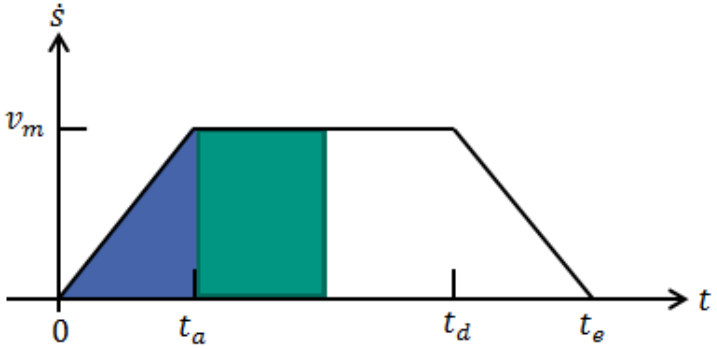
Karlsruher Institut für Technologie

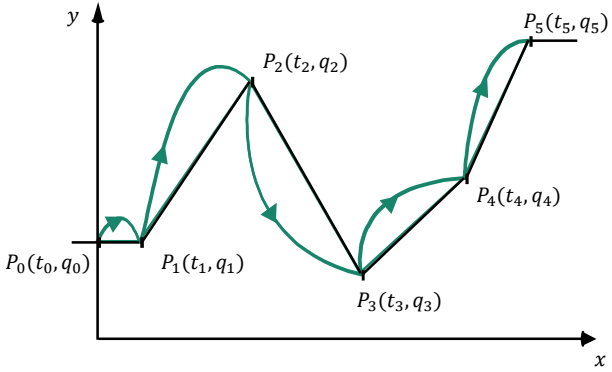
Hochperformante Humanoide Technologien

# Robotik I: Einführung in die Robotik
## Kapitel 6 – Bahnsteuerung

Tamim Asfour

http://www.humanoids.kit.edu



# Inhalt

- **Grundlagen der Bahnsteuerung**
- Programmierung der Schlüsselpunkte
- Interpolationsarten
- Approximierte Bahnsteuerung

# Grundlagen der Bahnsteuerung: Trajektorie

Bewegungen eines Roboters werden aufgefasst als

- **Zustandsänderungen**
  - Über die Zeit
  - Relativ zu einem festen Koordinatensystem (Arbeitsraum, Konfigurationsraum)

- mit **Einschränkungen** durch
  - Zwangsbedingungen
  - Gütekriterien
  - Neben- und Randbedingungen

# Grundlagen der Bahnsteuerung: Problem

- Gegeben
  - $S_{Start}$:
    Zustand zum **Startzeitpunkt**
  - $S_{Ziel}$:
    Zustand zum **Zielzeitpunkt**
- Gesucht
  - $S_i$:
    **Zwischenzustände** (Stützpunkte),
    damit die Trajektorie stetig wird.

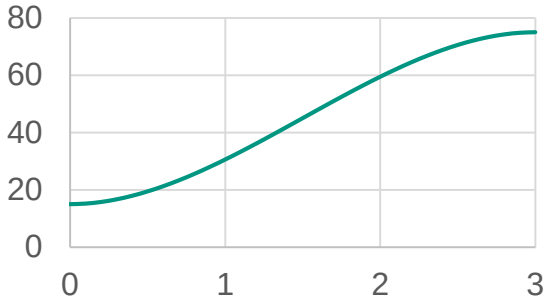
# Bahnsteuerung: Beispiel für ein Gelenk

- Anfangsbedingungen:

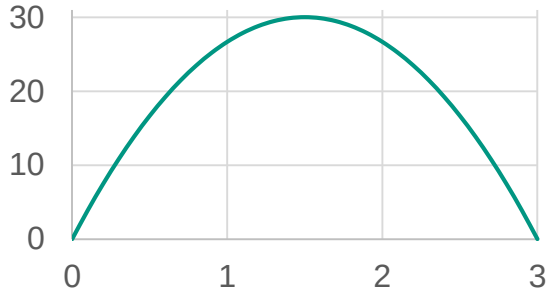
$$q(t_0) = 15°$$
$$\dot{q}(t_0) = 0 \frac{°}{sec}$$
$$\ddot{q}(t_0) = 40 \frac{°}{sec^2}$$

- Endbedingungen:

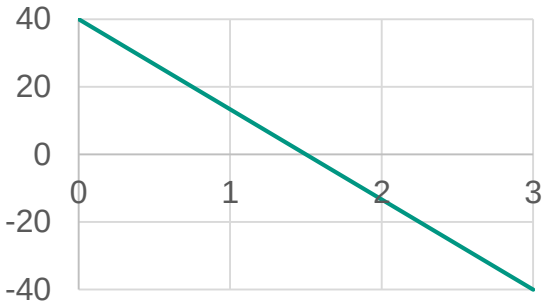
$$q(t_e) = 75°$$
$$\dot{q}(t_e) = 0 \frac{°}{sec}$$
$$\ddot{q}(t_e) = -40 \frac{°}{sec^2}$$




Position $q(t)$




Geschwindigkeit $\dot{q}(t)$




Beschleunigung $\ddot{q}(t)$

# Bahnsteuerung: Beispiel für ein Gelenk

- Anfangsbedingungen:

$$q(t_0) = 15°$$
$$\dot{q}(t_0) = 0 \frac{°}{sec}$$
$$\ddot{q}(t_0) = 40 \frac{°}{sec^2}$$

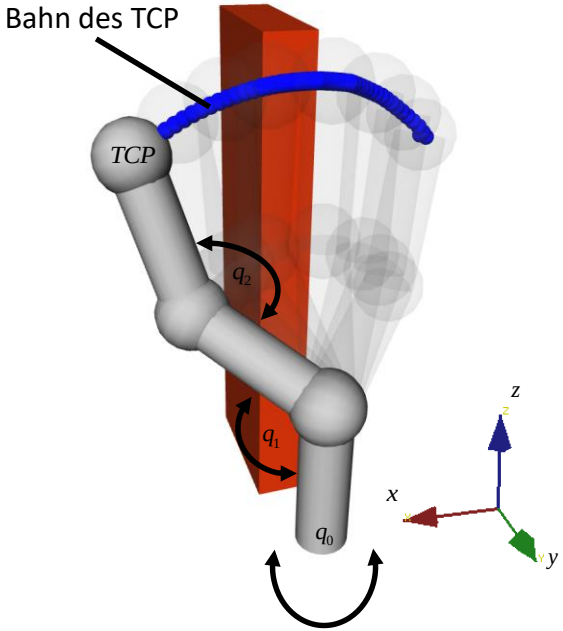
- Endbedingungen:

$$q(t_e) = 75°$$
$$\dot{q}(t_e) = 0 \frac{°}{sec}$$
$$\ddot{q}(t_e) = -40 \frac{°}{sec^2}$$

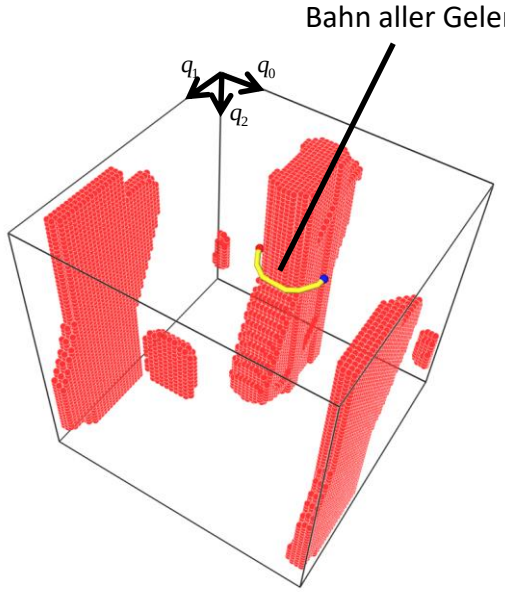
Mit den Bedingungen können wir ein Polynom dritten Grades bestimmen, dass die Anforderungen erfüllt

$$q(t) = -\frac{40}{9}t^3 + 20\,t^2 + 15 \qquad \dot{q}(t) = -\frac{40}{3}t^2 + 40\,t \qquad \ddot{q}(t) = -\frac{80}{3}t + 40$$

# Bahnsteuerung: Darstellung der Zustände (1)




Arbeitsraum




Konfigurationsraum

# Bahnsteuerung: Darstellung der Zustände (2)

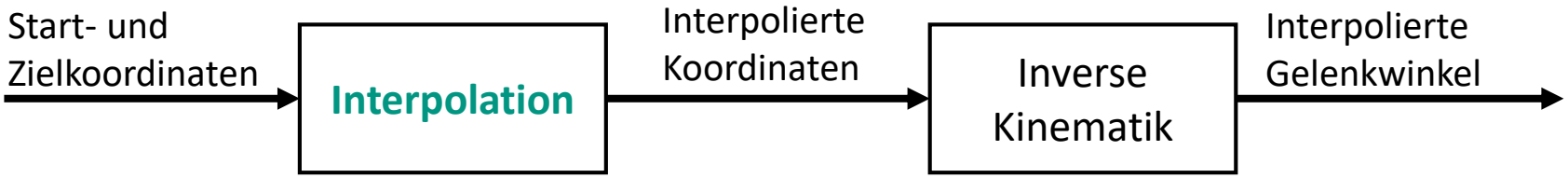
- **Zustände** können dargestellt werden im
  - Konfigurationsraum: $\mathbb{R}^n$
  - Arbeitsraum: $\mathbb{R}^3, SE(3)$

- Bahnsteuerung im **Konfigurationsraum** ist näher an der Ansteuerung der Teilsysteme des Roboter (Gelenke, Sensorik)

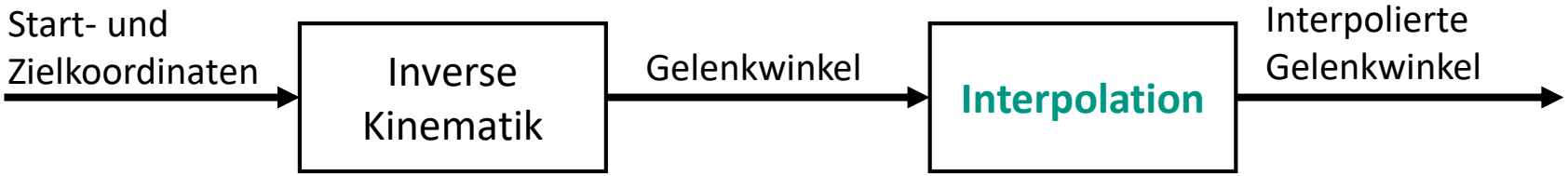
- Bahnsteuerung im **Arbeitsraum** ist näher an der zu lösenden Aufgabe
  - Bei Steuerung im Arbeitsraum ist das Lösen der **inversen Kinematik** nötig

