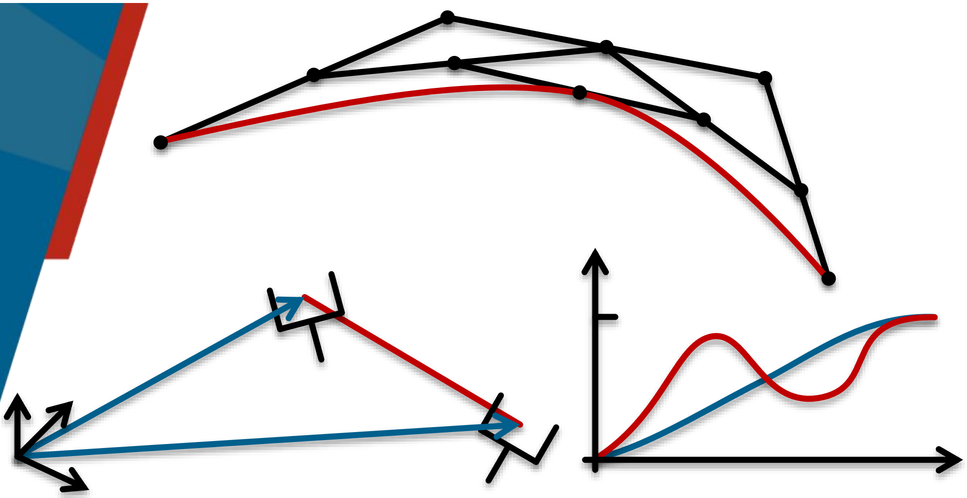
Continuous Path Control and Interpolation



Prof. Dr. Karsten Berns
Robotics Research Lab
Department of Computer Science
University of Kaiserslautern, Germany

Content

Basics of Continuous Path Control

- Movement of the robot are interpreted as state changes with respect to time (trajectory) relative to a stationary coordinate system (Cartesian space, joint space)
- Often constraints, boundary problems and other qualities are considered
- Given
  - Pose of the manipulator at start time (in Cartesian space $\vec{y}_{start}$and $\vec{\theta}_{start}$ in the joint space)
  - Pose of the manipulator at final time ($\vec{y}_{target}$ or $\vec{\theta}_{target}$)
- Wanted
  - Trajectory, which brings the manipulator from start to end point

Types of Planning

- PTP: Point to point
  - Planning of movement in configuration space
  - Time optimal path
  - Cartesian path not known
  - Use cases: Spot welding, handling tasks, …
- CP: Continuous path
  - Path control in Cartesian space
  - Path can be adapted to a desired shape
  - Path out of the work space is possible
  - Overstepping limits of the joints is possible
  - Use cases: Path welding, laser cutting, varnishing

PTP: Movement Phase of Joints

1. Acceleration
2. Movement with maximal or desired velocity
3. Slowing down, resting in target pose

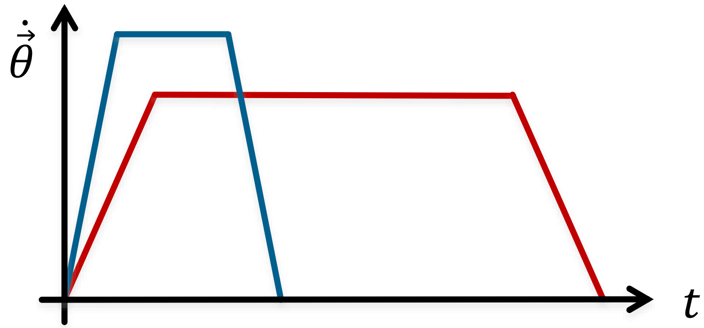
PTP: Phases of Planning

- Calculation of change for every actuating variable $\vec{\theta}_{target} - \vec{\theta}_{Start}$
- Determining the acceleration and deceleration duration
- Calculating the time, for which the joint will move at maximal velocity
  - Omitted if change is to small to reach maximal velocity
- Generation of trajectory
- Given actuating variables can be given by
  - Teach-in
  - Direct specification
  - Result of inverse kinematics

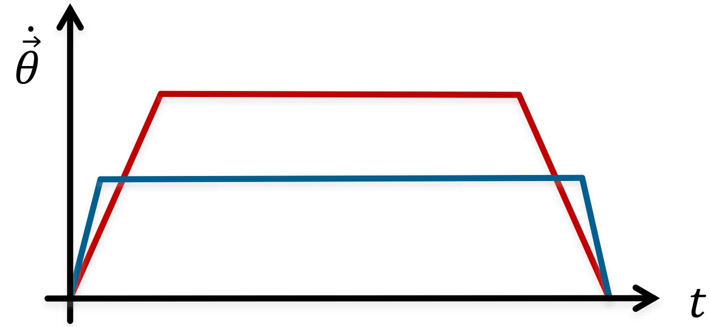
PTP: Types of Synchronization

- Asynchronous
  - Planning of axes independently
- Synchronous
  - Movement of all axes starts and ends simultaneously
  - Slowest joint as reference (leading axle)
  - Pro: Only little stress on mechanics
- Fully synchronous
  - Simultaneous acceleration and deceleration
  - Pros: Smooth movement in Cartesian space
  - Cons: Acceleration needs to be specified

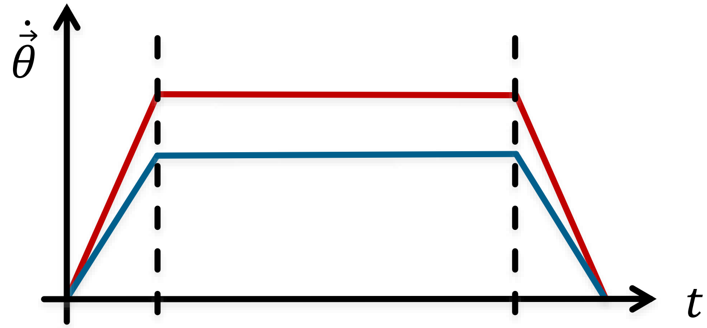
PTP: Types of Synchronization



Asynchronous PTP



Synchronous PTP



Fully synchronous PTP

PTP: Point-to-Point Control

- Pros
  - Joint trajectories are easy to calculate
  - No singularities during computation
  - Robot specific constraints, e.g. angle limitations, maximal speed and acceleration, are easy to consider
  - Time optimal path of movement
- Cons
  - Exact Cartesian path is difficult to foresee

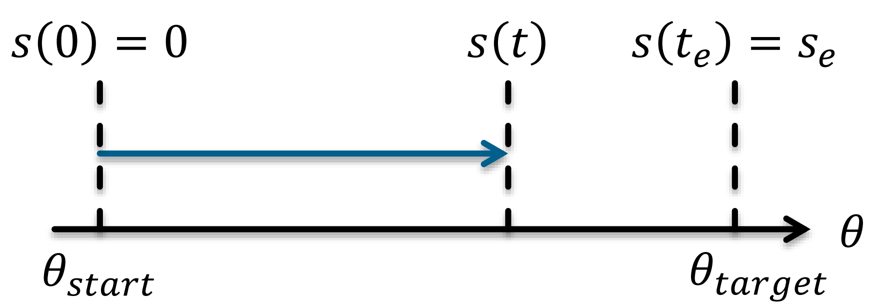
PTP: Constraints

- Start and target state are known
  - $\vec{\theta}(t_{start}) = \vec{\theta}_{start}$
  - $\vec{\theta}(t_{target}) = \vec{\theta}_{target}$
- Velocity is zero at start and end
  - $\dot{\vec{\theta}}(t_{start}) = \vec{0}$
  - $\dot{\vec{\theta}}(t_{target}) = \vec{0}$
- Working space, velocity and acceleration are limited

  $\vec{\theta}_{\min} \leq \vec{\theta}(t_j) \leq \vec{\theta}_{\max} \quad \vec{0} \leq \dot{\vec{\theta}}(t_j) \leq \dot{\vec{\theta}}_{\max} \quad \ddot{\vec{\theta}}_{\min} \leq \ddot{\vec{\theta}}(t_j) \leq \ddot{\vec{\theta}}_{\max}$
  - Limits can be chosen independent of mechanics, e.g. fast acceleration during parallel slow deceleration

PTP: Phases of Control

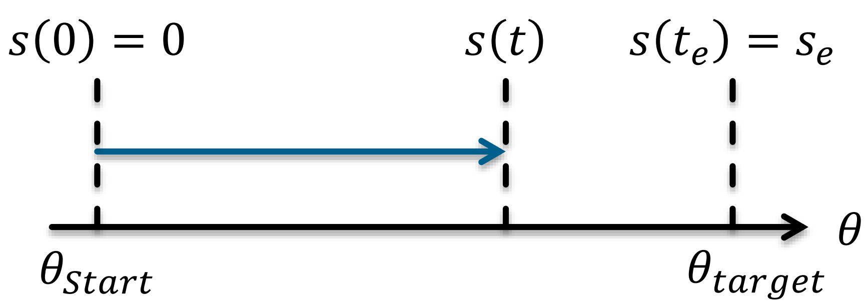
- Path parameter $s(t)$: describes …
  - … distance which should be covered for linear joints
  - … angle which should be rotated for rotational joints
- Given
  - General parameters $s(t), v(t), a(t)$
  - Maximal velocity v_{max} and acceleration a_{max}
  - Start and target position $\theta_{Start}, \theta_{target}$



