Simulationsgestützte Auslegung von Reglern mithilfe von Machine Learning

Dominic Brown[1*], Martin Strube[1]

[1]Institut für Kommunikationssysteme und Technologien, Ostfalia Hochschule für angewandte Wissenschaften, Salzdahlumer Str. 46/48, 38302 Wolfenbüttel; *dominic.brown@itison-ikt.de

Abstract. Ein zunehmend interessantes Einsatzgebiet der künstlichen Intelligenz (KI) stellt die Auslegung von Reglern für technische Systeme mithilfe von Machine Learning und dabei besonders mit Reinforcement Learning dar. Die Anwendung von Machine Learning ermöglicht die Auslegung von Reglern, ohne die üblicherweise mit dem Entwurfsprozess einhergehenden Aufwände für menschliche Experten. Stattdessen wird durch die Verwendung von neuronalen Netzen in Kombination mit Reinforcement Learning ein neuronaler Regler entwickelt, der die Regelung eines technischen Systems selbstständig erlernt. Dabei ist das Vorhandensein eines Simulationsmodells des zu regelnden Systems bei vielen Anwendungen eine Grundvoraussetzung für einen KI-basierten Ansatz zur Reglerauslegung. In dem Beitrag wird dargestellt, wie ein neuronaler Regler mit Reinforcement Learning die Regelung eines nichtlinearen Systems in Form eines inversen Pendels erlernt. Es werden die genutzte Lernmethode und die einzelnen Phasen beim Training des neuronalen Reglers beschrieben. Durch einen Vergleich der KI-basierten Reglerauslegung mit und ohne Simulationsmodell wird anschließend erläutert, welche Vorteile das simulationsgestützte Auslegen von Reglern mithilfe von Machine Learning bietet. Abschließend werden der neuronale Regler und ein konventioneller Regler gegenübergestellt und deren Verhalten bei der Regelung des inversen Pendels verglichen.

Einleitung

Der Einsatz von künstlicher Intelligenz (KI) innerhalb technischer Systeme gewinnt zusehends an Bedeutung und gilt als ein Innovationsmotor in der Fertigungs- und Verarbeitungsindustrie. „Durch den Einsatz von KI-Technologien soll die Effizienz und Effektivität industrieller Prozesse gesteigert werden. [...] In der Regel ist für die Bewältigung komplexer und komplizierter Prozessherausforderungen Erfahrungswissen und intelligentes Vorgehen erforderlich. Daher erscheint KI neben einfachen Wenn-dann-Routinen und klassischer Automatisierungs- und Regelungstechnik sehr gut geeignet, komplexe Situationen in industriellen Prozessen zu meistern.“ [1].

Bei der Reglerauslegung mit den Methoden der klassischen Regelungstechnik analysiert ein menschlicher Experte das technische System und legt anschließend eine Regelung für das System aus. Dabei muss der Experte über die notwendigen Kenntnisse und Fähigkeiten verfügen, um die komplexe Struktur des technischen Systems zu analysieren und einen geeigneten konventionellen Regler auszulegen. Ein konventioneller Regler ist durch seine linearen Eigenschaften für einige Anwendungen nur bedingt geeignet. "These conventional controllers are linear in nature and have a fixed control law. However many systems where these controllers are used are very non-linear and have changing characteristics depending on the operational parameters." [2].

Als Alternative zu konventionellen Reglern haben sich neuronale Regler etabliert, wie aus Veröffentlichungen zum Thema [3], [4], [5] hervorgeht. Durch den Einsatz künstlicher neuronaler Netze in Kombination mit Machine Learning und der Lernmethode Reinforcement Learning wird ein neuronaler Regler ausgelegt. Der neuronale Regler erlernt die Regelung eines technischen Systems durch die Beziehung zwischen den Eingangs- und Ausgangsgrößen des Systems. Dadurch kann ein Regler ohne die Aufwände menschlicher Experten ausgelegt werden, die normalerweise mit dem Entwicklungsprozess eines Reglers verbunden sind. Die Vorteile des neuronalen Reglers liegen darin, dass Abweichungen von der erlernten Systemdynamik gut kompensiert werden können und durch seine nichtlinearen Eigenschaften ist der neuronale Regler prädestiniert für die Regelung nichtlinearer Systeme.

Ebenso wie bei der klassischen Regelungstechnik ist beim KI-basierten Ansatz zur Reglerauslegung das Vorhandensein eines Simulationsmodells vor Vorteil, in manchen Anwendungsfällen sogar zwingend notwendig.

