

Bachelorthesis

Fakultät Optik & Mechatronik

Entwurf und Erprobung einer Regelung für ein pneumatisches Robotergelenk für sich wiederholende Bewegungen

Betreuer: Prof. Dr.-Ing. Jürgen Baur (Hochschule Aalen)
Prof. Dr.-Ing. Fabian Holzwarth (Hochschule Aalen)
Dr. Rainer Nitsche (Festo SE & Co. KG)

Timo Heubach
Mechatronik
79976
8. Semester

Waldstraße 50
71384 Weinstadt
+49 157 78985751
timo.heubach@gmx.de

Zeitraum: 01. März 2023 - 31. August 2023

Sperrvermerk

Die nachfolgende Bachelorarbeit beinhaltet vertrauliche und interne Daten der Firma Festo SE & Co. KG. Veröffentlichungen und Vervielfältigungen - auch nur auszugsweise - sind ohne ausdrückliche schriftliche Genehmigung des Unternehmens nicht gestattet. Die Arbeit ist nur den Gutachtern sowie den Mitgliedern des Prüfungsausschusses zugänglich zu machen.

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Esslingen den 24. August 2023

T. Heubach

(Ort, Datum, Unterschrift)

Gender-Hinweis

In dieser Arbeit wird aus Gründen der besseren Lesbarkeit das generische Maskulinum verwendet. Weibliche und anderweitige Geschlechteridentitäten werden dabei ausdrücklich mitgemeint, soweit es für die Aussage erforderlich ist.

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	V
Skriptverzeichnis	VI
1 Einleitung	1
1.1 Problemstellung	1
1.2 Ziel der Arbeit	1
1.3 Aufbau der Arbeit	2
2 Iterative Learning Control (ILC)	3
2.1 Einführung	3
2.2 Funktionsweise	4
2.3 Stabilität und Konvergenz	7
2.4 ILC-Gesetze	9
2.4.1 P-Type	10
2.4.2 PD-Type	10
2.4.3 Inversionsbasiertes ILC	11
2.4.4 Akausale Filterung	11
2.5 Matlab-Simulink	12
2.5.1 Untersuchung Stabilität und Konvergenz	13
2.5.2 Simulink-Modell ILC-Ansatz	14
2.5.3 Simulink-Modell ILC-Ansatz + PID-Regler	19
2.6 Ergebnisse	20
2.6.1 PT1-System	20
2.6.2 PT2-System	26
2.6.3 Zusammenfassung	32
3 Konstantes Volumen	33
3.1 Simulation	33
3.1.1 Modellierung Gesamtsystem	33
3.1.2 Flachheitsbasierter Regler	35
3.1.3 ILC-Ansatz	38
3.1.4 Ergebnisse	39

3.2 Reale Hardware	40
3.2.1 Aufbau	40
3.2.2 Echtzeitfähigkeit ILC-Ansatz	42
3.2.3 Ergebnisse	45
3.3 Zusammenfassung	46
4 Versuchsaufbau pJoint	47
4.1 Aufbau und Funktionsweise	47
4.2 Simulation	49
4.2.1 Aufbau Gesamtsystem	49
4.2.2 Reglerkaskade	50
4.2.3 ILC-Ansatz	51
4.2.4 Ergebnisse	52
4.3 Reale Hardware	54
4.3.1 Aufbau	54
4.3.2 Ergebnisse	56
4.4 Zusammenfassung	59
5 Fazit und Ausblick	60
A Literaturverzeichnis	61
B Messergebnisse pJoint	63

Abbildungsverzeichnis

2.1	Grundaufbau Regelkreis	4
2.2	Grundidee ILC [1]	5
2.3	Sollsignal	13
2.4	Matlab Modell ILC-Ansatz	15
2.5	ILC Subsystem Maske	16
2.6	Matlab Modell ILC-Ansatz mit PID-Regler in paralleler Struktur . .	19
2.7	Matlab Modell ILC-Ansatz mit PID-Regler in serieller Struktur . .	19
2.8	Stabilitäts- und Konvergenzuntersuchung PT1-System	21
2.9	Simulationsergebnis P-Typ ILC anhand PT1-System	22
2.10	Simulationsergebnis Inversionsbasiertes ILC anhand PT1-System . .	23
2.11	Simulationsergebnis ILC-Ansatz + PI-Regler in paralleler Struktur anhand PT1-System	24
2.12	Simulationsergebnis ILC-Ansatz + PI-Regler in serieller Struktur anhand PT1-System	25
2.13	Stabilitäts- und Konvergenzuntersuchung PT2-System	27
2.14	Simulationsergebnis P-Typ ILC anhand PT2-System	28
2.15	Simulationsergebnis Inversionsbasiertes ILC anhand PT2-System . .	29
2.16	ILC-Ansatz + flachheitsbasierte Vorsteuerung	30
2.17	flachheitsbasierte Vorsteuerung	30
2.18	Simulationsergebnis ILC-Ansatz + flachheitsbasierte Vorsteuerung anhand PT2-System	31
3.1	Blockschaltbild Druckregelung konstantes Volumen	33
3.2	Matlab Modell pneumatisches Volumen	34
3.3	Matlab Modell Piezoventile	34
3.4	Matlab Modell Gesamtsystem konstantes Volumen	35
3.5	Matlab Inverses Ventil	35
3.6	Schematischer Aufbau reduziertes konstantes Volumenmodell . . .	37
3.7	Matlab Modell reduziertes Gesamtsystem mit Druckregler	38
3.8	Matlab Modell reduziertes Gesamtsystem mit Druckregler und ILC- Ansatz	39
3.9	Simulationsergebnis Druckregelung konstantes Volumen	40
3.10	Versuchsaufbau konstantes Volumen	41
3.11	Pneumatikplan konstantes Volumen	42
3.12	Messergebnis Versuchsaufbau <i>konstantes Volumen</i>	46

4.1	Blockschaltbild Reglerkaskade pneumatisches Gelenk	47
4.2	Schematischer Aufbau eines pneumatischen Gelenks [12]	48
4.3	Funktionsweise Piezoventil [10]	49
4.4	Matlab Modell Gesamtsystem pneumatisches Gelenk	49
4.5	Matlab Modell pneumatisches Gelenk unterlagerter Druckregler . . .	50
4.6	Matlab Modell pneumatisches Gelenk überlagerter Positionsregler . .	51
4.7	Matlab Modell Kaskadenregelung pneumatisches Gelenk mit ILC-Ansatz	52
4.8	Simulationsergebnis Positionsregelung pneumatisches Gelenk	53
4.9	Versuchsaufbau pneumatisches Gelenk	54
4.10	Pneumatikplan pneumatisches Gelenk	55
4.11	Messergebnis Versuchsaufbau pneumatisches Gelenk, unterlagerter Druckregler (Abschnitt 4.2.2) mit überlagertem Positionsregler und ILC-Ansatz	57
4.12	Messergebnis Versuchsaufbau pneumatisches Gelenk, unterlagerter Druckregler (Festo Bibliothek) mit überlagertem Positionsregler und ILC-Ansatz	58
4.13	Messergebnis Versuchsaufbau pneumatisches Gelenk Regelabweichung nach 70 Iterationen	58
B.1	Messergebnis Versuchsaufbau pneumatisches Gelenk mit ROPA-DR- 10, unterlagerter Druckregler (Festo Bibliothek) mit überlagertem Positionsregler und ILC-Ansatz	63
B.2	Messergebnis Versuchsaufbau pneumatisches Gelenk mit ROPA-DR-5, unterlagerter Druckregler (Festo Bibliothek) mit überlagertem Positi- onsregler und ILC-Ansatz	64

Tabellenverzeichnis

2.1 Vergleich der Regelkonzepte anhand PT1-System	26
2.2 Vergleich der Regelkonzepte anhand PT2-System	32

Skriptverzeichnis

2.1	Berechnung maximaler Singulärwert	8
2.2	Untersuchung Stabilität und Konvergenz	13
2.3	ILC-Algorithmus	16
3.1	ILC-Algorithmus - Echtzeitfähig	43

1 Einleitung

Das erste Kapitel gibt eine erste Einführung in die Thematik der Arbeit und erläutert die Problemstellung. Anschließend wird die Zielsetzung und der Aufbau der Arbeit beschrieben.

1.1 Problemstellung

Festo entwickelt pneumatische Leichtbauroboter für den industriellen Einsatz, bei denen die Gelenke pneumatisch aktuiert werden. Pneumatische Gelenke bieten insbesondere in der kollaborativen Robotik den Vorteil, dass durch eine Mehrgrößenregelung neben der Position auch die Gelenksteifigkeit geregelt werden kann. Im Vergleich zu elektrisch betriebenen Robotern ist die Positioniergenauigkeit jedoch schlechter.

Für typische zyklische Aufgaben, wie z.B. Pick-and-Place Anwendungen, bietet sich die Verbesserung der Reglerperformance durch Iterative Learning Control (ILC) (dt. Iterativ Lernende Regelung (ILR)) an. Bei diesem Konzept wird der Fehler des vorangegangenen Zyklus analysiert und daraus gelernt, um die Regelgüte des nächsten Zykluses zu verbessern [1]. Durch die Anwendung der ILR kann der Roboter den Fehler aus vorangegangenen Zyklen korrigieren und somit eine höhere Genauigkeit und Reproduzierbarkeit erreichen.

1.2 Ziel der Arbeit

Das Ziel dieser Arbeit besteht darin, ein grundlegendes Verständnis des Regelkonzepts Iterative Learning Control (ILC) zu erlangen und das Konzept für einfache Systeme in Matlab-Simulink zu implementieren. Hierbei sollen insbesondere die Stabilitäts- und Konvergenzkriterien untersucht werden. Zusätzlich soll das ILC-Konzept parallel zu einem herkömmlichen PID-Regler eingebaut und simuliert werden.

Sobald das Regelkonzept erfolgreich für einfache Systeme implementiert wurde, soll es an komplexeren Systemen getestet werden. Das Hauptziel dieser Arbeit ist es, den ILC-Ansatz in Kombination mit einer Reglerkaskade für ein pneumatisches Gelenk zu implementieren und zu testen. Hierbei sollen sowohl Simulationen in Matlab-Simulink als auch reale Tests an der Hardware mithilfe eines Rapid Control

Prototyping Systems der Firma dSpace durchgeführt werden.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit ist in drei Hauptkapitel unterteilt. Das zweite Kapitel widmet sich zunächst der Einführung in das Regelkonzept Iterative Learning Control (ILC). Dabei werden die Funktionsweise, die Stabilitäts- und Konvergenzkriterien sowie die verschiedenen ILC-Gesetze erläutert. Des Weiteren erfolgt die Implementierung des Regelkonzepts anhand einfacher Systeme in Matlab-Simulink. Dabei werden die verschiedenen ILC-Gesetze getestet und miteinander verglichen. Das Kapitel schließt mit der Präsentation der Simulationsergebnisse.

Im dritten Kapitel wird der ILC-Ansatz erstmals an einem nichtlinearen System, sowohl in der Simulation als auch an realer Hardware getestet. Dafür wird ein vereinfachter Versuchsaufbau für eine Druckregelung eines konstanten Volumens verwendet. Dieser dient als Vorstufe zur Zielhardware. Dabei wird die Modellierung des Gesamtsystems und der Einsatz eines modellbasierten Reglers beschrieben. Zudem wird der Versuchsaufbau erläutert und vorgestellt. Das Kapitel endet mit der Darstellung der Ergebnisse an der realen Hardware und einer kurzen Zusammenfassung des Kapitels. Das letzte Hauptkapitel erläutert zunächst den Aufbau und die Funktionsweise eines pneumatischen Gelenks. Anschließend wird das Regelkonzept Iterative Learning Control, sowohl in Simulationen als auch an realer Hardware, an einem pneumatischen Gelenk getestet. Dabei wird das Regelkonzept in einer Reglerkaskade eingesetzt. Durch die Präsentation der Ergebnisse und einer kurzen Zusammenfassung des Kapitels ist der Hauptteil dieser Arbeit abgeschlossen.

Abschließend erfolgen ein Fazit und Ausblick der vorliegenden Bachelorarbeit.

2 Iterative Learning Control (ILC)

In diesem Kapitel wird das Regelungskonzept Iterative Learning Control (ILC) erläutert und seine Anwendung anhand einfacher linearer Systeme untersucht. Dabei werden zunächst die theoretischen Grundlagen von ILC dargelegt, einschließlich der Ableitung des Regelkonzepts und der Beschreibung der relevanten Begriffe und Parameter. Anschließend werden die Stabilitäts- und Konvergenzkriterien von ILC untersucht und diskutiert.

Um das Konzept von ILC in der Praxis zu testen, werden einfache lineare Systeme in Matlab-Simulink implementiert. Dabei werden die Ergebnisse der Simulationen verwendet, um die Stabilität und Konvergenz von ILC zu überprüfen. Zusätzlich wird das ILC-Konzept zusammen mit einem klassischen PID-Regler implementiert, simuliert und untersucht.

Das Ziel dieses Kapitels ist es, ein grundlegendes Verständnis von ILC zu vermitteln und seine Anwendung anhand einfacher linearer Systeme zu demonstrieren. Die gewonnenen Erkenntnisse und Erfahrungen werden in den nächsten Kapiteln auf nichtlineare Systeme angewendet, um die Wirksamkeit von ILC in der Regelung eines pneumatischen Gelenks zu untersuchen.

2.1 Einführung

Das Konzept von Iterative Learning Control basiert auf der Idee, dass wiederholte Durchläufe einer kontinuierlichen Aufgabe genutzt werden können, um die Leistung der Regelung schrittweise zu verbessern. Im Gegensatz zur klassischen Regelungstechnik, bei der das Regelergebnis nur auf Basis der aktuellen Messwerte berechnet wird, nutzt ILC Informationen aus vergangenen Durchläufen, um die Steuerung des Systems für die nächsten Durchläufe zu optimieren [1]. ILC ist deshalb speziell für Systeme entwickelt worden, welche wiederholende Aufgaben ausführen. Ein solches Konzept findet in vielen Bereichen Anwendungen. Beispielsweise bei der Regelung der Profiltemperatur beim Strangpressen, bei der Automatisierung eines verfahrenstechnischen Prozesses und vor allem bei der Automatisierung von Roboterbewegungen [2].

Das Ziel von ILC ist es, die Regelgüte solcher Systeme iterativ durch Lernen zu verbessern. Dafür wird der Fehler und das Stellsignal des vorangegangenen Zyklus

analysiert und für die Anpassung des neuen Stellsignals genutzt. Aufgrund der Tatsache, dass das neue Stellsignal vor jedem Zyklus bereits berechnet wird, ist es möglich, nicht kausale Algorithmen und Filter anzuwenden. Dadurch wird beispielsweise eine Tiefpassfilterung ohne Phasenverschiebung und eine Kompensation der Verzögerungszeit des Systems ermöglicht. Dies ist ein großer Vorteil gegenüber einer klassischen PID-Regelung [2]. Der ILC-Ansatz kann als eine Art Steuerungsanpassung verstanden werden, da das Stellsignal nur vom vorherigen Zyklus abhängt [1].

Ein weiterer Vorteil von ILC besteht darin, dass der Algorithmus in der Lage ist, mit Ungenauigkeiten in der Modellierung des Systems umzugehen. Da der Algorithmus iterativ arbeitet und sich auf die Fehlerkorrektur konzentriert, können Änderungen des Systemverhaltens über die Zeit, die in jedem Zyklus auftreten, korrigiert werden [3].

Der ILC-Ansatz gehört zur Klasse der Feedforward-Regelungen und kann im Sinne einer Zwei-Freiheitsgrad-Struktur durch einen Feedback-Anteil erweitert werden. Hierbei kann beispielsweise ein klassischer PID-Regler zum Einsatz kommen. Durch die Kombination von Feedforward- und Feedback-Regelung wird eine verbesserte Regelgüte und Störungsunterdrückung erreicht, da der ILC-Regler die periodischen Fehler reduziert und der Feedback-Regler die Störungen ausgleicht [1].

Um das Verständnis der nachfolgenden Abschnitte zu erleichtern, ist in Abbildung 2.1 der Aufbau eines Regelkreises mit den grundlegenden Größen dargestellt.

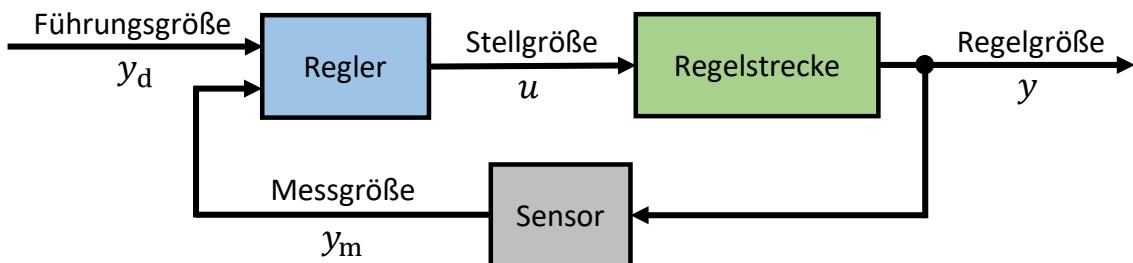


Abbildung 2.1: Grundaufbau Regelkreis

2.2 Funktionsweise

Iterativ Learning Control funktioniert wie folgt:

In jeder Iteration ($j = 0, 1, \dots$) wird eine Steuerung \mathbf{u}_{j+1} berechnet. Dafür muss der Ausgangsfehler \mathbf{e}_j und die Steuerung \mathbf{u}_j bestimmt und abgespeichert werden. Der

Ausgangsfehler

$$\mathbf{e} = \mathbf{y}_d - \mathbf{y}_m \quad (2.1)$$

wird dabei aus der Differenz zwischen dem Sollsignal \mathbf{y}_d (desired) und dem gemessenen Signal \mathbf{y}_m (measured) gebildet. In Abbildung 2.2 ist der Ablauf des ILC-Ansatzes abgebildet.

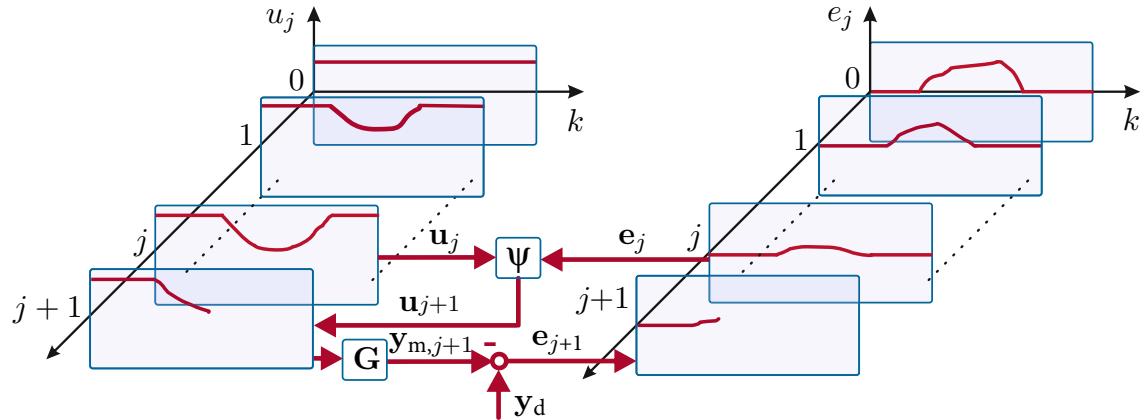


Abbildung 2.2: Grundidee ILC [1]

In jeder Iteration wird die Steuerung \mathbf{u}_{j+1} durch die Funktion $\psi(\mathbf{u}_j, \mathbf{e}_j(\mathbf{u}_j))$ berechnet. Die neue Steuerung ist somit nur von der vorherigen Iteration abhängig. Das daraus resultierende Stellsignal wird auf das System \mathbf{G} angewendet, um den Ausgang $\mathbf{y}_{m,j+1}$ zu messen. Anhand des Sollsignals \mathbf{y}_d wird der neue Ausgangsfehler \mathbf{e}_{j+1} ermittelt und zwischengespeichert. Gleichzeitig wird auch das an das System übergebene Stellsignal \mathbf{u}_{j+1} abgespeichert. Dieser Prozess kann beliebig oft wiederholt werden, wobei das Ziel darin besteht, die Steuerung \mathbf{u} so anzupassen, dass der Ausgangsfehler \mathbf{e} gegen null konvergiert.

Die Berechnung des Stellsignals anhand einer Minimierungsaufgabe wird in den Vorlesungsunterlagen der Technischen Universität (TU) Wien [1, S. 9] hergeleitet und im Folgenden als gegeben angenommen. Zum Verständnis der Gleichung wird eine Verstärkungsmatrix $\mathbf{L} \in R^{N \times N}$ und eine Filtermatrix $\mathbf{Q} \in R^{N \times N}$ definiert, wobei N die Anzahl der Abtastschritte während einer Iteration darstellt.

Das ILC-Gesetz hängt bei seiner Herleitung vom relativen Grad r des Systems ab, welcher bei linearen Systemen auch als Polüberschuss bekannt ist. Für lineare zeitkontinuierliche Systeme kann der relative Grad mit der Gleichung

$$r = n - m \quad (2.2)$$

berechnet werden, wobei der höchste Exponent des Zählerpolynoms der Übertra-

gungsfunktion durch m und der höchste Exponent des Nennerpolynoms durch n gegeben ist. Für zeitdiskrete Systeme gilt immer $r = 1$. Der relative Grad eines diskreten Systems gibt den Zeitindex r an, ab dem eine direkte Wirkung des Eingangs auf den Ausgang erfolgt. Er kann daher auch als Verzögerung interpretiert werden. In praktischen Anwendungen kann es neben der Verzögerung durch den relativen Grad r des Systems auch zu Verzögerungen aufgrund von Digital-Analog- oder Analog-Digital-Wandlungen kommen. Diese Verzögerung ist durch die Anzahl der Abtastschritte w gegeben, welche für die Wandlung benötigt werden. Damit kann eine Gesamtverzögerung

$$v = r + w \quad (2.3)$$

eingeführt und im ILC-Gesetz berücksichtigt werden.

Mit der Definition der Vektoren:

$$\mathbf{u}_j = [u_j[0] \ u_j[1] \dots u_j[N - 1]]^T \in R^{N \times 1} \quad (2.4)$$

$$\mathbf{y}_{d,j} = [y_{d,j}[v] \ y_{d,j}[v + 1] \dots y_{d,j}[v + N - 1]]^T \in R^{N \times 1} \quad (2.5)$$

$$\mathbf{y}_{m,j} = [y_{m,j}[v] \ y_{m,j}[v + 1] \dots y_{m,j}[v + N - 1]]^T \in R^{N \times 1} \quad (2.6)$$

$$\mathbf{e}_j = \mathbf{y}_{d,j} - \mathbf{y}_{m,j} = [e_j[v] \ e_j[v + 1] \dots e_j[v + N - 1]]^T \in R^{N \times 1} \quad (2.7)$$

ergibt sich das ILC-Gesetz in der Lifted-System Darstellung zu:

$$\mathbf{u}_{j+1} = \mathbf{Q} (\mathbf{u}_j + \mathbf{L} \mathbf{e}_j) . \quad (2.8)$$

Die Verstärkungsmatrix \mathbf{L} hat die Aufgabe, den Einfluss des Ausgangsfehlers \mathbf{e}_j in Relation zur vorherigen Steuerung \mathbf{u}_j zu bestimmen. Durch die Wahl der Matrixelemente kann also gesteuert werden, wie stark der Einfluss des Ausgangsfehlers auf die neue Steuerung ist. Die Filtermatrix \mathbf{Q} dient hingegen dazu, Messrauschen und sich wiederholende Störungen zu unterdrücken. Dabei wirkt sie wie ein Tiefpassfilter, welcher nur niedrige Frequenzen passieren lässt und hohe Frequenzen herausfiltert. Das ILC-Gesetz Gleichung (2.8) kann auch in der diskreten Form angegeben werden:

$$u_{j+1}[k] = q[k] (u_j[k] + l[k] e_j[k + v]) . \quad (2.9)$$

Dafür wird der Abtastindex $k = 0, 1, \dots, N - 1$ definiert.

2.3 Stabilität und Konvergenz

In diesem Abschnitt werden die Stabilitäts- und Konvergenzkriterien aufgezeigt. Die Gleichungen sind den Vorlesungsunterlagen der TU Wien entnommen [1].

Um die Stabilitäts- und Konvergenzkriterien zu überprüfen, wird die Impulsantwort \mathbf{g} des Systems benötigt. Aus der Impulsantwort kann die Matrix

$$\mathbf{G} = \begin{pmatrix} \mathbf{g}[v] & 0 & \cdots & 0 \\ \mathbf{g}[v+1] & \mathbf{g}[v] & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \mathbf{g}[v+N-1] & \mathbf{g}[v+N-2] & \cdots & \mathbf{g}[v] \end{pmatrix} \in R^{N \times N} \quad (2.10)$$

gebildet werden. Es ist wichtig zu beachten, dass $\mathbf{g}[v] \neq 0$ gilt. Für die Berechnungen wird zusätzlich die Einheitsmatrix $\mathbf{E} \in R^{N \times N}$ herangezogen.

Die Stabilität des ILC-Gesetzes Gleichung (2.8) kann über Satz 1 und Satz 2 nachgewiesen werden. Dafür wird als Hilfe der Spektralradius ρ verwendet. Dieser ist definiert als der betragsmäßig größte Eigenwert einer quadratischen Matrix und kann wie folgt berechnet werden [4]:

$$\rho(\mathbf{A}) = \max |\lambda_i(\mathbf{A})| \quad (2.11)$$

Satz 1 (Stabilität der Eingangsfehleriteration) *Die Eingangsfehleriteration des ILC-Gesetzes, angewendet auf die Lifted-System Darstellung (2.8) ist asymptotisch stabil, falls:*

$$\rho(\mathbf{Q}(\mathbf{E} - \mathbf{L}\mathbf{G})) < 1 \quad (2.12)$$

Satz 2 (Stabilität der Ausgangsfehleriteration) *Die Ausgangsfehleriteration des ILC-Gesetzes, angewendet auf die Lifted-System Darstellung (2.8) ist asymptotisch stabil, falls:*

$$\rho(\mathbf{G}\mathbf{Q}(\mathbf{E} - \mathbf{L}\mathbf{G})\mathbf{G}^{-1}) < 1 \quad (2.13)$$

Eine Gegenüberstellung der Eingangs- und der Ausgangsfehleriteration ergibt, dass die Matrizen aus Satz 1 und Satz 2 dieselben Eigenwerte aufweisen. Dies bedeutet, dass die asymptotische Stabilität der Eingangsfehleriteration automatisch die asymptotische Stabilität der Ausgangsfehleriteration nachweist. Deshalb muss nur die Stabilität für eine Fehleriteration nachgewiesen werden.