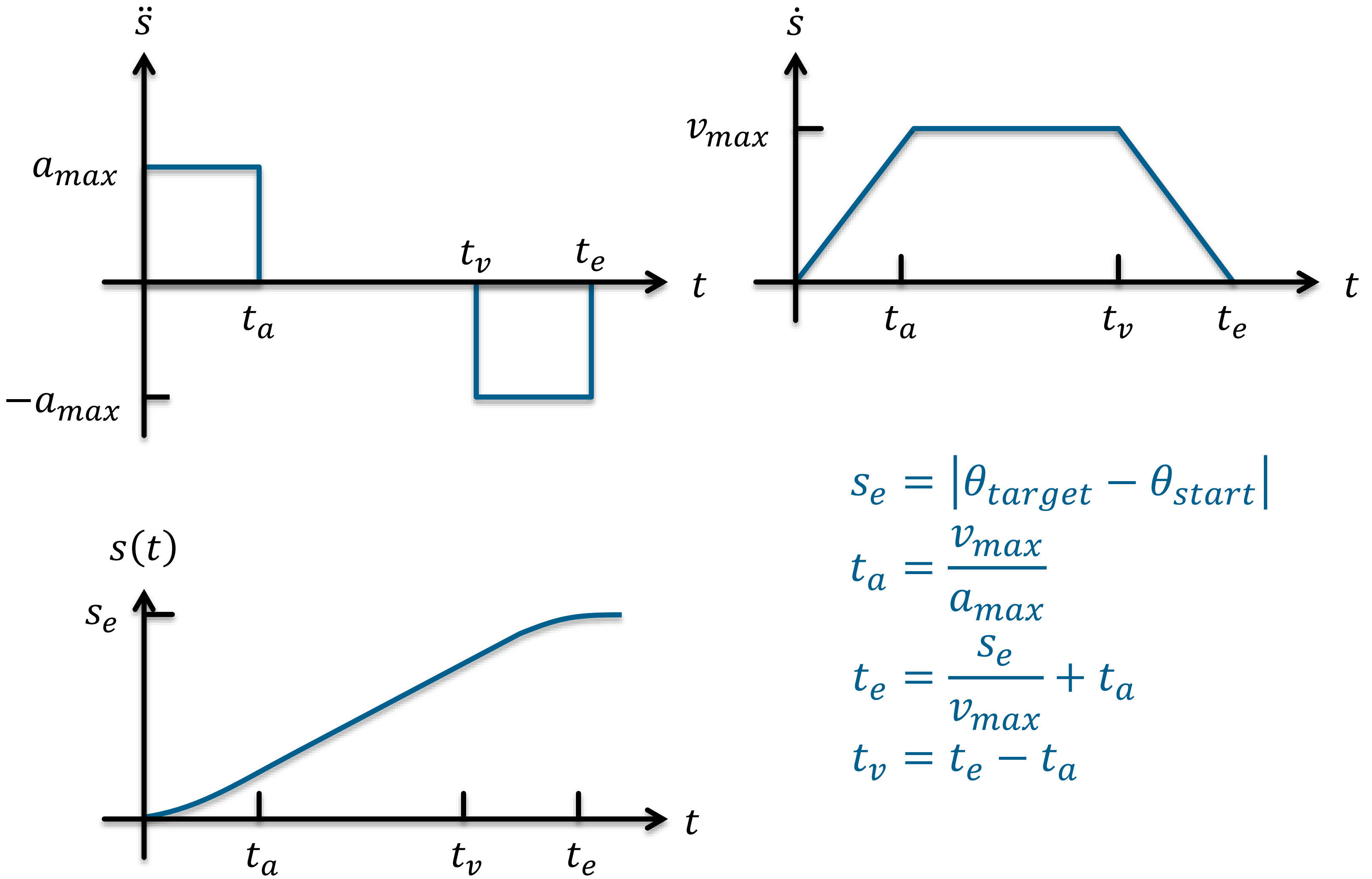
$$s(0) = \dot{s}(0) = v(0) = 0$$
$$\dot{s}(t_e) = v(t_e) = 0$$

PTP: Phases of Control

1. Calculation of path which should be covered s_e for each joint
$$s_e = \left|\theta_{target} - \theta_{start}\right|$$
2. Modification of inputs v_{max} and a_{max} for synchronous or fully synchronous PTP
3. Calculation of in-motion time t_e, acceleration time t_a and start of deceleration time t_v
4. Interpolation: Calculation of intermediate points $s(t), \dot{s}(t), \ddot{s}(t)$
5. Determination of reference values $\theta(t), \dot{\theta}(t), \ddot{\theta}(t)$



PTP: Square Wave Graphs for Interpolation



$$s_e = \left|\theta_{target} - \theta_{start}\right|$$

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

$$t_e = \frac{s_e}{v_{max}} + t_a$$

$$t_v = t_e - t_a$$

PTP: Calculation of Parameters

- Acceleration time $t_a = \frac{v_{\max}}{a_{\max}}$
- Integration of velocities
  $s_e = s(t_e) = v_{\max} \cdot t_a + v_{\max} \cdot (t_v - t_a) = v_{\max} \cdot t_a + v_{\max} \cdot (t_e - 2 \cdot t_a)$
- Calculation of in-motion time

$$t_e = \frac{s_e}{v_{\max}} + t_a = \frac{s_e}{v_{\max}} + \frac{v_{\max}}{a_{\max}}$$

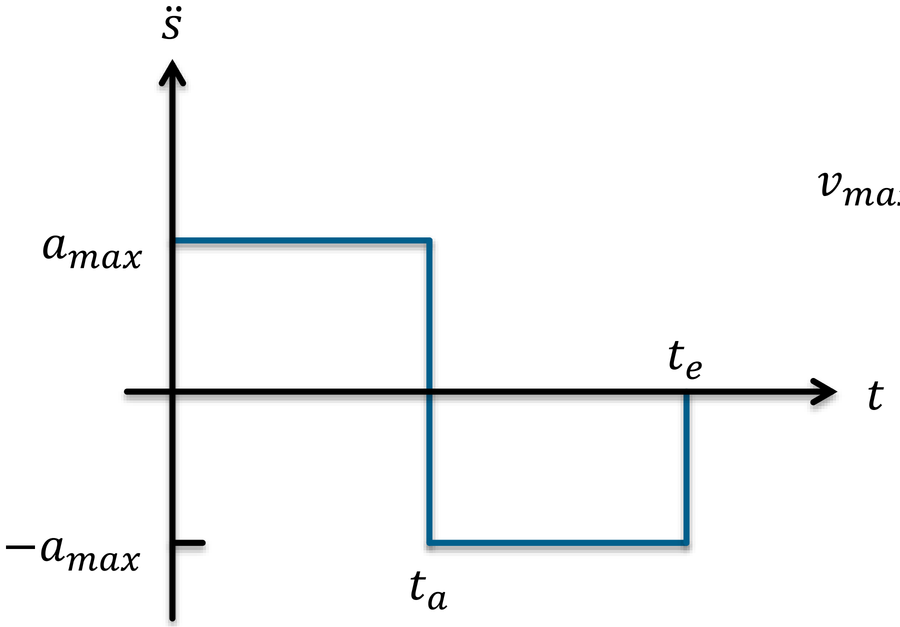
- Parameter for PTP

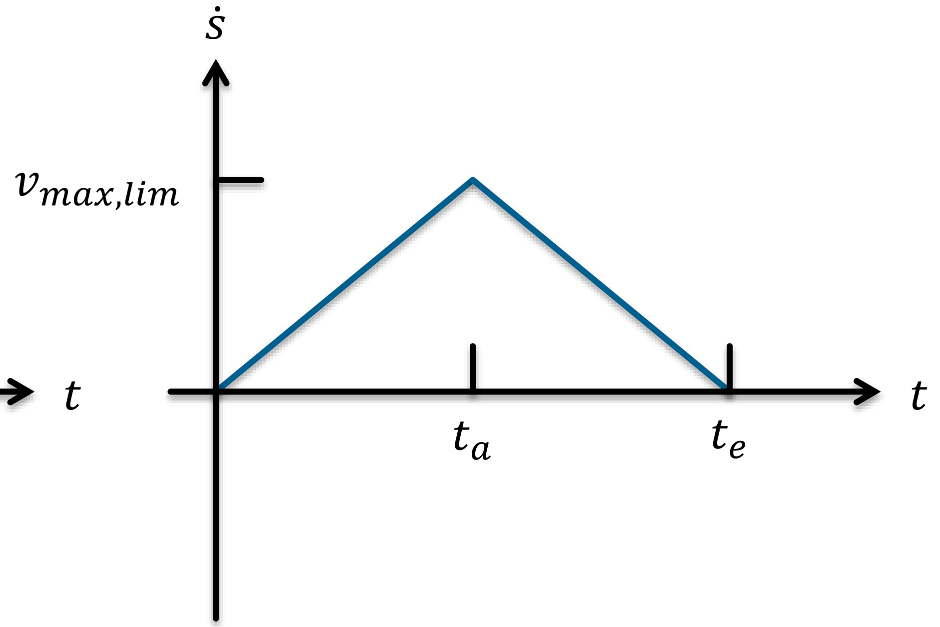
	$\ddot{s}(t)$	$\dot{s}(t)$	$s(t)$
$0 \leq t \leq t_a$	$a_{\max}$	$a_{\max} \cdot t$	$\frac{1}{2} \cdot a_{\max} \cdot t^2$
$t_a \leq t \leq t_v$	0	$v_{\max}$	$v_{\max} \cdot t - \frac{1}{2} \cdot \frac{v_{\max}^2}{a_{\max}}$
$t_v \leq t \leq t_e$	$-a_{\max}$	$v_{\max} - a_{\max} \cdot (t - t_v)$	$v_{\max} \cdot (t_e - t_a) - \frac{a_{\max}}{2} \cdot (t_e - t)^2$

