

Robuste, zentralisierte, dynamische Regelung eines anthropomorphen Manipulators

Projektbericht

13. Juli 2024

Modul: DLBROPIRC01_D

Tutor: Ha Ngo

Studiengang: Fernstudium Bachelor of Engineering Robotics DEU

Verfasser: Griesche, Nico

Matrikelnummer.: 92100873

Inhalt

Abbildungsverzeichnis:	iii
Abkürzungsverzeichnis:	iv
Variablenverzeichnis:	v
1. Einleitung	1
1.1. Aufgabenanalyse.....	1
1.2. Benötigte Ressourcen und Angaben zur Ausführung.....	1
1.3. Literatur Recherche.....	2
1.4. Projektmanagement	2
1.5. Solver Einstellungen, Manipulator Geometrie und Motor-Datenblatt.....	2
2. Modellierung des Manipulators	2
2.1. Kinetische Energie T	2
2.2. Potenzielle Energie U.....	3
2.3. Generalisierte Kräfte	3
3. Modellierung der Aktorik.....	4
3.1. Modellierung der Kommutierung und Inverter-Schaltung.....	4
3.2. Mathematische Grundlagen Aktorik.....	5
3.3. Modellierung des BLDCM	6
3.4. Transmission	6
3.5. Encoder.....	6
4. Trajektorien Planung	7
5. Regelkreis.....	7
6. Messung des Tracking-Fehler	8
6.2. Diskussion des Tracking-Fehler	9
7. Projektevaluation	10
Literaturverzeichnis:.....	11
Verzeichnis der Anhänge:	12

Abbildungsverzeichnis:

Abbildung 1 - Baba, M. A., Naoui, M., Cherkaoui, M. (2023). Digital Technologies and Applications. Modeling and Simulation of a BLDC Motor Speed Control in Elekctic Vehicles. Springer Nature Switzerland AG. Vol. 1, S. 885.

Abbildung 2 – Momenzadeh, M. M., Ahmed, A. F., Tolba, A. (2013). Modelling and Simulation Of The BLDC Electric Drive System Using SIMULINK/MATLAB for a Hybrid Vehicle. Universität Paderborn. S. 15

Bei allen Abbildungen, die nicht explizit aufgezählt sind, handelt es sich um eigene Bildschirmaufnahmen.

Abkürzungsverzeichnis:

1. BLDC(M) – Brushless Direct Current (Motor), Bürstenloser Gleichstrommotor
2. EMF – Electromotive Force, Elektromotorische Kraft
3. DC – Direct Current, Gleichstrom
4. PWM – Pulse Width Modulation, Pulsweitenmodulation
5. MOSFET – Metal-Oxide-Semiconductor Field-Effect Transistor, Metall-Oxid-Halbleiter-Feldefektttransistor
6. PI-Controller – Proportional-Integral Controller, Proportional-Integral-Regler
7. RPM – Revolutions Per Minute, Umdrehungen pro Minute
8. DH – Denavit-Hartenberg
9. DOF – Degrees of Freedom, Freiheitsgrade

Variablenverzeichnis:

Kapitel 2:

1. $I_{inertia}$ – Trägheitsmatrix
2. \vec{v} – Lineargeschwindigkeit
3. $\vec{\omega}$ – Winkelgeschwindigkeit
4. h - Höhe
5. L – Lagrangian
6. R – Rotationsmatrix
7. T – kinetische Energie
8. U – Potenzielle Energie
9. g - Gravitationskraft
10. ε - nicht-konservative Kräfte.
11. τ – Drehmoment

Kapitel 3:

1. $f_i(\theta)$ - Phasenfunktionen zur Berechnung der Rück-EMK als Funktion der Rotorposition θ für i ($i \in a, b, c$)
2. τ_e - Elektromagnetisches Drehmoment
3. τ_l - Lastdrehmoment
4. e_i - Elektromotorische Kraft der Phase i ($i \in a, b, c$)
5. f - Viskoser Reibungskoeffizient
6. I_i - Strom der Phase i ($i \in a, b, c$)
7. J – Trägheitsmoment
8. k_e - Rück-EMK-Konstante
9. k_t - Drehmomentkonstante
10. L – Induktivität
11. R – Widerstand
12. V_{ac}, V_{cb} - Spannungsdifferenz zwischen den Phasen a und c sowie den Phasen c und b
13. V_i - Spannung der Phase i ($i \in a, b, c$)
14. w_m - Mechanische Winkelgeschwindigkeit
15. θ – elektrische Rotorposition

1. Einleitung

Der vorliegende Projektbericht beschreibt die im Rahmen der Aufgabenstellung 1.1. „Robuste zentralisierte dynamische Regelung eines anthropomorphen Manipulators“ durchgeführten Arbeiten. Im ersten Kapitel wird die Aufgabenstellung analysiert und die Ziele definiert. Darüber hinaus wird die Literatur-Recherche und das Projektmanagement erläutert.

Im zweiten Kapitel wird die Modellierung des Manipulators betrachtet. Dabei wird das grundlegende Modell erläutert sowie die mathematische Aufstellung der Bewegungsgleichungen.

Das dritte Kapitel behandelt die Entwicklung der Aktorik, inklusive Details zur Kommutierung, Inverter-Schaltung und den mathematischen Gleichungen der BLDC-Motoren. Zudem werden die Getriebe und die Positionsgeber erläutert.

Im vierten Kapitel wird die Erstellung der drei Trajektorien und den damit verbunden Geschwindigkeitsprofil aufgezeigt.

Im fünften Kapitel wird der aufgestellte Regelkreis erläutert und grafisch aufgezeigt.

Im sechsten Kapitel wird der Tracking-Fehler für die drei Trajektorien aufgenommen. In einem vierten Szenario wird eine Zufallsvariation auf die Manipulator Parameter angewendet und der Fehler erneut aufgezeigt. Der Tracking-Fehler wird im Anschluss diskutiert.

Abschließend evaluiert das siebte Kapitel, inwieweit die definierten Ziele erreicht wurden, und reflektiert die Durchführung und Umsetzung des Projekts.

1.1. Aufgabenanalyse

Die Projektziele folgen zu:

1. Definition und Modellierung eines anthropomorphen Manipulators mit drei Gelenken. Der Manipulator soll so konzipiert werden, dass er einen Punkt erreichen kann, der sich zwei Meter entfernt von seiner Basis befindet. Dabei beträgt die Gesamtmasse des Manipulators 35kg.
2. Definition der Verstärkung und des dynamischen Verhaltens der Aktorik und Sensorik.
3. Implementieren eines robusten, zentralisierten Regelkreises, welcher die Steueraktionen erzeugt.
4. Durchführung von drei Simulationsexperimenten von jeweils zehn Sekunden sowie einen Simulationsexperiment mit fünfprozentiger Zufallsvariation auf den Manipulator Parametern.
5. Messung der Tracking-Fehler für jedes Simulationsexperiment.

1.2. Benötigte Ressourcen und Angaben zur Ausführung

Für die MATLAB Files müssen folgende Erweiterungen installiert sein: MATLAB R2023b, Simscape, Simscape Multibody, Simscape Electrical Simulink, Robotic System Toolbox, Symbolic Math Toolbox.

Vor dem Öffnen der „Griesche_Nico_92100873_DL BROPIRC01_SIMULATION“-Datei muss die „Griesche_Nico_92100873_DL BROPIRC01_MAIN“-Datei ausgeführt werden, um die Trajektorien im Workspace zu hinterlegen.

1.3. Literatur Recherche

Die Recherche erfolgte hauptsächlich über das Portal „ebsco“ und „google scholar“, mit Verwendung von Schlüsselbegriffen wie „manipulator“, „bldc motor“, „modellig“, „matlab“ und wurde in verschiedenen Kombinationen durchgeführt.

1.4. Projektmanagement

Im Rahmen des Projektmanagements wurde bewusst auf aufwendige und komplexe Methoden verzichtet. Aufgrund der Tatsache, dass das Projekt als Einzelperson durchgeführt wurde und keine konkreten zeitlichen Vorgaben bestanden, erwies sich diese einfache Herangehensweise als ausreichend und effektiv.

1.5. Solver Einstellungen, Manipulator Geometrie und Motor-Datenblatt.

Die Solver Einstellungen können Anhang 24 entnommen werden. Für die Manipulator Geometrie wird auf Anhang 25 verwiesen. Die Motor Parameter wurden einen RPX52-750V24 nachempfunden. Das entsprechende Datenblatt ist im Anhang 26 hinterlegt.

2. Modellierung des Manipulators

Mithilfe der inversen Dynamik werden die Drehmomente bestimmt, die benötigt werden, um das Mehrkörpersystem so zu manipulieren, dass sich die gewünschten Gelenkwinkel einstellen. Im Rahmen dieses Projektes werden die Drehmomente als Lastmoment auf die Aktorik geschaltet.

Für die Berechnung wird die Lagrange Formulierung genutzt. Der Vorteil der Formulierung liegt darin, dass die Bewegungsgleichungen mit einer systematischen Berechnungsvorschrift kalkuliert werden können (Siciliano et al., 2009, S.248).

Die Lagrangian ergibt sich aus der Differenz zwischen der kinetischen Energie „T“ und der potenziellen Energie „U“.

$$L = T - U \quad (2.1.)$$

Die Lagrange-Formulierung folgt dann, sofern nicht-konservative Kräfte „ ε “ – wie z.B. Reibungskräfte – zwischen den Gelenken vorliegen, zu:

$$\left(\frac{d}{dt}\right) \left(\frac{\partial L}{\partial \dot{q}_i}\right) - \left(\frac{\partial L}{\partial q_i}\right) = \varepsilon_i \quad i \in 0,1,2 \quad (2.2.)$$

Daraus kann gefolgert werden, dass 1) die kinetische Energie, 2) die potenzielle Energie sowie wie 3) die generalisierten Kräfte bzw. nicht-konservative Kräfte zunächst berechnet werden müssen.

2.1. Kinetische Energie T

Die kinetische Energie eines starren Körpers ist die Summe aus der Translationsenergie T_{trans} und der Rotationsenergie T_{rot} . Die Translationsenergie eines Massenpunktes ist proportional zu seiner Masse m und dem Quadrat seiner Lineargeschwindigkeit \vec{v} (Wikipedia, 2023):

$$T_{trans} = \frac{1}{2} m \vec{v}^T \vec{v} = \frac{1}{2} m (\dot{x}^2 + \dot{y}^2 + \dot{z}^2) \quad (2.3.)$$

Das heißt, zunächst müssen die kartesischen Koordinaten für jede einzelne Massenposition definiert, abgeleitet und quadriert werden, um die Lineargeschwindigkeit zu bestimmen.

Die Rotationsenergie ergibt sich aus dem Trägheitstensor I und der Winkelgeschwindigkeit $\vec{\omega}$ zum Quadrat (Wikipedia, 2022):

$$T_{rot} = \frac{1}{2} \vec{\omega}^T I_{inertia} \vec{\omega} \quad (2.4.)$$

Der Trägheitstensor, oder im Englischen als „inertia tensor“ bezeichnet, ist eine 3x3 symmetrische Matrix, welche auf Ihrer Diagonalen die Massen Trägheitsmomente und auf den nicht-diagonalen Elementen die Massen Produkte der Trägheit aufweist (Kasadin & Paley, 2011, S. 481). Für den hier behandelten Manipulator wird von einer komplexen Form abgesehen und für alle drei Körper ein voller Zylinder angenommen. Die entsprechende Trägheitstensor Matrix kann aus Tabellen entnommen werden.

Es ist wichtig zu beachten, dass die Elemente der Trägheitsmatrix im lokalen Frame des jeweiligen Körpers definiert sind. Daher muss, um zu vermeiden, dass die Trägheitsmatrix von der Manipulator Konfiguration abhängig ist, die Winkelgeschwindigkeit auf den lokalen Frame des Körpers bezogen werden.

Wenn die Joints in der Denavit-Hartenberg Konvention definiert worden sind, kann mit der Transponierten Rotationsmatrix R_i von Link_i zu Link_{i-1} die Winkelgeschwindigkeit übertragen werden (Sicilliano et al., 2009, S.251):

$$\omega_{i-1} = R_i^T \omega_i \quad (2.5.)$$

Die gesamte kinetische Energie ergibt sich zu:

$$T = T_{trans} + T_{rot} \quad (2.6.)$$

(Anhang 1 und 2)

2.2. Potenzielle Energie U

Für die Lagrange-Formulierung fehlt weiterhin die potenziellen Energien. Bei Vernachlässigung der Federenergie, entspricht diese der Lageenergie (Wikipedia, 2023):

$$U = mgh \quad (2.7.)$$

Die Höhen ergeben sich aus den Positionsvektoren der Massen, und zwar aus den Z-Elementen (im Welt-Koordinatensystem) (Anhang 3).

2.3. Generalisierte Kräfte

Die generalisierte Kraft „ ε “ ergibt sich aus den Beiträgen des Aktuierungsdrhmmoments „ τ “ am Gelenk und der viskosen Reibungsdrehmomente (Sicilliano et al., 2009, S.249).

$$\varepsilon = \tau - b\dot{\theta} \quad (2.8.)$$

Durch Freischneiden der Körper, können für alle Gelenke Drehmoment-Matrizen aufgestellt werden. In diesem Projekt wird vereinfacht angenommen, dass an eindimensionalen Gelenken Kräfte ausschließlich auf der Gelenkkachse wirken (Anhang 4).

3. Modellierung der Aktorik

3.1. Modellierung der Kommutierung und Inverter-Schaltung

Für die Realisierung der Gelenkbewegungen werden Aktoren benötigt, die das erforderliche Drehmoment (bzw. die Kräfte) bereitstellen. Diese können pneumatischer, hydraulischer oder elektrischer Nature sein und kommen jeweils mit eigenen Vorteilen. Generell gilt jedoch, dass die Aktoren einige kritische Leistungsmerkmale aufweisen sollten. So ist bspw. ein niedriges Trägheitsmoment, kombiniert mit einem hohen Leistungs-Gewicht-Verhältnis wichtig. Dies ermöglicht es den Aktoren, schnelle und präzise Bewegungen auszuführen, ohne durch ihre eigene Masse signifikant behindert zu werden (Sicilliano et al. 2009, S. 193f).

Nach Sicilliano et al. (2009, S. 194f) handelt es sich bei elektrischen Servomotoren um die am häufigsten genutzten Aktoren in robotischen Anwendungen. Dabei sind die zwei am meisten verbreiteten Typen Permanent-Magnet-Gleichstrommotoren (PMDCM) und Bürstenlose-Gleichstrommotoren (BLDCM).

Im Rahmen dieser Projektarbeit, wurde ein dreiphasiger BLDCM modelliert. Im Gegensatz zu herkömmlichen Gleichstrommotoren, besteht der Stator aus mehrphasigen Wicklungen, während der Rotor mit Permanentmagneten ausgestattet ist. Durch die Eliminierung von Bürsten und Kommutator werden elektrische und mechanische Verluste reduziert und eine höhere Effizienz und Langlebigkeit erreicht. Durch die Verlagerung der Wicklungen auf den Stator verbessert sich die Wärmeableitung. Insgesamt führt das zu einer kleineren Bauweise und ermöglicht trägeitätsärmere Rotoren im Vergleich zu herkömmlichen permanentmagnetischen Gleichstrommotoren (Sicilliano et al. 2009, S. 195).