1. Inverse Pendel

Das inverse Pendel ist eine klassische Aufgabenstellung der nichtlinearen Regelungstechnik. Das Prinzip des inversen Pendels findet sich auch in praktischen Anwendungen wieder. Beispielsweise beim Ausbalancieren eines Segway Personal Transporter oder bei den aufrecht

ARGESIM Report 59 (ISBN 978-3-901608-93-3), p 141-147, DOI: 10.11128/arep.59.a59020

landenden Raketen von SpaceX. Hier wird die Bauform eines rotierenden inversen Pendels genutzt, dass einen Aktor und zwei Freiheitsgrade besitzt. Das System des rotierenden inversen Pendels ist in Abbildung 1 dargestellt. Es besteht aus einem Pendel, das drehbar an einem Arm gelagert ist. Der Arm ist an einer Welle befestigt, die über einen Elektromotor mit einem Drehmoment beaufschlagt wird.

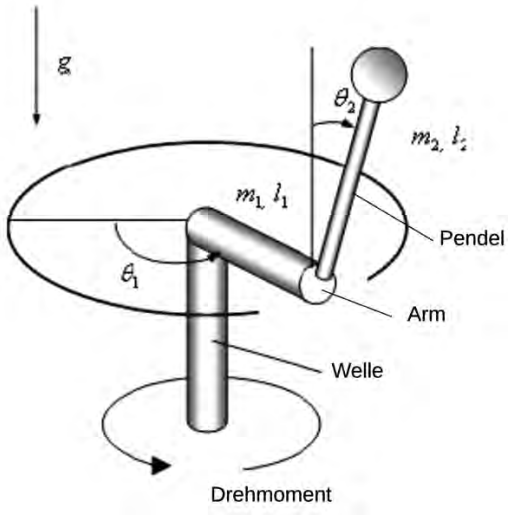

Abbildung 1: Rotierendes inverses Pendel [6]

Bei der Regelung eines rotierenden inversen Pendels wird zunächst das Pendel aus der stabilen Ruhelage in die instabile Ruhelage aufgeschwungen, also senkrecht nach oben zeigend. Dafür muss durch den Regler die benötigte Spannung an den Elektromotor anlegt werden. Dadurch wird ein Drehmoment auf die Welle aufgebracht und Bewegungen mit dem Arm ausgeführt, die das Pendel nach oben schwingen. Anschließend soll das Pendel vom Regler in der instabilen Ruhelage ausbalanciert werden. In der instabilen Ruhelage wird das Pendel den Zustand der geringsten Gesamtenergie anstreben, der erreicht wird, wenn das Pendel umfällt und die stabile Ruhelage erreicht hat. Um das Pendel in der instabilen Ruhelage auszubalancieren, muss mit dem Elektromotor ein Drehmoment aufgebracht werden, um Ausgleichsbewegungen mit dem Arm auszuführen.

Als rotierendes inverses Pendel wird der Quanser QUBE-Servo 2 verwendet. Beim Quanser QUBE-Servo 2 ist der Bewegungsbereich für den Arm durch Anschläge begrenzt. Ein optischer Drehimpulsgeber erfasst die Winkelposition des Arms und die Winkelposition des Pendels. Die Winkelposition wird zur Berechnung der Winkelgeschwindigkeit und der Winkelbeschleunigung verwendet. Als Schnittstelle wird das QFLEX 2 USB-Panel (USB 2.0) genutzt.

2. Reinforcement Learning

Machine Learning, im Deutschen maschinelles Lernen, gilt als Schlüsseltechnologie der künstlichen Intelligenz. Die Grundidee des maschinellen Lernens besteht darin, Computer mit speziellen Algorithmen in die Lage zu versetzen, selbstständig Lösungen für ein konkretes Problem zu finden. Ähnlich wie beim Menschen spielt dabei die Weiterentwicklung durch gewonnene Erfahrung eine wichtige Rolle.

Beim maschinellen Lernen werden u.a. künstliche neuronale Netze genutzt, deren Struktur dem Aufbau des menschlichen Gehirns nachempfunden ist. Mit den dazugehörigen Lern-Algorithmen können neuronale Netze ähnlich wie Menschen durch Versuch, Wiederholung und Feedback, zu den erzielten Ergebnissen, lernen.

