Motivation: Wie kann die Geschwindigkeit des TCP berechnet werden?

Angenommen, es gibt eine Trajektorie für die Gelenkwinkel, dann kann zu jedem Zeitpunkt t über die Vorwärtskinematik die Lage/Pose des TCP p(t) berechnet werden.

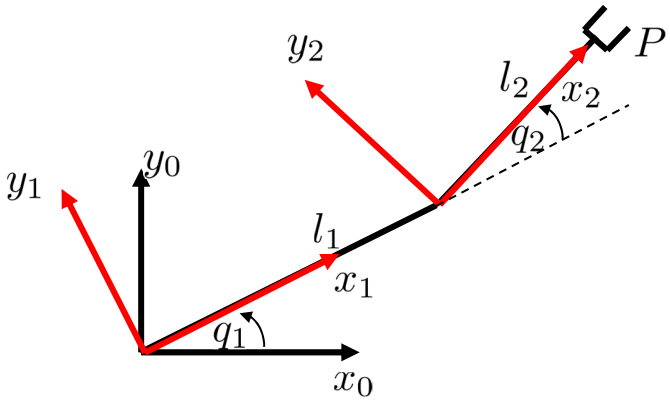
Geschwindigkeit mittels Differenzenquotient

$$\lim_{\Delta t \to 0} = \frac{p(t+\Delta t) - p(t))}{\Delta t} = \dot{p}(t)$$

Geschwindigkeit ist die Ableitung der Pose nach der Zeit

→ Formel für Pose erforderlich

Beispiel 2D RR



$${}^0T_2 = {}^0T_1\,{}^1T_2 = \begin{bmatrix} \cos(q_1+q_2) & -\sin(q_1+q_2) & \cos q_1 l_1 \\ \sin(q_1+q_2) & \cos(q_1+q_2) & \sin q_1 l_1 \\ 0 & 0 & 1 \end{bmatrix}$$

Punkt P im Koordinatensystem {0}

$${}^0\tilde{p} = {}^0T_2\,{}^2\tilde{p} = {}^0T_2 \begin{bmatrix} l_2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(q_1+q_2)l_2 + \cos q_1 l_1 \\ \sin(q_1+q_2)l_2 + \sin q_1 l_1 \\ 1 \end{bmatrix}$$

Punkt P im Koordinatensystem {0}

$$ {}^0p = \begin{bmatrix} \cos(q_1+q_2)l_2 + \cos q_1 l_1 \\ \sin(q_1+q_2)l_2 + \sin q_1 l_1 \end{bmatrix} = \begin{bmatrix} c_{12}l_2 + c_1 l_1 \\ s_{12}l_2 + s_1 l_1 \end{bmatrix} $$

Gelenkwinkel sind zeitabhängig, d.h.

$$ q_1 = q_1(t) \qquad q_2 = q_2(t) $$

Zeitableitung

$$ {}^0\dot{p} = \begin{bmatrix} -s_{12}\dot{q_1}l_2 - s_1\dot{q_1}l_1 - s_{12}\dot{q_2}l_2 \\ c_{12}\dot{q_1}l_2 + c_1\dot{q_1}l_1 + c_{12}\dot{q_2}l_2 \end{bmatrix} $$

Matrix-Vektor-Schreibweise

$$ {}^0\dot{p} = \begin{bmatrix} -s_{12}l_2 - s_1 l_1 & -s_{12}l_2 \\ c_{12}l_2 + c_1 l_1 & c_{12}l_2 \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} $$

$$ {}^0\dot{p} = \begin{bmatrix} -s_{12}l_2 - s_1 l_1 & -s_{12}l_2 \\ c_{12}l_2 + c_1 l_1 & c_{12}l_2 \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} $$