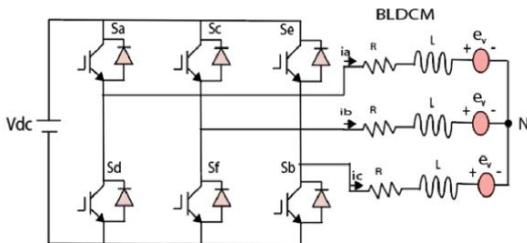


Abbildung 1 – Schematische Darstellung eines BLDCM inklusive Inverter Schaltung

Zunächst wurde die oben dargestellte dreiphasige Inverter Schaltung nachgebildet. Hierfür wurde eine Gleichspannungs-Quelle mit sechs MOSFETs aus der Simscape Bibliothek verschaltet. Da die direkten Spannungssignale von der Spannungsquelle nur mit Simscape Bausteinen kompatibel ist, wurde ein Simscape Three-Phase Measurement Block eingefügt und die Spannungen am Messausgang abgegriffen (Anhang 5).

Die Funktionsweise eines BLDCM basiert auf der Interaktion zweier magnetischer Felder. Ein elektromagnetisches Feld einer Spule am Stator und ein magnetisches Feld eines Permanentmagneten auf dem Rotor. Die magnetische Anziehungskraft zwischen Nord und Südpol erzeugt ein Drehmoment und dreht den Rotor. Die Drehung des Rotors wird durch das Wechseln der Stromrichtung in

den Stator-Wicklungen aufrechterhalten, was als Kommutierung bezeichnet wird. Um die Stator-Wicklungen zum richtigen Zeitpunkt zu wechseln, gilt es die Position des Rotors zu bestimmen. Hierfür werden mindestens drei Hall-Sensoren benötigt, welche jeweils um 120 Grad versetzt sind (Anhang 8) (Yamashita et al. 2017, S. 3).

Die Hall-Sensoren werden in einen Simulink Funktion-Block durch eine if-Gruppe simuliert. Dabei wird die elektrische Rotorposition θ_e verwendet. Je nach Hall-Status und Drehrichtung wird in einer weiteren if-Gruppe die Ansteuerung der MOSFETs vorgenommen (Anhang 6 und 7). Als Ergebnis wird die DC-Spannung in eine AC-Spannung invertiert (Anhang 8).

3.2. Mathematische Grundlagen Aktorik

Es ist anzumerken, dass in der recherchierten Literatur keine einheitlichen mathematischen Berechnungsvorschriften vorliegen. Obwohl die zugrundeliegenden Gleichungen oftmals ähnlich sind, variieren die spezifischen Formulierungen und Herangehensweisen von Quelle zu Quelle im Detail.

Die verwendeten spannungs-Gleichungen eines dreiphasigen BLDCM folgen zu (Baba et al. 2023, S.885ff, Nalli et al. 2019, S. 1550f):

$$V_i = RI_i + L \frac{di_i}{dt} + e_i \quad i \in a, b, c \quad (3.1.)$$

$$\begin{bmatrix} V_a \\ V_b \\ V_c \end{bmatrix} = \begin{bmatrix} R & 0 & 0 \\ 0 & R & 0 \\ 0 & 0 & R \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \begin{bmatrix} L & 0 & 0 \\ 0 & L & 0 \\ 0 & 0 & L \end{bmatrix} \frac{d}{dt} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} + \begin{bmatrix} e_a \\ e_b \\ e_c \end{bmatrix} \quad (3.2.)$$

Die Spannungsgleichung von Phase zu Phase ergeben sich zu:

$$V_{ac} = R(i_a - i_c) + L \left(\frac{di_a}{dt} - \frac{di_c}{dt} \right) + e_a - e_c \quad V_{cb} = R(i_c - i_b) + L \left(\frac{di_c}{dt} - \frac{di_b}{dt} \right) + e_c - e_b \quad (3.3.)$$

Nach dem ersten Kirchhoffschen Gesetz ist in einem Knotenpunkt die Summe der hineinfließenden Ströme immer gleich der Summe der herausfließenden Ströme. Demnach gilt:

$$\frac{di_c}{dt} = - \left(\frac{di_a}{dt} + \frac{di_b}{dt} \right) \quad (3.4.)$$

Die Phasenströme können durch Einsetzen und Umformen gelöst werden:

$$\frac{di_a}{dt} = \left(\frac{2}{3L} V_{ac} + \frac{1}{3L} V_{cb} - \frac{1}{L} i_a R - \frac{2}{3L} e_{ac} - \frac{1}{3L} e_{cb} \right) \quad (3.5.)$$

$$\frac{di_b}{dt} = \left(\frac{1}{3L} V_{ac} + \frac{2}{3L} V_{cb} - \frac{1}{L} i_b R - \frac{1}{3L} e_{ac} - \frac{2}{3L} e_{cb} \right) \quad (3.6.)$$

Die elektromotorischen Kräfte folgen zu:

$$e_a = k_e w_m f_a(\theta) \quad e_b = k_e w_m f_b(\theta) \quad e_c = k_e w_m f_c(\theta) \quad (3.7.)$$

Die grundlegende Bewegungsgleichung folgt zu:

$$\frac{dw_m}{dt} = \frac{\tau_e - f * w_m - \tau_l}{J} \quad (3.8.)$$

Das Motor-Drehmoment folgt zu:

$$\tau_e = \frac{e_a i_a + e_b i_b + e_c i_c}{w_m} \quad (3.9.)$$

$$\tau_e = k_t i_a \quad (3.10.)$$

3.3. Modellierung des BLDCM

Aus den mathematischen Grundlagen gehen vier Simulink Funktions-Blöcke hervor, welche die folgenden Aspekte berechnen: Strom, elektromotorische Kraft, elektromagnetisches Drehmoment und Rotorgeschwindigkeit (Anhang 8 und 9).

Grundlegend werden die oben aufgeführten Gleichungen schriftlich in Funktions-Blöcke eingetragen und die entsprechenden Variablen angebunden. An dieser Stelle wird auf eine detaillierte Ausführung der Blöcke verzichtet und nur einige Erwähnenswerte Aspekte aufgezählt.

- 1) Die elektromotorische Kraft wird mit einer Trapezfunktion simuliert. Dafür wird der elektrische Rotorwinkel, das ist der mechanische Rotorwinkel multipliziert mit den Polpaaren, in einer if-Anweisung abgefragt. Die EMF ist immer dann am stärksten, wenn die Magnetpole des Rotors direkt an den Spulen des Stators vorbeilaufen (Anhang 8 und 10).
- 2) Da die Rotorgeschwindigkeit eine grundlegende Variable in den Motorgleichungen ist, muss zum Start der Simulation sichergestellt werden, dass der Rotor aus der Nullstellung kommt. Dies geschieht mittels zweier „Delay“ Bausteinen am „Rotor Speed Block“. Einen Z^{-1} der für den ersten Simulationsschritt ein Motordrehmoment ungleich Null anlegt und einen Z^4 , der das Lastmoment erst nach vier Simulationsschritten zu schaltet (Anhang 11).
- 3) Ein weiterer wichtiger Punkt, sind die genaue Einhaltung der Einheiten. Das Augenmerk liegt hier auf den Rotorenvariablen. So kommt w_m zunächst in der Einheit rad/s aus dem Motor-Speed-Block. Während die Spannungskonstante in V/kRPM vorlag. Auch wird die Rotorposition in einem Bereich von null bis zwei π benötigt wird, kommt aber zunächst in einen Wert, der die gesamte Umdrehungszahl in Radianen widerspiegelt.

3.4. Transmission

Bei der Ausführung von Gelenkbewegungen von Manipulatoren sind oft niedrige Geschwindigkeiten bei gleichzeitig hohen Drehmomenten gefordert. Diese Anforderungen stehen im Widerspruch zu den optimalen Betriebsbedingungen von Servomotoren, die in der Regel hohe Geschwindigkeiten bei niedrigen Drehmomenten liefern. Um die mechanische Leistung des Motors effektiv auf das Gelenk zu übertragen, ist daher der Einsatz von Getrieben notwendig (Sicilliano et al. 2009, S. 192).

Im Rahmen dieses Projekts wird auf eine komplexe Modellierung verzichtet und ein Stirnradgetriebe mit einem definierten Übersetzungsverhältnis simuliert. Elastische- und Dämpfungskräfte werden nicht berücksichtigt. Die erforderlichen Gelenkbewegung und Drehmoment werden ebenfalls mit dem Übersetzungsverhältnis in die Motoren eingespeist (Anhang 12).

3.5. Encoder

Als Positionssensoren kommen absolut Drehgeber zum Einsatz. Das Signal wird mit der Auflösung des Drehzahlgebers quantisiert. Der im Modell verwendete Encoder SIKO WH5850 hat 19 Bit, folglich eine Auflösung von $\text{maxUmdrehungswinkel}/2^{19}$. Bei einer maximalen möglichen Gelenkdrehung von 2π multipliziert mit der Getriebeübersetzung bspw. $i=100$ entspricht die Auflösung $200\pi/2^{19}$ was

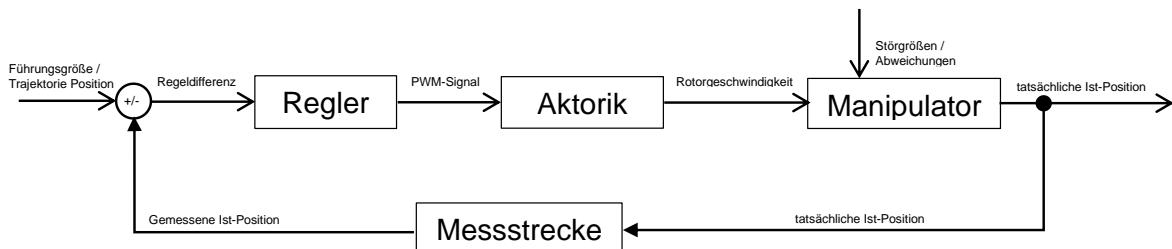
gerundet 0.0012 Radianen entspricht. Zusätzlich wird eine geringe Zeitverzögerung simuliert. Von einen zusätzlichen Störrauschen wird abgesehen, da die Motorsignale bereits starken Schwankungen unterliegen (Anhang 13).

4. Trajektorien Planung

Mit der MATLAB Funktion *trapveltraj* kann ein Trapezprofil generiert werden. Dabei wird der Funktion, einmal die Gelenkwinkel-Matrix, die Sample-Anzahl und die Peak-Velocity übergeben. Die minimale Peak Velocity kann dabei errechnet werden: $|PeakVelocity| > \frac{2|s|}{endTime}$, wobei $|s|$ der höchsten Winkeländerung der Gelenke entspricht. Die *trapveltraj*-Funktion gibt nun drei Matrizen bezüglich der Position, Geschwindigkeit und Beschleunigung mit entsprechendem Profil zurück (MathWorks, n.d.). Nach Definieren der geforderten Zeitspanne von 10 Sekunden, können diese Matrizen zum einen zur besseren Übersicht geplottet werden und zum anderen, können *timeseries* Objekte erstellt und von Simulink verwendet werden.

Für dieses Projekt sind drei vordefinierte Trajektorien verfügbar. Trajektorie eins fährt die Zwei-Meter-Bedingung an. Trajektorie zwei fährt langsam eine Position an. Trajektorie drei steuert zwei Position an und beinhaltet zudem eine Änderung der Drehrichtung, was zusätzliche Komplexität und Manövrierfähigkeit demonstriert (Anhang 14).

5. Regelkreis



In der Robotik stellen komplexe Manipulatoren mit mehreren Gelenken und beweglichen Teilen eine besondere Herausforderung für die Regelungstechnik dar. Die flüssige und präzise Bewegung solcher Systeme wird durch die Wechselwirkungen zwischen den Gelenken erschwert. Diese Wechselwirkungen, auch bekannt als nichtlineare Kopplungsterme, beschreiben die gegenseitigen Einflüsse der Gelenke aufeinander, die durch ihre Bewegungen, Trägheitskräfte und andere dynamische Faktoren verursacht werden. Diese Kopplungen können zu unerwünschten Bewegungen und Instabilitäten führen, wenn sie nicht adäquat in der Steuerung berücksichtigt werden. Ein Regelkreis, der diese Kopplungsterme bzw. das dynamische Verhalten berücksichtigt wird als zentralisierter Regelkreis beschrieben (Sicilliano et al. 2009, S. 327). In der endgültigen Projekt Ausführung konnte ein zentralisierter Ansatz nicht mehr verfolgt werden.

Im Verlauf des Projektes wurde die Ansteuerung der Gelenke von Drehmoment auf Bewegung umgestellt. Das aktuelle Model verwendet die zentralisierte Berechnung der Drehmomente nur um die Motoren mit dem Lastmoment zu belegen. Da die Gelenke allerdings entweder Drehmoment oder

Bewegung als Eingabe annehmen und nicht beides, dient dies nur der Simulation der dynamischen Verhalten der Motoren.

Die direkte Motorregelung basiert auf der Differenz zwischen der Sollposition und der gemessenen Gelenk-Positionen. Diese Regeldifferenz wird in einem Simulink-Funktionsblock gebildet, in welchem zwei if-Anweisungen abgefragt werden. Zum einen wird geprüft, ob die Zielposition negativ oder positiv ist und dementsprechend wird im oder gegen den Uhrzeigersinn kommutiert. Dies allein reicht allerdings nicht, denn bei einer Drehrichtungsänderung, liegt die Zielposition bspw. weiterhin im positiven Bereich, die Geschwindigkeit aber im negativen. Dies wird mit der Zielgeschwindigkeit abgefragt. Bei einer Drehung gegen den Uhrzeigersinn, wird die Regeldifferenz mit minus eins multipliziert.

Im Anschluss wird mit einem PI-Controller die Regelgröße für eine Sigmoidfunktion skaliert, welche in einen weiteren Funktionsbaustein definiert ist (Anhang 15).

$$u(x) = \frac{1}{1+e^{-x}} \quad (3.11)$$

Die Sigmoidfunktion nähert sich dem Wert eins bei positiv unendlichen Eingaben und strebt gegen null bei negativ unendlichen Eingaben. Das Ziel ist es, einen Wert zwischen Null und Eins für den „Duty Cycle“ der PWM-Generierung zu erzeugen. Insgesamt erwies sich diese Charakteristik als optimal, da die Motoren starken Schwankungen ausgesetzt sind. Die Periodendauer für die PWM-Generierung wird fest auf 0.0001 gesetzt. Da die Rotorgeschwindigkeit eine direkte Verbindung zur Kommutierung aufweist, könnte in einen weiteren Schritt die Periodendauer variable von dieser abhängig gemacht werden.

Das PWM-Signal wird schließlich mithilfe eines variablen Pulse Generator Blocks erzeugt und in den Motor-Subsystemen mit den Spannungen multipliziert.

Die Parameter wurden empirisch aufgestellt. Die Grenzen des Regelkreises werden im folgenden Kapitel diskutiert (Anhang 15).

6. Messung des Tracking-Fehler

Der Tracking-Fehler wird berechnet, indem die Zielkoordinaten von den aktuellen Gelenkkoordinaten subtrahiert werden. Diese Differenz wird zunächst in Radianen gemessen. Um ein intuitiveres Verständnis der Abweichungen in praktischen Maßeinheiten zu ermöglichen, werden die Gelenkkoordinaten durch Vorwärtsskinematik in kartesische Koordinaten umgerechnet. Diese Transformation erfolgt mittels eines Simulink-Funktionsblocks (Anhang 16).

