2 Dynamik und Regelung

Inhalt des Kapitels

2.0 Übungsaufbau und Materialien

In dieser zweiten Übung werden Sie sich mit der Roboterdynamik und -regelung beschäftigen. Dafür wird der KUKA LBR iiwa 14 R820, ein Leichtbauroboter (LBR) mit sieben rotatorischen Freiheitsgraden, betrachtet, siehe Abbildung 2.1. Er wird in Abschnitt 2.1 eingeführt.

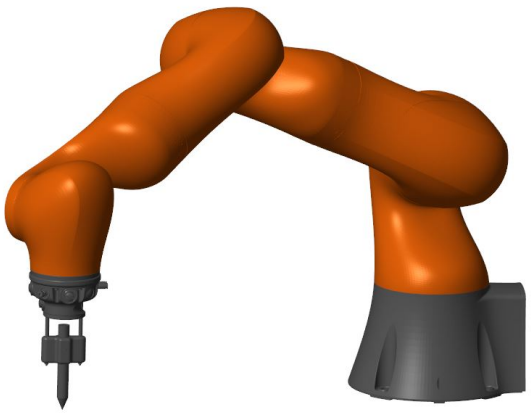


Abbildung 2.1: Für die Übung verwendete Visualisierung des KUKA LBR iiwa mit einem Werkzeug am Endeffektor.

Für diese Übung steht Ihnen ein Dynamikmodell des KUKA LBR iiwa zur Verfügung.

Bevor Sie es verwenden, werden Sie in Abschnitt 2.2 zunächst Plausibilitätschecks durchführen. In der Aufgabe aus Abschnitt 2.3 werden Sie den Verlauf der Kontaktkräfte berechnen, die auf den Roboter wirken, während dieser eine Wischbewegung auf einer Tischplatte ausführt. Schließlich werden Sie in Abschnitt 2.4 verschiedene Regler für den KUKA LBR iiwa im Konfigurations- und Aufgabenraum implementieren und analysieren.

2.0.1 Softwarevoraussetzungen

Zur Bearbeitung der Aufgaben benötigen Sie MATLAB 2022b. Stellen Sie sicher, dass Ihre MATLAB-Installation mindestens die Toolboxen *Matlab, Simulink, Simscape* und *Simscape Multibody* umfasst. Die Verwendung weiterer Software steht Ihnen frei, ist jedoch nicht erforderlich.

2.0.2 Ordnerstruktur

Bitte arbeiten Sie für die Übung im Ordner *Uebung2*, den Sie sich vom TUWEL-Kurs herunterladen können. Darin finden Sie vier Unterordner:

- *contact_force*:
  In diesem Ordner finden Sie das Material zur Aufgabe aus Abschnitt 2.3. Bitte legen Sie dort auch alle Dateien ab, die Sie zu deren Lösung erstellen.
- *control*:
  In diesem Ordner finden Sie Vorlagen für die Regelungsaufgaben aus Abschnitt 2.4. Bitte legen Sie dort auch alle Dateien ab, die Sie zu deren Lösung erstellen.
- *dynamics_iiwa*:
  In diesem Ordner finden Sie mehrere mögliche Funktionen zur Berechnung der Dynamik des Roboters. In Abschnitt 2.2 werden Sie die Korrekte auswählen und für weitere Aufgaben verwenden.
- *visualization*:
  Dieser Ordner enthält eine SIMULINK-Datei zur Visualisierung des Roboters mit SIMSCAPE sowie das MATLAB-Skript *init_Visualization.m* zum Starten des SIMULINK-Modells.

Hinweis: Stellen Sie während der Bearbeitung sicher, dass stets alle Ordner in Ihrem MATLAB-Pfad sind. Eine nützliche Funktion, um das programmatisch sicherzustellen, ist `addpath(folderName1,...,folderNameN)` auch in Kombination mit `genpath(folderName)`.

2.1 KUKA LBR iiwa

Der KUKA LBR iiwa ist seriell aus $n = 7$ Drehgelenken aufgebaut. Seine rotatorischen Freiheitsgrade q_i, mit $i = 1, ..., 7$, sind zum Vektor $\mathbf{q} = \begin{bmatrix} q_1 & \dots & q_7 \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^7$ der Konfiguration zusammengesetzt. Die Gelenkgeschwindigkeiten werden mit $\dot{\mathbf{q}} \in \mathbb{R}^7$ bezeichnet.

