HTWK

Hochschule für Technik,
Wirtschaft und Kultur Leipzig

Belegarbeit im Fach Robotersteuerungen

Dozenten:

Prof. Dr.-Ing. Jens Jäkel

Robotersteuerungen Dokumentation

Vorgelegt von:	Béla Schroth
Studiengang:	Elektro- und Informationstechnik
Seminargruppe:	23-EIM ESS
Matrikelnummer:	83917
E-Mail-Adresse:	bela.schroth@stud.htwk-leipzig.de

Vorgelegt von:	Mouncef Loukili
Studiengang:	Elektro- und Informationstechnik
Seminargruppe:	23-EIM AT
Matrikelnummer:	83799
E-Mail-Adresse:	mouncef.loukili@stud.htwk-leipzig.de

Vorgelegt von:	Erik Hertwig
Studiengang:	Elektro- und Informationstechnik
Seminargruppe:	23-EIM ESS
Matrikelnummer:	75029
E-Mail-Adresse:	erik.hertwig@stud.htwk-leipzig.de

12. März 2025

Einleitung

Dieses Dokument dient zur strukturierten Bearbeitung und Dokumentation der gestellten Aufgaben im Rahmen der Analyse und Simulation des Roboterarms KUKA LBR iiwa 14 R820 7-axis robot. Für jede Aufgabe wird eine ausführliche Erklärung bereitgestellt, einschließlich einer detaillierten Dokumentation der Antworten sowie einer grafischen Darstellung relevanter Diagramme.
Alle MATLAB- und Simulink-Skripte, die zur Lösung der Aufgaben verwendet wurden, sind in einem GitHub-Repository verfügbar. Der gesamte

Quellcode kann unter folgendem Link gefunden werden: `https://github.com/bschroth2011/RobotSteuerung_WS24.git`

Das Dokument ist wie folgt strukturiert:

- Jedes Kapitel behandelt eine spezifische Aufgabe.
- Jede Aufgabe enthält eine detaillierte Erklärung der Herangehensweise.
- Falls erforderlich, werden Diagramme und Abbildungen zur Veranschaulichung der Ergebnisse eingefügt.
- Das jeweilige MATLAB- oder Simulink-Skript wird erwähnt und sein Speicherort angegeben.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 URDF-Datei des Robotermodells

1.1 Die URDF des Roboters

Um die allgemeinen Roboter-Eigenschaften zu beschreiben, wird in der Regel ein sogenanntes URDF-File (Unified Robotics Description Format) benutzt, allgemein für die Modellierung von Mehrkörpersystemen benutzt wird. Die genauen Informationen werden im Rahmen dieser Aufgabe beantwortet.

1.2 Name des Roboters

Der im Rahmen unseres Projektes benutzte Roboter ist der **Kuka_lbr_iiwa_14_r820**. Er besitzt eine maximale Reichweite von 820 mm, eine maximale Traglast von 14 kg und verfügt über 7 Achsen (Degrees of Freedom, **DOF** = 7).

Dank seiner integrierten Sensorik ist er in der Lage, feine Bewegungen und Berührungen zu erkennen, was ihn beispielsweise für Anwendungen wie das **Messen**, **Beschichten** oder andere Applikationen in der **Materialbearbeitung** besonders geeignet macht.

1.3 Inhalte des URDF-Files

Das URDF-File beschreibt die Gelenke sowie die Armsegmente und deren Parameter unseres Roboters. (3)

In unserem URDF, welches im GitHub-Repository unter `Aufgabe1/lbr_iiwa_14_r820.urdf` gespeichert ist, werden nach Angabe des Roboternamens die Armsegmente (Links) sowie die Gelenke (Joints) in der Reihenfolge vom **Base-Link** bis hin zum **Endeffektor** initialisiert und beschrieben. Zuerst sind alle **Links** aufgeführt, danach folgen alle **Joints**.

Das URDF-File und somit alle enthaltenen Informationen wurden mithilfe der Sprache **xacro** und der Datei `lbr_iiwa_14_r820.xacro` generiert.

1.4 Parameter der Koordinatensysteme

Danach werden die Joints bezüglich ihres Namens und Typs (hier: revolute, rotierend) sowie ihrer Ursprungsparameter wie Position (x, y, z) in Metern und Orientierung (Roll, Pitch, Yaw) in Radiant deklariert. Außerdem wird

das zuvor angebrachte Robotersegment angegeben (child → Gelenk, parent → vorheriges Segment). Zusätzlich werden die Achsen, um die sich die Gelenke drehen, sowie die gelenkspezifischen Grenzwerte für die einstellbaren Winkel in Radiant und für die Winkelgeschwindigkeit in rad/s beschrieben (siehe Datenblatt (2) oder Umrechnung für °). Zu jedem Teil wird auch ein Mesh für ein 3D-Modell zur Anzeige innerhalb der Simulation angegeben.

1.5 kinematisches sowie dynamisches Modell

Mithilfe der Parameter x, y, z, Roll, Pitch, Yaw des URDF-Files innerhalb der ¡joint¿ lässt sich für unseren Roboter das kinematische Modell bilden. Für das dynamische Modell jedoch fehlen Parameter wie die Masse der Segmente oder deren Schwerpunkte.

1.6 kinematische und dynamische Beschränkungen

Für beide Modelle ist es jedoch wichtig, die unter ¡limits¿ angegebenen Einschränkungen einzuhalten, um keine Schäden an der Umgebung oder dem Roboter selbst zu verursachen.
Zu den kinematischen Einschränkungen gehören z.B. die Gelenkwinkelgrenzen sowie die Gelenkgeschwindigkeitsgrenzen oder die Arbeitsraumbegrenzung (Reichweite des Roboters). Die dynamischen Einschränkungen belaufen sich auf z.B. die maximale Last sowie die Grenzen der Trägheit des Roboters oder der Last.
In dem URDF-File sind unter Limits nur die jeweiligen kinematischen Beschränkungen der Gelenke vorzufinden. Jedoch kann man aus dem Datenblatt des Roboters die maximale Arbeitslast nehmen, sowie mit dem im folgenden Kapitel betrachteten RigitBodyTree-Objekt und deren Zusatzinformationen wie die Trägheit oder Masse der einzelnen Segmente wie in Abb. 1 zu sehen ablesen.

2 Analyse des Robotermodells

Für die Simulierung des Roboters kann in Matlab durch den Befehl loadrobot der gewünschte Roboter in Form eines RigitBodyTree-Objektes (RBT-Objekt) geladen werden. Dieses beinhaltet in Kombination mit der Robot-Systems-Toolbox alles, was für die Darstellung Modellierung und Regelung des Roboters im Rahmen des Projektes benötigt wird. Zusätzlich kann man sich auch die zusätzlichen (Im Projekt in der Variable DatenRobotdata) Informationen ausgeben lassen, welche im ManipulatorMotionModel weitere

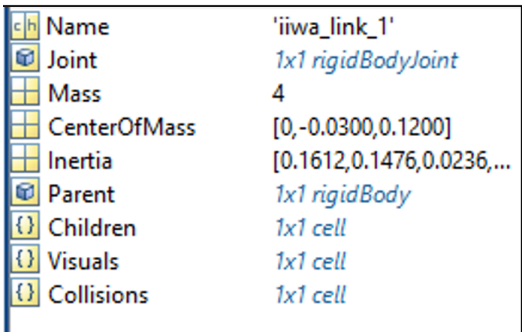


Abbildung 1: dynamische Eigenschaften Joint 1

dynamische Eigenschafften besitzt.

2.1 Aufbau des Objektes und deren wesentlichen Objekte

In der folgenden Tabelle werden die einzelnen Teile des RTB-Objektes benannt und beschrieben. Dabei ist zu beachten, dass sich alle Angaben auf den **Kuka_lbr_iiwa_14_r820** beziehen.

Teil	Wert	Funktion
NumBodies	10	Anzahl der Körper
Bodies	1x10 Cell	Beinhaltet einzelne Körper -> Infos zu Armsegmenten und Gelenken
Base	1x1 rigidBody	Beschreibt Infos des Basis-Körpers
BodyNames	1x10 Cell	Liste der Körper-Namen
BaseName	‚World'	Name des Basiskörpers
Gravity	[0,0,0]	Eingestellte Schwerkraft (hier Standardwert)
DataFormat	column	Form des RBT-Objektes

Abbildung 2: Aufbau des RTB-Objektes

2.2 Gelenkparameter, Transformationsmatrizen und Begrenzungen

Eine der wichtigsten Inhalte des RTB-Objektes sind die Informationen über die **Armsegmente und Gelenke**. Diese können unter folgendem Pfad in dem RTB-Objekt in Matlab gefunden werden:

- `Robot -> Bodies {1, 2 bis 8}` für Armsegment-Drehgelenk-Paar → Joint
- oder: `Robot.Bodies{1,3}.Joint`

Hier stehen die **gesammelten Gelenkparameter**, welche in **folgender Abbildung 3** noch einmal gezeigt werden.

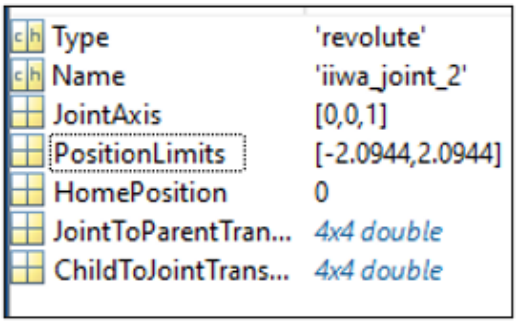
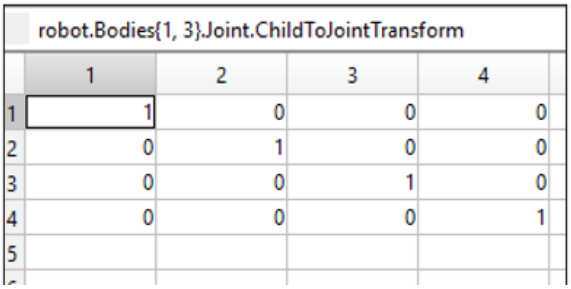


Abbildung 3: 2 Gelenkparameter im RBT-Objekt

Dabei wird hier der Typ, in unserem Fall rotierend, der Name der Achse, um die sich das Gelenk dreht, Die Gelenkbegrenzungen in Rad, die Heimatposition in Rad, sowie die Transformationsmatrizen für die Kopplung des Gelenkes an die angrenzenden Armsegmente und die kinematische Modellierung. Die Transformationsmatrizen liegen in der Form 4x4 vor, wie in Abbildung 3 beispielhaft visualisiert



	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5				

Abbildung 4: Anbindung folgendes Armsegment an Achse 2

Die Begrenzung der Achsengeschwindigkeiten sind nicht in dem RBT-Objekt enthalten und müssen deswegen extern recherchiert (z.B. URDF-File oder Datenblatt) und eingetragen werden.

