

Zusammenfassung - Robotik

Julian Shen

4. Juni 2023

1 Mathematische Grundlagen

Kinematik ist die reine geometrische Beschreibung von Bewegung eines Manipulators oder Roboters. Das essentielle Konzept ist die **Position**.

Statik behandelt Kräfte und Momente, die sich auf einen ruhenden Mechanismus auswirken. Das essentielle Konzept ist die **Steifigkeit**.

Dynamik analysiert die Kräfte und Momente, die durch Bewegung und Beschleunigung eines Mechanismus und einer zusätzlichen Last entstehen.

Freiheitsgrade (DoF) ist die Anzahl unabhängiger Parameter, die zur kompletten Spezifikation der Lage eines Objekts benötigt werden, z.B. Starrkörper hat in 2D 3 DoF und in 3D 6 DoF.

Starrkörperbewegungen werden durch zwei Eigenschaften charakterisiert:

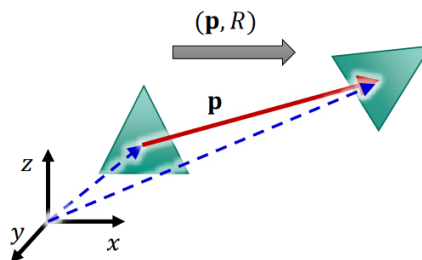
1. Distanz zweier beliebiger Punkte ist konstant
2. Orientierungen im Körper bleiben erhalten

SO(3) und SE(3):

- SO(3): **Spezielle Orthogonale Gruppe**, die **Rotationen** repräsentiert
- SE(3): **Spezielle Euklidische Gruppe**, die **Transformationen** repräsentiert
- Elemente aus SO(3) werden als reale 3×3 orthogonale Matrizen R (Zeilen- und Spaltenvektoren orthonormal) beschrieben und erfüllen

$$R^T R = 1 \quad \text{mit} \quad \det(R) = 1$$

- Elemente aus SE(3) sind von der Form (\mathbf{p}, R) mit $\mathbf{p} \in \mathbb{R}^3$ und $R \in SO(3)$ und beschreiben Verknüpfungen von Rotationen und Translationen



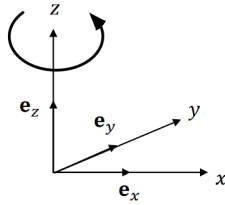
Euklidischer Raum: Vektorraum \mathbb{R}^3 mit dem Skalarprodukt.

- Punkt \mathbf{a} im euklidischen Raum wird durch Vielfache der Einheitsvektoren $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ beschrieben
- Wir benutzen **rechtsdrehende Koordinatensysteme**

**Rechtsdrehendes
Koordinatensystem**

$$\mathbf{e}_x \times \mathbf{e}_y = \mathbf{e}_z$$

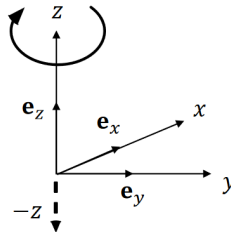
$$\mathbf{x} \times \mathbf{y} = \mathbf{z}$$



**Linksdrehendes
Koordinatensystem**

$$\mathbf{e}_x \times \mathbf{e}_y = -\mathbf{e}_z$$

$$\mathbf{x} \times \mathbf{y} = -\mathbf{z}$$



\times : Kreuzprodukt

Lineare Abbildungen (Transformationen), die den euklidischen Raum auf sich selbst abbilden, nennt man **Endomorphismen**:

$$\phi(\cdot): \mathbb{R}^3 \rightarrow \mathbb{R}^3$$

- Endomorphismen können durch quadratische Matrizen repräsentiert werden:

$$\phi(\mathbf{a}) = A \cdot \mathbf{a}, \quad A \in \mathbb{R}^{3 \times 3}$$

- A beschreibt einen Basiswechsel zwischen den originalen Basisvektoren $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ und den neuen Basisvektoren $\mathbf{e}'_x, \mathbf{e}'_y, \mathbf{e}'_z$:

$$A = (\mathbf{e}'_x \ \mathbf{e}'_y \ \mathbf{e}'_z) \cdot (\mathbf{e}_x \ \mathbf{e}_y \ \mathbf{e}_z)^{-1}$$

Bijektive Endomorphismen nennt man **Isomorphismen**.

- Eigenschaften:
 1. Winkel bleiben erhalten
 2. Längen bleiben erhalten
 3. Händigkeit bleibt erhalten
- Eine spezielle Art von Isomorphismen ist die **Rotationsgruppe** $SO(3)$

Rotationsgruppe $SO(3)$:

- $SO(3)$ ist nicht kommutativ: $A \cdot B \cdot \mathbf{x} \neq B \cdot A \cdot \mathbf{x}$ mit $\mathbf{x} \in \mathbb{R}^3$ und $A, B \in SO(3)$
- Für alle $R \in SO(3)$ ist $R^{-1} = R^\top$, die Inverse kann also leicht berechnet werden

Rotationen in 2D:

- Rotation in der xy -Ebene um $(0,0)$ ist eine **lineare Transformation**
- **Rotationsmatrix:** $R_\alpha(\mathbf{x}) = \begin{pmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{pmatrix} \cdot \mathbf{x}$ mit $RR^\top = R^\top R = I$ und $\det(R) = 1$
- Rotation um einen Punkt $\mathbf{c} \neq (0,0)$ ist keine lineare Transformation. Verschiebe dafür die Ebene um $-\mathbf{c}$, rotiere und verschiebe wieder um $+\mathbf{c}$ zurück:

$$R_{\mathbf{c},\alpha} = R_\alpha(\mathbf{x} - \mathbf{c}) + \mathbf{c} = R_\alpha(\mathbf{x}) + (-R_\alpha(\mathbf{c}) + \mathbf{c})$$

- $R_{\mathbf{c},\alpha}$ ist eine nichtlineare Transformation und heißt **affine Transformation**. Sie unterscheidet sich von R_α nur durch das Addieren einer Konstante

Rotationen in 3D:

- Eine 2D Rotation in der xy -Ebene ist eine 3D Rotation um die z -Achse:

$$R_{\mathbf{z},\alpha} = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

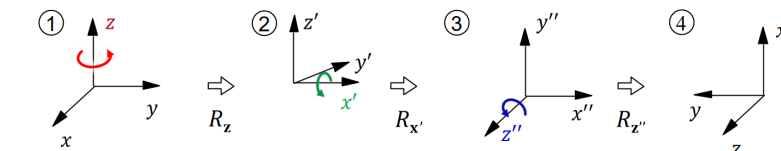
- Rotationen können verkettet werden: $\phi_{\mathbf{z},\gamma}(\phi_{\mathbf{y},\beta}(\phi_{\mathbf{x},\alpha}(\mathbf{a})))$, $\mathbf{a} \in \mathbb{R}^3$

