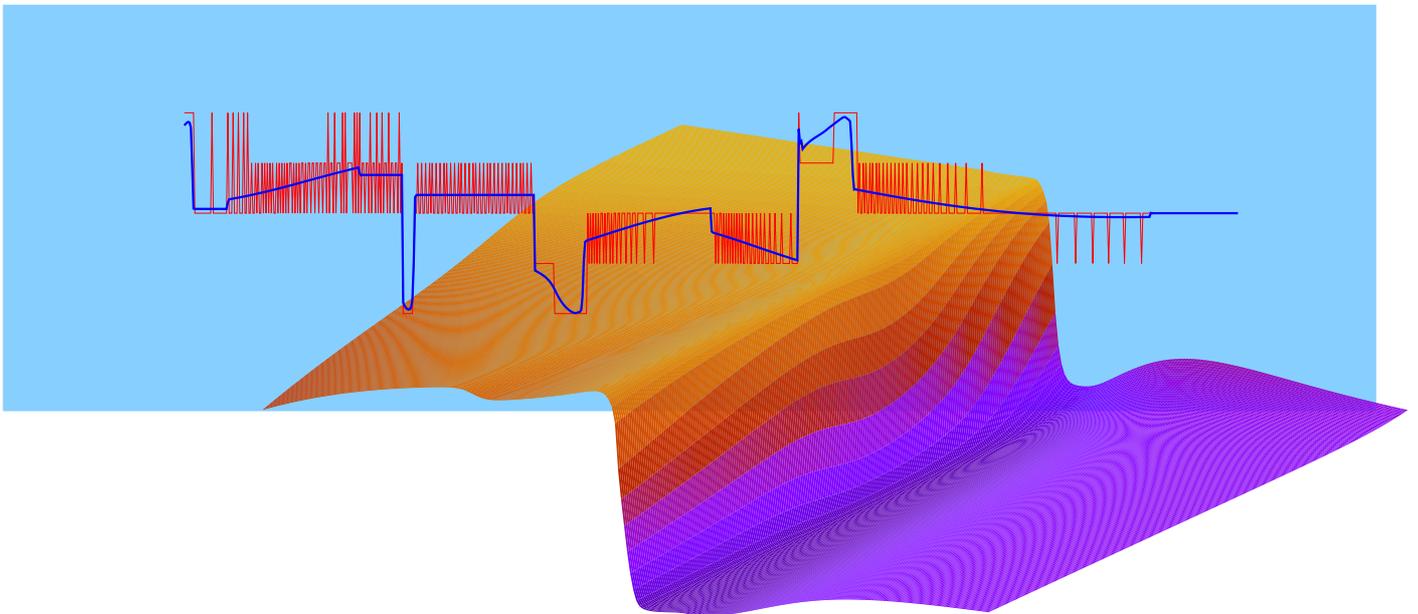


Dateneffiziente selbstlernende neuronale Regler

- Dissertation -
zur Erlangung des Doktorgrades
der Naturwissenschaften

Vorgelegt beim
Fachbereich Mathematik / Informatik
Universität Osnabrück



von

Dipl. Inform. Roland Hafner

Osnabrück 2009

Erstgutachter und Betreuer : Prof. Dr. Martin Riedmiller
Universität Osnabrück
Osnabrück
Fachbereich Mathematik / Informatik

Abstract

Die vorliegende Arbeit untersucht den Entwurf und die Anwendung selbstlernender Regler als intelligente Reglerkomponente im Wirkungsablauf eines Regelkreises für regelungstechnische Anwendungen. Der aufwändige Prozess der Analyse des dynamischen Systems und der Reglersynthese, welche die klassischen Entwurfsmuster der Regelungstechnik benötigen, wird dabei ersetzt durch eine lernende Reglerkomponente. Diese kann mit sehr wenig Wissen über den zu regelnden Prozess eingesetzt werden und lernt direkt durch Interaktion eine präzise Regelung auf extern vorgegebene Führungsgrößen. Der Lernvorgang basiert dabei auf einem Optimierungsprozess mit einem leistungsfähigen Batch-Reinforcement-Lernverfahren, dem 'Neural Fitted Q-Iteration'. Dieses Verfahren wird auf seine Verwendung als selbstlernender Regler untersucht. Für die in den Untersuchungen festgestellten Unzulänglichkeiten des Verfahrens bezüglich der geforderten präzisen, zeitoptimalen Regelung werden verbesserte Vorgehensweisen entwickelt, die ebenfalls auf ihre Leistungsfähigkeit untersucht werden. Für typische regelungstechnische Problemstellungen sind die diskreten Aktionen des NFQ-Verfahrens nicht ausreichend, um eine präzise Regelung auf beliebige Führungsgrößen zu erzeugen. Durch die Entwicklung einer Erweiterung des NFQ für kontinuierliche Aktionen wird die Genauigkeit und Leistungsfähigkeit der selbstlernenden Regler drastisch erhöht, ohne die benötigte Interaktionszeit am Prozess zu erhöhen. An ausgewählten Problemen der Regelung linearer und nichtlinearer Prozesse wird die Leistungsfähigkeit der entwickelten Verfahren empirisch evaluiert. Es zeigt sich dabei, dass die hier entwickelten selbstlernenden Regler mit wenigen Minuten Interaktionszeit an einem Prozess eine präzise Regelungsstrategie für beliebige externe Führungsgrößen lernen, ohne dass Expertenwissen über den Prozess vorliegt.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Zielsetzung und Aufbau der Arbeit	3
1.2.1	Zielsetzung	3
1.2.2	Aufbau der Arbeit	5
2	Methoden des Reinforcement-Lernens	9
2.1	Grundlagen des Reinforcement-Lernens	9
2.1.1	Markovscher Entscheidungsprozess	10
2.1.2	Die Strategie	11
2.1.3	Lösung des Optimierungsproblems	13
2.1.4	Batchverfahren im Reinforcement-Lernen	16
2.2	Künstliche neuronale Netze	17
2.2.1	Mehrschichtige neuronale Netze	17
2.2.2	Lernen im mehrschichtigen neuronalen Netz	19
2.2.3	Resilient Propagation (RProp)	19
2.2.4	Weitere Eigenschaften neuronaler Netze	20
2.3	Neural Fitted Q Iteration	21
2.3.1	Die Approximation der Q-Funktion mit neuronalen Netzen	22
2.3.2	Aktualisierung der Q-Funktion	25
2.3.3	NFQ als Lernverfahren	26
2.3.4	Praktische Aspekte bei der Anwendung von NFQ	28
2.4	Zusammenfassung	30
3	Selbstlernende Regler als Optimierungsprozess mit NFQ	31
3.1	Grundzüge der Regelungstechnik	32
3.1.1	Gütekriterien für die Regelung	33
3.1.2	Dynamische Systeme	34
3.2	Modellierung selbstlernender Regler als Optimierungsprozess mit NFQ	39
3.2.1	Modellierung des Optimierungskriteriums	40
3.2.2	Modellierung des Lernprozesses	43
3.2.3	Das Konditionieren der Q-Funktion	45
3.3	Korrektur des systematischen Fehlers in der Q-Funktion	46
3.3.1	Herleitung des Verfahrens	47
3.3.2	Arbeitsweise und Leistungsfähigkeit	48

3.4	Differenzierbare Kostenfunktionen	50
3.4.1	Probleme und Leistungsfähigkeit der Bereichsdefinition	50
3.4.2	Zeitoptimale Regelung mit differenzierbaren Kostenfunktionen	52
3.5	Die Regelung mit Führungsgröße	56
3.5.1	Modellierung des Lernprozesses	57
3.5.2	Lernen unter Zielwertveränderung	58
3.6	Verbesserung des Regelverhaltens durch Erweiterung der Systemdynamik	62
3.6.1	Dynamische Ausgabeelemente	62
3.6.2	Intelligente dynamische Ausgabeelemente	65
3.7	Zusammenfassung	68
4	Die Erweiterung von NFQ zur Verwendung kontinuierlicher Aktionen	71
4.1	Voruntersuchungen	72
4.1.1	NFQ in dynamischen Systemen mit kontinuierlichen Eingriffsgrößen	72
4.1.2	Die nichtlineare Dynamik eines Unterwasserfahrzeuges	74
4.1.3	NFQ am Beispiel des Unterwasserfahrzeuges	77
4.1.4	NFQ und kontinuierliche Aktionen	80
4.1.5	Diskussion	82
4.2	Kontinuierliche Aktionen durch Suche in der Q-Funktion	83
4.2.1	Minimums-Suche mittels Gradientenabstieg	83
4.2.2	NFQG: Ein Verfahren für kontinuierliche Aktionen	85
4.2.3	NFQG mit expliziter neuronaler Strategie	88
4.3	NFQCA	92
4.3.1	Die Verwendung von Aktor- und Kritiker-Komponente	92
4.3.2	Das Dreiecks-Netz	93
4.3.3	Die Aktualisierung des Aktors	94
4.3.4	Das NFQCA-Verfahren	96
4.4	Zusammenfassung	101
5	Regelungstechnische Versuche	103
5.1	Geschwindigkeitsregelung eines DC-Motors	104
5.1.1	Geschwindigkeitsregelung mit kontinuierlichen Aktionen	105
5.1.2	Versuche mit variabler externer Last	109
5.1.3	Robustheit gegen Rauschen	112
5.1.4	Diskussion	115
5.2	Autopilot für Verkehrsflugzeug	118
5.2.1	Entwurf eines Vergleichsreglers mittels LQR-Methode	119
5.2.2	Selbstlernender Regler zur Nickwinkelregelung	120
5.2.3	Diskussion	123
5.3	Aktive Schwingungs-Dämpfung	126
5.3.1	Lernender Regler zur Schwingungsdämpfung	129
5.3.2	Diskussion	134
5.4	Elektromagnetisches Schweben einer Metallkugel	134
5.4.1	Selbstlernender Regler zum Schweben der Stahlkugel	137
5.4.2	Die Annäherung der Trajektorienregelung	141
5.4.3	Diskussion	142
5.5	Die Regelung einer Heizspirale	142

5.5.1	Selbstlernender Regler zur Regelung der Heizspirale	145
5.5.2	Lernen unter veränderlichen Umgebungsvariablen	145
5.5.3	Diskussion	147
5.6	Zusammenfassung	148
6	Einbettung in den wissenschaftlichen Kontext	151
6.1	Batch-Verfahren im Reinforcement-Lernen	151
6.2	Reinforcement-Lernverfahren mit kontinuierlichen Aktionen	153
6.2.1	NFQCA als 'Adaptive Critic Design'	156
6.3	Reinforcement-Lernverfahren in regelungstechnischen Anwendungen	157
6.4	Zusammenfassung	160
7	Zusammenfassung und Ausblick	163
7.1	Überblick	163
7.1.1	Entwurfsaufwand selbstlernender Regler	166
7.2	Ausblick	166
7.2.1	MDPs mit externen Aktionen	166
7.2.2	Erweiterung der Lernverfahren	167
7.2.3	Erweiterung des Lernprozesses	169
7.3	Fazit	170
	Literaturverzeichnis	171

Einleitung

1.1 Motivation

Der Fortschritt in den Naturwissenschaften und der Technik führt zu immer komplexer werdenden Wirkungsgefügen, die in technischen Systemen beachtet werden müssen. Um das Potential der erreichten Fortschritte voll auszuschöpfen, müssen bei der gezielten Beeinflussung dieser Prozesse durch informationsverarbeitende Systeme immer mehr Wechselwirkungen einbezogen werden, um die vorhandenen Ressourcen auszunutzen. Dies wiederum erhöht die Komplexität der benötigten Lösung. Die gleiche Konsequenz erzeugt die Notwendigkeit, immer mehr Systeme mit ihrer Umwelt interagieren zu lassen. Diese Entwicklung hat zur Folge, dass der benötigte Entwurfsaufwand für eine annähernd optimale Lösung kaum mehr beherrschbar ist, da alle möglichen Zustände des Systems und deren störungsbedingte Abweichung vorab bedacht werden müssen. Für die konventionelle Programmierung bedeutet dies, dass sich der Mensch ein Domänenwissen über die Eigenschaften des Systems aneignen muss, um eine geeignete Lösung zu erzeugen. Im Umkehrschluss ist die erzeugte Lösung jedoch auf die vom Menschen erfassten Teilaspekte des Problems beschränkt und anfällig für Fehler und Vereinfachungen, die bei der Akquirierung dieses Wissens gemacht wurden. Durch die Entwicklungen im maschinellen Lernen ist abzusehen, dass zukünftige informationsverarbeitende Systeme in zunehmendem Maße nicht mehr von Hand programmiert sein werden. Sie werden immer mehr lernende Komponenten besitzen, welche die komplexen Zusammenhänge der Systeme selbstständig erfassen und dadurch handhabbar machen. Im Bereich der Analyse großer Datenmengen sind die Methoden des maschinellen Lernens bereits heute ein unverzichtbarer Bestandteil, ohne die viele praktische Anwendungen in der Bioinformatik, Prognoseverfahren und Data-Mining nicht möglich wären. Für die aufgabengemäße Beeinflussung eines komplexen Systems sind diese Analyseverfahren allein jedoch nicht ausreichend, da sie die Zusammenhänge der Daten erfassen, aber das System selbst nicht beeinflussen können. Für diese Aufgabenstellung bietet das maschinelle Lernen verschiedene Ansätze. Beim überwachten Lernen werden Beispiele für eine geeignete Beeinflussung des Systems vom Benutzer vorgegeben. Das Ziel der Methoden des Maschinellen Lernens ist die Entdeckung der Zusammenhänge auf den gesehenen Beispielen und die sinnvolle Generalisierung für Situationen, die nicht gesehen wurden. Ein Nachteil dieser Methoden ist es, dass eine Lösung für das Problem bekannt sein muss, um geeignete Beispiele zu generieren. Zudem kann die von den lernenden Methoden erzeugte Lösung nur so gut sein, wie sie in den Beispielen vorgegeben wurde. Eine interessante Alter-

native zu diesem Vorgehen sind die selbstlernenden Methoden. Diese basieren auf den Paradigmen des 'Reinforcement Learning' und des dynamischen Programmierens [13, 15, 16, 47]. Der Begriff 'Reinforcement Learning' (RL) kann im deutschen Sprachgebrauch nur unzureichend mit verstärkendem Lernen übersetzt werden und soll im Rahmen dieser Arbeit deshalb als Reinforcement-Lernen bezeichnet werden. Es bezeichnet das ursprünglich biologisch motivierte Lernen aus Belohnung und Bestrafung. Im Unterschied zum überwachten Lernen müssen vom Benutzer keine Beispiele über die Lösung des Problems vorgegeben werden, sondern nur das zu erreichende Ziel in Form eines Kostensignals. Die Verfahren des RL lernen eine entsprechend dieser Kosten optimierte Beeinflussung durch Interaktion mit dem vorliegenden System. In Abbildung 1.1 ist die Grundidee der selbstlernenden Verfahren des maschinellen

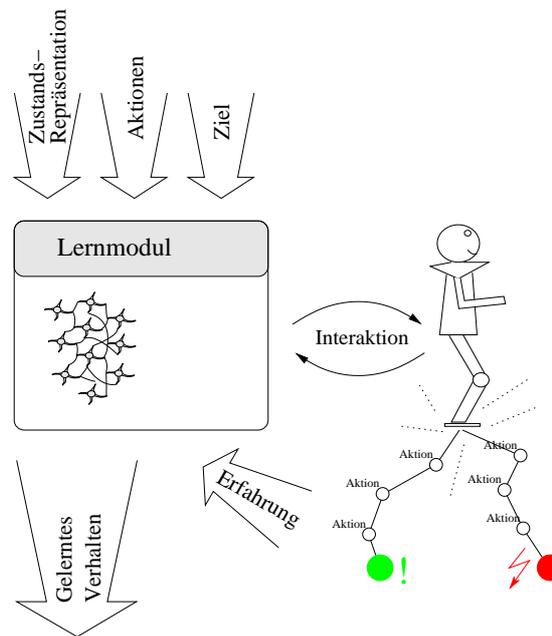


Abbildung 1.1: Die grundlegende Idee des Reinforcement-Lernens ist das Lernen durch Interaktion am System. Dem Lernmodul wird das zu erreichende Ziel, nicht aber die Lösung des Problems vorgegeben.

Lernens schematisch dargestellt. Die durch Interaktion mit dem zu beeinflussenden System gesammelte Erfahrung kann ausschließlich aufgrund der Vorgabe des zu erreichenden Zieles und einer geeigneten Repräsentation des Systems für das Erlernen eines geeigneten Verhaltens dienen. Die selbstlernenden Verfahren des RL sind somit eine vielversprechende Grundlage für die Erzeugung eines autonomen Verhaltens für komplexe Systeme, für das der Benutzer kein Wissen über die genauen Wirkungszusammenhänge und dessen störungsbedingte Abweichungen besitzt. Durch die Entwicklung dateneffizienter Verfahren im Reinforcement-Lernen wurde in den letzten Jahren eine drastische Verbesserung in Bezug auf die benötigte Interaktionszeit mit dem zu beeinflussenden System erreicht. Diese wurden um viele Größenordnungen reduziert und liegen nun in einem für das Lernen an realen Systemen realistischen Bereich. Mit der Entwicklung des 'Neural Fitted Q-Iteration'-Verfahren (NFQ) [79] wurde ein dateneffizientes Reinforcement-Lernverfahren entwickelt, das in der Lage ist, mit sehr wenig Vorwissen und unter Verwendung standardisierter Modellierungshilfsmitteln, eine breite Palette an verschiedenen Problemstellungen zu lösen. Die Leistungsfähigkeit von NFQ wurde in verschiedenen Anwendungen bereits demonstriert. Unter Anderem wurde das Verfahren vom Autor

auf dem ersten internationalen 'Reinforcement Learning Benchmark'-Wettbewerb [40, 103] in einer regelungstechniknahen Problemklasse eingesetzt, wo es als leistungsfähigstes Verfahren ausgezeichnet wurde.

Im Fokus dieser Arbeit steht die Untersuchung der Leistungsfähigkeit des NFQ für praktische Anwendungen im Bereich der Beeinflussung technischer Systeme, wie sie in der Robotik, der Automatisierungstechnik und der Prozesssteuerung vorkommen. Speziell diese technischen Systeme werden durch Fortschreiten der Entwicklung immer komplexer. Durch die parallele Weiterentwicklung der Verarbeitungsgeschwindigkeit und die Miniaturisierung von Computern können immer mehr Technologien aus dem Bereich des maschinellen Lernens direkt in realen Anwendungen integriert werden. Allgemein können die Methoden des maschinellen Lernens auf verschiedenen Ebenen zur Erzeugung eines autonomen Verhaltens eines komplexen Systems beitragen [29, 56, 83]. Zumeist werden mehrere Verfahren kombiniert um ein komplexes autonomes Verhalten zu generieren. Diese Arbeit konzentriert sich dabei auf die Beeinflussung eines Prozesses als regelungstechnische Problemstellung. Die präzise und optimale Regelung ist eine grundlegende Voraussetzung, welche die aufgabengemäße Funktionsweise vieler Systeme erst möglich macht. Dies gilt sowohl für die Funktionsweise mechatronischer Geräte des Alltags wie z.B. Laufwerken oder Automobilen, als auch für komplexe Maschinen in der Automatisierungstechnik und Robotik. Der Entwurfsprozess klassischer Regler ist ein aufwändiger Prozess, der ein hohes Maß an Expertenwissen benötigt. In einer Analyse wird die Struktur des Prozesses durch einen Entwickler, basierend auf dessen physikalischen Hintergrundwissen, mathematisch formuliert. Anschließend werden die genauen Parameter und physikalischen Eigenschaften dieser Modellierung am realen System identifiziert und auf dem entstandenen Modell eine Reglersynthese durchgeführt. Dieser gesamte Prozess benötigt nicht nur ein hohes Maß an Expertenwissen, sondern ist für viele Prozesse auch nicht ohne Informationsverlust durchzuführen. Der Einsatz des maschinellen Lernens in Form der selbstlernenden Regler auf Basis des NFQ soll diesen Prozess ersetzen. Mit nur sehr wenig Vorwissen soll es möglich sein, einen selbstlernenden Regler auf einen Prozess anzuwenden. Somit wird das Maß an Expertenwissen für die Erzeugung eines Reglers reduziert. Durch den durchgeführten Optimierungsprozess können Regelgesetze gefunden werden, welche die Ressourcen neuer Technologien voll ausschöpfen, ohne dass die erzeugten Lösungen durch die Komplexität der Wirkungszusammenhänge eingeschränkt werden.

1.2 Zielsetzung und Aufbau der Arbeit

1.2.1 Zielsetzung

Mit der vorliegenden Arbeit sollen selbstlernende Regler als intelligente Reglerkomponenten für den Einsatz in geschlossenen Regelkreisen entwickelt werden. Der Entwurfsaufwand der klassischen Reglersynthese für die aufgabengemäße Prozessbeeinflussung im Regelkreis wird dabei ersetzt durch eine lernende Komponente, die aus Erfahrung von vorangegangenen Regelungsvorgängen ein geeignetes Regelverhalten erlernt. Der Entwurf und die Untersuchung der selbstlernenden Regler basiert auf dem Lernparadigma des Reinforcement-Lernens mit neuronalen Netzen. Ausgehend von einem leistungsfähigen Reinforcement-Lernverfahren, dem Neural Fitted Q-Iteration (NFQ), wird die Modellierung der betrachteten regelungstechnischen Problemstellung als Optimierungsproblem diskutiert. Dabei wird insbesondere die Integration der aufgabengemäßen Beeinflussung des Prozesses unter einer extern vorgegebenen Führungsgröße betrachtet. Dies ist eine grundlegende Eigenschaft der zu entwickelnden selbstlernenden

Regler, die es ermöglicht, sie im Wirkungskreis der klassischen Regelungstechnik als Reglerkomponente einzusetzen. Die vorliegende Arbeit hat die Zielsetzung zeitoptimale und präzise Regelstrategien durch den Optimierungsprozess des Reinforcement-Lernens zu erhalten. Das zeitoptimale Verhalten der Regelung wird dadurch charakterisiert, dass sie bei Änderungen des vorgegebenen Zielwertes oder bei auftretenden Störungen, das gewünschte Verhalten des Prozesses in möglichst kurzer Zeit wieder sicherstellt. Von einer präzisen Regelung wird erwartet, dass sie die zu beeinflussende Regelgröße möglichst nahe an den vorgegebenen Zielwert heranhöhrt. Neben den Anforderungen der präzisen und schnellen Beeinflussung des Prozesses unter beliebigen Zielwerten, soll der Aufwand des Benutzers für die Anwendung der selbstlernenden Regler auf einen zu regelnden Prozess möglichst klein sein. Diese Forderung beinhaltet zum einen, dass das benötigte Wissen des Benutzers über den zu beeinflussenden Prozess für die Formulierung der Regelungsaufgabe möglichst gering sein soll. Zum anderen darf die Anzahl der benötigten Regelungsvorgänge am Prozess nicht zu hoch werden, damit die Ressourcen des Prozesses schnell wieder produktiv sein können.

Für die zu entwickelnden Verfahren und durchzuföhrenden Untersuchungen bedeutet dies im Einzelnen:

Präzise zeitoptimale Regelung Die bestehende zeitoptimale Formulierung des Optimierungskriteriums ermöglicht eine einfache, standardisierte Spezifikation des Regelungszieles für unterschiedliche Prozesse und Regelungsziele. Da die Formulierung des Optimierungskriteriums kein Wissen über die Systemgleichungen des Prozesses verwendet, benötigt es kein weiteres Vorwissen, das vom Benutzer über den zu regelnden Prozess akquiriert werden muss. Ein Nachteil der Formulierung besteht darin, dass die Genauigkeit der Regelung nur bis auf einen relativ großen Toleranzbereich um die Führungsgröße möglich ist. Eine präzise Regelung im Toleranzbereich ist nicht möglich. Hierzu ist die Entwicklung einer alternativen Formulierung des Reinforcement-Signals für eine zeitoptimale und präzise Regelung notwendig. Diese soll die zeitoptimalen Eigenschaften außerhalb des Toleranzbereiches beibehalten, darf jedoch kein zusätzliches Vorwissen über den Prozess notwendig machen.

Optimierungsproblem mit unendlichem Horizont Der Vorgang der Regelung ist ein andauernder Prozess, der nicht nach einer gewissen Zeit endet. Die Forderung nach einem zeitoptimalen Verhalten des zu lernenden Reglers bedingt dabei einen nicht beschränkten Optimierungshorizont für das Reinforcement-Lernen. Da eine Diskontierung des Kostensignals der Grundidee eines zeitoptimalen Verhaltens widerspricht, soll sie in dieser Arbeit nicht verwendet werden. Eine bestehende Lösung für die Probleme, die sich für das Lernverfahren daraus ergeben, benötigt einen hohen Grad an Vorwissen über das System und ist nicht allgemein anwendbar. Es soll ein neues Verfahren entwickelt werden, das kein Vorwissen über den zu regelnden Prozess benötigt.

Regelung auf beliebige Führungsgrößen Im klassische Wirkungsablauf der Regelung wird eine dem Regler von außen vorgegebene Führungsgröße verwendet, bezüglich derer der Regler den Prozess beeinflussen muss. In der Formulierung bestehender lernender Regler mit den Methoden des Reinforcement-Lernens wird diese externe Führungsgröße zumeist vernachlässigt, als konstant angesehen oder durch zusätzliche externe Reglerstrukturen behandelt. Es soll eine Modellierung entwickelt werden, die es erlaubt diese externe Führungsgröße im Lernprozess des Reinforcement-Lernens zu beachten und es

den selbstlernenden Regler erlaubt den Prozess bezüglich beliebiger Führungsgrößen zu regeln.

Beeinflussung des Prozesses mit kontinuierlichen Eingriffsgrößen Die Verwendung des NFQ-Verfahrens für die Erzeugung selbstlernender Regler bedingt die Einschränkung der lernbaren Regelstrategien auf die Verwendung einer diskreter Menge von möglichen Eingriffsgrößen. Die Verwendung dieser Regelstrategien mit diskreten Eingriffsgrößen hat negative Auswirkungen auf die präzise und zeitoptimale Regelung, die untersucht werden sollen. Da sich diese negativen Auswirkungen auch durch die Verwendung zusätzlicher Verfahren für allgemeine Prozesse nicht vollständig beheben lassen, ist eine Erweiterung des NFQ-Verfahrens auf die Verwendung kontinuierlicher Aktionen notwendig. Es soll daher ein dateneffizientes Reinforcement-Lernverfahren mit kontinuierliche Aktionen entwickelt werden, das es erlaubt eine sehr präzise Regelung auf beliebigen Prozessen zu lernen ohne die Anzahl der benötigten Interaktionen mit dem Prozess zu erhöhen.

Empirische Untersuchungen an exemplarischen Problemstellungen Die Diskussion der bestehenden Methoden und deren Nachteile, sowie die Darstellung des erreichten Fortschritts durch die neu entwickelten Verfahren soll durch empirische Untersuchungen an exemplarischen Problemstellungen der Regelungstechnik begleitet werden.

Empirische Untersuchung der Leistungsfähigkeit selbstlernender Regler Ein weiterer Schwerpunkt der Arbeit soll auf der generellen Untersuchung der Leistungsfähigkeit der entwickelten Verfahren für die Erzeugung selbstlernender Regler für ausgewählte Problemstellungen aus der regelungstechnischen Praxis liegen. Im Gegensatz zu den empirischen Untersuchungen der erzielten Fortschritte im Vergleich zu den bestehenden Verfahren soll der Schwerpunkt hier auf der Untersuchung der Relevanz der entwickelten selbstlernenden Regler für den praktischen Einsatz liegen.

1.2.2 Aufbau der Arbeit

Die Arbeit gliedert sich in 7 Kapitel. Die Grundlagen für die Entwicklung der selbstlernenden Regler werden in Kapitel 2 gegeben. Ausgehend von den Grundlagen des Reinforcement-Lernens und der sequentiellen Entscheidungsprobleme, die ihnen als mathematische Formulierung zugrunde liegt, werden die für diese Arbeit relevanten Aspekte vorgestellt und diskutiert. Basierend auf den Grundlagen der neuronalen Netze und der dateneffizienten Weiterentwicklung der RL-Verfahrenen wird das NFQ-Verfahren zur Lösung allgemeiner Problemstellungen vorgestellt. Durch die Verwendung der neuronalen Netze in NFQ unter der Idee des dateneffizienten RL ergibt sich ein leistungsfähiges Verfahren, dessen grundlegenden Aspekte bei der praktischen Verwendung beim Lernen durch Interaktion mit einem Prozess diskutiert werden. Die neuronalen Netze ermöglichen die Anwendung des Verfahrens auf Prozesse mit kontinuierlichen Zustandskodierungen, was das Verfahren für den Einsatz in selbstlernenden Reglern interessant macht.

Die Grundlagen der betrachteten regelungstechnischen Problemstellung wird in Kapitel 3 betrachtet. Aus der Analyse der geforderten Gütekriterien für die Regelung werden die zugrunde liegenden Problemklassen in Form dynamischer Systeme vorgestellt. Aus dieser Betrachtungsweise lassen sich die Anforderungen an die selbstlernenden Regler formalisieren und in die Modellierung des Optimierungsprozesses mit NFQ übertragen. Ausgehend von der

klassischen Modellierung der stochastischen kürzesten Pfadprobleme wird der Problemtyp der zeitoptimalen Regelung vorgestellt und die Auswirkungen auf das Lernverfahren untersucht. Die sich aus der Untersuchung ergebenden Unzulänglichkeiten für den Entwurf der selbstlernenden Regler unter der Zielsetzung der Arbeit werden herausgearbeitet. Dies sind vordringlich, dass sich der Problemtyp der Regelung nur durch Integration von Vorwissen über den Prozess realisieren lässt, um eine Degeneration der Q-Funktion zu verhindern und sich durch die zeitoptimale Formulierung des Optimierungskriteriums keine präzise Regelung ergibt. Für beide Unzulänglichkeiten des NFQ-Verfahrens werden Lösungen entwickelt, welche eine zentrale Basis zur Erreichung der Zielsetzung der Arbeit darstellen. Diese entwickelten Methoden erlauben eine höhere Genauigkeit bei der zeitoptimalen Regelung ohne die benötigte Interaktionszeit mit dem System zu erhöhen und ohne Vorwissen über den Prozess zu integrieren. Wie in den empirischen Untersuchungen gezeigt wird verbessert sich dadurch ebenfalls die Stabilität des Lernvorganges. Die selbstlernenden Regler werden auf die Modellierung des vollständigen Problems der Regelung unter einer externen Führungsgröße untersucht. Durch die Anpassung der Modellierung ist das Lernen unter dieser externen Führungsgröße möglich und stellt sich als effiziente Möglichkeit der Erzeugung einer intelligenten Reglerkomponente in einem Regelkreis dar. Die Leistungsfähigkeit des Verfahrens und der entwickelten Anpassungen werden anhand eines nichtlinearen Regelungsproblem untersucht. Ein zentrales Merkmal des NFQ-Verfahrens ist die Verwendung einer diskreten Menge von Ausgaben, aus der das Verfahren in jedem Zeitschritt eine Auswahl trifft, um den Prozess zu beeinflussen. Die Beschränkung der realisierbaren Regelungsstrategien durch diese diskreten Aktionen beeinträchtigt die erzielbare Genauigkeit und die gesamte Regelgüte der selbstlernenden Regler. Durch die Verwendung von dynamischen Ausgabeelementen wird ein Konzept vorgestellt, das dieses Problem handhabbar macht. Speziell durch die in dieser Arbeit entwickelte Erweiterung der dynamischen Ausgabeelemente zur Integration klassischer Regelgesetze lässt sich eine deutlich erhöhte Genauigkeit der Regler erzeugen.

In Kapitel 4 wird die Auswirkungen der diskreten Aktionen auf Regelungsprobleme mit kontinuierlichen Stellgrößen weiter untersucht. An einem Beispiel der Regelung eines Unterwasserfahrzeuges wird gezeigt, dass für eine optimale Regelung die Verwendung von kontinuierlichen Aktionen eine unabdingbare Voraussetzung für eine effektive und genaue Regelung darstellt. Anhand von NFQ wird die Erweiterung des Verfahrens auf die Verwendung kontinuierlicher Aktionen untersucht. Das daraus entwickelte Neural Fitted Q-Iteration mit kontinuierliche Aktionen (NFQCA) ist ein leistungsstarkes Verfahren, das die Verwendung kontinuierlicher Ausgaben des Reglers zulässt und direkt eine neuronale Regelungsstrategie lernt. Mit dem Entwurf des NFQCA, basierend auf den Entwicklungen in Kapitel 3, kann die Zielsetzung der Arbeit vollständig erreicht werden. Die Einführung der kontinuierlichen Aktionen bedingt dabei keine Erhöhung der benötigten Interaktionszeit und benötigt kein zusätzliches Vorwissen über den Prozess.

Eine ausführliche Untersuchung der Leistungsfähigkeit der entwickelten Verfahren wird in Kapitel 5 anhand von ausgewählten Beispielen der regelungstechnischen Praxis durchgeführt. Die verwendeten regelungstechnischen Probleme dienen als empirische Grundlage zur Bewertung der Leistungsfähigkeit der entwickelten Verfahren. In der gegebenen Form können sie als neuer Typ von Benchmarks für das Reinforcement-Lernen dienen, die eine Reihe von interessanten Eigenschaften besitzen, die in dieser Form in klassischen Benchmarks wenig Beachtung finden. Der Fokus der Untersuchung liegt auf der Anwendung der vorgestellten selbstlernenden Regler und der Untersuchung der erzielbaren Regelgüte auf anspruchsvolle regelungstechnische Problemstellung. Zusätzlich werden praktische Problemstellungen der Regelungs-

technik wie externe Störungen, Robustheit und Anwendbarkeit der Regler untersucht.

Das Kapitel 6 stellt die Einbettung der vorgestellten Arbeiten in den aktuellen wissenschaftlichen Kontext vor. Hierbei liegt der Schwerpunkt vor allem auf der Eingliederung des NFQ- und des entwickelten NFQCA-Verfahren in existierende Arbeiten. Ein Überblick über die Verwendung des Reinforcement-Lernens in regelungstechniknahen Problemstellungen zeigt, dass die hier betrachtete Zielsetzung für die selbstlernenden Regler ein selten betrachtetes Problem darstellt für das wenige vergleichbare Arbeiten existieren. Die Arbeit endet mit einer Zusammenfassung der erreichten Fortschritte und der durchgeführten Untersuchungen in Kapitel 7, das auch einen Ausblick auf weitere Arbeiten gibt.

In Abbildung 1.2 ist die Struktur der vorliegenden Arbeit schematisch dargestellt.

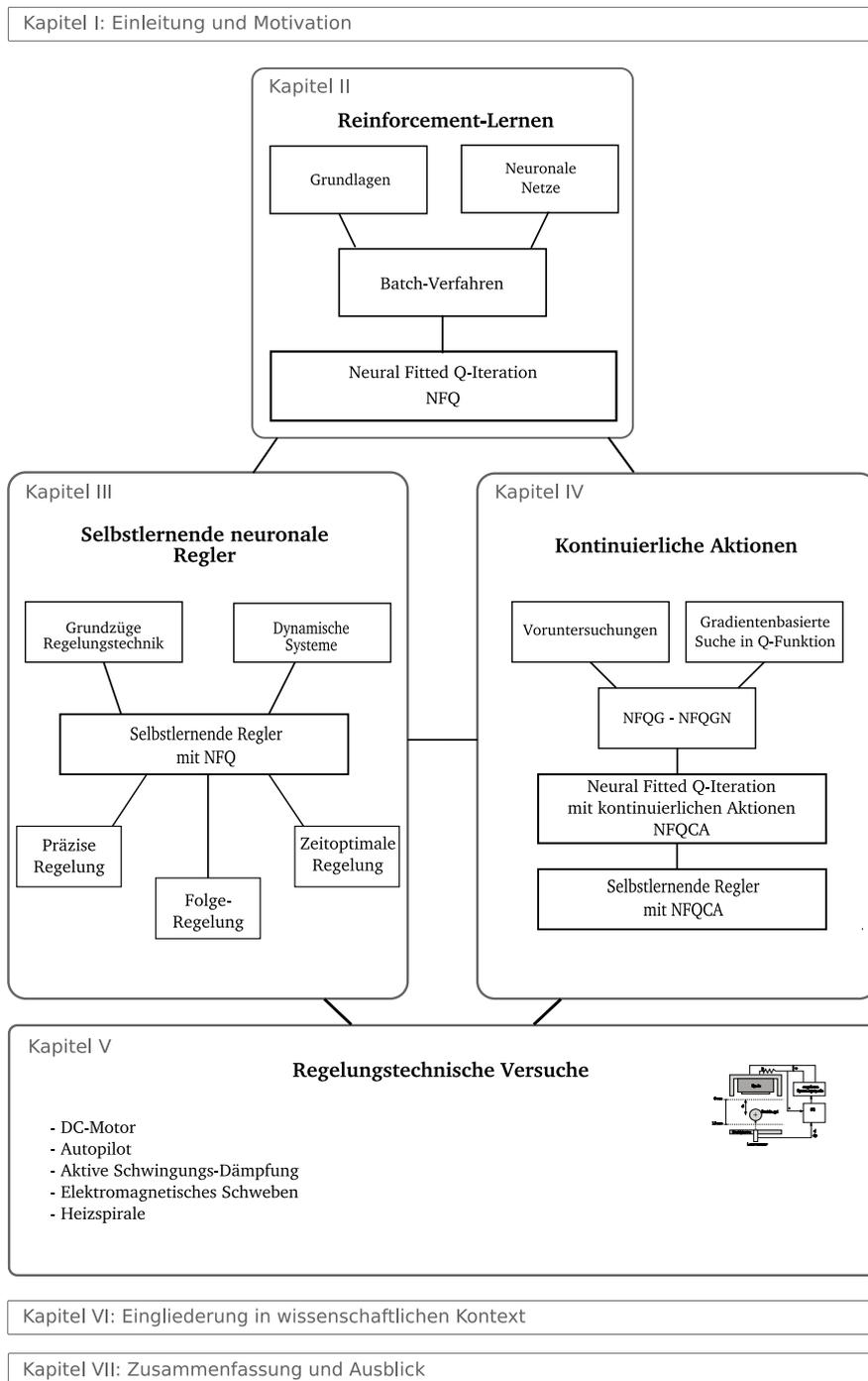


Abbildung 1.2: Der Aufbau der vorliegenden Arbeit gliedert sich in 7 Kapitel. Die drei Hauptkapitel behandeln die Erstellung selbstlernender Regler und die Untersuchung ihrer Leistungsfähigkeit an Problemstellungen der regelungstechnischen Praxis.

Methoden des Reinforcement-Lernens

Das vorliegende Kapitel präsentiert die Grundlagen der in dieser Arbeit verwendeten Methoden und Verfahren des Reinforcement-Lernens, welche die Basis der selbstlernenden Regler bilden. Hierzu werden in Abschnitt 2.1 die mathematische Modellierung von sequentiellen Entscheidungsproblemen und die Formulierung des im Reinforcement-Lernen betrachteten Optimierungsproblems vorgestellt. Die grundlegenden Verfahren des dynamischen Programmierens zur Lösung des Optimierungsproblems und das Q-Lernen als Methode des modellfreien Lernens werden in ihren Grundzügen behandelt. Durch die Einführung der Batch- bzw. Fitted-Methoden im wertfunktionsbasierten Reinforcement-Lernen konnten in aktuellen Arbeiten große Fortschritte im Hinblick auf die Erstellung dateneffizienter Verfahren erreicht werden (2.1.4). In Abschnitt 2.2 werden die für diese Arbeit relevanten Eigenschaften der mehrschichtigen neuronalen Netze vorgestellt. Diese dienen als allgemeine, nichtlineare Funktionsapproximatoren, für die leistungsstarke Lernmethoden zur Verfügung stehen. In Abschnitt 2.3 wird das dieser Arbeit zugrunde liegende 'Neural Fitted Q-Iteration'-Verfahren diskutiert. Dies ist ein dateneffizientes Verfahren zum modellfreien Reinforcement-Lernen, basierend auf der Generalisierungsfähigkeit der neuronalen Netze und der Idee der Batchverfahren.

2.1 Grundlagen des Reinforcement-Lernens

Das Reinforcement-Lernen, motiviert durch das Lernen aus Belohnung und Bestrafung, ist ein auf dem Dynamischen Programmieren basierendes Verfahren zur Lösung von sequentiellen Entscheidungsproblemen. Ziel des Lernens ist die Erzeugung eines Verhaltens, das die Gesamtheit der während der Ausführung des sequentiellen Entscheidungsprozesses auftretenden Kosten minimiert bzw. die auftretenden Belohnungen maximiert. In diesem Sinne wird im Reinforcement-Lernen die Optimierung einer Entscheidungsfunktion in Bezug auf eine Zielfunktion betrachtet. Im Gegensatz zu anderen Lernverfahren ist diese Zielfunktion nicht vollständig bekannt. Vielmehr ist sie definiert über ein Kosten- bzw. Belohnungssignal, das nach jeder Entscheidung anfällt und in Kombination mit dem sequentiellen Entscheidungsproblem über mehrere Entscheidungen die eigentlich zu minimierende bzw. maximierende Zielfunktion ergibt.

Im Reinforcement-Lernen sind sowohl die Betrachtung von Kostensignalen als auch die Betrachtung von Belohnungssignalen verbreitet. In dieser Arbeit wird durchgängig von einem Kostensignal ausgegangen und somit eine Minimierung durchgeführt.

2.1.1 Markovscher Entscheidungsprozess

Markovsche Entscheidungsprozesse (Markov Decision Processes, MDPs) sind eine allgemeine mathematische Formulierung für zeitdiskrete stochastische Entscheidungsprobleme. Sie sind in einem weiten Bereich von Optimierungsproblemen anwendbar, die mit Methoden des Dynamischen Programmierens und des Reinforcement-Lernens gelöst werden. Durch einen MDP wird ein sequenzielles Entscheidungsproblem definiert, bei dem ein Agent mit seiner Umwelt in diskreten Zeitschritten interagiert. Der Agent ist dabei eine schematische Sichtweise eines Verfahrens zur Entscheidungsfindung. In jedem Zeitschritt beobachtet der Agent den Zustand des MDPs und trifft eine Entscheidung. Diese Entscheidung lässt den MDP im folgenden Zeitschritt in einen neuen Zustand übergehen, den der Agent beobachten kann. Dieser Übergang im MDP wird als eine stochastische Übergangsfunktion modelliert. Zusätzlich erhält der Agent in jedem Zeitschritt ein Kostensignal, das vom beobachteten Zustand und der gewählten Aktion abhängig ist. In Abbildung 2.1 ist das zur Motivation des Reinforcement-Lernens oft verwendete Agentenmodell dargestellt, das diesen Zusammenhang verdeutlicht. Formal wird ein MDP charakterisiert durch ein 5-Tupel:

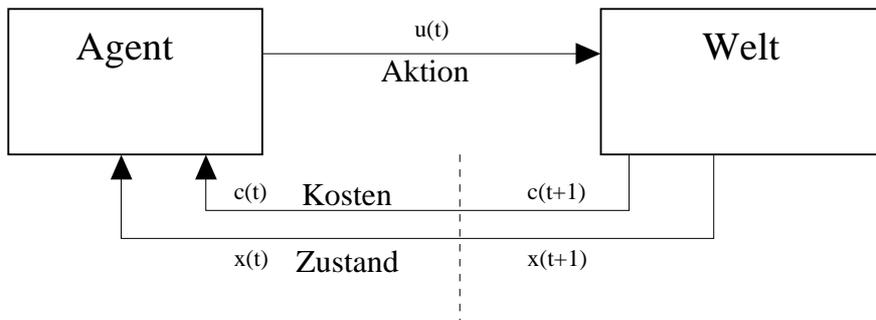


Abbildung 2.1: In der Formulierung des Reinforcement-Lernens interagiert ein Agent mit seiner Umwelt in diskreten Zeitschritten. In jedem Zeitschritt wählt er eine Aktion und beobachtet im darauf folgenden Zeitschritt die Auswirkung der gewählten Aktion in Form eines Folgezustandes und eines Kostensignals.

$$(T, X, U, P_u(x, x'), c(x, u)) \quad (2.1)$$

T ist die Menge der diskreten Entscheidungszeitpunkte des sequentiellen Entscheidungsproblems. X ist die Menge der möglichen Zustände, in der sich der MDP in jedem Zeitschritt befinden kann. Zu jedem diskreten Zeitschritt t beobachtet der Agent einen Zustand $x_t \in X$ und wählt eine Aktion $u_t \in U$ aus der Menge der dem Agenten zur Verfügung stehenden Aktionen. Die stochastischen Übergänge im MDP werden als Wahrscheinlichkeitsverteilung der Folgezustände, abhängig von der angewendeten Aktion modelliert. Dabei definiert $P_u(x, x')$ die Wahrscheinlichkeit, dass der Agent nach Anwendung der Aktion u im Zustand x in den Folgezustand x' übergeht.

$$P_u(x, x') = Pr(x_{t+1} = x' | x = x_t, u = u_t) \quad (2.2)$$

Um einen MDP zu modellieren, muss diese Übergangsfunktion die sogenannten Markov Eigenschaft besitzen. Bei Kenntnis des aktuellen Zustandes x_t und der Aktion u_t muss die Wahrscheinlichkeitsverteilung des Folgezustandes x_{t+1} eindeutig bestimmt sein. Formal lässt sich

dies für einen MDP als

$$\begin{aligned} P_u(x, x') &= Pr(x_{t+1} = x' | x = x_t, u = u_t) \\ &= Pr(x_{t+1} = x' | x = x_t, x_{t-1}, x_{t-2}, \dots, u = u_t, u_{t-1}, u_{t-2}, \dots) \end{aligned} \quad (2.3)$$

schreiben.

Das Kostensignal $c(x, u)$ beschreibt die unmittelbar auftretenden Kosten, die bei Anwendung der Aktion u im Zustand x entstehen. Das Ziel der Optimierung, das mit Methoden des Reinforcement-Lernens erreicht werden kann, bezieht sich nicht auf die Minimierung des direkten Kostensignals $c(x, u)$. Vielmehr wird die Summe der Übergangskosten über mehrere Zeitschritte betrachtet. Eine typische Funktion, die der Minimierung zu Grunde liegt, ist $\sum_{t \in T} \gamma^t c(x_t, u_t)$. Dabei ist $0 \leq \gamma \leq 1$ ein Diskontierungsfaktor, der den Einfluss der direkten Kosten aus früheren Zeitschritten bestimmt. Die in dieser Arbeit betrachteten Verfahren können sowohl auf diskontierte Probleme ($\gamma < 1$) als auch auf nicht diskontierte Probleme ($\gamma = 1$) angewandt werden. Für die betrachteten Problemstellungen ist die Verwendung eines Diskontierungsfaktors nicht notwendig. Aus diesem Grund wird zu Gunsten der Darstellung im folgenden $\gamma = 1$ gewählt.

2.1.2 Die Strategie

Die Entscheidungsfindung des Agenten wird als Strategie (engl. Policy) bezeichnet. In jedem Zeitschritt wählt dieser, abhängig vom aktuell beobachteten Zustand x , aus der Menge der zur Verfügung stehenden Aktionen U eine Aktion aus. Diese Abbildung von der Menge der Zustände in die Menge der Aktionen wird als Strategiefunktion $\pi(x)$ modelliert.

$$\text{Strategiefunktion } \pi : X \rightarrow U, \pi(x_t) = u_t, u_t \in U \quad (2.4)$$

Für Entscheidungsprobleme, die nicht in jedem Zustand die selben Aktionen zulassen, kann die Modellierung auf eine zustandsabhängige Aktionsmenge U_x verfeinert werden. Mit Hilfe dieser Verfeinerung können spezielle Eigenschaften eines MDPs modelliert werden. Ein Beispiel hierfür sind Brettspiele, bei denen nicht in jedem Zustand die gleichen Züge möglich sind.

In dieser Arbeit werden ausschließlich deterministische und stationäre Strategien betrachtet, wie sie in Formel 2.4 angegeben sind. Deterministisch bezeichnet dabei die Eigenschaft der Strategie, in einem bestimmten Zustand immer die selbe Aktion zu wählen. Im Unterschied dazu wird bei stochastischen Strategien die Auswahl der Aktion, basierend auf einer Wahrscheinlichkeitsverteilung über den zur Verfügung stehenden Aktionen, im aktuellen Zustand getroffen. Stationäre Strategien wählen die Aktion in einem Zustand unabhängig vom aktuellen Zeitschritt t des sequentiellen Entscheidungsprozesses. Nicht stationäre Strategien wenden verschiedene Strategien für verschiedene Zeitpunkte des sequenziellen Entscheidungsproblems an. Dies kann für Probleme mit einer fest definierten Anzahl von Entscheidungsschritten sinnvoll sein. So kann die Entscheidung des Agenten über die Durchführung eines Torschusses in einem Fußballspiel, abhängig von der Spielzeit zu Beginn des Spieles und direkt am Ende des Spieles, unterschiedlich ausfallen.

Durch die Strategiefunktion interagiert der Agent mit dem MDP. Die zu minimierende Zielfunktion kann nun über die Strategiefunktion des Agenten in Form eines Erwartungswertes der kumulierten Pfadkosten definiert werden, die ein Agent, beginnend in einem Zustand x_0 mit

Strategie π , erzielt.

$$J^\pi(x) = \mathbb{E} \left[\sum_{t \in T} c(x_t, \pi(x_t)) \right], x_0 = x, P_{u_t}(x_t, x_{t+1}) \quad (2.5)$$

Das Ziel der Optimierung ist es nun, aus der Menge der möglichen Startegien Π eine optimale Strategie π^* zu finden, die die erwarteten kumulierten Pfadkosten für jeden Zustand minimiert: $J^{\pi^*}(x) = \min_{\pi \in \Pi} J^\pi(x)$.

Problemtypen

Der Horizont eines sequentiellen Entscheidungsproblems bezeichnet die Anzahl der zu durchlaufenden Entscheidungsstufen. Man unterscheidet zwischen Problemen mit endlichem Horizont, die eine feste Anzahl von Entscheidungsstufen durchlaufen $T = \{1, \dots, N\}$ und Problemen mit unendlichem Horizont ohne zeitliche Begrenzung $T = \{1, \dots\}$. Bei Problemen mit endlichem Horizont betrachtet man N-stufige Entscheidungsprobleme. Zu jedem Zustand werden Terminalkosten $\hat{c}(x)$ festgelegt, die anfallen, wenn das sequentielle N-stufige Entscheidungsproblem in x endet. Die erwarteten Pfadkosten aus 2.5 werden für diese Probleme angepasst zu:

$$J_N^\pi(x) = \mathbb{E} \left[\hat{c}(x_N) + \sum_{t=0}^{N-1} c(x_t, \pi(x_t)) \right], x_0 = x, P_{u_t}(x_t, x_{t+1}) \quad (2.6)$$

Bei Problemen mit unendlichem Horizont können die zu einem Zustand zugehörigen Pfadkosten unbeschränkt sein. Daher wird für die Definition der kumulierten Pfadkosten ein Diskontierungsfaktor $\gamma \in (0, 1]$ eingeführt. Bei $\gamma < 1$ kann so sichergestellt werden, dass für jeden Zustand nur beschränkte Pfadkosten auftreten. Die erwarteten Pfadkosten für Probleme mit unendlichem Horizont ergeben sich als Grenzfall des N-stufigen Entscheidungsproblems zu:

$$J^\pi(x) = \mathbb{E} \left[\lim_{N \rightarrow \infty} \sum_{t=0}^N \gamma^t c(x_t, \pi(x_t)) \right], x_0 = x, P_{u_t}(x_t, x_{t+1}) \quad (2.7)$$

Ein weiterer Problemtyp mit unendlichem Horizont sind die Stochastischen Kürzester Pfad (SKP)-Probleme¹. Bei diesen kann ohne die Verwendung eines Diskontierungsfaktors sichergestellt werden, dass für jeden Zustand die erwarteten Pfadkosten endlich bleiben. Um dies zu erreichen, wird die Existenz mindestens eines absorbierenden Terminalzustandes gefordert. Wird ein solcher Terminalzustand x^{term} besucht, treten einmalig die Terminalkosten $\hat{c}(x)$ auf. In darauffolgenden Zeitschritten kann der Terminalzustand durch keine Aktion mehr verlassen werden und jede Aktion im Terminalzustand erzeugt keine weiteren Kosten mehr. In einem SKP-Problem wird nun gefordert, dass mindestens einer dieser absorbierenden Terminalzustände existiert. Es soll angenommen werden, dass in einem SKP-Problem ein speziell ausgezeichnete Terminalzustand x^+ existiert, dessen Terminalkosten und damit auch seine Pfadkosten 0 sind. Somit hat jede Strategie, die x^+ erreicht, endliche Pfadkosten. Eine Strategie wird als erfüllend bezeichnet (proper policy), wenn durch ihre Anwendung, beginnend in jedem beliebigen Startzustand, der Terminalzustand x^+ in höchstens $\#X$ Schritten mit positiver Wahrscheinlichkeit erreicht wird. Somit führt eine erfüllende Strategie, startend in einem beliebigen Zustand, mit Wahrscheinlichkeit 1 (irgendwann) in den Terminalzustand und besitzt für

¹Der Sonderfall eines SKP-Problems, definiert auf einem deterministischen MDP, wird als (deterministisches) kürzester Pfad Problem bezeichnet.

jeden Zustand endliche Pfadkosten. Sowohl diskontierte Probleme mit unendlichem Horizont ohne Terminalzustand, als auch Probleme mit endlichem Horizont, können in SKP-Probleme umformuliert werden.