Dabei werden unterschiedliche Lernmethoden genutzt, wie z.B. das Reinforcement Learning. Beim Reinforcement Learning (RL), im Deutschen bestärkendes Lernen, lernt ein aus neuronalen Netzen bestehendes KI-System (Agent) nach dem Grundsatz Versuch und Irrtum durch die Interaktion mit der Umgebung eine bestimmte Aufgabe zu lösen. Dabei wird nach jeder durchgeführten Aktion mithilfe einer Belohnungsfunktion bestimmte, wie gut diese Aktion in der aktuellen Situation geeignet war, um die gestellte Aufgabenstellung zu lösen. "The essence of RL is learning through interaction. An RL agent interacts with its environment and, upon observing the consequences of its actions, can learn to alter its own behaviour in response to rewards received. This paradigm of trial-and-error learning has its roots in behaviorist psychology and is one of the main foundations of RL.“ [7].

Zu Beginn des Lernprozesses besitzt der Agent kein Wissen über die Umgebung und kann Aktionen nur zufällig auswählen. Durch die während des Lernens erhaltenen Belohnungen entwickelt der Agent eine Strategie, um immer besser geeignete Aktionen auszuführen, um dadurch die erhaltenen Belohnungen zu maximieren.

Beim simulationsgestützten Ansatz lernt der Agent durch Interaktion mit einem Simulationsmodell anstelle von der realen Umgebung. Dies bietet im Falle der Reglerauslegung im Wesentlichen drei Vorteile:

1.Die durch das Prinzip von Versuch und Irrtum gene-

rierten Stellgrößen können zu keiner Beschädigung des zu regelnden Systems führen.

2. Die Entkopplung von physikalischen Randbedingungen ermöglicht eine stark beschleunigte Simulation der Regeleingriffe und damit auch des Lernverfahrens.

3. Die Reglerauslegung kann bereits zu einem Zeitpunkt erfolgen, in dem das reale System noch nicht bereitsteht.

Da das bestärkende Lernen zur Steuerung des Lernprozesses die erhaltenen Belohnungen nutzt, besteht hier die besondere Herausforderung darin, eine für den konkreten Anwendungsfall geeignete Belohnungsfunktion zu definieren.

Der in dieser Untersuchung genutzte Algorithmus für das bestärkende Lernen basiert auf der Methode Actor-Critic. "Actor-critic approaches achieve a high performance because they require minimal computation for action selections even in case of very high or infinite number of possible actions. Actor-critics can also learn an explicitly stochastic policy which is very useful in continuous learning problems." [8]. In Abbildung 2 ist der Informationsfluss bei der Methode Actor-Critic innerhalb des Agenten dargestellt und die Interaktion des Agenten mit der Umgebung abgebildet.

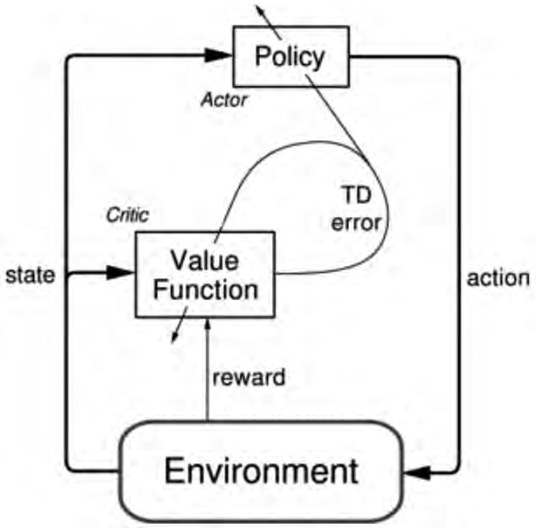

Abbildung 2: Informationsfluss bei der Methode Actor-Critic [9]

Der Aktor (Actor) erhält den Zustand (state) von der Umgebung (Environment) und führt abhängig von seiner Strategie (Policy) eine Aktion (action) aus. Die Strategie bildet damit das erlernte Verhalten des Aktors ab. „A policy defines the learning agent's way of behaving at a given time. Roughly speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. […] The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior.“ [9]

Der Kritiker (Critic) evaluiert die Strategie des Aktors mit einer Wertfunktion (Value Function). Mit der Ausgabe des Kritikers wird das Lernverfahren beim Aktor und beim Kritiker durchgeführt. „Almost all reinforcement learning algorithms are based on estimating value functions--functions of states (or of state-action pairs) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state). The notion of "how good" here is defined in terms of future rewards that can be expected, or, to be precise, in terms of expected return. Of course the rewards the agent can expect to receive in the future depend on what actions it will take. Accordingly, value functions are defined with respect to particular policies.“ [9]