Die Konvergenz ist über Satz 3 und Satz 4 nachweisbar. Dafür wird der maximale Singulärwert $\bar{\sigma}$ einer Matrix \mathbf{A} benötigt. Der maximale Singulärwert dieser Matrix entspricht dabei ihrer Spektralnorm und kann daher mit

$$\bar{\sigma} = \sqrt{\max |\lambda_i (\mathbf{A}^T \mathbf{A})|} \quad (2.14)$$

berechnet werden [5].

Zur Berechnung des maximalen Singulärwerts einer Matrix in Matlab kann die Funktion `svd(A)` verwendet werden. Die Funktion führt eine Singulärwertzerlegung durch und gibt alle Singulärwerte zurück. Um nur den gewünschten maximalen Singulärwert zu erhalten, kann die Funktion `max` benutzt werden [6]. Dadurch wird nur der größte Singulärwert zurückgegeben, siehe Skript 2.1:

```
sigma = max(svd(A));
```

Skript 2.1: Berechnung maximaler Singulärwert

Satz 3 (Monotone Konvergenz der Eingangsfehleriteration) *Die Eingangsfehleriteration der Lifted-System Darstellung (2.8) ist monoton konvergent gegen \mathbf{u}_∞ , wenn gilt:*

$$\|\mathbf{u}_{j+1} - \mathbf{u}_\infty\|_2 \leq \alpha \|\mathbf{u}_j - \mathbf{u}_\infty\|_2$$

für $0 \leq \alpha < 1$

$$\bar{\sigma}(\mathbf{Q}(\mathbf{E} - \mathbf{L}\mathbf{G})) < 1 \quad (2.15)$$

Satz 4 (Monotone Konvergenz der Ausgangsfehleriteration) *Die Ausgangsfehleriteration der Lifted-System Darstellung (2.8) ist konvergent gegen \mathbf{e}_∞ , wenn gilt:*

$$\|\mathbf{e}_{j+1} - \mathbf{e}_\infty\|_2 \leq \beta \|\mathbf{e}_j - \mathbf{e}_\infty\|_2$$

für $0 \leq \beta < 1$

$$\bar{\sigma}(\mathbf{G}\mathbf{Q}(\mathbf{E} - \mathbf{L}\mathbf{G})\mathbf{G}^{-1}) < 1 \quad (2.16)$$

Für den Beweis der monotonen Konvergenz in Satz 3 und Satz 4, werden die gleichen Matrizen wie in Satz 1 und Satz 2 verwendet. Daraus folgt, dass die Matrizen die gleichen Singulärwerte besitzen und es ausreicht, monotone Konvergenz für eine Fehleriteration nachzuweisen.

Wenn die Stabilitäts- und Konvergenzkriterien erfüllt sind, kann der asymptotische

Ausgangsfehler laut de Castro und Törres [7] mit

$$\mathbf{e}_\infty = (\mathbf{E} - \mathbf{G}(\mathbf{E} - \mathbf{Q}(\mathbf{E} - \mathbf{L}\mathbf{G}))^{-1}\mathbf{Q}\mathbf{L})(\mathbf{y}_d - \mathbf{y}_{m,0}) \quad (2.17)$$

berechnet werden. Dafür wird der Anfangszustand $\mathbf{y}_{m,0}$ des Istsignals benötigt. Der Ausgangsfehler kann nur zu null werden, wenn keine Filterung durchgeführt wird, das heißt

$$\mathbf{Q} = \mathbf{E} \quad \rightarrow \quad \mathbf{e}_\infty = 0. \quad (2.18)$$

Die Verwendung eines Filters im ILC-Gesetz führt dazu, dass der Regelfehler nicht auf null konvergieren kann. Dies liegt daran, dass der Filter den Ausgang des Systems glättet und somit die hochfrequenten Anteile des Regelfehlers reduziert. Dadurch wird auch die Fähigkeit der Regelung beeinträchtigt auf hochfrequente Änderungen zu reagieren. In praktischen Anwendungen ist es allerdings häufig notwendig, einen Filter je nach Anwendung zu verwenden, um Rauschen und Störungen zu unterdrücken. In diesem Fall ist es wichtig, eine geeignete Filtermatrix zu wählen, welche die unerwünschten Frequenzen herausfiltert, während gleichzeitig eine ausreichende Regelgenauigkeit erreicht werden kann.

2.4 ILC-Gesetze

In diesem Abschnitt geht es um die verschiedenen Arten von ILC-Gesetzen, die in der Praxis eingesetzt werden können. Es werden drei Arten von ILC-Gesetzen beschrieben.

Die P-Type und PD-Type Iterative Learning Control verwenden bekannte Konzepte der PID-Ausgangsregelung. Dabei wird der Ausgangsfehler \mathbf{e}_j , je nach Typ der Regelung, proportional und/oder differentiell zurückgeführt. Es wird jedoch in der Regel kein Integralanteil verwendet, da das ILC-Gesetz, durch die Aufsummierung des Stellsignals, bereits ein integrierendes Verhalten aufweist und somit die Vorteile des I-Anteils besitzt. Eine stationäre Genauigkeit ist somit bereits gegeben.

Das dritte Konzept ist das sogenannte inversionsbasierte Konzept, welches auf Basis des invertierenden Modells der Regelstrecke aufgebaut ist.

Außerdem wird der Entwurf einer akausalen Filterung erläutert.

2.4.1 P-Type

Wie auch bei einem klassischen Feedbackregler, ist das einfachste ILC Gesetz ein P-Type. Dabei wird die Verstärkungsmatrix \mathbf{L} zur einer diagonalen Verstärkungsmatrix mit dem Verstärkungsfaktor $k_p > 0$ gewählt.

$$\mathbf{L} = k_p \mathbf{E} \quad (2.19)$$

Daraus ergibt sich aus Gleichung (2.8) das P-Type ILC-Gesetz in der Lifted-System Schreibweise zu

$$\mathbf{u}_{j+1} = \mathbf{Q} (\mathbf{u}_j + k_p \mathbf{E} \mathbf{e}_j). \quad (2.20)$$

und analog dazu die diskrete Darstellung

$$u_{j+1}[k] = q[k] (u_j[k] + k_p e_j[k + v]). \quad (2.21)$$

2.4.2 PD-Type

Die PD-Type Iterative Learning Control verwendet sowohl den Ausgangsfehler als auch eine numerische Approximation der Zeitableitung des Ausgangsfehlers. Es ist daher eine Kombination aus Proportional- und Differentialglied. Um den Differentialanteil einzustellen, wird die Verstärkungsmatrix \mathbf{L} um einen zusätzlichen Einstellparameter $k_d > 0$ und einer Ableitungsfiltermatrix \mathbf{D} erweitert. Daraus ergibt sich

$$\mathbf{L} = k_p \mathbf{E} + k_d \mathbf{D}. \quad (2.22)$$

Je nach gewähltem zeitdiskreten Ansatz für die Gestaltung der Matrix \mathbf{D} kann diese unterschiedlich strukturiert sein. Im Folgenden ist beispielhaft ein möglicher Aufbau der Ableitungsfiltermatrix dargestellt:

$$\mathbf{D} = \frac{1}{T_s} \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix} \in R^{N \times N}. \quad (2.23)$$

Die Schrittweite ist dabei durch die Variable T_s gegeben.

Aus Gleichung (2.8) ergibt sich das PD-Type ILC-Gesetz in der Lifted-System Darstellung zu

$$\mathbf{u}_{j+1} = \mathbf{Q} (\mathbf{u}_j + (k_p \mathbf{E} + k_d \mathbf{D}) \mathbf{e}_j) \quad (2.24)$$

und analog dazu die diskrete Schreibweise

$$u_{j+1}[k] = q[k] \left(u_j[k] + k_p e_j[k+v] + \frac{k_d}{2T_s} (e_j[k+v+1] - e_j[k+v-1]) \right). \quad (2.25)$$

2.4.3 Inversionsbasiertes ILC

Inversionsbasiertes Iterative Learning Control ist eine Erweiterung des klassischen ILC-Konzepts, bei dem das invertierte Modell der Regelstrecke verwendet wird. Dafür muss die Impulsantwort \mathbf{g} der Regelstrecke und damit auch die Matrix \mathbf{G} bekannt sein. Für das ILC-Gesetz bedeutet dies, dass die Verstärkungsmatrix \mathbf{L} zur inversen Matrix von \mathbf{G} gewählt wird:

$$\mathbf{L} = \mathbf{G}^{-1}. \quad (2.26)$$

Dadurch ergibt sich aus Gleichung (2.8) das inversionsbasierte ILC-Gesetz in der Lifted System Schreibweise zu

$$\mathbf{u}_{j+1} = \mathbf{Q} (\mathbf{u}_j + \mathbf{G}^{-1} \mathbf{e}_j). \quad (2.27)$$

Mit diesem Verfahren ist es möglich, den Ausgangsfehler \mathbf{e}_j in einer Iteration auf den Grenzwert \mathbf{e}_∞ zu bringen. Wenn die Filtermatrix zu $\mathbf{Q} = \mathbf{E}$ gewählt wird, gilt zudem $\mathbf{e}_\infty = 0$. Allerdings ist dies in der Praxis oft schwer realisierbar, da in realen Anwendungen \mathbf{G} nie exakt bekannt und der Einsatz eines Filters essentiell ist.

2.4.4 Akausale Filterung

Der Einsatz einer Filterung hat den Zweck, das Lernen von nicht wiederkehrenden Störungen und Messrauschen zu unterbinden. Dies erweist sich insbesondere als relevant, wenn der ILC-Ansatz auf realer Hardware eingesetzt wird.

Wie bereits erwähnt, können für die Filterung auch akausale Filter verwendet werden. Ein einfacher Ansatz für einen akausalen Tiefpassfilter besteht in der Anwendung

eines Gauß-Filters [1]. Dafür wird die diskrete Gauß-Verteilung

$$\mathbf{q}_G[k] = \frac{T_s}{\sigma_q \sqrt{2\pi}} \exp\left(-\frac{(kT_s)^2}{2\sigma_q^2}\right) \quad (2.28)$$

mit der Standartabweichung

$$\sigma_q = \frac{\sqrt{\ln(2)}}{2\pi f_c} \quad (2.29)$$

verwendet. Hierbei steht der Parameter f_c für die Grenzfrequenz und bestimmt die Bandbreite bis zur -3dB-Absenkung.

Um eine Fensterung mit einer Länge von $2N_q + 1$ zu erhalten, ergibt sich der Vektor für den Gaußfilter zu

$$\mathbf{q}[k] = \frac{\mathbf{q}_G[k]}{\sum_{m=-N_q}^{N_q} \mathbf{q}_G[m]} \in R^N. \quad (2.30)$$

Mithilfe des Vektors \vec{q} kann die Filtermatrix \mathbf{Q} konstruiert werden. Hierfür werden die Parameter N und N_q verwendet. Im Folgenden ist ein Beispiel für die Erstellung einer Filtermatrix mit $N = 5$ und $N_q = 2$ gegeben:

$$\mathbf{Q} = \begin{pmatrix} \mathbf{q}[0] & \mathbf{q}[-1] & \mathbf{q}[-2] & 0 & 0 \\ \mathbf{q}[1] & \mathbf{q}[0] & \mathbf{q}[-1] & \mathbf{q}[-2] & 0 \\ \mathbf{q}[2] & \mathbf{q}[1] & \mathbf{q}[0] & \mathbf{q}[-1] & \mathbf{q}[-2] \\ 0 & \mathbf{q}[2] & \mathbf{q}[1] & \mathbf{q}[0] & \mathbf{q}[-1] \\ 0 & 0 & \mathbf{q}[2] & \mathbf{q}[1] & \mathbf{q}[0] \end{pmatrix} \in R^{5 \times 5}. \quad (2.31)$$

2.5 Matlab-Simulink

Dieses Kapitel stellt die Matlap Skripte zur Untersuchung der Stabilität- und Konvergenzkriterien sowie das Simulink Modell mit dem ILC-Ansatz vor. Für die vorliegende Abschlussarbeit wurde die Matlab Version 2021a verwendet.

Für alle weiteren Untersuchungen wird das in Abbildung 2.3 dargestellte Sollsignal y_d eingesetzt. Das Sollsignal besteht aus einer Sinusschwingung, einer sprungförmigen Ansteigung und einer rampenförmigen Ansteuerung. Dadurch kann das Systemverhalten für unterschiedliche Signalarten überprüft werden. Die Kanten sind durch die Faltung mit einem Gausfilter etwas abgeflacht. In der Praxis wird dies durch die Bahnplanung vorgenommen. Die Periodendauer des Sollsignals beträgt 42 Sekunden.

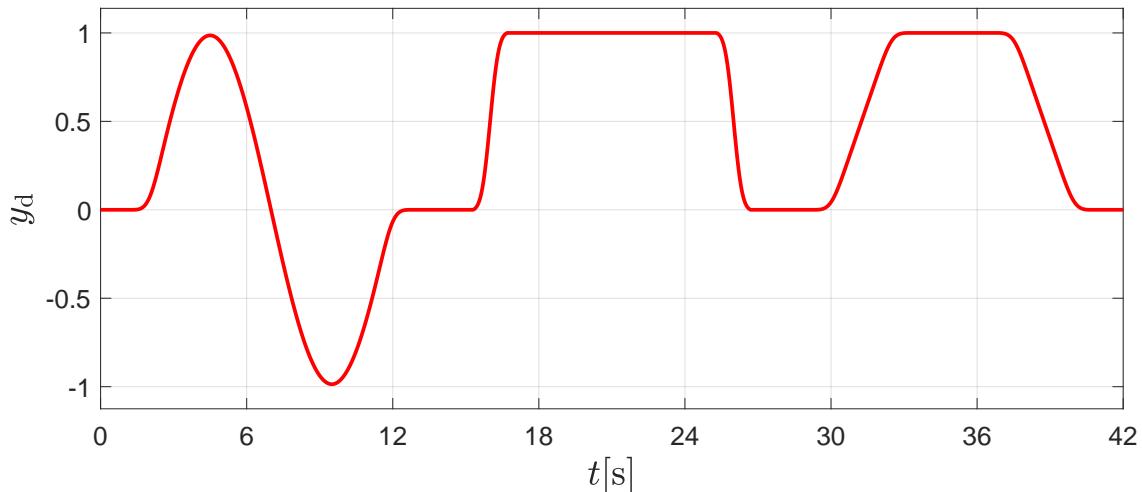


Abbildung 2.3: Sollsignal

2.5.1 Untersuchung Stabilität und Konvergenz

Zur Untersuchung der Stabilitäts- und Konvergenzkriterien wurde ein Skript erstellt, welches aus einer gegebenen Übertragungsfunktion die Impulsantwort berechnet und für die unterschiedlichen ILC-Gesetze die zulässigen Parameter k_p und gegebenenfalls k_d bestimmt. Anschließend werden die Ergebnisse in einem Schaubild aufgezeigt. In Skript 2.2 ist ein Ausschnitt aus dem Matlab Skript zur Parameterbestimmung abgebildet.

```

1 while(finish ~= 1)
2     switch state
3         case 1 % kp
4             disp('kp ermitteln... ')
5             for kp = kp_test
6                 L = kp * E;
7                 A = Q*(E-L*G);
8                 rho(n) = max(abs(eig(A)));
9                 sigma(n) = max(svd(A));
10                if rho(n) < 1 && sigma(n) < 1
11                    kp_sol(end+1) = kp;
12                    e_inf_seq(:,n) = (E - G/(E-A)*Q*L) *
13                        reshape(y_d,numel(y_d),1);
14                    e_inf_max(end+1) = max(abs(e_inf_seq(:,n)));
15                    e_inf_rms(end+1) = rms(e_inf_seq(:,n));
16                end
17                n = n + 1;

```

```

17      end
18      if isempty(kp_sol)
19          n = 1;
20          state = 2;
21          disp("Keine passenden Werte für kp gefunden")
22      else
23          finish = 1;
24          disp("Passende Werte für kp gefunden")
25      end

```

Skript 2.2: Untersuchung Stabilität und Konvergenz

Zu Beginn wird versucht, die Stabilitäts- und Konvergenzkriterien eines P-Type ILC zu erfüllen. Dafür sind die Einheitsmatrix **E** und die Matrix **G** bereits definiert und berechnet. Die Matrix **G** berechnet sich aus einer gegebenen Übertragungsfunktion. Durch den Vektor **kp_test** wird eine Wertespanne festgelegt, für die der Parameter k_p überprüft werden soll. In einer Vorschleife wird die Verstärkungsmatrix **L** für jeden gewünschten k_p -Wert berechnet. In den Zeilen acht und neun werden die Stabilitäts- und Konvergenzkriterien gemäß den Gleichungen (2.12) und (2.15) bestimmt. Die berechneten Werte werden in Vektoren gespeichert, um sie später in einem Schaubild darzustellen. In Zeile zehn werden die Stabilitäts- und Konvergenzkriterien gemäß Satz 1 und Satz 3 überprüft. Wenn beide Kriterien erfüllt sind, wird der k_p -Wert in einer Liste abgespeichert. Außerdem wird die asymptotische Regelabweichung mit Hilfe von Gleichung (2.17) bestimmt und der maximale Wert und der Effektivwert des Vektors gebildet. Wenn kein passender k_p -Wert gefunden werden kann, wird im nächsten Schritt ein sehr ähnlicher Ablauf für ein PD-Type ILC durchgeführt.

2.5.2 Simulink-Modell ILC-Ansatz

In diesem Abschnitt wird das Matlab-Simulink Modell mit dem ILC-Ansatz betrachtet, welches in Abbildung 2.4 dargestellt ist. Das Modell besteht aus zwei Subsystemen. Im grünen Subsystem wird das Sollsignal gemäß Abbildung 2.3 mit einer Abtastrate T_s am Ausgang ausgegeben. Das orangene Subsystem ist ein sogenanntes Resettable Subsystem, welches über den oberen Eingang ein- oder ausgeschaltet werden kann. Dieses Subsystem enthält den ILC-Ansatz.

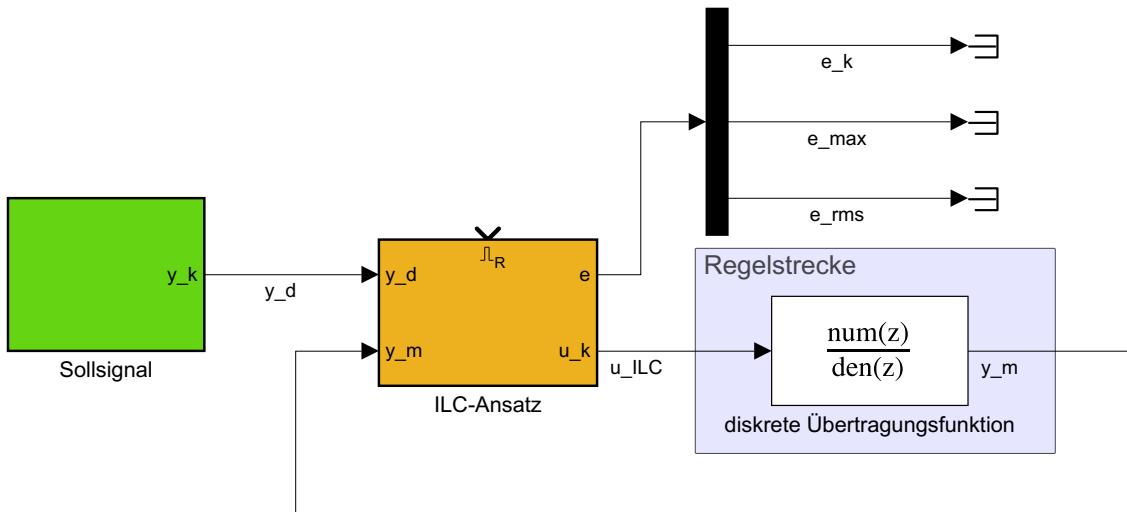


Abbildung 2.4: Matlab Modell ILC-Ansatz

Als Eingänge erhält der Block das Sollsignal y_d und das gemessene Signal y_m . Als Ausgänge werden das Stellsignal u_k , welches direkt auf die Regelstrecke wirkt, und der Regelfehler e bereitgestellt. Der Regelfehler e ist ein Vektor, der die aktuelle Regelabweichung e_k , die maximale Regelabweichung e_{\max} und den Effektivwert *Root Mean Square* e_{rms} der letzten Iteration enthält.

Der ILC-Ansatz arbeitet im diskreten Zeitbereich, deshalb wird die Regelstrecke als diskrete Übertragungsfunktion angegeben. Dafür muss die kontinuierliche Übertragungsfunktion $G_s(s)$ in eine diskrete Übertragungsfunktion $G_s(z)$ transformiert werden.

Maske

Die Maske ist eine grafische Benutzeroberfläche, die es ermöglicht, die Einstellungen und Parameter eines Simulink-Blocks zu konfigurieren. In diesem Fall wird die Maske verwendet, um die benötigten Parameter an den ILC-Block zu übergeben (siehe Abbildung 2.5). Der Parameter *Modus* ermöglicht die Auswahl zwischen verschiedenen ILC-Typen, während der Parameter *Filterung* entscheidet, ob eine Filterung des Signals vorgenommen werden soll oder nicht. Im vorherigen Kapitel wurde der D-Typ ILC-Ansatz nicht detailliert behandelt, jedoch nun aus Testzwecken aufgenommen. Für den inversionsbasierten ILC-Ansatz wird die Impulsantwort \mathbf{g} der Übertragungsfunktion, welche zuvor berechnet werden muss, im Hintergrund in den ILC-Block geladen.

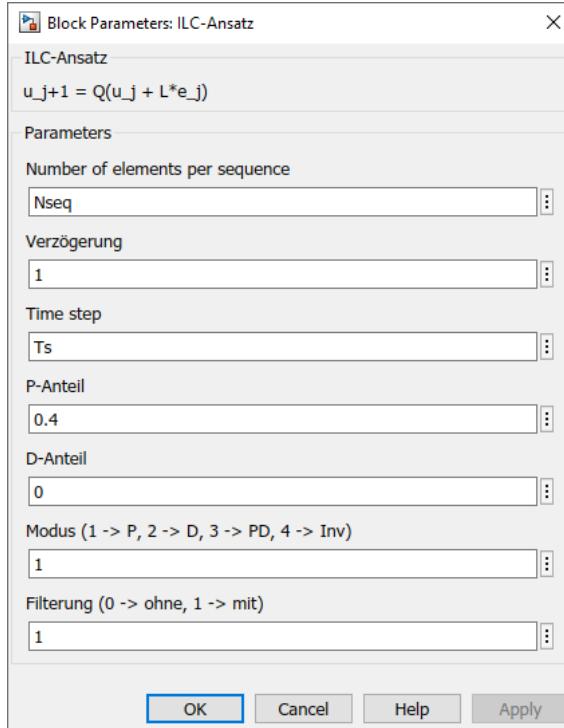


Abbildung 2.5: ILC Subsystem Maske

ILC-Algorithmus

Innerhalb des orangenen Subsystems befindet sich eine Matlab-Funktion (siehe Skript 2.3), welche den ILC-Algorithmus enthält.

```

1 function [u_k,e_k,e_max,e_rms] = fcn(y_d,y_m,kp,kd,Nseq,v,modus,Ts,g,filt)
2 % Variablen definition und Initialisierung
3 assert(Nseq <= 10001);
4 persistent Useq e e_rms_tmp e_max_tmp k E Q D G_inv
5 if isempty(Useq), Useq = zeros(Nseq,1); end
6 if isempty(e), e = zeros(Nseq,1); end
7 if isempty(e_rms_tmp), e_rms_tmp = 0; end
8 if isempty(e_max_tmp), e_max_tmp = 0; end
9 if isempty(k), k = 1; end
10 if isempty(E), E = eye(Nseq); end
11 if isempty(Q) || isempty(D)
12     [Q, D] = calc_QD(Nseq, Ts);
13 end
14 if isempty(G_inv)
15     G_inv = zeros(Nseq,Nseq);
16     for i = 1:Nseq

```

```

17      G_inv(i:Nseq,i) = g(1+r:Nseq-i+r+1);
18    end
19    G_inv = inv(G_inv);
20 end
21 %% ILC-Algorithmus
22 % Stellsignal extrahieren
23 u_k = Useq(k);
24 % Regelfehler berechnen
25 e_k = y_d - y_m;
26 e(k) = e_k;
27 % interne Variablen an Ausgänge übergeben
28 e_rms = e_rms_tmp;
29 e_max = e_max_tmp;
30 % Zähler erhöhen
31 k = k + 1;
32 % berechne vorgezogenes Stellsignal (aufgrund von Verzögerung)
33 if k == (v+1)
34   if modulus == 1
35     Useq(Nseq-v+1:Nseq) = Useq(Nseq-v+1:Nseq) + kp * E(1:v,:) * e(:,1);
36   elseif modulus == 2
37     Useq(Nseq-v:Nseq) = Useq(Nseq-v:Nseq) + kd * D(1:v,:) * e(:,1);
38   elseif modulus == 3
39     Useq(Nseq-v:Nseq) = Useq(Nseq-v:Nseq) + (kp * E(1:v,:) + kd *
40       D(1:v,:)) * e(:,1);
41   elseif modulus == 4
42     Useq(Nseq-v+1:Nseq) = Useq(Nseq-v+1:Nseq) + G_inv(1:v,:) * e(:,1);
43   end
44 end
45 % Zähler zurücksetzen Stellsignal für nächsten Zyklus berechnen und ggf.
46   filtern
47 if k > Nseq
48   k = 1;
49   e = circshift(e,-v);
50   e_rms_tmp = rms(e);
51   e_max_tmp = max(abs(e));
52   if modulus == 1
53     Useq(1:Nseq-v) = Useq(1:Nseq-v) + kp * E(1:Nseq-v,:) * e;
54   elseif modulus == 2
55     Useq(1:Nseq-v) = Useq(1:Nseq-v) + kd * D(1:Nseq-v,:) * e;
56   elseif modulus == 3

```

```

55      Useq(1:Nseq-v) = Useq(1:Nseq-v) + (kp * E(1:Nseq-v,:)) + kd *
56          D(1:Nseq-v,:)) * e;
57      elseif modus == 4
58          Useq(1:Nseq-v) = Useq(1:Nseq-v) + G_inv(1:Nseq-v,:) * e(:,1);
59      end
60      e = zeros(Nseq,1);
61      if filt
62          Useq = Q * Useq;
63      end
64 end

```

Skript 2.3: ILC-Algorithmus

Die Funktion erhält als Eingangsparameter die Eingänge des Subsystems sowie die in der Maske definierten Parameter. Die Ausgabeparameter der Funktion entsprechen den Ausgängen des Subsystems. Zunächst werden die internen Variablen definiert und über die `if isempty(...)` Abfrage initialisiert. Die Filtermatrix **Q** und die Ableitungsmatrix **D** werden in einer externen Funktion berechnet und bei der Initialisierung geladen.