Zur weiteren Analyse werden die berechneten Tracking-Fehlerwerte im MATLAB Workspace gespeichert (Anhang 16). Die durchschnittlichen Tracking-Fehler für die drei verschiedenen Trajektorien des Endeffektors werden wie folgt dargestellt:

Average	X [in mm]	Y [in mm]	Z [in mm]
Trajektorie 1	-0.1773	-0.1077	-0.0446
Trajektorie 2	-0.0656	-0.0159	-0.0103
Trajektorie 3	-0.1142	-0.1610	-0.0487

Es ist zu beachten, dass der Manipulator starken Schwankungen unterliegt, die nicht durch die Durchschnittswerte repräsentiert werden. Die (absoluten) maximalen Tracking Fehler folgenden zu:

Maximum	X [in mm]	Y [in mm]	Z [in mm]
Trajektorie 1	0.4928	3.9638	1.332
Trajektorie 2	0.7306	3.9192	1.4425
Trajektorie 3	2.3257	9.3310	3.4720

Ergänzend zu den drei Trajektorien wird eine fünfprozentige Zufallsvariation auf die Manipulator Parameter, Längen, Radien und Maßen der einzelnen Körper, ausgeführt. Dies geschieht mit der *rand()-Funktion*. Die Werte Folgen zu:

	Länge [m]	Radius [m]	Maße [kg]
Körper 1	0.4975	0.2050	19.167
Körper 2	1.0413	0.0973	9.6524
Körper 3	1.5058	0.0516	5.2481

Die Parameter werden ausschließlich am Simulink Model verändert. Die Werte in den Regelbausteinen bleiben unverändert. Die durchschnittlichen Tracking-Fehler folgen nun zu:

	X [in mm]	Y [in mm]	Z [in mm]
Trajektorie 1	-0.1780	-0.1074	-0.0446
Trajektorie 2	-0.0662	-0.0169	-0.0099
Trajektorie 3	-0.1143	-0.2384	-0.0577

Für eine detaillierte grafische Darstellung dieser Ergebnisse siehe Anhang 17 bis 22.

6.2. Diskussion des Tracking-Fehler

Bei einer genaueren Analyse der Tracking-Fehler-Diagramme treten mehrere bemerkenswerte Muster auf. Zu Beginn und am Ende der Simulation sind die größten Abweichungen festzustellen. Während die initialen Diskrepanzen durch das Anlaufverhalten der Aktorik begründet werden können, zeigt sich am Ende der Simulation ein anderes Phänomen: Die Ist-Position übersteigt die Soll-Position, was eine negative Regeldifferenz zur Folge hat. Der Regelkreis reagiert darauf mit einem Duty Cycle von null, was zwar ein weiteres Anwachsen der Differenz verhindert, jedoch nicht dazu beiträgt, diese aktiv zu reduzieren (Anhang 23). Um die Positions korrektur zu ermöglichen, wäre eine Aktivierung der CCW-Kommutierung notwendig, damit der Motor in entgegengesetzter Richtung

drehen kann. Dies könnte bspw. mit einer weiteren if-Anweisung passieren, die abfragt, ob die Rotorgeschwindigkeit kleiner als ein definierter Grenzwert ist und die Regeldifferenz negativ ist. Ein weiteres Hauptproblem besteht in den durch die BLDC-Motoren erzeugten Schwankungen, welche das gesamte Modell beeinflussen. In einem realen System könnten Mechanismen wie Getriebe und Kupplungen diese Schwankungen zumindest teilweise ausgleichen. Innerhalb dieses Projekts war es jedoch nicht möglich, diese Effekte mit „Moving Average“-Bausteinen adäquat nachzubilden. Eine detailliertere Modellierung der Transmission könnte dahingehend eine Lösung bieten.

7. Projektevaluation

Abschließend kann durch die Projektziele und Durchführung iteriert.

Es wurde deutlich, dass nicht alle Projektziele vollständig erreicht werden konnten. Insbesondere wurde keine zentralisierte Regelung entwickelt. Es ist jedoch zu beachten, dass in diesem Projekt sowohl die Rotorgeschwindigkeit als auch das Drehmoment an den Motoren erfolgreich simuliert wurden. Ein zentralisierter Regelkreis, der nicht nur das dynamische Verhalten eines Manipulators, sondern auch das von dreiphasigen BLDC-Motoren berücksichtigt, stellt ein äußerst umfassendes und komplexes Unterfangen dar.

Es lässt sich argumentieren, dass die Detailtiefe in den Bereichen Transmission und Sensorik nicht ausreichend ist. In Anbetracht des Projektumfangs wurden diese Aspekte jedoch bewusst nicht weiter vertieft.

Insgesamt konnte aber ein Manipulator entwickelt werden, der durch die modellierte Aktorik, durchaus in der Lage ist, vorgegebene Trajektorien abzufahren. Ermittelt man den durchschnitts Tracking-Fehler, wird das Potenzial sichtbar.

Ein weiteres Problem, welches immer wieder im Verlauf des Projektes auftrat, ist die Simulationsdauer. Daher wurde bspw. die powergui diskretisiert und die maximale Solver-Step-Size und die Anzahl der Polpaare beschränkt. Dies könnte auch Auswirkungen auf das Simulationsergebnis haben.

Literaturverzeichnis:

- Baba, M. A., Naoui, M., Cherkaoui, M. (2023). *Digital Technologies and Applications. Modeling and Simulation of a BLDC Motor Speed Control in Elekctic Vehicles*. Springer Nature Switzerland AG. Vol. 1.
- Momenzadeh, M. M., Ahmed, A. F., Tolba, A. (2013). *Modelling and Simulation Of The BLDC Electric Drive System Using SIMULINK/MATLAB for a Hybrid Vehicle*. Universität Paderborn.
- Sicilliano, B., Sciavicco, L., Villani, L., Oriolo, G., (2009). Robotics. *Modelling, Planning and Control*. Springer-Verlag London Limited. 10.1007/978-1-84628-642-1
- Kasdin, N. J. & Paley, D. A. (2011). Engineering Dynamics. *A Comprehensive Intoduction*. Princeton University Press. ISBN 978-0-691-13537-3.
- The MathWorks, Inc. (n.d.). Design Trajectory with Velocity Limits Using Trapezoidal Velocity Profile [<https://de.mathworks.com/help/robotics/ug/design-a-trajectory-with-velocity-limits-using-a-trapezoidal-velocity-profile.html>]
- The MathWorks, Inc. (n.d.). Robotics System Toolbox. *Entwicklung, Simulation, Test und Bereitstellung von Robotik-Anwendungen*. [<https://de.mathworks.com/products/robotics.html>]
- The MathWorks, Inc. (n.d.). Symbolic Math Toolbox. *Ausführung symbolischer mathematischer Berechnungen*. [<https://de.mathworks.com/products/symbolic.html>]
- The MathWorks, Inc. (n.d.). Teaching Rigid Body Dynamics. *Two-Degrees-of-Freedom Non-Planar Robotic Manipulator Case Study | Teaching Rigid Body Dynamics, Part 3*. [<https://de.mathworks.com/videos/teaching-rigid-body-dynamics-part-3-two-degrees-of-freedom-non-planar-robotic-manipulator-case-study-1509356325784.html>]
- The MathWorks, Inc. (n.d.). timeseries. *Create timeseries object*. [<https://de.mathworks.com/help/matlab/ref/timeseries.html>]
- Wikimedia Foundation Inc. (2022). Rotationsenergie. [<https://de.wikipedia.org/wiki/Rotationsenergie>]
- Wikimedia Foundation Inc. (2023). Kinetische Energie. [https://de.wikipedia.org/wiki/Kinetische_Energie]
- Wikimedia Foundation Inc. (2023). Liste von Trägheitstensoren. [https://de.wikipedia.org/wiki/Liste_von_Tr%C3%A4gheitstensoren]
- Wikimedia Foundation Inc. (2023). Potentielle Energie. [https://de.wikipedia.org/wiki/Potentielle_Energie]
- Yamashita, R. Y., Silva, F. L., Santiciolli, F. M., Eckert, J. J., Dedini, F. G. (2018). *Comparison between two models of BLDC motor, simulation and data Acquisition*. Journal of the Brazilian Society of Mechanical Sciences and Engineering. Vol. 40.

Verzeichnis der Anhänge:

Anhang 1: Kinetische Energien	13
Anhang 2: Kinetische Energien	14
Anhang 3: Potenzielle Energien	15
Anhang 4: Generalisierte Kräfte	16
Anhang 5: Inverter-Schaltung.....	17
Anhang 6: Auswertung und Erzeugung der Hall Signale	18
Anhang 7: Kommutierung nach Hall Signalen	19
Anhang 8: Spannungen, BEMF-Signal, Hall-Signal.....	20
Anhang 8.1: Motor-Strom, Motor-Drehmoment	21
Anhang 9: Übersicht BLDCM Subsystem.....	22
Anhang 10: BEMF Block.....	23
Anhang 11: Rotor-Speed-Calculation	24
Anhang 12: Getriebe	25
Anhang 13: Encoder	26
Anhang 14: Trajektorien	27
Anhang 15: Regelkreis Differenzbildung, Drehrichtung und Sigmoidfunktion	28
Anhang 16: Tracking-Fehler Erfassung	29
Anhang 17: Tracking-Fehler Trajektorie 1	30
Anhang 18: Tracking-Fehler Trajektorie 2	31
Anhang 19: Tracking-Fehler Trajektorie 3	32
Anhang 20: Tracking-Fehler Trajektorie 1 mit Zufallsvariation	33
Anhang 21: Tracking-Fehler Trajektorie 2 mit Zufallsvariation	34
Anhang 22: Tracking-Fehler Trajektorie 3 mit Zufallsvariation	35
Anhang 23: Diskussion Tracking Fehler	36
Anhang 24: Solver Einstellungen	37
Anhang 25: Manipulator Geometrie	38
Anhang 26: Motor Datenblatt	39

Anhang 1: Kinetische Energien

```
%-----
% Inertia Cylinder
%-----
disp("%----- Inertia Cylinder -----")
% basic cylindrical inertia matrix with central mass point https://de.wikipedia.org/wiki/Liste_von_Tr%C3%A4gheitstensoren
inertiaMatrix = @(m, l, r) [m*(3*r^2 + l^2)/12, 0, 0;
                           0, m*(3*r^2 + l^2)/12, 0;
                           0, 0, m*(r^2)/2];
% setting up the inertia tensor matrices
inertiaLink0 = inertiaMatrix(m0, l0, r0)
inertiaLink1 = inertiaMatrix(m1, l1, r1)
inertiaLink2 = inertiaMatrix(m2, l2, r2)
%-----
% Translational Velocity of Mass in the World Frame
%-----
disp("%----- Translational Velocity of Mass -----")
% location of mass 0
WF_PosM0 = [0; 0; 10/2];
WF_PosM0 = rotationZAxis(theta0(t))*WF_PosM0
% linear velocity of mass 0
WF_TransVelM0 = diff( WF_PosM0, t);
WF_TransVelM0
% location of mass 1
WF_PosM1 = [(l1*cos(theta1(t)))/2; 0; 10 + (l1*sin(theta1(t)))/2];
WF_PosM1 = rotationZAxis(theta0(t))*WF_PosM1
% linear velocity of mass 1
WF_TransVelM1 = diff( WF_PosM1, t);
WF_TransVelM1
% location of mass 2
WF_PosM2 = [l1*cos(theta1(t)) + (l2*cos(theta1(t)+theta2(t)))/2; 0; 10 +
l1*sin(theta1(t)) + (l2*sin(theta1(t)+theta2(t)))/2];
WF_PosM2 = rotationZAxis(theta0(t)) * WF_PosM2
% linear velocity of mass 2
WF_TransVelM2 = diff( WF_PosM2, t);
WF_TransVelM2
```

Anhang 2: Kinetische Energien

```
%-----
% Translation Energy
%-----
disp("%----- Translation Energy -----")
% calculate the translational energy
KE_trans = translationalEnergy(m0, WF_TransVelM0) + translationalEnergy(m1,
WF_TransVelM1) + translationalEnergy(m2, WF_TransVelM2)
%-----
% Rotational Velocity of Mass in the Base Frame
%-----
disp("%----- Angular Velocity of Mass -----")
% The angular velocity is transformed to each link's local frame to simplify
calculations.
% This approach avoids the need to transform the inertia matrix.
% transform dh
transformLinkToLink0 = [1,0,0;
                      0, 0,1;
                      0,-1,0];
% local body frame already in alignment
LF_angVelM0 = [0; 0; diff(theta0(t), t)]
% rotate into body frame
LF_angVelM1 = rotationYAxis(-theta1(t))*LF_angVelM0+transformLinkToLink0.*[
0; 0; diff(theta1(t)) ]
LF_angVelM2 = rotationYAxis(-(theta1(t)+theta2(t)))*LF_angVelM0+transform-
LinkToLink0.*[ 0; 0; diff(theta1(t)+theta2(t)) ]
%-----
% Rotation Energy
%-----
disp("%----- Rotation Energy -----")
% calculate the rotational energy
KE_rot = rotationEnergy(LF_angVelM0, inertiaLink0) + rota-
tionEnergy(LF_angVelM1, inertiaLink1) + rotationEnergy(LF_angVelM2, iner-
tiaLink2)
%-----
% Kinetic Energy
%-----
disp("%----- Kinetic Energy -----")
KE_total = KE_trans + KE_rot
%-----
function KE_trans = translationalEnergy(m, transVel)
    KE_trans = 1/2*(transVel')*m*transVel;
End
function KE_rot = rotationEnergy(angVel, Inertia)
    KE_rot = 1/2*(angVel')*Inertia*angVel;
end
```