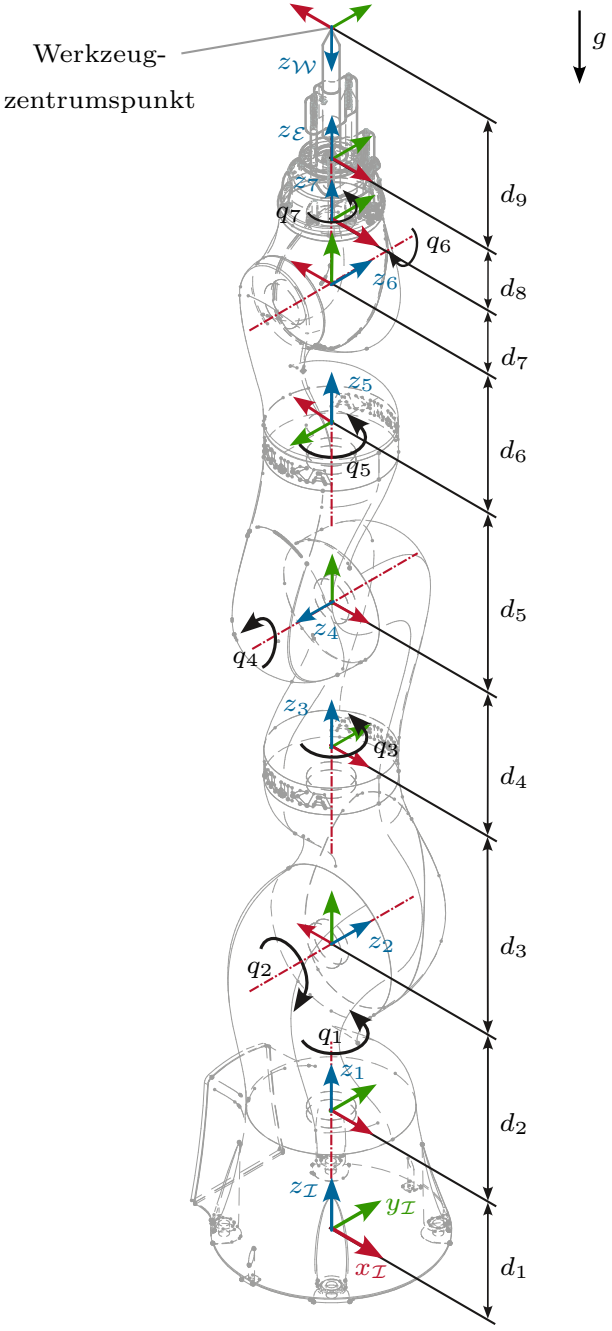


Abbildung 2.2: Schematischer Aufbau des KUKA LBR iiwa.

In Abbildung 2.2 sind der schematische Aufbau des Roboters, dessen Freiheitsgrade für $\mathbf{q} = \mathbf{0}$ und die wichtigsten Koordinatensysteme dargestellt. In der Übung wird angenommen, dass der Roboter stehend auf einer sich am Boden befindenden Basisplatte

montiert ist. Die Gravitation wirkt in negative $z_{\mathcal{I}}$-Richtung. Am Endeffektor des Roboters befindet sich ein Werkzeug. Für die Roboterkinematik sind im Folgenden insbesondere das Inertialkoordinatensystem $\mathcal{I}$ und das Werkzeugkoordinatensystem $\mathcal{W}$ entscheidend. Bitte beachten Sie, dass das Werkzeugkoordinatensystem $\mathcal{W}$ so orientiert ist, dass die $z_{\mathcal{W}}$-Achse in das Werkzeug hinein zeigt.

Im Folgenden wird der Roboter als Starrkörpersystem modelliert. Die Roboterdynamik lässt sich folglich in der bekannten Form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} - \boldsymbol{\tau}_{\text{ext}} \tag{2.1}$$

schreiben. Darin werden die Motormomente in $\boldsymbol{\tau} \in \mathbb{R}^7$ und die externen Momente aus der Interaktion mit der Umgebung in $\boldsymbol{\tau}_{\text{ext}} \in \mathbb{R}^7$ zusammengefasst. Die Massenmatrix wird als $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{7\times7}$, die Coriolismatrix als $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{7\times7}$ und die Gravitationsterme als $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^7$ bezeichnet. Letztgenannte Dynamikmatrizen erfüllen die in der Vorlesung spezifizierten Eigenschaften.

Wie bereits erwähnt, steht Ihnen im Ordner *visualization* ein SIMSCAPE-Modell zur Bewegungsvisualisierung des Roboters zur Verfügung. Als Eingang benötigt das Modell die Verläufe von $\mathbf{q}(t)$ und $\dot{\mathbf{q}}(t)$. Es können sowohl Bewegungen des Roboters als auch einzelne Konfigurationen ($\dot{\mathbf{q}} = \mathbf{0}$) veranschaulicht werden.

2.2 Plausibilitätsprüfung der Dynamikmodellierung

Bevor Sie in den folgenden Aufgaben ein gegebenes Dynamikmodell des KUKA LBR iiwa verwenden, sollen Sie es in dieser Aufgabe zunächst analysieren und auf Plausibilität überprüfen. Ziel wird es sein, die Funktion `[M,C,g,H,Jg,dJg] = dynamics_iiwa(q,dq)` im Ordner *dynamics_iiwa* korrekt auszuwählen. Sie gibt für eine gegebene Konfiguration $\texttt{q} = \mathbf{q} \in \mathbb{R}^7$ und Gelenksgeschwindigkeit $\texttt{dq} = \dot{\mathbf{q}} \in \mathbb{R}^7$ die dynamischen und kinematischen Größen des Roboters zurück. Konkret sind die Ausgänge wie folgt definiert:

