Ein neuro-adaptives Regelungskonzept mit on-line-fähigem Kalman-Filter Lernverfahren für stochastisch gestörte nicht-lineare Prozesse

Dipl.-Ing. C. W. Frey, Dipl.-Ing. M. Sajidman, Dr.-Ing. H.-B. Kuntze,

Fraunhofer-Institut IITB, Karlsruhe

Durch den Einsatz von neuronalen Netzen eröffnen sich prinzipiell viele neuartige Möglichkeiten, um leistungsfähige, adaptive Regelungskonzepte zu realisieren [1]. Der vergleichsweise hohe Rechenaufwand sowie der Mangel an schnell konvergierenden on-line-fähigen Lernverfahren waren bisher wesentliche Gründe für die mangelnde Akzeptanz in der industriellen Praxis. Im vorliegenden Beitrag wird ein adaptives Regelungskonzept basierend auf einem Kalman-Filter Lernalgorithmus vorgestellt, das sich im Gegensatz zu anderen Lernverfahren, wie z.B. dem Backpropagation- oder dem Levenberg-Marquardt-Algorithmus, auch für stochastisch gestörte Prozesse mit a priori unbekanntem Systemverhalten anwenden läßt. Am Beispiel eines nichtlinearen Prozesses wird gezeigt, daß der adaptive neuronale PID-Algorithmus trotz starker stochastischer Störungen in der Lage ist, sich nicht nur an transiente Sollverläufe, sondern auch an beliebige stationäre Arbeitspunkte des Prozesses ausgezeichnet anzupassen.

1 Einleitung

Klassische PID-Regelalgorithmen finden in der industriellen Praxis seit vielen Jahren eine weite Verbreitung. Ein wesentlicher Grund hierfür ist die einfache Realisierbarkeit des Reglers und die Interpretierbarkeit der Regelparameter, in Form von Proportional-, Integral- und Differentialanteil (k_p, k_i und k_d). Abbildung 1.1 zeigt einen zeitdiskreten PID-Regelalgorithmus in einer für die Parameteradaption sinnvollen Darstellungsform:

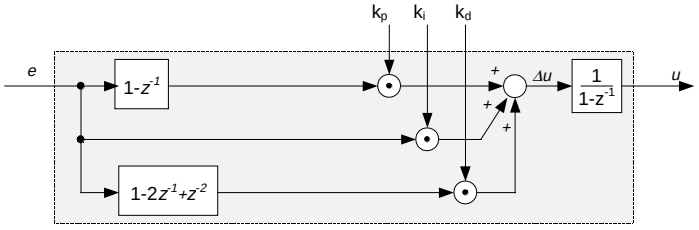

Abbildung 1.1: PID-Regelalgorithmus

Bedingt durch die starre Einstellung der Regelparameter eines solchen PID-Reglers gelten für einen Einsatz in der Praxis folgende Einschränkungen:

- Der Regler kann sich nicht an ein zeitlich veränderliches Verhalten des zu regelnden Systems anpassen.
- Nichtlineare Systeme, die eine arbeitspunktabhängige Einstellung des Reglers erfordern, können nur unbefriedigend geregelt werden.

Zur Adaption von PID-Regelalgorithmen an das veränderliche Verhalten der Regelstrecke wurden in der Vergangenheit zahlreiche Konzepte basierend auf analytischen Modellen vorgeschlagen [2,3]. Mit zunehmender Rechenleistung wurden mehr und mehr auf neuronalen Netzen basierende Konzepte entwickelt, die sich in hybride und kooperative Ansätze unterscheiden lassen [4,5]. Das hier vorgestellte Konzept basiert auf einem kooperativen Ansatz, und zeichnet sich durch folgende Merkmale aus:

- Strukturerhalt: Der PID-Regelalgorithmus soll in seiner Struktur erhalten bleiben.
- Echtzeitfähigkeit: Der Adaptionsalgorithmus muß in der Lage sein, den Regler in Echtzeit für ein unbekanntes oder zeitlich veränderliches Systemverhalten zu optimieren.
- Nichtlinearität: Bei nichtlinearem Prozeßverhalten soll der Adaptionsalgorithmus in der Lage sein, den Regler möglichst schnell für veränderte Arbeitspunkte zu optimieren.
- Erinnerungsvermögen: Die optimierten Regelparameter sollen in Abhängigkeit der Prozeßarbeitspunkte in einem neuronalen Netz abgespeichert werden.
- Lernen bei festem Arbeitspunkt: Beim Betrieb des Reglers in einem festen Arbeitspunkt soll der Adaptionsalgorithmus den Regler allein anhand der Störcharakteristik des Prozesses optimieren können.