→ TCP-Geschwindigkeit = (Jacobi-)Matrix mal Gelenkwinkelgeschwindigkeit

→ Matrix ist von der Stellung abhängig

z.B. gestreckte Lage $q_1 = 0, q_2 = 0$

$$ {}^0\dot{p} = \begin{bmatrix} 0 & 0 \\ l_2 + l_1 & l_2 \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} $$

wenn q1abl 0 ist der Hebelarm l2+l1

Anteil von Gelenk 1: $(l_2 + l_1)\dot{q_1} = \omega_{0,1} r_{0,P}$ r0p Heblearm

Anteil von Gelenk 2: $l_2\dot{q_2} = \omega_{1,2} r_{1,P}$

Winkelgeschwindigkeit im Punkt P

$$\omega = \dot{q}_1 + \dot{q}_2 = \omega_{0,1} + \omega_{1,2}$$

Verallgemeinerung

1. Formeln für Vorwärtskinematik aufstellen
2. Zeitableitung

Beispiel UR Roboter

$$^0T_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

mit Formeln für die einzelnen Matrixeinträgen

$$
\begin{aligned}
n_x &= c_6(s_1s_5 + ((c_1c_{234} - s_1s_{234})c_5)/2.0 + ((c_1c_{234} + s_1s_{234})c_5)/2.0) - (s_6((s_1c_{234} + c_1s_{234}) - (s_1c_{2}\\
&\quad c_1s_{234})))/2.0\\
n_y &= c_6(((s_1c_{234} + c_1s_{234})c_5)/2.0 - c_1s_5 + ((s_1c_{234} - c_1s_{234})c_5)/2.0) + s_6((c_1c_{234} - s_1s_{234})/2.0 - (c_1c_{234} +\\
&\quad s_1s_{234})/2.0)\\
n_z &= (s_{234}c_6 + c_{234}s_6)/2.0 + s_{234}c_5c_6 - (s_{234}c_6 - c_{234}s_6)/2.0
\end{aligned}
$$

$$
\begin{aligned}
o_x &= -(c_6((s_1c_{234} + c_1s_{234}) - (s_1c_{234} - c_1s_{234})))/2.0 - s_6(s_1s_5 + ((c_1c_{234} - s_1s_{234})c_5)/2.0 + ((c_1c_{234} +\\
&\quad s_1s_{234})c_5)/2.0)\\
o_y &= c_6((c_1c_{234} - s_1s_{234})/2.0 - (c_1c_{234} + s_1s_{234})/2.0) - s_6(((s_1c_{234} + c_1s_{234})c_5)/2.0 - c_1s_5 + ((s_1c_{234} -\\
&\quad c_1s_{234})c_5)/2.0)\\
o_z &= (c_{234}c_6 + s_{234}s_6)/2.0 + (c_{234}c_6 - s_{234}s_6)/2.0 - s_{234}c_5s_6
\end{aligned}
$$

$$
\begin{aligned}
a_x &= c_5s_1 - ((c_1c_{234} - s_1s_{234})s_5)/2.0 - ((c_1c_{234} + s_1s_{234})s_5)/2.0\\
a_y &= -c_1c_5 - ((s_1c_{234} + c_1s_{234})s_5)/2.0 + ((c_1s_{234} - s_1c_{234})s_5)/2.0\\
a_z &= (c_{234}c_5 - s_{234}s_5)/2.0 - (c_{234}c_5 + s_{234}s_5)/2.0
\end{aligned}
$$

$$
\begin{aligned}
p_x &= -(d_5(s_1c_{234} - c_1s_{234}))/2.0 + (d_5(s_1c_{234} + c_1s_{234}))/2.0 + d_4s_1 - (d_6(c_1c_{234} - s_1s_{234})s_5)/2.0 - (d_6(c_1c_{234} +\\
&\quad s_1s_{234})s_5)/2.0 + a_2c_1c_2 + d_6c_5s_1 + a_3c_1c_2c_3 - a_3c_1s_2s_3)\\
p_y &= -(d_5(c_1c_{234} - s_1s_{234}))/2.0 + (d_5(c_1c_{234} + s_1s_{234}))/2.0 - d_4c_1 - (d_6(s_1c_{234} + c_1s_{234})s_5)/2.0 - (d_6(s_1c_{234} -\\
&\quad c_1s_{234})s_5)/2.0 - d_6c_1c_5 + a_2c_2s_1 + a_3c_2c_3s_1 - a_3s_1s_2s_3)\\
p_z &= d_1 + (d_6(c_{234}c_5 - s_{234}s_5))/2.0 + a_3(s_2c_3 + c_2s_3) + a_2s_2 - (d_6(c_{234}c_5 + s_{234}s_5))/2.0 - d_5c_{234}
\end{aligned}
$$