3 Numerischer inverser Kinematik

Die Konfiguration in einer Singularität wurde zuerst ausgewählt. Dabei wurde eine Ellenbogen-Singularität gesucht. Diese ließ sich durch einfaches Äusstrecken"des Roboterarms realisieren. Hierfür wurde ein Punkt angefahren, der genau über dem Ursprung, also der Basis liegt. Die Höhe des Punktes ist genau gleich der Länge des gestreckten Roboterarms. In den beiden Abbildungen in Figure 5 ist diese Konfiguration zu erkennen. Als Singularität bestätigt wurde diese Pose mittels des Matlab-Skripts Aufgabe3/drei_inverse_kinematik.m, da die Berechnungsdauer von 0,9666 Sekunden bei 5000 Iterationen lag, was als maximale Anzahl definiert wurde. Die Endeffektorposition liegt bei (0; 0; 1.31) (x; y; z) und seine Orientierung ist (0; -1.57; 0) (Roll; Pitch; Yaw).

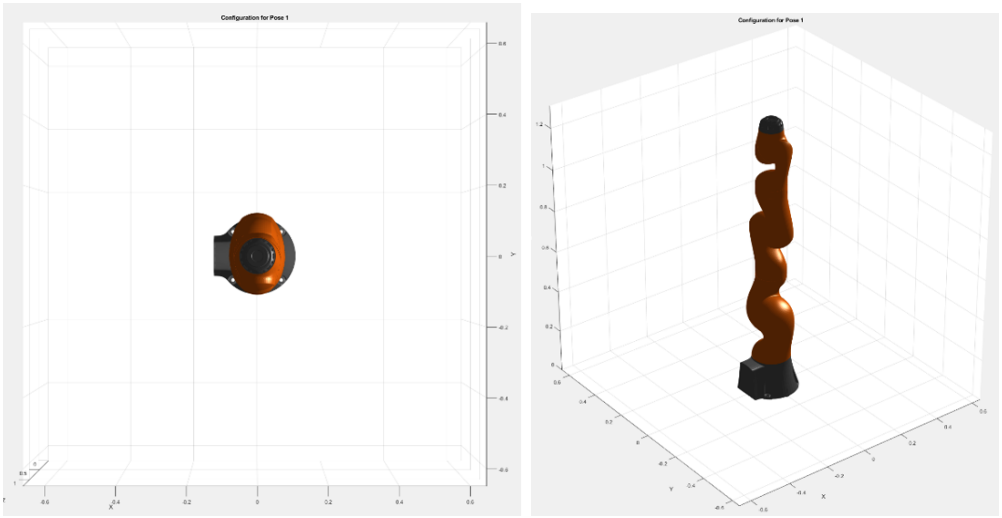


Abbildung 5: Links: Singularität x-y, Rechts: Singularität isometrisch

Von dieser ersten Konfiguration wurde eine weitere abgeleitet, welche sich möglichst nahe an einer Singularität befinden sollte. Hierbei wurde der Zielpunkt weiterhin direkt über der Basis definiert, jedoch diesmal mit einer Höhe geringfügig niedriger als die Länge des ausgestreckten Arms. Somit muss der Roboter leicht einknicken, wie in Figure 6 zu sehen. Die Ermittlung der genauen Position ist allerdings nicht deterministisch, da es theoretisch für diese Konfiguration, wie bei Konfiguration 1, unendlich viele Lösungen gibt. Das führt zu stark variierenden Berechnungsdauern, die meist zwischen ca. 0,0327 Sek. bei 162 Iterationen und 0,1117 Sek. bei 1226 Iterationen liegen. Figure 6 zeigt ein Beispiel für eine Berechnung von 0,1934 Sek. bei 985 Iterationen. Die Endeffektorposition liegt bei (0; 0; 1.30) (x; y; z) und

seine Orientierung ist (0; -1.57; 0) (Roll; Pitch; Yaw).

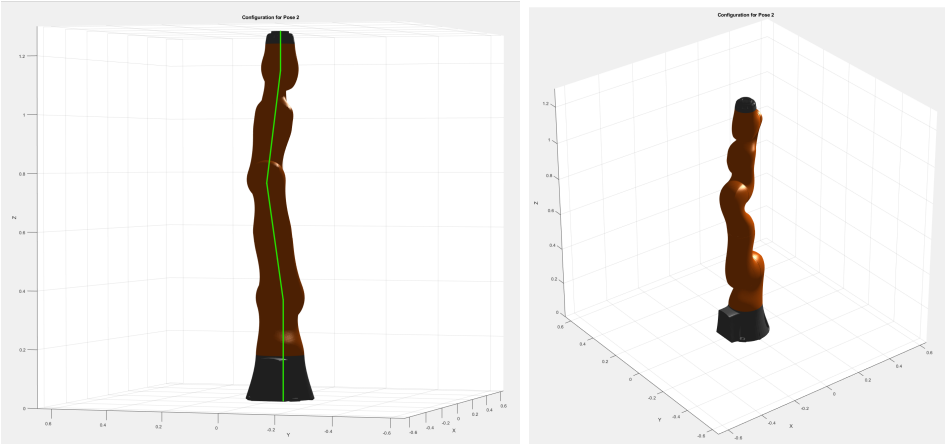


Abbildung 6: Links: Fast-Singularität seitlich, Rechts: Fast-Singularität isometrisch

Die dritte und letzte Konfiguration, welche sich weder in noch nah an einer Singularität befindet, wurde durch Ausprobieren und unter Beachtung der Regeln für Singularitäten ermittelt. Sie befindet sich möglichst weit weg von Ellenbogen-, Handgelenk- und Basis-Singularitäten und ist in Abbildung 8 zu sehen. Die Berechnungsdauer betrug 0,0065 Sek. bei 28 Iterationen. Diese Werte waren in allen Versuchen ähnlich. Die Endeffektorposition liegt bei (0.2; 0.6; 0.4) (x; y; z) und seine Orientierung ist (1.57; 0; 0) (Roll; Pitch; Yaw).

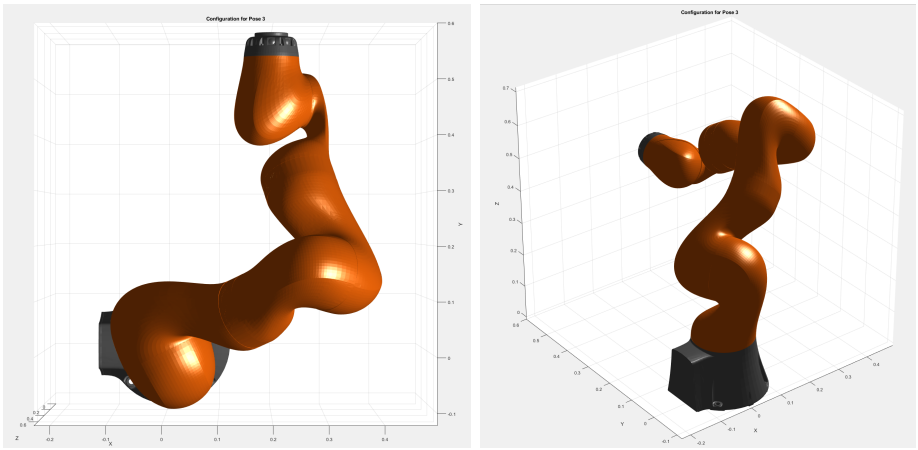


Abbildung 7: Links: Nicht-Singularität x-y, Rechts: Nicht-Singularität isometrisch

3.1 Fazit

In Abbildung (8) sind alle erwähnten Konfigurationen zusammengefasst.

```
% Define end-effector poses [X, Y, Z, Roll, Pitch, Yaw] (angles in radians)
poses = [0.0, 0.0, 1.4, 0, -pi/2, 0;    % Pose 1
         0.0, 0.0, 1.3, 0, -pi/2, 0;    % Pose 2
         0.2, 0.6, 0.4, pi/2, 0, 0];    % Pose 3
%poses = Pose;
```

Abbildung 8: Konfigurationen

In Singularitäten steigen sowohl die Berechnungszeit als auch die Anzahl der Iterationen scheinbar gegen unendlich. Während eine leichte Verschiebung der Endeffektorlage diese Werte verringern kann, bleibt die tatsächlich benötigte Berechnungszeit im Voraus schwer abschätzbar. Eine Position, die sich klar außerhalb einer Singularität befindet, benötigt hingegen stets sehr ähnliche Berechnungszeiten und erfordert generell einen wesentlich geringeren Ressourcenaufwand.

4 Analytische Inverse Kinematik

Die Analytische Inverse Kinematik (AIK) für den KUKA LBR iiwa 14 ermöglicht die direkte Berechnung der Gelenkwinkel mittels trigonometrischer und algebraischer Methoden, im Gegensatz zur numerischen IK, die iterative Näherungsverfahren verwendet. Die Voraussetzung für Diese ist das bestehen sogenannter kinematischer Gruppen, welche innerhalb von Matlab angezeigt und ausgewählt werden können, solange Sie existieren. Die Auswahl für unseren Roboter sind als Bild unter Github Aufgabe4 einzusehen. Die verschiedenen Gruppen haben dabei einen variablen Endeffektor

4.1 Definition der DH-Parameter

Zur Beschreibung der Kinematik des Roboters definieren wir die Denavit-Hartenberg (DH)-Parameter, die die Gliederlängen und Verdrehungen enthalten. Diese Parameter sind die Grundlage für die Berechnung der Vorwärts- und Inversen Kinematik.
Die gewünschte Endeffektorpose ist gegeben durch:

- **Position:** $X = 0.2$, $Y = 0.6$, $Z = 0.4$.
- **Orientierung:** Roll $= \frac{\pi}{2}$, Pitch $= 0$, Yaw $= 0$.

4.2 Berechnung der Gelenkwinkel

4.2.1 Gelenk 1 (Basisrotation)

Der erste Gelenkwinkel wird berechnet mit:

$$q_1 = \tan^{-1}\left(\frac{P_y}{P_x}\right) \tag{1}$$

Dies bestimmt die notwendige Rotation in der XY-Ebene.