3. Auslegung des neuronalen Reglers

Der neuronale Regler wird mit der Programmiersprache Python und dem Machine Learning Framework Tensorflow ausgelegt. Unabhängig davon, ob das RL simulationsgestützt oder am physikalischen Modell des Pendels erfolgt, besteht der zu entwickelnde neuronale Regler aus zwei künstlichen neuronalen Netzen, eins für den Aktor und eins für den Kritiker. Als physikalisches Modell wird der Quanser QUBE-Servo 2 genutzt (siehe Abbildung 3). Das Setup für das Training des neuronalen Reglers ist unabhängig von der Anwendung auf dem physikalischen Modell oder dem Simulationsmodell. Vor diesem Hintergrund werden die hierzu notwendigen Schritte im Folgenden anhand des Setups am physikalischen Modell erklärt.

3.1. Architektur der künstlichen neuronalen Netze

Der Aktor und der Kritiker verwenden für die künstlichen neuronalen Netze dieselbe Architektur. Es wird jeweils ein vollvermaschtes mehrschichtiges Perzeptron (MLP) mit zwei verborgenden Schichten mit je 64 Einheiten und der tanh-Aktivierungsfunktion verwendet. Die künstlichen neuronalen Netze werden auf einem Computer mit dem Betriebssystem Ubuntu 18.10 und der CPU Intel CORE i7-7700HQ trainiert.

3.2. Proximal Policy Optimization

Für das Training der künstlichen neuronalen Netze wird der Algorithmus Proximal Policy Optimization (PPO) in der Implementierung PPO2 von der Organisation OpenAI genutzt. Bei der Implementierung des Algorithmus werden die Hyperparameter aus der Publikation von PPO [10] und von OpenAI verwendet.

Das künstliche neuronale Netz des Aktors repräsentiert die Strategie und nimmt den Zustand des inversen Pendels als Input, d.h. die Winkelposition des Arms und des Pendels, sowie jeweils die Winkelgeschwindigkeit und -beschleunigung. Als Output gibt das künstliche neuronale Netz einen Wert aus, der mittels Controller als Eingangsspannung an dem Gleichstrommotor des inversen Pendels anliegt.

Das künstliche neuronale Netz des Kritikers stellt die Wertfunktion dar und schätzt die zukünftigen Belohnungen bei Einhaltung der Strategie ab. Mit der Ausgabe der Wertfunktion und der erhaltenen Belohnung wird das neuronale Netz des Kritikers angepasst und mit dem Algorithmus PPO das neuronale Netz des Aktors.

Der Algorithmus PPO stellt dabei sicher, dass sich die neue Strategie des Aktors nicht zu stark von der alten Strategie unterscheidet, damit es nicht zu Einbrüchen beim Lernprozess kommt.

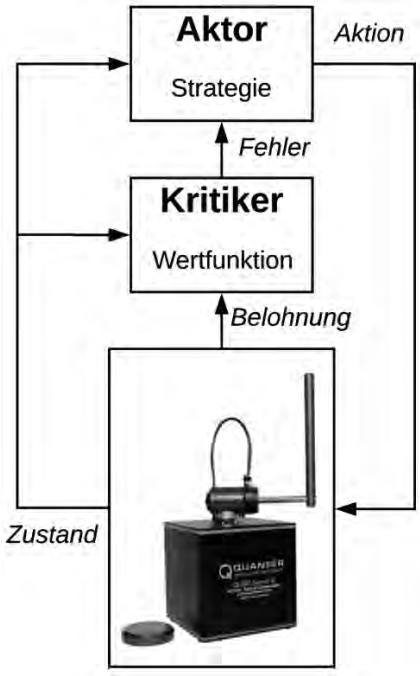

Abbildung 3: Aufbau des neuronalen Reglers und des Quanser QUBE-Servo 2

3.3. Definition der Belohnungsfunktionen

Es muss eine geeignete Funktion zur Berechnung der Belohnungen definiert werden, um den neuronalen Regler für das beabsichtigte Verhalten zu trainieren. Das Ziel ist, dass der neuronale Regler das Pendel aufschwingen und in der instabilen Ruhelage ausbalancieren kann. Der Arm soll beim Ausbalancieren in der instabilen Ruhelage mittig liegen.

