Gleitzustandstraining Höherer Ordnung für die Adaptive Flugregelung eines Unbemannten Flugsystems

Philipp Schnetter, Jonas Kaste, Thomas Krüger
Institut für Luft- und Raumfahrtsysteme - Technische Universität Braunschweig
Hermann-Blenk-Str. 23, 38108 Braunschweig, Deutschland

Zusammenfassung

Im Rahmen des Forschungsprojekts *Bürgernahes Flugzeug* am *Campus Forschungsflughafen* in Braunschweig forscht das Institut für Luft- und Raumfahrtsysteme an adaptiven Flugregelungsverfahren. Ein wohl untersuchter Ansatz ist dabei die Erweiterung eines Reglers auf Basis der modellbasierten dynamischer Inversion um künstliche neuronale Netzwerke. Sollte es auf Grund vorliegender Modellunsicherheiten zu Inversionsfehlern gegenüber der tatsächlichen Regelungsstrecke kommen, wird die universelle Approximationseigenschaft eines neuronalen Netzwerks genutzt, um diesen Fehler zu adaptieren und ihn so zu kompensieren. Zur Sicherstellung eines transparenten Lernprozesses wird ein von der strukturvariablen Regelung abgeleitetes Lernverfahren verwendet. Dieses sogenannte Gleitzustandslernverfahren als Erweiterung des klassischen Gradientenabstiegs betrachtet das neuronale Netz als abstraktes Regelungssystem. Diese Betrachtung ermöglicht mit Hilfe von Schaltfunktionen eine transparente Gewichtsänderung und birgt zusätzlich den Vorteil einer dynamischen Bestimmung der optimalen Lernrate innerhalb stabiler Grenzen. In dieser Arbeit soll der Gleitzustand höherer Ordnung vorgestellt werden, welcher die Konvergenzgeschwindigkeit der trainierten Netzwerke weiter beschleunigt und auf diese Weise die Robustheit gegenüber Inversionsfehlern und nicht zuletzt auch Systemschäden weiter erhöht. Zur Analyse der Netzwerkeigenschaften wird die automatische Bahnfolge eines unbemannten Flugsystems unter dem Einfluss atmosphärischer Störungen und Modellunsicherheiten simuliert. Die Ergebnisse zeigen, dass die Erweiterung durch den Ansatz der Gleitzustandsregelung höherer Ordnung einen stabilen und robusten Flug trotz erheblicher Modellunsicherheiten gewährleistet.

1 Einleitung

Getrieben durch das stetige Wachstum der Anwendungsmöglichkeiten von unbemannten Flugsystemen (engl. Unmanned Aircraft Systems - UAS) im sowohl kommerziellen als auch wissenschaftlichen Sektor, steigen auch die Anforderungen an eine vollständig automatisch durchgeführte Flugmission. Immer mehr Anwendungen benötigen eine hoch präzise Bahnfolge und somit ein leistungsstarkes Flugregelungskonzept für die erfolgreiche Durchführung der Mission. Doch gerade auf Grund der oftmals geringen Größe und des durch Leichtbaumethoden reduzierten Gewichts, sind kleine UAS sehr Störanfällig gegenüber Wind und Turbulenz. Klassische Ansätze wie das *gain scheduling* für ausgewählte Arbeitspunkte von Linearreglern [1] verlassen unter Einfluss starker atmosphärischer Störungen jedoch ihren robusten und stabilen Auslegungsbereich, wodurch die Regelqualität deutlich verringert wird. Zu diesem Zweck wird gerade auf dem Gebiet der kleinen Flugsysteme vermehrt auf nichtlineare Regelungsansätze, wie der modellbasierten dynamischen Inversion mit der Erweiterung um künstliche neuronale Netzwerke (KNN), zurückgegriffen [2, 3, 4, 5]. Die Lerneigenschaften künstlicher neuronaler Netze bieten in diesem Zusammenhang die Möglichkeit das Regelungssystem unter Echtzeitbedingungen an nicht modellierte Störungen und Dynamiken sowie Systemschäden anzupassen.

Der vorgestellte Ansatz eines Reglers auf Basis der dynamischen Inversion, auch Eingangs-/Ausgangslinearisierung genannt, in Kombination mit künstlichen neuronalen Netzwerken ist in den letzten Jahren umfangreich untersucht und in Flugtests validiert worden [6, 3, 4, 5, 7]. Dabei wird oftmals der Backpropagation Algorithmus für die Approximation des bestehenden Modell- oder hier Inversionsfehler durch ein KNN verwendet [8, 9]. Nachteilhaft ist dabei die meist sim Sinne der Stabilität konservative Wahl einer geeigneten Lernrate, um Oszillationen des Trainingsverfahrens und letztendlich des Netzwerkausgangs zu vermeiden, was eine Reduktion der Konvergenzgeschwindigkeit zur Folge hat. Die so verlangsamte Fehlerapproximation des Netzwerks reduziert wiederum die Robustheit des adaptiven Regelungskonzepts. Um diesen Effekt zu vermeiden, können alternative Lernverfahren wie das Sliding Mode Training genutzt werden [10, 11, 12, 6, 2]. Durch die Betrachtung des KNN als zu regelndes System mit dem Netzwerkfehler und dessen Ableitungen als Zustandsgrößen, können Algorithmen aus dem Bereich der strukturvariablen Regelung für das Training verwendet werden. Durch die Wahl geeigneter Schaltgesetze kann das Netzwerk in einen sogenannten Gleitzustand (engl. Sliding Mode) überführt werden, was zum einen Robustheit gegenüber Störungen garantiert und zum anderen die dynamische Berechnung der Lernrate innerhalb stabiler Grenzen erlaubt. Abhängig von den Zustandsgrößen des Netzwerks können verschiedene Ordnungen des Gleitzustands vorgegeben werden und auf ihre Auswirkung auf das Netzwerktraining hin untersucht werden. Um bereits vor dem Eintreffen der Zustände im Gleitzustand die Dynamik des Netzwerkfehlers zu beeinflussen, können sogenannte

Reaching Laws definiert werden. In Abhängigkeit der gewählten Schaltfunktion wird so die Erreichbarkeit einer vorgegebenen Ordnung des Gleitzustands garantiert.

Die hier vorgestellten Ergebnisse sind zunächst in einer Testumgebung für das Training neuronaler Netzwerke und anschließend mit dem unbemannten Kleinflugzeug *CAROLO P360* und der dazugehörigen Simulationsumgebung entstanden. Das Fluggerät ist in Abb. 1 dargestellt. Das UAS hat ein Abfluggewicht von maximal 25kg, eine Spannweite von 360 cm und kann mit elektrischem Antrieb ca. 45 Minuten bei einer Reisegeschwindigkeit von 23 m/s operieren. Die Simulation nutzt die nichtlinearen Bewegungsgleichungen, identifizierte Aktuator- und Sensormodelle, ein komplett modelliertes integriertes Navigationssystemund ein Atmosphärenmodell mit einem Dryden-Turbulenzspektrum [1] wie es in [6] beschrieben ist.

Abbildung 1: Unbemanntes Flugsystem *CAROLO P360*.

2 Theorie des Regelungsverfahrens

2.1 Nichtlineare dynamische Inversion

Das grundlegende Prinzip der dynamischen Inversion ist es, einem nichtlinearen System mit Hilfe einer Koordinatentransformationen sowie einer Zustandsrückführung ein lineares Eingangs-/Ausgangsverhalten aufzuprägen [13, 14]. Der Übersichtlichkeit halber sei ein nichtlinearen Eingrößensystem (engl. single-input-single-output - SISO) mit Affinität in der Steuerung u gegeben:

$$\begin{aligned} \dot{\vec{x}} &= f(\vec{x}) + g(\vec{x}) \cdot u \,, \\ y &= h(\vec{x}) \,. \end{aligned} \tag{1}$$

Dabei sind f and g hinreichend glatte Vektorfelder im Wertebereich $D \subset \mathbf{R}^n$. Für die Inversion des Eingangs-/Ausgangsverhaltens bedarf es einer direkten Abhängigkeit der Systemantwort y von der Steuergröße u. Liegt dieser Zusammenhang nicht in der Ausgangsgleichung vor, muss diese abhängig vom relativen Grad r-fach abgeleitet werden, bis die Steuergröße auf der rechten Seite der Gleichung auftaucht [13]. Zu diesem Zweck kann die Ableitung von y zu

$$y^{(r)} = L_f^r h(\vec{x}) + L_g L_f^{r-1} h(\vec{x}) \cdot u \tag{2}$$

bestimmt werden. Der Ausdruck $\frac{\partial h}{\partial \vec{x}} \cdot f(\vec{x}) = L_f h(\vec{x})$ wird als Lie-Ableitung bezeichnet. Für ein Mehrgrößensystem (engl. multiple-input-multiple-output - MIMO) mit m Ausgängen und einem vektoriellen relativen Grad $\{r_1, \ldots, r_m\}$ kann Gleichung (2) umgeschrieben werden zu:

$$\vec{y}^{r} = b(\vec{x}) + \mathbf{A}(\vec{x}) \cdot \vec{u} \,, \tag{3}$$

mit

$$\begin{aligned} \mathbf{A}(\vec{x}) &= \begin{bmatrix} L_{g1} L_f^{r_1-1} h_1(x) & \ldots & L_{gm} L_f^{r_1-1} h_1(x) \\ L_{g1} L_f^{r_2-1} h_2(x) & \ldots & L_{gm} L_f^{r_2-1} h_2(x) \\ \vdots & \ddots & \vdots \\ L_{g1} L_f^{r_m-1} h_m(x) & \ldots & L_{gm} L_f^{r_m-1} h_m(x) \end{bmatrix}, \\ b(\vec{x}) &= \begin{bmatrix} L_f^{r_1} h_1(x) \\ L_f^{r_2} h_2(x) \\ \vdots \\ L_f^{r_m} h_m(x) \end{bmatrix} . \end{aligned} \tag{4}$$