Probleme mit Rotationsmatrizen:

- **Redundanz:** Neun Werte für eine Rotationsmatrix
- Probleme im Bereich des maschinellen Lernens

Eulerwinkel:

- Es ist möglich jede Rotation durch drei Rotationen um jeweils eine Rotationsachse darzustellen
- **Euler-Konvention:** $\mathbf{z} \mathbf{x}' \mathbf{z}''$ (lokale Drehung, Drehung verändert Achsen) oder $\mathbf{x} \mathbf{y} \mathbf{z}$ (globale Drehung, Drehung um feste Achsen)
- Winkel α, β, γ sind **Eulerwinkel** und beschreiben den Grad der Drehungen



- **Vorteile:** Kompakter und aussagekräftiger als Rotationsmatrizen

- **Nachteile:**

- Nicht eindeutig: In der Euler-Konvention $\mathbf{x} \mathbf{y}' \mathbf{z}''$ beschreiben die Eulerwinkel $(45^\circ, -90^\circ, 45^\circ)$ und $(30^\circ, -90^\circ, 60^\circ)$ die gleiche Rotation
- Nicht kontinuierlich: Kleine Änderung in der Orientierung können zu großen Änderungen der Eulerwinkel führen
- **Gimbal Lock:** Bei bestimmten Winkeln werden zwei Achsen voneinander abhängig \Rightarrow Ein Freiheitsgrad geht verloren

Bewertung der Darstellung von Orientierung mit 3×3 -Matrizen:

- **Vorteil:** Vektor und Rotationsmatrix sind anschaulich \Rightarrow übliche Form der Eingabe von Posen
- **Nachteil:** Darstellung als (\mathbf{p}, R) mit $\mathbf{p} \in \mathbb{R}^3$ und $R \in \text{SO}(3)$ führt dazu, dass Translation und Rotation getrennt durchgeführt werden müssen

\rightarrow **Ziel:** Geschlossene Darstellung von Rotation und Translation in einer Matrix

Affine Transformationen:

- Der **affine Raum** ist eine Erweiterung zum euklidischen Raum
- Beinhaltet Vektoren, die in **erweiterten, homogenen Koordinaten** ausgedrückt werden: $a = (a_x \ a_y \ a_z \ h)^\top, h \in \{0, 1\}$, wobei a für $h = 0$ einen Ortsvektor und für $h = 1$ einen Richtungsvektor beschreibt
- Für Rotationsmatrix R und Translation \mathbf{t} gilt nun:

$$\mathbf{b} = A \cdot \mathbf{x} + \mathbf{t} \Leftrightarrow \begin{pmatrix} \mathbf{b} \\ 1 \end{pmatrix} = \begin{pmatrix} A & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} + \begin{pmatrix} \mathbf{t} \\ 0 \end{pmatrix} = \begin{pmatrix} A & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}$$

womit also nun Translation und Rotation als eine allgemeine homogene 4×4 -Matrix beschrieben werden kann:

$$T = \begin{pmatrix} R & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}, \quad T \in \text{SE}(3) \text{ mit } \mathbf{t} \in \mathbb{R}^3 \text{ und } R \in \text{SO}(3)$$

- Eine **Translationsmatrix**, die eine Verschiebung um $\mathbf{t} = (t_x \ t_y \ t_z)^\top$ beschreibt, ist demnach:

$$T_{\text{trans}} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Es gilt $T^{-1} = \begin{pmatrix} R^\top & -R^\top \cdot \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix}$. Diese bildet $\mathbf{b} = R \cdot \mathbf{x} + \mathbf{t}$ wieder auf \mathbf{x} ab

- **Interpretationen von homogenen 4×4 -Matrizen:**

- **Lagebeschreibung:** ${}^A P_B$ beschreibt die Lage des Koordinatensystems B relativ zum Koordinatensystem A
- **Transformationsabbildung** zwischen Koordinatensystemen:

$${}^A T_B: {}^B P \rightarrow {}^A P, \quad {}^A P = {}^A T_B \cdot {}^B P$$

transformiert Koordinatensystem B in Koordinatensystem A

- **Transformationsoperator** innerhalb eines Koordinatensystems:

$$T: {}^A P_1 \rightarrow {}^A P_2, \quad {}^A P_2 = T \cdot {}^A P_1$$

transformiert einen Punkt P_1 in einen Punkt P_2 innerhalb des Koordinatensystems A

Beispiele 1/58-59

- Lagebeschreibungen können als Matrixprodukt verkettet werden, z.B.

$${}^{\text{BKS}} T_{O_3} = {}^{\text{BKS}} T_{O_1} \cdot {}^{O_1} T_{O_2} \cdot {}^{O_2} T_{O_3}$$

Quaternionen:

- Repräsentation ohne Nachteile von Rotationsmatrizen und Eulerwinkeln
- Menge der **Quaternionen** \mathbb{H} ist definiert durch:

$$\mathbb{C} + \mathbb{C}j \quad \text{mit } j^2 = -1 \text{ und } i \cdot j = -j \cdot i$$

wobei i die imaginäre Einheit ist

- Ein Element $\mathbf{q} \in \mathbb{H}$ hat die Form:

$$\mathbf{q} = (a, \mathbf{u})^\top = a + u_1 i + u_2 j + u_3 k \quad \text{mit } a \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^3 \text{ und } k = i \cdot j$$

- a heißt **Realteil** und $\mathbf{u} = (u_1, u_2, u_3)^\top$ heißt **Imaginärteil**
- **Rechenregeln:**

