Hochschule Aalen FESTO

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

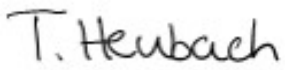
Sperrvermerk

Die nachfolgende Abschlussarbeit beinhaltet vertrauliche und interne Daten der Firma Festo SE & Co. KG. Veröffentlichungen und Vervielfätigungen - 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 1. August 2023 

(Ort, Datum, Unterschrift)

Gender Disclaimer

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

Skriptverzeichnis

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. 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 simulative Tests in Matlab-Simulink als auch reale Tests an der Hardware mithilfe eines dSpace Systems 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 eine kurze Zusammenfassung des Kapitels ist der Hauptteil dieser Thesis abgeschlossen.
Abschließend erfolgen ein Fazit und Ausblick der vorliegenden Bachelorthesis.

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 eingebaut, 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 nicht lineare 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 nächste Durchführung 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 [3]. 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 Modellierungsfehler und eine Änderung des Systemverhaltens über die Zeit, die in jedem Zyklus auftreten, korrigiert werden [2].
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].

2.2 Funktionsweise

Interativ Learning Control funktioniert wie folgt:
In jeder Iteration ($j = 0, 1, \ldots$) wird eine Steuerung u_{j+1} berechnet. Dafür muss der Ausgangsfehler e_j und die Steuerung u_j bestimmt und abgespeichert werden. Der Ausgangsfehler

$$e = y_\mathrm{d} - y_\mathrm{m} \tag{2.1}$$

wird dabei aus der Differenz zwischen dem Sollsignal y_d (desired) und dem gemessenen Signal y_m (measured) gebildet. In Abbildung 2.1 ist der Ablauf des ILC-Ansatzes abgebildet. In jeder Iteration wird die Steuerung u_{j+1} durch die Funktion $\psi\big(u_j, e_j(u_j)\big)$ 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 $y_{\mathrm{m},j+1}$ zu messen. Anhand des Sollsignals y_d wird der neue Ausgangsfehler e_{j+1} ermittelt und zwischengespeichert. Gleichzeitig wird auch das an das System übergebene Stellsignal u_{j+1} abgespeichert. Dieser Prozess kann beliebig oft wiederholt werden, wobei das Ziel darin besteht, die Steuerung u so anzupassen, dass der

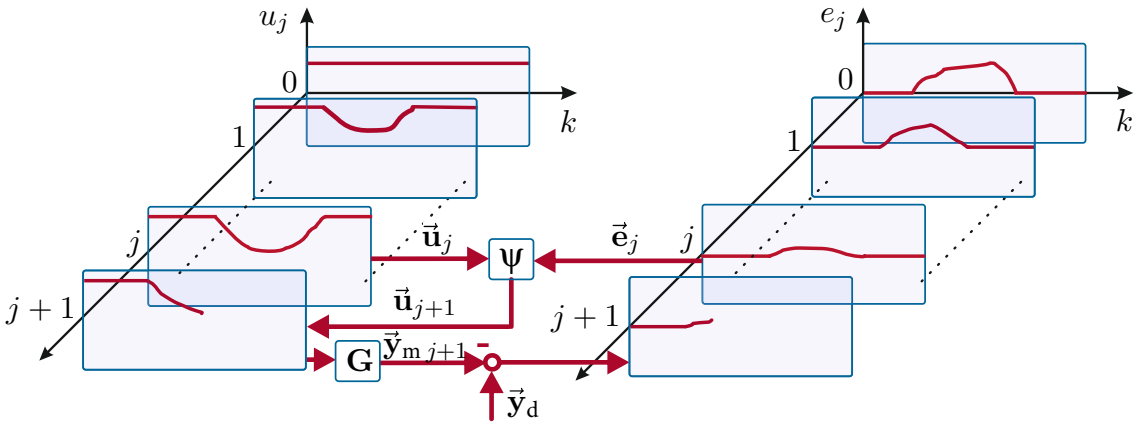


Abbildung 2.1: Grundidee ILC [1]

Ausgangsfehler e gegen null konvergiert.
Die Berechnung des Stellsignals anhand einer Minimierungsaufgabe wird in den Vorlesungsunterlagen der Technischen Universität (TU) Wien [1] 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, auch bekannt als Polüberschuss. Für zeitkontinuierliche Systeme kann der relative Grad mit der Gleichung

$$r = n - m \tag{2.2}$$

berechnet werden, wobei der höchste Exponent des Zählerpolynoms der Übertragungsfunktion 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

$$v = r + w \tag{2.3}$$

eingeführt und im ILC-Gesetz berücksichtigt werden.

Mit der Definition der Vektoren:

$$\vec{\boldsymbol{u}}_j = \begin{bmatrix}\vec{\boldsymbol{u}}_j[0] & \vec{\boldsymbol{u}}_j[1] & \dots & \vec{\boldsymbol{u}}_j[N-1]\end{bmatrix}^T \in R^N \tag{2.4}$$

$$\vec{\boldsymbol{y}}_{\mathrm{d},j} = \begin{bmatrix}\vec{\boldsymbol{y}}_{\mathrm{d},j}[v] & \vec{\boldsymbol{y}}_{\mathrm{d},j}[v+1] & \dots & \vec{\boldsymbol{y}}_{\mathrm{d},j}[v+N-1]\end{bmatrix}^T \in R^N \tag{2.5}$$

$$\vec{\boldsymbol{y}}_{\mathrm{m},j} = \begin{bmatrix}\vec{\boldsymbol{y}}_{\mathrm{m},j}[v] & \vec{\boldsymbol{y}}_{\mathrm{m},j}[v+1] & \dots & \vec{\boldsymbol{y}}_{\mathrm{m},j}[v+N-1]\end{bmatrix}^T \in R^N \tag{2.6}$$

$$\vec{\boldsymbol{e}}_j = \vec{\boldsymbol{y}}_{\mathrm{d},j} - \vec{\boldsymbol{y}}_{\mathrm{m},j} = \begin{bmatrix}\vec{\boldsymbol{e}}_j[v] & \vec{\boldsymbol{e}}_j[v+1] & \dots & \vec{\boldsymbol{e}}_j[v+N-1]\end{bmatrix}^T \in R^N \tag{2.7}$$

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

$$\vec{\boldsymbol{u}}_{j+1} = \mathbf{Q}\left(\vec{\boldsymbol{u}}_j + \mathbf{L}\vec{\boldsymbol{e}}_j\right). \tag{2.8}$$

Die Verstärkungsmatrix $\mathbf{L}$ hat die Aufgabe, den Einfluss des Ausgangsfehlers e_j in Relation zur vorherigen Steuerung 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

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

angegeben werden. 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 $\vec{\boldsymbol{g}}$ des Systems benötigt. Aus der Impulsantwort kann die Matrix

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

gebildet werden. Es ist wichtig zu beachten, dass $g[m] \neq 0$ 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]:

$$\rho\left(\mathbf{A}\right) = \max\left|\lambda_i\left(\mathbf{A}\right)\right| \tag{2.11}$$

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

$$\rho\left(\boldsymbol{Q}\left(\boldsymbol{E}-\boldsymbol{LG}\right)\right) < 1 \tag{2.12}$$

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

$$\rho\left(\boldsymbol{GQ}\left(\boldsymbol{E}-\boldsymbol{LG}\right)\boldsymbol{G}^{-1}\right) < 1 \tag{2.13}$$

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

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

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 u_∞, wenn gilt:*

$$\|u_{j+1} - u_\infty\|_2 \leq \alpha \|u_j - u_\infty\|_2$$

für $0 \leq \alpha < 1$

$$\bar{\sigma}\left(\boldsymbol{Q}\left(\boldsymbol{E} - \boldsymbol{L}\boldsymbol{G}\right)\right) < 1 \tag{2.15}$$