Auf der Grundlage dieser Anforderungen wurde ein adaptiver neuronaler PID-Regler entwickelt, in der Simulation erprobt und mit einem fest eingestellten, herkömmlichen PID-Regler verglichen.

2 Adaptiver neuronaler PID-Algorithmus

Der entwickelte adaptive PID-Regler basiert auf der Kombination eines Multilayer-Perceptron-Netzes (MLP) mit einem klassischen PID-Regelalgorithmus. Kennzeichnend für den Ansatz ist, daß die Struktur des PID-Reglers erhalten bleibt, das neuronale Netz in Abhängigkeit von den Prozeßgrößen, wie z.B. Führungsgröße *w*, Regelgröße *y* und

externen produktionsbedingten Prozeßparametern *p*, die erforderlichen Regelparameter k_p , k_i und k_d bereitstellt [6]:

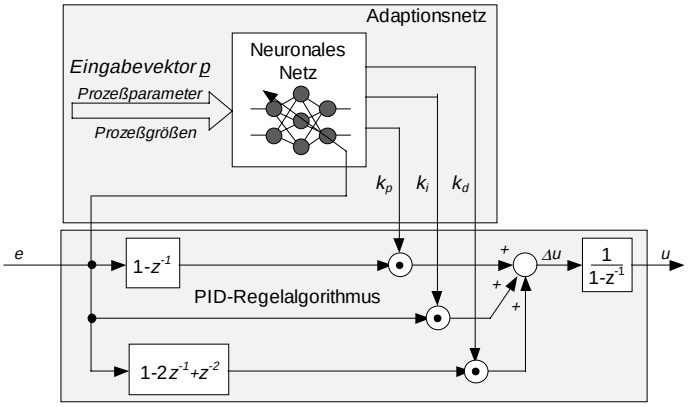

Abbildung 2.1: Neuronaler PID-Regler

Die Funktionsweise dieses adaptiven Ansatzes kann folgendermaßen beschrieben werden:

- Basierend auf der Regelabweichung $e(k)$ berechnet der PID-Regelalgorithmus mit den aktuellen Regelparametern $k_p(k)$, $k_i(k)$ und $k_d(k)$ zum Zeitpunkt k die Stellgröße $u(k)$.
- Die Regelabweichung $e(k)$ wird als Lernsignal zur Adaption der Regelparameter k_p, k_i und k_d verwendet. Das Netz wird durch einen geeigneten Lernalgorithmus so trainiert, daß beim Anlegen des Eingabevektors $\underline{p}(k)$ am Eingang des Netzes die auf der zugrundeliegenden Gütefunktion optimalen Regelparameter ausgegeben werden.
- Der Lernalgorithmus trainiert das neuronale Netz über so viele Abtastzyklen hinweg, bis die Gütefunktion eine vorgegebene Schranke ε unterschreitet. Wird diese Schranke im Verlauf der Prozeßdauer wieder überschritten, so setzt der Adaptionsmechanismus wieder ein, um den Regler für das geänderte Prozeßverhalten zu optimieren.

Die Vorteile dieses Ansatzes gegenüber herkömmlichen neuronalen Reglern sind:

- Die Interpretierbarkeit des Regelverhaltens, da die Regelparameter k_p, k_i und k_d „ablesbar“ sind, es entsteht also kein „Black-Box-Verhalten“.
- Das neuronale Netz kann die optimierten Regelparametereinstellungen der entsprechenden Prozeßzustände (Arbeitspunkte) abspeichern und somit bei sich wiederholenden Arbeitspunkten des Prozesses die optimalen Einstellungen des Reglers ohne erneutes Adaptieren bereitstellen.

3 Kalman-Filter Lernverfahren für Multi-Layer-Perceptron-Netze

Die am häufigsten in der Regelungstechnik eingesetzten Netze sind vorwärts gerichtete, mehrschichtige Netzwerke, wie z.B. das hier verwendete Multi-Layer-Perceptron Netzwerk (MLP) [7]. Das in Abbildung 2.2 dargestellte Netzwerk beinhaltet logarithmisch-sigmoide Neuronen in der Eingabeschicht bzw. lineare Neuronen in der Ausgabeschicht. Der Eingabevektor $\underline{p}(k)$ wird aus den Prozeßgrößen des Systems gebildet, wobei hier zur Vereinfachung lediglich die Führungsgröße $w(k)$ und die Ausgangsgröße $y(k)$ mit jeweils einem vergangenem Wert $w(k\text{-}1)$ und $y(k\text{-}1)$ berücksichtigt werden. Durch Vorwärtspropagierung des Eingabevektors $\underline{p}(k)$ durch das Netz erhält man in Verbindung mit den Gewichten w_{ij} und Bias-Werten b_i des Netzes den Ausgabevektor $\underline{a}^{(2)}(k)$, der aus den drei Regelparameter k_p, k_i und k_d besteht.