Der eigentliche ILC-Algorithmus beginnt ab Zeile 21. Zunächst wird das neue Stellsignal **u_k** aus dem Vektor **Useq** mithilfe des Abtastindex **k** extrahiert. Danach wird der Regelfehler **e_k** mit Gleichung (2.1) berechnet und in einem Vektor **e** abgespeichert. Zur Bestimmung des maximalen Regelfehlers **e_max** und des Effektivwertes der Regelabweichung **e_rms** werden zusätzliche lokale Variablen benötigt, damit die Berechnung am Ende jeder Iteration stattfinden kann.

In Zeile 31 wird der Abtastindex **k** um eins erhöht. Durch die Verschiebung des Regelfehlers **e** um die Verzögerung **v**, verschiebt sich ein Teil des Stellsignals in die vorherige Iteration. Ab Zeile 32 wird das vorgezogene Stellsignal für das Ende der aktuellen Iteration berechnet. Dafür wird abhängig vom eingestellten Modus die Gleichung (2.20), (2.24) oder (2.27) verwendet.

Zuletzt erfolgt die Überprüfung, ob der Abtastindex **k** größer als die maximale Anzahl der Schritte **Nseq** ist. Trifft dies zu, wird der Abtastindex auf eins zurückgesetzt und das neue Stellsignal wird berechnet. Dafür wird der Vektor **e** um die Verzögerung **v**, laut Gleichung (2.7) verschoben. Anschließend wird, abhängig davon welcher Modus in der Maske ausgewählt wird, das neue Stellsignal **Useq** berechnet. Je nach Einstellung des **filt** Parameters in der Maske, wird die Filtermatrix **Q** in Zeile 60 auf das neu berechnete Stellsignal angewendet. Dadurch wird das Signal unabhängig vom eingestellten Modus gefiltert.

2.5.3 Simulink-Modell ILC-Ansatz + PID-Regler

Das in Abbildung 2.4 gezeigte Modell wird um einen Feedback-Anteil erweitert. Dafür wird ein klassischer PID-Regler eingesetzt. Hierbei gibt es zwei verschiedene Strukturen, die angewendet werden können. Bei der parallelen Struktur (Abbildung 2.6), welche auch als Zwei-Freiheitsgrade-Regelung bezeichnet wird, wird der ILC-Anteil zum Ausgang des Reglers addiert. Anders ist es bei der seriellen Struktur (Abbildung 2.7); hier wird der ILC-Anteil zum Eingang des Reglers addiert. Die parallele Struktur wird als Zwei-Freiheitsgrade-Regelung bezeichnet.

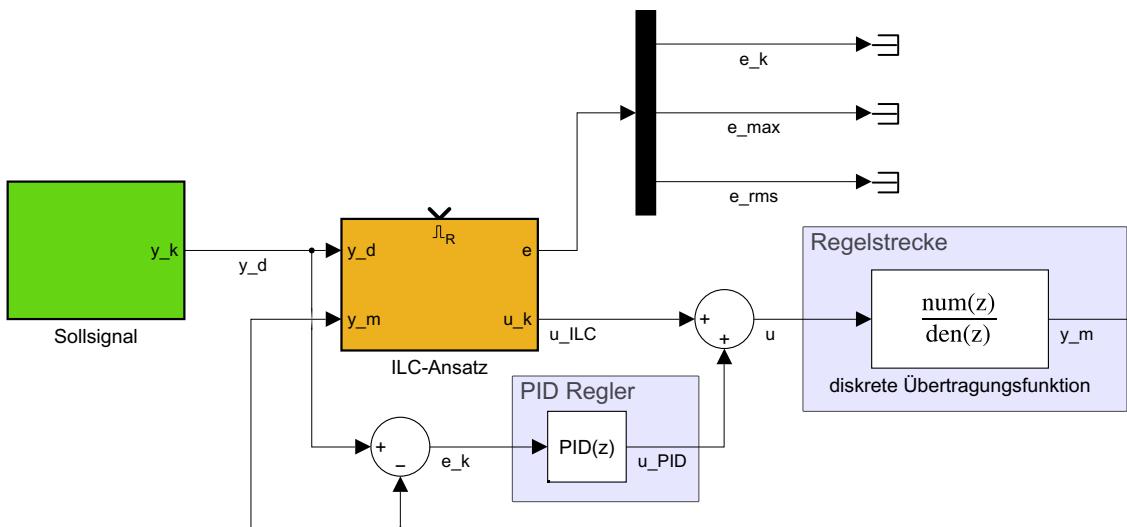


Abbildung 2.6: Matlab Modell ILC-Ansatz mit PID-Regler in paralleler Struktur

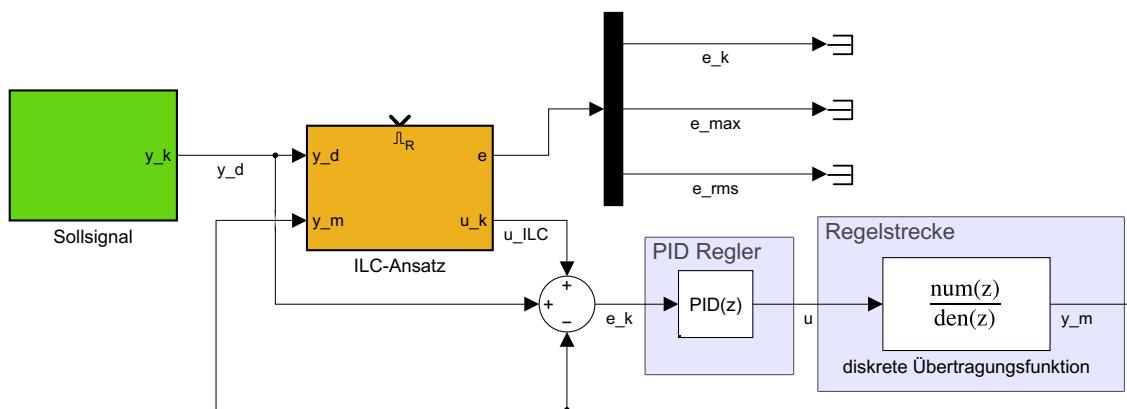


Abbildung 2.7: Matlab Modell ILC-Ansatz mit PID-Regler in serieller Struktur

2.6 Ergebnisse

In diesem Abschnitt werden die Ergebnisse aus der Untersuchung der Stabilitäts- und Konvergenzkriterien präsentiert. Dabei wird außerdem der ILC-Algorithmus an verschiedenen Regelstrecken mit unterschiedlichen ILC-Gesetzen getestet. Die Regelfehlerverläufe und die Konvergenzgeschwindigkeiten der Regelung sollen hierbei unter anderem untersucht und verglichen werden. Außerdem wird die Kombination aus ILC-Ansatz und PID-Regelung simuliert und ebenfalls untersucht. Bei den Simulationen wird der ILC-Ansatz ohne Filterung ($\mathbf{Q} = \mathbf{E}$) eingesetzt.

2.6.1 PT1-System

Als erstes wird der ILC-Ansatz an einem System mit PT1-Verhalten untersucht. Die Übertragungsfunktion im kontinuierlichen Zeitbereich der Regelstrecke $G_s(s)$ ist gegeben durch

$$G_s(s) = \frac{0.1}{0.4s + 1.1}. \quad (2.32)$$

Wie in Abschnitt 2.5.2 erläutert, wird bei der Simulation die Regelstrecke im diskreten Zeitbereich benötigt. Dafür muss die kontinuierliche Übertragungsfunktion $G_s(s)$ mithilfe der Z-Transformation in eine diskrete Übertragungsfunktion $G_s(z)$ transformiert werden. Dies kann zum Beispiel durch eine Rechteck-Vorwärts-Transformation, auch Euler-Transformation genannt, der kontinuierlichen Übertragungsfunktion mit $s = \frac{1}{T_s}(z - 1)$ erfolgen [8]. In Matlab kann die kontinuierliche Übertragungsfunktion mit dem Befehl `c2d(Gs, Ts)` in den diskreten Zeitbereich transformiert werden. Für die Untersuchungen und Simulationen wird eine Abtastzeit von $T_s = 0.1\text{s}$ gewählt. Dadurch ergibt sich die diskrete Übertragungsfunktion (bestimmt mit Matlab) zu

$$G_s(z) = \frac{0.02186}{z - 0.7596}. \quad (2.33)$$

P-Type ILC

Zunächst wird der P-Typ ILC-Ansatz am System angewendet, wobei der Parameter k_p durch Skript 2.2 eingegrenzt wird. Die Ergebnisse der Untersuchung sind in Abbildung 2.8 dargestellt. Im oberen und unteren Diagramm sind die Auswirkungen des Parameters k_p auf die Stabilität, Konvergenz und der asymptotischen Regelabweichung ($e_{\max,\infty}$ und $e_{\text{rms},\infty}$) des Systems zu sehen.

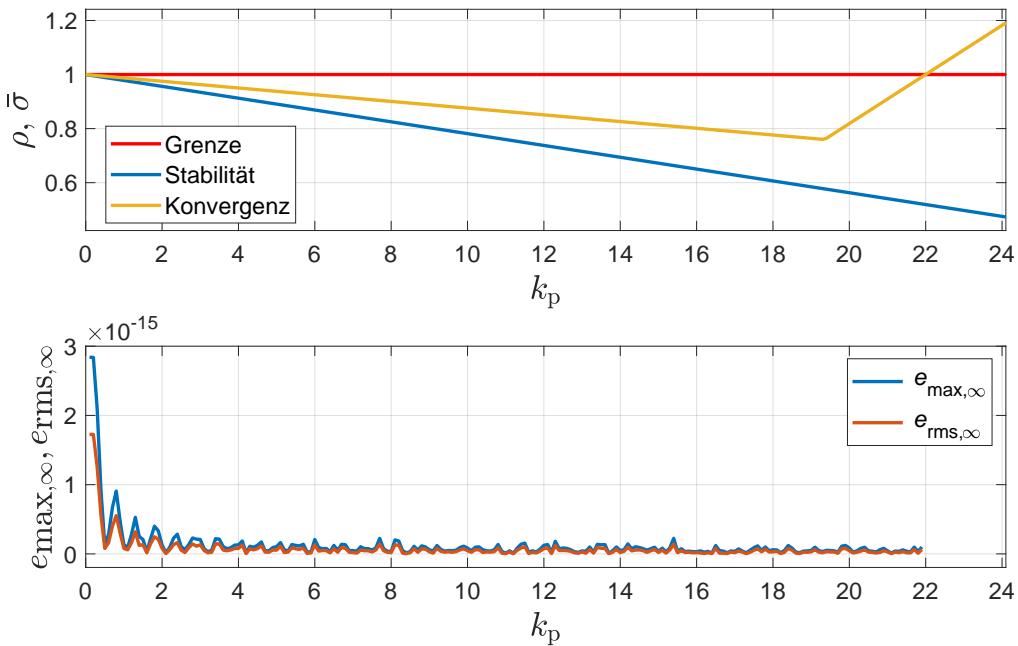


Abbildung 2.8: Stabilitäts- und Konvergenzuntersuchung PT1-System

Das obere Diagramm zeigt die Stabilitäts- und Konvergenzergebnisse für verschiedene k_p -Werte. Um die Stabilitäts- und Konvergenzkriterien zu erfüllen, muss der berechnete Wert kleiner als eins sein (vgl. Satz 1 und Satz 3). Die Ergebnisse zeigen, dass der ILC-Ansatz für k_p -Werte unter 22 stabil ist und eine monotone Konvergenz aufweist. Für Werte ab 22 ist zwar das Stabilitätskriterium weiterhin erfüllt, allerdings ist das Konvergenzkriterium nicht mehr erfüllt.

Im unteren Schaubild ist die maximale Regelabweichung und der Effektivwert der Regelabweichung in Abhängigkeit von k_p dargestellt. Je höher der k_p -Wert, desto kleiner ist die Regelabweichung. Die Formel zur Berechnung der asymptotischen Regelabweichung ist nur anwendbar, solange die Stabilitäts- und Konvergenzkriterien erfüllt sind. Aus diesem Grund wurde ab einem k_p -Wert von 22 keine Berechnung durchgeführt.

Die Ergebnisse zeigen, dass der P-Typ ILC-Ansatz mit einem k_p -Wert im Bereich von $0 < k_p < 22$ für die Strecke (2.32) beziehungsweise (2.33) eingesetzt werden kann, um eine stabile Regelung mit einer Konvergenz der Regelabweichung auf Werte im Bereich von 10^{-16} zu erreichen.

Diese theoretischen Untersuchungen werden im nächsten Schritt, anhand einer Simulation in Matlab-Simulink, untersucht. Dafür wird das Modell aus Abbildung 2.4 mit der definierten diskreten Übertragungsfunktion (2.33) verwendet. Als P-Anteil wird ein k_p -Wert von 4 eingestellt, welcher heuristisch bestimmt wurde. In Abbildung 2.9 sind die Ergebnisse der Simulation zu sehen.

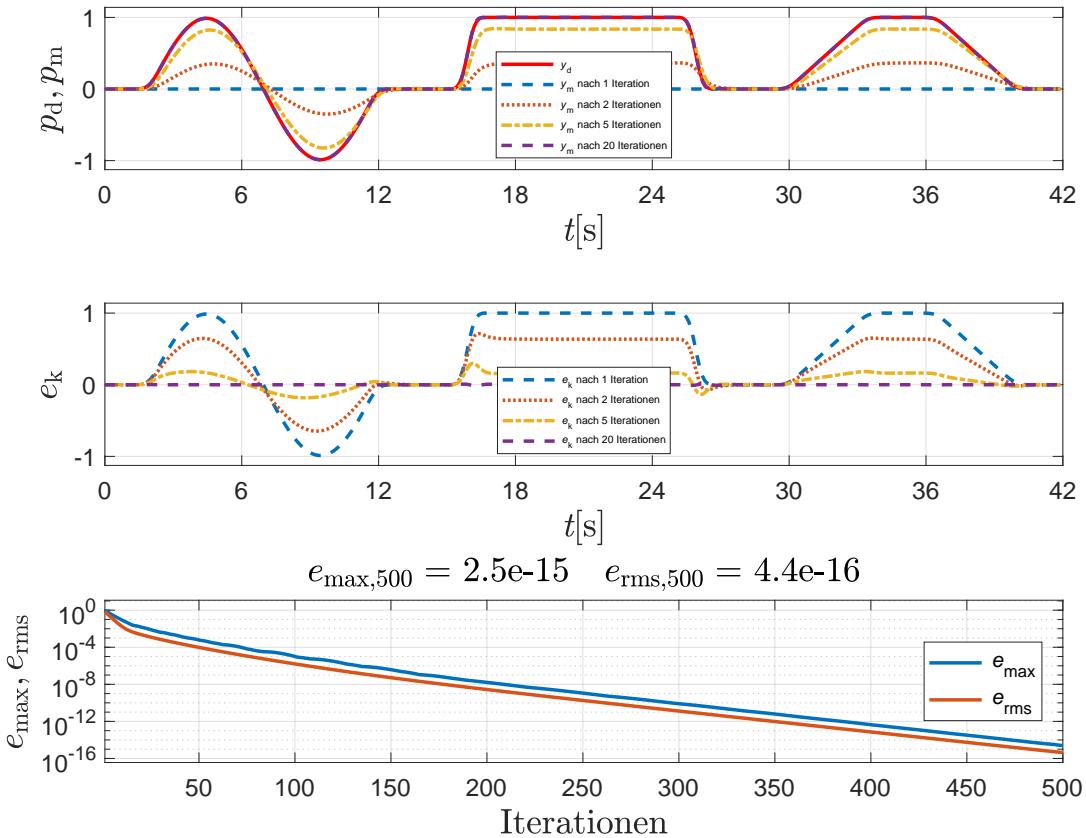


Abbildung 2.9: Simulationsergebnis P-Typ ILC anhand PT1-System

Im oberen Schaubild ist das Sollsignal y_d in rot und die gemessenen Signale y_m , nach einer bestimmten Anzahl an Iterationen, dargestellt. Die Regelabweichung zwischen diesen Signalen, welche zu jedem Zeitschritt k aufgezeigt wird, ist im mittleren Schaubild zu sehen. Das ILC-Gesetz hängt nur von der vorherigen Iteration ab, wodurch das Istsignal nach einer Iteration gleich null ist und die Regelabweichung dem Sollsignal entspricht. Mit zunehmender Anzahl von Iterationen nähert sich das gemessene Signal dem Sollsignal an und die Regelabweichung wird kleiner. Nach 20 Iterationen ist im oberen Schaubild fast kein Unterschied zwischen Sollsignal und gemessenem Signal erkennbar. Lediglich bei der impulsartigen Ansteigung ist ein kleiner Knick bei der Regelabweichung zu erkennen.

Im unteren Schaubild wird die Regelabweichung (e_{\max} und e_{rms}) in Abhängigkeit der Iterationen aufgezeigt. Die y-Achse ist dabei logarithmisch dargestellt. Mit zunehmender Anzahl von Iterationen wird die Regelabweichung kleiner. Nach 500 Iterationen erreicht der e_{\max} -Wert $2.5 \cdot 10^{-15}$ und der e_{rms} -Wert $4.4 \cdot 10^{-16}$.

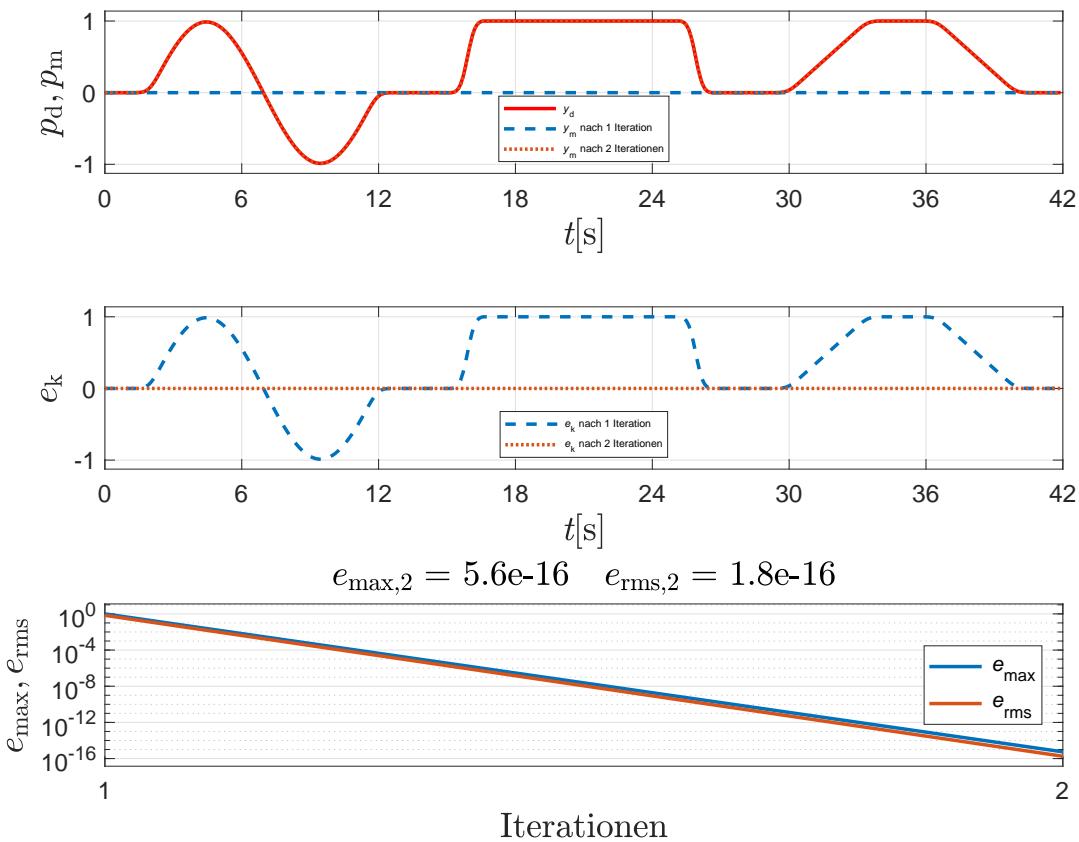
Die Wahl des Parameters k_p beeinflusst die Konvergenzgeschwindigkeit der Regelabweichung. Größere k_p -Werte führen zu einer schnelleren Konvergenz, während kleinere Werte zu einer langsameren Konvergenz führen. Wenn der k_p -Wert kurz unterhalb

des erlaubten Bereichs liegt, kommt es zum Überschwingen des Istsignals im Vergleich zum Sollsignal, was ein vergleichbares Verhalten wie bei einer Sprungantwort eines PID-Reglers darstellt.

Für k_p -Werte ab 22 konvergiert das Istsignal nicht und die Regelabweichung steigt ins Unendliche. Die Simulationsergebnisse stimmen mit den theoretischen Untersuchungen überein.

Inversionsbasiertes ILC

Im Folgenden wird der inversionsbasierte ILC-Ansatz untersucht. Dafür wird der Modus vier in der Maske des ILC-Subsystems eingestellt. Die Ergebnisse der Simulation sind in Abbildung 2.10 zu sehen. Das Schaubild hat den gleichen Aufbau wie die Abbildung 2.9.



In den oberen beiden Schaubildern ist zu erkennen, dass bereits nach zwei Iterationen (einer Wiederholung) das Istsignal dem Sollsignal entspricht. Die Regelabweichung ist dabei auf ihren minimalen Wert konvergiert.

ILC-Ansatz + PI-Regler

In diesem Abschnitt erfolgt eine Untersuchung einer Kombination aus ILC-Ansatz und PI-Regler. Hierbei werden die Simulink-Modelle aus Abschnitt 2.5.3 in Verbindung mit der gegebenen diskreten Übertragungsfunktion Gleichung (2.33) verwendet. Die Einstellung der Regelparameter des PI-Reglers (ideal Struktur mit Anti-Wind-Up) erfolgt mittels der in Matlab verfügbaren PID Tuner App und wird durch verschiedene Simulationen in Kombination mit dem ILC-Ansatz angepasst. Es werden Werte von 6.1 für den P-Anteil und 2.9 s^{-1} für den I-Anteil ermittelt. Der ILC-Ansatz wird für beide Untersuchungen als P-Type eingesetzt. Die k_p -Parameter des ILC-Ansatzes wurden experimentell ermittelt.

Im ersten Schritt erfolgt eine Untersuchung der parallelen Struktur. Dafür wird ein Wert von 15 für den k_p -Parameter des ILC-Ansatzes eingestellt. Die Ergebnisse der Simulation sind in Abbildung 2.11 dargestellt.

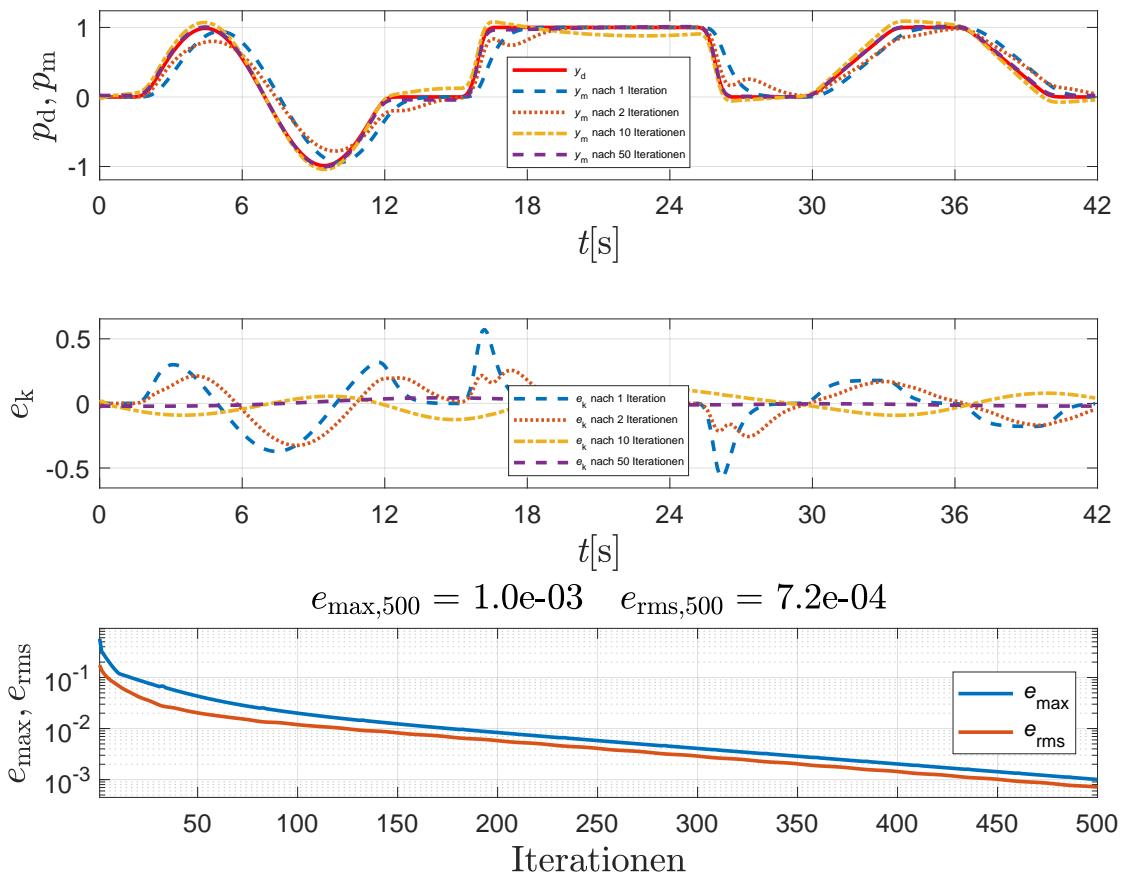


Abbildung 2.11: Simulationsergebnis ILC-Ansatz + PI-Regler in paralleler Struktur anhand PT1-System

Die blauen Linien in den oberen beiden Diagrammen zeigen das Ergebnis der Rege-

lung allein durch den PI-Regler (ILC-Ansatz benötigt eine Iteration bis er wirksam wird). Durch den ILC-Ansatz nähert sich das Istsignal, nach mehreren Iterationen, schrittweise dem Sollsignal an. Der Effektivwert und die maximale Regelabweichung betragen nach 500 Iterationen $7.2 \cdot 10^{-4}$ und $1.0 \cdot 10^{-3}$. Die Regelgüte des PI-Reglers konnte somit signifikant verbessert werden.