Anhang 3: Potenzielle Energien

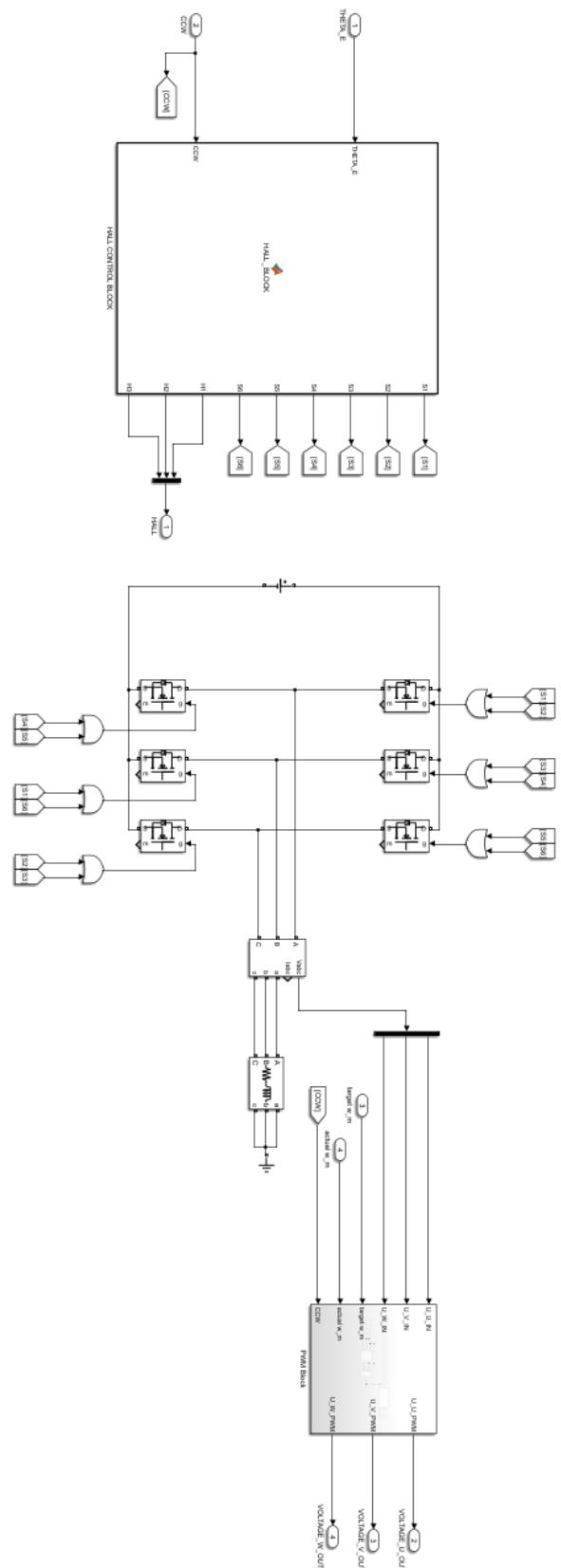
```
%-----
% Potential Energy
%-----
disp("%----- Potential Energy -----")
% calculate the potential energy
PE_total = potentialEnergy(g, m0, WF_PosM0(3)) + potentialEnergy(g, m1,
WF_PosM1(3)) + potentialEnergy(g, m2, WF_PosM2(3))

function PE_part = potentialEnergy(g, m, heightZ)
    PE_part = g*m*heightZ;
end
```

Anhang 4: Generalisierte Kräfte

```
%-----
% Generalized forces
%-----
disp("%----- Generalized forces -----")
% Switching to generalized variables
LF_angVelM0 = subs(LF_angVelM0, angular_variables, generalized_variables);
LF_angVelM1 = subs(LF_angVelM1, angular_variables, generalized_variables);
LF_angVelM2 = subs(LF_angVelM2, angular_variables, generalized_variables);
% Setting up the torque matrix for joint 0
tauM0Matrix = [ ...
    0,           0,           0,           0;
    0,           0,           (-b1*dq1),   (t1);
    (t0),        (-b0*dq0),  0,           0; ];
% Setting up the associated angular velocitys of joint 0
angVelM0Matrix = [LF_angVelM0, LF_angVelM0, LF_angVelM0, LF_angVelM0];
% Setting up the torque matrix for joint 1
tauM1Matrix = [ ...
    0,           0,           0,           0;
    (-t1),      (b1*dq1),   (t2),        (-b2*dq2);
    0,           0,           0,           0; ];
% Setting up the associated angular velocitys of joint 1
angVelM1Matrix = [LF_angVelM1, LF_angVelM1, LF_angVelM1, LF_angVelM1];
% Setting up the torque matrix for joint 2
tauM2Matrix = [ ...
    0,           0;
    (-t2),      (b2*dq2);
    0,           0; ];
% Setting up the associated angular velocitys of joint 2
angVelM2Matrix = [LF_angVelM2, LF_angVelM2];
% Setting up the torque matrix for all joint
tauMatrix = [tauM0Matrix, tauM1Matrix, tauM2Matrix];
% Setting up the associated angular velocitys of all joints
angVelMatrix = [angVelM0Matrix, angVelM1Matrix, angVelM2Matrix];
% calling the generalizedForces function to calculate the generalized forces
Q2 = generalizedForces(tauMatrix, angVelMatrix, dq2);
Q1 = generalizedForces(tauMatrix, angVelMatrix, dq1);
Q0 = generalizedForces(tauMatrix, angVelMatrix, dq0);
% Switching to base variables
Q2 = subs(Q2, generalized_variables, angular_variables)
Q1 = subs(Q1, generalized_variables, angular_variables)
Q0 = subs(Q0, generalized_variables, angular_variables)
```

Anhang 5: Inverter-Schaltung



Anhang 6: Auswertung und Erzeugung der Hall Signale

```
function [S1, S2, S3, S4, S5, S6, H1, H2 ,H3] = HALL_BLOCK(THETA_E, CCW)
%#codegen

% Normalisierung
%THETA_E = mod(THETA_E, 2*pi);
% Initialisieren Sie alle Signale auf 0

THETA = mod(THETA_E, 2*pi);

% electrical theta 0°-60°
if (0 <= THETA) && (THETA < 1*pi/3)
    H1 = 1;
    H2 = 0;
    H3 = 1;

% electrical theta 60°-120°
elseif (1*pi/3 <= THETA) &&(THETA < 2*pi/3)
    H1 = 1;
    H2 = 0;
    H3 = 0;
% electrical theta 120°-180°
elseif (2*pi/3 <= THETA) && (THETA < pi)
    H1 = 1;
    H2 = 1;
    H3 = 0;

% electrical theta 180°-240°
elseif (pi <= THETA) && (THETA < 4*pi/3)
    H1 = 0;
    H2 = 1;
    H3 = 0;

% electrical theta 240°-300°
elseif (4*pi/3 <= THETA) && (THETA < 5*pi/3)
    H1 = 0;
    H2 = 1;
    H3 = 1;

% electrical theta 300°-360°
elseif (5*pi/3 <= THETA) && (THETA < 2*pi)
    H1 = 0;
    H2 = 0;
    H3 = 1;

% something went wrong
else
    H1 = 0;
    H2 = 0;
    H3 = 0;
End

[Anhang 7...]
```

Anhang 7: Kommutierung nach Hall Signalen

```
function [S1, S2, S3, S4, S5, S6, H1, H2 ,H3] = HALL_BLOCK(THETA_E, CCW)

[Anhang 5...]

if CCW ~= 1
    if (H1 == 1) && (H3 == 1)
        S3 = 0; S2 = 0; S4 = 0; S5 = 0; S6 = 0;
        S1 = 1;
    elseif (H1 == 1) && (H2 ~= 1 && H3 ~= 1)
        S1 = 0; S4 = 0; S3 = 0; S5 = 0; S6 = 0;
        S2 = 1;
    elseif (H1 == 1) && (H2 == 1)
        S1 = 0; S2 = 0; S5 = 0; S4 = 0; S6 = 0;
        S3 = 1;
    elseif (H2 == 1) && (H1 ~= 1 && H3 ~= 1)
        S1 = 0; S2 = 0; S3 = 0; S6 = 0; S5 = 0;
        S4 = 1;
    elseif (H2 == 1) && (H3 == 1)
        S2 = 0; S3 = 0; S4 = 0; S1 = 0; S6 = 0;
        S5 = 1;
    elseif (H3 == 1) && (H1 ~= 1 && H2 ~= 1)
        S1 = 0; S3 = 0; S4 = 0; S5 = 0; S2 = 0;
        S6 = 1;
    else
        S1 = 0; S2 = 0; S3 = 0; S4 = 0; S5 = 0; S6 = 0;
    end
else
    if (H1 == 1) && (H3 == 1)
        S3 = 0; S2 = 0; S1 = 0; S5 = 0; S6 = 0;
        S4 = 1;
    elseif (H1 == 1) && (H2 ~= 1 && H3 ~= 1)
        S1 = 0; S4 = 0; S3 = 0; S2 = 0; S6 = 0;
        S5 = 1;
    elseif (H1 == 1) && (H2 == 1)
        S1 = 0; S2 = 0; S5 = 0; S4 = 0; S3 = 0;
        S6 = 1;
    elseif (H2 == 1) && (H1 ~= 1 && H3 ~= 1)
        S4 = 0; S2 = 0; S3 = 0; S6 = 0; S5 = 0;
        S1 = 1;
    elseif (H2 == 1) && (H3 == 1)
        S5 = 0; S3 = 0; S4 = 0; S1 = 0; S6 = 0;
        S2 = 1;
    elseif (H3 == 1) && (H1 ~= 1 && H2 ~= 1)
        S1 = 0; S6 = 0; S4 = 0; S5 = 0; S2 = 0;
        S3 = 1;
    else
        S1 = 0; S2 = 0; S3 = 0; S4 = 0; S5 = 0; S6 = 0;
    end
end
end
```

Anhang 8: Spannungen, BEMF-Signal, Hall-Signal

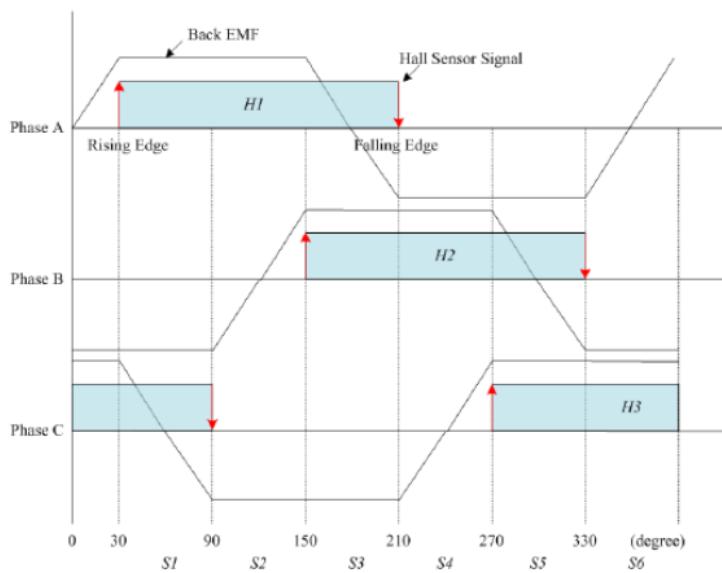
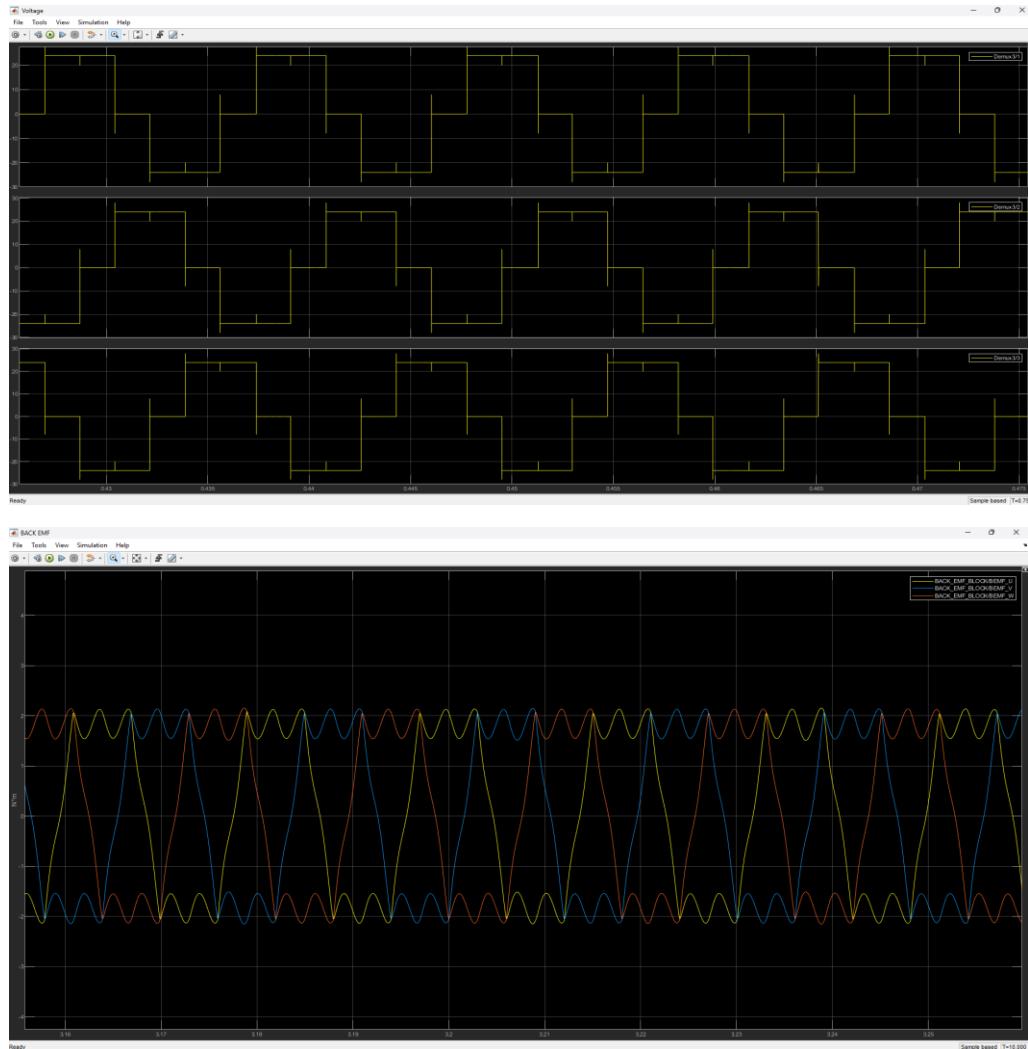
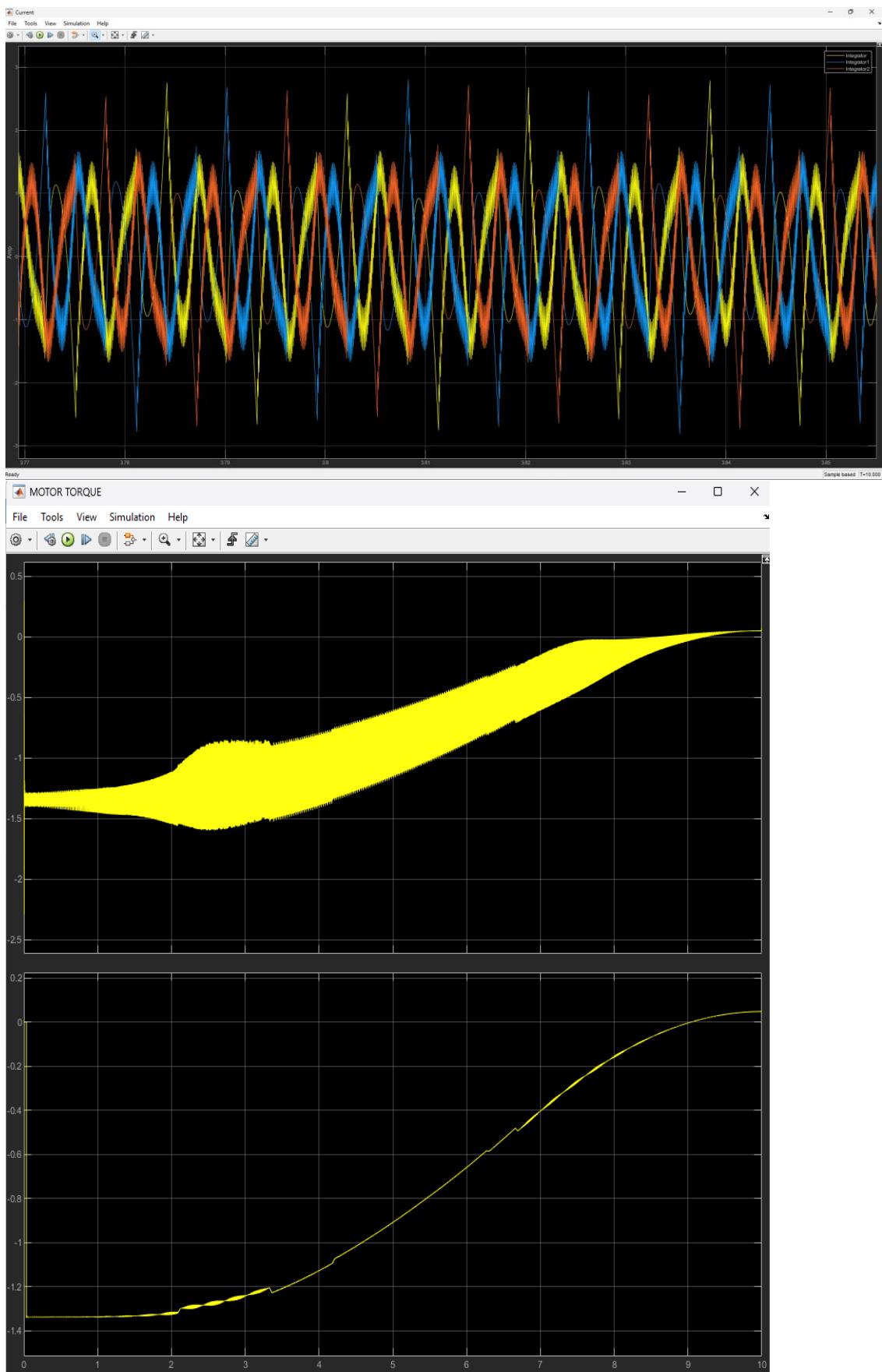
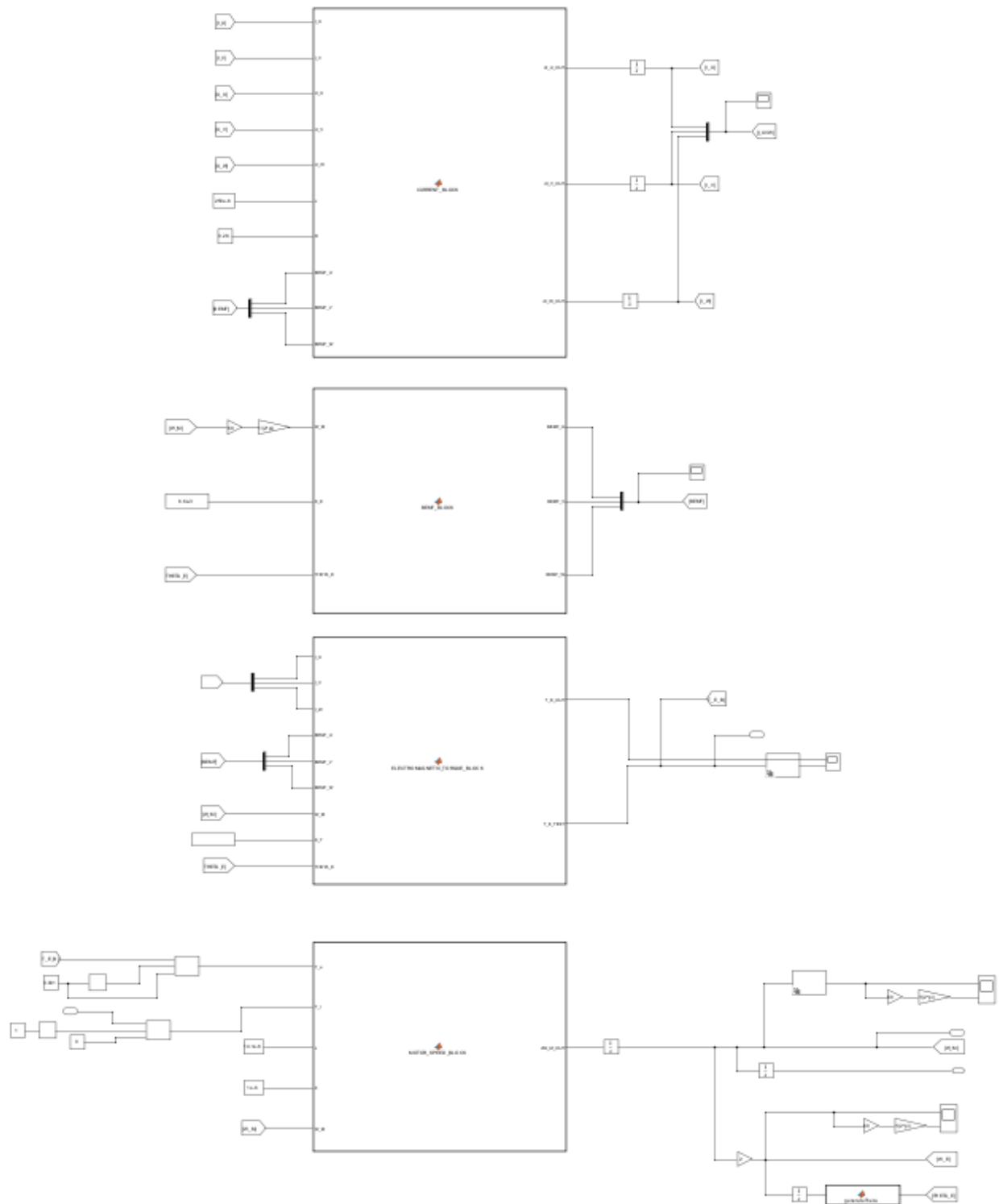