- $\texttt{M} = \mathbf{M}(\mathbf{q})$: Massenmatrix.
  In der Massenmatrix sind die effektiven Rotorträgheiten J_i für $i = 1, ..., 7$ mitberücksichtigt, indem die Werte auf die korrespondierenden Diagonalelemente $m_{ii}(\mathbf{q})$ der Massenmatrix addiert wurden. Für den KUKA LBR iiwa wurde dafür $J_i = 0.1\,\text{kg}\,\text{m}^2$ für jedes Gelenk $i = 1, ..., 7$ angenommen. Vergleichen Sie dafür auch Kapitel 6.2.3 des Vorlesungsskriptums.
- $\texttt{C} = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$: Coriolismatrix.
- $\texttt{g} = \mathbf{g}(\mathbf{q})$: Gravitationsterme.
- $\texttt{H} = \mathbf{H}_{\mathcal{I}}^{\mathcal{W}}(\mathbf{q})$: homogene Transformation vom Werkzeugsystem ins Inertialsystem.
- $\texttt{Jg} = \mathbf{J}_g(\mathbf{q}) \in \mathbb{R}^{6\times7}$: geometrische Jacobi-Matrix.
  Die geometrische Jacobi-Matrix bezieht sich auf den Werkzeugzentrumspunkt. Es gilt der Zusammenhang $\mathbf{J}_g(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix}\dot{\mathbf{p}}_e^{\text{T}} & \boldsymbol{\omega}_e^{\text{T}}\end{bmatrix}^{\text{T}}$, wobei die zeitliche Ableitung der Werkzeugsposition $\dot{\mathbf{p}}_e$, sowie die Winkelgeschwindigkeit $\boldsymbol{\omega}_e$ in Bezug auf das Inertialsystem $\mathcal{I}$ gegeben sind.

- `dJg` $= \dot{\mathbf{J}}_g(\mathbf{q}, \dot{\mathbf{q}})$: zeitliche Ableitung der geometrischen Jacobi-Matrix.

Im Ordner stehen Ihnen des Weiteren fünf MATLAB-Funktionen `[M, C, g, H, Jg, dJg] = dynamics_iiwa_i(q, dq)` mit `i = {1,...,5}` als Kandidaten für die gesuchte Funktion `dynamics_iiwa` zur Verfügung. Jede der fünf Funktionen gibt den identischen und korrekten Satz der kinematischen Matrizen $\mathbf{H}_{\mathcal{I}}^{\mathcal{W}}(\mathbf{q})$, $\mathbf{J}_g(\mathbf{q})$ und $\dot{\mathbf{J}}_g(\mathbf{q}, \dot{\mathbf{q}})$ zurück. Allerdings sind die ausgegebenen Dynamikmatrizen $\mathbf{M}(\mathbf{q}), \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ und $\mathbf{g}(\mathbf{q})$ in vier der fünf Funktionen fehlerhaft.

Aufgabe 2.1. Bestimmen Sie, welche dieser fünf Funktionen eine gültige Kombination an Dynamikmatrizen zurückgibt, die die Dynamik des Roboters im in den Abschnitten 2.1 und 2.2 beschriebenen Setup modellieren.

Hinweis: Lösen Sie diese Aufgabe ausdrücklich nicht, indem Sie ein eigenes Dynamikmodell aufbauen und die Ergebnisse vergleichen. Wenden Sie stattdessen das **Ausschlussverfahren** an. Für das Abgabegespräch wird erwartet, dass Sie begründen können, anhand welcher Eigenschaften Sie welche Funktion ausgeschlossen haben. Es sind keine aufwändigen Rechnungen nötig, um die Aufgabe zu lösen.

Hinweis: Achten Sie auf die erwarteten Eigenschaften der Dynamikmatrizen $\mathbf{M}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ und $\mathbf{g}(\mathbf{q})$ des Roboters. **Alle drei Matrizen** müssen analysiert werden, um die richtige Funktion zu finden.
Die kinematischen Größen $\mathbf{H}_{\mathcal{I}}^{\mathcal{W}}(\mathbf{q})$, $\mathbf{J}_g$ und $\dot{\mathbf{J}}_g$ sind in jeder Funktion korrekt und für diese Aufgabe nicht relevant.

Hinweis: Betrachten Sie die Dynamikmatrizen in **aussagekräftigen Konfigurationen** ($\dot{\mathbf{q}} = \mathbf{0}$) und in der **Bewegung**. Dabei kann die SIMSCAPE-Visualiserung hilfreich sein. Um eine Trajektorie zu analysieren, bietet sich die Verwendung von SIMULINK an. Sollten Sie zeitliche Ableitungen benötigen, können Sie dort auf den Block *Derivative* zurückgreifen und die benötigten Signale numerisch ableiten.

Hinweis: Auch **numerische Ungenauigkeiten** können dazu führen, dass geforderte Eigenschaften der Dynamikmatrizen nicht exakt eingehalten werden. Verletzungen in kleiner Größenordnung sollen für diese Aufgabe vernachlässigt werden.

Vervollständigen Sie nun die Funktion `[M,C,g,H,Jg,dJg] = dynamics_iiwa(q,dq)`. Sie soll einzig die von Ihnen als korrekt bestimmte Dynamikfunktion aufrufen. Die bereitgestellten Templates der folgenden Aufgaben rufen diese Funktion auf und hängen somit von der von Ihnen gewählten Dynamikfunktion ab. Wir empfehlen, dass auch Sie für alle weiteren Implementierungen die Funktion `dynamics_iiwa` verwenden.