Satz 4 (Monotone Konvergenz der Ausgangsfehleriteration) *Die Ausgangsfehleriteration der Lifted-System Darstellung (2.8) ist konvergent gegen e_∞, wenn gilt:*

$$\|e_{j+1} - e_\infty\|_2 \leq \beta \|e_j - e_\infty\|_2$$

für $0 \leq \beta < 1$

$$\bar{\sigma}\left(\boldsymbol{G}\boldsymbol{Q}\left(\boldsymbol{E} - \boldsymbol{L}\boldsymbol{G}\right)\boldsymbol{G}^{-1}\right) < 1 \tag{2.16}$$

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 Tôrres [7] mit

$$\vec{\boldsymbol{e}}_\infty = \left(\mathbf{E} - \mathbf{G}\left(\mathbf{E} - \mathbf{Q}\left(\mathbf{E} - \mathbf{L}\mathbf{G}\right)\right)^{-1}\mathbf{Q}\mathbf{L}\right)\left(\vec{\boldsymbol{y}}_\mathrm{d} - \vec{\boldsymbol{y}}_{\mathrm{m},0}\right) \tag{2.17}$$

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

$$\mathbf{Q} = \mathbf{E} \qquad \rightarrow \qquad \vec{\boldsymbol{e}}_\infty = 0. \tag{2.18}$$

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 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 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 eine P-Type. Dabei wird die Verstärkungsmatrix $\mathbf{L}$ zur einer diagonalen Verstärkungsmatrix mit dem Verstärkungsfaktor $k_\mathrm{p} > 0$ gewählt.

$$\mathbf{L} = k_\mathrm{p}\mathbf{E} \tag{2.19}$$

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

$$\vec{\boldsymbol{u}}_{j+1} = \mathbf{Q}\left(\vec{\boldsymbol{u}}_j + k_\mathrm{p}\mathbf{E}\,\vec{\boldsymbol{e}}_j\right). \tag{2.20}$$

und analog dazu die diskrete Darstellung

$$u_{j+1}[k] = q[k]\big(u_j[k] + k_\mathrm{p}\, e_j[k+v]\big). \tag{2.21}$$

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 Differenzialanteil einzustellen, wird die Verstärkungsmatrix $\mathbf{L}$ um einen zusätzlichen Einstellparameter $k_\mathrm{d} > 0$ und einer Ableitungsfiltermatrix $\mathbf{D}$ erweitert. Daraus ergibt sich

$$\mathbf{L} = k_\mathrm{p}\mathbf{E} + k_\mathrm{d}\mathbf{D} \tag{2.22}$$

mit der Ableitungsfiltermatrix

$$\mathbf{D} = \frac{1}{T_\mathrm{s}} \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}. \tag{2.23}$$

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

$$\vec{\boldsymbol{u}}_{j+1} = \mathbf{Q}\left(\vec{\boldsymbol{u}}_j + \left(k_\mathrm{p}\mathbf{E} + k_\mathrm{d}\mathbf{D}\right)\vec{\boldsymbol{e}}_j\right) \tag{2.24}$$

und analog dazu die diskrete Schreibweise

$$u_{j+1}[k] = q[k]\left(u_j[k] + k_\mathrm{p}\, e_j[k+v] + \frac{k_\mathrm{d}}{2T_\mathrm{s}}\left(e_j[k+v+1] - e_j[k+v-1]\right)\right). \tag{2.25}$$

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 $\vec{\boldsymbol{g}}$ 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.

$$\mathbf{L} = \mathbf{G}^{-1} \tag{2.26}$$

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

$$\vec{\boldsymbol{u}}_{j+1} = \mathbf{Q}\left(\vec{\boldsymbol{u}}_j + \mathbf{G}^{-1}\vec{\boldsymbol{e}}_j\right). \tag{2.27}$$

Mit diesem Verfahren ist es möglich, den Ausgangsfehler e_j in einer Iteration auf den Grenzwert e_∞ zu bringen. Wenn die Filtermatrix zu $\mathbf{Q} = \mathbf{E}$ gewählt wird, gilt zudem $e_\infty = 0$. 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

$$q_{\mathrm{G}}[k] = \frac{T_{\mathrm{s}}}{\sigma_{\mathrm{q}}\sqrt{2\pi}}\, exp\left(-\frac{(kT_{\mathrm{s}})^2}{2\sigma_{\mathrm{q}}^2}\right) \tag{2.28}$$

mit der Standartabweichung

$$\sigma_{\mathrm{q}} = \frac{\sqrt{\ln(2)}}{2\pi f_{\mathrm{c}}} \tag{2.29}$$

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 $2N\mathrm{q}+1$ zu erhalten, ergibt sich der Vektor für den Gaußfilter zu

$$q[k] = \frac{q_{\mathrm{G}}[k]}{\sum_{m=-N_{\mathrm{q}}}^{N_{\mathrm{q}}} q_{\mathrm{G}}[m]} \in R^N. \tag{2.30}$$

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 $N_{\mathrm{q}} = 2$ gegeben:

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

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 2.2 dargestellte Sollsignal $\vec{y}_\mathrm{d}$ eingesetzt. Das Sollsignal besteht aus einer Sinusschwingung, einer sprungförmigen

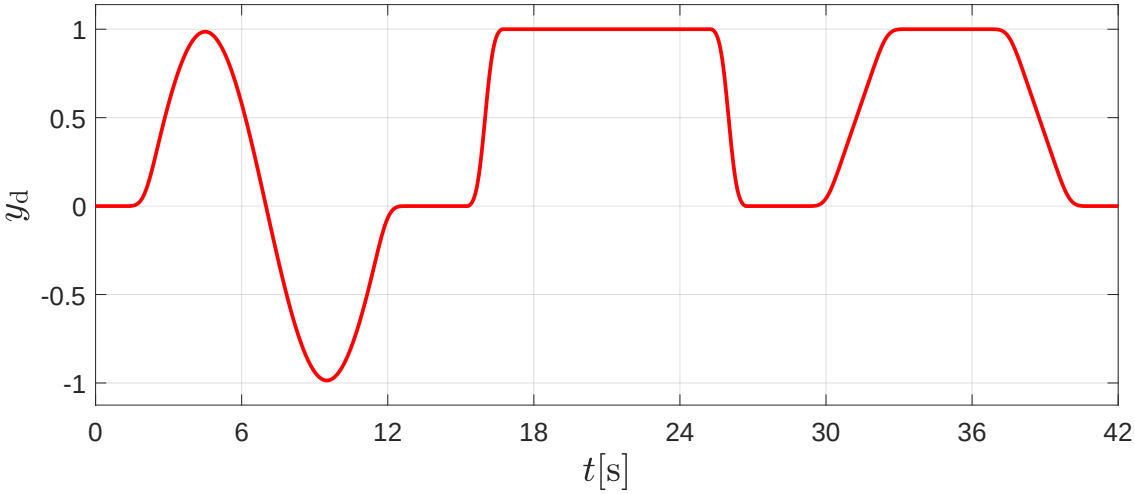


Abbildung 2.2: Sollsignal

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.

