



Robotics 1

Trajectory planning in Cartesian space

Prof. Alessandro De Luca

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI



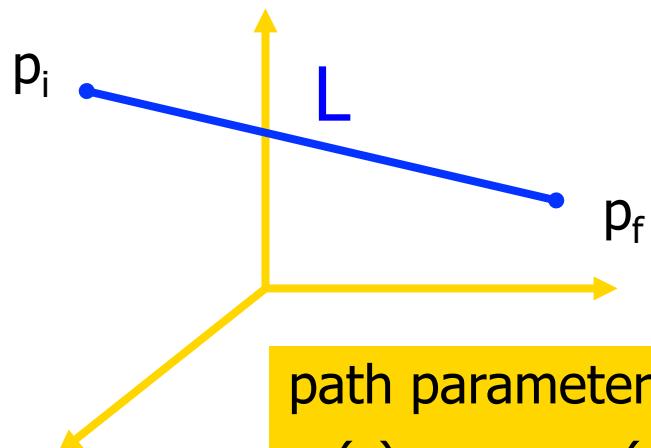


Trajectories in Cartesian space

- in general, the trajectory planning methods proposed in the joint space can be applied also in the Cartesian space
 - consider **independently** each component of the task vector (i.e., a position or an angle of a minimal representation of orientation)
- however, when planning a trajectory for the three orientation angles, the resulting global motion cannot be intuitively **visualized** in advance
- if possible, we still prefer to plan Cartesian trajectories **separately** for **position** and **orientation**
- the number of knots to be interpolated in the Cartesian space is typically low (e.g., 2 knots for a PTP motion, 3 if a “via point” is added) ⇒ use **simple** interpolating paths, such as straight lines, arc of circles, ...



Planning a linear Cartesian path (position only)



path parameterization
 $p(s) = p_i + s(p_f - p_i)$

GIVEN
 $p_i, p_f, v_{\max}, a_{\max}$
 v_i, v_f (typically = 0)

$$L = \|p_f - p_i\|$$

$\frac{p_f - p_i}{\|p_f - p_i\|}$ = unit vector of directional cosines of the line

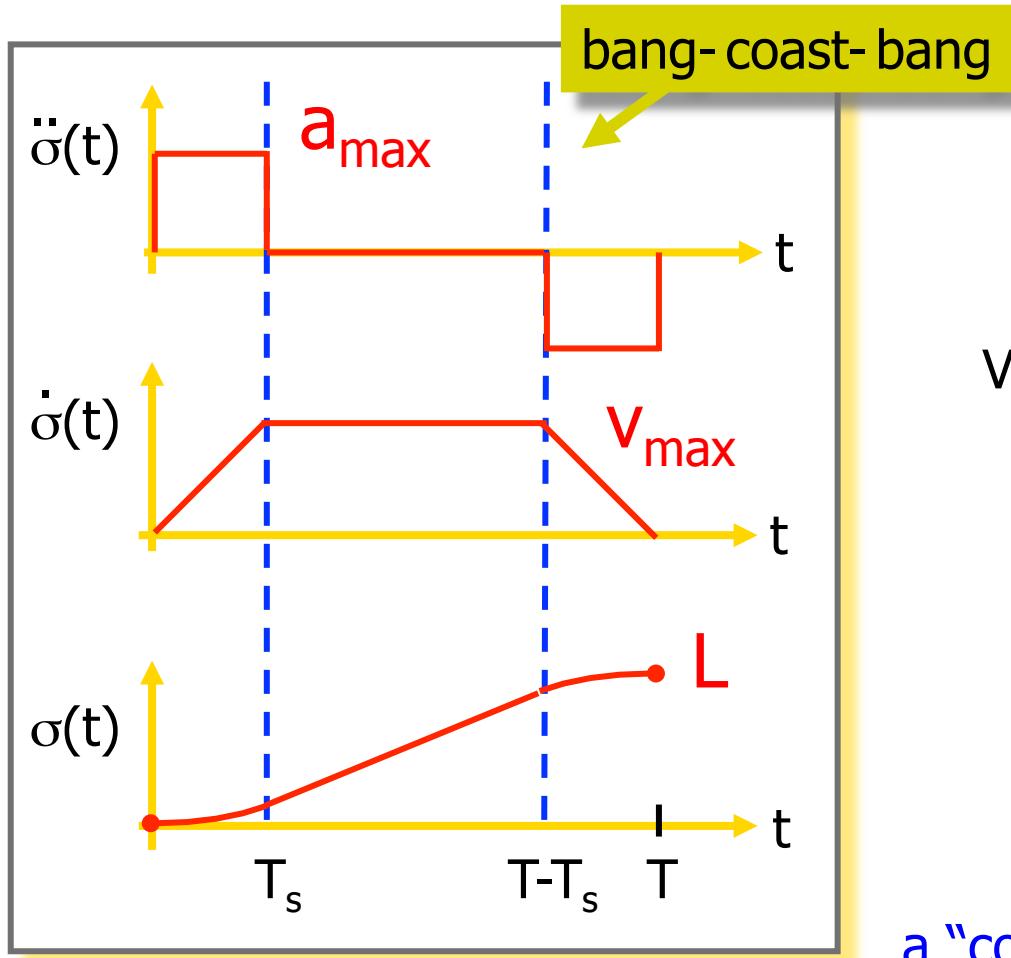
$s \in [0,1]$ ← setting $s = \sigma/L$, $\sigma \in [0,L]$ is the **arc length**
(gives the current length of the path)

$$\begin{aligned}\dot{p}(s) &= \frac{dp}{ds} \dot{s} = (p_f - p_i) \dot{s} \\ &= \frac{p_f - p_i}{L} \dot{\sigma}\end{aligned}$$

$$\begin{aligned}\ddot{p}(s) &= \frac{d^2p}{ds^2} \dot{s}^2 + \frac{dp}{ds} \ddot{s} = (p_f - p_i) \ddot{s} \\ &= \frac{p_f - p_i}{L} \ddot{\sigma}\end{aligned}$$



Timing law with trapezoidal speed - 1



given*: L, v_{\max}, a_{\max}
find: T_s, T

$v_{\max}(T - T_s) = L$ ← = area of the speed profile

$$T_s = \frac{v_{\max}}{a_{\max}}$$

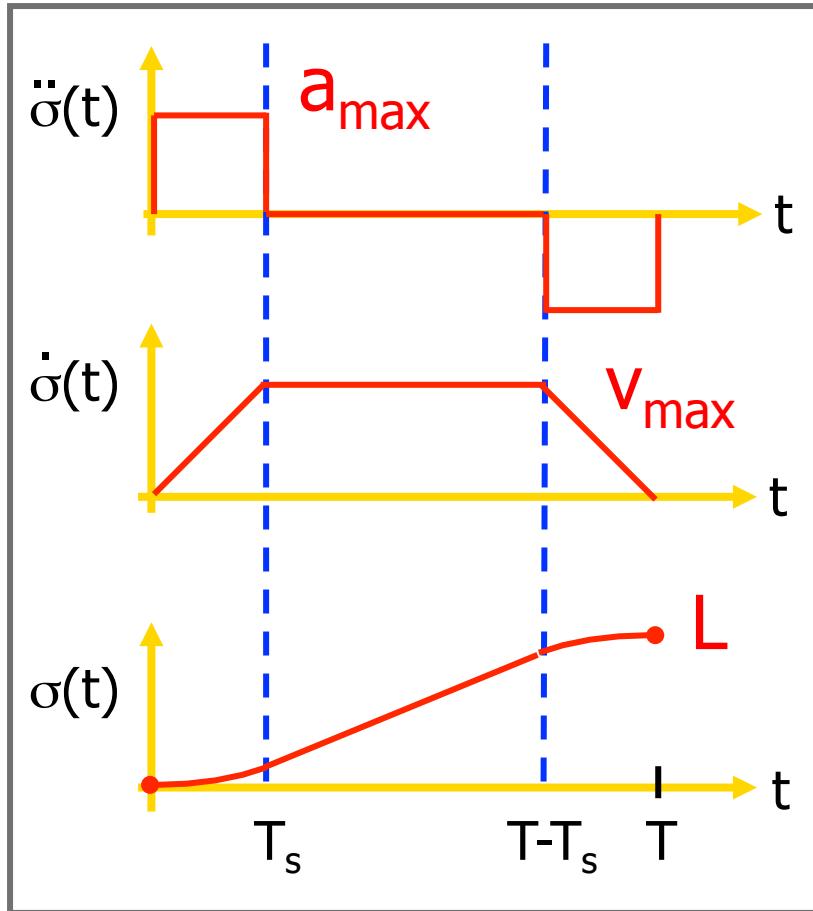
$$T = \frac{L a_{\max} + v_{\max}^2}{a_{\max} v_{\max}}$$

a "coast" phase exists iff: $L > v_{\max}^2/a_{\max}$

* = other input data combinations are possible (see textbook)