Abbildung 2: Back EMF and hall-effect sensor signal

Anhang 8.1: Motor-Strom, Motor-Drehmoment



Anhang 9: Übersicht BLDCM Subsystem



Anhang 10: BEMF Block

```
function [BEMF_U, BEMF_V, BEMF_W] = BEMF_BLOCK(W_M, K_E, THETA_E)
%%codegen

% This function was generated by the Symbolic Math Toolbox version 23.2.
% 20-Mar-2024 19:08:31

%THETA = mod(THETA_E, 2*pi);
THETA = THETA_E;
% electrical theta 0°-60°
if (0 <= THETA) && (THETA < 1*pi/3)
    F_U = 1;
    F_V = -1;
    F_W = 1-6*THETA/pi;

% electrical theta 60°-120°
elseif (1*pi/3 <= THETA) &&(THETA < 2*pi/3)
    F_U = 1;
    F_V = 6*THETA/pi - 3;
    F_W = -1;

% electrical theta 120°-180°
elseif (2*pi/3 <= THETA) && (THETA < pi)
    F_U = -6*THETA/pi + 5;
    F_V = 1;
    F_W = -1;

% electrical theta 180°-240°
elseif (pi <= THETA) && (THETA < 4*pi/3)
    F_U = -1;
    F_V = 1;
    F_W = 6*THETA/pi - 7;

% electrical theta 240°-300°
elseif (4*pi/3 <= THETA) && (THETA < 5*pi/3)
    F_U = -1;
    F_V = -6*THETA/pi + 9;
    F_W = 1;

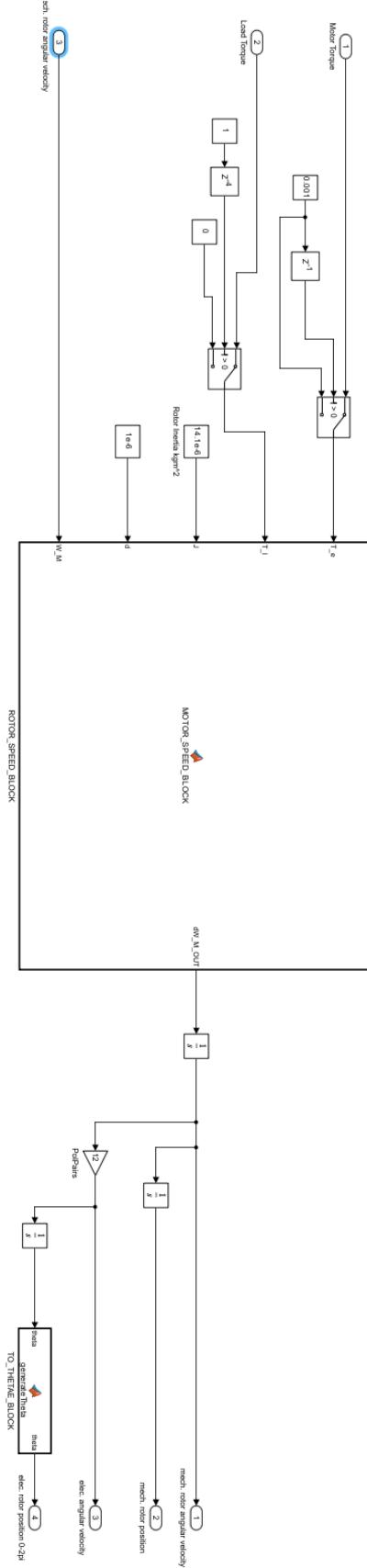
% electrical theta 300°-360°
elseif (5*pi/3 <= THETA) && (THETA < 2*pi)
    F_U = 6*THETA/pi - 11;
    F_V = -1;
    F_W = 1;

% something went wrong
else
    F_U = NaN;
    F_V = NaN;
    F_W = NaN;
end

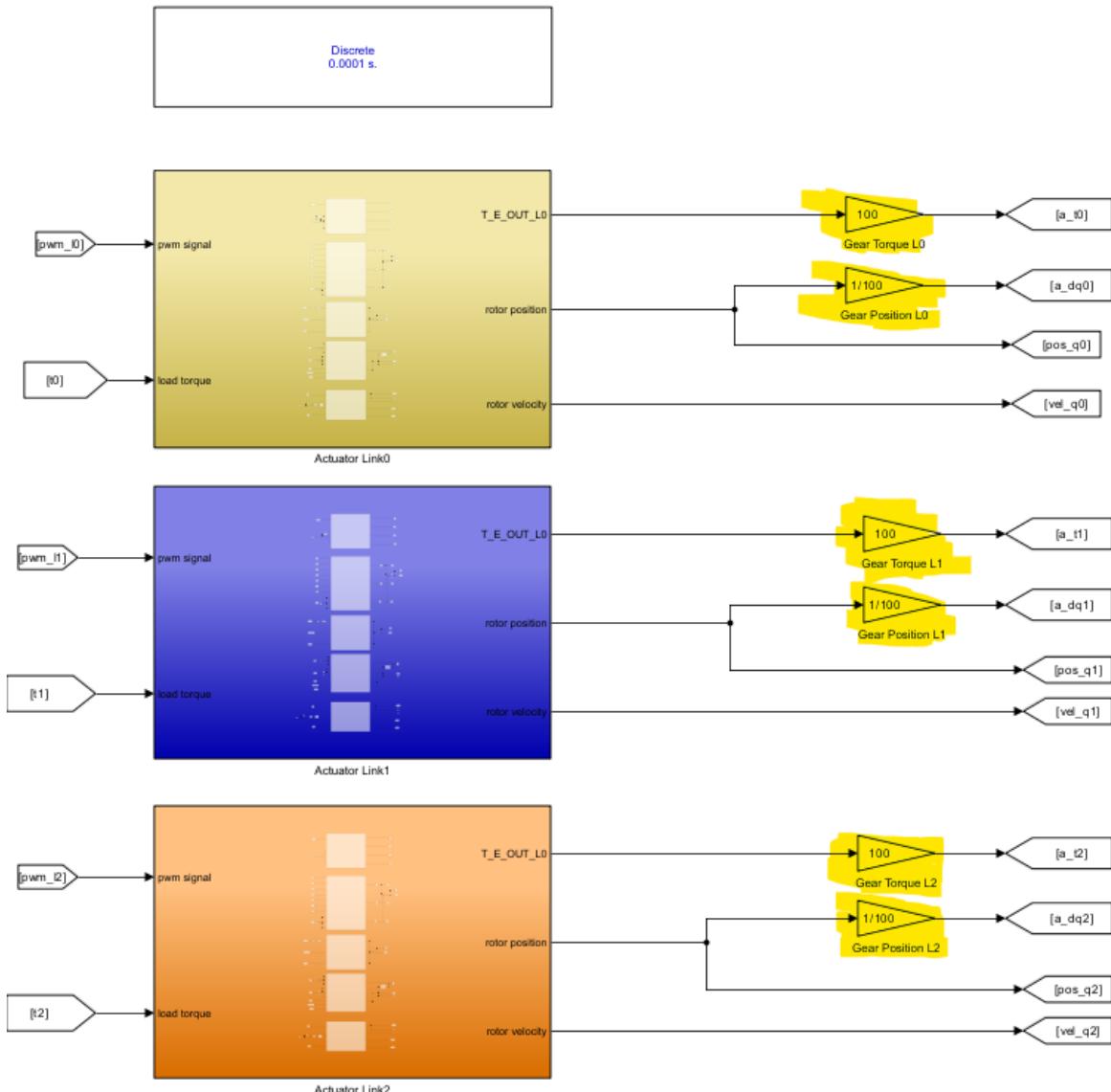
BEMF_U = K_E*W_M*F_U;
BEMF_V = K_E*W_M*F_V;
BEMF_W = K_E*W_M*F_W;

end
```

Anhang 11: Rotor-Speed-Calculation



Anhang 12: Getriebe



Anhang 13: Encoder

```
function encoder_output = simulateEncoder(actual_position)

    % https://www.siko-global.com/de-de/produkte/rotoline-drehgeber/absolute-
    % drehgeber/wh5850
    % Auflösung wäre theoretisch 2pi/2^19 / 200pi/2^19
    resolution = 0.0012;
    delay = 0.0001;
    dt = 2e-4;
    %noise_level = 0.0001;

    % quantizes
    quantized_position = round(actual_position / resolution) * resolution;

    % delay
    persistent position_queue;
    if isempty(position_queue)
        position_queue = repmat(quantized_position, 1, round(delay/dt));
    end
    encoder_output = position_queue(1)/100;
    position_queue = [position_queue(2:end), quantized_position];

end
```