PTP: Time Optimal Path

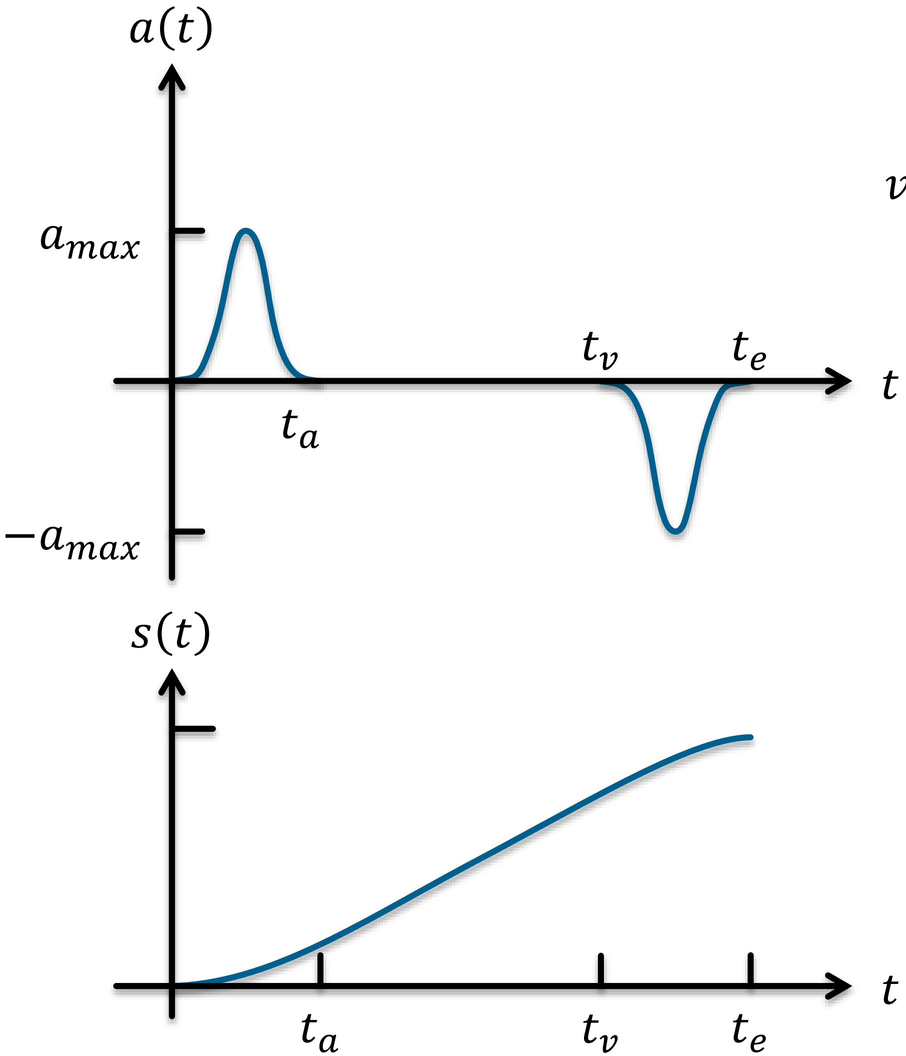
- If v_{max} is too big compared to acceleration and path length, a time optimal path can be calculated by:

$$s_e = t_a \cdot v_{max,lim} = \frac{v_{max,lim}^2}{a_{max}} \Rightarrow \sqrt{a_{max} \cdot s_e} \leq v_{max}$$





PTP: Sine Wave Graphs for PTP-Control



$$s_e = \left|\theta_{target} - \theta_{start}\right|$$

$$t_a = \frac{2 \cdot v_{max}}{a_{max}}$$

$$t_e = \frac{s_e}{v_{max}} + t_a$$

$$t_v = t_e - t_a$$

PTP: Sine Wave Graphs for PTP-Control

$$\ddot{s}(t) = a_{\max} \cdot \sin^2\left(\frac{\pi}{t_a} \cdot t\right) \tag{10.1}$$

- Integrating (10.1) over time yields the velocity

$$\dot{s}(t) = a_{\max} \cdot \left(\frac{1}{2} \cdot t - \frac{t_a}{4 \cdot \pi} \cdot \sin\left(\frac{2 \cdot \pi}{t_a} \cdot t\right)\right) \tag{10.2}$$

- For $t = t_a$ one gets v_{max} and (10.2) yields

$$t_a = \frac{2 \cdot v_{\max}}{a_{\max}} \tag{10.3}$$

PTP: Sine Wave Graphs for PTP-Control

- Covered path or angles during acceleration time can be calculated by integrating (10.2) …

$$s(t) = a_{\max} \cdot \left(\frac{1}{4} \cdot t^2 + \frac{t_a^2}{8 \cdot \pi} \cdot \left(\cos\left(\frac{2 \cdot \pi}{t_a} \cdot t \right) - 1 \right) \right) \tag{10.4}$$

- … over the whole covered path or angle distance

$$s_e = 2 \cdot s(t_a) + v_{\max} \cdot (t_e - 2 \cdot t_a)$$

$$s(t_a) = \frac{1}{4} \cdot a_{\max} \cdot t_a^2 = \frac{v_{\max}^2}{a}$$

$$t_e = \frac{s_e}{v_{\max}} + \frac{2 \cdot v_{\max}}{a_{\max}} = \frac{s_e}{v_{\max}} + t_a \tag{10.5}$$

PTP: Sine Wave Graphs for PTP-Control

- During phase of uniform movement

$$\dot{s}(t) = v_{\max}$$

$$s(t) = s(t_a) + v_{\max} \cdot (t - t_a) = v_{\max} \cdot \left(t - \frac{1}{2} \cdot t_a\right) \qquad (10.6)$$

- Velocity and path during deceleration

$$\dot{s}(t) = v_{\max} - \int_{t-t_v}^{t} a(\tau - t_v) \cdot d\tau$$

$$= v_{max} - a_{max} \cdot \left(\frac{1}{2} \cdot (t - t_v) - \frac{t_a}{4 \cdot \pi} \cdot \sin\left(\frac{2 \cdot \pi}{t_a} \cdot (t - t_v)\right)\right)$$

$$s(t) = s(t_v) + \int_{t-t_v}^{t} \dot{s}(\tau - t_v) \cdot d\tau \qquad (10.7)$$

$$= \frac{a_{\max}}{2} \cdot \left[t_e \cdot (t + t_a) - \frac{t^2 + t_e^2 + 2 \cdot t_a^2}{2} + \frac{t_a^2}{4 \cdot \pi^2} \cdot \left(1 - \cos\left(\frac{2 \cdot \pi}{t_a} \cdot (t - t_v)\right)\right)\right]$$

Synchronous PTP: Approach

1. Determine path length $s_{e,i}$ for each joint i
2. Determine PTP-parameter $v_{max,i}, a_{max,i}$
3. Calculate time in-motion $t_{e,i}$
4. Determine axes with maximal time in-motion $t_e = t_{e,max} = \max(t_{e,i})$
   - Determined axle is leading axle
5. Set $t_{e,i} = t_e$ for all joints
6. Calculate new velocities for each joint

Synchronous PTP

- Transformation of time in-motion t_e and calculation of new velocities
- Graphs

$$t_e = \frac{s_{e,i}}{v_{\max,i}} + \frac{v_{\max,i}}{a_{\max,i}}$$

  - After transformation $v_{\max,i}^2 - v_{\max,i} \cdot a_{\max,i} \cdot t_e + s_{e,i} \cdot a_{\max,i} = 0$
  - Solution is the smaller value since else $2 \cdot t_{a,i} > t_e$ and

$$v_{\max,i} = \frac{a_{\max,i} \cdot t_e}{2} - \sqrt{\frac{a_{\max,i}^2 \cdot t_e^2}{4} - s_{e,i} \cdot a_{\max,i}}$$

- Sine wave path $v_{\max,i} = \frac{a_{\max,i} \cdot t_e}{4} - \sqrt{\frac{a_{\max,i}^2 \cdot t_e^2 - 8 \cdot s_{e,i} \cdot a_{\max,i}}{16}}$

Fully Synchronous PTP

- Takes acceleration and deceleration times into account
- Determination of leading axle with t_e and $t_a \rightarrow t_v = t_e - t_a$
- Determination of velocity and acceleration of the other axes via $v_{max,i} = \frac{s_{e,i}}{t_v}$ and $a_{max,i} = \frac{v_{max,i}}{t_a}$

Path Control: Continuous Path (CP) Control in Cartesian Space

- Description of trajectory as a function of the TCPs pose
  - E.g. with a description vector: $\vec{y}_{TCP}(t),\ \dot{\vec{y}}_{TCP}(t), \ddot{\vec{y}}_{TCP}(t)$
- Function e.g. linear, polynomial or spline path
- Pros
  - Definition of trajectory explicitly in Cartesian space
  - Planning independent of robot kinematics
- Cons
  - Calculation of transformation to joint angles for each point of the trajectory needed
  - Planned trajectory not always executable (limits of working space, singularities of the robot)
  - Constraints of joints can not be taken into account