Timing law with trapezoidal speed - 2

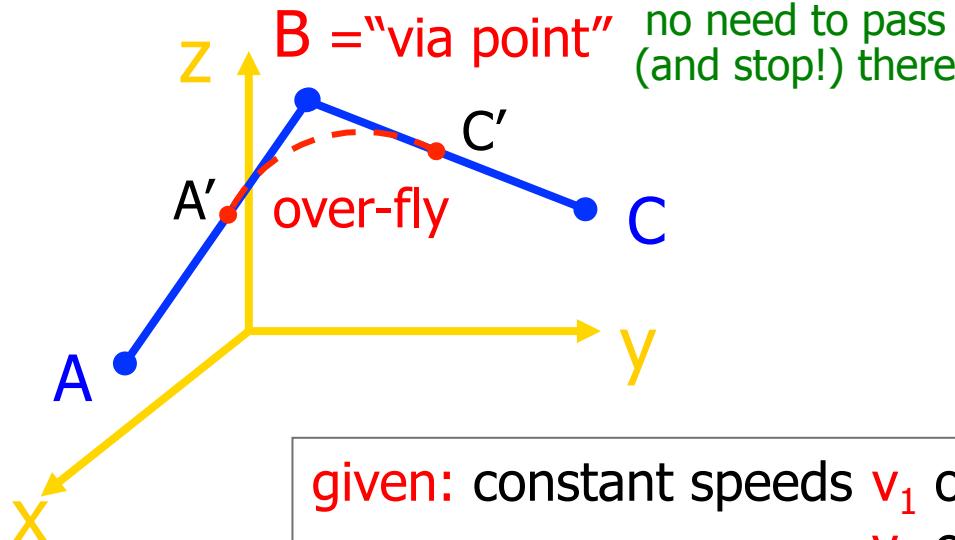


$$\sigma(t) = \begin{cases} a_{\max} t^2/2 & t \in [0, T_s] \\ v_{\max} t - \frac{v_{\max}^2}{2 a_{\max}} & t \in [T_s, T-T_s] \\ - a_{\max} (t-T)^2/2 + v_{\max} T - \frac{v_{\max}^2}{a_{\max}} & t \in [T-T_s, T] \end{cases}$$

can be used also
in the joint space!



Concatenation of linear paths



$$\frac{\mathbf{B} - \mathbf{A}}{\|\mathbf{B} - \mathbf{A}\|} = \mathbf{k}_{AB}$$

$$\frac{\mathbf{C} - \mathbf{B}}{\|\mathbf{C} - \mathbf{B}\|} = \mathbf{k}_{BC}$$

unit vectors of
direction cosines

given: constant speeds v_1 on linear path AB
 v_2 on linear path BC

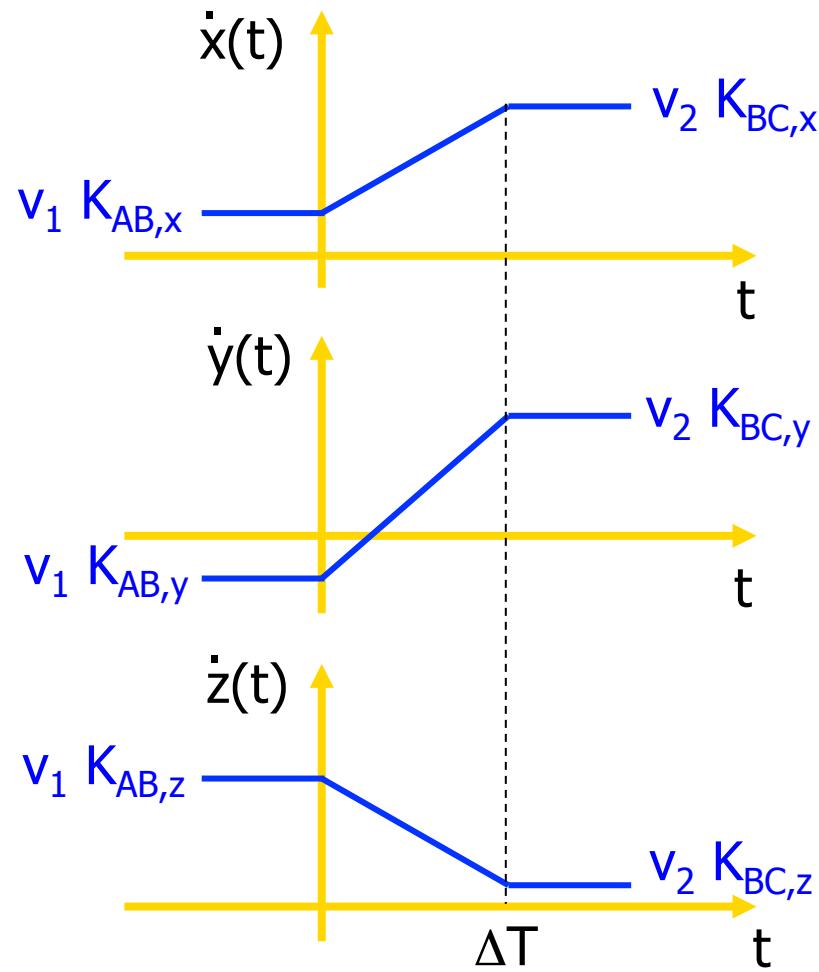
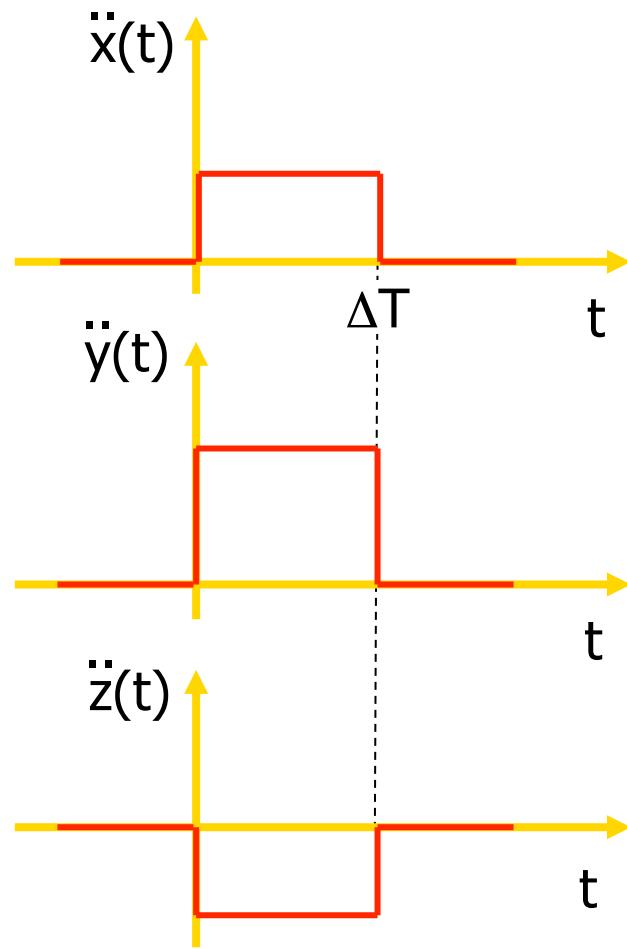
desired transition: with constant acceleration for a time ΔT

$$\mathbf{p}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} \quad t \in [0, \Delta T] \text{ (transition starts at } t = 0\text{)}$$

note: during over-fly, the path remains always in the plane specified by the two lines intersecting at B (in essence, it is a **planar problem**)

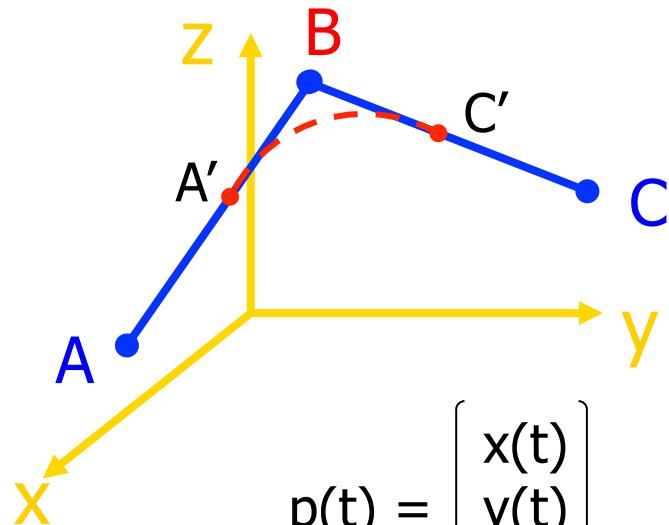


Time profiles on components





Timing law during transition



$$p(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}$$

$$\frac{B - A}{\|B - A\|} = K_{AB}$$

$$\frac{C - B}{\|C - B\|} = K_{BC}$$

unit vectors of
direction cosines

$t \in [0, \Delta T]$ (transition starts at $t = 0$)

$$\ddot{p}(t) = (v_2 K_{BC} - v_1 K_{AB})/\Delta T$$

$$\int$$

$$\dot{p}(t) = v_1 K_{AB} + (v_2 K_{BC} - v_1 K_{AB}) t/\Delta T$$

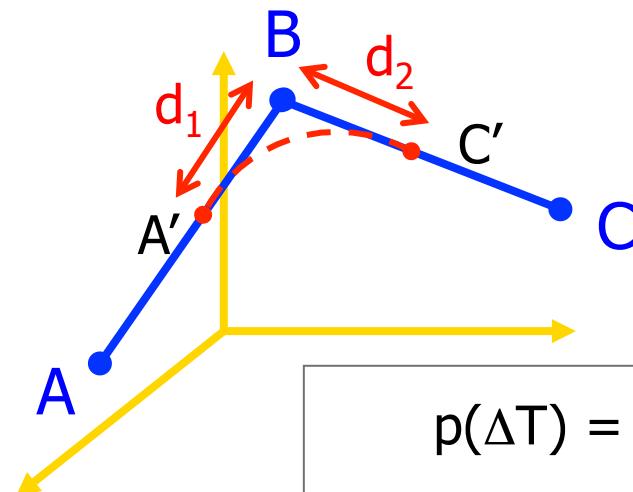
$$\int$$

$$p(t) = A' + v_1 K_{AB} t + (v_2 K_{BC} - v_1 K_{AB}) t^2/(2\Delta T)$$

thus, we obtain a
parabolic blending
(see textbook
for this same approach
in the joint space)