Die Belohnungsfunktion wird mit dem Zustand des inversen Pendels definiert. Dabei wird die Winkelposition des Arms (THETA) und die Winkelposition des Pendels (ALPHA) genutzt. THETA kann einen Wert von bis zu 120° annehmen, da der Bewegungsbereich des Arms durch Anschläge begrenzt ist. Ein Vorzeichen gibt an, ob der Arm sich nach links oder rechts bewegt und wird bei der Berechnung der Belohnung nicht berücksichtigt. Bei 0° ist der Arm zentriert. ALPHA hat einen Wert von 0°, wenn das Pendel senkrecht nach oben zeigt, und 180°, wenn das Pendel nach unten zeigt. Ein Vorzeichen gibt an, ob das Pendel sich nach rechts oder links bewegt und wird nicht berücksichtigt. Zur einfacheren Berechnung und Definition der Belohnungsfunktion wird der Betrag von ALPHA und THETA gebildet und die Werte im Bereich von [0, 1] skaliert.

Die Abläufe beim Aufschwingen und Ausbalancieren des inversen Pendels sind sehr verschieden. Dies wird auch beim Training des neuronalen Reglers berücksichtigt, indem dieser in unterschiedlichen Phasen trainiert wird. In der ersten Phase des Trainings soll der neuronale Regler lernen, das Pendel in der instabilen Ruhelage auszubalancieren. Dabei befindet sich das Pendel zu Beginn in der instabilen Ruhelage und soll dort vom neuronalen Regler ausbalanciert werden. Wenn das Pendel die instabile Ruhelage verlässt, wird das Training unterbrochen und das Pendel wird von einem konventionellen Regler in die instabile Ruhelage aufgeschwungen, wo das Training wieder beginnt. Um das beabsichtigte Verhalten zu trainieren, wird bei der Belohnungsfunktion für diese Phase der Winkel des Pendels (ALPHA) deutlich höher gewichtet als der Winkel des Arms (THETA). Dadurch kann der neuronale Regler eine höhere Belohnung bekommen, wenn das Pendel aufrecht steht als wenn der Arm zentriert ist. Dementsprechend passt der neuronale Regler sein Verhalten bei der Regelung des inversen Pendels an. Nach einer Trainingsdauer von 3,5 Stunden kann der neuronale Regler das Pendel in der instabilen Ruhelage ausbalancieren. Die Belohnungsfunktion in (1) wird verwendet.

Belohnung = 1 - 0,9 * |ALPHA| - 0,1 * |THETA| (1)

In der zweiten Phase soll der neuronale Regler lernen, das Pendel in die instabile Ruhelage aufzuschwingen. Dabei ist das Pendel anfangs in der stabilen Ruhelage, d.h. nach unten gerichtet, und soll in die instabile Ruhelage aufgeschwungen werden. Wenn das Pendel für 10 Sekunden in der instabilen Ruhelage ausbalanciert wurde, wird das Training unterbrochen und das Pendel kehrt wieder in die stabile Ruhelage zurück, wo das Training wieder beginnt. Nach einer Trainingsdauer von 4,8 Stunden kann der neuronale Regler das inverse Pendel zuverlässig in die instabile Ruhelage aufschwingen. Die Belohnungsfunktion in (1) wird dabei verwendet.

Nach der zweiten Phase neigt der Arm dazu, in Richtung der Anschläge zu driften, wenn der neuronale Regler das Pendel ausbalanciert. Daher ist eine dritte Trainingsphase erforderlich. Das Ziel in der dritten Phase ist, dass der Arm beim Ausbalancieren mittig bleibt und nicht mehr in Richtung der Anschläge driftet. Dabei ist das Pendel in der instabilen Ruhelage und wird durch den neuronalen Regler ausbalanciert. Wenn das Pendel die instabile Ruhelage verlässt, wird das Training unterbrochen und das Pendel wird durch einen konventionellen Regler in die instabile Ruhelage aufgeschwungen, wo das Training wieder beginnt. In (2) ist die Belohnungsfunktion für die dritte Phase definiert. Der Winkel des Arms wird höher als zuvor gewichtet, um das beabsichtigte Verhalten mit dem neuronalen Regler zu trainieren.