2.1.3 Lösung des Optimierungsproblems

Die Lösung des Optimierungsproblems bezieht sich auf die Bestimmung einer optimalen Strategie π^* für einen gegebenen MDP. Die Methoden des Dynamischen Programmierens sind in der Lage, die zur optimalen Strategie π^* gehörenden optimalen Pfadkosten J^* zu berechnen. Die Berechnung dieser optimalen Pfadkosten für jeden Zustand basieren auf dem Bellmanschen Optimalitätsprinzip. Dieses besagt, dass die optimalen Pfadkosten für einen Zustand gegeben sind durch:

$$J^*(x) = \min_{u \in U} \sum_{x' \in X} P_u(x, x')(c(x, u) + J^*(x')) \quad (2.8)$$

Aus dem Bellmanschen Optimalitätsprinzip folgt, dass für eine beliebige, aber festen Strategie π für die zugehörigen Pfadkosten J^π gelten muss $J^\pi(x) = \sum_{x' \in X} P_{\pi(x)}(x, x')(c(x, u) + J^\pi(x'))$. Unterstellt man die vollständige Kenntnis der Übergangswahrscheinlichkeiten $P_u(x, x')$ des MDPs, so können diese optimalen Pfadkosten und damit auch die optimale Strategie berechnet werden. Ein Verfahren hierzu ist das sogenannte Wertiterationsverfahren (engl. Value Iteration). Dies ist ein iteratives Verfahren, das, ausgehend von beliebig initialisierten Pfadkosten, die optimalen Pfadkosten für jeden Zustand berechnen kann. Hierzu wird die aus dem Bellmanschen Optimalitätsprinzip abgeleitete Iterationsgleichung verwendet. In

Algorithmus 1 Wertiterationsverfahren

$k \leftarrow 0$

Wähle J_0 beliebig

repeat

$\forall x \in X : J_{k+1}(x) = \min_{u \in U} \sum_{x' \in X} P_u(x, x')(c(x, u) + J_k(x'))$

$k \leftarrow k + 1$

until (konvergiert)

Algorithmus 1 ist das Wertiterationsverfahren angegeben. Das Wertiterationsverfahren konvergiert für SKP-Probleme unter zwei Voraussetzungen. Voraussetzung eins ist die Existenz zumindest einer erfüllenden Strategie. Die zweite Voraussetzung für die Konvergenz auf SKP-Problemen ist, dass die Pfadkosten für jede nicht erfüllende Strategie zumindest in einem Zustand unendlich werden. Die zweite Voraussetzung kann auch durch eine stärkere Forderung $c(x, u) > 0 \forall x \neq x^+, u \in U$ ersetzt werden, die besagt, dass alle direkten Kosten außerhalb des Terminalzustandes x^{term} größer Null sein müssen. Anstatt die Aktualisierung im Schritt k parallel auf allen Zuständen gleichzeitig durchzuführen, kann das Wertiterationsverfahren auch mit asynchronen Aktualisierungen durchgeführt werden. Für die Konvergenz des Verfahrens muss in jedem Zustand die Aktualisierung jedoch unendlich oft durchgeführt werden. Die optimale Strategie ergibt sich durch ein gierige (engl. greedy) Auswertung der optimalen Pfadkosten. Eine Strategie $\pi(x) = \operatorname{argmin}_{u \in U} \sum_{x' \in X} P_u(x, x')(c(x, u) + J(x'))$ wird als Gierige Strategie bezüglich der Kostenfunktion J bezeichnet.

Ein weiteres Grundverfahren des Dynamischen Programmierens zur Lösung des Optimierungsproblems für SKP-Probleme auf endlich vielen Zuständen und Aktionen ist das Strategieiterationsverfahren (engl. Policy Iteration). Beginnend mit einer Strategie π_0 iteriert das

Verfahren über zwei Schleifen. Die innere Schleife iteriert über alle Zustände, um die zu der aktuellen Strategie zugehörigen Pfadkosten zu ermitteln. Sind diese ermittelt, wird in einer äußeren Schleife ein Schritt zur Verbesserung der aktuellen Strategie durch Auswerten der aktuellen Pfadkostenfunktion nach der Bellman Gleichung durchgeführt. Ist die Strategie, mit der das Verfahren gestartet wird, erfüllend, so generiert es eine Sequenz von sich monoton verbessernden erfüllenden Strategien und konvergiert zur optimalen Strategie (siehe Algorithmus 2).

Algorithmus 2 Strategieiterationsverfahren

$h \leftarrow 0$

Wähle π_0 erfüllende Strategie

repeat

$k \leftarrow 0$

Bestimme die zu π_h zugehörigen Pfadkosten:

repeat

$$\forall x \in X : J_{k+1}^{\pi_h}(x) = \min_{u \in U} \sum_{x' \in X} P_u(x, x') (c(x, u) + J_k^{\pi_h}(x'))$$

$k \leftarrow k + 1$

until (J^{π_h} konvergiert)

Strategie Verbesserungs-Schritt:

$$\pi_{h+1}(x) = \operatorname{argmin}_{u \in U} \sum_{x' \in X} P_u(x, x') (c(x, u) + J_k^{\pi_h}(x'))$$

until (π konvergiert)

Von diskreten Zustandsmengen zu kontinuierlichen Zustandsräumen

Die vorgestellte Modellierung und die Verfahren zur Lösung des Optimierungsproblems sind definiert auf einer diskreten Menge von Zuständen und Aktionen. Für anspruchsvolle Probleme mit sehr vielen Zuständen sind sie in dieser Form nicht geeignet. Um für praxisnahe Problemstellungen geeignet zu sein, müssen in der Modellierung der MDPs die diskreten Zustandsmengen X ersetzt werden durch kontinuierliche Zustandsräume \mathcal{X} . Ein Zustand des MDPs ist nun ein Vektor mit reellwertigen Einträgen aus einem n -dimensionalen Zustandsraum $x \in \mathcal{X} \subset \mathbb{R}^n$.

Die Einführung von kontinuierlichen Zustandsräumen impliziert zwei Änderungen, die die Methoden des Reinforcement-Lernens von den vorgestellten Methoden des Dynamischen Programmierens unterscheiden. Die Menge der beobachtbaren Zustände, die der MDP annehmen kann, ist nicht mehr abzählbar. Zum Einen impliziert dies, dass die Wertfunktionen über dem Zustandsraum nicht mehr effizient dargestellt werden können. Zum Anderen kann die Aktualisierungsvorschrift in Algorithmus 1 und 2 nicht mehr über alle vorhandenen Zustände iteriert werden. Das Problem der Repräsentation der Pfadkosten bzw. einer Wertfunktion über dem kontinuierlichen Zustandsraum wird durch die Verwendung von geeigneten Funktionsapproximatoren gelöst. Da die Lernverfahren nicht in allen möglichen Zuständen ihre Aktualisierungsvorschrift anwenden können und somit auch Zustände existieren, die selbst nicht aktualisiert werden, spielt die Generalisierungsleistung dieser Approximatoren eine zentrale Rolle in der Entwicklung leistungsfähiger Reinforcement Lernverfahren.

Da die Lernverfahren nicht mehr über alle existierenden Zustände ihre Lernvorschrift iterieren können, geht man bei kontinuierlichen Zustandsräumen zum Lernen auf Trajektorien über. Der Agent interagiert mit dem MDP auf Trajektorien, indem er sich bezüglich seiner aktuellen

Strategie verhält. Im Fall des Wertiterationsverfahrens wertet er hierzu seine aktuelle Pfadkostenfunktion gierig aus. Die Trajektorien werden von einer Menge von Startzuständen aus durchgeführt. Das Lernen entlang von Trajektorien beschränkt die Aktualisierung der Wertfunktionen auf die in diesen Trajektorien durchlaufenen Zustände. Die Aktualisierung der Wertfunktion kann direkt nach dem Beobachten eines Übergangs oder nach Durchlaufen einer ganzen Trajektorie stattfinden. Dies ist für viele praxisnahe Problemstellungen eine effiziente Herangehensweise, da nur Zustände in die Aktualisierung eingehen, die auch im betrachteten Problem vorkommen können. Andere Zustände, die die zugrunde liegende Übergangsfunktion von relevanten Startzuständen aus nicht erreichen kann, werden nicht berücksichtigt. Das Wertiterationsverfahren auf Trajektorien kann als Wertiterationsverfahren mit asynchroner Aktualisierung der Wertfunktion betrachtet werden, das im Prinzip unter Kenntnis der Übergangsfunktion konvergiert. Die Konvergenz auf Trajektorien kann jedoch die optimale Kostenfunktion nur auf der Menge der während der Ausführung des Algorithmus unendlich oft besuchten Zustände gezeigt werden. Eine Bedingung, die für die Konvergenz zusätzlich eingehalten werden muss, ist, dass die initialen Pfadkosten J_0 die optimalen Pfadkosten J^* für alle Zustände unterschätzen. Diese Eigenschaft ist speziell bei der Verwendung von Funktionsapproximatoren für die Wertfunktionen nicht einfach zu garantieren. Es wird deshalb beim Lernen auf Trajektorien eine Exploration eingeführt, die durch zufällige Abweichung von der gierigen Strategie eine gewisse 'Umgebung' der Trajektorie im Zustandsraum erkundet. Eine typische Vorgehensweise im Reinforcement-Lernen ist, dass in jedem Schritt mit einer gewissen Wahrscheinlichkeit (der Explorationsrate ϵ) von der Gierigen Strategie abgewichen wird und eine rein zufällige Aktion gewählt wird. Auf diese Weise werden Zustände, die durch eine schlechte Initialisierung der Wertfunktion fehlerhaft als unattraktiv erscheinen, dennoch besucht.

Lernen ohne Modell

Die Kenntnis des vollständigen Modells der Übergangsfunktionen im MDP mit ihren Übergangswahrscheinlichkeiten ist für praxisnahe Problemstellungen sehr schwer zu realisieren. Im Speziellen würde die Kenntnis des vollständigen Modells eines regelungstechnischen Problems die Anwendung von klassischen Verfahren der Regelungstechnik zur Lösung des Problems zulassen. Ohne die Kenntnis der Übergangswahrscheinlichkeiten muss die Pfadkostenfunktion durch eine auf Zustand und Aktion basierende Wertfunktion ersetzt werden. Diese Idee wurde von Watkins [111, 112] als Q-Lernen eingeführt. Dabei ersetzt eine Funktion $Q^\pi(x, u)$ die Pfadkostenfunktion. Diese repräsentiert die Pfadkosten, die auftreten, wenn in einem Zustand x die Aktion u angewendet wird und danach der aktuellen Strategie π weiter gefolgt wird.

$$Q^\pi(x, u) = \sum_{x' \in X} P_{\pi(x)}(x, x')(c(x, u) + J^\pi(x')) \quad (2.9)$$

Die zu $Q^\pi(x, u)$ gehörende Gierige Strategie ergibt sich dann zu $\pi(x) = \operatorname{argmin}_{u \in U} Q^\pi(x, u)$. Aus einer vorhandenen Q-Funktion kann die zugehörige Gierige Strategie somit ohne Kenntnis der Übergangswahrscheinlichkeiten berechnet werden.

Bei kontinuierlichen Zustandsräumen wird das Q-Lernen auf Trajektorien am MDP ausgeführt. Durch die Unkenntnis des Modells sind die beobachteten Übergänge die einzige Information, die dem Q-Lernen für die Aktualisierung der Q-Funktion zur Verfügung steht. Somit muss bei stochastischen MDPs der Erwartungswert durch stochastische Approximation gewonnen werden. In Formel 2.10 ist die Aktualisierungsvorschrift für das Q-Lernen mit stochastischer

Approximation mit Robbins-Monro dargestellt.

$$Q_{k+1}(x, u) = (1 - \alpha) * Q_k(x, u) + \alpha(c(x, u) + \gamma \min_b Q_k(x', b)) \quad (2.10)$$

Der Parameter α beeinflusst die stochastische Approximation in der Wertfunktion. Das Q-Lernen ermöglicht das Lernen mit einem Minimum an 'a priori Wissen' über das zu lösende Problem. Durch die Interaktion auf Trajektorien mit dem MDP kann die Q-Funktion iterativ mit Formel 2.10 jeweils an der Stelle aktualisiert werden über die der Übergang Informationen enthält. Die Information, auf der die Aktualisierung beruht, ist in der Form (x, u, x', c) und repräsentiert den vom Agenten auf der Trajektorie beobachteten Folgezustand und die Kosten bei Anwendung einer Aktion in einem Zustand.

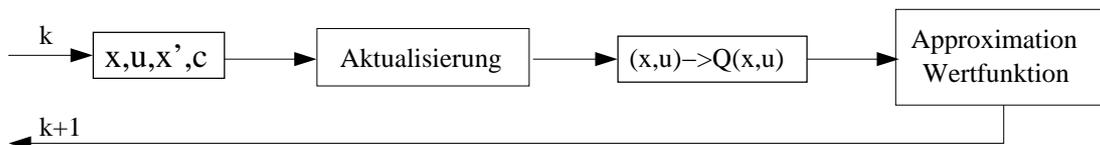


Abbildung 2.2: Die Schematische Darstellung der Aktualisierung der Wertfunktion beim Q-Lernen auf kontinuierlichen Zustandsräumen beim Lernen entlang von Trajektorien.

2.1.4 Batchverfahren im Reinforcement-Lernen

Die Familie der 'Batch'- und 'Fitted'-Verfahren ist eine aktuelle Entwicklung im Reinforcement-Lernen [5, 30, 48, 106]. Ihre Grundidee ist die Steigerung der Dateneffizienz der Reinforcement Lernverfahren speziell im modellfreien Lernen basierend auf Wertfunktionen, in kontinuierlichen Zustandsräumen.

Beim Lernen entlang von Trajektorien werden nacheinander direkt die beobachteten Übergänge (x, u, x', c) zur Aktualisierung der Wertfunktion verwendet. Durch die iterative Form der Aktualisierungsvorschriften der Wertfunktionen muss dieser Übergang sehr oft in die Aktualisierung eingehen, um sie an dieser Stelle möglichst gut anzupassen. Der Agent muss somit diesen Übergang für jede Aktualisierung der Wertfunktion durch das Ablaufen einer Trajektorie besuchen. Da die betrachteten MDPs, über die durch Interaktion in Trajektorien die Information gesammelt wird, stochastische Eigenschaften besitzen, muss der Agent eine einzelne Trajektorie sehr oft ausführen um die stochastische Approximation zu gewährleisten. Dies führt dazu, dass klassische Verfahren, basierend auf Wertfunktionen, eine sehr hohe Anzahl von Interaktionen und Aktualisierungsschritten benötigen, die sie für die praktische Anwendung nur bedingt einsatzfähig machen. Ein zusätzliches Problem stellt die Integration eines einzelnen aktualisierten Wertes in eine bestehende Approximation der Wertfunktion dar. Die neue Information, mit der die Approximation aktualisiert werden soll, liegt nur in einem Punkt des Eingaberaumes vor. Sie soll sich möglichst nur lokal um diesen Punkt auf die Approximation auswirken, um die Approximation in anderen Bereichen nicht zu verschlechtern. Dies ist ein Grund, weshalb Funktionsapproximatoren mit globalen Generalisierungseigenschaften in der praktischen Anwendung im Reinforcement-Lernen schlechte Ergebnisse erzeugen können.

Die Batchverfahren erweitern den Agenten um eine Speicherkomponente, in der die gesamten während des Verlaufs des Lernvorganges beobachteten Übergänge gespeichert werden. Diese Speicherkomponente wird als Menge \mathcal{D} von Übergängen der Form (x, u, x', c) repräsentiert. In jeder Iteration des Lernverfahrens wird die Aktualisierung der Wertfunktion nun parallel für

alle Übergänge in der Datenbasis berechnet. Dieses Vorgehen ist im weiteren Sinne ähnlich der parallelen Aktualisierungsvorschrift des Wertiterationsverfahren aus 2.1.3. Im Unterschied dazu sind jedoch nicht alle möglichen Übergänge des MDPs bekannt, sondern nur diejenigen, die auf Trajektorien durch Interaktion mit dem MDP beobachtet wurden. Durch die parallele

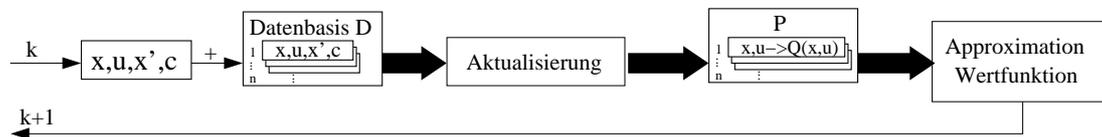


Abbildung 2.3: Die Verwendung der Batchverfahren führt ein Speicherelement im Agenten ein in dem alle beobachteten Übergänge des MDPs gespeichert werden. Die Aktualisierung der Wertfunktion wird in jeder Iteration des Lernverfahrens auf der gesamten Menge der gespeicherten Übergänge durchgeführt.

Die Anwendung der Aktualisierungsvorschrift auf die Datenbasis \mathcal{D} im Batchlernen wird eine Menge \mathcal{P} von Trainingsmustern für die Aktualisierung der Approximation der Wertfunktion berechnet. Diese Menge von Trainingsmustern repräsentiert die zu lernende Approximation der aktualisierten Wertfunktion, die mit Methoden des überwachten Lernens in den verwendeten Approximator eingelernt wird. Aus diesem Grund werden die Batchverfahren auch als 'Fitted'-Verfahren bezeichnet, was ihre Vorteile für die Approximation der Wertfunktion betont. Diese Approximation kann nun auf der gesamten Menge von Trainingsmustern erfolgen, was die Anwendung von leistungsstarken Funktionsapproximatoren und Lernverfahren zulässt. Insbesondere können Funktionsapproximatoren mit globalen Generalisierungseigenschaften verwendet werden, da der Approximationsfehler auf der ganzen Datenbasis reduziert wird.

2.2 Künstliche neuronale Netze

Bei der Anwendung der Reinforcement Lernverfahren auf regelungstechnische Fragestellungen ist die Verwendung eines Funktionsapproximators zur Repräsentation kontinuierlicher Zustandsräume eine wichtige Voraussetzung. Im Rahmen dieser Arbeit werden mehrschichtige neuronale Netze (engl. 'Multi Layer Perceptron (MLP)', 'Feed Forward Neural Network') verwendet. Im folgenden werden die Eigenschaften der neuronalen Netze, basierend auf ihrer Relevanz für die in dieser Arbeit verwendeten Methoden, dargestellt. Für eine ausführliche Diskussion neuronaler Netze sowie anderer Netz-Modelle und eine vertiefende Betrachtung der Lernverfahren für neuronale Netze sei auf die Vielzahl einführender Literatur zu diesem Thema (z.B. [86, 118]) verwiesen.

2.2.1 Mehrschichtige neuronale Netze

Das 'Multi Layer Perceptron' (MLP) ist ein vorwärts gerichtetes neuronales Netz (Abbildung 2.4). Dieser konnektionistische Ansatz berechnet eine Antwort bzw. Ausgabe auf eine anliegende Aktivierung bzw. Eingabe unter Benutzung vieler durch Gewichte verbundener, einfacher Berechnungselemente. Die Struktur (oder Topologie) eines solchen Netzes kann in verschiedene Schichten unterteilt werden. Die Eingabeschicht enthält den Eingabevektor, der an das Netz anliegt. Die Aktivierung eines Neurons dieser Schicht ist direkt durch den entsprechenden Eintrag des externen Eingabevektors bestimmt. Über gewichtete Verbindungen wird diese Aktivierung an die Neuronen der nächsten verdeckten Schicht weitergegeben. Diese berechnen

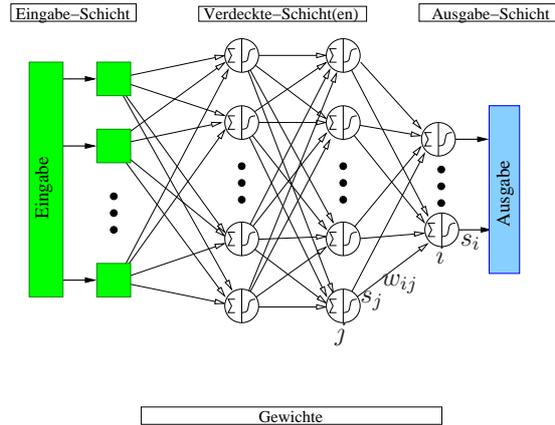


Abbildung 2.4: Schematische Darstellung des Aufbaus eines mehrschichtigen neuronalen Netzes. Auf die Darstellung des Schwellwertneurons bzw. des Schwellwertgewichts wurde in der Darstellung verzichtet.

Tabelle 2.1: Gebräuchliche Aktivierungsfunktionen in neuronalen Netzen und ihre Ableitungen.

linear	$s_i = f_{lin}(net_i) = net_i$	$\mathbb{R} \Rightarrow \mathbb{R}$	$f'_{lin}(net_i) = 1$
logistisch	$s_i = f_{log}(net_i) = \frac{1}{1+e^{-net_i}}$	$\mathbb{R} \Rightarrow (0, 1)$	$f'_{log}(net_i) = s_i(1 - s_i)$
symmetrisch	$s_i = f_{sym}(net_i) = \frac{2}{1+e^{-2net_i}} - 1$	$\mathbb{R} \Rightarrow (-1, 1)$	$f'_{sym}(net_i) = 1 - s_i^2$

hieraus ihre eigene Aktivierung und geben diese wiederum an die nächste Schicht weiter. Dieses Fortpflanzen der Aktivierung bzw. Vorwärtspropagieren der Aktivierung durch das Netz findet statt, bis die Aktivierung der Ausgabeschicht berechnet wird. Diese ist die Antwort des Netzes auf den anliegenden Eingabevektor. Die Aktivierung s_i eines Neurons i außerhalb der Eingabeschicht wird ausgehend von seiner Eingabe net_i berechnet. net_i ist die Summe der gewichteten Aktivierungen aller eingehenden Verbindungen zu Neuronen aus vorhergehenden Schichten (siehe Formel 2.11).

$$net_i = \left(\sum_{j \in pred(i)} s_j w_{ij} \right) - \Theta_i \quad (2.11)$$

Hierbei bezeichnet $pred(i)$ die Neuronen aus vorhergehenden Schichten, die mit Neuron i verbunden sind, w_{ij} das Verbindungsgewicht zwischen Neuron j und Neuron i und Θ_i den Schwellwert oder Bias des Neurons i . Aus Gründen der homogenen Darstellung wird der Bias eines Neurons oft ebenfalls als eingehende Verbindung modelliert, die mit $-\Theta_i$ gewichtet ist und eine konstante Aktivierung von 1 besitzt. Die Aktivierung s_i des Neurons i ergibt sich aus der Berechnung einer Aktivierungsfunktion seiner Eingabe: $s_i = f_{act}^i(net_i)$, $f_{act}^i : \mathbb{R} \Rightarrow \mathbb{R}$. Die gebräuchlichsten Aktivierungsfunktionen und ihre Ableitungen $\frac{\partial s_i}{\partial net_i} = f'_{act}(net_i)$ sind in Tabelle 2.1 aufgeführt. Die Funktion, die ein mehrschichtiges neuronales Netz repräsentiert, hängt folglich von seiner Topologie und den Werten der Gewichten ab. Zugunsten der besseren Darstellung wird im folgenden die von einem mehrschichtigen neuronalen Netz \mathcal{N} mit Eingabevektor x und Gewichtsvektor w realisierte Funktion als $\mathcal{N}(x, w)$ dargestellt.

2.2.2 Lernen im mehrschichtigen neuronalen Netz

Um das neuronale Netz an neue Information anzupassen, müssen die Gewichte im Netz verändert werden. Diese Veränderung wird typischerweise mit überwachtem Lernen durchgeführt. Beim überwachten Lernen ist die zu lernende Abbildung von Netzeingabe auf Netzausgabe durch Beispiele aus dieser Abbildung in einer Menge aus Trainingsmustern \mathcal{P} gegeben. Jedes Trainingsmuster $p \in \mathcal{P}$ besteht aus einem Vektor x von Netzeingaben und der bei diesen Eingaben erwarteten Ausgabewerten des Netzes in einem Vektor z . Der überwachte Lernvorgang hat das Ziel, die Gewichte des Netzes so anzupassen, dass bei Eingabe x^p an das Netz die Antwort des Netzes s^p den entsprechenden Zielwerten z^p angenähert wird. Dies lässt sich als eine zu minimierende Funktion über \mathcal{P} definieren:

$$E(w) := \frac{1}{2} \sum_{p \in \mathcal{P}} (z^p - s^p)^2. \quad (2.12)$$

Das Verfahren, das typischerweise zur Minimierung des Fehlers eingesetzt wird, ist der Gradientenabstieg. Hierbei wird die partielle Ableitung des Fehlers nach den Parametern bzw. den Gewichten gebildet und diese in die entgegengesetzte Richtung des Gradienten verschoben. Für die Fehlerfunktion des Überwachten Lernens im neuronalen Netz ergibt sich dieser Anpassungsschritt im einfachsten Fall zu: $\Delta w = -\alpha \nabla E(w)$. Hierbei bezeichnet $\nabla E(w)$ den Vektor der partiellen Ableitungen der Netzgewichte w nach dem Fehler $E(w)$. Über die Lernrate α wird die Schrittweite der Anpassung der Gewichte Δw beeinflusst. Der neue Gewichtsvektor des Netzes ergibt sich dann zu $w = w + \Delta w$.

Im mehrschichtigen neuronalen Netz kann die partielle Ableitung des Fehlers ∂E nach einem bestimmten Netzgewicht w_{ij} durch Zurückpropagieren des Fehlers durch das Netz von der Ausgabeschicht zur Eingabeschicht sukzessive berechnet werden. Dieses Vorgehen wird auch als 'Backpropagation' bezeichnet. Die Berechnung des Fehlergradienten lässt sich direkt durch mehrfache Anwendung der Kettenregel herleiten. Im ersten Schritt ergibt sich $\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial s_i} \frac{\partial s_i}{\partial w_{ij}}$, wobei sich die partielle Ableitung der Ausgabe eines Neurons nach den Gewichten seiner Eingabe zu $\frac{\partial s_i}{\partial w_{ij}} = f'_{act}(net_i) s_j$ ergibt. Den Einfluss der Aktivierung des Neurons i auf die Fehlerfunktion $\frac{\partial E}{\partial s_i} = -(z^i - s_i)$ kann für die Ausgabeneuronen direkt aus der Differenz des Zielwertes und der Ausgabe des entsprechenden Ausgabeneurons berechnet werden. Für Neuronen in der verdeckten Schicht ergibt sich die partielle Ableitung ihrer Ausgabe nach dem Fehler $\frac{\partial E}{\partial s_i} = \sum_{k \in succ(i)} \frac{\partial E}{\partial s_k} f'_{act}(net_k) w_{ki}$ durch Zurückpropagieren des Fehlersignales durch das Netz. Die Funktion $succ(i)$ gibt die Menge der Neuronen an, die in den Folgeschichten (zur Ausgabe hin) mit Neuron i verbunden sind.

Für die Anwendung als Lernverfahren ist der Zeitpunkt der Gewichtsanzpassung entscheidend. Beim musterbasierten Lernen wird nach der Präsentation eines einzelnen Trainingsmusters, beim blockbasierten Lernen nach mehreren und beim epochenbasierten Lernen nach der Präsentation aller Trainingsmuster die Aktualisierung durchgeführt.

Das reine Backpropagation ist sehr empfindlich gegen die Einstellung der Lernrate α und benötigt sehr viele Anpassungsschritte, um eine Funktion mit ausreichender Genauigkeit zu approximieren.

2.2.3 Resilient Propagation (RProp)

Eine sehr leistungsfähige Erweiterung des Backpropagation stellt das 'Resilient Propagation (RProp)' Verfahren dar [80–82]. Im Unterschied zum Standardverfahren ist bei RProp die Län-

ge der durchgeführten Gewichtsanzpassung nicht konstant. Sie wird vielmehr iterativ über die Lernschritte des Verfahrens, basierend auf dem Gradienten des Fehlers, für jedes Gewicht angepasst. Für den Vektor w mit Einträgen w_i für jedes Gewicht im neuronalen Netz und iterativer Aktualisierung nach dem Gradientenabstiegsschema $w^{(k+1)} = w^k + \Delta w^k$ ergibt sich die Anpassung zu:

$$\Delta w_i^k = \begin{cases} -\Delta_i^k & \frac{\partial E^k}{\partial w_i} > 0 \\ +\Delta_i^k & \frac{\partial E^k}{\partial w_i} < 0 \\ 0 & \text{sonst} \end{cases} \quad (2.13)$$

$$\Delta_i^k = \begin{cases} \eta^+ \Delta_i^{(k-1)} & \frac{\partial E^{k-1}}{\partial w_i} \frac{\partial E^k}{\partial w_i} > 0 \\ \eta^- \Delta_i^{(k-1)} & \frac{\partial E^{k-1}}{\partial w_i} \frac{\partial E^k}{\partial w_i} < 0 \\ \Delta_i^{(k-1)} & \text{sonst} \end{cases} \quad (2.14)$$

Das RProp-Verfahren kann seine Stärke bei der Anpassung der Lernraten für jedes Gewicht vor allem im epochenbasierten Lernen voll entwickeln. Bei rein musterbasiertem oder blockbasiertem Lernen geht dieser Vorteil teilweise verloren. Das RProp-Verfahren ist nicht ausschließlich für die Minimierung der Fehlerfunktion eines neuronalen Netzes bezüglich seiner Gewichte anwendbar. Es ist vielmehr ein Verfahren zur Minimierung allgemeiner parametrisierter Funktionen $E(w)$ mit Parameter-Vektor w mittels Gradientenabstieg. Die Stärken des Verfahrens liegen vor allem in der Reduktion der benötigten Lernschritte für das Erlernen einer gegebenen Funktion und seiner Robustheit gegen die verwendeten Parameter. Im Unterschied zum Backpropagation-Verfahren müssen diese nicht für jeden Lernversuch angepasst werden. Die von Riedmiller publizierten Werte von $\eta^- = 0.5$ und $\eta^+ = 1.2$ in Formel 2.14 werden für alle in dieser Arbeit durchgeführten Lernversuche verwendet.

2.2.4 Weitere Eigenschaften neuronaler Netze

Eine im Verlauf dieser Arbeit ausgenutzte Eigenschaft der neuronalen Netze ist, dass sie durch ihren Aufbau und die verwendeten Aktivierungsfunktionen differenzierbare Funktionen repräsentieren. Durch Vorwärts- und Rückwärtspropagieren durch das Netz können die partiellen Ableitungen der durch das Netz repräsentierten Funktion (bei mehrdimensionalen Ausgaben der Funktionen) nach der Eingabe gebildet werden. Für ein neuronales Netz $\mathcal{O}(x, w) = y$ mit Eingabevektor $x \in \mathbb{R}^n$, Gewichtsvektor w und Ausgabe $y \in \mathbb{R}^m$ ergibt sich nach Vorwärtspropagieren eines Wertes x und Zurückpropagieren eines Wertes von 1 für ein einzelnes Ausgabeneuron h an jedem Eingabeneuron l somit die partielle Ableitung: $\frac{\partial \mathcal{O}_h(x, w)}{\partial x_l}$. Eine direkte Anwendung der auf diese Weise berechneten Gradienten ergibt sich bei der Minimumssuche in einer durch ein neuronales Netz mit festen Gewichten repräsentierten Funktion. So kann $\text{argmin}_x \mathcal{O}(x, w)$ durch Berechnung des Gradienten im Netz und Anwendung eines Gradientenabstiegsverfahrens für eine Funktion berechnet werden, die nur in Beispielen bzw. Trainingsmustern vorliegt. Eine weitere Eigenschaft der neuronalen Netze, die in dieser Arbeit verwendet wird, ist die Möglichkeit, mehrere Netze zu verketteten. Ein gekoppeltes neuronales Netz ist in Abbildung 2.5 schematisch dargestellt. Es besteht aus zwei mehrschichtigen neuronalen Netzen, von denen das erste die Abbildung $y = \mathcal{G}(x, w_g)$ mit Gewichtsvektor w_g und das zweite die Abbildung $z = \mathcal{F}(y, w_f)$ mit Gewichtsvektor w_f realisiert. Werden die Ausgabeneuronen von $\mathcal{G}(x, w_g)$ direkt mit den Eingabeneuronen von $\mathcal{F}(y, w_f)$ verbunden, so realisieren sie beide zusammen die Funktion $\mathcal{F}(\mathcal{G}(x, w_g), w_f)$. Es soll der spezielle Fall

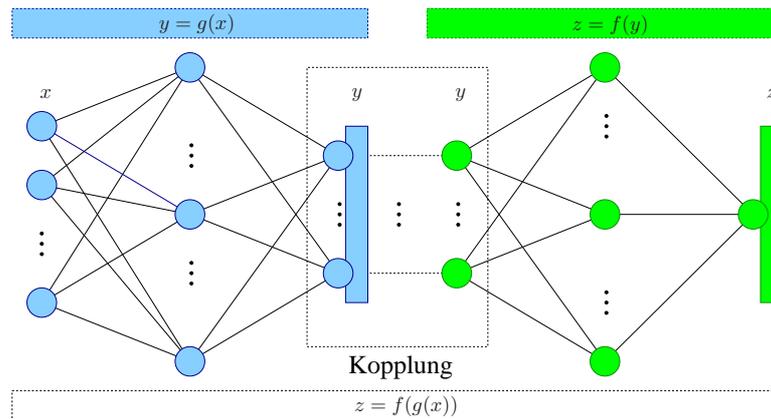


Abbildung 2.5: Eine Verketzung von zwei neuronalen Netzen $y = \mathcal{G}(x, w_g)$ und $z = \mathcal{F}(y, w_f)$. Durch Verbindung der Ausgabeneuronen des einen Netzes mit den Eingabeneuronen des zweiten Netzes realisieren sie die Funktion $\mathcal{F}(\mathcal{G}(x, w_g), w_f)$.

betrachtet werden, bei dem das Netz $\mathcal{F}(y, w_f)$ eine durch Trainingsmuster gegebene Abbildung mit $z \in \mathbb{R}$ repräsentiert. Wie bereits diskutiert, kann für das in seinen Gewichten feste Netz \mathcal{F} die partielle Ableitung $\frac{\partial \mathcal{F}(y, w_f)}{\partial y}$ direkt durch Vorwärts- und Rückwärtspropagieren im neuronalen Netz berechnet werden. Das Vorwärtspropagieren geht in diesem Falle über beide Netze, das Rückwärtspropagieren bis zur Eingabeschicht des Netzes \mathcal{F} . In der Eingabeschicht des Netzes \mathcal{F} liegt dann der Vektor $\frac{\partial \mathcal{F}(\mathcal{G}(x, w_g), w_f)}{\partial y}$ an. Die Berechnung von $\frac{\partial \mathcal{F}(\mathcal{G}(x, w_g), w_f)}{\partial w_g}$, die partielle Ableitung der gesamten Netzausgabe nach den Gewichten im Netz \mathcal{G} , kann nun ebenfalls direkt nach dem Backpropagation-Schema durchgeführt werden. Durch Vorwärtspropagieren eines anliegenden Musters über das gesamte Netz und Zurückpropagieren einer Eins, ausgehend vom Ausgabeneuron von \mathcal{F} , kann somit im Netz \mathcal{G} der geeignete Gradient für jedes Gewicht gebildet werden, um die Veränderung der Netzgewichte in w_g mit Gradientenabstieg vorzunehmen. Diese Veränderung der Gewichte des Netzes \mathcal{G} kann als zusätzlicher Lernschritt eingeführt werden um eine Fehlerfunktion des gesamten Netzes weiter zu verkleinern. Der interessantere Fall wird in 4.3 betrachtet, bei dem die gesamte Funktion die Form $\mathcal{F}(x, y = \mathcal{G}(x, w_g), w_f)$ besitzt. In dieser Kombination ist es möglich für feste Gewichte w_f , die Gewichte w_g , so einzulernen, dass \mathcal{G} die gesamte Funktion für jede Eingabe x , minimiert.

2.3 Neural Fitted Q Iteration

Das 'Neural Fitted Q-Iteration'-Verfahren (NFQ) ist ein aktuelles, dateneffizientes Batch-Reinforcement-Lernverfahren zum modellfreien Q-Lernen mit kontinuierlichen Zustandsräumen und diskreten Aktionen [79]. Als Funktionsapproximator für die Q-Funktion werden neuronale Netze verwendet. Durch die Eigenschaft der Batchverfahren (siehe 2.1.4) kann die Funktionsapproximation der Q-Funktion mit epochenbasiertem Lernen unter Verwendung eines leistungsfähigen Lernverfahrens wie RProp durchgeführt werden. Auf diese Weise können die Vorteile der neuronalen Netze in Bezug auf ihre Generalisierungseigenschaften und ihre Fähigkeit, allgemeine nichtlineare Funktionen zu repräsentieren, für das Reinforcement-Lernen eingesetzt werden. Der Nachteil der globalen Änderung der repräsentierten Funktion bei lokaler Änderung eines einzelnen Musters wird durch das Batchverfahren aufgehoben.

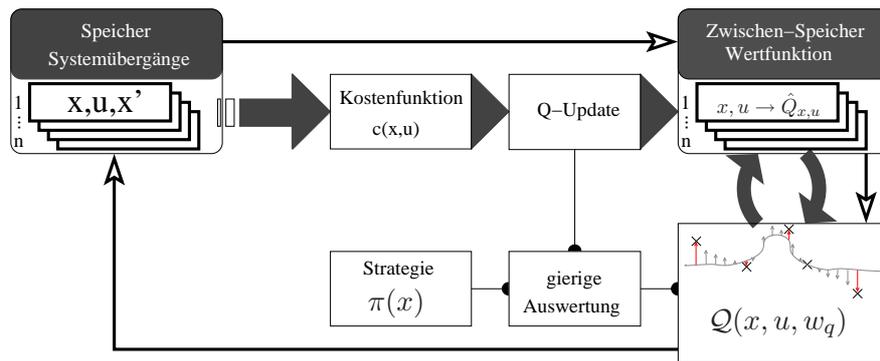


Abbildung 2.6: Die schematische Darstellung des NFQ-Verfahrens als Batchverfahren des Reinforcement-Lernens mit neuronaler Approximation der Q-Funktion. Die Aktualisierung der Q-Funktion wird parallel auf der gesamten Datenbasis durchgeführt. Die neuronale Q-Funktion wird mit der resultierenden Trainingsmenge und Epochen basierendem Lernen mittels RProp eingelernt.

2.3.1 Die Approximation der Q-Funktion mit neuronalen Netzen

Die Approximation der Q-Funktion kann für eine kleine Anzahl von diskreten Aktionen in der Aktionsmenge $U = \{u^0, \dots, u^m\}$ durch n unabhängige neuronale Netze $Q_i(x, w_q^i)$ dargestellt werden. Diese Repräsentation eignet sich für abstrakte Aktionen, die in ihrer Auswirkung auf den MDP in der Q-Funktion keine Generalisierung über den Aktionen zulassen. Für die in dieser Arbeit betrachteten Problemstellungen wird davon ausgegangen, dass ähnliche Aktionen ähnliche Q-Werte besitzen. In diesem Fall ist die Generalisierung über den Aktionen in der Q-Funktion erwünscht. Auf diese Weise enthält die Q-Funktion in einem Zustand auch für Aktionen sinnvolle Werte, die selbst vom Agenten noch nicht besucht worden sind bzw. nicht in der Datenbasis vorhanden sind. Dies gilt zumindest dann, wenn ein Übergang mit ähnlichem Zustand und ähnlicher Aktion in der Datenbasis vorhanden ist. Um diese Generalisierung mit einem neuronalen Netz zu realisieren, werden die diskreten Aktionen als Punkte aus einem mehrdimensionalen Aktionsraum $u^i \in U \subset \mathbb{R}^m$ repräsentiert. Eine Aktion ist somit ein Vektor, dessen Einträge nur diskrete Werte annehmen können. Die Repräsentation der Q-Funktion kann somit als neuronales Netz $Q(x, u, w_q)$ dargestellt werden. Der Eingabevektor, der am neuronalen Netz anliegt, ist zusammengesetzt aus dem Zustandsvektor und dem Aktionsvektor.

Leistungsfähigkeit von RProp am Beispiel einer Wertfunktion

Die Verwendung neuronaler Netze als Funktionsapproximatoren für Q-Funktionen im Reinforcement-Lernen hat eine lange Tradition. Ohne die Idee der Batchverfahren wird die Aktualisierung der Q-Funktion auf einzelnen Übergängen vorgenommen. Als Folge daraus muss das Lernen der neuronalen Approximation der Q-Funktion mit musterbasiertem Lernen durchgeführt werden. Das Backpropagation mit Standard-Gradientenabsitzig erweist sich jedoch als sehr sensibel gegen die Variation seiner Parameter. Dies wird als ein zentraler Nachteil bei der Verwendung von neuronalen Netzen im Reinforcement-Lernen angesehen. Die Verwendung von neuronalen Netzen in Kombination mit RProp als Beispiel eines leistungsfähigen Lernverfahrens zur Approximation der Q-Funktion ist ein zentraler Bestandteil des NFQ-Verfahrens. In der Praxis erweist sich diese Kombination als sehr robust sowohl gegen die Variation der Topologie des Netzes, als auch gegen die Parameter des RProp-Verfahrens. Zudem ist der Ap-

proximationsfehler der Q-Funktion schon nach wenigen hundert Epochen auf einem sehr guten Wert, der für die meisten praktischen Probleme ausreichend ist.

Dieser Unterschied der Lernverfahren soll an einem Beispiel verdeutlicht werden. In Abbildung 2.7 ist eine für das Reinforcement-Lernen exemplarische Wertfunktion abgebildet. Diese repräsentiert eine analytisch bestimmte Kostenfunktion eines Standardbenchmarks des Reinforcement-Lernens.² Die betrachtete Funktion enthält die typischen Merkmale, welche Q-

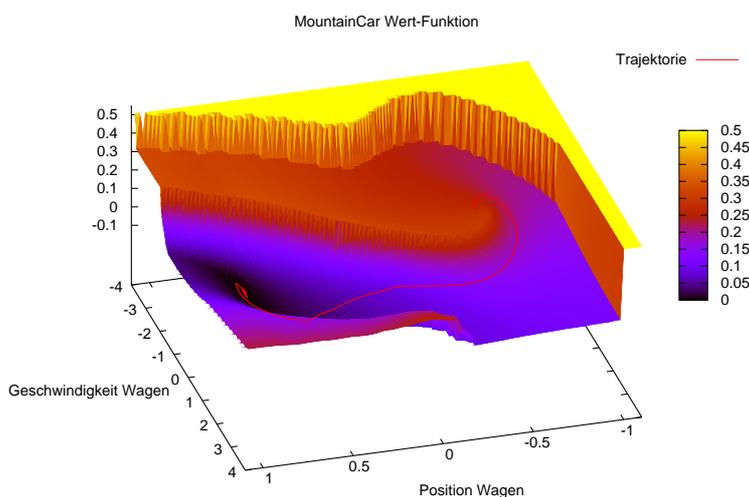


Abbildung 2.7: Die betrachtete exemplarische Wertfunktion für den Vergleich der Lernverfahren weist typische Merkmale von Q-Funktionen im Reinforcement-Lernen auf, die die Approximation erschweren.

Funktionen im Reinforcement-Lernen aufweisen und die Approximation dieser Funktionen erschweren. Die für die Approximation problematischen Eigenschaften sind vor allem steile Flanken, wie sie in Randbereichen oft auftreten, sowie beliebig gekrümmte Flächen. Es werden aus der analytisch berechneten Funktion 1000 zufällig gezogene Trainingsmuster und 2000 zufällig gezogene Testmuster erstellt. Die Lernversuche werden jeweils mit RProp und Backpropagation mit epochenbasiertem Lernen durchgeführt. Nach jeweils 50 Epochen wird der Testfehler (RMS) auf der Testmenge bestimmt. Für Backpropagation wird zusätzlich das musterbasierte Lernen durchgeführt, dessen Testfehler ebenfalls nach 50-facher Präsentation der gesamten Trainingsmenge berechnet wird. Das RProp-Verfahren wird ohne weitere Anpassungen mit seinen Standardparametern aus der Literatur verwendet. Für das Backpropagation-Verfahren wird die Lernrate für eine Topologie von 2 verdeckten Schichten mit jeweils 40 Neuronen durch ausgiebige Suche festgelegt. Jeder Lernversuch wird jeweils 3000 Epochen lang auf 20 unabhängigen Versuchen mit unterschiedlicher Initialisierung des neuronalen Netzes durchgeführt. In Abbildung 2.8 ist der Verlauf der Testfehler der drei Verfahren über die Trainingsepochen auf einer Topologie mit 2 verdeckten Schichten mit jeweils 40 Neuronen aufgetragen. An diesem Beispiel zeigt sich eine typische Eigenschaft des RProp-Verfahrens gegenüber normalem Backpropagation. Der Fehler in der Approximation sinkt sehr schnell auf einen sehr kleinen Wert und die durch die unterschiedliche Initialisierung des Netzes verur-

²Die exemplarische Wertfunktion stammt aus dem 'Mountain Car' Versuch, bei dem ein untermotorisierter Wagen aus einem Tal heraus auf die Flanke eines Berges gesteuert werden muss, ohne einen bestimmten Bereich zu verlassen.

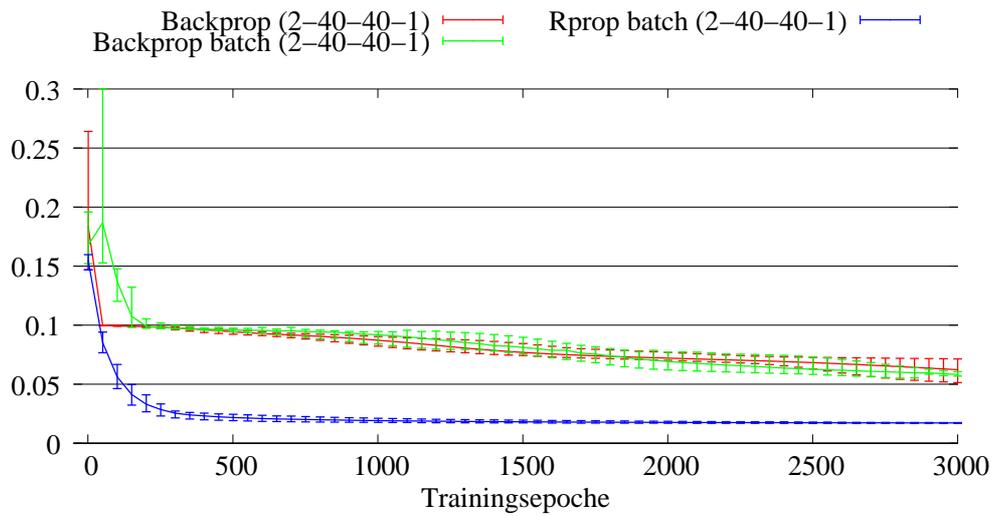


Abbildung 2.8: Verlauf des Testfehler beim überwachten Lernen der exemplarischen Wertfunktion am Beispiel einer 2-40-40-1 Topologie. Dargestellt ist der mittlere Testfehler über 20 unabhängige Lernversuche mit den minimalen und maximalen Fehlern des RProp-Verfahrens mit Epochen basiertem Lernen, des Backprop Verfahrens mit Epochen basiertem sowie Muster basierten Lernen.

sachte Varianz nimmt schnell ab. Das Backpropagation-Verfahren kann durch die Verwendung des epochenbasierten Lernens keine signifikante Verbesserung erzielen. Der Approximationsfehler nimmt nur langsam ab. Der Lernversuch mit den selben Parametern der Verfahren wird

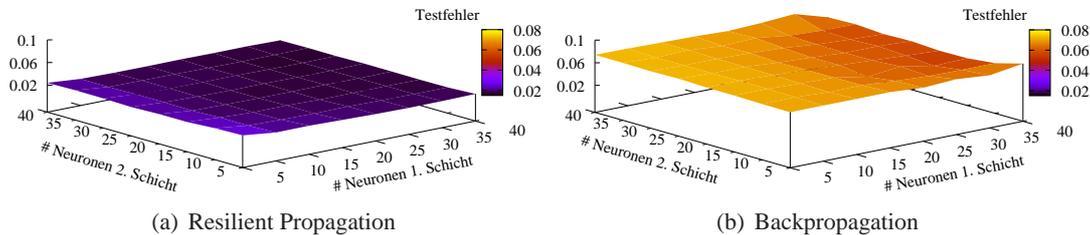


Abbildung 2.9: Der über jeweils 20 unabhängige Versuche gemittelte durchschnittliche Testfehler nach 3000 Lernschritten bei überwachten einlernen der exemplarischen Wertfunktion über unterschiedliche Anzahl von Neuronen in den zwei verdeckten Schichten eines neuronalen Netzes.

nun auf verschiedenen Topologien durchgeführt. Dabei werden zweischichtige Topologien mit einer variierenden Anzahl von Neuronen in jeder verdeckten Schicht verwendet. Für jede Topologie werden wiederum 20 Versuche mit den verschiedenen Verfahren durchgeführt. In Abbildung 2.9 sind die Ergebnisse als Mittelwert des Testfehlers (RMS) für die verschiedenen Topologien des epochenbasierten Lernens mit RProp und Backpropagation nach 3000 Epochen dargestellt. Das RProp-Verfahren ist robust gegen die Variation der Netztopologie und erzielt mit einer ausreichenden Anzahl von Neuronen den annähernd gleichen Testfehler. Das Backpropagation-Verfahren erzielt mit 40 Neuronen in der ersten verborgenen Schicht und 20 Neuronen in der zweiten verborgenen Schicht die besten Ergebnisse. Die Ergebnisse des musterbasierten Lernvorgangs unterscheiden sich nicht wesentlich von denen des epochenbasierten Lernens mit Backpropagation.

Der vorgestellte Vergleich ist keine empirische Untersuchung im engeren Sinne. Er verdeutlicht

vielmehr Erfahrungswerte, die im praktischen Umgang mit NFQ gesammelt wurden. Einer dieser Erfahrungswerte ist, dass wenige hundert Epochen Training mit RProp eine ausreichende Approximation der Wertfunktion ermöglichen, was für die benötigte Rechenzeit des Lernverfahrens ein wichtiger Faktor ist. Zum anderen ist das Verfahren in weiten Bereichen robust gegen die Wahl der Topologie des Netzes und benötigt darüber hinaus keine Veränderung der Parameter des Lernverfahrens.

2.3.2 Aktualisierung der Q-Funktion

Der Kern des NFQ-Verfahrens ist eine Iterationsvorschrift zur Berechnung einer aktualisierten neuronalen Approximation der Q-Funktion. Diese wird basierend auf einer Datenbasis von beobachteten Übergängen eines MDPs und der aktuellen neuronalen Approximation der Q-Funktion durchgeführt. Dabei liegt die über den vorliegenden MDP gesammelte Erfahrung in Form einer Menge \mathcal{D} von beobachteten Übergängen der Form $d = (x, u, x', c)$ des MDPs vor. Ausgehend von einer Approximation der Q-Funktion Q wird in einem Batch-Aktualisierungsschritt für jeden Übergang in \mathcal{D} unter Benutzung von Formel 2.10 ein neuer Q-Wert \hat{Q} berechnet. Diese aktualisierten Q-Werte bilden mit den entsprechenden Zuständen und Aktionen, aus denen sie berechnet wurden die Menge \mathcal{P} von Trainingsmustern für das überwachte Lernen der aktualisierten Approximation der Q-Funktion. Die Trainingsmenge \mathcal{P} , als Datenbasis der Approximation der Q-Funktion, hat Einträge der Form $p = (p^{input}, p^{target}) = ((x, u), \hat{Q}_{x,u})$. Dabei ist $\hat{Q}_{x,u}$ der neue Zielwert für die Approximation der Q-Funktion für den entsprechenden Übergang. Dieser Batchschritt des Verfahrens zur Erzeugung der Trainingsmenge \mathcal{P} ist in Algorithmus 3 dargestellt. Durch die Verwendung diskreter Aktionen wird die

Algorithmus 3 BatchQUpdate(Q, \mathcal{D})

Require: Q : Approximation der Q-Funktion, \mathcal{D} : Menge der beobachteten Übergänge

Ensure: \mathcal{P} : Trainingsdatenmenge mit aktualisierten Q-Werten

$\mathcal{P} \leftarrow \emptyset$

for $i = 1, \dots, \#\mathcal{D}$ **do**

mit $d_i = (x_i, u_i, x'_i, c_i)$ i -tes Element $\in \mathcal{D}$

$\hat{Q} = (1 - \alpha) * Q(x_i, u_i) + \alpha * (c_i + \gamma MIN(Q, x'_i))$

$\mathcal{P} \leftarrow \mathcal{P} \cup ((x_i, u_i), \hat{Q})$

end for

return \mathcal{P}

benötigte Minimumsbildung über die in einem Zustand zur Verfügung stehenden Aktionen in Formel 2.10 durch eine vollständige Suche durchgeführt. Bei n zur Verfügung stehenden Aktionen muss somit die neuronale Q-Funktion für jede der n Aktionen in einem Zustand x berechnet werden, um daraus das Minimum zu ermitteln (siehe Algorithmus 4). Die aus dem Batch-Aktualisierungsschritt erzeugte Trainingsdatenmenge wird für das überwachte Lernen der neuen neuronalen Q-Funktion benutzt. Dieser Lernvorgang des neuronalen Netzes wird mit epochenbasiertem Lernen mit RProp durchgeführt. Dabei wird das überwachte Lernen in jedem dieser Iterationsschritte mit einem zufällig initialisiertem Netz und mit reinitialisierten Schrittweiten des RProp-Verfahrens durchgeführt. Dies verhindert, dass die Approximation der Q-Funktion über mehrere Iterationen hinweg in einem lokalen Minimum der Fehlerfunktion gefangen ist. In Algorithmus 5 ist das gesamte Basisverfahren des NFQ zur Berechnung einer Iterationsschritte der neuronalen Q-Funktion dargestellt. Die Aktualisierung der Q-Werte in

Algorithmus 4 $MIN(Q, x)$

Require: x : Zustand, Q : neuronale Approximation der Q-Funktion**Ensure:** $q_{min} = \min_{u^i \in U, i=1, \dots, \#U} Q(x, u^i)$ $q_{min} \leftarrow \infty$ **for** $i = 1, \dots, n$ **do** $q_{min} = \min(Q(x, u^i), q_{min})$ **end for****return** q_{min}

Algorithmus 5 $NFQUpdate(Q_k, \mathcal{D})$

Require: Q_k : Approximation der Q-Funktion, \mathcal{D} : Menge der beobachteten Übergänge**Ensure:** Q_{k+1} : aktualisierte neuronale Approximation der Q-Funktion $\mathcal{P} = BatchQUpdate(Q_k, \mathcal{D})$ $Q_{k+1} = RPropTraining(Q_0, \mathcal{P})$ **return** Q_{k+1}

Algorithmus 3 ist mit der stochastischen Approximation aus Formel 2.10 mit Parameter α angegeben. Durch die Verwendung eines neuronalen Netzes mit epochenbasiertem Lernen werden durch das neuronale Netz selbst die Erwartungswerte aus den Trainingsmustern gelernt. Somit ist die zusätzliche Verwendung einer stochastischen Approximation nicht zwingend erforderlich. Für die in dieser Arbeit durchgeführten Lernversuche wird deshalb $\alpha = 1$ gewählt und somit keine zusätzliche, explizite stochastische Approximation durchgeführt.

