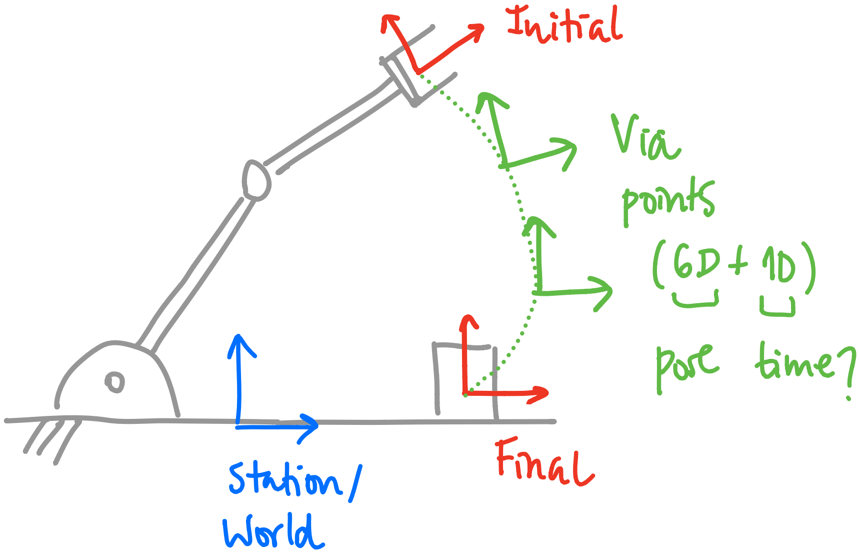


We want to compute the trajectories that join some initial and final poses (often the tool defined wrt a world frame), which additionally go smoothly through some via points

Continuous function + continuous first derivative

The complete trajectory is often called path, and two schemes are used for computing it: joint-scheme & cartesian-scheme.

① Path Generation with Joint-Schemes

Joints schemes are the easiest to compute and have the least issues: we set via points expressed in joint angles and these values are interpolated, usually with cubics. Singularities do not matter.

$$\vec{\theta}(t) \begin{cases} \vec{\theta}(t=0) = \vec{\theta}_0 = (\theta_{01}, \theta_{02}, \theta_{03}, \theta_{04}, \theta_{05}, \theta_{06})^T & \text{START} \\ \vec{\theta}(t=t_f) = \vec{\theta}_f = (\theta_{f1}, \dots, \theta_{f6})^T & \text{END / FINAL} \end{cases}$$

We treat each joint angle independently, and start defining function constraints so that we have a smooth interpolation.

$$\boxed{\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3}$$

We want to obtain the value of the coefficients a_i

Conditions:

$$\theta(t=0)=\theta_0$$
$$\theta(t=t_f)=\theta_f$$
$$\boxed{\begin{aligned}\dot\theta(t=0)&=0\\ \dot\theta(t=t_f)&=0\end{aligned}}$$

smooth start and end

Derivatives:

$$\dot\theta(t)=a_1+2a_2t+3a_3t^2$$
$$\ddot\theta(t)=2a_2+6a_3t$$

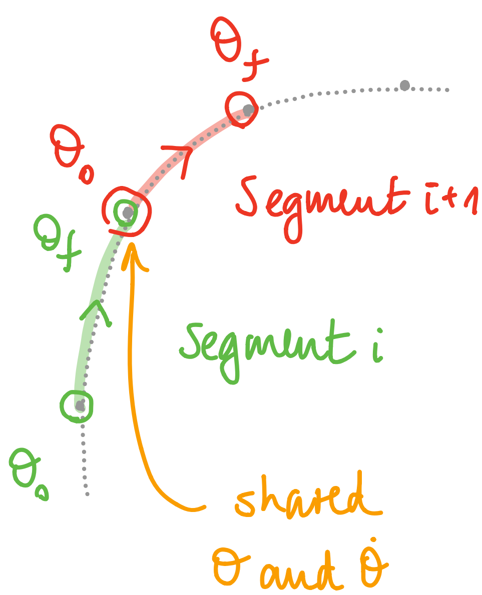
Coefficients after replacing the conditions in the $\theta, \dot\theta, \ddot\theta$ functions:

$$\boxed{\begin{aligned}&a_0=\theta_0\ ,\quad a_1=0\\ &a_2=\frac{3}{2t_f^2}(\theta_f-\theta_0)\qquad a_3=-\frac{2}{t_f^3}(\theta_f-\theta_0)\end{aligned}}$$