Hinweis: Zwei der bereitgestellten Funktionen liefern ungültige Dynamikmatrizen. Die anderen beiden fehlerhaften Funktionen geben zwar gültige Matrizen zurück, diese können jedoch nicht zum beschriebenen Roboter gehören. Solange Sie in dieser Aufgabe eine Funktion mit gültigem Dynamkikmodell auswählen, können Sie alle weiteren Aufgaben bearbeiten.

2.3 Kontaktkraftberechnung

Der Roboter führt eine Wischbewegung auf einem $h_{\text{Tisch}} = 20\,\text{cm}$ hohen Tisch aus, das heißt die Tischplatte liegt bei $z_{\mathcal{I}} = h_{\text{Tisch}}$. Es wird im Folgenden angenommen, dass der Roboter dabei weder in die Tischplatte eindringen noch sich von ihr lösen kann. Es gilt demnach eine holonome Zwangsbedingung in $z_{\mathcal{I}}$-Richtung. Parallel zur Tischplatte ist eine Bewegung möglich. Ziel dieses Abschnittes ist es, aus aufgezeichneten Daten die Kontaktkraft zu bestimmen, mit der der Roboter während eines Versuchs auf den Tisch wirkte.

Aufgabe 2.2. Stellen Sie die holonome Zwangsbedingung $\phi(\mathbf{q}) = 0$ auf und bestimmen Sie daraus die Matrizen $\mathbf{A}(\mathbf{q}) = \frac{\partial \phi(\mathbf{q})}{\partial \mathbf{q}} \in \mathbb{R}^{1\times 7}$ und $\dot{\mathbf{A}}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{1\times 7}$. Finden Sie Ausdrücke für die beiden Matrizen, die nur von $\mathbf{q}$ beziehungsweise $\mathbf{q}$ und $\dot{\mathbf{q}}$ abhängen. Sie können in Ihren Ausdrücken geeignete kinematischen Größen verwenden, die von `dynamics_iiwa` zurückgegeben werden.

Die Bewegung des Roboters im Kontakt mit dem Tisch wurde mit einer Abtastzeit von $T_a = 0.1\,\text{ms}$ aufgenommen. Die gesamte Bewegung dauerte $T = 6\,\text{s}$, sodass sich insgesamt $N = 1 + T/T_a = 60001$ Abtastpunkte ergeben. Zu jedem Zeitpunkt $t_k = kT_a$ für $k = 0, \ldots, N-1$ sind die momentanen Werte von $\mathbf{q}(t_k), \dot{\mathbf{q}}(t_k)$ und $\boldsymbol{\tau}(t_k)$ aufgenommen. Die Werte der Konfigurationen `q_Force` $= [\mathbf{q}(t_0) \ \ldots \ \mathbf{q}(t_{N-1})] \in \mathbb{R}^{7\times N}$, der Gelenkgeschwindigkeiten `dq_Force` $= [\dot{\mathbf{q}}(t_0) \ \ldots \ \dot{\mathbf{q}}(t_{N-1})] \in \mathbb{R}^{7\times N}$, der Motormomente `tau_Force` $= [\boldsymbol{\tau}(t_0) \ \ldots \ \boldsymbol{\tau}(t_{N-1})] \in \mathbb{R}^{7\times N}$ und der Zeitpunkte `time_Force` $= [t_0 ... t_{N-1}] \in \mathbb{R}^N$ sind im Ordner *contact_force* in der MATLAB-Datei *ContactForceData.mat* hinterlegt.

Aufgabe 2.3. Laden Sie die Werte von `q_Force`, `dq_Force`, `tau_Force` und `time_Force` in Ihren MATLAB-Workspace. Schreiben Sie ein MATLAB-Skript, mit dem Sie die Kontaktkraft $\lambda \in \mathbb{R}$ zu jedem Zeitpunkt aus `time_Force` mithilfe der Dynamikfunktion aus Aufgabe 2.1 und der Matrizen $\mathbf{A}(\mathbf{q}), \dot{\mathbf{A}}(\mathbf{q}, \dot{\mathbf{q}})$ bestimmen. Stellen Sie schließlich den Zeitverlauf von λ grafisch dar. Wählen Sie aus den möglichen qualitativen Verläufen (1) bis (5) in Abbildung 2.3 den richtigen aus.