2.3.3 NFQ als Lernverfahren

Die in Abschnitt 2.3.2 angegebene Iterationsvorschrift kann auf verschiedene Weise im Reinforcement-Lernen angewandt werden. Der für diese Arbeit interessanteste Fall ist die Verwendung des Verfahrens in einem explorativen Lernprozess. Unter einem explorativen Lernprozess wird der Fall verstanden, in dem der Agent zu Beginn des Lernprozesses keinerlei Erfahrung über den MDP besitzt. Er beginnt mit einer leeren Datenbasis und einem zufällig initialisierten neuronalen Q-Netz und somit mit einer rein zufälligen Strategie. Auf sogenannten Explorations- oder Interaktionsepisoden sammelt der Agent unter Verwendung seiner aktuellen Strategie Erfahrung über das System, indem er die Übergänge des Systems während der Episode beobachtet. Nach jeder Explorationsepisode wird mit der gesamten bis dorthin gesammelten Erfahrung ein Aktualisierungsschritt mit dem NFQ-Grundverfahren und der aktuellen Q-Funktion durchgeführt. Anschließend wird mit der daraus entstandenen Q-Funktion wiederum eine Explorationsepisode durchgeführt, auf der neue Daten gesammelt werden. Diese Schritte werden abwechselnd immer weiter durchgeführt. In Algorithmus 6 ist die Durchführung der Interaktionsepisode während eines explorativen Lernprozesses als Pseudocode dargestellt. In jedem Zeitschritt berechnet der Agent die Strategie im beobachteten Zustand durch gierige Auswertung der aktuellen Q-Funktion und wendet diese auf den MDP an. Die beobachteten Übergänge werden gespeichert und stehen nach Beendigung der Episode dem Lernprozess zur Verfügung. Das Hinzufügen einer zufälligen Exploration mit einer Explorationsrate $0 \leq \epsilon \leq 1$, in der die Auswahl der Aktion rein zufällig durchgeführt wird, unterstützt den Lernprozess und lässt das Entdecken von Bereichen des Zustandsraumes zu, die die aktuelle Strategie nicht

Algorithmus 6 InteraktionsEpisode(\mathcal{Q} , ϵ)**Require:** \mathcal{Q} Approximation der Q-Funktion, ϵ Explorationsrate $\mathcal{D} \leftarrow \emptyset$, $t \leftarrow 0$, $x_t \leftarrow$ Beobachtung aus Welt**repeat** $u_t = \begin{cases} \text{zufällig } \in U & \text{mit Wahrscheinlichkeit } \epsilon \\ \text{Strategie}(\mathcal{Q}, x_t) & \text{sonst} \end{cases}, u_t \rightarrow \text{Welt}$ $(x_{t+1}, c_t) \leftarrow$ Beobachtung aus Welt $\mathcal{D} \leftarrow \mathcal{D} \cup (x_t, u_t, x_{t+1}, c_t)$, $t \leftarrow t + 1$ **until** $t = t_{max}$ oder x_t Terminalzustand**return** \mathcal{D}

erreichen würde. In Algorithmus 2.3.3 ist die Berechnung der Strategie durch gierige Auswer-

Algorithmus 7 Strategie(\mathcal{Q} , x)**Require:** Zustand x , neuronale Approximation der Q-Funktion \mathcal{Q} **Ensure:** $u_{min} = \underset{u^i \in U, i=1, \dots, \#U}{\operatorname{argmin}} \mathcal{Q}(x, u^i)$ $q_{min} \leftarrow \infty$ **for** $i = 1, \dots, n$ **do** $q = \mathcal{Q}(x, u^i)$ **if** $q < q_{min}$ **then** $q_{min} = q$, $u_{min} = u^i$ **end if****end for****return** u_{min}

tung der Q-Funktion dargestellt. Im Unterschied zur Berechnung des minimalen Q-Wertes in einem Zustand ist die Aktion die dieses Minimum erzeugt nicht eindeutig. Es können mehrere Aktionen auftreten, die den gleichen Q-Wert besitzen. In diesem Fall wird die gewählte Aktion zufällig unter den möglichen Kandidaten mit gleichem Q-Wert gezogen. In Algorithmus 2.3.3

Algorithmus 8 Exploratives NFQ $k \leftarrow 0$, $\mathcal{D} \leftarrow \emptyset$, $\mathcal{Q}_0 \leftarrow$ Neuronales Netz, initialisiert mit zufälligen Gewichten**repeat** $\mathcal{D} = \mathcal{D} \cup \text{InteraktionsEpisode}(\mathcal{Q}_k)$ $\mathcal{Q}_{k+1} = \text{NFQUpdate}(\mathcal{Q}_k, \mathcal{D})$ **until** $k = K$ **return** \mathcal{Q}_K

ist der gesamte Lernprozess des explorativen Lernens mit NFQ dargestellt.

Im Gegensatz zum explorativen Lernprozess kann die Iteration des NFQ-Basisverfahrens auch direkt auf einer bestehenden Datenbasis von Übergängen des MDPs durchgeführt werden (siehe Algorithmus 2.3.3). Die Daten selbst können auf verschiedene Arten am MDP gesammelt worden sein. Je nach Problemstellung können die Daten auf suboptimalen, heuristischen Lösungen mit Exploration oder durch die Anwendung einer rein zufälligen Strategie stammen. Eine interessante Möglichkeit ist die Wiederverwertung einer Datenbasis, die mit einem explorativen Lernprozess gesammelt wurde. Die Datenbasis enthält in ihren Übergängen wichtige

Algorithmus 9 Datenbasiertes NFQ (\mathcal{D})

Require: \mathcal{D} : Menge der beobachteten Übergänge $k \leftarrow 0$, $Q_0 \leftarrow$ Neuronales Netz, initialisiert mit zufälligen Gewichten**repeat** $Q_{k+1} = NFQUpdate(Q_k, \mathcal{D})$ **until** $k = K$ **return** Q_K

Information über die Übergangswahrscheinlichkeiten des MDPs. Das Optimierungsziel lässt sich nun ändern, indem die direkten Kostensignale in der Datenbasis mit einer neuen Funktion $c(x, u)$ berechnet werden. Auf diese Weise kann mit einem rein datenbasierten Lernprozess in wenigen Schritten eine Strategie gelernt werden.

Die vorgestellten Ausprägungen des Lernprozesses sind die zwei am häufigsten benutzten Vorgehensweisen, in denen NFQ angewandt wird. Es lassen sich jedoch beliebige Kombinationen aus diesen Vorgehensweisen bilden. So kann ein Wechsel von einem datenbasierten Vorgehen zu einem explorativen Vorgehen sinnvoll sein, wenn bereits Daten über den Prozess vorliegen, diese aber den MDP noch nicht ausreichend beschreiben. Andererseits kann ein Wechsel von einem explorativen zu einem datenbasierten Lernprozess vorgenommen werden, wenn genügend Erfahrung in der Datenbasis vorliegt. Zusätzlich sind Lernprozesse denkbar, bei denen beliebig zwischen den zwei Vorgehen gewechselt wird.

2.3.4 Praktische Aspekte bei der Anwendung von NFQ

Das NFQ-Verfahren ist unter dem Eindruck von Erfahrungen entstanden, die beim praktischen Einsatz von Reinforcement Lernverfahren in anwendungsbezogenen Problemstellungen gesammelt wurden. In diesem Sinne wurde darauf Wert gelegt, ein Verfahren mit wenigen Parametern zu entwickeln. Im folgenden werden die für die praktische Verwendung des Verfahrens relevanten Parameter beschrieben. Für möglichst viele Parameter werden dabei Standardwerte angegeben, die sich in der praktischen Anwendung bewährt haben und in dieser Arbeit für die durchgeführten Versuche benutzt werden.

Einer dieser freien Parameter ist die Wahl der Topologie und der verwendeten Aktivierungsfunktionen des neuronalen Netzes. Mit der Anzahl der verwendeten Neuronen wächst die Fähigkeit des Netzes, komplexere Q-Funktionen zu repräsentieren. Gleichzeitig steigt jedoch ebenfalls die benötigte Rechenzeit für das Verfahren. Die praktische Anwendung des Verfahrens hat gezeigt, dass wenige Kandidaten von Topologien ausreichen, um ein sehr weites Spektrum von Lernproblemen zu lösen. Zumeist wird die Netztopologie nach wenigen Versuchen festgelegt und muss nicht mehr verändert werden. Verwendet werden in der Praxis neuronale Netze mit ein bis zwei verdeckten Schichten, die jeweils 5 bis 20 Neuronen enthalten. Für die verdeckten Schichten in der neuronalen Repräsentation der Q-Funktion $Q(x, u, w_q)$ werden in dieser Arbeit ausschließlich sigmoide Aktivierungsfunktionen verwendet. Die Verwendung von symmetrischen Aktivierungsfunktionen ist ebenfalls möglich und beeinflusst den Lernvorgang nicht signifikant. Als Konvention für diese Arbeit soll ebenfalls festgelegt sein, dass das Ausgabeneuron eine sigmoide Aktivierungsfunktion besitzt. Die Verwendung von linearen Aktivierungsfunktionen, sowohl in den verdeckten Schichten als auch im Ausgabeneuron, verschlechtert den Lernvorgang typischerweise deutlich. Die Eingabeneuronen des neuronalen Netzes haben per Definition eine lineare Aktivierungsfunktion. Dies bedeutet, dass sie, ge-

wichtet durch die entsprechenden Verbindungen, direkt in die Berechnung der Aktivierung der Neuronen der darauffolgenden Schichten eingehen. Es ist somit möglich, die Eingabe, bestehend aus Zustand und Aktion des Trainingsmusters, aus der Datenmenge direkt an das Netz anzulegen. Da sich die Zustände und Aktionen in verschiedenen Anwendungsproblemen in ihrem Wertebereich um mehrere Größenordnungen unterscheiden können, führt dies zu extrem kleinen oder großen Gewichten, die für die Repräsentation der zu lernenden Funktion notwendig sind. In der Praxis werden deshalb die Daten aus der Datenmenge in jeder Dimension der Eingabe an die Q-Funktion auf Mittelwert Null und Varianz Eins normalisiert. Dies wird automatisch vor jedem Iterationsschritt des NFQ-Verfahrens, basierend auf den in der Datenmenge zu diesem Zeitpunkt vorhandenen Daten, durchgeführt.

Da für das Ausgabeneuron der neuronalen Q-Funktion eine sigmoide Aktivierungsfunktion Verwendung findet, kann sie ausschließlich Werte zwischen Null und Eins repräsentieren. Durch geeignete Skalierung und Verschiebung, ähnlich zur Normalisierung der Netzeingaben, können mit dem verwendeten neuronalen Netz auch beliebige Q-Werte dargestellt werden. In der Praxis wird die Q-Funktion für das Lernverfahren auf ein Intervall $[Q_{min}^{bound}, Q_{max}^{bound}]$ beschränkt. Die Intervallgrenzen müssen so gewählt sein, dass alle Werte der zu erwartenden optimalen Q-Funktion in diesem Intervall liegen. Während des Lernvorganges werden in Algorithmus 3 Werte \hat{Q} außerhalb des Intervalles auf die Intervallgrenzen zurückgesetzt. Dies hat den Vorteil, dass die Q-Funktion während des Lernvorganges beschränkt bleibt. Für die Versuche in dieser Arbeit werden diese Werte in allen Versuchen zu $Q_{min}^{bound} = 0.05$ und $Q_{max}^{bound} = 0.95$ gewählt. Durch diese Begrenzung der Q-Werte wird der Wertebereich der sigmoiden Aktivierungsfunktion des Ausgabeneurons in der neuronalen Q-Funktion bewusst nicht voll ausgenutzt, um die benötigten Gewichte nicht zu groß werden zu lassen. Die geeignete Wahl der Kostenfunktionen stellt sicher, dass die zu erwartenden Q-Werte aus diesem Intervall stammen. Bei SKP-Problemen ist eine offensichtliche Anpassung, dass der Terminalzustand x^+ Terminalkosten von $\hat{c}(x^+) = Q_{min}^{bound}$ besitzen muss.

Aus der Sicht des Reinforcement-Lernens ist das zu lösende Optimierungsproblem vollständig durch den betrachteten MDP modelliert. Für die Anwendung des Reinforcement-Lernens auf eine praktische Problemstellung muss diese somit in Form eines MDPs modelliert werden. Dabei gliedern sich aus der Sicht der Problemstellung die durchzuführenden Modellierungsschritte in die geeignete Wahl der Repräsentation des Zustands- und Aktionsvektors, die Auswahl geeigneter diskreter Aktionen, mit denen das Problem zu lösen ist und die Herstellung der Interaktionsmöglichkeit, mit der durch das Problem gegebenen Übergangsfunktion. Dies kann ein simuliertes oder auch ein reales System sein, dessen Übergangsfunktionen selbst dem Agenten nicht zur Verfügung stehen. Das für die Problemstellung relevante Optimierungsziel wird durch die Kostenfunktion $c(x, u)$ modelliert. Da die Kostenfunktion für die Lösung von praktischen Problemen vom Benutzer vorgegeben werden muss, ist ihre Repräsentation in der Datenbasis \mathcal{D} nicht zwingend notwendig. In Abbildung 2.6 ist sie deshalb schematisch von der Datenbasis getrennt aufgeführt.

Durch die Verwendung von neuronalen Netzen als nichtlineare Funktionsapproximatoren können keine allgemeinen Konvergenzaussagen für das Verfahren getroffen werden. In der aktuellen Entwicklung im Bereich des NFQ-Verfahrens wird an Metaverfahren gearbeitet, die es erlauben, die Wahrscheinlichkeit für die Auswahl einer annähernd optimalen Strategie bei sehr wenigen Testepisoden zu erhöhen. Speziell für die effektive Interaktionszeit an realen Systemen ist dies ein wichtiger Aspekt. In dieser Arbeit ist dieser Aspekt jedoch nicht im Fokus des Interesses. Nach jeder Trainingsiteration werden die Regler zur besseren Bewertung an einer großen Menge von Testepisoden aus verschiedenen Startzuständen getestet, ohne die dadurch

Tabelle 2.2: Zusammenfassung der zentralen Parameter des NFQ Lernverfahrens für praktische Problemstellungen

Modellierung:
Repräsentation der Aktionen
Auswahl der diskreten Aktionsmenge
Repräsentation des Zustands
Definition der Kostenfunktion
Exploratives Lernen:
Explorationsrate
Startzustände
Länge der Interaktionsepisoden
Parameter des Grundverfahrens:
Topologie des Q-Netzes
Anzahl der Epochen für überwachtes Lernen

entstehende Interaktionszeit zu beachten. Dadurch ist sichergestellt, dass die Leistungsfähigkeit der Verfahren ausreichend evaluiert werden kann.

2.4 Zusammenfassung

Das Reinforcement-Lernen erlaubt die Lösung von sequentiellen Entscheidungsproblemen, die in Form eines MDPs vorliegen. Für praktische Problemstellungen ist der modellfreie Fall auf kontinuierlichen Zustandsräumen interessant, für den die Übergangswahrscheinlichkeiten des MDPs nicht a priori bekannt sind, sondern durch Interaktion entlang von Trajektorien in den Lernprozess einfließen. Die neuronalen Netze bilden eine wichtige Grundlage für die in dieser Arbeit betrachteten Verfahren. Durch ihre Fähigkeit, beliebige, nichtlineare Funktionen zu realisieren, werden sie als allgemeine Funktionsapproximatoren für das Reinforcement-Lernen eingesetzt. Ihre Eigenschaft, differenzierbare Funktionen zu realisieren, deren Ableitung im Netz berechnet werden kann, wird in einem folgenden Kapitel zur Erweiterung des NFQ-Verfahrens auf kontinuierliche Aktionen genutzt. Im Verlauf dieser Arbeit dient das NFQ als zentrales Verfahren für die Erstellung und Untersuchung selbstlernender Regler. NFQ ist ein modellfreies Batch Reinforcement Lernverfahren für kontinuierliche Zustände und diskrete Aktionen. Durch die Verwendung neuronaler Netze zur Approximation der Q-Funktion mit RProp als leistungsfähiges Lernverfahren ergibt sich ein sehr dateneffizientes und gegen seine Parameter in weiten Bereichen robustes Lernverfahren. Seine Dateneffizienz macht das Verfahren für explorative Lernprozesse an realen Systemen interessant, bei denen durch wenige Interaktionsepisoden ohne Vorwissen eine Strategie gelernt werden soll. In Tabelle 2.2 sind die bei der Modellierung des Lernproblems festzulegenden Parameter zusammengefasst.

Selbstlernende Regler als Optimierungsprozess mit NFQ

Mit dem 'Neural Fitted Q-Iteration' (NFQ) steht ein Reinforcement-Lernverfahren zur Verfügung, das es ermöglicht, anspruchsvolle Optimierungsprobleme in Form eines Markovschen Entscheidungsprozesses effizient zu lösen. Insbesondere ist die Anzahl von benötigten Interaktionen mit dem zu optimierenden Prozess im Vergleich zum herkömmlichen Q-Lernen um viele Größenordnungen gefallen. Dies ermöglicht die Anwendung des NFQ-Verfahrens auf praktische Problemstellungen und reale Systeme. Das vorliegende Kapitel behandelt die Modellierung der selbstlernenden Regler für regelungstechnische Problemstellungen als Optimierungsprozess mit NFQ. Das Leitbild für dieser Modellierung bildet der Wunsch nach einer intelligenten Regler-Komponente für praktische Problemstellungen. Für diese wird vom Benutzer nicht spezifiziert, wie sie ein Problem zu lösen hat, sondern nur, welches Ziel sie zu erreichen hat. Oder anders ausgedrückt, der Entwurf der selbstlernenden Regler und ihre Modellierung soll mit möglichst wenig Wissen über das zu lösende Regelungsproblem durchführbar sein.

In Abschnitt 3.1 wird die Problemstellung der Regelung anhand der Begrifflichkeiten der klassischen Regelungstechnik diskutiert. Basierend auf diesen wird in Abschnitt 3.2 die Modellierung der zeitoptimalen Regelung mit NFQ vorgestellt. Ein Hauptbeitrag des vorliegenden Kapitels ist die Entwicklung eines Verfahrens zur Korrektur des systematisch auftretenden Fehlers in der Q-Funktion in Abschnitt 3.3. Da die Regelung einen andauernden Prozess ohne festen Endzeitpunkt darstellt, ist die Modellierung des Optimierungsprozesses als SKP-Problem mit Terminalzustand nicht möglich. Als Konsequenz degeneriert die Q-Funktion im Verlauf des Lernprozesses mit NFQ. Der dafür hauptsächlich verantwortliche systematische Fehler in der Aktualisierung der Q-Funktion wird durch das entwickelte Verfahren effizient korrigiert. Ein weiterer Beitrag des Kapitels besteht in der Verfeinerung der zeitoptimalen Kostenfunktion in Abschnitt 3.4. Diese entwickelte Kostenfunktion ist eine zentrale Voraussetzung für die Erzeugung präzise arbeitender, selbstlernender Regler. Darüber hinaus behebt diese Verfeinerung der Kostenfunktion ein Problem der ursprünglichen zeitoptimalen Formulierung, das zu Schwierigkeiten bei der Approximation führt. Die Modellierung selbstlernenden Regler unter einer externen Führungsgröße wird in Abschnitt 3.5 betrachtet. Dies ist eine Aufgabenstellung, die in der Literatur zum Reinforcement-Lernen kaum Beachtung findet. Sie stellt jedoch eine zentrale Voraussetzung für die Erzeugung praxisrelevanter, selbstlernender Regler dar. Die Verwendung von diskreten Aktionen im NFQ-Verfahren führt nicht bei allen regelungs-

technischen Prozesses zu einer zufriedenstellenden Genauigkeit der selbstlernenden Regler. In Abschnitt 3.6 wird mit den dynamischen Ausgabe-Elementen (DAE) ein Verfahren zur künstlichen Erweiterung der Systemdynamik solcher Prozesse vorgestellt. Diese Erweiterung erlaubt, die Genauigkeit mit diskreten Aktionen deutlich zu verbessern. Die Weiterentwicklung der DAE zu intelligenten DAE (IDAE) in 3.6.2 ermöglicht die Integration klassischer Regelungskonzepte, mit denen, unter Benutzung von Vorwissen, die Genauigkeit weiter verbessert werden kann. Ein DC-Motor mit nichtlinearer Endstufe dient in diesem Kapitel als Beispiel einer Regelungstechnischen Aufgabenstellung. An diesem werden die vorgestellten Verfahren empirisch evaluiert.

3.1 Grundzüge der Regelungstechnik

Beim Vorgang der Regelung wird der Istwert der Regelgröße einer Regelstrecke erfasst und mit einer Sollwert bzw. Führungsgröße verglichen¹. Weichen beide voneinander ab, z.B. durch Einwirkung einer Störgröße oder schnelle Veränderung des Sollwerts, dann wird mit einem entsprechenden Signal an die Stelleinrichtung der Istwert an den Sollwert herangeführt. In der

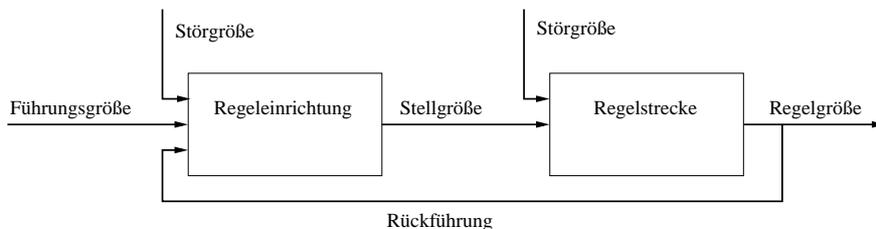


Abbildung 3.1: Die schematische Darstellung eines Regelkreises.

Regelungstechnik beschreibt der in Abbildung 3.1 dargestellte Regelkreis den geschlossenen Wirkungsablauf aller Glieder, die an einer Regelung beteiligt sind. Der Regelkreis wird unterteilt in Regeleinrichtung und Regelstrecke. Die Regeleinrichtung beinhaltet die Komponenten zur Erfassung des Istwertes der Regelgröße (Messeinrichtung, Messsystem), zum Vergleich dieses Istwertes mit der Führungsgröße und zur Bildung der Stellgröße (Steller). Häufig wird die gesamte Regeleinrichtung als Regler bezeichnet. Im engeren Sinn versteht man darunter nur das Rechenglied (Gerät, Programm, Algorithmus) innerhalb der Regeleinrichtung, welches aus der Regelabweichung, der Differenz der Regelgröße und der Führungsgröße eine Ausgabe (Reglerausgabe) an den Steller generiert. Um eine aufgabengemäße Beeinflussung der Regelstrecke über das Stellglied zu erreichen, müssen die Eigenschaften des Reglers auf die Regelstrecke abgestimmt sein. Die Regelstrecke ist derjenige Teil einer Regelung, der den aufgabengemäß zu beeinflussenden Bereich darstellt und an dem die Regelgröße auftritt. Die Regelstrecke beinhaltet das Stellglied, das, basierend auf der Stellgröße, die Regelstrecke beeinflusst sowie die mechanischen, physikalischen oder elektrischen Übertragungsglieder des Prozesses. Sowohl auf die Regeleinrichtung als auch auf die Regelstrecke können Störgrößen einwirken. Die Begriffe Sollwert und Führungsgröße einer Regelung sollen in dieser Arbeit synonym verwendet werden. Sie stellen eine von der Regelung nicht unmittelbar beeinflussbare Größe dar, die ihr von außen zugeführt wird. In der klassischen Regelungstechnik nimmt der Sollwert direkt seine Größe an, während die Führungsgröße die Veränderung der Regelgröße,

¹Die verwendeten Begrifflichkeiten entstammen der DIN 19226 in ihrer Neufassung.

z.B. auf einer Rampe, bewirkt, um den Sollwert zu erreichen. Im stationären Bereich der Regelung decken sich beide. Die Führungsgröße kann jedoch auch als Sollwert gesehen werden, der sich über die Zeit ändert.

Im Unterschied zur Regelung beschreibt Föllinger [28] die 'Steuerung' eines dynamischen Systems als offene Wirkungskette. Ähnlich der Regelung beeinflusst die Steuerung einen Prozess basierend auf einer Führungsgröße. Föllinger unterscheidet zwischen der Regelung mit Rückführung und der Steuerung mit Rückführung. Bei letzterer findet keine kontinuierliche Rückführung der Regelgröße statt und es wird explizit kein Soll-Istwert-Vergleich durchgeführt. Die in dieser Arbeit betrachtete Beeinflussung eines dynamischen Systems soll somit explizit als Regelung aufgefasst werden, die als ein Teil einer Steuerung eines komplexeren Wirkungsablaufes eingesetzt werden kann.

Für die vorliegende Arbeit ist die getrennte Betrachtung der Eigenschaften von Messeinrichtung, Stellglied, Steller und Regelstrecke für den Entwurf der selbstlernenden Regler nicht von Bedeutung. Es wird die rein digitale Form eines Regelkreises betrachtet, bei welcher der Regler in Form eines Algorithmus implementiert wird, der die aufgabengemäße Reglerausgabe für einen technischen oder physikalischen Prozess berechnet. Wie in Abbildung 3.2 darge-

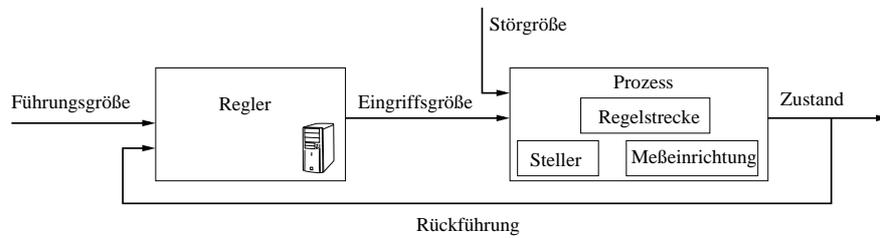


Abbildung 3.2: Die für diese Arbeit verwendete schematische Darstellung eines Regelkreises mit Trennung zwischen physikalischen Komponenten und Regelalgorithmus.

stellt, werden alle Komponenten des Regelkreises, die nicht den eigentlichen Regelalgorithmus betreffen, in einem Prozess zusammengefasst. Dieser regelungstechnische Prozess beschreibt somit das Verhalten aller physikalischen und technischen Eigenschaften des Regelkreises, ihr Verhalten auf Ausgaben des Reglers und auf Störungen von außen. Als Regler wird nur der eigentliche Regelalgorithmus aufgefasst, der auf einem digitalen System implementiert wird. Der Regler erhält die Informationen über den aktuellen Zustand des Prozesses in Form eines Zustandsvektors x aus Prozessgrößen (Prozessvariablen). Diese wurden im Prozess selbst durch geeignete Messeinrichtungen erfasst und liegen in digitaler Form vor. Die Regelgröße selbst ist im Zustandsvektor enthalten, der darüber hinaus weitere Informationen über den Prozess enthält. Die Ausgabe des Reglers wird als Eingriffsgröße für den Prozess bezeichnet. Diese berechnet er, basierend auf der aktuellen Führungsgröße und dem aktuellen Zustand des Prozesses. Treten Abweichungen der Regelgröße von der Führungsgröße auf, so muss der Regler seine Ausgabe geeignet verändern, um dieser Regelabweichung entgegenzuwirken. Bei der digitalen Regelung können sich Führungs- und Regelgrößen auf mehrere Prozessvariablen beziehen und somit mehrdimensionale Größen darstellen.

3.1.1 Gütekriterien für die Regelung

Das in dieser Arbeit betrachtete Reglerkonzept basiert auf einem lernenden Verfahren. Es müssen Gütekriterien definiert werden, die für die Untersuchung der Leistungsfähigkeit der ge-

lernten Regler geeignet sind und den direkten Vergleich verschiedener Verfahren und Regler ermöglichen. In der klassischen Regelungstechnik wird die Güte eines Reglers durch verschiedene Maße ermittelt, die im folgenden diskutiert werden.

Das zentrale Gütekriterium für die Bewertung eines Reglers ist die Regelabweichung (Sollwertabweichung), die Differenz zwischen Führungsgröße und Regelgröße. Man unterscheidet zur Bewertung des Reglers zwei Fälle, in denen diese Regelabweichung von Interesse ist. Zum einen die bleibende Regelabweichung als die im Beharrungszustand der Regelgröße vorhandene Regelabweichung. Zum anderen die vorübergehende Regelabweichung die während des Übergangs von einem Beharrungszustand in einen anderen nach einer Änderung der Störgröße oder der Führungsgröße auftritt. Die Bezugsgröße in diesem Fall, ist der Sollwert der Regelgröße im neuen Beharrungszustand. In diesem Fall sind die relevanten Kenngrößen gegeben durch die Ausregelzeit und den Toleranzbereich. Die Ausregelzeit ist eine Zeitspanne, die beginnt, wenn der Wert der Regelgröße nach einem Sprung der Störgröße oder der Führungsgröße den vorgegebenen Toleranzbereich verlässt und die endet, wenn er in den Bereich wieder eintritt und dort dauerhaft verbleibt. Ein weiteres Gütekriterium der Regelung ist ihr Führungsverhalten. Dieses gibt Auskunft über das Verhalten der Regelgröße bei sich verändernder Führungsgröße.

Für die Bewertung der in dieser Arbeit betrachteten Regler werden drei Kennzahlen definiert. Diese Kennzahlen stellen eine quantitative Erfassung der Bewertungskriterien dar, mit denen die Regler unter verschiedenen Parametern und Konzepten direkt verglichen werden können. Die erste dieser Kennzahlen, \bar{N} , bewertet die Zeit, die sich die Regelgröße außerhalb des Toleranzbereiches der Regelung befindet. Da in dieser Arbeit ausschließlich zeitdiskrete Regler betrachtet werden, wird direkt die Anzahl der Zeitschritte erfasst, die sich ein Regler außerhalb des Toleranzbereiches befindet. Typischerweise wird diese Kennzahl als Durchschnittswert über mehrere Regeltrajektorien gebildet. Je nach Regelungsaufgabe werden die Regeltrajektorien, ausgehend von verschiedenen Prozesszuständen und unter verschiedenen Sollwerten, durchgeführt. Die Sollwerte werden auf der Trajektorie konstant gehalten. Auf diese Weise wird sichergestellt, dass die Kennzahl eine möglichst gute Auskunft über die Ausregelzeit im gesamten Arbeitsbereich der Regelung gibt. Über dieselben Trajektorien wird auch die Kennzahl \bar{e}_∞ erfasst, um die bleibende Regelabweichung zu bewerten. Für diese wird am Ende jeder Regeltrajektorie die mittlere Regelabweichung über eine bestimmte Anzahl von Schritten bestimmt. Die Kennzahl selbst wird wiederum als Durchschnitt über alle Trajektorien angegeben. Für die Bewertung des Führungsverhaltens wird die Kennzahl \bar{e}_T eingeführt. Diese stellt die mittlere Regelabweichung über eine gesamte Regeltrajektorie dar. Typischerweise wird diese erfasst, indem in einem Profil das zeitliche Verhalten der Führungsgröße über eine Trajektorie festgelegt wird. An diesem Profil wird das Folgeverhalten des Reglers in Form der mittleren Regelabweichung gemessen.

3.1.2 Dynamische Systeme und ihre Eigenschaften aus regelungstechnischer Sicht

Die Analyse der Eigenschaften dynamischer Systeme stellt für den klassischen Reglerentwurf eine zentrale Voraussetzung dar. Aus ihr werden die benötigten Regelkonzepte abgeleitet und geeignet parametrisiert. Für die Idee der selbstlernenden Regler dient die Analyse und die Erfassung der Systemgleichungen ausschließlich der Diskussion der vorhandenen Problemklassen und deren Eigenschaften.

Ein allgemeines dynamisches System ist, aus regelungstechnischer Sicht, die mathematische

Repräsentation eines Prozessverhaltens im Zeitbereich. Diesem Prozess liegen typischerweise technische und physikalische Zusammenhänge zu Grunde. Aus der Beschreibung des Verhaltens können ihnen bestimmte, für die Beherrschung dieser Systeme wichtige, Eigenschaften zugeordnet werden. Die Beschreibung eines dynamischen Systems im Zeitbereich basiert auf einem Vektor $x \in \mathbb{R}^n$ von Systemvariablen, die den Zustand des Prozesses zu jedem Zeitpunkt charakterisieren. Zudem ist, in dem für diese Betrachtung interessanten Fall, eine externe Eingriffsgröße $u \in \mathbb{R}^m$ vorhanden, die den Prozess basierend auf seiner Dynamik beeinflusst. Die Entwicklung des Zustandes eines Prozesses über der Zeit wird als Differenzialgleichung in der Form

$$\dot{x} = f(x_t, u_t) \quad (3.1)$$

dargestellt. Dies ist die allgemeinste Form eines dynamischen Systems, dessen Dynamik durch beliebige, insbesondere nichtlineare Funktionen, beschrieben wird. Das Systemverhalten hängt ausschließlich von den Systemvariablen und den Eingriffsgrößen zum Zeitpunkt t ab und nicht von der Zeit der Ausführung. Dies zeigt sich in der dargestellten mathematischen Formulierung insbesondere dadurch, dass die Funktion f über die Zeit nicht veränderlich sind. Allgemeine dynamische Systeme mit dieser Eigenschaft werden als autonom bezeichnet. In einem Regelkreis mit digitalem Mess- und Regelungsglied wird die Information in diskreten Zeitschritten verarbeitet. Dies entspricht der mathematischen Formulierung eines zeitdiskreten dynamischen Systems. Bei diesem wird das Systemverhalten zu festen Zeitschritten $t = 0, 1, 2, 3, \dots$ betrachtet, zwischen denen jeweils eine konstante Zeitspanne Δ_t liegt. Ein zeitdiskretes, autonomes, dynamisches System wird als dynamischer Prozess mit Systemdynamik

$$x_{t+1} = f(x_t, u_t) \quad (3.2)$$

dargestellt.

Eine für die klassische Regelungstechnik besonders wichtige Untermenge der dynamischen Systeme sind die linearen Systeme. Ein System wird als linear bezeichnet, wenn sich die Systemdynamik in der Form $\dot{x} = Ax_t + Bu_t$ bzw. im zeitdiskreten Fall $x_{t+1} = Ax_t + Bu_t$ mit geeigneten Matrizen A und B schreiben lässt. Für diese Untermenge der dynamischen Systeme steht eine große Anzahl von theoretischen Betrachtungsweisen, Hilfsmitteln und standardisierten Entwurfsmustern für den Reglerentwurf zur Verfügung. Dies gilt vor allem für die linearen, zeitinvarianten Systeme (Linear Time Invariant, LTI), die eine Untermenge der autonomen Systeme darstellen. Annähernd alle in der Praxis betrachteten physikalischen und technischen Prozesse besitzen jedoch ein gewisses Maß an Nichtlinearität in ihrem Systemverhalten. Dabei unterscheidet man zwischen den vom zu regelnden Prozess selbst verursachten Nichtlinearitäten und den durch den Regelkreis künstlich hinzugefügten Nichtlinearitäten. Prozessnichtlinearitäten sind durch die physikalischen und technischen Eigenschaften des Prozesses bedingt. Sie werden z.B. durch Haft- und Gleitreibung zwischen Oberflächen und in Lagern verursacht, betreffen jedoch auch Beschränkungen des Arbeitsbereiches und der Eingriffsgrößen. Die künstlichen Nichtlinearitäten hingegen werden vom Regler durch seine Struktur und Arbeitsweise in den Regelkreis eingeführt. Durch ein nichtlineares Regelgesetz kann der Regler Prozessnichtlinearitäten geeignet entgegenwirken. Doch auch bei Systemen mit linearem Prozessverhalten kann ein solches Regelgesetz notwendig werden, wenn das Optimierungskriterium nichtlineare Eigenschaften besitzt.

Der klassische Entwurfsprozess für lineare und nichtlineare Regelgesetze basiert auf der Analyse der Systemdynamik. Für diese Analyse müssen die Systemgleichungen für den zu regelnden Prozess möglichst genau erfasst und parametrisiert werden. Im Gegensatz zu den linearen

Systemen ist der Reglerentwurf für nichtlineare Systeme ein weitaus aufwändigerer Prozess. Dies liegt vor allem daran, dass die Auswirkungen der Prozessnichtlinearitäten auf das Systemverhalten nur schwer analysierbar und vorhersehbar sind. Der Reglerentwurf für nichtlineare Systeme basiert häufig auf der Linearisierung der Systemdynamik um einen speziellen Arbeitspunkt der Regelung. Für die linearisierte Systemdynamik wird dann analytisch ein linearer Regler entworfen. Dieser Entwurfsprozess ist jedoch aufwändig und kann den Entwurf von mehreren Reglern für verschiedene Arbeitspunkte der Regelung notwendig machen. Zusätzlich wird durch die Linearisierung ein Teil der Systemdynamik vernachlässigt, was die erzielbare Regelgüte beeinträchtigt. Der Entwurfsprozess nichtlinearer Regelgesetze, andererseits, ist weitaus weniger standardisiert und basiert auf einem hohen Maß an Expertenwissen über den zu regelnden Prozess (z.B. 'Sliding Mode Control' [95]).

Die in dieser Arbeit betrachteten selbstlernenden Regler stellen durch ihre Struktur und Arbeitsweise ein nichtlineares Regelgesetz dar. Die aufwändige Analyse des zu regelnden Prozesses entfällt und wird durch einen Lernprozess in Form einer Optimierung ersetzt. In diesem Sinne stellen sie ein einheitliches Entwurfsmuster für Regler dar, bei dem nicht explizit zwischen linearen und nichtlinearen Prozessen unterschieden werden muss. Bei der Entwicklung der selbstlernenden Regler liegt der Fokus auf der Betrachtung autonomer, zeitdiskreter, dynamischer Systeme. Erweiterungen der selbstlernenden Regler auf zeitabhängige dynamische Systeme ist denkbar, liegt jedoch nicht im Fokus dieser Arbeit. Des Weiteren wird davon ausgegangen, dass alle für die Beschreibung der Dynamik notwendigen Prozessvariablen voll beobachtbar sind. In praktischen Anwendungen kann dies den Entwurf von geeigneten Beobachtern notwendig machen. Die hierfür notwendigen Verfahren reichen von einfachen Filtern über komplexe Schätzer bis hin zu Sensorfusions- und Prediktionsaufgaben in der Robotik [29] und bilden ein eigenes Forschungsgebiet.

Der Entwurf eines Reglers ist eng verbunden mit der Forderung nach einem stabilen Systemverhalten des, durch den Regler geschlossenen, dynamischen Systems. Oder anders ausgedrückt, auch wenn das Systemverhalten eines offenen dynamischen Systems nicht zufriedenstellend ist, so ist es das Ziel, einen geeigneten Regler zu erstellen, der gewährleistet, dass die Stabilitätseigenschaften des geschlossenen Systems zufriedenstellend sind. In der allgemeinen Betrachtung dynamischer Systeme ohne die Beschränkung auf den linearen Fall gibt es mehrere Definitionen des Begriffes der Stabilität. Gegeben sei ein zeitdiskretes dynamisches System $x_{t+1} = f(x_t)$ mit $x \in \mathbb{R}^n$. Im Speziellen kann das System in dieser Form sowohl ein offenes dynamisches System, als auch ein geschlossenes dynamisches System repräsentieren, dessen Eingabe u_t , basierend auf dem aktuellen Zustand, x_t definiert ist. Der Begriff der Stabilität ist eng verbunden mit dem eines Gleichgewichtspunktes eines dynamischen Systems. Ein Zustand x^e ist ein Gleichgewichtspunkt des zeitdiskreten dynamischen Systems f , wenn gilt $f(x^e) = x^e$. Ein Gleichgewichtspunkt ist ein stabiler Punkt der Systemdynamik, da er nicht mehr verlassen wird. Für die Untersuchung der Dynamischen Systeme von besonderem Interesse, ist das Verhalten des Systems in der unmittelbaren und weiteren Umgebung eines Gleichgewichtspunktes. Es existieren verschiedene starke Forderungen an die Stabilität dynamischer Systeme, die sich in verschiedenen Definitionen ausdrücken. Eine dieser Definitionen ist die asymptotische Stabilität eines dynamischen Systems. Ein Gleichgewichtspunkt x^e ist uniform lokal asymptotisch stabil (uniform asymptotic stable(UAS)), wenn eine kompakte Menge $S \subset \mathbb{R}^n$ existiert, so dass für alle initialen Zustände $x_0 \in S$ folgt, dass $\|x_t - x^e\| \rightarrow 0$ wenn $t \rightarrow \infty$. Der Zustand x_t konvergiert somit zu x^e . Wenn $S = \mathbb{R}^n$, wird das System als uniform global asymptotisch stabil (UGAS) bezeichnet. Die Eigenschaft uniform bezeichnet den betrachteten Sonderfall der autonomen Systeme, da die Eigenschaften unabhängig für den Startzeitpunkt des Systems gelten.

Asymptotische Stabilität ist eine sehr strenge Forderung, die in geschlossenen Regelkreisen nur sehr schwierig herzustellen ist. Die hauptsächliche Ursache hierfür sind unvorhersagbare Störungen, Messrauschen oder numerische Fehler bei der Berechnung der Regelgesetze. Sie gibt jedoch einen Eindruck darüber, was die Stabilität eines Gleichgewichtspunktes bedeutet. Ist ein geschlossenes System GAS, so ist garantiert, dass der Gleichgewichtspunkt im asymptotischen Sinn von jedem möglichen Zustand aus erreicht wird. Eine weitere, oft verwendete Definition für Stabilität ist die Stabilität im Sinne von Lyapunov (stable in the sense of Lyapunov (SISL)). Diese besagt, dass ein Gleichgewichtspunkt x^e stabil im Sinne von Lyapunov ist, wenn für alle $\epsilon > 0$ ein $\delta > 0$ existiert, so dass aus $\|x_0 - x^e\| < \delta$ folgt, dass für alle $t > 0$ gilt $\|x_t - x^e\| < \epsilon$. In anderen Worten, eine Systemtrajektorie kann beliebig nahe am Gleichgewichtspunkt gehalten werden, indem man den Startpunkt nahe genug am Gleichgewichtspunkt wählt. Auch diese Forderung für Stabilität ist noch immer zu streng für den Entwurf selbstlernender Regler. Aus diesem Grund folgt nun eine etwas schwächere Definition für die Stabilität eines dynamischen Systems. Diese dient als Grundlage für die selbstlernenden Regler in dieser Arbeit. Ein Gleichgewichtspunkt x^e wird uniform ultimativ beschränkt (UUB) bezeichnet, wenn eine kompakte Menge $S \subset \mathbb{R}^n$ existiert und eine Zahl $N(x_0) \geq 0$, so dass für alle initialen Zustände $x_0 \in S$ folgt, dass $\|x_t - x^e\| < \mu$ für $t > t_0 + N$. Diese Forderung ist gleichbedeutend mit einem Systemverhalten, das garantiert, dass ein System, beginnend in einem Zustand aus S , nach einer begrenzten Anzahl von Zeitschritten in eine beschränkte Nachbarschaft von x^e überführt wird und dort verbleibt. In praktischen Anwendungen kann μ meist klein gewählt werden, so dass die erzeugte Regelgüte zufriedenstellend ist. Der Unterschied von UUB zu SISL ist, dass bei UUB die Schranke μ für einen Startzustand x_0 nicht beliebig verkleinert werden kann, indem er näher am Gleichgewichtspunkt gewählt wird. Uniform bezieht sich wieder auf den Fakt, dass die Forderung für alle Zeitpunkte t_0 gelten muss und ultimativ darauf, dass nach N Zeitschritten das Systemverhalten der Beschränkung unterliegt.

Ein Beispiel: Der DC-Motor mit nichtlinearer Endstufe

Für die Diskussion der Verfahren zum Entwurf der selbstlernenden Regler mit NFQ wird im folgenden das Beispiel der Geschwindigkeitsregelung eines DC-Motors eingeführt. Das Beispiel beruht auf einem realen Prozess, der in Kapitel 5 vorgestellt wird. Betrachtet wird die

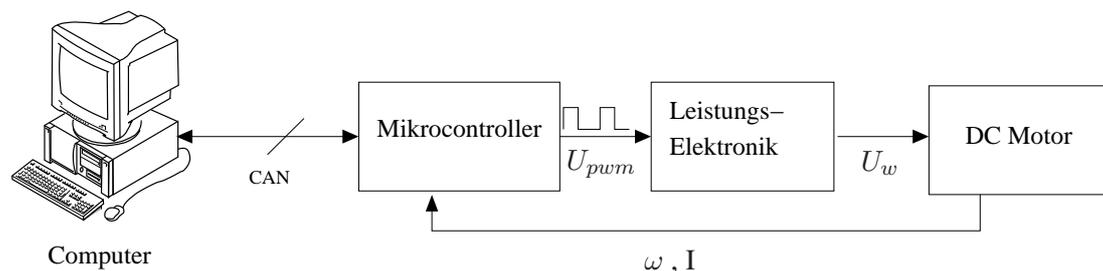


Abbildung 3.3: Schematische Darstellung der dem Beispiel zugrunde liegenden realen Motorregelung, das zur Untersuchung der vorgestellten Verfahren dient.

Geschwindigkeitsregelung eines bürstenbehafteten Gleichstrom-Motors. Dieser besteht aus einem statischen Magnetfeld, in dem sich ein Anker mit Wicklungen dreht. Durch das Anlegen einer Spannung an die Wicklung mit dem Innenwiderstand R_a entsteht ein Drehmoment auf der Motorachse bzw. dem Anker. Diesem entgegengesetzt ist ein Drehmoment, das durch die

Tabelle 3.1: Die für die Simulation verwendeten Parameter des DC-Motors. Die Parameter entsprechen den Werten des Datenblattes eines realen Motors.

R_a	1.23	ohmscher Widerstand der Armaturwicklung	[Ohm]
L	0.00034	Rotor-Induktivität	[H]
k_{emf}	0.03881	emf-Konstante des Motors	[V / s / rad]
k_m	0.03890	Drehmoment-Konstante des Motors	
J	0.0001667	Trägheitsmoment des Rotors (und Last)	[kg m^2 / s^2]
k_f	0.0001	viskose Reibung	[Nms]
τ_{ap}		appliziertes, externes Drehmoment	[Nm]
vcc	24.00	Betriebsspannung des Motors	[V]

Induktions-Spannung bei Drehung des Motors mit Geschwindigkeit ω entsteht. Der Zustandsvektor eines Gleichstrom-Motors lässt sich durch $x = (x_1, x_2)$ mit $x_1 = \omega$ und $x_2 = I$ beschreiben. Neben der aktuellen Drehgeschwindigkeit ω ist somit der Ankerstrom I , der in seiner Ankerwicklung zu einem bestimmten Zeitpunkt fließt, für das Systemverhalten bestimmend. Die Systemdynamik des betrachteten Prozesses ist in Formel 3.3 in Form der zeitkontinuierlichen Dynamik $\dot{x} = f(x, U_w)$ mit einer Eingangsspannung U_w dargestellt.

$$\dot{x}_1 = \frac{k_m}{J} x_2 - \frac{k_f}{J} x_1 - \frac{1}{J} \tau_{ap} \quad (3.3)$$

$$\dot{x}_2 = -\frac{R_a}{L} x_2 - \frac{k_{emf}}{L} x_1 + \frac{1}{L} U_w \quad (3.4)$$

Das in Formel 3.3 und 3.4 definierte System ist ein Beispiel für einen Prozess, der für jede Eingangsspannung U_w einen stabilen Gleichgewichtspunkt in seiner Systemdynamik besitzt. Die bleibende Regelabweichung im ausgeregelten Zustand hängt somit direkt von den dort gewählten Aktionen ab. In diesem Sinne ist das System ein Beispiel für eine ganze Problemklasse, wie sie in der technischen Anwendung häufig anzutreffen sind. Das in Formel 3.3 angegebene Modell ist das Standardmodell eines idealen DC-Motors mit linearer Systemdynamik. Eine technische Umsetzung eines solchen Prozesses ist in Abbildung 3.3 dargestellt. Der Regler selbst wird als Algorithmus auf einem Computer berechnet. Die Anbindung an den physikalischen Prozess wird durch eine auf CAN-Bus basierende Kommunikation mit einem Mikrocontroller hergestellt. Der Regler kommuniziert sein Reglerausgangssignal u_{pwm} an den Mikrocontroller, der über eine Leistungselektronik die für den Motor geeignete Spannung U_w erzeugt. Aus technischer Sicht erzeugt der Mikrocontroller ein pulsweiten-moduliertes (PWM) Signal U_{pwm} , das unter Verwendung einer H-Brücke, basierend auf der Versorgungsspannung vcc , die Stellgröße U_w ergibt, die die Motorgeschwindigkeit beeinflusst. Zudem fungiert der Mikrocontroller als Messglied, das die aktuelle Geschwindigkeit des Motors über die Auswertung eines Inkrementalkoders sowie den aktuell fließenden Strom auf dem Motorkanal durch eine Strommessung auswertet. Diese Messwerte werden dem Regler über den CAN-Bus übermittelt. Für die Geschwindigkeitsregelung des DC-Motors ist somit das Prozessverhalten aus den Eigenschaften der Endstufe, des Mikrocontrollers und des DC-Motors gegeben. Sie beschränkt sich somit nicht mehr alleine auf das vom Prinzip her lineare Prozessverhalten des DC-Motors. Je nach Qualität der Endstufe kann diese durch parasitäre Induktivitäten das PWM Signal stark nichtlinear verändern. An Messungen am realen Gerät wurde für die Simulation ein geeignetes Modell entworfen, um diese Nichtlinearitäten abzubilden. Die Reglerausgangsgröße $u_{pwm} \in [-1, 1]$, die an den Mikrocontroller kommuniziert wird, sei proportional zum erzeug-

ten PWM-Tastverhältnis $U_{pwm} \in [-100\%, 100\%]$. Ein Wert von $u_{pwm} = 1$ entspricht hierbei einem Wert von 100% PWM-Tastverhältnis und somit der vollen positiven Betriebsspannung, die an den Motor angelegt wird. Analog dazu entspricht ein Wert von $u_{pwm} = -1$ der vollen negativen Betriebsspannung (-100% PWM) an den Motor. Die nichtlinearen Eigenschaften der Endstufe werden durch eine nichtlineare Eingangskennlinie (Hammerstein-Modell) realisiert, die die PWM Werte auf eine effektiv am Motor wirkende Spannung U_w abbildet. In Formel 3.5 ist diese nichtlineare Abbildung der Reglerausgangsgröße auf die Stellgröße des Prozesses angegeben.

$$U_w = \min[0, \tanh((|u_{pwm}| - 0.03)3.5)] * SGN(u_{pwm}) * vcc \quad (3.5)$$

Diese nichtlineare Abbildung wurde aus Messwerten des realen Systems mit Versorgungsspannung $vcc = 24V$ erstellt. Sie beinhaltet zusätzlich eine Totzone um den Nullpunkt, bei der die H-Brücken nicht durchschalten. Der zu regelnde Prozess wird durch die Dynamik $\dot{x} = f(x, u_{pwm})$ repräsentiert und besitzt im Gegensatz zum Standardmodell des DC-Motors keine lineare Systemdynamik mehr. Die Erzeugung der benötigten zeitdiskreten Dynamik für die weiteren Betrachtungen wird durch die numerische Integration mit Runge-Kutta-Methoden vierter Ordnung in der Simulation auf einer Zykluszeit von $\Delta_t = 0.01s$ erzeugt.

3.2 Modellierung selbstlernender Regler als Optimierungsprozess mit NFQ

Durch die Modellierung des zu regelnden Prozessverhaltens als sequentielles Entscheidungsproblem auf Basis eines MDP wird der Vorgang des klassischen Reglerentwurfs umformuliert zu einem Optimierungsproblem. Unter Verwendung des NFQ-Verfahrens mit einem explorativen Lernprozess wird somit das selbstständige Erlernen eines optimalen Verhaltens des Reglers bezüglich eines vorgegebenen Optimierungskriteriums ermöglicht. Aus Sicht des Reinforcement-Lernens realisiert der Agent dabei die Entscheidungsfindung, basierend auf dem aktuellen Zustandsvektor des Prozesses und schließt somit den Regelkreis. Der zu regelnde Prozess wird als offenes dynamisches System mit zeitdiskreter Systemdynamik betrachtet. Wie in 3.1 ausgeführt, repräsentiert er aus Sicht des Agenten alle physikalischen und technischen Eigenschaften des klassischen Regelkreises. Für die Modellierung des zu regelnden Prozesses wird eine zu der grundlegenden Definition eines MDP analoge Darstellungsform gewählt. Bei dieser wird ein MDP definiert als 6 Tupel $(\mathcal{T}, \mathcal{X}, U, f, w, c)$. Die Übergangswahrscheinlichkeiten $P_u(x, u)$ werden ersetzt durch die Übergangsfunktion $x_{t+1} = f(x_t, u_t, w_t)$ mit Störgröße w_t . Der Rauschparameter w_t repräsentiert das stochastische Systemverhalten im Sinne einer Zufallsvariablen mit Wahrscheinlichkeitsverteilung $P(w_t|x_t, u_t)$. Sowohl die Funktion f als auch das durch die Wahrscheinlichkeitsverteilung von w definierte Rauschmodell sind für den Agenten nicht bekannt. Die Informationen über diese Zusammenhänge werden vom Agenten durch die Interaktion mit dem Prozess gesammelt. Der Zustandsvektor aus Prozessvariablen muss genügend Informationen enthalten, damit grundsätzlich eine beliebige, im Speziellen auch nichtlineare Funktion f und Wahrscheinlichkeitsverteilung für w existiert, die das zeitdiskrete autonome dynamische System repräsentieren. Durch die gegebene Modellierung des Prozesses als autonomes zeitdiskretes System ist somit implizit die Einhaltung der Markovbedingung erfüllt. Die grundlegende Vorgehensweise bei der Erstellung selbstlernender Regler mit NFQ und die zentralen Eigenschaften des Verfahrens werden vorerst an einer Festwertregelung dis-

kutiert. Durch die Definition einer geeigneten Kostenfunktion sind somit die Voraussetzungen für einen Optimierungsprozess in Form eines MDPs gegeben.