# Bahnsteuerung: Interpolation

- Interpolation der **Weltkoordinaten**




- Interpolation der **Gelenkwinkel**



# Bahnsteuerung im Konfigurationsraum

- Bahnsteuerung als Funktion der **Gelenkwinkelzustände**
  - Verlauf der **punktweise** in Gelenkwinkeln spezifizierten Bahn muss im Arbeitsraum nicht definiert sein
- Abfahren dieser punktweise spezifizierten Trajektorien
  - **Asynchron:** Steuerung der Achsen unabhängig voneinander
    - Anwendung: Punktschweißen, Handhabungsaufgaben
  - **Synchron:** achsinterpolierte Steuerung
    - Bewegung aller Achsen beginnt und endet zum gleichen Zeitpunkt
    - Leitachse
    - Anwendung: Bahnschweißen, Lackieren, Montieren

# Bahnsteuerung im Arbeitsraum

- Angabe der Trajektorie erfolgt als Funktion der **Zustände des Roboters**
  - Beispiel: Beschreibungsvektor des Endeffektors
  - Position, Geschwindigkeit, Beschleunigung
- **Continuous Path** (CP):
  Endeffektor folgt in Position und Orientierung einer **definierten Bahn**
- **Bahntypen**
  - Lineare Bahnen
  - Polynombahnen
  - Splines

# Bahnsteuerung: Vor- und Nachteile der Darstellungen

| Arbeitsraum | Konfigurationsraum |
|---|---|
| + Bahn einfacher zu formulieren<br>+ Interpolation ist einfacher | + Ansteuerung der Gelenke ist einfacher<br>+ Trajektorie ist eindeutig und berücksichtigt die Gelenkwinkelgrenzen |
| – Inverse Kinematik für jeden Punkt der Trajektorie zu lösen<br>– Geplante Trajektorie nicht immer ausführbar | – Interpolation für mehrere Gelenke<br>– Formulierung der Trajektorie umständlicher |

# Inhalt

- Grundlagen der Bahnsteuerung
- **Programmierung der Schlüsselpunkte**
- Interpolationsarten
- Approximierte Bahnsteuerung

# Direkte Programmierung: Teach-In (1)

- **Anfahren** markanter **Punkte** der Bahn mit **manueller Steuerung**
  - Teach Box
  - Teach Panel
  - Spacemouse
  - Teach-Kugel

- **Funktionalität** einer **Teach Box:**
  - Einzelbewegung der Gelenke
  - Bewegung des Effektors in 6 Freiheitsgraden
  - Speichern / Löschen von Anfahrpunkten
  - Eingabe von Geschwindigkeiten
  - Eingabe von Befehlen zur Bedienung des Greifers
  - Starten / Stoppen ganzer Programme

# Direkte Programmierung: Teach-In (2)

- Vorgehen
  - **Anfahren** markanter **Schlüsselpunkte** der Bahn
  - **Speichern** der **Gelenkwerte**
  - **Ergänzung** der gespeicherten Werte um Parameter wie Geschwindigkeit, Beschleunigung usw.

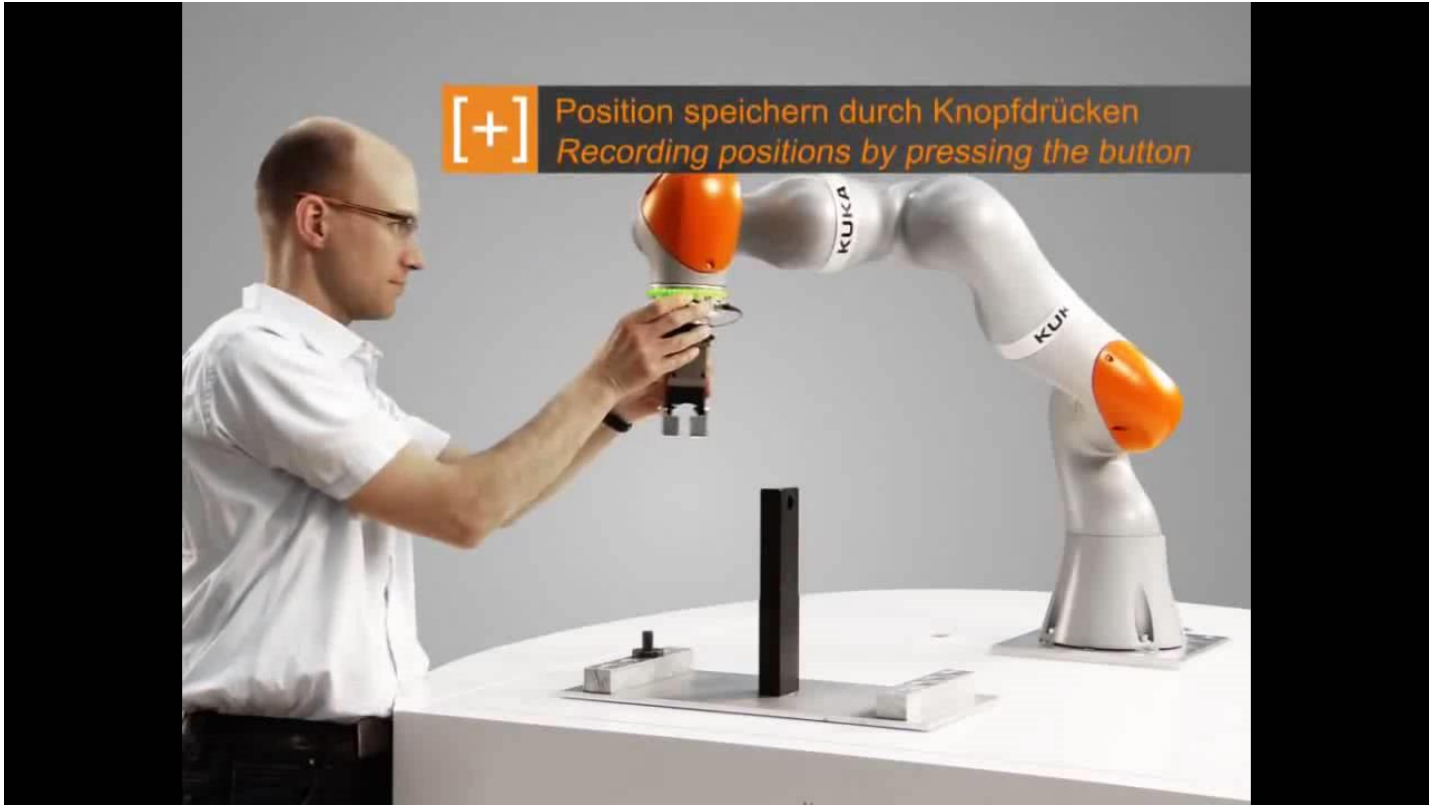
- Anwendung:
  - Fertigungsindustrie
    - Punktschweißen
    - Nieten
  - Handhabungsaufgaben
    - Pakete vom Fließband nehmen

# Direkte Programmierung: Playback (1)

- Roboter im **Zero-Force-Control** Modus
  - Roboter kann **durch den Bediener bewegt** werden
  - **Abfahren** der gewünschten Bahn
  - **Speichern** der **Gelenkwerte** (2 Möglichkeiten):
    - Automatisch (definierte Abtastfrequenz)
    - Manuell (durch Tastendruck)


- Anwendung:
  - Mathematisch schwer beschreibbare Bewegungsabläufe
  - Integration der handwerklichen Erfahrung
  - Typische Einsatzbereiche:
    - Lackieren
    - Kleben

# Direkte Programmierung: Playback (2)



# Direkte Programmierung: Playback (4)

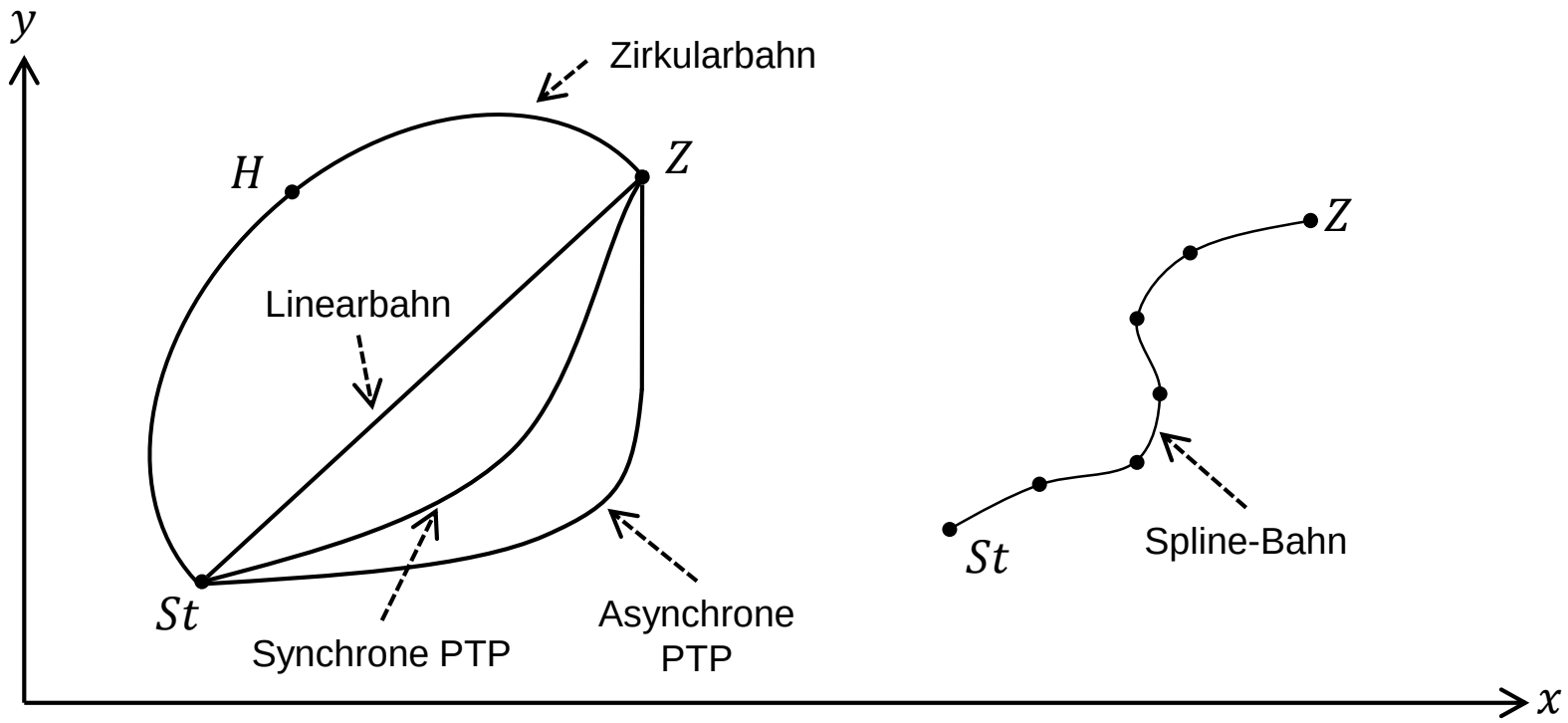
- Vorteile
  - **Schnell** für komplexe Bahnen
  - **Intuitiv**

