KIT-Fakultät für Informatik
Institut für Anthropomatik und Robotik (IAR)
Hochperformante Humanoide Technologien (H^2T)

Robotik I im WS 2017/18

Musterlösungen zum 3. Übungsblatt

Prof. Dr.-Ing. Tamim Asfour
Dipl.-Inform. Peter Kaiser
M.Sc. Fabian Paus
M.Sc. Jonas Beil
Adenauerring 2, Geb. 50.20
Web: http://h2t.anthropomatik.kit.edu

Lösung 1 (Differentielle Inverse Kinematik)

1. Berechnung der Jacobi-Matrix

   Die Funktion für die Position des Endeffektors ist gegeben als

$$f(\boldsymbol{q}) = \begin{pmatrix} -500sin(\theta_2)cos(\theta_3) - 500cos(\theta_2)sin(\theta_3) - 500sin(\theta_2) \\ 500cos(\theta_2)cos(\theta_3) - 500sin(\theta_2)sin(\theta_3) + 100 + 500cos(\theta_2) \\ d_1 \end{pmatrix}$$

   Die Jacobi-Matrix berechnet sich als

$$J(\boldsymbol{q}) = \frac{\partial f}{\partial \boldsymbol{q}} = \begin{pmatrix} \frac{\partial f}{\partial d_1} & \frac{\partial f}{\partial \theta_2} & \frac{\partial f}{\partial \theta_3} \end{pmatrix} = \begin{pmatrix} 0 & -500cos(\theta_2+\theta_3) - 500cos(\theta_2) & -500cos(\theta_2+\theta_3) \\ 0 & -500sin(\theta_2+\theta_3) - 500sin(\theta_2) & -500sin(\theta_2+\theta_3) \\ 1 & 0 & 0 \end{pmatrix}$$

   Invertieren der Jacobi-Matrix ergibt

$$J^{-1}(\boldsymbol{q}) = \begin{pmatrix} 0 & 0 & 1 \\ -\frac{1}{500}\frac{sin(\theta_2+\theta_3)}{sin(\theta_3)} & \frac{1}{500}\frac{cos(\theta_2+\theta_3)}{sin(\theta_3)} & 0 \\ \frac{1}{500}\frac{sin(\theta_2+\theta_3)+sin(\theta_2)}{sin(\theta_3)} & \frac{1}{500}\frac{-cos(\theta_2+\theta_3)-cos(\theta_2)}{sin(\theta_3)} & 0 \end{pmatrix}$$

2. Gelenkwinkelgeschwindigkeit

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

$$\dot{\boldsymbol{q}} = J^{-1}((1, 0, \frac{\pi}{2})^T) \cdot (1000, 0, 0)^T = \begin{pmatrix} 0 & 0 & 1 \\ -\frac{1}{500} & 0 & 0 \\ \frac{1}{500} & -\frac{1}{500} & 0 \end{pmatrix} \cdot \begin{pmatrix} 1000 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ -2 \\ 2 \end{pmatrix}$$

3. Singularitäten

   Für quadratische Matrizen könne wir Singularitäten bestimmen, indem wir die Determinante gleich Null setzen. In dem Fall hat die Matrix keinen vollen Rang mehr.

$$\det J(\boldsymbol{q}) = 500^2 \cdot \sin\theta_3 = 0$$

   Die Jacobi-Matrix ist singulär für $\theta_3 = 0°, 180°$. Die inverse Matrix existiert für diese Gelenkwinkelstellungen nicht.

Lösung 2 (Dynamikmodellierung nach Lagrange)

Für unseren Roboter gelten folgende vereinfachende Annahmen:

1. m_1 und m_2 sind als Punktmassen in der Mitte der entsprechenden Armsegmente definiert,
2. in dem System gibt es keine Reibung,
3. der Roboter ist auf einer Höhe von $h = 0$ installiert.