3.2.1 Modellierung des Optimierungskriteriums

Der Zustandsvektor der Prozessgrößen enthält unter anderem auch die betrachtete Regelgröße, die in Verbindung mit dem Sollwert, der vorerst als fest angenommen wird, die zu eliminierende Regelabweichung ergibt. Um diese Regelabweichung zu eliminieren, muss der Agent seine Aktionen in einem sequentiellen Entscheidungsprozess geeignet wählen. Er überführt den Zustand des Systems somit über die Zeit in einen Zustand, bei dem die Regelabweichung möglichst klein wird. Betrachtet man das durch den Regler geschlossene dynamische System, so ist das Regelungsziel definiert durch einen Equilibriumspunkt x^e , der durch den Sollwert der Regelung und die Prozessdynamik definiert wird. Die Forderung an den Optimierungsprozess ergibt sich nun aus den Gütekriterien der Regelung und des UUB-Kriteriums für die Stabilität dynamischer Systeme. Zum Einen besteht die Forderung nach einem stabilen Verhalten des Reglers bezüglich des Equilibriumspunkt x^e . Nach dem für diese Arbeit zugrunde gelegten UUB-Stabilitätskriterium bedeutet dies, dass sich das System, ausgehend von einem Startzustand nach einer beschränkten Anzahl von Schritten, in einem Bereich um x^e befinden muss und diesen Bereich nicht mehr verlassen soll. Zum Anderen besteht von den Gütekriterien der Regelung her die Forderung nach einer möglichst geringen Ausregelzeit, die gleichbedeutend mit der Anzahl der Schritte bis zum Erreichen des Toleranzbereiches ist, und einer möglichst geringen bleibenden Regelabweichung im Beharrungszustand nach dem Erreichen dieses Toleranzbereiches.

Das Reinforcement-Lernen optimiert eine Strategie bezüglich einer definierten Funktion, die auf einzelnen Übergängen der beobachteten Systemdynamik definiert wird. Das erwünschte Regelungsverhalten muss somit durch die Modellierung dieser Kostenfunktion c , abhängig vom aktuellen Zustand und der aktuellen Aktion, definiert werden. In klassischen Benchmarks des Reinforcement-Lernens ist die Kostenfunktion fester Bestandteil des betrachteten Problems. Im regelungstechnischen Kontext ist die Definition der Kostenfunktion als Optimierungskriterium ein zentraler Bestandteil des Entwurfes eines selbstlernenden Reglers. Bei verschiedenen Ansätzen in der Literatur werden Funktionen verwendet, die aus der Systemdynamik des Prozesses abgeleitet werden. Im Extremfall enthalten die direkten Kosten in jedem Übergang bereits selbst einen hohen Informationsgehalt über die optimale Aktion, die in diesem Zustand zu wählen ist. Um der Zielsetzung eines selbst lernenden Reglers mit minimalem Wissen über die Systemdynamik gerecht zu werden, soll an dieser Stelle auf die Verwendung solchen Vorwissens verzichtet werden. Für eine systematische Betrachtung wird die Formulierung der Kostenfunktion in zwei Anteile aufgespalten: $c(x, u) = c^t(x, u) + c^o(x, u)$. Die Funktion $c^t(x, u)$ modelliert das zu erreichende Regelungsziel im ausgeregelten Zustand und das Verhalten während des Vorhandenseins einer Regelabweichung. Mit der Funktion $c^o(x, u)$ können zusätzliche Randbedingungen für die Optimierung vorgegeben werden. Typische Randbedingungen für den praktischen Einsatz von selbstlernenden Reglern sind physikalische oder technische Beschränkungen des Zustandsraumes, kritische Zustände des zu regelnden Prozesses oder Kosten, welche die Anwendung von Aktionen auf den Prozess erzeugen.

Zeitoptimales Verhalten: Die Ziel-Kostenfunktion als Bereichsdefinition

Der Fokus der Untersuchungen liegt auf der Zielkostenfunktion c^t , die das grundlegende Optimierungskriterium der selbstlernenden Regler spezifiziert. Hierzu wird eine zeitoptimale Definition der Kostenfunktion verwendet [74], die auf beliebige regelungstechnische Problemstellungen ohne Kenntnis der zugrunde liegenden Systemgleichungen angewendet werden kann. Wie in Abschnitt 3.1.2 angegeben, soll ein stabiles Verhalten dadurch gekennzeichnet sein, dass es einen vorgegebenen Bereich um den zu erreichenden Equilibriumspunkt nach einer Anzahl von Schritten erreicht und das System anschließend in diesem Bereich hält. Der Begriff 'zeitoptimal' drückt aus, dass die Anzahl der Schritte bis zum Erreichen dieses Zielbereiches möglichst klein sein soll. Mit anderen Worten, über einer gegebenen Systemtrajektorie sollen die kumulierten Kosten dann klein sein, wenn das UUB-Kriterium für Stabilität eingehalten wird und zusätzlich die Anzahl der Schritte bis zum Erreichen des Toleranzbereiches klein sind.

In Form einer zeitoptimalen Kostenfunktion wird das Optimierungskriterium realisiert, indem jeder Übergang außerhalb des Toleranzbereiches der Regelung konstante positive Kosten erzeugt. Zustandsübergänge im Toleranzbereich hingegen erzeugen keine Kosten. Somit erzeugt eine Trajektorie Pfadkosten, die die Anzahl der Schritte außerhalb des Toleranzbereiches repräsentieren. Die minimalen Pfadkosten erzielt eine Strategie, die den Prozess schnellstmöglich in den Toleranzbereich regelt und ihn dort hält. Eine solche Strategie erzeugt somit ein stabiles Verhalten im Sinne des UUB-Kriteriums.

Die Definition des Toleranzbereiches beruht auf der zulässigen Regelabweichung μ und somit ausschließlich auf der Regelgröße im Zustandsvektor des Optimierungsprozesses. Für den im Kontext der selbstlernenden Regler betrachteten Zustandsraum wird ein Zielbereich $\mathcal{X}^+ \subset \mathcal{X}$ definiert. Dieser enthält alle Zustände des Zustandsraumes, in denen die Regelgröße im entsprechenden Toleranzbereich liegt. Übergänge in \mathcal{X}^+ erhalten keine direkten Kosten, außerhalb sind sie auf einem kleinen positiven Wert $\mathcal{C} \in \mathbb{R}, \mathcal{C} > 0$. Die Definition der zeitoptimalen Kostenfunktion lässt sich somit, basierend auf der Bereichsdefinition \mathcal{X}^+ , als,

$$c^t(x, u) = c^t(x) = \begin{cases} 0 & x \in \mathcal{X}^+ \\ \mathcal{C} & \text{sonst} \end{cases} \quad (3.6)$$

definieren. In Abbildung 3.4 ist eine solche Zielkostenfunktion im Zustandsraum eines eindimensionalen Regelungsproblems abgebildet. Bei Prozessen mit mehrdimensionalem Zustandsvektor ist die Beschränkung von \mathcal{X}^+ nur auf der Regelgröße im Zustandsraum festgelegt. Die zulässige Regelabweichung μ wird bei mehrdimensionalen Regelgrößen auf jeder Dimension separat festgelegt.

Ist der Optimierungsprozess erfolgreich, so liegt der Equilibriumspunkt des durch den lernenden Regler geschlossenen dynamischen Systems im Bereich \mathcal{X}^+ . Das geschlossene dynamische System ist jedoch nicht in allen Zuständen dieses Bereichs im Sinne des UUB-Kriteriums stabil. Dieser stabile Bereich muss durch den Regler während des Lernvorganges gefunden werden und spiegelt sich in der gelernten Wertfunktion wieder. Zur besseren Unterscheidbarkeit der beiden Bereiche soll dieser stabile Bereich des geschlossenen dynamischen Systems im Sinne des UUB-Kriteriums als \mathcal{X}^{++} bezeichnet werden. Dies ist der Bereich, in dem das Systemverhalten der Beschränktheit im Sinne des UUB-Kriteriums unterliegt.

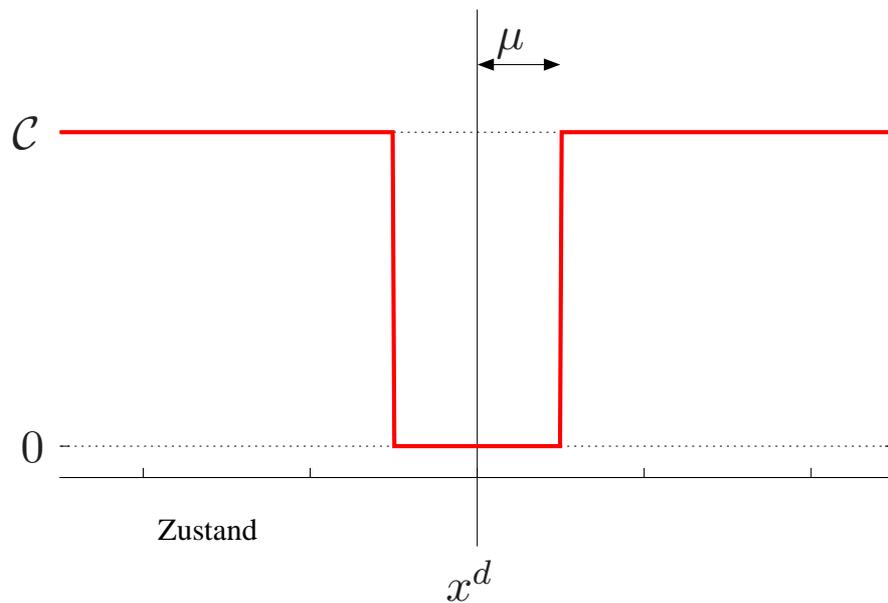


Abbildung 3.4: Darstellung einer Kostendefinition auf einem eindimensionalen Regelungsproblem. Die Kosten außerhalb des Toleranzbereiches um den Sollwert x^d der Regelung sind durch einen konstanten positiven Wert gegeben. Im Toleranzbereich sind sie Null.

Betrachtung am DC-Motor

In Abschnitt 3.1.2 wurde die Geschwindigkeitsregelung des DC-Motors mit nichtlinearer Endstufe als Beispiel eines regelungstechnischen Prozesses eingeführt. Für dieses Beispiel wird nun ein selbst lernender Regler modelliert, der im folgenden zur Evaluation der betrachteten Verfahren dient.

Das Ziel ist die Regelung des durch den DC-Motor und seiner Ansteuerungskomponenten gegebenen dynamischen Systems. Der Zustandsvektor des Optimierungsproblems wird als Vektor der Prozessvariablen $x = (\omega, I)$ definiert. Der vorerst feste Sollwert für die Geschwindigkeitsregelung des DC-Motors sei mit $\omega^d = 421.7 \text{ rad/s}$ gegeben. Es soll ein Regler gelernt werden, der, ausgehend von dem Systemzustand $(0, 0)$, die Regelabweichung zeitoptimal eliminiert. In anderen Worten, die Aufgabe für den lernenden Regler besteht darin, ausgehend von dem definierten Startzustand die Zielgeschwindigkeit in so wenigen Schritten wie möglich zu erreichen und diese zu halten. Für die Definition der Kostenfunktion wird der zulässige Toleranzbereich auf der Regelabweichung mit $\mu = 20 \text{ rad/s}$ festgelegt. Dies entspricht einer maximal zulässigen, absoluten Abweichung von 3.33% der Maximalgeschwindigkeit von 600 rad/s. Für die Vereinfachung der Notation soll die Festlegung der Bereichsdefinition in diesem Fall als $\mathcal{X}^+ = ([\omega^d - \mu \ \omega^d + \mu])$ geschrieben werden. Im konkreten Fall bedeutet dies, dass bei der Definition der Kostenfunktion der Strom des Motors nicht relevant ist. Die Kostenfunktion hängt nur von der Motorgeschwindigkeit ab. Somit ergibt sich im konkreten Fall $\mathcal{X}^+ = ([401.7 \ 441.7])$. Die konstanten Kosten außerhalb des Zielbereiches werden zu $C = 0.01$ gewählt. Die zur Verfügung gestellte Aktionsmenge besteht aus geeignet gewählten diskreten Aktionswerten, die für den festen Sollwert speziell ausgewählt wurden. In diesem Sinne enthält die verwendete Aktionsmenge die für die Zielgeschwindigkeit entsprechende Aktion von 0.3 (30% PWM), eine Aktion 1.0 für die volle Spannung (100% PWM) und -1.0 (-100% PWM) für die volle Spannung in Gegenrichtung. Die verwendete Aktionsmenge

ergibt sich somit zu $U = \{-1.0, 0.3, 1.0\}$. Das NFQ-Verfahren wird in einem explorativen Lernprozess auf das Regelungsproblem angewendet. Er beginnt somit ohne jedes Wissen über den Prozess und wechselt zwischen Interaktionsepisoden und Lerniterationen. Eine Interaktionsepisode hat eine Dauer von 1 Sekunde, in der 100 Übergänge am zu regelnden Prozess gesammelt werden. Die Interaktion wird mit 30% Exploration ($\epsilon = 0.3$) durchgeführt. Für die Repräsentation der Q-Funktion wird ein mehrschichtiges neuronales Netz mit einer (3-10-10-1) Topologie verwendet. An den Eingängen der neuronalen Q-Funktion liegen somit in diesem Fall die zwei Systemvariablen und die Aktion an. Diese Parameter des NFQ-Verfahrens sollen als Standardwerte für das gesamte Kapitel dienen. Was sich bei der schrittweisen Erweiterung der Problembetrachtung ändert, ist die verwendete Zustandsinformation und somit die Dimension der Eingabeschicht des neuronalen Netzes.

Zugunsten der Darstellung, wird der Prozess der Regelung in diesem Beispiel vorerst durch den festen Sollwert vereinfacht. Des weiteren werden dem Agenten durch die für die Zielgeschwindigkeit benötigte Aktion sehr spezifische Informationen über den Prozess zur Verfügung gestellt. Die in dieser Arbeit entwickelten Methoden tragen schrittweise dazu bei, das Lernen von effizienten Reglern auch ohne das Vorliegen von spezifischer Information über den Prozess zu ermöglichen. Das gewählte Beispiel ist in dieser Form somit nicht repräsentativ für die Zielsetzung der Arbeit. Es soll vorerst jedoch dazu dienen, die auftretenden Probleme und deren Lösungen miteinander vergleichbar zu machen.

3.2.2 Modellierung des Lernprozesses

Die Regelung ist ein andauernder Prozess ohne festen Endzeitpunkt. Während der gesamten Ausführung stellt der Regler sicher, dass das Verhalten des geschlossenen dynamischen System stabil um den gegebenen Gleichgewichtspunkt bleibt und bei Auftreten von Störungen das System wieder in einen stabilen Zustand überführt wird. Für das Reinforcement-Lernen ergibt sich somit ein Problemtyp mit unendlichem Horizont. Dies bedarf typischerweise einer Diskontierung der Kosten, was in diesem Falle aber unerwünscht ist, da das Optimierungskriterium dadurch verfälscht wird. Die Problemstellung der zeitoptimalen Regelung soll deshalb als Problem ohne Diskontierung, in Anlehnung an die stochastischen kürzesten Pfad-Probleme betrachtet werden. Im Unterschied zu den SKP-Problemen gibt es bei dieser Betrachtung jedoch keinen Terminalzustand x^+ , der mit keiner Aktion mehr verlassen werden kann. Vielmehr wird dieser ersetzt durch eine Menge \mathcal{X}^+ , in welcher die Kosten für einen Übergang Null sind [77]. Eine Strategie, die \mathcal{X}^+ erreicht und den Zustand des Systems in diesem Bereich hält, erzeugt endliche Pfadkosten. Eine Strategie ist somit dann erfüllend, wenn sie \mathcal{X}^+ erreicht und nicht mehr verlässt. Mit der Bereichsdefinition der Kosten sind für alle Zustände außerhalb von \mathcal{X}^+ die direkten Kosten auf einem kleinen, positiven Wert. Somit erzeugt jede Strategie, die nicht erfüllend ist, unendliche Kosten. Das alleinige vorübergehende Erreichen von \mathcal{X}^+ , erzeugt jedoch keine endlichen Pfadkosten. Zusätzlich werden analog zu den SKP-Problemen Terminalzustände zugelassen, in die das System übergehen kann und die mit keiner Aktion mehr verlassen werden können. Diese Terminalzustände beenden eine Trajektorie und stellen zumeist einen Fehlerfall dar, der über die Kostenfunktion c^o behandelt wird. Auf diese Weise kann das Verlassen eines sicheren Bereiches der Systemdynamik oder des Zustandsraumes selbst bestraft werden.

Der Wegfall des eindeutigen Terminalzustandes x^+ , für den die Pfadkosten in einem SKP-Problem bekannt sind, ist ein zentrales Problem bei der Anwendung von NFQ auf dem Problemtyp der zeitoptimalen Regelung. Bei SKP-Problemen basieren die Wertfunktionen, hier

im Speziellen die Q-Funktion, auf diesem bekannten Wert. Wird ein Übergang in den Terminalzustand beobachtet, so wird die Wertfunktion für diesen Übergang direkt im Approximator gesetzt. Durch die iterativen Eigenschaften des Lernverfahrens ergeben sich die Werte der nicht terminalen Zustände somit relativ zu diesem Wert. Bei Anwendung des NFQ-Verfahrens auf den Problemtyp der zeitoptimalen Regelung entsteht daraus das Problem des Wegdriftens der Q-Funktion. Dies ist bedingt durch die zufällige Initialisierung des neuronalen Q-Netzes zu Beginn des Lernverfahrens, das dadurch beliebige Werte im gesamten Zustandsraum annimmt. Die Aktualisierung der Q-Funktion mit NFQ basiert auf der Kostenfunktion und dieser zufällig initialisierten Q-Funktion. Die lernbare Q-Funktion kann somit keine kleineren Werte annehmen als den kleinsten Wert im initialen Netz. Das Problem wird weiter verschärft durch den explorativen Lernprozess. Bei diesem ist nicht zu erwarten, dass frühe Interaktions-episoden auf einer erfüllenden Strategie basieren. Somit werden vorwiegend Übergänge mit positiven Kosten gesammelt. Die Aktualisierung, basierend auf der gesammelten Information, lässt die Q-Funktion deshalb noch stärker anwachsen. Dies verhindert, dass die Informationen aus Trajektorien durch \mathcal{X}^+ effektiv ausgenutzt werden. Die Konsequenzen daraus sind, dass die Q-Funktion einen nicht vorhersagbaren Wertebereich annimmt bzw. durch zusätzliche Approximationsfehler alle Werte unbeschränkt wachsen. In der praktischen Anwendung stößt die Q-Funktion nach kurzer Zeit für alle Zustände gegen die Begrenzung Q_{max}^{bound} der Q-Funktion und nimmt diesen Wert für den ganzen Zustandsraum an. Die Q-Funktion degeneriert somit und die durch die gierige Auswertung der Q-Funktion bestimmte Strategie wird rein zufällig.

Betrachtung am DC-Motor

Das Wegdriften der Q-Funktion wird am DC-Motor durch die Anwendung des bisher beschriebenen Lernversuches direkt deutlich. Zur Bewertung des Verhaltens der Q-Funktion, wird die Menge der Trainingsmuster \mathcal{P} in einer Lerniteration betrachtet. Auf dieser Menge wird der minimale Zielwert $\hat{Q}_{min}^{\mathcal{P}}$ und der maximale Zielwert $\hat{Q}_{max}^{\mathcal{P}}$ für das überwachte Lernen der Approximation der Q-Funktion in jeder Lerniteration bestimmt. Der Verlauf dieser Werte ist in Abbildung 3.5 über den explorativen Lernprozess der Geschwindigkeitsregelung auf eine konstante Geschwindigkeit am DC-Motor über die ersten 150 Trainingsepisoden aufgetragen. Deutlich ist das Wegdriften der Werte in der Q-Funktion zu sehen. Der Agent lernt trotz des

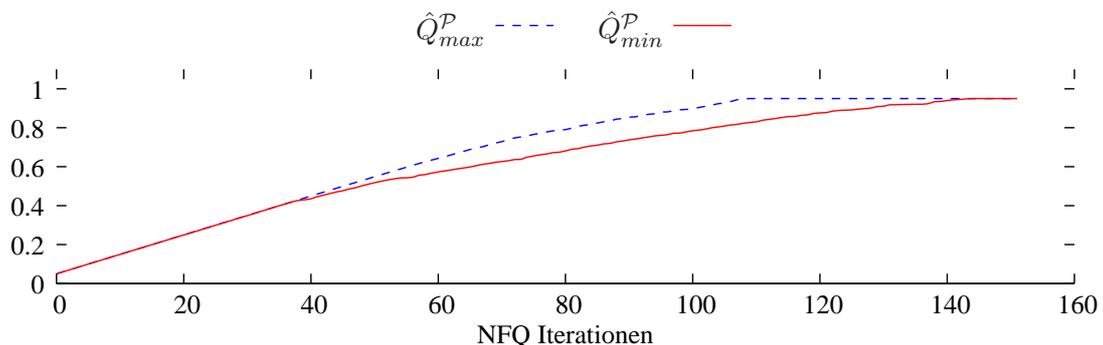


Abbildung 3.5: Der Verlauf der minimalen und maximalen Q-Werte in der Trainingsdatenmenge während des Lernprozesses am simulierten DC-Motor. Ohne einen festgelegten Terminalzustand, in dem die Kosten feststehen, driften die Werte stark ab. Dies hat zur Folge, dass nur sehr schlecht gelernt werden kann und bei Fortsetzung des Lernverlaufs beide Werte gegen den maximal erlaubten Q Wert im Approximator streben. Somit degeneriert die Q-Funktion und die Strategie wird rein zufällig.

Wegdriftens eine geeignete Strategie für das definierte Problem. Nach 60 Trainingsepisoden wird die erste erfüllende Strategie gefunden. Der theoretisch optimale Wert von $\bar{N} = 16$ Schritten bis zum Ziel wird erst nach 65 Trainingsepisoden erreicht. Es zeigt sich somit ein relativ langsamer Lernprozess, der für dieses einfache Problem die optimale Strategie findet. Bei weiterem Verlauf des Lernprozesses verliert er diese jedoch und erzeugt keine erfüllenden Strategien mehr.

3.2.3 Das Konditionieren der Q-Funktion

Eine Lösung des Problems der wegdriftenden Q-Werten bei dem Problemtyp der zeitoptimalen Regelung ist das Konditionieren der Q-Funktion mit künstlichen Trainingsmustern. In jeder Iteration des NFQ-Verfahrens wird die Datenbasis der Approximation der Q-Funktion \mathcal{P} mit zusätzlichen Trainingsmustern erweitert, für die der Wert der Q-Funktion bekannt ist. Durch das überwachte Lernen der Q-Funktion mit dieser erweiterten Trainingsdatenmenge geht die Information der künstlich erzeugten Muster in die Approximation der Q-Funktion ein. In dieser Form ist es möglich, die Approximation an diesen Stellen auf korrekte Werte zu konditionieren. Die Information verbreitet sich durch den iterativen Lernvorgang, ausgehend von diesen Mustern in der gesamten Q-Funktion. Diese Trainingsmuster repräsentieren ein gewisses Vorwissen, das über den Prozess und seine Regelung vorhanden ist. Grundsätzlich können dies alle Zustands-Aktionspaare aus der stabilen Untermenge \mathcal{X}^{++} (vgl. 3.2.1) des geschlossenen Systems sein, die eine Strategie repräsentieren, welche das System in der stabilen Menge hält. Diese besitzen in der praktischen Realisierung des Verfahrens den Q-Wert Q_{min}^{bound} . Für die praktische Anwendung des Verfahrens, ist die Bestimmung dieser Zustands-Aktions-Paare meist nicht einfach. Deshalb wird zumeist der Gleichgewichtspunkt des zu entwerfenden geschlossenen Systems selbst als künstliches Trainingsmuster benutzt. Die Information, die für die Konditionierung der Q-Funktion vorliegen muss, ist somit die Ausprägung des Gleichgewichtspunktes in allen Prozessvariablen und die Aktion, die diesen Zustand bezüglich der Prozessdynamik stabil macht. Dies ist für viele Systeme ein sehr hohes Maß an Vorwissen, das die Kenntnis und Analyse der Systemgleichungen notwendig machen kann und zusätzlich Wissen über die Strategie in diesem Punkt erfordert.

Das beschriebene Vorgehen der Konditionierung der Q-Funktion wurde von Riedmiller als 'Hint to Goal'-Heuristik bezeichnet [79], was auf einen weiteren Vorteil dieses Verfahren hinweist. Dem Agenten wird nicht nur der Q-Wert an einer Stelle der Q-Funktion mitgeteilt, sondern auch direkt die Information über die benötigte Strategie in diesem Punkt. Durch die Approximation der Q-Funktion mit neuronalen Netzen findet im Nahbereich um diesen Punkt eine Generalisierung statt. Sind erst wenige Übergänge am System gesammelt, erscheint dieser Bereich für den Agenten attraktiv und zieht ihn während der Explorationsepisoden an, womit sich die Lerngeschwindigkeit erhöhen lässt.

Für die praktische Umsetzung dieser Methode stellt sich die Frage, wie viele solcher künstlicher Werte benötigt werden, um das Training des neuronalen Netzes geeignet zu beeinflussen. Dies ist ein weiterer Parameter, der den Lernprozess stark beeinflussen kann. In der Praxis hat sich ein Wert von 1% der in der Datenmenge vorhandenen Übergänge als geeignet erwiesen. Wie das nachstehende Beispiel zeigt, ist es für allgemeine regelungstechnische Probleme nicht immer einfach oder überhaupt möglich, solche Punkte zu finden.

Betrachtung am DC-Motor

Für die Anwendung der Konditionierung der Q-Funktion am Beispiel der Geschwindigkeitsregelung des DC-Motors muss der Gleichgewichtspunkt des zu entwerfenden Systems bestimmt werden. Dies ist somit zum einen der Zustand, in dem die Regelgröße ω den Sollwert $\omega_d = 421.72$ im ausgeglichenen Zustand annimmt. Zum anderen muss der Strom bestimmt werden, der im ausgeglichenen Zustand bei dieser Geschwindigkeit fließt. Dieser muss durch die Analyse der Systemgleichungen oder durch Experimente bestimmt werden. Des Weiteren muss die Aktion vorgegeben werden, die im ausgeglichenen Zustand die Sollgeschwindigkeit beibehält. Das künstliche Trainingsmuster für die Konditionierung der Q-Funktion ergibt sich in diesem Beispiel somit zu $p = ((x, u), \hat{Q}_{x,u}) = (((421.72, 1.0841), 0.3), Q_{min}^{bound})$. In Abbildung 3.5 ist der Verlauf der minimalen und maximalen Q-Werte über 150 Trainingsiterationen mit NFQ unter Verwendung der Konditionierung der Q-Funktion aufgetragen. Die Anzahl der in die Trainingsmenge eingefügten künstlichen Trainingsmuster beträgt in jeder Iteration 1% der bis dort hin gesammelten Übergänge. Durch die Verwendung der künstlichen Muster bleiben die Q-Werte in der Trainingsmenge über die Lerniterationen in einem beschränkten Bereich. Das Wegdriften der Q-Werte wird effektiv verhindert und die minimalen Q-Werte sind nahe an dem vorgegebenen Wert von $Q_{min}^{bound} = 0.05$. Nach nur 25 Iterationen hat das NFQ-Verfahren die erste erfüllende Strategie gefunden, die die Geschwindigkeit des Motors in den Toleranzbereich regelt und dort hält. Die optimalen Pfadkosten von $\bar{N} = 16.00$ werden nach nur 40 Iterationen des Lernverfahrens erreicht. Dies ist eine deutliche Verbesserung des Lernprozesses, der sich durch die Anwendung des Verfahrens ergibt. Um diese Methode am Beispiel des DC-Motors

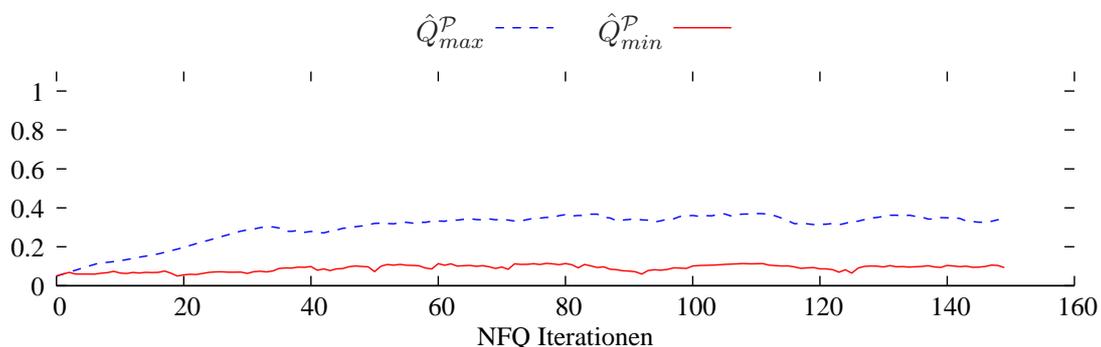


Abbildung 3.6: Der Verlauf der minimalen und maximalen Q-Werte in der Trainingsdatenmenge, während des Trainings am simulierten DC-Motor. Um das Wegdriften der Q-Werte zu vermeiden, wurde im künstlich berechneten Gleichgewichtspunkt die 'Hint to Goal'-Heuristik angewandt. Der Lernverlauf wird dadurch stabiler und die Q-Werte bleiben im erwarteten Bereich. Ein Degenerieren der Q-Funktion wird verhindert.

anzuwenden, musste jedoch der stabile Gleichgewichtspunkt für diesen Zustand aus den Systemgleichungen berechnet werden. Für allgemeine regelungstechnische Problemstellungen ist dies jedoch oft nicht möglich und bedarf eines fundierten Wissens über die Systemgleichungen.

3.3 Korrektur des systematischen Fehlers in der Q-Funktion

In Abschnitt 3.2.3 wurde ein bestehendes Verfahren vorgestellt, um das Wegdriften der Q-Werte beim Problemtyp der zeitoptimalen Regelung, zu verhindern. Die Verwendung dieses Verfahrens stabilisiert den Lernprozess und verhindert die Degeneration der Q-Funktion im

Verlauf des Lernvorganges. Für die Anwendung müssen jedoch sehr spezifische Informationen über den zu regelnden Prozess vorliegen. Die zur Verfügung gestellte Information entspricht der vollständigen Kenntnis des Gleichgewichtspunktes des zu entwerfenden geschlossenen dynamischen Systems.

Im folgenden wird ein Verfahren entwickelt, das ohne die zusätzliche Information über die Systemdynamik, ausschließlich basierend auf der Kostenfunktion der zeitoptimalen Regelung, dieses Wegdriften der Q-Werte effizient verhindert. Hierzu wird der systematische Fehler näher untersucht, der in jeder Iteration des NFQ zum Wegdriften der Q-Funktion führt. Basierend darauf wird ein geeignetes Vorgehen entwickelt, das diesem systematischen Fehler entgegenwirkt.

3.3.1 Herleitung des Verfahrens

Ausgangspunkt für die Herleitung des Verfahrens ist die Betrachtung der Eigenschaften einer optimalen Q-Funktion unter dem Problemtyp der zeitoptimalen Regelung mit NFQ. Der kleinste darstellbare Wert der Approximation der Q-Funktion sei vorerst festgelegt auf 0. Das direkte Kostensignal des MDP, sei $c(x, u) = C > 0$ außerhalb \mathcal{X}^+ und $c(x, u) = 0$ innerhalb \mathcal{X}^+ , wie bei der zeitoptimalen Regelung mit der Bereichsdefinition festgelegt. Unter diesen Voraussetzungen kann die Q-Funktion nur an den Stellen den Wert 0 annehmen, für die der Zustand selbst in \mathcal{X}^+ liegt und die Aktion in einen Folgezustand führt, von dem aus eine erfüllende Strategie existiert, die das System in \mathcal{X}^+ hält. Die optimale Q-Funktion hat somit für diejenigen Zustands-Aktions-Paare den Wert 0, die zum stabilen Bereich des geschlossenen Systems im Sinne des UUB-Kriteriums gehören. Für alle anderen Zustands-Aktions-Paare ist der Q-Wert größer 0. In diesem Sinne soll die optimale Q-Funktion als absolute Q-Funktion gelten, da ihre Funktionswerte in keiner Form gegen die Pfadkostenfunktion verschoben sind. In einer Iteration des NFQ-Verfahrens liegt die aktuelle Schätzung der Q-Funktion in einem Schritt k in Form eines neuronalen Netzes Q_k vor. Basierend auf dieser Information wird die Trainingsdatenmenge für die Approximation der aktualisierten Q-Funktion in Schritt $k+1$ berechnet. Startet das Verfahren mit einem zufällig initialisiertem Netz Q_0 , so besitzt es beliebige Werte über den ganzen Zustands-Aktions-Raum. Durch die iterativen Eigenschaften der Aktualisierung mit der Q-Lernregel nehmen Übergänge im stabilen Bereich des geschlossenen dynamischen Systems den Pfadkostenwert ihres Nachfolgezustandes an. Übergänge außerhalb dieses stabilen Bereiches nehmen einen um C größeren Wert als die Pfadkosten ihres Folgezustandes an. Ist die Dynamik des Prozesses in der Datenbasis des Verfahrens ausreichend repräsentiert, so bildet sich durch die wiederholte Anwendung des NFQ-Verfahrens eine Q-Funktion heraus, die relativ zum kleinsten Wert der Initialisierung der Q-Funktion im stabilen Bereich ist. Es tritt somit eine systematische Verschiebung der Q-Funktion bzw. der geschätzten Pfadkosten auf, die relativ zur Initialisierung der Q-Funktion ist.

Durch den explorativen Lernprozess wird das Problem weiter verschärft. Da zu Beginn des Lernprozesses die Dynamik des Prozesses noch nicht ausreichend in der Datenbasis repräsentiert ist, kommt es auch zu einem Anwachsen der geschätzten Pfadkosten im stabilen Bereich des Zustands-Aktions-Raumes. Die Quantifizierung des systematischen Fehlers basiert auf dem kleinsten Zielwert, der in der Datenbasis der Trainingsmenge \mathcal{P} für die Q-Funktion vorliegt. Bei einer gegebenen Trainingsdatenmenge \mathcal{P} , mit Einträgen der Form (x^p, u^p, \hat{Q}^p) , $p = 1, \dots, \#\mathcal{P}$, wird dieser kleinste Zielwert für die Approximation der Q-Funktion als

$$\hat{Q}_{min}^{\mathcal{P}} = \min_{p \in \mathcal{P}} \hat{Q}^p$$

bezeichnet. Die vorliegende Trainingsdatenmenge \mathcal{P} wurde durch die Anwendung der Q-Lernregel, basierend auf der Datenmenge der gesammelten Übergänge, \mathcal{D} , und der aktuellen Schätzung der Q-Funktion erzeugt. Nach der Berechnung von \mathcal{P} kann $\hat{Q}_{min}^{\mathcal{P}}$ als eine Schätzung für einen konstanten Beitrag betrachtet werden, der in die Berechnung aller Zielwerte mit eingegangen ist, ohne selbst Information zu tragen. Um diesen systematischen Fehler, der bei der Anwendung der Lernregel auf der Datenbasis entstanden ist, zu korrigieren kann dieser Wert von allen Zielwerten in \mathcal{P} abgezogen werden.

In der praktischen Anwendung wird von einer beschränkten Q-Funktion im Wertebereich $[Q_{min}^{bound}, Q_{max}^{bound}]$ ausgegangen. In diesem Fall wird nicht der Wert $\hat{Q}_{min}^{\mathcal{P}}$ direkt zur Korrektur der Zielwerte verwendet, sondern vielmehr die Differenz zu der unteren Beschränkung der Q-Funktion $\hat{Q}_{min}^{\mathcal{P}} - Q_{min}^{bound}$ von den Zielwerten abgezogen. Das entwickelte Verfahren wird aufgrund der Korrektur mit dem kleinsten Q-Wertes in der Trainingsdatenmenge im folgenden auch als Q_{min} -Verfahren bezeichnet.

3.3.2 Arbeitsweise und Leistungsfähigkeit

Im Vergleich zu der 'Hint to Goal'-Heuristik arbeitet das vorgestellte Verfahren zur Korrektur des systematischen Fehlers in der Q-Funktion global auf der gesamten Datenbasis. Die Korrekturen müssen sich nicht durch den iterativen Prozess des Lernvorganges und über den Approximator im Zustandsraum ausbreiten, sondern können sich direkt auf alle vorhandenen Trainingsmuster auswirken. Da für ihre Anwendung kein Wissen über die Systemdynamik notwendig ist, kann sie auf beliebige Prozesse direkt angewendet werden, ohne weitere Parameter für den Lernprozess einzuführen. Für die Anwendung im regelungstechnischen Kontext kann nur durch die Angabe des Sollwertes der Regelgröße ein stabiler Bereich des geschlossenen dynamischen Systems gefunden werden. Die zulässigen Werte für die Prozessvariablen und Aktionen für diesen stabilen Bereich müssen nicht vorgegeben werden sondern werden während des Lernprozesses durch Interaktion mit dem System gefunden.

Die in der Q-Funktion vorhandenen Abstufungen verschiedener Zustands-Aktions-Paare bleiben durch die Korrektur vollständig erhalten. Auf diese Weise unterstützt die Arbeitsweise des Verfahrens den explorativen Lernprozess. Dies ist zum einen der Fall, wenn noch keine Trajektorie den Bereich \mathcal{X}^+ betreten hat. In diesem Fall bewirkt die Korrektur, dass die Q-Funktion nicht gegen den maximalen Wert im Approximator strebt und somit keinerlei Information mehr trägt. Zum anderen erzeugen Trajektorien, die den Bereich \mathcal{X}^+ kurzfristig betreten, einen für den Agenten anziehenden Bereich, der die Exploration des Zustandsraumes in interessanten Bereichen unterstützt. Sind die Anforderungen an eine absolute Q-Funktion erfüllt, ist also der Wert $\hat{Q}_{min}^{\mathcal{P}} = Q_{min}^{bound}$ so wird keine Korrektur durchgeführt und das NFQ-Verfahren liegt in seiner originalen Form vor. Das Verfahren dient als eine Art Überwachung des Lernvorganges und greift ein, wenn ein systematischer Fehler vorliegt. In diesem Sinne dient das Verfahren ebenfalls zur Korrektur von Approximationsfehlern in der Q-Funktion, die eine Verschiebung der Q-Werte bedingen können.

Eine Eigenschaft des Verfahrens ist, dass es sich nur für Problemstellungen eignet, deren geschlossenes Systemverhalten einen zusammenhängenden stabilen Bereich besitzt. Sind mehrere solcher Bereiche vorhanden, kann im Allgemeinen der systematische Fehler global nicht mehr bestimmt werden. Die 'Hint to goal'-Heuristik ist hier im Vorteil, da bei Kenntnis der Gleichgewichtspunkte die Trainingsmuster an mehreren Stellen im Zustands-Aktionsraum erzeugt werden können.

Betrachtung am DC-Motor

In Abbildung 3.7 ist der Verlauf der minimalen und maximalen Q-Werte über 150 Trainingsepisoden bei Verwendung des Q_{min} -Verfahrens dargestellt. Die einzigen Informationen, die für die Anwendung des Q_{min} -Verfahrens vorliegen müssen, sind der Sollwert $\omega_d = 421.72$, sowie der Toleranzbereich der Regelung $\mathcal{X}^+ = ([401.7441.7])$ für die Bereichsdefinition der Kosten. Der Strom im stabilen Zustand und die dort benötigte Aktion wird explizit nicht benötigt. Ohne dass zusätzliche Informationen über die Systemdynamik vorliegen müssen, bleiben die Werte in einem beschränkten Bereich und das Wegdriften wird verhindert. Nach nur 20 Lernschritten hat der Regler gelernt, die Zielgeschwindigkeit zu erreichen. Die optimalen Pfadkosten von 16.00 werden ebenfalls nach nur 20 Iterationen des Lernverfahrens erreicht. Somit ist die vorgestellte Heuristik beim Erreichen des optimalen Wertes für den Regler deutlich schneller als die zuvor vorgestellte 'Hint to Goal'-Heuristik. Wie ebenfalls aus der Abbildung hervor geht,

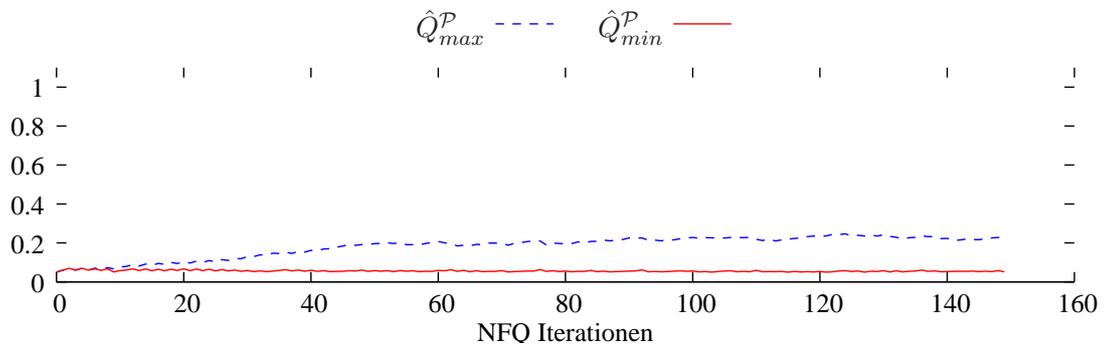


Abbildung 3.7: Der Verlauf der minimalen und maximalen Q-Werte in der Trainingsdatenmenge während des Lernprozesses am simulierten DC-Motor. Um das Wegdriften der Q-Werte zu vermeiden, wurde das Q_{min} -Verfahren angewandt. Das Lernen funktioniert sehr stabil und die Q-Werte bleiben im erwarteten Bereich.

wird der kleinste Q-Wert in der Trainingsdatenmenge \hat{Q}_{min}^P über den Lernverlauf sehr nahe bei dem geforderten Wert von Q_{min}^{bound} gehalten. In Abbildung 3.8 sind die Ergebnisse von drei

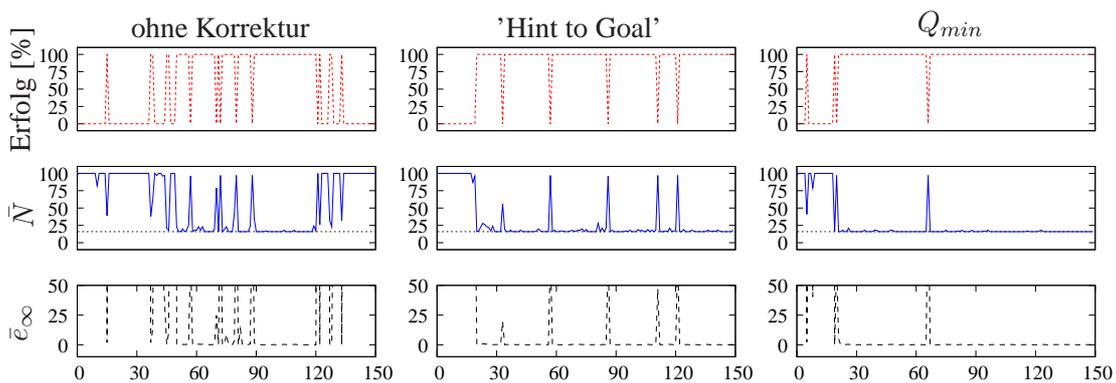


Abbildung 3.8: Der direkte Vergleich des Lernverlaufes über eine Anzahl von 150 Iterationen mit NFQ. Dargestellt sind die Ergebnisse eines Lernversuches ohne Korrektur, eines Lernversuches mit Konditionierung der Q-Funktion durch die 'Hint to Goal'-Heuristik und eines Lernversuches mit Korrektur des systematischen Fehlers mit dem Q_{min} -Verfahren.

Lernversuchen über 150 Iterationen des NFQ-Verfahrens im direkten Vergleich dargestellt. Je-

weils eine Spalte enthält die Ergebnisse eines Lernversuches. Diese wurden ohne Korrektur der Q-Funktion, mit der 'Hint to Goal'-Heuristik und mit Korrektur des systematischen Fehlers unter dem Q_{min} -Verfahren, durchgeführt. In der ersten Zeile ist die Erfolgsrate der Regler Aufgetragen. Diese gibt an, ob der Regler die aktuelle Testtrajektorie in den Toleranzbereich der Regelung überführt hat (100%) oder nicht (0%). Die zweite Zeile gibt das Gütekriterium der Zeitoptimalität wieder, bzw. die Anzahl der Zeitschritte, in denen sich die Regelgröße außerhalb des Toleranzbereiches befunden hat. Die dritte Zeile enthält den Verlauf des Gütekriteriums der bleibenden Regelabweichung über die Iterationen des Lernverfahrens. Ohne Korrektur ist die vollständige Degeneration der Q-Funktion nach c.a. 140 Iterationen zu sehen. Der Lernverlauf mit der 'Hint to Goal'-Heuristik degeneriert nicht, bricht über den Lernverlauf jedoch häufig ein. Die Korrektur des systematischen Fehlers in der Q-Funktion ergibt einen wesentlich stabileren Lernverlauf.

3.4 Differenzierbare Kostenfunktionen

Im Folgenden wird die Formulierung der zeitoptimalen Kostenfunktion im Kontext der selbstlernenden Regler noch einmal genauer betrachtet. Das Ziel der Betrachtung ist die Verfeinerung der Kostenfunktion, um zeitoptimale Regler zu lernen, die eine sehr hohe Genauigkeit im ausgeregelten Zustand aufweisen. Dies ist eine zentrale Forderung im praktischen Einsatz der Regler, welche mit der vorgestellten Bereichsdefinition der Kosten nicht zu erfüllen ist.

3.4.1 Probleme und Leistungsfähigkeit der Bereichsdefinition

Die Formulierung der Kostenfunktion mit Hilfe der Bereichsdefinition erzeugt im Sinne des UUB-Kriteriums zeitoptimale und stabile Regler. Durch die einfache Definition der Kostenfunktion ist sie ohne Vorwissen für verschiedenste Prozesse direkt anwendbar. Da im einfachsten Fall nur die Regelabweichung oder der Sollwert in die Definition der Kosten eingeht, kann auch für komplexe Prozesse mit mehrdimensionalen Größen eine einfache zeitoptimale Kostenfunktion angegeben werden. Ein Problem der Kostenfunktion ist jedoch, dass sie ausschließlich die Stabilität in einem Teilbereich von \mathcal{X}^+ bewirkt. Das Verhalten des geschlossenen Systems wird auf einen Verbleib in diesem Bereich optimiert. Innerhalb dieses Bereiches ist es jedoch nicht weiter spezifiziert. Ein weiterer Nachteil wird durch die Form der sich ergebenden Q-Funktion bedingt. Durch die harten Grenzen der Kostenfunktion werden diese über die Systemdynamik in die Q-Funktion überführt. Die dadurch bedingten Stufen in der Q-Funktion lassen sich nur schlecht in einem neuronalen Netz repräsentieren.

Die bei der Verwendung in regelungstechnischen Problemen mit kontinuierlichem Zustandsraum auftretenden Probleme werden im Folgenden an einem einfachen Beispiel diskutiert.

Durch die Systemgleichung $x_{t+1} = x_t + u_t$ sei ein einfacher zeitdiskreter Prozess mit eindimensionalem Zustand und eindimensionaler Aktion gegeben. Die Regelgröße ist somit gegeben durch x , die auf eine Sollgröße von 0 geregelt werden soll. Dazu wird ein Agent betrachtet, der eine Aktionsmenge mit diskreten Aktionen im Bereich zwischen -0.1 und 0.1 besitzt. Die Aktionsmenge ergibt sich somit zu $U = \{-0.1, \dots, 0, \dots, 0.1\}$. Die Kostenfunktion für die zeitoptimale Regelung sei durch die Bereichsdefinition mit Toleranz μ bzw. Breite des Toleranzbereiches von 2μ gegeben. In Abbildung 3.9 sind exemplarisch drei verschiedene Trajektorien dieses einfachen Systems unter einer optimalen Strategie schematisch dargestellt. Für die dargestellten Bereiche besteht diese optimale Strategie aus der n-maligen Anwendung der Aktion 0.1 zum Erreichen des Zielbereiches und anschließender Anwendung der Aktion 0, um in

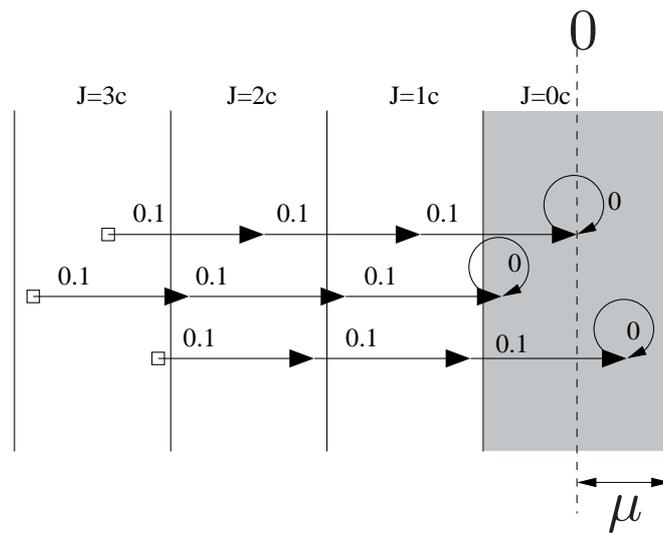


Abbildung 3.9: Drei Trajektorien des Einfachen Systems $x_{t+1} = x_t + u_t$ zur Überführung des Zustandes in den grau unterlegten Bereich unter einer optimalen Strategie. Die Pfeile stellen die Übergänge des Systems dar, die mit der gewählten Aktion gekennzeichnet sind. Mit der Bereichsdefinition der Kosten, mit Zielbereich der Breite 2μ , ergeben sich Regionen gleicher optimaler Pfadkosten in der Wertfunktion.

diesem zu verbleiben. Es ergeben sich ganze Bereiche gleicher Kosten, die, abhängig von den zur Verfügung stehenden Aktionen, nahezu beliebige optimale Strategien ermöglichen. Diese sind vor allem im Toleranzbereich störend, da dort die Kosten für Übergänge 0 sind und die Genauigkeit der Regelung verschlechtern. In anderen Worten ausgedrückt, im Toleranzbereich kann der Agent beliebige Aktionen anwenden, solange dieser Bereich nicht verlassen wird, ohne durch die Kostenfunktion bestraft zu werden. In Abbildung 3.10 ist die analytisch bestimmte optimale Kostenfunktion und Q-Funktion für das gegebene Beispiel mit einer hohen Auflösung von diskreten Aktionen dargestellt. Der beschriebene Effekt bewirkt ausgeprägte Treppenstufen in den Wertfunktionen, die sich mit Hilfe neuronaler Netze nur schlecht lernen lassen. Für die Verwendung mit NFQ bedeutet dies, dass die exakte Q-Funktion des optimalen Systems nur angenähert werden kann. Die sich ergebende Q-Funktion wird als Glättung über die Treppenfunktion repräsentiert, die die Generalisierung im neuronalen Netz bewirkt. Die Verkleinerung des Toleranzbereiches bzw. von \mathcal{X}^+ durch die Reduzierung von μ verbessert die Genauigkeit der Regelung und schwächt die Ausprägung der Treppenstufen ab. Für die praktische Anwendung stellt dies jedoch keine Lösung dar. Bei Verkleinerung von μ bzw. des Toleranzbereiches wird die Wahrscheinlichkeit für das Betreten eines Zustandes in diesem Toleranzbereich immer kleiner. Speziell beim explorativen Lernprozess ohne Vorwissen steigt die Anzahl der benötigten Interaktionsepisoden dadurch stark an. Im Grenzfall $\mu \rightarrow 0$ geht die Wahrscheinlichkeit für das Betreten des Bereiches gegen Null. Dies ist ein bekanntes Ergebnis aus dem Bereich der evolutionären Algorithmen. Wird bei diesen die Fitnessfunktion auf dem gesamten Parameterbereich konstant gehalten und nur in einem Punkt auf einen kleineren Wert gesetzt (in Form einer Golfplatzfunktion), so geht die Wahrscheinlichkeit für das Auffinden dieses Punktes gegen 0 .

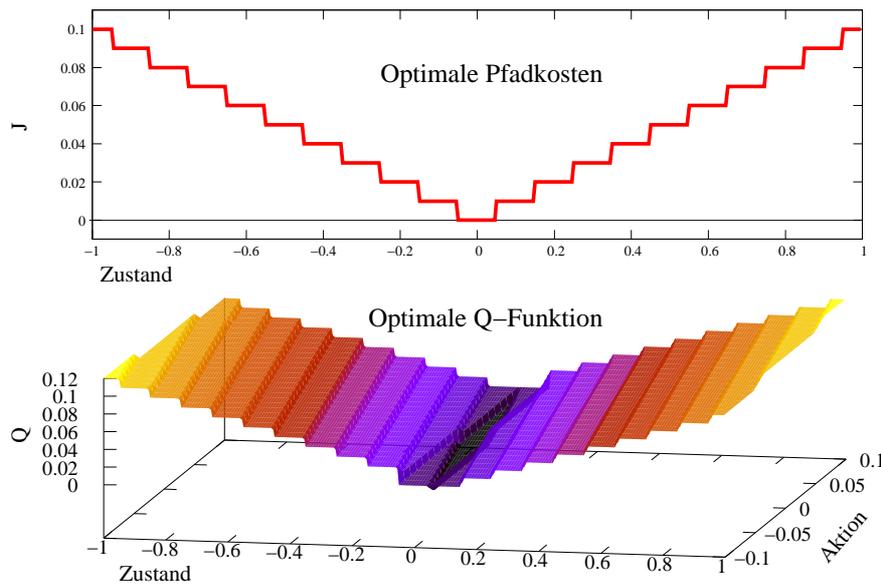


Abbildung 3.10: Die sich ergebende optimale Pfadkostenfunktion und Q-Funktion bei Lernen des einfachen Systems $x_{t+1} = x_t + u_t$ mit einer hohen Anzahl diskreter Aktionen zwischen -0.1 und 0.1. Verwendet wurde die Bereichsdefinition der direkten Kosten. Dadurch ergeben sich Treppenstufen, die sehr schwer in einem Funktionsapproximator zu repräsentieren sind.

3.4.2 Zeitoptimale Regelung mit differenzierbaren Kostenfunktionen

Im Folgenden wird eine Verfeinerung der Bereichsdefinition entwickelt. Das Hauptaugenmerk liegt dabei auf der Erzeugung genauer Regler, die eine für die Approximation der Q-Funktion geeignetere Form besitzen. Dabei sollen die zeitoptimalen Eigenschaften des Optimierungsprozesses erhalten bleiben.