- Nachteile
  - **Schwere Roboter** sind oft auch schwierig zu bewegen
  - Wenig **Platz** in engen Fertigungszellen für Bediener (Sicherheitsrisiko)
  - **Schlechte Korrekturmöglichkeiten**
  - **Optimierung** und Kontrolle durch Interpolationsmethoden **schwierig** (Suboptimale Bahnen)

# Inhalt

- Grundlagen der Bahnsteuerung
- Programmierung der Schlüsselpunkte
- **Interpolationsarten**
  - **Punkt-zu-Punkt (PTP)**
  - **Linear- und Zirkularinterpolation**
  - **Spline-Interpolation**
- Approximierte Bahnsteuerung

# Interpolationsarten: Überblick



# Punkt-zu-Punkt-Steuerung (PTP) (1)

- Roboter führt **Punkt-zu-Punkt-Bewegung** aus
  - PTP: Point-to-Point

- Vorteile:
  - Die Berechnung der Gelenkwinkeltrajektorie ist **einfach**
  - **Keine Probleme** mit **Singularitäten**

- Sequenz von **Gelenkwinkelvektoren**

$$\boldsymbol{q}(t_j) = \left(q_1(t_j), q_2(t_j), \dots, q_n(t_j)\right)^T$$

mit $q_i(t_j)$: Winkel des Gelenks $i$ zum Zeitpunkt $t_j$ mit $j = 0, \dots, k$

# Punkt-zu-Punkt-Steuerung (PTP) (2)

Randbedingungen

- **Start- und Zielzustand** sind bekannt

$$\boldsymbol{q}(t_0) = \boldsymbol{q}_{Start}$$
$$\boldsymbol{q}(t_e) = \boldsymbol{q}_{Ziel}$$

- Beispiel: Geschwindigkeiten zu Beginn und am Ende sind Null

$$\dot{\boldsymbol{q}}(t_0) = 0$$
$$\dot{\boldsymbol{q}}(t_e) = 0$$

- Der **Gelenkwinkelbereich** sowie **Geschwindigkeiten** und **Beschleunigungen** sind **begrenzt** (z.B. schnelles Beschleunigen, langsames Abbremsen)

$$\boldsymbol{q}_{min} < \boldsymbol{q}(t_j) < \boldsymbol{q}_{max}$$
$$|\dot{\boldsymbol{q}}(t_j)| < \dot{\boldsymbol{q}}_{max}$$
$$|\ddot{\boldsymbol{q}}(t_j)| < \ddot{\boldsymbol{q}}_{max}$$

# Punkt-zu-Punkt-Steuerung (PTP) (3)

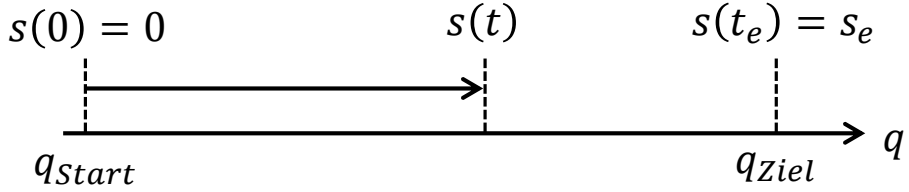
## Ablauf der Steuerung

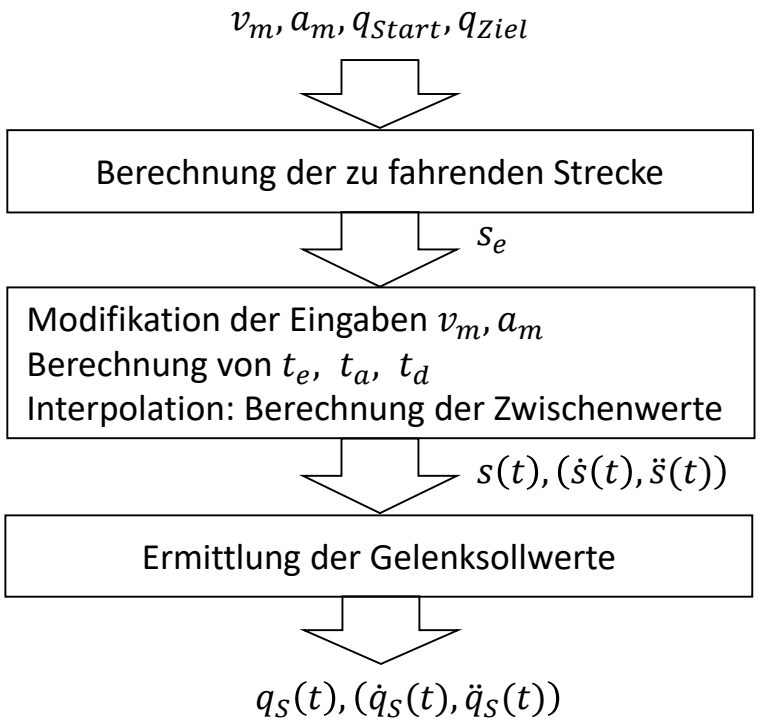
- Fahrzeit $t_e$
- Beschleunigungszeit $t_a$
- Beginn der Bremszeit $t_d$
- Maximale Geschwindigkeit $v_m$
- Maximale Beschleunigung $a_m$

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

$$s(t_e) = s_e = |q_{Ziel} - q_{Start}|$$

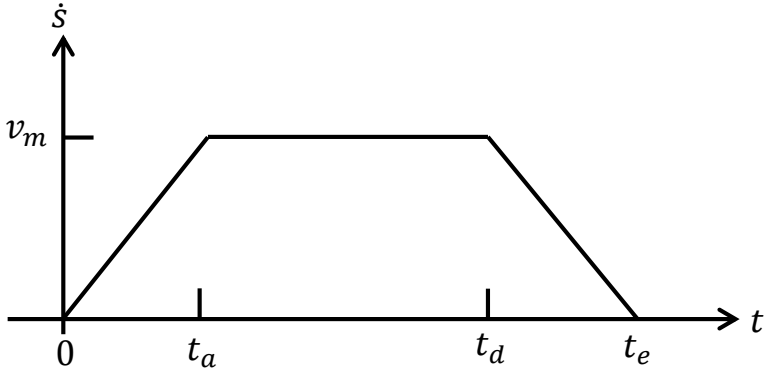
$$\dot{s}(t_e) = v(t_e) = 0$$



# Interpolation für PTP mit Rampenprofil (1)

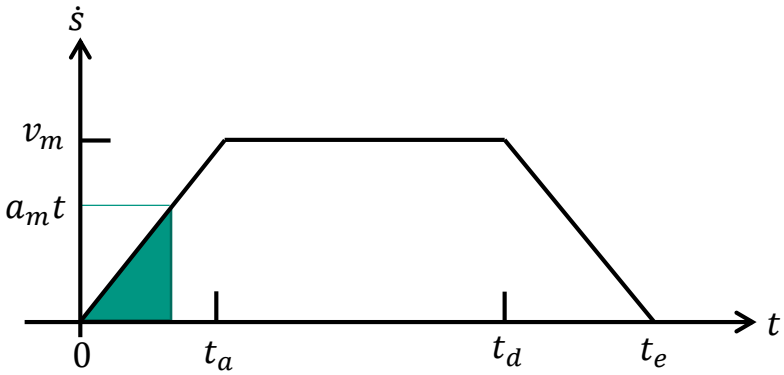
- Vorteil:
  **Einfache** Art zur **Berechnung** der Bahnparameter $s(t)$
- Nachteil:
  **Sprungförmige** Aufschaltung der **Beschleunigung** (ruckartig) kann zu Eigenschwingungen von mechanischen Teilen führen



# Interpolation für PTP mit Rampenprofil (2)

Phase I: **Beschleunigung**

$$0 \leq t \leq t_a$$




$$\ddot{s}(t) = a_m$$

$$\dot{s}(t) = a_m t + \dot{s}(0) \qquad mit\ \dot{s}(0) = 0$$

$$= a_m t$$

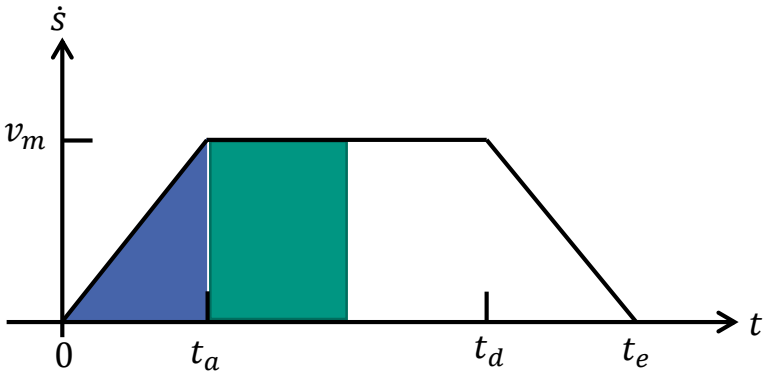
$$s(t) = \frac{1}{2} a_m t^2 + s(0) \qquad mit\ s(0) = 0$$

$$= \frac{1}{2} a_m t^2$$

# Interpolation für PTP mit Rampenprofil (3)

Phase II: **Gleichmäßigen Fahrt**

$$t_a \leq t \leq t_d$$




Aus Phase I wissen wir:

$$\dot{s}(t_a) = a_m t_a = v_m \rightarrow t_a = \frac{v_m}{a_m}$$

$$\boldsymbol{s(t_a) = \frac{1}{2} a_m {t_a}^2}$$

$$\ddot{s}(t) = 0$$

$$\dot{s}(t) = \dot{s}(t_a) = v_m$$

$$s(t) = \boldsymbol{v_m(t - t_a) + s(t_a)}$$

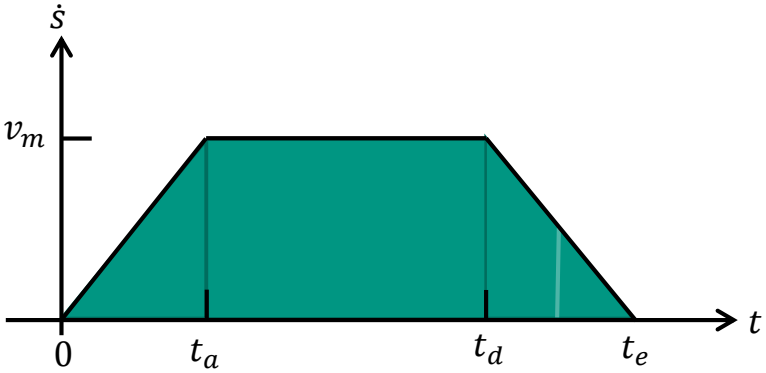
$$= v_m \left(t - \frac{v_m}{a_m}\right) + \frac{1}{2} a_m {t_a}^2$$