Solution (various options)



$$\begin{aligned} B - A' &= d_1 K_{AB} \\ C' - B &= d_2 K_{BC} \end{aligned}$$

1

$$p(t) = A' + v_1 K_{AB} t + (v_2 K_{BC} - v_1 K_{AB})t^2/(2\Delta T)$$

$$p(\Delta T) = A' + (\Delta T/2) (v_1 K_{AB} + v_2 K_{BC}) = C'$$

→ $-B + A' + (\Delta T/2) (v_1 K_{AB} + v_2 K_{BC}) = C' - B$

1 → $d_1 K_{AB} + d_2 K_{BC} = (\Delta T/2) (v_1 K_{AB} + v_2 K_{BC})$

→ $d_1 = v_1 \Delta T/2$

$d_2 = v_2 \Delta T/2$

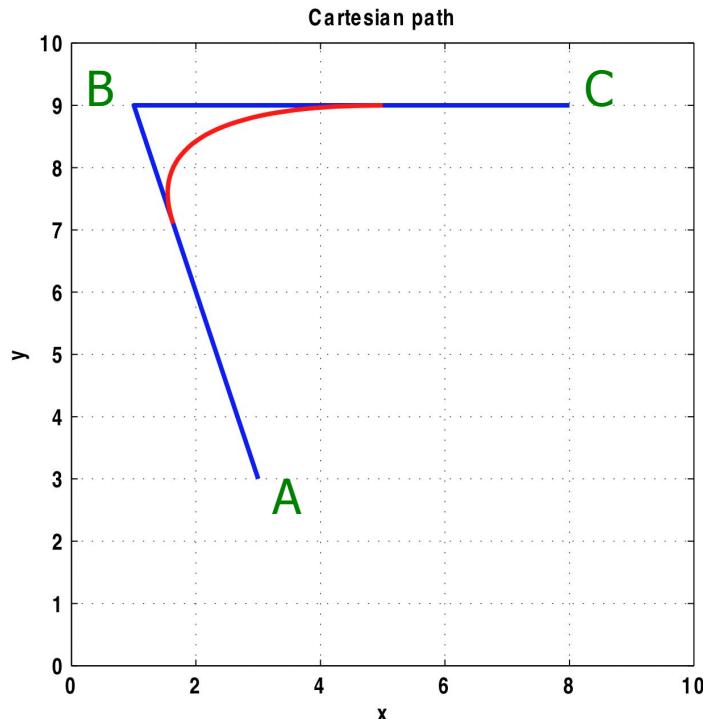
by choosing, e.g., d_1
(namely A')

$\Delta T = 2d_1/v_1 \rightarrow d_2 = d_1 v_2/v_1$

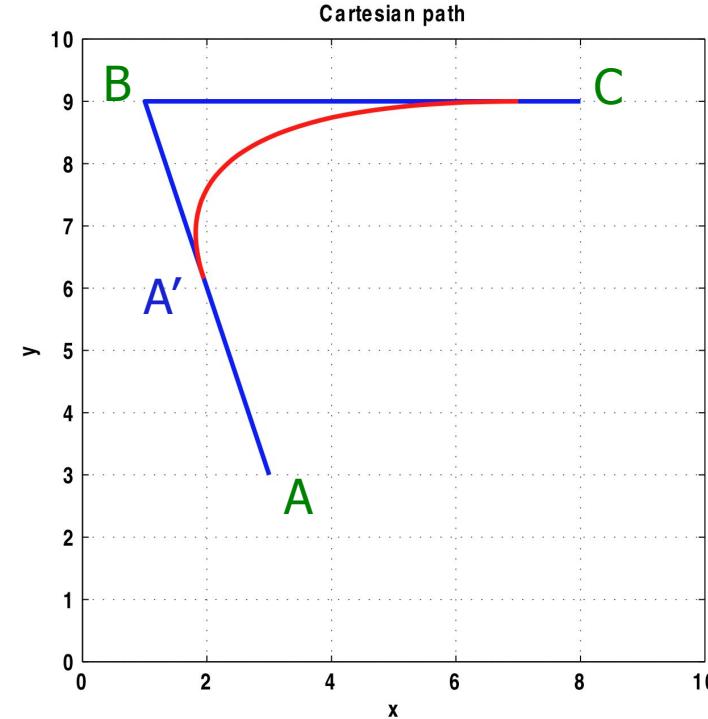


A numerical example

- transition from $A=(3,3)$ to $C=(8,9)$ via $B=(1,9)$, with speed from $v_1=1$ to $v_2=2$
- exploiting **two options** for solution (resulting in **different paths!**)
 - assign transition time: $\Delta T=4$ (we re-center it here for $t \in [-\Delta T/2, \Delta T/2]$)
 - assign distance from B for departing: $d_1=3$ (assign d_2 for landing is handled similarly)



$\Delta T=4$

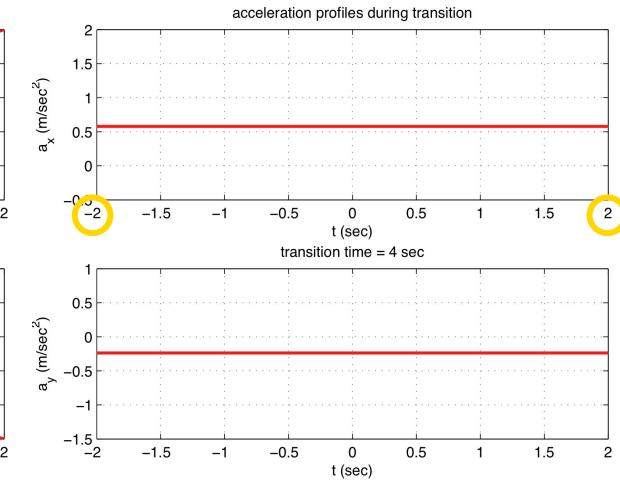
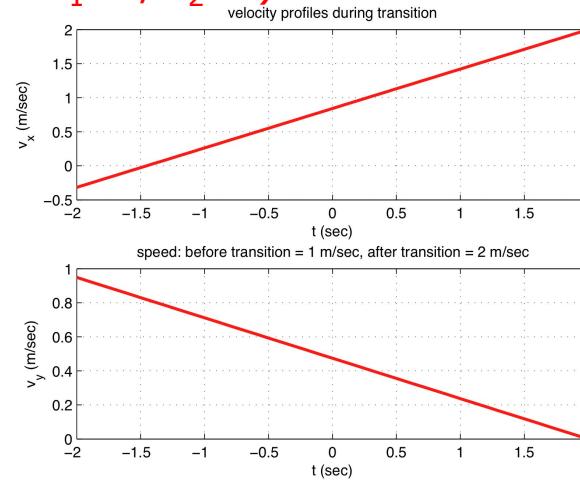
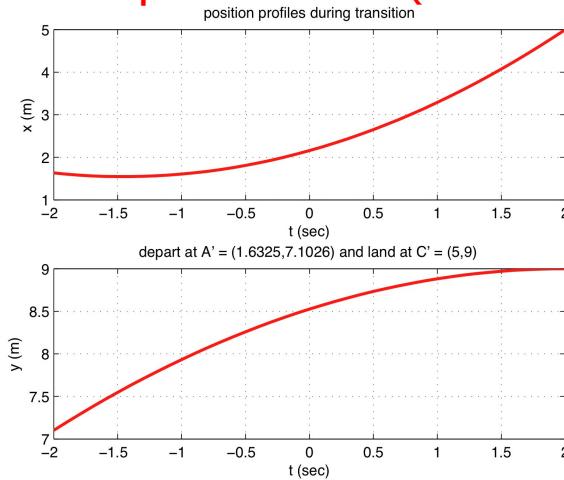