In Formel 3.7 ist die verbesserte Kostenfunktion für einen Prozess mit einer Regelgröße x und einem Sollwert x^d angegeben.

$$\begin{aligned}
 c^t(x, u) &= c^t(x) & (3.7) \\
 &= \tanh^2(|x - x^d| * w) * \mathcal{C} \\
 w &= \tanh^{-1}\left(\frac{\sqrt{(0.95)}}{\hat{\mu}}\right)
 \end{aligned}$$

Diese verbesserte Kostenfunktion basiert auf der Regelabweichung des Systems, die über einen quadrierten Tangenshyperbolicus auf den Wertebereich von 0 bis 1 abgebildet wird. Somit entstehen in einem Nahbereich um den Sollwert die Kosten für einen Übergang, abhängig vom Abstand der Regelgröße vom Sollwert. Wie bei der Bereichsdefinition sind die Kosten in einem größeren Abstand jedoch konstant. Durch die Multiplikation mit \mathcal{C} wird die Funktion auf den Wertebereich von 0 bis \mathcal{C} skaliert. In Abbildung 3.11 ist die sich ergebende Kostenfunktion dargestellt. Die Weite des Toleranzbereiches wird über eine Gewichtung des Abstandes mit Faktor w durchgeführt. Zur besseren Handhabung der Kostenfunktion wird dieser Faktor über eine Breite des Toleranzbereiches $\hat{\mu}$ definiert. Diese gibt an, in welchem Abstand vom Sollwert die Kostenfunktion auf 95% ihres Wertebereiches sein soll. Die so definierte Funktion kann als stetige und differenzierbare Annäherung der Bereichsdefinition angesehen werden, die ein eindeutiges Minimum besitzt. Die Spezifikation der Kostenfunktion für einen Lernversuch ba-

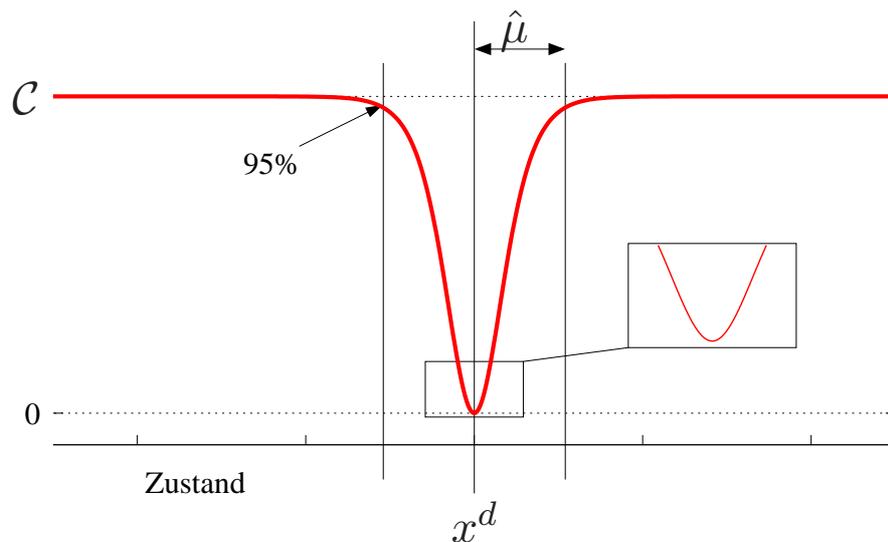


Abbildung 3.11: Die Kostenfunktion mit quadrierter tanh-Funktion im Zustandsraum, auf einer Variable dargestellt. Die Kosten außerhalb des erwünschten Zielbereichs sind auf einem kleinen, positiven Wert. Die direkten Kosten, die bei einem Übergang im Zielbereich vergeben werden, sind abhängig von ihrem Abstand zum erwünschten Zielpunkt. Dies kombiniert die zeitoptimale Regelung der Bereichsdefinition mit einer auf euklidischem Abstand beruhenden Kostenfunktion.

siert somit auf der Regelabweichung und dem Toleranzbereich, der als Annäherung durch $\hat{\mu}$ gegeben ist. Somit führt die gegebene Kostenfunktion im Vergleich zur Bereichsdefinition keine neuen Parameter ein. Die Spezifikation der Breite des Toleranzbereiches mit dem 95% Wert wird in der Arbeit durchgehend als analog zu μ betrachtet. Die Festlegung des Wertes auf 95% ist ein auf Erfahrung basierender Wert und dient ausschließlich der besseren Handhabbarkeit, um für die Gewichtung des Abstandes eine physikalisch relevante Größe berechnen zu können. In Abbildung 3.12 sind die sich ergebenden Wertfunktionen für das einfache System mit dieser Bereichsdefinition zu sehen. Bei geeigneter Wahl der Weite der Kostenfunktion sind die sich ergebenden optimalen Wertfunktionen für das System deutlich glatter als mit der Bereichsdefinition (vergleiche Abbildung 3.10). In der analytisch berechneten, optimalen Q-Funktion ergeben sich unter der Verwendung der verfeinerten Kostenfunktion deutlich bessere Eigenschaften im Zielbereich. Dort besitzt der Q-Wert auch für Bereiche nahe des Sollwertes Informationen, die mit einer gierigen Strategie ausgewertet werden können. Dies führt zu deutlich genaueren Reglern.

Für die praktische Anwendung der so definierten Kostenfunktion, auch in mehrdimensionalen Systemen, wird die obige Definition auf den mehrdimensionalen Fall erweitert. Für ein System mit m Systemvariablen wird ein m -dimensionaler Weitenvektor μ definiert, der für jede Dimension die Weite der Zielregion angibt. Zusätzlich sei ein m dimensionaler Vektor x^d gegeben, der den Sollwert für die einzelnen Dimensionen beinhaltet. Ist die Dimension nicht maßgebend für die Definition des Regelungszieles, so sind die Weiten auf Null. Diese Dimen-

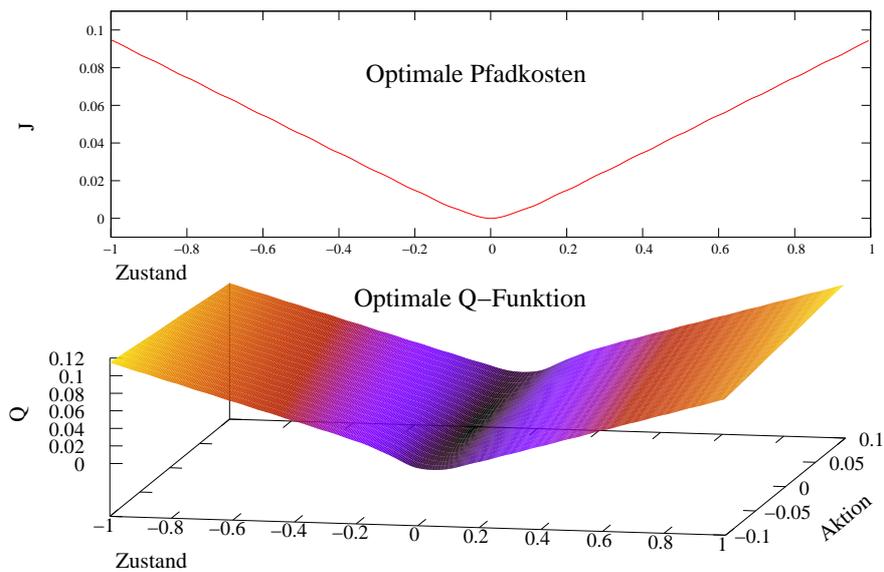


Abbildung 3.12: Die sich ergebende optimale Pfadkosten- und Q-Funktion, bei Lernen des einfachen Systems $x_{t+1} = x_t + u_t$ mit Aktionen aus $U = \{-0.1, 0, 0.1\}$ mit der \tanh^2 -Kostenfunktion. Die Treppenfunktion, die durch die Bereichsdefinition verursacht wurde, wird stark geglättet. Dadurch vereinfacht sich die Approximation der Wertfunktionen deutlich. Zusätzlich wird die Genauigkeit der Optimierung stark verbessert, da das Kostensignal auch im Toleranzbereich definiert ist.

Tabelle 3.2: Auswertung der Genauigkeit der Festwertregelung für den DC-Motor

Kostenfunktion	Iterationen bis N^*	\bar{e}_∞
\mathcal{X}^+	25	5.5
\tanh^2	20	0.53

sionen gehen somit auch nicht in das Optimierungskriterium mit ein.

$$\begin{aligned}
 c^t(x, u, x') &= c^t(x) \\
 &= \tanh^2(\|(x - x^d)^T * w\|) * c \\
 w_i &= \begin{cases} \tanh^{-1}\left(\frac{\sqrt{0.95}}{\mu_i}\right) & \text{falls } i \text{ Zielvariable, bzw. } \mu_i \neq 0 \\ 0 & \text{sonst} \end{cases}
 \end{aligned} \tag{3.8}$$

Betrachtung am DC-Motor

Die Betrachtung der verfeinerten Kostenfunktion erfolgt wiederum an der Festwertregelung des DC-Motors auf die Geschwindigkeit 421.72 rad/s. Im Unterschied zu den vorigen Versuchen wird dem Agenten nun die für diese Sollgeschwindigkeit benötigte Aktion von 0.3 (30% PWM) nicht mehr direkt zur Verfügung gestellt. Anstelle dieser werden der Aktionsmenge zwei ähnliche Aktionen hinzugefügt, die jede für sich jedoch die eigentliche Zielgeschwindigkeit im ausgeregelten Zustand nicht genau erreicht. Die verwendete Aktionsmenge ergibt sich zu $U = \{-1, 0.315, 0.285, 1\}$. Unter der Verwendung des Q_{min} -Verfahrens werden zwei verschiedene, lernende Regler betrachtet. Ein Regler wird unter der Verwendung der Bereichsdefinition mit $\mu = 20 \text{ rad/s}$ um die Sollgeschwindigkeit ($\mathcal{X}^+ = ([401.72, 441.72])$) eingelernt.

Dieser soll verglichen werden mit einem Regler mit \tanh^2 -Kostenfunktion mit 95%-Weite von $\mu = 20\text{rad/s}$. Der Regler hat jeweils die Aufgabe, den Motor aus dem Zustand $(0, 0)^T$ zeitoptimal auf Geschwindigkeit $x^d = 421.72\text{rad/s}$ zu bringen und ihn dort zu stabilisieren.

Eine äquivalente Notation, welche die Kostenfunktion für beide Regler beschreibt, kann durch die Angabe eines Vektors $\mu = (20.0, 0)$ für die Breite des Toleranzbereiches und eines Vektors $x^d = (421.7, -)$ mit $C = 0.01$ angegeben werden. Für die Versuche wurde jeweils ein explorativer Lernprozess über 100 Lerniterationen durchgeführt, die jeweils unabhängig voneinander 100 Mal wiederholt wurden. Wie in den vergangenen Versuchen hat jede Interaktionsepisode eine Dauer von 100 Schritten. Nach jedem Iterationsschritt des NFQ-Verfahrens wurde zur Bewertung der aktuell gelernte Regler ohne Exploration auf die Regelung des Prozesses angewandt. Zur Bewertung der Zeitoptimalität der Regler dienen die benötigten Zeitschritte außerhalb des Zielbereiches von $[-401.72, 441.72]$, die für diesen Versuch mit $\bar{N} = 16.00$ den optimalen Wert N^* darstellen. Die Bewertung der Genauigkeit der Regler wird aus der mittleren absoluten Regelabweichung über die jeweils letzten 50 Zeitschritte einer Testtrajektorie \bar{e}_∞ berechnet.

Der lernende Regler mit Bereichsdefinition kommt im Durchschnitt nach 25 Iterationen zum zeitoptimalen Verhalten von 16 Schritten außerhalb des Zielbereiches. Ist die Geschwindigkeit

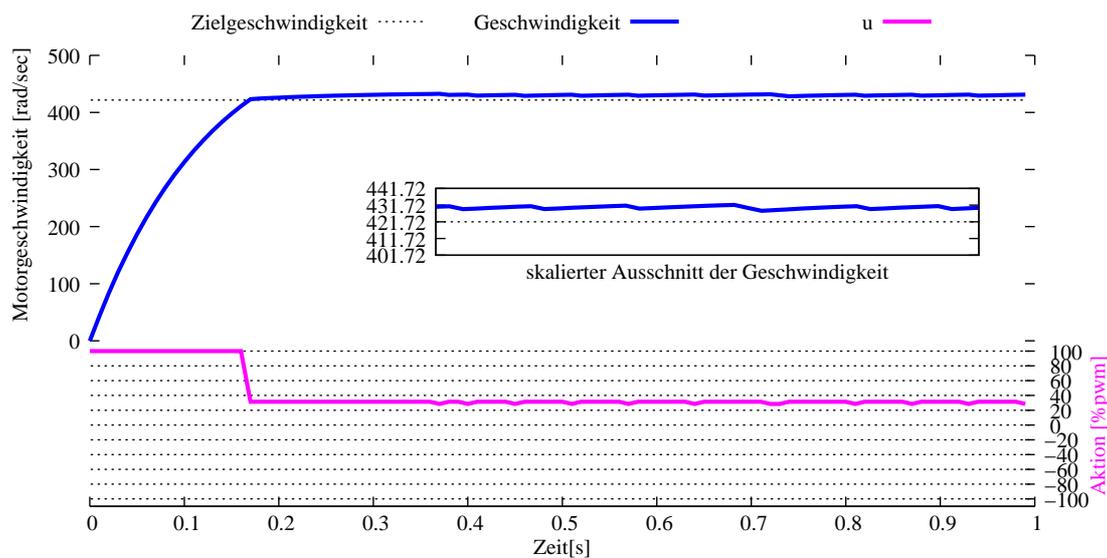


Abbildung 3.13: Die Systemtrajektorie eines mittels NFQ und der Bereichsdefinition der Kosten gelernten Reglers auf die Zielgeschwindigkeit 421.72. Dem lernenden Regler stehen zwei, der eigentlich benötigten Aktion von 0.3 (30%PWM), ähnliche Aktionen zur Verfügung. Die Regelung erreicht den Zielbereich sehr schnell und hält die Geschwindigkeit in diesem Toleranzbereich. Innerhalb dieses Bereiches ist die Regelung jedoch ungenau da das Optimierungskriterium dort keine Information enthält.

in diesem Zielbereich angelangt, ist das Optimierungskriterium für den lernenden Regler erfüllt. Mit anderen Worten, die Strategie im Zielbereich ist mehr oder weniger zufällig. Trotz diesem ist die mittlere Regelabweichung deutlich besser als die in der Kostenfunktion vorgegebene maximal erlaubte Abweichung von 20rad/s . Nach durchschnittlich 41 Iterationen erreicht diese einen Wert von $\bar{e}_\infty = 5.5\text{rad/s}$. In Abbildung 3.13 ist eine typische Systemtrajektorie des DC-Motors unter Regelung eines mit Bereichsdefinition eingelernten Reglers zu sehen.

Bei der Anwendung der \tanh^2 -Kostenfunktion ist der Lernvorgang bis zum Erreichen des zeitoptimalen Verhaltens mit im Durchschnitt 20 Iterationen etwas kleiner als bei der Bereichsdefinition. Da bei dieser Formulierung der Kosten die Strategie auch innerhalb des Zielbereichs

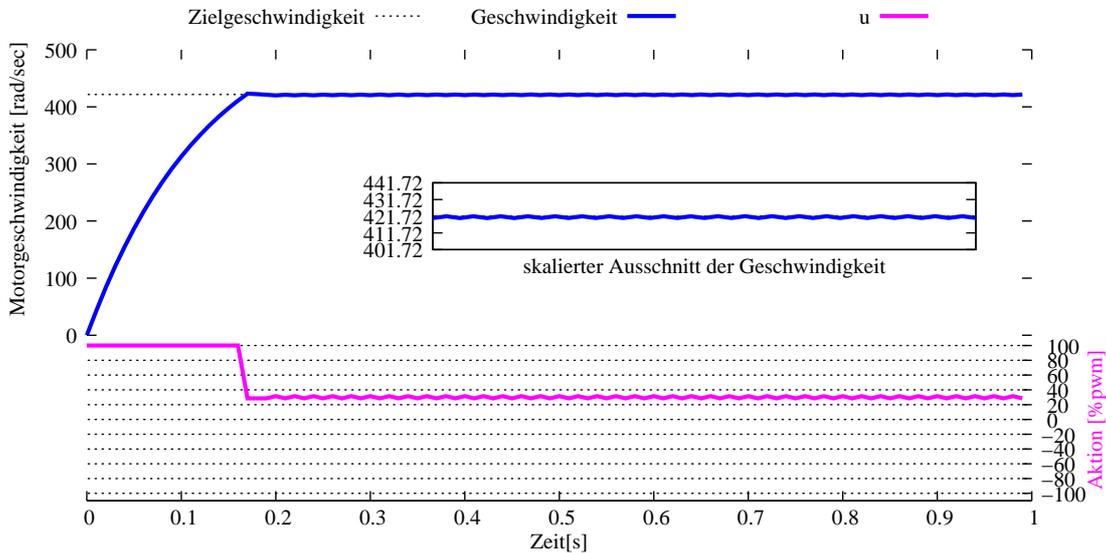


Abbildung 3.14: Die Systemtrajektorie eines mittleren NFQ und der \tanh^2 Kostendefinition gelernten Regler auf die Zielgeschwindigkeit 421.72. Dem lernenden Regler stehen zwei, der eigentlich benötigten Aktion von 0.3 (30%PWM), ähnliche Aktionen zur Verfügung. Die Regelung ist deutlich genauer als mit der Bereichsdefinition, da der lernende Regler versucht auch im Zielbereich den gewünschten Zielwert anzunähern.

ches definiert ist, ist auch der entstehende absolute Fehler deutlich kleiner. Diese liegt bei $\bar{e}_\infty = 0.53 \text{ rad/s}$ mittlere absolute Abweichung über die letzten 50 Schritte und wird im Durchschnitt nach 33 Schritten erreicht. In Abbildung 3.14 ist eine typische Systemtrajektorie des DC-Motors unter Regelung eines mit der \tanh^2 -Kostenfunktion eingelernten Reglers zu sehen. Die mittlere Aktion des Reglers im Zielbereich versucht, die real benötigte Aktion von 0.3 anzunähern. Der Fehler bleibt dadurch, bezogen auf die Genauigkeit, klein. Zusätzlich bleibt die Güte im zeitoptimalen Kriterium bestehen und die gelernten Regler sind mit 16 Schritten bis zum Ziel ebenso zeitoptimal wie mit der Bereichsdefinition.

In Tabelle 3.2 sind die zentralen Ergebnisse der Versuche zusammengefasst. Es zeigt sich, dass die \tanh^2 -Kostenfunktion das zeitoptimale Verhalten schneller erreicht und deutlich präziser ist als die Bereichsdefinition. Für einen effizient und genau arbeitenden selbstlernenden Regler ist somit der Einsatz der \tanh^2 -Kostenfunktion in Kombination mit dem Q_{min} -Verfahren eine zentrale Voraussetzung.

3.5 Die Regelung mit Führungsgröße

Die Aufgabe der Regelung eines Prozesses wurde bisher als reine Festwertregelung mit festem Sollwert betrachtet. In dieser Sichtweise ist das Regelungsziel fester Bestandteil der Definition des Optimierungsprozesses und kann direkt im Zustandsraum des Prozesses über die Kostenfunktion definiert werden. In den meisten Arbeiten mit Reinforcement-Lernen wird diese Sichtweise eingenommen. Der für die Regelungstechnik jedoch weitaus interessantere

Fall ist die Regelung mit variierendem Sollwert, bzw. mit Führungsgröße.

Im vorliegenden Kontext kann diese Aufgabe durch die Betrachtung eines unabhängigen dynamischen Systems für jeden möglichen Sollwert betrachtet werden. Als Konsequenz muss ebenfalls für jeden möglichen Sollwert ein unabhängiger Regler gelernt werden. Dies ist wenig effizient und erlaubt keine Generalisierung für Sollwerte, die in den Interaktionsepisoden am Prozess nicht vorgekommen sind. Das Ziel ist es deshalb, einen geschlossenen Ansatz zu verfolgen, der die Regelung auf beliebige Sollwerte in einem selbstlernenden Regler modelliert.

3.5.1 Modellierung des Lernprozesses

Der Regler erhält in jedem Zeitschritt eine potentiell vektorielle Führungsgröße x^d . Basierend auf dieser Führungsgröße soll das System so ausgeregelt werden, dass die Regelabweichung zeitoptimal minimiert wird. Vorerst soll davon ausgegangen werden, dass der Regler beliebige Zielwerte aus seinem Arbeitsbereich als Führungsgröße bekommen kann, sich diese während einer Trajektorie jedoch nicht verändern. Die Regelgrößen, auf die sich die Führungsgrößen beziehen, sind eine Teilmenge der Zustandsvariablen, die der Regler in jedem Zeitschritt beobachten kann. Diese Teilmenge der Zustandsvariablen sei in einem Vektor von Regelgrößen \hat{x} zusammengefasst. Die Regelabweichung sei dann als Vektor der Differenzen dieser Variablen $\tilde{x} = x^d - \hat{x}$ definiert.

Der Optimierungsprozess im selbstlernenden Regler basiert nun nicht mehr ausschließlich auf den Zustandsvariablen des Prozesses. Die Information über die Führungsgröße muss ebenfalls in die Zustandsbeschreibung des MDPs aufgenommen werden. Es ergeben sich potentiell zwei Möglichkeiten, in welcher Form diese Information in die Zustandsbeschreibung integriert werden kann. Wird die Führungsgröße direkt in den Zustand des MDP integriert, ergibt sich dieser zu (x, x^d) . Ein Nachteil dieser Modellierung ist, dass sich die Definition der Kosten erschweren würde, da dort die Regelabweichung gebildet werden muss. In regelungstechnischen Systemen ist es daher Praxis, die Regelabweichung in den Zustand des Systems zu integrieren. Für die hier betrachteten Systeme soll der Zustand des MDPs deshalb als (x, \tilde{x}) definiert werden. Aus schematischer Sicht des Regelkreises wird dem Prozess somit auch die Bildung der Regelabweichung, der Ist-Sollwert-Vergleich, zugeschlagen. In der praktischen Anwendung wird dies durch ein virtuelles Glied zur Differenzenbildung realisiert.

Die Kostenfunktion c^t kann in dieser Modellierung mit Hilfe der Bereichsdefinition oder der \tanh^2 -Funktion direkt auf dem Zustand des MDPs definiert werden. Typischer Weise beziehen sich die Kostenfunktionen ausschließlich auf die im Zustand repräsentierte Regelabweichung. Die Zustandsmodellierung kodiert auf diese Weise den aktuellen Sollwert durch die im Prozesszustand vorhandene Regelgröße und die Regelabweichung. Derselbe Prozesszustand kann somit mit verschiedenen Regelabweichungen auftreten, was für den Agent in diesem Sinne die unterschiedlichen Regelungsziele kodiert. Das durch den gelernten Regler geschlossene dynamische System besitzt nun für jeden Sollwert einen Gleichgewichtspunkt im Zustandsraum. Eine Änderung des Sollwertes bewirkt eine Änderung des Zustandes in den Variablen der Regelabweichung, worauf der Agent das System in den entsprechenden stabilen Bereich überführt, bei dem die Regelabweichung im Toleranzbereich liegt. Über die Repräsentation der Q-Funktion durch ein neuronales Netz und die geschlossene Modellierung des Reglers für beliebige Sollwerte kann eine gute Generalisierung des Reglers auch für verschiedene Sollwerte erwartet werden. Dies ist eine essentielle Eigenschaft der selbstlernenden Regler, da sie in einem ex-

plorativen Lernprozess sowohl die Systemdynamik als auch die Regeldynamik für beliebige Sollwerte aus wenigen Beispielen lernen müssen.

3.5.2 Lernen unter Zielwertveränderung

Die Modellierung der Regelabweichung im Zustand des MDPs wurde unter der Einschränkung eingeführt, dass sich der Sollwert während einer Interaktionsepisode nicht verändert. Der Grund für diese Einschränkung ist aus der Betrachtung der Dynamik der Regelgröße im MDP ersichtlich. Die Regelgröße ändert ihren Wert bei einem festen Sollwert ausschließlich aufgrund der Beeinflussung des Agenten durch die angewendete Aktion. Die externe Veränderung des Sollwertes beeinflusst die Regelgröße jedoch ebenfalls. Da die Veränderung der Führungsgrößen nicht auf bestimmte Verläufe beschränkt sein soll, ergibt sich hieraus in einem Zustandsübergang des MDPs eine zusätzliche Dynamik, die nicht durch die Prozessdynamik definiert ist und nicht durch die Aktion beeinflusst werden kann. Im strengen Sinne verletzt dies die Markovbedingung für die Modellierung des Prozesses als MDP. Wird der Sollwert während der Interaktionsepisoden nur in bestimmten Abständen verändert, ohne dass er eine kontinuierliche Veränderung erfährt, so kann diese sprunghafte Veränderung als Störung des Prozesses modelliert werden. Aus Sicht der Modellierung des MDPs gibt es in jedem Zustand somit eine geringe Wahrscheinlichkeit für einen Sprung der Regelgröße, die anschließend wieder ausgegelt werden muss.

Für die Betrachtung von kontinuierlich veränderten Führungsgrößen können diese nicht mehr als Störung modelliert werden, da die Dynamik der Sollwertveränderung die Dynamik des Prozesses überlagert. Um auch bei kontinuierlicher Veränderung des Sollwertes während der Interaktionsepisoden das Sammeln von korrekten Übergängen für den MDP zu ermöglichen, wird ein intelligentes Fehlerglied eingeführt. Die Prozessdynamik ist unabhängig vom Sollwert und der Sollwertveränderung. Im intelligenten Fehlerglied wird für jeden Übergang der Prozessdynamik ein konsistenter Übergang für den MDP gebildet (siehe Abbildung 3.15). Beobachtet

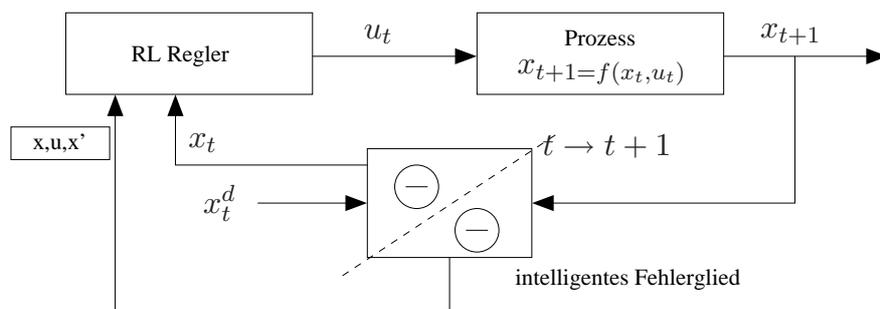


Abbildung 3.15: Die schematische Darstellung der Regelung unter veränderlichem Führungswert. Im Kontext der Modellierung des Prozesses als Reinforcement Lernproblem, werden die Regelabweichungen als zusätzliche Zustandsinformation betrachtet. Um die Veränderung des Führungswertes aus den Übergängen zu eliminieren, wird ein intelligentes Fehlerglied benutzt.

der Agent in Zeitpunkt t einen Zustand des MDPs x_t unter dem aktuellen Sollwert x_t^d und wendet eine Aktion u_t an, so würde er im nächsten Zeitschritt einen Zustand x_{t+1} beobachten, in dem die Regelabweichung auf dem Sollwert x_{t+1}^d basiert. Ist der Sollwert im Zeitschritt $t+1$ gegenüber dem Zeitschritt t verändert, ist dies ein Übergang im MDP, der nicht erwünscht ist. Das intelligente Fehlerglied speichert den Sollwert x_t^d und erzeugt die Regelabweichung im

Folgezustand, basierend auf diesem Wert. Somit erzeugt er einen bezüglich der Prozessdynamik konsistenten Übergang, welcher der Datenbasis des NFQ-Verfahrens hinzugefügt werden kann. Die Strategie der Interaktionsepisode bzw. der Zustand, auf der diese Strategie berechnet wird, basiert in jedem Zeitschritt auf der aktuellen Regelabweichung bezüglich des aktuellen Sollwertes.

Um das Wegdriften der Q-Werte zu verhindern, kann die 'Hint to Goal'-Heuristik in praktischen Anwendungen nicht mehr eingesetzt werden. Da die zu erzeugenden künstlichen Trainingsmuster jeden möglichen Sollwert der Regelung ausreichend repräsentieren müssten, würde eine Bestimmung der vollständigen Prozessvariablen im stabilen Bereich des zu lernenden geschlossenen Systems für sehr viele Sollwerte aus den Systemgleichungen, sowie die Bestimmung der optimalen Aktion für jeden Sollwert notwendig.

Das in 3.3 beschriebene Q_{min} -Verfahren ist somit für das Lernen mit veränderlichen Führungsgrößen eine zentrale Voraussetzung. Wie in 3.3 ausgeführt, bezieht sich das Verfahren zur Korrektur der Q-Werte jedoch nur auf einen Equilibriumspunkt des geschlossenen Regelkreises bzw. den stabilen Bereich um diesen Punkt im Zustandsraum. Diese stabilen Bereiche sind für die Regelung mit veränderlichem Sollwert jedoch abhängig vom aktuellen Sollwert, wodurch sehr viele dieser stabilen Bereiche existieren. In diesem Sonderfall kann das beschriebene Verfahren jedoch angepasst werden. In jedem gespeicherten Übergang ist der Sollwert, unter dem er beobachtet wurde, über die Regelabweichung und Regelgröße kodiert, enthalten. Für jeden Überhang in der Datenbasis des Verfahrens kann somit bestimmt werden, bezüglich welchem Equilibriumspunktes die Korrektur der Q-Werte durchgeführt werden muss. Werden für die Interaktionsepisoden nur eine begrenzte Menge an Führungsgrößen vorgegeben, so kann die Trainingsdatenmenge aufgespalten werden und für jeden Führungsgröße separat eine Korrektur der entsprechenden Übergänge vorgenommen werden.

In der Praxis hat sich die Anwendung des globalen Verfahrens auf die gesamte Menge der Trainingsmuster für viele Versuche als ausreichend erwiesen. Dies ist vor allem dann der Fall, wenn während der Interaktionsepisoden beliebige Führungsgrößen aus dem Arbeitsbereich der Regelung auftreten und der Regler für unterschiedlichste Führungsgrößen die Systemdynamik erkunden kann. Die in der Q-Funktion repräsentierten stabilen Bereiche sind dann ein zusammenhängender Bereich und es genügt die globale Anwendung der Korrektur, ohne die separate Korrektur der Trainingsmuster für die verschiedenen Sollwerte.

Betrachtung am DC-Motor

Die selbstlernenden Regler werden nun am Beispiel der Geschwindigkeitsregelung des DC-Motors auf beliebige Sollgeschwindigkeiten untersucht. Im Unterschied zu den vorigen Versuchen wird nun nicht mehr ausschließlich die Festwertregelung mit einem einzelnen Sollwert betrachtet, vielmehr ist es das Ziel, einen Regler zu lernen, der den Motor auf beliebige Geschwindigkeiten regeln kann. Für die Modellierung des Lernproblems wird der Zustandsvektor des MDPs um die Regelabweichung erweitert. Der Zustandsvektor des Lernproblems besteht somit nun aus 3 Dimensionen $x = (\omega, I, \omega^d - \omega)$ und beinhaltet somit die aktuelle Geschwindigkeit des Motors ω , den Strom I und die Regelabweichung $\omega^d - \omega$. Unter Verwendung der globalen Korrektur der Q-Werte werden die erzielbaren Ergebnisse bezüglich der Regelgüte für die Kostendefinition mit der \tanh^2 -Funktion und der Bereichsdefinition untersucht. Die \tanh^2 -Kostenfunktion sei definiert durch $x^d = (-, -, 0.00)$ mit $\mu = (0, 0, 20)$. Es wird somit eine zeitoptimale Regelung definiert, die die Regelabweichung minimiert. Für die Bereichsdefinition ist die äquivalente Modellierung des Regelungsproblems in der zuvor vereinbarten

Tabelle 3.3: Auswertung der Genauigkeit der Folgeregelung für den DC-Motor

Kostenfunktion	\bar{N}	\bar{e}_∞	\bar{e}_T
\mathcal{X}^+	16.86	11.61	20.61
\tanh^2	9.64	5.64	15.68

Definition gegeben mit $\mathcal{X}^+ = ([-20, 20])$. Für beide Kostenfunktionen sei $\mathcal{C} = 0.01$. Als Aktionen werden 17 Werte aus dem Bereich $[-1, 1]$ (entspricht einem Bereich von -100% PWM bis 100% PWM) mit gleichen Abständen gewählt. Für den explorativen Lernprozess werden Interaktionsepisoden mit einer Dauer von 100 Zeitschritten und einer Explorationsrate von 0.3 (30%) gewählt. Zu Beginn jeder Interaktionsepisode wird ein zufälliger Sollwert aus dem Bereich $[-500, 500]$ gewählt und der Motor aus dem Zustand $(0, 0)$ gestartet. Auf einer Interaktionsepisode wird der Sollwert konstant gehalten.

Für die Auswertung der Versuche wird nach jeder Iteration des Lernverfahrens der entstandene Regler auf 100 zufälligen Zielgeschwindigkeiten, jeweils ausgehend vom Zustand $(0, 0)$ mit einer Episodenlänge von 100 Schritten getestet. Das Gütemaß \bar{e}_∞ wird auf den jeweils letzten 50 Schritten einer Episode ermittelt. Für \bar{N} sind die durchschnittlichen Schritte außerhalb eines Toleranzbereiches von $\pm 20 \text{ rad/s}$ um die jeweilige Sollgeschwindigkeit maßgebend. Zusätzlich wird der gelernte Regler an einem Profil einer sich verändernden Sollgeschwindigkeit über 1000 Schritte getestet. Über dieses gesamte Profil wird das Gütemaß \bar{e}_T als mittlere absolute Regelabweichung über die gesamte Trajektorie bestimmt. Jeder Versuch wird 10 mal unabhängig durchgeführt und die entsprechenden Gütemaße werden gemittelt. In Tabelle 3.3 und Abbildung 3.16 sind die Ergebnisse der Versuche zusammengefasst. Mit den 17 zur Verfügung gestellten Aktionen kann der Regler in den ersten 100 Iterationen des Lernverfahrens ein sehr gutes Regelverhalten einlernen. Die Generalisierungsfähigkeit über die Regelabweichung ist sehr hoch. Im Mittel wird bei der \tanh^2 -Kostenfunktion schon nach weniger als 44 (im schlechtesten Fall 52) Iterationen der beste Regler erreicht. Dies bedeutet, dass der Regler erst wenige Zielgeschwindigkeiten und nur einen kleinen Teil der Systemdynamik beobachtet hat, aber schon sehr gut über die Testfälle generalisiert, die er noch nie gesehen hat.

Wie zu erwarten, kann mit der \tanh^2 -Kostendefinition ein genaueres Ergebnis erzielt werden als mit der Bereichsdefinition. In Abbildung 3.16 sind die Resultate der zwei verschiedenen Methoden mit ihren mittleren, minimalen und maximalen Werten über die 10 unabhängigen Versuche aufgetragen. Sowohl im schlechtesten Fall als auch im mittleren Fall ist die \tanh^2 -Kostenfunktion in allen Kriterien der Bereichsdefinition überlegen. Bemerkenswert ist ebenfalls, dass sich die Robustheit des Lernverfahrens steigern lässt, was sich in der deutlich kleineren Varianz der Gütekriterien über die 10 unabhängigen Versuche ausdrückt. In den Interaktionsepisoden wurde der Regler ausgehend von einem einzigen Startzustand bezüglich zufälliger Sollwerte geregelt. Daher zeigt sich die Leistungsfähigkeit der Generalisierung des Reglers insbesondere in der Anwendung auf ein Profil von sich verändernden Führungsgrößen. In Abbildung 3.17 ist eine Trajektorie des Reglers auf einem solchen Profil dargestellt. Der Regler hat zu diesem Zeitpunkt nur 44 Iterationen des explorativen Lernvorganges durchgeführt und somit die Systemdynamik ebenfalls nur unter 44 verschiedenen Sollwerten beobachtet. Über die gesamte Regeltrajektorie ist der Regler in der Lage, die Regelgröße im Toleranzbereich der Regelung zu halten. Die Ausregelphasen zu Beginn der Trajektorie werden zeitoptimal durchgeführt. Auch auf der Folgeregelung zeigt sich hierbei bei der \tanh^2 -Kostenfunktion eine bessere Regelgüte \bar{e}_T als bei der Bereichsdefinition.

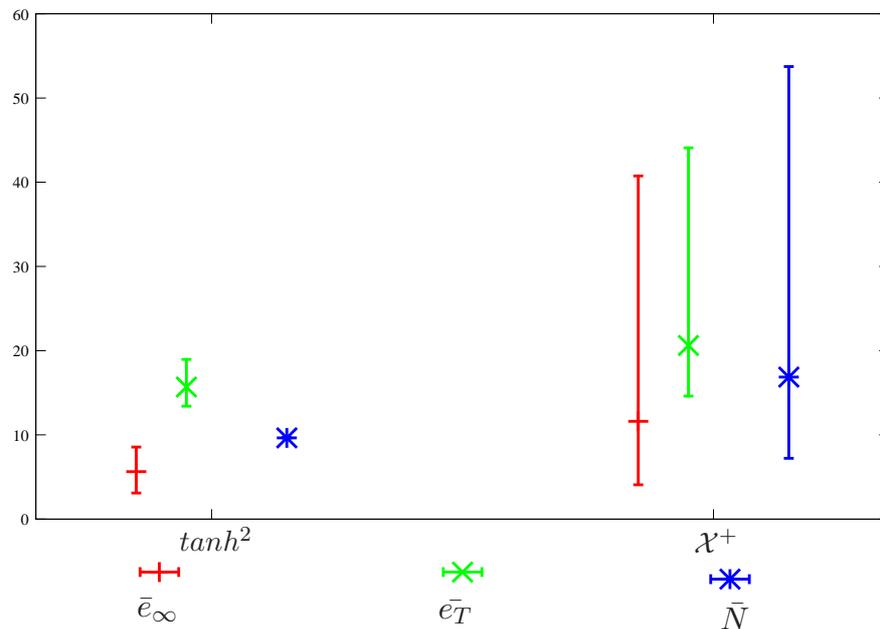


Abbildung 3.16: Die Auswertung der beiden Kostenfunktionen auf der Folgeregelung des DC-Motors. Aufgetragen sind die mittleren, minimalen und maximalen Werte über drei Kriterien aus 10 unabhängigen Versuchen. Kleinere Werte bedeuten bessere Resultate. Die \tanh^2 Kostenfunktion hat nicht nur einen kleineren mittleren Fehler, sie lernt auch wesentlich stabiler als die Bereichsdefinition, was sich in einer kleineren Varianz der Werte ausdrückt.

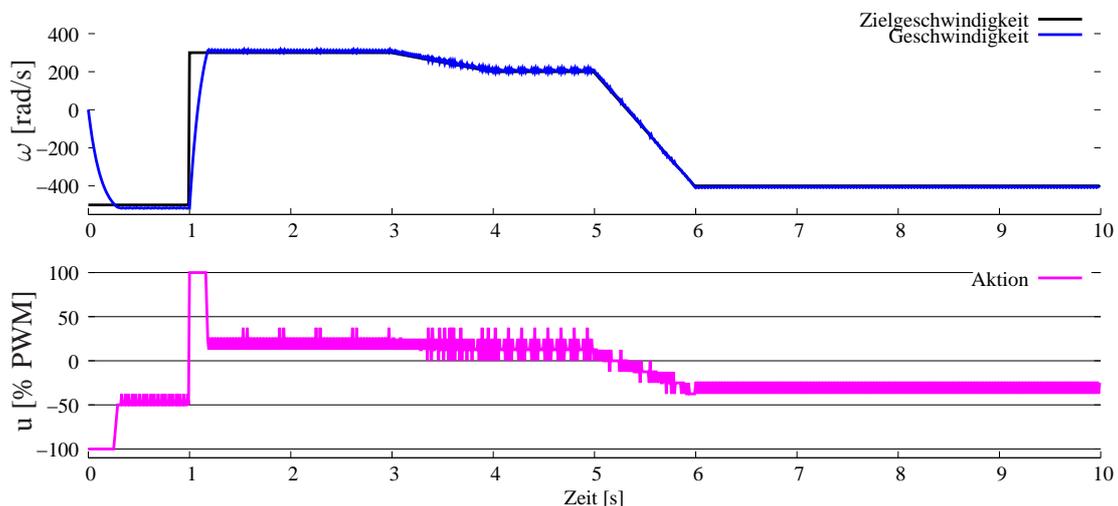


Abbildung 3.17: Eine Folgeregelung am DC-Motor mit 17 Aktionen aus dem Bereich $[-1, 1]$, in gleichen Abständen gewählt. Die Genauigkeit der Regelung ist durch die Anwendung der \tanh^2 -Kostenfunktion im Gegensatz zur Bereichsdefinition deutlich gesteigert. Durch die Verwendung diskreter Aktionen muss die Regelstrategie die benötigten Spannungen für den Motor durch zeitliche Änderung der Aktion annähern.

3.6 Verbesserung des Regelverhaltens durch Erweiterung der Systemdynamik

Selbstlernende Regler auf Basis des NFQ zeigen in vielen Bereichen sehr gute Eigenschaften, die sie für den Einsatz an komplexen, nicht linearen und mehrdimensionalen Systemen geeignet machen. Speziell ihre Beschränktheit auf diskrete Aktionen ist jedoch bezüglich eines präzisen Regelverhaltens problematisch. Die Formulierung des regelungstechnischen Problems der Folgeregelung ist hierbei eine besondere Herausforderung, da die Anzahl der diskreten Aktionen besonders hoch sein muss, um ein gutes Regelverhalten für beliebige Zielwerte zu erreichen. Ein Verfahren, um dieses Problem beherrschbar zu machen, ist die künstliche Erweiterung der Systemdynamik des betrachteten Prozesses. Aus der Sicht des lernenden Reglers wird der zu regelnde Prozess um weitere Systemvariablen mit typischer Weise nichtlinearer Dynamik erweitert, die ihrerseits den Prozess beeinflussen. Dies macht den gesamten Prozess für menschliche Begriffe deutlich schwieriger zu verstehen. Die selbstlernenden Regler können diese Eigenschaften der erweiterten Dynamik jedoch durch ihre optimierenden Eigenschaften hervorragend ausnutzen, um mit diskreten Aktionen eine kontinuierliche Eingriffsgröße zu erzeugen.

3.6.1 Dynamische Ausgabelemente als Erweiterung der Systemdynamik

Die dynamischen Ausgabelemente (DAEs) sind eine künstliche Erweiterung der Systemdynamik des zu regelnden Prozesses [41, 75, 76]. Dies sind Elemente, die zwischen den lernenden Regler und den zu regelnden Prozess geschaltet werden, um die Menge der erreichbaren Eingriffsgrößen des lernenden Reglers zu erhöhen (siehe Abbildung 3.18). Das dynamische

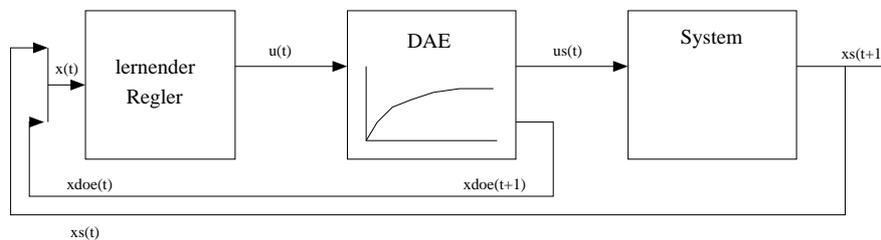


Abbildung 3.18: Eine schematische Darstellung der erweiterten Systemdynamik, mittels dynamischem Ausgabelement. Das DAE besitzt eine eigene Systemdynamik, dessen Systemvariablen zusätzlich in die Zustandsbeschreibung des lernenden Reglers mit eingehen.

Ausgabelement besitzt eine eigene Systemdynamik und eigene Prozessvariablen, die seinen Zustand im Zeitbereich beschreiben. Diese Systemdynamik muss, wie auch der zu regelnde Prozess, ein autonomes dynamisches System sein. Es bestehen sonst keine weiteren Einschränkungen. Insbesondere darf das System eine nichtlineare Dynamik aufweisen. Die Systemdynamik eines DAE ist gegeben durch ein zeitdiskretes dynamisches System der Form:

$$\xi_{t+1} = g(\xi_t, u_t) \tag{3.9}$$

$$\kappa_t = O(\xi_t) \tag{3.10}$$

Die Reglerausgabe des Agenten u beeinflusst direkt den internen Zustand ξ des DAEs durch die gegebene zeitdiskrete Übergangsfunktion g . Basierend auf dem internen Zustand des DAEs

berechnet es seine Ausgabe κ durch eine Abbildung $\kappa = O(\xi)$, die als Eingriffsgröße für den zu regelnden Prozess dient. Aus Sicht des lernenden Reglers ergibt sich ein zu regelndes dynamisches System, das sich aus der Kombination des zu regelnden Prozesses und der künstlich hinzugefügten Systemdynamik ergibt. Um die Eigenschaften eines MDPs in der Modellierung des kombinierten Prozesses zu wahren, muss der Zustandsvektor des Reglers um den internen Zustand des DAEs erweitert werden. Das kombinierte System besitzt die Systemdynamik

$$\begin{pmatrix} x_{t+1} \\ \xi_{t+1} \end{pmatrix} = \begin{pmatrix} f(x_t, O(g(\xi_t, u_t))) \\ g(\xi_t, u_t) \end{pmatrix} \quad (3.11)$$

mit den eingesetzten Systemgleichungen des DAEs in die Systemdynamik des zu regelnden Prozesses. Hierbei wird davon ausgegangen, dass sich die Systemdynamik des DAEs als rein virtuelles Berechnungsglied ohne Zeitverlust berechnen lässt und instantan bei Wahl der Eingriffsgröße zur Verfügung steht. Dies ist plausibel, da es sich um ein künstliches Übertragungsglied handelt und nicht die Dynamik eines realen Systems repräsentiert.

Betrachtung am DC-Motor

Für die Regelung der Geschwindigkeit des DC-Motors wird ein integrierendes DAE benutzt. Dies bedeutet, dass die Ausgabe des lernenden Reglers die aktuell an den Motor angelegte Spannung um diskrete Werte erhöhen oder erniedrigen kann. Zusätzlich wird im DAE eine Beschränkung der Ausgabe auf den erlaubten Bereich von $[-1, 1]$ durchgeführt. Erhöht der lernende Regler die Ausgabe über den erlaubten Bereich, so wird diese auf die Intervallgrenze zurückgesetzt. In Formel 3.12 sind die Systemgleichungen des verwendeten DAE aufgeführt. Die Funktion $bound(\cdot)$ beschreibt hierbei die Begrenzung des Ausgabewertes auf den erlaubten Bereich.

$$\xi_{t+1} = bound(\xi_t + u_t) \quad (3.12)$$

$$\kappa_t = \xi_t \quad (3.13)$$

Der Zustandsvektor des gesamten MDPs besteht somit aus 4 Dimensionen $x = (\omega, I, \chi, \tilde{\omega})$. Er beinhaltet die aktuelle Geschwindigkeit des Motors ω , den Strom I , die Regelabweichung $\tilde{\omega}$ und zusätzlich den Zustand des DAEs ξ . In diesem speziellen Fall ist der Zustand des DAEs gleichbedeutend mit dem aktuell an den Mikrocontroller kommunizierten PWM-Wert u_{pwm} . Die Aktionen, die der lernende Regler in jedem Zeitschritt wählen kann, sind auf eine Menge von 9 diskreten Werten begrenzt: $U = \{-1.0, -0.3, -0.1, -0.01, 0, 0.01, 0.1, 0.3, 1.0\}$. Diese sind so gewählt, dass sowohl schnelle Sprünge der Ausgabespaltungen, als auch kleine Korrekturen möglich sind.

Der Lernprozess und die Auswertung bleiben ansonsten zum vorigen Abschnitt unverändert. Die einzige Änderung aus der Sicht des lernenden Reglers ist die Erhöhung der Dimension des Zustandsvektors und somit eine Erhöhung der Dimension der Netzeingabe und der Definitionen der Kostenfunktionen. Die \tanh^2 -Kostenfunktion wird somit notiert, als $x^d = (-, -, -, 0.00)$ mit $\mu = (0, 0, 0, 20)$. Für die Bereichsdefinition ist die äquivalente Modellierung des Regelungsproblems in der zuvor vereinbarten Definition gegeben mit $\mathcal{X}^+ = ([[]][[-20 \ 20])$.

In Abbildung 3.19 ist eine beispielhafte Regeltrajektorie des selbstlernenden Reglers auf einer ausgewählte Zielgeschwindigkeit zu sehen. Der Regler hat gelernt, die Möglichkeiten des DAEs zur Wahl der Spannung voll auszunutzen. Zusätzlich wird diese im Zielbereich weiter korrigiert, um einen möglichst kleine bleibende Regelabweichung zu erzeugen.

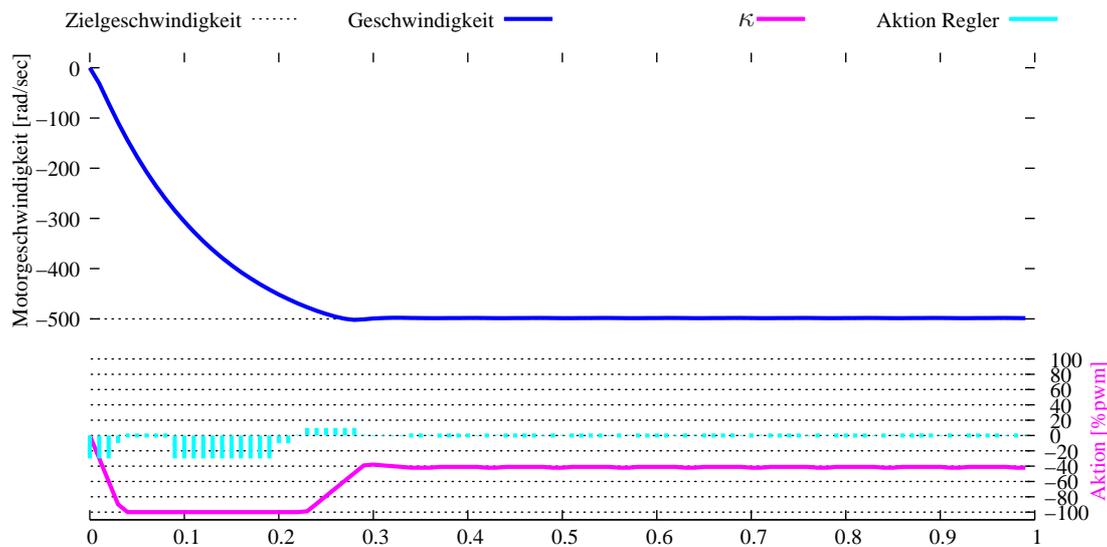


Abbildung 3.19: Systemtrajektorie des DC-Motors bei Folgeregelung auf einen bestimmten Zielwert. Die Systemdynamik wurde mit einem integrierenden dynamischen Ausgabe Element erweitert. Die Aktionen vom gelernten Regler sind als Impulse dargestellt, die an den Motor weitergegebene Spannung, als pinke Linie dargestellt. Mit einer sehr kleinen Menge von Aktionen ist der lernende Regler in der Lage alle Zielgeschwindigkeiten im Arbeitsbereich zu erreichen. Der Regler lernt dabei die kombinierte Dynamik, bestehend aus Systemdynamik und künstlicher Dynamik für sich auszunutzen.

Tabelle 3.4: Auswertung der Reglergüte unter Verwendung von NFQ und eines integrierenden DAE am DC-Motor

Kostenfunktion	\bar{N}	\bar{e}_∞	\bar{e}_T
\mathcal{X}^+	17.90	8.11	18.24
\tanh^2	9.76	1.48	12.01

In Tabelle 3.4 und Abbildung 3.20 sind die Ergebnisse der Auswertung des lernenden Reglers aufgeführt. Zum Vergleich wurden wiederum Versuche mit den zwei möglichen Kostenfunktionen durchgeführt. Die \tanh^2 -Kostenfunktion kann die Möglichkeiten des Reglers voll ausschöpfen und erzielt einen deutlich kleineren Fehler als ohne die Verwendung des DAEs. Die Bereichsdefinition profitiert nicht so stark von der Benutzung dieses dynamischen Ausgabe Elements und kann den mittleren Fehler nur gering verbessern. In Abbildung 3.20 kann für die \tanh^2 -Kostenfunktion eine signifikante Verbesserung über die 10 unabhängig durchgeführten Versuche, sowohl im Vergleich zu der Bereichsdefinition als auch zu den Versuchen ohne DAE festgestellt werden. Die mittleren Gütekriterien haben sich in allen Kennzahlen verbessert und ihre Varianz über die Versuche ist sehr gering. In Abbildung 3.21 ist eine Regeltrajektorie auf dem zur Auswertung definierten Profil der Führungsgröße aufgetragen. An diesem Profil zeigt sich die erhöhte Genauigkeit für beliebige Zielwerte. Durch die Verwendung des DAE können die zur Stabilisierung der Geschwindigkeit benötigten Eingriffsgrößen für den Prozess deutlich besser angenähert werden. Dies ergibt glattere Reglerausgaben und eine genauere Einhaltung der Führungsgröße.