Cubic Polynomials of Joint Values with Via Points

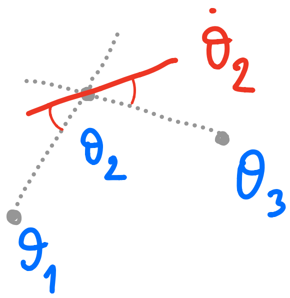
We apply the same equations as before for segments constituted by the paths between via points, but now we have a non-zero $\dot\theta_0$ & $\dot\theta_f$; therefore:

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



We have 2 options to choose $\dot{\theta}_0$ and $\dot{\theta}_f$:

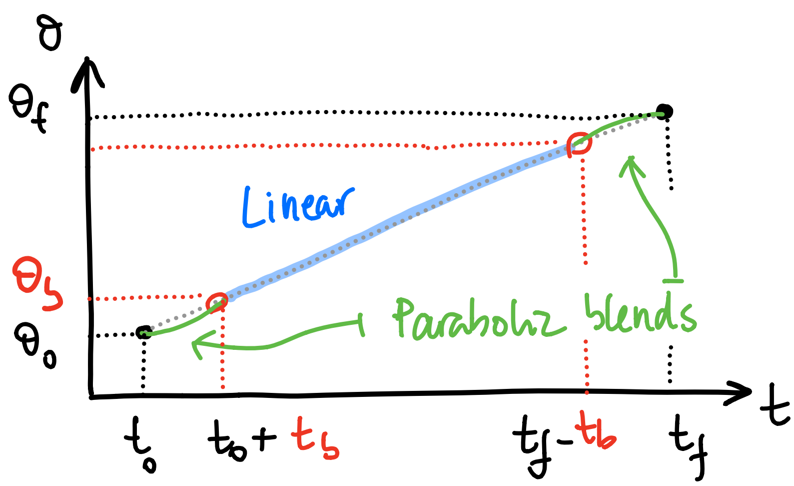
1) The user specifies the Cartesian velocities and we convert them to joint space using the Jacobian



2) A heuristic automatically selects them either to
   - ensure a sensible tangent between via point lines
   - ensure a continuous acceleration at the via points

     → a 5th order polynomial is required for that

Linear Functions with Parabolic Blends



We can similarly define θ interpolations which are linear except for at small regions at the start (0) and end (f) points, where a parabolic blend is forced: $\theta_b = \theta_0 + \frac{1}{2}\ddot{\theta} t_b^2$

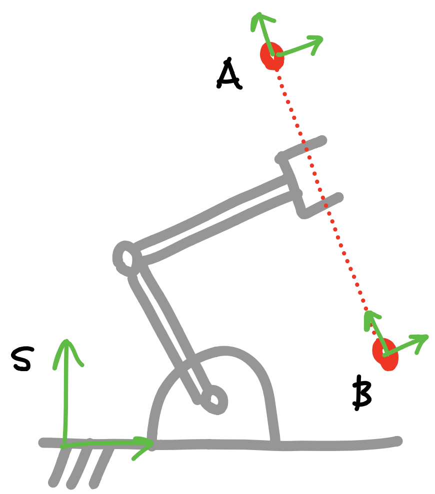
Via points can be considered, too.

② Path Generation with Cartesian-Schemes

Using a joint-scheme, the endeffector won't follow a known trajectory except for the via points. If the in-between trajectory is relevant, we can use Cartesian interpolation: we interpolate the path in cartesian space and then apply the inverse kinematics on all its points. That's:

- computationally more expensive
- riskier, because we might fall into singularities

