?

Video example

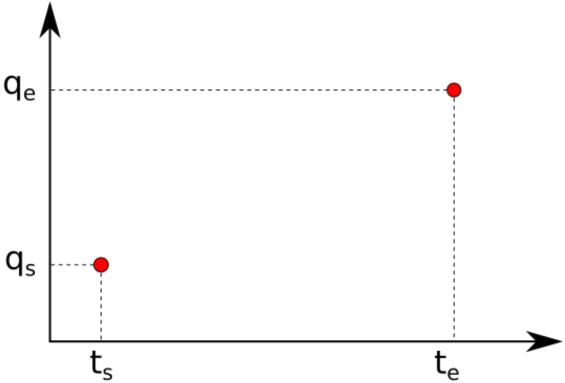


Why trajectories?

- We often do not care only about the final pose of a movement
- It helps in obstacle avoidance
- We can avoid singular configurations

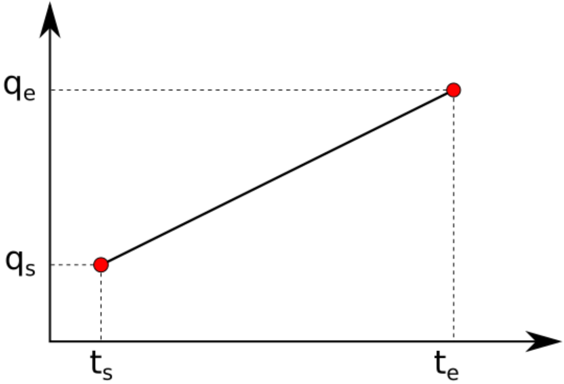
Interpolation basics

Linear interpolation



Interpolation basics

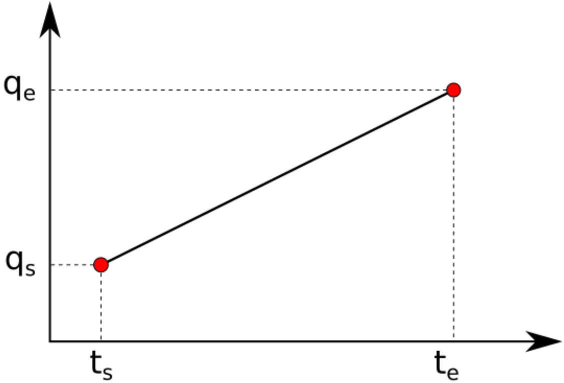
Linear interpolation



$$q(t) = (1 - \frac{t - t_s}{t_e - t_s}) q_s + \frac{t - t_s}{t_e - t_s} q_e$$

Interpolation basics

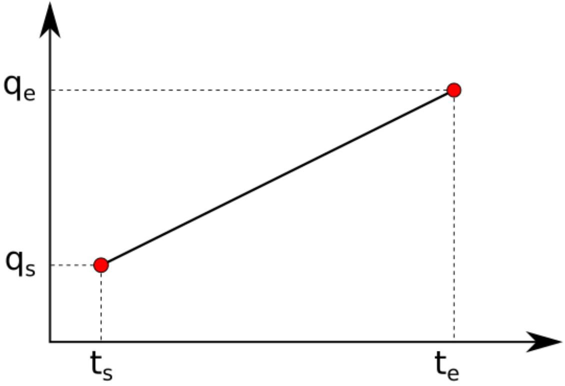
Linear interpolation



$q(t) = (1 - \frac{t-t_s}{t_e-t_s})q_s + \frac{t-t_s}{t_e-t_s}q_e$
when $t = t_s, q(t_s) = q_s$

Interpolation basics

Linear interpolation

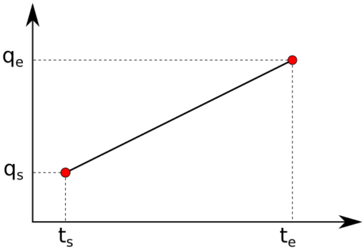


$q(t) = (1 - \frac{t-t_s}{t_e-t_s})q_s + \frac{t-t_s}{t_e-t_s}q_e$
when $t = t_s, q(t_s) = q_s$
when $t = t_e, q(t_e) = q_e$

Interpolation basics

Linear interpolation

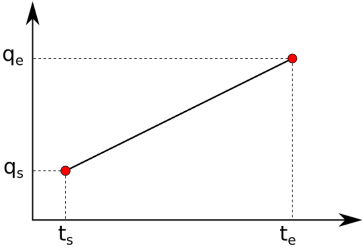
$$q(t) = (1 - \frac{t-t_s}{t_e-t_s})q_s + \frac{t-t_s}{t_e-t_s}q_e$$



To simplify matters a bit, we define a function s, which will be our 'interpolator'

Interpolation basics

Linear interpolation



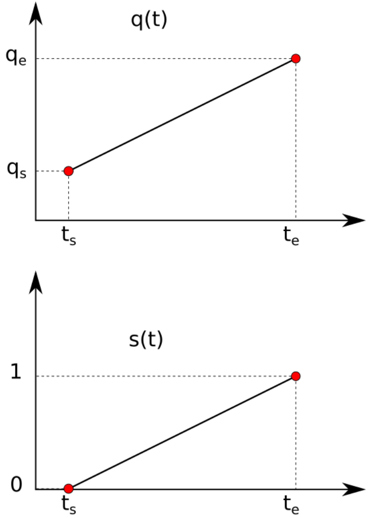
$$q(t) = (1 - \frac{t - t_s}{t_e - t_s}) q_s + \frac{t - t_s}{t_e - t_s} q_e$$

To simplify matters a bit, we define a function s, which will be our 'interpolator'

$$s(t) = \frac{t - t_s}{t_e - t_s}, s[0, 1]$$

Interpolation basics

Linear interpolation



$$q(t) = (1 - \frac{t - t_s}{t_e - t_s})q_s + \frac{t - t_s}{t_e - t_s} q_e$$

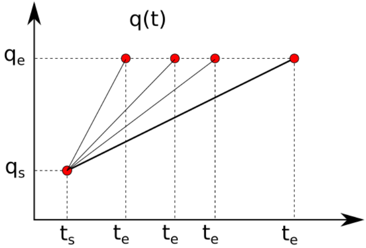
To simplify matters a bit, we define a function s, which will be our 'interpolator'

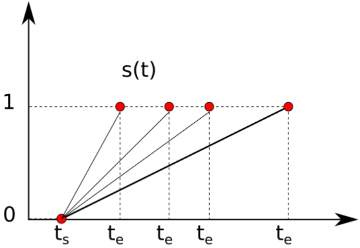
$$s(t) = \frac{t - t_s}{t_e - t_s}, s[0, 1]$$

$$q(t) = (1 - s(t))q_s + s(t)q_e$$

Interpolation basics

Linear interpolation





$$q(t) = (1 - \frac{t - t_s}{t_e - t_s})q_s + \frac{t - t_s}{t_e - t_s} q_e$$

To simplify matters a bit, we define a function s, which will be our 'interpolator'

$$s(t) = \frac{t - t_s}{t_e - t_s}, s[0, 1]$$

$$q(t) = (1 - s(t))q_s + s(t)q_e$$

The 'slope' of s defines the velocity

Interpolation

Application in Robotics

We have initial and target Pose P_s, and P_e, respectively.