\cdot	1	i	j	k
1	1	i	j	k
i	i	-1	k	$-j$
j	j	$-k$	-1	i
k	k	j	$-i$	-1

- **Rechenoperationen:** Seien $\mathbf{q} = (a, \mathbf{u})^\top, \mathbf{r} = (b, \mathbf{v})^\top$ zwei Quaternionen
 - **Addition:** $\mathbf{q} + \mathbf{r} = (a + b, \mathbf{u} + \mathbf{v})^\top$
 - **Skalarprodukt:** $\langle \mathbf{q} | \mathbf{r} \rangle = a \cdot b + \langle \mathbf{u} | \mathbf{v} \rangle = a \cdot b + v_1 \cdot u_1 + v_2 \cdot u_2 + v_3 \cdot u_3$
 - **Multiplikation:** $\mathbf{q} \cdot \mathbf{r} = (a + u_1 i + u_2 j + u_3 k) \cdot (b + v_1 i + v_2 j + v_3 k)$
 - **Konjugierte Quaternion:** $\mathbf{q}^* = (a, -\mathbf{u})^\top$
 - **Norm:** $|\mathbf{q}| = \sqrt{\mathbf{q} \cdot \mathbf{q}^*} = \sqrt{\mathbf{q}^* \cdot \mathbf{q}} = \sqrt{a^2 + u_1^2 + u_2^2 + u_3^2}$
 - **Inverse:** $\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{|\mathbf{q}|^2}$
- **Einheitsquaternion** $\mathbb{S}^3 = \{\mathbf{q} \in \mathbb{H} \mid \|\mathbf{q}\|^2 = 1\}$
- Beschreibung eines Vektors $\mathbf{p} \in \mathbb{R}^3$ als Quaternion \mathbf{q} : $\mathbf{q} = (0, \mathbf{p})^\top$
- Beschreibung eines Skalars $s \in \mathbb{R}$ als Quaternion \mathbf{q} : $\mathbf{q} = (s, \mathbf{0})^\top$
- Sei eine Rotation beschrieben durch eine Drehachse \mathbf{a} mit $\|\mathbf{a}\| = 1$ und einen Drehwinkel θ , dann existiert hierfür eine Repräsentation als Quaternion: $\mathbf{q} = (\cos \frac{\theta}{2}, \mathbf{a} \sin \frac{\theta}{2})$
- Ein Punkt \mathbf{v} wird mit einer Quaternion \mathbf{q} rotiert durch: $\mathbf{v}' = \mathbf{q}\mathbf{v}\mathbf{q}^{-1} = \mathbf{q}\mathbf{v}\mathbf{q}^*$, wobei die letzte Gleichheit gilt, weil \mathbf{q} ein Einheitsquaternion ist
- Verkettung von Rotationen $f \circ h$: $f(h(\mathbf{v})) = \mathbf{q}(\mathbf{r}\mathbf{v}\mathbf{r}^*)\mathbf{q}^*$

Beispiel 1/72

Bewertung von Quaternionen:

- **Vorteile:** Kompakt, Anschaulich, Kein Gimbal Lock, Verkettung möglich, Stetige Repräsentation
- **Nachteil:** Nur Beschreibung von Rotation, keine Translation

SLERP Interpolation:

- SLERP Interpolation von \mathbf{q}_1 nach \mathbf{q}_2 mit Parameter $t \in [0, 1]$:

$$\text{SLERP}(\mathbf{q}_1, \mathbf{q}_2, t) = \frac{\sin((1-t) \cdot \theta)}{\sin \theta} \cdot \mathbf{q}_1 + \frac{\sin(t \cdot \theta)}{\sin \theta} \cdot \mathbf{q}_2 \quad \text{mit } \langle \mathbf{q}_1 | \mathbf{q}_2 \rangle = \cos \theta$$

- Ergebnis: Rotation mit konstanter Winkelgeschwindigkeit
 - **Problem:** Orientierungen in $\text{SO}(3)$ werden durch Einheitsquaternionen doppelt abgedeckt, weil \mathbf{q} und $-\mathbf{q}$ der gleichen Rotation entsprechen
- \Rightarrow SLERP berechnet deshalb nicht immer die kürzeste Rotation. Es muss geprüft werden, ob die Rotation von \mathbf{q}_1 zu \mathbf{q}_2 oder $-\mathbf{q}_1$ zu \mathbf{q}_2 kürzer ist

Duale Quaternionen:

- Erlauben es auch Translationen zu berücksichtigen
- **Duale Zahlen** sind Zahlen der Form:

$$d = p + \varepsilon \cdot s, \quad \text{wobei } \varepsilon^2 = 0$$

mit **Primärteil** p , **Sekundärteil** s

- **Rechenoperationen:** Seien $d_1 = p_1 + \varepsilon \cdot s_1$ und $d_2 = p_2 + \varepsilon \cdot s_2$ duale Zahlen
 - **Addition:** $d_1 + d_2 = p_1 + p_2 + \varepsilon \cdot (s_1 + s_2)$
 - **Multiplikation:** $d_1 \cdot d_2 = p_1 \cdot p_2 + \varepsilon \cdot (p_1 \cdot s_2 + p_2 \cdot s_1)$

- **Duale Quaternionen:**

$$DQ = (d_1, d_2, d_3, d_4), \quad d_i = dp_i + \varepsilon \cdot ds_i$$

- Primärteil dp_i enthält den Winkelwert $\theta/2$
- Sekundärteil ds_i enthält die Translationsgröße $d/2$
- **Multiplikationstabelle für duale Einheitsquaternionen:**

\cdot	1	i	j	k	ε	εi	εj	εk
1	1	i	j	k	ε	εi	εj	εk
i	i	-1	k	$-j$	εi	$-\varepsilon$	εk	$-\varepsilon j$
j	j	$-k$	-1	i	εj	$-\varepsilon k$	$-\varepsilon$	εi
k	k	j	$-i$	-1	εk	εj	$-\varepsilon i$	$-\varepsilon$
ε	ε	εi	εj	εk	0	0	0	0
εi	εi	$-\varepsilon$	εk	$-\varepsilon j$	0	0	0	0
εj	εj	$-\varepsilon k$	$-\varepsilon$	εi	0	0	0	0
εk	εk	εj	$-\varepsilon i$	$-\varepsilon$	0	0	0	0

- Rotation um eine Achse \mathbf{a} mit dem Winkel θ : $\mathbf{q}_r = \left(\cos \frac{\theta}{2}, \mathbf{a} \sin \frac{\theta}{2}\right) + \varepsilon \cdot (0, 0, 0, 0)$
- Translation mit dem Vektor $\mathbf{t} = (t_x, t_y, t_z)$: $\mathbf{q}_t = (1, 0, 0, 0) + \varepsilon \cdot \left(0, \frac{t_x}{2}, \frac{t_y}{2}, \frac{t_z}{2}\right)$
- Kombination zu einer Transformation T : $\mathbf{q}_T = \mathbf{q}_t \mathbf{q}_r$
- Eine Transformation \mathbf{q}_T wird auf einen Punkt \mathbf{p} als duale Quaternion wie folgt angewendet: $\mathbf{p}' = \mathbf{q}_T \mathbf{p} \mathbf{q}_T^*$, mit $\mathbf{q}_T^* = (\mathbf{q}_t \mathbf{q}_r)^* = \mathbf{q}_r^* \mathbf{q}_t^*$
- Konjugieren von $\mathbf{q} = \mathbf{p} + \varepsilon \cdot \mathbf{s}$: $\mathbf{q}^* = \mathbf{p}^* - \varepsilon \cdot \mathbf{s}^*$