Continuous Path (CP) Control in Cartesian Space

- Given
  - Target position $\vec{p}_{target} = \left(x_{target}, y_{target}, z_{target}\right)^T$
  - Target orientation (Euler) $\vec{w}_{target} = \left(\alpha_{target}, \beta_{target}, \gamma_{target}\right)^T$
  - Intermediate point (optional) $\vec{p}_H = (x_H, y_H, z_H)^T$
  - Linear velocity and acceleration v_p, a_p
  - Rotational velocity and acceleration v_w, a_w
- Constraints
  - Maximal velocity and acceleration of each joint

$$\vec{y}_{target} = \begin{pmatrix} \vec{p}_{target} \\ \vec{w}_{target} \end{pmatrix}, (\vec{p}_H)$$

$$v_p, a_p, v_w, a_w$$

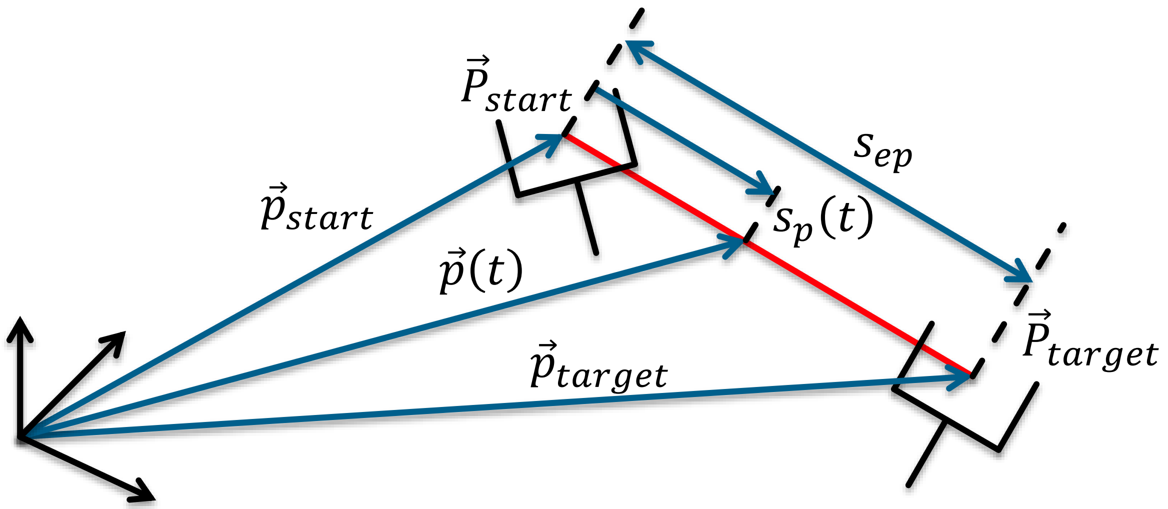
$$\xrightarrow{} \boxed{\text{Interpolation}} \xrightarrow{\vec{y}(t) = \begin{pmatrix} \vec{p}(t) \\ \vec{w}(t) \end{pmatrix}} \boxed{\text{Inverse transformation}} \xrightarrow{\vec{\theta}(\mathrm{t})}$$

$$\left(\dot{\vec{\theta}}(t), \ddot{\vec{\theta}}(t)\right)$$

CP: Linear Interpolation

- Path parameter $s_p(t)$ describes covered path at time t
- Complete path

$$s_{ep} = \left|\vec{p}_{target} - \vec{p}_{start}\right|$$
$$= \sqrt{\left(x_{target} - x_{start}\right)^2 + \left(y_{target} - y_{start}\right)^2 + \left(z_{target} - z_{start}\right)^2}$$



CP: Linear Interpolation

- Constraints

$$s_p(0) = \dot{s}_p(0) = v_p(0) = 0$$
$$\dot{s}_p(t_e) = v_p(t_e) = 0$$

- with

$$v_{\max} = v_p \qquad a_{\max} = a_p$$
$$t_e = t_{ep} \qquad t_a = t_{ap} \qquad t_v = t_{vp}$$
$$s_e = s_{ep} \qquad s = s_p$$

  $s_p(t)$ can be calculated from the already described equations in PTP, via sine or square wave

- Position of the TCP at time t

$$\vec{p}(t) = \vec{p}_{start} + s_p(t) \cdot \frac{\left(\vec{p}_{target} - \vec{p}_{start}\right)}{s_{ep}}$$

CP: Linear Interpolation

- Calculation of change in orientation analogous to calculation of change in position
- Complete orientation change

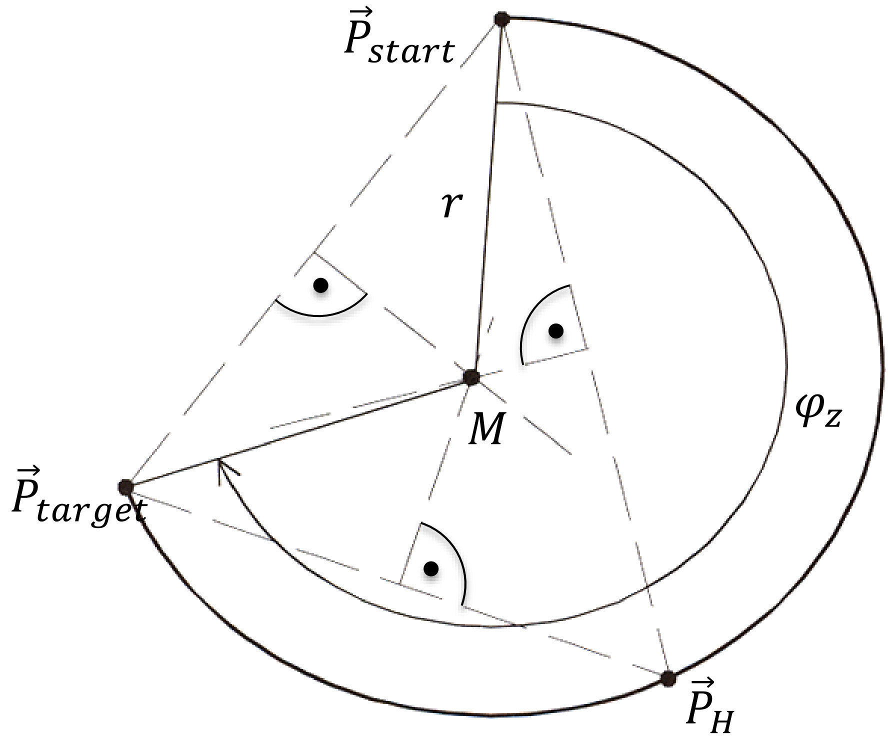
$$s_{ew} = \left|\vec{w}_{target} - \vec{w}_{start}\right|$$

$$= \sqrt{\left(\alpha_{target} - \alpha_{start}\right)^2 + \left(\beta_{target} - \beta_{start}\right)^2 + \left(\gamma_{target} - \gamma_{start}\right)^2}$$

- Position and orientation change should be finished at the same time
  - Adapt time in-motion to maximal one
  - Reduce velocity accordingly
  - $t_e = \max\left(t_{ep}, t_{ew}\right)$
- For a robot control the calculated Cartesian poses must be transformed to joint angles at every sampling interval

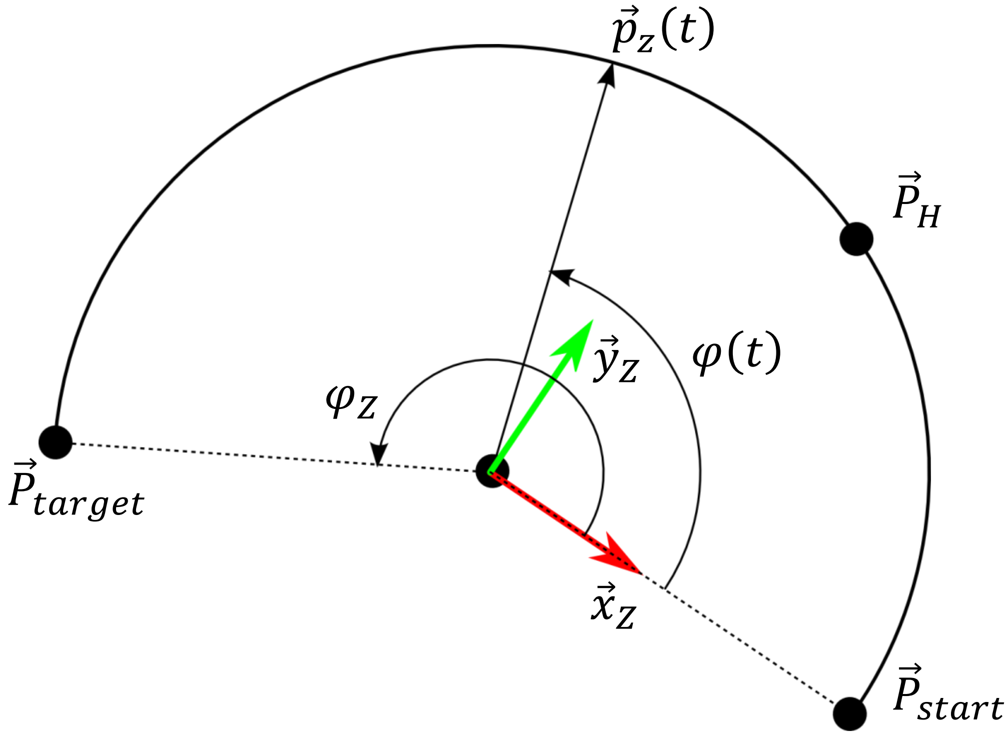
CP: Circular Interpolation

- Apart from lines often times also an arc of a circle can be used as parts of a path
- Easies model of an arc of a circle via a start point $\vec{P}_{Start}$, an end point $\vec{P}_{target}$ and an Intermediate point $\vec{P}_H$
- Can be determined via the intersection points of the perpendicular bisectors
  - Center point M
  - Radius r
  - Angle φ_z