Interpolation

Application in Robotics

We have initial and target Pose P_s, and P_e, respectively.
We calculate the Inverse Kinematics for P_s and P_e

Interpolation

Application in Robotics

We have initial and target Pose P_s, and P_e, respectively.
We calculate the Inverse Kinematics for P_s and P_e

$$\begin{gathered} q_{1s} \rightarrow q_{1e} \\ q_{2s} \rightarrow q_{2e} \\ \vdots \\ q_{ns} \rightarrow q_{ne} \end{gathered}$$

Interpolation

Application in Robotics

We have initial and target Pose P_s, and P_e, respectively.
We calculate the Inverse Kinematics for P_s and P_e

$$\begin{gathered} q_{1s} \rightarrow q_{1e} \\ q_{2s} \rightarrow q_{2e} \\ \vdots \\ q_{ns} \rightarrow q_{ne} \end{gathered}$$

We interpolate each one of them separately

Interpolation

Application in Robotics

We have initial and target Pose P_s, and P_e, respectively.
We calculate the Inverse Kinematics for P_s and P_e

$$\begin{gathered} q_{1s} \rightarrow q_{1e} \\ q_{2s} \rightarrow q_{2e} \\ \vdots \\ q_{ns} \rightarrow q_{ne} \end{gathered}$$

We interpolate each one of them separately

$$q_n(t) = (1 - s(t))q_{ns} + s(t)q_{ne}$$

Interpolation

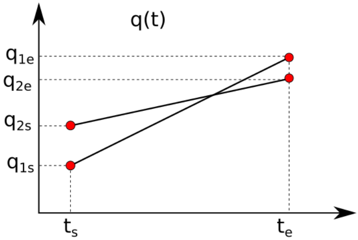
Application in Robotics

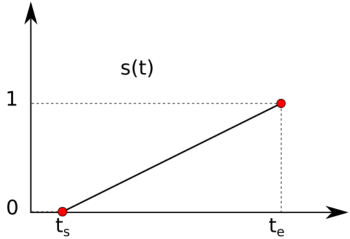
We can either interpolate each joint over the same time period

Interpolation

Application in Robotics

We can either interpolate each joint over the same time period

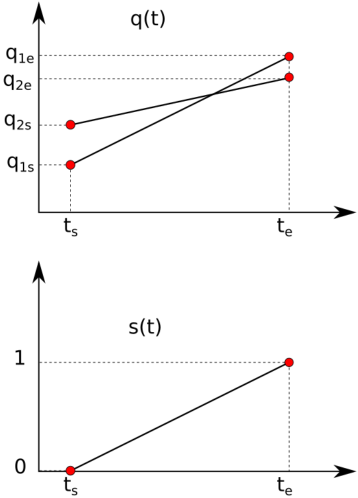




Interpolation

Application in Robotics

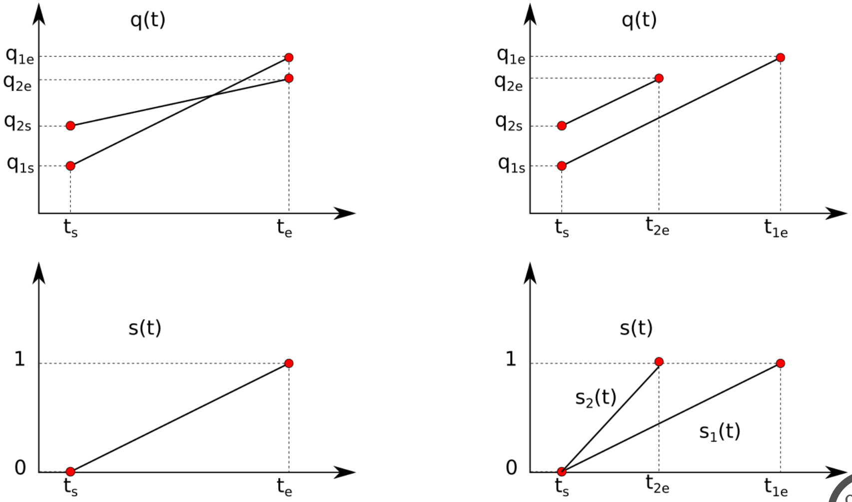
We can either interpolate each joint over the same time period or interpolate each joint to have the same velocity



Interpolation

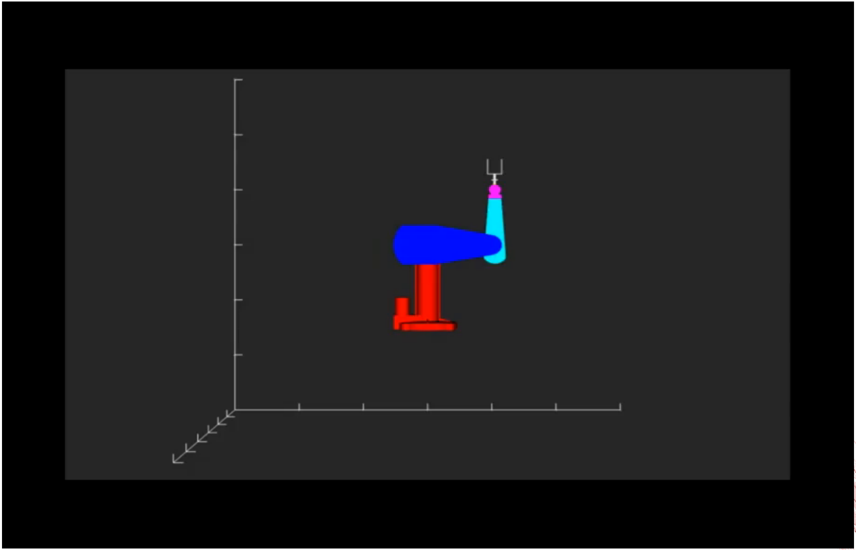
Application in Robotics

We can either interpolate each joint over the same time period or interpolate each joint to have the same velocity