Belohnung = 1 - 0,7 * |ALPHA| - 0,3 * |THETA| (2)

Nach ca. 20 Minuten Trainingszeit werden in der dritten Phase gute Ergebnisse mit dem neuronalen Regler erzielt. Insgesamt dauert das Training ca. 9 Stunden, wobei 2,7 Millionen Aktionen während des Trainings vom neuronalen Regler auf dem inversen Pendel ausgeführt wurden. Im Ergebnis kann der neuronale Regler das inverse Pendel aus der stabilen Ruhelage in die instabile Ruhelage aufschwingen und dort ausbalancieren. Somit kann der hier entworfene neuronale Regler zur Regelung des inversen Pendels genutzt werden.

4. Vergleich KI-basierte Reglerauslegung mit und ohne Simulationsmodell

Aufgrund der bereits unter Kapitel 2 aufgeführten Vorteile eines simulationsgestützten Ansatzes, konnte die benötigte Zeitpanne für das Training eines neuronalen Reglers am Simulationsmodell im Verglich zum Training an dem realen System sehr deutlich verkürzt werden. Dies führt vor allem dazu, dass unterschiedliche Algorithmen und Parameter für den Entwurf und das Training eines neuronalen Reglers bei der simulationsgestützten Vorgehensweise schnell und kostengünstig evaluiert werden können.

Weiterhin hat sich gezeigt, dass aggressive Lernstrategien beim Training am physikalischen Modell mit einem erheblichen Risiko von Beschädigungen des Systems oder sogar Verletzungen von Menschen einhergehen.

Ein weiterer Vorteil beim Training am Simulationsmodell ist, dass kritische Situationen häufiger simuliert werden können, als sie tatsächlich in der Realität vorkommen. Dadurch kann der neuronale Regler das korrekte Verhalten in der Simulation lernen, bevor diese kritischen Situationen an einem realen System geregelt werden müssen. Dadurch kann die Robustheit der Regelung gesteigert werden.

Ein Nachteil der simulationsgestützten Auslegung entsteht durch Abweichungen zwischen dem Simulationsmodell und dem realen zu regelnden System. Bei den durchgeführten Untersuchungen hat sich gezeigt, dass geringfügige Vereinfachungen im Simulationsmodell schnell dazu führen können, dass die daran trainierten neuronalen Regler nicht auf das zu regelnde physikalische System übertragen werden können.

Die Vorteile einer simulationsgestützten Herangehensweise müssen daher Anwendungsfall spezifisch mit den durch eine notwendige hohe Modellgenauigkeit erforderlichen Aufwänden abgeglichen werden.

5. Vergleich zwischen dem neuronalen Regler und einem konventionellen Regler

Das Verhalten des trainierten neuronalen Reglers wird mit dem Verhalten eines konventionellen Reglers bei der Regelung des inversen Pendels verglichen. Ein konventioneller Regler wird von Quanser, dem Hersteller des genutzten inversen Pendels, für Matlab Simulink zur Verfügung gestellt [11].

Der konventionelle Regler nutzt beim Aufschwingen des Pendels einen energiebasierten Regelungsansatz und zum Ausbalancieren einen PD-Regler (siehe Abbildung 4).