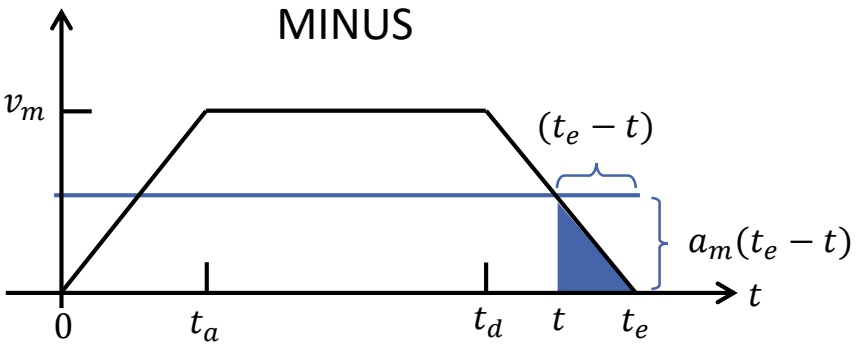
$$= v_m t - \frac{1}{2} \frac{v_m^2}{a_m}$$

# Interpolation für PTP mit Rampenprofil (4)

Phase III: **Bremsvorgang**

$$t_d \leq t \leq t_e \quad mit \quad t_d = t_e - t_a$$



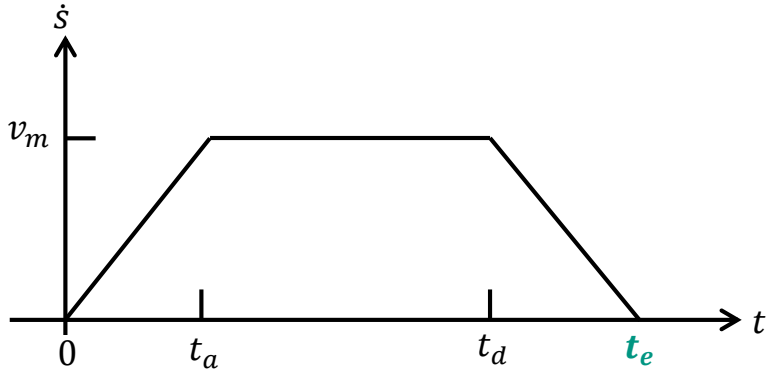

$$\ddot{s}(t) = -a_m$$

$$\dot{s}(t) = -a_m(t - t_d) + \dot{s}(t_d)$$
$$= -a_m(t - t_d) + v_m$$

$$s(t) = \boldsymbol{v_m(t_e - t_a) - \frac{a_m}{2}(t_e - t)^2}$$

# Interpolation für PTP mit Rampenprofil (5)

Berechnung der **Fahrtzeit**




Aus Phase III wissen wir:

$$s(t_e) = s_e = v_m(t_e - t_a)$$

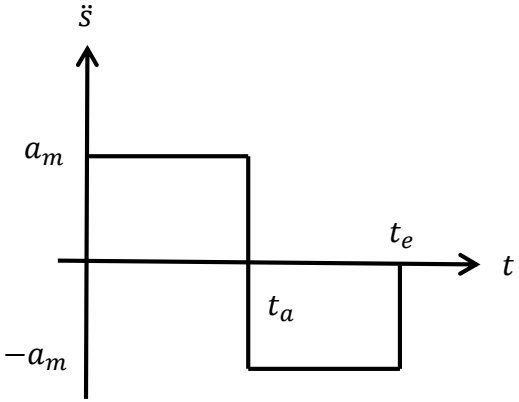
Nach $t_e$ auflösen, $t_a = \frac{v_m}{a_m}$

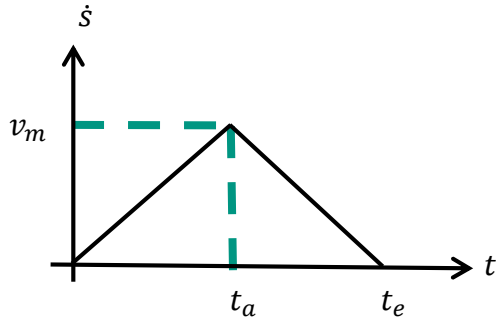
$$t_e = \frac{s_e}{v_m} + t_a = \frac{s_e}{v_m} + \frac{v_m}{a_m}$$

# Zeitoptimale Bahn

Falls $v_m$ zu groß in Bezug auf Beschleunigung und Bahnlänge:
Bestimmung einer zeitoptimalen Bahn nach

$$\boldsymbol{s_e = t_a \cdot v_m} = \frac{{v_m}^2}{a_m} \rightarrow v_m = \sqrt{a_m s_e}$$

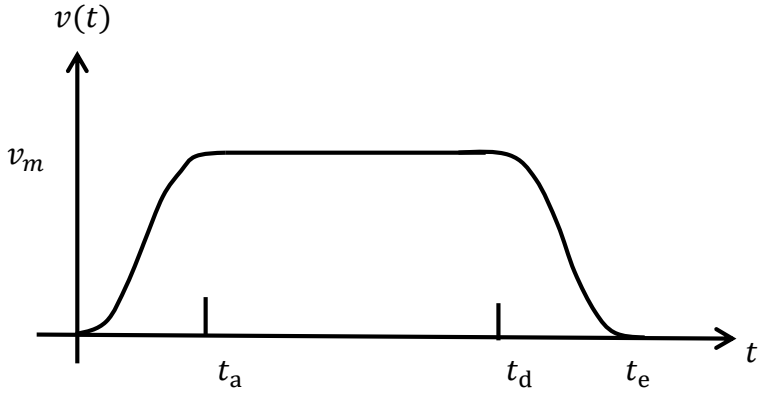


# Interpolation für PTP mit Sinoidenprofil (1)

- **Weichere Bewegung** durch Verwendung einer sinusförmigen Zeitfunktion
- Vorteil:
  - Roboter wird weniger beansprucht
- Nachteil:
  - Längere Beschleunigungs- und Bremsphase als beim Rampenprofil

- Bestimmung der Kurvenparameter für die drei Phasen:
  - Beschleunigung
  - Gleichförmige Bewegung
  - Bremsvorgang



# Interpolation für PTP mit Sinoidenprofil (2)

- Phase der **Beschleunigung**

$$\ddot{s}(t) = a_m sin^2\left(\frac{\pi}{t_a}t\right) \quad 0 \le t \le t_a$$

$$\dot{s}(t) = a_m\left(\frac{1}{2}t - \frac{t_a}{4\pi}\sin\left(\frac{2\pi}{t_a}t\right)\right)$$

$$s(t) = a_m\left(\frac{1}{4}t^2 + \frac{{t_a}^2}{8\pi^2}\left(\cos\left(\frac{2\pi}{t_a}t\right) - 1\right)\right)$$

- Aus $\dot{s}(t_a) = a_m\frac{1}{2}t_a = v_m$ folgt $t_a = \frac{2v_m}{a_m}$

- Phase der **gleichmäßigen Fahrt**

$$\ddot{s}(t) = 0 \quad t_a \le t \le t_d$$
$$\dot{s}(t) = v_m$$
$$s(t) = v_m(t - \frac{1}{2}t_a)$$

# Interpolation für PTP mit Sinoidenprofil (3)

- Phase des **Bremsvorganges**

$$\dot{s}(t) = v_m - \int_{t-t_d}^{t} a(\tau - t_d)d\tau = v_m - a_m\left(\frac{1}{2}(t - t_d) - \frac{t_a}{4\pi}\sin\left(\frac{2\pi}{t_a}(t - t_d)\right)\right) \quad t_d \le t \le t_e$$

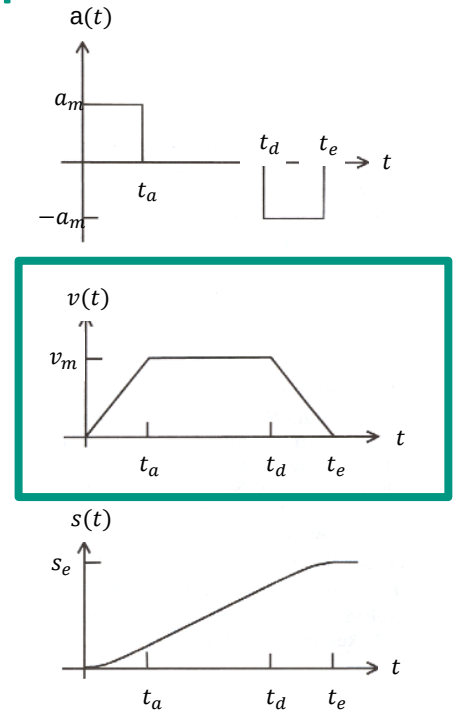
$$s(t) = s(t_d) + \int_{t-t_d}^{t} \dot{s}\,(\tau - t_d)d\tau = \frac{a_m}{2}\left(t_e(t + t_a) - \frac{t^2 + t_e^2 + 2\,t_a^2}{2} + \frac{t_a^2}{4\pi}\left(1 - \cos\left(\frac{2\pi}{t_a}(t - t_d)\right)\right)\right)$$