$d_1=3$

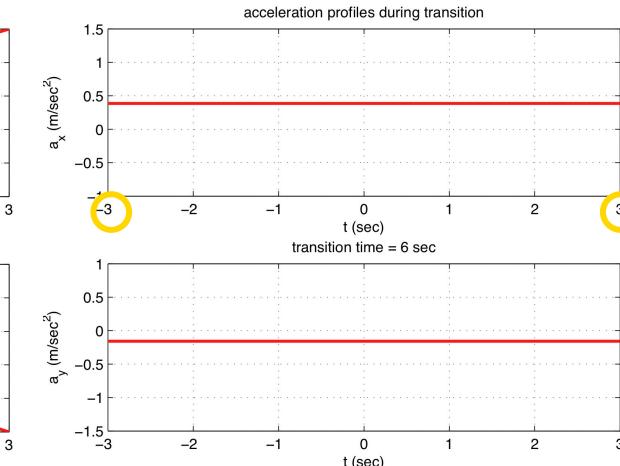
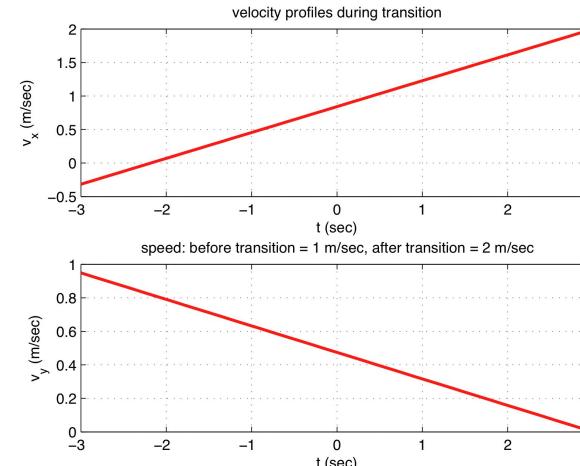
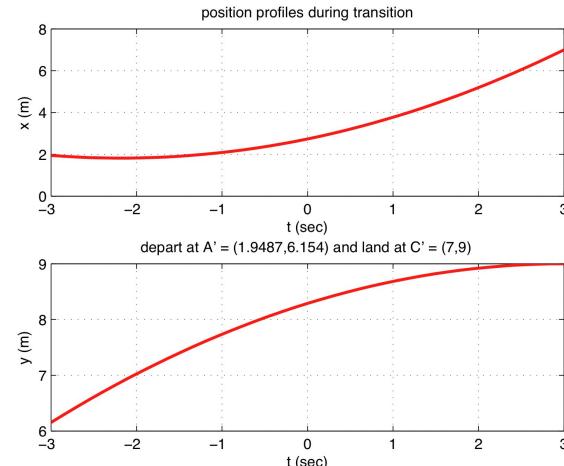


A numerical example (cont'd)

first option: $\Delta T=4$ (resulting in $d_1=2$, $d_2=4$)



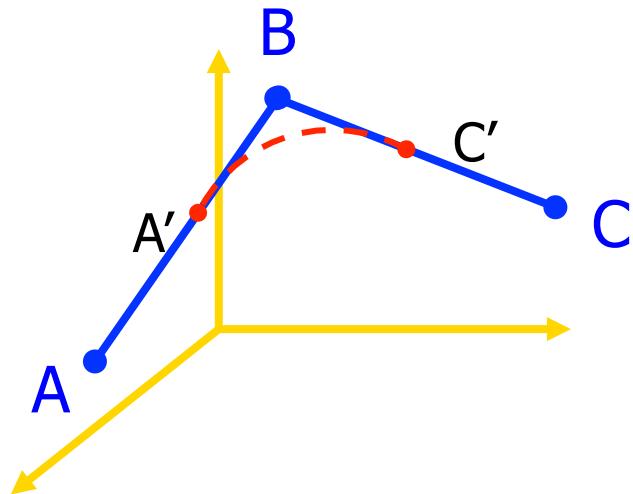
second option: $d_1=3$ (resulting in $\Delta T=6$, $d_2=6$)



actually, the same vel/acc profiles only with a different time scale!!



Alternative solution (imposing acceleration)



$$\ddot{p}(t) = 1/\Delta T (v_2 K_{BC} - v_1 K_{AB})$$

$v_1 = v_2 = v_{\max}$ (for simplicity)

$$\|\ddot{p}(t)\| = a_{\max}$$

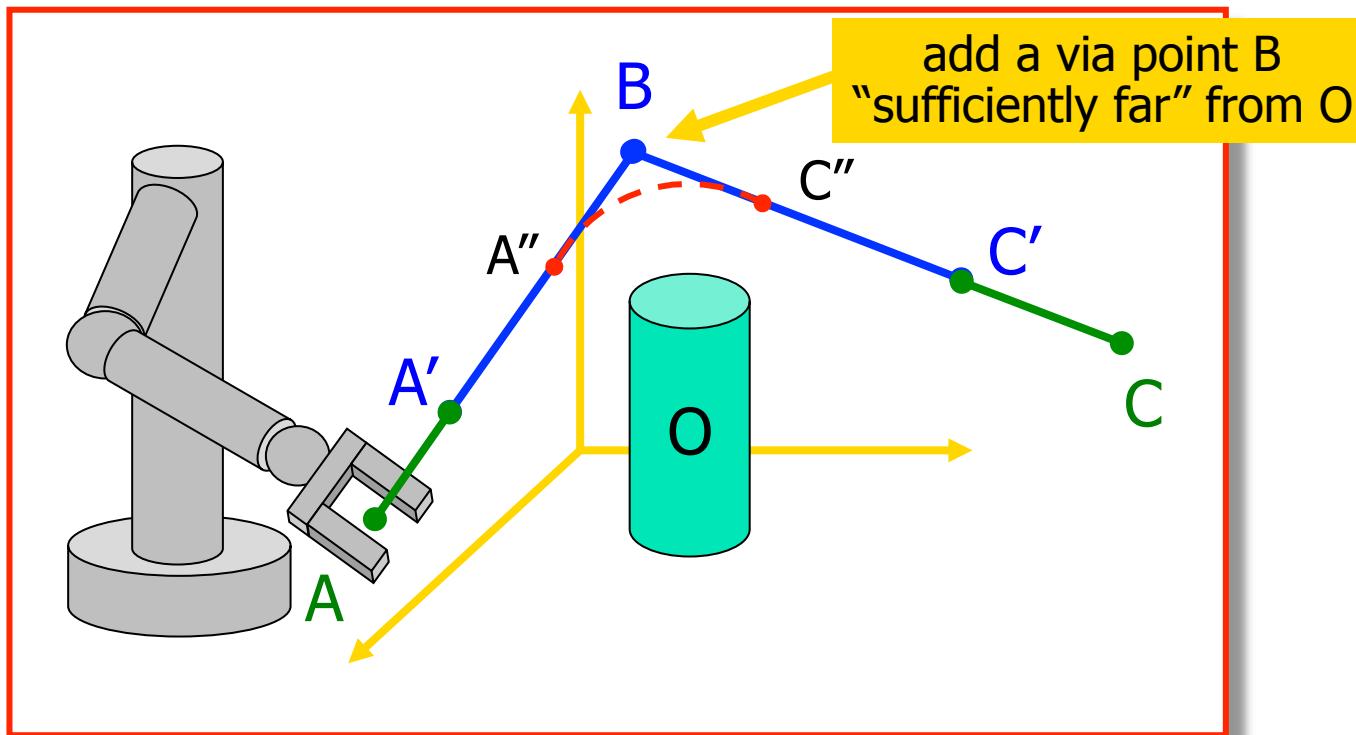
$$\begin{aligned}\Delta T &= (v_{\max} / a_{\max}) \|K_{BC} - K_{AB}\| \\ &= (v_{\max} / a_{\max}) \sqrt{2(1 - K_{BC,x}K_{AB,x} - K_{BC,y}K_{AB,y} - K_{BC,z}K_{AB,z})}\end{aligned}$$

then, $d_1 = d_2 = v_{\max} \Delta T / 2$



Application example

plan a Cartesian trajectory from A to C (rest-to-rest)
that avoids the obstacle O, with $a \leq a_{\max}$ and $v \leq v_{\max}$



on $\overline{AA'}$ $\rightarrow a_{\max}$

on $\overline{A'B}$ and $\overline{BC'}$ $\rightarrow v_{\max}$
+ over-fly between A'' e C''

on $\overline{C'C}$ $\rightarrow -a_{\max}$



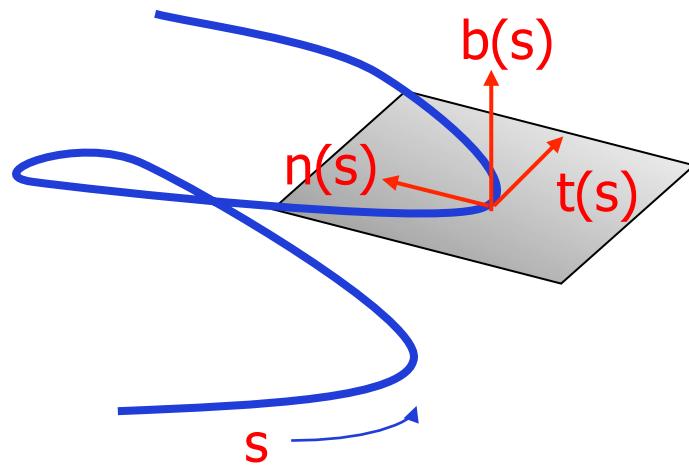
Other Cartesian paths

- **circular path** through 3 points in 3D (often built-in feature)
- linear path for the end-effector with **constant orientation**
- in robots with **spherical wrist**: planning may be **decomposed** into a path for wrist center and one for E-E orientation, with a common timing law
- though more complex in general, it is often **convenient** to parameterize the Cartesian geometric path $p(s)$ in terms of its **arc length** (e.g., with $s = R\theta$ for circular paths), so that
- **velocity**: $dp/dt = dp/ds \cdot ds/dt$
 - dp/ds = unit vector ($\|\cdot\|=1$) tangent to the path: **tangent** direction $t(s)$
 - ds/dt = absolute value of tangential velocity (= **speed**)
- **acceleration**: $d^2p/dt^2 = d^2p/ds^2 \cdot (ds/dt)^2 + dp/ds \cdot d^2s/dt^2$
 - $\|d^2p/ds^2\| = \text{curvature } \kappa(s)$ (= 1/radius of curvature)
 - $d^2p/ds^2 \cdot (ds/dt)^2 = \text{centripetal acceleration}$: **normal** direction $n(s)$ \perp to the path, on the osculating plane; **binormal** direction $b(s) = t(s) \times n(s)$
 - d^2s/dt^2 = scalar value (**with any sign**) of tangential acceleration