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) *
                        reshape(y_d,numel(y_d),1);
13                  e_inf_max(end+1) = max(abs(e_inf_seq(:,n)));
14                  e_inf_rms(end+1) = rms(e_inf_seq(:,n));
15              end
16              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 2.3 dargestellt ist. Das Modell besteht aus zwei Subsystemen. Im

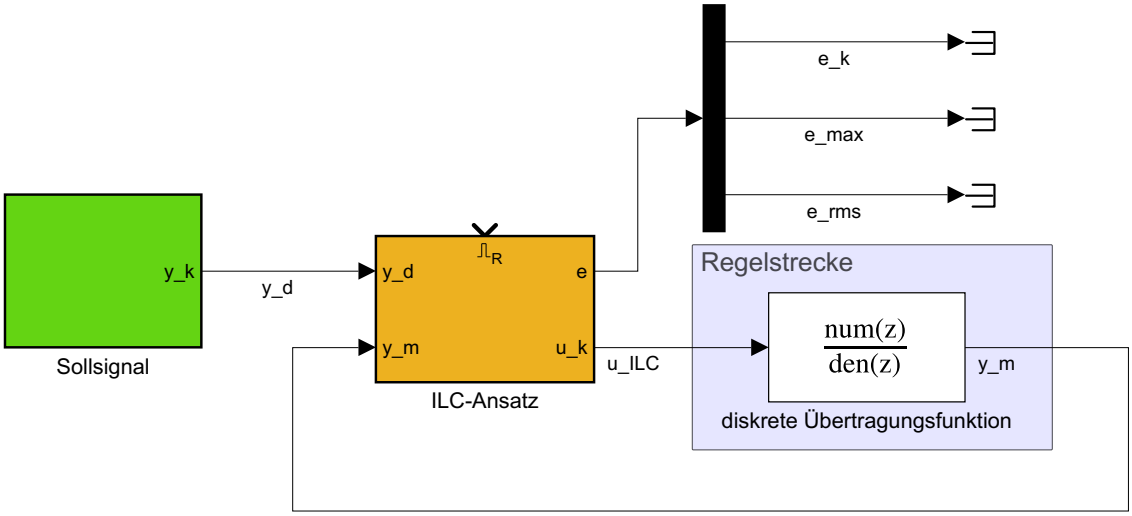


Abbildung 2.3: Matlab Modell ILC-Ansatz

grünen Subsystem wird das Sollsignal 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. 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 $G_\mathrm{s}(s)$ in eine diskrete Übertragungsfunktion $G_\mathrm{s}(z)$ 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 2.4). Der Parameter *Modus* ermöglicht die Auswahl zwischen verschiedenen ILC-Typen, während der Parameter *Filterung* entscheidet, ob eine

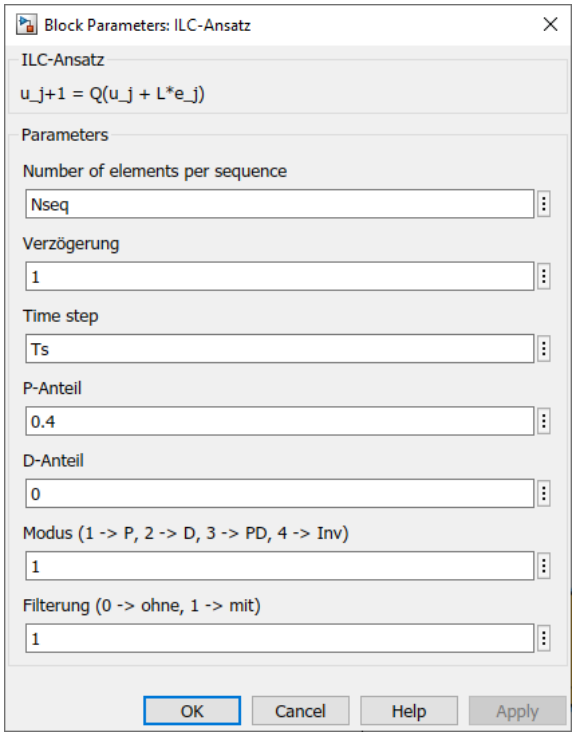


Abbildung 2.4: ILC Subsystem Maske

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 $\vec{g}$ der Übertragungsfunktion, welche zuvor berechnet werden muss, im Hintergrund in den ILC-Block geladen.