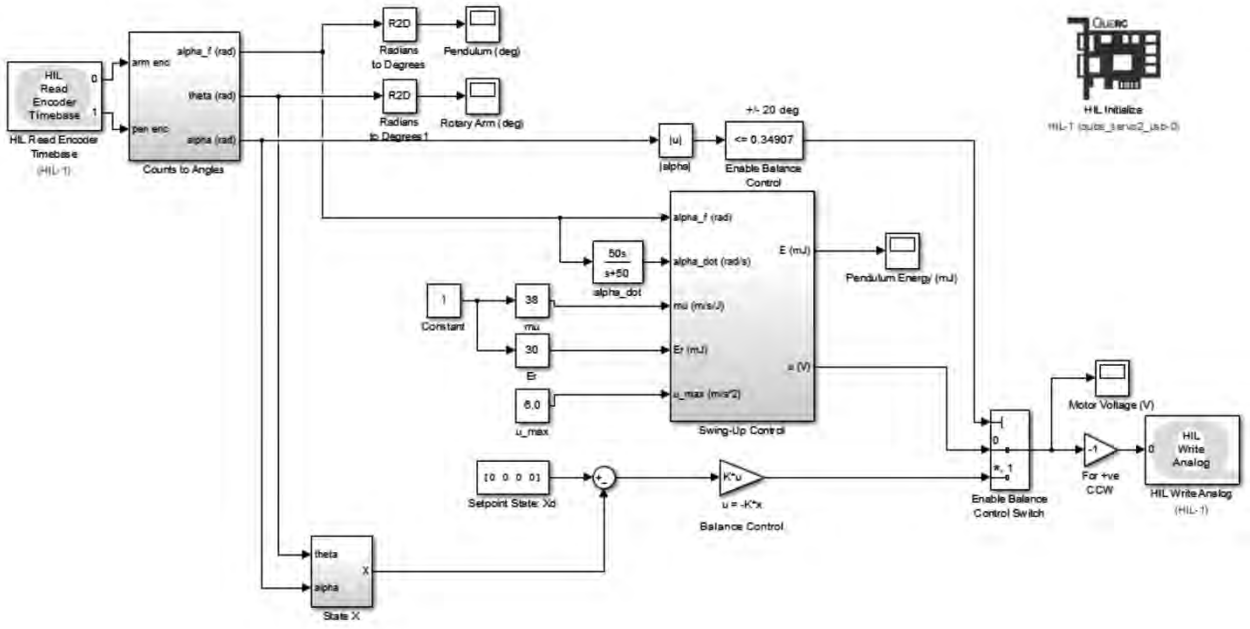

Abbildung 4: Aufbau konventioneller Regler in Matlab Simulink [11]

5.1. Verhalten des konventionellen Reglers

In Abbildung 5 ist das Verhalten des konventionellen Reglers bei der Regelung des inversen Pendels dargestellt.

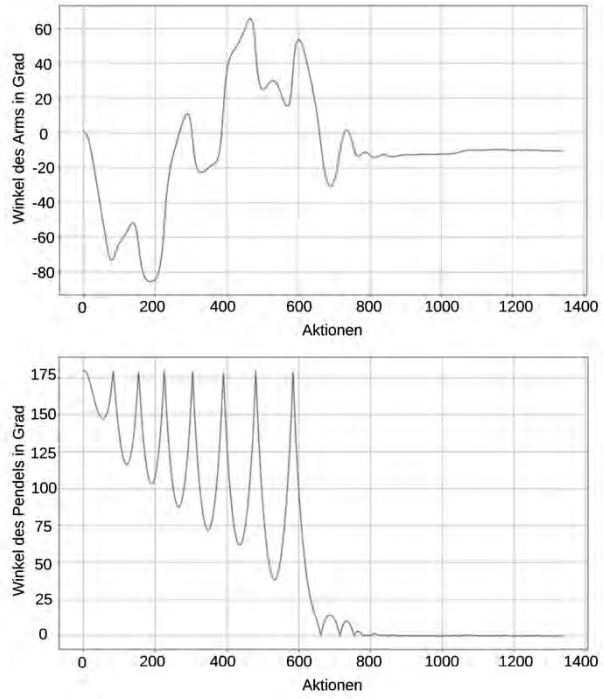

Abbildung 5: Verhalten des konventionellen Reglers

Zunächst ist der Arm mittig und hat einen Winkel von 0°. Der Regler lenkt den Arm ungleichmäßig nach links und rechts aus, um das Pendel aufzuschwingen. Das Pendel befindet sich zu Beginn in der stabilen Ruhelage mit einem Winkel von 180° und zeigt nach unten. Um den Winkel des Pendels in der Abbildung darzustellen, wird das Vorzeichen nicht berücksichtigt. Das Pendel wird vom Regler immer weiter ausgelenkt, bis das Pendel die instabile Ruhelage bei 0° erreicht, in der es ausbalanciert wird. Der Regler benötigt ca. 700 Aktionen, bis das Pendel die instabile Ruhelage erreicht hat. Beim Ausbalancieren ist der Arm um ca. 10° außerhalb der Mitte.

5.2. Verhalten des neuronalen Reglers

Das Verhalten des neuronalen Reglers bei der Regelung des inversen Pendels ist in Abbildung 6 dargestellt. Es ist zu erkennen, dass sich das Verhalten des neuronalen Reglers vom Verhalten des konventionellen Reglers deutlich unterscheidet. Der Arm wird gleichmäßig ausgelenkt, um das Pendel aus der stabilen Ruhelage in die instabile Ruhelage aufzuschwingen. Das Pendel erreicht die instabile Ruhelage nach ca. 400 Aktionen und damit deutlich schneller als mit dem konventionellen Regler. Der Arm wird anschließend mittig ausgerichtet und erreicht schließlich einen Winkel von 0°. Der neuronale Regler zeigt bei der Regelung des inversen Pendels das mit den Belohnungsfunktionen beabsichtigte Verhalten. In den Belohnungsfunktionen wird der Winkel des Pendels höher gewichtet als der Winkel des Arms. Daher ist

es für den neuronalen Regler wichtiger, das Pendel in die instabile Ruhelage aufzuschwingen und dort auszubalancieren. Der Winkel des Arms wird weniger stark gewichtet, daher wird der Arm erst zentriert, wenn sich das Pendel in der instabilen Ruhelage befindet.