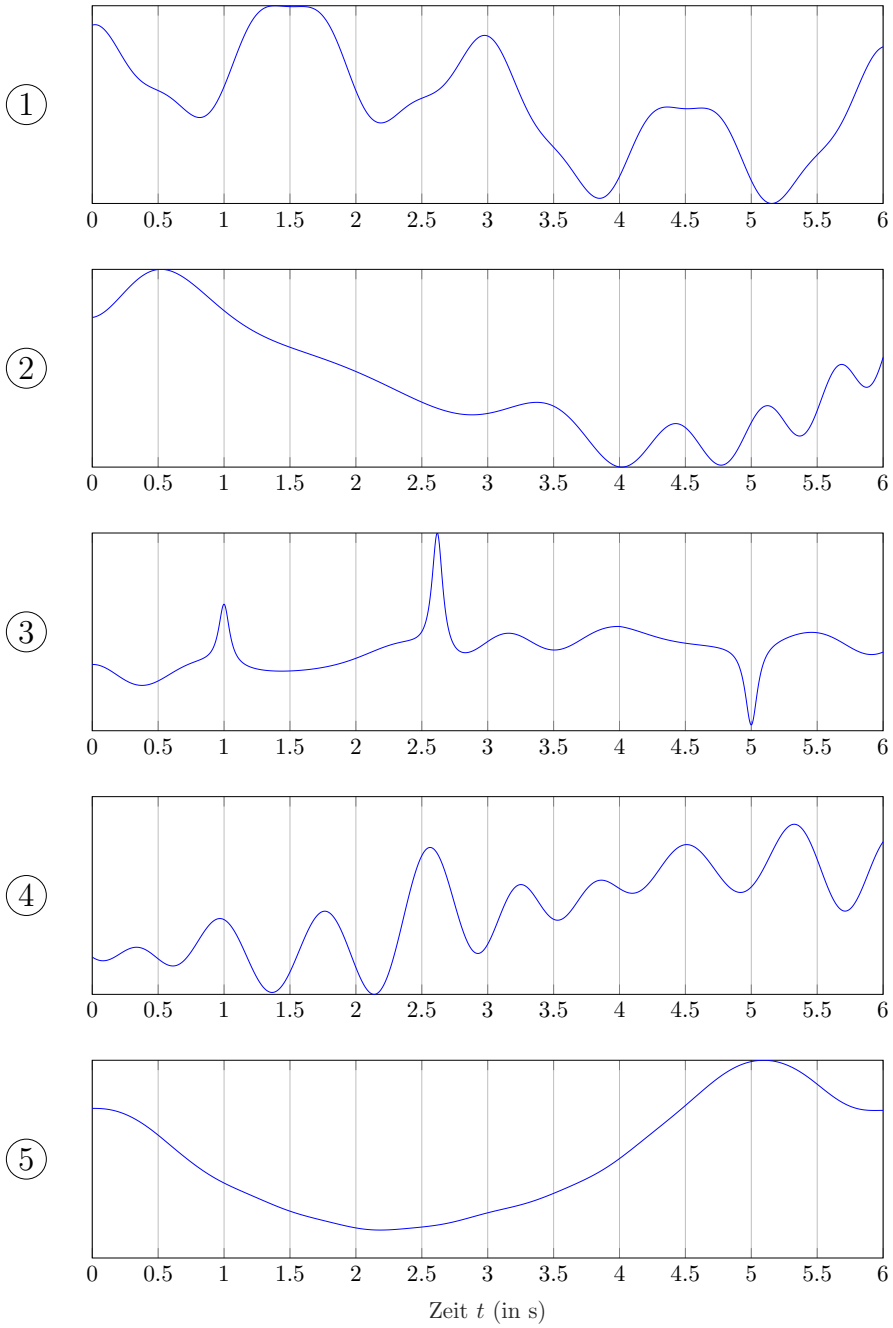


Abbildung 2.3: Mögliche qualitative Verläufe von λ für Aufgabe 2.3

2.4 Regelung

Für die Regelung des Roboters werden Sie in den Aufgaben dieses Abschnitts Regler im Konfigurations- und auch im Aufgabenraum implementieren und untersuchen. Arbeiten Sie

für alle folgenden Aufgaben im Ordner *control* der zur Verfügung gestellten Ordnerstruktur.

2.4.1 Computed-Torque-Regler im Konfigurationsraum

Zunächst wird die Trajektorienfolgeregelung im Konfigurationsraum betrachtet. Exemplarisch werden Sie einen Computed-Torque-Regler implementieren, um einer zweifach stetig differenzierbaren Trajektorie $\mathbf{q}_d(t) \in \mathbb{R}^7$ im Konfigurationsraum des Roboters zu folgen.

Aufgabe 2.4. Vervollständigen Sie in dieser Aufgabe das SIMULINK-Modell *ComputedTorque_iiwa.slx*. Darin enthalten ist ein Trajektoriengenerator, der Ihnen die Solltrajektorie in der Form `qd` = $\mathbf{q}_d(t)$, `dqd` = $\dot{\mathbf{q}}_d(t)$ und `ddqd` = $\ddot{\mathbf{q}}_d(t)$ zur Verfügung stellt. Es gilt

$$\mathbf{q}_d(t=0) = \begin{bmatrix} 0 & \pi/4 & 0 & -\pi/2 & 0 & \pi/4 & 0 \end{bmatrix}^{\mathrm{T}} \mathrm{rad},$$
$$\dot{\mathbf{q}}_d(t=0) = \mathbf{0}.$$

Desweitern sind auch die Roboterdynamiksimulation und eine Visualisierung des KUKA LBR iiwa schon im SIMULINK-Modell enthalten. Zum Starten des Modells können Sie das MATLAB-Skript *init_ComputedTorque.m* verwenden. Bei Bedarf können Sie es um zusätzliche Parameter oder Berechnungen ergänzen. Bearbeiten Sie die folgenden Aufgabenteile:

(a) Implementieren Sie einen Computed-Torque-Regler im Konfigurationsraum zur Trajektorienfolge und vervollständigen Sie damit das SIMULINK-Modell. Verwenden Sie als Eingänge für Ihren Regler die Solltrajektorie $\mathbf{q}_d(t)$ mit allen nötigen Zeitableitungen, die Konfiguration $\mathbf{q}$, die Gelenkgeschwindigkeit $\dot{\mathbf{q}}$ sowie weitere benötigte Parameter.