ILC-Algorithmus

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

```
function [u_k,e_k,e_max,e_rms] = fcn(y_d,y_m,kp,kd,Nseq,v,modus,Ts,g,filt)
% Variablen definition und Initialisierung
assert(Nseq <= 10001);
persistent Useq e e_rms_tmp e_max_tmp k E Q D G_inv
if isempty(Useq), Useq = zeros(Nseq,1); end
if isempty(e), e = zeros(Nseq,1); end
if isempty(e_rms_tmp), e_rms_tmp = 0; end
if isempty(e_max_tmp), e_max_tmp = 0; end
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 modus == 1
35         Useq(Nseq-v+1:Nseq) = Useq(Nseq-v+1:Nseq) + kp * E(1:v,:) * e(:,1);
36     elseif modus == 2
37         Useq(Nseq-v:Nseq) = Useq(Nseq-v:Nseq) + kd * D(1:v,:) * e(:,1);
38     elseif modus == 3
39         Useq(Nseq-v:Nseq) = Useq(Nseq-v:Nseq) + (kp * E(1:v,:) + kd *
               D(1:v,:)) * e(:,1);
40     elseif modus == 4
41         Useq(Nseq-v+1:Nseq) = Useq(Nseq-v+1:Nseq) + G_inv(1:v,:) * e(:,1);
42     end
43 end
44 % Zähler zurücksetzen Stellsignal für nächsten Zyklus berechnen und ggf.
       filtern
45 if k > Nseq
46     k = 1;
47     e = circshift(e,-v);
```

```
48      e_rms_tmp = rms(e);
49      e_max_tmp = max(abs(e));
50      if modus == 1
51          Useq(1:Nseq-v) = Useq(1:Nseq-v) + kp * E(1:Nseq-v,:) * e;
52      elseif modus == 2
53          Useq(1:Nseq-v) = Useq(1:Nseq-v) + kd * D(1:Nseq-v,:) * e;
54      elseif modus == 3
55          Useq(1:Nseq-v) = Useq(1:Nseq-v) + (kp * E(1:Nseq-v,:) + kd *
                D(1:Nseq-v,:)) * e;
56      elseif modus == 4
57          Useq(1:Nseq-v) = Useq(1:Nseq-v) + G_inv(1:Nseq-v,:) * e(:,1);
58      end
59      e = zeros(Nseq,1);
60      if filt
61          Useq = Q * Useq;
62      end