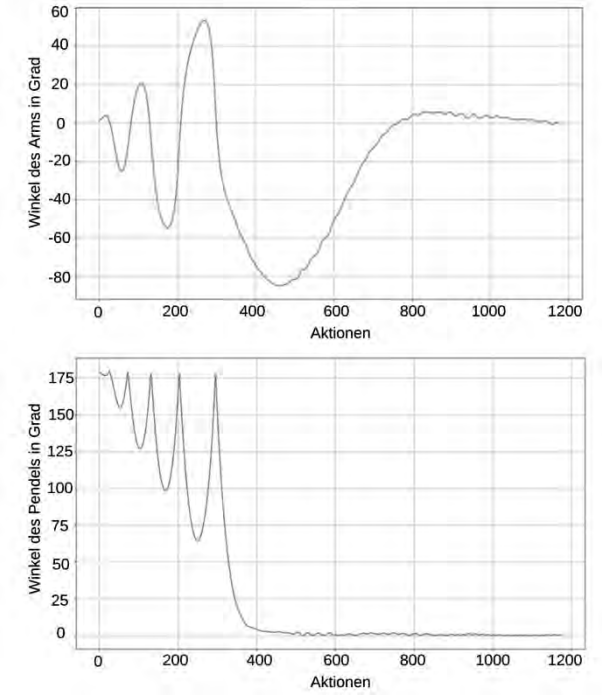

Abbildung 6: Verhalten des neuronalen Reglers

6. Fazit und Ausblick

In diesem Beitrag wurde gezeigt, wie ein neuronaler Regler für die Regelung eines rotierenden inversen Pendels mithilfe von maschinellen Lernverfahren ausgelegt werden kann. In der Gegenüberstellung des Regelverhaltens ist der durch maschinelles Lernen erzeugte Regler einem konventionellen Regler ebenbürtig. Des weiteren wurde verdeutlicht, welche Vorteile, aber auch Herausforderungen sich durch den simulationsgestützten Entwurf neuronaler Regler ergeben.

Die durchgeführten Untersuchungen lassen deutliches Potenzial simulationsgestützter Auslegungen von Regelungen mithilfe maschinellen Lernens erkennen, da sehr leistungsfähige Rechensysteme heute ein paralleles und schnelles Training verschiedenster neuronaler Regler ermöglichen.

Referenzen

1. Bundesministerium für Wirtschaft und Energie, Technologieszenario „Künstliche Inteligenz in der Industrie 4.0”, 2019
2. D. M. Charney and G. M. Josin, "Neural network servo control of a robot manipulator joint in real-time," 1991 IEEE International Joint Conference on Neural Networks, Singapore, 1991, pp. 1989-1994
3. S. K. Suman, M. K. Gautam, R. Srivastava and V. K. Giri, "Novel approach of speed control of PMSM drive using neural network controller," 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), Chennai, 2016, pp. 2780- 2783
4. N. Yadaiah, A. K. Priya and G. T. Ram Das, "Design of Neural Hysteris Band PWM Current Controller," 2006 1ST IEEE Conference on Industrial Electronics and Applications, Singapore, 2006, pp. 1-5
5. D. V. Samokhvalov and I. S. Nosirov, "Feed drive of the metal cutting machines with neural network controller," 2017 XX IEEE International Conference on Soft Computing and Measurements (SCM), St. Petersburg, 2017, pp. 373-375
6. Teng Fong, Tang & Jamaludin, Z. & Abdullah, Lokman. (2014). SYSTEM IDENTIFICATION AND MODELLING OF ROTARY INVERTED PENDULUM. International Journal of Advances in Engineering & Technology. 6. 2342-2353.
7. K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," in IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, Nov. 2017.
8. Marochko, Vladimir. (2017). Pseudorehearsal in actor-critic agents
9. R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, Cambridge, MA:MIT Press, 2018
10. John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.
11. Quanser CUBE Servo 2, https://www.quanser.com/products/qube-servo-2/