Beispiel 1/83-85

Bewertung von dualen Quaternionen:

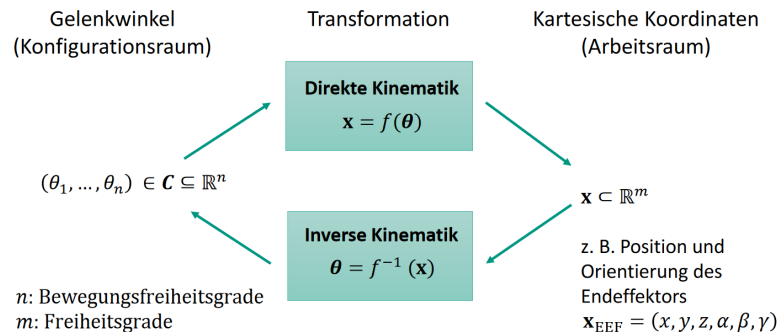
- **Vorteile:** Erlauben Lagebeschreibung und Transformationen, Geringere Redundanz (nur 8 statt 12 Werte bei homogener Matrix)
- **Nachteile:** Lage durch Angabe einer Dualquaternion zu beschreiben ist relativ schwierig, Komplexe Verarbeitungsvorschriften

2 Kinematik

Kinematisches Modell: Beschreibt Zusammenhänge zwischen **Gelenkwinkelraum** (Konfigurationsraum) und **Posenraum des Endeffektors** (Arbeitsraum)

Direkte und Inverse Kinematik:

- **Direkte Kinematik:**
 - Eingabe: Gelenkwinkelstellungen des Roboters
 - Ausgabe: Pose des Endeffektors
 - z.B. Wo befindet sich meine Hand?
- **Inverse Kinematik:**
 - Eingabe: Zielpose des Endeffektors
 - Ausgabe: Gelenkwinkelstellungen
 - z.B. Wie bewege ich meine Hand zum Ziel?



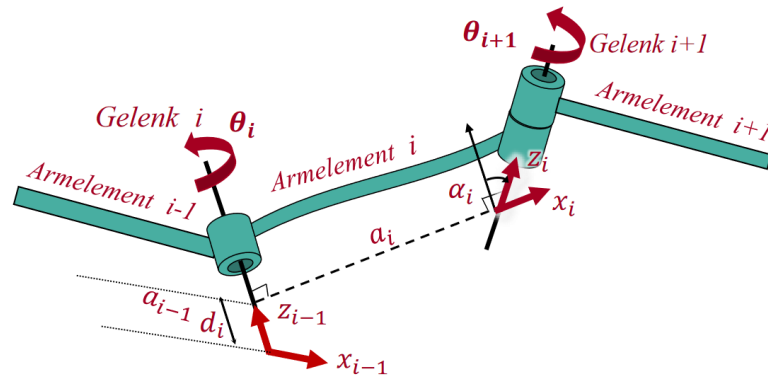
Kinematische Kette wird von mehreren Körpern gebildet, die durch Gelenke kinematisch verbunden sind (z.B. Roboterarm). Unterscheidung zwischen:

- **Offene** kinematische Kette: Nur ein Ende ist fest, anderes Ende frei bewegbar
- **Geschlossene** kinematische Kette: Beide Enden der Kette sind fest

Für jedes Glied müssen **6 Parameter** für die Transformation zwischen Gelenken bestimmt werden (3 Rotationsparameter, 3 Translationsparameter)

Denavit-Hartenberg (DH) Konvention:

- Durch die geschickte Wahl der Koordinatensysteme lassen sich die Parameter zur Beschreibung eines Armelements auf 4 reduzieren
- **Regeln für Koordinatensysteme:**
 - z_{i-1} -Achse liegt entlang der Bewegungsachse des i -ten Gelenks
 - x_i -Achse verläuft entlang der gemeinsamen Normalen (Kreuzprodukt von z_{i-1} und z_i) von z_{i-1} und z_i
 - y_i -Achse vervollständigt das Koordinatensystem entsprechend der Rechte-Hand-Regel
- **Parameter des Armelements (DH-Parameter):**
 - **Armelementlänge** a_i beschreibt den Abstand von z_{i-1} zu z_i entlang x_i
 - **Armelementverdrehung** α_i beschreibt den Winkel von z_{i-1} zu z_i um x_i
 - **Gelenkabstand** d_i ist der Abstand zwischen der x_{i-1} -Achse und x_i -Achse entlang der z_{i-1} -Achse
 - **Gelenkwinkel** θ_i ist der Winkel von x_{i-1} zu x_i um z_{i-1}



- DH-Parameter beschreiben wie aufeinanderfolgende Gelenke ineinander transformiert werden
- **DH-Transformationsmatrizen:** Beschreibung mit homogenen Matrizen

1. Rotation θ_i : $R_{z_{i-1}}(\theta_i) = \begin{pmatrix} \cos \theta_i & -\sin \theta_i & 0 & 0 \\ \sin \theta_i & \cos \theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

$$2. \text{ Translation } d_i: T_{z_{i-1}}(d_i) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$3. \text{ Translation } a_i: T_{x_i}(a_i) = \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$4. \text{ Rotation } \alpha_i: R_{x_i}(\alpha_i) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha_i & -\sin \alpha_i & 0 \\ 0 & \sin \alpha_i & \cos \alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Zusammenführen zu einer Matrix:

$$\begin{aligned} A_{i-1,i} &= R_{z_{i-1}}(\theta_i) \cdot T_{z_{i-1}}(d_i) \cdot T_{x_i}(a_i) \cdot R_{x_i}(\alpha_i) \\ &= \begin{pmatrix} \cos \theta_i & -\sin \theta_i \cdot \cos \alpha_i & \sin \theta_i \cdot \sin \alpha_i & a_i \cdot \cos \theta_i \\ \sin \theta_i & \cos \theta_i \cdot \cos \alpha_i & -\cos \theta_i \cdot \sin \alpha_i & a_i \cdot \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

- Inverse DH-Transformation:

$$A_{i-1,i}^{-1} = A_{i,i-1} = \begin{pmatrix} \cos \theta_i & \sin \theta_i & 0 & -a_i \\ -\cos \alpha_i \cdot \sin \theta_i & \cos \theta_i \cdot \cos \alpha_i & \sin \alpha_i & -d_i \cdot \sin \alpha_i \\ \sin \theta_i \cdot \sin \alpha_i & -\sin \alpha_i \cdot \cos \theta_i & \cos \alpha_i & -d_i \cdot \cos \alpha_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Durch Multiplikation der DH-Matrizen lässt sich die Lage der einzelnen Koordinatensysteme bezüglich des Bezugskordinatensystems bestimmen

Direktes kinematisches Problem: Stellung des Endeffektors (EFF) in Bezug auf das BKS ist gegeben durch:

$$S_{\text{Basis, EFF}} = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot \dots \cdot A_{n-2,n-1}(\theta_{n-1}) \cdot A_{n-1,n}(\theta_n)$$

Lösung des Problems ergibt sich aus Einsetzen der Gelenkwinkel in obige Gleichung.