- Berechnung der **Fahrtzeit**

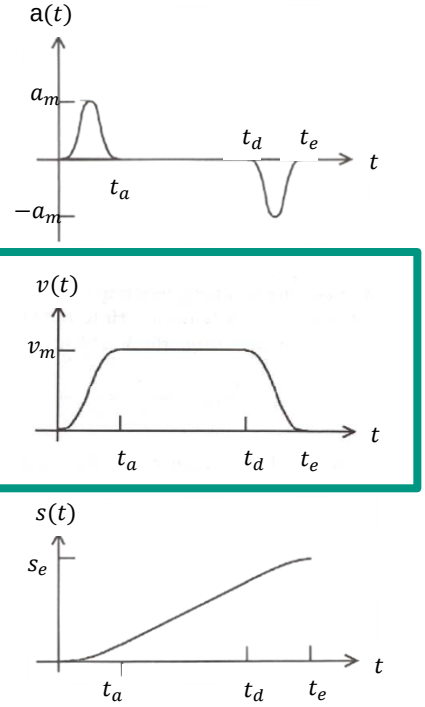
$$t_e = \frac{s_e}{v_m} + t_a = \frac{s_e}{v_m} + \frac{2v_m}{a_m}$$

# Interpolationsarten: Rampen- vs. Sinoidenprofil

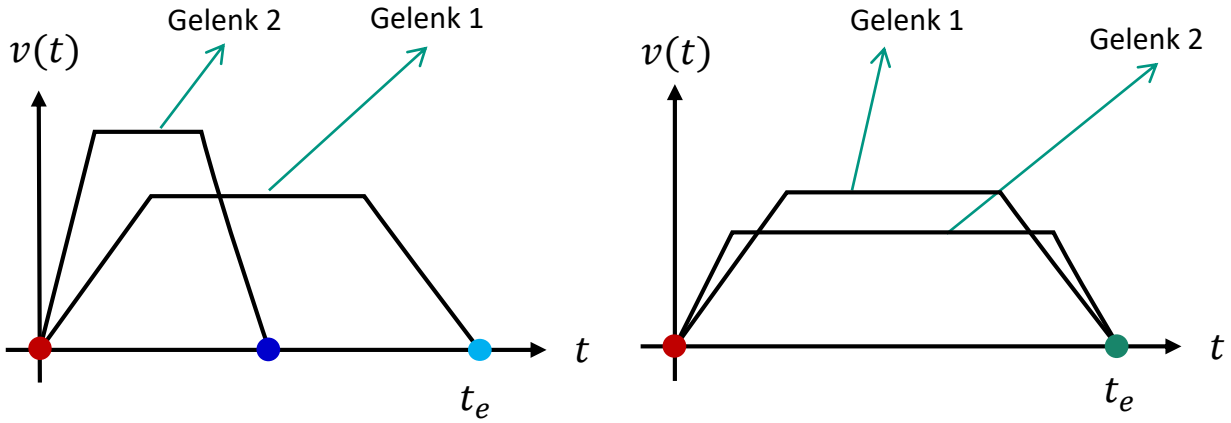
## Rampenprofil



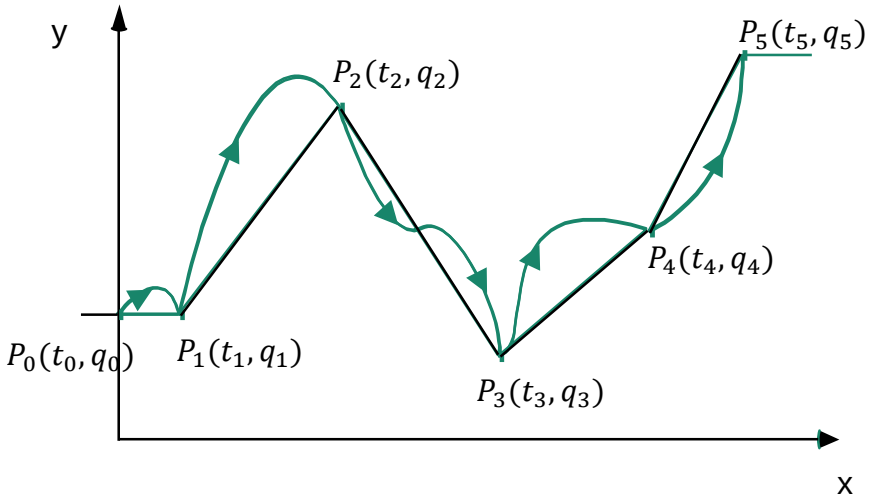

## Sinoidenprofil



# Asynchrone und synchrone PTP-Bahnen

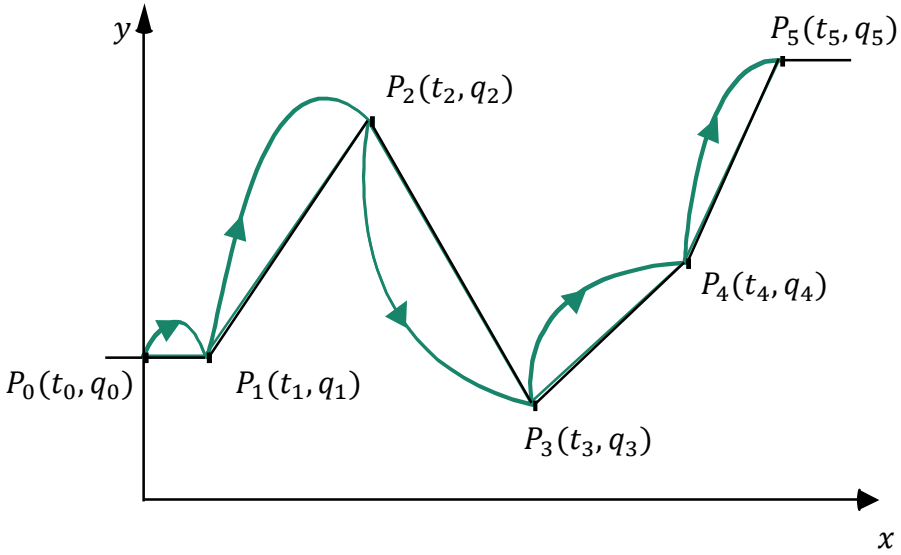


# Asynchrone PTP-Bahnen




- Jedes Gelenk wird **sofort** mit der **maximalen Beschleunigung** angesteuert.
- Jede Gelenkbewegung **endet unabhängig** von den anderen.

# Synchrone PTP-Bahnen




- Alle Gelenke **beginnen und beenden** ihre **Bewegungen zum gleichen Zeitpunkt** (synchron).

# Synchrone PTP-Bahnen: Vorgehen (1)

- Bestimme für **jedes Gelenk $i$** die **PTP-Parameter** (analog zur asynchronen PTP)
  - $s_{e,i}$
  - $v_{m,i}$
  - $a_{m,i}$
  - $t_{e,i}$ (Fahrzeit)
- Bestimme die **maximale Fahrzeit**
  - $t_e = t_{e,max} = max(t_{e,i})$
  - Achse mit max. Fahrzeit ist Leitachse
- Setze die **maximale Fahrzeit** als Fahrzeit **für alle Gelenke**
  - $t_{e,i} = t_e$

# Synchrone PTP-Bahnen: Vorgehen (2)

- Bestimme die **neuen maximalen Geschwindigkeiten** für **alle Gelenke**
  - **Umformung Fahrzeit** und Berechnung der neuen Geschwindigkeiten

    - **Rampenprofil:**

$$t_e = \frac{s_{e,i}}{v_{m,i}} + \frac{v_{m,i}}{a_{m,i}} \rightarrow v_{m,i}^2 = v_{m,i} a_{m,i} t_e - s_{e,i} a_{m,i}$$

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

    - Analoge Berechnung für **Sinoidenbahn:**

$$v_{m,i} = \frac{a_{m,i} t_e}{4} - \sqrt{\frac{a_{m,i}^2 t_e^2 - 8 s_{e,i} a_{m,i}}{16}}$$

# Vollsynchrone PTP-Bahnen

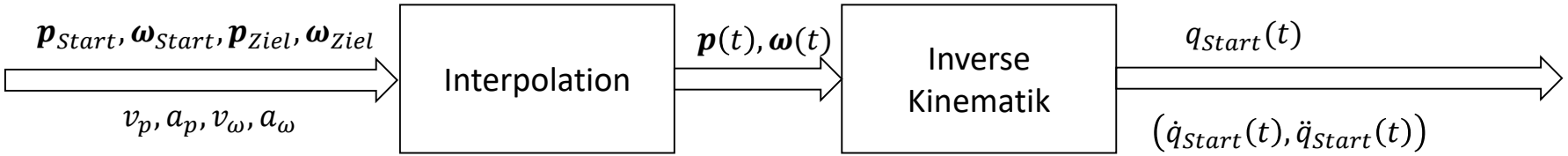
- Zusätzliche Berücksichtigung der **Beschleunigungs- und Bremszeit**
- **Bessere Annäherung** der Start- und Zielpunkte im Arbeitsraum
- Bestimmung der Leitachse mit $t_e$ und $t_a \rightarrow t_d = t_e - t_a$
- Bestimmung der maximalen Geschwindigkeit und Beschleunigung der anderen Achsen mit

$$v_{m,i} = \frac{s_{e,i}}{t_d} \qquad a_{m,i} = \frac{v_{m,i}}{t_a}$$

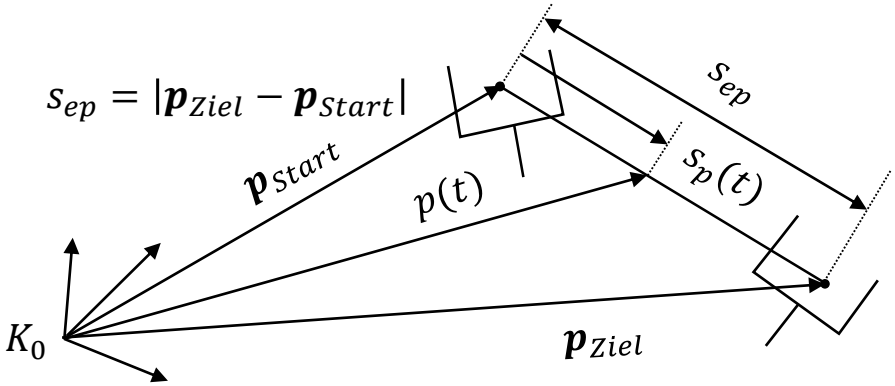
- Nachteil: Beschleunigung jeder Achse wird vorgegeben