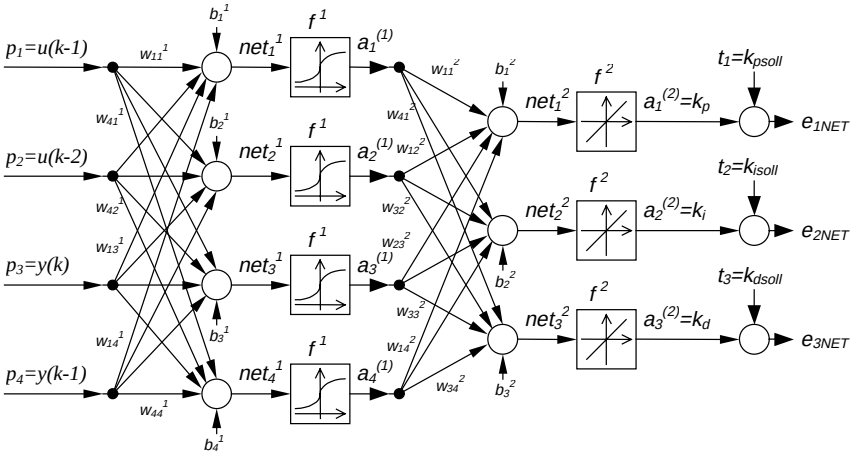

Abbildung 2.2: Neuronales Netz mit 2 Schichten

Die Aufgabe des Lernalgorithmus ist es, anhand des auf der Regelabweichung $e(k)$ basierenden Gütekriteriums die optimalen Reglereinstellungen zu finden und diese im Netz abzuspeichern. Bei der Auswahl eines geeigneten Verfahrens standen folgende Punkte im Vordergrund:

- Bei stochastisch gestörten Prozessen, die sich über einen weiten Zeithorizont in einem stationären Arbeitspunkt befinden, muß das Lernverfahren in der Lage sein, die Regelparameter ausschließlich anhand der Störcharakteristik einzustellen.
- Um ein optimales Störverhalten des Regelkreises zu erhalten, sollen die Regelparameter so eingestellt werden, daß die Varianz der Regelabweichung minimiert wird. Als Gütekriterium wird deshalb die minimale Varianz der Regelabweichung verwendet.

In der Theorie der neuronalen Netze finden sich verschiedenste Lernverfahren für vorwärtsgerichtete Multilayer-Perceptron-Netze (MLP), so z.B. das weit verbreitete Backpropagation- oder das Levenberg-Marquardt-Verfahren [8,9]. Beide Lernalgorithmen basieren auf quadratischen Gütekriterien und sind deshalb nicht in der Lage, den Regler bei einer reinen Störregelung ohne Arbeitspunktänderungen in der gewünschten Weise zu optimieren. Daher kommt hier das relativ unbekannte Kalman-Filter Lernverfahren zum Einsatz, welchem die minimale Varianz der Regelabweichung als Gütefunktion zugrunde liegt [10].

Das vorgestellte Lernverfahren basiert auf den Gleichungen des erweiterten Kalman-Filters (EKF). Zur Darstellung des Lernalgorithmus wird deshalb kurz auf die Grundlagen des Filters eingegangen, weitergehende Erläuterungen zur Theorie des Kalman-Filters finden sich in [11,12].

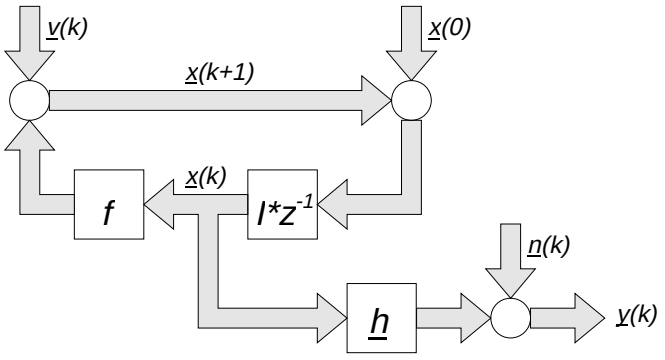

Abbildung 2.3: Markov-Signalprozeß

Das Filter geht von einer Darstellung des Prozesses im Zustandsraum aus, wobei ein Vektorprozeß erster Ordnung, der sogenannte Markov-Signalprozeß, zugrunde liegt (Abbildung 2.3). Das System wird durch den Zustandsgrößenvektor $\underline{x}(k)$, die nichtlineare Systemfunktion $\underline{f}(\underline{x}(k))$, die nichtlineare Ausgangsfunktion $\underline{h}(\underline{x}(k))$ sowie das mittelwertfreie, vektorielle Eingangsrauschen $\underline{v}(k)$ beschrieben. Zudem ist die Messung der Ausgangsgröße $\underline{y}(k)$ mit einem mittelwertfreien, Meßrauschen $\underline{n}(k)$ überlagert. Somit ergeben sich folgende Signalgleichungen zur Beschreibung dieses Prozesses:

$$\underline{x}(k+1) = \underline{f}(\underline{x}(k)) + \underline{v}(k) \tag{3.1}$$

$$\underline{y}(k) = \underline{h}(\underline{x}(k)) + \underline{n}(k) \tag{3.2}$$

Überträgt man die Systemgleichungen des Signalprozesses nun auf das Netz aus Abbildung 2.1, so ergibt sich folgendes:

- Die unbekannten Gewichte w_{ij} und Bias-Werte b_i werden als Zustandsvektor $\underline{x}(t)$ des Systems definiert:

$$\underline{x} = \left[w_{11}^1 \quad w_{12}^1 \quad \Lambda \quad b_1^1 \quad b_2^1 \qquad w_{11}^2 \quad w_{12}^1 \quad \Lambda \quad b_1^2 \quad b_2^2 \right]^T \tag{3.3}$$

- Die Systemfunktion $\underline{f}(\underline{x}(k))$ des Filters ist die Identitätsfunktion, das Eingangsrauschen $\underline{v}(k)$ entfällt:

$$\underline{x}(k+1) = \underline{f}(\underline{x}(k)) + \underline{v}(k) = \underline{x}(k) \tag{3.4}$$

- Die nichtlineare, zeitvariante Ausgangsfunktion $\underline{h}(\underline{x}(k))$ beschreibt die Gewichtsstruktur des Netzes in Verbindung mit dem Eingabevektor $\underline{p}(k)$:

$$\underline{a}(k) = \underline{h}(\underline{x}(k), \underline{p}(k)) \tag{3.5}$$

- Der Ausgabevektor $\underline{y}(k)$ des Systems entspricht dem Zielausgabevektor $\underline{t}$ des Netzes, das Meßrauschen $\underline{n}(k)$ kann als Modellfehler $\underline{e}_{NET}(k)$ interpretiert werden.

$$\underline{y}(k) = \underline{t}(k) \tag{3.6}$$

$$\underline{t}(k) = \underline{h}(\underline{x}(k)) + \underline{n}(k) \tag{3.7}$$

$$\underline{t}(k) = \underline{a}^{(2)}(k) + \underline{e}_{NET}(k) \tag{3.8}$$

Werden die abgeleiteten Signalgleichungen des neuronalen Netzes auf die Gleichungen des erweiterten Kalman-Filters übertragen, so ergeben sich folgende rekursive Beziehungen:

- Der neu geschätzte Zustandsgrößenvektor (d.h. die Gewichte) ergibt sich rekursiv aus der Korrekturmatrix $\underline{K}$ sowie dem Ausgabefehlervektor $\underline{e}_{NET}(k)$:

$$\hat{\underline{x}}(k+1) = \hat{\underline{x}}(k) + \underline{K}(k+1) \cdot \left[\underline{e}_{NET}(k) \right] \tag{3.9}$$

- Der Ausgabefehlervektor $\underline{e}_{NET}(k)$ des Netzes wird durch Anwendung der Kettenregel im Bezug auf die Regelabweichung $e(k)$ angegeben. Der Parameter μ kann als Lernrate des Verfahrens aufgefaßt werden:

$$\underline{e}_{NET}(k) = \begin{bmatrix} e(k) - e(k-1) \\ e(k) \\ e(k) - 2 \cdot e(k-1) + e(k-2) \end{bmatrix} \cdot \mu \cdot e(k) \tag{3.10}$$

- Die Korrekturmatrix $\underline{K}$ (Kalman-Verstärkungsmatrix) wird so berechnet, daß die Varianz des Schätzfehlers minimal wird. Dazu bestimmt man die Kovarianzmatrix $\underline{N}$ des Meßrauschens und die Ableitung der vektoriellen Ausgangsfunktion $\underline{h}$ nach dem Zustandsgrößenvektor $\underline{x}$:

$$\underline{N}(k+1) = \underline{N}(k) + \frac{1}{k} \cdot \left(\underline{e}_{NET}{}^{T} (\underline{x}(k)) \cdot \underline{e}_{NET} \ (\underline{x}(k)) - \underline{N}(k) \right) \tag{3.11}$$

$$\underline{H}(k+1) = \left(\frac{\partial \underline{a}^{(2)}(\underline{x}(k))}{\partial \underline{x}(k)} \right) \tag{3.12}$$

$$\underline{K}(k+1) = \underline{P}(k) \cdot \underline{H}(k+1)^{T} \cdot \left[\underline{H}(k+1) \cdot \underline{P}(k) \cdot \underline{H}(k+1)^{T} + \underline{N}(k+1) \right]^{-1} \tag{3.13}$$