63  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 2.3 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 2.5), 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 2.6), hier wird der ILC-Anteil zum Eingang des Reglers addiert. Die parallele Struktur

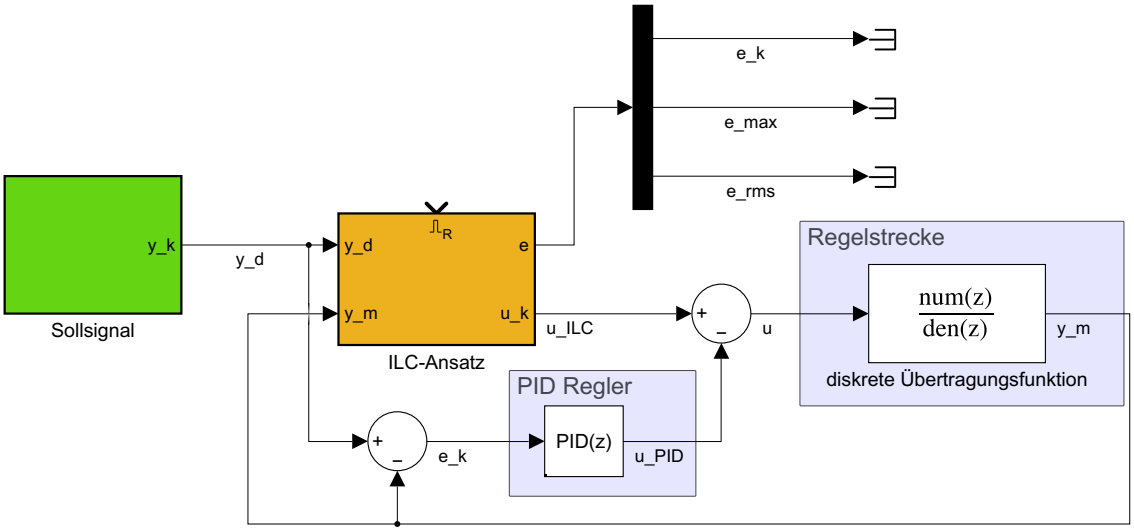


Abbildung 2.5: Matlab Modell ILC-Ansatz mit PID-Regler in paralleler Struktur

wird als Zwei-Freiheitsgrade-Regelung bezeichnet.

2.6 Ergebnisse

In diesem Abschnitt werden die Ergebnisse aus der Untersuchung der Stabilitäts- und Konvergenzkriterien präsentiert. Dabei wird außerdem der ILC-Algorithmus

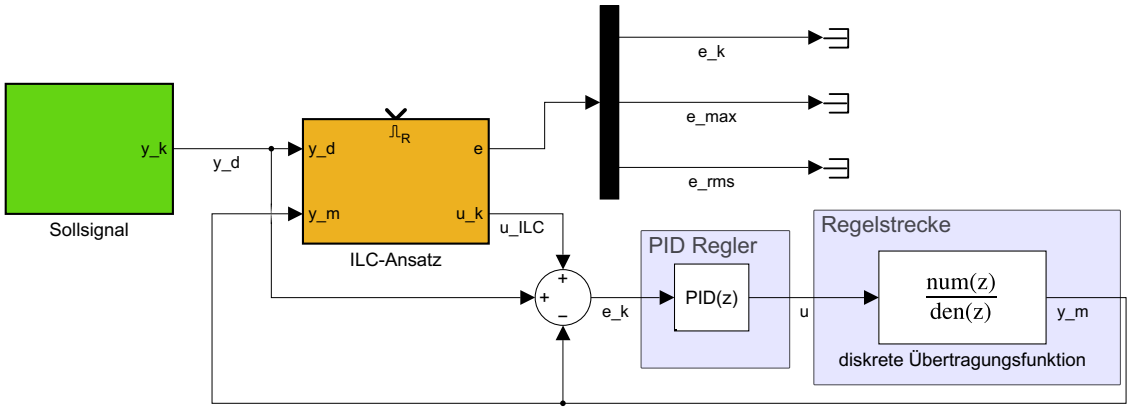


Abbildung 2.6: Matlab Modell ILC-Ansatz mit PID-Regler in serieller Struktur

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 $G_\mathrm{s}(s)$ ist gegeben durch

$$G_\mathrm{s}(s) = \frac{0.1}{0.4s + 1.1}. \tag{2.32}$$

Wie in Abschnitt 2.5.2 erläutert, wird bei der Simulation die Regelstrecke im diskreten Zeitbereich benötigt. Dafür muss die kontinuierliche Übertragungsfunktion $G_\mathrm{s}(s)$ mithilfe der Z-Transformation in eine diskrete Übertragungsfunktion $G_\mathrm{s}(z)$ transformiert werden. Dies kann zum Beispiel durch eine Rechteck-Vorwärts-Transformation, auch Euler-Transformation genannt, der kontinuierlichen Übertragungsfunktion mit $s = \frac{1}{T_\mathrm{s}}(z-1)$ 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 $T_\mathrm{s} = 0.1\,\mathrm{s}$ gewählt. Dadurch ergibt sich die diskrete Übertragungsfunktion (bestimmt mit Matlab) zu

$$G_\mathrm{s}(z) = \frac{0.02186}{z - 0.7596}. \tag{2.33}$$

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 2.7 dargestellt. Im oberen und unteren Diagramm sind die Auswirkungen

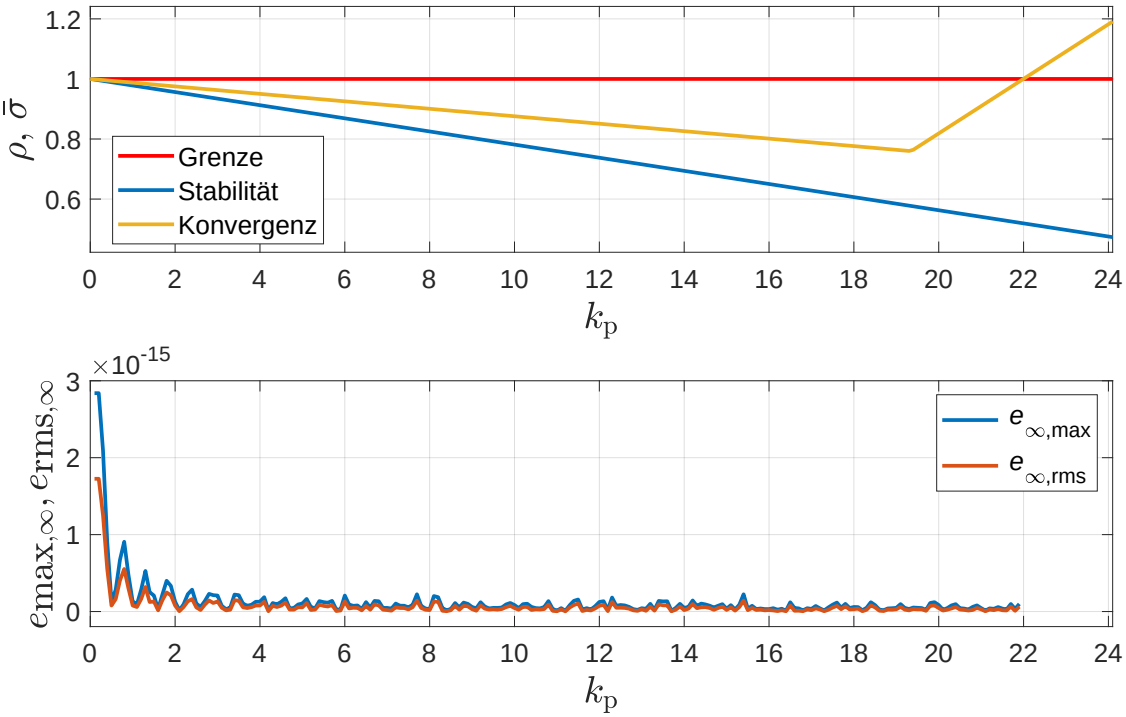


Abbildung 2.7: Stabilitäts- und Konvergenzuntersuchung PT1-System