# Steuerung im Arbeitsraum

- Continuous Path (CP)
  - Endeffektor folgt in Position und Orientierung einer **definierten** Bahn
- **Pose** des Endeffektors im **Arbeitsraum**
  - $\boldsymbol{p} = (x, y, z)^T \in \mathbb{R}^3$: **Position**
  - $\boldsymbol{\omega} = (\alpha, \beta, \gamma)^T \in \mathbb{R}^3$: **Orientierung** (z. B. als Eulerwinkel)
- Maximale Geschwindigkeiten und Beschleunigungen im Arbeitsraum
  - $v_p \in \mathbb{R}$: Lineare Geschwindigkeit
  - $a_p \in \mathbb{R}$: Lineare Beschleunigung
  - $v_\omega \in \mathbb{R}$: Winkelgeschwindigkeit
  - $a_\omega \in \mathbb{R}$: Winkelbeschleunigung



# Linearinterpolation (1)




$$p(t) = \boldsymbol{p}_{Start} + \frac{s_p(t)}{s_{ep}} \cdot (\boldsymbol{p}_{Ziel} - \boldsymbol{p}_{Start})$$

Berechnung von $s_p(t)$ mit Rampen- oder Sinoidenprofil

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

$$v_m = v_p, a_m = a_p, t_e = t_{ep}, t_a = t_{ap}, t_d = t_{dp}, s_e = s_{ep}, s = s_p$$

# Linearinterpolation (2)

- Orientierung in Eulerwinkel: $\boldsymbol{\omega} = (\alpha, \beta, \gamma)^T$

$$s_{e\omega} = |\boldsymbol{\omega}_{Ziel} - \boldsymbol{\omega}_{Start}|$$
$$= \sqrt{(\alpha_{Ziel} - \alpha_{Start})^2 + (\beta_{Ziel} - \beta_{Start})^2 + (\gamma_{Ziel} - \gamma_{Start})^2}$$

- Berechnung von $s_\omega(t)$ mit Rampen- oder Sinoidenprofil:

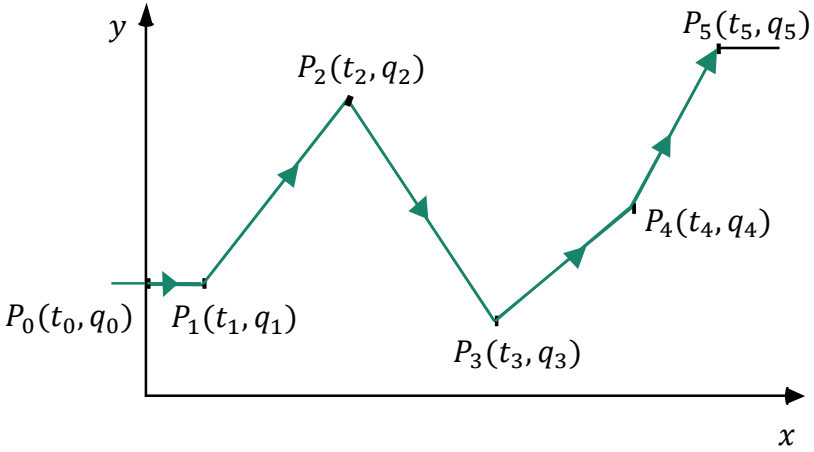
$$v_m = v_\omega, \quad a_m = a_\omega, \quad t_e = t_{e\omega}, \quad t_a = t_{a\omega}, \quad t_d = t_{d\omega},$$
$$s_e = s_{e\omega}, \quad s = s_\omega$$

- Angleichen der Fahrzeiten $t_{ep}$ (Position) und $t_{e\omega}$ (Orientierung)

$$t_e = \max(t_{ep}, t_{e\omega})$$

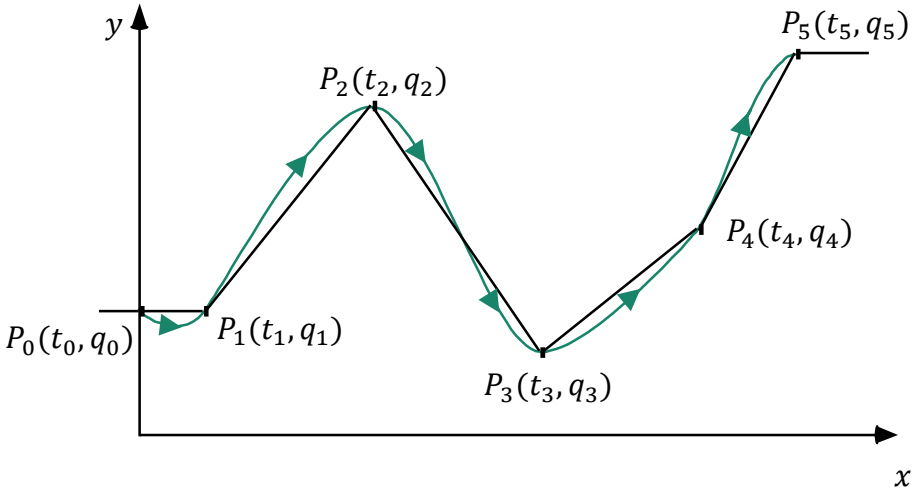
- Analog zur Anpassung der Geschwindigkeiten bei synchronen PTP
  - Falls $t_e = t_{ep}$: $\quad v_\omega = \frac{a_\omega t_e}{2} - \sqrt{\frac{a_\omega^2 t_e^2}{4} - s_{e\omega} a_\omega}$
  - Falls $t_e = t_{e\omega}$: $\quad v_p = \frac{a_p t_e}{2} - \sqrt{\frac{a_p^2 t_e^2}{4} - s_{ep} a_p}$

# Linearinterpolation: Beispiel




- Die Robotersteuerung interpoliert die Bahn zwischen je 2 Teiltrajektorien.

# Segmentweise Bahninterpolation




- Die Endbedingungen der Teiltrajektorie $j-1$ (Richtung, Geschwindigkeit, Beschleunigung) und die Anfangsbedingungen der Teiltrajektorie $j$ werden aneinander angeglichen
- Teiltrajektorien werden separat beschrieben (Beispiel: Splines)

# Interpolation mit Kubischen Splines (1)

- Polynom

$$f(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \qquad (a_0, a_1, a_2, a_3 \in \mathbb{R})$$

- Gegeben
  - Anfangspunkt $f(0) = s_a$
  - Endpunkt $f(t_e) = s_e$
  - Anfangsgeschwindigkeit $\dot{f}(0) = v_a$
  - Endgeschwindigkeit $\dot{f}(t_e) = v_e$
- Gesucht: $a_0, a_1, a_2, a_3 \in \mathbb{R}$
- Ziel: Bestimmen der Parameter für das Polynom

# Kubische Splines: Bestimmung der Parameter (1)

- $f(0) = s_a$
- $\dot{f}(0) = v_a$
- $\dot{f}(t_e) = v_e$

# Kubische Splines: Bestimmung der Parameter (1)

- $f(0) = s_a$

$$f(t=0) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 = a_0$$
$$\Rightarrow a_0 = s_a$$

- $\dot{f}(0) = v_a$

$$\dot{f}(t=0) = a_1 + 2a_2 t + 3a_3 t^2 = a_1$$
$$\Rightarrow a_1 = v_a$$

- $\dot{f}(t_e) = v_e$

$$a_1 + 2a_2 t_e + 3a_3 t_e^2 = v_e$$
$$v_a + 2a_2 t_e + 3a_3 t_e^2 = v_e$$
$$2a_2 t_e = v_e - v_a - 3a_3 t_e^2$$
$$a_2 = \frac{v_e - v_a}{2t_e} - \frac{3}{2} a_3 t_e$$

# Kubische Splines: Bestimmung der Parameter (2)

- $a_0 = s_a$
- $a_1 = v_a$
- $a_2 = \frac{v_e - v_a}{2t_e} - \frac{3}{2} a_3 t_e$
- $f(t_e) = s_e$

$$a_0 + a_1 t_e + a_2 t_e^2 + a_3 t_e^3 = s_e$$

$$s_a + v_a t_e + \left( \frac{v_e - v_a}{2t_e} - \frac{3}{2} a_3 t_e \right) t_e^2 + a_3 t_e^3 = s_e$$

$$2v_a t_e + (v_e - v_a) t_e - 3a_3 t_e^3 + 2a_3 t_e^3 = 2(s_e - s_a)$$

$$(v_e + v_a) t_e - a_3 t_e^3 = 2(s_e - s_a)$$

$$-a_3 t_e^3 = -(v_e + v_a) t_e$$

$$\Rightarrow a_3 = \frac{(v_e + v_a)}{t_e^2} - \frac{2(s_e - s_a)}{t_e^3}$$

# Kubische Splines: Bestimmung der Parameter (3)

- $a_0 = s_a$
- $a_1 = v_a$
- $a_2 = \frac{v_e - v_a}{2t_e} - \frac{3}{2} a_3 t_e$
- $a_3 = \frac{(v_e + v_a)}{t_e^2} - \frac{2(s_e - s_a)}{t_e^3}$

$$a_2 = \frac{v_e - v_a}{2t_e} - \frac{3}{2} a_3 t_e$$

$$a_2 = \frac{v_e - v_a}{2t_e} - \frac{3}{2}\left(\frac{(v_e + v_a)}{t_e^2} - \frac{2(s_e - s_a)}{t_e^3}\right) t_e$$

$$a_2 = \frac{1}{2t_e}(v_e - v_a - 3v_e - 3v_a) + \frac{3(s_e - s_a)}{t_e^2}$$

$$\Rightarrow\ a_2 = \frac{3(s_e - s_a)}{t_e^2} - \frac{v_e + 2v_a}{t_e}$$

# Kubische Splines: Bestimmung der Parameter (4)

- Kubisches Polynom

$$f(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

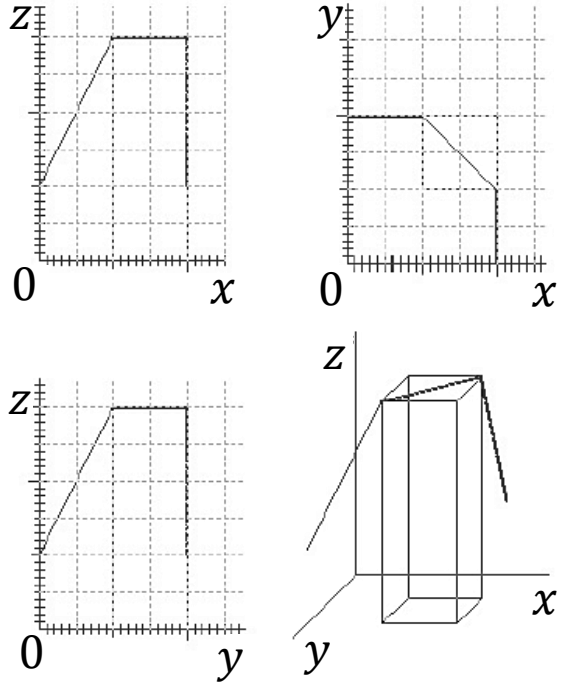
- Gewünschte Eigenschaften:
  - Anfangspunkt $f(0) = s_a$
  - Endpunkt $f(t_e) = s_e$
  - Anfangsgeschwindigkeit $\dot{f}(0) = v_a$
  - Endgeschwindigkeit $\dot{f}(t_e) = v_e$