Mittels Simulationen mit verschiedenen Reglerparametern wird festgestellt, dass der Integral-Anteil des Reglers die Konvergenzgeschwindigkeit der ILC-Methode entgegenwirkt. Bei Nullsetzung des Integral-Anteils des Reglers kann die Konvergenzgeschwindigkeit erhöht und die Regelgüte nach 500 Iterationen verbessert werden. Allerdings ist die anfängliche Regelgüte (Regelung nur mit P-Regler) deutlich schlechter.

Als nächstes wird die serielle Struktur untersucht. Dafür wird ein Wert von eins als k_p -Parameter für den ILC-Ansatz eingestellt. Dieser Wert muss deutlich geringer sein als bei der parallelen Struktur, da der P-Anteil im Regler das Stellsignal des ILC-Ansatzes zusätzlich verstärkt. Die Simulationsergebnisse sind in Abbildung 2.12 zu sehen.

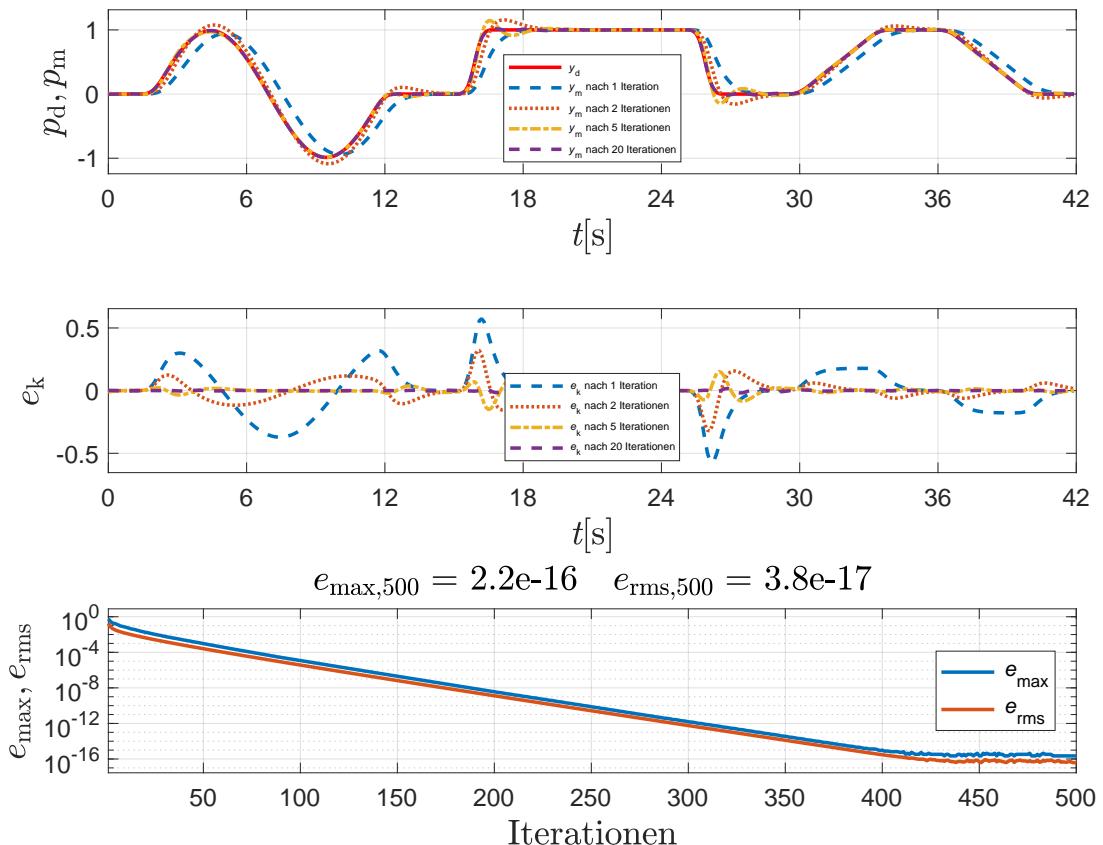


Abbildung 2.12: Simulationsergebnis ILC-Ansatz + PI-Regler in serieller Struktur anhand PT1-System

Die oberen beiden Schaubilder zeigen in blauer Farbe wieder die Ergebnisse einer Regelung, die ausschließlich durch den PI-Regler erfolgt. Die Anwendung des ILC-Ansatzes in der seriellen Struktur führt zu einer verbesserten Performance im Vergleich zur parallelen Struktur. Die Konvergenzgeschwindigkeit ist erhöht und die bleibende Regelabweichung ist signifikant geringer.

Die Performance der Regelung hängt jedoch stark von den Regelparametern des PI-Reglers und des zu regelnden Systems ab. Deshalb kann keine allgemeine Aussage darüber getroffen werden, ob eine serielle Struktur im Vergleich zur parallelen Struktur zu besseren Ergebnissen führt.

Zusammenfassung Untersuchung PT1-System

Um einen besseren Überblick über die erzielten Ergebnisse zu erhalten sind in Tabelle 2.1 die einzelnen Regelkonzepte aufgeführt. Dabei wird die anfangs Regelabweichung, also der Regelabweichung ohne ILC-Ansatz, sowie die Konvergenzgeschwindigkeit, als auch die Regelabweichung nach vielen durchgeföhrten Iterationen aufgeführt.

Regelkonzept	Regelabweichung bei erster Iteration ($j = 1$)	Konvergenzgeschwindigkeit	Stationäre Regelabweichung ($j \rightarrow \infty$)
P-Type ILC	--	0	++
Inversionsbasiertes ILC	--	++	++
P-Type ILC + PI-Regler	+	0	+

Tabelle 2.1: Vergleich der Regelkonzepte anhand PT1-System

2.6.2 PT2-System

Analog zu den Untersuchungen am PT1-System, wird der ILC-Ansatz an einem System mit PT2-Verhalten untersucht. Die nachfolgenden Schaubilder sind gleich wie im vorherigen Abschnitt 2.6.1 zu interpretieren.

Aus der Vorlesung *Regelungstechnik* ist die Übertragungsfunktion für ein vereinfachtes Modell eines Gleichstrommotors bekannt [8]. Die kontinuierliche Übertragungsfunkti-

on ist gegeben durch

$$G_s(s) = \frac{4}{7 \cdot 10^{-7}s^2 + 7.5 \cdot 10^{-3}s + 1}. \quad (2.34)$$

Daraus folgt durch die Z-Transformation mit einer Abtastzeit von $T_s = 0.1\text{ s}$ die diskrete Übertragungsfunktion

$$G_s(z) = \frac{4z + 1.371 \cdot 10^{-7}}{z^2 - 1.668 \cdot 10^{-7}z}. \quad (2.35)$$

P-Type ILC

In Abbildung 2.13 sind die Ergebnisse für die Stabilitäts- und Konvergenzuntersuchung zu sehen.

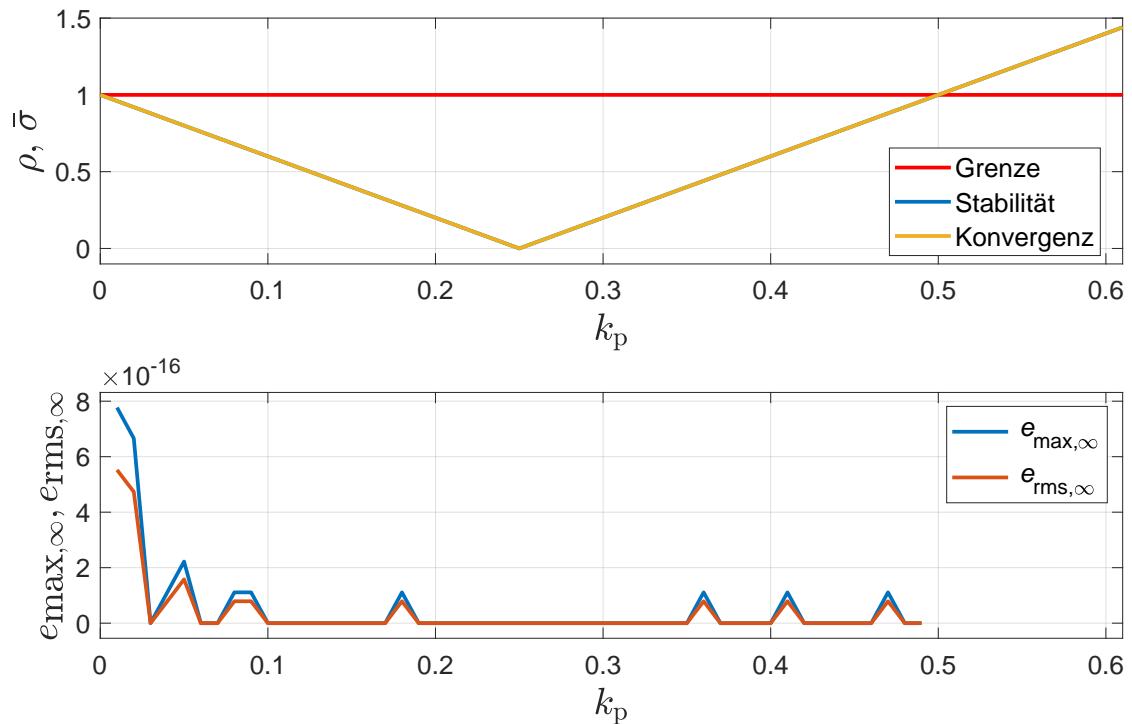


Abbildung 2.13: Stabilitäts- und Konvergenzuntersuchung PT2-System

Für das gegebene System erfüllen k_p -Werte von $0 < k_p < 0.5$ die Stabilitäts- und Konvergenzkriterien und die Regelabweichung konvergiert im Bereich von $0.2 < k_p < 0.35$ auf Werte von bis zu $5 \cdot 10^{-26}$.

Für die Simulation wird ein k_p -Wert von $k_p = 0.2$ eingestellt. Die Ergebnisse dieser Simulation sind in Abbildung 2.14 aufgezeigt.

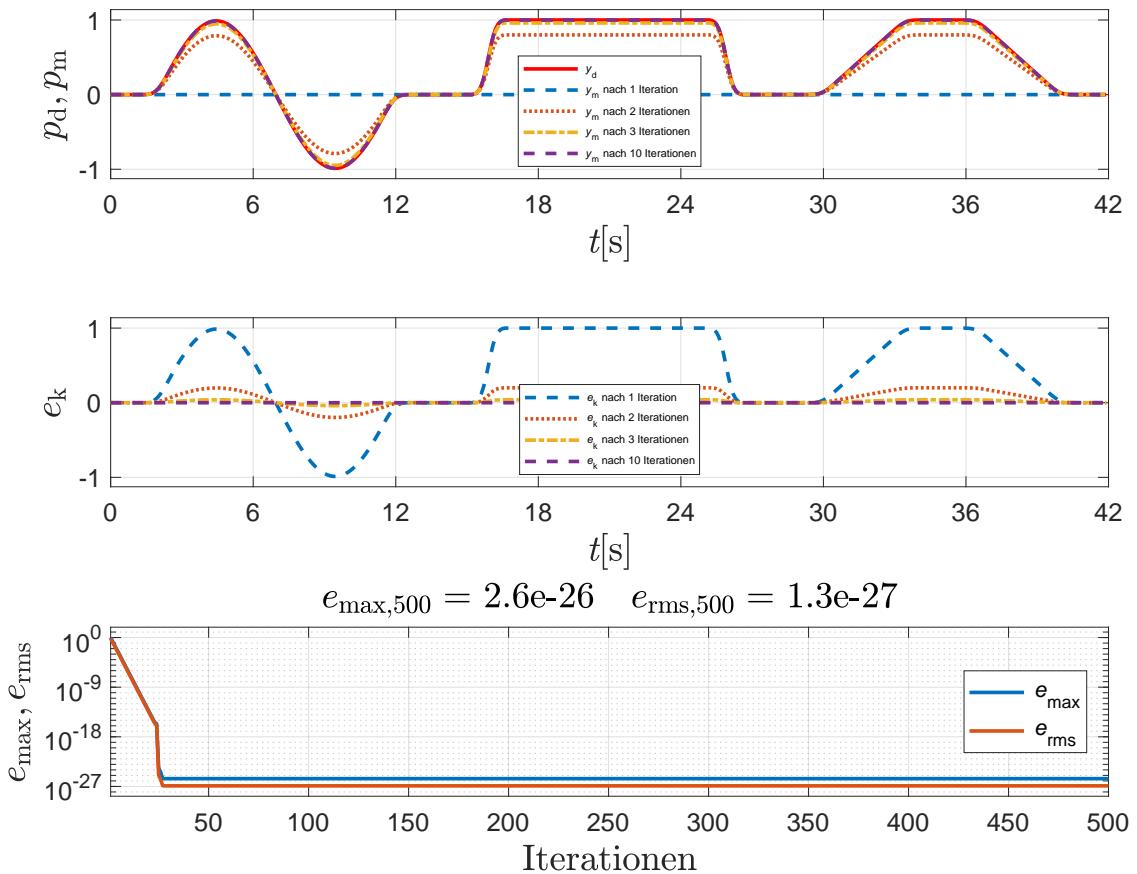


Abbildung 2.14: Simulationsergebnis P-Typ ILC anhand PT2-System

Nach bereits 10 Iterationen ist im Schaubild kein Unterschied zwischen Soll- und Istsignal zu erkennen. Der maximale Regelfehler und der Effektivwert der Regelabweichung konvergieren auf Werte von $2.6 \cdot 10^{-26}$ und $1.3 \cdot 10^{-27}$. Die davor berechneten Werte stimmen mit der Simulation überein. Auch hier gilt, umso größer der k_p -Wert, desto höher die Konvergenzgeschwindigkeit. Der unerklärlich große Sprung, der bei Iteration 25 auftritt, ist in Anbetracht der geringen Werte nicht von Bedeutung.

Inversionsbasiertes ILC

Als nächstes wird der inversionsbasierte ILC-Ansatz an dem System mit PT2-Verhalten getestet. Die Simulationsergebnisse sind in Abbildung 2.15 dargestellt. Es ist das gleiche Verhalten wie an dem System mit PT1-Verhalten zu erkennen. Der Regelfehler konvergiert innerhalb einer Iteration auf seinen stationären Endwert.

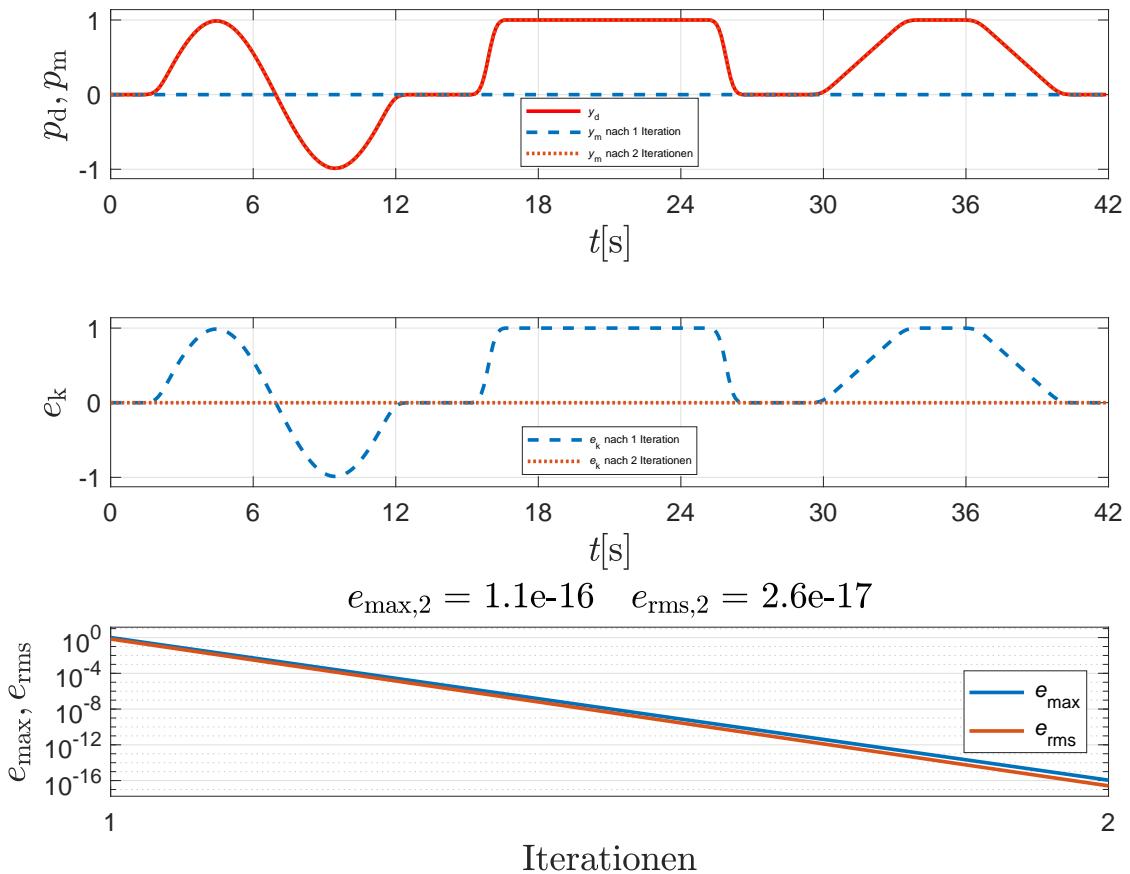


Abbildung 2.15: Simulationsergebnis Inversionsbasiertes ILC anhand PT2-System

ILC-Ansatz + Vorsteuerung

Für das betrachtete System mit PT2-Verhalten zeigt die Kombination aus ILC-Ansatz und PI-Regler ein identisches Verhalten im Vergleich zum System mit PT1-Verhalten (vgl. Abschnitt 2.6.1). Aus diesem Grund wird im nachfolgenden Versuch eine Zweifreiheitsgrade-Regelung mit einer Vorsteuerung für das PT2-System getestet. Hierbei wird ein flachheitsbasierter Ansatz verwendet, bei dem das gegebene System Gleichung (2.35) invertiert und der Ausgang zu dem Stellsignal des ILC-Ansatzes addiert wird [9]. Das entsprechende Simulink-Modell ist in Abbildung 2.16 dargestellt.

Für die flachheitsbasierte Vorsteuerung werden als Eingänge das Sollsignal y_d sowie dessen erste und zweite Ableitungen (\dot{y}_d , \ddot{y}_d) benötigt. Hierfür wird ein von Festo eigen erstellter Zustandsvariablenfilter verwendet. Dieser Block erhält als Eingänge das gewünschte Sollsignal sowie die Stellgrößenbeschränkungen und die Initialisierungswerte für die Integratoren. Als Ausgang wird das Sollsignal sowie dessen Ableitungen bereitgestellt.

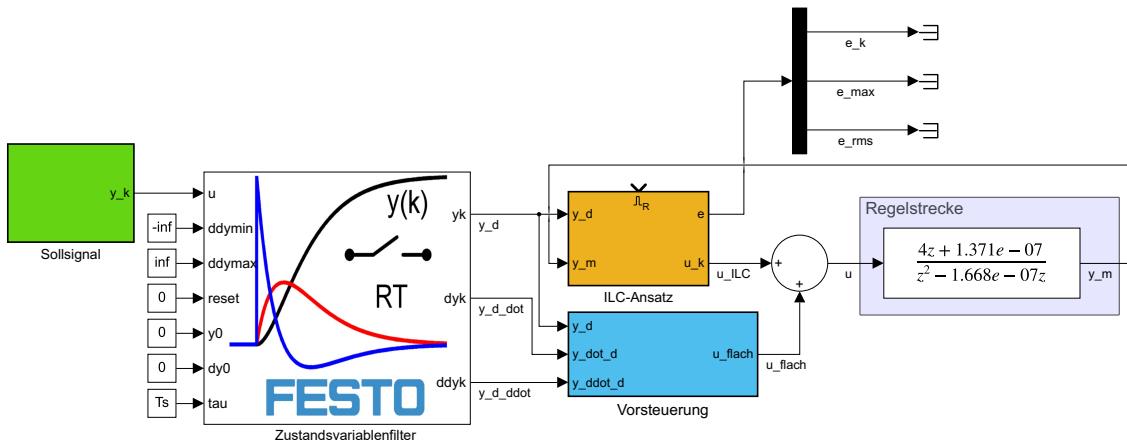


Abbildung 2.16: ILC-Ansatz + flachheitsbasierte Vorsteuerung

Das in blau dargestellte Subsystem des Modells beinhaltet die flachheitsbasierte Vorsteuerung, dessen Ausgang auf das Stellsignal des ILC-Ansatzes addiert wird. Die restlichen Komponenten des Modells sind bereits aus den vorherigen Abschnitten bekannt.

Wie bereits erwähnt, wird die Vorsteuerung auf Basis der Invertierung des gegebenen Systems konstruiert. Hierfür wird die kontinuierliche Übertragungsfunktion Gleichung (2.34) invertiert und jedes s durch die erste bzw. s^2 durch die zweite Ableitung ersetzt. Daraus ergibt sich die Vorsteuerung gemäß folgender Gleichung:

$$u_{\text{flach}} = \frac{1}{4} (y_d + 7.5 \cdot 10^{-3} \dot{y}_d + 7 \cdot 10^{-7} \ddot{y}_d) . \quad (2.36)$$

Aus Gleichung (2.36) ergibt sich das dazugehörige Simulink-Modell wie in Abbildung 2.17 abgebildet.

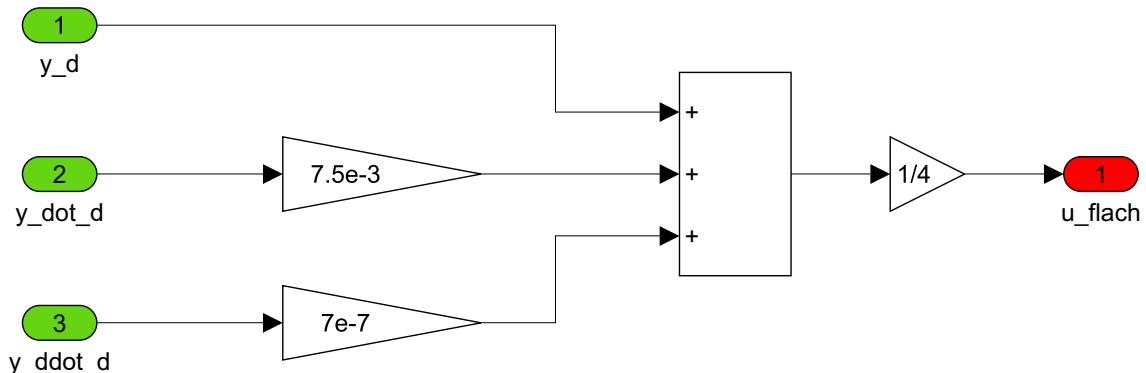


Abbildung 2.17: flachheitsbasierte Vorsteuerung

Für die Durchführung der Simulation wird der P-Type ILC-Ansatz mit einem k_p -

Parameter von 0.2 eingesetzt. Zusätzlich wird das Messsignal y_m um fünf Zeitschritte verzögert, um eine zusätzliche Totzeit im System zu erzeugen und das Verhalten des ILC-Ansatzes unter diesen Bedingungen genauer zu untersuchen. Der Verzögerungsparameter v im ILC-Algorithmus setzt sich aus dem relativen Grad der diskreten Übertragungsfunktion $r = 1$ und der eingebauten Verzögerung von fünf Abtastschritten zusammen. Dadurch muss der Parameter $v = 6$ eingestellt werden. Die Ergebnisse der Simulation sind in Abbildung 2.18 aufgezeigt.