des Parameters k_p auf die Stabilität, Konvergenz und der asymptotischen Regelabweichung ($e_{\mathrm{max},\infty}$ und $e_{\mathrm{rms},\infty}$) des Systems zu sehen. 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 $0 < k_\mathrm{p} < 22$ eingesetzt werden kann, um eine stabile Regelung mit einer Konvergenz

mit Werten 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 2.3 mit der definierten diskreten Übertragungsfunktion (Gleichung 2.33) verwendet. Als P-Anteil wird ein k_p-Wert von 4 eingestellt, welcher experimentell bestimmt wurde. In Abbildung 2.8 sind die Ergebnisse der Simulation zu sehen. Im oberen Schaubild

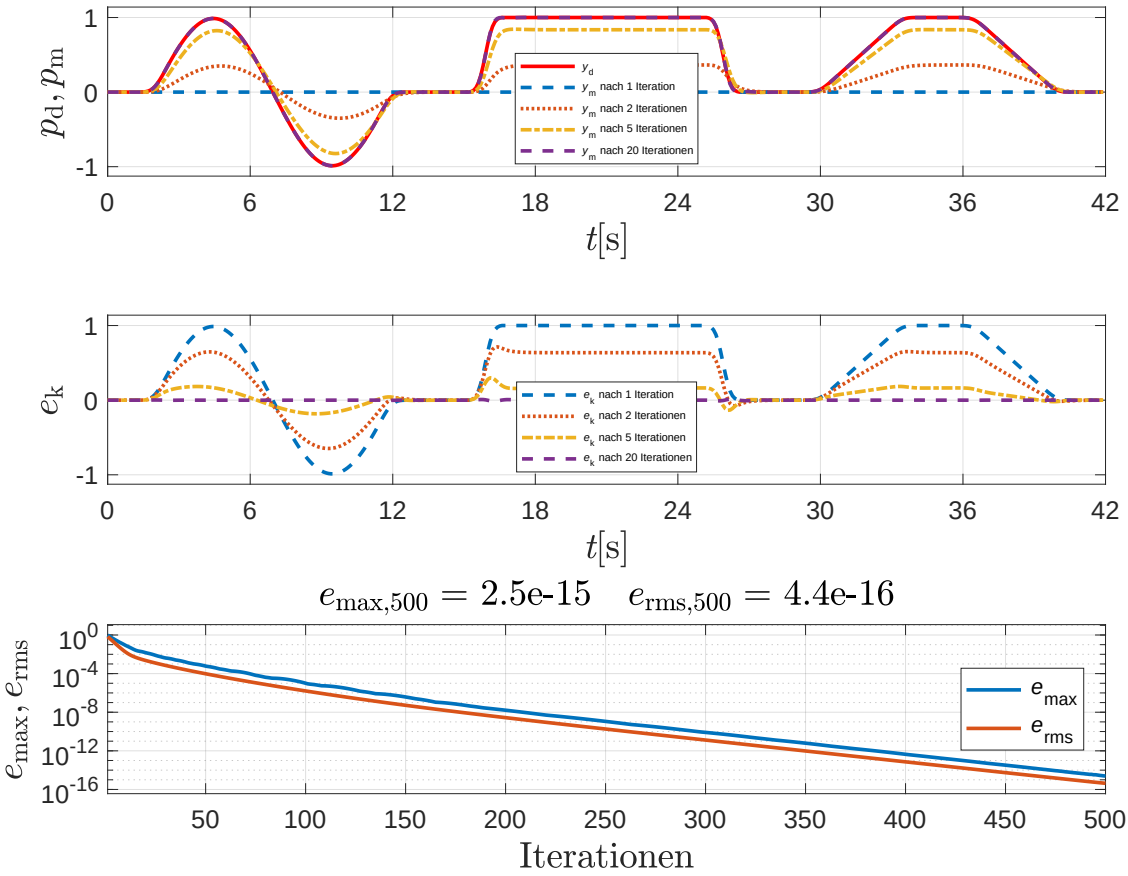


Abbildung 2.8: Simulationsergebnis P-Typ ILC anhand PT1-System

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. 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 2.9 zu sehen. Das Schaubild hat den gleichen Aufbau wie Abbildung 2.8. 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 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

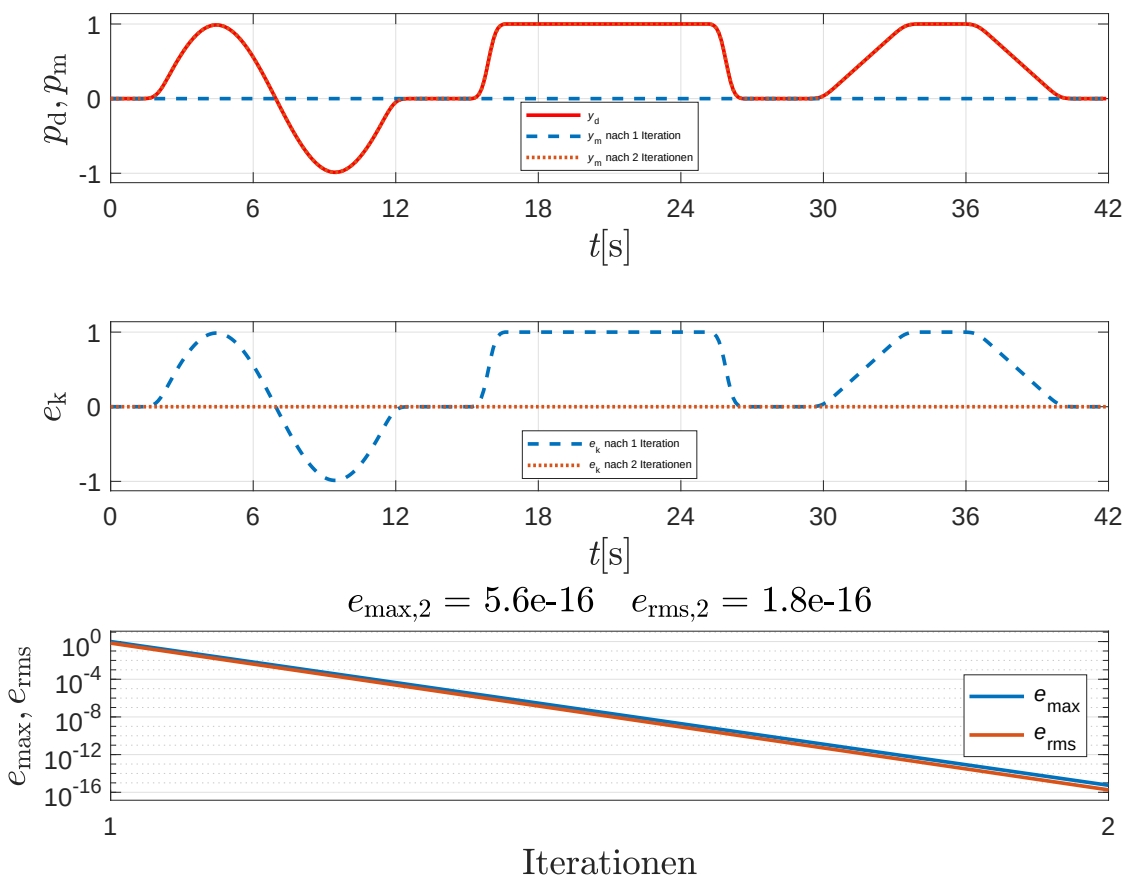


Abbildung 2.9: Simulationsergebnis Inversionsbasiertes ILC anhand PT1-System

der Simulation sind in Abbildung 2.10 dargestellt. Die blauen Linien in den oberen beiden Diagrammen zeigen das Ergebnis der Regelung 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 der 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

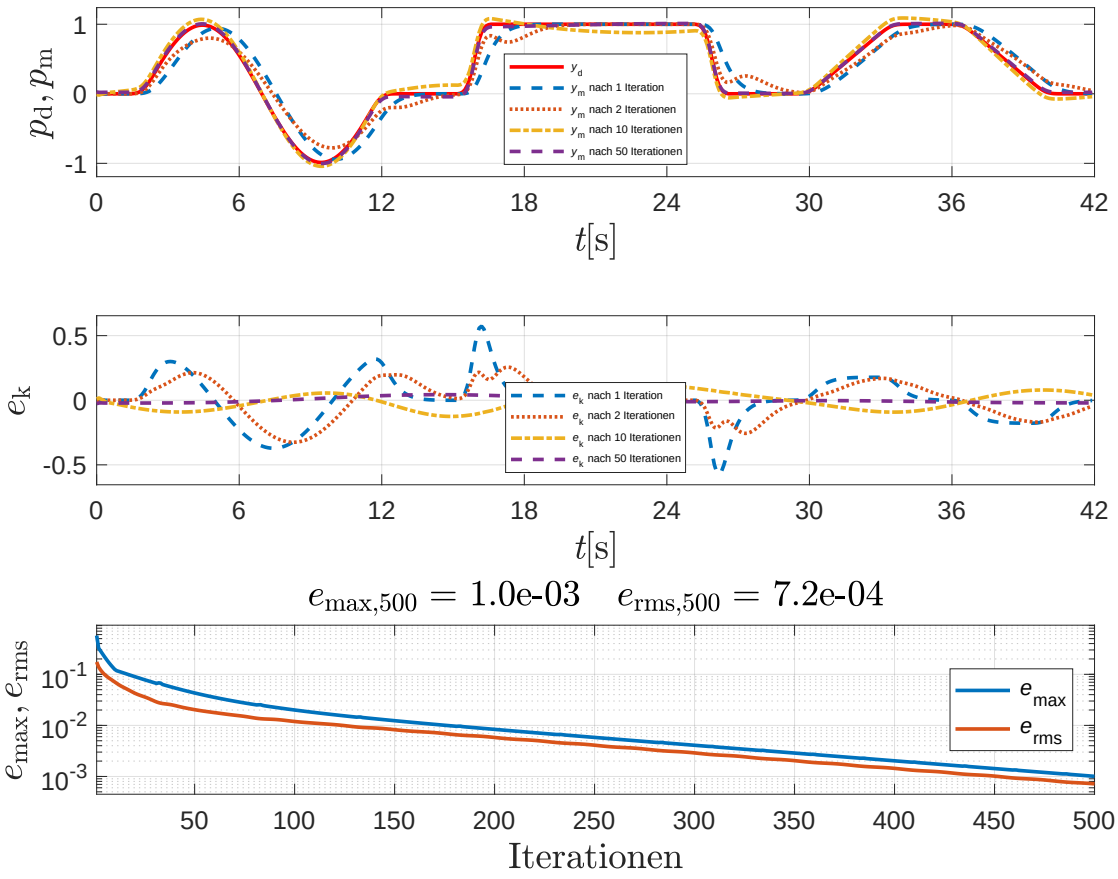


Abbildung 2.10: Simulationsergebnis ILC-Ansatz + PI-Regler in paralleler Struktur anhand PT1-System

ILC-Ansatzes zusätzlich verstärkt. Die Simulationsergebnisse sind in Abbildung 2.11 zu sehen. 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.

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

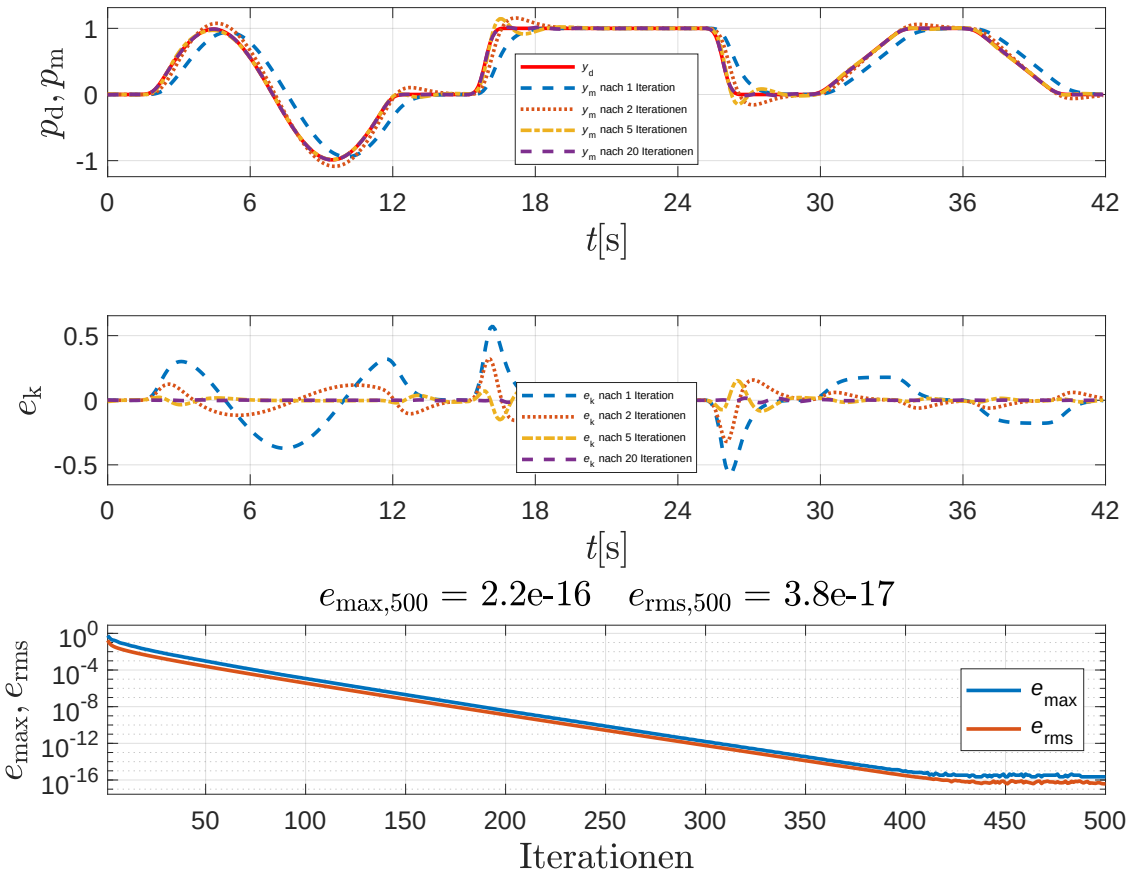


Abbildung 2.11: Simulationsergebnis ILC-Ansatz + PI-Regler in serieller Struktur anhand PT1-System