Anhang 14: Trajektorien

```
for i = 1:3
    if i == 1
        % Testing 2m Condition
        %angularMatrix = [0,0,0; 0, 0.5110, -1.2310];
        angularMatrix = [0,0,0; 1.5708,      -1.9325,      1.2125];
        peakVelocity = 2*findMaxDifference(angularMatrix)/10;
    elseif i == 2
        % Rotation to the (right) positiv q0
        angularMatrix = [0,0,0; pi/1.2, -5*pi/18, 5*pi/15];
        peakVelocity = 2*findMaxDifference(angularMatrix)/10;
    else
        % Rotation to the (left) negativ q0, and a return to the positiv q0
        angularMatrix = [0,0,0; -pi/2, -pi/2, pi/2; pi/2, -5*pi/18, 5*pi/15];
        peakVelocity = 6*findMaxDifference(angularMatrix)/10;
    end
% Reference to MATLAB documentation for designing trajectories with velocity
limits using a trapezoidal velocity profile.
% https://de.mathworks.com/help/robotics/ug/design-a-trajectory-with-velocity-
limits-using-a-trapezoidal-velocity-profile.html
% Trapez Profil PeakVelocity > (2*s/endtime) 0.16
[trapezPosition, trapezVelocity, trapezAcceleration, tSamples, pp] =
trapveltraj(angularMatrix', 1000, 'PeakVelocity', peakVelocity);
trapezPosition = trapezPosition'
trapezVelocity = trapezVelocity';
trapezAcceleration = trapezAcceleration';
% Create a series of time points based on the desired total duration.
desiredDuration = 10;
numberoftimePoints = linspace(0, desiredDuration, size(trapezPosition, 1))

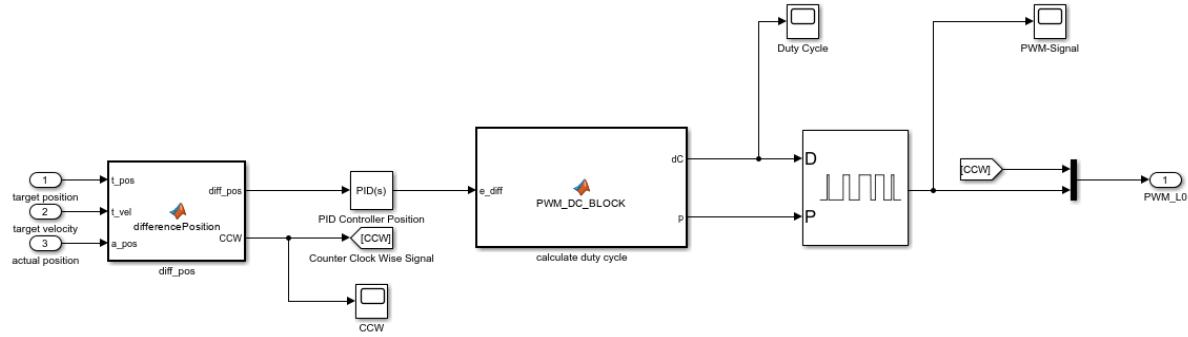
....  

if i == 1
    % Create time series objects for simulink simulation.
    ts1_q0_pos = timeseries(trapezPosition(:,1), numberoftimePoints);
    ts1_q1_pos = timeseries(trapezPosition(:,2), numberoftimePoints);
    ts1_q2_pos = timeseries(trapezPosition(:,3), numberoftimePoints);

    ts1_q0_vel = timeseries(trapezVelocity(:,1), numberoftimePoints);
    ts1_q1_vel = timeseries(trapezVelocity(:,2), numberoftimePoints);
    ts1_q2_vel = timeseries(trapezVelocity(:,3), numberoftimePoints);

    ts1_q0_acc = timeseries(trapezAcceleration(:,1), numberoftimePoints);
    ts1_q1_acc = timeseries(trapezAcceleration(:,2), numberoftimePoints);
    ts1_q2_acc = timeseries(trapezAcceleration(:,3), numberoftimePoints);
....
```

Anhang 15: Regelkreis Differenzbildung, Drehrichtung und Sigmoidfunktion



Inside diff_pos:

```
function [diff_pos, CCW] = differencePosition(t_pos, t_vel, a_pos)

if t_pos >= 0
    diff_pos = t_pos - a_pos;
    CCW = 0;
else
    if t_pos < 0
        diff_pos = (t_pos - a_pos)*(-1);
        CCW = 1;
    end
end
else
    diff_pos = (t_pos - a_pos)*(-1);
    CCW = 1;
    if t_pos > 0
        diff_pos = (t_pos - a_pos);
        CCW = 0;
    end
end
end
```

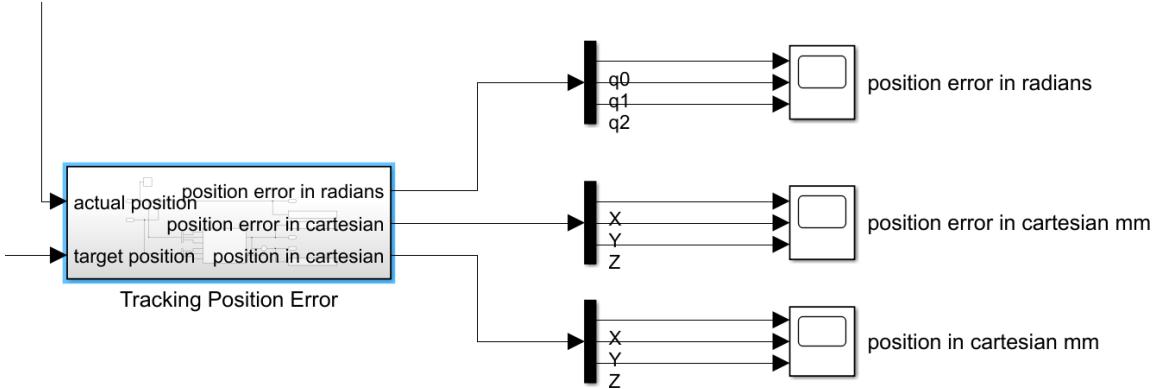
Inside calculate duty cycle:

```
function [dC, p] = PWM_DC_BLOCK(e_diff)

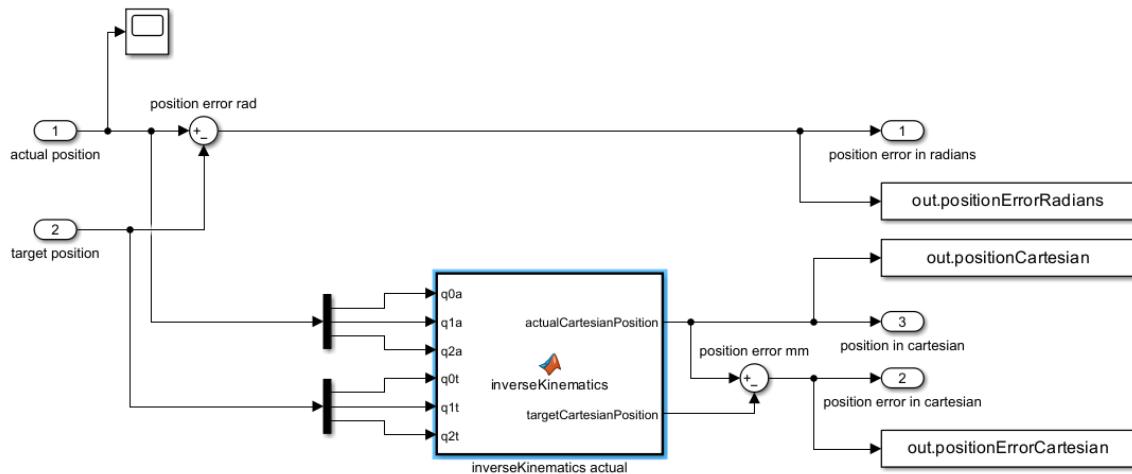
p = 0.0001;
%p = max(1e-4, min(0.001, p));

% Sigmoidfunktion
dC = 1/(1+exp(-e_diff));
end
```

Anhang 16: Tracking-Fehler Erfassung



Inside Tracking Position Error:



Inside inverseKinematics actual:

```

function [actualCartesianPosition, targetCartesianPosition] =
inverseKinematics(q0a, q1a, q2a, q0t, q1t, q2t)
    % Beispiel: Linear von 0 bis 2pi über 10 Sekunden
10 = 500;
11 = 1000;
12 = 1000;

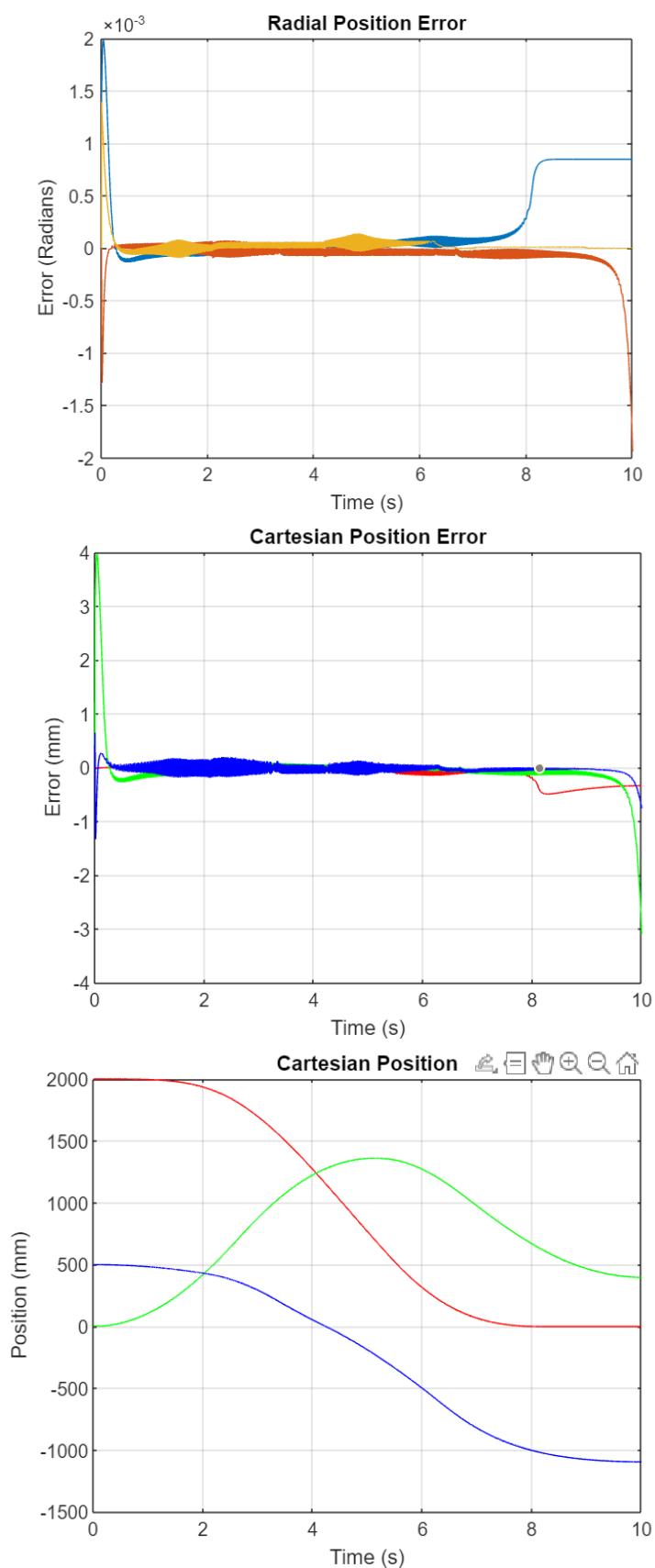
actualCartesianPosition = [cos(q0a)*(l1*cos(q1a)+l2*cos(q1a+q2a));
                           sin(q0a)*(l1*cos(q1a)+l2*cos(q1a+q2a));
                           10+l1*sin(q1a)+l2*sin(q1a+q2a)];

targetCartesianPosition = [cos(q0t)*(l1*cos(q1t)+l2*cos(q1t+q2t));
                           sin(q0t)*(l1*cos(q1t)+l2*cos(q1t+q2t));
                           10+l1*sin(q1t)+l2*sin(q1t+q2t)];

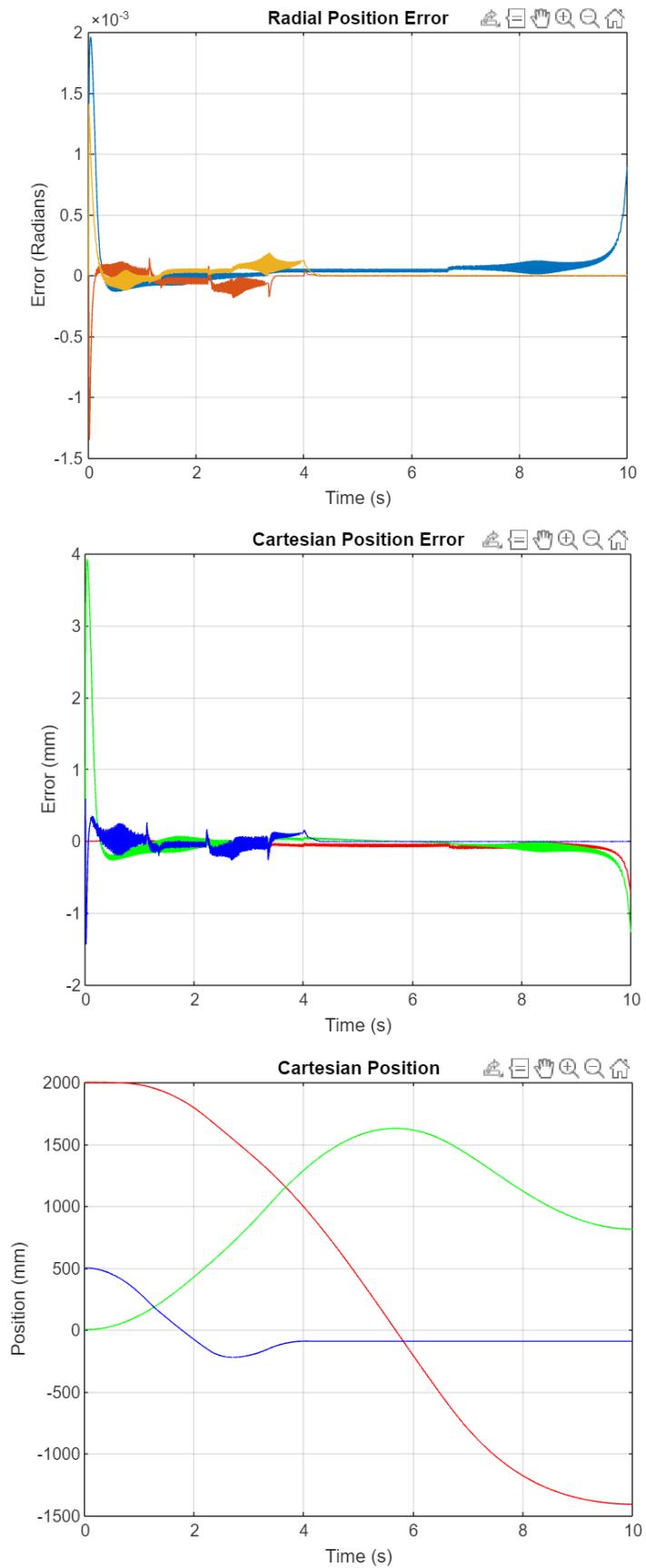
end