4.2.2 Gelenke 2 und 3 (Armkonfiguration in der XZ-Ebene)

Zur Berechnung von q_2 und q_3:

- Berechnung der projizierten Distanz in der XY-Ebene:

$$r = \sqrt{P_x^2 + P_y^2} \tag{2}$$

- Berechnung der Höhendifferenz von der Basis:

$$s = P_z - d_1 \tag{3}$$

- Anwendung des **Kosinussatzes** zur Bestimmung von q_3:

$$\cos(q_3) = \frac{r^2 + s^2 - L_1^2 - L_2^2}{2L_1L_2} \tag{4}$$

- Sicherstellung, dass $\cos(q_3)$ im gültigen Bereich $[-1, 1]$ liegt, und Berechnung von:

$$q_3 = \tan^{-1}\left(\frac{\sqrt{1 - \cos^2(q_3)}}{\cos(q_3)}\right) \tag{5}$$

- Berechnung von q_2:

$$q_2 = \tan^{-1}\left(\frac{s}{r}\right) - \tan^{-1}\left(\frac{L_2 \sin(q_3)}{L_1 + L_2 \cos(q_3)}\right) \tag{6}$$

4.2.3 Gelenke 4, 5 und 6 (Endeffektor-Orientierung)

Die gewünschte Endeffektor-Orientierung wird in eine Rotationsmatrix umgewandelt:

$$R_{06} = R_z(\text{roll})R_y(\text{pitch})R_x(\text{yaw}) \tag{7}$$

Dann berechnen wir die Rotationsmatrix für die ersten drei Gelenke:

$$R_{03} = R_z(q_1)R_y(q_2)R_y(q_3) \tag{8}$$

Daraus extrahieren wir:

$$R_{36} = R_{03}^{-1}R_{06} \tag{9}$$

Berechnung von q_4, q_5 und q_6:

$$q_4 = \tan^{-1}\left(\frac{R_{36}(2,3)}{R_{36}(1,3)}\right) \tag{10}$$

$$q_5 = \tan^{-1}\left(\frac{\sqrt{1 - R_{36}(3,3)^2}}{R_{36}(3,3)}\right) \tag{11}$$

$$q_6 = \tan^{-1}\left(\frac{R_{36}(3,2)}{-R_{36}(3,1)}\right) \tag{12}$$

4.2.4 Gelenk 7 (Redundanzauflösung)

Da der KUKA LBR iiwa 14 sieben Freiheitsgrade besitzt, bleibt ein zusätzlicher Freiheitsgrad bestehen. Zur Lösung dieser Redundanz setzen wir:

$$q_1 = 0 \tag{13}$$

Dieser Wert kann weiter optimiert werden, je nach Ellbogen-oben- oder Ellbogen-unten-Konfiguration.

4.3 Fazit

Die analytische inverse Kinematik berechnet für die dritte Konfiguration (ohne Singularität) aus Aufgabe 3 [0.2, 0.6, 0.4, pi/2, 0, 0] mehrere Lösungskonfigurationen. die verschiedenen Lösungen des Endeffektors sind im Github unter Aufgabe4 zu finden. Die Anzahl der Lösungen ist dabei von Position zu Position des Endeffektors unterschiedlich.

5 Dynamische Modellierung des Roboters

5.1 Bestimmung der verallgemeinerten Schwerkraft

Zur Untersuchung der Gravitationsmomente wurde das MATLAB-Skript `min_max_torques.m` verwendet, das die Schwerkraftmomente für zwei unterschiedliche Konfigurationen des Roboterarms berechnet. Das Skript kann in folgendem Verzeichnis gefunden werden:
`Aufgabe5/min_max_torques.m`

- **Vertikale Konfiguration:** Der Arm ist nach oben ausgerichtet (Home-Position).
- **Horizontale Konfiguration:** Der Arm ist gestreckt und parallel zum Boden ausgerichtet.

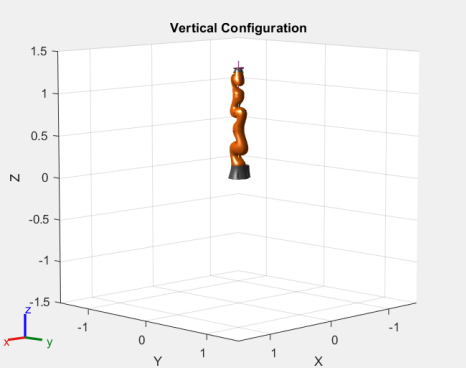


Abbildung 9: Vertikale Konfiguration des Roboterarms

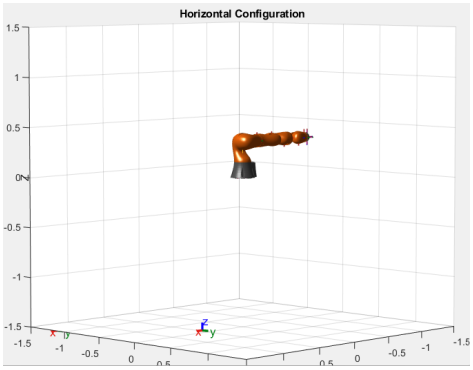


Abbildung 10: Horizontale Konfiguration des Roboterarms

- **Vertikale Konfiguration:** Geringe Drehmomente, da die Schwerkraft direkt nach unten wirkt und kaum eine Drehung um die Gelenke verursacht.
- **Horizontale Konfiguration:** Höhere Drehmomente, insbesondere an bestimmten Gelenken:
  - **Gelenk 2**: Erfordert ein hohes Drehmoment, da es das gesamte Gewicht des ausgestreckten Arms trägt.
  - **Gelenk 4**: Erzeugt ein signifikantes Gegendrehmoment, um einen Teil des Armgewichts auszugleichen.

Die berechneten Minimal- und Maximalwerte für die Drehmomente werden in folgender Tabelle dargestellt:

Vertikale Konfiguration	Horizontale Konfiguration
-0.0000	0
0.0134	53.6080
-0.0000	0.3608
-0.0017	-14.6672
-0.0000	0.3432
-0.0000	0.3078
0	0

Tabelle 1: Gravitationsmomente für vertikale und horizontale Konfigurationen

Gelenk	Min. Drehmoment (Nm)	Max. Drehmoment (Nm)
1	0.0000	0.0000
2	0.0134	53.6000
3	-0.0000	0.3608
4	-14.6672	0.3432
5	-0.0000	0.3078
6	-0.0000	0.0000
7	0.0000	0.0000

Tabelle 2: Berechnete minimale und maximale Drehmomente für die Gelenke

- **Hebelwirkung der Schwerkraft**: Je länger der Hebelarm, desto höher das erforderliche Drehmoment. In der horizontalen Konfiguration muss Gelenk 2 ein hohes Drehmoment aufbringen, um das gesamte Armgewicht zu tragen.
- **Vergleich der Konfigurationen**: Während die vertikale Position nur geringe Drehmomente benötigt, steigen sie in der horizontalen Position deutlich an.
- **Einfluss auf die Regelung**: Die Ergebnisse zeigen, dass in bestimmten Positionen des Roboters deutlich höhere Momente notwendig sind, was bei der Reglerauslegung berücksichtigt werden muss.

5.2 Untersuchung der verallgemeinerten Massenmatrix

Zur Extraktion der Massenmatrix wurde das MATLAB-Skript `aufgabe5/Mass_matrix.m` verwendet. Die Berechnung erfolgt mit der

Methode `massMatrix(.)`.
Die verallgemeinerte Massenmatrix $M(q)$ beschreibt die Trägheitseigenschaften des Roboters. Sie gibt an, wie schwer es ist, die einzelnen Gelenke zu bewegen. Mathematisch kann sie wie folgt dargestellt werden:

$$M(q) = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} & M_{16} & M_{17} \\ M_{21} & M_{22} & M_{23} & M_{24} & M_{25} & M_{26} & M_{27} \\ M_{31} & M_{32} & M_{33} & M_{34} & M_{35} & M_{36} & M_{37} \\ M_{41} & M_{42} & M_{43} & M_{44} & M_{45} & M_{46} & M_{47} \\ M_{51} & M_{52} & M_{53} & M_{54} & M_{55} & M_{56} & M_{57} \\ M_{61} & M_{62} & M_{63} & M_{64} & M_{65} & M_{66} & M_{67} \\ M_{71} & M_{72} & M_{73} & M_{74} & M_{75} & M_{76} & M_{77} \end{bmatrix} \tag{14}$$

Die **diagonalen Elemente** M_{ii} geben an, wie schwer es ist, ein einzelnes Gelenk i zu bewegen (Trägheit). Die **nicht-diagonalen Elemente** M_{ij} $(i \neq j)$ zeigen, wie die Bewegung eines Gelenks die Bewegung eines anderen Gelenks beeinflusst.

5.2.1 Ergebnisse für verschiedene Konfigurationen

Die Massenmatrix wurde für drei verschiedene Konfigurationen des Roboters berechnet:

- **Konfiguration 1 (Vertikal):** Der Arm ist in der vertikalen Standardposition.
- **Konfiguration 2 (Horizontal):** Der Arm ist ausgestreckt in horizontaler Lage.
- **Konfiguration 3 (Zwischenposition):** Eine Zwischenstellung des Arms.