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 Übertragungsfunktion ist gegeben durch

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

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

$$G_\mathrm{s}(z) = \frac{4z + 1.371 \cdot 10^{-7}}{z^2 - 1.668 \cdot 10^{-7} z}. \tag{2.35}$$

P-Type ILC

In Abbildung 2.12 sind die Ergebnisse für die Stabilitäts- und Konvergenzuntersuchung zu sehen. Für das gegebene System erfüllen k_p-Werte von $0 < k_\mathrm{p} < 0.5$ die Stabilitäts-

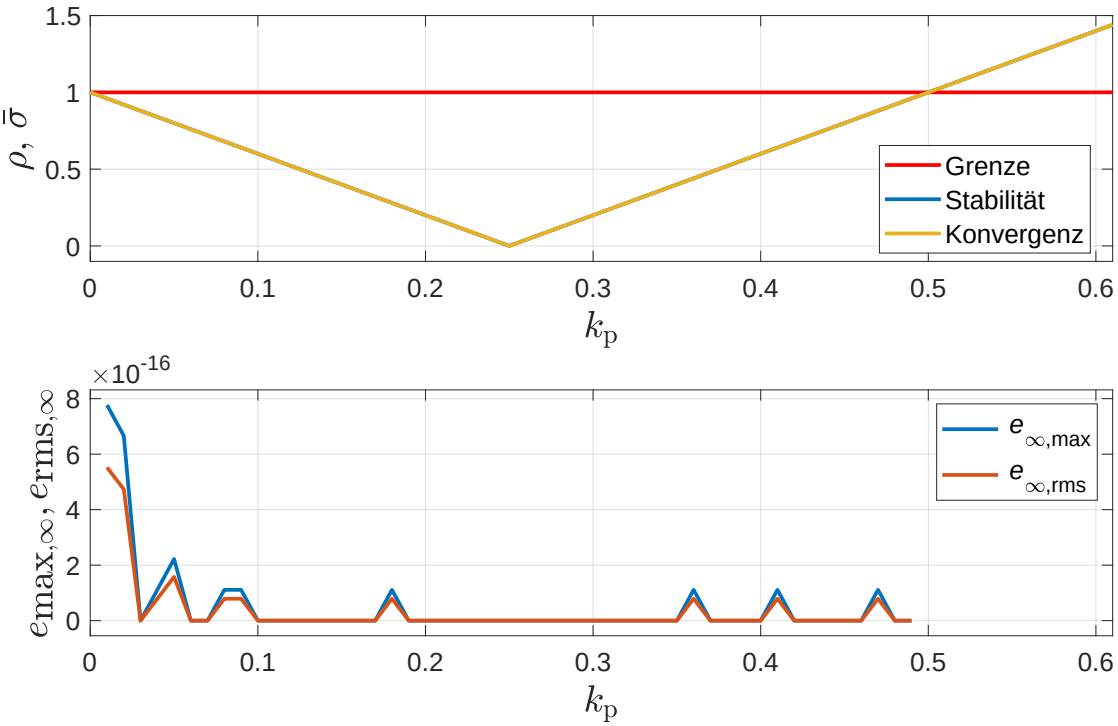


Abbildung 2.12: Stabilitäts- und Konvergenzuntersuchung PT2-System

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 2.13 aufgezeigt. 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 2.14 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.

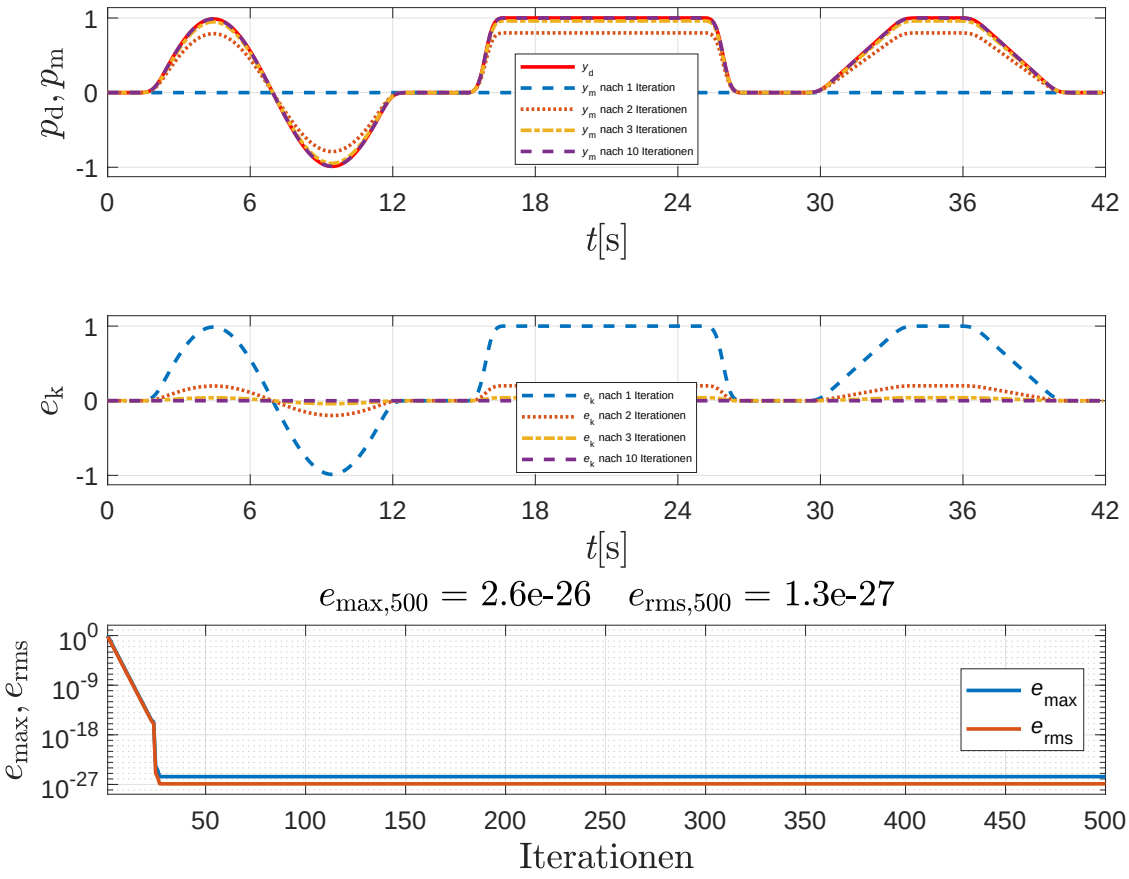


Abbildung 2.13: Simulationsergebnis P-Typ ILC anhand PT2-System

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 Zwei-Freiheitsgrade-Regelung mit einer Vorsteuerung für das PT2-System getestet. Hierbei wird ein flachheitsbasierter Ansatz verwendet, bei dem das gegebene System (2.35) invertiert und der Ausgang zu dem Stellsignal des ILC-Ansatzes addiert wird. Das entsprechende Simulink-Modell ist in Abbildung 2.15 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. Das in blau dargestellte Subsystem des Modells

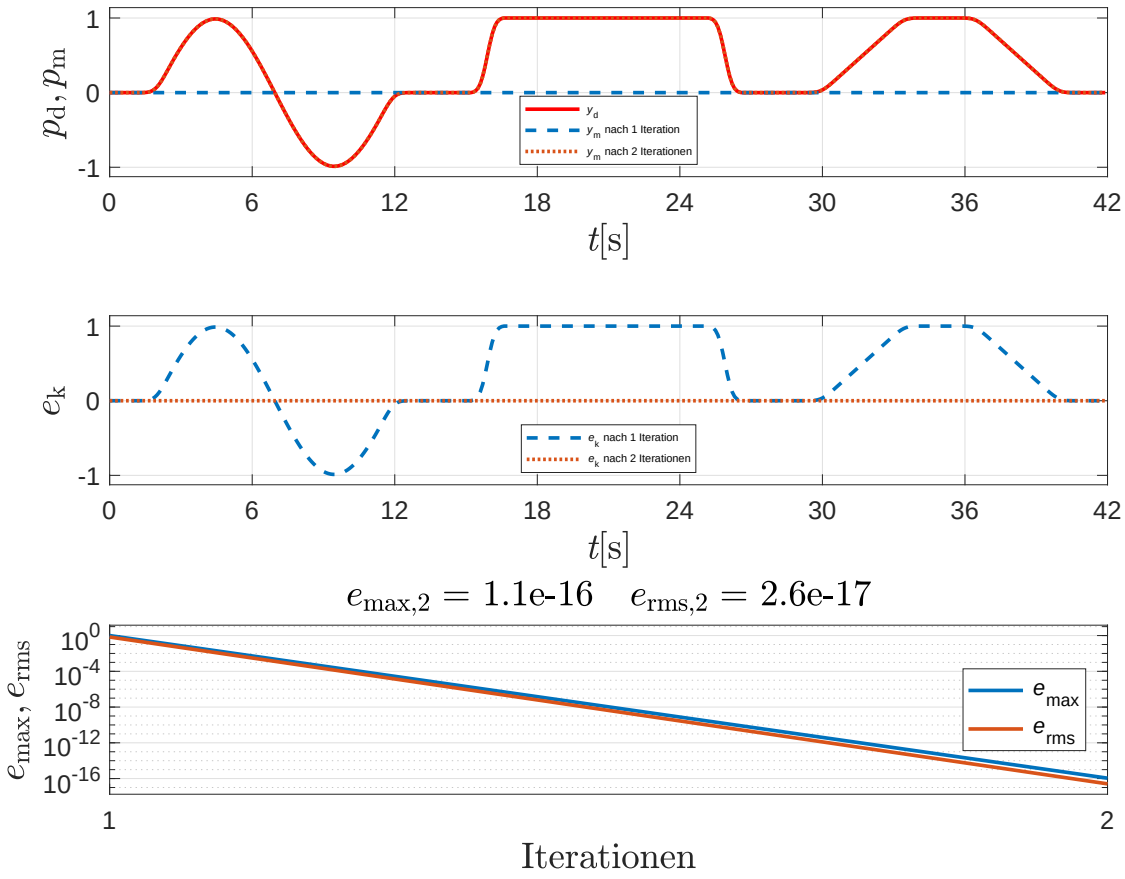


Abbildung 2.14: Simulationsergebnis Inversionsbasiertes ILC anhand PT2-System

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 (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:

$$u_\mathrm{flach} = \frac{1}{4}\left(y_\mathrm{d} + 7.5 \cdot 10^{-3}\ddot{y}_\mathrm{d} + 7 \cdot 10^{-7}\ddot{y}_\mathrm{d}\right). \tag{2.36}$$