- Die Kovarianzmatrix $\underline{P}$ des Schätzfehlers berechnet sich rekursiv aus folgender Gleichung:

$$\underline{P}(k+1) = \underline{P}(k) - \underline{K}(k+1) \cdot \underline{H}(k+1) \cdot \underline{P}(k) \tag{3.14}$$

Mit Hilfe der hergeleiteten Gleichungen wurde der neuronale PID-Regler implementiert und in der Simulation erprobt. Weitergehende Ausführungen zur Theorie des Regelalgorithmus sind in [13] dargestellt.

4 Anwendung des neuro-adaptiven Reglers auf einen verfahrenstechnischen Prozeß

Zur Simulation der Funktion des entwickelten Konzeptes wurde der Algorithmus auf einen verfahrenstechnischen Prozeß in der Glasindustrie angewendet. Eine Beschreibung des Verfahrens wird in [14] vermittelt. Bei dem betrachteten Prozeß handelt es sich um einen stark gestörtes, nichtlineares System mit a priori unbekanntem Streckenverhalten. Abbildung 4.1 zeigt den simulierten Regelkreis mit der Ausgangsgröße *y* der Führungsgröße *w* , der Stellgröße *u* sowie dem produktionsbedingten Prozeßparameter *p*.

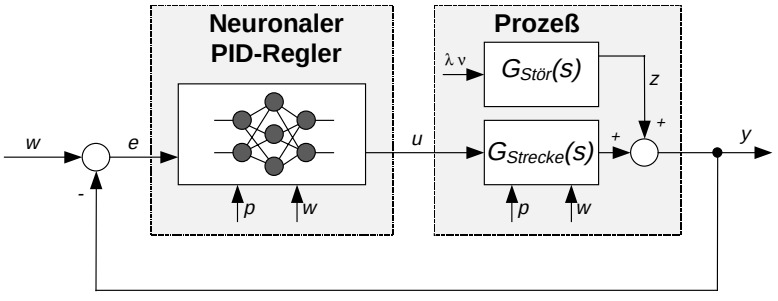

Abbildung 4.1: Simulierter Regelkreis

Das Streckenverhalten kann als ein Verzögerungsglied erster Ordnung angenommen werden, wobei für die Streckenparameter $k_{Strecke}$ und $T_{Strecke}$ folgender arbeitspunktabhängiger nichtlinearer Zusammenhang gilt:

$$G_{Strecke}(s) = \frac{k_{Strecke}(w)}{1 + T_{Strecke}(p) \cdot s} \tag{4.1}$$

$$k_{Strecke}(w) = -(0.5 + \log(w)), \quad w = 1\text{K } 1.5, \tag{4.2}$$

$$T_{Strecke}(p) = (2\,\text{sec} + 24\,\text{sec} \cdot p), \quad p = 1\text{K } 1.5 \tag{4.3}$$

Mit Hilfe dieses Models der Strecke wird in der Simulation die Funktion des Regelalgorithmus überprüft. Um die Adaptionsfähigkeit des Algorithmus darzustellen, werden die zu Beginn der Simulation verwendeten Regelparametereinstellungen zufällig gewählt, d.h. dem Regler ist das Streckenverhalten unbekannt. Abbildung 4.2 zeigt den Simulationsverlauf der Führungsgröße *w*, des externen Prozeßparameters *p* sowie die adaptierten Regelparameter k_p, k_i und k_d. Durch die Änderung der Führungsgröße *w* und des Prozeßparameter *p* wurde das Streckenverhalten *G*(*s*) verändert. Betrachtet man den Verlauf der Regelparameter, so zeigt sich, daß der Adaptionsmechanismus des Reglers die Regelparameter ändert, um ein optimales Regelverhalten zu erzielen.