Die lernenden Regler können durch diese künstliche Erweiterung der Systemdynamik deutlich genauer werden. Dies ist vor allem der kontinuierlichen Natur des Aktionsraumes des dynami-

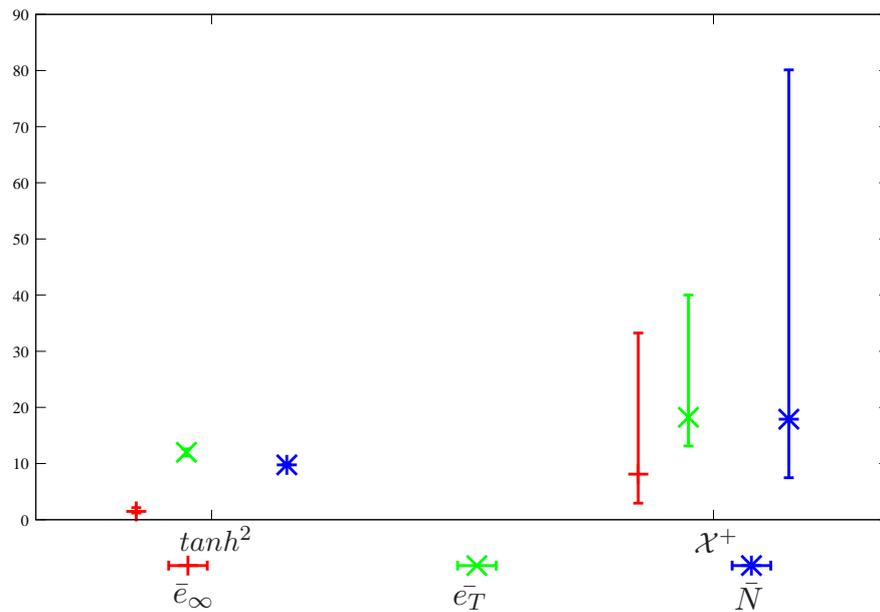


Abbildung 3.20: Die Auswertung der beiden Kostenfunktionen auf der Folgeregelung des DC-Motors. Aufgetragen sind die mittleren, minimalen und maximalen Werte über drei Kriterien aus 10 unabhängigen Versuchen bei Verwendung eines integrierenden DAEs. Kleinere Werte bedeuten bessere Resultate. Die $\tan h^2$ Kostendefinition kann die Möglichkeiten des DAEs zur Steigerung der Genauigkeit voll ausnutzen.

schen Systems zuzuschreiben. Mit Hilfe dieser Erweiterungen wird dieses Problem für praktische Anwendungen beherrschbar, es liefert jedoch noch keine vollständig zufriedenstellende Lösung für Problemstellungen wie den DC-Motor. Dies liegt daran, dass die Systemdynamik des erweiterten Prozesses immer noch mit diskreten Aktionen beeinflusst wird. Ist das DAE geeignet gewählt, sind die Auswirkungen der diskreten Natur der Aktionen abgeschwächt. Ein Nachteil der DAEs ist jedoch, dass nicht alle Eingriffsgrößen des Prozesses direkt erreichbar sind. Diese müssen erst über mehrere Zeitschritte mit der Dynamik des DAEs selbst hergestellt werden. Dadurch werden die entstehenden Regler etwas schlechter im Gütekriterium des zeitoptimalen Verhaltens \bar{N} und die Komplexität der benötigten Regelstrategie steigt.

3.6.2 Intelligente dynamische Ausgabeelemente zur Integration klassischer Regelkonzepte

Die Verwendung der dynamischen Ausgabeelemente als künstliche Erweiterung der Systemdynamik ist ein hilfreiches Konzept bei der Verwendung diskreter Aktionen an Prozessen mit kontinuierlichen Eingriffsgrößen. Im Gegensatz zu klassischen Regelgesetzen wird die Regelabweichung jedoch durch die Verwendung eines neuronalen Funktionsapproximatoren zur Bestimmung der Strategie nicht exakt zu Null.

Im Rahmen dieser Arbeit wurde deshalb eine Weiterentwicklung der DAE entwickelt, die es erlaubt, klassische Regelgesetze in die DAE-Struktur zu integrieren. Die so entstandenen Ausgabeelemente werden durch ihre erweiterten Berechnungsmöglichkeiten als intelligente dynamische Ausgabeelemente (IDAE) bezeichnet. Auf diese Weise können dem gesamten Prozess Eigenschaften verliehen werden, die eine sehr exakte Ausregelung ermöglichen. Die zeitoptimale Regelung durch den selbstlernenden Regler kann in diesem Prozess ausgenutzt werden,

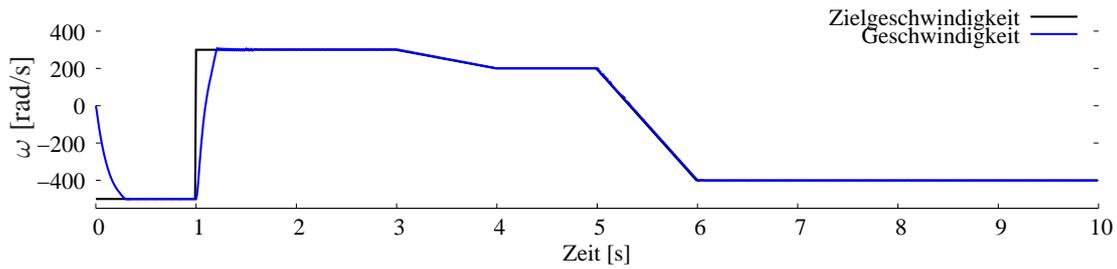


Abbildung 3.21: Die Trajektorie des mit NFQ unter Verwendung eines integrierenden DAE gelernten Reglers auf dem definierten Profil der Führungsgröße. Durch die Erweiterung der Systemdynamik kann die Reglergüte deutlich gesteigert werden.

um Bereiche zu optimieren, in denen das klassische Regelgesetz nicht zufriedenstellend arbeitet. In diesem Sinn müssen die im IDAE integrierten Regelgesetze nicht optimal für den zu regelnden Prozess sein und können nur Teilaspekte des Problems lösen. Der Grad an Intelligenz im verwendeten IDAE kann, je nach Wissen um den vorliegenden Prozess, unterschiedlich gewählt werden.

Die Systemdynamik des IDAE kann als Erweiterung der zeitdiskreten Systemdynamik des DAE betrachtet werden

$$\xi_{t+1} = g(\xi_t, x_t, u_t) \quad (3.14)$$

$$\kappa_t = O(\xi_{t+1}, x_t) \quad (3.15)$$

Dabei basiert die Berechnung seiner internen Dynamik nicht mehr ausschließlich auf seinem aktuellen Zustand ξ und der Eingabe vom lernenden Regler u , sondern kann zusätzlich auf dem Zustand des gesamten MDPs basieren. Die Ausgabe des IDAE erfolgt wiederum über eine Ausgabefunktion O , die, basierend auf dem internen Zustand und dem Zustand des MDP, eine Ausgabe berechnet. Wie bei der Verwendung der DAE muss der Zustand des IDAE in den Zustandsvektor des MDP integriert werden. Das intelligente DAE berechnet somit in einem Zeitschritt, basierend auf dem aktuellen Zustand des Prozesses und seinem internen Zustand, seine Dynamik. Diese Berechnung kann wiederum als instantan angesehen werden, da es sich um ein künstliches Berechnungsglied handelt. Basierend auf dem neuen Zustand des IDAE und des Prozesszustandes wird die Ausgabe als Eingriffsgröße an den Prozess ausgegeben.

Auf diese Weise lassen sich sehr viele klassische Regelgesetze in den Prozess integrieren, da sie ihre Ausgabe, basierend auf der Regelabweichung in der Funktion O , berechnen können. Durch die Verwendung eines internen Zustandes im IDAE, lassen sich auch integrierende Regelgesetze verwenden, die besonders geeignet sind, bleibende Regelabweichungen zu eliminieren.

Betrachtung am DC-Motor

Für die Regelung des DC-Motors wird eine PI-Reglerkomponente in das IDAE integriert. Die Ausgabefunktion des IDAE ist durch das PI Regelgesetz mit Parametern k_p und k_i gegeben.

$$\kappa_t = O(\xi_{t+1}, x_t) = \text{bound}(k_p * (\omega_t^d - \omega_t) + k_i * \xi_{t+1}) \quad (3.16)$$

Entspricht der interne Zustand des IDAE dem aktuellen Wert des Fehlerintegrals des PI-Reglers, das als Anpassung an die zeitdiskrete Eigenschaft des Prozesses als Summe $\sum_t (\omega_t^d -$

$\omega_t)\Delta t$ realisiert wird, so ergibt sich das IDAE als einfacher PI-Regler. Die Parameter dieses PI-Regelgesetzes wurden zu $k_p = 0.01$ und $k_i = 0.6$ festgelegt. Dieser PI-Regler besitzt ein deutliches Überschwingen und arbeitet durch die Begrenzung der Aktionen im Prozess nicht optimal. Dies wird durch die Bildung des Fehlerintegrals während der vollen Aussteuerung der Ausgabe des Reglers verursacht, wodurch sich das Fehlerintegral zu einem sehr hohen Wert aufsummiert. Bei kleinen Sprüngen der Führungsgröße besitzt der PI-Regler hingegen nur ein schwaches Überschwingen und eliminiert die bleibende Regelabweichung schnell. Durch das Regelgesetz selbst wird die bleibende Regelabweichung im Rahmen der numerischen Genauigkeit vollständig eliminiert. Zur Optimierung des Verhaltens der vorgegebenen PI-Komponente beeinflusst der lernende Regler die Bildung des Fehlerintegrals. Dies wird in der Übergangsfunktion des IDAE realisiert, indem die Ausgabe des lernenden Reglers als Faktor bei der Aufsummierung der internen Zustandsvariable fungiert.

$$\xi_{t+1} = g(\xi_t, x_t, u_t) = \xi_t + u_t(\omega_t^d - \omega_t)\Delta t \quad (3.17)$$

Der Zustand des gesamten MDPs ergibt sich zu $x = (\omega, I, \xi, e)$. Basierend auf diesen Informationen entscheidet der lernende Regler, mit welchem Faktor das Fehlerintegral mit dem aktuellen Fehler aufsummiert werden soll. Wendet der Agent konstant die Aktion 1 an, so ergibt sich das Verhalten der reinen PI-Reglerkomponente. Die diskrete Aktionsmenge für den Lernversuch werden zu $U = \{-0.1, 0, 0.25, 0.5, 0.75, 1\}$ gewählt. Somit kann der Regler die Bildung des Fehlerintegrals ganz oder teilweise unterbinden und es zusätzlich durch die negative Aktion entgegengesetzt zum aktuellen Regelabweichung beeinflussen. Für die Lernversuche mit NFQ wird dieselbe Vorgehensweise wie bei den vorhergehenden Lernversuchen verwendet.

In Abbildung 3.22 ist eine Trajektorie des gelernten Reglers im Vergleich zum Verhalten der reinen PI-Reglerkomponente bei einem Sprung der Führungsgröße abgebildet. Der lernende

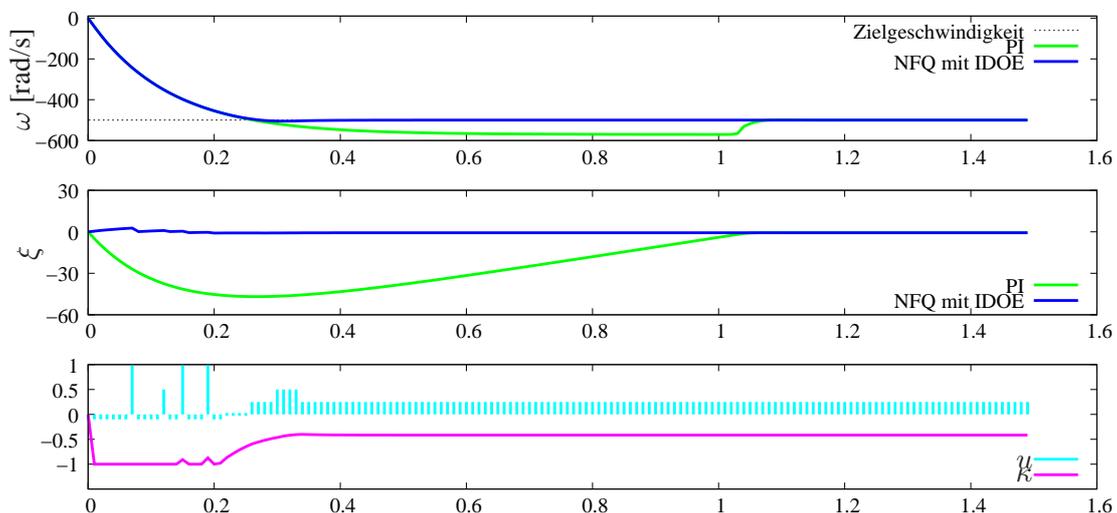


Abbildung 3.22: Das Regelverhalten am Beispiel eines Sprunges der Führungsgröße für den PI-Regler und NFQ mit IDAE.

Regler kann die verwendete PI-Komponente ausnutzen und beeinflusst die Bildung des Fehlerintegrals, um ein Überschwingen zu verhindern. Im Durchschnitt nach 19 Iterationen lernt der Regler mit dem gegebenen IDAE die Regelung der Geschwindigkeit in den Toleranzbereich

Tabelle 3.5: Regelgüte bei Verwendung eines IDAE für die Geschwindigkeitsregelung des DC-Motors

Regler	\bar{N}	\bar{e}_∞	\bar{e}_T
PI	28.98	8.8	22.5
NFQ mit IDAE	9.63	0	10.49

der 100 Sollwerte in den Testtrajektorien. Nach 60 Iterationen des explorativen Lernprozesses wird der beste Regler gefunden, der auf allen Testtrajektorien eine bleibende Regelabweichung von Null besitzt und ein zeitoptimales Verhalten zeigt. In Tabelle 3.5 sind die erzielten Werte der Regelgüte auf der betrachteten Problemstellung zusammengefasst. Die Verwendung der PI-Komponente in Form eines IDAE ist in diesem Fall eine effektive Möglichkeit, den Fehler im Beharrungszustand zu verkleinern. Voraussetzung dafür ist, dass der lernende Regler die Möglichkeit besitzt, die Arbeitsweise der integrierten Reglerkomponente in den Bereichen zu beeinflussen, in denen sie nicht optimal arbeitet. Die IDAE werden als generelle Möglichkeit angesehen, um für Systeme mit sehr hohen Anforderungen an die Genauigkeit die bleibende Regelabweichung zu eliminieren. Der Entwurf solcher Regelgesetze kann jedoch ein sehr hohes Maß an Vorwissen und die Suche nach geeigneten Parametern notwendig machen. In der vorliegenden Arbeit wird die Verwendung der IDAE nicht weiter untersucht. Vielmehr steht die Untersuchung der Leistungsfähigkeit der selbstlernenden Regler ohne Vorwissen über den Prozess im Vordergrund. Die vorgestellte Methode ist jedoch eine effektive Möglichkeit, alle vorgestellten Verfahren bei Vorliegen von Vorwissen über den Prozess weiter zu verbessern.

3.7 Zusammenfassung

Im vorliegenden Kapitel wurde die Modellierung selbstlernender Regler als Optimierungsprozess mit NFQ diskutiert. Durch die entwickelte Modellierung sind die gelernten Regler in der Lage, ihren Zustand, basierend auf einer in ihm repräsentierten Regelgröße in Bezug auf eine externe Sollwert geeignet zu beeinflussen. Es werden somit selbstlernende Regler mit externem Sollwert bzw. Führungsgröße betrachtet. Da die Regelung ein andauernder Prozess ohne festen Endzeitpunkt darstellt, ist die Modellierung des Lernprozesses als SKP Problem nicht mehr möglich. Für den Lernprozess ergibt sich dadurch ein Wegdriften der Q-Werte, das zu einer Degeneration der Q-Funktion führt. Wie diskutiert, ist die existierende 'Hint to Goal'-Heuristik eine Möglichkeit, dieses Wegdriften zu vermeiden. Sie erfordert jedoch ein hohes Maß an Vorwissen, das für allgemeine Systeme und die Regelung auf beliebige Sollwerte meistens nicht zur Verfügung steht. Es wurde ein Verfahren zur Korrektur des systematischen Fehlers in der Q-Funktion entwickelt, das in der Lage ist, das Wegdriften der Q-Werte im NFQ-Verfahren zu verhindern. Dieses Verfahren benötigt kein Vorwissen über den Prozess und ist für beliebige Systeme anwendbar, ohne weitere Parameter in den Lernprozess einzuführen. Trotz des Verzichtes auf Vorwissen über den zu regelnden Prozess lernt das Verfahren schneller als die 'Hint to Goal'-Heuristik und erzeugt einen stabileren Lernverlauf. Für die Modellierung der zeitoptimalen Regelung ist die Bereichsdefinition der direkten Kosten eine effektive Methode, um mit wenig Vorwissen über den Prozess einen zeitoptimalen, stabilen Regler zu lernen. Durch ihre Form ist im Toleranzbereich der Regelung das Verhalten des Reglers nicht weiter spezifiziert. Trotzdem sind die gelernten Regler in ihrer bleibenden Regelabweichung besser als der vorgegebene Toleranzbereich. Durch die Entwicklung einer verbesserten Form der Kostenfunktion durch die Benutzung einer \tanh^2 -Funktion konnte die Genauigkeit der

Regelung deutlich verbessert werden. Diese Verbesserung basiert zum einen auf der genaueren Definition der Kostenfunktion im Toleranzbereich, zum anderen auf der Vermeidung von Treppenstufen in der Q-Funktion, die für die Approximation im neuronalen Netz störend sind. Diese geforderte Genauigkeit der Regelung kann durch die dynamischen Ausgabeelemente noch weiter verbessert werden. Trotz der Verwendung von wenigen diskreten Aktionen in NFQ ermöglichen die DAE eine sehr hohe Anzahl von erreichbaren Eingriffsgrößen für den zu regelnden Prozess. Wie gezeigt wurde, kann durch die Erweiterung der DAE zu intelligenten DAE Vorwissen in den Regler integriert werden, das die bleibende Regelabweichung trotz der Verwendung diskreter Aktionen vollends eliminiert.

Die Erweiterung von NFQ zur Verwendung kontinuierlicher Aktionen

Die Lösung regelungstechnischer Problemstellungen auf Basis des NFQ-Verfahrens hat sich in vielen Anwendungen als effiziente Methode für die Erstellung selbstlernender Regler bewährt. Insbesondere die Anpassungen des Verfahrens auf die regelungstechnischen Prozesse, die im letzten Kapitel vorgestellt wurden, ermöglichen eine deutliche Erhöhung der Genauigkeit und die Anwendung des Verfahrens auf ein breiteres Spektrum von Problemstellungen.

Eine zentrale Eigenschaft des NFQ-Verfahrens ist die Verwendung von diskreten Aktionen. Typischerweise besitzen technische und physikalische Prozesse jedoch kontinuierliche Eingriffsgrößen, deren genaue Einstellung und Nachführung für den regelungstechnischen Prozess wichtig ist. Durch die Verwendung diskreter Aktionen stößt das NFQ-Verfahren hier an seine Grenzen.

Im vorliegenden Kapitel werden die Probleme, die bei der Verwendung diskreter Aktionen in dynamischen Systemen mit kontinuierlichen Eingriffsgrößen auftreten, eingehend diskutiert. Anschließend werden die Auswirkungen der Verwendung kontinuierlicher Aktionen auf das NFQ-Verfahren herausgearbeitet. Aus den Untersuchungen folgt eine Erweiterung des NFQ-Verfahrens auf kontinuierliche Aktionen, die direkt auf der Generalisierungsleistung der neuronalen Q-Funktion basiert. Dieses Verfahren dient der empirischen Untersuchung der Vorteile kontinuierlicher Aktionen. Ein Nachteil des Verfahrens ist die drastische Erhöhung der Rechenzeit, nicht nur beim Lernen, sondern auch bei der Ausführung. Diese macht eine Anwendung auf echtzeitnahe Systeme als selbstlernender Regler unpraktikabel. In Abschnitt 4.3 wird mit NFQCA ein effizientes Verfahren entwickelt, das NFQ für die Verwendung mit kontinuierlichen Aktionen unter Echtzeitanforderungen erweitert. Mit Hilfe dieses Verfahrens ist es möglich, eine kontinuierliche Strategie in Form eines explizit repräsentierten neuronalen Netzes zu lernen. Dabei werden die Vorteile des NFQ-Verfahrens und kontinuierlicher Aktionen zu einem effektiven Instrument zur Erstellung selbstlernender Regler kombiniert. Die durchgeführten Untersuchungen zeigen, dass das Verfahren trotz der Verwendung kontinuierlicher Aktionen keine Verschlechterung der Lerngeschwindigkeit mit sich bringt. Die vorgestellten Methoden werden an der stark nichtlinearen Systemdynamik eines Unterwasserfahrzeuges untersucht. Unter anderem wird mit diesem System aufgezeigt, dass NFQ durch die Verwendung diskreter Aktionen in regelungstechnischen Anwendungen auf bestimmte Grenzen stößt.

4.1 Voruntersuchungen

Die für die Erweiterung des NFQ-Verfahrens auf kontinuierliche Aktionen notwendigen Voruntersuchungen werden im Folgenden durchgeführt. Hierzu werden die Auswirkungen diskreter Aktionen in Systemen mit kontinuierlichen Eingriffsgrößen näher betrachtet und das NFQ-Verfahren selbst auf seine Verwendung mit kontinuierlichen Aktionen hin untersucht.

4.1.1 NFQ in dynamischen Systemen mit kontinuierlichen Eingriffsgrößen

Die Modellierung eines regelungstechnischen Prozesses als MDP mit diskreten Aktionen bedeutet nicht zwingend die Einschränkung der betrachteten Prozesse auf dynamische Systeme mit diskreten Eingriffsgrößen. Wie in Kapitel 3 am DC-Motor mit nichtlinearer Endstufe gezeigt wurde, können auch regelungstechnisch relevante Prozesse unter Verwendung diskreter Aktionen geschlossen werden. Die erreichbare Genauigkeit kann dabei in der Praxis durchaus zufriedenstellend sein [41]. Im Speziellen ist dies der Fall bei Prozessen, die in ihrer Systemdynamik eine gewisse Trägheit besitzen und bei denen der Regler in sehr kurzen Zeitabständen eingreifen kann. Wie im letzten Kapitel dargestellt, kann eine solche Systemträgheit auch durch eine künstlich hinzugefügte Systemdynamik in Form eines DAEs herbeigeführt werden. Beispiele für solche Systeme finden sich vor allem im Bereich der Dämpfung von schwingenden Systemen und der Regelung elektromagnetisch angetriebener Prozesse.

Es existieren jedoch dynamische Systeme, deren Eigenschaften eine zeitoptimale und genaue Regelung mit diskreten Aktionen nicht zulassen. In Bezug auf die Zeitoptimalität des Reglers ist die Verwendung diskreter Aktionen vor allem dann problematisch, wenn die Aktion in jedem Zeitschritt optimal an das dynamische System angepasst sein muss, um ein zeitoptimales Regelungsverhalten zu erzeugen. Beispiele für solche regelungstechnischen Problemstellungen finden sich in dynamischen Systemen mit direktem Einfluss der Eingriffsgröße auf die Systemvariablen ohne nennenswerte Beiträge von Trägheitsverhalten. Ein weiterer Nachteil diskreter Aktionen zeigt sich in ihrem Verhalten im Toleranzbereich der Regelung. Ein Beispiel an einem einfachen dynamischen System verdeutlicht dies:

Beispiel 4.1.1.1 Gegeben sei ein zeitdiskretes dynamische System der Form:

$$x_{t+1} = x_t + u_t, x_t, u_t \in \mathbb{R}$$

Ausgehend von einem beliebigen Startzustand $x_0 \in \mathcal{X}$ ist es das Regelungsziel, in einer minimalen Anzahl von Schritten dem Zielzustand $x = 0$ möglichst nahe zu kommen. In anderen Worten: Gesucht ist eine optimale Strategie, die das System, ausgehend von einem Startzustand x_0 , in einer minimalen Anzahl von Schritten in einen Zielbereich überführt, für den gilt: $|x| \leq \mu$, mit $\mu > 0$. Die Systemvariable des Systems sei beschränkt auf das Intervall $x \in [-x_{max}, x_{max}]$, ebenso gilt diese Beschränkung auch für die Eingriffsgröße $u \in [-x_{max}, x_{max}]$. Somit existiert in der Systemdynamik für jeden Zustand eine Eingriffsgröße, die das System in einem Schritt in den Zielzustand überführt.

Für die zeitoptimale Regelung dieses Systems in diesem Beispiel ist somit bekannt, dass eine Strategie existiert, die für jeden beliebigen Startzustand in einem Schritt in den Toleranzbereich der Regelung gelangt und dort verbleiben kann. Diese Strategie benötigt jedoch eine optimale diskrete Aktionsmenge $\mathcal{U} = (u_1, \dots, u_n)$, die sich für eine gegebene Breite des Zielbereiches μ spezifizieren lässt als:

$$\forall x_0 \exists u \in \mathcal{U} : u \in [-x - \epsilon, -x + \epsilon]$$

Eine solche Aktionsmenge lässt sich für dieses Beispiel generieren, indem auf der Systemvariablen Intervalle der Länge 2μ definiert werden, für die jeweils eine Aktion festgelegt werden kann.

Es folgt hieraus jedoch, dass selbst für dieses einfache dynamische System bei Reduzierung des zulässigen Fehlers bzw. der Breite des Toleranzbereiches durch μ , die Aktionsmenge immer mehr anwachsen muss, um die geforderte Genauigkeit unter einem zeitoptimalen Verhalten erreichen zu können. Geht die geforderte Fehlergrenze gegen Null, so benötigt man unendlich viele Aktionen aus dem zulässigen Bereich der Eingriffsgröße.

Selbst die Betrachtung dieses sehr einfachen Systems legt somit nahe, dass die Verwendung kontinuierlicher Aktionen, die jeden beliebigen Wert im zulässigen Bereich der Eingriffsgröße annehmen können, für die genaue Regelung eine notwendige Voraussetzung darstellen. In der Praxis kann es schwierig sein, eine ausreichende Menge an diskreten Aktionen zur Verfügung zu stellen, die eine zufriedenstellende Genauigkeit der Regelung des Systems zulässt. Dies gilt vor allem bei dem in dieser Arbeit betrachteten Problem der selbstlernenden Regler ohne Kenntnis der Systemgleichungen und der Regelung auf beliebige Führungsgrößen. Ein weiteres Beispiel, das diesen Zusammenhang aufzeigt, wurde in Kapitel 3.4.2 vorgestellt. In Abbildung 3.13 und 3.14 konnte gezeigt werden, dass durch geeignete Wahl der Kostenfunktion die Genauigkeit des Reglers erhöht werden kann, die Regelung jedoch aufgrund der fehlenden optimalen Aktion in Zyklen um den Zielpunkt der Regelung schwingen muss, um diese Genauigkeit im Mittel zu erreichen.

Durch die Erhöhung der Anzahl der diskreten Aktionen kann der zu regelnde Prozess feiner beeinflusst und somit auch die Regelgüte gesteigert werden. Das grundsätzliche Problem der fehlenden optimalen Aktionen für beliebige Zustände kann jedoch durch eine reine Erhöhung der Anzahl der diskreten Aktionen nicht gelöst werden. Zusätzlich ist zu beachten, dass eine Erhöhung der Anzahl der diskreten Aktionen ebenfalls Konsequenzen für die Effizienz des Lernverfahrens hat. Die Erhöhung der Anzahl der zur Verfügung stehenden diskreten Aktionen bedingt eine Verschlechterung der Lerngeschwindigkeit beziehungsweise eine Erhöhung der benötigten Anzahl der Lerniterationen des Verfahrens. Dies wiederum führt zu einer

unerwünschten Erhöhung der benötigten Interaktionsepisoden mit dem Prozess. Desweiteren bedeutet die Verwendung diskreter Aktionen bei mehrdimensionalen Eingriffsgrößen ein exponentielles Anwachsen der benötigten diskreten Aktionen. Dies ist zumindest dann der Fall, wenn jede Eingriffsgröße des Systems mit einer bestimmten Genauigkeit in der Menge der diskreten Aktionen repräsentiert sein soll. In diesem Fall muss jede mögliche Kombination der Werte, die eine Eingriffsgröße annehmen kann, über alle Eingriffsgrößen gebildet werden und der Aktionsmenge zugeschlagen werden.

4.1.2 Die nichtlineare Dynamik eines Unterwasserfahrzeuges

Für die Untersuchung der Eigenschaften der in diesem Kapitel diskutierten Verfahren wird im folgenden die Dynamik eines Unterwasserfahrzeuges betrachtet. An der Regelung dieses dynamischen Systems werden die Probleme, die die Verwendung diskreter Aktionen mit sich bringen, diskutiert und die vorgestellten Verbesserungen untersucht. Die Dynamik des Unterwasserfahrzeuges wurde hierfür so angepasst, dass ein idealer Benchmark für die Untersuchung zeitoptimaler Regler entsteht. Im Speziellen wurde das System so gewählt, dass seine Systemdynamik einen hohen Anteil an Nichtlinearität und nichtsymmetrische Eigenschaften über seinen Zustandsraum besitzt. Zudem muss der Regler die gewählte Eingriffsgröße in jedem Zeitschritt auf den aktuellen Zustand des Prozesses anpassen, um eine zeitoptimale Regelung zu erreichen. Das betrachtete System hat eine Systemvariable und eine Eingriffsgröße und erlaubt daher eine ideale Darstellung und Auswertung der verschiedenen Methoden.

Die Geschwindigkeit v des Unterwasserfahrzeuges im Wasser ist die einzige Systemvariable. Die Masse m des Fahrzeuges und der Widerstandsbeiwert c sind keine Konstanten, sondern repräsentieren als Ersatzmasse und Ersatzwiderstandsbeiwert die komplexe Dynamik, die sich bei einer Bewegung eines Körpers in einem Fluid ergibt. Die Eingriffsgröße des Systems u ist die als ideal angenommene Vorschubkraft, die zum Beispiel von einer Schraube durch eine entsprechende Drehzahl erzeugt werden kann. Diese wird durch einen Faktor k_{wirk} , dem Wirkungsgrad, auf die am Fahrzeug wirkende Vorschubkraft umgerechnet. Insbesondere wird durch den Wirkungsgrad ein Abriss der real wirkenden Vorschubkraft bei zu hohen Drehzahlen an der Schraube simuliert. Die das System beschreibenden Differenzialgleichungen ergeben sich zu:

$$\dot{v} = f(v, u) = \frac{u * k_{wirk}(v) - c(v) * v * |v|}{m(v)} \quad (4.1)$$

mit:

$$c(v) = 1.2 + 0.2 * \sin(|v|)$$

$$m(v) = 3.0 + 1.5 * \sin(|v|)$$

$$k_{wirk}(v) = \tanh [|(c * v * |v|) - u| - 30.0] * 0.1 * -0.5 + 0.5$$

In Abbildung 4.1 ist die Dynamik des Fahrzeuges als dreidimensionale Grafik dargestellt. Aufgetragen ist die sich ergebende Beschleunigung des Fahrzeuges über die aktuelle Geschwindigkeit und die gewählte Eingriffsgröße. Diese Systemdynamik hat mehrere für den Entwurf nichtlinearer Regler interessante Eigenschaften. Die Beschleunigung des Fahrzeuges in einem Zustand ist für jede Aktion unterschiedlich. Sie ist nicht symmetrisch, da der Wirkungsgrad des Antriebs ebenfalls nicht symmetrisch ist. Zusätzlich besitzt sie mehrere lokale Minima und Maxima in ihrer Beschleunigung, in denen sich ein optimierender Regler festsetzen kann, ohne die optimale Lösung zu finden. In Abbildung 4.2 ist ein Schnitt durch die Systemdynamik für

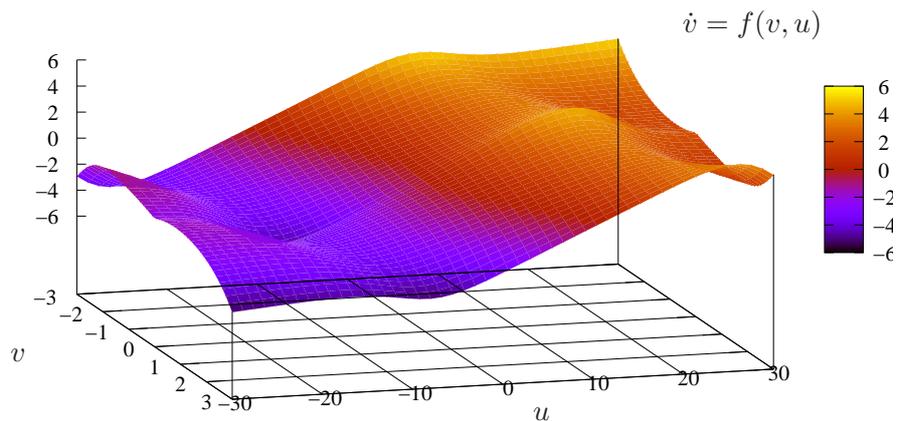


Abbildung 4.1: Die Systemdynamik des Unterwasserfahrzeuges mit erweiterter nichtlinearer Dynamik. Dargestellt ist die sich bei einer Geschwindigkeit und Anwendung einer Aktion ergebende Beschleunigung (Systemdynamik).

zwei unterschiedliche Geschwindigkeiten des Unterwasserfahrzeuges dargestellt. Für jede Geschwindigkeit ist die sich ergebende Beschleunigung des Fahrzeuges über der Eingriffsgröße aufgetragen.

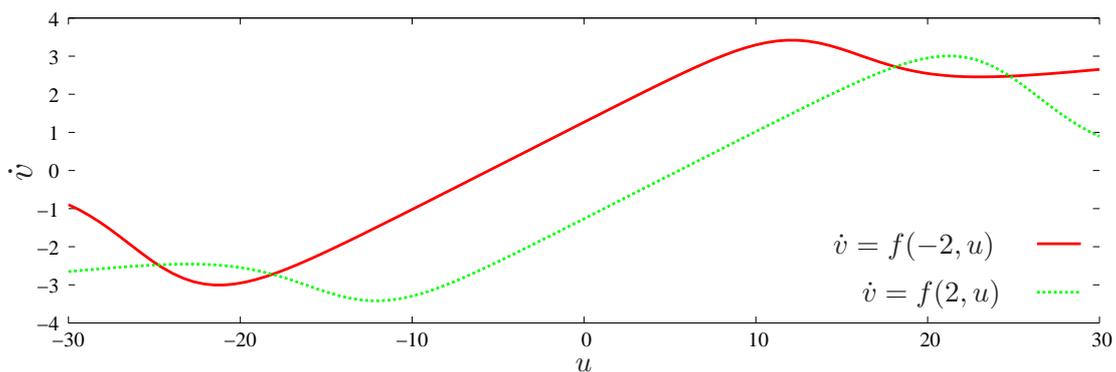


Abbildung 4.2: Zwei Schnitte durch die Systemdynamik des simulierten Unterwasserfahrzeuges bei Geschwindigkeit $v = 2 \frac{m}{s}$ und $v = -2 \frac{m}{s}$. Die sich ergebende Beschleunigung ist unsymmetrisch und optimale Beschleunigungen werden nicht durch die maximale Aktion erreicht.

Zu Gunsten der Auswertung der durchgeführten Untersuchungen und der Darstellung wird vorerst wieder von einer Festwertregelung mit unverändertem Sollwert ausgegangen. Die Regelungsaufgabe sei definiert als das zeitoptimale Abbremsen des Unterwasserfahrzeuges von beliebigen Geschwindigkeiten im Bereich $v \in [-5, 5]$ auf Geschwindigkeit $v = 0$. Diese Festwertregelung mit konstantem Sollwert dient im Verlauf des Kapitels dazu, die Nachteile der diskreten Aktionen und die Vorteile für die kontinuierlichen Aktionen herauszuarbeiten. Am Ende des Kapitels wird die Folgeregelung mit Zielwertvorgabe an diesem Beispiel zusätzlich demonstriert.

Für die Messung der Regelgüte wird ein Toleranzbereich der Regelung mit der Breite $\mu = 0.3$ betrachtet. Hierzu wird ein Regler auf einer Menge von 101 zufällig gezogenen Startgeschwindigkeiten getestet und die mittlere Anzahl der Schritte, \bar{N} , außerhalb dieses Zielbereiches angegeben.

Tabelle 4.1: Regelgüte der Vergleichsregler bei Regelung auf Geschwindigkeit 0 von 101 Startgeschwindigkeiten.

Regler	\bar{N}	\bar{e}_∞
optimal	14.74	0 (durch Design)
Bang-Bang	20.53	0 (durch Design)

Für die Versuche wird die Systemdynamik im Rechner simuliert. Die Simulation basiert auf der numerischen Integration mit dem Runge-Kutta-Verfahren 4. Ordnung auf einem Zeitintervall von $\Delta_t = 0.03$ Sekunden.

Vergleichsregler

Als Vergleichsregler für die betrachtete Regelungsaufgabe der Festwertregelung wurde analytisch ein optimaler Regler für das zeitdiskrete System entworfen. Für die betrachtete Regelungsaufgabe kann dieser bestimmt werden, indem für jede Geschwindigkeit die maximale Entschleunigung (negative Beschleunigung) in der Systemdynamik bestimmt wird. Der Entwurf dieses Reglers erlaubt somit ein ideales Regelverhalten, das den Sollwert $v = 0$ zeitoptimal und exakt erreicht. Diese optimale Regelstrategie ist in Abbildung 4.3 als Verlauf der optimalen Eingriffsgröße über der Geschwindigkeit aufgetragen. Für die optimale Strategie muss demnach für jede Geschwindigkeit eine bestimmte Aktion gewählt werden, die eine maximale Entschleunigung des Fahrzeuges bewirkt. Diese Strategie unterscheidet sich deutlich von anderen Strategien für vergleichbare Systeme, da ein einfaches Bang-Bang-Verhalten des Reglers nicht zu einer zeitoptimalen Strategie führt. Zusätzlich wurde ein erweiterter, idealer Bang-Bang Regler entworfen. Dieser besitzt außerhalb eines Toleranzbereiches von $\pm 0.3 \frac{m}{s}$ um das Regelungsziel die Aktionen -30 und 30. Innerhalb des Toleranzbereiches wählt er die analytisch optimale Aktion der idealen Strategie. In Tabelle 4.1 ist die Regelgüte der Vergleichsregler für die betrachtete Regelungsaufgabe zusammengefasst. Durch das Design der Regler unter Kenntnis der Systemgleichungen können beide Vergleichsregler die bleibende Regelabweichung vollständig eliminieren. Der Bang-Bang-Regler erzielt durch die Anwendung der maximalen Aktion eine Regelgüte von $\bar{N} = 20.53$ im Zeitkriterium. Dies zeigt, dass die Anwendung von maximalen Aktionen nicht zu einer zufriedenstellenden Regelgüte führt und sich deutlich von der idealen Strategie mit $\bar{N} = 14.74$ unterscheidet.

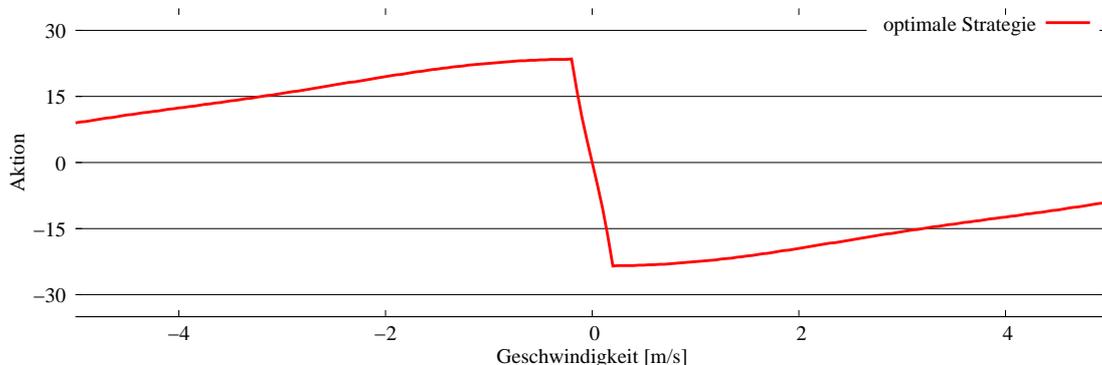


Abbildung 4.3: Die analytisch erzeugte optimale Strategie um das Unterwasserfahrzeug von beliebigen Anfangsgeschwindigkeiten zeitoptimal auf Geschwindigkeit Null zu regeln.

Tabelle 4.2: Parameter für die Lernversuche am Unterwasserfahrzeug mit NFQ

Länge Interaktionsepisode	50 Schritte
Länge Testepisode	100 Schritte
Startzustände für Training	zufällig aus $[-5, 5]$
Startzustände für Test	101 feste Geschwindigkeiten aus $[-5, 5]$
Kostenfunktion	\tanh^2 mit $x^e = 0$, $\mu = 0.3$, $c = 0.01$
Zielbereich für Auswertung	$[-0.3, 0.3]$
Topologie \mathcal{Q}	(2-10-10-1)

Tabelle 4.3: Die Regelgüte des NFQ-Verfahrens am Unterwasserfahrzeug bei Verwendung verschiedener Anzahl von diskreten Aktionen.

Regler	\bar{N}	\bar{e}_∞
NFQ 3 diskrete Aktionen	20.53	0.043
NFQ 7 diskrete Aktionen	15.70	0.022
NFQ 13 diskrete Aktionen	15.68	0.016
NFQ 61 diskrete Aktionen	15.45	0.002

4.1.3 NFQ am Beispiel des Unterwasserfahrzeuges

Die Modellierung des selbstlernenden Reglers mit NFQ für das beschriebene regelungstechnische Problem basiert auf der Stabilisierung der geschlossenen Systemdynamik in seinem Zustandsraum, um die Geschwindigkeit 0. Der Zustand dieses Problems ist eindimensional und enthält die Systemvariable, welche die Geschwindigkeit des Unterwasserfahrzeuges v repräsentiert, welche gleichzeitig die Regelabweichung darstellt. Die Aufgabe ist es, diesen Zustand auf eine zeitoptimale Weise von beliebigen Ausgangsgeschwindigkeiten in einen Toleranzbereich um die Geschwindigkeit Null zu bringen. Für das NFQ-Verfahren wird eine neuronale Q-Funktion mit einer (2-10-10-1) Topologie¹ verwendet. Das Lernen mit dem NFQ-Verfahren wird mit einem explorativen Lernprozess auf Interaktionsepisoden mit einer Länge von 100 Zeitschritten und einer zufälligen Explorationsrate von 30%, durchgeführt. Nach jeder Iteration des Lernverfahrens wird der Regler auf 101 Testepisoden mit einer Länge von 100 Schritten angewendet. Jede Trainings- und Testepisode beginnt dabei aus einem zufälligen Startzustand $v \in [-5, 5]$. Als Kostenfunktion wird die \tanh^2 -Kostenfunktion mit $x^d = 0$ und $\mu = 0.3$ eingesetzt. Die Versuche werden auf den ersten 300 Iterationen des Lernverfahrens ausgewertet. Jeder dieser Versuche wird 10 Mal unabhängig wiederholt und die erzielten Kennzahlen für die Regelgüte gemittelt. Die Parameter für die Lernversuche sind in Tabelle 4.2 zusammengefasst. Für die Versuche mit NFQ wurden 4 verschiedene Aktionsmengen U mit einer unterschiedlichen Anzahl an diskreten Aktionen betrachtet. Die kleinste Aktionsmenge mit 3 diskreten Aktionen besteht aus $U_3 = \{-30, 0, 30\}$. Die weiteren Aktionsmengen U_7 , U_{13} und U_{61} wurden durch das symmetrische Einfügen von Zwischenwerten in der Aktionsmenge erzeugt. In Tabelle 4.3 sind die durchschnittlichen Ergebnisse über 10 unabhängige Lernversuche für jede der Aktionsmengen angegeben. Die statistischen Auswertung der Versuche sind in Abbildung 4.4 mit Mittelwert, minimalem und maximalem Wert der Kennzahl des Zeitkriteriums \bar{N} über

¹Die Wahl der Topologie der neuronalen Approximation der Q-Funktion entspricht wiederum einem Standardwert, mit dem die Leistungsfähigkeit der Verfahren gegeneinander vergleichbar gemacht wird und repräsentiert keine optimierte Lösung durch intensive Parametersuche.

die 10 unabhängigen Versuche aufgetragen. Ein mit NFQ und 3 diskreten Aktionen gelernter

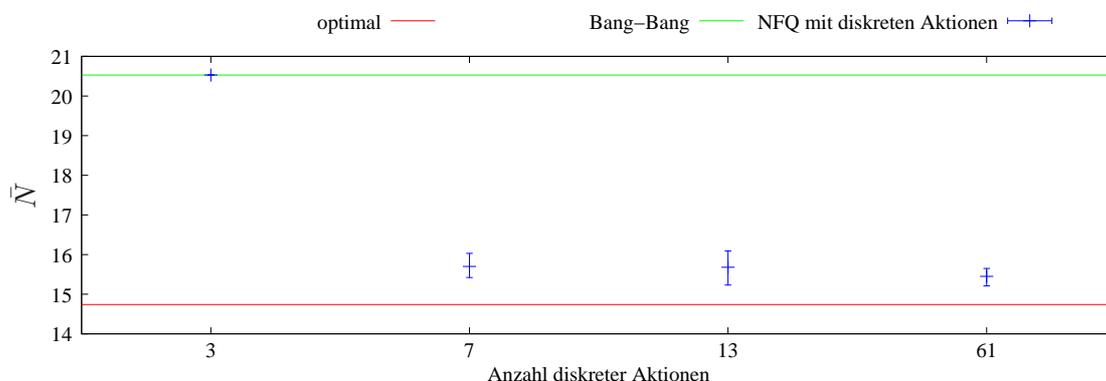


Abbildung 4.4: Erzielte Regelgüte in Bezug auf das zeitoptimale Verhalten der Regler \bar{N} über eine Reihe von 10 unabhängigen Versuchen für jede der definierten Aktionsmengen. Die Auswertung der Lernversuche mit diskreten Aktionen zeigt, dass die Steigerung der Anzahl der diskreten Aktionen eine Verbesserung der Regelgüte ergibt, jedoch nicht zur optimalen Strategie führt.

Regler ist in der Lage, das Regelungsproblem zu lösen und erzeugt eine im Toleranzbereich der Regelung zulässige bleibende Regelabweichung \bar{e}_∞ . Allerdings kann diese nicht beliebig klein werden, was ein Schwingen des Systems um den Zielpunkt erzeugt. Dies wird durch den relativ hohen Wert im Gütekriterium \bar{e}_∞ deutlich, das über die letzten 50 Schritte jeder Testepisode berechnet wird. Der gelernte Regler besitzt bezüglich des Zeitverhaltens außerhalb des Zielbe-

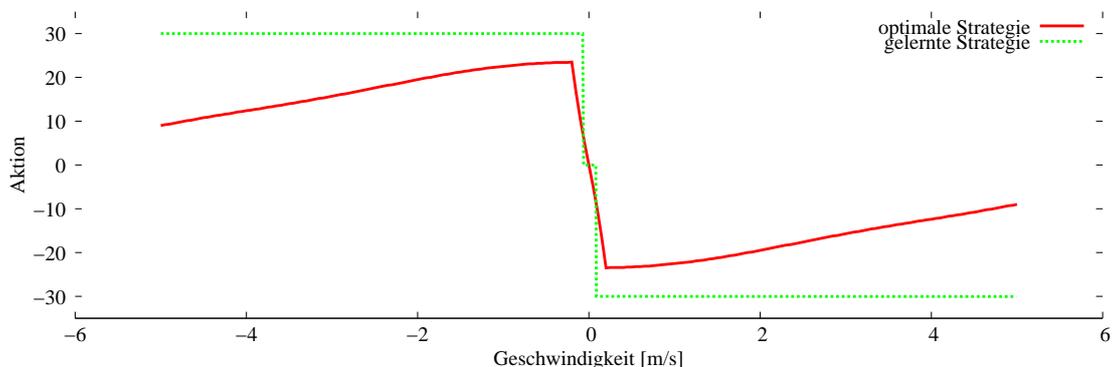


Abbildung 4.5: Die gelernte Strategie mit 3 diskreten Aktionen. Diese entspricht ausserhalb des Toleranzbereiches dem Verhalten des Bang-Bang Reglers. Im Vergleich zu diesem stehen dem Regler jedoch nur 3 Aktionen zur Verfügung, während der Bang-Bang-Regler im Toleranzbereich die analytisch optimale Aktion wählt. Die Regelung ist in der Lage den Toleranzbereich der Regelung um die Zielgeschwindigkeit von Null zu erreichen. Da die zur Verfügung gestellte Aktionsmenge nicht ausreichend ist, wird die Güte des Reglers in Bezug auf die Genauigkeit und die Zeitoptimalität jedoch nicht zufriedenstellend.

reichs exakt das Verhalten des Bang-Bang-Reglers. Dieses Verhalten wird in allen 10 Versuchen aus der Interaktion mit dem System erlernt. Die sich ergebende Strategie im Vergleich zur optimalen Strategie ist in Abbildung 4.5 über der Geschwindigkeit des Unterwasserfahrzeuges aufgetragen. Bei Erhöhung der Anzahl der diskreten Aktionen wird die optimale Strategie immer mehr angenähert. In Abbildung 4.6 und 4.7 sind die mit 7 und 13 diskreten Aktionen gelernten Strategien im Vergleich zur optimalen Strategie dargestellt. Durch die Erhöhung der

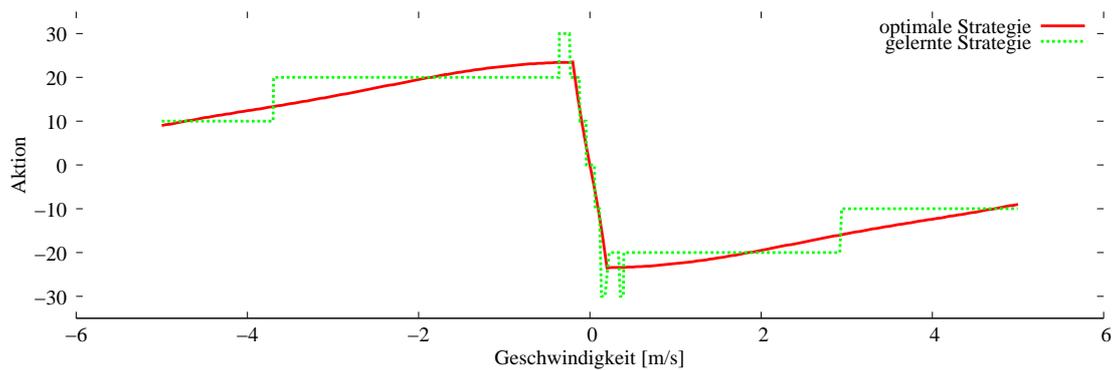


Abbildung 4.6: Die gelernte Strategie mit 7 diskreten Aktionen. Die gelernte Strategie nähert die optimale Strategie mit den zur Verfügung stehenden Aktionen an.

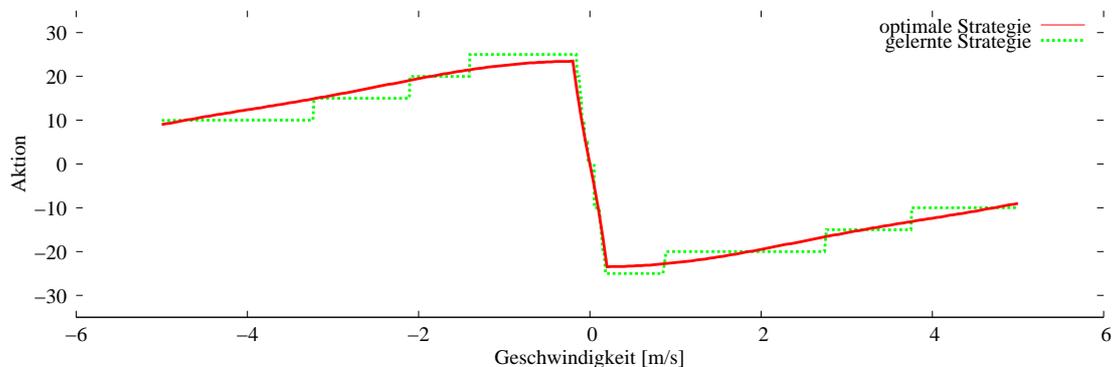


Abbildung 4.7: Die gelernte Strategie mit 13 diskreten Aktionen. Die gelernte Strategie nähert die optimale Strategie mit den zur Verfügung stehenden Aktionen an.

Anzahl der Aktionen kann die Genauigkeit des Reglers drastisch verbessert werden. Die Zyklen um den eigentlichen Zielpunkt werden mit der Erhöhung der Anzahl der Aktionen immer kleiner und das Fehlerkriterium nimmt immer weiter ab. Bei 61 Aktionen kann der Regler in Bezug auf die Genauigkeit immer noch deutlich von der Erhöhung der Anzahl der diskreten Aktionen profitieren und erzeugt eine bleibende Regelabweichung von $\bar{e}_\infty = 0.002 \frac{m}{s}$.