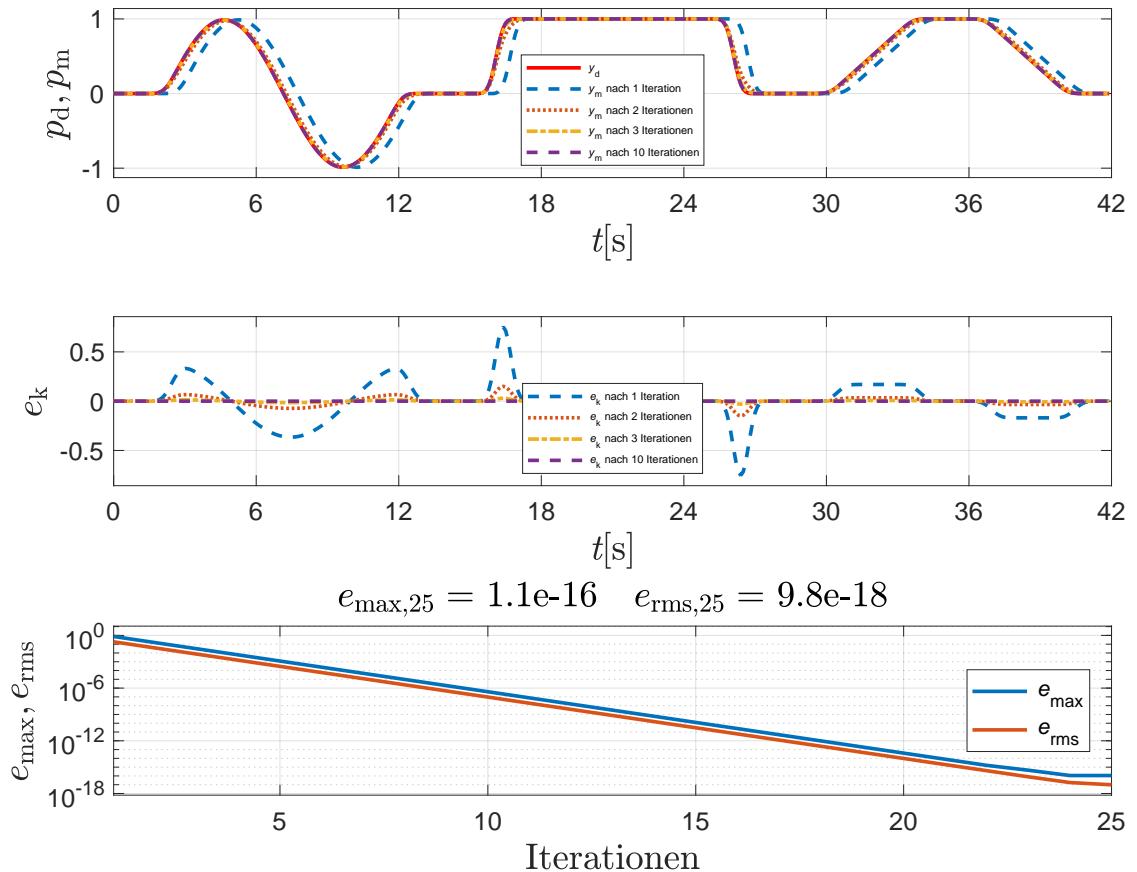


Abbildung 2.18: Simulationsergebnis ILC-Ansatz + flachheitsbasierte Vorsteuerung anhand PT2-System

Die dargestellten blauen Linien zeigen das Systemverhalten, welches allein durch die Vorsteuerung erreicht wird. In diesem Fall folgt das Istsignal dem Sollsignal mit einer bestimmten Phasenverschiebung, welche durch die manuell eingefügte Verzögerung entsteht. Durch den Einsatz des ILC-Ansatzes kann die Phasenverschiebung bereits nach wenigen Iterationen vollständig kompensiert werden. Nach 25 Iterationen beträgt der maximale Regelfehler lediglich $1.1 \cdot 10^{-16}$ und der Effektivwert der Regelabweichung $9.8 \cdot 10^{-18}$.

Zusammenfassung Untersuchung PT2-System

Analog zu den Untersuchungen des PT1-Systems, sind die verschiedenen Regelkonzepte für das PT2-System in Tabelle 2.2 aufgeführt. Hierbei werden erneut die anfängliche Regelabweichung, also die Abweichung ohne den ILC-Ansatz, die Konvergenzgeschwindigkeit sowie die Regelabweichung nach zahlreichen Iterationen bewertet.

Regelkonzept	Regelabweichung bei erster Iteration ($j = 1$)	Konvergenzgeschwindigkeit	Stationäre Regelabweichung ($j \rightarrow \infty$)
P-Type ILC	--	0	++
Inversionsbasiertes ILC	--	++	++
P-Type ILC + Vorsteuerung	+	+	++

Tabelle 2.2: Vergleich der Regelkonzepte anhand PT2-System

2.6.3 Zusammenfassung

In diesem Abschnitt werden die Ergebnisse und Erkenntnisse der durchgeführten Untersuchungen zusammengefasst. Die Stabilitäts- und Konvergenzanalysen haben es ermöglicht, die Einstellparameter für den ILC-Ansatz bei verschiedenen ILC-Gesetzen einzuschränken und zu bestimmen. Durch Simulationen mit Matlab-Simulink wurde der ILC-Ansatz an verschiedenen linearen Systemen getestet. Wenn der ILC-Ansatz als alleiniger Regler eingesetzt wird, konvergiert die Regelabweichung auf numerisch null. Bei der Kombination aus ILC-Ansatz und einem PI-Regler verbessert sich die Regelung signifikant. Allerdings arbeiten beide Regler teilweise gegeneinander, weshalb die stationäre Regelgüte nicht so gering ist als bei der Regelung mit alleinigem ILC-Ansatz. Die Regelparameter haben darauf einen starken Einfluss.

Der größte Vorteil des ILC-Ansatzes wurde bei der letzten Untersuchung, der Kombination aus ILC-Ansatz und flachheitsbasierter Regelung, deutlich. Durch das iterative Lernen der Sollsignal ist der ILC-Ansatz in der Lage, Phasenverschiebungen und Totzeiten im System zu kompensieren.

3 Konstantes Volumen

In diesem Kapitel findet die praktische Anwendung der gewonnenen Erkenntnisse aus dem vorherigen Kapitel statt. Hierfür wird ein Versuchsaufbau zur Druckregelung eines konstanten Volumens eingesetzt. Der Versuchsaufbau wird sowohl simulativ mithilfe eines Matlab-Simulink-Modells als auch mit realer Hardware getestet. Dieser Versuchsaufbau stellt eine Vorstufe zum pneumatischen Gelenk dar und ermöglicht somit eine erstmalige Überprüfung des Verhaltens des ILC-Ansatzes an einem nicht-linearen System, sowie unter realen Bedingungen.

Um das Verständnis der Regelaufgabe für den Versuchsaufbau zu erleichtern, ist in Abbildung 3.1 das dazugehörige Blockschaltbild abgebildet.

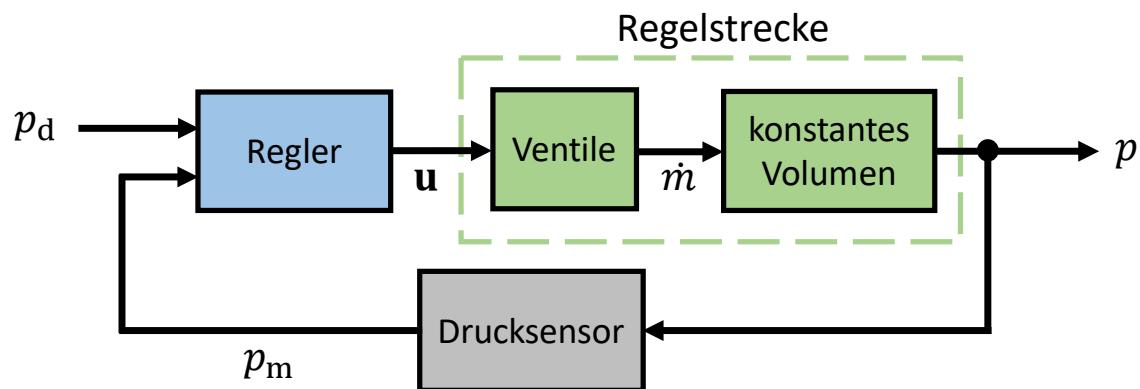


Abbildung 3.1: Blockschaltbild Druckregelung konstantes Volumen

3.1 Simulation

Als erstes wird der Versuchsaufbau in Matlab-Simulink modelliert und simuliert. Für die Druckregelung wird ein flachheitsbasierter Regler in Kombination mit einem ILC-Ansatz eingesetzt.

3.1.1 Modellierung Gesamtsystem

Das zu regelnde System besteht aus zwei Piezoventilen zur Be- und Entlüftung eines konstanten Volumens. Sowohl für die Ventile als auch für das Volumen sind bereits vorhandene Simulink-Blöcke in der Festo internen Bibliothek verfügbar. In Abbildung 3.2 ist der Block für das pneumatische Volumen dargestellt.

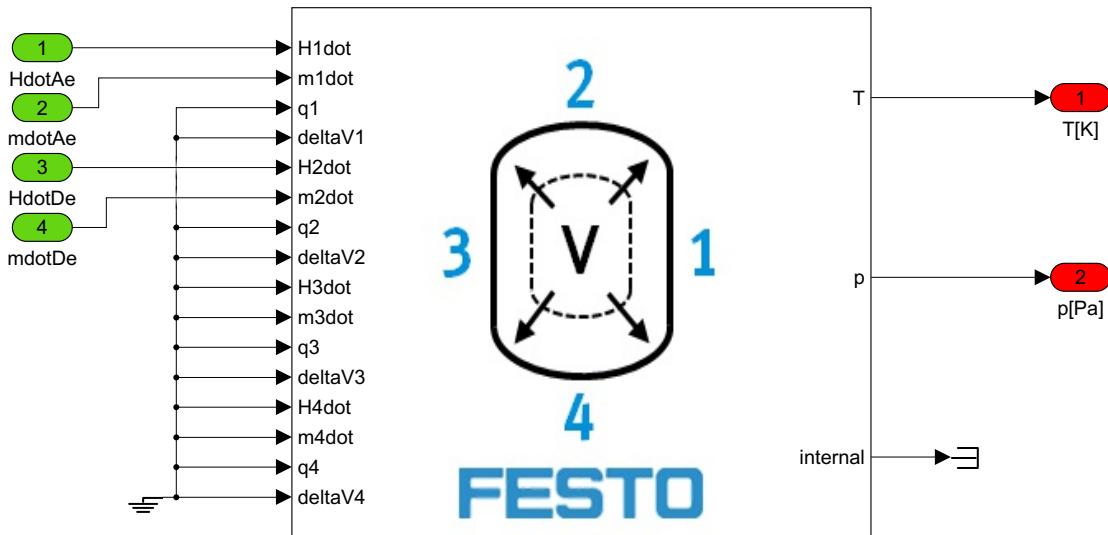


Abbildung 3.2: Matlab Modell pneumatisches Volumen

Der Block erhält als Eingänge den Massenfluss für das Be- und Entlüften sowie die dazugehörigen Enthalpieströme. Als Ausgänge liefert der Block den aktuellen Druck und die Temperatur. Bei diesem Versuchsaufbau wird die Temperatur als konstant angenommen. Dies kann durch die Parameter in der Maske gesteuert werden.

Der vorliegende Block ermöglicht die Simulation von variablen Volumen. Für diesen speziellen Versuch ist es jedoch erforderlich, die Eingänge δV auf null zu setzen, um ein konstantes Volumen darzustellen.

In Abbildung 3.3 ist der Block für zwei Piezoventile aufgezeigt.

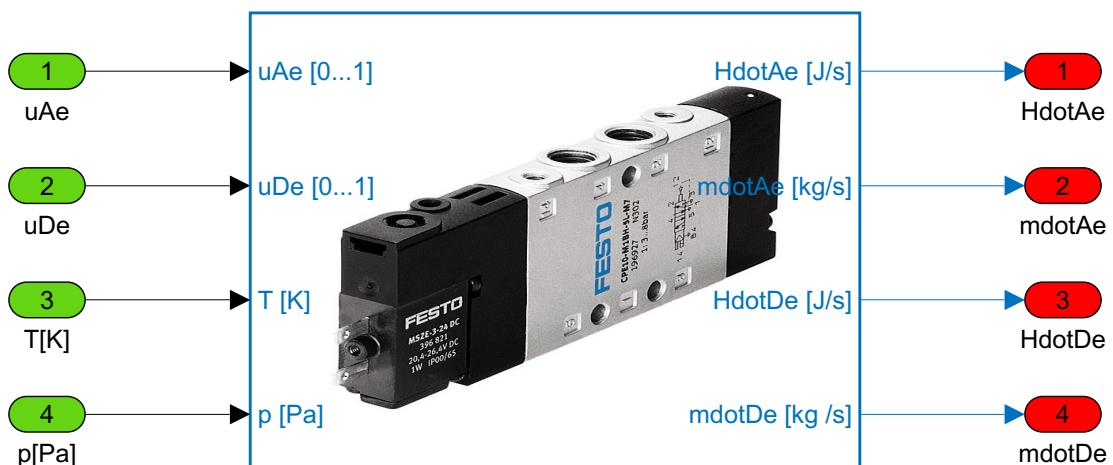


Abbildung 3.3: Matlab Modell Piezoventile

Die erforderlichen Parameter werden über die Maske an den Block übergeben. Der

Block erfordert als Eingangssignale das Stellsignal für die Be- und Entlüftung sowie den Druck und die Temperatur zwischen den beiden Ventilen. Anhand dieser Eingangssignale werden die Massenströme und die zugehörigen Enthalpieströme berechnet, die am Ausgang dieses Blocks zur Verfügung gestellt werden.

Das Gesamtsystem in Abbildung 3.4, bestehend aus den Abbildungen 3.2 und 3.3, besitzt als Eingänge die Stellsignale für die Be- und Entlüftung und als Ausgänge die Temperatur in Kelvin sowie den absoluten Druck in Pascal im Druckluftbehälter.

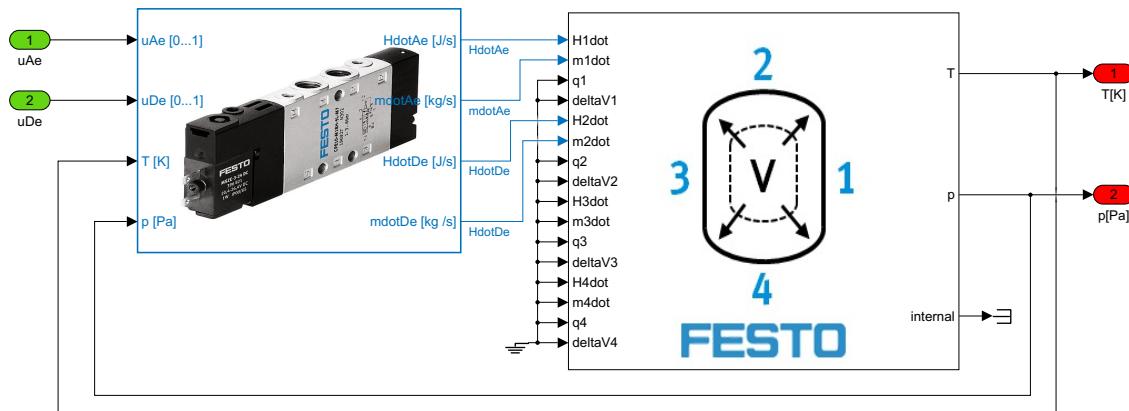


Abbildung 3.4: Matlab Modell Gesamtsystem konstantes Volumen

3.1.2 Flachheitsbasierter Regler

Für den Entwurf des flachheitsbasierten Reglers wird das inverse System benötigt. In einer vorherigen Bachelorarbeit (siehe [10]), wurde bereits ein flachheitsbasierter Regler für das gesamte pneumatische Gelenk entworfen. Daraus stammt das inverse Ventil für diesen Entwurf. Das dazugehörige Matlab Modell ist in Abbildung 3.5 dargestellt.

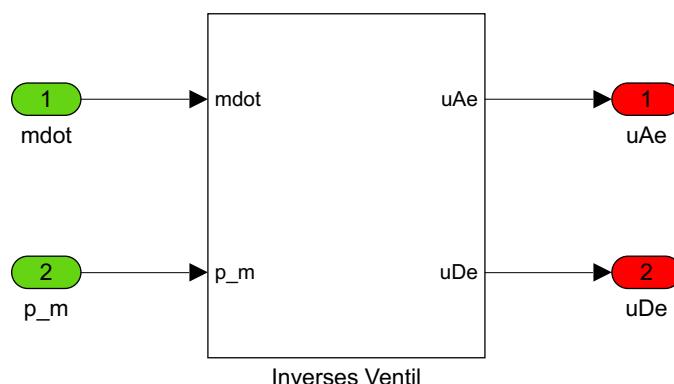


Abbildung 3.5: Matlab Inverses Ventil

Aus dem gemessenen Druck und dem berechneten Massenfluss wird das dazugehörige Stellsignal für das Be- und Entlüften der Ventile berechnet.

Die Gleichungen zur Modellierung des inversen pneumatischen Volumens sind der Festo-Dokumentation über die Simulink-Blöcke aus Quelle [11] entnommen.

Im ersten Schritt wird die Gleichung für die Druckableitung \dot{p} in Abhängigkeit des Massenstroms \dot{m} bzw. Enthalpiestroms \dot{H} hergeleitet. Hierbei werden die Gleichungen für ideale Gase

$$pV = mRT \quad (3.1)$$

für die innere Energie

$$U = c_v m T \quad (3.2)$$

und für die Ableitung der inneren Energie mit $V = \text{const.} \rightarrow \dot{V} = 0$

$$\dot{U} = \dot{H} - p\dot{V} = \dot{H} \quad (3.3)$$

herangezogen. Die Variable m steht für die Masse im Volumen (Ableitung entspricht dem Massenstrom \dot{m}). Der Faktor R steht für die ideale Gaskonstante und c_v für die spezifische Wärmekapazität bei konstantem Volumen. Durch Umstellen der Gleichung (3.1) ergibt sich für die Masse

$$m = \frac{pV}{RT}. \quad (3.4)$$

Durch Einsetzen von Gleichung (3.4) in Gleichung (3.2) ergibt sich

$$U = \frac{c_v V}{R} p \quad (3.5)$$

und die dazugehörige Ableitung (nur p zeitabhängig)

$$\dot{U} = \frac{c_v V}{R} \dot{p}. \quad (3.6)$$

Durch Gleichsetzen der Gleichung (3.3) mit Gleichung (3.6) kann der Enthalpystrom mit

$$\dot{H} = \frac{c_v V}{R} \dot{p} \quad (3.7)$$

berechnet werden. Mit

$$\dot{H} = c_p T \dot{m} \quad (3.8)$$

(c_p = spezifische Wärmekapazität bei konstantem Druck) und dem Polytropenexpo-

nant

$$n = \frac{c_p}{c_v} \quad (3.9)$$

ergibt sich aus Gleichung (3.7)

$$\dot{p} = \frac{nRT}{V} \dot{m}. \quad (3.10)$$

Im letzten Schritt muss die Gleichung (3.10) umgestellt werden, um den Massenstrom in Abhängigkeit der Druckableitung berechnen zu können. Dadurch ergibt sich das Inverse pneumatische Volumen zu

$$\dot{m} = \frac{V}{nRT} \dot{p}. \quad (3.11)$$

Der Nachweis, dass es sich um ein flaches System handelt, wurde bereits in der herangezogenen Bachelorarbeit (siehe [10]) erbracht und wird daher nicht erneut aufgeführt.

Durch die Verwendung des invertierten Systems wird das Gesamtmodell auf eine reduzierte lineare Form des nichtlinearen konstanten Volumenmodells gebracht. Dies ist in Abbildung 3.6 schematisch abgebildet.

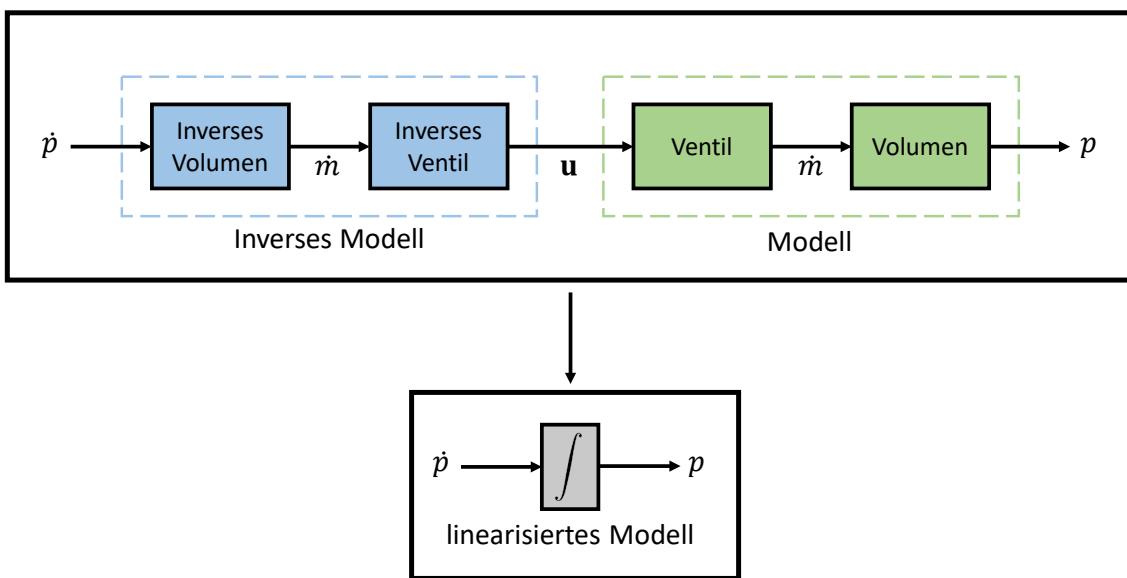


Abbildung 3.6: Schematischer Aufbau reduziertes konstantes Volumenmodell

Durch das reduzierte System kann das Modell einfacher geregelt werden. Für die Berechnung des Stellsignals u_R wird die Regelabweichung (Differenz zwischen Soll- und Istdruck) sowie die Ableitung des Solldrucks verwendet. Daraus ergibt sich für die Stellsignalberechnung

$$u_R = \dot{p}_d + k(p_d - p_m) \quad (3.12)$$

mit dem Einstellparameter k . Das Stellsignal u_R entspricht dabei dem Eingang des inversen Systems \dot{p} [10].

Das Modell aus Abbildung 3.4 wird um den modellbasierten Regler erweitert. Für die Berechnung des Stellsignals (siehe Gleichung (3.12)) wird die Ableitung des Solldrucks benötigt. Dafür wird ein Zustandsvariablenfilter (Festo Bibliothek) eingesetzt. Das vollständig modellierte Matlab Modell ist in Abbildung 3.7 zu sehen.

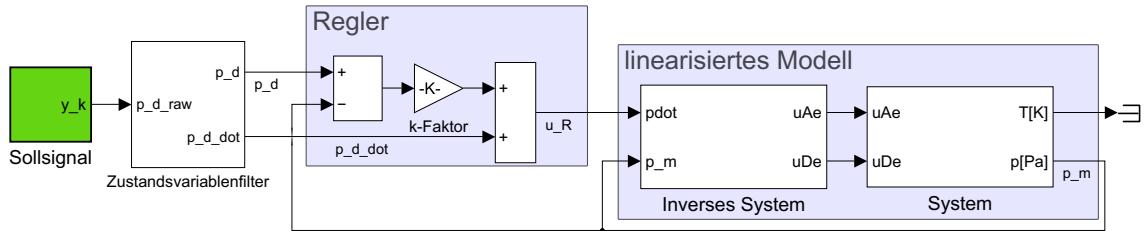


Abbildung 3.7: Matlab Modell reduziertes Gesamtsystem mit Druckregler

3.1.3 ILC-Ansatz

Aus Gründen der Vereinfachung wird im Folgenden ausschließlich der P-Type ILC-Ansatz in der parallelen Struktur verwendet. Für die Implementierung des ILC-Ansatzes wird der in Kapitel 2 erstellte Simulink-Block verwendet. Dieser kann problemlos parallel zum flachheitsbasierten Regler eingefügt werden. Der Eingang des reduzierten Systems ergibt sich somit durch die Addition des Stellsignals des Reglers gemäß Gleichung (3.12) und des Stellsignals des ILC-Ansatzes u_{ILC} :

$$\dot{p} = u_R + u_{ILC}. \quad (3.13)$$

In Abbildung 3.8 ist das um den ILC-Ansatz erweiterte Modell abgebildet. Zur besseren Verständlichkeit der Abbildung ist auf die entsprechenden Gleichungen und Abbildungen referenziert.

Der Verzögerungsparameter v im ILC Regelgesetz (2.20) muss im Vergleich zu den vorherigen Untersuchungen aus Kapitel 2 auf den Wert zwei erhöht werden, da im Ventilmodell eine Totzeit von einem Abtastschritt implementiert ist.

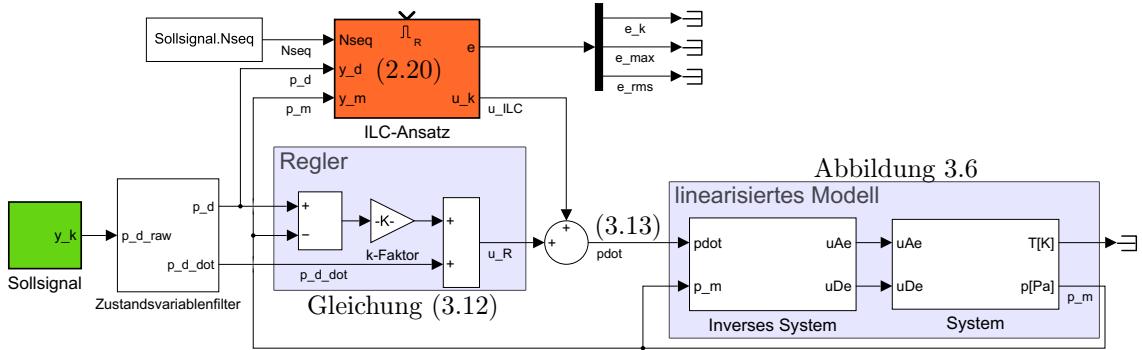


Abbildung 3.8: Matlab Modell reduziertes Gesamtsystem mit Druckregler und ILC-Ansatz

3.1.4 Ergebnisse

Für die Durchführung der Simulation wird ein ähnliches Sollsignal wie in Kapitel 2 verwendet. Das Sollsignal hat eine Periodendauer von 26 Sekunden und eine maximale Amplitude von 2 bar. Dabei bezieht sich der Druck immer auf den absoluten Druck. Zur besseren Darstellung der Diagramme wird der Druck in bar und nicht in Pascal angegeben.

Als ILC-Ansatz wird das P-Type ILC-Gesetz mit einem Verstärkungsfaktor von $k_p = 3 \text{ s}^{-1}$ (experimentell ermittelt) ohne Filterung verwendet. Der Verstärkungsfaktor für den flachheitsbasierten Regler beträgt $k = 12 \text{ s}^{-1}$ (ebenfalls experimentell ermittelt).

In Abbildung 3.9 ist das Simulationsergebnis für das Modell konstantes Volumen (Abbildung 3.8) abgebildet. Der Aufbau des Diagramms ist bereits aus Kapitel 2 bekannt. Zu beachten ist die logarithmische Darstellung der y-Achse im unteren Schaubild. Im oberen Diagramm ist kein Unterschied zwischen dem roten Sollsignal und den Istsignalen zu erkennen. Dies ist darauf zurückzuführen, dass der flachheitsbasierte Regler bereits eine äußerst geringe Regelabweichung erbringt.

Im mittleren Diagramm wird die Regelabweichung ohne den ILC-Anteil in blau dargestellt, da der ILC-Ansatz erst ab der zweiten Iteration aktiv wird. Die Regelabweichung nach bestimmter Anzahl von Iterationen ist in verschiedenen Farben dargestellt. Je mehr Iterationen durchgeführt werden, desto geringer wird die Regelabweichung. Nach 20 Iterationen ist keine Regelabweichung mehr erkennbar.