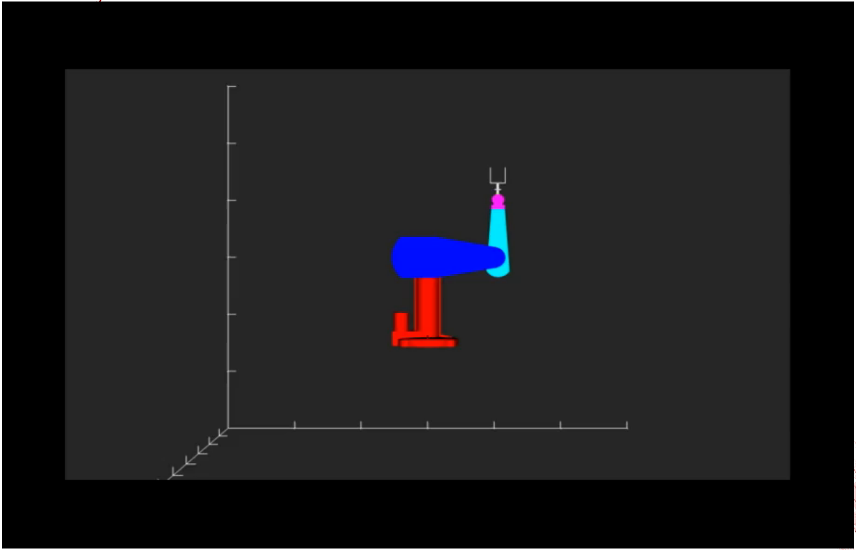
Interpolation

Same duration



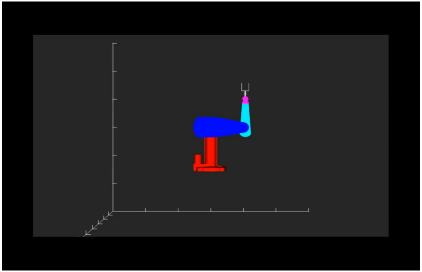
Interpolation

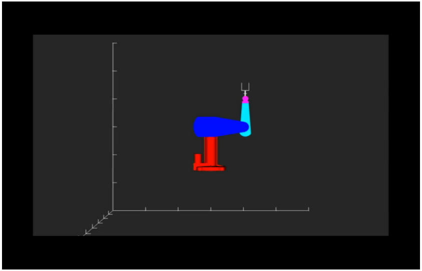
Max velocity



Interpolation

Both





Linear interpolation

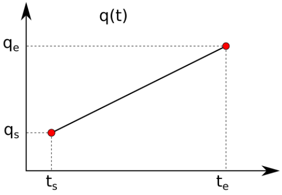
Problems?

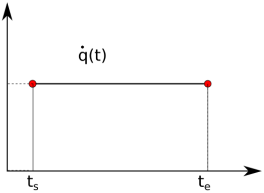
Are there any issues with the linear interpolation?

Linear interpolation

Problems?

Are there any issues with the linear interpolation?

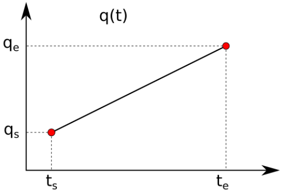


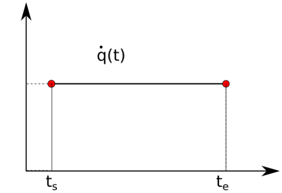


Linear interpolation

Problems?

Are there any issues with the linear interpolation?





What about accelerations?

Polynomial interpolation

I want to ensure a specific starting and ending position, with zero velocities and accelerations at the beginning and end of the trajectory.

Polynomial interpolation

I want to ensure a specific starting and ending position, with zero velocities and accelerations at the beginning and end of the trajectory.

I need a polynomial!

Polynomial interpolation

I want to ensure a specific starting and ending position, with zero velocities and accelerations at the beginning and end of the trajectory.

I need a polynomial! To simplify things, I use again my interpolator 's' to define the polynomial

Polynomial interpolation

I want to ensure a specific starting and ending position, with zero velocities and accelerations at the beginning and end of the trajectory.
I need a polynomial! To simplify things, I use again my interpolator 's' to define the polynomial

$$s(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F, t \in [0, T]$$

Polynomial interpolation

I want to ensure a specific starting and ending position, with zero velocities and accelerations at the beginning and end of the trajectory.
I need a polynomial! To simplify things, I use again my interpolator 's' to define the polynomial

$$s(t) = At^5 + Bt^4 + Ct^3 + Dt^2 + Et + F, t \in [0, T]$$

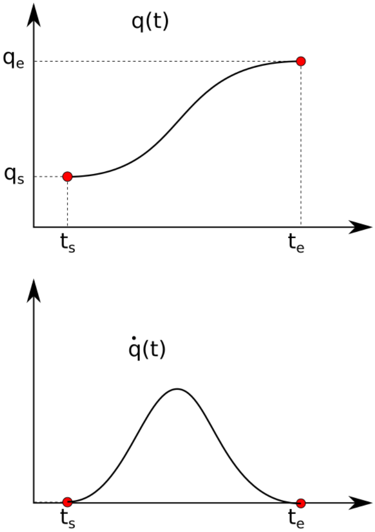
With conditions:

$$s(0) = 0$$
$$s(T) = 1$$
$$\dot{s}(0) = 0$$
$$\dot{s}(T) = 0$$
$$\ddot{s}(0) = 0$$
$$\ddot{s}(T) = 0$$

Polynomial Interpolation

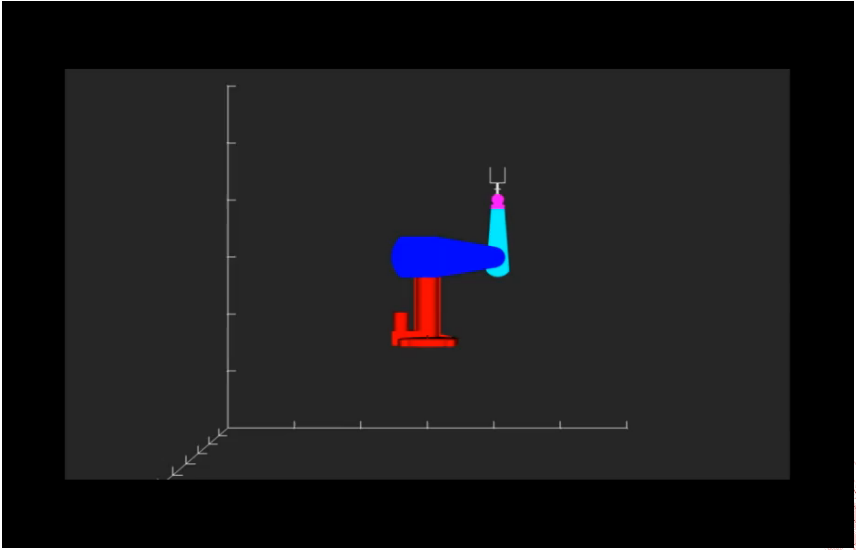
$$\begin{bmatrix} s(0) \\ s(T) \\ \dot{s}(0) \\ \dot{s}(T) \\ \ddot{s}(0) \\ \ddot{s}(T) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \end{bmatrix}$$

Polynomial Interpolation



Much smoother movement!

Polynomial Interpolation



Interpolation

Trajectories

What if I want to follow a specific trajectory?

Trajectories

Position

When we only care about the position of the end-effector, things are easy:

Trajectories

Position

When we only care about the position of the end-effector, things are easy:
I interpolate each coordinate between starting and ending position (either linear or polynomial interpolation):

$$
\begin{gathered}
P_x(0) \to P_x(T) \\
P_y(0) \to P_y(T) \\
P_z(0) \to P_z(T) \\
\text{e.g. } P_x(t) = (1 - s(t))P_x(0) + sP_x(T)
\end{gathered}
$$

Trajectories

Position

When we only care about the position of the end-effector, things are easy:
I interpolate each coordinate between starting and ending position (either linear or polynomial interpolation):

