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

Robotik I im WS 2017/18

Musterlösungen zum 4. Übungsblatt

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

Lösung 1 (Puma 560 aus der Robotics Toolbox verwenden)

Teilaufgaben:

1. Laden Sie das Modell des Roboters (Befehl: `mdl_puma560`).

```
mdl_puma560
```

2. Lassen Sie sich die Informationen über den Roboter und insbesondere seine DH-Parameter anzeigen (Befehl: `p560`).

```
p560
```

Ausgabe:

```
p560 =

        Puma 560 [Unimation]:: 6 axis, RRRRRR, stdDH, slowRNE
        - viscous friction; params of 8/95;
        [DH-Parameter ausgelassen]
```

3. Verwenden Sie die Funktion `p560.plot(q)`, um sich den Roboter in einem bestimmten Zustand $\boldsymbol{q} \in \mathbb{R}^6$ anzuzeigen.

```
q = [0 0 0 0 0 0]
p560.plot(q)
```

Ausgabe: Ein Fenster öffnet sich mit dem Roboterarm in der angegebenen Konfiguration.

4. Machen Sie sich mit der Kinematik des Roboters vertraut, indem Sie die Funktion `p560.teach(q)` verwenden.

```
q = [0 0 0 0 0 0]
p560.teach(q)
```

Ausgabe: Ein Fenster öffnet sich mit dem Roboterarm in der angegebenen Konfiguration. Die Konfiguration kann über Slider in dem Fenster angepasst werden.

5. Berechnen Sie mithilfe der Vorwärtskinematik (Funktion: `p560.fkine(q)`) die Position des Endeffektors in der Konfiguration $\boldsymbol{q}_0 = (0, 0, 0, 0, 0, 0)$.

```
q = [0 0 0 0 0 0]
p = p560.fkine(q)
```

Ausgabe:

```
p =
        1        0        0   0.4521
        0        1        0    -0.15
        0        0        1   0.4318
        0        0        0        1
```

6. Berechnen Sie mithilfe der inversen Kinematik (Funktion: `p560.ikine(p)`) eine Konfiguration $\boldsymbol{q}_p$, die die Endeffektorpose p erreicht. Die Pose p setzt sich aus einer Translation von $\boldsymbol{t}_p = (0.7, 0.3, 0)^T$ und einer Rotation von 90 deg um die y-Achse zusammen.

```
Tp = transl(0.7, 0.3, 0) * troty(90, 'deg')
p = SE3(Tp)
qp = p560.ikine(p)
```

Ausgabe:

```
qp =
    0.6032   -0.5278   -0.4689   -0.9032   -0.8072    0.7200
```

7. Bestimmen Sie eine Trajektorie zwischen der Konfiguration $\boldsymbol{q}_0$ und $\boldsymbol{q}_p$. Die Trajektorie soll in zwei Sekunden abgefahren werden. Hinweis: Verwenden Sie die Funktion `mtraj` zur Erzeugung der Trajektorie mit `lspb` als skalarer Trajektorienfunktion.

   Wir erzeugen ein Zeitintervall von $0s$ bis $2s$ mit einer Diskretisierung von $0.05s$.

```
t = [0:0.05:2]
[q, qd, qdd] = mtraj(@lspb, q0, qp, t)
```

Ausgabe:
Werte für die Gelenkwinkelpositionen, -geschwindigkeiten und -beschleunigungen.

```
q   = ...
qd  = ...
qdd = ...
```

Lösung 2 (Roboter mit zwei Gelenken modellieren)

1. Bestimmen Sie die DH-Parameter für die beiden Gelenke des Roboters.

   DH-Parameter:

Gelenk	θ	d	α	a
1	90 deg	d_1 (variabel)	90 deg	0
2	θ_2 (variabel)	0	0 deg	1

   Hinweis: Je nach Wahl der Koordinatensysteme sind mehrere Lösungen möglich. Sie können bei der Modellierung auch Ihre DH-Parameter verwenden und so prüfen, ob diese ebenfalls korrekt sind.