Berechnung der Zeitableitung sehr aufwendig

→ Aufteilung der kinematischen Kette und gelenkweise Berechnung

Nützliche Formeln

Kinematische Kette

$$\omega_{0,i} = \omega_{0,i-1} + \omega_{i-1,i}$$

$$\dot{p}_i = \dot{p}_{i-1} + \omega_{0,i} \times r_{i-1,i}$$

zi-1 =Vektor (0,0,1)

Winkelgeschwindigkeit einer DH-Transformation

Standard DH (klassisch) (Rotation um die z i-1 Achse) $\omega_{i-1,i} = \dot{q}_i z_{i-1}$

DH (modifiziert) (Rotation um die z i Achse) $\omega_{i-1,i} = \dot{q}_i z_i$

Geometrische Jacobimatrix (mit Winkelgeschwindigkeit ohne Eulerwinkel)

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = J\dot{q} = \begin{bmatrix} J_{trans} \\ J_{rot} \end{bmatrix} \dot{q}$$

v und omega 3 Werte

Beitrag eines Gelenks zur Winkelgeschwindigkeit

$$\omega = \omega_{0,6} = \omega_{0,1} + \omega_{1,2} + ... + \omega_{5,6}$$

$$\omega = \dot{q}_1 z_0 + \dot{q}_2 z_1 + ... + \dot{q}_6 z_5$$

DH klassisch

$$\omega = \begin{bmatrix} z_0 & \dots & z_5 \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_6 \end{bmatrix} = J_{rot}\dot{q}$$

Geometrische Jacobimatrix (mit Winkelgeschwindigkeit ohne Eulerwinkel)

Translationsgeschwindigkeit des Tools $v = \dot{p}_6$

Herleitung mit $\dot{p}_i = \dot{p}_{i-1} + \omega_{0,i} \times r_{i-1,i}$

$$\dot{p}_1 = \dot{p}_0 + \omega_{0,1} \times r_{0,1} = \omega_{0,1} \times r_{0,1} = (z_0 \times r_{0,1})\dot{q}_1$$

DH klassisch

$$\dot{p}_2 = \dot{p}_1 + \omega_{0,2} \times r_{1,2} = \omega_{0,1} \times r_{0,1} + \omega_{0,1} \times r_{1,2} + \omega_{1,2} \times r_{0,1}$$
$$= \omega_{0,1} \times r_{0,2} + \omega_{1,2} \times r_{1,2} = (z_0 \times r_{0,2})\dot{q}_1 + (z_1 \times r_{1,2})\dot{q}_2$$

...

$$v = \dot{p}_6 = \begin{bmatrix} z_0 \times r_{0,6} & \dots & z_5 \times r_{5,6} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_6 \end{bmatrix} = J_{trans}\dot{q}$$

Geometrische Jacobimatrix (mit Winkelgeschwindigkeit ohne Eulerwinkel)

$$J = \begin{bmatrix} z_0 \times r_{0,6} & \dots & z_5 \times r_{5,6} \\ z_0 & \dots & z_5 \end{bmatrix}$$

DH klassisch

$$J = \begin{bmatrix} z_1 \times r_{1,6} & \dots & z_6 \times r_{6,6} \\ z_1 & \dots & z_6 \end{bmatrix}$$

DH modifiziert

Implementierung

$$ {}^0T_i = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{matrix} & & z_i & p_i \end{matrix}$$

$$r_{i,6} = p_6 - p_i$$

Pseudocode

```
T_0_6 = forward_kinematics(dh_para, q)
p = T_0_6[0:3, 3]

for i from 0 to 5:
        T = denavit_hartenberg(dh_para[i], q[i])
        T_0_i = np.dot(T_0_i, T)
        z_i = T_0_i[0:3, 2]
        p_i = T_0_i[0:3, 3]
        r = p - p_i
        J[0:3, i] = np.cross(z_i, r)
        J[3:6, i] = z_i
```

Anwendung Jacobimatrix

- Berechnung der TCP-Geschwindigkeit

- Berechnung der Gelenkgeschwindigkeit (Inverse Jacobimatrix)