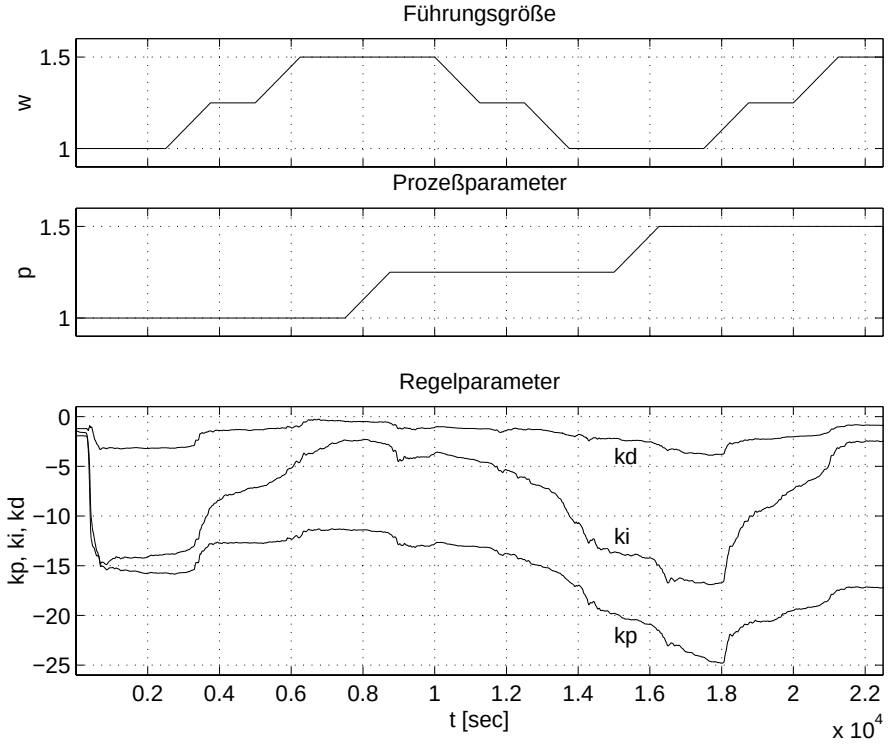

Abbildung 4:2: Verlauf der Prozeßgrößen sowie der Regelparameter

Der so trainierte Regler ist nun in der Lage für die jeweiligen Arbeitspunkte des Prozesses die optimalen Regelparameter zu verwenden. Die Abbildung 4.3 veranschaulicht das

Ergebnis nochmals anhand der sich einstellenden Kennfelder der Regelparameter in Abhängigkeit der Arbeitspunkte.

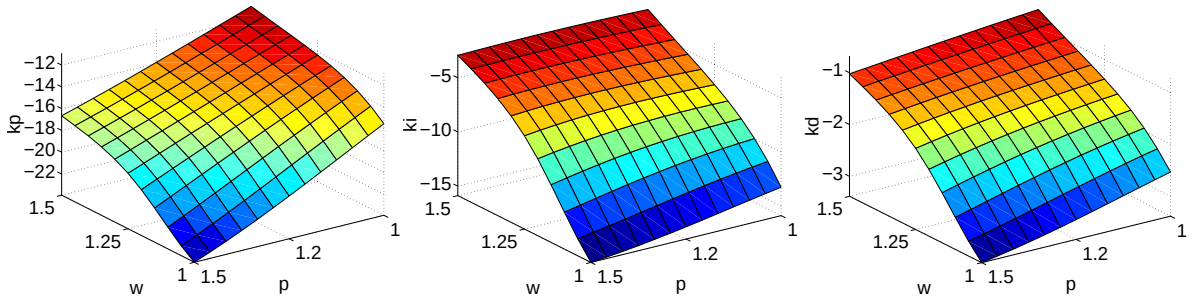

Abbildung 4.3: Kennfelder der Regelparameter *kp*, *ki* und *kd*

Zur Beurteilung des Stör- bzw. Führungsverhaltens des neuronalen PID-Reglers wird dieser mit einem fest eingestellten PID-Regler verglichen. Abbildung 4.4 zeigt zur Veranschaulichung des Störverhaltens die Varianz der Regelgröße *y*. Zu Beginn der Simulation verwenden beide Regler dieselben Regelparameter und erreichen somit auch dieselbe Regelgüte. Nach 1000 Abtastschritten wurden die Führungsgröße *w*, der Prozeßparameter *p* und somit das Streckenverhalten *G*(*s*) verändert. Durch die Adaptionskomponente paßt sich der neuronale PID-Regler an das sich ändernde Streckenverhalten an, was im Vergleich zu dem fest eingestellten PID-Regler in einem besseren Störverhalten des Regelkreises resultiert. Angemerkt sei hier, daß sich die Varianz *var*(*y*) während der Änderung der Arbeitspunkte bei beiden Reglern stark erhöht. Die Ursache dafür liegt in der Änderung der Führungsgröße *w*, sowie in dem relativ langen Zeithorizont zur Ermittlung der Varianz (150 Abtastschritte)

Das unterschiedliche Führungsverhalten des neuronalen PID-Reglers bzw. des fest eingestellten PID-Reglers ist in Abbildung 4.5 dargestellt. Auch hier wurde zu Beginn der Simulation von der selben Einstellung der Regelparameter beider Regler ausgegangen. Die Änderung der Führungsgröße *w* bewirkt eine Änderung des Streckenverhaltens *G*(*s*), der neuronale PID-Regler adaptiert die Regelparameter, und erzielt somit ein optimales Führungsverhalten. Im Gegensatz dazu führt die feste Regelparametereinstellung des herkömmlichen PID-Reglers zu einem instabilen Regelverhalten.