2. Erstellen Sie mithilfe der DH-Parameter einen `SerialLink`, der diesen Roboter als Verkettung eines `Prismatic`- und eines `Revolute`-Gelenks modelliert.

   Erstellen der beiden Gelenke:

```
%% DH-Parameter und Dynamikeigenschaften
%% DH-Parameter 'd' ist variabel (Schubgelenk)
prismatic_joint = Prismatic( ...
        'theta', pi/2, ...
        'a', 0, ...
        'alpha', pi/2, ...
        ...
        'm', 1, ...
        'r', [0 0 -0.5], ...
        'I', [0.25 0.25 0], ...
        'B', 0, ...
        'G', 0, ...
        'Jm', 0, ...
        'standard')

%% DH-Parameter und Dynamikeigenschaften
%% DH-Parameter 'theta' ist variabel (Drehgelenk)
revolute_joint = Revolute( ...
        'd', 0, ...
        'a', 1, ...
        'alpha', 0, ...
        ...
        'm', 1, ...
        'r', [-0.5 0 0], ...
        'I', [0.25 0.25 0], ...
        'B', 0, ...
        'G', 0, ...
        'Jm', 0, ...
        'standard')
```

   Den Roboter erstellen:

```
two_link = SerialLink([
        prismatic_joint
```

```
        revolute_joint
]);

%% Rotieren des BKS
two_link.base=SE3(0,0,0) * SE3.Ry(pi/2);
two_link.links(2).offset = pi/2;
two_link.links(1).offset = 1;
two_link.links(1).qlim = [0 0.5];

two_link.gravity=([9.81 0 0]);
```

Den Roboter anzeigen:

```
ws = [0 3 -1 1 -0.5 3]; % Size of Workspace
plot_options = {'workspace',ws,'nobase'};
two_link.plotopt = plot_options;
two_link.teach([0 0], 'view','top');
```

3. Testen Sie die Vorwärts- und inverse Kinematik sowie die Trajektoriengenerierung. Orientieren Sie sich dabei an dem Vorgehen in Aufgabe 1.

   Hier werden beispielhaft einige Funktionen aufgerufen, mit denen Sie selbst weiter experimentieren können:

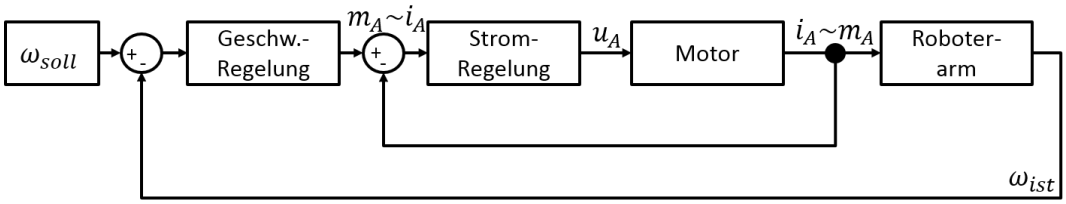
```
%% Trajectory in joint space
FK = two_link.fkine([0.5 pi/2])
FK.toeul % In Eulerwinkel

%% Mask: Genutzte Robotergelenke in kartesischen Koordinaten
%%       im Endeffektorkoordinatensystems.
%% Hier bewegung in XY Ebene
IK = two_link.ikine(FK,'mask',[1 1 0 0 0 0]);

T1 = transl(2,0,0); % Startpunkt der Translation
T2 = transl(2,1,0)*trotz(pi/2); % Endpunkt der Translation

%% Inverse Kinematik des Start und Zielpunktes
q1 = two_link.ikine(T1,'mask',[1 1 0 0 0 0]);
q2 = two_link.ikine(T2,'mask',[1 1 0 0 0 0]);

%% Zeitvektor
t1 = [0:0.05:2]';

%% Trajektorie im Jointspace (Standard: Polynomapprox.)
[q qd qdd] = mtraj(@lspb,q1,q2,t1);

%% Plotten der Trajektorie
two_link.plot(q)
hold on
```

```
%% Berechnen und Plotten der Endeffektorbewegung
FK1 = two_link.fkine(q);
p = transl(FK1);
plot3(p(:,1),p(:,2),p(:,3));

%% Plotten der Jointtrajektorien
figure(2)
plot(t1,q);
legend('q1','q2');

%%Jacobi Matrix
J1 = two_link.jacob0([0 0]);
figure(3);two_link.plot([ 0 0], 'view','top');

%% In diesem Fall dreht sich das Gelenk nicht in die
%% positive Richtung. D.h. Drehung
um die Z Achse und
%% Translation in negative X und Y Richtung
vB1 = J1*[0.5 0]';


J2 = two_link.jacob0([0 0]);
vB2 = J2*[0 pi/2]';


J3 = two_link.jacob0([0 pi/4]);
vB3 = J3*[0 pi/2]';
two_link.plot([ 0 pi/4], 'view','top');

%% Recursives Newton-Euler-Verfahren zum Berechnen der
%% Kraefte und Momenten in
den Gelenken

qn = [0 0];
qz = [0 0];
two_link.plot(qn, 'view','top');

%% Berechnen der Joint Torque/Force fuer Position 0 0.
Q = two_link.rne(qn,qz,qz);
```