Wenn $\mathbf{A}$ an der Stelle $x_0 = 0$ nicht singulär ist, kann mit Hilfe der Inversion der Systemdynamik ein linearisierender Ausdruck für das Steuersignal gefunden werden

$$\vec{u} = \mathbf{A}^{-1}(\vec{x}) \cdot \left[\vec{v} - b(\vec{x}) \right] \,. \tag{5}$$

Dieser eingesetzt in Gleichung (3) reduziert das ursprünglich nichtlineare System zu einer linearen Integratorkette zwischen der r-ten Ableitung des Ausgangs und der Eingangsgröße des invertierten Systems, der sogenannten Pseudosteuergröße

$$\left[y_i^{(r_i)} \right]_{m \times 1} = \vec{v}_i \,. \tag{6}$$

Es ist wichtig zu beachten, dass die Aktuatordynamik des Systems, zur Reduzierung des relativen Grades, nicht mit invertiert wird. Um dennoch Nichtlinearitäten aufgrund von Begrenzungen zu verhindern, existieren verschiedene Verfahren zum Schutz vor beispielsweise Integrator-windup und Anpassung von neuronalen Netzwerken an gesättigte Aktuatordynamiken [15, 16]. Bei der in dieser Arbeit verwendeten Methode handelt es sich dabei um das sogenannte Pseudo-Control Hedging (PCH), bei dem die Referenzmodelle in den Regelschleifen soweit herab gebremst werden, dass sie nur noch Signale ausgeben, denen das beschränkte Gesamtsystem folgen kann [5, 17].

2.2 Fehlerdynamik

Modellungenauigkeiten, wie sie häufig bei kostengünstigen Identifikationsmethoden für kleine UAS der Fall sind, oder auch Degradationen und Systemschäden führen zwangsläufig zu Inversionsfehlern. Diese reduzieren die Leistungsfähigkeit des Regelungsansatzes. Unter dieser Annahme, dass $\mathbf{A}(\vec{x}) \neq \widehat{\mathbf{A}}(\vec{x})$ und $b(\vec{x}) \neq \widehat{b}(\vec{x})$ gilt, folgt entsprechend

$$\vec{u} = \widehat{\mathbf{A}}^{-1}(\vec{x}) \cdot \left[\vec{v} - \widehat{b}(\vec{x}) \right] \tag{7}$$

und somit

$$\vec{y}^{(r)} = b(\vec{x}) + \mathbf{A}(\vec{x}) \cdot \widehat{\mathbf{A}}^{-1}(\vec{x}) \cdot \left[\vec{v} - \widehat{b}(\vec{x}) \right] \neq \vec{y}_R^{(r)} \,. \tag{8}$$

Hierbei sind die fehlerbehafteten Modellgrößen mit $\widehat{\ }$ markiert. Die in Gleichung (8) dargestellte Differenz zwischen

der r-ten Ableitung des Systemausgangs $\vec{y}^{(r)}$ und des Referenzmodellausgangs $\vec{y}_R^{(r)}$ wird als Inversionsfehler bezeichnet:

$$\Delta = y^{(r)} - y_R^{(r)} \,. \tag{9}$$

Zusätzlich zum Inversionsfehler kann ein Regelfehler angegeben werden. Dieser setzt sich aus den $k \leq r-1$ Ableitungen von $e(t) = y - y_R$ zusammen und wird in einem Fehlerzustandsvektor zusammengefasst:

$$\vec{\chi}(t) = \begin{bmatrix} e(t) \\ \dot{e}(t) \\ \vdots \\ e^{(r-1)} \end{bmatrix} \,. \tag{10}$$

In der Form einer Zustandsraumdarstellung, mit dem Inversionsfehler als deren Eingangsgröße, gilt nun für die Fehlerdynamik [5]:

$$\dot{\vec{\chi}} = \mathbf{A}_E \vec{\chi} + \vec{b}_E \Delta \,. \tag{11}$$

Dabei ist $\mathbf{A}_E$ die Hurwitz Matrix und enthält die positiven Reglerverstärkungen, sodass Stabilität der Fehlerdynamik in einem Ljapunow Ansatz garantiert werden kann. Es folgt für die Ljapunow Gleichung

$$\mathbf{A}_E^T \mathbf{P}_E + \mathbf{P}_E \mathbf{A}_E = -\mathbf{Q}_E \,, \tag{12}$$

wobei $\mathbf{P}_E$ und $\mathbf{Q}_E$ positiv sein müssen.

2.3 Einsatz neuronaler Netzwerke

Die universale Approximationsfähigkeit neuronaler Netzwerke [8, 18] wird verwendet um sich an resultierenden Inversionsfehler Δ zu adaptieren und so auszugleichen. Auf diese Weise können modellbasierte Regelungsverfahren selbst für nur ungenau bestimmte Modelle kleiner UAS verwendet werden. Eine für diese Aufgabe oftmals verwendete Netzwerktopologie besitzt eine sigmoide Transferfunktion in der einen versteckten Netzwerkschicht und lineare Funktionen in der Ein- und Ausgangsschicht [17, 19, 20, 21, 4]. Darüber hinaus handelt es sich um ein Netzwerk mit nur einer Ausgangsgröße. Mit dem Eingangsvektor $\underline{\vec{x}}$, den Gewichtsmatrizen $\mathbf{w}^{(1)}$ und $\mathbf{w}^{(2)}$ sowie der Transferfunktion der versteckten Schicht $\vec{f}^{(2)}$ ergibt sich der Netzwerkausgang in vektorieller Schreibweise zu:

$$\vec{y} = \mathbf{w}^{(2)} \cdot \vec{f}^{(2)}\left(\mathbf{w}^{(1)} \underline{\vec{x}}\right) \,. \tag{13}$$

Das Wissen des Netzwerks ist in den Gewichten gespeichert und kann zu einer generellen Gewichtsmatrix $\mathbf{w}$ zusammengefasst werden. Um das Netzwerk zu trainieren und so den Inversionsfehler zu adaptieren, werden die Gewichte in Abhängigkeit ihres Einflusses auf den Netzwerkfehler angepasst. Für den oftmals verwendeten gradientenbasierten Backpropagation-Algorithmus kann diese Gewichtsanpassung mit dem quadratischen Fehler E wie folgt beschrieben werden:

$$\Delta \mathbf{w} = -\mu \frac{\partial E}{\partial \mathbf{w}} \,. \tag{14}$$

Die Schrittweite des Verfahrens wird durch die Lernrate μ vorgegeben. Diese beeinflusst nicht ausschließlich die Konvergenzgeschwindigkeit, sondern auch maßgeblich die Stabilität der Methode. Zu hohe Lernraten können zu Oszillationen führen und sind deswegen zu vermeiden [6]. Mit der Annahme optimaler Gewichte (markiert durch den Index $*$) bei denen der Approximationsfehler $\vec{\varepsilon}$ minimal ist, folgt für die Adaption des Inversionsfehlers:

$$\vec{\Delta} = \mathbf{w}_*^{(2)} \cdot \vec{f}^{(2)}\left(\mathbf{w}_*^{(1)} \underline{\vec{x}}\right) + \vec{\varepsilon}\left(\underline{\vec{x}}\right) \,. \tag{15}$$

Wobei $\|\varepsilon\| < \overline{\varepsilon}$ gilt und $\overline{\varepsilon}$ begrenzt ist. Der Einfluss der neuronalen Netzwerke auf die Fehlerdynamik wird nun anhand der um den Netzwerkausgang ν_{ad} und den robustifizierenden Term ν_r erweitere Gleichung (11) beschrieben:

$$\dot{\vec{\chi}} = \mathbf{A}_E \cdot \vec{\chi} + \mathbf{b}_E \left(\Delta - \nu_{ad} - \nu_r \right) \,. \tag{16}$$

Durch die Definition einer Gewichtsfehlermatrix $\widetilde{\mathbf{w}}^{(1)} = \mathbf{w}^{(1)} - \mathbf{w}_*^{(1)}$ und einer Taylorreihenentwicklung der verdeckten Schicht mit Abbruch nach dem ersten Glied ergibt sich der finale Ausdruck der Fehlerdynamik zu:

$$\begin{aligned} \dot{\vec{\chi}} =& \mathbf{A}_E \cdot \vec{\chi} + \mathbf{b}_E \Big[\mathbf{z} - \widetilde{\mathbf{w}}^{(2)} \Big[\vec{f}\left(\mathbf{w}^{(1)} \underline{\vec{x}}\right) - \mathbf{f}'(\mathbf{w}^{(1)} \underline{\vec{x}}) \mathbf{w}^{(1)} \underline{\vec{x}} \Big] \\ & - \mathbf{w}^{(2)} \mathbf{f}'(\mathbf{w}^{(1)} \underline{\vec{x}}) \widetilde{\mathbf{w}}^{(1)} \underline{\vec{x}} - \nu_r \Big] \,. \end{aligned} \tag{17}$$

Hier ist $\mathbf{z}$ ein nach oben begrenzter Störterm der Netzwerklinearisierung [22, 5, 4]. Es gilt:

$$\mathbf{z} = \varepsilon\left(\underline{\vec{x}}\right) - \widetilde{\mathbf{w}}^{(2)} \mathbf{f}'(\mathbf{w}^{(1)} \underline{\vec{x}}) \, \mathbf{w}_*^{(1)} \underline{\vec{x}} + \mathbf{w}_*^{(2)} \vec{O}\left[\widetilde{\mathbf{w}}^{(1)} \underline{\vec{x}}\right]^2 . \tag{18}$$