5.2.2 Massenmatrix für Konfiguration 1 (Vertikal)

Die Massenmatrix für die vertikale Konfiguration ist:

$$M_{\text{vertikal}} = \begin{bmatrix} 0.0848 & 0.0284 & 0.0362 & -0.0152 & 0.0104 & -0.0000 & 0.0010 \\ 0.0284 & 3.4155 & 0.0383 & -1.1723 & 0.0237 & 0.0345 & 0.0000 \\ 0.0362 & 0.0383 & 0.0362 & -0.0152 & 0.0104 & -0.0000 & 0.0010 \\ -0.0152 & -1.1723 & -0.0152 & 0.5443 & -0.0090 & -0.0213 & 0.0000 \\ 0.0104 & 0.0237 & 0.0104 & -0.0090 & 0.0104 & -0.0000 & 0.0010 \\ -0.0000 & 0.0345 & -0.0000 & -0.0213 & -0.0000 & 0.0088 & -0.0000 \\ 0.0010 & 0.0000 & 0.0010 & 0.0000 & 0.0010 & -0.0000 & 0.0010 \end{bmatrix} \tag{15}$$

5.2.3 Massenmatrix für Konfiguration 2 (Horizontal)

Die Massenmatrix für die horizontale Konfiguration ist:

$$M_{\text{horizontal}} = \begin{bmatrix} 3.4670 & 0.0000 & 0.0001 & 0.0000 & 0.0001 & 0.0000 & 0.0000 \\ 0.0000 & 3.4155 & 0.0383 & -1.1723 & 0.0237 & 0.0345 & 0.0000 \\ 0.0001 & 0.0383 & 0.0362 & -0.0152 & 0.0104 & -0.0000 & 0.0010 \\ 0.0000 & -1.1723 & -0.0152 & 0.5443 & -0.0090 & -0.0213 & 0.0000 \\ 0.0001 & 0.0237 & 0.0104 & -0.0090 & 0.0104 & -0.0000 & 0.0010 \\ 0.0000 & 0.0345 & -0.0000 & -0.0213 & -0.0000 & 0.0088 & -0.0000 \\ 0.0000 & 0.0000 & 0.0010 & 0.0000 & 0.0010 & -0.0000 & 0.0010 \end{bmatrix} \tag{16}$$

5.2.4 Massenmatrix für Konfiguration 3 (Zwischenposition)

Die Massenmatrix für die Zwischenposition ist:

$$M_{\text{Intermediate}} = \begin{bmatrix} 1.7562 & 0.0211 & 0.0139 & 0.5773 & 0.0029 & -0.0063 & 0.0007 \\ 0.0211 & 3.4216 & 0.0332 & -0.8285 & 0.0229 & 0.0333 & 0.0000 \\ 0.0139 & 0.0332 & 0.0362 & -0.0140 & 0.0104 & -0.0000 & 0.0010 \\ 0.5773 & -0.8285 & -0.0140 & 0.5451 & -0.0078 & -0.0185 & 0.0000 \\ 0.0029 & 0.0229 & 0.0104 & -0.0078 & 0.0104 & -0.0000 & 0.0010 \\ -0.0063 & 0.0333 & -0.0000 & -0.0185 & -0.0000 & 0.0088 & -0.0000 \\ 0.0007 & 0.0000 & 0.0010 & 0.0000 & 0.0010 & -0.0000 & 0.0010 \end{bmatrix} \tag{17}$$

Die Massenmatrix wurde für drei verschiedene Konfigurationen des Roboters analysiert. Die wichtigsten Beobachtungen sind wie folgt:

- **Konfiguration 1: Vertikal (Home Position)**

1. Das erste Gelenk (**Schulter**) hat eine sehr geringe Trägheit ($0.0848\,\mathrm{kg} \cdot \mathrm{m}^2$), da es fast mit der Schwerkraft ausgerichtet ist.
2. **Gelenk 2** ($3.4155\,\mathrm{kg}{\cdot}\mathrm{m}^2$) hat die höchste Trägheit und trägt somit den größten Teil der Masse.
3. **Gelenk 4 ist stark mit Gelenk 2 gekoppelt** (-1.1723), d.h. eine Bewegung in Gelenk 4 beeinflusst Gelenk 2 erheblich.

- **Konfiguration 2: Horizontal (Arm ausgestreckt)**

  1. **Gelenk 2 und Gelenk 4 sind weiterhin stark gekoppelt** ($M_{24} = -1.1723$).
  2. Andere Gelenke haben nahezu keine Kopplung (**die Nicht-Diagonalelemente sind fast null**). Dies bedeutet, dass sich die Gelenke unabhängiger bewegen, wenn der Arm vollständig ausgestreckt ist.

- **Konfiguration 3: Zwischenposition**

  1. **Gelenk 1 und Gelenk 4 zeigen nun eine spürbare Kopplung** ($M_{14} = 0.5773$).
  2. **Gelenk 2 und Gelenk 4 bleiben stark gekoppelt** ($M_{24} = -0.8285$).
  3. **Die Kopplung ist im Vergleich zur horizontalen Konfiguration weiter verteilt**.

5.2.5 Interpretation der Ergebnisse

- **Gelenk 2 und Gelenk 4 sind in allen Konfigurationen stark gekoppelt.** Wenn sich Gelenk 2 bewegt, beeinflusst es Gelenk 4 und umgekehrt.

- **In der horizontalen Konfiguration sind die Gelenke unabhängiger**, da die meisten Nicht-Diagonalelemente nahezu null sind.

- **In der Zwischenposition ist die Kopplung komplexer**, d.h. Bewegungen in einem Gelenk beeinflussen mehrere Gelenke.

- **Die Kopplung bleibt für jede Konfiguration gleich, während sich die Diagonalelemente der Massenmatrix aufgrund des Hebelgesetzes der Schwerkraft ändern.**

5.3 Analyse der Euler- und Coriolis-Kräfte

Zur Untersuchung der Euler- und Coriolis-Kräfte wurde das MATLAB-Skript `aufgabe5/Corioli.m` verwendet. Die Ergebnisse zeigen die folgenden Eigenschaften:
1. Größenordnung der Kräfte: Die Werte der **Euler-Kräfte** (bedingt durch Trägheit und Beschleunigung) liegen hauptsächlich zwischen **0.0003 Nm und 1.1654 Nm**.

- Die größte Kraft tritt bei **Gelenk 2** auf (**1.1654 Nm**), während die geringsten Werte bei **Gelenk 6 und 7** (**0.0003–0.0018 Nm**) zu finden sind.
- Dies zeigt, dass einige Gelenke eine wesentlich größere Trägheit haben als andere.

Die **Coriolis-Kräfte** hängen von der Geschwindigkeit ab:

- Für $\dot{q} = 0.3$ rad/s liegen sie zwischen **0.0001 Nm und 0.1924 Nm**.
- Für $\dot{q} = 0.6$ rad/s steigen sie auf **0.0004 Nm bis 0.7695 Nm**.
- Für $\dot{q} = 0.9$ rad/s erreichen sie maximal **1.7314 Nm**.
- Dies bestätigt die **quadratische Abhängigkeit** der Coriolis-Kräfte von der Geschwindigkeit.

2. Kopplung zwischen den Achsen: Ja, insbesondere **Gelenk 2 und Gelenk 4** zeigen starke Kopplungseffekte:

- **Gelenk 2** weist die höchsten Euler- und Coriolis-Kräfte auf, was auf eine hohe Trägheit und starke Kopplung mit anderen Gelenken hindeutet.
- **Gelenk 4** zeigt ebenfalls signifikante Coriolis-Kräfte mit einem **negativen Vorzeichen**, was bedeutet, dass es entgegengesetzt zu anderen Gelenken gekoppelt ist.
- **Gelenke 6 und 7** sind kaum beeinflusst, da ihre Kräfte fast null sind – sie sind also wenig mit anderen Achsen gekoppelt.

Fazit:

- **Stärkste Kopplung:** Gelenk **2 und 4**.
- **Kaum beeinflusst:** Gelenk **6 und 7**.
- **Größenordnung der Kräfte:** Euler-Kräfte bis **1.16 Nm**, Coriolis-Kräfte bis **1.73 Nm** (bei hoher Geschwindigkeit).

5.4 Einfluss externer Kräfte auf die Gelenkmomente

In dieser Untersuchung wird analysiert, wie eine externe Kraft, die auf den Endeffektor wirkt, die Gelenkmomente beeinflusst. Dazu wurde das MATLAB-Skript `aufgabe5/Ext_Force.m` verwendet, um die entsprechenden Momente in den Gelenken zu berechnen.

Berechnungsmethode: Die Gelenkmomente τ_{ext} aufgrund externer Kräfte F_{ext} werden mit der transponierten Jacobi-Matrix J^T berechnet:

$$\tau_{\text{ext}} = J^T \cdot F_{\text{ext}} \tag{18}$$

Dabei gilt:

- τ_{ext} sind die durch externe Kräfte verursachten Gelenkmomente,
- J ist die geometrische Jacobi-Matrix des Roboters,
- F_{ext} ist die auf den Endeffektor einwirkende Kraft.

Angewandte Konfiguration: Die Berechnung wurde für die folgende Gelenkkonfiguration durchgeführt:

$$q = [0.1; -0.5; 0.3; -0.1; 0.2; -0.4; 0.5] \tag{19}$$

Eine externe Kraft wurde in Y-Richtung auf den Endeffektor angewendet:

$$F_{\text{ext}} = \begin{bmatrix} 0 \\ 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{20}$$

5.4.1 Ergebnisse der Berechnung

Die berechneten Gelenkmomente aufgrund der äußeren Kraft sind:

$$\tau_{\text{ext}} = \begin{bmatrix} -0.0000 \\ 4.9750 \\ -0.2393 \\ -4.6234 \\ -0.0496 \\ 4.1531 \\ -1.1297 \end{bmatrix} \tag{21}$$

5.4.2 Interpretation der Ergebnisse

Nach Berücksichtigung der Hauptachsen der Gelenke aus der URDF-Datei lassen sich folgende Beobachtungen machen:

- **Eine Kraft in Y-Richtung beeinflusst besonders Gelenke, deren Hauptachse mit der Y-Richtung übereinstimmt.** Dies betrifft insbesondere **Gelenk 2, 4 und 6**, die um die Y-Achse rotieren.
- **Gelenk 2 erfährt das höchste Drehmoment (4.9750 Nm)**, da es eine der Hauptachsen ist, die mit der Bewegung des Endeffektors in der Y-Richtung verbunden ist.
- **Gelenk 4 zeigt eine negative Reaktion (-4.6234 Nm)**, was darauf hinweist, dass es entgegenwirken muss, um das System stabil zu halten.
- **Gelenk 6 erfährt ebenfalls ein hohes Moment (4.1531 Nm)**, was bedeutet, dass diese Achse eine große Rolle bei der Kraftkompensation spielt.
- **Das letzte Gelenk (Gelenk 7) hat eine vergleichsweise hohe Reaktion (-1.1297 Nm), obwohl es um die Z-Achse rotiert.** Dies könnte daran liegen, dass es sich nahe der Kraftanwendungsstelle befindet.
- **Geringe Momente an Gelenk 1 und 5**, da sie weniger mit der Y-Richtungsbewegung des Endeffektors verbunden sind.

5.4.3 Zusammenfassung

- Die Haupteffekte treten in **Gelenk 2, 4 und 6** auf, da sie direkt mit der Y-Richtung verknüpft sind
- Gelenk 7 zeigt ebenfalls ein relevantes Moment, möglicherweise aufgrund seiner Nähe zur Kraftquelle
- **Gelenk 1 und 5 sind weniger betroffen**, da sie nicht direkt mit der Y-Richtung gekoppelt sind

5.5 Berücksichtigung von Reibungsmomenten im Modell

Das MATLAB-Robotermodell `loadrobot('kukaIiwa14')` enthält standardmäßig **keine Reibungsmomente**. Das bedeutet, dass die Berechnungen der Gelenkmomente unter der Annahme eines **reibungsfreien Systems** durchgeführt wurden.