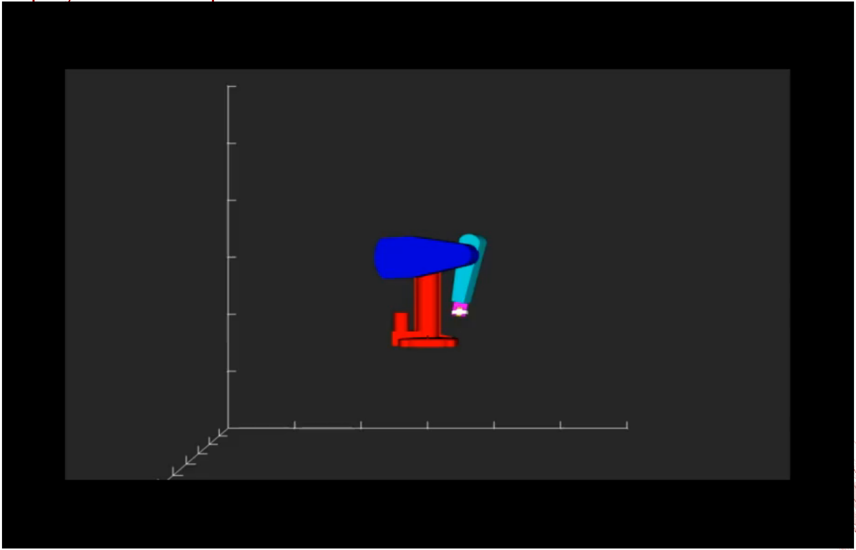
$$P_x(0) \rightarrow P_x(T)$$
$$P_y(0) \rightarrow P_y(T)$$
$$P_z(0) \rightarrow P_z(T)$$
$$\text{e.g. } P_x(t) = (1 - s(t))P_x(0) + sP_x(T)$$

I solve the inverse kinematics for each of the interpolated positions:

$$q(t) = g(P_x(t), P_y(t), P_z(t))$$

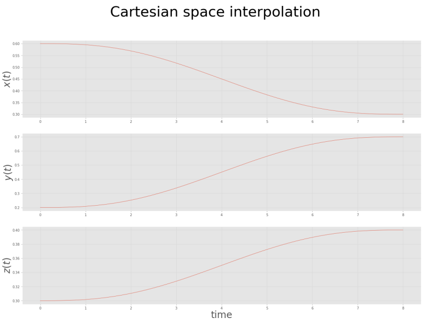
Position Trajectory

With polynomial interpolation



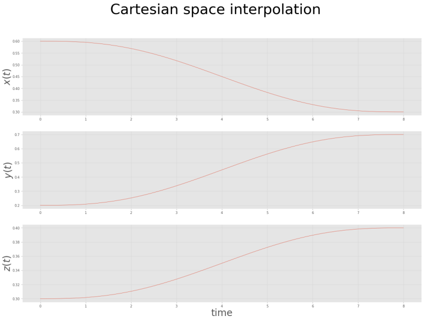
Position Trajectory

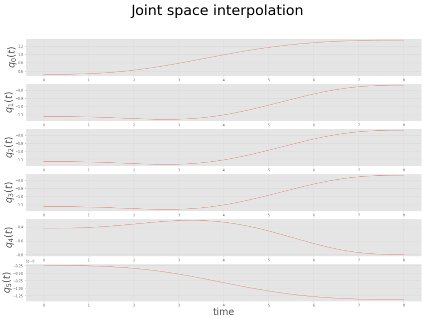
Cartesian space and joint space



Position Trajectory

Cartesian space and joint space





Orientation Trajectory

What about orientation?

Orientation Trajectory

What about orientation? How can we interpolate?

$$R_m^n = \begin{bmatrix} X_x & Y_x & Z_x \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{bmatrix}$$

Orientation Trajectory

What about orientation? How can we interpolate?

$$R_m^n = \begin{bmatrix} X_x & Y_x & Z_x \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{bmatrix}$$

We cannot interpolate each value individually :(

Orientation Trajectory

What about orientation? How can we interpolate?

$$R_m^n = \begin{bmatrix} X_x & Y_x & Z_x \\ X_y & Y_y & Z_y \\ X_z & Y_z & Z_z \end{bmatrix}$$

We cannot interpolate each value individually :(
We can decompose to Euler angles!

Orientation Trajectories

Euler angles

Arbitrary **orientation**

The transition between two arbitrarily **oriented** coordinate frames can **always** be described in terms of **three elementary rotations**

Orientation Trajectories

Euler angles

Arbitrary **orientation**

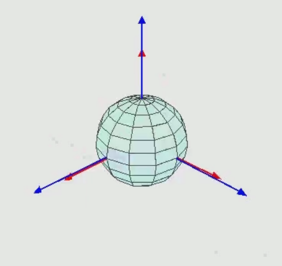
The transition between two arbitrarily **oriented** coordinate frames can **always** be described in terms of **three elementary rotations**

Orientation Trajectories

Euler angles

Arbitrary **orientation**

The transition between two arbitrarily **oriented** coordinate frames can **always** be described in terms of **three elementary rotations**

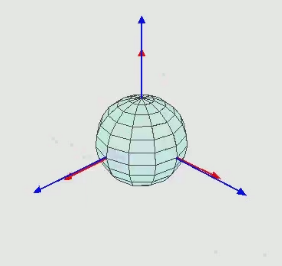


Orientation Trajectories

Euler angles

Arbitrary **orientation**

The transition between two arbitrarily **oriented** coordinate frames can **always** be described in terms of **three elementary rotations**



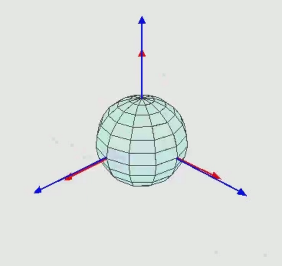
Three subsequent rotations, along specific axes

Orientation Trajectories

Euler angles

Arbitrary **orientation**

The transition between two arbitrarily **oriented** coordinate frames can **always** be described in terms of **three elementary rotations**



Three subsequent rotations, along specific axes

- Z-X-Z
- X-Y-X
- X-Y-Z
- ...

Trajectories

Orientation

If we have starting and ending Euler angles, we can interpolate as usual:

Trajectories

Orientation

If we have starting and ending Euler angles, we can interpolate as usual:
I interpolate each coordinate between starting and ending position (either linear or polynomial interpolation):

$$R(0) \to R(T)$$
$$P(0) \to P(T)$$
$$Y(0) \to Y(T)$$
$$\text{e.g. } R(t) = (1 - s(t))R(0) + sR(T)$$

Trajectories

Orientation

If we have starting and ending Euler angles, we can interpolate as usual:
I interpolate each coordinate between starting and ending position (either linear or polynomial interpolation):

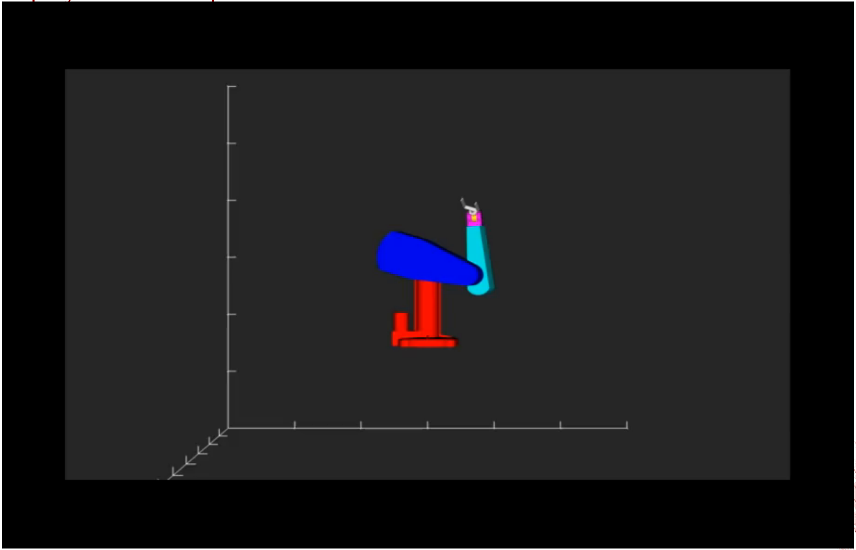
$$
\begin{gathered}
R(0) \rightarrow R(T) \\
P(0) \rightarrow P(T) \\
Y(0) \rightarrow Y(T) \\
\text{e.g. } R(t) = (1 - s(t))R(0) + sR(T)
\end{gathered}
$$