Beispiele 2/38-48

Oft interessiert man sich für verwandte Beziehungen wie z.B. Gelenkwinkelgeschwindigkeiten → Endeffektor-Geschwindigkeit. Dafür muss man die Vorwärtskinematik ableiten → **Jacobi-Matrix**

Jacobi-Matrix: Für eine differenzierbare Funktion $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$, $f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix}$ und

$\mathbf{x} \in \mathbb{R}^n$ ist die Jacobi-Matrix für ein $\mathbf{a} \in \mathbb{R}^n$ wie folgt:

$$J_f(\mathbf{a}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{a}) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{a}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\mathbf{a}) & \cdots & \frac{\partial f_m}{\partial x_n}(\mathbf{a}) \end{pmatrix} \in \mathbb{R}^{m \times n}$$

Problem: Vorwärtskinematik ist matrixwertig \rightarrow Jacobi-Matrix nicht definiert

Lösung: Vektorwertige Repräsentation wählen, z.B. mit Eulerwinkel

Geschwindigkeitsraum und Krafraum:

- Annahme: Kinematische Kette bewege sich entlang einer Trajektorie $\theta: \mathbb{R} \rightarrow \mathbb{R}^n$, wobei θ die Gelenkwinkelstellungen zu einem Zeitpunkt t beschreibt
- Pose des End-Effektors $\mathbf{x}(t) \in \mathbb{R}^6$ zum Zeitpunkt t : $\mathbf{x}(t) = \mathbf{f}(\theta(t))$, wobei f die Funktion für die Vorwärtskinematik ist
- **Beziehung zwischen Endeffektor- und Gelenkwinkelgeschwindigkeiten:**

$$\dot{\mathbf{x}}(t) = J_f(\theta(t)) \cdot \dot{\theta}(t)$$

- **Beziehung zwischen Kräfte und Momente am End-Effektor und Drehmomenten in den Gelenken:**

$$\boldsymbol{\tau}(t) = J_f^\top(\theta(t)) \cdot \mathbf{F}(t)$$

wobei $\mathbf{F}: \mathbb{R} \rightarrow \mathbb{R}^6$ der Kraft-Momenten-Vektor am End-Effektor und $\boldsymbol{\tau}: \mathbb{R} \rightarrow \mathbb{R}^n$ die Drehmomente in Gelenken ist

- Im Geschwindigkeits- und Krafraum lässt sich die Frage nach der Inversen Kinematik durch die Inverse der Jacobi-Matrix lösen, z.B. Welche Gelenkwinkelgeschwindigkeiten sind notwendig, um eine End-Effektor-Geschwindigkeit zu realisieren? \rightarrow Löse nach $\dot{\theta}(t)$ auf

Berechnung der Jacobi-Matrix:

- Jede Spalte der Jacobi-Matrix gehört zum Gelenk θ_i der kinematischen Kette
- Wenn j -tes Gelenk ein Translationsgelenk ist, das eine Translation in Richtung des Einheitsvektors $\mathbf{z}_j \in \mathbb{R}^3$ durchführt, gilt für die j -te Spalte der Jacobi-Matrix:

$$J_j(\theta) = \begin{bmatrix} \mathbf{z}_j \\ \mathbf{0} \end{bmatrix} \in \mathbb{R}^6$$

- Wenn j -tes Gelenk ein Rotationsgelenk ist, das eine Rotation um die Rotationsachse $\mathbf{z}_j \in \mathbb{R}^3$ an der Position $\mathbf{p}_j \in \mathbb{R}^3$ durchführt, gilt für j -te Spalte der Jacobi-Matrix:

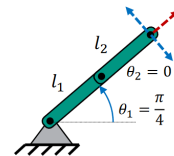
$$J_j(\boldsymbol{\theta}) = \begin{bmatrix} \mathbf{z}_j \times (\mathbf{f}(\boldsymbol{\theta}) - \mathbf{p}_j) \\ \mathbf{z}_j \end{bmatrix} \in \mathbb{R}^6$$

Beispiel 2/68-72

Kinematische Singularitäten:

- **Definition:** Wenn sich Roboter in einer Konfiguration $\boldsymbol{\theta}_{\text{singular}} \in C$ befindet, in der er sich nicht instantan in eine oder mehr Richtungen bewegen kann
- Bedeutet, dass Jacobi-Matrix $J(\boldsymbol{\theta})$ keinen vollen Rang bzw. min. zwei linear abhängige Spalten hat, also nicht invertierbar ist

Es existiert keine Gelenkwinkelgeschwindigkeit, die eine Endeffektor-Geschwindigkeit in die **rote Richtung** erzeugt.
 \Rightarrow Die Konfiguration ist singular.

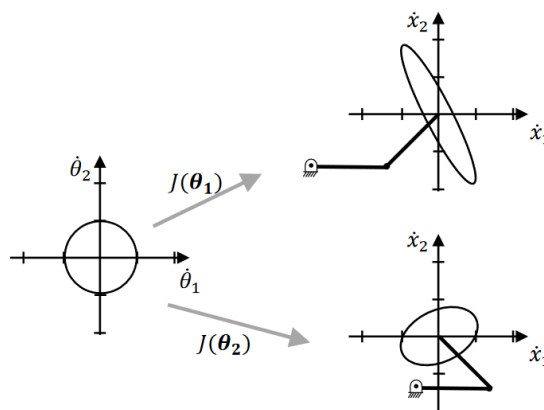


Manipulierbarkeit: Maß für die Bewegungsfreiheit des Endeffektors, bzw. wie nahe eine Konfiguration an einer Singularität ist