Durch die geeignete Wahl des Lerngesetztes kann die Stabilität dieser Dynamik garantiert werden. Für die Gewichtsänderung in Abhängigkeit des gefilterten Fehlerterms $\vec{\zeta} = \vec{\chi}^T \mathbf{P}_E \mathbf{b}_E$ muss gelten:

$$\dot{\mathbf{w}}^{(1)} = \Gamma^{(1)} \cdot \left[\underline{\vec{x}} \cdot \vec{\zeta} \cdot \mathbf{w}^{(2)} \cdot \mathbf{f}'(\mathbf{w}^{(1)} \underline{\vec{x}}) - \lambda \cdot \|\vec{\zeta}\|_2 \cdot \mathbf{w}^{(1)^T} \right], \tag{19}$$

$$\dot{\mathbf{w}}^{(2)} = \Gamma^{(2)} \cdot \left[\vec{f}\left(\mathbf{w}^{(1)} \underline{\vec{x}}\right) \cdot \vec{\zeta} - \mathbf{f}'(\mathbf{w}^{(1)} \underline{\vec{x}}) \cdot \mathbf{w}^{(1)} \underline{\vec{x}} \cdot \vec{\zeta} - \lambda \cdot \|\vec{\zeta}\|_2 \cdot \mathbf{w}^{(2)^T} \right]. \tag{20}$$

Zur Stabilitätsuntersuchung wird eine geeignete Ljapunow-Kandidatenfunktion gewählt [4, 5, 17]:

$$\begin{aligned} V(\vec{\chi}) =& \frac{1}{2} \vec{\chi}^T \mathbf{P}_E \vec{\chi} + \frac{1}{2} \mathrm{trace}\left[\widetilde{\mathbf{w}}^{(1)} \Gamma^{(1)^{-1}} \widetilde{\mathbf{w}}^{(1)^T} \right] \\ & + \frac{1}{2} \mathrm{trace}\left[\widetilde{\mathbf{w}}^{(2)} \Gamma^{(2)^{-1}} \widetilde{\mathbf{w}}^{(2)^T} \right]. \end{aligned} \tag{21}$$

Abschließend ergibt sich für die Ableitung von $V(\vec{\chi})$ mit Einsetzen der Gewichtsänderungen aus (19) und (20):

$$\begin{aligned} \dot{V}(\vec{\chi}) =& -\frac{1}{2} \mathbf{Q}_E \vec{\chi} - \vec{\zeta} \left[k_{r0} + k_{r1} \cdot \left(\|\mathbf{w}\|_F + \overline{\mathbf{w}}_* \right) \right] \cdot \vec{\zeta}^T \\ & + \vec{\zeta} \, \mathbf{z} + \lambda \cdot \|\vec{\zeta}\|_2 \cdot tr\left(\widetilde{\mathbf{w}} \cdot \mathbf{w}^T \right). \end{aligned} \tag{22}$$

In den Arbeiten [4, 5, 17] wird die Begrenztheit der Fehlerdynamik der Netzwerkgewichte sowie die negative Beschaffenheit von $\dot{V}(\vec{\chi})$ bewiesen und verifiziert. In Abbildung 2 ist der Aufbau des Regelungsarchitektur am Beispiel der Rotationsdynamik dargestellt.

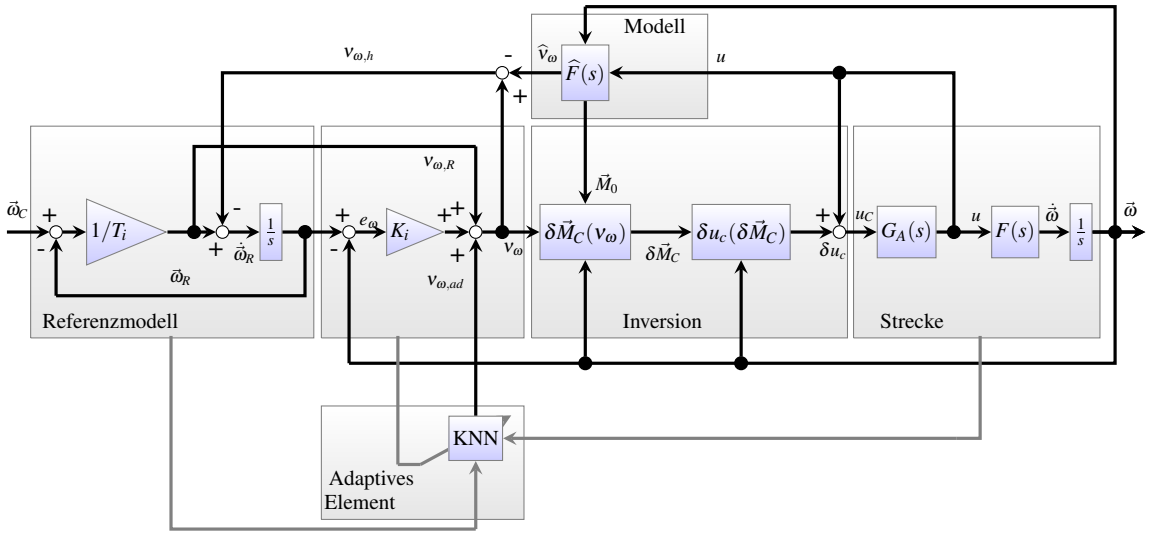

Abbildung 2: Erweiterte Regelungsarchitektur der Rotationsdynamik eines Flugzeuges [5].

3 Sliding Mode Lernverfahren

Die Betrachtung der Netzwerktrainings als ein Optimierungsproblem ist oftmals verknüpft mit einer intransparenten Gewichtsänderung, was im Besonderen beim Training im Flug als nicht ideal empfunden wird. Die Theorie der strukturvariablen Regelung bietet den nichtlinearen Ansatz der Gleitzustandsregelung (engl. Sliding Mode Control - SMC), der übertragen auf das Netzwerktraining, das KNN als ein zu regelndes System betrachtet. Dabei sind der Netzwerkfehler und dessen Ableitungen die Systemzustände und die Gewichtsänderung $\Delta\vec{w}$ das Steuersignal. Die Grundidee besteht darin, den Zustandsraum des Netzwerks in unterschiedliche Segmente zu teilen. Zu diesem Zweck werden sogenannte diskontinuierliche Schaltfunktionen $\vec{S}(x)$ definiert, die in Kombination mit einem Schaltgesetzt und einer Erreichbarkeitsbedingung dazu führen, dass die Fehlerzustände auf vordefinierte Bahnen im Phasenraum gebracht werden, sobald vordefinierte Schaltflächen überquert werden. So kann eine Trajektorie vorgegeben werden, auf der die Netzwerkzustände in Richtung des stabilen Ursprungs gleiten [23, 10, 24]. Dieser spezielle Zustand nennt sich Gleitzustand. Es kann gezeigt werden, dass ein System im Gleitzustand robust gegenüber externen Störungen ist und sich global stabil verhält [25, 26, 27].

Die Übertragung der Gleitzustandsregelung auf das Training neuronaler Netzwerke erfolgt durch die Erweiterung des konventionellen Gradienabstiegverfahrens. Dafür wird die Änderung der Netzwerkgewichte mit dem Vorzeichen der Schaltfunktion $\vec{S}(x)$ erweitert

$$\Delta\vec{w} = \left(\frac{\partial\vec{y}(\vec{w},\vec{x},\vec{y}_d)}{\partial\vec{w}(t)}\right)^T \cdot \mu \cdot \mathrm{diag}\left(\mathrm{sign}(\vec{S})\right) \cdot |\vec{\varepsilon}| \,. \tag{23}$$

Für sowohl die Definition der Schaltfunktionen als auch der Erreichbarkeitsbedingungen existieren verschiedene Ansätze in der Literatur in Bezug auf ihre Ordnung [28, 11, 12]. Es kann gezeigt werden, dass eine Schaltfunktion zweiter Ordnung das Lernen neuronaler Netze verbessert [2]. Der Klarheit wegen sei zunächst das Vorgehen für den Gleitzustand erster Ordnung dargestellt. Es wird eine Schaltlinie durch

$$\vec{S} = \dot{\vec{\varepsilon}} + \lambda \cdot \vec{\varepsilon} \tag{24}$$

definiert. Für $\vec{S} = 0$ befindet sich das System direkt auf der Schaltlinie, auf der der Netzwerkfehler gegen 0 läuft, sofern der Faktor λ positiv gewählt wird:

$$\vec{S} = \dot{\vec{\varepsilon}} + \lambda \cdot \vec{\varepsilon} = 0 \quad \Rightarrow \quad \vec{\varepsilon} = \vec{\varepsilon}(t_0) \cdot \mathrm{e}^{-\lambda(t-t_0)}. \tag{25}$$

Für den Beweis der Existenz und Erreichbarkeit des Gleitzustandes wird nun ein Ljapunow-Ansatz gewählt. Für ein zeitdiskretes System wird die folgende Bedingung gefordert:

$$|\vec{S}_{k+1}| < |\vec{S}_k| \tag{26}$$

wobei $\vec{S}(t) = \vec{S}_k$ den aktuellen und $\vec{S}(t+T_s) = \vec{S}_{k+1}$ den nächsten Zeit- beziehungsweise Lernschritt beschreibt. Solange der absolute Betrag der Schaltfunktion $\vec{S}$ über die Zeit abnimmt, streben die Zustände der Gleitfläche entgegen. Im Weiteren wird die Ableitung des Netzwerkfehlers $\vec{\varepsilon}$ über die Taktrate des Regelkreises T_s bestimmt.

$$\dot{\vec{\varepsilon}} \approx \frac{\vec{\varepsilon}(t) - \vec{\varepsilon}(t-T_s)}{T_s} \tag{27}$$