Aus Gleichung 2.36 ergibt sich das dazugehörige Simulink-Modell wie in Abbildung 2.16 abgebildet.

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

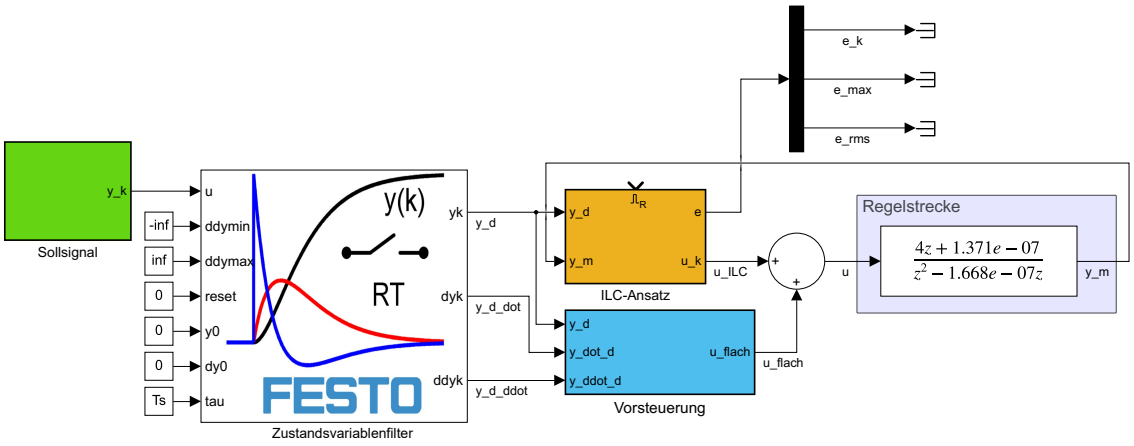


Abbildung 2.15: ILC-Ansatz + flachheitsbasierte Vorsteuerung

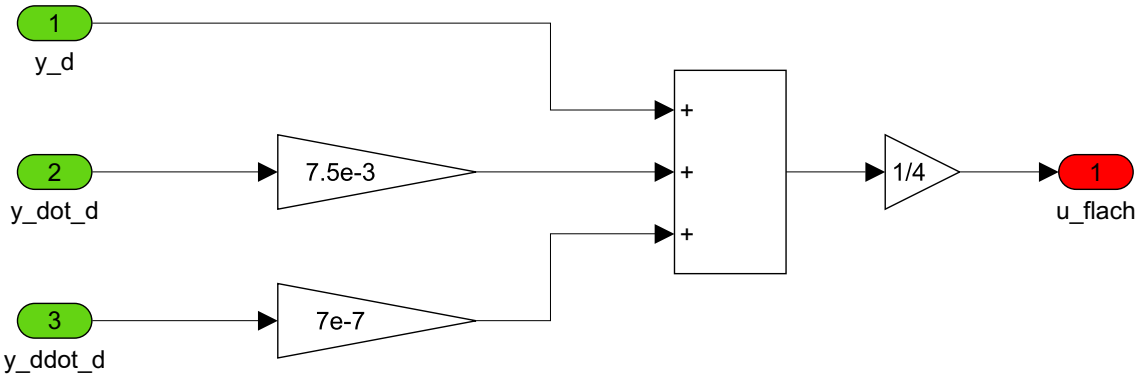


Abbildung 2.16: flachheitsbasierte Vorsteuerung

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 2.17 dargestellt. 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 $9.8 \cdot 10^{-18}$.

2.6.3 Zusammenfassung

In diesem Abschnitt werden die Ergebnisse und Erkenntnisse der durchgeführten Untersuchungen zusammengefasst. Die Stabilitäts- und Konvergenzanalysen haben es

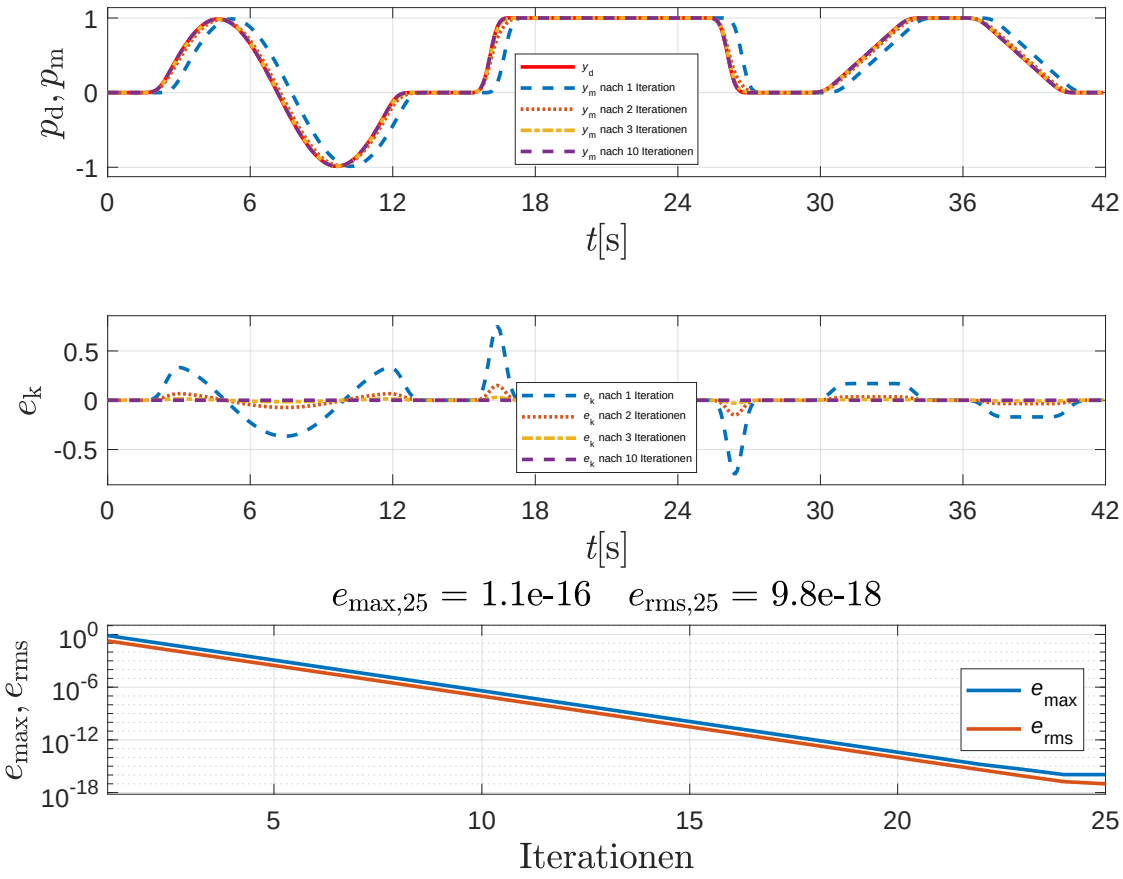


Abbildung 2.17: Simulationsergebnis ILC-Ansatz + flachheitsbasierte Vorsteuerung anhand PT2-System

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 Solltrajektorie ist der ILC-Ansatz in der Lage, Phasenverschiebungen und Totzeiten im System zu kompensieren.

A Literaturverzeichnis

[1] M. Böck, T. Glück, A. Kugi und A. Steinböck. „Fortgeschrittene Methoden der nichtlinearen Regelung“. In: *Vorlesungsskript, Technische Universität Wien* (2022).

[2] 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.

[3] Technische Universität Kaiserslauter. *Forschungsbericht 2000*. URL: https://www.eit.uni-kl.de/pandit/haupt/forschung/jahresbericht00.htm (besucht am 25. 04. 2023).

[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/analysis/spektralradius.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] Shruti Sharma. „Model-Based Design and Control of a Pneumatic Robot Joint“. Bachelor thesis. Birla Institute of Technology und Science, Pilani, 2019.

[10] 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.

[11] 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