In Abbildung 4.8 ist der Verlauf des Zeitkriteriums über die Iterationen des NFQ-Verfahrens für die verschiedenen Versuche dargestellt. Die Abbildung stellt die Anzahl der benötigten Iterationen des Lernverfahrens in Bezug zu der erreichten Qualität des Reglers im Zeitkriterium \bar{N} . Dargestellt sind die im Durchschnitt benötigten Iterationen, bis eine bestimmte Schranke des Zeitkriteriums zum ersten Mal auf der Testmenge unterschritten wird. Zusätzlich ist die minimale und maximale Anzahl der Iterationen dargestellt, die auf den 10 unabhängigen Versuchen benötigt wurde. Lernversuche, welche die Schranke nicht unterschritten haben, wurden in die Auswertung nicht aufgenommen. Daher sind in der Abbildung auch Werte für Lernversuche eingetragen, die im Mittel über die 10 durchgeführten Versuche die entsprechende Schranke nicht unterschritten haben (vgl. Tabelle 4.3). Im Vergleich der Versuche mit 13 und 61 Aktionen in Abbildung 4.8 wird die Auswirkung der Erhöhung der Anzahl der Aktionen auf das Lernverfahren deutlich. Mittlere Fehlerschranken werden mit einer ähnlichen Anzahl von Iterationen erreicht. Für kleine Schranken benötigt das Verfahren mit erhöhter Anzahl von Aktionen jedoch mehr Zeit, um diese zu unterschreiten. Die Erhöhung der Aktionen ist somit

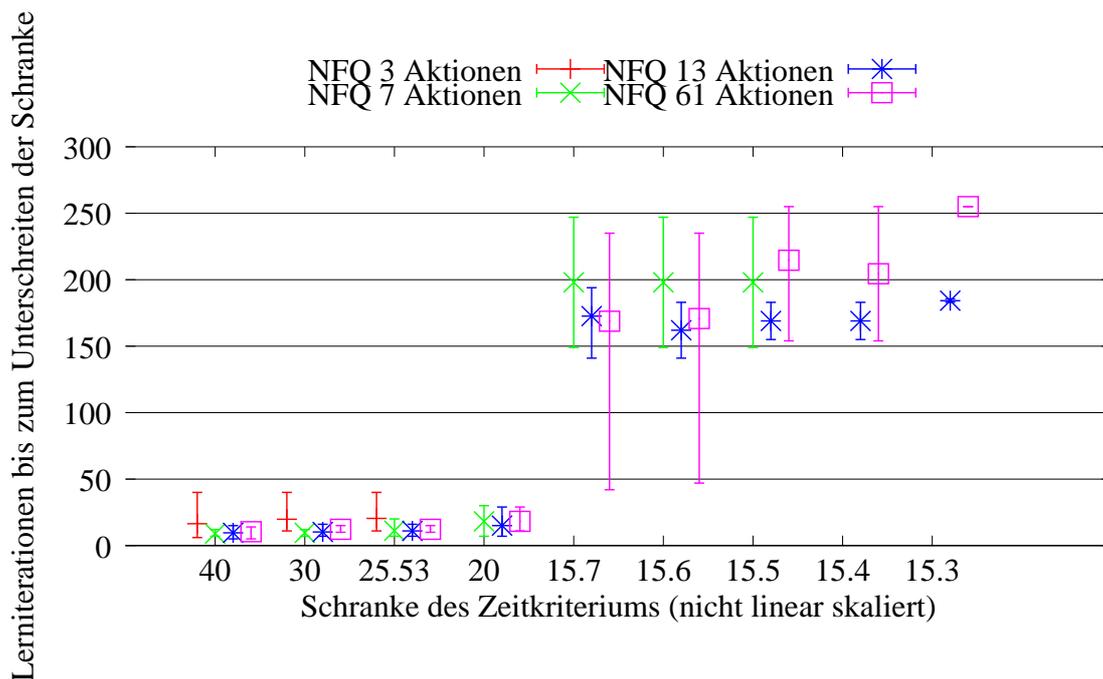


Abbildung 4.8: Die Auswertung der Lerngeschwindigkeit des NFQ-Verfahrens mit diskreten Aktionen. Für unterschiedliche Anzahl von Aktionen sind die durchschnittliche Anzahl der Lerniterationen angegeben, nach denen eine bestimmte Schranke des Zeitkriteriums zum ersten mal unterschritten wird. Die Werte sind gemittelt über die 10 durchgeführten Läufe. Zusätzlich ist das Minimum und Maximum aufgetragen.

bis zu einem gewissen Punkt sinnvoll, führt dann aber dazu, dass sich der Optimierungsprozess verlangsamt.

4.1.4 NFQ und kontinuierliche Aktionen

Das NFQ-Verfahren ist in seiner Grundform beschränkt auf die Verwendung diskreter Aktionen. Diese Beschränkung ist ein zentraler Nachteil des Verfahrens beim Einsatz als Optimierungsprozess für die Erstellung von selbstlernenden Reglern für allgemeine Systeme. Wie in diesem Abschnitt diskutiert wird, erfüllen die meisten Teilschritte des Algorithmus und seiner Formulierung bereits die Erfordernisse, welche die Verwendung kontinuierlicher Aktionen mit sich bringen. Durch die Verwendung der kontinuierlichen Aktionen ergibt sich jedoch ein zentraler Punkt in der Grundformel zur Aktualisierung der Q-Werte, die, unter Verwendung kontinuierlichen Aktionen nicht mehr durchführbar ist. Die Verwendung der neuronalen Netze in NFQ bringt hier jedoch zentrale Vorteile mit sich, die auf eine generelle Lösbarkeit des Problems hinweisen.

Die Verwendung kontinuierlicher Aktionen macht die Ersetzung der diskreten Aktionsmenge $U = \{u^0, \dots, u^h\}$, $u^i \in \mathbb{R}^m$, bestehend aus h diskreten, m -dimensionalen Aktionen, durch einen m -dimensionalen Aktionsraum \mathcal{U} notwendig. Für den Optimierungsprozess muss dieser Aktionsraum beschränkt sein. Die diskrete Aktionsmenge des NFQ wird somit ersetzt durch

eine kontinuierliche Aktionsmenge:

$$\mathcal{U} = \left\{ u = \begin{pmatrix} u_1 \\ \dots \\ u_m \end{pmatrix} \in \mathbb{R}^m : u_i \in [u_i^{\min}, u_i^{\max}] \right\} \quad (4.2)$$

Dies definiert eine Teilmenge des \mathbb{R}^m . Die Beschränkung der einzelnen Dimensionen des Aktionsraumes u_i^{\min} und u_i^{\max} sind anwendungsspezifische Werte, die für regelungstechnische Prozesse durch die physikalische oder technische Beschränkung der Eingriffsgrößen gegeben sind. Die in Kapitel 3 gegebenen Anpassungen für regelungstechnische Fragestellungen können ohne Änderungen auf kontinuierliche Aktionen angewandt werden. Die Verwendung der differenzierbaren Kostenfunktion, in Kombination mit kontinuierlichen Aktionen, ist ihrer Grundidee zufolge noch effektiver. Auch die Korrektur des systematischen Fehlers in der Q-Funktion und die Formulierung der Regelung unter Führungsgröße lassen sich direkt ohne Änderung übertragen.

Ein zentraler Teil des NFQ-Grundverfahrens ist die Approximation der Q-Funktion in Form eines mehrschichtigen neuronalen Netzes $Q(x, u, w_q)$. An den Eingaben des Netzes liegen die Zustandsvariablen und die Eingriffsgrößen an. Dies hat in diesem Kontext den Vorteil, dass nicht nur eine Generalisierung über den Zustandsraum, sondern auch eine Generalisierung über den Aktionsraum stattfindet. Diese Generalisierung führt typischer Weise zu einem sehr effizienten Lernvorgang, bei dem ein Agent in einem explorativen Lernprozess nicht alle Aktionen ausführen muss, um den Q-Wert in einem Zustand ausreichend zu schätzen. Die Approximationsleitung des neuronalen Netzes führt jedoch auch dazu, dass die Q-Funktion prinzipiell für beliebige Aktionen einen Wert enthält. Diese Zwischenwerte sind beim Lernen nicht in der Datenbasis repräsentiert, da sie nicht in der diskreten Aktionsmenge enthalten sind. Somit werden sie auch im Lernschritt nicht verwendet. Ihre Werte sind ausschließlich ein Resultat der Generalisierung des neuronalen Netzes und basieren nicht auf den direkten Eigenschaften des MDPs. Dennoch enthalten sie implizit Informationen, die für regelungstechnische Problemstellungen ausgewertet werden können und einen Lernvorgang mit kontinuierlichen Aktionen unterstützen. Für einen Agenten mit kontinuierlichen Aktionen reichen somit in einem Zustand wenige Beispiele von Aktionen aus, um die Q-Funktion zu lernen.

Die grundsätzliche Idee der Erweiterung des NFQ-Verfahrens auf die Verwendung kontinuierlicher Aktionen basiert auf dieser Generalisierungsleistung der neuronalen Approximation der Q-Funktion im Zustands-Aktions-Raum. Wenige Beispiele von beobachteten Zustands-Aktions-Paaren aus der Interaktion mit dem System sollen ausreichend sein, um eine gute Schätzung der Kostenfunktion und eine näherungsweise optimale Strategie mit kontinuierlichen Aktionen zu berechnen.

Beispiel am Unterwasserfahrzeug

Die Generalisierungsleistung der neuronalen Q-Funktion wird am Beispiel der Regelung des Unterwasserfahrzeuges mit NFQ unter Verwendung diskreter Aktionen deutlich. Hierzu wird die neuronale Q-Funktion eines mit 13 diskreten Aktionen eingelernten Reglers betrachtet. Dieser Regler erzeugt einen Wert $\bar{N} = 15.83$ für das Zeitkriterium ². Die Generalisierungsleistung der neuronalen Approximation der Q-Funktion zeigt sich nun, wenn diese mit einer höheren Anzahl von diskreten Aktionen gierig ausgewertet wird. Auf einer Menge von 61 diskreten Aktionen erzeugt die so entstandene Strategie einen Wert $\bar{N} = 15.71$. Die Güte der

²Der Regler wurde zufällig aus den 10 durchgeführten Versuchen mit 13 diskreten Aktionen ausgewählt.

Regelung lässt sich somit ausschließlich basierend auf der Generalisierungsleistung des neuronalen Netzes verbessern. Die Zwischenwerte der Aktionen, die bei der Auswertung verwendet wurden, können vom Netz in gewisser Weise interpretiert werden und zur Verbesserung der Strategie beitragen. In Abbildung 4.9 ist die originale Strategie mit 13 Aktionen und die sich

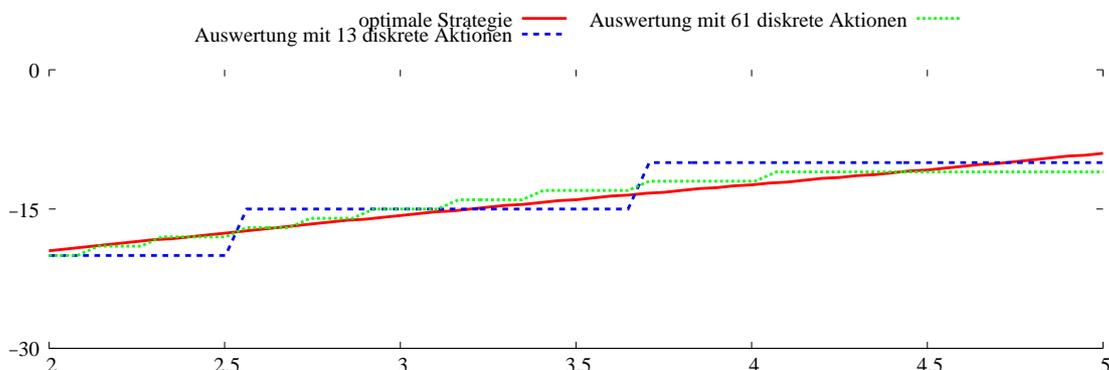


Abbildung 4.9: Der Vergleich zweier Strategien, die sich durch die Auswertung derselben neuronalen Q-Funktion ergeben, als Ausschnitt im Zustandsraum. Die Q-Funktion wurde mit 13 diskreten Aktionen eingelernt. Wird diese, ohne neu zu lernen, mit 61 diskreten Aktionen abgetastet, so wird die Generalisierungsleistung des neuronalen Netzes deutlich. Die Strategie mit 61 diskreten Aktionen ist an den meisten Stellen eine feinere Approximation der optimalen Strategie, ohne dass die diskreten Aktionen beim Lernen verwendet wurden.

ergebende Strategie bei Auswertung mit 61 Aktionen gegen die optimale Strategie auf einem Ausschnitt des Zustandsraumes des Unterwasserfahrzeuges aufgetragen. Deutlich ist zu sehen, dass die Strategie mit 61 Aktionen eine Verfeinerung der originalen Strategie darstellt. Diese ist nicht an allen Stellen optimal, da die Approximation an diesen Stellen keine Trainingsdaten besitzt. Um die Strategie an diesen Stellen anzupassen muss ein Lernverfahren in der Lage sein beliebige, kontinuierliche Aktionen zu wählen, um Übergänge an diesen Stellen zu beobachten und die Approximation der Wertfunktion anzupassen.

4.1.5 Diskussion

Die Verwendung kontinuierlicher Aktionen für die Erstellung selbstlernender Regler auf Basis des NFQ ist ein vielversprechender Ansatz. Speziell die guten Generalisierungsfähigkeiten der neuronalen Netze und die Repräsentation der Q-Funktion stellen sich als geeignet dar, um mit der gegebenen Modellierung der selbstlernenden Regler kontinuierliche Aktionen zu verwenden.

Die problematische Stelle für die Verwendung kontinuierlicher Aktionen in NFQ ist die Berechnung des minimalen Q-Wertes in der neuronalen Q-Funktion. Diese wird an zwei Stellen im Verfahren benötigt. Zum einen für die Aktualisierung der Q-Werte in der Lernvorschrift des Q-Lernens. Zum anderen in der gierigen Auswertung der Q-Funktion zur Berechnung der aktuellen Strategie.

$$\min_{u \in \mathcal{U}} Q(x, u, w_q) \quad \text{bzw.} \quad \operatorname{argmin}_{u \in \mathcal{U}} Q(x, u, w_q) \quad (4.3)$$

Eine vollständige Suche über die zur Verfügung stehenden Aktionen ist bei kontinuierlichen Aktionen nicht mehr möglich.

Die Herleitungen der Verfahren zur Erweiterung des NFQ auf kontinuierliche Aktionen werden mit der Definition der Aktionsräume als Teilmenge des \mathbb{R}^m gegeben, bei denen jede Dimension des Aktionsraumes auf ein bestimmtes Intervall beschränkt ist. Dies ist eine Annahme, die für die meisten physikalischen und technischen Prozesse realistisch ist. Aus diesem Grund wird in den vorgestellten Verfahren nur auf diese Randbedingungen bei der Optimierung eingegangen. Für Spezialfälle mit der Beschränkung einer Aktionsdimension auf mehrere Intervalle oder zustandsabhängige Aktionsintervalle für eine Dimension können die Verfahren erweitert werden. Diese zusätzlichen Randbedingungen können analog zu den hier behandelten Randbedingungen in die Verfahren integriert werden.

4.2 Kontinuierliche Aktionen durch Suche in der Q-Funktion

Das erste Verfahren zur Integration kontinuierlicher Aktionen in NFQ basiert direkt auf den im vorigen Abschnitt diskutierten Eigenschaften der Generalisierungsleistung und auf der Repräsentation der neuronalen Q-Funktion. In diesem Sinne wurde es hauptsächlich entwickelt, um eine erweiterte Untersuchung dieser Eigenschaften durchzuführen. Das resultierende Verfahren basiert auf einem Gradientenabstieg auf der neuronalen Q-Funktion und wird daher im Folgenden als NFQ mit Gradientenabstieg auf der Q-Funktion (NFQG) bezeichnet.

4.2.1 Minimums-Suche mittels Gradientenabstieg

Die Repräsentation der Q-Funktion mittels einem neuronalen Netz $Q(x, u, w_q)$ besitzt den Vorteil, dass über den gesamten Zustands-Aktionsraum generalisiert werden kann. Zusätzlich bietet diese Approximation der Zustands-Aktions-Wertfunktion in einem mehrschichtigen neuronalen Netz einen weiteren entscheidenden Vorteil. Die neuronale Repräsentation der Q-Funktion stellt eine Struktur dar, die Informationen über den Gradienten der Q-Funktion an beliebigen Stellen des Eingaberaumes enthält. Aus der Theorie der neuronalen Netze ist bekannt, dass ein neuronales Netz eine differenzierbare Funktion darstellt, deren Ableitung an beliebiger Stelle durch Vor- und Zurückpropagieren, mit Hilfe des Backpropagation-Schemas berechnet werden kann (siehe 2.2.4). In diesem Fall kann somit durch Vorwärtspropagieren des Tupels (x, u) der Q-Wert an dieser Stelle bestimmt werden. Nach dem Zurückpropagieren liegt an den Eingabeneuronen des Netzes die partielle Ableitung der Q-Funktion nach der entsprechenden Eingabe an. Hierzu wird am Ausgabeneuron des Netzes nicht das Fehlersignal angelegt, sondern direkt die Aktivierung des Ausgabeneurons durch das Netz zurückpropagiert. Man berechnet somit die partielle Ableitung des Netzes an einem Punkt im Zustands-Aktions-Raum nach den einzelnen Eingaben:

$$\nabla Q(x, u) = \left(\frac{\partial Q(x, u)}{\partial x_1}, \dots, \frac{\partial Q(x, u)}{\partial x_n}, \frac{\partial Q(x, u)}{\partial u_1}, \dots, \frac{\partial Q(x, u)}{\partial u_m} \right) \quad (4.4)$$

Die Aufgabe der Bestimmung des Minimum in der Q-Funktion an einer bestimmten Stelle x des Zustandsraumes kann mit Hilfe des Gradienten aus der neuronalen Q-Funktion somit als Suche mit Gradientenabstieg realisiert werden. Da in diesem Fall nur der partielle Gradient der Q-Funktion nach dem Aktionsvektor benötigt wird, soll dieser vereinfachend als $\nabla Q_x(u)$ geschrieben werden. Die Gewichte w_q des Netzes Q und der Zustand, an dem das Minimum gesucht wird, sind in diesem Fall fest. Es wird nach einer Aktion u^* gesucht die das Q-Netz im Zustand x minimiert: $u^* = \operatorname{argmin}_u Q_x(u)$. Bei der Suche mit dem Gradientenabstiegsverfahren wird die minimale Aktion in einem Zustand, ausgehend von einer Aktion u^0 , über

ein iterative Suche mit Anpassung $u^{k+1} = u^k + \Delta u^k$ durchgeführt. Das Standardverfahren verwendet dabei eine konstante Lernrate α , womit sich die Änderung in einem Suchschritt zu $\Delta u^k = -\alpha \nabla Q_x(u^k)$ ergibt.

Durch Gradientenabstieg mit Resilient Propagation (RProp, siehe 2.2.3) kann die Suche der optimalen Aktion weiter verbessert werden. Der Vorteil durch die Verwendung von RProp ist im Vergleich zum Standardverfahren, dass die Lernrate nicht für jedes Problem aufwändig bestimmt werden muss, sondern die in der Literatur angegebenen Standardparameter für die Optimierung sehr robust arbeiten und sich während des Suchvorganges adaptieren.

Da die verwendete Repräsentation der Q-Funktion als neuronales Netz grundsätzlich den Wertebereich für die Aktionen nicht beschränkt, muss die Minimumssuche an den Randstellen des Aktionsraumes künstlich beschränkt werden. Tritt dieser Fall auf, d.h. $u^{k+1} \notin \mathcal{U}$ mit einem (oder mehreren) $u_i^{k+1} \notin [u_i^{min}, u_i^{max}]$, so wird ein Korrekturschritt notwendig. Bei diesem Korrekturschritt wird der aktuelle Wert von u^{k+1} zurückprojiziert in den Aktionsraum \mathcal{U} . Dies bedeutet, dass diejenigen Dimensionen des Aktionsvektors u_i^{k+1} , die ihren zulässigen Bereich $[u_i^{min}, u_i^{max}]$ verlassen, auf die nächst liegende Intervallgrenze gesetzt werden. Zusätzlich werden die Parameter des RProp-Verfahren für diese Dimensionen des Aktionsvektors zurückgesetzt.

In Algorithmus 10 ist diese Suche, ausgehend von einem Startpunkt u^0 , als Pseudocode zusam-

Algorithmus 10 GRADDESCMIN(\mathcal{Q}, x, u^0) Suche nach lokalem Minimum in neuronaler Q-Funktion

Require: Zustand x , neuronale Q-Funktion $\mathcal{Q}(x, u, w_q)$, Ausgangspunkt für die Suche u^0

Ensure: q_{min} lokales Minimum über $u \in \mathcal{U}$ von $\mathcal{Q}(x, u, w_q)$ ausgehend von u^0

Ensure: $u_{min} \in \mathcal{U} \subset \mathbb{R}^m$ Aktion an der das lokale Minimum gefunden wurde

$q_{min} \leftarrow \infty$

$k \leftarrow 1$

RPropGradDesc_init

repeat

$q = forward_propagate(\mathcal{Q}, x, u^{k-1})$

if $q < q_{min}$ **then**

$q_{min} = q ; u_{min} = u^{k-1}$

end if

$(\frac{\partial q}{\partial x_1}, \dots, \frac{\partial q}{\partial x_n}, \frac{\partial q}{\partial u_1}, \dots, \frac{\partial q}{\partial u_m}) \leftarrow backward_propagate(\mathcal{Q}, 1)$

$u^k \leftarrow RPropGradDesc(u^{k-1}, (\frac{\partial q}{\partial u_1}, \dots, \frac{\partial q}{\partial u_m}))$

bound(u^k)

$k \leftarrow k + 1$

until u konvergiert

return q_{min}, u_{min}

mengenfasst. Die Funktionen $forward_propagate(\mathcal{Q}, x, u)$ und $backward_propagate(\mathcal{Q}, 1)$ realisieren das Vorwärts- und Rückwärtspropagieren der Werte durch das neuronale Netz \mathcal{Q} . $RPropGradDesc(u, \nabla Q_x(u))$ bezeichnet den RProp-Gradientenabstiegsschritt und die Funktion bound(u) das Zurückprojizieren der Aktion in den zulässigen Wertebereich sowie das Zurücksetzen der Parameter des RProp-Verfahrens bei Verlassen des Aktionsraumes.

Ausgehend von einer Aktion u^0 kann so mit Hilfe des Gradientenabstieges zumindest ein lokales Minimum der Q-Funktion und die zugehörige Aktion, die die Q-Funktion an dieser Stelle

minimiert, gefunden werden.

4.2.2 NFQG: Ein Verfahren für kontinuierliche Aktionen

Die hier vorgestellte Grundidee zum Auffinden geeigneter Minima in einer neuronalen Q-Funktion kann für die erste Erweiterung des NFQ-Verfahrens für kontinuierliche Aktionen benutzt werden. Hierzu wird die Minimumssuche über diskrete Aktionen in NFQ ersetzt durch die Minimumssuche über kontinuierliche Aktionen durch Gradientenabstieg in der Q-Funktion. Wie in Abbildung 4.10 dargestellt, ändert dies für das gesamte Verfahren ausschließlich die gierige Auswertung der Q-Funktion, welche zur Bestimmung des minimalen Q-Wertes bzw. der Strategie notwendig ist. Im Unterschied zu NFQ wird nun nicht mehr eine Menge von

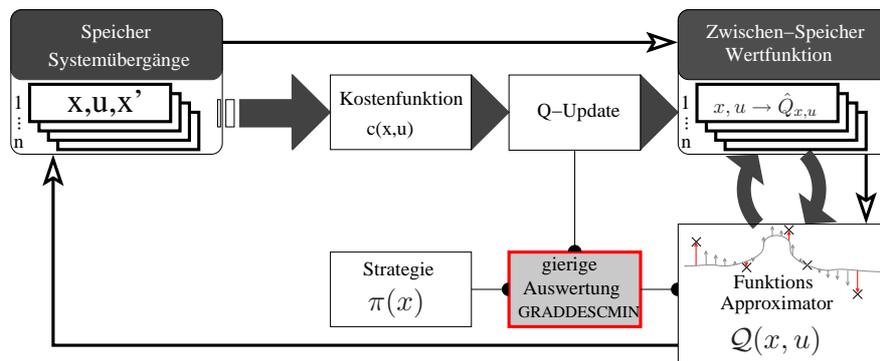


Abbildung 4.10: Der NFQG Algorithmus gehört zu den Neural Fitted Verfahren. Im Unterschied zu NFQ sind die Aktionen kontinuierlich. Die Strategie ergibt sich aus der gierigen Auswertung der Q-Funktion.

diskreten Aktionen betrachtet, sondern ein kontinuierlicher Aktionsraum \mathcal{U} mit reellwertigen Aktionsvektoren. Wie bei NFQ wird die durch Interaktion mit dem System gesammelte Erfahrung als Menge \mathcal{D} von beobachteten Übergängen der Form $d = (x, u, x', c)$ gespeichert. Auf dieser Menge von gespeicherten Übergängen wird die Aktualisierung der Q-Funktion in einem Lernschritt durchgeführt, die für jeden Übergang eine Minimumssuche mit Gradientenabstieg in der Q-Funktion benötigt. Der Lernprozess ist ebenfalls wie bei NFQ in mehreren Formen möglich, insbesondere auch in Form eines explorativen Lernprozesses, der mit einer zufälligen Q-Funktion und leerer Datenbasis beginnt. Bei diesem wird in jedem Zeitschritt zur Auswertung der Strategie wiederum eine Minimumssuche mit Gradientenabstieg in der Q-Funktion benötigt.

Die Verwendung multipler Startpunkte in der Minimum-Suche

Die gradientenbasierte Minimumssuche in der Q-Funktion ist abhängig von einer Aktion u_0 als Startpunkt für den Gradientenabstieg. Speziell zu Beginn des Lernvorganges ist zu erwarten, dass die Q-Funktion viele lokale Minima enthält, in denen das Suchverfahren enden wird, ohne ein evtl. vorhandenes globales Minimum in der Q-Funktion zu finden.

Um dieses Problem in der Praxis handhabbar zu machen, wird die Suche von mehreren Aktionen aus gestartet und das kleinste gefundene Minimum der Q-Funktion als Näherung des globalen Minimums verwendet. In Algorithmus 11 ist die Vorgehensweise zum Auffinden einer möglichst guten Annäherung des globalen Minimums in der Q-Funktion zusammengefasst.

Algorithmus 11 GRADDESCMIN(\mathcal{Q}, x) Minimum-Suche mit multiplen Startpunkten**Require:** Zustand x , neuronale Q-Funktion $\mathcal{Q}(x, u, w_q)$ **Ensure:** $q_{min} = \min_{u \in \mathcal{U}} \mathcal{Q}(x, u, w_q)$ möglichst global**Ensure:** $u_{min} = \operatorname{argmin}_{u \in \mathcal{U}} \mathcal{Q}(x, u, w_q) : u \in \mathcal{U} \subset \mathbb{R}^m$ möglichst global $q_{min} \leftarrow \infty$ **for** $t = 1, \dots, T$ **do** init $u^0 \in \mathcal{U} \subset \mathbb{R}^m$ $q, u = \text{GRADDESCMIN}(\mathcal{Q}, x, u^0)$ **if** $q < q_{min}$ **then** $q_{min} = q ; u_{min} = u$ **end if****end for****return** q_{min}, u_{min}

Durch die Wahl der Startpunkte für die Suche in der Q-Funktion können verschiedene Verfahren abgeleitet werden. Eine Möglichkeit ergibt sich durch die Verwendung einer vom Benutzer vorgegebenen Menge von diskreten Aktionen U^0 als Startpunkte für die Suche in der Q-Funktion. Bei der Minimumssuche in der Q-Funktion wird für jede Aktion der diskreten Aktionsmenge ein Suchlauf durchgeführt und der kleinste gefundene Q-Wert über die Suchläufe als das gesuchte Minimum verwendet. Stellt man sicher, dass bei der Ausführung des Verfahrens in der Interaktionsphase die Aktionen aus der Menge U^0 in jedem Zustand mit Wahrscheinlichkeit größer Null gewählt werden, so kann man dieses Vorgehen als direkten Übergang der diskreten Aktionen zu kontinuierlichen Aktionen betrachten. Der gefundene Regler ist mindestens so gut wie der Regler, der unter Verwendung der diskreten Aktionen gelernt werden kann. Beim Lernen und bei der Ausführung besitzt er alle Möglichkeiten, die ein mit diskreten Aktionen eingelernter Regler ebenfalls besitzen würde. Zusätzlich kann er jedoch beliebige Aktionen aus dem gesamten Aktionsraum wählen und somit die Generalisierungsleistung der neuronalen Q-Funktion über den Zustands-Aktionsraum voll ausnutzen.

Eine weitere praktikable Möglichkeit, die Startpunkte für die Suche der minimalen Q-Werte zu wählen, basiert auf einer zufälligen Wahl der Startpunkte aus dem Aktionsraum. In jedem Suchdurchlauf wird u^0 gleichverteilt aus \mathcal{U} gezogen und der Gradientenabstieg basierend auf der Gradienteninformation im neuronalen Q-Netz durchgeführt. Bei Start des Lernvorganges und geringer Anzahl von Lerniterationen ist hierbei zu erwarten, dass die Anzahl der lokalen Minima relativ hoch ist und somit die Anzahl der zufällig gezogenen Startpunkte und die Anzahl der darauf basierenden Suchvorgänge ebenfalls groß gewählt werden muss. Im Verlauf des Lernvorganges kann die Anzahl der Suchvorgänge kleiner werden, da die systembedingten lokalen Minima in der Q-Funktion typischer Weise gering ist. Für jeden einzelnen Suchdurchgang ist die Wahrscheinlichkeit, einen Startpunkt im Einflussbereich des globalen Minimas zu ziehen, größer Null. Durch diese Vorgehensweise bei der gierigen Auswertung der Q-Funktion erhält man somit eine probabilistische Strategie. Diese Strategie selbst wird in der Interaktionsphase mit dem System verwendet. Attraktiv erscheinende lokale Minima werden auf diese Weise vom System besucht und entsprechende Übergänge in die Datenbasis hinzugefügt. Über die Dauer des Lernvorganges ist daher zu erwarten, dass sich lokale Minima, die sich auf Grund einer noch fehlerhaften Q-Funktion ergeben, abschwächen. Die probabilistische Strategie dient somit in diesem Fall als zusätzliche Explorationskomponente des Lernverfahrens, die das Sam-

Tabelle 4.4: Regelgüte der gelernten Regler mit NFQG an der Aufgabe der Regelung der Geschwindigkeit des Unterwasserfahrzeuges auf den Zielwert 0, ausgehend von 101 Startgeschwindigkeiten. Angegeben sind die Ergebnisse bei der Verwendung von verschiedenen Anzahlen von Startpunkten für die Minimumssuche.

Regler	\bar{N}	\bar{e}_∞
NFQG 3 Stratpunkte für Suche	14.92	0.0
NFQG 5 Stratpunkte für Suche	14.88	0.0
NFQG 10 Stratpunkte für Suche	14.85	0.0

meln der Daten an für die Approximation der Q-Funktion interessanten Stellen verstärkt. Die rein zufällige Wahl einer vorgegebenen Anzahl von Startpunkten hat sich in der Praxis als robuste und leistungsfähige Methode erwiesen. Die Auswertungen der Leistungsfähigkeit des NFQG-Verfahrens werden deshalb an dieser Vorgehensweise mit konstanter Anzahl von Startpunkten über den Lernvorgang durchgeführt. Die Verfahren lassen sich durch geeignete Wahl der Parameter und Kombination der Vorgehensweisen noch weiter verbessern, was hier nicht Fokus der Untersuchung ist.

Betrachtung am Unterwasserfahrzeug

Für die Untersuchung des NFQG-Verfahrens an der Regelungsaufgabe des Unterwasserfahrzeuges wird die Aktionsmenge U ersetzt durch einen Aktionsraum $\mathcal{U} = \{u : u \in [-30, 30]\}$. Es stehen dem selbstlernenden Regler somit in jedem Zustand alle Eingriffsgrößen im Intervall $[-30, 30]$ zur Verfügung. Die Durchführung der Lernversuche und die Parameter des NFQ-Verfahrens werden ansonsten von den vorangegangenen Lernversuchen übernommen. Mit dem NFQG-Verfahren werden, mit zufälliger Wahl der Startpunkte, Lernversuche mit 3, 5 und 10 Startpunkten pro Suchvorgang durchgeführt. Jeder Versuch wird wiederum 10 Mal unabhängig durchgeführt und die Ergebnisse der erzielbaren Reglergüte gemittelt. In Tabelle 4.4 sind die Ergebnisse der Lernversuche dargestellt. Bemerkenswert ist, dass für alle Versuche die Regelabweichung im ausgeregelten Zustand zu Null wird. Dies zeigt, dass das NFQG-Verfahren die kontinuierlichen Aktionen in Verbindung mit der \tanh^2 -Kostenfunktion effektiv ausnutzen kann. Auch im Gütekriterium für die Zeitoptimalität \bar{N} sind die Regler mit NFQG den

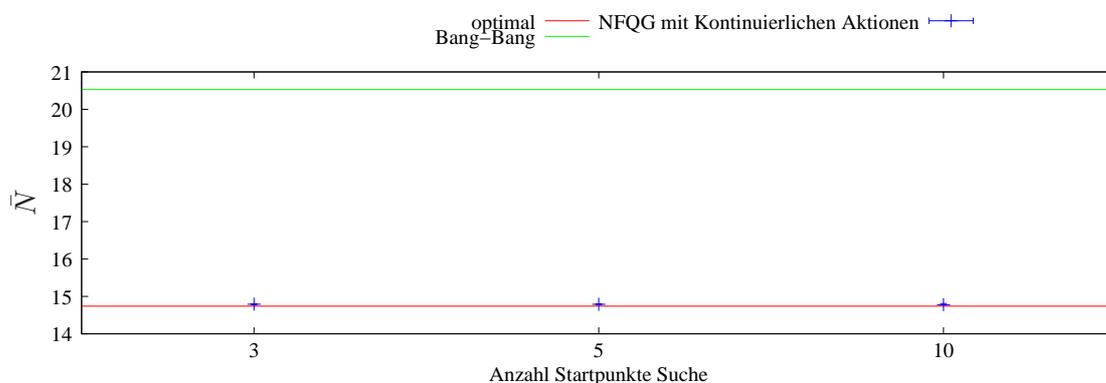


Abbildung 4.11: Erzielte Regelgüte in Bezug auf das zeitoptimale Verhalten der Regler \bar{N} über eine Reihe von 10 Versuchen mit NFQG. Die kontinuierlichen Aktionen bringen eine deutliche Verbesserung im Vergleich zu den diskreten Aktionen.

Reglern mit diskreten Aktionen deutlich überlegen (Abbildung 4.11 und 4.4) und kommen mit $\bar{N} = 14.85$ sehr nahe an die optimale Strategie ($\bar{N} = 14.74$). In Abbildung 4.12 ist der

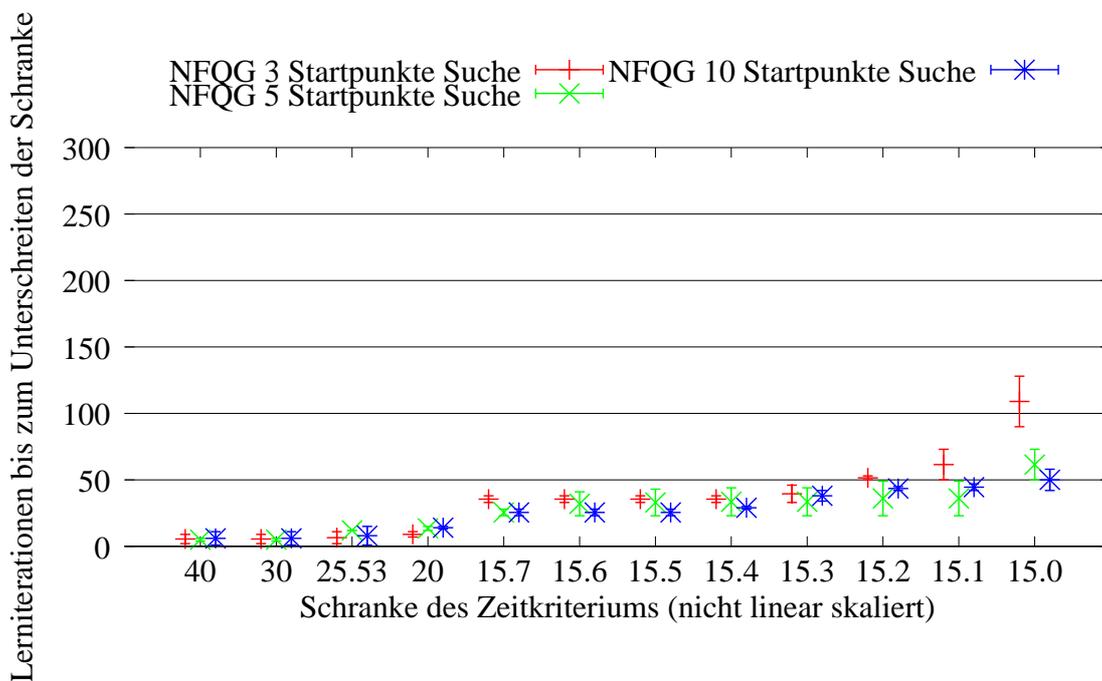


Abbildung 4.12: Die Auswertung der Lerngeschwindigkeit des NFQG-Verfahrens mit kontinuierlichen Aktionen. Für unterschiedliche Anzahl von Startpunkten, für die Suche in der Q-Funktion, sind die durchschnittliche Anzahl der Lerniterationen angegeben, nach denen eine bestimmte Schranke des Zeitkriteriums \bar{N} zum ersten Mal unterschritten wird. Die Werte sind gemittelt über die 10 durchgeführten Versuche. Zusätzlich ist das Minimum und Maximum aufgetragen.

Lernverlauf über die Iterationen des NFQG-Verfahrens bezüglich des Gütekriteriums der Zeitoptimalität dargestellt. Die Erhöhung der Anzahl der Startpunkte ermöglicht ein schnelleres Lernen und das Gütekriterium sinkt schneller unter bestimmte Schranken. Der Regler mit 10 Startpunkten erzielt dabei das beste Ergebnis und besitzt im Durchschnitt nach c.a. 45 Iterationen ein besseres Ergebnis, als die Regler mit diskreten Aktionen im gesamten Lernverlauf erreichen. Durch die Verwendung der kontinuierlichen Aktionen lernt das Verfahren schneller als mit diskreten Aktionen (vergleiche Abbildung 4.8). Dies ist eine erstaunliche Eigenschaft des Verfahrens, da die Verwendung kontinuierlicher Aktionen die Suche in einem unendlich großen Strategieraum bedingt. Es zeigt sich jedoch, dass die Verwendung kontinuierlicher Aktionen für regelungstechnisch relevante Problemstellungen zu einer deutlichen Verbesserung der Genauigkeit und Lerngeschwindigkeit führen. Dies liegt darin begründet, dass die diskreten Aktionen bei der Optimierung auf Zeitoptimalität und hohe Genauigkeit die Strategie komplexer werden lassen, als sie mit kontinuierlichen Aktionen sein müssen.

4.2.3 NFQG mit expliziter neuronaler Strategie

Durch die Ausnutzung der Generalisierung in der neuronalen Repräsentation der Q-Funktion im NFQG-Verfahren wird die Berechnung einer annähernd optimalen Strategie mit kontinuierlichen Aktionen möglich. Die gierige Auswertung der Q-Funktion zur Berechnung der Strategie

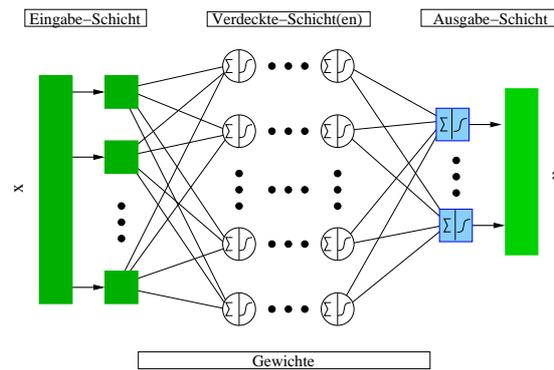


Abbildung 4.13: Schematische Darstellung der Struktur eines künstlichen Neuronales Netzes als Approximator einer expliziten Strategie-Funktion (Darstellung ohne Bias).

gie ist jedoch sehr rechenintensiv und erfüllt daher nur sehr bedingt die Echtzeitanforderungen, die die meisten Systeme im regelungstechnischen Kontext stellen. Diese Echtzeitanforderungen bestehen in der Interaktionsphase des Lernprozesses, beziehungsweise während der Ausführung des gelernten Reglers. Die Lernphase des Reglers verläuft unabhängig vom betrachteten Prozess und basiert ausschließlich auf der Datenbasis \mathcal{D} . Notwendige Berechnungen, welche die Echtzeitfähigkeit einschränken, können daher in diese Phase des Lernprozesses verschoben werden. Um das NFQG-Verfahren in der Interaktionsphase echtzeitfähig zu machen, muss die gierige Auswertung der Q-Funktion durch die rechenintensive Suche mit Gradientenabstieg durch eine effizientere Methode ersetzt werden. Dies wird durch die Verwendung einer expliziten Repräsentation der Strategie in Form eines zusätzlichen mehrschichtigen neuronalen Netzes realisiert.

Die Verwendung einer expliziten Repräsentation der Strategie in Form eines neuronalen Netzes $\Pi(x, w_\pi)$ hat bei der Verwendung in Reinforcement-Lernverfahren mehrere Vorteile. Zum Einen kann die Strategie durch eine Anfrage an das Netz direkt berechnet werden. Dies hat zeitlich große Vorteile und führt zu einem besseren Echtzeitverhalten während der Ausführung des Reglers. Zum Anderen bewirkt die Repräsentation der Strategie als explizite Funktion, dass die Strategie stabiler gegen kleine Änderungen der Wertfunktion wird. Kleine Änderungen im Zustandsraum bewirken kleine Änderungen in der Strategie. Bei der Minimum-Berechnung in der Q-Funktion dagegen können kleine Änderungen in der Wertfunktion sehr große Änderungen in der Strategie bewirken. Die explizit repräsentierte Strategie wirkt somit ähnlich wie eine zusätzliche Glättung der Strategie und generalisiert über den Zustandsraum.

Die Eingabe an das neuronale Netz ist der aktuelle Zustand des Systems, der durch die neuronale Strategiefunktion mit Gewichten w_π auf eine Aktion abgebildet wird $u_t = \Pi(x_t, w_\pi)$. An einer Erweiterung von NFQG wird untersucht, wie sich die explizite Repräsentation der Strategie nutzen lässt, um das Verfahren echtzeitfähig zu machen.

Die neuronale Strategie wird mit überwachtem Lernen aus einer Trainingsdatenmenge \mathcal{T} mit Einträgen der Form: $t = (t^{input}, t^{target}) = (x, \hat{\pi}_x)$ eingelernt. Diese Trainingsmenge wird im Lernverfahren nach dem Batch-Aktualisierungsschritt der Q-Funktion und dem überwachten Einlernen dieser Funktion erzeugt. Hierzu wird für alle Übergänge in der Datenbasis die Q-Funktion durch gradientenbasierte Suche gierig ausgewertet. Analog zur Grundidee der Fitted-Verfahren erfolgt das Lernen der neuronalen Strategiefunktion durch epochenbasiertes Lernen mit RProp, wodurch ein leistungsfähiges, überwachtes Lernverfahren Anwendung findet. Da

Tabelle 4.5: Regelgüte der gelernten Regler mit NFQG unter Verwendung einer expliziten neuronalen Strategie auf Geschwindigkeit 0 von 101 Startgeschwindigkeiten. Angegeben sind die Ergebnisse für verschiedene Anzahlen von Startpunkten für die Minimum-Suche.

Regler	\bar{N}	\bar{e}_∞
NFQG mit neuronaler Strategie	14.79	0.0

Tabelle 4.6: Empirische Werte zur benötigten Rechenzeit der Verfahren auf einem P4 2.6 GHz Rechner.

Verfahren	Lernen [ms / Übergang]	Strategie [ms / Aktion]
NFQ 3 diskrete Aktionen	3.11	0.022
NFQ 7 diskrete Aktionen	3.29	0.045
NFQ 13 diskrete Aktionen	3.36	0.085
NFQ 61 diskrete Aktionen	3.80	0.380
NFQG 3 Startpunkte	17.15	10.33
NFQG 5 Startpunkte	23.10	17.42
NFQG 10 Startpunkte	41.54	35.56
NFQGN 10 Startpunkte	79.74	0.003

dies nicht die Dauer der benötigten Interaktion mit dem System beeinflusst, steht im Prinzip unbegrenzte Rechenzeit zur Verfügung. Die neuronale Strategiefunktion wird ausschließlich zur Bestimmung der Strategie im Agenten verwendet. Für die Aktualisierung der Q-Funktion wird direkt die Minimumssuche in der neuronalen Q-Funktion verwendet. Ein Nachteil des beschriebenen Ansatzes ist die Einführung eines weiteren Fehlers, basierend auf der Approximationsleistung des neuronalen Netzes $\Pi(x, w_\pi)$.

Betrachtung am Unterwasserfahrzeug

Als Strategie-Repräsentation des NFQG-Verfahrens mit expliziter neuronaler Strategie (NFQGN) auf das Regelungsproblem des Unterwasserfahrzeuges, wird ein neuronales Netz mit einer (1-10-1) Topologie verwendet. Das NFQG-Verfahren wird mit 10 Startpunkten für die Suche angewendet und die Lernversuche analog zu den vorigen Versuchen durchgeführt. In Tabelle 4.5 sind die Ergebnisse des Verfahrens dargestellt. Die Genauigkeit der Regelung bleibt unverändert gut und die bleibende Regelabweichung wird über den 101 Testtrajektorien zu $\bar{e}_\infty = 0$. Das Gütekriterium für die Zeitoptimalität \bar{N} kann im Vergleich zum NFQG-Verfahren ohne explizite neuronale Strategie noch einmal verbessert werden. Dies zeigt, dass die Generalisierung in der neuronalen Strategiefunktion alleine schon einen Vorteil für die gelernten Regler erzeugen kann, ohne dass der durch die zusätzliche Approximation erzeugte Fehler den Lernprozess negativ beeinflusst. In Abbildung 4.14 ist der Lernverlauf des Verfahrens dargestellt. Im Vergleich zu Abbildung 4.12 ist die Lerngeschwindigkeit des Verfahrens durch die Einführung der neuronalen Strategiefunktion deutlich verbessert worden. Da die neuronale Strategie sich ausschließlich auf die Strategie des Reglers auswirkt und die Aktualisierung der Q-Funktion nicht beeinflusst, ist diese Verbesserung so nicht zu erwarten. Im Wesentlichen wird diese Verbesserung ebenfalls der Generalisierung der Strategie zugeschrieben.

In Abbildung 4.15 ist die gelernte Strategie im Vergleich zur optimalen Strategie dargestellt. Die Nichtlinearitäten der Systemdynamik sind in der Strategie angenähert aber noch nicht vollständig erfasst.

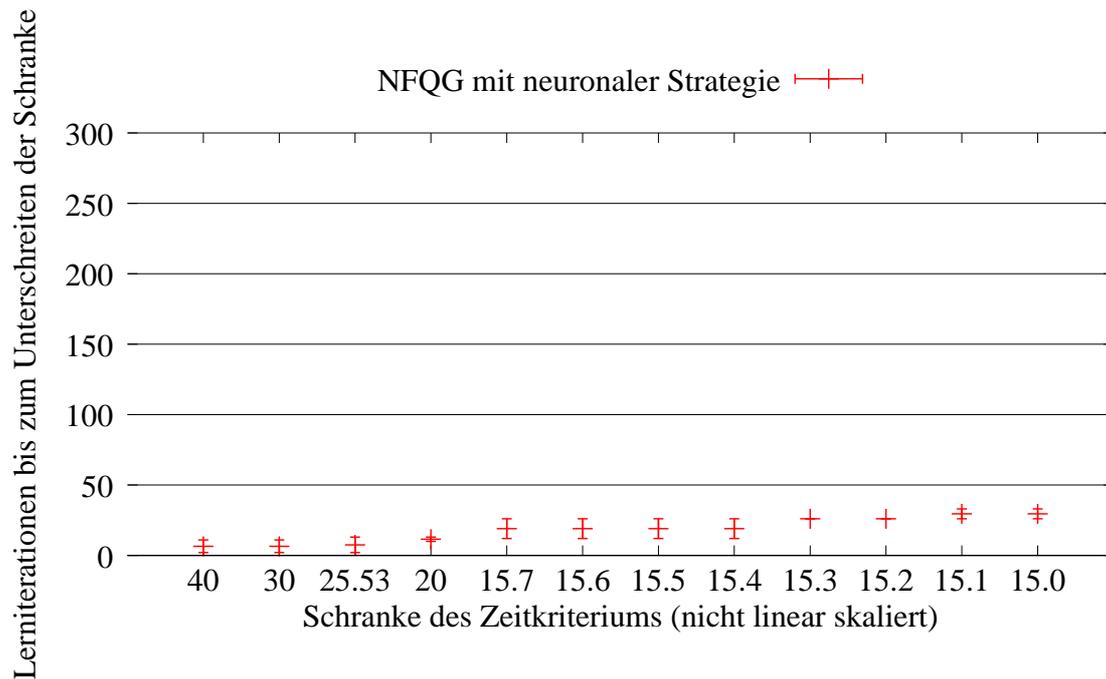


Abbildung 4.14: Die Auswertung der Lerngeschwindigkeit des NFQG-Verfahrens mit kontinuierlichen Aktionen unter expliziter Verwendung einer neuronalen Strategie. Angegeben sind die durchschnittliche Anzahl der Lerniterationen, nach denen eine bestimmte Schranke des Zeitkriteriums zum ersten Mal unterschritten wird. Die Werte sind gemittelt über jeweils 10 durchgeführten Versuche. Zusätzlich ist das Minimum und Maximum aufgetragen.

Die Einführung der expliziten Strategie in NFQG wurde motiviert durch die Echtzeitanforderungen bei der Ausführung der Strategie. Ein Vergleich der Rechenzeit der verschiedenen vorgestellten Verfahren ist in Tabelle 4.6 dargestellt. Gemessen wurde für alle Verfahren die benötigte Rechenzeit für eine Iteration des Lernverfahrens. Da in einem explorativen Lernprozess die Anzahl der Übergänge in der Datenbasis wächst, wurde der angegebene Wert in Rechenzeit pro Übergang in der Datenbasis angegeben. Dieser Wert bleibt für die Verfahren über den ganzen Lernverlauf konstant. Zusätzlich wurde die Rechenzeit für die Auswertung der Strategie in der Interaktionsphase gemessen. Dieser Wert ist angegeben in Rechenzeit pro Berechnung einer Aktion. Für die NFQ-Verfahren mit diskreten Aktionen wächst die Rechenzeit für eine Iteration des Lernverfahrens mit der Anzahl der verwendeten Aktionen annähernd linear, ebenfalls die Zeit für die Auswertung der Strategie. Bei NFQG ist die Zeit für eine Lerniteration deutlich höher und wächst mit der Anzahl der verwendeten Startpunkte annähernd linear an. Bei der Bestimmung der Strategie benötigen die NFQG-Verfahren eine um mehrere Größenordnungen erhöhte Rechenzeit und liegen mit 10 bis 30 Millisekunden in einem für reale Systeme nicht akzeptablen Bereich. Durch die Einführung der neuronalen Strategie erhöht sich die Rechenzeit für das Lernen noch einmal fast um Faktor 2. Die Rechenzeit für die Bestimmung der Strategie ist jedoch eine Größenordnung kleiner als selbst bei den NFQ-Verfahren und bleibt konstant, abhängig von der Netzgröße.

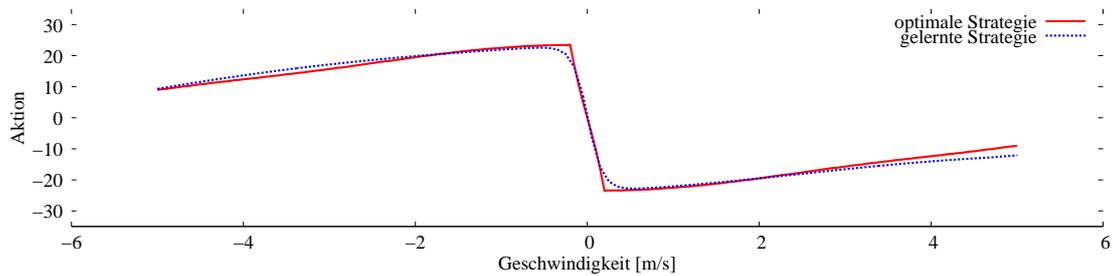


Abbildung 4.15: Die gelernte Strategie mit NFQG unter Verwendung einer expliziten neuronalen Strategie im Vergleich zu der optimalen Strategie. Die kontinuierlichen Aktionen sind in der Lage die optimale Strategie anzunähern.

4.3 NFQCA: Ein dateneffizientes Reinforcement Lernverfahren für kontinuierliche Aktionen

Aus den bisherigen Betrachtungen wird nun ein Verfahren zur Erweiterung von NFQ auf kontinuierliche Aktionen entwickelt, das ein praxistaugliches Lernverfahren für selbstlernende Regler mit kontinuierlichen Aktionen darstellt. Das neuronal gefittete Q-Lernen mit kontinuierlichen Aktionen (Neural Fitted Q-Iteration with Continuous Actions, NFQCA) vereint die Vorteile der expliziten Repräsentation der Q-Funktion in Form eines neuronalen Netzes und die daraus berechenbaren Gradienten der Q-Funktion mit den Vorteilen der expliziten Repräsentation der Strategie in Form eines neuronalen Netzes. Der rechenaufwändige Schritt der separaten Minimums-Suche wird durch eine Kopplung der Netze ersetzt, was ein geschlossenes Lernverfahren ermöglicht.

4.3.1 Die Verwendung von Aktor- und Kritiker-Komponente

Aktor-Kritiker-Ansätze (Actor-Critic) sind im Reinforcement-Lernen weit verbreitet. Diese Methoden haben die Eigenschaft, eine explizite Repräsentation der Strategie, unabhängig von einer Wertfunktion, zu besitzen. Die Strategie wird als Aktor bezeichnet, da sie für die Auswahl der Aktion benutzt wird und somit für die Interaktion mit dem System verantwortlich ist. Der Kritiker hingegen hat die Aufgabe, die durch den Aktor durchgeführten Aktionen, basierend auf einer Wertfunktion, zu bewerten. Die hier vorgestellte Erweiterung des NFQ-Verfahrens auf kontinuierliche Aktionen kann als solcher Actor-Kritiker-Ansatz eingeordnet werden.

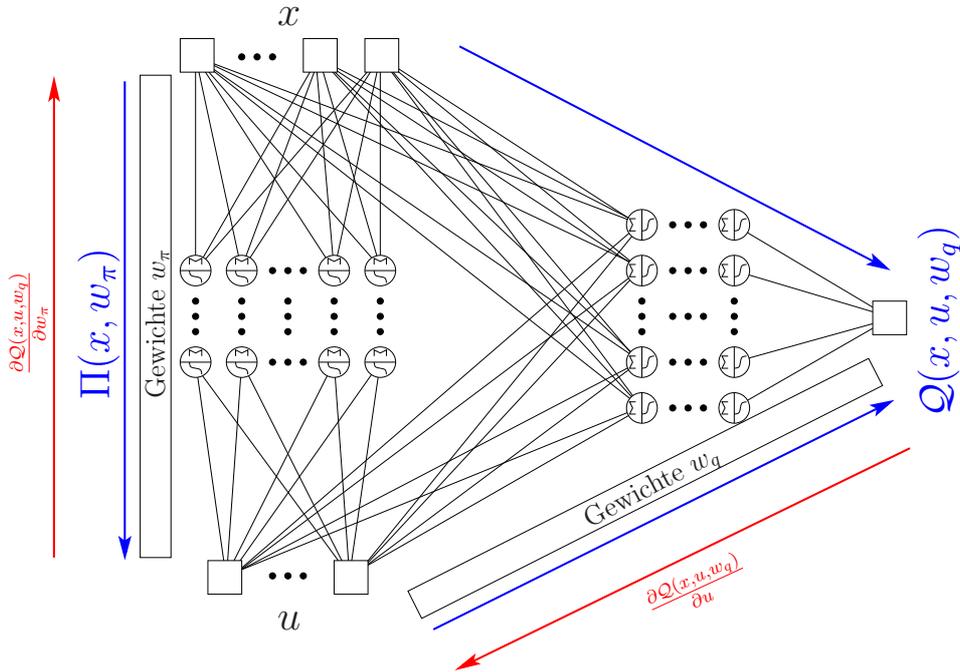


Abbildung 4.17: Struktur des neuronalen Dreiecks-Netztes, bestehend aus neuronalem Q-Netz und neuronalem Strategie-Netz. Dargestellt ist der Informationsfluss beim vorwärts Propagieren (blau) durch die Netze und die für die Aktualisierung des Aktors wichtigen Gradienten durch Rückpropagieren durch die Netze (rot).

koppelten Ausführung als einzelnes mehrschichtiges neuronales Netz betrachtet werden. Im Unterschied zu den in 2.2.4 diskutierten gekoppelten Netzen bestimmt die Ausgabe des Strategienetzes die Eingabe des Q-Netztes nicht vollständig. Sie betrifft nur die Eingabeneuronen, an denen der Aktionsvektor im Q-Netz anliegt.