Definition of Frenet frame

- For a generic (smooth) path $p(s)$ in \mathbb{R}^3 , parameterized by s (not necessarily its arc length), one can define a reference frame as in figure



$$p' = \frac{dp}{ds} \quad p'' = \frac{d^2p}{ds^2}$$

derivatives w.r.t. the parameter

$$t(s) = p'(s) / \|p'(s)\|$$

unit tangent vector

$$n(s) = p''(s) / \|p''(s)\|$$

unit normal vector
(\in osculating plane)

$$b(s) = t(s) \times n(s)$$

unit binormal vector

- general expression of path curvature (at a path point $p(s)$)

$$\kappa(s) = \|p'(s) \times p''(s)\| / \|p'(s)\|^3$$

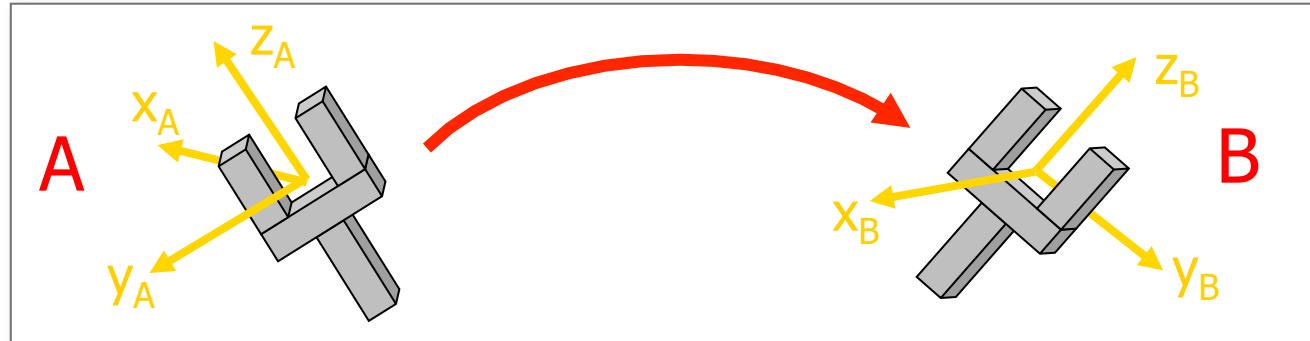


Optimal trajectories

- for Cartesian robots (e.g., PPP joints)
 1. the straight line joining two position points in the Cartesian space is **one** path that can be executed in **minimum time** under velocity/acceleration constraints (but other such paths may exist, if (joint) motion can also be **not coordinated**)
 2. the optimal timing law is of the bang-coast-bang type in acceleration (in this special case, also in terms of actuator torques)
- for articulated robots (with at least a R joint)
 - 1. e 2. are no longer true in general in the **Cartesian** space, but time-optimality still holds in the **joint** space when assuming **bounds** on **joint velocity/acceleration**
 - straight line paths in the joint space **do not correspond** to straight line paths in the Cartesian space, and vice-versa
 - bounds on joint acceleration are **conservative** (though **kinematically tractable**) w.r.t. actual ones on actuator torques, which involve the robot dynamics
 - when changing robot configuration/state, different torque values are needed to impose the same joint accelerations



Planning orientation trajectories



- using minimal representations of orientation (e.g., ZXZ Euler angles ϕ, θ, ψ), we can plan independently a trajectory for each component
 - e.g., a linear path in space $\phi \theta \psi$, with a cubic timing law
 ⇒ but poor prediction/understanding of the resulting intermediate orientations
- alternative method: based on the axis/angle representation
 - determine the (neutral) axis r and the angle θ_{AB} : $R(r, \theta_{AB}) = R_A^T R_B$ (rotation matrix changing the orientation from A to B ⇒ inverse axis-angle problem)
 - plan a timing law $\theta(t)$ for the (scalar) angle θ interpolating 0 with θ_{AB} (with possible constraints/boundary conditions on its time derivatives)
 - $\forall t, R_A R(r, \theta(t))$ specifies then the actual end-effector orientation at time t



A complete position/orientation Cartesian trajectory

- initial **given** configuration $q(0) = (0 \ \pi/2 \ 0 \ 0 \ 0 \ 0)^T$
- **initial** end-effector position $p(0) = (0.540 \ 0 \ 1.515)^T$
- **initial** orientation

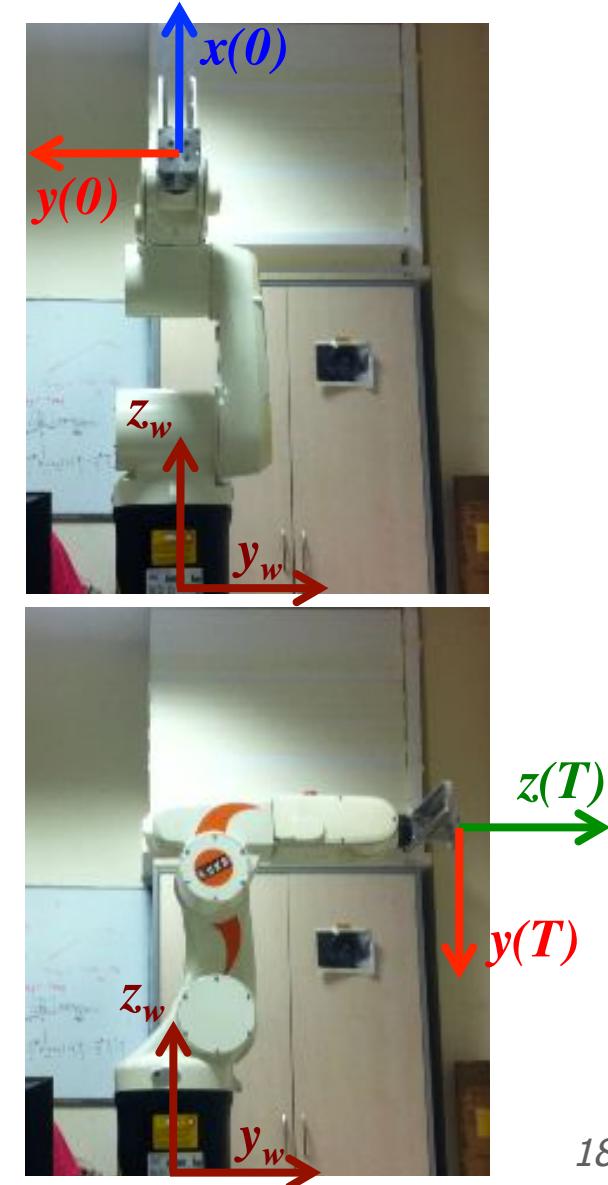
$$R(0) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

linear path
for position



axis-angle method
for orientation

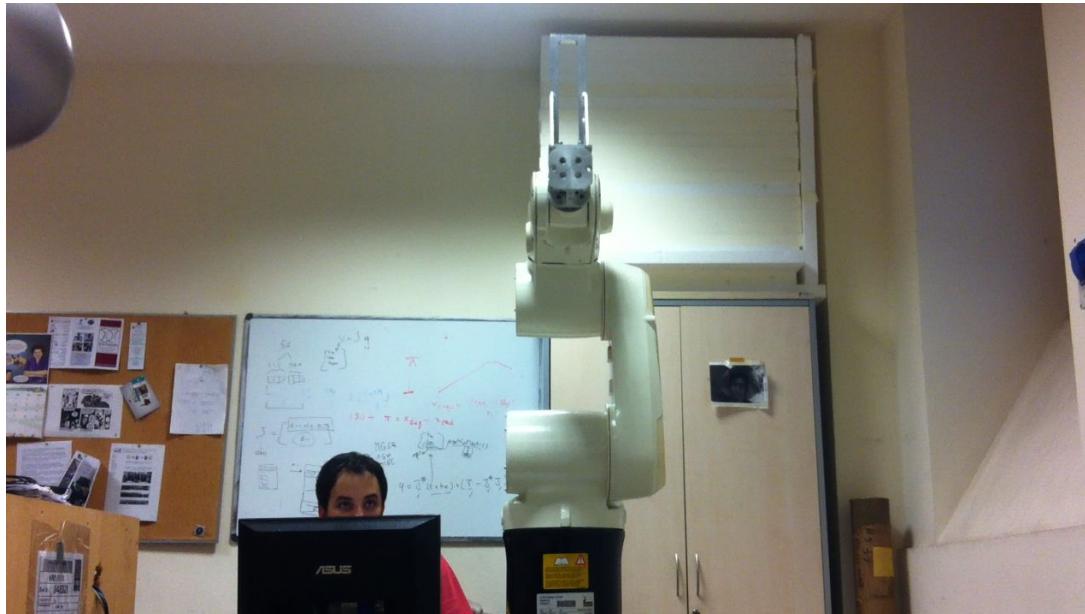
- **final** end-effector position $p(T) = (0 \ 0.540 \ 1.515)^T$
 - **final** orientation
- $$R(T) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$
- the final configuration is **NOT** specified a priori