CP: Circular Interpolation

- Path parameter $s(t)$ describes covered angle $\varphi(t)$
- Can be calculated as in linear CP with equations from PTP
- To calculate Cartesian position introduce subsidiary coordinate system XYZ_Z



CP: Circular Interpolation

- Position $\vec{p}_Z(t)$ on the arc of the circle XYZ_Z can be computed with r and $\varphi(t)$

$$\vec{p}_Z(t) = \begin{pmatrix} r \cdot \cos\big(\phi(t)\big) \\ r \cdot \sin\big(\phi(t)\big) \\ 0 \end{pmatrix}$$

- $\vec{\mathrm{p}}_Z(t)$ can be transformed homogeneously in the BCS
- Interpolation of orientation as in the linear interpolation case
- For a robot control the calculated Cartesian poses must be transformed to joint angles at every sampling interval

CP: Spline Interpolation

- If the trajectory of a kinematic system consists of linear segments, very high accelerations occur at each start and end point of a segment.
- To avoid the associated jerk and the high forces on the kinematics,
  - Either decelerate at each segment end, or smooth the paths.
- A suitable spline consisting of the segments $S_0, \dots, S_{n-1}$ would have to traverse the points $P_0, \dots, P_n$ of the trajectory,
- Usually, such splines are interpolated by means of polynomials of degree n.

CP: Piecewise Interpolation

- Path is defined by piecewise polynomials, called splines
- Usual case: Cubical splines

$$\vec{p}(t) = \vec{a}_3 \cdot t^3 + \vec{a}_2 \cdot t^2 + \vec{a}_1 \cdot t + \vec{a}_0$$

- $\vec{p}(t)$: Path between position $\vec{p}_{start}$ and $\vec{p}_{target}$, with a time duration of t_e
- 4 conditions are needed to calculate the parameters $\vec{a}_j$ of a spline $\vec{p}(t)$ uniquely
- Two conditions are described by the interpolation at the supporting points

$$\vec{p}(t = 0) = \vec{p}_{start}$$
$$\vec{p}(t = t_e) = \vec{p}_{target}$$

CP: Piecewise Interpolation

- The two remaining conditions can be determined by the desired velocity vectors

$$\dot{\vec{p}}(t=0) = \dot{\vec{p}}_{start}$$
$$\dot{\vec{p}}(t=t_e) = \dot{\vec{p}}_{target}$$

- Calculating the parameters from the described conditions yields:

$$\vec{a}_0 = \vec{p}_{start}$$
$$\vec{a}_1 = \dot{\vec{p}}_{Start}$$
$$\vec{a}_2 = \frac{3}{t_e^2}\left(\vec{p}_{target} - \vec{p}_{start}\right) - \frac{1}{t_e}\left(\dot{\vec{p}}_{target} + 2\dot{\vec{p}}_{start}\right)$$
$$\vec{a}_3 = -\frac{2}{t_e^3}\left(\vec{p}_{target} - \vec{p}_{start}\right) + \frac{1}{t_e^2}\left(\dot{\vec{p}}_{target} + \dot{\vec{p}}_{start}\right)$$

CP: Piecewise Interpolation - An Example

- Given
  $\vec{p}_I = \begin{pmatrix}0\\0\end{pmatrix}, \vec{p}_{II} = \begin{pmatrix}1\\2\end{pmatrix}, \vec{p}_{III} = \begin{pmatrix}2\\1\end{pmatrix}, \vec{p}_{IV} = \begin{pmatrix}4\\4\end{pmatrix}, \dot{\vec{p}}_I = \cdots = \dot{\vec{p}}_{IV} = \begin{pmatrix}1\\0\end{pmatrix}, t_e = 1$
- Solution: Parameter for the first polynomial; others analogously

$$\vec{a}_0 = \vec{p}_I \quad \vec{a}_2 = \frac{3}{1}(\vec{p}_{II} - \vec{p}_I) - \frac{1}{1}(\dot{\vec{p}}_{II} + 2\dot{\vec{p}}_I)$$

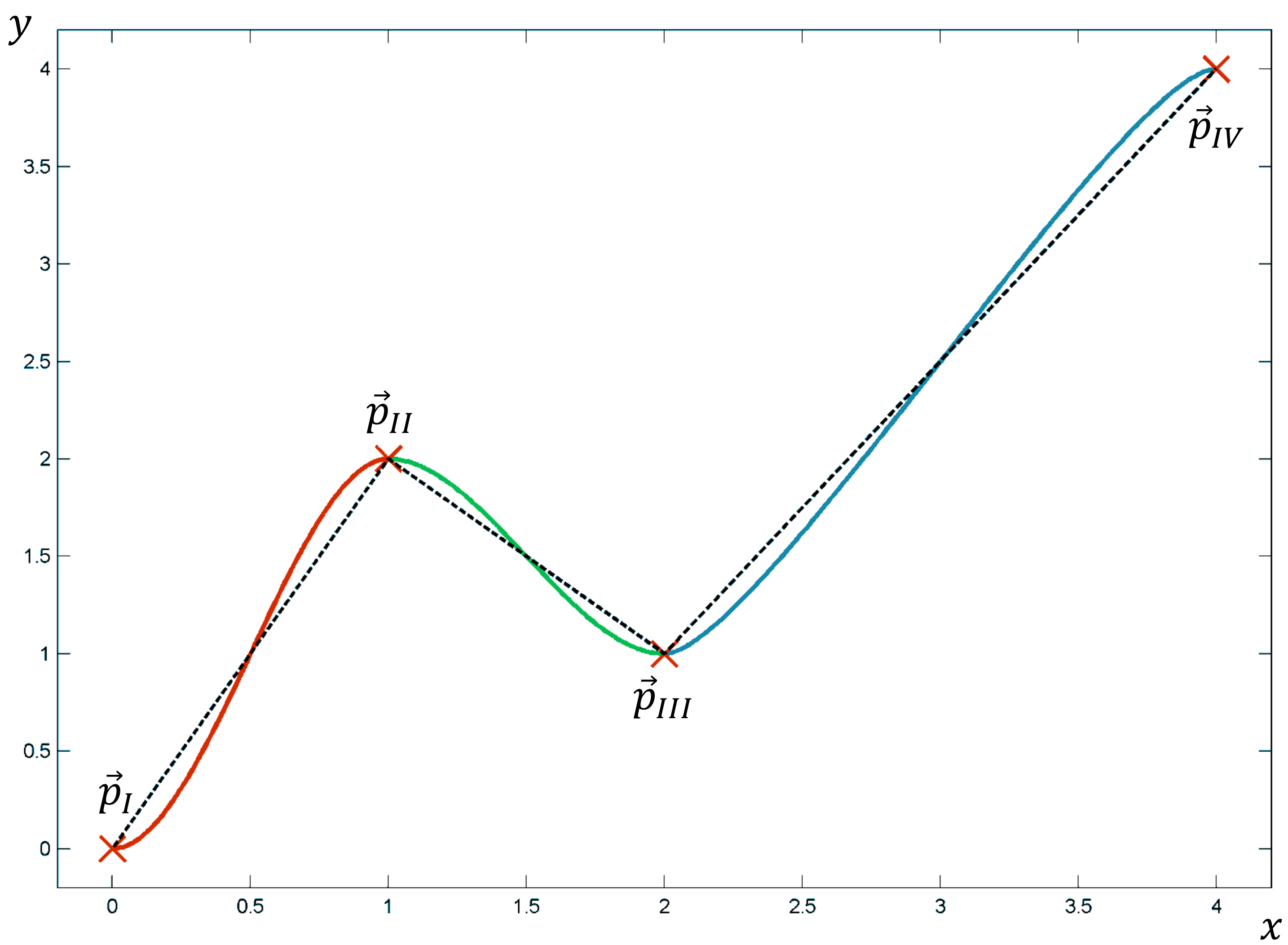
$$\vec{a}_1 = \dot{\vec{p}}_I \quad \vec{a}_3 = -\frac{2}{1}(\vec{p}_{II} - \vec{p}_I) + \frac{1}{1}(\dot{\vec{p}}_{II} + \dot{\vec{p}}_I)$$

$$\vec{p}_1(t) = \begin{pmatrix}0\\-4\end{pmatrix} \cdot t^3 + \begin{pmatrix}0\\6\end{pmatrix} \cdot t^2 + \begin{pmatrix}1\\0\end{pmatrix} \cdot t + \begin{pmatrix}0\\0\end{pmatrix}$$

$$\vec{p}_2(t) = \begin{pmatrix}0\\2\end{pmatrix} \cdot t^3 + \begin{pmatrix}0\\-3\end{pmatrix} \cdot t^2 + \begin{pmatrix}1\\0\end{pmatrix} \cdot t + \begin{pmatrix}1\\2\end{pmatrix}$$