- Nutze $J(\boldsymbol{\theta})$ um Einheitskreis der Gelenkwinkel-Geschwindigkeiten in den Raum der Endeffektor-Geschwindigkeiten abzubilden

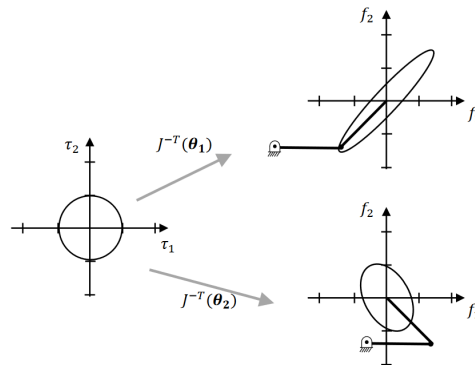
\rightarrow Resultat: **Ellipsoid der Manipulierbarkeit**

- Kreis: Bewegungen des Endeffektors in alle Richtungen uneingeschränkt möglich
- Degenerierte Fälle (gestauchtes Ellipsoid): Endeffektor-Bewegung ist in bestimmten Richtungen eingeschränkt



- Singulärwerte von $J(\boldsymbol{\theta})$ sind die Wurzeln der Eigenwerte von $J(\boldsymbol{\theta}) \cdot J(\boldsymbol{\theta})^\top$, also $\sigma_i = \sqrt{\lambda_i}$, wobei λ_i ein EW von $J(\boldsymbol{\theta}) \cdot J(\boldsymbol{\theta})^\top$ ist
- **Skalare Maße** für die Manipulierbarkeit:
 - Kleinster Singulärwert $\mu_1(\boldsymbol{\theta}) = \sigma_{\min}(A(\boldsymbol{\theta}))$
 - Inverse Kondition: $\mu_2(\boldsymbol{\theta}) = \frac{\sigma_{\min}(A(\boldsymbol{\theta}))}{\sigma_{\max}(A(\boldsymbol{\theta}))}$
 - Determinante: $\mu_3(\boldsymbol{\theta}) = \det A(\boldsymbol{\theta})$
- **Kraft-Ellipsoid:** Selbiges kann man auch für den Krafraum einführen

$$\boldsymbol{\tau}(t) = J_f^\top(\boldsymbol{\theta}(t)) \cdot \mathbf{F}(t) \quad \rightarrow \quad \mathbf{F}(t) = J_f^{-\top}(\boldsymbol{\theta}(t)) \cdot \boldsymbol{\tau}(t)$$



Geometrisches Modell:

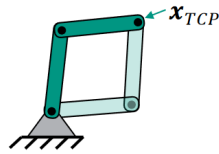
- **Einsatzbereiche:**
 - Abstandsmessung und Kollisionserkennung
 - Graphische Darstellung von Körpern
 - Berechnung der Bewegungen von Körpern
 - Ermittlung der wirkenden Kräfte und Momente
- **Klassifizierung** nach **Raum** (2D oder 3D) oder **Grundprimitiven** (Kanten- bzw. Drahtmodelle, Flächen- bzw. Oberflächenmodelle, Volumenmodell)

3 Inverse Kinematik

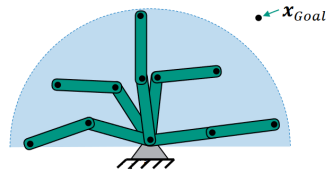
Problemstellung: siehe S. 9

Damit Inverse $\boldsymbol{\theta} = f^{-1}(x)$ existiert, muss die Vorwärtskinematik f bijektiv sein, aber:

- Vorwärtskinematik ist i. A. nicht injektiv

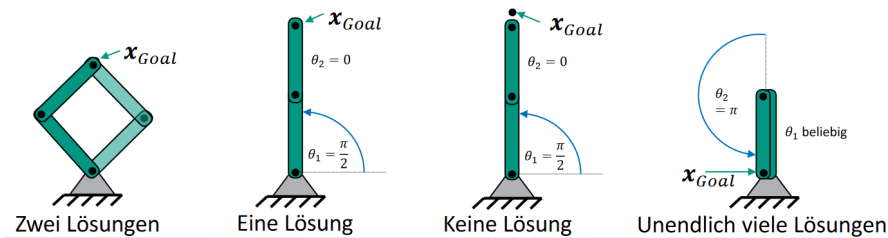


- Vorwärtskinematik ist i. A. nicht surjektiv



Unterschiedliche Fälle der inversen Kinematik:

- Es gibt zwei unabhängige Lösungen (Normalfall).
- Es gibt genau eine Lösung (Rand des Arbeitsraums).
- Es gibt keine Lösung (Außerhalb des Arbeitsraums).
- Es gibt unendlich viele Lösungen (Zielpunkt in der Basis)



Geometrische Methoden:

- Nutze geometrische Beziehungen (z.B. Sinus-/Kosinussatz), um θ aus T_{TCP} zu bestimmen
- Bei mehreren Gelenken kann das aber sehr schwierig werden
- **Beispiel:** 3/23-25

Algebraische Methoden:

- Führe Koeffizientenvergleich der beiden Matrizen der gewünschten TCP-Pose und der Transformation aus dem kinematischen Modell durch
→ 16 Gleichungen bei homogenen Matrizen, von denen 4 trivial sind (letzte Zeile immer $\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}$)

- **Beispiel:** 3/29-32
- **Problem:** Oft können nicht alle Gelenkwinkel aus den 12 Gleichungen bestimmt werden
- **Lösung:** Kenntnis der Transformationen erhöht Chance, die Gleichungen zu lösen, z.B. sei Gleichung $P_{TCP} = A_{0,1}(\theta_1) \cdot A_{1,2}(\theta_2) \cdot A_{2,3}(\theta_3) \cdot A_{3,4}(\theta_4) \cdot A_{4,5}(\theta_5) \cdot A_{5,6}(\theta_6)$ gegeben
 1. Invertiere $A_{0,1}(\theta_1)$ und multipliziere beide Seiten der Gleichung mit $A_{0,1}^{-1}$
 2. Versuche im neuen Gleichungssystem den Wert einer Variablen zu bestimmen
 3. Versuche weitere Gleichung zu finden, die durch Substitution der im letzten Schritt gefundenen Lösung lösbar ist
 4. Falls keine Lösungen gefunden werden kann, so muss eine weitere Matrix $A_{1,2}(\theta_2)$ invertiert werden
 5. Wiederhole die Schritte 1 - 4 bis alle Gelenkwinkel ermittelt sind