5.5.1 Zusammenfassung

- **Falls erforderlich, kann sie in MATLAB explizit hinzugefügt werden**, indem zusätzliche Reibungsmomente in die Bewegungsgleichungen integriert werden.
- **Die Implementierung kann über eine externe Kraft oder ein angepasstes Reglermodell erfolgen.**

Die Reibung kann dabei modelliert werden als:

$$\tau_{\text{friction}} = F_c \cdot \text{sign}(\dot{q}) + F_v \cdot \dot{q} \tag{22}$$

Dabei sind:

- F_c die Coulomb-Reibung,
- F_v die viskose Reibung,
- $\text{sign}(\dot{q})$ das Vorzeichen der Geschwindigkeit.

5.6 Simulation der Dynamik in Simulink

In dieser Aufgabe wurde untersucht, wie sich der Roboter aufgrund der Gravitation verhält, wenn keine Antriebsmomente wirken. Dazu wurde eine **Simulink-Forward-Dynamics-Simulation** durchgeführt.

Steuerung der Gravitation Um die Simulation sowohl mit als auch ohne Gravitation durchzuführen, wurde der folgende MATLAB-Befehl verwendet:

- `evalin('base', 'robot.Gravity = [0; 0; 0];')` → **Deaktiviert die Gravitation**.
- `evalin('base', 'robot.Gravity = [0; 0; -9.81];')` → **Aktiviert die Gravitation**.

Durch diesen Befehl wird sichergestellt, dass die Gravitation entweder entfernt oder angewendet wird, um den Einfluss auf die Roboterdynamik zu analysieren.

Simulationssetup

- Die Roboterkonfiguration wurde auf $\mathbf{q} = [0,0,0,0,0,0,0]$ gesetzt.
- Alle **Antriebsmomente wurden deaktiviert** ($\tau = 0$).
- Die Simulation wurde mit dem **Simulink Forward Dynamics Block** durchgeführt.
- Zwei Szenarien wurden getestet:
  1. **Ohne Gravitation:** `robot.Gravity = [0; 0; 0]`
  2. **Mit Erdanziehungskraft:** `robot.Gravity = [0; 0; -9.81]`

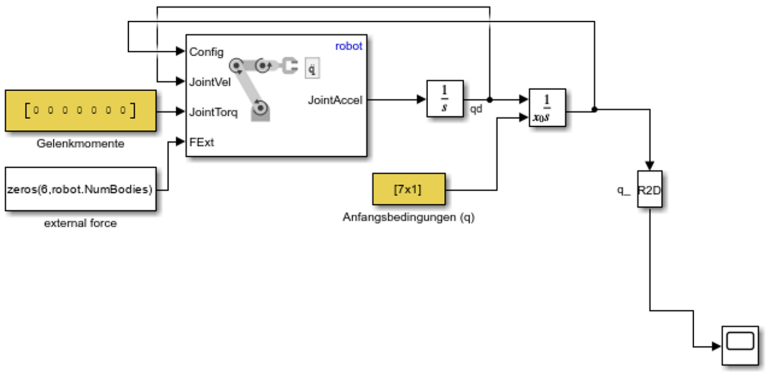


Abbildung 11: Simulink-Blockdiagramm für die Forward-Dynamics-Simulation

5.6.1 Simulationsergebnisse

Die folgende Grafik zeigt den Vergleich der Gelenkbewegungen (q) unter zwei Bedingungen:

- **Ohne Gravitation:** Die Gelenke bleiben stabil in ihrer Ausgangsposition.
- **Mit Gravitation:** Bestimmte Gelenke beginnen zu driften oder fallen aufgrund der Gewichtskraft.

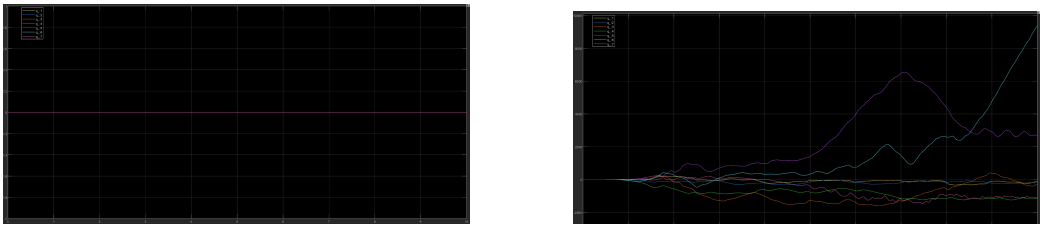


Abbildung 12: Vergleich der Gelenkbewegungen ohne Gravitation (links) und mit Gravitation (rechts).

5.6.2 Interpretation der Ergebnisse

- Einige Gelenke zeigen starke Drift oder unkontrollierte Bewegung (z.B. Gelenk 5 und 6), da sie weniger strukturelle Unterstützung haben.
- Andere Gelenke bleiben relativ stabil (kleinere Ausschläge in den Positionskurven).
- Die Gravitation hat einen signifikanten Einfluss auf die Dynamik des Roboters, insbesondere auf die Gelenke, die nicht aktiv stabilisiert werden.

5.6.3 Zusammenfassung

- Der Roboter bewegt sich unter Gravitation **frei** entsprechend seiner Massenverteilung.
- **Bestimmte Gelenke driften stark**, während andere stabil bleiben.
- Ohne zusätzliche **Regelung oder aktiven Antrieb** führt dies zu unkontrollierter Bewegung.

6 Bewegung im Arbeitsraum

Für die Simulierung des Roboters soll nun eine Bahn im 3D-Arbeitsraum unter Berücksichtigung der kinematischen und dynamischen Beschränkungen abgefahren werden. Folgende Punkte werden benutzt:

Startpunkt = [0.0, -0.8, 0.6, -pi/2, 0, 0]
Endpunkt = [-0.6, -0.1, 0.5, -pi, 0, 0]

Dabei entsprechen die Werte folgendem Format [x, y, z, r, p, y], wobei x, y und z der Position und r, p und y der Orientierung im Raum entsprechen. Zusätzlich sind die Beschränkungen der Achsen folgendermaßen:

Gelenk	1	2	3	4	5	6	7
Winkelbegrenzung (+-) in Rad	2.9668	2.0942	2.9668	2.0942	2.9668	2.0942	3.0541
Geschwindigkeitsbegr. In Rad/s	1.4834	1.4834	1.7452	1.3089	2.2688	2.356	2.356

Abbildung 13: Kinematische Beschränkungen

Da der Bahnablauf ohne eine Last am Roboter simuliert wird, ist das Überschreiten der dynamischen Beschränkungen nicht zu erwarten.

6.1 Bahn mit trapeziodalem Geschwindigkeitsprofil

Für diese Aufgabe wurde im Github das Matlabfile: Aufgabe 6/trajectory_a.m benutzt.

Es soll nun eine Bahn mit trapeziodalem Geschwindigkeitsprofil erstellt werden. Für das Ablaufen der Bahn werden 5 Zwischenpunkte (ZW) benutzt, um das Fahren der Bahn genauer anzupassen. Die Koordinaten, inklusive Start- und Endpunkt sind in der kommenden Tabelle visualisiert:

	X (m)	Y (m)	Z (m)	Roll (rad)	Pitch (rad)	Yaw (rad)
Startpunkt	0.0000	-0.8000	0.6000	-1.5708	0	0
ZW 1	-0.1000	-0.6000	0.6000	-1.5708	0	0
ZW 2	-0.3000	-0.6000	0.6000	-1.9635	0	0
ZW 3	-0.5000	-0.6000	0.6000	-2.4166	0	0
ZW 4	-0.6000	-0.6000	0.6000	-3.1416	0	0
ZW 5	-0.6000	-0.6000	0.5000	-3.1416	0	0
Endpunkt	-0.6000	-0.6000	0.5000	-3.1416	0	0

Abbildung 14: Koordinaten

Dabei wird jeder Punkt anfangs in der Form der Endeffektor-Position angegeben.

Nach dem Festlegen der einzelnen Punkte werden diese mittels der inversen Kinematik in Gelenkkonfigurationen dieser umgewandelt. Diese werden in der Funktion trapveltraj eingegeben, welche zusätzlich noch mit der gezielten Anzahl an Schritten (Samples) von 300 (beudetet man bekommt 300 Erbenisse) die Gelenkkonfigurationen (q), -geschwindigkeiten (qd) sowie -beschleunigungen (qdd) über diese Schrittanzahl ausgibt. außerdem werden die Zeitpunkte der Gelenkkonfigurationen in einer weiteren Variable (tSamples) gespeichert.

Mithilfe dieser Ergebnisse kann nun die Bewegung im Koordinatensystem angezeigt und animiert werden. Die Animation ist in dem Matlabfile: Aufgabe 6/trajectory_a.m und die Trajektorie in folgender Abbildung 15 zu finden.

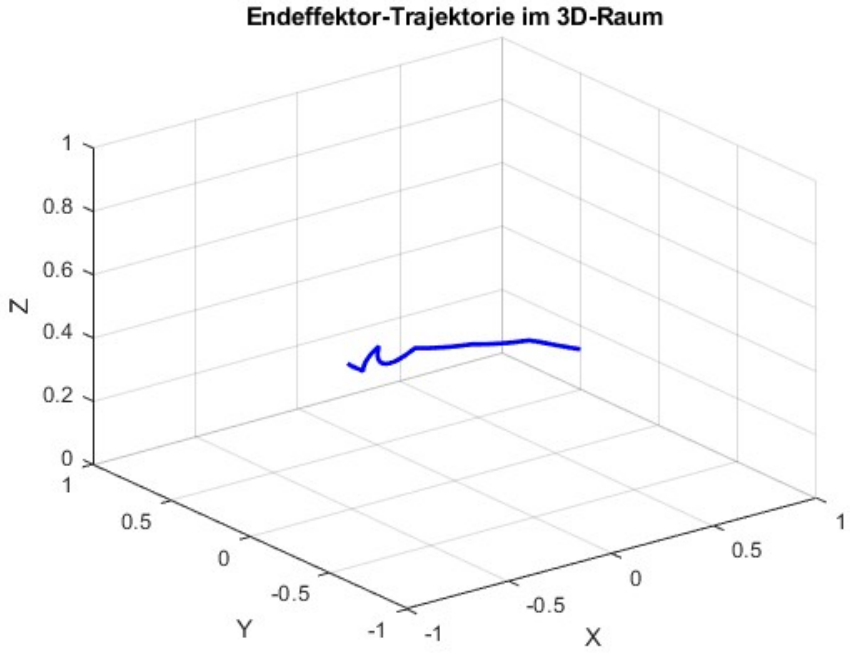


Abbildung 15: Bahnverlauf(a) im Arbeitsraum

Die genauen Koordinaten sind hier allerdings schwer zu erkennen, weshalb auch hier die Ansicht innerhalb von Matlab geeigneter ist.