- Berechnung von Singularitäten (Determinante Jacobimatrix)

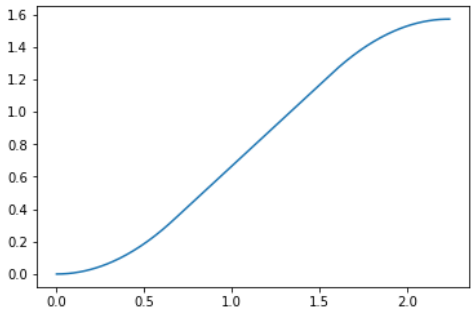
- Berechnung von Manipulierbarkeitsmaßen (Betrag der Determinante)

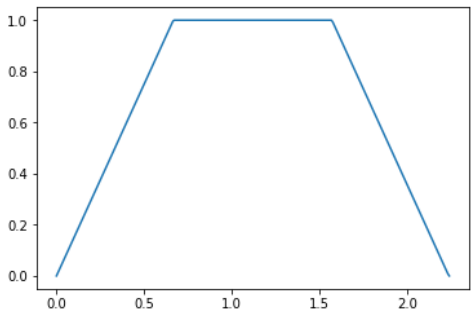
- Berechnung von statischen Kräften/Momenten (Transponierte Jacobimatrix)

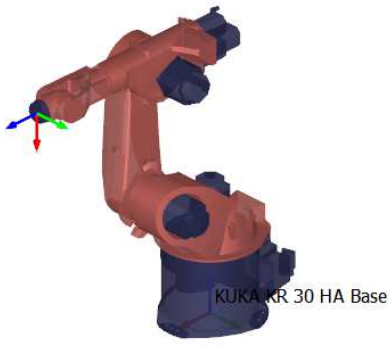
- Redundante Roboter (Pseudo Inverse)