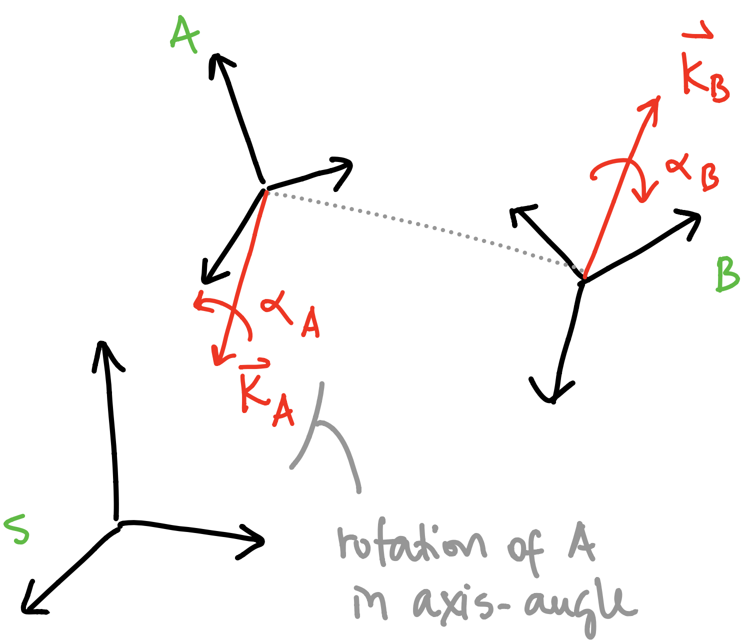
Cartesian Straight-Line Motion



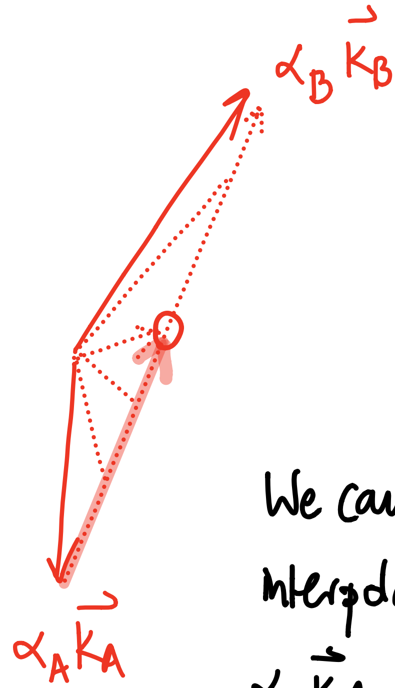
We want to interpolate a line in Cartesian space without cheating. Given $\underset{\sim}{H}$:

- all three components of the position can be interpolated linearly
- but not rotation components!
  ↳ for rotation, we use the angle-axis representation

$$ {}^{S}\underset{\sim}{H}_{A} = \begin{bmatrix} {}^{S}\underset{\sim}{R}_{A} & {}^{S}\vec{P}_{AORG} \\ 0\;0\;0 & 1 \end{bmatrix} \longrightarrow \underbrace{{}^{S}\vec{X}_{A}}_{\text{generalized position with axis-angle}} = \underbrace{\Big[\underbrace{{}^{S}\vec{P}_{AORG}}_{x,y,z \text{ coords of A origin wrt S}}{}^{T}, \overbrace{\underbrace{\alpha}_{\text{angle}} \cdot \underbrace{\vec{K}}_{\text{axis}}}^{{}^{S}\underset{\sim}{R}_{A}}{}^{T} \Big]^{T}}_{6\times 1} $$



If we go from A to B in a straight line rotations can change also linearly



We can linearly interpolate from $\alpha_A \vec{K}_A$ to $\alpha_B \vec{K}_B$, as with positions

If we use the generalized position vector $\vec{X} = (\vec{P}, \alpha \vec{K})$ with the axis-angle representation, we can interpolate as in the joint-scheme! However, if we use blends, the blend time for all degrees of freedom must be the same!

Problems with Cartesian Paths

1) Intermediate points might be unreachable, although start & end/final points are reachable.