Axis-angle orientation trajectory

video



$$L = \|p_{final} - p_{init}\| \\ = 0.763 \text{ [m]}$$

$$\omega = r\dot{\theta} \rightarrow \|\omega\| = |\dot{\theta}|$$

$$\dot{\omega} = r\ddot{\theta} \rightarrow \|\dot{\omega}\| = |\ddot{\theta}|$$

$$p(s) = p_{init} + s(p_{final} - p_{init}) \\ = (0.540 \quad 0 \quad 1.515)^T + s(-0.540 \quad 0.540 \quad 0)^T, \quad s \in [0, 1]$$

$$R_{init} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = R_{init}^T$$

$$R_{final} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$$



$$R_{init}^T R_{final} = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{pmatrix} = Rot(r, \theta_{if})$$

$$r = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}, \quad \theta_{if} = \frac{2\pi}{3} \text{ [rad]} \quad (= 120^\circ)$$

coordinated
Cartesian motion
with bounds on

$$v_{max} = 0.4 \text{ [m/s]}$$

$$a_{max} = 0.1 \text{ [m/s}^2]$$

$$\omega_{max} = \pi/4 \text{ [rad/s]}$$

$$\dot{\omega}_{max} = \pi/8 \text{ [rad/s}^2]$$



triangular
speed profile $\dot{s}(t)$
with minimum
time $T = 5.52 \text{ s}$
(imposed by the bounds
on linear motion)



$$R(s) = R_{init} Rot(r, \theta(s))$$

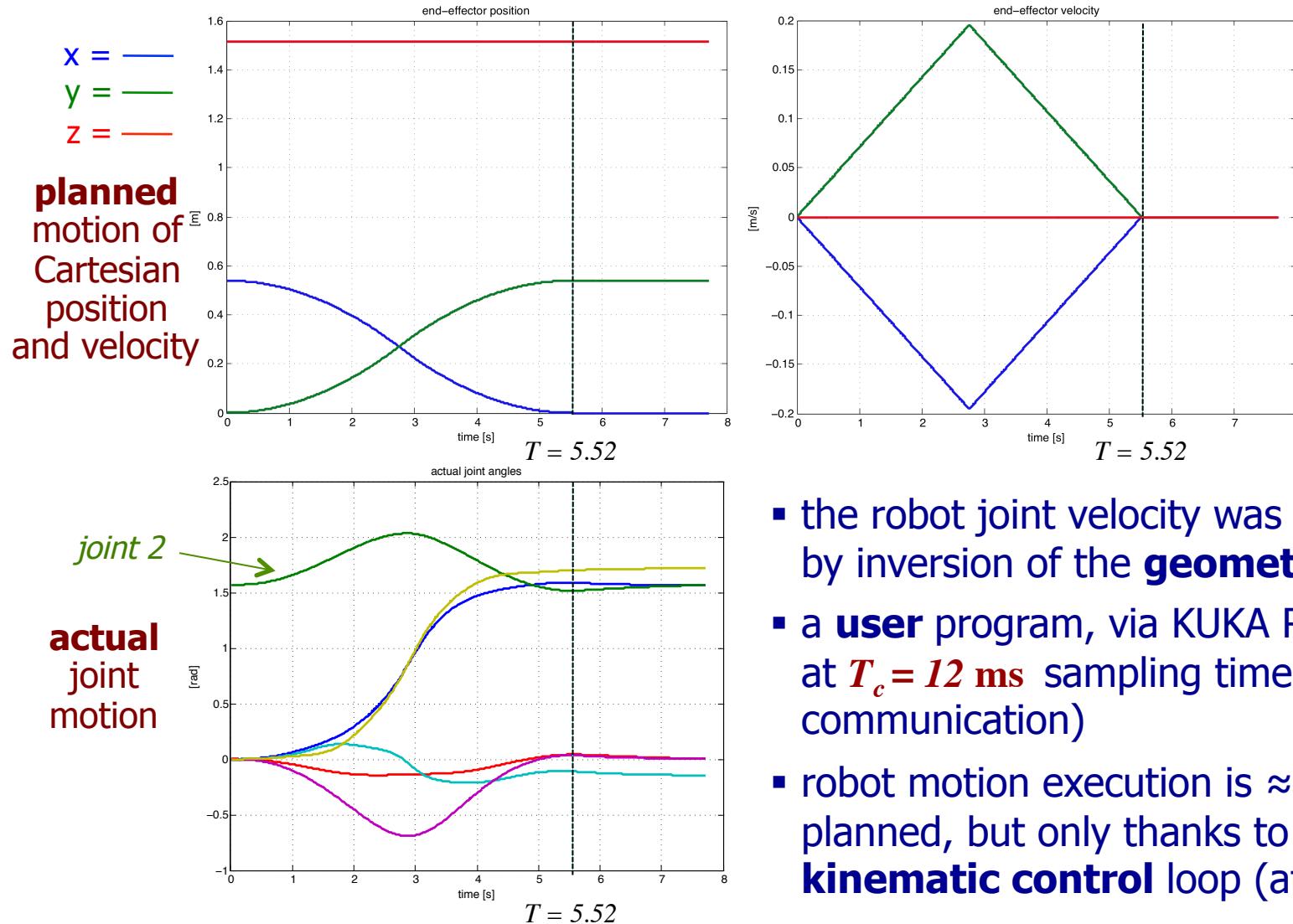


$$\theta(s) = s\theta_{if}, \quad s \in [0, 1]$$

$$s = s(t), \\ t \in [0, T]$$



Axis-angle orientation trajectory



triangular profile for linear speed
 $T = 5.52$ s

- the robot joint velocity was commanded by inversion of the **geometric** Jacobian
- a **user** program, via KUKA RSI interface at $T_c = 12$ ms sampling time (one-way communication)
- robot motion execution is \approx what was planned, but only thanks to an external **kinematic control** loop (at **task** level)

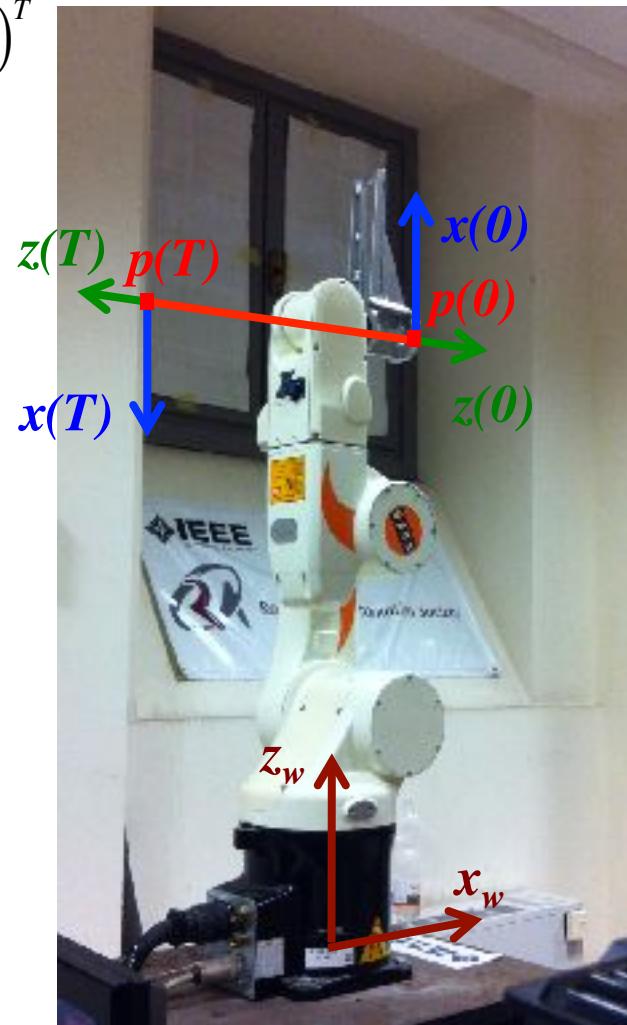


Comparison of orientation trajectories Euler angles vs. axis-angle method

- initial configuration $q(0) = (0 \quad \pi/2 \quad \pi/2 \quad 0 \quad -\pi/2 \quad 0)^T$
 - initial end-effector position $p(0) = (0.115 \quad 0 \quad 1.720)^T$
 - initial orientation
- $$R(0) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$
- initial Euler ZYZ angles $\phi_{ZYZ}(0) = (0 \quad \pi/2 \quad \pi)^T$

via a **linear path** (for position)

- final end-effector position $p(T) = (-0.172 \quad 0 \quad 1.720)^T$
 - final orientation
- $$R(T) = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$
- final Euler ZYZ angles $\phi_{ZYZ}(T) = (-\pi \quad \pi/2 \quad 0)^T$