Im unteren Diagramm sind der maximale Regelfehler und der Effektivwert der Regelabweichung über die Anzahl der Iterationen abgebildet. Die Regelabweichung konvergiert nach 500 durchgeföhrten Iterationen auf einen maximalen Regelfehler von $e_{\max} = 1.1 \cdot 10^{-8} \text{ bar}$ und auf einen Effektivwert von $e_{\text{rms}} = 4.4 \cdot 10^{-10} \text{ bar}$.

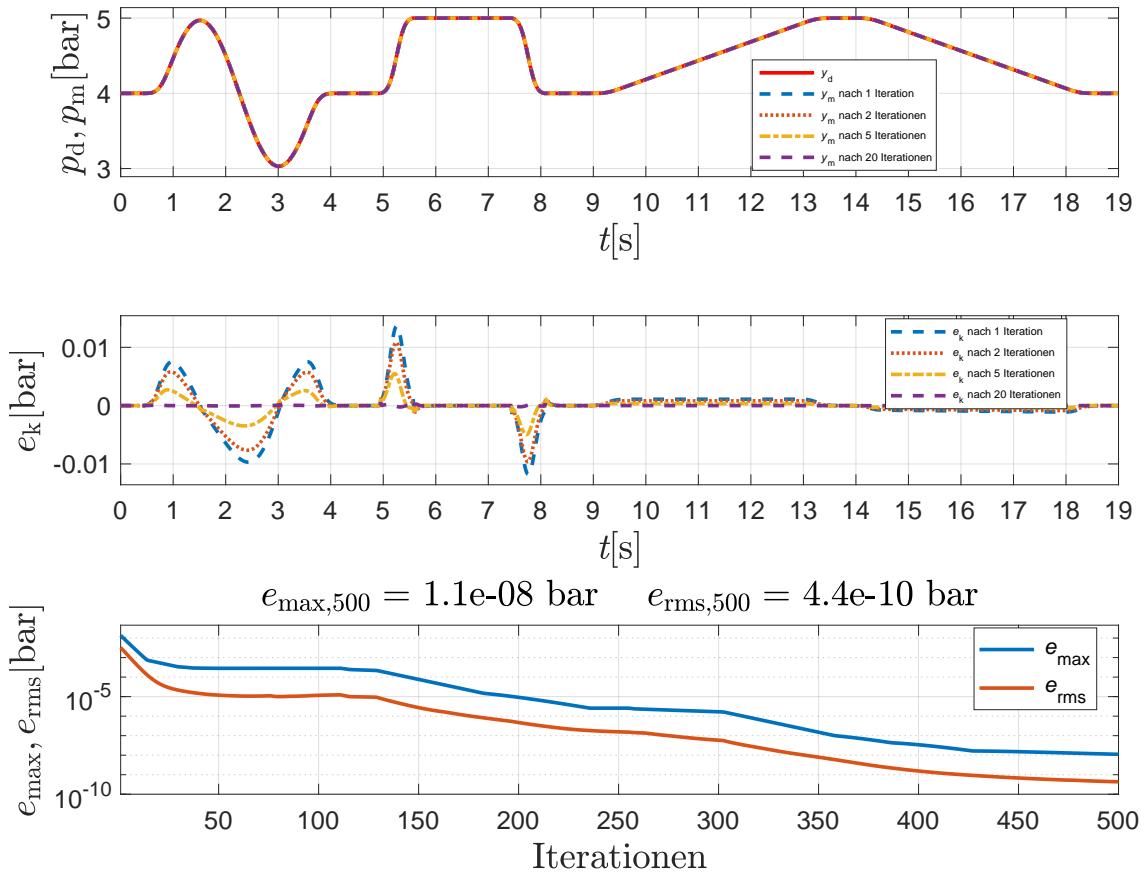


Abbildung 3.9: Simulationsergebnis Druckregelung konstantes Volumen

3.2 Reale Hardware

In diesem Abschnitt wird der Versuchsaufbau - Druckregelung an einem konstanten Volumen - mit realer Hardware vorgestellt. Außerdem werden die Anpassungen des ILC-Ansatzes für eine Echtzeitfähigkeit und die erzielten Ergebnisse präsentiert.

3.2.1 Aufbau

Die Abbildung 3.10 zeigt den experimentellen Aufbau und Abbildung 3.11 den Pneumatikplan für den Versuch mit konstantem Volumen. Ursprünglich war dieser Aufbau für eine andere Versuchsreihe vorgesehen, weshalb nicht alle Komponenten für diesen Versuch genutzt werden.

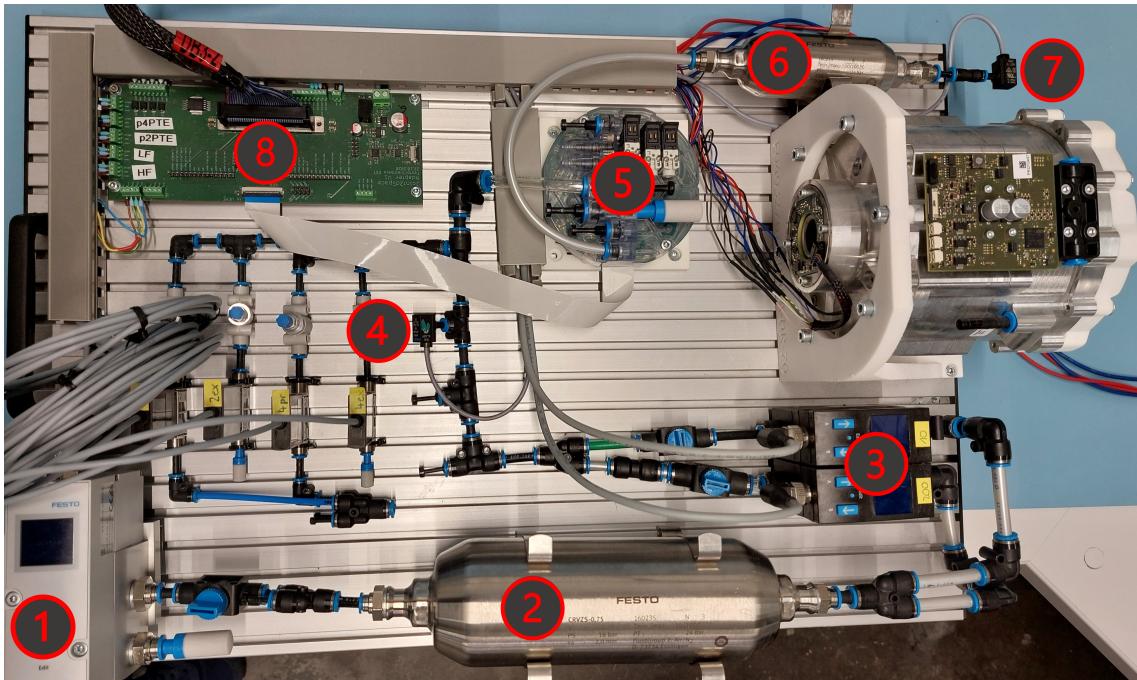


Abbildung 3.10: Versuchsaufbau konstantes Volumen

1. Druckregelventil
2. Druckluftbehälter
3. Durchflusssensoren
4. Drucksensor (Versorgungsdruck)
5. Ventilblock
6. zu regelnder Druckluftbehälter
7. Drucksensor (Sensor für Druckregelung)
8. Festo to dSpace Platine (Schnittstelle zwischen realer Hardware und dSpace)

Mithilfe des Druckregelventils (Position eins) wird der Versorgungsdruck des Systems präzise eingestellt. Um sicherzustellen, dass stets ausreichend Luft im System vorhanden ist, wird ein großvolumiger Druckluftbehälter (Position zwei) als Pufferspeicher eingesetzt. Die Durchflusssensoren (Position drei) ermöglichen die präzise Messung des Volumenstroms und bieten die Möglichkeit, zwischen zwei Messbereichen manuell umzuschalten. Der Drucksensor (Position vier) dient der kontinuierlichen Überwachung des Versorgungsdrucks. Für die Umsetzung dieses Versuchsaufbaus kommen

zwei Ventile des Ventilblocks (Position fünf) zum Einsatz. Ein Ventil wird verwendet, um den Druckbehälter zu befüllen, während das andere Ventil zum Entlüften des Druckbehälters dient. Der detaillierte Aufbau und die Funktionsweise des Ventilblocks sind in Kapitel 4 beschrieben. Der Drucksensor (Position sieben) erfasst den Druck im zu regelnden Druckluftbehälter. Die Platine (Position acht) stellt die Schnittstelle zwischen der realen Hardware und der Steuerung dar, wodurch Signale eingelesen und Aktoren gesteuert werden können. Als Steuerung wird eine MicroLabBox von der Firma dSpace verwendet. Dafür wird aus einem Matlab-Simulink-Modell Code für die Steuerung generiert. Anschließend lässt sich der Versuchsaufbau über das Programm ControlDesk steuern.

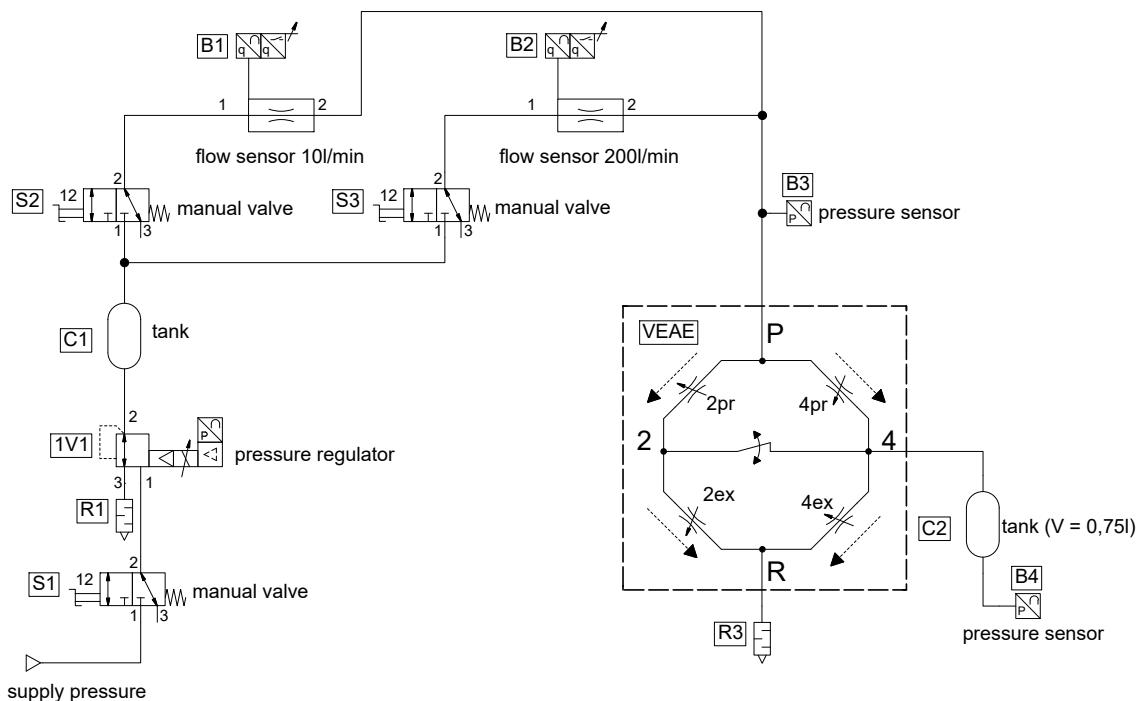


Abbildung 3.11: Pneumatikplan konstantes Volumen

3.2.2 Echtzeitfähigkeit ILC-Ansatz

Um sicherzustellen, dass der Simulink-Block in Echtzeit mit einer geringen Abtastzeit arbeiten kann, sind einige Code-Anpassungen erforderlich. Daher ist es notwendig, den Algorithmus aus Skript 2.3 anzupassen. Die Verwendung von Vektoren und Matrizen in den Berechnungen erfordert zu viel Zeit und verhindert die Realisierung kleiner Abtastschritte. Daher darf der Algorithmus nur einfache mathematische Operationen verwenden. Der angepasste Algorithmus für ein P-Type ILC-Ansatz ist in Skript 3.1 dargestellt. Die Initialisierung und die Definition der Variablen sind

nicht aufgeführt.

```

1 % Stellsignal extrahieren
2 u_k = Useq(k);
3 % Regelfehler berechnen
4 e_k = y_d - y_m;
5 % Interne Variablen übergeben
6 e_rms = e_rms_seq;
7 e_max = e_max_seq;
8 j = j_int;
9 % max Regelfehler berechnen
10 if abs(e_k) > e_max_tmp
11     e_max_tmp = abs(e_k);
12 end
13 % für rms Regelfehler berechnung
14 e_rms_tmp = e_rms_tmp + e_k^2;
15 % k_u bestimmen
16 if (k - v) >= 1
17     k_u = k - v;
18 else
19     k_u = Nseq - (v - k);
20 end
21 % neues Stellsignal berechnen
22 Useq(k_u) = Useq(k_u) + kp * e_k;
23 % filtern, wenn aktiviert
24 if filt
25     if (k_u - Nq_int) >= 1
26         k_filt = k_u - Nq_int;
27     else
28         k_filt = Nseq - (Nq_int - k_u);
29     end
30     if k_filt-Nq_int < 1
31         Useq_shift = circshift(Useq,Nq_int-k_filt+1);
32         Useq_gaus = Useq_shift(1:numel(q));
33     elseif k_filt+Nq_int > Nseq
34         Useq_shift = circshift(Useq,Nseq-(k_filt+Nq_int));
35         Useq_gaus = Useq_shift(Nseq-numel(q)+1:Nseq);
36     else
37         Useq_gaus = Useq(k_filt-Nq_int:k_filt+Nq_int);
38     end

```

```

39     Useq(k_filt) = q * Useq_gaus;
40 end
41 % Zähler erhöhen
42 k = k + 1;
43 % Zähler zurücksetzen, max und rms Fehler bestimmen
44 if k > Nseq
45     k = 1;
46     j_int = j_int + 1;
47     e_rms_seq = sqrt(e_rms_tmp/Nseq);
48     e_rms_tmp = 0;
49     e_max_seq = e_max_tmp;
50     e_max_tmp = 0;
51 end

```

Skript 3.1: ILC-Algorithmus - Echtzeitfähig

Für die Bestimmung des maximalen Regelfehlers wird anstatt der Matlab-Funktion `max()` ab Zeile 11 zu jedem Zeitpunkt überprüft, ob der aktuelle Regelfehler `e_k` größer ist als der bisherige maximale Regelfehler `e_max_tmp`. Wenn dies der Fall ist, wird die Variable `e_max_tmp` aktualisiert.

Bei der Berechnung des Effektivwerts der Regelabweichung `e_rms` wird in Zeile 15 zu jedem Zeitpunkt der quadratische Regelfehler aufsummiert. Am Ende des Zyklus wird in Zeile 48 der Effektivwert aus dieser Summe berechnet.

Die Berechnung des neuen Stellsignals erfolgt zu jedem Abtastschritt an der Position `k_u` (Zeile 23). Die Variable `k_u` ist um den Wert der Verzögerung `v` kleiner als der aktuelle Abtastschritt. Auf diese Weise wird die Verzögerung direkt bei der Berechnung des Stellsignals berücksichtigt.

Als Filter wird ein Moving-Gauss-Filter eingesetzt, dessen Länge (Parameter `Nq`) und Eckfrequenz (Parameter `fc`) über die Maske des Subsystems eingestellt werden können. Der Gaußfiltervektor `q` wird bei der Initialisierung mithilfe einer externen Funktion entsprechend den genannten Parametern gemäß Gleichung (2.30) berechnet. Bei jedem Abtastschritt an der Position `k_filt` wird in Zeile 40 die Filterung durchgeführt, indem der Vektor `q` mit dem entsprechenden Bereich von `Useq` multipliziert wird. Die Variable `k_filt` ist dabei um den Wert `Nq` kleiner als `k_u`.

Der Parameter `Nseq` wird nicht mehr über die Maske eingestellt, sondern als Eingang des Subsystems an den Algorithmus übergeben. Dadurch ist es möglich, zwischen unterschiedlichen Sollsignalen mit unterschiedlichen Periodenlängen zu wechseln.

Um die Anzahl der durchgeführten Iterationen nachzuverfolgen, steht am

Ausgang des Blocks eine zusätzliche Variable j zur Verfügung.

Alle Variablen werden beim Zurücksetzen des Subsystems auf ihre Anfangswerte gesetzt. Zudem wird der Filtervektor entsprechend der eingestellten Parameter neu berechnet. Daher werden Änderungen der Parameter erst wirksam, wenn der gesamte Block zurückgesetzt ist.

3.2.3 Ergebnisse

In diesem Abschnitt wird beispielhaft ein Messergebnis für den Versuchsaufbau *Druckregelung für ein konstantes Volumen an realer Hardware* vorgestellt. Das Sollsignal und die Reglerparameter sind identisch wie in Abschnitt 3.1.4. Dadurch können beide Ergebnisse miteinander verglichen werden.

Während der Durchführung verschiedener Messungen wurde festgestellt, dass das Stellsignal des ILC-Ansatzes nach einer großen Anzahl von Iterationen anfängt sich aufzuschwingen und instabil zu werden. Dieses Verhalten ist auf das Rauschen im Messsignal zurückzuführen. Dieses Rauschen wird vom ILC-Ansatz zusätzlich verstärkt. Um diesem Effekt entgegenzuwirken, ist eine Filterung gemäß Gleichung (2.30) im ILC Ansatz erforderlich. Die Verwendung von Filterparametern wie $Nq = 20$ und $f_c = 30 \text{ Hz}$ haben sich als wirksam erwiesen. Durch die Anwendung dieser Filterung wird das Rauschen effektiv reduziert und das Stellsignal des ILC-Ansatzes bleibt stabil.

Das Messergebnis ist in Abbildung 3.12 dargestellt. Im mittleren Diagramm wird erneut die Regelabweichung e_k dargestellt. Die blaue Kurve zeigt die Abweichung bei der Regelung ausschließlich mit dem flachheitsbasierten Regler. Mit zunehmender Anzahl an Iterationen nimmt die Regelabweichung kontinuierlich ab. Dieses Verhalten wird auch im unteren Diagramm deutlich sichtbar. Sowohl der maximale Regelfehler als auch der Effektivwert der Regelabweichung konvergieren nach 100 Iterationen zu den Werten $e_{\max} = 1.5 \cdot 10^{-3} \text{ bar}$ und $e_{\text{rms}} = 3.7 \cdot 10^{-4} \text{ bar}$.

Das Messsignal besitzt ein Messrauschen von ungefähr $2 \cdot 10^{-3} \text{ bar}$. Daher ist es nicht möglich, den maximalen Regelfehler und damit auch den Effektivwert weiter zu reduzieren. Das Messrauschen stellt somit den limitierenden Faktor dar und nicht die bereits implementierte Filterung.

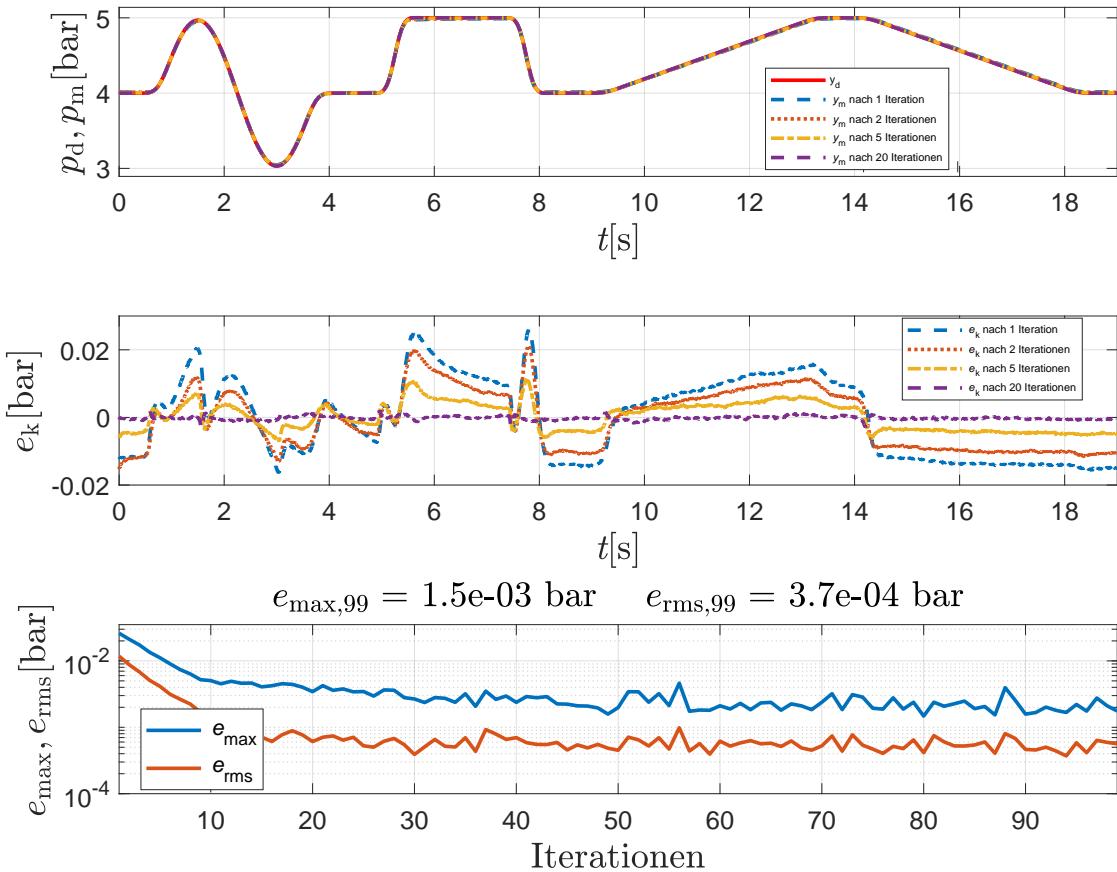


Abbildung 3.12: Messergebnis Versuchsaufbau *konstantes Volumen*

3.3 Zusammenfassung

In diesem Kapitel wurde der ILC-Ansatz erstmals an einem nichtlinearen System sowohl simulativ als auch an realer Hardware getestet. Dafür wurde ein Versuchsaufbau für eine Druckregelung eines konstanten Volumens in Matlab modelliert und mit realer Hardware aufgebaut. Die Simulationsergebnisse als auch die Messergebnisse fallen sehr positiv aus. Durch den Einsatz eines ILC-Ansatzes, welcher parallel zu einem modellbasierten Regler eingesetzt wird, verbessert sich die Regelgüte bei der Durchführung vieler Iterationen signifikant. Bei der realen Hardware wird durch den Einsatz eines Filters das Messrauschen unterdrückt und das System bleibt in einem stabilen Zustand. Der limitierende Faktor für die Genauigkeit ist dabei das Messrauschen.

4 Versuchsaufbau pJoint

In diesem Kapitel wird der Aufbau und die Funktionsweise eines pneumatischen Gelenks (engl. pJoint) eingehend erläutert. Des Weiteren wird der ILC-Ansatz sowohl simulativ als auch an einem realen Aufbau mit einem pneumatischen Gelenk getestet und analysiert. Dabei kommt eine Reglerkaskade zum Einsatz, bei der ein modellbasierter Regler unterlagert ist.

Um das Verständnis der Regelaufgabe für den Versuchsaufbau zu erleichtern, ist in Abbildung 4.1 das dazugehörige Blockschaltbild abgebildet.

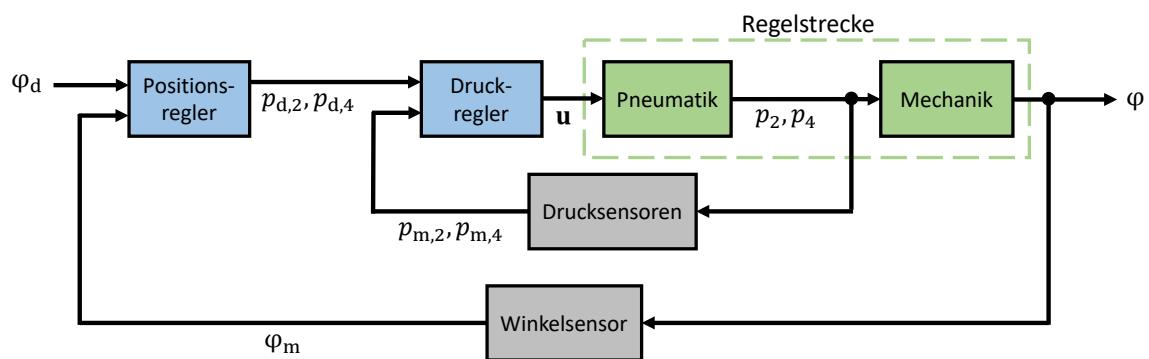


Abbildung 4.1: Blockschaltbild Reglerkaskade pneumatisches Gelenk

4.1 Aufbau und Funktionsweise

Die Abbildung 4.2 zeigt den schematischen Aufbau eines pneumatischen Gelenks. Jedes Gelenk setzt sich aus einem Ventilblock und einem Schwenkantrieb (in grau dargestellt) zusammen. Der doppeltwirkende Schwenkantrieb besteht aus zwei Druckkammern, die über den Ventilblock angesteuert werden. Im Gegensatz zu elektrischen Gelenken wird hier kein zusätzliches Getriebe benötigt. Durch eine Druckdifferenz Δp zwischen Kammer 4 und Kammer 2 entsteht ein Drehmoment am keilförmigen Kolben, der in der Abbildung 4.2 in Schwarz dargestellt ist. Dadurch wird eine Drehbewegung des Gelenks erzeugt. Es ist möglich, diese Bewegung sowohl im Uhrzeigersinn als auch gegen den Uhrzeigersinn auszuführen. Je größer die Druckdifferenz Δp ist, desto größer ist das erzeugte Drehmoment und die resultierende Winkelbeschleunigung $\ddot{\varphi}$. Der Ventilblock besteht aus vier Piezoventilen. Für jede Kammer gibt es jeweils zwei Ventile, eins für das Belüften (aerate) und eins für das Entlüften (deaerate). Durch diese kann der Druck in den Kammern unabhängig voneinander eingestellt

und geregelt werden.