```

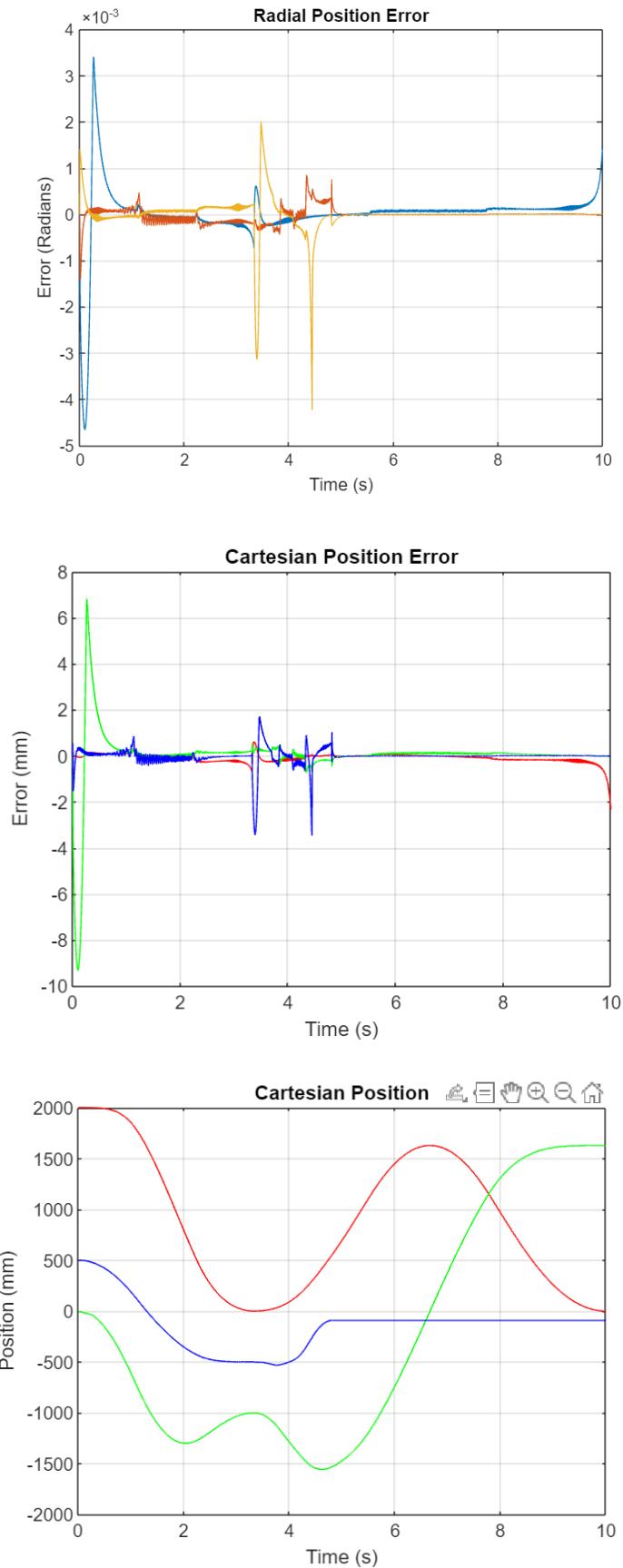
Anhang 17: Tracking-Fehler Trajektorie 1



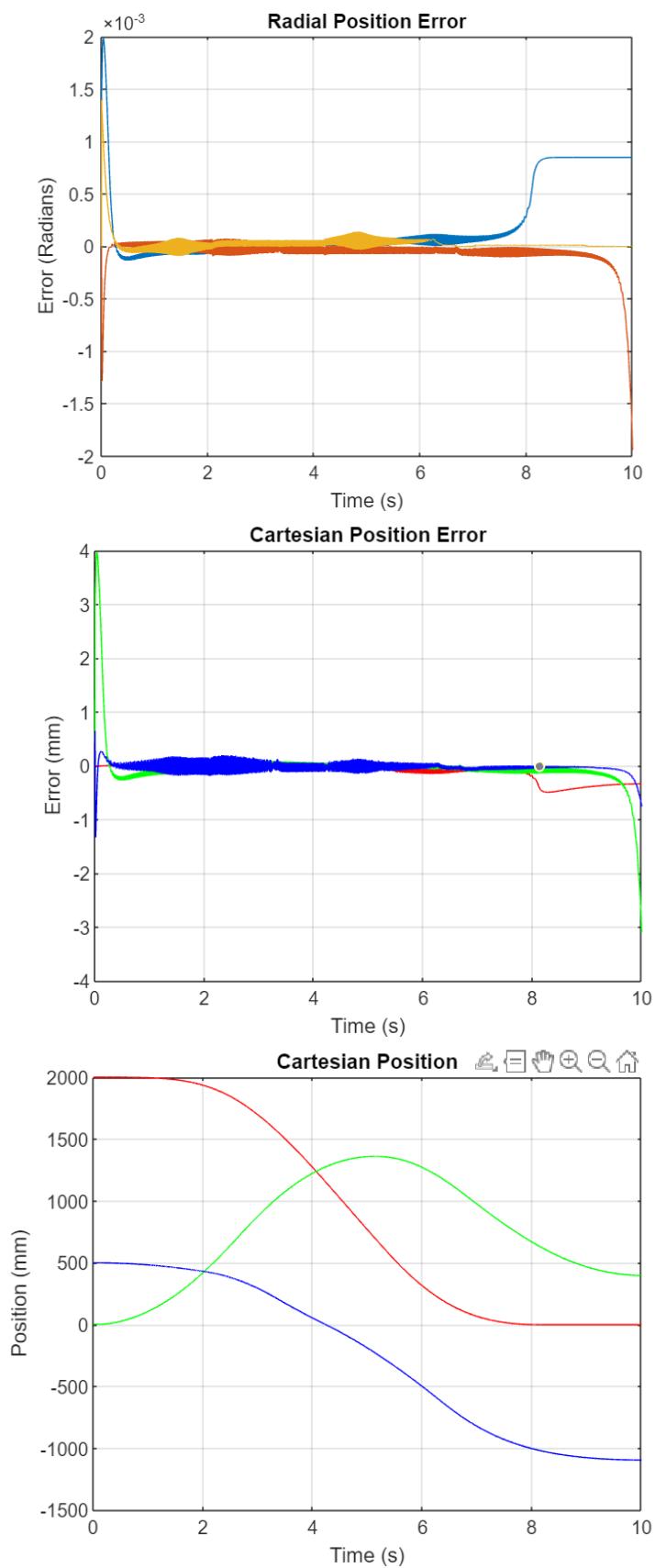
Anhang 18: Tracking-Fehler Trajektorie 2



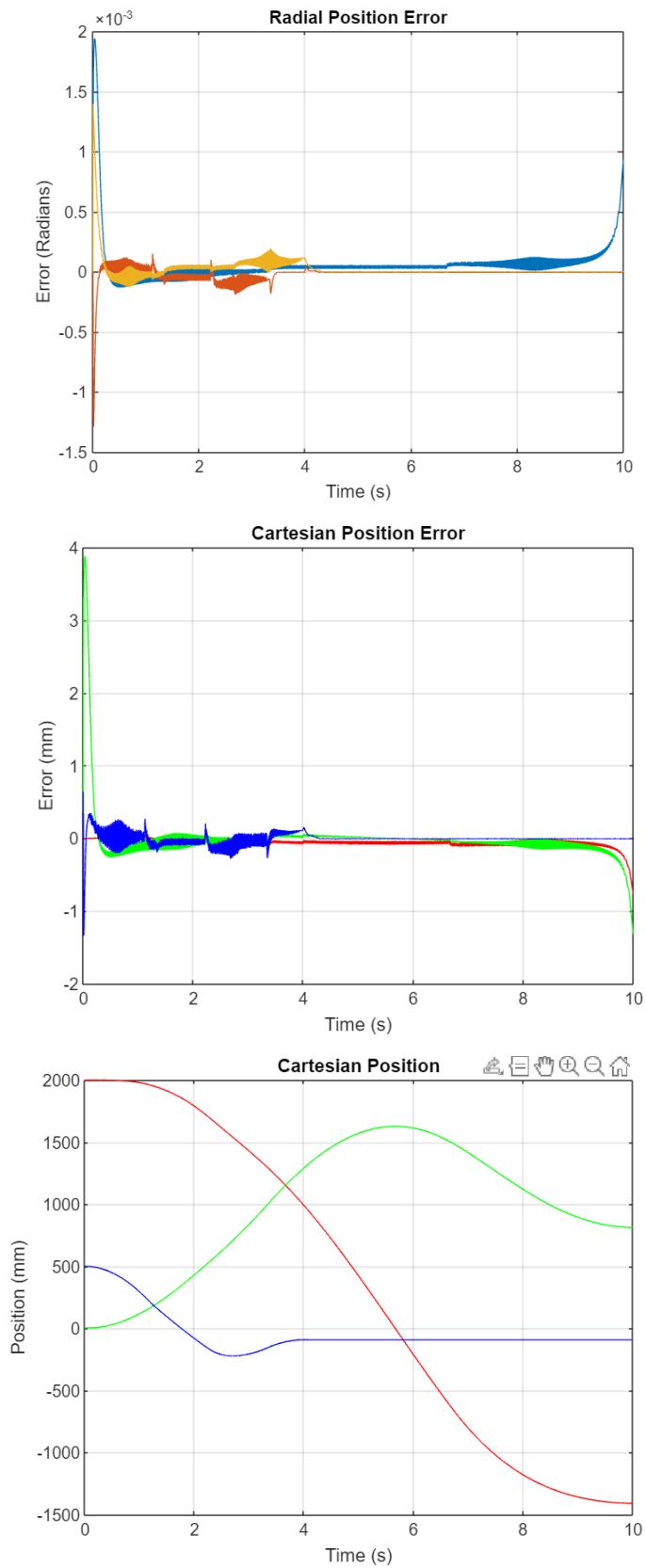
Anhang 19: Tracking-Fehler Trajektorie 3



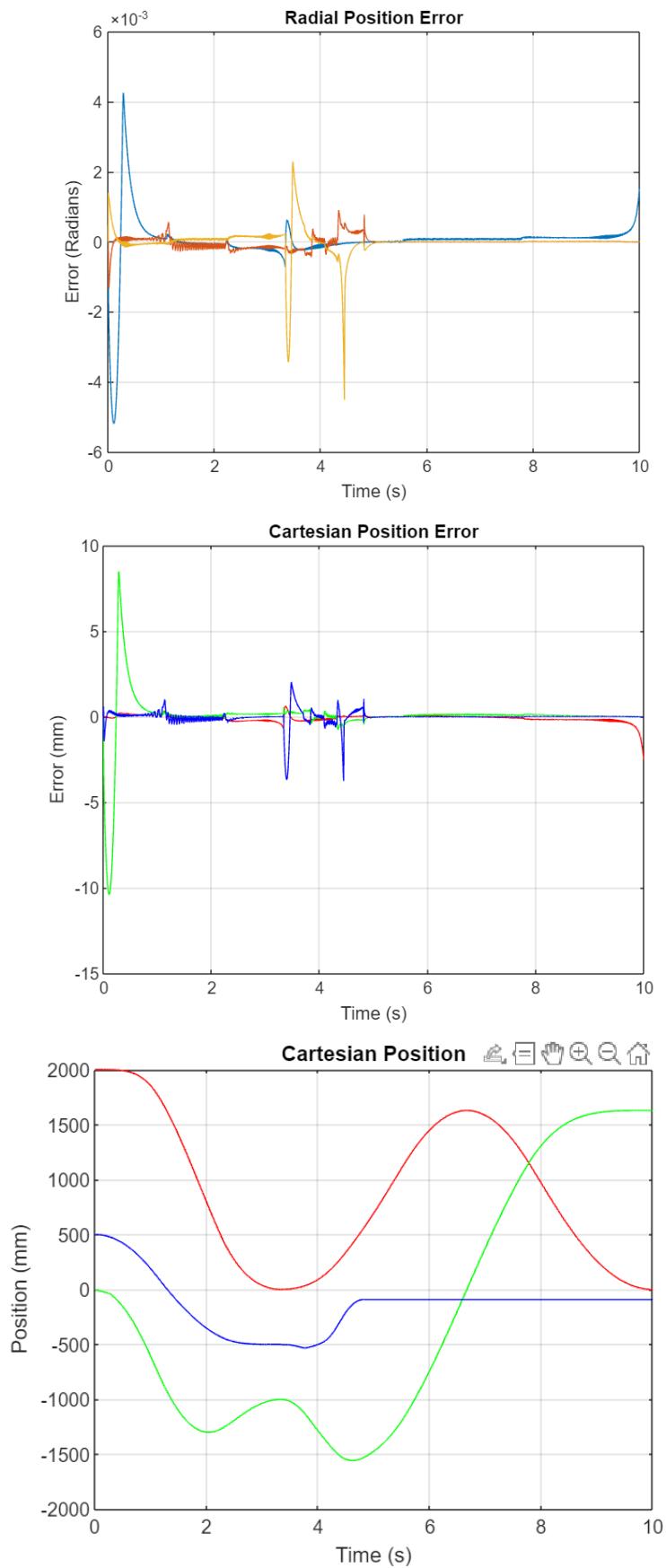
Anhang 20: Tracking-Fehler Trajektorie 1 mit Zufallsvariation



Anhang 21: Tracking-Fehler Trajektorie 2 mit Zufallsvariation

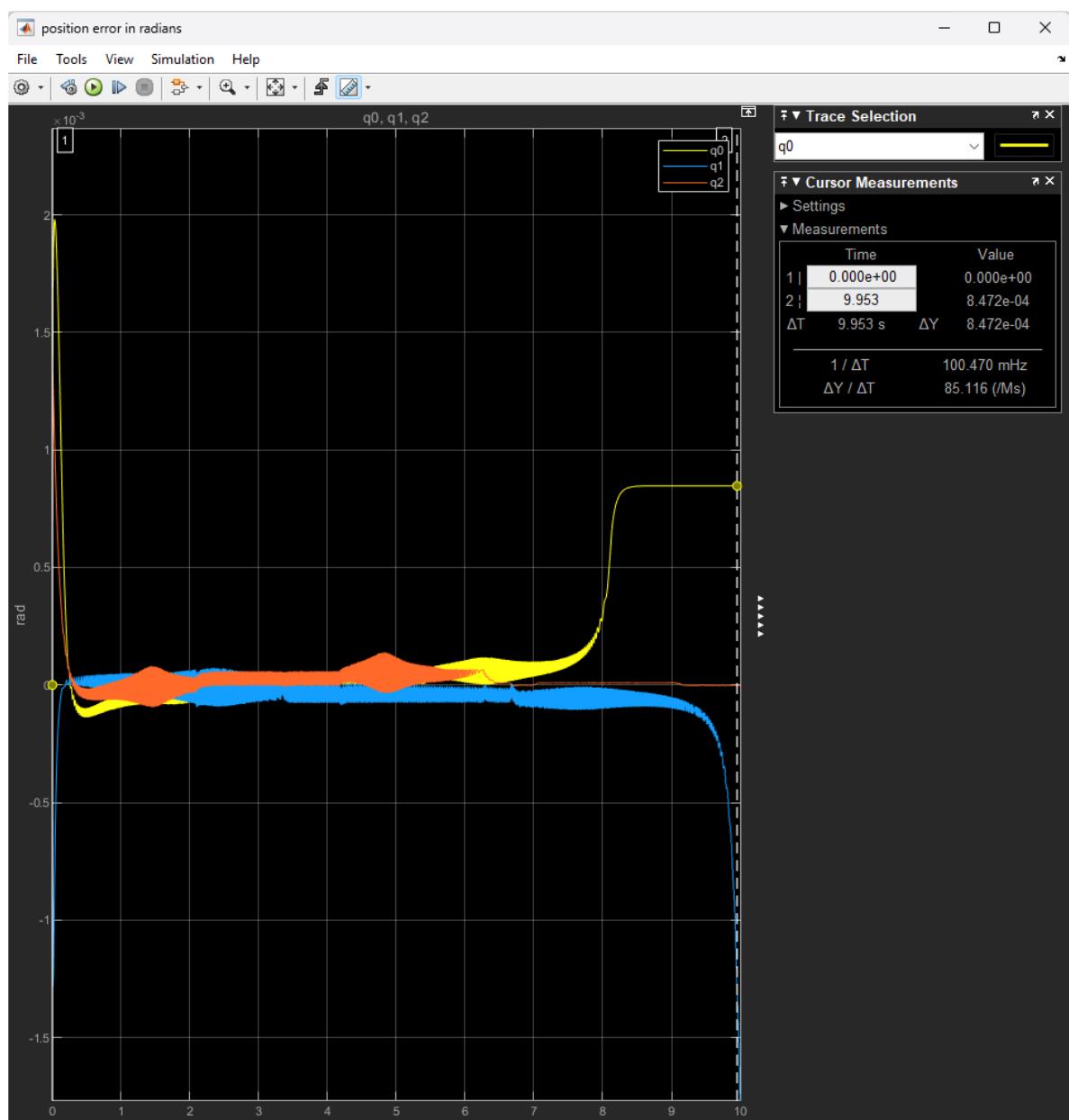
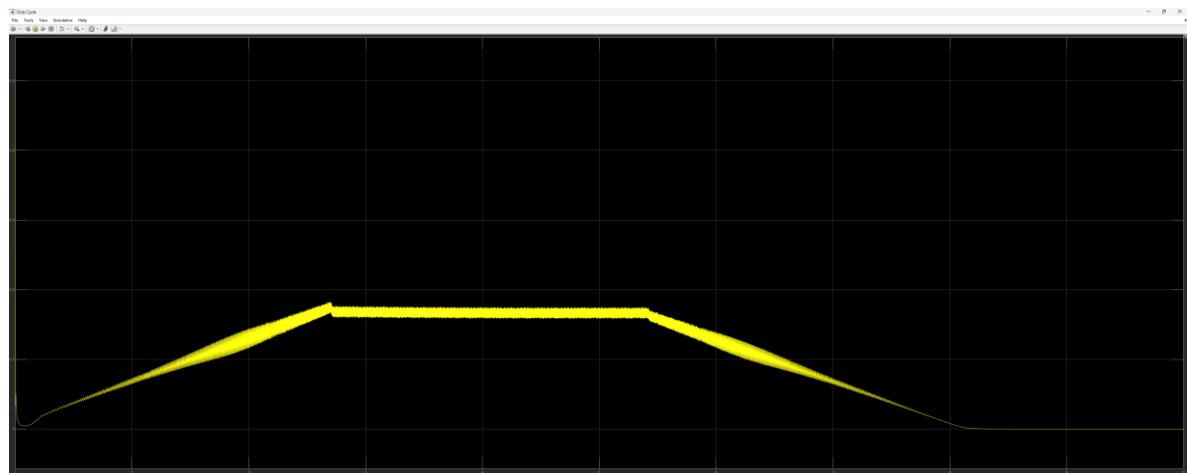


