

Aufgabenorientierte Regelung – Entkopplung im Task-Raum

Motivation



Robotic Assembly of Complex Planar Parts: an Experimental Evaluation

P. Robuffo Giordano, A. Stemmer, K. Arbter
and A. Albu-Schäffer

German Aerospace Center (DLR)

 Deutsches Zentrum
für Luft- und Raumfahrt e.V.
an der Technischen Universität München

 **SAPIENZA**
UNIVERSITÀ DI ROMA

Robotergleichungen in aufgabenbezogenen Koordinaten

Robotermodell im Gelenkraum:

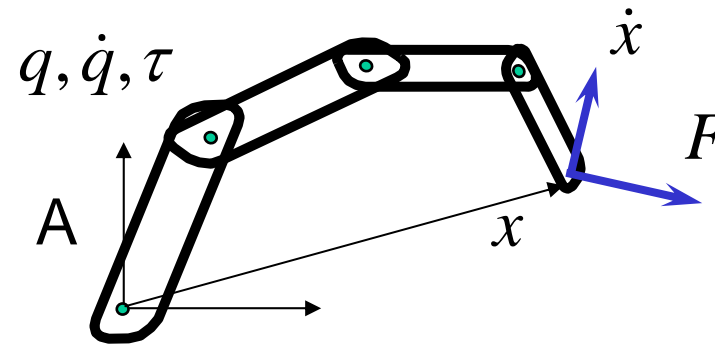
$$M(q)\ddot{q} + c(q, \dot{q}) + g(q) = \tau$$

q -Koordinaten von Q (\mathbb{R}^6 oder 6-Torus)

\dot{q} - Vektor

$c(q, \dot{q}), g(q), \tau$ - Kovektoren

$M(q)$ - kovarianter Tensor zweiter Ordnung



Vorwärtskinematik: lokale Abbildung zwischen Mannigfaltigkeiten (Q und $SE3$)

$$f : Q \rightarrow SE3, \quad x = f(q)$$

f ist eine **lokale** 1:1 Abbildung zwischen Mannigfaltigkeiten

- für **nicht-redundante Manipulatoren** ($\dim(Q)=6$) **und**
- **in nicht-singulären Konfigurationen**

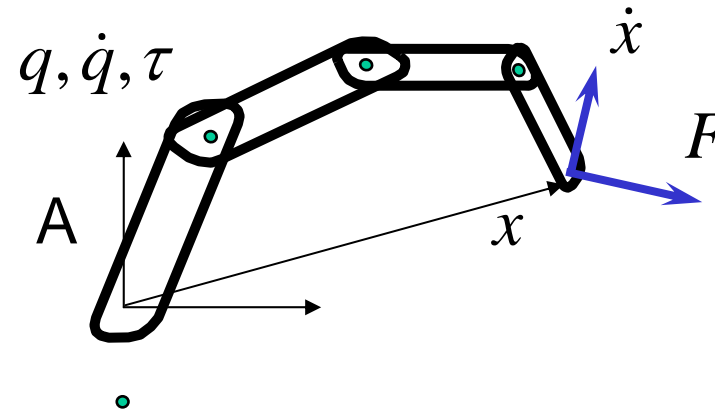
Dann wissen wir wie alle Größen transformieren!

Robotergleichungen in aufgabenbezogenen Koordinaten

Robotermodell in $SE3$ Koordinaten:

$$M_K(q)\ddot{x} + c_K(q, \dot{q}) + g_K(q) = F$$

$$\begin{aligned} x &= f(q); & q &= f^{-1}(x) \\ \dot{x} &= J(q)\dot{q}; & \dot{q} &= J^{-1}(q)\dot{x} \end{aligned}$$



Und die Beschleunigung?