Comparison of orientation trajectories Euler angles vs. axis-angle method

$$R_{init} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\Rightarrow \phi_{ZYX,init} = \begin{pmatrix} 0 \\ \pi/2 \\ \pi \end{pmatrix}$$

$$R_{final} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{pmatrix}$$

$$\Rightarrow \phi_{ZYX,final} = \begin{pmatrix} -\pi \\ \pi/2 \\ 0 \end{pmatrix}$$

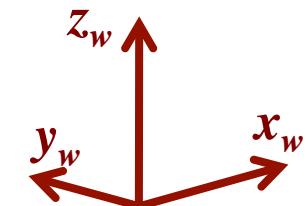
video



using ZYZ Euler angles

$$R^T R_{final} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

$$\Rightarrow r = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}, \quad \theta = \pi$$



video

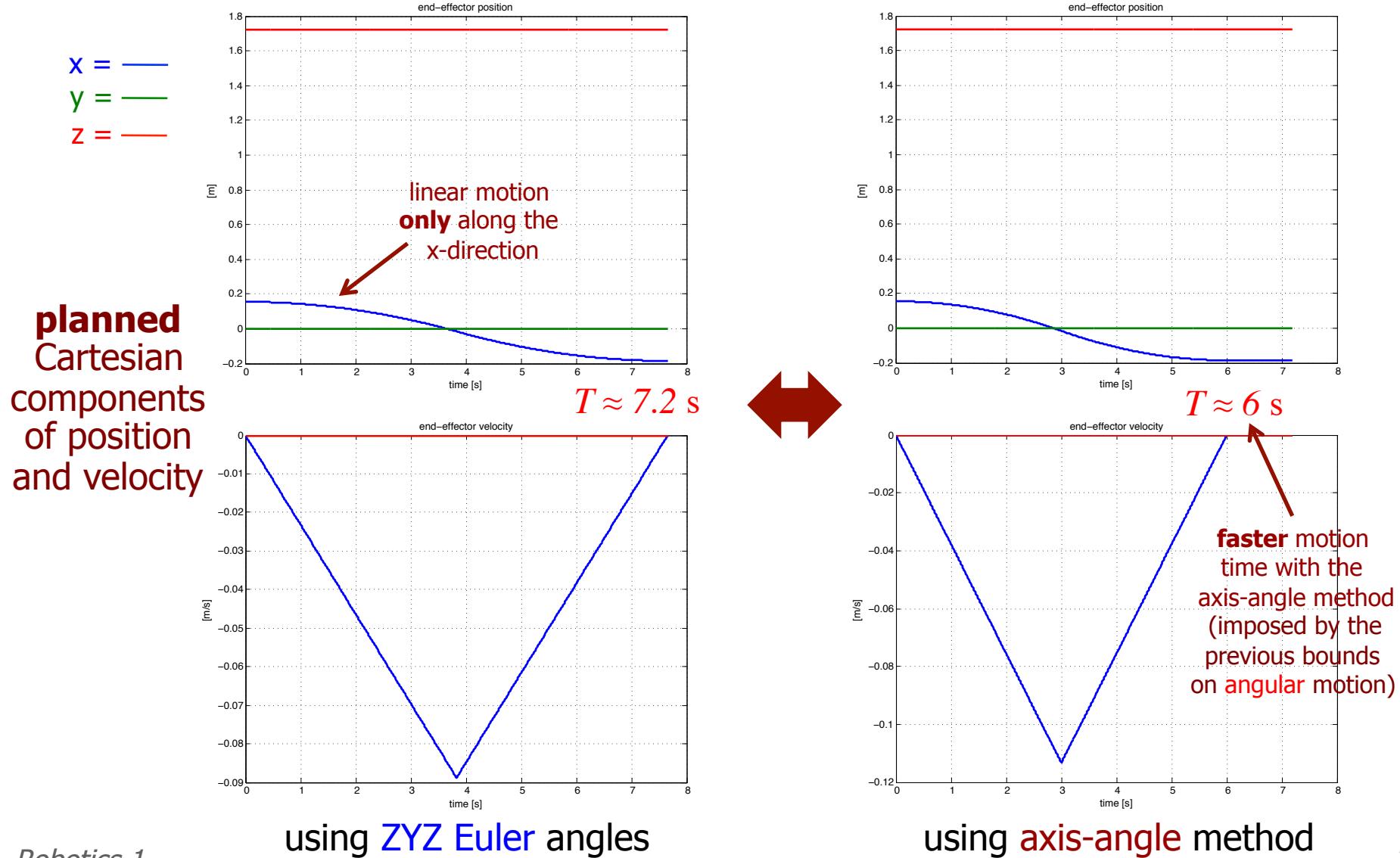


using axis-angle method



Comparison of orientation trajectories

Euler angles vs. axis-angle method



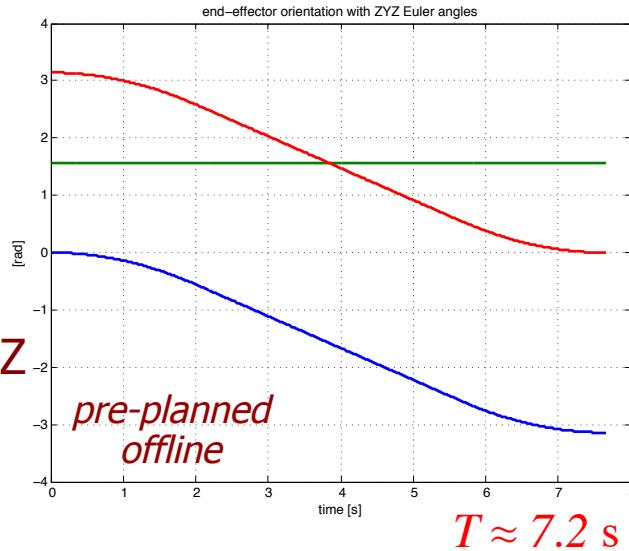


Comparison of orientation trajectories

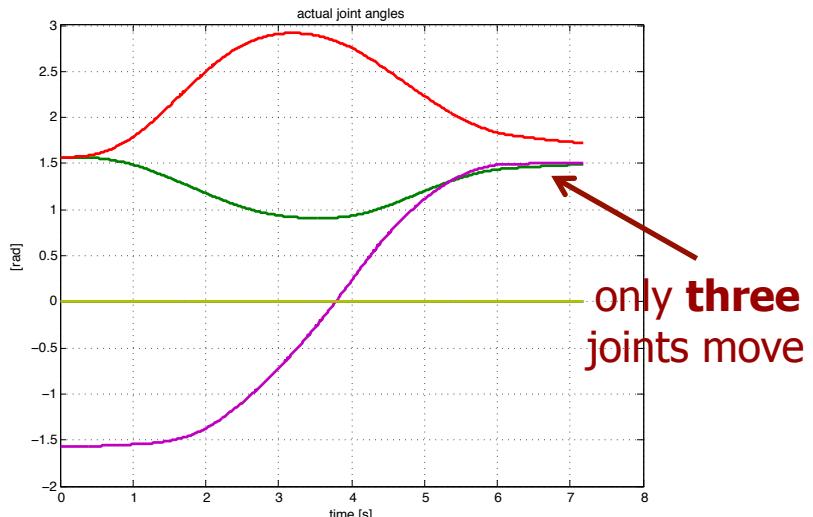
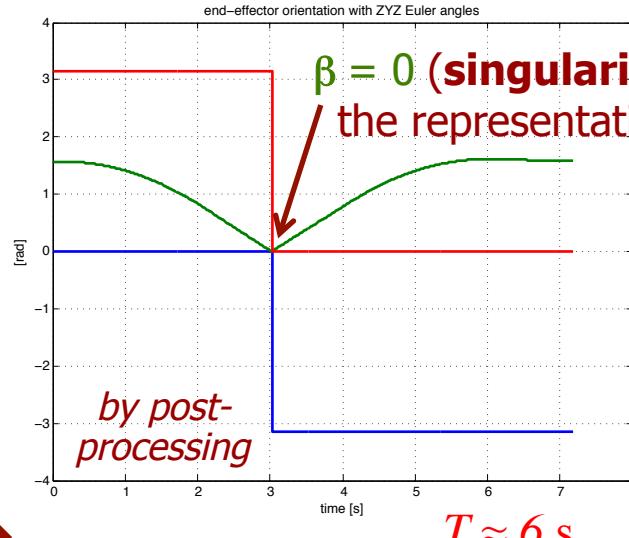
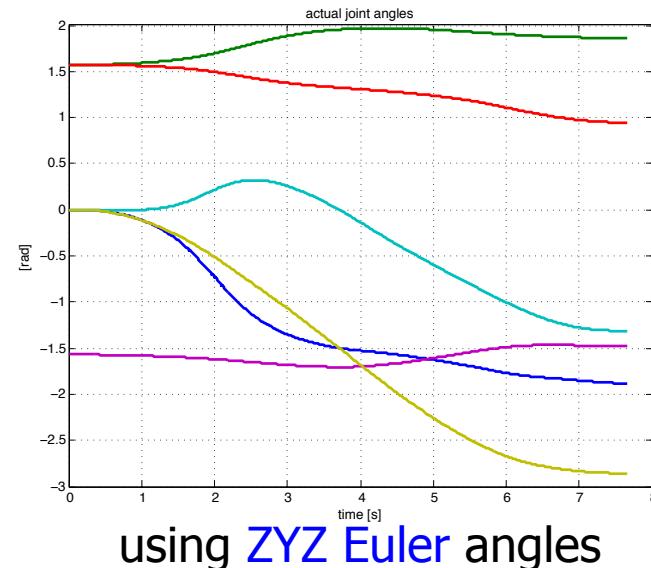
Euler angles vs. axis-angle method