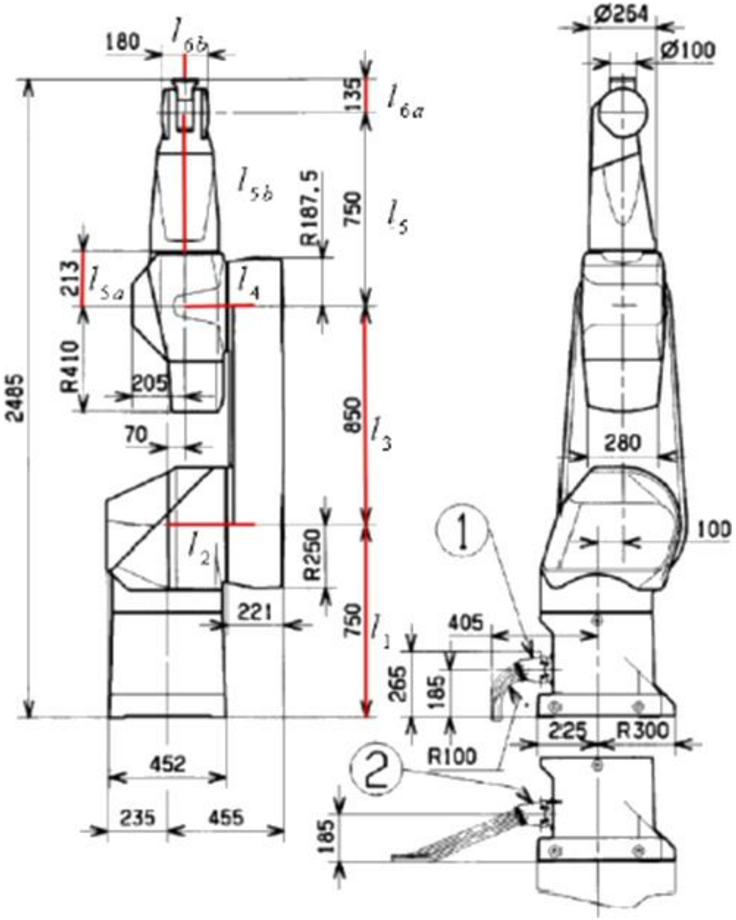
$$\vec{p}_3(t) = \begin{pmatrix}-2\\-6\end{pmatrix} \cdot t^3 + \begin{pmatrix}3\\9\end{pmatrix} \cdot t^2 + \begin{pmatrix}1\\0\end{pmatrix} \cdot t + \begin{pmatrix}2\\1\end{pmatrix}$$

CP: Piecewise Interpolation - An Example



Spline-Interpolation: Bernstein Polynomial

Determining an appropriate Path for Stäubli RX 170



Spline-Interpolation: Bernstein Polynomial

- The function is:

$$X(t) = \sum_{i=0}^{n} b_i B_i^n(t) \qquad (10 \cdot 8)$$

- The polynomial is of degree n, and has the location vectors $\vec{p}_i$ of the nodes p_i. The following applies

$$\mathrm{B_i^n(t)} = \binom{\mathrm{n}}{\mathrm{i}} \mathrm{(1-t)^{n-i} t^i} \qquad (10 \cdot 9)$$

- The Bernstein polynomials have the property of adding up to 1 for any t between [0,1]:

$$\sum_{i=0}^{n} B_i^n(t) = 1 \qquad (10 \cdot 10)$$

- Always symmetrical

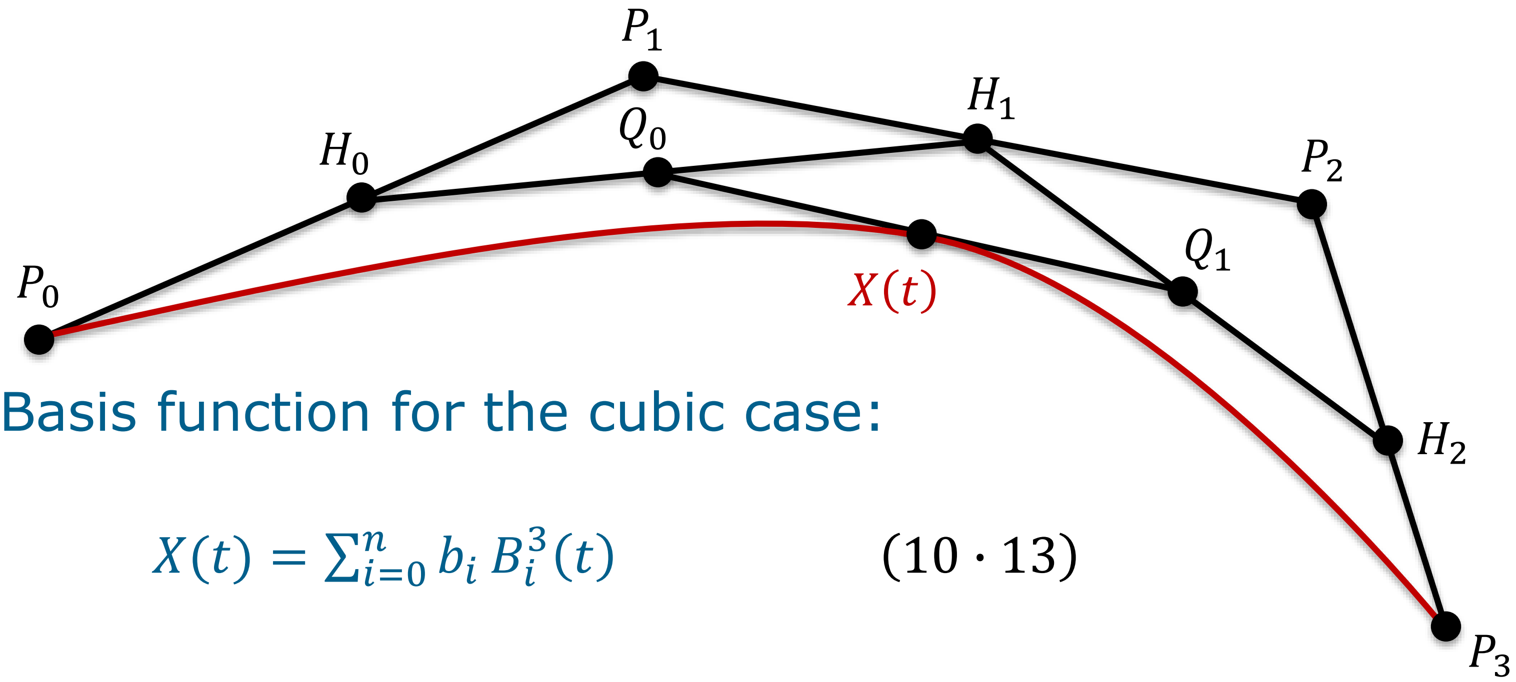
$$B_i^t(t) = B_{n-i}^n(1-t) \qquad (10 \cdot 11)$$

- And positive:

$$B_i^n(t) \geq 0 \qquad (10 \cdot 12)$$

Spline-Interpolation: Intermediate Step

- Approach:



- Basis function for the cubic case:

$$X(t) = \sum_{i=0}^{n} b_i \, B_i^3(t) \qquad (10 \cdot 13)$$

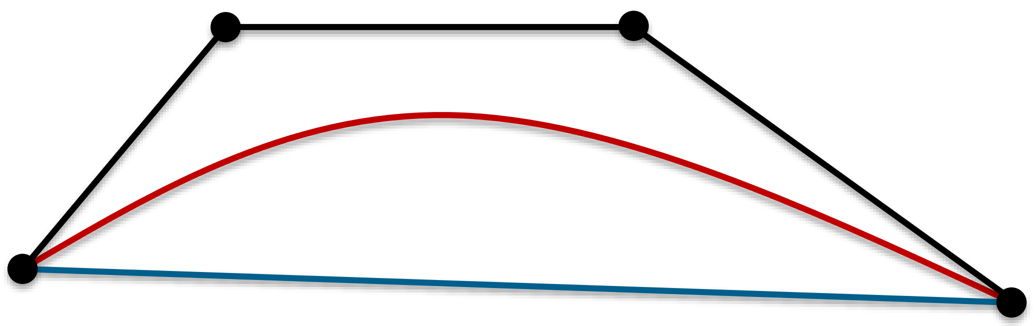
Spline-Interpolation: Intermediate Step

- Calculation of arbitrary intermediate steps
- Bernstein polynomial in the cubic case

$$B_i^n = \binom{3}{i} t(1-t)^{3-i} \qquad (10 \cdot 14)$$

$$\vec{x}(t) = P_0(1-t)^3 + P_1 \cdot 3(1-t)^2 t + P_2(1-t)t^2 + P_3 t^3 \qquad (10 \cdot 15)$$