(b) Verifizieren Sie das Trajektorienfolgeverhalten. Wählen Sie dafür geeignete Parameter für Ihren Regler und betrachten Sie das Trajektorienfolgeverhalten ausgehend von verschiedenen Startkonfigurationen $\mathbf{q}_0 = \mathbf{q}(t=0)$ und -geschwindigkeiten $\dot{\mathbf{q}}_0 = \dot{\mathbf{q}}(t=0)$. Sie können dafür die entsprechenden Variablen `q0` und `dq0` im MATLAB-Skript *init_ComputedTorque.m* geeignet setzen. Nehmen Sie $\boldsymbol{\tau}_{ext} = \mathbf{0}$ an. Erstellen Sie aussagekräftige MATLAB-Figures mit Ihren Simulationsergebnissen und speichern Sie diese für das Abgabegespräch ab.

Hinweis: Die Dynamik des KUKA LBR iiwa wird in der Simulation mit dem zur Verfügung gestellten Block *RobotDynamics* simuliert. Er erhält die Motormomente $\boldsymbol{\tau}$ und den Vektor der externen Momente auf jedes Gelenk $\boldsymbol{\tau}_{ext} \in \mathbb{R}^7$ als Eingänge. In diesem Block wird die Gelenkbeschleunigung gemäß

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1}(\boldsymbol{\tau} - \boldsymbol{\tau}_{ext} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q})) \tag{2.2}$$

berechnet und numerisch integriert, um Werte für $\mathbf{q}$ und $\dot{\mathbf{q}}$ ausgehend von den Startwerten $\mathbf{q}_0$ und $\dot{\mathbf{q}}_0$ zu erhalten. Beachten Sie, dass die Funktion `dynamics_iiwa` und damit Ihre Lösung von Aufgabe 2.1 zur Berechnung der Dynamikmatrizen in (2.2) verwendet wird.

2.4.2 Regelung im Aufgabenraum

In den folgenden Aufgabenteilen sollen Regler im Aufgabenraum implementiert, getestet und verglichen werden. Dabei sollen zusätzlich externe Kräfte $\mathbf{F}_{ext} \in \mathbb{R}^3$ auf das Werkzeug berücksichtigt werden.

Die Pose im Aufgabenraum wird durch

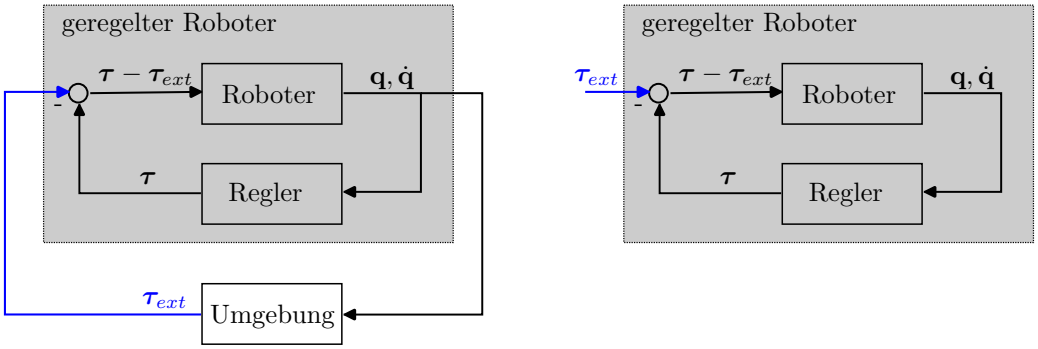
$$\mathbf{x}_e = \begin{bmatrix} \mathbf{p}_e^{\mathrm{T}} & \boldsymbol{\phi}_e^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^6 \tag{2.3}$$

beschrieben. Sie setzt sich aus der Position $\mathbf{p}_e \in \mathbb{R}^3$ des Werkzeuges im Inertialsystem und dessen Orientierung $\boldsymbol{\phi}_e = \boldsymbol{\phi}_{zyx}(\mathbf{R}_{\mathcal{I}}^{\mathcal{W}}(\mathbf{q})) \in \mathbb{R}^3$, parametriert mit den RPY-Winkeln, zusammen. Vergleichen Sie dazu auch Kapitel 4.1.1 des Vorlesungsskriptums.

Aus einer Kraft $\mathbf{F}_{ext}$ auf das Werkzeug resultieren die Drehmomente $\boldsymbol{\tau}_{ext} \in \mathbb{R}^7$ auf die Gelenke. In der Praxis kann sich eine solche Kraft $\mathbf{F}_{ext}$, und damit $\boldsymbol{\tau}_{ext}$, aus der Interaktion des Roboters mit der Umgebung ergeben. Der Vektor $\boldsymbol{\tau}_{ext}$ ist dadurch direkt von der Bewegung des Roboters abhängig (vgl. Abbildung 2.4 (a)). Für diese Übung wird die Umgebung zur Vereinfachung nicht simuliert. Die externen Drehmomente $\boldsymbol{\tau}_{ext}$ werden stattdessen als Systemeingang betrachtet, der nicht vom Systemzustand abhängt (vgl. Abbildung 2.4 (b)).