- Lösung:

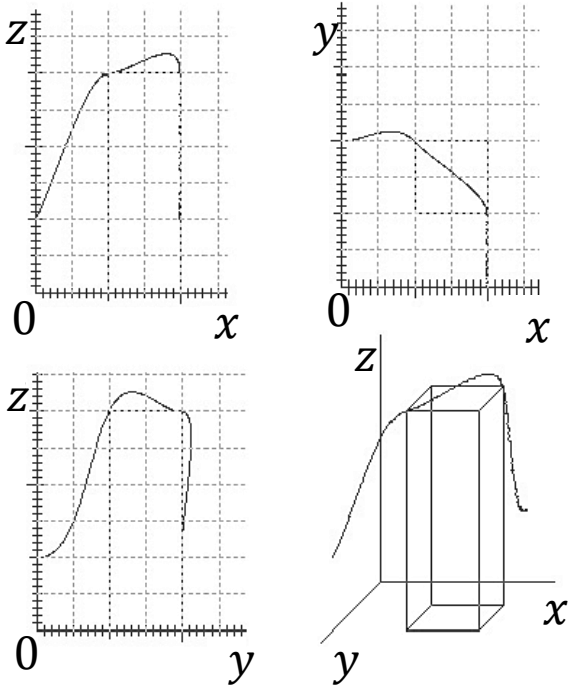
$$f(t) = s_a + v_a t + \left(\frac{3(s_e - s_a)}{t_e^2} - \frac{v_e + 2v_a}{t_e}\right) t^2 + \left(\frac{(v_e + v_a)}{t_e^2} - \frac{2(s_e - s_a)}{t_e^3}\right) \mathrm{t}^3$$

# Beispiele für Spline-Interpolation

- Bahn (4 Stützpunkte)




- Spline-Interpolation

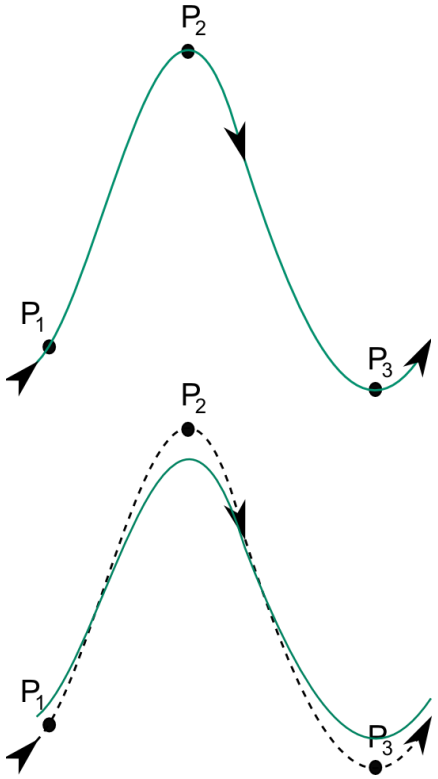


# Inhalt

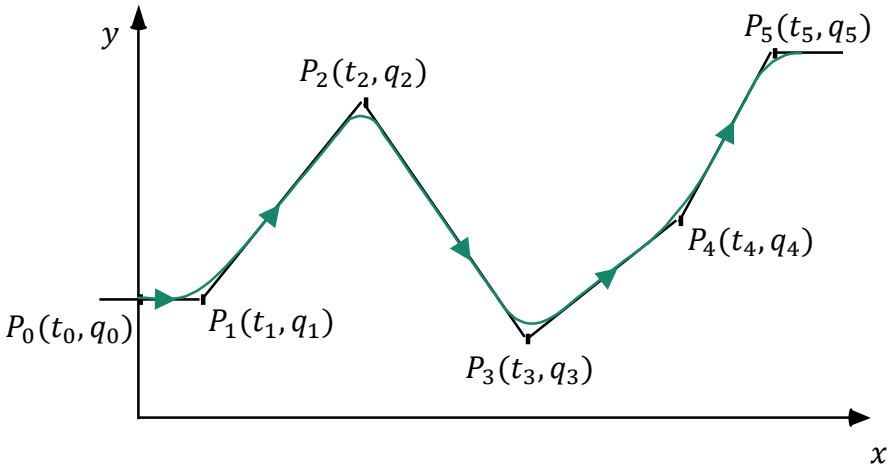
- Grundlagen der Bahnsteuerung
- Programmierung der Schlüsselpunkte
- Interpolationsarten
- **Approximierte Bahnsteuerung**
  - **Bernsteinpolynome**

# Approximierte Bahnsteuerung: Definition

- Bahninterpolation
  - Die ausgeführte Bahn verläuft **durch alle Stützpunkte** der Trajektorie
- Bahnapproximation
  - Die Kontrollpunkte beeinflussen den Bahnverlauf und werden **approximiert**



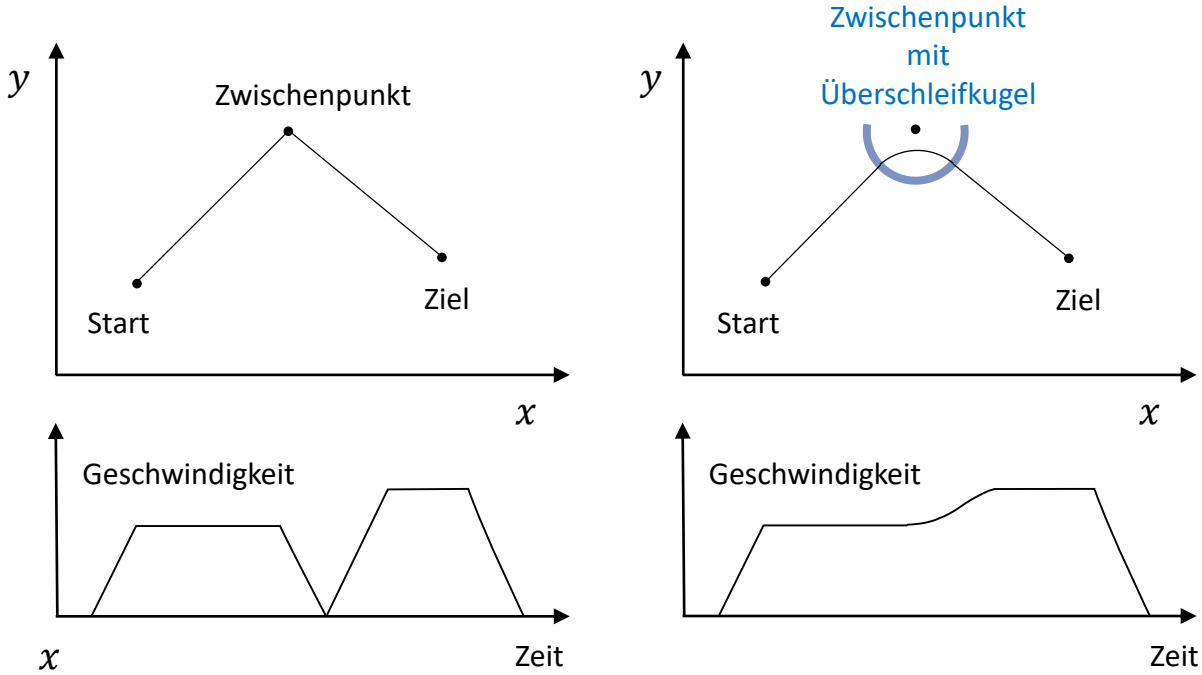
# PTP und CP mit Überschleifen (1)




- Zum Zeitpunkt $t_j - \varepsilon$ beginnen, die Parameter (Richtung und Geschwindigkeit) der Teiltrajektorie $j - 1$ auf die Parameter der Teiltrajektorie $j$ zu überführen.

- In der Regel wird der **Stützpunkt $i$ nicht erreicht**.

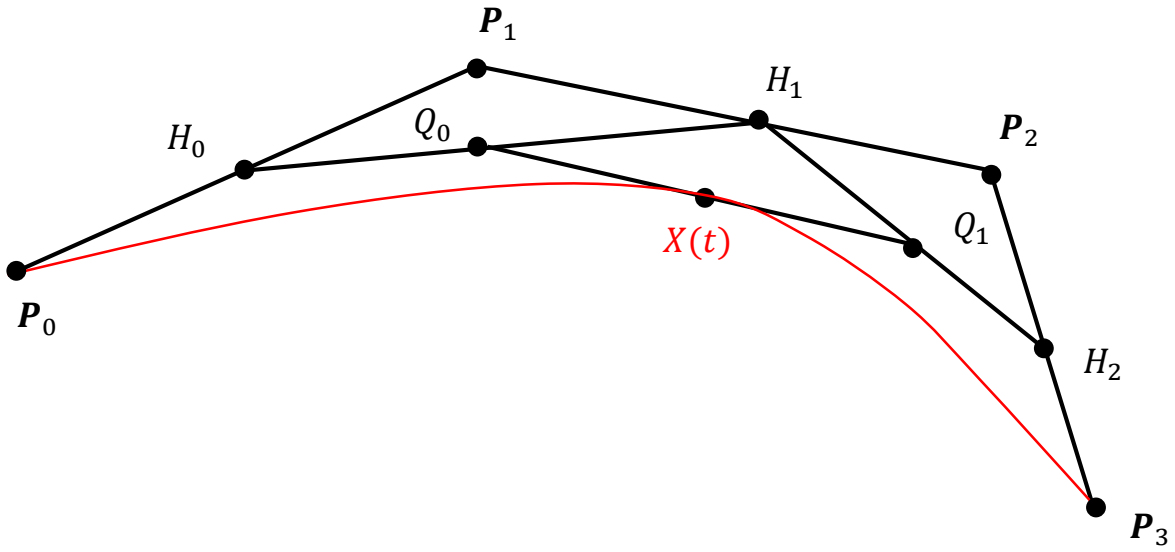
# PTP und CP mit Überschleifen (2)



# PTP und CP mit Überschleifen (3)

- **Geschwindigkeitsüberschleifen**
  - Beginn, wenn die Geschwindigkeit einen festgelegten Minimalwert unterschreitet
  - **Nachteil**: Abhängig vom Geschwindigkeitsprofil
- **Positionsüberschleifen**
  - Beginn, wenn der Endeffektor in die Überschleifkugel eintritt
  - Außerhalb der Überschleifkugel wird die Bahn exakt eingehalten
  - **Vorteil**: Gut kontrollierbar

# Approximation mit Bernsteinpolynomen



# Bézierkurven (1)

- Im Unterschied zu kubischen Splines verlaufen **Bézierkurven nicht durch alle Stützpunkte $P_i$**, sondern werden nur von ihnen beeinflusst.