Zusätzlich kann die sich verändernde Gelenkkonfiguration für jedes Glenk über die Zeit dargestellt werden. Dies wird in der Abbildung 16 visualisiert.

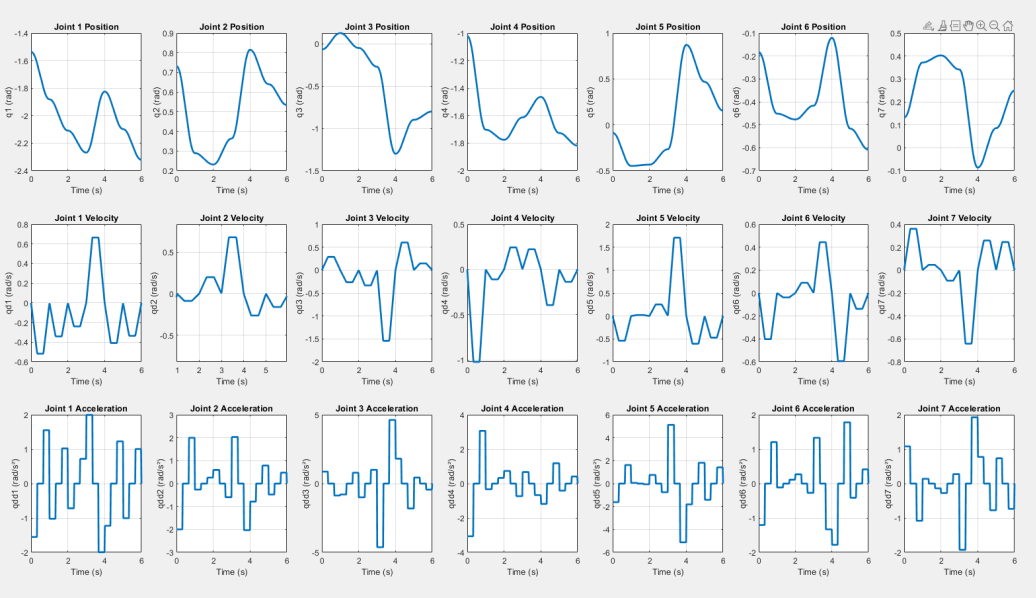


Abbildung 16: Gelenkverläufe (a)

In der ersten Reihe sind die Gelenkkonfigurationen über die Zeit zu erkennen. Hier kann man auch durch scharfes Hinsehen oder über die Nulldurchgänge der Gelenkgeschwindigkeiten die einzelnen Wegpunkte ausmachen. Unter Inbezugnahme der Begrenzungen in Abbildung 13 ist auch zu sehen, dass keine der Gelenke seine maximale Konfiguration über/unterschreitet.

In der zweiten Reihe sind die Gelenkgeschwindigkeiten über die Zeit dargestellt. Dabei sei besonders auf die Trapezförmigkeit der Verläufe zu achten, welche die Hauptcharakteristik dieses Bahnverlaufes darstellt. Bei jedem Punkt ungleich null befindet sich der Endeffektor zwischen zwei Wegpunkten auf der Bewegungsbahn. Auch hier sei noch einmal auf die Tabelle 2 und deren Begrenzungen zu verweisen, welche auch für die Geschwindigkeiten keine der Grenzwerte erreicht.

In der dritten Reihe ist die Beschleunigung der einzelnen Gelenke dargestellt. Die Nulldurchgänge/-phasen der Beschleunigung sind hierbei ein Maß für das Abfahren der Bahn mit maximaler Geschwindigkeit und die Spitzen geben die beschleunigungs- und Bremsvorgänge der Gelenke an.

Zusammengefasst sei gesagt, dass die komplette Bewegung über die Wegpunkte ohne das Auftreten von Singularitäten und unter Einhaltung der Gelenkbegrenzungen stattfindet.

6.2 Transformierte Bahngenerierung

Für die Bearbeitung dieser Aufgabe wurden dieselben Parameter aus der vorherigen Aufgabe sowie auch die Funktion trapveltraj mit denselben Input-Argumenten benutzt. Die Ergebnisse der Gelenkparameter (q, qd, qdd) wurden nun allerdings normiert (Werte zwischen Null und Eins dimensioniert), um als ein Timescaling-Vektor für die Funktion transformtraj benutzt zu werden. Innerhalb der Funktion transformtraj werden nun mittels eines Start- und Endpunktes (keine Zwischenpunkte), der Zeitintervalle zwischen zwei Gelenkkonfigurationen sowie der Schrittanzahl und des Timescalings die Transformations-Matrizen sowie die Geschwindigkeits- und Beschleunigungsprofile der Schritte (300) generiert. Die Gelenkpositionen in Matrizen-Form werden mittels der inversen Kinematik in eine Gelenkkonfiguration mit 7 Werten (Gelenke) konvertiert und die Gelenkgeschwindigkeiten und -beschleunigungen mittels der Jacobimatrix und deren zeitlichen Ableitung ermittelt.

Für den in Abbildung 17 dargestellten Bahnverlauf wurde aus den Transformationsmatrizen der Funktion transformtraj jeweils nur der Positions-Vektor entnommen und dargestellt.

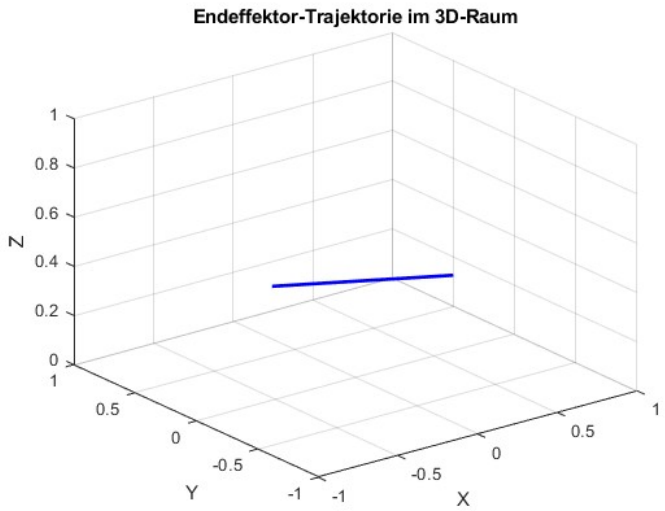


Abbildung 17: Bahnverlauf(b) im Arbeitsraum

Dieser Verlauf ist im Vergleich zum vorherigen Verlauf aus Abbildung 15 eine gerade und nicht kurvig/eckig, da diese Bahn nur mittels eines Start- und eines Endpunktes erzeugt wurde, ohne Wegpunkte zwischendrin zu benutzen.

In der Abbildung 18 sind weiter Unterschiede zwischen den 2 Bahnen zu erkennen.

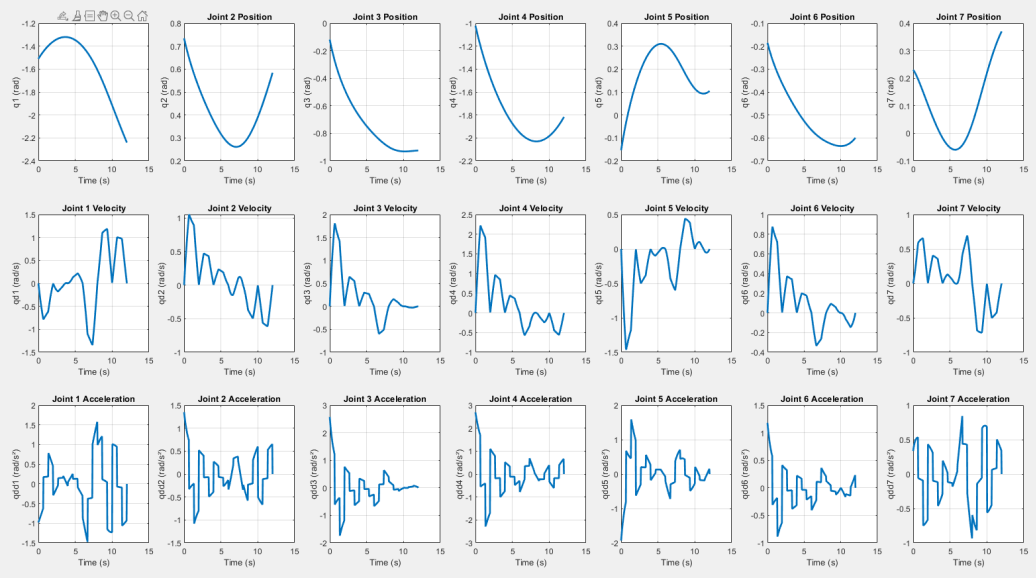


Abbildung 18: Bahnverlauf(b) im Arbeitsraum

Abbildung 14 zeigt erneut, dass die Beschränkungen für die Gelenkposition eingehalten wurden, obwohl es bei Joint 4 sehr eng wird (Grenze: -2.0942, Wert: -2.0293).

Das Geschwindigkeitsprofil ähnelt dem in Abbildung 16 dargestellten, was jedoch auf die Nutzung des Timescalings durch die Funktion ‘trapveltraj‘ zurückzuführen ist. Diese wurde ausschließlich anhand eines Gelenks (dem ersten) generiert, wodurch die Gelenkgeschwindigkeiten und -beschleunigungen stark variieren und ungleichmäßig erscheinen.

Die Gelenkgeschwindigkeitsbegrenzungen werden bei Joint 1, 3 sowie 4 überschritten. Zusammenfassend wird die Bahn zwischen zwei Punkten mit Timescaling auf einer Trajektorie mit sieben Wegpunkten abgefahren, wodurch die Bewegung für den Roboter in der aktuellen Konfiguration nicht optimal ist.

Für eine saubere und sichere Abfahrt müsste entweder eine alternative Strecke festgelegt, das Geschwindigkeitsprofil angepasst oder die Zeit, in der die Bahn abgefahren wird, verlängert werden.

Der Ablauf der Aufgabe kann über das Matlab-File auf Github ('Aufgabe6/trajectory') nachvollzogen werden.