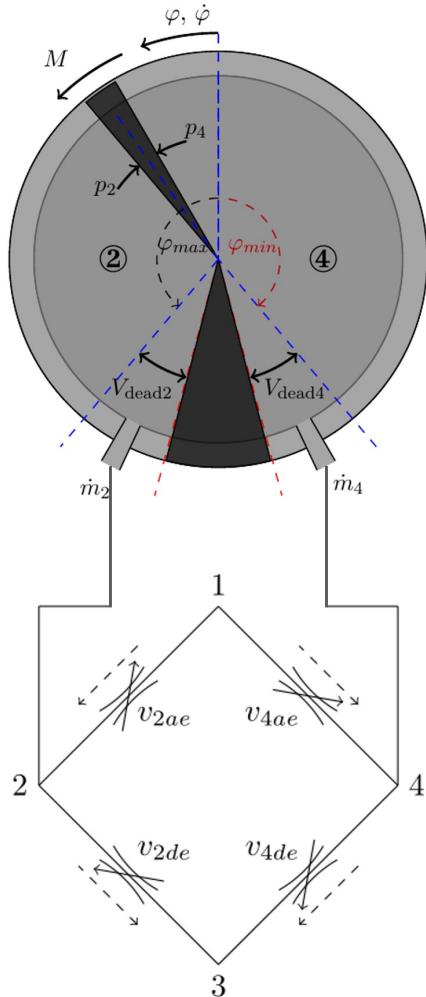


Abbildung 4.2: Schematischer Aufbau eines pneumatischen Gelenks [12]

Die Ventile bestehen, wie in Abbildung 4.3 zu sehen, aus einem Bieger aus Piezokeramik und einem passiv leitfähigem Träger. Es werden die piezoelektrischen Eigenschaften ausgenutzt. Durch Anlegen einer Spannung am Bieger, entsteht eine mechanische Verformung, weshalb sich der Bieger nach oben biegt. Beim Abschalten der Spannung behält der Bieger seine Position bei, ähnliches Verhalten wie bei einem Kondensator. Es wird also lediglich Energie zum Wechseln der Position benötigt. Die maximal angelegte Spannung liegt bei 310 Volt, welche durch die Leiterplatte im Ventilblock erzeugt wird. Piezoventile gehören zu den Proportionalventilen, das heißt der Massenfluss ist proportional zum Hub.

An der unteren Seite des Trägers befindet sich ein kleiner Magnet (in der Abbildung 4.3 nicht dargestellt), welcher zum messen der Öffnung verwendet wird. Dafür

wird ein Hall Sensor eingesetzt, welcher abhängig von der Stärke des Magnetfeldes eine Spannung liefert. Dadurch kann der Durchfluss abhängig von der Hall-Sensor-Spannung bestimmt und geregelt werden.

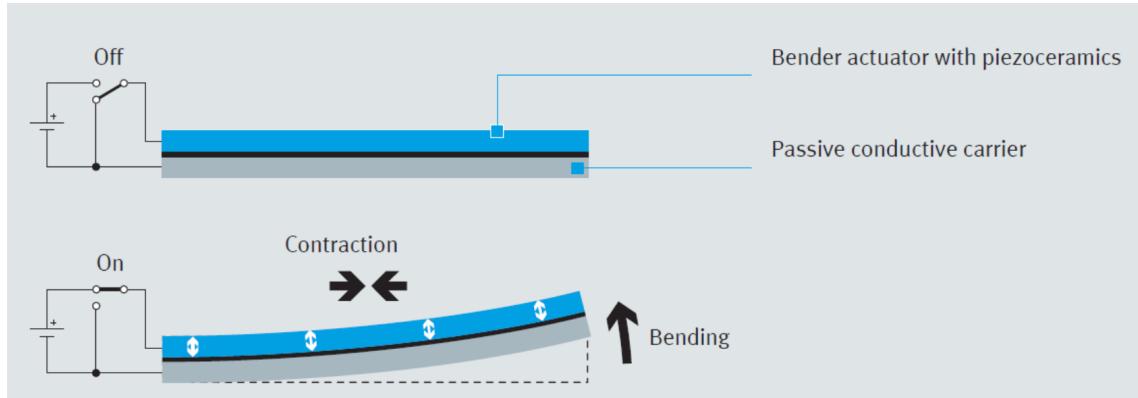


Abbildung 4.3: Funktionsweise Piezoventil [10]

4.2 Simulation

Für die Simulation wird das Matlab-Simulink-Modell verwendet, welches in der Bachelorarbeit von Sharma [10] entwickelt wurde. Dieses Modell umfasst die vollständige Modellierung eines pneumatischen Gelenks.

4.2.1 Aufbau Gesamtsystem

In Abbildung 4.4 ist das Gesamtsystem in Matlab-Simulink abgebildet.

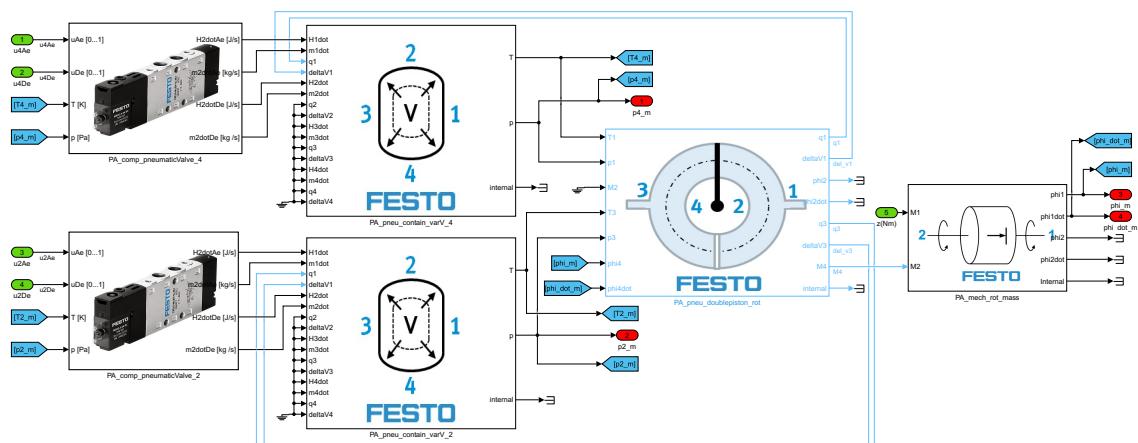


Abbildung 4.4: Matlab Modell Gesamtsystem pneumatisches Gelenk

Es enthält bereits bekannte Blöcke für die Piezoventile und das pneumatische Volumen, die im vorherigen Kapitel vorgestellt wurden. Aufgrund der Struktur des Swivel Drive, mit seinen beiden Kammern und vier Ventilen, werden jeweils zwei dieser Blöcke benötigt. Die beiden hinteren Blöcke berechnen das resultierende Drehmoment, die Position und die Geschwindigkeit anhand der Druck- und Temperaturwerte in den Kammern, sowie den mechanischen Parametern wie Reibung und Massenträgheitsmoment der Antriebswelle. Da sich das Volumen in den Kammern verändert, wird die Volumenänderung an den Block für das pneumatische Volumen zurückgeführt. Das Gesamtsystem verfügt über Eingänge für die Steuersignale der vier Piezoventile und über einen Eingang, um ein Störmoment auf den Antrieb zu geben. Die Ausgänge des Systems liefern die Druckwerte in den einzelnen Kammern sowie die aktuelle Position und Geschwindigkeit des Gelenks. Alle Größen sind in ihren Einheitsgrößen (SI-Basiseinheiten) angegeben (Position [rad], Geschwindigkeit [rad/s], Druck [Pa]).

4.2.2 Reglerkaskade

Für die Regelung des pneumatischen Gelenks wird eine Kaskadenregelung verwendet. Die untergeordneten Regler sind modellbasierte Druckregler, die aus der Bachelorarbeit von Sharma [10] entnommen wurden. Diese Regler dienen zur Regelung des Drucks in den jeweiligen Kammern. Als übergeordneter Positionsregler wird ein PI-Regler eingesetzt.

In Abbildung 4.5 sind die unterlagerten Druckregler dargestellt.

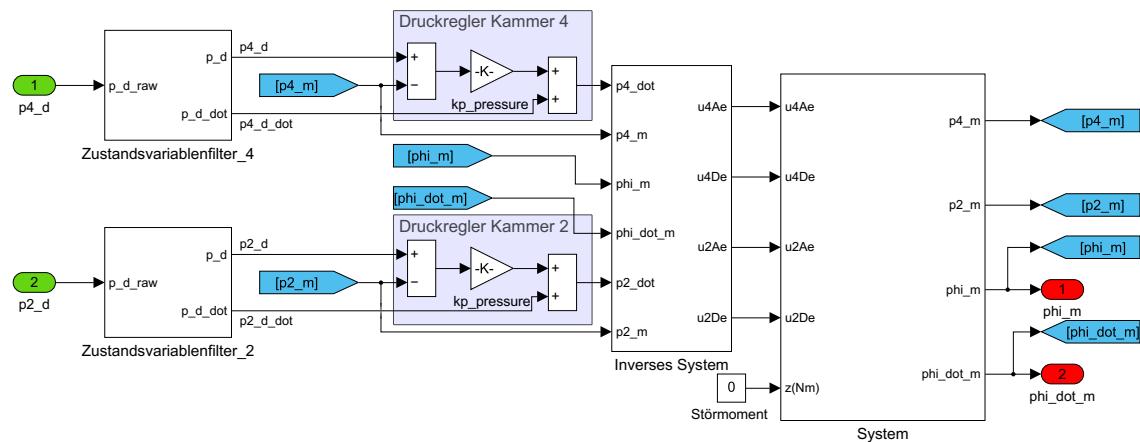


Abbildung 4.5: Matlab Modell pneumatisches Gelenk unterlagerter Druckregler

Der Aufbau dieser Druckregler wurde bereits im vorherigen Kapitel 3 erläutert. In diesem Aufbau werden jedoch zwei Regler benötigt, um den Druck in jeder Kammer

zu regulieren. Als Eingang bedarf es den Solldruck für jede Kammer. Mithilfe des Zustandsvariablenfilters wird die Ableitung des Drucks berechnet. Durch das Inverse System wird das Steuersignal für die entsprechenden Ventile erzeugt und auf das System übertragen. Anschließend werden die aktuellen Druckwerte in den Kammern an den Regler zurückgeführt. Die Position und Geschwindigkeit werden dabei nach außen an den überlagerten Positionsregler weitergegeben.

In Abbildung 4.6 ist der überlagerte Positionsregler abgebildet.

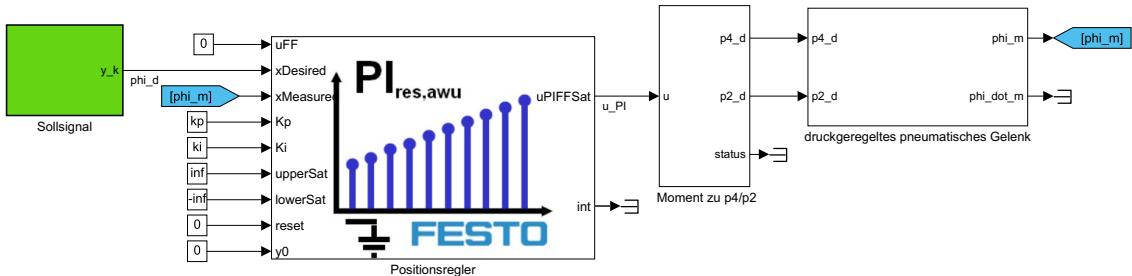


Abbildung 4.6: Matlab Modell pneumatisches Gelenk überlagerter Positionsregler

Als Positionsregler wird ein PI-Regler in paralleler Struktur verwendet, der die Sollposition und die aktuelle Position erhält. Das Stellsignal des Reglers entspricht dem Drehmoment an der Welle des Antriebes. Dieses Drehmoment wird mithilfe einer Matlab-Funktion innerhalb des Blocks *Moment zu p2/p4* in Solldrücke für die jeweiligen Kammern umgewandelt. Dabei wird das Drehmoment mithilfe der Parameter, mittlerer Radius und effektive Fläche des beweglichen Kolbens, in eine Druckdifferenz umgerechnet. Der Mitteldruck ist dabei auf einen Wert von 4.5 bar absolut festgelegt. Die Solldrücke für Kammer vier und Kammer zwei werden an das pneumatische Gelenk mit Druckregelung übergeben. Die aktuelle Position wird an den Regler zurückgeführt.

4.2.3 ILC-Ansatz

Bei einer Kaskadenregelung muss der ILC-Ansatz parallel zum äußersten Regler implementiert werden. Dies ist erforderlich, da das Sollsignal bei den untergeordneten Reglern über die Zyklen hinweg minimale Änderungen aufweisen kann und somit die Voraussetzung für den Einsatz des ILC-Konzepts nicht erfüllt ist. Außerdem hat das System ein integrierendes Verhalten und zwischen der Druckdifferenz Δp und der Position φ besteht keine eindeutige Zuordnung. Aus diesem Grund kann der ILC-Ansatz nicht als alleinstehender Regler funktionieren und wird deshalb

parallel zum übergeordneten Positionsregler (PI-Regler) eingesetzt. Hierbei kommt der angepasste ILC-Block aus Abschnitt 3.2.2 zum Einsatz.

In Abbildung 4.7 ist das um den ILC-Ansatz erweiterte Matlab-Modell zu sehen. Das Stellsignal des ILC-Ansatzes wird zum Stellsignal des PI-Reglers addiert, um die Druckdifferenz in beiden Kammern gemeinsam zu regeln. Das übrige Modell bleibt unverändert.

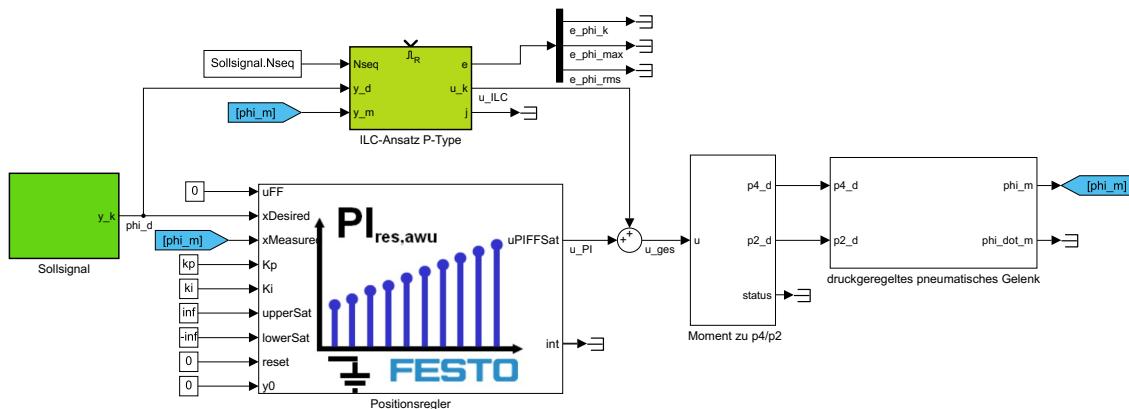


Abbildung 4.7: Matlab Modell Kaskadenregelung pneumatisches Gelenk mit ILC-Ansatz

4.2.4 Ergebnisse

In diesem Abschnitt werden die Ergebnisse der Simulationen erläutert. Wie in Kapitel 3 genannt, kommt nur der P-Type ILC-Ansatz zum Einsatz. Für die Simulation werden die Parameter für den größten Antrieb im Roboter verwendet. Die Regelparameter wurden experimentell bestimmt. Für die Parameter des PI-Reglers werden $k_p = 1.2 \frac{\text{Nm}}{\text{s}^\circ}$ und $k_i = 0.3 \frac{\text{Nm}}{\text{s}^\circ}$ ermittelt. Für den Verstärkungsfaktor im ILC-Gesetz wird $kp = 0.8 \frac{\text{Nm}}{\text{s}^\circ}$ gewählt. Der Verzögerungsfaktor v im ILC-Ansatz muss im Vergleich zum vorherigen Versuch (Kapitel 3) um eins erhöht werden. Das liegt an den Zustandsvariablenfilter, um die Ableitung des Druckes zu bestimmen (siehe Abbildung 4.5). In dieser Simulation wird die Filterung im ILC-Ansatz aktiviert, um die Stabilität des Systems zu gewährleisten. Dies ist notwendig, da der PI-Regler und der ILC-Ansatz teilweise gegeneinander wirken können und das System bei ungefiltertem Signal zu Schwingungen neigen kann. Eine Filtergröße von 13 mit einer Grenzfrequenz von 15 Hz wird verwendet. Die Simulationsergebnisse sind in Abbildung 4.8 dargestellt.

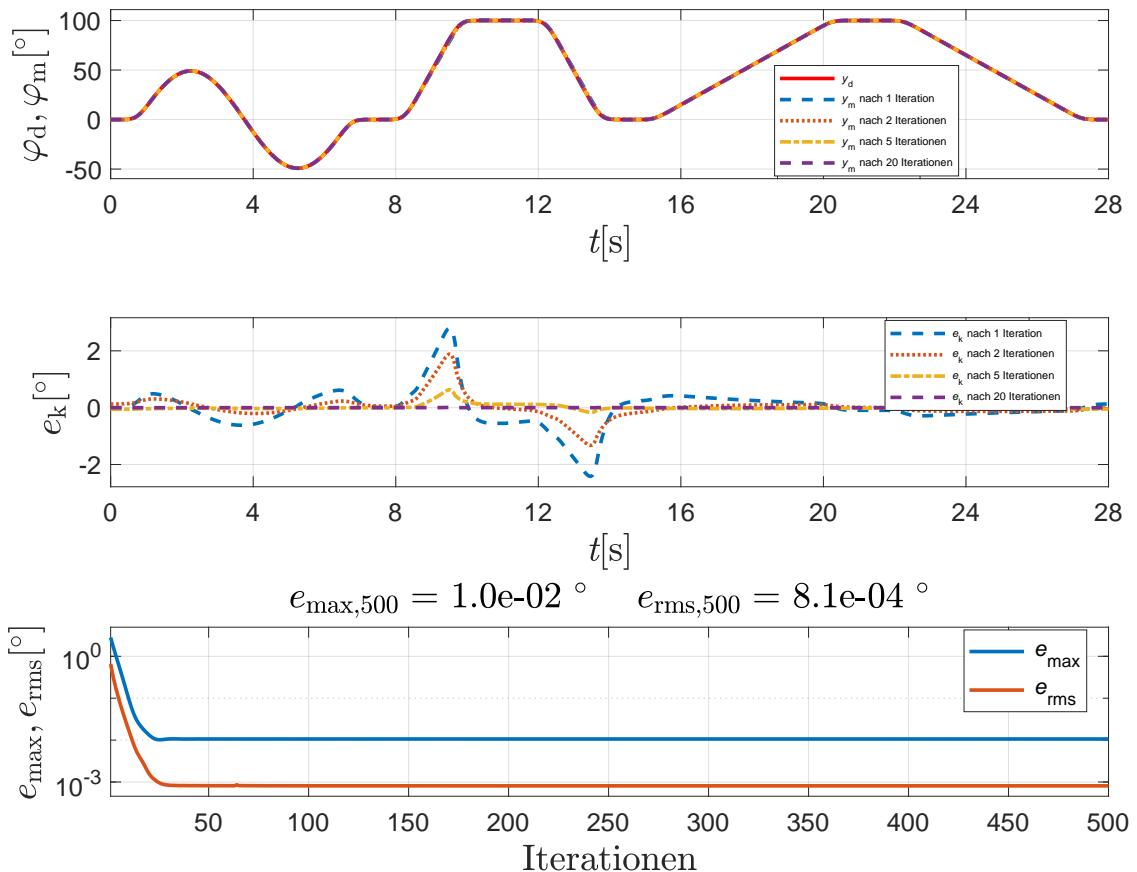


Abbildung 4.8: Simulationsergebnis Positionsregelung pneumatisches Gelenk

Im oberen Diagramm ist die Sollposition und die Istposition in Grad nach einer bestimmten Anzahl von Iterationen dargestellt. Die zugehörige Regelabweichung ist im mittleren Diagramm zu sehen. Durch den Einsatz des PI-Reglers liegt der Effektivwert der Regelabweichung bereits knapp unter einem Grad. Durch den ILC-Ansatz konvergiert die Regelabweichung nach 25 Iterationen auf Werte von $e_{\max} = 1.0 \cdot 10^{-2}^\circ$ und $e_{\text{rms}} = 8.1 \cdot 10^{-4}^\circ$.

Die Konvergenzgeschwindigkeit wird sowohl durch den k_p -Faktor des ILC-Gesetzes als auch durch die Aggressivität des PI-Reglers beeinflusst. Je aggressiver der Regler ist, desto mehr beeinflusst er den ILC-Ansatz und desto schneller wird das System instabil.

Der stationäre Endwert der Regelabweichung wird durch die Filterung im ILC-Ansatz begrenzt. Daher muss ein Kompromiss zwischen Filterung und Stabilität gefunden werden.

4.3 Reale Hardware

In diesem Abschnitt wird der Versuchsaufbau zur Positionsregelung an einem pneumatischen Gelenk sowie die erzielten Ergebnisse mit dem ILC-Ansatz erläutert. Der Aufbau wurde mit verschiedenen Antrieben getestet. Anhand des größten Antriebes (ROPA-DR-70) sind beispielhaft der Aufbau und die Messergebnisse aufgezeigt. Die Zahl 70 steht dabei für das maximale Drehmoment in Nm.

4.3.1 Aufbau

Es wird derselbe Versuchsaufbau wie in Kapitel 3 verwendet, allerdings wird der Schwenkantrieb und nicht der Druckluftbehälter an den Ventilblock angeschlossen. Die Abbildung 4.9 zeigt den experimentellen Aufbau und Abbildung 4.10 den Pneumatikplan für den Versuch mit einem pneumatischen Gelenk.

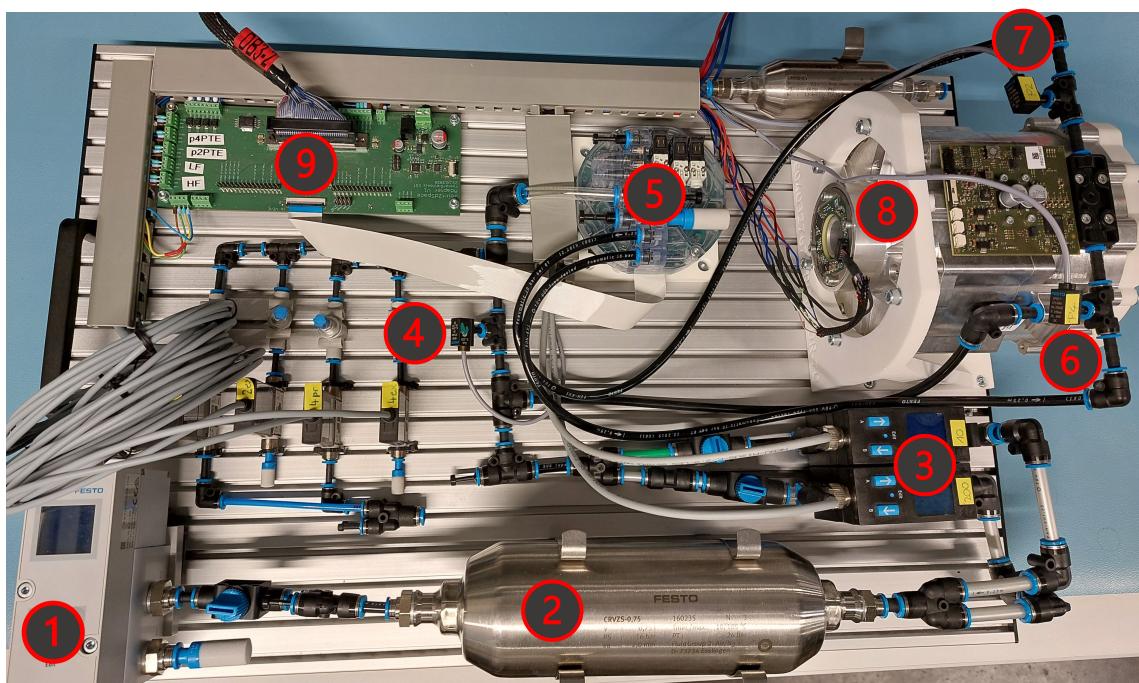


Abbildung 4.9: Versuchsaufbau pneumatisches Gelenk

1. Druckregelventil
2. Druckluftbehälter
3. Durchflusssensoren

4. Drucksensor (Versorgungsdruck)
5. Ventilblock
6. Drucksensor (Kammer vier)
7. Drucksensor (Kammer zwei)
8. Swivel Drive ROPA-DR-70
9. Festo to dSpace Platine (Schnittstelle zwischen realer Hardware und dSpace)

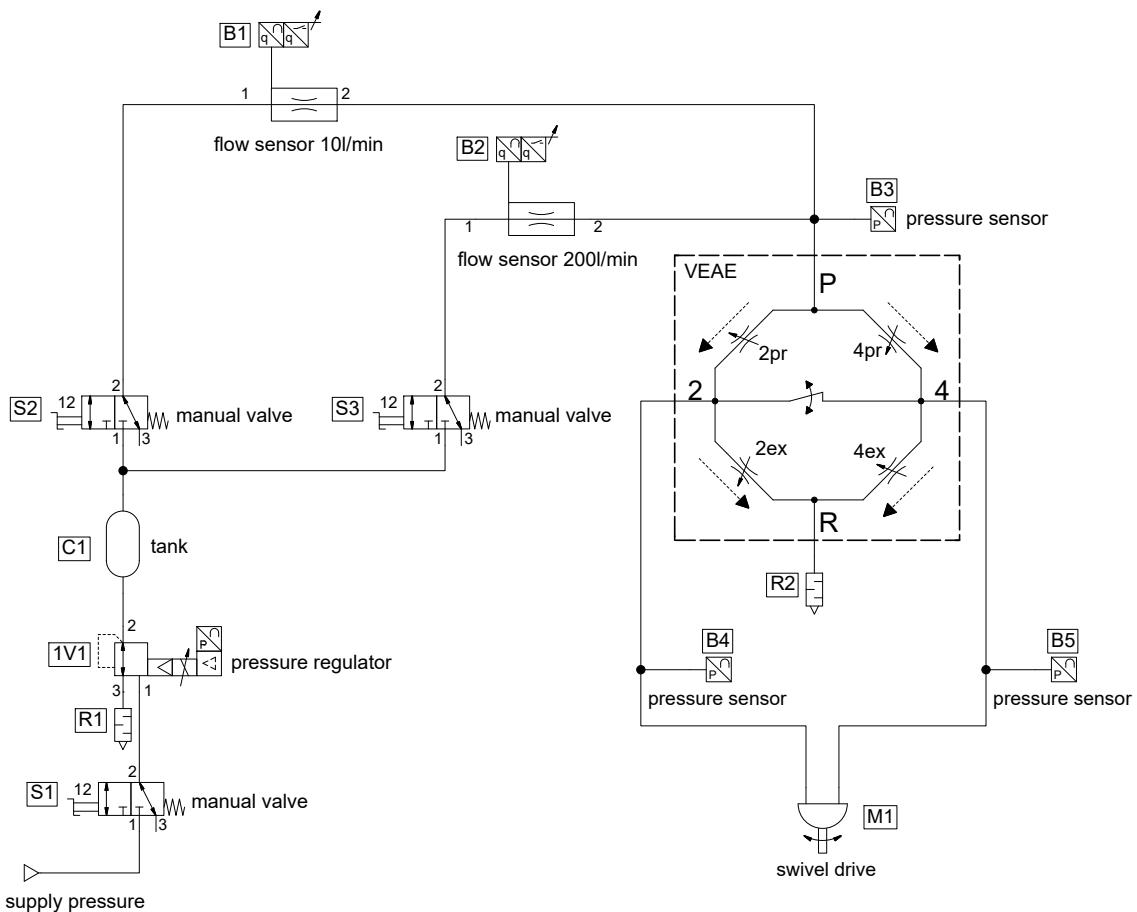


Abbildung 4.10: Pneumatikplan pneumatisches Gelenk

Die Funktionen der Bauteile Position 1-5 und 9 wurden bereits in Abschnitt 3.2.1 beschrieben. Für diesen Versuchsaufbau ist ein zusätzlicher Drucksensor erforderlich, um den aktuellen Druck in beiden Kammern zu messen (Position 6 und 7). Der Swivel Drive (Position 8) ist mit einem integrierten Encoder ausgestattet, der über eine 10-adrige Leitung mit der Hauptplatine verbunden ist. Dadurch kann die aktuelle Position abgerufen werden.