Beispiel: Berechnung der TCP-Geschwindigkeit

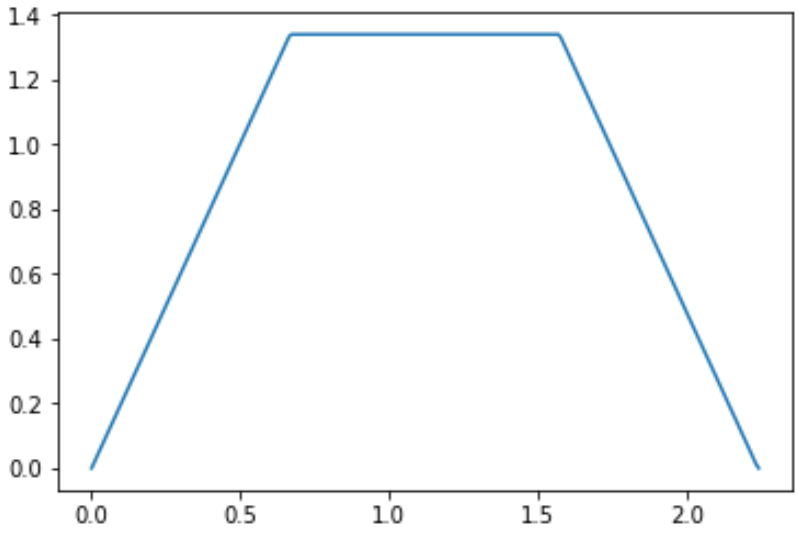
Gegeben Trajektorie für Gelenk 1



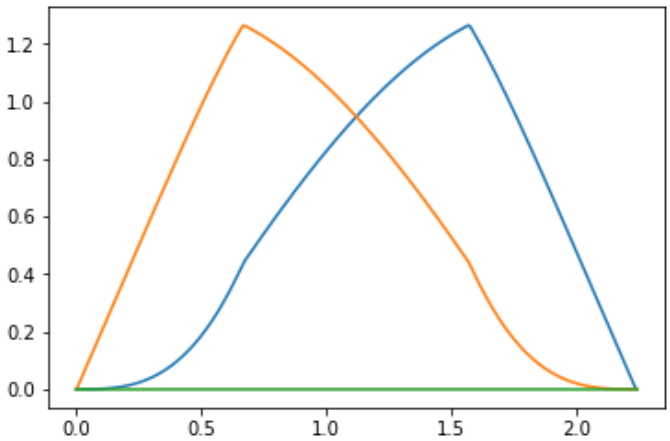




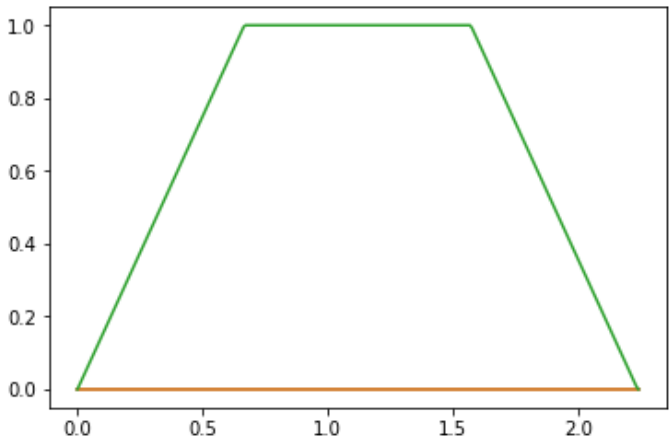
```
fo (q_i, qd_i) in zip(traj[1], traj[2]):
q = np.array([q_i , -np.pi/2, np.pi/2 , 0, 0, 0])
qd = np.array([qd_i , 0., 0, 0., 0., 0.])
J = r.jacobi_kuka(kr30_dhm, q)
v = J.dot(qd)
V_abs.append(np.linalg.norm(v[0:2]))
```



Translationgeschw.



Rotationgeschw.

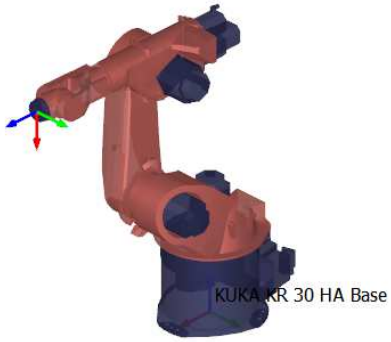


Beispiel: Berechnung der statischen Kräfte/Momente

Gegeben Kraft / Moment am TCP

```
q = np.array([0 , -np.pi/2, np.pi/2 , 0, 0, 0])
J = r.jacobi_kuka(kr30_dhm, q)
JT = np.transpose(J)
F = np.array([0, -10, 0, 0, 0, 0])
t = np.dot(JT, F)
print(t)
```

F(x,y,z,...) Koordinaten




```
Ausgabe:
[-1.34000000e+01 -6.06200166e-16 -6.06200166e-16
1.04094978e-16 -1.04094978e-16  0.00000000e+00]

F = np.array([0, 0, 10, 0, 0, 0])
t = np.dot(JT, F)
print(t)

Ausgabe:
[ 0.00000000e+00 -9.90000000e+00 -9.90000000e+00
4.62223187e-32 -1.70000000e+00  0.00000000e+00]
```

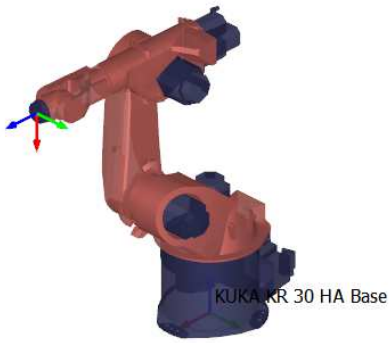

Beispiel: Berechnung von Singularitäten

Gegeben Gelenkwinkel

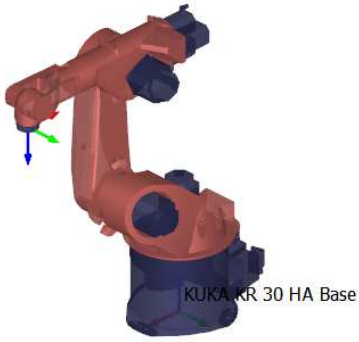
```
q = np.array([0 , -np.pi/2, np.pi/2 , 0, 0, 0])
J = r.jacobi_kuka(kr30_dhm, q)
d = np.linalg.det(J)
print(d)

Ausgabe:
-7.2554199615485e-18
```