7 Simulation mit Joint Space Motion Model

Die Aufgabe wurde mittels des Matlab-Skripts `Aufgabe7/sieben_Rob_Simulink.mlx` und der Simulink-Datei `Aufgabe7/sl_robreg1_CR_MM_dezPD.slx` umgesetzt.
PD-Regler (PD = Proportional-Derivativ) sind einfach in der Implementierung und im Tuning. Selbst ohne ein komplexes dynamisches Modell kann durch die dezentrale Berechnung eine Bahn im Gelenkraum ermittelt werden. Außerdem sorgen PD-Regler für gut gedämpfte Bewegungen mit minimalen Oszillationen. Damit sinkt auch der Rechenaufwand, da ein Regler nur Positions- und Geschwindigkeitsdifferenzen, aber keine Kopplungseffekte benötigt.
Die Regelung erfolgt durch die Formel:

$$\tau = k_p(q_d - q) + k_d(\dot{q_d} - \dot{q}) \tag{23}$$

Der Nachteil besteht darin, dass nichtlineare Effekte wie die Corioliskräfte, Trägheiten und Gravitation nicht berücksichtigt werden. Außerdem können schnelle Bewegungen ohne Lasten zu ungenauerer Bahnverfolgung führen. Somit muss der Regler bei neuen Lasten oder anderen Geschwindigkeiten neu abgestimmt werden. In Abbildung 19 ist das zugehörige Simulink-Modell sichtbar.

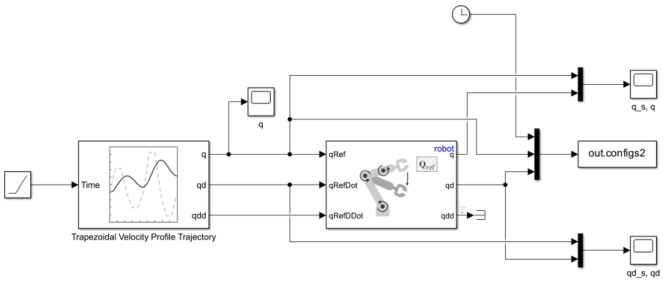


Abbildung 19: Simulink-Model

Das Skript beginnt mit dem Laden des Roboter-Modells und der Spezifikation der Bahnpunkte. Hier wurden die Bahnpunkte aus der vorherigen Aufgabe verwendet.
Nachfolgend wird eine Trajektorienplanung im Gelenkraum (Joint Space) durchgeführt. Die Gelenkwinkel werden dabei mittels Inverser Kinematik aus den Bahnpunkten berechnet und in C1 bis C7 gespeichert. Aus diesen Gelenkkonfigurationen wird mittels `trapveltraj()` eine Bahn im Gelenkraum mit jeweils linearer Beschleunigung, konstanter Geschwindigkeit und linearer Verzögerung erzeugt. Dieses trapezförmige Geschwindigkeitsprofil sorgt für eine glatte Bewegung ohne Sprünge.
Geschwindigkeit unter Einhaltung der kinematischen und dynamischen Beschränkungen des Roboters

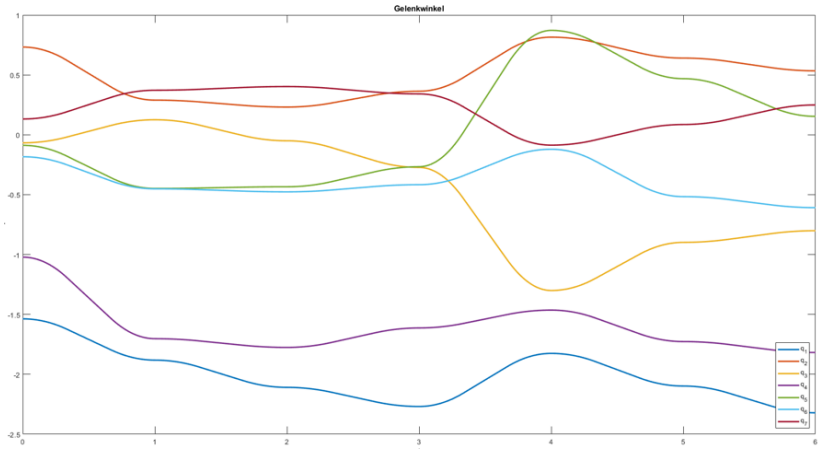


Abbildung 20: Gelenkwinkel

Dabei sind q (Abbildung 9) die berechneten Gelenkwinkel, q (Abbildung 10) die Gelenkwinkelgeschwindigkeiten und q die Gelenkwinkelbeschleunigungen. Die daraus folgende Bewegung wird schlussendlich in einem Live-Plot simuliert.

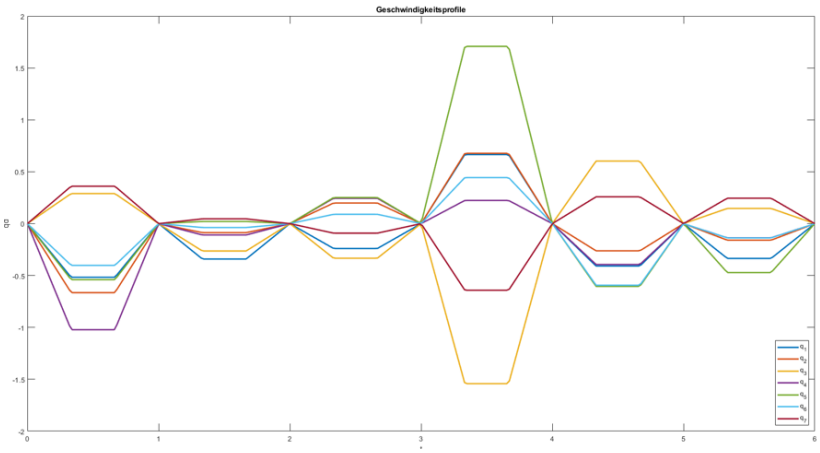


Abbildung 21: Geschwindigkeitsprofil

Auswertung

Dank geringer Geschwindigkeit und keiner zusätzlicher Last verfolgt der Roboterarm die vorgegebene Bahn sehr präzise. In (Abbildung 11) ist beispielhaft ein Einschwingvorgang des Reglers erkennbar. Verglichen wird hier die vorgegebene Geschwindigkeit des Gelenk 3 mit der tatsächlichen. Es kommt kaum zu Abweichungen.

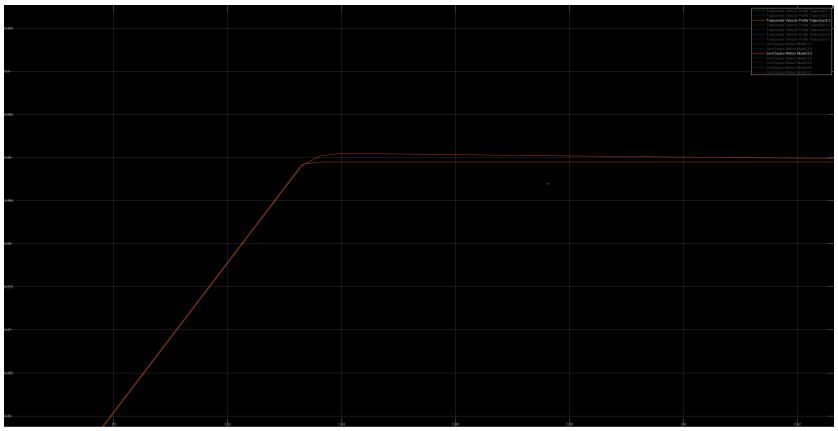


Abbildung 22: Einschwingverhalten

Vergleich mit dem Beispiel „Plan and Execute Task- and Joint-Space Trajectories Using KINOVA Gen3 Manipulator“ aus der RSTB-Dokumentation:

Kriterium	Beispiel RSTB-Doku	unsere Umsetzung
Trajektorienraum	Kombination Joint-Space und Task-Space	nur Joint-Space
IK-Verwendung	Zuerst im Task-Space, dann Umwandlung in Joint-Space	direkte Nutzung der IK für Wegpunkte
Trajektorienerzeugung	transformtraj -> Task Space cubicpolytraj -> Joint-Space	trapveltraj -> glatte Bewegung im Joint-Space
Simulationsmethode	ODE-Solver -> berücksichtigt Dynamik	direkte Animation + Simulink -> keine Dynamik
Geschwindigkeitsprofil	automatische Berechnung der Zeit auf Basis der gewünschten Endeffektor-Geschwindigkeit	trapezförmiges Geschwindigkeitsprofil

Abbildung 23: Vergleich mit der Trajektorienplanungsansätze der RSTB-Dokumentation

8 Bewegungssteuerung durch (CTC)

Der **CTC** ist eine modellbasierte nichtlineare Regelungsmethode, die die Systemdynamik durch *inverse Dynamikkompensation* linearisiert. Sie wird häufig in Robotermanipulatoren eingesetzt, um eine präzise Trajektorienverfolgung unter Berücksichtigung der Roboterdynamik zu gewährleisten.

8.1 Modell der Roboterdynamik

Die Bewegung eines Robotermanipulators mit n-Freiheitsgraden wird durch folgende Gleichung beschrieben:

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) = \tau \tag{24}$$

wobei:

- $\mathbf{q} \in R^n$ der Vektor der Gelenkpositionen ist.
- $M(\mathbf{q}) \in R^{n \times n}$ die **Massenmatrix** darstellt.
- $C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ die **Coriolis- und Zentrifugalkräfte** beschreibt.
- $G(\mathbf{q})$ das **Gravitationsmoment** ist.
- $\tau \in R^n$ das Vektor der **Steuermomente** darstellt.

Das Ziel der berechneten Drehmomentregelung besteht darin, ein Reglergesetz zu definieren, das den Roboter mit hoher Genauigkeit einer gewünschten Trajektorie $\mathbf{q}_d(t)$ folgen lässt.

8.2 Ableitung des Regelgesetzes

Um die Trajektorienverfolgung zu gewährleisten, wird ein Steuerbefehl τ definiert, der die Nichtlinearitäten des Systems kompensiert und das geschlossene Regelsystem in ein lineares Zweitordnungssystem überführt.
Das geregelte Drehmoment ergibt sich aus:

$$\tau = M(\mathbf{q})\mathbf{u} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) \tag{25}$$

wobei die Hilfssteuerung $\mathbf{u}$ wie folgt definiert ist:

$$\mathbf{u} = \ddot{\mathbf{q}}_d + K_p(\mathbf{q}_d - \mathbf{q}) + K_d(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) \tag{26}$$

mit:

- $\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d$: **gewünschte Position, Geschwindigkeit und Beschleunigung**.
- K_p, K_d: **positive diagonale Verstärkungsmatrizen**, die die PD-Regelungsgewinne bestimmen.