$\alpha = \text{---}$
 $\beta = \text{---}$
 $\gamma = \text{---}$

orientation
in terms of ZYZ
Euler angles



actual
joint motion





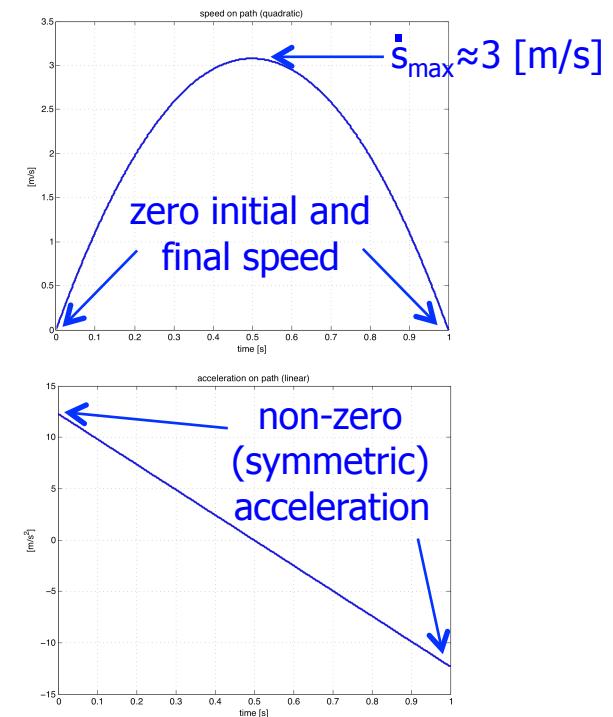
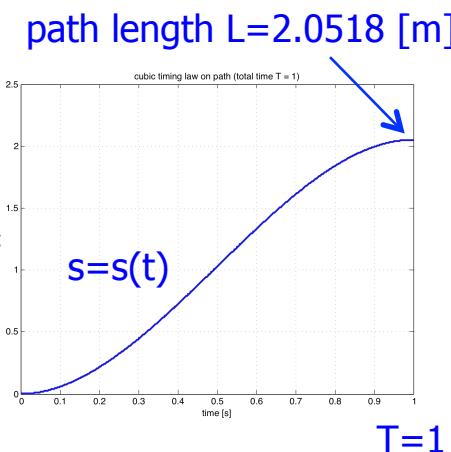
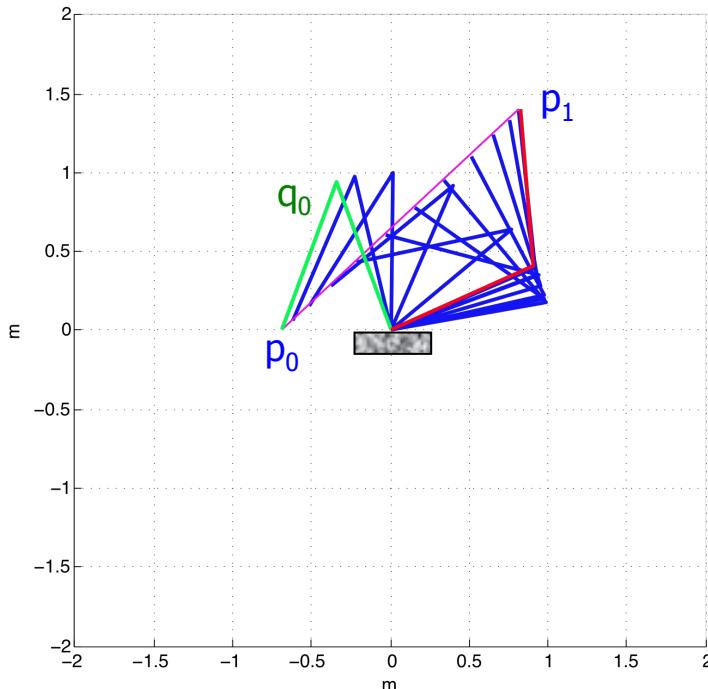
Uniform time scaling

- for a given path $p(s)$ (in joint or Cartesian space) and a given timing law $s(\tau)$ ($\tau=t/T$, T =“motion time”), we need to **check if existing bounds v_{\max} on (joint) velocity and/or a_{\max} on (joint) acceleration are violated or not**
 - ... unless such constraints have already been taken into account during the trajectory planning, e.g., by using a bang-coast-bang acceleration timing law
- **velocity scales linearly** with motion time
 - $dp/dt = dp/ds \cdot ds/d\tau \cdot 1/T$
- **acceleration scales quadratically** with motion time
 - $d^2p/dt^2 = (d^2p/ds^2 \cdot (ds/d\tau)^2 + dp/ds \cdot d^2s/d\tau^2) \cdot 1/T^2$
- if motion is unfeasible, **scale (increase)** time $T \rightarrow kT$ ($k>1$), based on the “most violated” constraint (max of the ratios $|v|/v_{\max}$ and $|a|/a_{\max}$)
- if motion is “too slow” w.r.t. the robot capabilities, **decrease** T ($k<1$)
 - in both cases, after scaling, there will be (at least) one instant of saturation (for at least one variable)
 - no need to re-compute motion profiles from scratch!



Numerical example - 1

- 2R planar robot with links of unitary length (1 [m])
- linear Cartesian path $p(s)$ from $q_0=(110^\circ, 140^\circ) \Rightarrow p_0=f(q_0)=(-.684, 0)$ [m] to $p_1=(0.816, 1.4)$, with rest-to-rest cubic timing law $s(t)$, $T=1$ [s]
- bounds in joint space: max (absolute) velocity $v_{\max,1}= 2$, $v_{\max,2}= 2.5$ [rad/s], max (absolute) acceleration $a_{\max,1}= 5$, $a_{\max,2}= 7$ [rad/s²]

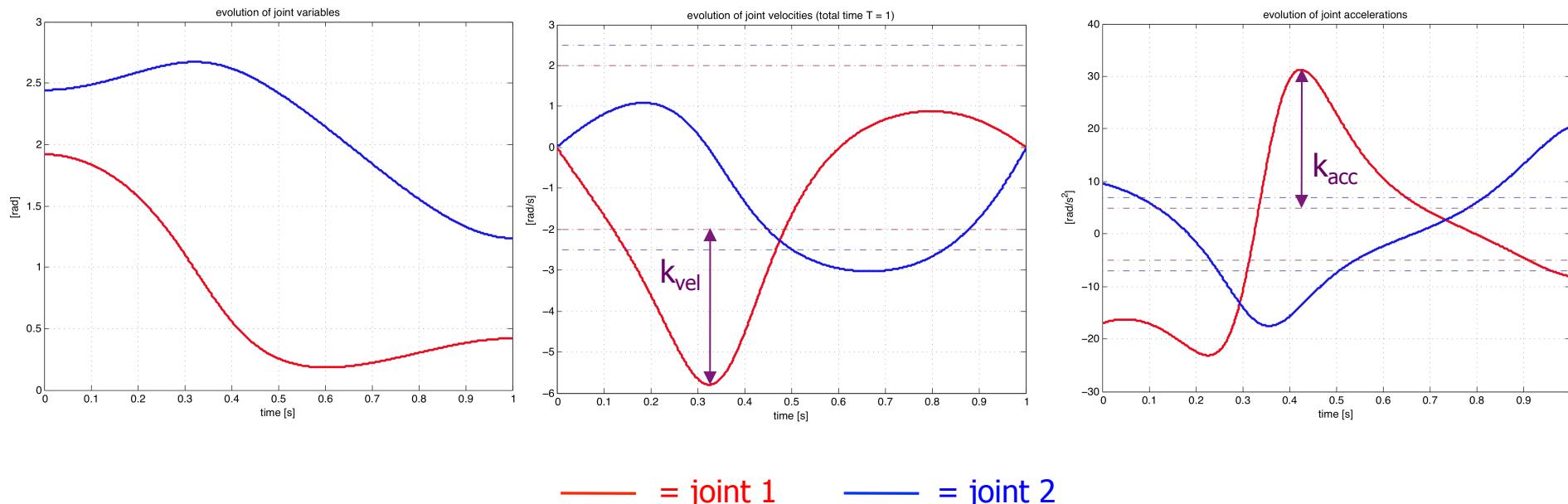




Numerical example - 2

- **violation** of both joint velocity and acceleration bounds with $T=1$ [s]
 - max relative violation of joint velocities: $k_{\text{vel}} = 2.898$
 - max relative violation of joint accelerations: $k_{\text{acc}} = 6.2567$
- minimum **uniform time scaling** of Cartesian trajectory to **recover feasibility**

$$k = \max \{1, k_{\text{vel}}, \sqrt{k_{\text{acc}}} \} = 2.898 \Rightarrow T_{\text{scaled}} = kT = 2.898 > T$$





Numerical example - 3

- scaled trajectory with $T_{\text{scaled}} = 2.898 \text{ [s]}$
 - speed [acceleration] on path and joint velocities [accelerations] scale linearly [quadratically]