Die Bewegungsgleichung kann nach der Methode von Lagrange entsprechend folgender Formel ermittelt werden:

$$\tau = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L}{\partial \boldsymbol{q}}$$

mit der Lagrange-Funktion:

$$L = E_{kin} - E_{pot}.$$

1. Kinetische Energie für s_1 und s_2

   Für s_1 ergibt sich aus der Translationsgeschwindigkeit $\dot{d}$ folgende kinetische Energie:

   $$E_{kin,1} = \frac{1}{2} m_1 \dot{d}^2.$$

   Für s_2 ergibt sich aus der Geschwindigkeit $(\dot{x}_2, \dot{y}_2)$ die folgende kinetische Energie:

   $$E_{kin,2} = \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_2 \left(\dot{x}_2^2 + \dot{y}_2^2\right). \quad (1)$$

   Die Geschwindigkeiten $\dot{x}_2$ und $\dot{y}_2$ ergeben sich aus x_2 und y_2 durch Ableiten nach der Zeit:

   $$\begin{aligned} \dot{x}_2 &= \dot{d} - \frac{1}{2} l_2 \dot{\theta} sin(\theta), \\ \dot{y}_2 &= \frac{1}{2} l_2 \dot{\theta} cos(\theta). \end{aligned}$$

   Eingesetzt in Gleichung 1 ergibt sich die kinetische Energie für das zweite Armsegment:

   $$E_{kin,2} = \frac{1}{2} m_2 \dot{d}^2 + \frac{1}{8} m_2 l_2^2 \dot{\theta}^2 - \frac{1}{2} m_2 l_2 \dot{d} \dot{\theta} sin(\theta).$$

2. Potentielle Energie für s_1 und s_2

   Für den allgemeinen Fall, dass sich der der Ursprung des Roboters auf einer Höhe von $h > 0$ befindet, kann die potentielle Energie mit dem Gravitationsvektor g für s_1 wie folgt beschrieben werden:

   $$E_{pot,1} = m_1 g h.$$

   Entsprechend gilt für s_2

   $$E_{pot,2} = m_2 g \left(h - \frac{1}{2} l_2 sin(\theta) \right).$$

   Aus $h = 0$ folgt:

   $$\begin{aligned} E_{pot,1} &= 0 \\ E_{pot,2} &= -\frac{1}{2} m_2 g l_2 sin(\theta). \end{aligned}$$

3. Lagrange-Funktion und deren Ableitungen

   Die Lagrange-Funktion setzt sich aus der kinetischen und potentiellen Energie für s_1 und s_2 entsprechend folgender Regel zusammen:

   $$L = E_{kin,1} + E_{kin,2} - E_{pot,1} - E_{pot,2}.$$

   L wird mittels vorheriger Ergebnisse wie folgt beschrieben:

   $$L = \frac{1}{2}(m_1 + m_2)\dot{d}^2 + \frac{1}{2}\frac{1}{4} m_2 l_2^2 \dot{\theta}^2 - \frac{1}{2} m_2 l_2 \dot{d}\dot{\theta} sin(\theta) + \frac{1}{2} m_2 g l_2 sin(\theta).$$

Die Bewegungsgleichung setzt sich aus den beiden Komponenten τ_1 und τ_2 zusammen, zu deren Bestimmung die folgenden Ableitungen bestimmt werden müssen:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{d}}, \frac{\partial L}{\partial d}, \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}}, \frac{\partial L}{\partial \theta}.$$

Die Ableitungen von L für das Translationgelenk mit dem Stellsignal d sind wie folgt definiert:

$$\begin{aligned} \frac{\partial L}{\partial \dot{d}} &= (m_1 + m_2)\dot{d} - \frac{1}{2} m_2 l_2 \dot{\theta} sin(\theta) \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{d}} &= (m_1 + m_2)\ddot{d} - \frac{1}{2} m_2 l_2 (\ddot{\theta} sin(\theta) + \dot{\theta}^2 cos(\theta)) \\ \frac{\partial L}{\partial d} &= 0. \end{aligned}$$

Die Ableitungen von L für das Rotationsgelenk mit dem Stellsignal θ sind wie folgt definiert:

$$\begin{aligned}
\frac{\partial L}{\partial \dot{\theta}} &= \frac{1}{4} m_2 l_2^2 \dot{\theta} - \frac{1}{2} m_2 l_2 \dot{d} sin(\theta) \\
\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} &= \frac{1}{4} m_2 l_2^2 \ddot{\theta} - \frac{1}{2} m_2 l_2 (\ddot{d} sin(\theta) + \dot{d}\dot{\theta} cos(\theta)) \\
\frac{\partial L}{\partial \theta} &= -\frac{1}{2} m_2 l_2 \dot{d}\dot{\theta} cos(\theta) + \frac{1}{2} m_2 l_2 g cos(\theta).
\end{aligned}$$