→ Große Wahrscheinlichkeit, dass das Gleichungssystem gelöst werden kann

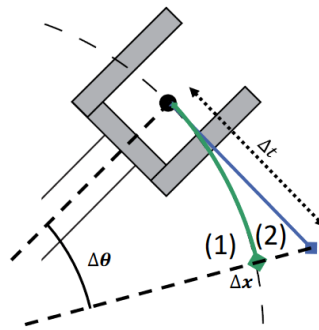
Im Folgenden werden **numerische Methoden** betrachtet:

Gradientenabstieg:

- **Fehlerfunktion** für die Zielpose $\mathbf{x}_{\text{Goal}} \in W$: $e(\boldsymbol{\theta}) = \|\mathbf{x}_{\text{Goal}} - f(\boldsymbol{\theta})\|^2$
- Wähle Startkonfiguration $\boldsymbol{\theta}_0 \in C, i = 0$ und Schrittlänge $\gamma \in \mathbb{R}$
- Solange $e(\boldsymbol{\theta}_i) > e_{\text{Threshold}}$:
 - $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \gamma \cdot 2 \cdot J^T(\boldsymbol{\theta}_i) \cdot (f(\boldsymbol{\theta}_i) - \mathbf{x}_{\text{Goal}})$, wobei J die Jacobi-Matrix der Vorwärtskinematik ist
 - $i = i + 1$
- Je nach gewählter Startkonfiguration erhält man unterschiedliche Lösungen

Pseudoinverse:

- **Differenzenquotient:**
 - Tatsächliche Bewegung gemäß $\dot{\mathbf{x}}(t) = J(\boldsymbol{\theta}(t)) \cdot \dot{\boldsymbol{\theta}}(t)$
 - Annäherung mittels Differenzenquotient: $\Delta \mathbf{x} \approx J(\boldsymbol{\theta}) \Delta \boldsymbol{\theta}$

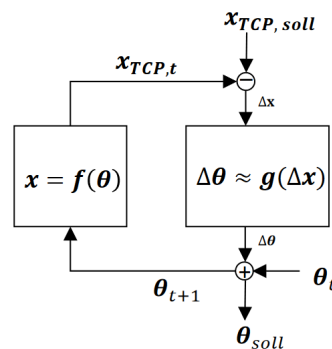


- Suche Lösung für das inverse Problem: $\Delta\theta = J_f^{-1}(\theta) \cdot \Delta\mathbf{x}$, aber J_f muss nicht unbedingt invertierbar sein \rightarrow Nutze Pseudoinverse
- Es lässt sich herleiten:

$$\Delta\theta = (J_f^\top J_f)^{-1} J_f^\top \Delta\mathbf{x} = J_f^+(\theta) \Delta\mathbf{x}$$

wobei J_f^+ die **Pseudoinverse** von J_f ist ($A^+ = (A^\top A)^{-1} A^\top$)

- **Algorithmus:**



- **Problem:** Pseudoinverse ist in der Nähe von Singularitäten instabil!
- **Lösung:** Damped least squares-Methode

Damped Least Squares:

- Führe Dämpfungskonstante $\lambda > 0$ ein und berechne

$$\Delta\theta = J^\top (J J^\top + \lambda^2 I)^{-1} \Delta\mathbf{x}$$

- Analyse der Stabilität über **Singulärwertzerlegung**: $J = U D V^\top = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$, wobei U, V orthogonale Matrizen, D Diagonalmatrix mit Singulärwerten auf der Diagonalen, $\mathbf{u}_i, \mathbf{v}_i$ die i -te Spalte von U bzw. V und r der rang von J ist

- Es lässt sich herleiten:

- Pseudoinverse: $J^+ = \sum_{i=1}^r \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^\top$

- Damped Least Squares: $J^\top (JJ^\top + \lambda^2 I)^{-1} = \sum_{i=1}^r \frac{\sigma_i}{\sigma_i^2 + \lambda^2} \mathbf{v}_i \mathbf{u}_i^\top$

→ Für $\sigma_i \rightarrow 0$ wird die Pseudoinverse instabil, Damped-Least-Squares bleibt aber wohldefiniert. Für große σ_i verglichen mit λ verhält sich Damped-Least-Squares wie die Pseudoinverse.

4 Dynamik

Dynamisches Modell: Beschreibt Zusammenhang zwischen Antriebs- und Kontaktkräfte, welche in einem mechanischen Mehrkörpersystem auftreten und deren resultierenden Beschleunigungen und Bewegungen

Allgemeine Bewegungsgleichung:

$$\boldsymbol{\tau} = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q}) + \varepsilon(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$$

mit:

- $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$: $n \times 1$ Vektor der generalisierten Koordinaten (Position, Geschwindigkeit und Beschleunigung)
- $\boldsymbol{\tau}$: $n \times 1$ Vektor der generalisierten Kräfte
- $M(\mathbf{q})$: $n \times n$ Massenträgheitsmatrix
- $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$: $n \times 1$ Vektor mit Zentripetal- und Corioliskomponenten
- $g(\mathbf{q})$: $n \times 1$ Vektor der Gravitationskomponenten
- $\varepsilon(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$: $n \times 1$ Nichtlineare Effekte, wie z.B. Reibung

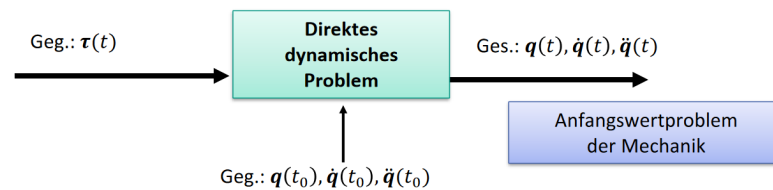
Generalisierte Koordinaten:

- **Definition:** Minimaler Satz an voneinander unabhängigen Koordinaten, der den aktuellen Systemzustand vollständig beschreibt
- Allgemein: Roboter besteht aus N Partikel. Für jeden Ortsvektor eines Partikels braucht man 3 Raumkoordinaten, insgesamt $3N$ Koordinaten, um das System zu beschreiben
- Partikel können sich wegen Verbindungen und Gelenken nicht unabhängig voneinander bewegen → Einführung von **Zwangsbedingungen**

- $3N$ Koordinaten lassen sich mit k unabhängigen Zwangsbedingungen auf $3N - k$ unabhängige generalisierte Koordinaten q_i reduzieren
- **Beispiel:** 4/11-15

Direktes Dynamisches Problem:

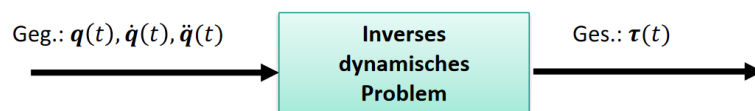
- Gegeben externe Kraft und aktueller Bewegungszustand, was ist die neue Bewegung des Systems?