I solve the inverse kinematics for each of the interpolated orientations (and positions):

$$
q(t) = g(R(t), P(t), Y(t), P_x(t), P_y(t), P_z(t))
$$

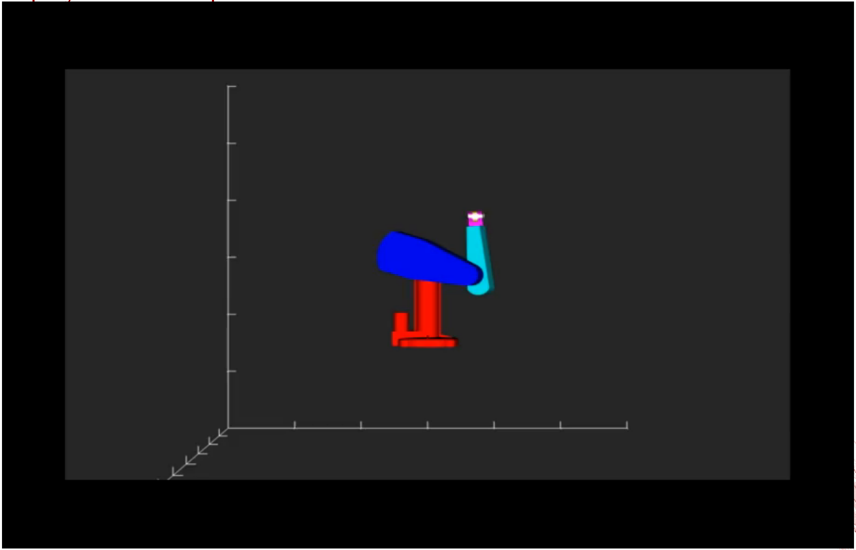
Orientation Trajectory

With polynomial interpolation



Full Trajectory

With polynomial interpolation



Interpolating orientation

Quaternions

Quaternions

Quaternions is a number system that extends the complex numbers. It can be used to represent relative orientation of two coordinate frames

$a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$

Interpolating orientation

Quaternions

Quaternions

Quaternions is a number system that extends the complex numbers. It can be used to represent relative orientation of two coordinate frames

$a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$

x	**1**	**i**	**j**	**k**
1	1	i	j	k
i	i	-1	k	-j
j	j	-k	-1	i
k	k	j	-i	-1

Interpolating orientation

Quaternions

We cannot interpolate the elements of quaternions using polynomials, because that would result in non valid quaternions (as with the orientation matrix)

Interpolating orientation

Quaternions

We cannot interpolate the elements of quaternions using polynomials, because that would result in non valid quaternions (as with the orientation matrix)

We can use **SLERP** (Spherical Linear Interpolation)!

$$Slerp(p_s, p_e, s(t)) = \frac{sin[(1-s(t))]\Omega}{sin(\Omega)}p_s + \frac{sin[s(t)\Omega]}{sin(\Omega)}p_e$$

Interpolating orientation

Quaternions

We cannot interpolate the elements of quaternions using polynomials, because that would result in non valid quaternions (as with the orientation matrix)

We can use **SLERP** (Spherical Linear Interpolation)!

$$Slerp(p_s, p_e, s(t)) = \frac{sin[(1 - s(t))]\Omega}{sin(\Omega)} p_s + \frac{sin[s(t)\Omega]}{sin(\Omega)} p_e$$

For quaternions it is a bit different:

Interpolating orientation

Quaternions

We cannot interpolate the elements of quaternions using polynomials, because that would result in non valid quaternions (as with the orientation matrix)

We can use **SLERP** (Spherical Linear Interpolation)!

$$Slerp(p_s, p_e, s(t)) = \frac{sin[(1 - s(t))]\Omega}{sin(\Omega)} p_s + \frac{sin[s(t)\Omega]}{sin(\Omega)} p_e$$

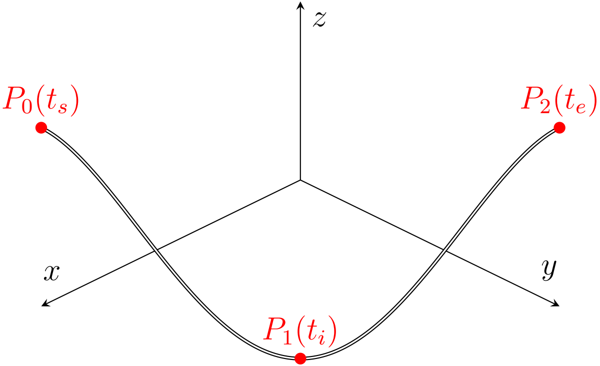
For quaternions it is a bit different:

$$Slerp(q_s, q_e, s(t)) = q_s(q_s^{-1} q_e)^{s(t)}$$

More complex trajectories

Via points

What if we have multiple poses that we want to pass from?



Via points

Higher order polynomials

With conditions:

$$
\begin{gathered}
q(t_s) = q_s \\
q(t_e) = q_e \\
q(t_i) = q_i \\
\dot{q}(t_s) = 0 \\
\dot{q}(t_e) = 0 \\
\dot{q}(t_i) = \dot{q}_i \\
\ddot{q}(t_s) = 0 \\
\ddot{q}(t_i) = 0 \\
\ddot{q}(t_e) = 0
\end{gathered}
$$

Via points

Higher order polynomials

With conditions:

$$
\begin{aligned}
q(t_s) &= q_s \\
q(t_e) &= q_e \\
q(t_i) &= q_i \\
\dot{q}(t_s) &= 0 \\
\dot{q}(t_e) &= 0 \\
\dot{q}(t_i) &= \dot{q}_i \\
\ddot{q}(t_s) &= 0 \\
\ddot{q}(t_i) &= 0 \\
\ddot{q}(t_e) &= 0
\end{aligned}
$$

Potential problems?