Lösung 3 (Kaskadierte Regler)

Bearbeiten Sie folgende Teilaufgaben:

1. Zeichnen Sie ein Blockdiagram des kaskadierten Reglers.



   Legende:

   - ω_{soll}: Vorgabe für die Winkelgeschwindigkeit
   - ω_{ist}: Gemessene Winkelgeschwindigkeit am Roboterarm
   - m_A: Drehmoment
   - i_A: Motorstrom
   - u_A: Spannung am Motor

2. Berechnen Sie die Laplace-Transformationen für folgende Gleichungen, welche den Zusammenhang von Regel- und Stellgröße in beiden Reglern beschreiben.

   Die Laplace-Transformation kann mithilfe des Linearitätssatzes und des Differentiationssatzes gelöst werden.

$$\begin{aligned} U_A(s) &= L\{u_A(t)\} \\ &= L\{R_A \cdot i_A(t) + L_A \cdot \frac{d}{dt} i_A(t)\} \\ &= R_A \cdot L\{i_A(t)\} + L_A \cdot L\{\frac{d}{dt} i_A(t)\} \\ &= R_A \cdot I_A(s) + L_A \cdot s \cdot I_A(s) \end{aligned}$$

$$\begin{aligned} M_A(s) &= L\{m_A(t)\} \\ &= L\{J \cdot \frac{d}{dt}\omega(t) + K_v \cdot \omega(t)\} \\ &= J \cdot L\{\frac{d}{dt}\omega(t)\} + K_v \cdot L\{\omega(t)\} \\ &= J \cdot s \cdot \Omega(s) + K_v \cdot \Omega(s) \end{aligned}$$

3. Stellen Sie die Übertragungsfunktionen für beide Regler auf.

   Die Übertragungsfunktion eines Reglers kann über die Division der Laplacetransformierte Regelgröße durch die Laplace-transformierten Stellgröße bestimmt werden.

$$\text{Übertragungsfunktion} = \frac{L\{\text{Regelgröße}\}}{L\{\text{Stellgröße}\}}$$

Motor:

$$\begin{aligned}\frac{I_A(s)}{U_A(s)} &= \frac{I_A(s)}{R_A \cdot I_A(s) + L_A \cdot s \cdot I_A(s)} \\ &= \frac{I_A(s)}{I_A(s) \cdot (R_A + L_A \cdot s)} \\ &= \frac{1}{R_A + L_A \cdot s}\end{aligned}$$

Mechanik:

$$\begin{aligned}\frac{\Omega(s)}{M_A(s)} &= \frac{\Omega(s)}{J \cdot s \cdot \Omega(s) + K_v \cdot \Omega(s)} \\ &= \frac{\Omega(s)}{\Omega(s) \cdot (J \cdot s + K_v)} \\ &= \frac{1}{J \cdot s + K_v}\end{aligned}$$