Daraus ergibt sich für τ_1 und τ_2:

$$\begin{aligned}
\tau_1 &= \frac{d}{dt}\frac{\partial L}{\partial \dot{d}} - \frac{\partial L}{\partial d} \\
&= (m_1 + m_2)\ddot{d} - \frac{1}{2} m_2 l_2 sin(\theta)\ddot{\theta} - \frac{1}{2} m_2 l_2 cos(\theta)\dot{\theta}^2 - 0 \\
\tau_2 &= \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} \\
&= \frac{1}{4} m_2 l_2^2 \ddot{\theta} - \frac{1}{2} m_2 l_2 \ddot{d} sin(\theta) - \frac{1}{2} m_2 l_2 \dot{d}\dot{\theta} cos(\theta) - (-\frac{1}{2} m_2 l_2 \dot{d}\dot{\theta} cos(\theta) + \frac{1}{2} m_2 l_2 g cos(\theta)) \\
&= \frac{1}{4} m_2 l_2^2 \ddot{\theta} - \frac{1}{2} m_2 l_2 sin(\theta)\ddot{d} - \frac{1}{2} m_2 l_2 g cos(\theta).
\end{aligned}$$

Die allgemeine Form der Bewegungsgleichung unter Vernachlässigung von Reibung lautet:

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

Die Massenträgheitsmatrix M erhält man durch das Zusammenfassen aller Terme die von $\ddot{\boldsymbol{q}}$ abhängen. Im Vektor $\boldsymbol{c}$ (Zentripetal- und Corioliskomponenten) werden alle Terme zusammengefasst, die sowohl von $\dot{\boldsymbol{q}}$ und $\boldsymbol{q}$ abhängen. Im Vektor $\boldsymbol{g}$ werden alle Terme zusammengefasst, in denen Gravitationskomponenten enthalten sind

Damit lässt sich die Bewegungsgleichung wie folgt beschreiben:

$$\boldsymbol{\tau} = \begin{pmatrix} \tau_1 \\ \tau_2 \end{pmatrix} = \begin{pmatrix} m_1 + m_2 & -\frac{1}{2} m_2 l_2 sin(\theta) \\ -\frac{1}{2} m_2 l_2 sin(\theta) & \frac{1}{4} m_2 l_2^2 \end{pmatrix} \begin{pmatrix} \ddot{d} \\ \ddot{\theta} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} m_2 l_2 \dot{\theta}^2 \cdot cos(\theta) \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -\frac{1}{2} m_2 l_2 g \cdot cos(\theta) \end{pmatrix}.$$

Hinweis: Die generalisierten Kräfte τ_1 und τ_2 haben unterschiedliche Einheiten. τ_1 ist eine Kraft (Einheit: Newton), während τ_2 ein Drehmoment ist (Einheit: Newtonmeter).

Lösung 3 (Matlab Einführung)

Die Aufgaben 1 bis 5 des ersten Übungsblattes werden im Folgenden im interaktiven *Command Window* von Matlab gelöst. Aus Gründen der Übersichtlichkeit sind die Matlab-Ausgaben zum Teil gekürzt dargestellt.

Aufgabe 1: Rotationsmatrizen

1. Die Rotationsmatrix R anlegen

```
>> R = [0.36,0.48,-0.8; -0.8,0.6,0; 0.48,0.64,0.6]
```

2. Die Robotics Toolbox beinhaltet das $SO3$-Objekt, welches Orientierungen im Raum repräsentiert und über die Rotationsmatrix R erzeugt werden kann:

```
>> p = SO3(R)
```

3. Über das $SO3$-Objekt lassen sich verschiedene Darstellungen der durch R repräsentierten Orientierung erzeugen:

```
>> p.torpy()          % Roll-Pitch-Yaw
>> p.toeul()          % Euler Z-Y-Z
>> p.toangvec()       % Angle-Axis
```

   Insbesondere lässt sich die Orientierung als Einheitsquaternion darstellen (Lösung zu Aufgabe 1.3):

```
>> p.UnitQuaternion()
ans =
0.8 < 0.2, -0.4, -0.4 >
```

Aufgabe 2: Homogene Transformationen

1. Die Transformationsmatrix T und den Vektor v anlegen:

```
>> T=[0,0,1,5; 0,1,0,0; -1,0,0,0; 0,0,0,1]
>> v=[1; 2; 3; 1]
```

2. Die Robotics Toolbox beinhaltet das $SE3$-Objekt, welches homogene Transformationen im Raum repräsentiert und über die Transformationsmatrix T erzeugt werden kann:

```
>> p = SE3(T)
```

3. Um herauszufinden, welche Transformation T darstellt, wird zunächst die Rotationskomponente in der gut lesbaren Roll-Pitch-Yaw-Konvention dargestellt:

```
>> p.SO3.torpy('deg')
ans = 0  90  0
```

Analog kann die Translationskomponente als Vektor dargestellt werden:

```
>> p.t
ans = 5  0  0
```

T repräsentiert also eine Drehung um 90° um die y-Achse und eine Translation um 5m entlang der x-Achse (Lösung zu Aufgabe 1.1).