- Basisfunktion:

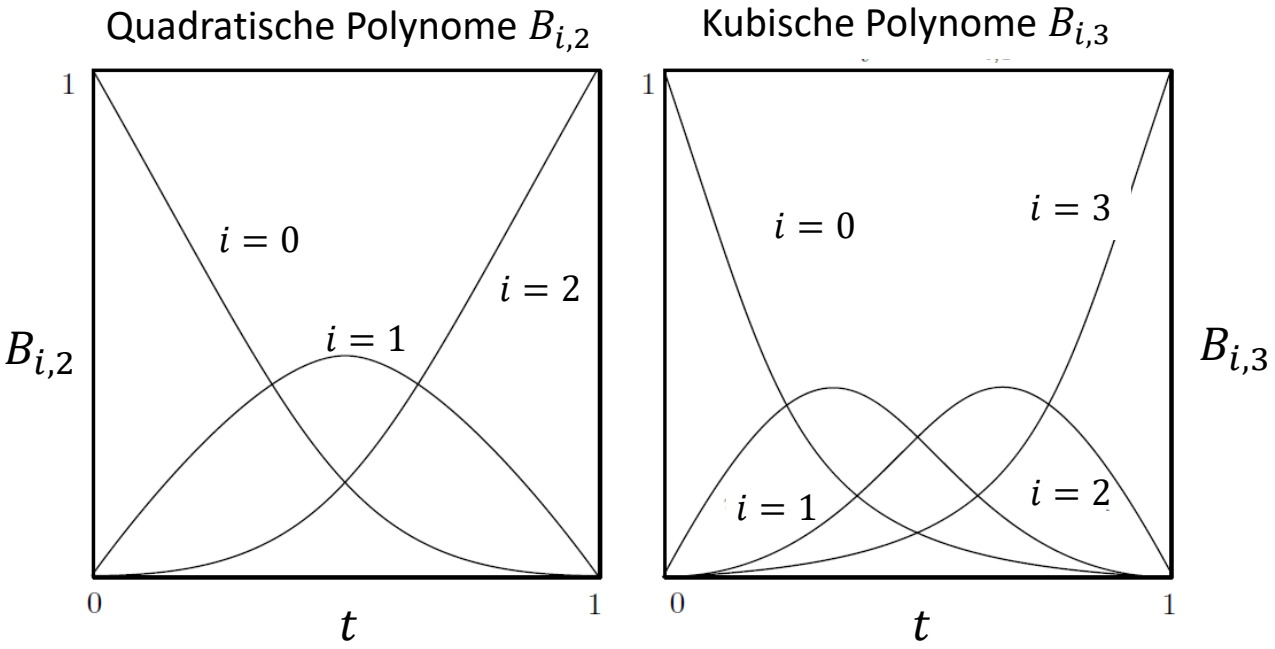
$$P(t) = \sum_{i=0}^{n} B_{i,n}(t)\, \boldsymbol{P}_i \quad 0 \le t \le 1$$

- $B_{i,n}(t)$: $i$–tes **Bernsteinpolynom** vom Grad $n$

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

# Beispiele für Bernsteinpolynome

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$



# Bézierkurven (2)

- Berechnung beliebiger Zwischenstellungen

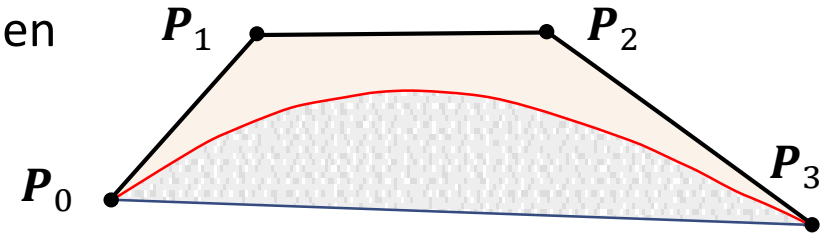
- Beispiel: Bernsteinpolynom für kubischen Fall (Grad $n = 3$)

$$B_{i,3}(t) = \binom{3}{i} t^i (1-t)^{3-i}$$

$$P(t) = (1-t)^3 \boldsymbol{P}_0 + 3(1-t)^2 t \boldsymbol{P}_1 + 3(1-t) t^2 \boldsymbol{P}_2 + t^3 \boldsymbol{P}_3$$

- Annähern von unten an Stützstellen

- Keine beliebige Form



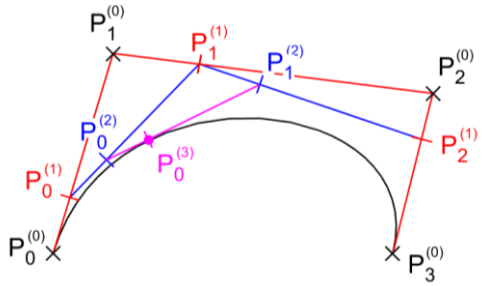
# De-Casteljau-Algorithmus (1)

- **Annäherung** an die **Bézierkurve:**
- Effiziente Berechnung einer Näherungsdarstellung von Bézierkurven durch einen Polygonzug
- **Idee:** Algorithmus basiert darauf, dass eine **Bézierkurve geteilt** und durch **zwei aufeinanderfolgende** Bézierkurven **dargestellt** wird
- **Iterative Berechnung**: Kann auch für große $n$ effizient berechnet werden

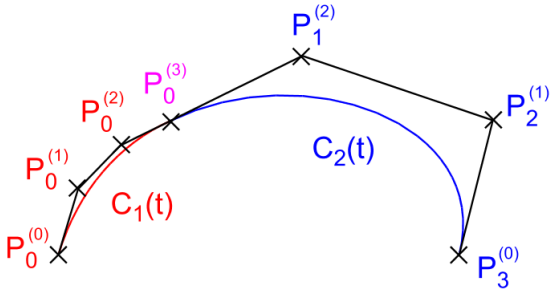
- Gegeben: $n$ Kontrollpunkte $\boldsymbol{P}_0, \ldots, \boldsymbol{P}_{n-1}$
- Start: $\boldsymbol{P}_i^0 = \boldsymbol{P}_i$
- Iteration k: $\boldsymbol{P}_i^{k+1} = (1 - t_0)\boldsymbol{P}_i^k + t_0\boldsymbol{P}_{i+1}^k$

# De-Casteljau-Algorithmus (2)

- Beispiel für $\boldsymbol{P}_0$ mit $k = 3$ und $t_0 = 0{,}25$:

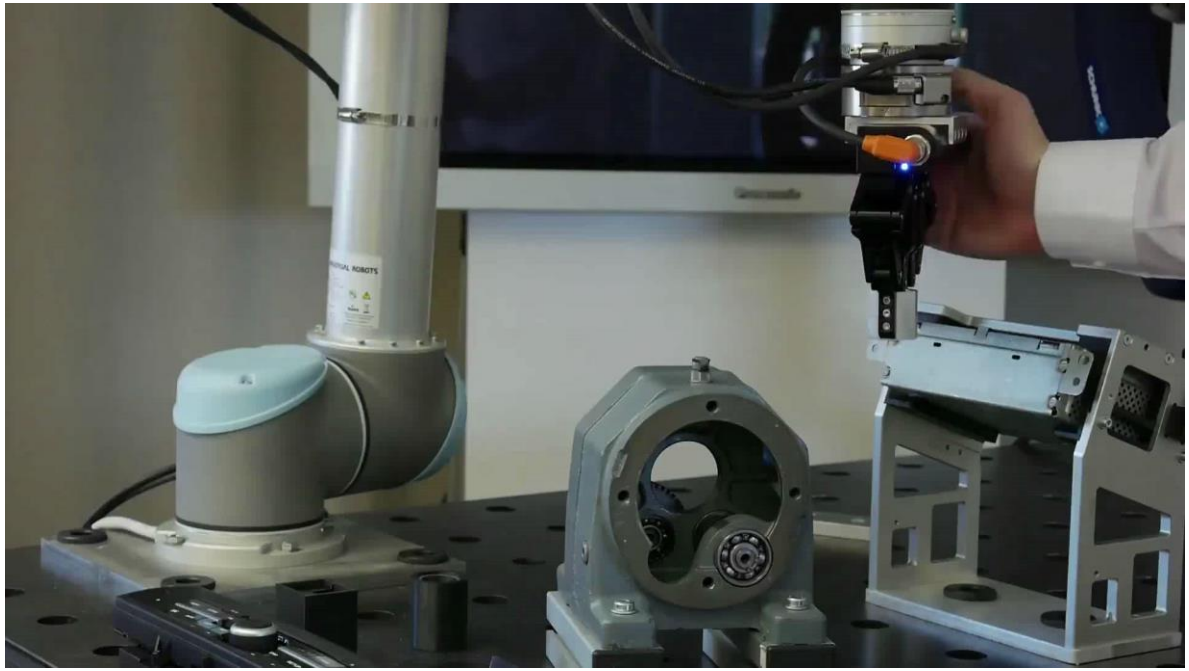



- **Zwei Bézierkurven** $C_1(t)$ und $C_2(t)$
- Approximation der Bézierkurve durch Polygonzug



# Ende!

# Direkte Programmierung: Playback (3)



http://www.artiminds.com/