Anhang 22: Tracking-Fehler Trajektorie 3 mit Zufallsvariation

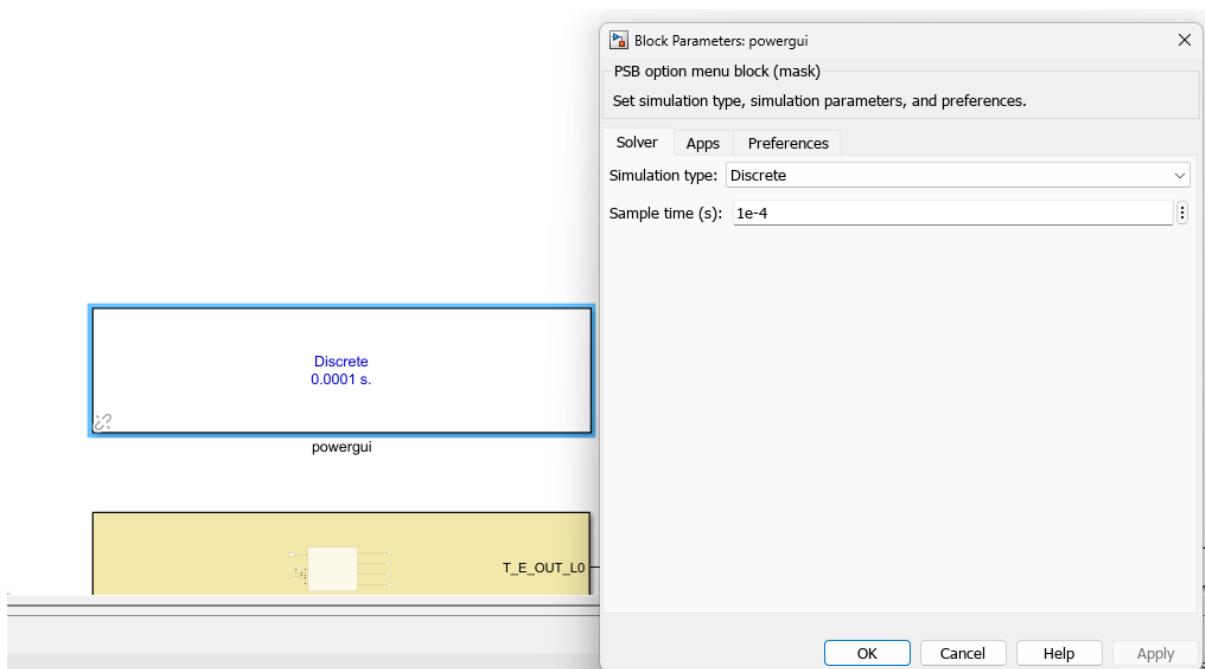
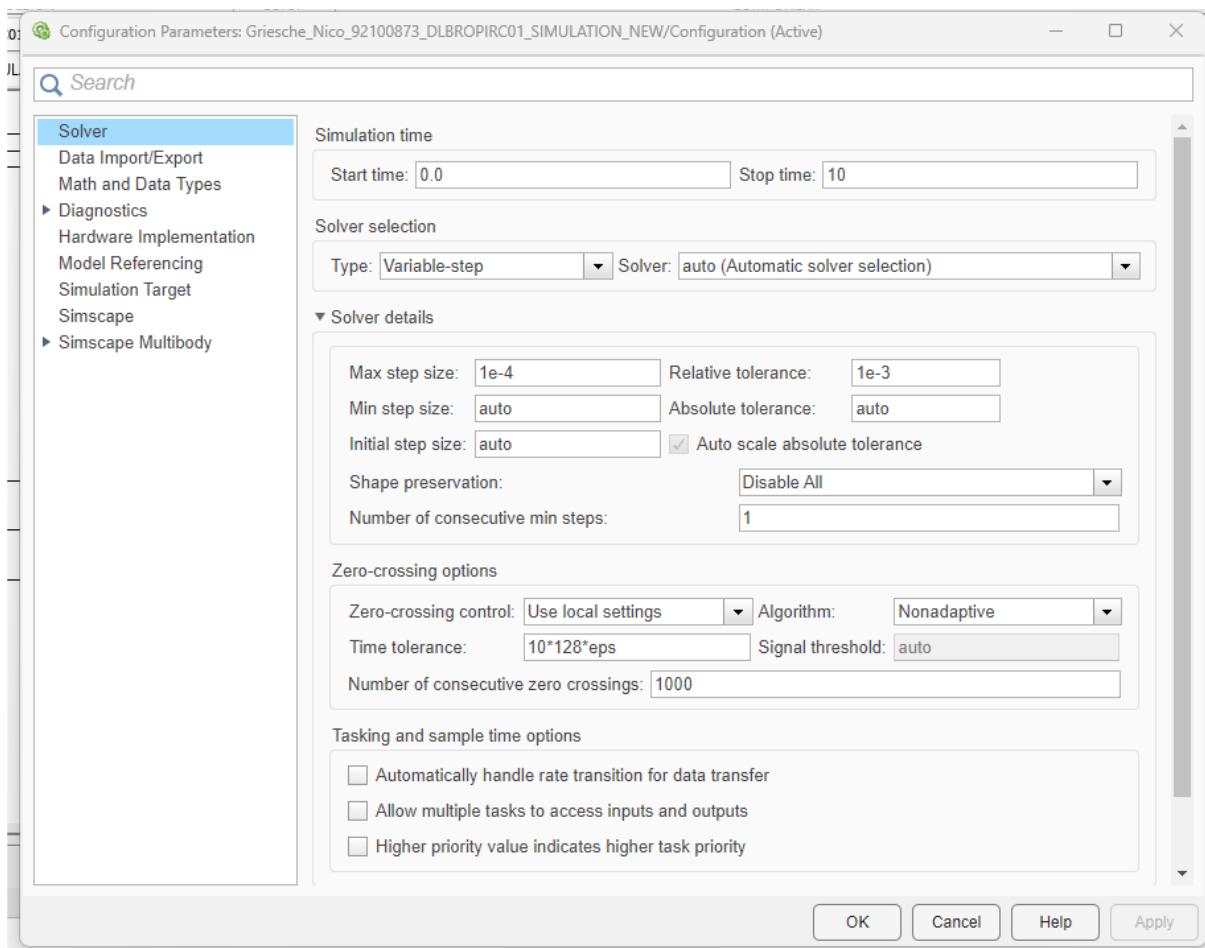


Anhang 23: Diskussion Tracking Fehler

PWM Signal q0:



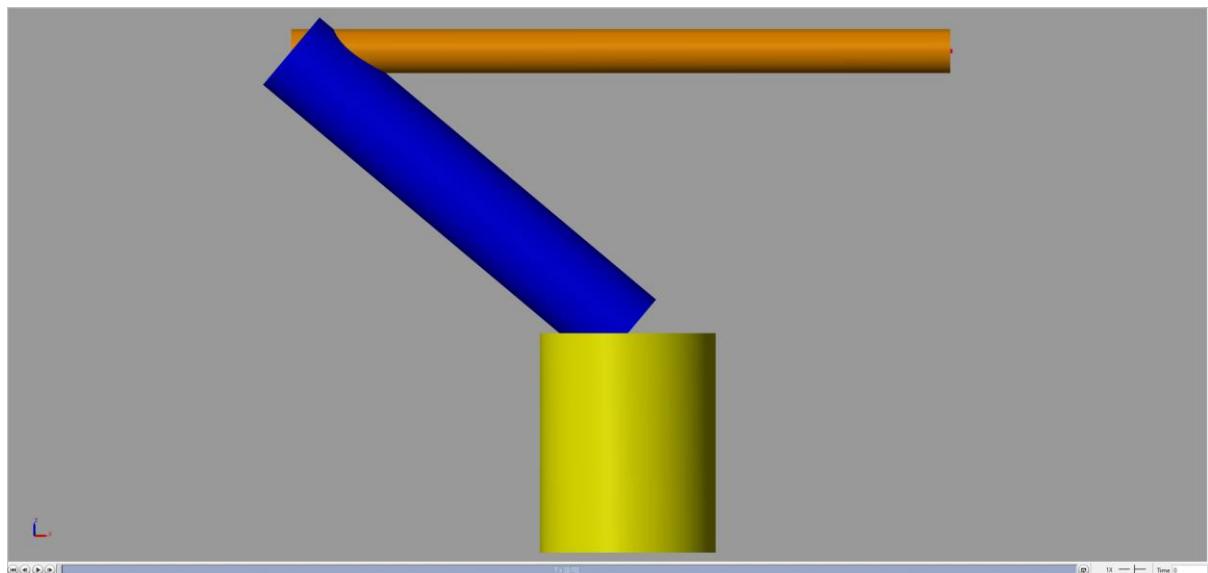
Anhang 24: Solver Einstellungen



Anhang 25: Manipulator Geometrie

	Link 0 (gelb)	Link 1 (blau)	Link 2 (orange)
Geometrischer Körper:	Zylinder	Zylinder	Zylinder
Länge [in Meter]:	0.5	1	1.5
Radius [in Meter]:	0.2	0.1	0.05
Gewicht [in Kg]	20	10	5

[In Degree]	Joint 0 (gelb)	Joint 1 (gelb zu blau)	Joint 2 (blau zu orange)
Drehung X-Z-X	-	[90 140 0]	-
Drehung Z-X-Y	-	-	[-140 0 0]
Unteres Limit	-180	-120	0
Oberes Limit	180	0	140



RPX52

ElectroCraft RapidPower™ Xtreme Brushless DC Servo Motor

High torque density. Excellent torque per frame size performance.



RPX52

RAPIDPOWER™ XTREME BRUSHLESS DC SERVO MOTOR

RPX52 Mechanical / Winding Data

Specifications	Stack Size and Winding Models							
	RPX52-300V12	RPX52-300V24	RPX52-300V48	RPX52-600V12	RPX52-600V24	RPX52-600V48	RPX52-750V24	RPX52-750V48
Design Voltage (VDC)	12	24	48	12	24	48	24	48
No load speed (RPM)		5,500			3,800			3,750
Peak Torque (oz-in)		99.2			233.7			446.2
Peak Torque (mNm)		700.0			1,650.0			3,150.0
Peak Current (Amps)	36.8	17.4	8.6	56.3	26.5	12.9	55.8	28.6
Continuous Stall Torque* (oz-in)		42.5			85.0			106.2
Continuous Stall Torque* (mNm)		300.0			600.0			750.0
Continuous Stall Current (Amps)	20.0	10.0	5.0	24.0	12.0	6.0	14.1	7.1
Continuous Rated Torque* (oz-in)		28.3			62.3			85.0
Continuous Rated Torque* (mNm)		200.0			440.0			600.0
Continuous Rated Current (Amps)	13.7	6.8	3.5	18.4	9.1	4.6	11.4	5.8
Continuous Rated Speed (RPM)		3,400			2,000			2,500
Voltage Constant (V / kRPM)	2.4	4.4	8.7	3.2	6.3	12.6	6.6	13.2
Torque Constant (oz-in / Amp)	2.7	5.7	11.6	4.2	8.8	18.1	8.9	17.9
Torque Constant (mNm / Amp)	19.0	40.2	81.7	29.3	62.3	127.8	62.7	126.1
Resistance (Ohms)	0.15	0.46	1.74	0.14	0.34	1.26	0.26	0.95
Inductance (mH)	0.09	0.37	1.40	0.10	0.33	1.28	0.25	0.98
Motor Constant (oz-in / Watt)	6.9	8.4	8.8	11.1	15.1	16.1	17.4	18.4
Motor Constant (mNm / Watt)	49.1	59.3	61.9	78.3	106.8	113.9	122.5	129.7
Electrical Constant (msec)	0.60	0.80	0.80	0.71	0.97	1.02	0.95	1.04
Mechanical Constant (msec)	2.41	1.91	1.79	1.61	0.93	0.84	0.93	0.84
Thermal Time Constant (sec)		1,200.0			1,240.0			6,000
Rotor Inertia (oz-in²)		0.3390			0.5631			0.7709
Rotor Inertia (g·cm²)		62.0			103.0			141.0
Thermal Resistance (C / Watt)		2.79			2.32			2.30
Axial Load (N) 5mm from face		35.0			35.0			35.0
Radial Load (N) 5mm from face		110.0			110.0			110.0
Weight (oz)		12.7			18.3			23.2
Weight (g)		360.00			520.00			657.00
Length (inch)		1.7			2.2			2.7
Length (mm)		44.0			56.0			68.0
Number of Poles		14.0			14.0			14.0

Notes: *Continuous rating based on a 25°C ambient temperature, winding temperature rise of 125°C. Mounted on a 152.4 X 152.4 X 6.4 mm aluminum heat sink.



Your Genius. Our Drive.

ElectroCraft, Inc.
2 Martin Way, Suite 3
Stratham, NH 03885-2578 USA

Tel: (844) 565-6144

Email: sales@electrocraft.com
www.electrocraft.com

PAGE 3 OF 3