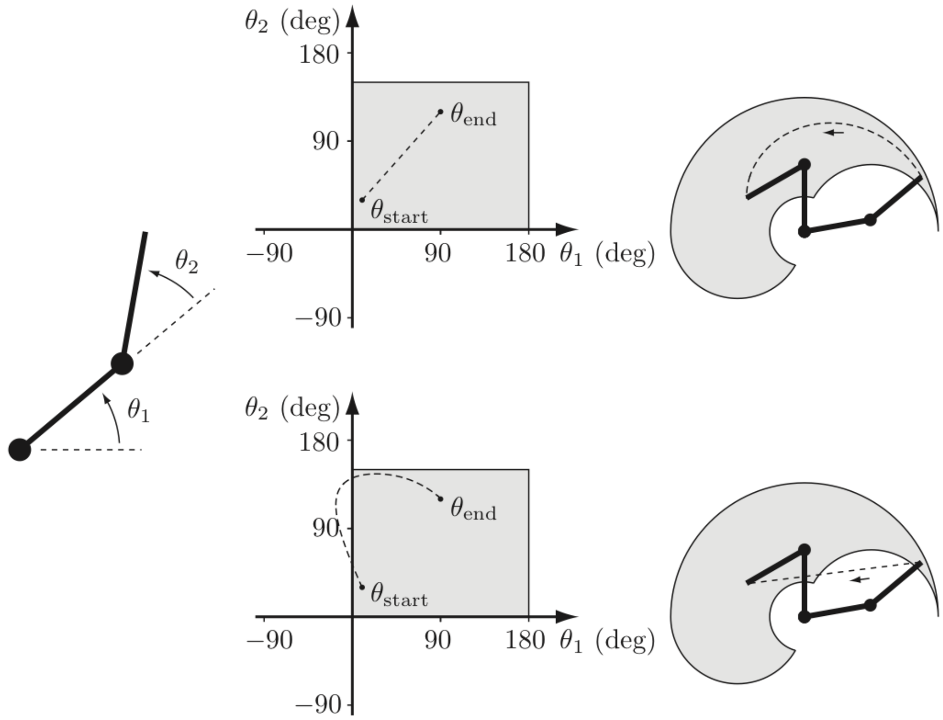


Figure 9.1: (Left) A 2R robot with joint limits $0^\circ \leq \theta_1 \leq 180^\circ$, $0^\circ \leq \theta_2 \leq 150^\circ$. (Top center) A straight-line path in joint space and (top right) the corresponding motion of the end-effector in task space (dashed line). The reachable endpoint configurations, subject to joint limits, are indicated in gray. (Bottom center) This curved line in joint space and (bottom right) the corresponding straight-line path in task space (dashed line) would violate the joint limits.

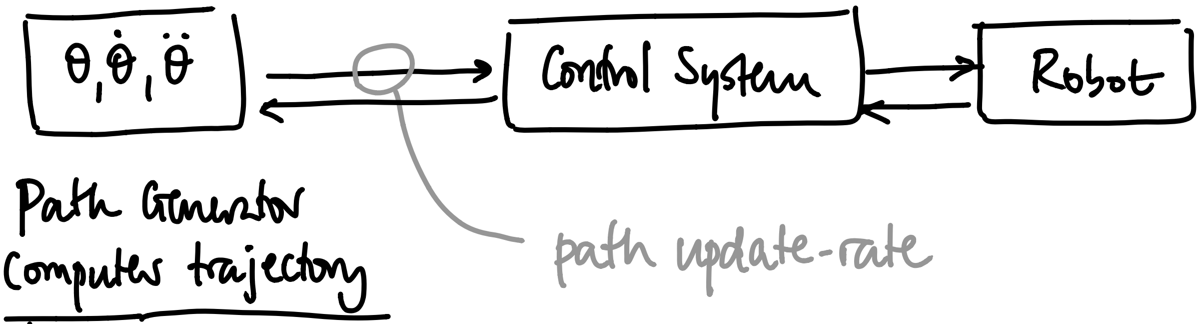
Lynch, p. 329

2) We might approach to singularities, which lead to high joint rates.

3) Start and goal/final points might be reachable in several configurations.

Therefore, the standard way to go is to use a joint-scheme for path generation — start, goal and via points can be defined in Cartesian space, but these are converted to joint space with the inv. kin. and then joint values interpolated.

③ Path Generation at Run Time



↳ The formulas/approaches explained in Section 1 are used. If cartesian schemes followed, Section 2 is used. However, the inverse kinematics converts each Cartesian pose to joint angle values. Usually, θ is differentiated to obtain $\dot{\theta}$ and $\ddot{\theta}$ instead of using the Jacobian...