Via points

Multiple polynomials

First polynomial:

$$
\begin{aligned}
q(t_s) &= q_s \\
q(t_e) &= q_i \\
\dot{q}(t_s) &= 0 \\
\dot{q}(t_e) &= \dot{q}_i \\
\ddot{q}(t_s) &= 0 \\
\ddot{q}(t_e) &= 0
\end{aligned}
$$

Via points

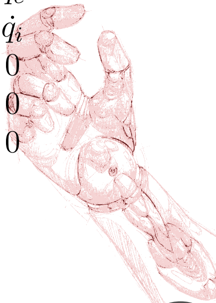
Multiple polynomials

First polynomial:

$$
\begin{aligned}
q(t_s) &= q_s \\
q(t_e) &= q_i \\
\dot{q}(t_s) &= 0 \\
\dot{q}(t_e) &= \dot{q}_i \\
\ddot{q}(t_s) &= 0 \\
\ddot{q}(t_e) &= 0
\end{aligned}
$$

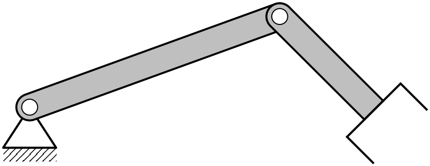
Second polynomial:

$$
\begin{aligned}
q(t_s) &= q_i \\
q(t_e) &= q_e \\
\dot{q}(t_s) &= \dot{q}_i \\
\dot{q}(t_e) &= 0 \\
\ddot{q}(t_s) &= 0 \\
\ddot{q}(t_e) &= 0
\end{aligned}
$$