Unter Verwendung von (24) und (27) können dann $\vec{S}_k$ sowie $\vec{S}_{k+1}$ bestimmt werden:

$$\vec{S}_k = \dot{\vec{\varepsilon}}_k + \lambda \cdot \vec{\varepsilon}_k = \left(\lambda + \frac{1}{T_s}\right)\vec{\varepsilon}_k - \frac{1}{T_s}\vec{\varepsilon}_{k-1} \,, \tag{28}$$

$$\vec{S}_{k+1} = \dot{\vec{\varepsilon}}_{k+1} + \lambda \cdot \vec{\varepsilon}_{k+1} = \left(\lambda + \frac{1}{T_s}\right)\vec{\varepsilon}_{k+1} - \frac{1}{T_s}\vec{\varepsilon}_k \,. \tag{29}$$

Für die Berechnung des Fehlers $\vec{\varepsilon}_{k+1}$ aus (29) kann die folgende Näherung getroffen werden:

$$\vec{\varepsilon}_{k+1} = \vec{\varepsilon}_k + \Delta\vec{\varepsilon}_k. \tag{30}$$

Die Änderung des Netzwerkfehlers innerhalb eines Zeitschrittes T_S wird als Differenz zwischen der Zielausgabe mit dem Index t und der dazugehörigen tatsächlichen Netzausgabe bestimmt werden:

$$\begin{aligned}\Delta\vec{\varepsilon}_k &= \vec{\varepsilon}_{k+1} - \vec{\varepsilon}_k = (\vec{y}_{t,k+1} - \vec{y}_{k+1}) - (\vec{y}_{t,k} - \vec{y}_k) \\ &= \Delta\vec{y}_{t,k} - \Delta\vec{y}_k. \end{aligned} \tag{31}$$

Die Änderung des gewünschten Netzausgangs $\Delta\vec{y}_{t,k}$ kann durch eine Extrapolation aus dem vorangegangenen Zeitschritt angenähert werden.

$$\Delta\vec{y}_{t,k} = \vec{y}_{t,k+1} - \vec{y}_{t,k} \approx \vec{y}_{t,k} - \vec{y}_{t,k-1}\ . \tag{32}$$

Für den unbekannten Parameter $\Delta\vec{y}_k$ wird eine Taylorreihenentwicklung angenommen:

$$\Delta\vec{y}_k = \frac{\partial\vec{y}_k(\vec{w}_k,\vec{x}_k)}{\partial\vec{w}_k}\Delta\vec{w}_k + \frac{\partial\vec{y}_k(\vec{w}_k,\vec{x}_k)}{\partial\vec{x}_k}\Delta\vec{x}_k. \tag{33}$$

Die partiellen Ableitungen $\frac{\partial\vec{y}_k(\vec{w}_k,\vec{x}_k)}{\partial\vec{w}_k}$ und $\frac{\partial\vec{y}_k(\vec{w}_k,\vec{x}_k)}{\partial\vec{x}_k}$ werden durch den Gradientenabstiegsalgorithmus bei der Rückwärtspropagation des Netzausgangs $\vec{y}_k$ durch das KNN bestimmt und liegen somit vor. Die Gewichtsänderung ist bereits durch Gleichung (23) gegeben. Die Änderung der Netzwerkeingänge $\Delta\vec{x}_k$ wird üblicherweise für kleine Zeitschritte T_s als minimal angesehen. Das Einsetzen dieser Beziehungen in Gleichung (26) ergibt:

$$\left|\vec{S}_k\right| > \left|\left(\lambda + \frac{1}{T_s}\right)\left(\vec{\varepsilon}_k + \Delta\vec{y}_{t,k} - \frac{\partial\vec{y}_k}{\partial\mathbf{w}_k}\Delta\mathbf{w}_k - \frac{\partial\vec{y}_k}{\partial\underline{\vec{x}}_k}\Delta\underline{\vec{x}}_k\right) - \frac{1}{T_s}\vec{\varepsilon}_k\right|. \tag{34}$$

Dieser Ausdruck beschreibt die Stabilitätsbedingung für den Gleitzustand in Abhängigkeit von bekannten Parametern und der Änderung der Netzwerkgewichte als Funktion der Lernrate. Nun kann die Gleichung nach der Lernrate aufgelöst und auf diese Weise innerhalb der aufgestellten Stabilitätsgrenzen in jedem Zeitschritt dynamisch berechnet werden. Mit den Parametern a_i und b_i

$$a_i = \left(\lambda + \frac{1}{T_s}\right)\left(\varepsilon_{k,i} + \Delta y_{d,k,i} - \left(\frac{\partial\vec{y}_k}{\partial\vec{x}_k}\Delta\vec{x}_k\right)_i\right) - \frac{1}{T_s}\varepsilon_{k,i} \tag{35}$$

$$b_i = \left(\lambda + \frac{1}{T_s}\right)\left(\frac{\partial\vec{y}_k}{\partial\vec{w}_k}\left(\frac{\partial\vec{y}_k}{\partial\vec{w}_k}\right)^T \mathrm{diag}\left(\mathrm{sign}(\vec{S}_k)\right)|\vec{\varepsilon}_k|\right)_i \tag{36}$$

wobei i den betrachteten Netzwerkausgang angibt, kann der finale Ausdruck für die Berechnung der Lernrate μ durch (34) und (35) umgeschrieben werden zu:

$$\left|\vec{S}_{k,i}\right| > |a_i - \mu\cdot b_i|\ . \tag{37}$$

Abschließend lassen sich auf dieser Grundlage Fallunterscheidungen treffen, die durch die stabile Berechnung von μ den Gleitzustand der Netzwerkzustände garantieren:

$$\begin{aligned}&\left\{-\frac{S_{k,i}}{b_i} + \frac{a_i}{b_i} < \mu < \frac{S_{k,i}}{b_i} + \frac{a_i}{b_i}\right\} \\ &\text{für } \left(S_{k,i} > 0 \wedge b_i > 0\right) \vee \left(S_{k_i} < 0 \wedge b_i < 0\right)\ , \\ &\left\{\frac{S_{k,i}}{b_i} + \frac{a_i}{b_i} < \mu < -\frac{S_{k,i}}{bi} + \frac{a_i}{b_i}\right\} \\ &\text{für } \left(S_{k,i} > 0 \wedge b_i < 0\right) \vee \left(S_{k_i} < 0 \wedge b_i > 0\right)\ .\end{aligned} \tag{38}$$

In anschließenden Kapitel soll auf die Besonderheit und die Erreichbarkeit des Gleitzustands höherer Ordnung eingegangen werden.

3.1 Sliding Mode Höherer Ordnung

Ursprünglich dafür entworfen das sogenannte Rattern (engl. Chattering) beim wiederholten schnellen Überschreiten der Schaltfläche zu reduzieren, haben Gleitzustandsregler höherer Ordnung (engl. High Order Sliding Mode - HOSM) außerdem eine höhere Regelgüte zeigen können und stellen so eine vielversprechende Erweiterung des bereits beschriebenen Gleitzustandstraining dar [29, 2]. Dabei können alle positiven Eigenschaften des klassischen Sliding Modes erster Ordnung beibehalten werden [30, 31]. Bei der Umsetzung dieser Regelstrategie wird die Dynamik der Zustandstrajektorie nicht ausschließlich durch die Schaltfunktion $\vec{S} = 0$ definiert, sondern entsprechend der gewünschten Ordnung r bestimmen die $r-1$-Derivative von $\vec{S}$ die Annäherung an den Ursprung. Das bedeutet, dass ein Sliding Mode r-ter Ordnung oder auch r-Sliding durch die folgende Bedingung definiert wird:

$$\vec{S} = \dot{\vec{S}} = \ddot{\vec{S}} = \cdots = \vec{S}^{(r-1)} = 0. \tag{39}$$

Nach [31] können dabei auch Schaltfunktionen erster Ordnung verwendet werden, wenn sichergestellt wird, dass die entsprechenden Ableitungen von $\vec{S}$ zum Schalten heran gezogen werden. Aufgrund des steigenden Informationsbedarf durch die benötigten Ableitungen der Schaltfunktion $\vec{S}$ und der damit verbundenen Komplexität beim Reglerentwurf haben sich in der Praxis Sliding Mode zweiter Ordnung, auch 2-Sliding Regler genannt, insbesondere auf Grundlage des *Twisting*-oder *Super Twisting-Algorithmus* durchgesetzt [31, 32, 33, 34, 35, 36].

Die Vorgehensweise für die Kombination von HOSM Ansätzen und dem Training neuronaler Netzwerke entspricht dabei grundlegend den Schritten des Standard Sliding Modes aus Abschnitt 3. Zunächst ist eine geeignete Schaltfunktion entsprechend der gewünschten Ordnung zu definieren. Diese kann in Anlehnung an (24) mit Hilfe des mathematischen Zusammenhangs:

$$\vec{S} = \left(\frac{d}{dt} + \lambda\right)^{n-1}\cdot\vec{\varepsilon} \tag{40}$$

gebildet werden. Dabei ist die Systemordnung exakt um den Faktor eins höher als die der Schaltfunktion [33].

Ein weiterer entscheidender Punkt ist die gewählte Erreichbarkeitsbedingung, die einen Übergang des Systemzustandes

in die Gleitphase und somit die Überführung in den stationären Ursprung in endlicher Zeit gewährleisten soll. Da das System außerhalb des Gleitzustandes nicht über die spezifischen Eigenschaften bezüglich der Robustheit gegenüber Störungen und Modellunsicherheiten verfügt, ist ein möglichst schneller Transfer auf die Gleitlinie/ -fläche zwar wünschenswert, zu hohe Geschwindigkeiten können jedoch zu einem starken Überschwingen führen, welches in der Folge Chattering begünstigt. Neben der Bedingung aus Gleichung (26) existieren Reaching Law-Ansätze, die durch unterschiedliche Näherung des ersten zeitlichen Derivativs der Schaltlinie, wie der Form

$$RL = \dot{\vec{S}} = -Q \cdot sgn(\vec{S}) \tag{41}$$

die Erreichbarkeit der Gleitlinie/ -fläche garantieren [37], [27]. Der Faktor Q stellt dabei eine Verstärkungsmatrix dar.