4.3.2 Ergebnisse

In diesem Abschnitt werden die Ergebnisse des Versuchsaufbaus an einem pneumatischen Gelenk erläutert. Dabei werden zwei unterschiedliche flachheitsbasierte Regler als unterlagerte Regler eingesetzt und miteinander verglichen. Einer der Regler wurde in Abschnitt 4.2.2 erläutert. Als zweiter Regler wird der modellbasierte Regler aus der Festo eigenen Simulink-Bibliothek für Robotik verwendet, der zusätzliche Massenflussbeobachter enthält.

Die Regelparameter wurden für beide Regler gleich eingestellt. Für den überlagerten Positionsregler wurde ein P-Anteil von $2.4 \frac{\text{Nm}}{\circ}$ und ein I-Anteil von $1.2 \frac{\text{Nm}}{\text{s} \circ}$ gewählt. Als k_p -Wert für das ILC-Gesetz wurde der Wert $1.2 \frac{\text{Nm}}{\circ}$ verwendet. Die Regelparameter für den PI-Regler wurden experimentell bestimmt. Es besteht die Möglichkeit, dass sie noch besser gewählt werden könnten, aber für diese Untersuchungen reicht die Performance des Reglers aus.

Die Filterung im ILC-Ansatz muss aktiviert sein, um Schwingungen und Stellgrößeninstabilitäten zu vermeiden. Als Filtergröße wird $N_q = 20$ und als Grenzfrequenz $f_c = 5 \text{ Hz}$ gewählt.

In Abbildung 4.11 und Abbildung 4.12 sind die Messergebnisse dargestellt. In beiden Abbildungen ist deutlich zu erkennen, dass die Regelperformance durch den ILC-Ansatz über mehrere Iterationen signifikant verbessert wird. Die Regelabweichung wird geringer und die Isttrajektorie nähert sich immer weiter der Solltrajektorie an. Besonders herausfordernd ist der Übergang zwischen Haftreibung und Gleitreibung, bei dem der Regler größere Schwierigkeiten hat. Durch den ILC-Ansatz wird dieses Verhalten kompensiert, indem beim Losfahren kurzzeitig ein großer Druckunterschied in den Kammern erzeugt wird, was zu einem erhöhten Drehmoment führt. Dadurch kann die Haftreibung effektiver überwunden werden und die Regelperformance verbessert sich.

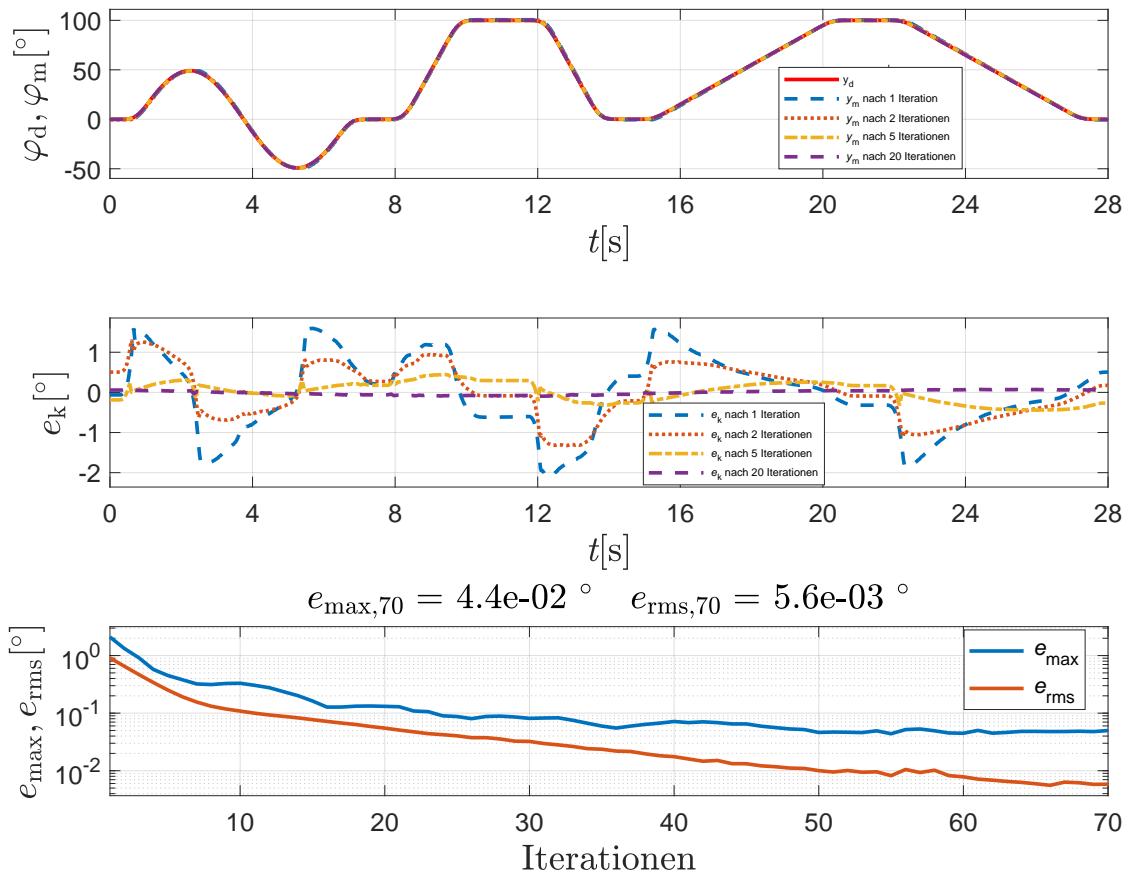


Abbildung 4.11: Messergebnis Versuchsaufbau pneumatisches Gelenk, unterlagelter Druckregler (Abschnitt 4.2.2) mit überlagertem Positionsregler und ILC-Ansatz

Nach 70 durchgeführten Iterationen konvergiert die Regelabweichung mit dem Regler aus Abschnitt 4.2.2 auf Werte von $e_{\max} = 4.4 \cdot 10^{-2} \text{ }^\circ$ und $e_{\text{rms}} = 5.6 \cdot 10^{-3} \text{ }^\circ$. Die Regelperformance mit diesem Regler ist besser als die mit dem Regler aus der Festo-Bibliothek. Aufgrund des Massenflussbeobachters im Festo Regler arbeitet der ILC-Ansatz etwas stärker gegen den Regler, wodurch die Regelperformance etwas schlechter ausfällt.

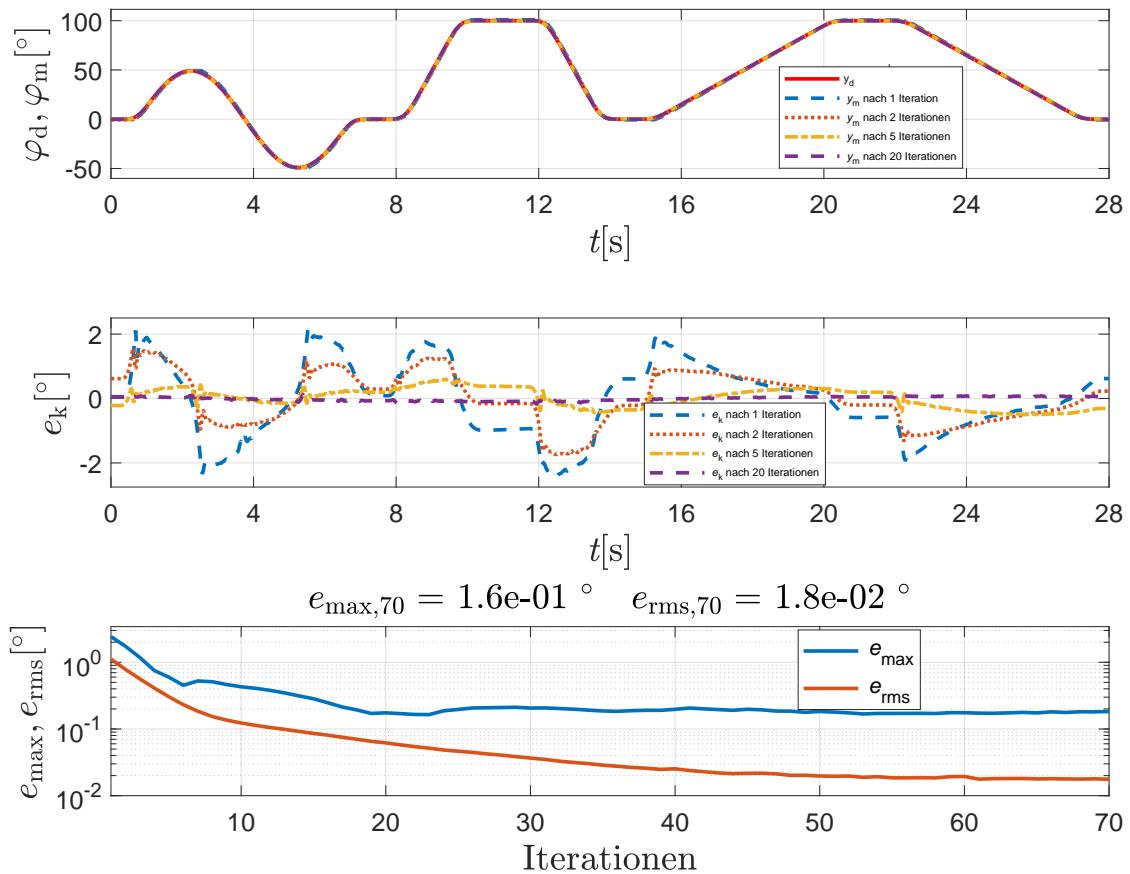


Abbildung 4.12: Messergebnis Versuchsaufbau pneumatisches Gelenk, unterlagelter Druckregler (Festo Bibliothek) mit überlagertem Positionsregler und ILC-Ansatz

Es ist anzumerken, dass bei beiden Reglern die maximale Regelabweichung nach 70 Iterationen beim Losfahren in der Sinusbewegung am größten ist (siehe Abbildung 4.13).

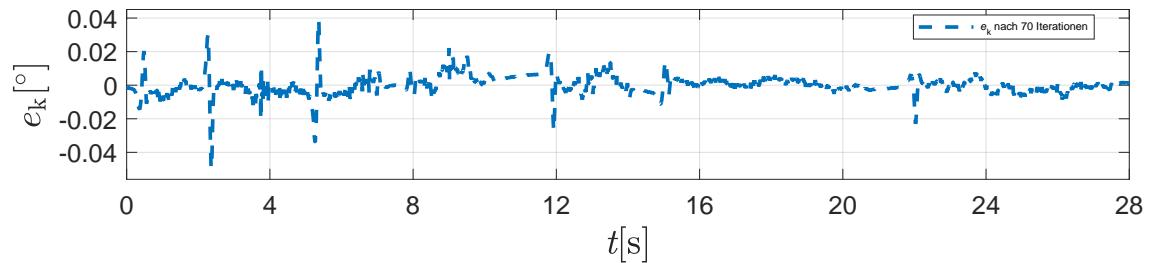


Abbildung 4.13: Messergebnis Versuchsaufbau pneumatisches Gelenk Regelabweichung nach 70 Iterationen

In Anhang B sind die Messergebnisse für die kleineren Antriebe (ROPA-DR-10 und den ROPA-DR-5) aufgeführt. Dabei wurde festgestellt, dass bei diesen Antrie-

ben bessere Ergebnisse mit dem ILC-Ansatz erzielt wurden, wenn der I-Anteil des Positionsreglers auf null gesetzt wurde.

4.4 Zusammenfassung

Der ILC-Ansatz wurde sowohl simulativ als auch an realer Hardware in Verbindung mit einer Reglerkaskade für das pneumatische Gelenk untersucht. Dabei wurden verschiedene modellbasierte Regler in Kombination mit dem ILC-Ansatz getestet. Sowohl die Simulationsergebnisse als auch die Messergebnisse an realer Hardware zeigen, dass durch den parallelen Einsatz eines ILC-Algorithmus zum überlagerten Regler, die Regelperformance bei der Durchführung vieler Iterationen deutlich verbessert wird. Die Verwendung einer Filterung im ILC-Gesetz gewährleistet die Stabilität des Systems und führt zu einer Konvergenz auf sehr geringe Regelabweichungen. Dabei ist zu beachten, dass die Filterung den limitierenden Faktor für die Genauigkeit darstellt. Ohne Filterung wird das System jedoch instabil und die Regelabweichung divergiert. Bei den kleineren Antrieben werden bessere Ergebnisse erzielt, wenn der I-Anteil des Positionsreglers auf null gesetzt wird.

5 Fazit und Ausblick

Die vorliegende Bachelorarbeit hat erfolgreich gezeigt, dass der Einsatz von Iterative Learning Control (ILC) die Regelperformance bei wiederholenden Bewegungen über die Zeit signifikant verbessert, sowohl bei einfachen linearen als auch bei komplexen nichtlinearen Systemen. Durch das Lernen aus vorangegangenen Zyklen kann das Stellsignal für den nächsten Zyklus angepasst werden, was zu einer verbesserten Regelgüte führt. Dabei ist die optimale Wahl der Regelparameter wie Verstärkungsfaktor und Verzögerung des Systems von großer Bedeutung.

Der größte Vorteil des ILC-Algorithmus besteht darin, dass er parallel zu jedem bestehenden Regler eingesetzt werden kann. Eine Voraussetzung dafür ist, dass das Sollsignal für jeden Zyklus identisch sein muss, weshalb der ILC-Ansatz in einer Reglerkaskade immer parallel zum überlagerten Regler Anwendung findet. Je nach Struktur der Regelung kann es vorkommen, dass der ILC-Ansatz dem eigentlichen Regler entgegenwirkt, was zu Instabilität führen kann.

Durch den vereinfachten Versuchsaufbau an einem konstanten Volumen konnte der ILC-Algorithmus echtzeitfähig gemacht und problemlos für zukünftige Versuchsaufbauten verwendet werden. Dadurch kann der Algorithmus mit geringem Aufwand in bestehende Regelstrukturen integriert und untersucht werden.

Die akausale Filterung im ILC-Algorithmus ermöglicht eine Filterung des Stellsignals ohne Phasenverschiebung. Dies ist besonders wichtig bei Messrauschen der Regelgröße und bei gleichzeitigem Einsatz mit einem bestehenden Regler.

Um den ILC-Ansatz für den gesamten Roboter nutzbar zu machen, sollten Erweiterungen hinzugefügt werden, wie beispielsweise eine Stellgrößenbeschränkung. Zudem muss das Verhalten untersucht werden, wenn der ILC-Algorithmus gleichzeitig für mehrere Achsen eingesetzt wird. Bei positiven Ergebnissen etwaiger Tests steht einer dauerhaften Integration des ILC-Algorithmus in die Regelstruktur für wiederholende Bewegungen nichts im Wege.

A Literaturverzeichnis

- [1] M. Böck, T. Glück, A. Kugi und A. Steinböck. „Fortgeschrittene Methoden der nichtlinearen Regelung“. In: *Vorlesungsskript, Technische Universität Wien* (2022).
- [2] Technische Universität Kaiserslauter. *Forschungsbericht 2000*. URL: <https://www.eit.uni-kl.de/pandit/haupt/forschung/jahresbericht00.htm> (besucht am 25.04.2023).
- [3] D. Andres, Heiko Hengen und Madhukar Pandit. „Optimierend iterativ lernende Regelungen (Optimizing Iterative Learning Control)“. In: 50.3 (2002), S. 112. DOI: doi:10.1524/auto.2002.50.3.112. URL: <https://doi.org/10.1524/auto.2002.50.3.112>.
- [4] Bianca Hoegel. *Spektralradius*. 12. Feb. 2023. URL: <https://www.biancahoegel.de/mathe/analysis/spektralradius.html> (besucht am 24.04.2022).
- [5] Bianca Hoegel. *Spektralnorm*. 13. Aug. 2018. URL: <https://www.biancahoegel.de/mathe/norm/spektralnorm.html> (besucht am 24.04.2022).
- [6] The MathWorks Inc. *MATLAB version: 9.10.0 (R2021a) Update 5*. Natick, Massachusetts, United States, 2021. URL: <https://www.mathworks.com>.
- [7] Adler Fonseca de Castro und Leonardo Antônio Borges Tôrres. „Iterative learning control applied to a recently proposed mechanical ventilator topology“. In: *IFAC-PapersOnLine* 52.1 (2019), S. 154–159.
- [8] Prof. Dr.-Ing Jürgen Baur. „Manuskript Regelungstechnik Einführung“. In: *Vorlesungsskript, Hochschule Aalen* (Wintersemester 2016/2017).
- [9] Michael Zeitz. „Differenzielle Flachheit: Eine nützliche Methodik auch für lineare SISO-Systeme Differential Flatness: A Useful Method also for Linear SISO Systems“. In: 58.1 (2010), S. 5–13. URL: <https://doi.org/10.1524/auto.2010.0815>.
- [10] Shruti Sharma. „Model-Based Design and Control of a Pneumatic Robot Joint“ Bachelor thesis. Birla Institute of Technology und Science, Pilani, 2019.
- [11] Rainer Nitsche und Christian Dieterich. *Components Subset for Simulink. Process Automation Standard Library (PASL) - Documentation*. Techn. Ber. Festo AG & Co. KG , Dept. BA-DEE, 2022.

-
- [12] Daniel Bergmann. „Design and Experimental Validation of a Robust Sliding Mode Position Control for a pneumatic Robot Joint“. Master Thesis. Universität Stuttgart Institut für Analysis, Dynamik und Modellierung, 30. Sep. 2021.

B Messergebnisse pJoint

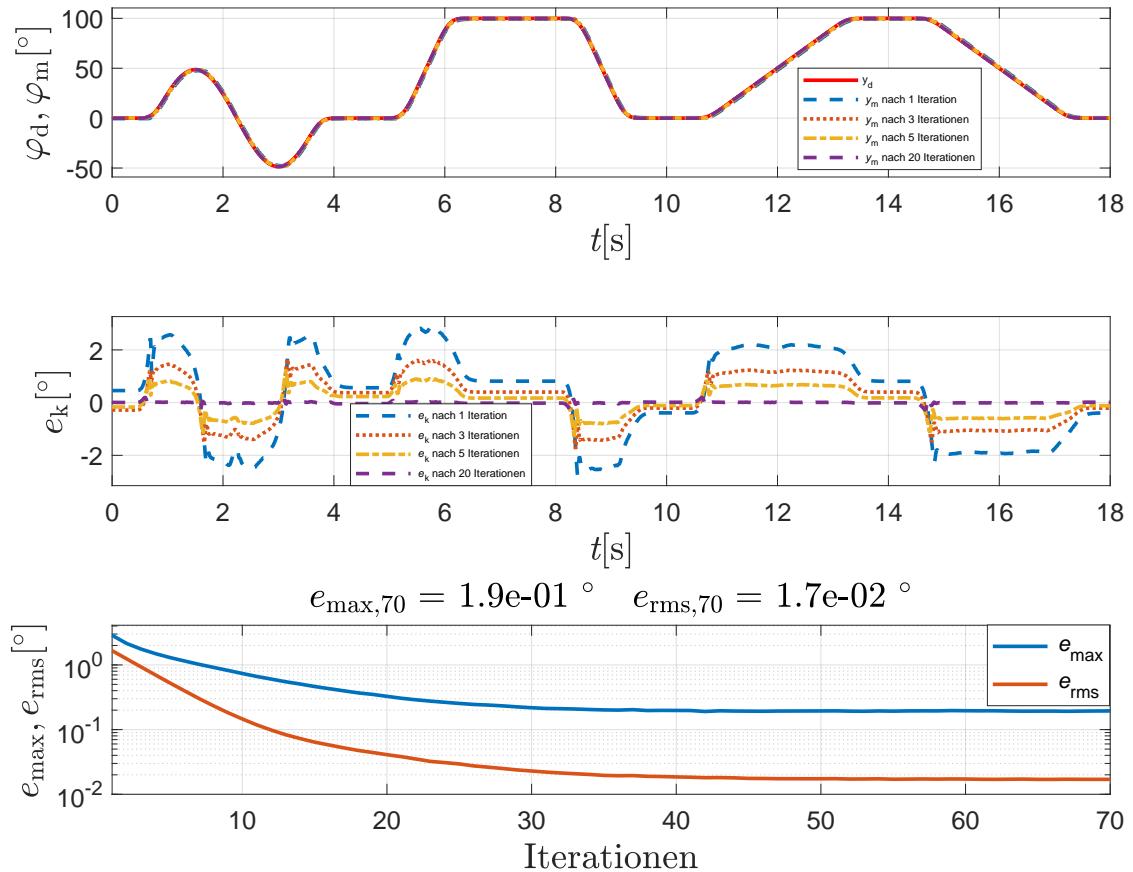


Abbildung B.1: Messergebnis Versuchsaufbau pneumatisches Gelenk mit ROPA-DR-10, unterlagerter Druckregler (Festo Bibliothek) mit überlagertem Positionsregler und ILC-Ansatz

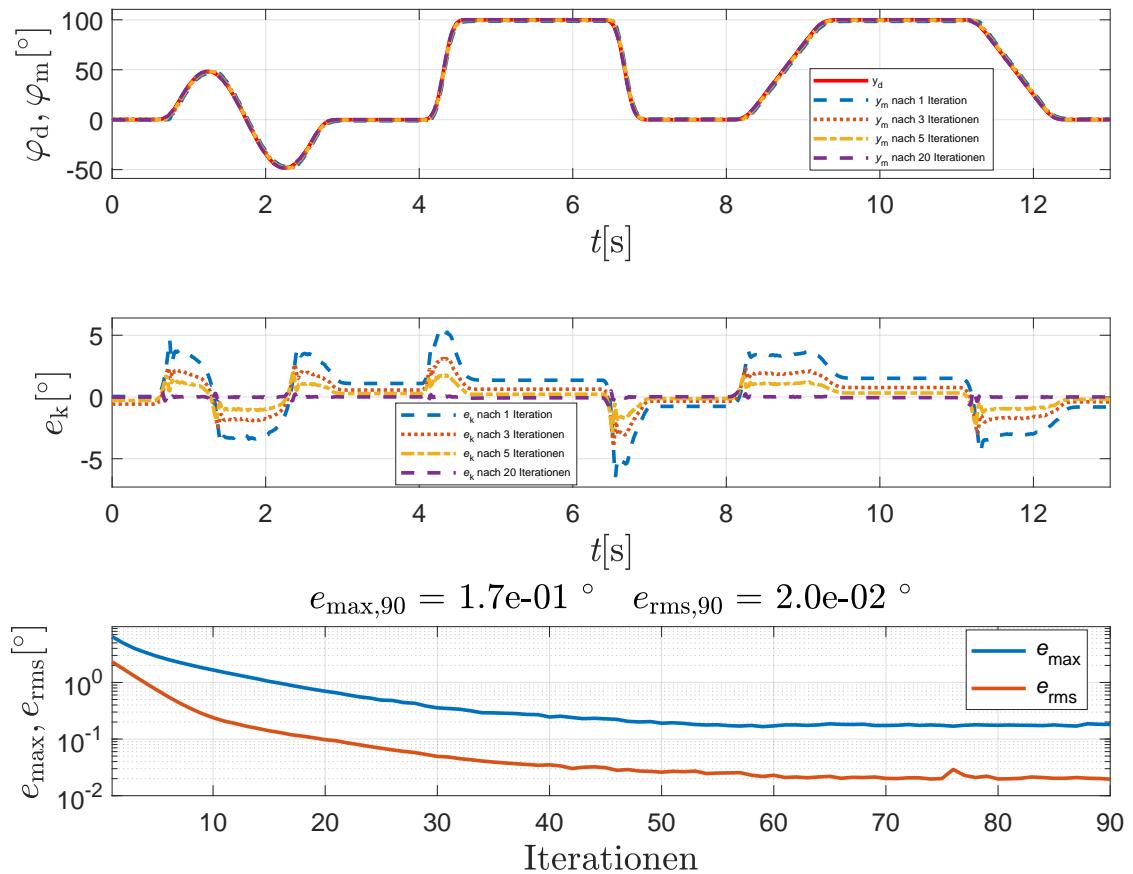


Abbildung B.2: Messergebnis Versuchsaufbau pneumatisches Gelenk mit ROPA-DR-5, unterlagerter Druckregler (Festo Bibliothek) mit überlagertem Positionsregler und ILC-Ansatz