v=J*qabl-->qabl=Jinvers*v

Jacobi matrix gegen 0
Geschwindigkeit gegen unendlich



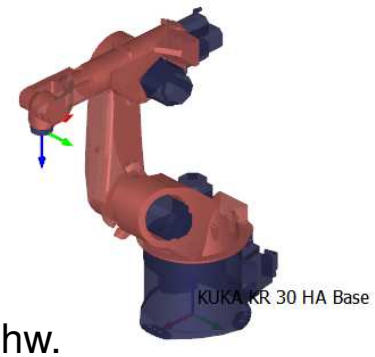

```
q = np.array([0 , -np.pi/2, np.pi/2 , 0, np.pi/2, 0])
J = r.jacobi_kuka(kr30_dhm, q)
d = np.linalg.det(J)
print(d)

Ausgabe:
-0.8154899999999999
```



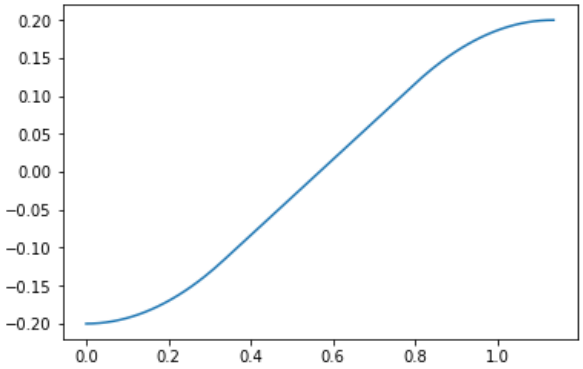
Beispiel: Lineare Bewegung (in der Nähe einer Singularität)

Gegeben TCP (Position und Geschwindigkeit)

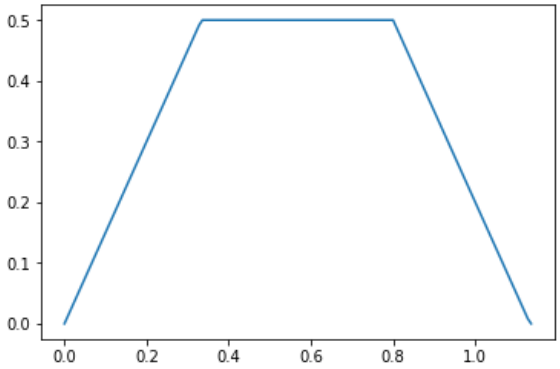
Ausgangsstellung in Abhängigkeit von q5
Lineare Bewegung von y = -200 mm zu y = +200 mm



Y-Pos.



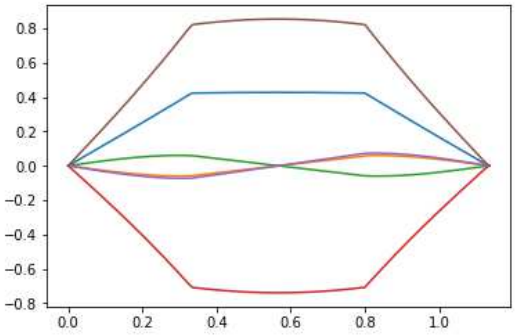
Y-Geschw.



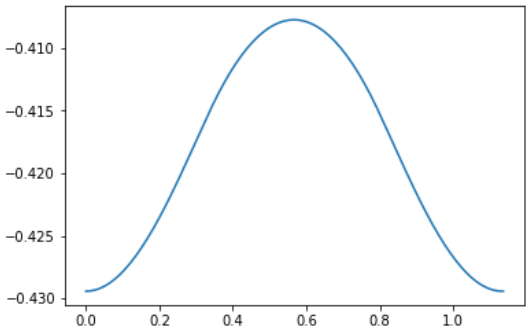

```
for (r_i, rd_i) in zip(traj[1], traj[2]):
    pos = p_0
    pos[1] = r_i
    rd = [0,rd_i,0,0,0,0]
    q = r.ik_kuka(kr30_dhm, pos, 0)

    J = r.jacobi_kuka(kr30_dhm, q)
    Jinv = np.linalg.inv(J)
    qd = Jinv.dot(rd)
    QD.append(qd)

    d = np.linalg.det(J)
    D.append(d)
```