Um eine Sliding Mode Bewegung zweiter Ordnung zu gewährleisten, kann der Reaching Law-Ansatz aus Gleichung (41) entsprechend Gleichung (42) erweitert werden [38]:

$$RL = \ddot{\vec{S}} = -Q \cdot sgn(\vec{S}) - K \cdot sgn(\dot{\vec{S}}), \tag{42}$$

wobei Q und K erneut Verstärkungsmatrizen darstellen. Ein weiteres Reaching Law zweiter Ordnung (RLzO) soll in der Folge in leicht abgeänderter Form entsprechend Gleichung (43) [38] betrachtet und für spätere Simulationen herangezogen werden. Es gilt:

$$RL = \ddot{\vec{S}} = -Q \cdot sgn(\vec{S}) \cdot |\vec{S}|^{\alpha} - K \cdot sgn(\dot{\vec{S}}) \cdot |\dot{\vec{S}}|^{\beta} \tag{43}$$

mit

$$0 < \alpha < 1, \quad \alpha = \frac{p}{q}, \quad \beta = \frac{2p}{p+q}\,. \tag{44}$$

Die RL-Parameter Q, K, p und q können dabei entsprechend den vorgegebenen Grenzen variiert werden und beeinflussen so die Dynamik der Annäherung des Systemzustandes. Dabei soll das in Gleichung (43) beschriebene Reaching Law die Annäherung von $\vec{S}_k$ und $\dot{\vec{S}}_k$ gegen Null gewährleisten.

Um die Anpassung der Verbindungsgewichte innerhalb des künstlichen neuronalen Netzwerkes, entsprechend Gleichung (23) zu gewährleisten, wird die Lernrate μ auch für Reaching Law-Ansätze zweiter Ordnung für jeden Simulationsschritt dynamisch berechnet. Die Vermeidung einer empirischen Auslegung dieses wichtigen Parameters und den beschrieben Vorteilen dieses Vorgehens bleibt somit bestehen.
Zunächst wird eine geeignete Schaltfunktion (40) gebildet, für die an dieser Stelle auf eine Funktion erster Ordnung entsprechend Gleichung (24) zurückgegriffen wird:

$$\vec{S} = \dot{\vec{\varepsilon}} + \lambda \cdot \vec{\varepsilon}. \tag{45}$$

Um die Lernrate dynamisch bestimmen zu können, ist es notwendig die ersten zeitlichen Derivative der Schaltfunktion $\dot{\vec{S}}$ und $\ddot{\vec{S}}$ abzuschätzen. Analog zum Derivativ des Netzfehlers $\dot{\vec{\varepsilon}}_k$ in Gleichung (27) erfolgt die Näherung der ersten Ableitung der Schaltfunktion zu:

$$\dot{\vec{S}}_k \approx \frac{\vec{S}_k - \vec{S}_{k-1}}{T_s}. \tag{46}$$

Für die Näherung des zweiten zeitlichen Dertivativs kann analog

$$\ddot{\vec{S}}_k \approx \frac{\vec{S}_k - 2\vec{S}_{k-1} + \vec{S}_{k-2}}{T_s^2} \tag{47}$$

angenommen werden. Da mit Hilfe des Reaching Law-Ansatzes die Überführung des Systemzustandes auf die Gleitfläche, beschrieben durch $\vec{S}_k = \dot{\vec{S}}_k = 0$, in endlicher Zeit erfolgen soll, ist es zweckmäßig die Gradienteninformation nicht aus den zwei vergangenen Zeitschritten zu beziehen. Stattdessen wird eine Näherung für den kommenden Zeitschritt $\vec{S}_{k+1}$ berücksichtigt. Um eine Aussage über den Systemzustand im folgenden Zeitschritt treffen zu können, werden zwei Gradienten der Schaltfunktion betrachtet, die sich aus den Intervallen des aktuellen $\vec{S}_k$ mit $\vec{S}_{k-1}$ sowie $\vec{S}_k$ und $\vec{S}_{k+1}$ zusammensetzen. Es wird demnach neben der derzeitigen Schaltfunktion auch die Funktion einen Zeitschritt in der Vergangenheit betrachtet, bzw. ein Schritt in der Zukunft abgeschätzt. Die Näherung der zweiten Ableitung folgt damit zu:

$$\ddot{\vec{S}}_k = RL \approx \frac{\vec{S}_{k+1} - 2 \cdot \vec{S}_k + \vec{S}_{k-1}}{T_s^2}. \tag{48}$$

Da die zur Bestimmung von μ notwendigen Größen von $\vec{S}_k$ und $\vec{S}_{k-1}$ während der Simulation des Regelungsprozesses verfügbar sind, ist zunächst $\vec{S}_{k+1}$, analog zu Gleichung (29) zu bestimmen:

$$\vec{S}_{k+1} = \dot{\vec{\varepsilon}}_{k+1} + \lambda \cdot \vec{\varepsilon}_{k+1} = \left(\lambda + \frac{1}{T_s}\right)\vec{\varepsilon}_{k+1} - \frac{1}{T_s}\vec{\varepsilon}_k \tag{49}$$

mit

$$\dot{\vec{\varepsilon}}_{k+1} = \frac{\vec{\varepsilon}_{k+1} - \vec{\varepsilon}_k}{T_s}. \tag{50}$$

Die folgenden Schritte entsprechen zunächst dem in Abschnitt 3 beschriebenen Verfahren mit der Erreichbarkeitsbedingung $|\vec{S}_{k+1}| < |\vec{S}_k|$. Anfangs wird der unbekannte Faktor $\vec{\varepsilon}_{k+1}$ sowie der Faktor $\Delta\vec{y}_k$ entsprechend den Gleichungen (30), bzw. (31) approximiert. Durch Einsetzen von Gleichung (23), welche die Gewichtsanpassung innerhalb des künstlichen neuronalen Netzwerkes beschreibt, kann der unbekannte Faktor $\Delta\vec{\varepsilon}_k$ ermittelt werden:

$$\Delta\vec{\varepsilon}_k = \Delta\vec{y}_{t,k} - \mu \cdot \frac{\partial\vec{y}_k}{\partial\vec{w}_k}\left(\frac{\partial\vec{y}_k}{\partial\vec{w}_k}\right)^T \mathrm{diag}\left(\mathrm{sign}(\vec{S}_k)\right)|\vec{\varepsilon}_k|. \tag{51}$$

Somit folgt für $\vec{S}_{k+1}$:

$$\vec{S}_{k+1} = \left(\lambda + \frac{1}{T_s}\right) \cdot \tag{52}$$

$$\left(\Delta\vec{y}_{t,k} - \mu \cdot \frac{\partial\vec{y}_k}{\partial\vec{w}_k}\left(\frac{\partial\vec{y}_k}{\partial\vec{w}_k}\right)^T \mathrm{diag}\left(\mathrm{sign}(\vec{S}_k)\right)|\vec{\varepsilon}_k|\right) + \lambda \cdot \vec{\varepsilon}_k. \tag{53}$$

Da auf diese Weise alle unbekannten Faktoren abgeschätzt worden sind, kann anschließend mit Hilfe der gewählten Er-

reichbarkeitsbedingung die Lernrate für jeden Zeitschritt ermittelt werden. Eingesetzt in (48) ergibt sich:

$$RL \cdot T_s^2 + 2\vec{S}_k - \vec{S}_{k-1} = \left(\lambda + \frac{1}{T_s}\right) \cdot \left(\Delta\vec{y}_{t,k} - \mu \cdot \frac{\partial \vec{y}_k}{\partial \vec{w}_k} \left(\frac{\partial \vec{y}_k}{\partial \vec{w}_k}\right)^T \mathrm{diag}\left(\mathrm{sign}(\vec{S}_k)\right) |\vec{\varepsilon}_k|\right) + \lambda \cdot \vec{\varepsilon}_k, \tag{54}$$

und nach entsprechenden Umformungen:

$$\mu = \left(\frac{-RL \cdot T_s^2 - 2\vec{S}_k + \vec{S}_{k-1} + \lambda \cdot \vec{\varepsilon}_k + \Delta\vec{y}_{t,k} \cdot \left(\lambda + \frac{1}{T_s}\right)}{\left(\frac{\partial \vec{y}_k}{\partial \vec{w}_k} \left(\frac{\partial \vec{y}_k}{\partial \vec{w}_k}\right)^T \mathrm{diag}\left(\mathrm{sign}(\vec{S}_k)\right) |\vec{\varepsilon}_k|\right) \cdot \left(\lambda + \frac{1}{T_s}\right)}\right). \tag{55}$$

Im Unterschied zu Abschnitt 3 wird aufgrund der unterschiedlichen Erreichbarkeitsbedingungen für die Lernrate kein Intervall sondern ein spezifischer Wert bestimmt. Dabei ist darauf zu achten, dass die Lernrate zu jeder Zeit positive Werte annimmt. Sollte dies durch den vorgegebenen Algorithmus nicht möglich sein, so wird μ in dem entsprechenden Zeitschritt zu Null gesetzt.

4 Simulationsergebnisse

Die ersten Analysen des Gleitzustands höherer Ordnung werden zunächst mit Hilfe einer Testsimulationsumgebung durchgeführt. In dieser bezieht das Netzwerk sinusförmige Eingangsgrößen und wird darauf trainiert verschiedene Sollverläufe zu approximieren. In der ersten Simulation soll das Verhalten der Netzwerkzustände nach einem einfachen Sprung in der Sollgröße untersucht werden. In Abb. 3 sind die Verläufe der Zustandsgrößen im Phasendiagramm für das Slidinge Mode Backpropagation Training (SMBP) erster Ordnung und das Reaching Law zweiter Ordnung (RLzO) dargestellt.