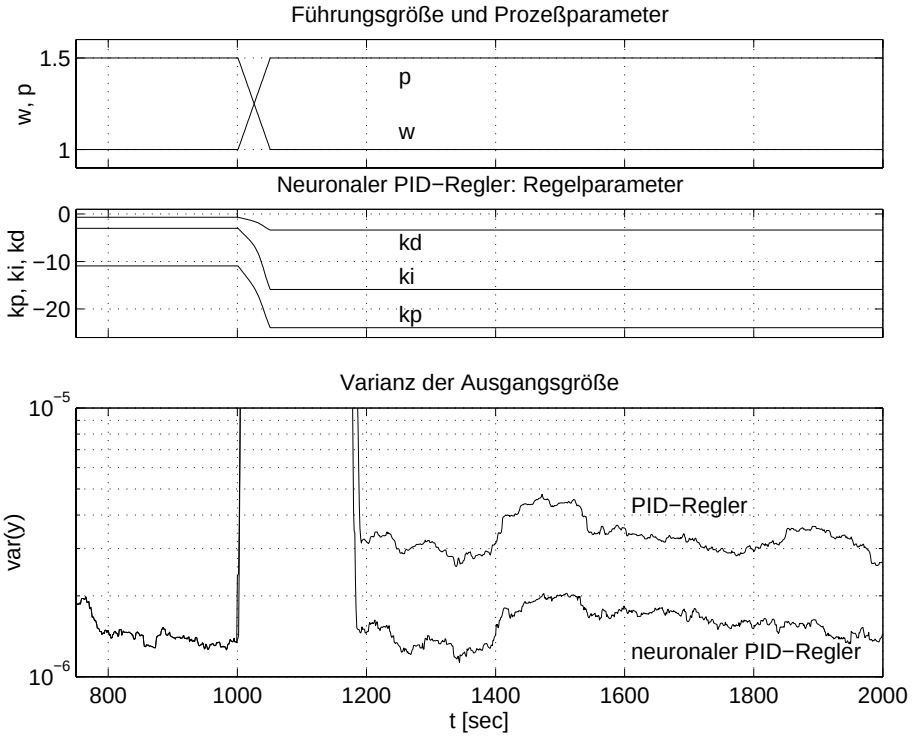

Abbildung 4.4: Störverhalten des Regelkreises

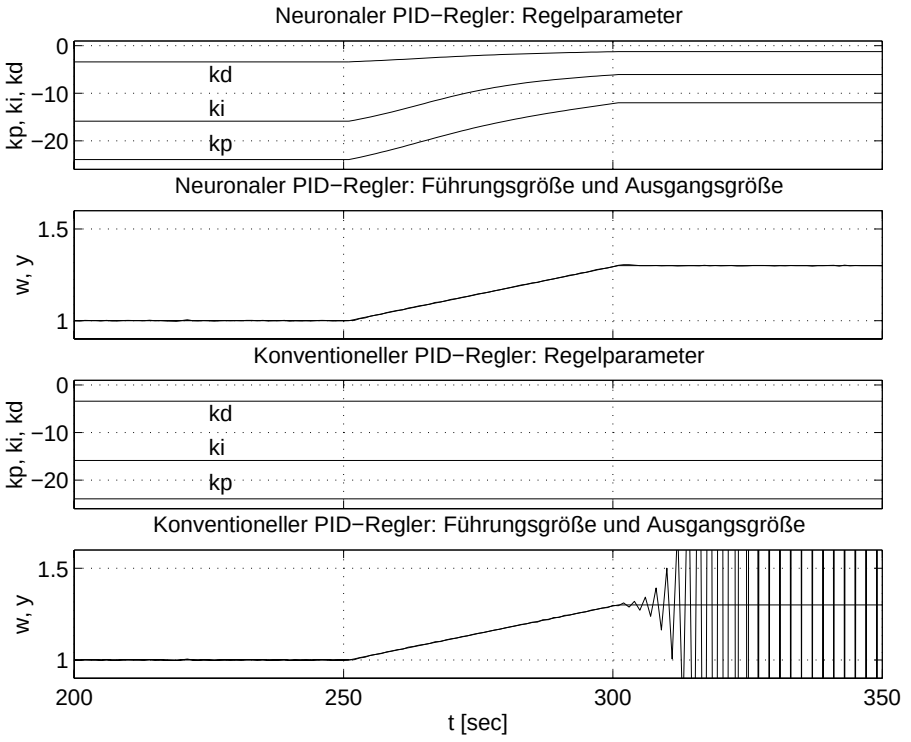

Abbildung 4.5: Führungsverhalten des Regelkreises

5 Zusammenfassung

Das vorgestellte neuro-adaptive Regelungskonzept basiert auf der Kombination eines neuronalen Netzes mit einem klassischen PID-Regelalgorithmus. Kennzeichnend ist, daß die Struktur des PID-Reglers erhalten bleibt und das neuronale Netz in Abhängigkeit von den Prozeßgrößen die optimalen Reglereinstellungen generiert. Als Lernalgorithmus des Netzes wurde das in diesem Zusammenhang bisher kaum beachtete Kalman-Filter Lernverfahren implementiert.

Die bei der Simulation des Reglerkonzeptes in Verbindung mit einem stark gestörten nicht-linearen Prozeß gewonnenen Ergebnisse lassen erkennen, daß der entwickelte Ansatz

- bei rein stochastischer Anregung des Prozesses in der Lage ist, die Regelparameter zu adaptieren und diese in dem neuronalen Netz abzuspeichern,
- arbeitspunktabhängige Nichtlinearitäten erfassen kann, und die jeweiligen optimalen Reglereinstellungen dazu bereitstellt,
- sich an ein veränderliches Prozeßverhalten on-line anpassen kann,
- das Stör- bzw. Führungsverhalten des Prozesses optimiert, und im Vergleich zu einem fest eingestellten herkömmlichen PID-Regler eine wesentlich bessere Regelgüte erzielt

Literatur

[1] Peter Firsching: „Der Einsatz künstlicher neuronaler Netze in Regelungstechnischen Anwendungen“, Dissertation, Technische Fakultät der Universität Erlangen-Nürnberg, 1995

[2] K.J., Aström, und Wittenmark, B.: „Adaptive Control“, Addison-Wesley Pub. Co., Massachusetts 1989

[3] Rolf Isermann: „Stand und Entwicklungstendenzen bei adaptiven Regelungen“, atp-Automatisierungstechnische Praxis (29) 1987