- Löse Differentialgleichung $\tau = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q})$ nach $\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t)$

Inverses Dynamisches Problem:

- Aus den gewünschten Bewegungsparametern sollen die dazu erforderlichen Kräfte und Momente ermittelt werden



- Berechne rechten Teil der Gleichung $\tau = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + g(\mathbf{q})$ durch Einsetzen

Modellierung der Dynamik: Versuche Terme der allgemeinen Bewegungsgleichung herzuleiten

Methode nach Lagrange:

1. Ermittle E_{kin} und E_{pot}
2. Drücke E_{kin} und E_{pot} in generalisierten Koordinaten als **Lagrange-Funktion** aus

$$L(\mathbf{q}, \dot{\mathbf{q}}) = E_{\text{kin}}(\mathbf{q}, \dot{\mathbf{q}}) - E_{\text{pot}}(\mathbf{q})$$

3. Für jedes Gelenk i ist die Bewegungsgleichung:

$$\tau_i = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i}$$

Beispiele: 4/26-35

Eigenschaften:

- Einfaches Aufstellen der Gleichungen
- Geschlossenes Modell, analytisch auswertbar
- Berechnung sehr umfangreich $\mathcal{O}(n^3)$
- Nur Antriebsmomente werden berechnet

Methode nach Newton-Euler:

- **Newton-Euler Gleichungen:**

$$\mathbf{f} = I\mathbf{a} + \mathbf{v} \times I\mathbf{v}$$

wobei

$$\mathbf{f} = \begin{pmatrix} \mathbf{n}_c \\ \mathbf{f} \end{pmatrix}$$

$$\mathbf{v} = \begin{pmatrix} \boldsymbol{\omega} \\ \mathbf{v}_c \end{pmatrix}$$

$$\mathbf{a} = \begin{pmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\mathbf{v}}_c \end{pmatrix}$$

\mathbf{v}_c : Lineare Geschwindigkeit des Körpers in Bezug auf CoM

$\dot{\mathbf{v}}_c$: Lineare Beschleunigung des Körpers in Bezug auf CoM

$\mathbf{f}, \mathbf{v}, \mathbf{a}$: 6D Kraft- bzw. Bewegungsvektoren welche alle auf den Körper wirkenden Kräfte und Bewegungen (Geschwindigkeit, Beschleunigung) beschreiben

- Die Beschleunigungen $\ddot{\mathbf{c}}_i$ und $\dot{\boldsymbol{\omega}}_i$ eines Armelementes i hängen von den Beschleunigungen der vorhergehenden Armelemente ab
→ Beschleunigungen können über kinematisches Modell von der Basis zum Greifer rekursiv berechnet werden → **Vorwärtsgleichungen**
- Die Kraft \mathbf{f}_i und das Drehmoment $\mathbf{n}_{C,i}$, die auf ein Armelement i wirken, hängen von den nachfolgenden Armelementen ab
→ Kräfte und Momente können vom Greifer zur Basis rekursiv berechnet werden
→ **Rückwärtsgleichungen**
→ **Rekursiver Newton-Euler Algorithmus (RNEA)**

Rekursiver Newton-Euler Algorithmus:

1. Rekursive Berechnung der Geschwindigkeit und Beschleunigung jedes einzelnen Armelements i von der Basis bis zum Endeffektor:
 - Geschwindigkeit: $\mathbf{v}_i = \mathbf{v}_{p(i)} + \Phi_i \dot{\mathbf{q}}_i$ mit $\mathbf{v}_0 = 0$
 - $\dot{\mathbf{q}}_i$: Generalisierte Geschwindigkeit des Armelements i
 - Φ_i : $6 \times n$ Bewegungsmatrix (Abhängig vom Gelenktyp)
 - $\mathbf{v}_{p(i)}$: Geschwindigkeit des Vorgängerelements $p(i)$
 - Beschleunigung: $\mathbf{a}_i = \mathbf{a}_{p(i)} + \Phi_i \ddot{\mathbf{q}}_i + \dot{\Phi}_i \dot{\mathbf{q}}_i$ mit $\mathbf{a}_0 = -\mathbf{a}_g$

2. Berechnung der Kräfte/Momente jedes einzelnen Armelements i mithilfe Newton-Euler:

$$\mathbf{f}_i^a = \mathbf{I}_i \mathbf{a}_i + \mathbf{v}_i \times \mathbf{I}_i \mathbf{v}_i$$

- \mathbf{f}_i^a : Kräfte, welche aufgrund von \mathbf{a}_i auf das Armelement i wirken
 - \mathbf{I}_i : Trägheitsmoment des Armelements i
 - \mathbf{v}_i : Geschwindigkeit des Armelements i (in Schritt 1 berechnet)
 - \mathbf{a}_i : Beschleunigung des Armelements i (in Schritt 1 berechnet)
3. Rekursive Berechnung der Kräfte zwischen den Armelementen und der generalisierten Kräfte für den jeweiligen Gelenktyp:

- $\mathbf{f}_i = \mathbf{f}_i^a - \mathbf{f}_i^e + \sum_{j \in c(i)} \mathbf{f}_j$
- $\boldsymbol{\tau}_i = \boldsymbol{\Phi}_i^\top \mathbf{f}_i$
 - \mathbf{f}_i : Resultierende Kraft am Armelement i
 - \mathbf{f}_i^e : Summe aller externen Kräfte, die an i wirken
 - \mathbf{f}_j : Kraft eines anliegenden Armelementes j
 - $c(i)$: Menge nachfolgender Armelemente i in kinematischer Kette
 - $\boldsymbol{\Phi}_i$: $6 \times n$ Bewegungsmatrix (Abhängig vom Gelenktyp)
 - $\boldsymbol{\tau}_i$: Generalisierte Kräfte/Momente an i

Eigenschaften der Methode nach Newton-Euler:

- Beliebige Anzahl von Gelenken
- Belastungen der Armelemente werden berechnet
- Aufwand $\mathcal{O}(n)$
- Rekursion

Herausforderungen der Dynamik:

- Nichtlineare Kräfte (wie z.B. Reibung) können nicht direkt modelliert werden, haben jedoch einen großen Einfluss
- Dynamik eines Roboters kann sich im Laufe der Zeit stark verändern z.B. durch Abnutzung
- Dynamik variiert stark in Abhängigkeit von der auszuführenden Aufgabe

5 Regelung