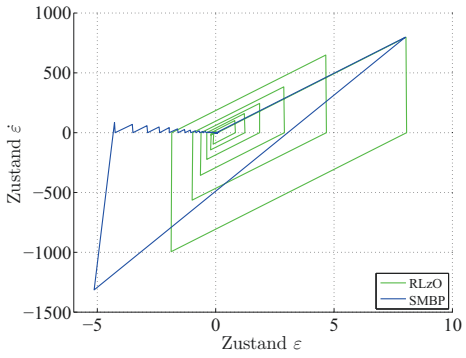

Abbildung 3: Phasendiagramm für unterschiedlichen Gleitzustandslernverfahren in einer Testumgebung

Es ist deutlich zu erkennen, wie die Trainingsverfahren die Zustände auf unterschiedlichen Wegen zurück in den Ursprung überführen. Beide Trajektorien starten im fehlerfreien Zustand bei $\vec{\varepsilon} = 0$ und $\dot{\vec{\varepsilon}} = 0$ bevor der Sprung auf die Trainingsgröße gegeben wird. Dieser führt zur ersten Auslenkung der Zustände. Das SMBP Training erreicht nach zwei Zeitschritten den Gleitzustand und folgt der Gleitlinie mit schwachem Chattering bis in den Ursprung. Der Reaching Law-Ansatz zeigt die für einen *Twisting-Algorithmus* typische Spiralbewegung, welche letztendlich im Ursprung endet. Es kann deutlich das Umschalten des Trainings durch die scharfen Änderungen der Trajektorie beobachtet werden.

In dieser Form kann jedoch noch keine Aussage über die Konvergenzgeschwindigkeit der Verfahren getroffen werden. Zu diesem Zweck werden in Abb. 4 die Zustandstrajektorien über der Zeit dargestellt, wobei der Sprung bei Sekunde eins auf das System gegeben wird. Es ist zu erkennen, dass beide Verfahren den Fehler ε nach 1,3 Sekunden zurück auf Null geführt haben und bereits nach nur 0,1 Sekunden den Großteil des Fehlers wieder abgebaut haben.

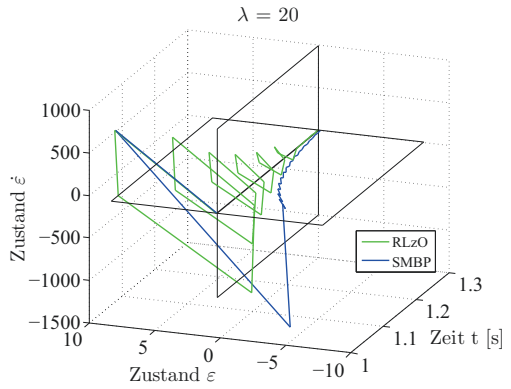

Abbildung 4: Phasendiagramm und Konvergenzzeiten für unterschiedlichen Gleitzustandslernverfahren in einer Testumgebung

Um die Robustheit gegenüber großen und mehrfach auftretenden Sprüngen der Trainingsgröße zu untersuchen, wird ein weiterer Sollverlauf vorgegeben. Dieser besteht aus der Überlagerung verschiedener Sinusschwingungen kombiniert mit diversen Sprüngen. In Abb. 5 sind die Zielgröße sowie die Netzausgaben der verschiedenen Lernverfahren im Vergleich dargestellt. Es kann gezeigt werden, dass sowohl der konventionelle Gradientenabstieg (engl. Gradient Descent - GD) als auch beide Gleitzustandsverfahren den sprunghaften Verlauf robust approximieren können. Dabei agiert der Reaching Law-Ansatz in den simulierten 20 Sekunden am schnellsten und zeigt die höchste Konvergenzgeschwindigkeit. Der Gradientenabstieg zeigt vermehrt ein starkes Überschwingen an sowohl den Sprüngen, als auch während kontinuierlichen Sollgrößenverläufen. Die Lernrate von $\mu = 0{,}08$ scheint für die großen Änderungen in der Trainingsgröße zu hoch zu sein, so dass wie in Kapitel 3 beschrieben, die Wahl der Lernrate konservativer ausfallen sollte, was wiederum zu einer langsameren Konvergenzgeschwindigkeit führen würde. Beim ersten Sprung nach einer Sekunde ist für beide Gleitzustandsverfahren zu sehen, dass die Gewichte zunächst an die vorgegebene Dynamik angepasst werden müssen, so dass es einmalig zu Überschwingern kommt, die im weiteren

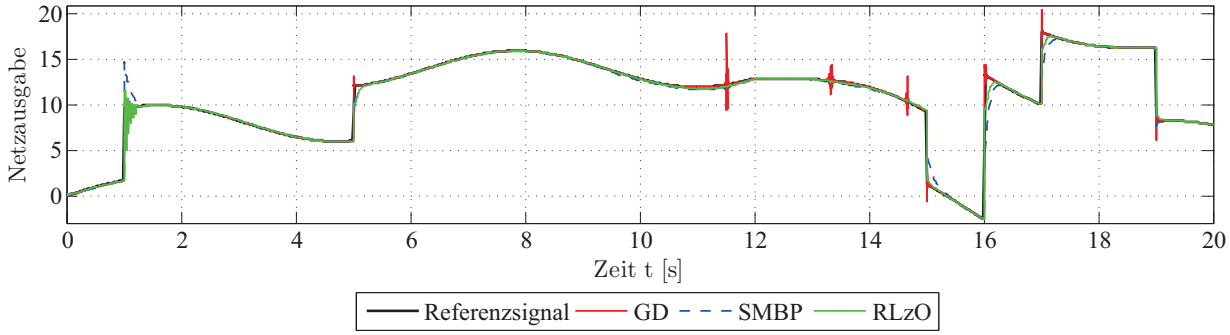

Abbildung 5: Netzwerkausgaben für unterschiedliche Lernverfahren in einer Testumgebung.

Verlauf nicht mehr auftreten.

Die Untersuchung der gesamten Reglerarchitektur wird mit der in Abschnitt 1 beschriebenen Simulationsumgebung des unbemannten Flugsystems *CAROLO P360* durchgeführt. Die Regelungsaufgabe ist dabei die automatische Bahnfolge unter Einfluss verschiedener atmosphärischen Bedingungen mit teilweise künstlich erzeugten Inversionsfehlern und Modellunsicherheiten. Der Verlauf der Testbahn ist in Abb. 6 dargestellt. Das gezeigte Profil stellt mit einer Abfolge von Steig- und Sinkflügen sowie verschiedenen Kurvenflügen ein dynamisches Szenario für ein kleines Starrflügel UAS dar.

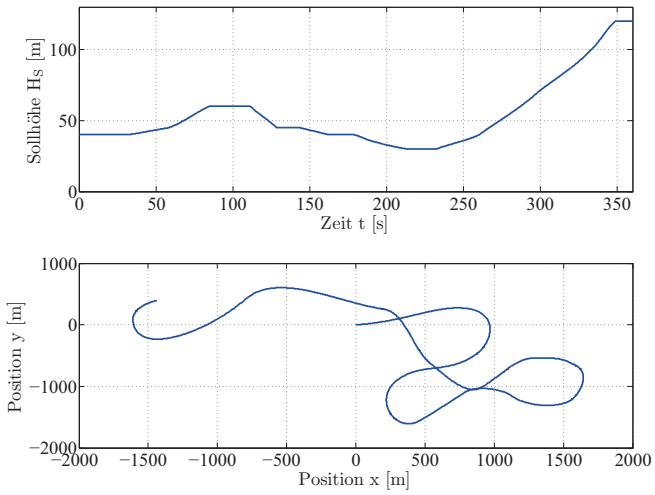

Abbildung 6: Horizontales und vertikales Profil der vorgegebenen Flugbahn.

Die erste Simulation widmet sich der Konvergenzgeschwindigkeit der verschiedenen Trainingsalgorithmen in einer vergleichbaren Fehlersituation. Zu diesem Zweck werden die ersten 15 Sekunden der gezeigten Flugbahn simuliert und bei Sekunde fünf ein konstanter Inversionsfehler von $\Delta = 5 rad/s^2$ auf die Nickrate gegeben. Dieser Fehler steht in keinem beschreibbaren Zusammenhang zum geregelten Modell und muss aus diesem Grund durch das neuronale Netzwerk im Nickkanal ausgeglichen werden, damit es nicht zur Destabilisierung bis hin zum Absturz des simmulierten Systems kommt. In Abb. 7 ist der Netzwerkfehler ε als Reaktion auf die künstlich aufgebrachte Störung dargestellt. Alle drei Trainingsverfahren gleichen den in Sekunde fünf aufgebrachten Fehler nach ungefähr zehn Sekunden aus. Es ist deutlich zu erkennen, dass für $\lambda = 80$ und $\mu = 0.08$ der Gradientenabstieg die längste Zeit benötigt und kurz vor dem Ausgleich des Fehlers eine leichte Schwingung induziert. Das HOSM-Training hält den Fehler auf dem geringsten Absolutwert und approximiert den Inversionsfehler nach ungefähr einer Sekunde vollständig.

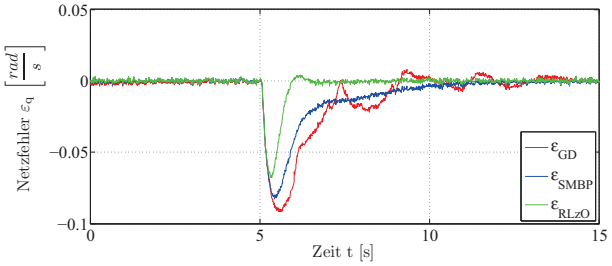

Abbildung 7: Vergleich zwischen GD und den beiden SMC-Trainingsverfahren für ein Referenzfehlerszenario der Längsbewegung ohne Wind und Turbulenz.