Im Folgenden soll das Werkzeug einer Trajektorie $\mathbf{x}_{e,d}(t) \in \mathbb{R}^6$ im Aufgabenraum folgen. Dabei soll sich der Roboter nachgiebig verhalten. Dazu können unter anderem ein kartesischer Impedanzregler und ein kartesischer Nachgiebigkeitsregler (PD+-Regler) im Aufgabenraum eingesetzt werden.

Für die gegebene Aufgabe ist der KUKA LBR iiwa redundant. Es gilt $m = 6$ und $n = 7$ und daher $m < n$. Aus diesem Grund soll im Folgenden neben der Trajektorienfolge im Aufgabenraum mit $\mathbf{x}_{e,d}(t)$ auch ein Verhalten im Nullraum geregelt werden.



(a) (b)

Abbildung 2.4: Externe Drehmomente $\boldsymbol{\tau}_{ext}$ als Resultat der Interaktion des Roboters mit seiner Umgebung (a) und vereinfachte Modellierung in der Übung (b).

Hinweis: Für die folgenden Aufgaben stehen Ihnen im Unterordner *helper_RPY* des Ordners *control* zwei Hilfsfunktionen zur Verfügung. Sie können Sie im Zusammenhang mit den RPY-Winkeln verwenden. Betrachtet werden zwei Koordinatensysteme $\mathcal{K}_1$ und $\mathcal{K}_2$. Mit $\mathbf{d}_{\mathcal{K}_1}^{\mathcal{K}_2} \in \mathbb{R}^3$ wird die Verschiebung von $\mathcal{K}_2$ gegenüber $\mathcal{K}_1$, angegeben in $\mathcal{K}_1$, bezeichnet. Der Vektor $\boldsymbol{\phi} \in \mathbb{R}^3$ beschreibt die Orientierung von $\mathcal{K}_2$ bezüglich $\mathcal{K}_1$ in RPY-Winkeln. Vergleichen Sie dazu auch Abschnitt 3.2.3 des Vorlesungsskriptums.

- `x = x_RPY_fromH(H)`: Für eine homogene Transformationsmatrix

$$\mathtt{H} = \mathbf{H}_{\mathcal{K}_1}^{\mathcal{K}_2} = \begin{bmatrix} \mathbf{R}_{\mathcal{K}_1}^{\mathcal{K}_2} & \mathbf{d}_{\mathcal{K}_1}^{\mathcal{K}_2} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3)$$

  von $\mathcal{K}_2$ nach $\mathcal{K}_1$ wird der zugehörige Vektor

$$\mathtt{x} = \mathbf{x} = \begin{bmatrix} \mathbf{d}_{\mathcal{K}_1}^{\mathcal{K}_2} \\ \boldsymbol{\phi}_{zyx}(\mathbf{R}_{\mathcal{K}_1}^{\mathcal{K}_2}) \end{bmatrix} \in \mathbb{R}^6 \tag{2.4}$$

  berechnet, der die Lage von $\mathcal{K}_2$ bezüglich $\mathcal{K}_1$ beschreibt.

- `T = T_RPY(x)`: Die Funktion erhält den Vektor `x`$=\mathbf{x}$ gemäß 2.4 als Eingang. Zurückgegeben wird eine Transformationsmatrix

$$\mathtt{T} = \mathbf{T} = \begin{bmatrix} \mathbf{E} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{zyx}(\boldsymbol{\phi}) \end{bmatrix} \in \mathbb{R}^{6\times 6},$$

  so dass

$$\begin{bmatrix} \dot{\mathbf{d}} \\ \boldsymbol{\omega} \end{bmatrix} = \mathbf{T} \begin{bmatrix} \dot{\mathbf{d}} \\ \dot{\boldsymbol{\phi}} \end{bmatrix} = \mathbf{T}\dot{\mathbf{x}} \tag{2.5}$$

gilt. Dabei bezeichnet $\boldsymbol{\omega}$ die relative Winkelgeschwindigkeit von $\mathcal{K}_2$ gegenüber $\mathcal{K}_1$, angegeben in $\mathcal{K}_1$. Vergleichen Sie dazu auch Abschnitt 4.1.3 des Vorlesungsskriptums. Beachten Sie, dass $\mathbf{T}$ als 6×6 Matrix definiert ist.

- `dT = dT_RPY(x, dx)`: Die Funktion erhält den Vektor `x` $= \mathbf{x}$ und dessen Zeitableitung `dx` $= \dot{\mathbf{x}}$ als Eingänge. Zurückgegeben wird die Zeitableitung `dT`$= \dot{\mathbf{T}} \in \mathbb{R}^{6\times6}$ der Transformationsmatrix $\mathbf{T}$ aus (2.5).