$$\ddot{x} = J(q)\ddot{q} + \dot{J}(q)\dot{q}$$

Zusatzinfo: (ein Vektor auf der $2n$ Mannigfaltigkeit mit Koordinaten (q, \dot{q}) : *Tangentenbündel*) -Tafelerklärung

$$F = J^{-T}(q)\tau; \quad \tau = J^T(q)F \quad (\text{das gleiche mit } g_K(q))$$

$$c_K(q, \dot{q}) = J^{-T}(q)c(q, \dot{q}) - \underline{M_K(q)\dot{J}(q)\dot{q}}$$

$$M_K(x) = J^{-T}(q)M(q)J^{-1}(q); \quad M(q) = J^T(q)M_K(x)J(q)$$

alternative Formulierung:

$$M_x(x)\ddot{x} + c_x(x, \dot{x}) + g_x(x) = F$$

eingeschränkt auf Konfigurationen in denen $x = f(q)$ eine 1:1 Abbildung ist

Zusatzfolie Jakobimatrix Manipulator

wenn $T_{i,j} = \begin{bmatrix} R_{i,j} & p_{i,j} \\ 0 & 1 \end{bmatrix}$

für serielle Manipulatoren mit rotatorischen Gelenken ist das basic Jacobian:

$$J_j(q) = \begin{bmatrix} R_{j,1}z_1 \times p_{1,n+1}, & \cdots & , R_{j,n}z_n \times p_{n,n+1} \\ R_{j,1}z_1, & \cdots & , R_{j,n}z_n \end{bmatrix} \quad \begin{array}{l} z_i : \text{Achse Gelenk } i \\ z_i, p_{i,n+1} \text{ ausgedrückt im} \\ \text{Koordinatensystem der Achse } i \end{array}$$

für serielle Manipulatoren mit translatorischen Gelenken ist das basic Jacobian:

$$J_j(q) = \begin{bmatrix} R_{j,1}z_1, & \cdots, & R_{j,n}z_n \\ 0, & \cdots, & 0 \end{bmatrix}$$

(Skizze an der Tafel)

zur Herleitung siehe z.B.
[Spong, Khalil, Murray]

SE3 Koordinaten und Jakobimatrizen

(Was bedeutet hier genau F ? Wie berechne ich J ?)

Robotermodell in SE3 Koordinaten x :

$$M_K(q)\ddot{x} + c_K(q, \dot{q}) + g_K(q) = F$$

Ist hier F ein Wrench $F = (f, m)^T$?

$$x = f(q); \quad q = f^{-1}(x)$$

$$x = \begin{bmatrix} p_x \\ p_y \\ p_z \\ \alpha \\ \beta \\ \gamma \end{bmatrix} \quad \dot{x} = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix}$$

F ist eine verallgemeinerte Kraft dual zu \dot{x} s.d. $F^T \dot{x} = P$

Jakobimatrizen allgemein:

$$\dot{x} = \frac{df(q)}{dt} = \frac{\partial f(q)}{\partial q} \dot{q} = J(q)\dot{q}$$

Der duale Vector zum m ist ω .

$$\dot{x}_j = J_j(q)\dot{q}, \quad \tau = J_j^T(q)F_j \quad \text{mit} \quad \dot{x}_j = \begin{bmatrix} v_j \\ \omega_j \end{bmatrix}, \quad F_j = \begin{bmatrix} f_j \\ m_j \end{bmatrix}$$

ausgedrückt im Koordinatensystem j
 (basic Jacobian, geometric Jacobian)

zur Herleitung on J_j siehe z.B. [Spong, Khalil, Murray]

z.B. ausgedrückt im TCP Koordinatensystem - **body** Jacobian

Zusatzfolie Jakobimatrizen für Orientierung

(wie komme ich von ω zur notwendigen Vektordarstellung von SO3?)

Die Winkelgeschwindigkeit ω ist zwar nicht zu lokalen Koordinaten von SO3 integrierbar, es besteht aber folgende Beziehung zur Ableitung von R :

$$\hat{\omega}_{i,j}^j = R_{i,j}^T \dot{R}_{i,j}$$

(Herleitung
z.B. Murray, S.52)

Matrix Repräsentierung eines Vektors

$$\hat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \text{ s.d. } \omega \times r = \hat{\omega} r$$