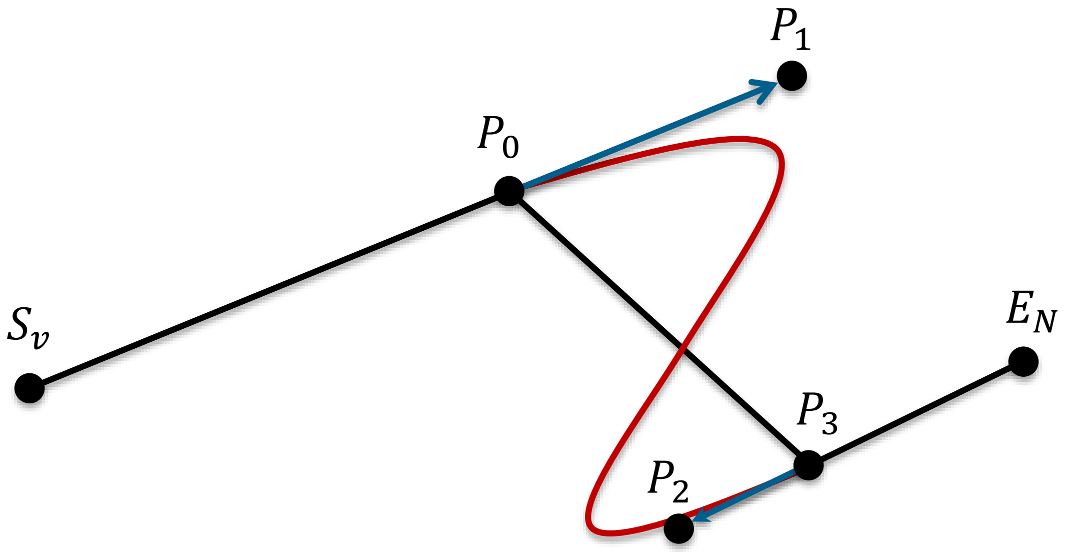
- Approximation for supporting points from below
- Not all forms are possible



Spline-Interpolation: Supporting Points

Calculation of supporting points in the 2D-case:

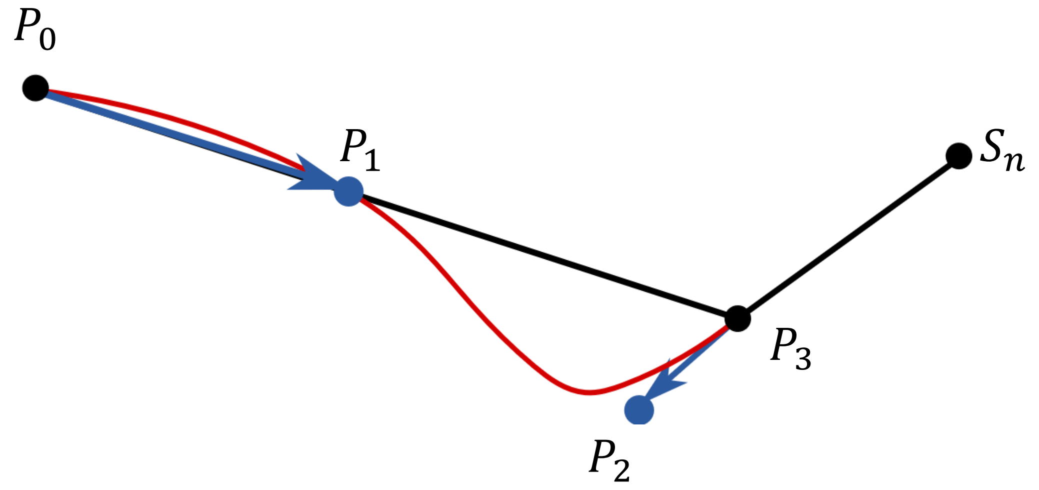
$$x(t) = P_{0x}(-t^3 + 3t^2 - 3t + 1) + P_{1x}(3t^3 - 6t^2 + 3t) + P_{2x}(-3t^3 + 3t^2) + P_{3x}t^3$$
$$y(t) = P_{0y}(-t^3 + 3t^2 - 3t + 1) + P_{1y}(3t^3 - 6t^2 + 3t) + P_{2y}(-3t^3 + 3t^2) + P_{3y}t^3$$



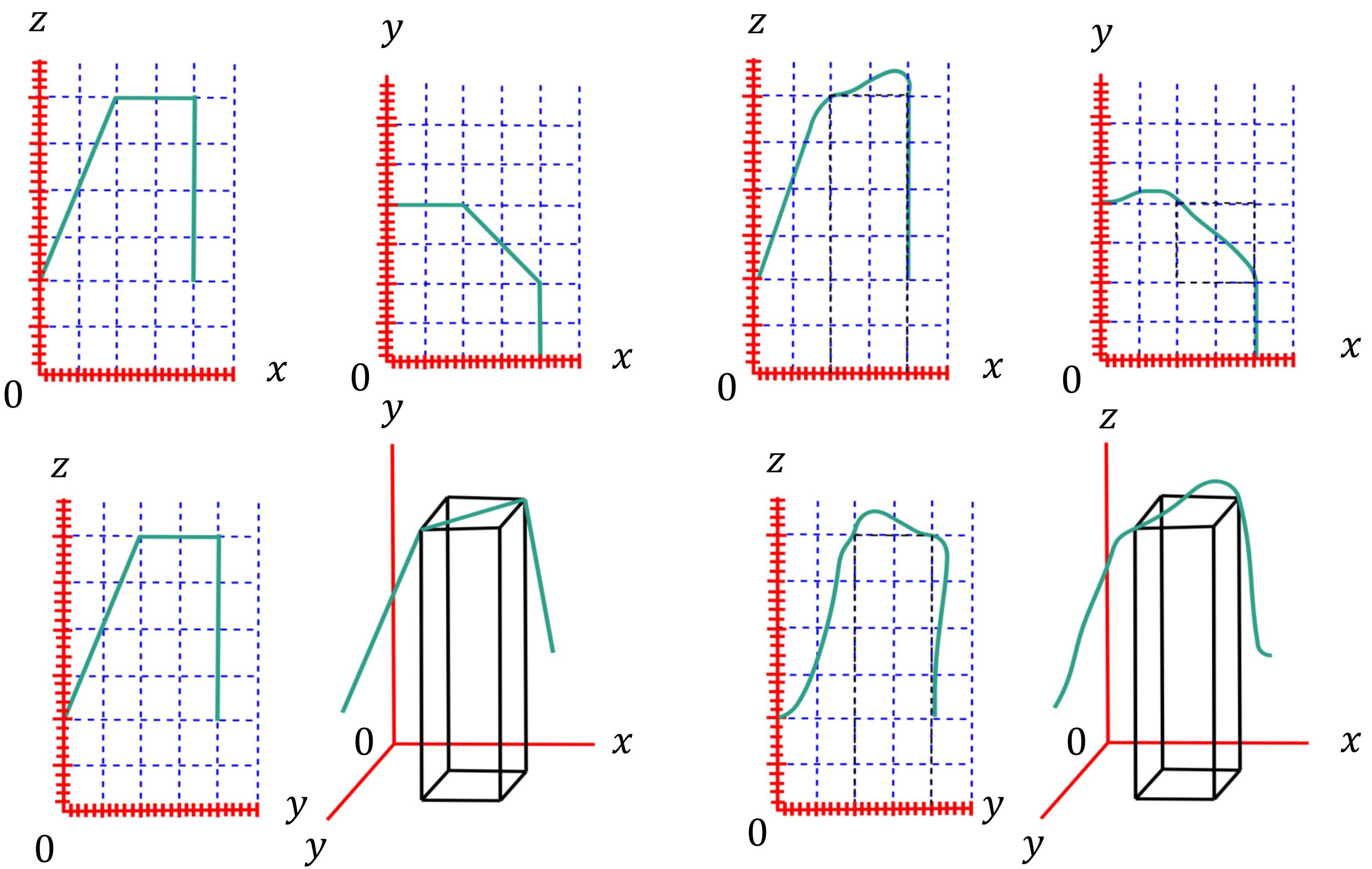
$$P_{1,x} = P_{0,x} + \tau\left(P_{0,x} - S_v\right)$$
$$P_{2,x} = P_{3,x} + \tau\left(P_{3,x} - E_n\right)$$
$$P_{1,y} = P_{0,y} + \tau\left(P_{3,y} - P_{0,y}\right)$$
$$P_{2,y} = P_{3,y} + \tau\left(P_{0,y} - P_{3,y}\right)$$