4. Durch den Multiplikationsoperator kann die Transformation auf den Vektor v angewandt werden (Lösung zu Aufgabe 1.2):

```
>> p.T * v
ans = 8  2  -1  1
```

5. Die zu T inverse Transformationsmatrix lässt sich folgendermaßen bestimmen (Lösung zu Aufgabe 1.3):

```
>> inv(p.T)
ans =
     0     0    -1     0
     0     1     0     0
     1     0     0    -5
     0     0     0     1
```

Aufgabe 3: Transformationsmatrizen

1. Die Transformationsmatrix T_{init} anlegen

```
>> T0 = [1,0,0,5; 0,1,0,3; 0,0,1,0; 0,0,0,1]
```

2. Rotation um 90° um die z-Achse:

```
>> T01 = trotz(90, 'deg')
```

3. Translation um 4 Einheiten in x-Richtung

```
>> T12 = transl(4, 0, 0)
```

4. Translation um 2 Einheiten in x-Richtung und um 3 Einheiten in y-Richtung, sowie eine Rotation um −45° um die z-Achse:

```
>> T23 = transl(2, 3, 0) * trotz(-45, 'deg')
```

5. Die Verkettung der Transformationsmatrizen löst Aufgabe 3:

```
>> T0 * T01 * T12 * T23
ans =
    0.7071   -0.7071         0    2.0000
    0.7071    0.7071         0    9.0000
         0         0    1.0000         0
         0         0         0    1.0000
```

Aufgabe 4: Translations- und Rotationsdifferenz

1. Transformationsmatrizen T_{tcp} und T_{goal} anlegen und korrespondierende $SE(3)$-Objekte erzeugen:

```
>> Ttcp = [0,-1,0,1; 1,0,0,2; 0,0,1,3; 0,0,0,1]
>> ptcp = SE3(Ttcp)

>> Tgoal = [0,0,-1,7; 0,1,0,6; 1,0,0,5; 0,0,0,1]
>> pgoal = SE3(Tgoal)
```

2. Die Positionsdifferenz kann als Länge der Differenz der Translationsvektoren berechnet werden:

```
>> dt = pgoal.t - ptcp.t
>> norm(dt)
ans = 7.4833
```

3. Zur Bestimmung der Rotationsdifferenz transformieren wir T_{goal} in das durch T_{tcp} definierte Koordinatensystem und überführen die resultierende Rotation in eine Angle-Axis-Darstellung, um den Rotationswinkel zu erhalten:

```
>> Rdiff = inv(ptcp.SO3) * pgoal.SO3
>> Rdiff.toangvec('deg')
Rotation: 120.000000 deg x [-0.577350 -0.577350 -0.577350]
```

   Die Rotationsdifferenz ergibt sich also zu 120°.

Aufgabe 5: Quaternionen

1. Erzeugen des Vektors p und Darstellung als Quaternion (Lösung zu Aufgabe 5.1):

```
>> p = [5,1,7]
>> v = Quaternion(0, p)
v =
0 << 5, 1, 7 >>
```

2. Erzeugen eines Rotationsquaternions aus einem Winkel ϕ und einer Achse a (Lösung zu Aufgabe 5.2):

```
>> phi = 90
>> a = [0,0,1]
>> q = UnitQuaternion.angvec(phi/180*pi, a)
q =
0.70711 < 0, 0, 0.70711 >
>> qc = q.conj()
qc =
0.70711 < 0, 0, -0.70711 >
```

3. Transformation des Punktes p mit q (Lösung zu Aufgabe 5.3):

```
>> q * v * qc
ans =
0 << -1, 5, 7 >>
```

4. SLERP-Interpolation (Lösung zu Aufgabe 5.4):

```
>> a1 = [1,0,0]
>> q1 = UnitQuaternion.angvec(pi,a1)
>> a2 = [0,1,0]
>> q2 = UnitQuaternion.angvec(pi,a2)
>> q2.interp(q1, 0.5)
ans =
8.6596e-17 < 0.70711, 0.70711, 0 >
```