Vektorprodukt als Matrixoperation

Da lokale Koordinaten x_R von SO3 auch in Abhängigkeit von R ausgedrückt sind

$$x_R = f_R(R_v) \Rightarrow \dot{x}_R = J_{f_R} \dot{R}_v$$

R_v enthält alle Elemente von R als Vektor angeordnet

kann man die Transformation zwischen den beiden Vektortypen herleiten:

$$\dot{x}_R = J_R \omega$$

Somit gilt für die Jakobimatrix

$$\dot{x} = \begin{bmatrix} v \\ \dot{x}_R \end{bmatrix} = \underbrace{\begin{bmatrix} I_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & J_R \end{bmatrix}}_{J_{TR}} \begin{bmatrix} v \\ \omega \end{bmatrix} = \underbrace{J_{TR} J_b}_J \dot{q} = J \dot{q}$$

Zusatzfolie Jakobimatrizen Orientierung

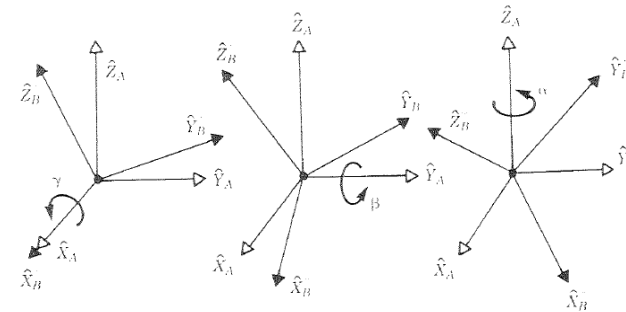
Beispiel RollPitchYaw-Winkel:

$$\gamma = \text{atan2}(r_{32}, r_{33})$$

$$\beta = \text{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right)$$

$$\alpha = \text{atan2}(r_{21}, r_{11})$$

(Singularität bei $\beta = \pm 90^\circ$)



$$\begin{bmatrix} \dot{\gamma} \\ \dot{\beta} \\ \dot{\alpha} \end{bmatrix} = J_R \omega = \begin{bmatrix} \frac{\cos \alpha}{\cos \beta} & \frac{\sin \alpha}{\cos \beta} & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ \tan \beta \cos \alpha & \tan \beta \sin \alpha & 1 \end{bmatrix} \omega$$

Beispiel Quaternionen (Euler Parameter):

$$\lambda = [\lambda_0, \lambda_1, \lambda_2, \lambda_3] = \left[\cos \frac{\theta}{2}, \vec{k} \sin \frac{\theta}{2} \right]$$

$$\lambda_0 = \frac{1}{2} \sqrt{r_{11} + r_{22} + r_{33} + 1}$$

$$\lambda_1 = \frac{1}{2} \text{sgn}(r_{32} - r_{23}) \sqrt{r_{11} - r_{22} - r_{33} + 1}$$

$$\lambda_2 = \frac{1}{2} \text{sgn}(r_{13} - r_{31}) \sqrt{-r_{11} + r_{22} - r_{33} + 1}$$

$$\lambda_3 = \frac{1}{2} \text{sgn}(r_{21} - r_{12}) \sqrt{-r_{11} - r_{22} + r_{33} + 1}$$

$$J_R = \frac{1}{2} \begin{bmatrix} -\lambda_1 & -\lambda_2 & -\lambda_3 \\ \lambda_0 & \lambda_3 & -\lambda_2 \\ -\lambda_3 & \lambda_0 & \lambda_1 \\ \lambda_2 & -\lambda_1 & \lambda_0 \end{bmatrix}$$

Entkoppelte Positionsregelung in Task-Koordinaten

(Feedback Linearisierung oder I/O Linearisierung oder Computed Torque Controller)

Robotermodell in $SE3$ Koordinaten x :

$$M_K(q)\ddot{x} + c_K(q, \dot{q}) + g_K(q) = F$$

Regler

$$F = M_K(q)[\ddot{x}_d + K(x_d - x) + D(\dot{x}_d - \dot{x})] + c_K(q, \dot{q}) + g_K(q)$$

$$\tau = J^T(q)F$$

Kommandierung eines Gelenkdrehmoments erfordert:

- Direktantriebe oder
- Gelenkdrehmomentregelung

Es folgt eine lineare, entkoppelte Fehlerdynamik:

$$\ddot{e} + D\dot{e} + Ke = 0$$

$$e = x_d - x$$

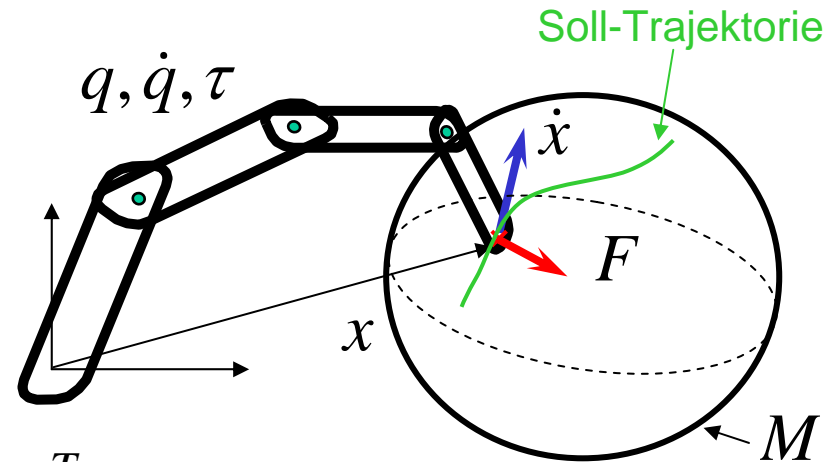
Drehmomentschnittstelle



Hybride Kraft-/Positionsregelung in Task-Koordinaten

Der Roboter soll entlang einer Trajektorie auf eine Mannigfaltigkeit M (Objektoberfläche) fahren und eine vorgegebene Kraft auf M ausüben!

wenn x_M : r -dim. lokale Koordinaten für M , wollen wir x_M bzw. \dot{x}_M regeln, sowie $m-r$ Wrenches F_R , die reziprok zu \dot{x}_M sind, d.h. $F_R^T \dot{x}_M = 0$



Wir wählen somit als Geschwindigkeitsvektor $\dot{x} = (\dot{x}_M, \dot{x}_R)$ dual zu F_R

Robotermodell in $SE3$ Koordinaten x :

$$M_K(q)\ddot{x} + c_K(q, \dot{q}) + g_K(q) = F - F_{R_{ext}} \quad \leftarrow \text{Kontaktkraft}$$

Der Regler F hat dann zwei Komponenten:

$$\begin{array}{l} F_M \text{ - für Positionsregelung} \\ F_R \text{ - für Kraftregelung} \end{array} \quad F_C = \begin{bmatrix} F_M \\ F_R \end{bmatrix}$$

Hybride Kraft-/Positionsregelung in Task-Koordinaten

$$M_K(q)\ddot{x} + c_K(q, \dot{q}) + g_K(q) = F - F_{Rext}$$

Regler:

$$F = M_K(q)F_C + c_K(q, \dot{q}) + g_K(q) + (I - M_K(q))F_{Rext}$$

$$\tau = J^T(q)F$$

zusätzliche Rückkopplung
der externen Kraft

Durch einsetzen, resultiert die Fehlerdynamik:

$$\ddot{x} = F_C - F_{Rext}$$

oder aufgetrennt in Koordinaten für Positions- und Kraftregelung:

$$\ddot{x}_M = F_M$$

$$\ddot{x}_R = F_R - F_{Rext}$$

Hybride Kraft-/Positionsregelung in Task-Koordinaten

$$\ddot{x}_M = F_M$$

$$\ddot{x}_R = F_R - F_{R\text{ext}}$$

- 1) Entkopplungsregler für Position (wie gehabt)

$$F_M = \ddot{x}_{Md} + D(\dot{x}_{Md} - \dot{x}_M) + K(x_{Md} - x_M)$$

Es folgt eine lineare, entkoppelte Fehlerdynamik:

$$\ddot{e}_M + D\dot{e}_M + Ke_M = 0 \quad e_M = x_{Md} - x_M$$

- 2) Kraftregler

$$F_R = F_{Rd} + K_F(\dot{F}_{Rd} - \dot{F}_{R\text{ext}})$$

Es folgt die Fehlerdynamik:

$$\ddot{x}_R = K_F \dot{e}_F + e_F \quad e_F = F_{Rd} - F_{R\text{ext}}$$

$$\ddot{x}_R = 0 \quad \text{mit idealem starren Kontakt}$$

Redundante Roboter

Aufgaben-Raum (Task Space): m dimensional $m \leq 6$

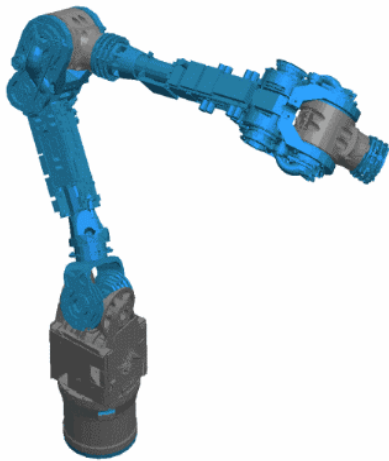
Gelenk-Raum Q : n dimensional

redundanter Manipulator: $n > m$

$n-m$: Redundanzgrad

$n=10, m=6$

$n=7, m=6$



Nullraumbewegung für Justin



Redundante Roboter: Dynamik

Für $m < n$ sind die Koordinaten x nicht ausreichend, um den Roboter komplett zu beschreiben. Allerdings kann das dynamische Verhalten des Endeffektors dadurch komplett beschrieben werden.

Für nichtredundante Roboter hatten wir:

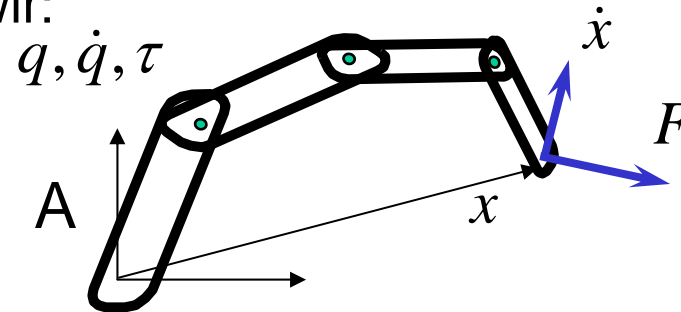
$$M_K(q)\ddot{x} + c_K(q, \dot{q}) + g_K(q) = F$$

$$\dot{x} = J(q)\dot{q};$$

$$\dot{q} = J^{-1}(q)\dot{x}$$

$$F = J^{-T}(q)\tau; \quad \tau = J^T(q)F$$

$$M_K(x) = J^{-T}(q)M(q)J^{-1}(q);$$



Für redundante Roboter ist $J_{m \times n}$ nicht quadratisch, also nicht invertierbar!

Lösung: pseudoinverse Matrix:

$$J_{m \times n} J_{n \times m}^{\#} = I_{m \times m}$$

$$(J_{m \times n}^{\#T} J_{n \times m}^T = I_{m \times m})$$

man bemerke, dass $J_{n \times m}^{\#} J_{m \times n} \neq I_{n \times n}$

$$(J_{n \times m}^T J_{m \times n}^{\#T} \neq I_{n \times n})$$

mit

$$J^{\#} = A J^T (J A J^T)^{-1}$$

A-p.d. matrix

A=I Sonderfall:
Moore-Penrose
Pseudoinverse
 $J^{\#} = J_{n \times m}^T (J J^T)^{-1}$

Redundante Roboter: Dynamik

dann ist:

nicht –singulärer Fall!

$$\dot{q} = J^\#(q)\dot{x} \quad F = J^{\#T}(q)\tau$$

$$M_R(x) = J^{\#T}(q)M(q)J^\#(q)$$

und wieder

$$M_R(q)\ddot{x} + c_R(q, \dot{q}) + g_R(q) = F$$

Aber: Wie wählt man die Matrix A richtig?

(Unterschiedliche A führen zu unterschiedlichen Ergebnissen für M_R !?!)

Alles, was man mit einer Pseudoinversen erreichen kann, kann man auch ohne lösen!

Ansatz: um M_R zu berechnen, berechnet man zuerst die Inverse M_R^{-1}

M_R^{-1} ist ein Tensor vom Typ (0,2) –zweimal kontravariant

Mobilitätsmatrix

transformiert also wie folgt:

$$M_R^{-1} = JM^{-1}J^T$$

folglich

$$M_R = (JM^{-1}J^T)^{-1}$$

weil $p^T M_R^{-1} p = E_{kin}$

$p = M_R \dot{x}$ Impuls,
kovariant