[4] P. J. Antsaklis, K. M. Passino: „an introduction to intelligent and autonomous control“, Kluwer Academic Publishers, 1993

[5] M. Brown, C. Harris: „neurofuzzy adaptive modelling and control“, Prentice Hall International, 1994

[6] Saiful Akhyar, Sigeru Omatu: „Neuromorphic Self-Tuning PID-Controller“, IEEE International Conference on Neural Networks, San Farnsisco 1993

[7] Oliver Nelles, Susanne Ernst, Rolf Iserman: „Neuronale Netze zur Identifikation nichtlinearer, dynamischer Systeme: Ein Überblick“, at-Automatisierungstechnik (45) 1997

[8] Detlef Nauck, Frank Klawonn, Rudolf Kruse: „Neuronale Netze und Fuzzy-Systeme“, Vieweg 1994

[9] Martin T. Hagan, Mohammad B. Menhaj: „Training Feedforward Networks with the Marquardt Algorithm“, IEEE Transactions on Neural Networks, Vol. 5, No. 6, 1994

[10] Youji Iiguni, Hideaki Sakai: „A Real-Time Learning Algorithm for Multilayerd Neural Network, Based on the Extended Kalman-Filter“, IEEE Transactions on Signal Processing, Vol. 40, No.4, 1992

[11] Rolf Isermann: „Digitale Regelsysteme“, Band I+II. Springer-Verlag, 1987

[10] K. Brammer, G. Siffling: „Kalman-Bucy-Filter“, Oldenburg, München, 1985

[11] Christian W. Frey: „Untersuchungen zur On-Line-Optimierung einer Fuzzy-Logic-Regelung eines verahrenstechnischen Prozesses mit Hilfe eines Neuronalen Netzes“, Diplomarbeit am Fraunhofer-Institut IITB in Karlsruhe, 1995

[12] Markoto Sajidman, Helge-Björn Kuntze: „Integration of Fuzzy Control and Model Based Concepts for Disturbed Industrial Plants with Large Dead-Times“, Sixth IEEE International Conference on Fuzzy Systems in Barcelona, July 1-5, 1997