q5 = 30

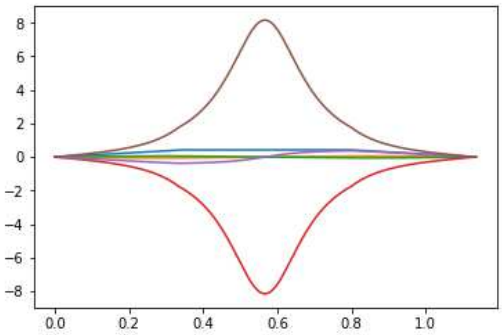
Gelenkgeschw.

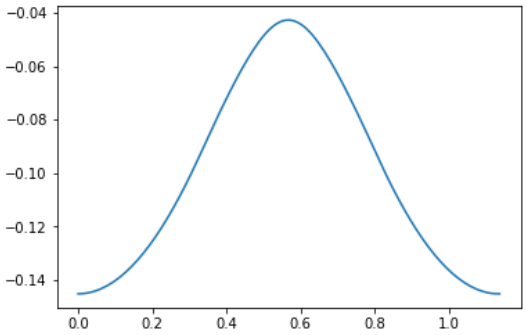


Jacobideterminante



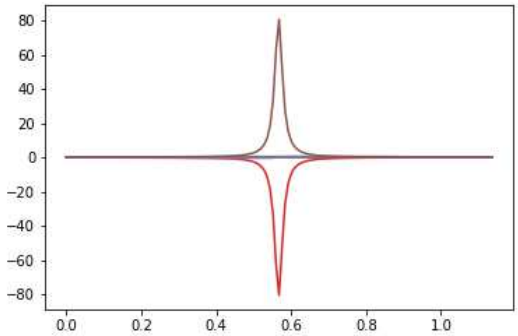
q5 = 3



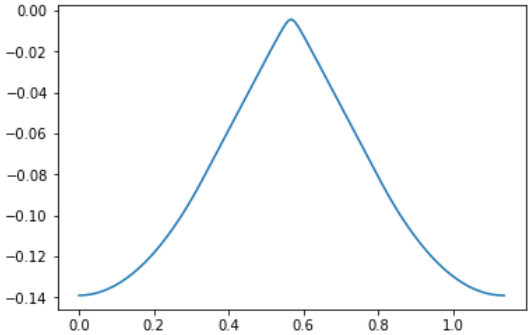


q5 = 0.3

Gelenkgeschw.



Jacobideterminante

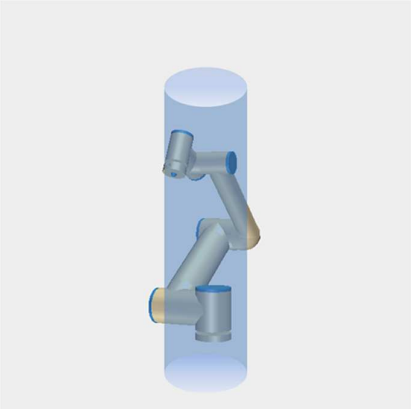


Singularitäten bei Universal Robots

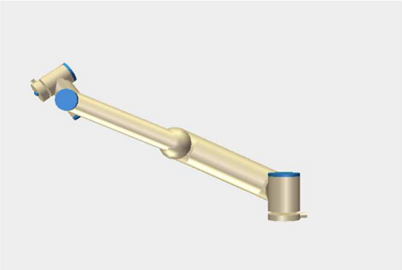
(FAQ Robot singularity – 28788)

"It is natural that 6-axis robot arms have singularity problems. However, the robot poses to cause singularity issues tend to be dependent on mechanical design. There are three known singular poses in UR robots.

First, the linear movement close to the cylindrical area directly above and below the robot base is restricted. It causes the joints to move fast even though the tool is moving slowly."



"The second singular area is around the workspace boundary. When the robot is stretched out, it would not be possible to stretch the robot more or to change orientation by keeping the tool position."



"Finally, if the wrist 2 joint is 0 or 180 degree, 4 axes are co-aligned with the same direction then the robot lost one or more degree of freedom in the certain direction."