Kartesischer Impedanzregler

Aufgabe 2.5. Vervollständigen Sie für diese Aufgabe das SIMULINK-Modell *ImpedanceController_iiwa.slx*. Es enthält bereits eine Simulation der Roboterdynamik, siehe Hinweis unter Aufgabe 2.4, und eine Visualisierung des KUKA LBR iiwa. Zum Starten des Modells können Sie das MATLAB-Skript *init_ImpedanceController.m* verwenden und dort auch alle von Ihnen benötigten Parameter ergänzen. Bearbeiten Sie folgende Aufgabenteile:

(a) Implementieren Sie einen kartesischen Impedanzregler als Subsystem im SIMULINK-Modell. Verwenden Sie mindestens $\mathbf{q}$ und $\dot{\mathbf{q}}$, die externe, kartesische Kraft $\mathbf{F}_{ext} \in \mathbb{R}^3$ auf den Werkzeugzentrumspunkt in einem geeigneten Koordinatensystem und die Solltrajektorie im Aufgabenraum $\mathbf{x}_{e,d}(t) \in \mathbb{R}^6$ mit allen nötigen Zeitableitungen als Eingänge. Ihre Implementierung sollte darüber hinaus Bewegungen des Roboters im Nullraum dämpfen.

> **Hinweis:** Achten Sie stets auf eine geeignete Wahl der benötigten Jacobi-Matrizen.

(b) Verifizieren Sie das Trajektorienfolgeverhalten und das Störverhalten Ihres Reglers. In diesem Aufgabenteil sind Sie sehr frei. Testen Sie das Verhalten Ihres Reglers ausgiebig. Erstellen Sie MATLAB-Figures Ihrer aussagekräftigen Simulationsergebnisse und speichern Sie diese für das Abgabegespräch ab. Solange die Eigenschaften Ihres Reglers in Bezug auf das Trajektorienfolgeverhalten und das Störverhalten deutlich werden, ist die Aufgabe erfüllt. Sie können sich jedoch grob an folgenden Punkten orientieren:

- Wählen Sie geeignete Startkonfigurationen `q0` $= \mathbf{q}_0 = \mathbf{q}(t=0)$ und Startgeschwindigkeiten `dq0` $= \dot{\mathbf{q}}_0 = \dot{\mathbf{q}}(t=0)$ und definieren Sie die entsprechenden Variablen im MATLAB-Skript *init_ImpedanceController.m*. Denken Sie dabei an Repräsentationssingularitäten.
- Definieren Sie die Parameter Ihres Reglers geeignet.
- Testen Sie die Lageregelung einer konstanten Sollpose $\mathbf{x}_{e,d} = \text{konst.}$. Betrachten Sie dabei die beiden Fälle $\mathbf{q}_0 = \mathbf{q}_d$ und $\mathbf{q}_0 \neq \mathbf{q}_d$, wobei sich $\mathbf{q}_d$ über die Inverskinematik aus $\mathbf{x}_{e,d}$ ergibt.

- Testen Sie die Trajektorienfolgeregelung. Definieren Sie dafür eine geeignete Trajektorie $\mathbf{x}_{e,d}(t)$.
- Führen Sie eine externe, kartesische Kraft $\mathbf{F}_{ext} \in \mathbb{R}^3$ auf den Werkzeugzentrumspunkt in einem geeigneten Koordinatensystem als Störung ein. Wählen Sie einen einfachen, gut zu analysierenden Verlauf für Ihre Kraft. Testen Sie das Trajektorienfolgeverhalten unter Einfluss der externen Kraft.

  Hinweis: Um den Einfluss der externen Kraft auf die Systemdynamik zu simulieren, müssen Sie den Vektor $\boldsymbol{\tau}_{ext}$ aus $\mathbf{F}_{ext}$ berechnen und ihn dem Block *RobotDynamics* als Eingang übergeben.

Kartesischer Nachgiebigkeitsregler (PD+-Regler)

Aufgabe 2.6. Vervollständigen Sie für diese Aufgabe das SIMULINK-Modell *PDplusController_iiwa.slx*. Wie in der vorigen Aufgabe, enthält es bereits eine Simulation der Roboterdynamik und eine Visualisierung des KUKA LBR iiwa. Zum Starten des Modells können Sie das MATLAB-Skript *init_PDplusController.m* verwenden und dort auch alle von Ihnen benötigten Parameter ergänzen. Bearbeiten Sie folgende Aufgabenteile:

(a) Implementieren Sie einen kartesischen Nachgiebigkeitsregler als Subsystem im SIMULINK-Modell. Verwenden Sie mindestens $\mathbf{q}$, $\dot{\mathbf{q}}$ und die Solltrajektorie im Aufgabenraum $\mathbf{x}_{e,d}(t) \in \mathbb{R}^6$ mit allen nötigen Zeitableitungen als Eingänge. Ihre Implementierung sollte außerdem Bewegungen des Roboters im Nullraum dämpfen.

Hinweis: Achten Sie auch bei dieser Aufgabe stets auf eine geeignete Wahl der benötigten Jacobi-Matrizen.

Hinweis: Denken Sie bei der Wahl des Regelgesetzes an die Redundanz.

(b) Verifizieren Sie das Trajektorienfolgeverhalten und das Störverhalten Ihres Reglers. Wie beim kartesischen Impedanzregler, sind Sie auch hier sehr frei. Testen Sie das Verhalten Ihres Reglers ausgiebig. Erstellen Sie MATLAB-Figures Ihrer aussagekräftigen Simulationsergebnisse und speichern Sie diese für das Abgabegespräch ab. Sie können sich an den Punkten aus Aufgabe 2.5(b) orientieren und Ihre Trajektorie wiederverwenden. Achten Sie auf Unterschiede zwischen dem kartesischen Impedanz- und Nachgiebigkeitsregler.