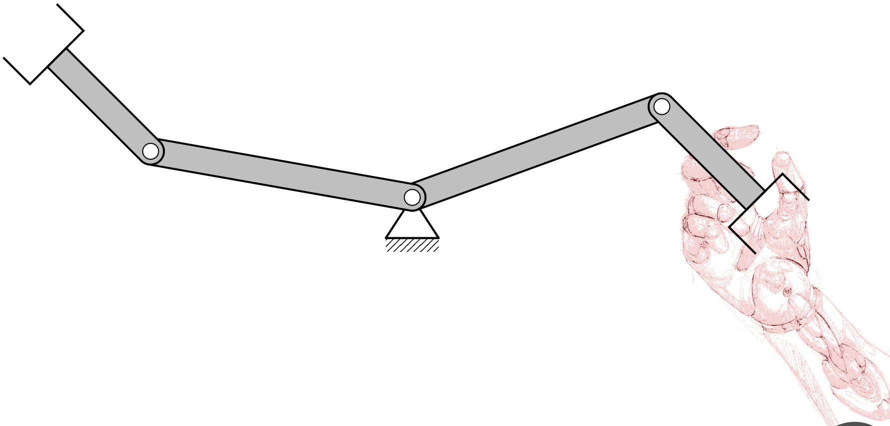
Cartesian path problems

Unrechable poses



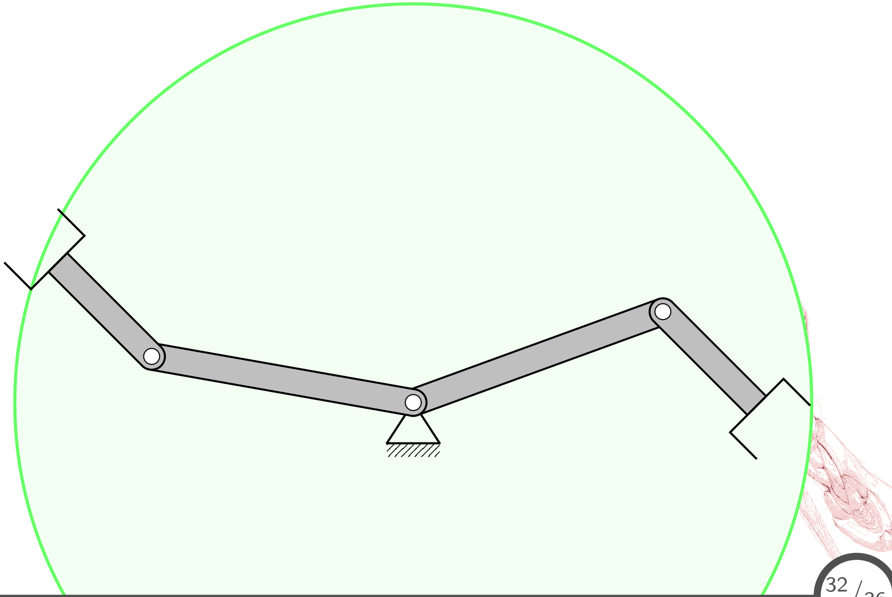
Cartesian path problems

Unrechable poses



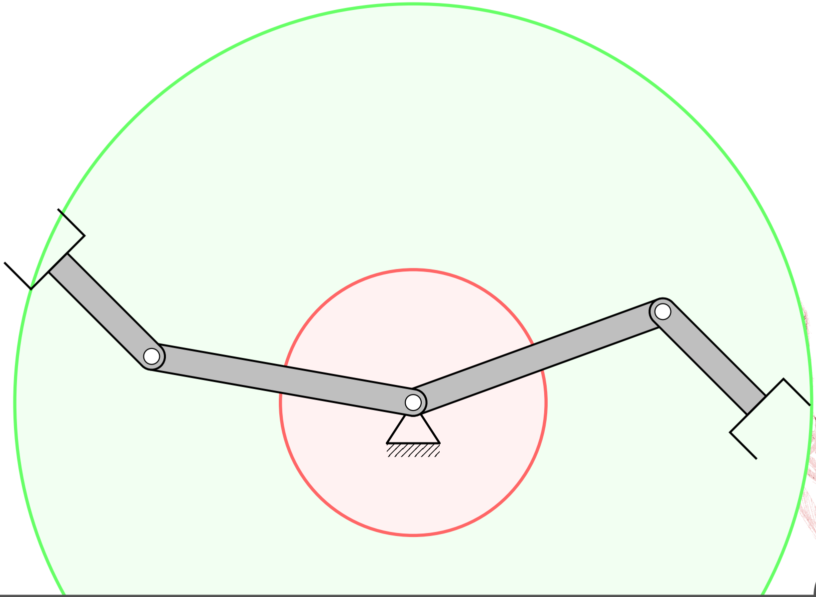
Cartesian path problems

Unrechable poses



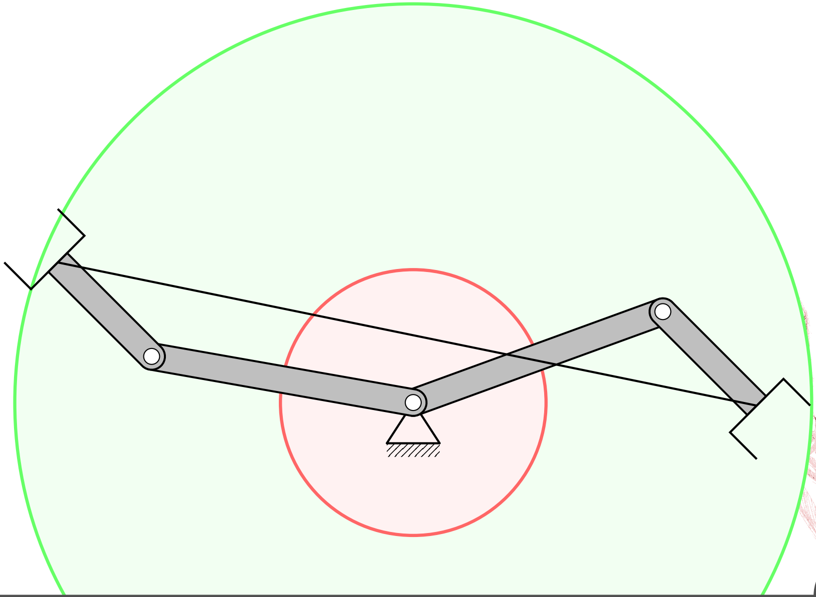
Cartesian path problems

Unrechable poses



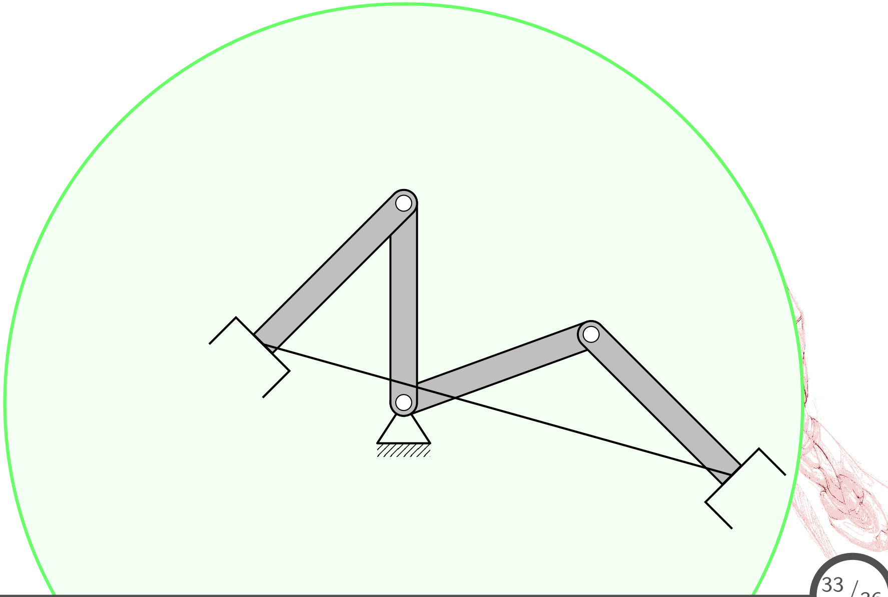
Cartesian path problems

Unrechable poses



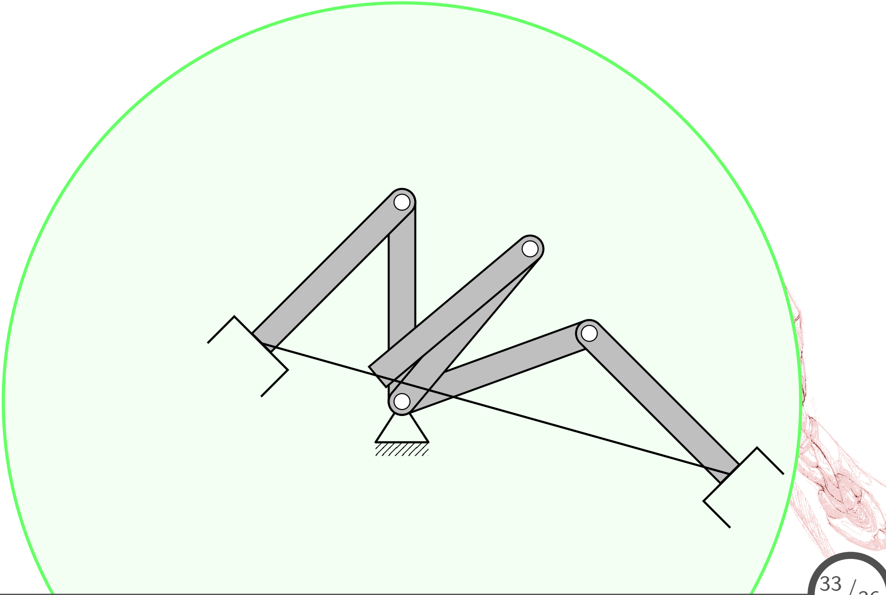
Cartesian path problems

High velocities at singularities



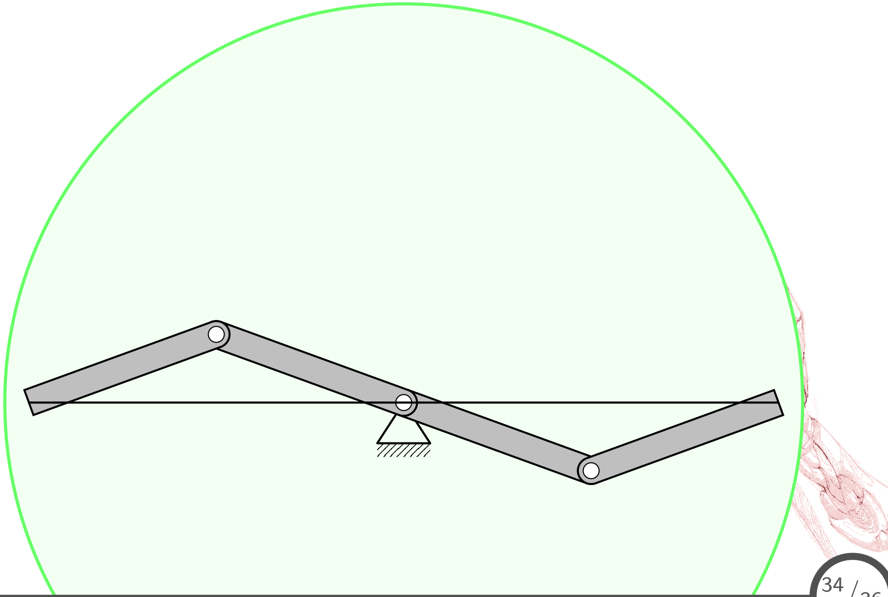
Cartesian path problems

High velocities at singularities



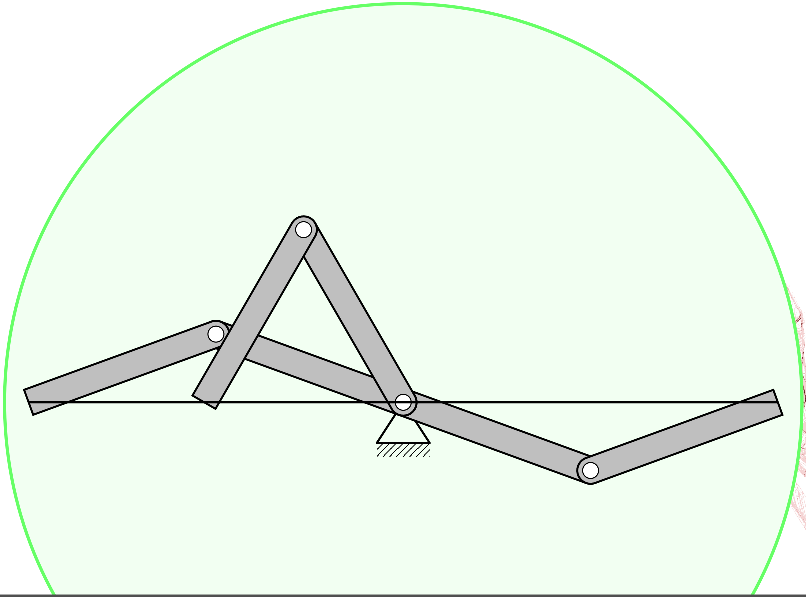
Cartesian path problems

Jumps due to joint limits



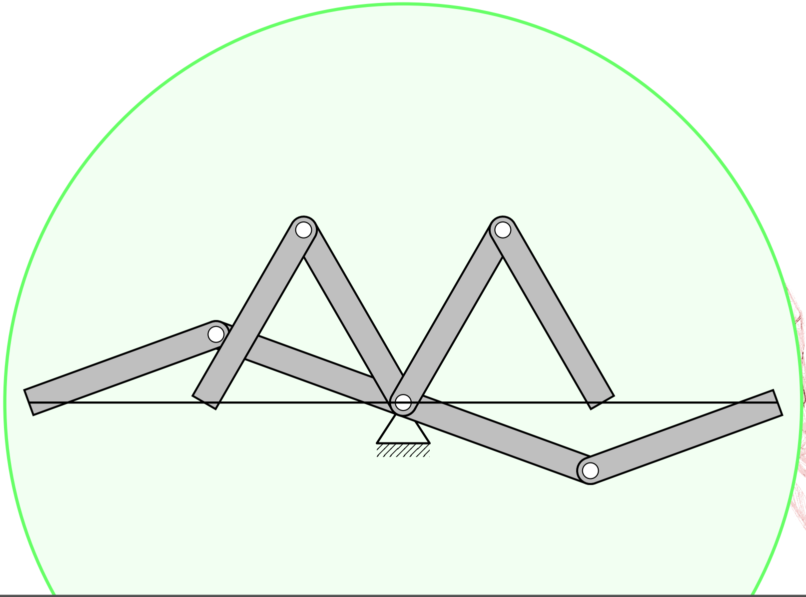