Die Auswirkung der unterschiedlichen Lernverfahren auf die kombinierten Netzwerkausgänge ist in Abb. 8 zusehen. Dabei stellen die Verläufe die Summe des robustifizierenden Terms ν_r und des Netzwerkausgangs ν_{ad} dar. Es ist zu erkennen, wie die verschieden trainierten Netzwerke nach Auftreten des Fehlers sich an das inverse Fehlersignal anpassen und so die aufgebrachte Störung kompensieren. Wie schon im Fehlerverlauf zu sehen, nähert sich das Netzwerk mit HOSM-Training schneller an die zu kompensierende Fehlergröße an, als es das Gleitzustandstraining erster Ordnung oder der klassische Gradientenabstieg tut.

Zur weiteren Untersuchung der unterschiedlichen Trainingsverhalten ist in Abb. 9 die Lernrate der einzelnen Verfahren abgebildet. Zu sehen ist die konstante und empirisch gewählte Lernrate des GD-Trainings und die beiden dynamisch kalkulierten Lernraten der Gleitzustandsansätze. Bei Sekunde fünf ist ein wesentlicher Vorteil der SMC-Verfahren zu erkennen. Vor der Aufbringung des Fehlers ist eine schnelle und betragsmäßig große Änderung der Lernrate zu beob-

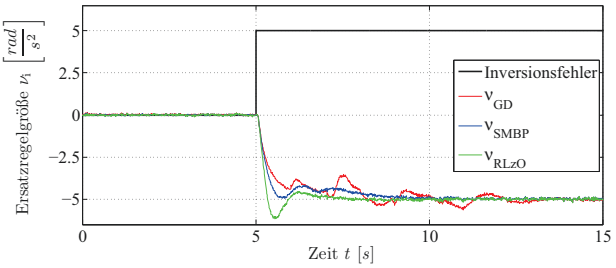

Abbildung 8: Vergleich des kombinierten Netzwerksignals für einen konstanten Inversionsfehler in der Nickrate.

achten. Bei Sekunde fünf sinken die Lernraten beider SMC-Lernverfahren, um auf diese Weise den Gleitzustand trotz großer Fehlerwerte sicherstellen zu können. Auf diese Weise wird der destabilisierende Effekt durch erheblichen Gewichtsveränderungen bei zu großen Lernraten vermieden. Erst nach der abgeschlossenen Approximation des Fehlers steigt die Lernrate erneut. Die getroffenen Annahmen für die stabile Berechnung der Lernrate kann somit experimentell bestätigt werden. Außerdem ist zu beobachten, dass die Lernrate für das RLzO-Training zwischenzeitlich zu Null wird und somit in der logarithmischen Darstellung verschwindet. Ebenfalls bemerkenswert ist die konstant höhere Lernratenberechnung des Gleitzustandstrainings höherer Ordnung gegenüber dem SMC-Training erster Ordnung. Da alle Trainingsverfahren in der gleichen Umgebung unter identisichen Randbedingungen simuliert werden, ist die höhere Lernrate und als Folge dessen die schnellere Änderung der Netzwerkgewichte der Grund für die höhere Konvergenzgeschwindigkeit des vorgestellten Ansatzes zweiter Ordnung.

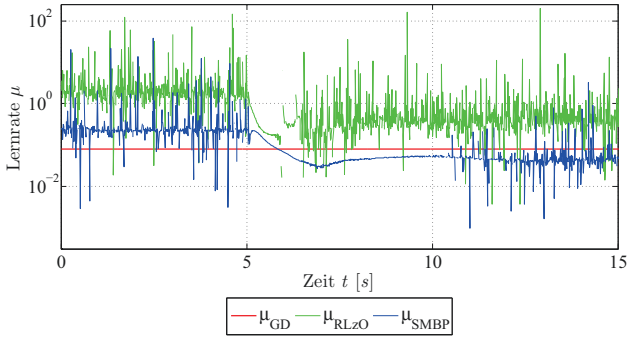

Abbildung 9: Vergleich der Lernrate zwischen GD- und den beiden SMC-Trainingsverfahren.

Um die Geschwindigkeit und Robustheit der vorgestellten Algorithmen nicht nur in Referenzfehlersituationen zu vergleichen, wird zum Ende der Untersuchung eine vollständige Bahnfolge bei Wind und Turbulenz sowie verschiedenen Inversionsfehlern simuliert. Zusätzlich zu dem konstanten Inversionsfehler in der Nickrate werden sämtliche aerodynamischen Parameter in der Inversionsregelung abgeändert [2]. Dies hat zur Folge, dass es zu gravierenden Modellfehlern zwischen dem Inversionsregler und der tatsächlichen Regelstrecke kommt. Auf diese Weise kann das Verhalten bei in der Realität oftmals vorliegenden Modellunsicherheiten untersucht werden. In Abb. 10 sind die beiden maßgeblichen Bewertungsgrößen der Reglereigenschaften, der Höhenfehler ΔH und die seitliche Bahnabweichung d, dargestellt.

Die drei Trainingsverfahren erzeugen über einen Zeitraum von ungefähr 260 Sekunden ähnliche Ergebnisse. Die langsamere Konvergenzgeschwindigkeit des GD-Trainings führt aufgrund der Summe an aufgebrachten Störungen zu Beginn der Simulation zu größeren Ablagen in sowohl der Höhe als auch der seitlichen Bahnführung. Ab Sekunde 260, dem Beginn des steilsten Steigflugabschnitts, kommt es jedoch zu kurzen aber starken Oszillationen die letztendlich zum Absturz des Flugzeugs führen. Die Kombination aus atmosphärischen Störungen sowie dem konstanten Inversionsfehler und den erheblichen Modellungenauigkeiten führt dazu, dass der Trainingsansatz mit konstanter Lernrate die induzierten Instabilitäten nicht mehr auszugleichen vermag. Die SMC-basierten Lernansätze hingegen zeigen über die gesamte Simulation stabiles verhalten und stellen somit eine zweckmäßige Erweiterung des adaptiv gestützten Inversionsregelkreis dar. In Tabelle 1 ist zu erkennen, dass der Gleitzustandsansatz zweiter Ordnung in Längs- und Seitenbewegung bessere Ergebnisse erzeugt und so die Beobachtungen aus der Testumgebung bestätigt.

	$E_{max}(d)$ $[m]$	$\sigma(d)$ $[m]$	MSE(d) $[m^2]$	$E_{max}(\Delta H)$ $[m]$	$\sigma(\Delta H)$ $[m]$	MSE(ΔH) $[m^2]$
GD	Absturz					
SMBP	4.2981	0.8591	0.7514	2.4418	0.2866	0.0821
RLzO	4.0739	0.8577	0.7492	2.1093	0.2363	0.0559

Tabelle 1: Höhen- und Splineablage bei Modellfehlern, einem konstanten Inversionsfehler und gestörter Atmosphäre.

Dabei beschreibt E_{max} den maximalen Fehler, σ die Standardabweichung und MSE den mittleren quadratischen Fehler. Der vorgestellte Reaching Law-Ansatz kann somit das Lernverhalten von gleitzustandtrainierten Netzwerken weiter verbessern.

5 Zusammenfassung

In dieser Arbeit ist die Übertragung von Gleitzustandsverfahren höherer Ordnung auf das Training künstlicher neuronaler Netzwerkwerke als Erweiterung eines dynamischen Inversionsreglers dargestellt und untersucht worden. Durch die Vorgabe eines Reaching Laws zweiter Ordnung kann die hohe Robustheit und die stabile und dynamische Berechnung der Lernrate von Ansätzen erster Ordnung beibehalten werden und zusätzlich die Konvergenzgeschwindigkeit weiter erhöht werden. Der SMC-Algorithmus zweiter Ordnung vermag die bereits beachtliche Leistungsfähigkeit eines Gleitzustandstrainings erster Ordnung, bei der Simulation einer automatischen Bahnfolge unter Einfluss von Wind und Turbulenz sowie signifikanten Modellunsicherheiten, noch weiter zu verbessern. Weiterführende Untersuchungen werden sich mit der Kombination von Reaching-Law-Ansätzen sowie Schaltfunktionen höherer Ordnung befassen.

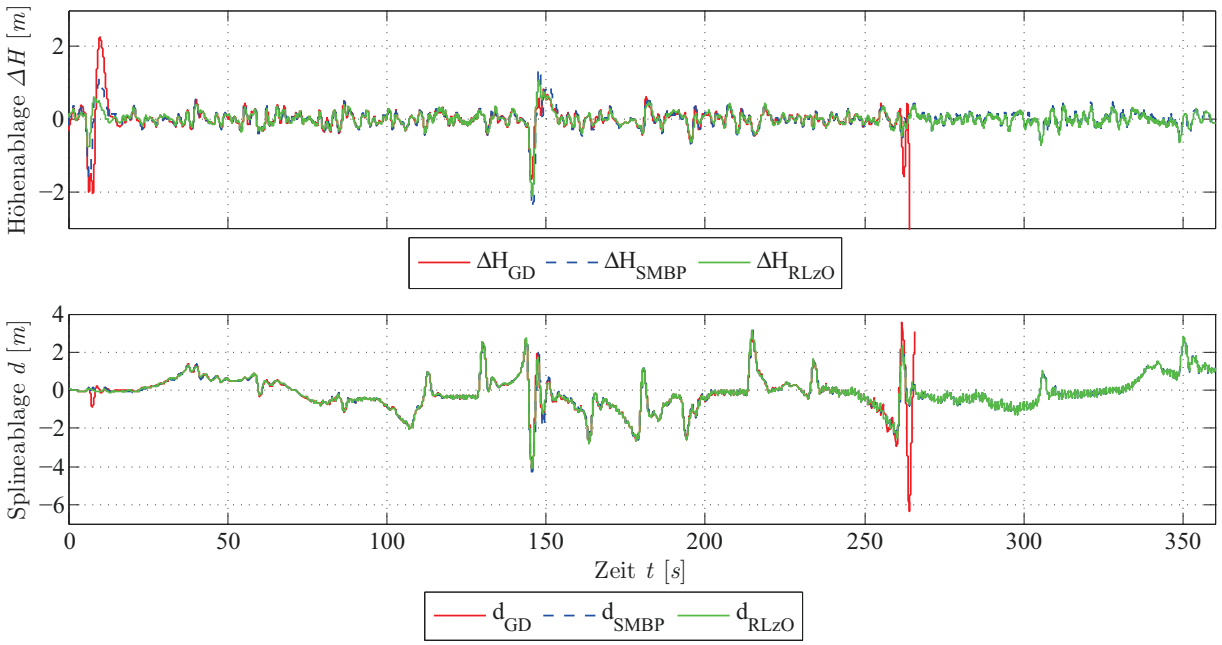

Abbildung 10: Vergleich von Höhenfehler und seitlicher Bahnabweichung bei Wind und Turbulenz, sowie einem konstanten Inversionsfehler und erheblichen Modellunsicherheiten.