Setzt man $\mathbf{u}$ in die Drehmomentgleichung ein, so ergibt sich ein lineares Fehlerdynamikmodell zweiter Ordnung:

$$\ddot{\mathbf{e}} + K_d\dot{\mathbf{e}} + K_p\mathbf{e} = 0 \tag{27}$$

mit dem Positionsfehler $\mathbf{e} = \mathbf{q}_d - \mathbf{q}$. Diese Gleichung garantiert eine **exponentielle Fehlerkonvergenz**, sofern K_p und K_d angemessen gewählt sind.

8.3 Implementierung in Simulink

Die Implementierung in Simulink umfasst folgende Schritte:

1. **Trajektoriengenerierung:** Eine vorgegebene Trajektorie (z. B. trapezförmiges Geschwindigkeitsprofil) liefert $\mathbf{q}_d, \dot{\mathbf{q}}_d, \ddot{\mathbf{q}}_d$.
2. **Fehlerberechnung:** Die Abweichung zwischen der gewünschten und der tatsächlichen Gelenkstellung wird berechnet.
3. **Inverse Dynamik-Kompensation:** Das berechnete Drehmoment wird angewendet, um die Roboterdynamik zu kompensieren.
4. **Vorwärtsdynamik-Simulation:** Die Gelenkbewegung wird mit numerischer Integration simuliert.

Das Simulink-Blockdiagramm enthält:

- Einen **Trajektoriengenerator** für gewünschte Gelenkpositionen, -geschwindigkeiten und -beschleunigungen.
- Einen **PD-Regler** zur Berechnung der Fehlerkorrektur.
- Einen **Roboterdynamik-Block** zur Berechnung der inversen Dynamik.
- Einen **numerischen Integrator**, um die Gelenkbewegung über die Zeit zu simulieren.

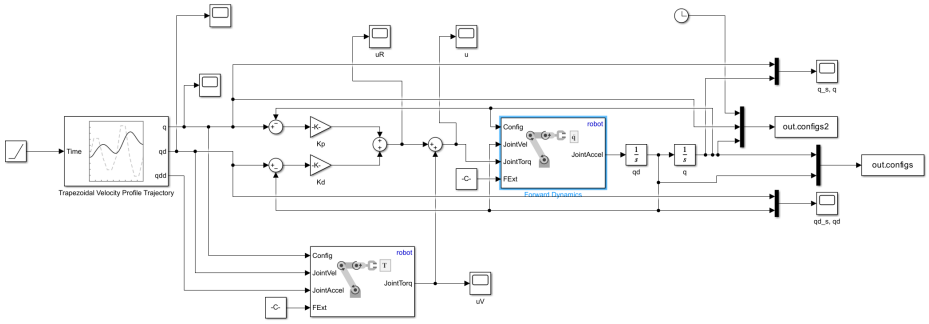


Abbildung 24: Blockdiagramm der Simulink-Implementierung des Computed-Torque-Reglers

8.4 Ergebnisse der Regelung

Nach einer Feinabstimmung der Regelungsparameter K_p und K_d konnte eine präzise Nachverfolgung der gewünschten Trajektorie erreicht werden. Die Simulationsergebnisse zeigen, dass die tatsächliche Gelenkposition q der vorgegebenen Solltrajektorie q_d eng folgt, während die Positions- und Geschwindigkeitsbeschränkungen eingehalten werden.
Wie in Abbildung 25 zu sehen ist, passt sich das System gut an die gewünschte Trapeztrajektorie an. Kleinere Abweichungen zu Beginn der Simulation resultieren aus der initialen Beschleunigungsphase, werden jedoch durch die PD-Regelung schnell korrigiert.
Die Geschwindigkeitsprofile der Gelenke zeigen ebenfalls eine gleichmäßige Anpassung an die gewünschten Werte:
In Abbildung 26 ist zu erkennen, dass die Gelenkgeschwindigkeiten den gewünschten Trapezprofilen folgen, während abrupte Änderungen vermieden

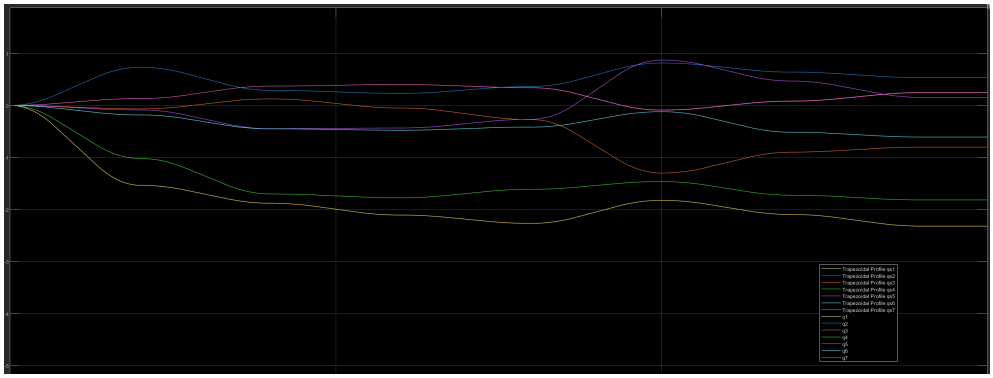


Abbildung 25: Vergleich zwischen gewünschter und tatsächlicher Gelenkposition

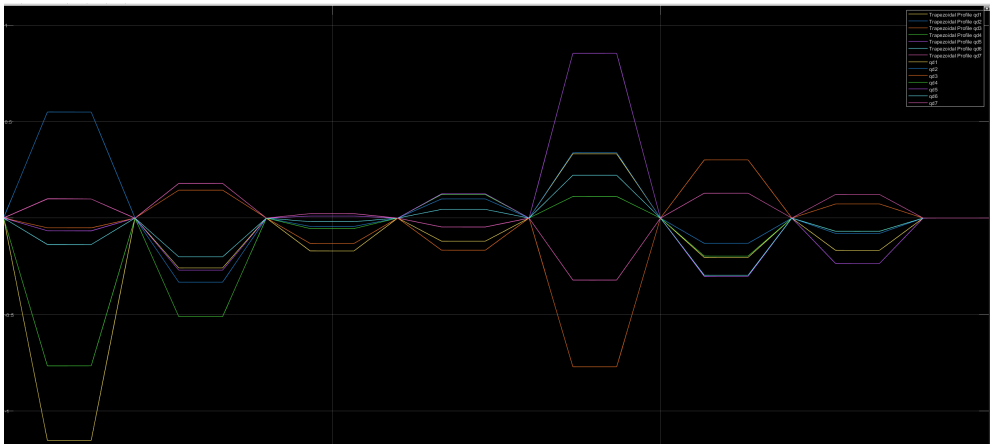


Abbildung 26: Vergleich zwischen gewünschter und tatsächlicher Gelenkgeschwindigkeit

werden. Dies zeigt, dass die Regelung sowohl Stabilität als auch Dynamik effizient berücksichtigt.

8.5 Reglerausgangsanalyse

Die berechnete Stellgröße U für die Computed-Torque-Regelung wurde analysiert, um die Dynamik der Regelung sowie die Stabilität der angewendeten Drehmomente zu bewerten.
Die Abbildung 27 zeigt die Drehmomentausgaben für die sieben Gelenke des Roboters. Folgende Beobachtungen können gemacht werden:

- Die **ersten Regelungszyklen zeigen hohe Stellgrößen**, insbesondere

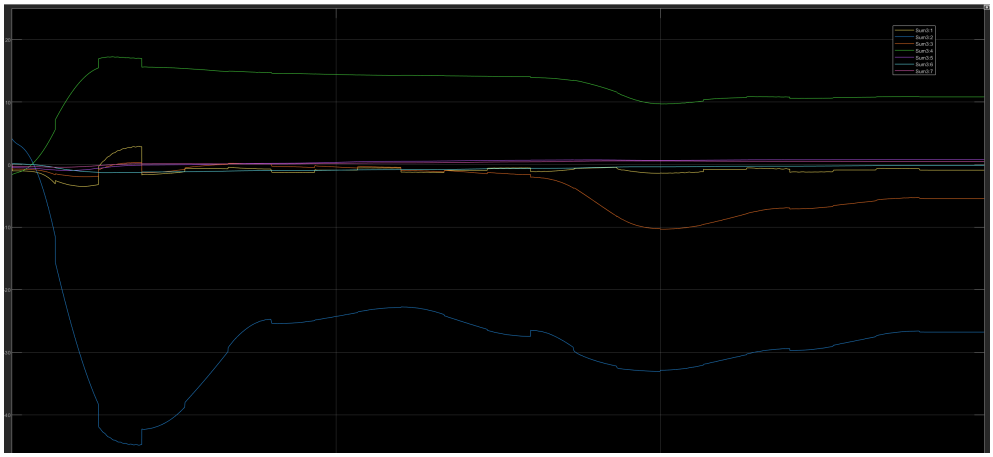


Abbildung 27: Reglerausgang U für die einzelnen Gelenke

für einige Gelenke. Dies ist auf die Korrektur der initialen Abweichungen von der Solltrajektorie zurückzuführen.

- Nach der Anfangsphase stabilisieren sich die Drehmomente und bleiben innerhalb akzeptabler Grenzen, was auf eine erfolgreiche Feinabstimmung der Regelparameter K_p und K_d hinweist.
- Bestimmte Gelenke (z. B. Gelenk 1 und Gelenk 3) zeigen größere Drehmomentschwankungen, was auf dynamische Kopplungseffekte im Robotermodell hinweisen kann.
- Die Stellgrößen bleiben insgesamt innerhalb der physikalischen Grenzen des Roboters, was sicherstellt, dass keine Sättigungsprobleme auftreten.

9 Quellen

- Daten Kuka_lbr_iiwa_14_r820: https://www.robots.com/industrial-robots/kuka-lbr-iiwa-14-r820 (1)
- Datenblatt Kuka: https://www.kuka.com/-/media/kuka-downloads/imported/8350ff3ca11642998dbdc81dcc2ed44c/0000246833_en.pdf?rev=7efde9ea5df74660a561068aef175e07&hash=97486EE79D01CB75C54A83EB2699970A (2)
- URDF Github https://github.com/Daniella1/urdf_files_dataset/blob/main/urdf_files/ros-industrial/kuka/kuka_lbr_iiwa_support/urdf/lbr_iiwa_14_r820.urdf (3)