Cartesian path problems

Jumps due to joint limits



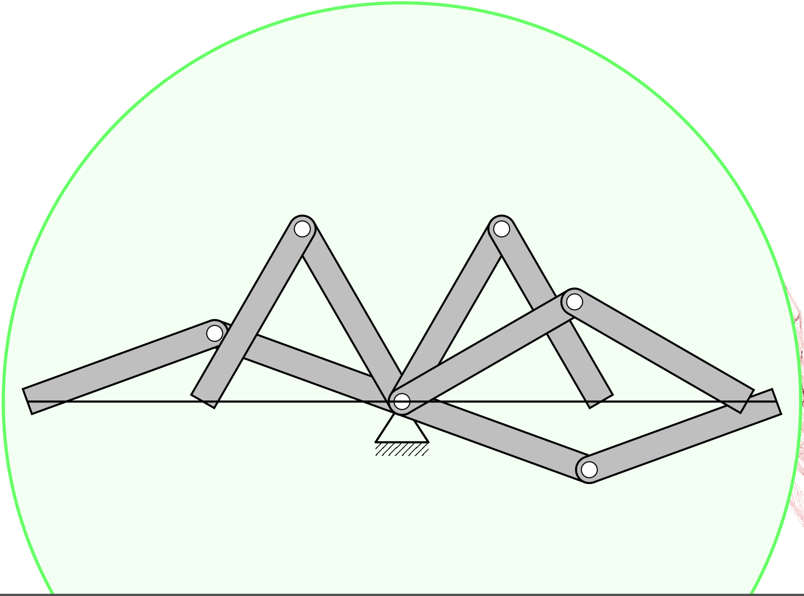
Cartesian path problems

Jumps due to joint limits



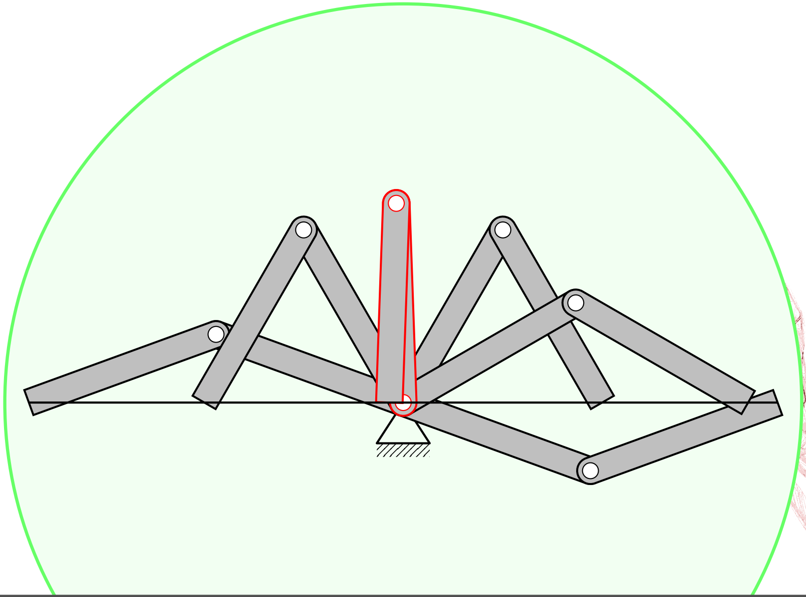
Cartesian path problems

Jumps due to joint limits



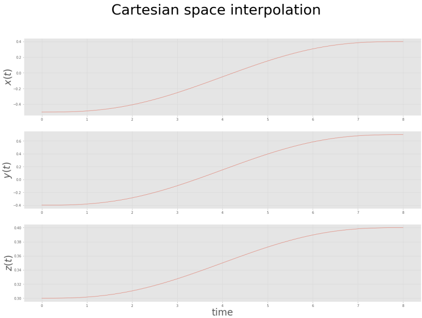
Cartesian path problems

Jumps due to joint limits



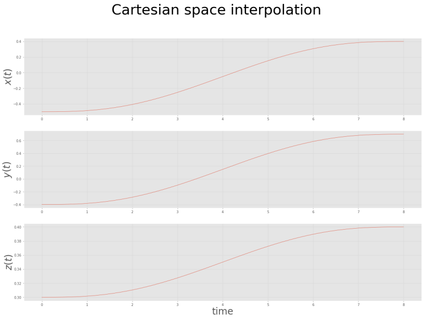
Cartesian path problems

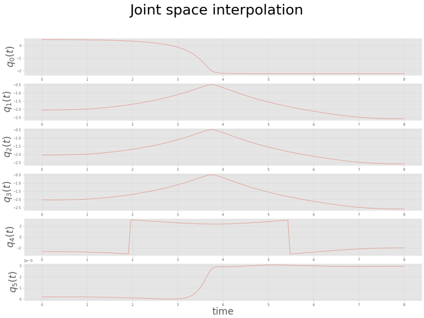
Jumps due to joint limits



Cartesian path problems

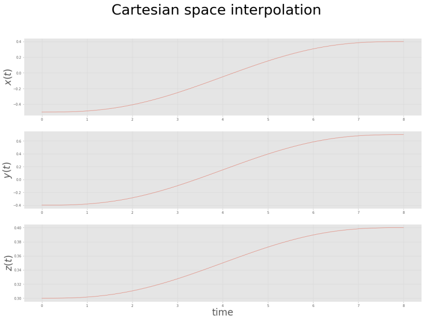
Jumps due to joint limits

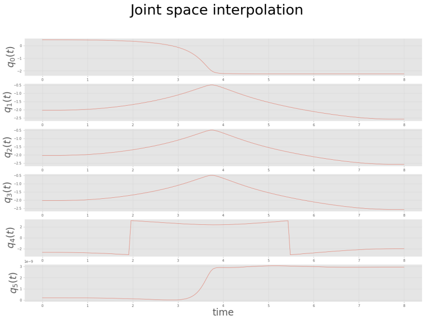




Cartesian path problems

Jumps due to joint limits





In general, we interpolate in joint space, cause it is more robust

Questions?