Literatur

[1] BROCKHAUS, R. ; ALLES, W. ; LUCKNER, R.: *Flugregelung*. ISBN 978-3-642-01442-0 : Springer Verlag, Berlin, 2011

[2] SCHNETTER, P. ; KRÜGER, T.: Compensation of Significant Parametric Uncertainties Using Sliding Mode Online Learning. In: *IEEE Aerospace Conference 2013*. Big Sky, Montana, USA, March 03 - March 09 2013

[3] CALISE, A. ; LEE, S. ; SHARMA, M.: Development of a Reconfigurable Flight Control Law for the X-36 Tailless Fighter Aircraft. In: *AIAA Guidance, Navigation, and Control Conference*. Denver, CO, September 2000. – AIAA 2000-3940

[4] RYSDYK, R. ; CALISE, A.: Robust Nonlinear Adaptive Flight Control for Consistent Handling Qualities. In: *IEEE Transactions on Control Sytems Technology* 13 (2005), Nr. 6, S. 896–910

[5] HOLZAPFEL, F.: *Nichtlineare adaptive Regelung eines unbemannten Fluggerätes*, Lehrstuhl für Flugmechanik und Flugregelung, Technische Universität München, Diss., 2004

[6] KRÜGER, T.: *Zur Anwendung neuronaler Netzwerke in adaptiven Flugregelungssystemen*, Institut für Luft- und Raumfahrtsysteme, Technische Universität Braunschweig, Diss., 2012

[7] CHOWDHARY, G. ; JOHNSON, E. N. ; CHANDRAMOHAN, R. ; KIMBRELL, M. S. ; CALISE, A.: Guidance and Control of Airplanes Under Actuator Failures and Severe Structural Damage. In: *Journal of Guidance, Control, and Dynamics* 36 (2013), S. 1093–1104

[8] HORNIK, K. ; STINCHCOMBE, M. ; WHITE, H.: Multilayer Feedforward Networks are Universal Approximators. In: *Neural Networks* 2 (1989), S. 359–366

[9] ROJAS, R.: *Neural Networks - A Systematic Introduction*. Berlin : Springer-Verlag, 1996

[10] KAYNAK, O. ; ERBATUR, K. ; ERTUGRUL, M.: The Fusion of Computationally Intelligent Methodologies and Sliding-Mode Control - A Survey. In: *IEEE Transactions on Industrial Electronics* 48 (2001), Nr. 1, S. 4–17

[11] TOPALOV, A. V. ; KAYNAK, O.: Online Learning in Adaptive Neurocontrol Schemes with a Sliding Mode Algorithm. In: *IEEE Transactions on Sytems, Man, and Cybernetics - Part B: Cybernetics* 31 (2001), Nr. 3, S. 445–450

[12] NIED, A. ; SELEME, S. I. ; PARMA, G. G. ; MENEZES, B. R.: On-line neural training algorithm with sliding mode control and adaptive learning rate. In: *Neurocomputing* 70 (2007), S. 2687–2691

[13] ISIDORI, A.: *Nonlinear Control Systems*. Springer Verlag, Berlin, 1996

[14] KHALIL, H.: *Nonlinear Systems*. Prentice Hall, New Jersey, 3rd ed., 2002

[15] N. HOVAKIMYAN, E. L.: Positive μ-modification for Stable Adaption in Dynamic Inversion based Adaptive Control with Input Saturation. In: *American Control Conference 2005*. Portland, OR, USA, June 8 - 10 2005

[16] MENON, P. P. ; LOWENBERG, M. ; HERMANN, G. ; TURNER, M. C. ; BATES, D. G. ; POSTLETHWAITE, I.: Experimental Implementation of a Nonlinear Dynamic

Inversion Controller with Antiwindup. In: *Journal of Guidance, Control, and Dynamics* 36 (2013), Nr. 4, S. 1035–1046

[17] JOHNSON, E. N.: *Limited Authority Adaptive Flight Control*, Georgia Institute of Technology, Diss., 2000

[18] FUNAHASHI, K. I.: On the Approximate Realization of Continuous Mappings by Neural Networks. In: *Neural Networks* 2 (1989), S. 183–192

[19] KIM, N.: *Improved Methods in Neural Network-Based Adaptive Output Feedback Control with Applications to Flight Control*, School of Aerospace Engineering, Georgia Institute of Technology, Diss., 2003

[20] BURKEN, J. ; NGUYEN, N. T. ; GRIFFIN, B.: Adaptive Flight Control Design with Optimal Control Modification on an F-18 Aircraft Model. In: *AIAA Infotech@Aerospace Conference*. Atlanta, Georgia, April 2010. – AIAA 2010-3364

[21] CALISE, A. ; LEE, H. ; N.KIM: High Bandwidth Adaptive Flight Control. In: *AIAA Guidance, Navigation, and Control Conference*. Denver, CO, September 2000. – AIAA 2000-4551

[22] LEWIS, F. L. ; YEGILDIREK, A. ; LIU, K.: Multilayer neural-net robot controller with guaranteed tracking performance. In: *IEEE Transactions on Neural Networks* 7 (1996), Nr. 2, S. 388–399. `http://dx.doi.org/10.1109/72.485674`. – DOI 10.1109/72.485674

[23] UTKIN, V.: *Sliding Modes in Control and Optimization*. Springer Verlag, Berlin, 1992

[24] YU, X. ; KAYNAK, O.: Sliding-Mode Control With Soft Computing: A Survey. In: *IEEE Transactions on Industrial Electronics* 56 (2009), Nr. 9, S. 3275–3285

[25] UTKIN, V. ; GULDNER, J. ; SHI, J.: *Sliding Mode Control in Electro-Mechanical Systems*. CRC Press, London, 2009

[26] SARPTURK, S. ; ISTEFANOPULOS, Y. ; KAYNAK, O.: On the Stability of Discrete-Time Sliding Mode Control Systems. In: *IEEE Transactions on Automatic Control* AC-32 (1987), October, Nr. 10, S. 930–932

[27] HUNG, J. ; GAO, W. ; HUNG, J.: Variable Structure Control: A Survey. In: *IEEE Transactions on Industrial Electronics* 40 (1993), Nr. 1, S. 2–22

[28] SHAKEV, N. G. ; TOPALOV, A. V. ; KAYNAK, O.: Sliding Mode Algorithm for Online Learning in Analog Multilayer Feedforward Neural Networks. In: *LNCS* 2714 (2003), S. 1064–1072

[29] I. CHAIREZ, T. P. A. Poznyak P. A. Poznyak ; G. BARTOLINI, A. Pisano E. U. L. Fridman F. L. Fridman (Hrsg.): *Modern Sliding Mode Theorie - New Perspectives and Applications*. Springer Verlag, 2008 (High Order Sliding Mode Neurocontrol for Uncertain Nonlinear SISO Systems: Theory and Application)

[30] LEVANT, A.: *Higher Order Sliding Modes and their application for controlling uncertain processes*, Institute for System Studiesof the USSR Acadamy of Science, Moskau, Diss., 1987

[31] LEVANT, A.: *Introduction to Higher-Order Sliding Modes*. Tel-Aviv University, 2002-2003

[32] PISANO, A.: *Second Order Sliding Modes: Theory and Applications*, Universita Studi di Cagliari, Diss., 2000

[33] M.MEUSER: *Nichtlineare Regelung pneumatischer Antriebe*. ISBN 978-3844-004168 : Shaker Verlag, Aachen, 2011

[34] FRIDMAN, L. ; LEVANT, A. ; PERRUQUETTI, W. (Hrsg.) ; BARBOT, J. P. (Hrsg.): *Higher Order Sliding Modes*. Sliding Mode Control in Engineering. Bertrams, 2002 (Control Engineering Series)

[35] VECCHIO, C.: *Sliding Mode Control: theoretical developments and applications to uncertain mechanical systems*, Universita Degli Studi di Pavia, Diss., 2008

[36] PUNTA, E.: Second Order Sliding Mode Control of Nonlinear Multivariable Systems. In: *Proc. of the 5th International Conference on Technology and Automation*. Thessaloniki, Greece, 2005

[37] HEBISCH, H.: *Grundlagen der Sliding-Mode-Regelung*. Univ.-GH, Meß-, Steuer- und Regelungstechnik, 1995 (Forschungsbericht / Meß-, Steuer- und Regelungstechnik)

[38] TRIVEDI, P.: A Simple Reaching Law Based Design Method for 2-Sliding Mode Control. In: *Proc. IEEE International Conference on Industrial Technology*, 2010