Spline-Interpolation: Supporting Points

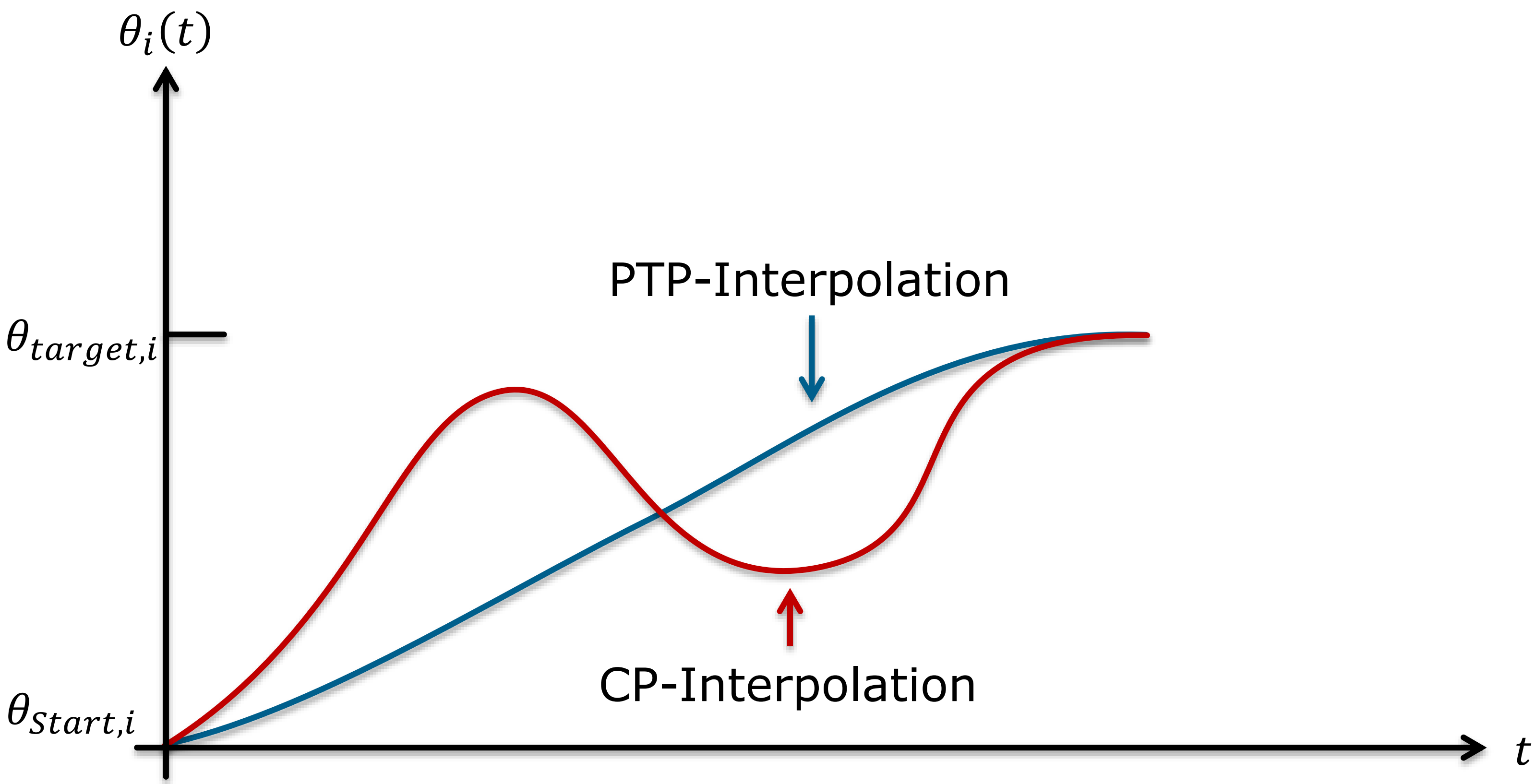
- The figure illustrates how the point P_1 is formed in the case of the first spline segment.



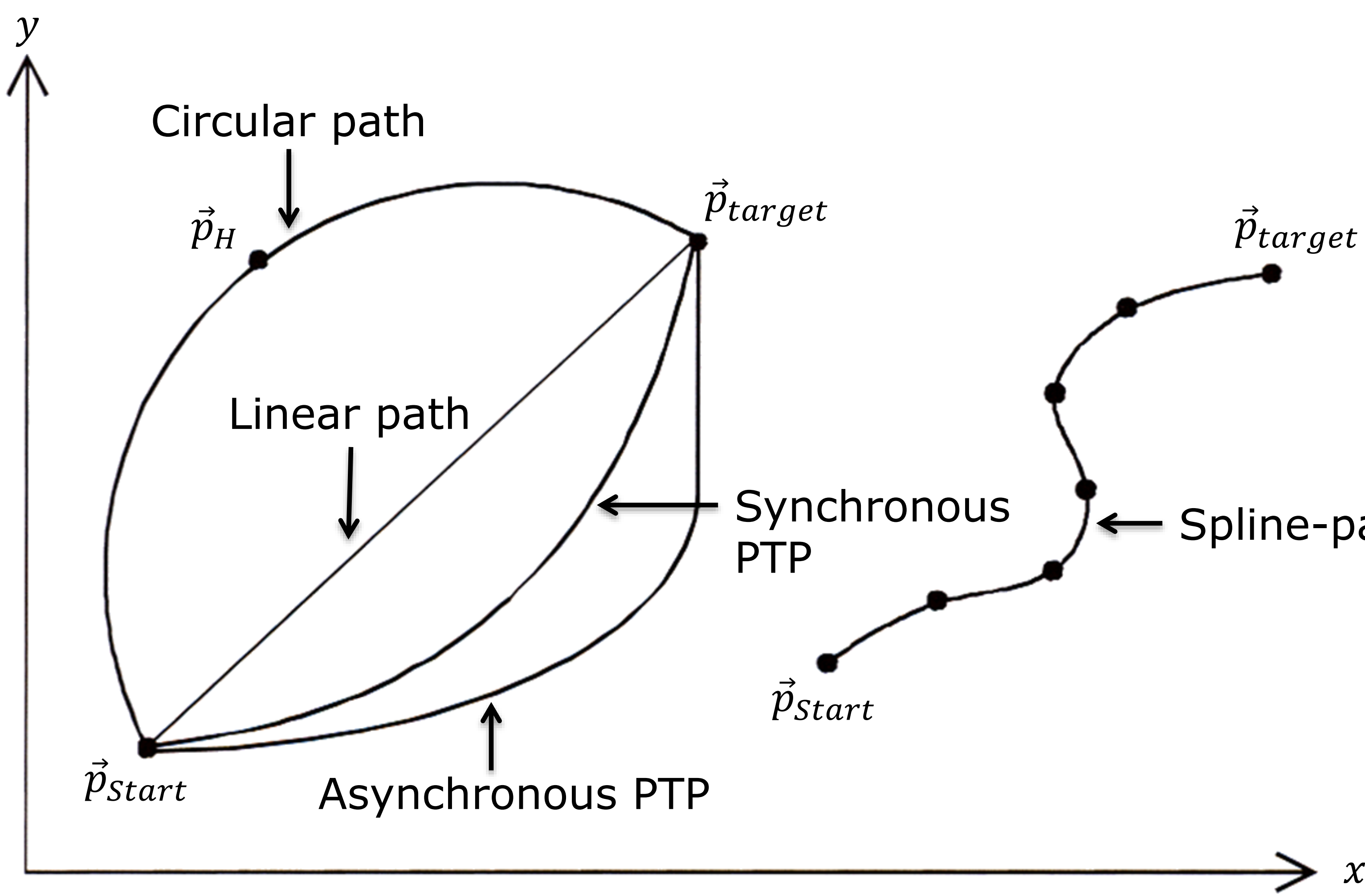
Example: Spline with 3 Segments $\tau=1\$$ and $\tau=1,3\$$



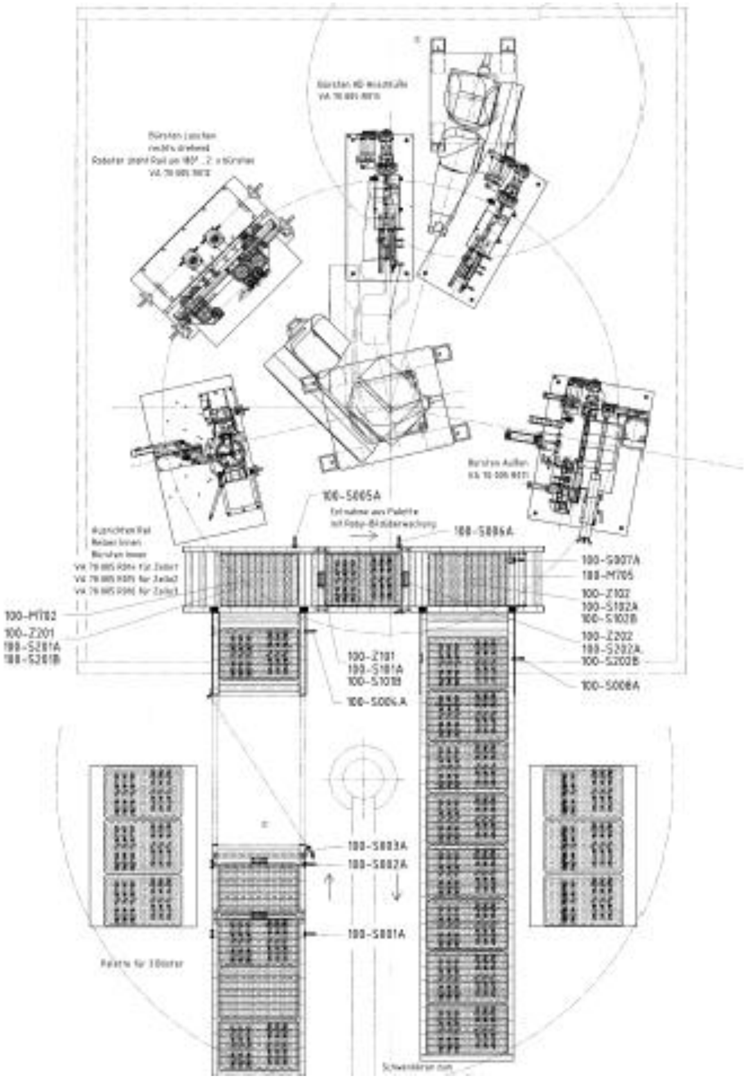
Comparison CP & PTP: Configuration Space



Comparison CP & PTP: Cartesian Space



Example: Bosch Homburg



Literature

- Weber, W. (2002),
  Industrieroboter – Methoden der Steuerung und Regelung,
  Fachbuchverlag Leipzig
- Stark, G. (2009),
  Robotik mit MATLAB,
  Fachbuchverlag Leipzig

Coming up Next...

End Effectors and Grip Planning