4.3.3 Die Aktualisierung des Aktors

Für eine gegebene neuronale Q-Funktion, in einem Iterationsschritt k des Verfahrens soll das Strategienetz die gierige Auswertung dieser repräsentierten Q-Funktion approximieren. Hierzu müssen die Gewichte w_π der neuronalen Strategiefunktion $\Pi_k(x, w_\pi)$ so eingestellt werden, dass die Strategiefunktion die gekoppelte Funktion für jeden Zustand minimiert.

$$\Pi_k(x, w_\pi) \approx \operatorname{argmin}_{u \in \mathcal{U}} Q_k(x, u, w_q) \quad (4.5)$$

Hat die Strategiefunktion diese Eigenschaften, so repräsentiert das gekoppelte Netz die aktuelle Pfadkostenschätzung in diesem Iterationsschritt:

$$Q_k(x, \Pi_k(x, w_\pi), w_q) \approx \min_{u \in \mathcal{U}} Q_k(x, u, w_q) \quad (4.6)$$

Die Aktualisierung des Aktors basiert nun direkt auf der in 2.2.4 diskutierten Minimierung gekoppelter Netze durch Propagieren des Gradienten durch das Netz. Während der Aktualisierung des Aktors wird die Q-Funktion als unveränderbare Funktion angesehen, die durch den

Aktor minimiert werden muss.

Im Detail wird somit ein Zustand im gekoppelten Netz vorwärtspropagiert. Dabei wird der Zustand zuerst durch das Strategie-Netz propagiert. Die Ausgabe des Strategie-Netzes ergibt mit dem Zustand zusammen die Eingabe für das Q-Netz, die durch dieses ebenfalls vorwärtspropagiert wird. Durch Zurückpropagieren durch das Q-Netz liegt in dessen Eingabeneuronen der Vektor der partiellen Ableitungen der Q-Funktion nach dem Zustand und der von der Strategie berechneten Aktion an $\nabla \mathcal{Q}(x, \Pi(x))$. Für die Einstellung der Gewichte des Strategienetzes sind die partiellen Ableitungen der Q-Funktion nach der Aktion von Interesse $\nabla \mathcal{Q}_x(\Pi(x)) = \left(\frac{\partial \mathcal{Q}(x, \Pi(x))}{\partial u_1}, \dots, \frac{\partial \mathcal{Q}(x, \Pi(x))}{\partial u_m} \right)$. Durch die Kopplung der Netze wird jeweils die partielle Ableitung eines Eingabeneurons u_i der Q-Funktion an das entsprechende Ausgabe-neuron u_i des Strategienetzes angelegt. Durch Zurückpropagieren mit dem Backpropagation-Schema durch das Strategienetz kann in diesem somit für jedes Neuron partielle Ableitung der Gewichte nach seiner Ausgabe gebildet werden. Für das gesamte Netz entspricht dieses Vorgehen der Anwendung der Kettenregel:

$$\frac{\partial}{\partial w_\pi} \mathcal{Q}(x, \Pi(x, w_\pi), w_q) = \frac{\partial \mathcal{Q}(x, \Pi(x, w_\pi), w_q)}{\partial \Pi(x, w_\pi)} \cdot \frac{\partial \Pi(x, w_\pi)}{\partial w_\pi} \quad (4.7)$$

Mit dieser partiellen Ableitung der Q-Funktion nach den Gewichten des Strategie-Netzes können diese Gewichte mit Hilfe des Gradientenabstieges eingestellt werden.

Dieser überwachte Lernvorgang wird auf allen Zuständen in der Datenbasis mit epochenbasiertem Lernen mit RProp durchgeführt. Das RProp ist hier wiederum auf Grund seiner Robustheit gegen seine Parameter und den schnellen Lernvorgang vorteilhaft. Es wurden wiederum die in der Literatur gegebenen Standardparameter verwendet. In Algorithmus 12 ist der Aktualisierungs-Schritt für den Aktor in Form eines Algorithmus dargestellt. Die Aktualisie-

Algorithmus 12 *AktorFittedTraining*(\mathcal{Q}, \mathcal{D})

Require: \mathcal{Q} : Approximation der Q-Funktion, \mathcal{D} : Menge der beobachteten Übergänge

Ensure: $\Pi(x)$: neuronale repräsentation der gierigen Strategie bezüglich \mathcal{Q}

$\Pi_0 \leftarrow$ neuronales Netz mit zufällig initialisierten Gewichten

$k \leftarrow 0$

repeat

for $i = 0, \dots, \#\mathcal{D}$ **do**

$u = \text{forward_propagate}(\Pi_k, x^i)$

$q = \text{forward_propagate}(\mathcal{Q}, x^i, u)$

$\left(\frac{\partial q}{\partial x_1^i}, \dots, \frac{\partial q}{\partial x_n^i}, \frac{\partial q}{\partial u_1}, \dots, \frac{\partial q}{\partial u_m} \right) = \text{backward_propagate}(\mathcal{Q}, 1)$

$\text{backward_propagate}(\Pi, \left(\frac{\partial q}{\partial u_1}, \dots, \frac{\partial q}{\partial u_m} \right))$ kumulieren von $\frac{\partial q}{\partial w_\pi}$ für alle Gewichte

end for

$\Pi_{k+1} \leftarrow \text{update_weights}(\Pi_k)$ mit RProp Gradientenabstieg

$k \leftarrow k + 1$

until konvergiert

return Π_k

rung des Aktors in der gegebenen Form wurde auf den Zuständen in der Datenbasis \mathcal{D} durchgeführt. Dies ist nicht zwingend notwendig, da die Form der Aktualisierung für eine gegebene Q-Funktion ohne Wissen über die Systemdynamik durchgeführt werden kann. Somit sind Abwandlungen des Algorithmus denkbar, die diese Aktualisierung an beliebigen Stellen des

Zustandsraumes durchführen. Die Aktualisierung, basierend auf den in der Datenbasis vorhandenen Zuständen, ist jedoch naheliegend, da die Q-Funktion mit diesen Daten eingelernt wurde. Somit kann an diesen Stellen die genaueste Information erwartet werden.

Die Aktualisierung des Aktors wird durch das Propagieren der Gradienten im Dreiecks-Netz durchgeführt. Im Vergleich zum NFQG-Verfahren mit expliziter Strategie wird somit keine separate Trainingsmenge durch die Minimumssuche erstellt. Vielmehr wird die Minimumssuche über das gekoppelte Netz, basierend auf dem Gradienten der Q-Funktion durchgeführt.

Analog zum NFQG-Verfahren muss in der Aktualisierung des Aktors die Beschränkung der Aktionen auf die entsprechenden Intervallgrenzen beachtet werden. Da ein explizites neuronales Netz als Strategierepräsentation verwendet wird, kann dies über eine Skalierung der Aktivierungsfunktionen der Ausgabeneuronen durchgeführt werden. Typischerweise werden symmetrische Aktivierungsfunktionen benutzt, die für jedes Neuron auf den Wertebereich des entsprechenden Intervalls normiert werden. Auf diese Weise entspricht der Wertebereich der Strategiefunktion dem Aktionsraum des Prozesses.

4.3.4 Das NFQCA-Verfahren

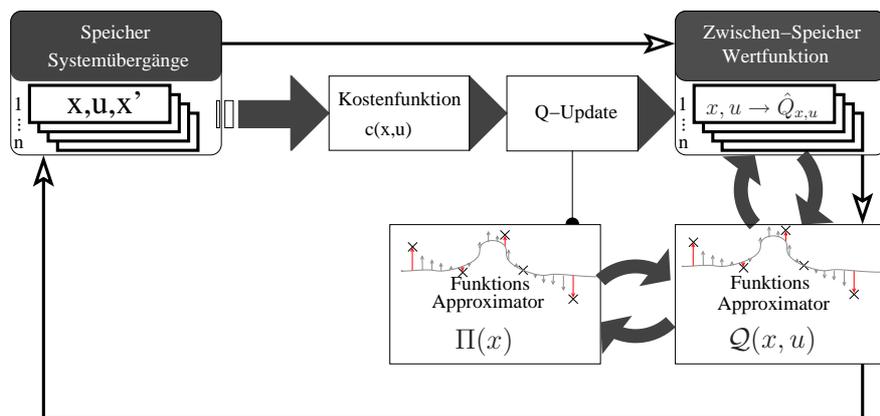


Abbildung 4.18: Das NFQCA als Batch-Verfahren. Die Strategie wird explizit durch ein neuronales Netz repräsentiert, was die schnelle Ausführung der Strategie unter Echtzeitanforderungen erlaubt. Der Datenfluss durch das System basiert auf der gesamten Datenbasis, dies erlaubt ein sehr effizientes Lernen.

Für die Aktualisierung der Q-Funktion wurde in NFQG die gradientenbasierte Minimumssuche im neuronalen Q-Netz verwendet. Dieser Vorgang ist sehr rechenintensiv, bestimmt jedoch direkt auf der Approximation der Q-Funktion die notwendigen Minima. Im Unterschied dazu wird bei NFQCA in einem Iterationsschritt direkt die Approximation der gierigen Strategie bezüglich der aktuellen Q-Funktion verwendet. Diese liegt in der explizit repräsentierten Strategienetz vor. In diesem Aktualisierungsschritt werden die zwei neuronalen Netze als unabhängige Netze betrachtet. Das neuronale Q-Netz kann, basierend auf der Trainingsmenge \mathcal{P} , mit RProp überwacht eingelernt werden. In Algorithmus 13 ist die Erzeugung dieser Trainingsdatenmenge \mathcal{P} für das NFQCA-Verfahren als Pseudocode dargestellt. Wurde die Q-Funktion aktualisiert, so kann mit Algorithmus 12, basierend auf der neuen Approximation der Q-Funktion, die Approximation der gierigen Strategie erzeugt werden. In diesem Teil des Verfahrens werden die Netze als gekoppelt betrachtet und der Gradient durch beide Netze hindurchpropagiert. In Algorithmus 14 ist die Aktualisierung eines Iterationsschrittes des NFQCA-Verfahrens dar-

Algorithmus 13 BatchQCAUpdate($\mathcal{Q}, \Pi, \mathcal{D}$)

Require: \mathcal{Q} : Approximation der Q-Funktion**Require:** Π : neuronale Approximation der gierigen Strategie bzgl. \mathcal{Q} **Require:** \mathcal{D} : Menge der beobachteten Übergänge**Ensure:** \mathcal{P} : Trainingsdatenmenge mit aktualisierten Q-Werten $\mathcal{P} \leftarrow \emptyset$ **for** $i = 1, \dots, \#\mathcal{D}$ **do**mit $d_i = (x_i, u_i, x'_i, c_i)$ i-tes Element $\in \mathcal{D}$ $\hat{Q} = (1 - \alpha) * \mathcal{Q}(x_i, u_i) + \alpha * (c_i + \gamma \mathcal{Q}(x'_i, \Pi(x'_i)))$ $\mathcal{P} \leftarrow \mathcal{P} \cup ((x_i, u_i), \hat{Q})$ **end for****return** \mathcal{P}

Algorithmus 14 NFQCAUpdate($\mathcal{Q}_k, \Pi_k, \mathcal{D}$)

Require: \mathcal{Q}_k : neuronale Approximation der Q-Funktion, Π_k : neuronale Approximation der Strategie**Require:** \mathcal{D} : Menge der beobachteten Übergänge**Ensure:** \mathcal{Q}_{k+1} : aktualisierte neuronale Approximation der Q-Funktion**Ensure:** Π_{k+1} : aktualisierte neuronale Approximation der Strategie $\mathcal{P} = \text{BatchQCAUpdate}(\mathcal{Q}_k, \Pi_k, \mathcal{D})$ $\mathcal{Q}_{k+1} = \text{RpropTraining}(\mathcal{Q}_0, \mathcal{P})$ $\Pi_{k+1} = \text{AktorFittedTraining}(\mathcal{Q}_{k+1}, \mathcal{D})$ **return** $\mathcal{Q}_{k+1}, \Pi_{k+1}$

gestellt.

Analog zu NFQ ergeben sich aus dieser Iterationsvorschrift verschiedene Möglichkeiten, einen Lernprozess durchzuführen. Der für diese Arbeit interessantere Fall des explorativen Lernprozesses ist ebenso möglich wie ein rein datenbasierter Lernprozess oder Kombinationen dieser Vorgehen. In allen Fällen beginnt das Lernverfahren mit einem zufällig initialisierten neuronalen Q-Netz \mathcal{Q}_0 und der dazu gehörenden Approximation der gierigen Strategie Π_0 . Das initiale Strategienetz wird somit zu Beginn des Lernverfahrens, basierend auf er zufällig initialisierten Q-Funktion, eingelernt.

Das NFQCA-Verfahren führt im Vergleich zum NFQ-Verfahren nur wenige neue Parameter ein, die für einen Lernprozess zu bestimmen sind. Im Wesentlichen muss nur die Topologie des Strategienetzes festgelegt werden. Ähnlich zur Wahl der Topologie für die Q-Funktion ist das Lernverfahren durch die Verwendung von epochenbasiertem Lernen mit RProp in gewissen Grenzen robust gegen die Wahl dieser Topologie. Auf der anderen Seite fällt durch die Verwendung kontinuierlicher Aktionen die Auswahl der diskreten Aktionen zur Lösung des Regelungsproblems weg.

Für den explorativen Lernprozess wurde in 2.3.3 die Verwendung einer zufälligen Komponente für die Exploration des Systems eingeführt. Im Fall der diskreten Aktionen wurde in jedem Schritt mit einer gewissen Wahrscheinlichkeit ϵ von der gierigen Strategie abgewichen und eine zufällige Aktion aus der Aktionsmenge gewählt. Bei kontinuierlichen Aktionen ist diese zufällige Exploration durch gleichverteiltes Ziehen der Aktion aus dem gesamten Aktionsraum nicht mehr ausreichend. Es wird deshalb dazu übergegangen, die Wahl der Aktion in einem zufälligen Explorationsschritt normalverteilt um die aktuelle Strategie zu wählen. Die

Tabelle 4.7: Regelgüte der gelernten Regler mit NFQCA auf Geschwindigkeit 0 von 101 Startgeschwindigkeiten. Angegeben sind die Ergebnisse für verschiedene Anzahlen von Neuronen im Aktornetz.

Regler	\bar{N}	\bar{e}_∞
NFQCA Policy Netz mit 5 Neuronen	14.792	0.0
NFQCA Policy Netz mit 10 Neuronen	14.790	0.0
NFQCA Policy Netz mit 15 Neuronen	14.781	0.0

Strategie, während der Interaktion mit dem System führt somit in jedem Zeitschritt mit einer gewissen Wahrscheinlichkeit (ϵ) die Aktionswahl mit: $\pi(x) = \Pi(x) + \Delta$ mit $\Delta \sim \mathcal{N}(0, \sigma)$ aus (eine ähnliche Explorationsstrategie veröffentlicht Hasselt (2007) [110]).

Betrachtung am Unterwasserfahrzeug

Für die Untersuchung des NFQCA-Verfahrens an der Regelungsaufgabe des Unterwasserfahrzeuges, werden drei verschiedene Topologien der neuronalen Strategiefunktion betrachtet. Die Netze besitzen alle eine verdeckte Schicht und haben eine (1-5-1), (1-10-1) und (1-15-1) Topologie. Die symmetrische Aktivierungsfunktion der Ausgabeneuronen wird auf den Wertebereich von -30 bis 30 skaliert. Die Anzahl der mit RProp durchgeführten Epochen für das Lernen des Strategienetzes wird auf 600 festgelegt. Ansonsten werden die Versuche mit den aus den vorigen Versuchen bekannten Parametern und Vorgehen durchgeführt. In Tabelle 4.7 sind die erzielten Ergebnisse zusammengefasst. Der selbstlernende Regler mit dem NFQCA-Verfahrens kann die bleibende Regelabweichung, wie schon das NFQG-Verfahren, vollständig eliminieren. Mit 10 Neuronen in der verdeckten Schicht ist das Gütekriterium der Zeitoptimalität \bar{N} gleich zu den Versuchen mit NFQG bei Verwendung einer expliziten Strategie. Die Erhöhung der Anzahl der Neuronen im Strategienetz kann das Gütekriterium der Zeitoptimalität noch einmal gering verbessern. Es kommt mit $\bar{N} = 14.781$ im Durchschnitt sehr nahe an das analytisch bestimmte Optimum von $\bar{N} = 14.74$. Selbst mit nur 5 Neuronen in der verdeck-

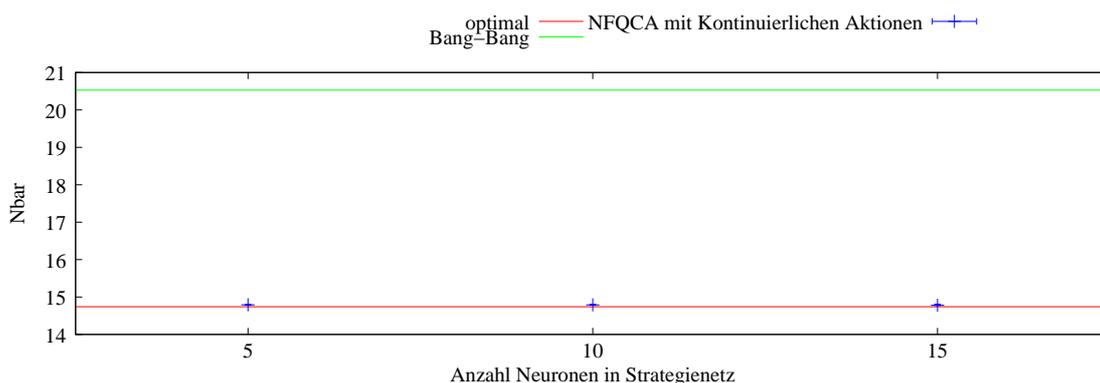


Abbildung 4.19: Erzielte Regelgüte in Bezug auf das zeitoptimale Verhalten der Regler \bar{N} über eine Reihe von 10 Versuchen mit NFQCA. Die kontinuierlichen Aktionen bringen eine deutliche Verbesserung im Vergleich zu den diskreten Aktionen und erreichen annähernd die optimale Strategie.

ten Schicht sind die Ergebnisse besser als die Ergebnisse von NFQ mit 61 diskreten Aktionen. Dies kann auch in Abbildung 4.20 am Verlauf des Gütekriteriums der Zeitoptimalität über den Lerniterationen des Verfahrens abgelesen werden. Dieser Verlauf ist deutlich schneller als bei

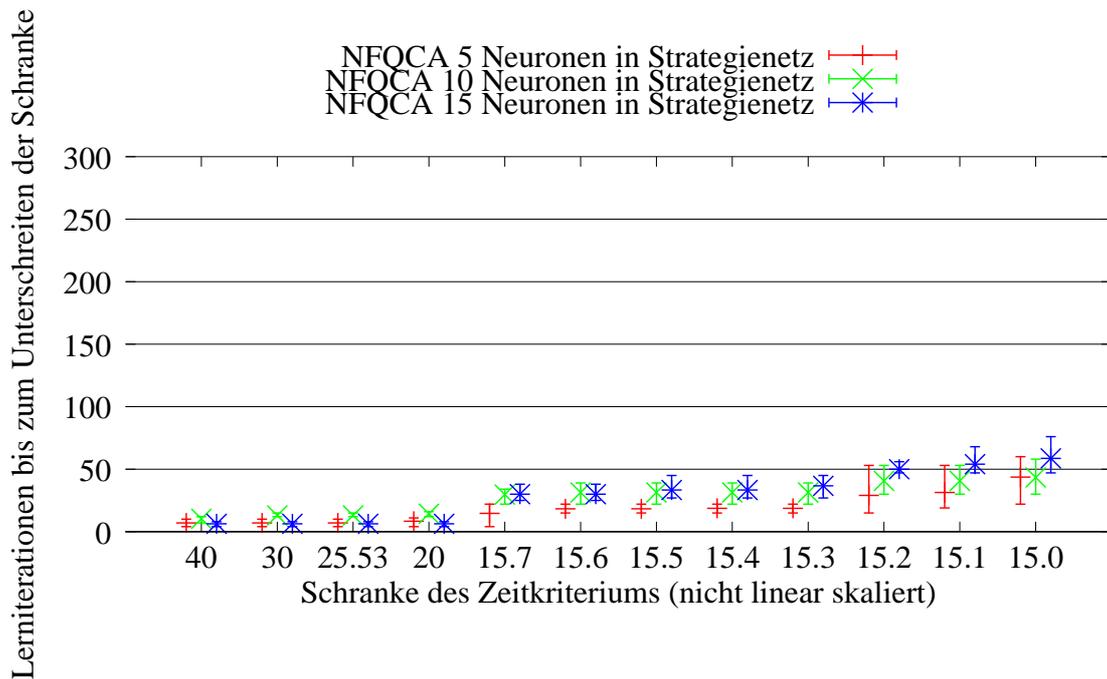


Abbildung 4.20: Die Auswertung der Lerngeschwindigkeit des NFQCA-Verfahrens mit kontinuierlichen Aktionen. Für unterschiedliche Anzahl von Neuronen im Strategienetz ist die durchschnittliche Anzahl der Lerniterationen angegeben, nach denen eine bestimmte Schranke des Zeitkriteriums \bar{N} zum ersten mal unterschritten wird. Die Werte sind gemittelt über die 10 durchgeführten Versuche. Zusätzlich ist das Minimum und Maximum aufgetragen.

der Verwendung diskreter Aktionen. Gegenüber den NFQG-Verfahren geht die Lerngeschwindigkeit etwas zurück, der zu erzielende Wert im Gütekriterium der Zeitoptimalität wird jedoch verbessert.

In Abbildung 4.21 ist eine mit NFQCA gelernte Strategie im Vergleich zu der optimalen Strategie des Regelungsproblems dargestellt. Die gelernte Strategie nähert die optimale Strategie

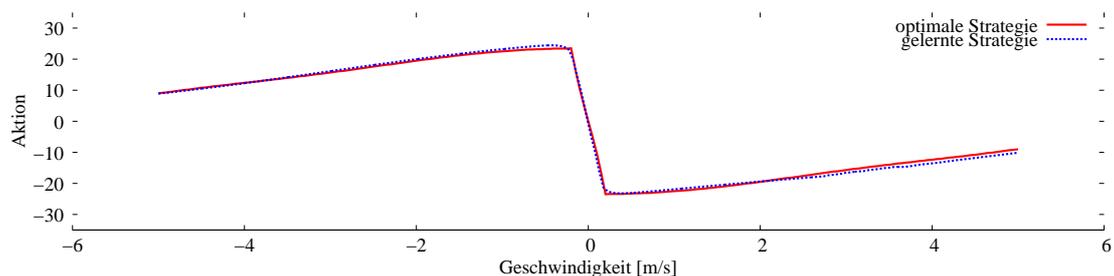


Abbildung 4.21: Die gelernte Strategie mit NFQCA im Vergleich zu der optimalen Strategie. Die kontinuierlichen Aktionen sind in der Lage die optimale Strategie sehr gut anzunähern.

sehr gut an und hat die Nichtlinearitäten der Systemdynamik erfasst.

Die Rechenzeit des NFQCA-Verfahrens ist in Tabelle 4.8 dargestellt. Für eine Iteration des Lernverfahrens wächst sie annähernd linear mit der Größe des Strategienetzes an und liegt in der Größenordnung des NFQG-Verfahrens mit 3 Startpunkten. Durch die Verwendung einer

Tabelle 4.8: Empirische Werte zur benötigten Rechenzeit der Verfahren auf einem P4 2.6 GHz Rechner.

Verfahren	Lernen [ms / Übergang]	Strategie [ms / Aktion]
NFQCA 5 Neuronen in Strategie	15.49	0.003
NFQCA 10 Neuronen in Strategie	15.81	0.003
NFQCA 15 Neuronen in Strategie	16.32	0.004

Tabelle 4.9: Auswertung der Reglergüte unter Verwendung von NFQ bzw. NFQCA an der Folgeregelung des Unterwasserfahrzeuges.

Regler	\bar{N}	\bar{e}_∞	\bar{e}_T
NFQ	18.22	0.054	0.34
NFQCA	16.01	0.015	0.28

expliziten neuronalen Strategie ist die Rechenzeit für die Auswertung der Strategie wiederum kleiner als bei NFQ und in einem echtzeitfähigen Bereich.

Folgeregelung des Unterwasserfahrzeuges

Zu Gunsten der besseren Vergleichbarkeit und Darstellung wurde in den letzten Abschnitten die Regelung des Unterwasserfahrzeuges anhand einer Festwertregelung auf Geschwindigkeit 0 betrachtet. Im Folgenden werden die Ergebnisse der selbstlernenden Regler auf eine variable Führungsgröße v^d diskutiert. Die Diskussion konzentriert sich auf den erreichten Fortschritt durch die Verwendung kontinuierlicher Aktionen mit NFQCA im Vergleich zur Verwendung diskreter Aktionen in NFQ. Auf den sehr aufwändigen Entwurf eines analytisch optimalen Reglers für das stark nichtlineare System, im Fall der Folgeregelung auf beliebige Führungsgrößen, wird verzichtet.

Der Zustandsvektor des MDPs erweitert sich um die aktuelle Regelabweichung $v^d - v$ und ergibt sich zu $x = (v, v^d - v)$. Somit wird auch das neuronale Q-Netz bzw. das neuronale Strategienetz um ein Eingabeneuron erweitert. Für den Lernprozess werden zufällige Sollwerte aus dem Intervall $v^d \in [-4, 4]$ zu Beginn einer Interaktionstrajektorie gewählt und auf dieser konstant gehalten. Die Startzustände werden ebenfalls vor jeder Interaktionstrajektorie zufällig, aus dem Intervall $[-5, 5]$, gewählt. Die Kostenfunktion sei mit Hilfe der \tanh^2 -Funktion, mit $\mu = (0, 0.3)$, $x^d = (-, 0)$ und $\mathcal{C} = 0.01$, definiert. Nach jeder Iteration des Lernverfahrens wird dieses auf 200 Episoden mit zufällig gewählten Startzuständen und Sollwerten getestet. Auf diesen wird die durchschnittliche bleibende Regelabweichung \bar{e}_∞ auf den jeweils letzten 50 Schritten der Trajektorie und die Schritte außerhalb des Toleranzbereiches \bar{N} gemessen. Zusätzlich wird der Regler an einem Profil des Verlaufs der Führungsgröße getestet, auf dem die mittlere Regelabweichung \bar{e}_T bestimmt wird.

Nach 110 Iterationen des NFQCA-Verfahrens wird der beste Regler gefunden, bei NFQ ist dies nach 140 Iterationen der Fall. In Tabelle 4.9 sind die Ergebnisse der Versuche zusammengefasst. Wie bereits bei der Festwertregelung, kann durch die Anwendung des NFQCA-Verfahrens eine Verbesserung sowohl im Kriterium der bleibenden Regelabweichung als auch im Kriterium der Zeitoptimalität erreicht werden. In Abbildung 4.22 sind die Ergebnisse mit NFQCA und NFQ mit 5 diskreten Aktionen in Form eines Verlaufes einer Trajektorie gegenübergestellt. Das Profil des Verlaufes der Führungsgröße ist so gewählt, dass mehrere Rampen enthalten sind, bei denen der Regler schrittweise die Aktion nachführen muss, um den Ziel-

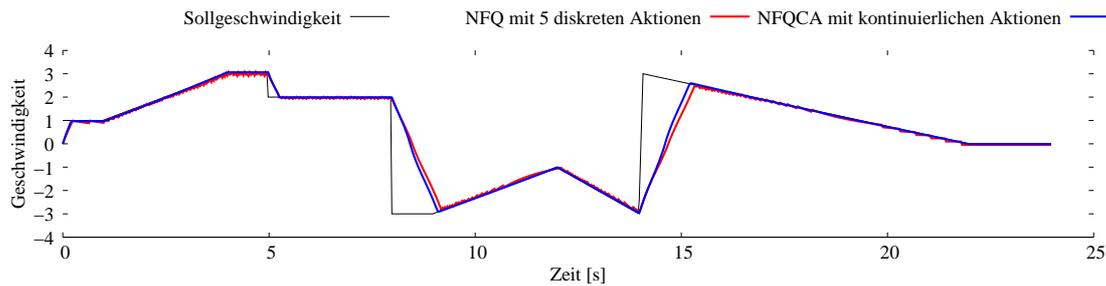


Abbildung 4.22: Die Aufgabe der Folgeregelung auf eine gegebene Trajektorie. Die mit NFQ und 5 diskreten Aktionen gefundene Lösung und die mit kontinuierlichen Aktionen gefundene Lösung ist über dem Verlauf der Geschwindigkeit zur Führungsgröße über der Zeit aufgetragen.

wert möglichst optimal zu halten. Zusätzlich sind Sprünge enthalten, bei denen sich der Führungswert schneller verändert, als die Systemdynamik eine Ausregelung der Regelabweichung zulässt. In Abbildung 4.23 sind die von den beiden gelernten Reglern erzeugten Eingriffsgrößen

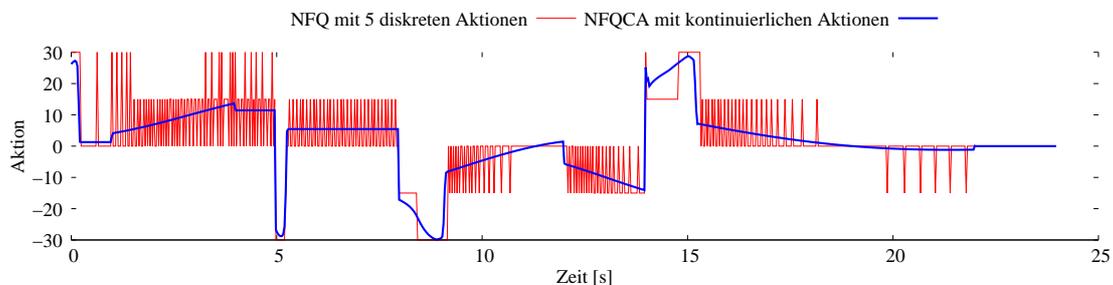


Abbildung 4.23: Der Verlauf der Eingriffsgröße eines Reglers mit kontinuierlichen Aktionen und eines Reglers mit diskreten Aktionen am Beispiel der Folgeregelung des Unterwasserfahrzeuges (vgl. Abbildung 4.22). Das NFQCA-Verfahren erzeugt ein glattes Regelgesetz, welches dennoch schnelle Änderungen der Eingriffsgröße vornehmen kann.

ßen über die Folgeregelung zu sehen. Das NFQCA-Verfahren erzeugt ein Regelgesetz, das sehr glatt ist und dennoch schnelle Änderungen der Eingriffsgröße zulässt. Im Gegensatz dazu muss das NFQ-Verfahren mit diskreten Aktionen die optimale Strategie durch eine zeitliche Abfolge der zur Verfügung stehenden Aktionen annähern. Im direkten Vergleich in Abbildung 4.23 kann diese zeitliche Kodierung mit diskreten Aktionen gegen die kontinuierliche Strategie des NFQCA sehr deutlich erkannt werden. Für das NFQ-Verfahren resultiert daraus ein Schwingen um den Führungswert, der in Abbildung 4.22 zu sehen ist. Im Gegensatz dazu erzeugt das NFQCA-Verfahren eine sehr glatte Trajektorie, die dem Führungswert eng folgt.

4.4 Zusammenfassung

Durch die Verwendung kontinuierlicher Aktionen entfällt eine zentrale Designentscheidung, die der Benutzer beim Entwurf selbstlernender Regler mit NFQ treffen muss. Dies ist die Wahl geeigneter diskreter Aktionen für die Aktionsmenge U , die eine Optimierung des Prozesses bezüglich des Regelungszieles erlauben. Ohne Kenntnis der Prozessdynamik und deren Eigenschaften in verschiedenen dynamischen Situationen, ist die Wahl geeigneter diskreter Aktionen für ein Regelungsproblem schwierig.

Bei der Verwendung kontinuierlicher Aktionen hingegen wird dem Verfahren nur der Wertebereich möglicher Eingriffsgrößen für den zu regelnden Prozess mitgeteilt. Die durchgeführten empirischen Untersuchungen zeigen, dass die Genauigkeit der Regler durch die Verwendung kontinuierlicher Aktionen deutlich gesteigert werden kann, ohne das zeitoptimale Verhalten zu verlieren. Die im letzten Kapitel entwickelten Verfahren zur Korrektur des systematischen Fehlers in der Q-Funktion und die \tanh^2 -Kostenfunktion können in Kombination mit kontinuierlichen Aktionen ihre volle Leistungsfähigkeit entwickeln.

Ein wichtiger Aspekt ist, dass die Einführung kontinuierlicher Aktionen die Anzahl der benötigten Interaktionsepisoden und Lerniterationen nicht erhöht. Obwohl der betrachtete Strategie-Suchraum unendlich groß wird, benötigen die in dieser Arbeit vorgestellten Verfahren in den empirischen Untersuchungen weniger Lern-Iterationen als das NFQ-Verfahren mit diskreten Aktionen.

Die Vorteile der Verwendung einer neuronalen Q-Funktion mit kontinuierlichen Aktionen wurden bereits am NFQG-Verfahren deutlich. Durch die Suche in der Q-Funktion ist das Verfahren jedoch sehr rechenintensiv, was speziell in der Auswertung der Strategie zu einem nicht mehr echtzeitfähigen Verhalten führt. Die Idee der Erweiterung des Verfahrens durch eine explizite neuronale Strategie hat nicht nur den Vorteil einer echtzeitfähigen Ausführung. Die Versuche zeigen auch eine bessere Regelgüte und einen schnelleren Lernprozess.

Das NFQCA ist ein praxistaugliches Verfahren, das die Vorteile einer expliziten Repräsentation einer neuronalen Strategie mit den Vorteilen der neuronalen Q-Funktion mit kontinuierlichen Aktionen kombiniert. Im Vergleich zu den NFQG-Verfahren hat es dabei den Vorteil, dass die Generalisierungsleistung des Strategie-Netzes in die Approximation der Q-Funktion mit einfließt. Die benötigte Rechenzeit des NFQCA ist im Vergleich zu den NFQG-Verfahren in einem akzeptablen Rahmen, jedoch deutlich höher als bei NFQ. Dies betrifft jedoch ausschließlich die Rechenzeit auf dem Computer. Die reine Interaktionszeit am System kann, im Vergleich zu NFQ, reduziert werden. Da sich die effektiv benötigte Rechenzeit durch den Einsatz schnellerer Rechner und effizienterer Programmierung noch steigern lassen, wird dies nicht als zentraler Nachteil des Verfahrens gewertet.

Ein weiterer Aspekt, der noch nicht diskutiert wurde, ist, dass das NFQG-Verfahren mit expliziter neuronaler Strategie in den präsentierten empirischen Ergebnissen eine etwas bessere Lerngeschwindigkeit zeigt als das NFQCA-Verfahren. Im Gegensatz dazu erreicht das NFQCA-Verfahren jedoch eine bessere Regelgüte. Dies bestätigen auch die in der praktischen Anwendung gemachten Erfahrungen. Motivieren lässt sich dies dadurch, dass das NFQG-Verfahren zu Beginn des Lernverfahrens lokale Minima in der Minimumssuche besser überwinden kann als das NFQCA-Verfahren. Eine Möglichkeit ist es deshalb, im NFQCA-Verfahren in bestimmten Abständen die Aktualisierung mit der im NFQG-Verfahren verwendeten expliziten Minimumssuche mit multiplen Startpunkten durchzuführen. In der Praxis kann dies zu einem schnelleren Lernverlauf des NFQCA-Verfahrens führen. Für die in dieser Arbeit durchgeführten Versuche soll jedoch die reine Form des NFQCA untersucht werden.

Regelungstechnische Versuche

Das nun folgende Kapitel untersucht die anwendungsorientierte Verwendung der selbstlernenden Regler an ausgewählten regelungstechnischen Beispielen. Ein Hauptaugenmerk liegt dabei auf der Betrachtung typischer Problemstellungen aus dem Bereich der regelungstechnischen Praxis und der Untersuchung der Leistungsfähigkeit der vorgestellten Verfahren in diesen Szenarien. Die hierbei vorgestellten Regelungsprobleme stammen aus dem Bereich der linearen und nichtlinearen Regelungstechnik. Durch die Betrachtung dieser Systeme unter dem Aspekt der selbstlernenden Regler ergeben sich interessante Benchmarkprobleme, die sich deutlich von klassischen Benchmarks im Reinforcement-Lernen unterscheiden. Im Speziellen enthalten diese Benchmarkprobleme den interessanten und im Reinforcement-Lernen selten betrachteten Aspekt der Regelung unter externer Sollwertvorgabe.

In Abschnitt 5.1 wird das in Kapitel 3.1.2 vorgestellte System des DC-Motors mit nichtlinearer Endstufe zur Untersuchung des generellen Regelverhaltens der betrachteten Verfahren verwendet. Hierzu wird die Leistungsfähigkeit der kontinuierlichen Aktionen an diesem System untersucht und das Verhalten der Regler unter einer nicht direkt beobachtbaren externen Last und Rauschen betrachtet. Diese Betrachtungen stehen exemplarisch für das Verhalten, das die Regler an verschiedenen Systemen zeigen und deren Leistungsfähigkeit unter realistischen Bedingungen. Am Beispiel eines selbstlernenden Reglers für einen Autopiloten eines Boeing-Verkehrsflugzeuges wird in Abschnitt 5.2 die Regelung von sehr trägen Systemen untersucht. Für die selbstlernenden Regler ist dieser Anwendungsfall interessant, da das Erreichen des stabilen Zustandes im Ziel mehrere hundert Zeitschritte benötigt und dabei die Regelgröße möglichst genau auf der Führungsgröße gehalten werden muss. Dies erfordert eine genaue Nachführung der Eingriffsgröße, die mit diskreten Aktionen nur schwer realisierbar sind. Der Einsatz der selbstlernenden Regler für die aktive Schwingungsdämpfung wird in Abschnitt 5.3 betrachtet. Am Beispiel einer aktiven Dämpfungsregelung einer Busfederung wird untersucht, wie extern auftretende Störungen in den Lernprozess integriert werden können. Die Regelung von elektromagnetischen Schwebevorgängen ist ein weiteres Beispiel eines instabilen Systems mit hohen Nichtlinearitäten in der Systemdynamik. An diesem wird in Abschnitt 5.4 die Leistungsfähigkeit der selbstlernenden Regler an einer genauen Positionsregelung demonstriert. Diese ist eine zentrale Voraussetzung, die die technische Anwendung des Prinzips des magnetischen Schwebens in vielen Bereichen erst möglich macht. Ein Beispiel eines Prozesses mit externen, beobachtbaren Systemvariablen wird in Abschnitt 5.5 an der Regelung einer Heizspirale einer Klimaanlage demonstriert. Dies ist ein interessantes nichtlineares Regelungspro-

blem, bei dem die Regelgröße, auch bei sich extern verändernden Umgebungsbedingungen, sehr schnell wieder an den Sollwert angenähert werden muss.

5.1 Geschwindigkeitsregelung des Gleichstrom-Motors mit nichtlinearer Endstufe

Im Folgenden wird das Beispiel des DC-Motors mit nichtlinearer Endstufe aus Kapitel 3.1.2 aufgegriffen, um weitere Eigenschaften der vorgestellten Verfahren zu untersuchen. Wie in 3.1.2 beschrieben, wurde für das verwendete Simulationsmodell ein realer DC-Motor mit Endstufe vermessen und das Modell qualitativ an die beobachteten Nichtlinearitäten angepasst. In Abbildung 5.1 ist die Messung der sich ergebenden Grenzdrehzahl des realen Motors bei verschiedenen Spannungen (resp. PWM-Signalen) in drei unterschiedlichen Lastfällen zu sehen. Die verschiedenen Lastfälle ergeben sich durch die Anwendung einer externen, konstanten Last, die der Motor antreiben muss. Für die Messung wurde der statische Fall betrachtet, in dem eine gegebene Spannung zu der sich ergebenden Grenzdrehzahl aufgetragen wurde. Durch die Eigenschaften der Endstufe ergibt sich ein deutlich nichtlinearer Verlauf der Grenzdrehzahlen. Diese Nichtlinearität ist im idealen, linearen Modell des DC-Motors nicht vorhanden und würde dadurch in einem klassischen Reglerentwurf unberecksichtigt bleiben. Zusätzlich kann eine Totzone beobachtet werden, bei der die Spannung nicht ausreicht, um die Haftreibung im Getriebe und Motor zu überwinden. Die Messungen wurden an einem realen omnidirektionalen Roboter durchgeführt, bei dem die Lastverteilung, die auf die Motoren wirkt, durch die Fahrtrichtung des Roboters bestimmt wird. Im Simulationsmodell wurde dieses Verhalten

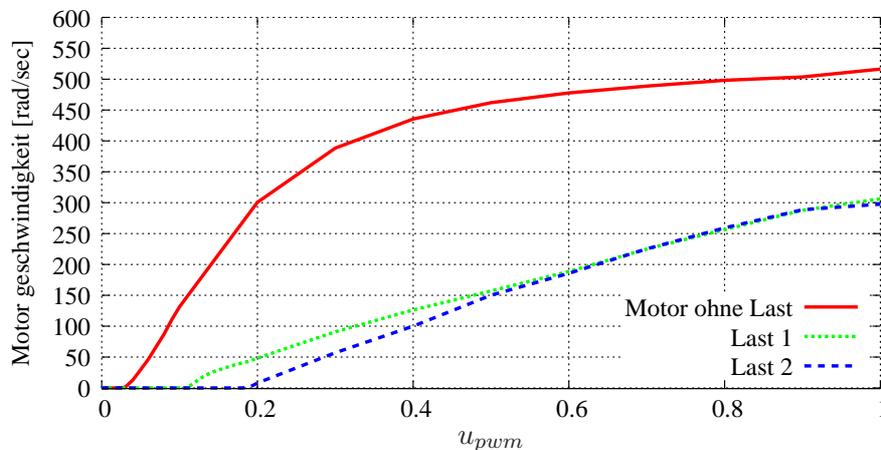


Abbildung 5.1: Gemessener Zusammenhang zwischen PWM-Eingriffsgröße u_{pwm} (konstant) und sich ergebender Grenzdrehzahl des Motors im ausbeschleunigten Zustand, an einem realen Motor unter drei verschiedenen Lastzuständen. Deutlich ist der vom Modell abweichende nichtlineare Verlauf der Grenzdrehzahl zu sehen, der durch die Endstufe des Motors zustande kommt.

durch eine nichtlineare Abbildung der Eingriffsgröße u_{pwm} auf die sich ergebende, wirksame Spannung U_w abgebildet (siehe Abbildung 5.2). Es wird somit eine nichtlineare Eingangskennlinie verwendet, die einem Hammerstein-Modell [45] entspricht. Die Abbildung der gemessenen Nichtlinearitäten ist nur eine Annäherung des realen Verhaltens des Motors. Hierbei wird davon ausgegangen, dass sämtliche beobachteten Nichtlinearitäten auf der Umsetzung des kommandierten PWM-Signals auf die wirksame Spannung am DC-Motor beruhen. Sie enthält

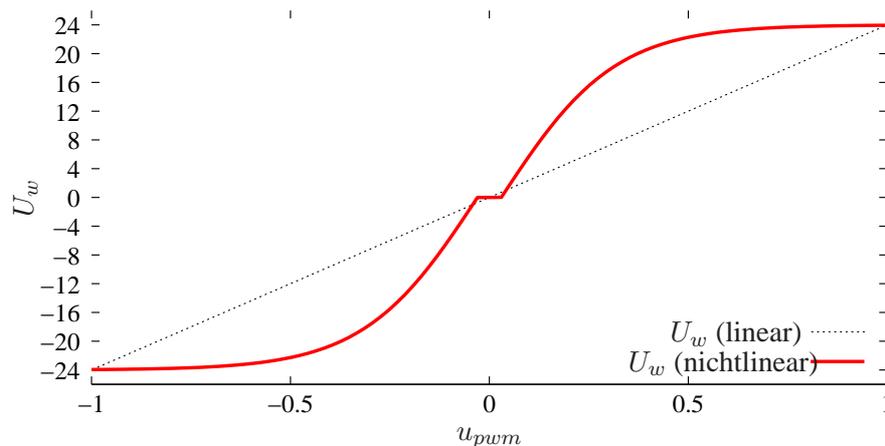


Abbildung 5.2: Nichtlineare Eingangskennlinie im simulierten DC-Motor. Das Eingangssignal wird in eine wirksame Spannung für das Modell umgerechnet. Im idealen Fall wäre diese Umrechnung die Identität. Diese nichtlineare Eingangskennlinie bildet die gemessenen Nichtlinearitäten im Simulationsmodell annäherungsweise ab.

Tabelle 5.1: Physikalische Modellparameter des DC-Motors für die Simulation.

ohmscher Widerstand der Armatuurwicklung	R_a	=	1.23	[Ohm]
Rotor-Induktivität	L	=	0.00034	[H]
emf-Konstante des Motors	k_{emf}	=	0.03881	[V / s / rad]
Drehmomentkonstante des Motors	k_m	=	0.03890	
Trägheitsmoment des Rotors	J	=	0.0001667	[kg m ² / s ²]
viskose Reibung	k_f	=	0.0001	[Nms]
externes Drehmoment	τ_{ap}	=	0	[Nm]
Betriebsspannung des Motors	vcc	=	24.00	[V]
Eingriffsgröße	u_{pwm}	∈	[-1.0, 1.0]	
Regelintervall	Δ_t	=	0.01	[s]

zusätzlich eine Totzone um den Nullpunkt des PWM-Signals. Dieses bewirkt, dass sehr kleine PWM-Signale nicht zur Dynamik des Motors beitragen. Die Simulation des Motors basiert auf den in Kapitel 3.1.2 angegebenen Systemgleichungen und wird mit numerischer Integration mit dem Runge-Kutta-Verfahren (4. Ordnung) der Systemgleichungen 3.3 und 3.5 mit einer Zyklusdauer von 10ms durchgeführt. Die physikalischen Standardparameter für die Versuche mit dem DC-Motor sind in Tabelle 5.1 zusammengefasst.

5.1.1 Geschwindigkeitsregelung mit kontinuierlichen Aktionen

Wie die Untersuchungen in Kapitel 3 gezeigt haben, ist es mit Hilfe von NFQ und unter Verwendung diskreter Aktionen möglich, einen selbstlernenden Regler zu erzeugen, der die Geschwindigkeit des DC-Motors auf beliebige Führungsgrößen regelt. Der direkte Einsatz der diskreten Aktionen hat jedoch den Nachteil, dass keine glatten Regeltrajektorien zustande kommen und der Regler die eigentlich benötigte Aktion durch das Anwenden von diskreten Aktionen über die Zeit annähern muss. Wie diskutiert, kann die Genauigkeit der Regler mit Hilfe der dynamischen Ausgabe-Elemente (DAE) erhöht werden. Trotz der Verwendung der DAE kann die eigentlich benötigte Eingriffsgröße jedoch nicht erreicht werden und muss durch ständiges Korrigieren des Reglers durch die vorhandenen diskreten Aktionen angenähert werden. Durch die Verwendung kontinuierlicher Aktionen mit dem NFQCA-Verfahren kann die Reglerqualität

Tabelle 5.2: Die für das Lernen verwendeten Parameter für die Geschwindigkeitsregelung des DC-Motors mit NFQCA.

Zustand x	3 dimensional	ω I $\tilde{\omega} = \omega_d - \omega$	Geschwindigkeit des Motors Strom in Armaturwicklung Regelabweichung
Kostenfunktion	\tanh^2	x^d μ C	$(-, -, 0)$ $(0, 0, 20)$ 0.01
Aktion u	1 dimensional	u_{pwm}	$\in [-1.0, 1.0]$
Q-Funktion	neuronal	Topologie	4-10-10-1 (400 Epochen RProp)
Strategie-Funktion	neuronal	Topologie	3-5-1 (600 Epochen RProp)
Exploration		ϵ Δ	0.30 $\sim N(0, 0.3)$
Interaktionsepisode		Länge	100 Schritte
Testepisode		Länge	100 Schritte

noch einmal deutlich verbessert werden. Für die Lernversuche mit kontinuierlichen Aktionen wird die gleiche Vorgehensweise wie bei den diskreten Aktionen verwendet. Es werden wiederum Standardparameter für die verwendeten neuronalen Netze und ihre Topologie eingesetzt. Zusätzlich wird ein Strategienetz mit einer verborgenen Schicht mit 5 Neuronen verwendet und die zufällige Exploration mit einer Normalverteilung um die aktuelle Strategie durchgeführt. Die Aktualisierung des Strategienetzes durch NFQCA wird mit 600 Epochen RProp durchgeführt. Dies ist wiederum ein Standardwert, der nicht durch extensive Versuche gefunden werden muss, da das Verfahren sehr robust gegen die Variation dieses Parameters ist. In Tabelle 5.2 sind die verwendeten Parameter des Lernversuches mit NFQCA zur Geschwindigkeitsregelung des DC-Motors aufgeführt. Mit Hilfe des NFQCA-Verfahrens kann mit wenigen Iterationen ein guter Regler auf beliebige Führungsgrößen eingelernt werden. Nach nur 30 Iterationen des Lernverfahrens erfüllt der gelernte Regler die geforderten Voraussetzungen auf allen 100 Testtrajektorien. Zu diesem Zeitpunkt hat der Regler 30 Interaktionsepisoden an 30 unterschiedlichen Führungsgrößen durchgeführt. Da der Regler mit einer zufälligen Strategie beginnt, ist die gesammelte Information über das System zu diesem Zeitpunkt bemerkenswert gering. Aus den gesammelten 3000 Übergängen am System, die unter den 30 Führungsgrößen gesammelt wurden, lernt der Regler eine sehr gute Regelstrategie. In Abbildung 5.3 ist der Verlauf der Gütekriterien \bar{N} und \bar{e}_∞ über die Iterationen des Lernverfahrens aufgetragen. Nach jeder Iteration des Lernverfahrens wird der Regler an 100 Trajektorien getestet. Auf jeder dieser Testtrajektorien wird die Führungsgröße zu Beginn aus dem Arbeitsbereich der Regelung zufällig gewählt und auf der Trajektorie konstant gehalten. Die Kennzahl für die bleibende Regelabweichung \bar{e}_∞ wird dabei wieder über die letzten 50 Schritte der Trajektorie bestimmt. Für ein Verfahren mit nichtlinearen Funktionsapproximatoren, ist der Verlauf der Gütekriterien über die Iterationen des Lernvorganges bemerkenswert stabil. Dies wird am Verlauf der Kurven in Abbildung 5.3 deutlich. Nach 30 Iterationen bleibt die Güte des Reglers in einem tolerierbaren Bereich. Durch das Ändern der Strategie während des Lernverlaufes kommen kleinere Ausreißer vor, bei denen die Reglergüte kurzfristig schlechter wird. Da bei der Regelung kontinuierliche Aktionen verwendet werden, können die für die Führungsgrößen benötigten Eingriffsgrößen vom Regler direkt erzeugt werden und müssen nicht durch die Verwendung diskreter Aktionen und DAE angenähert werden. In Abbildung 5.4 ist eine Regeltrajektorie des mit NFQCA gelernten Reglers zu sehen. Im Unterschied zu Abbildung 3.21 wird die vom Regler angewendete Aktion im ausgeregelten Zustand der Regeltrajektorie konstant. Der Regler lernt implizit für jede Führungsgröße eine zur Grenzdrehzahl passende Eingriffsgröße und erreicht somit einen Equi-

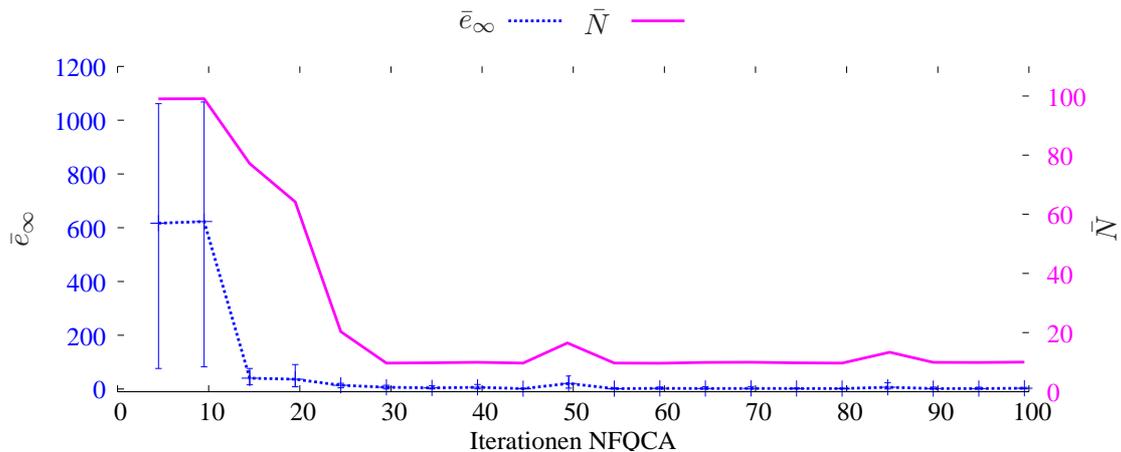


Abbildung 5.3: Der Lernverlauf des NFQCA-Verfahrens bei Anwendung auf die Geschwindigkeitsregelung des DC-Motors. Die Güte des Reglers ist bezüglich der Kennzahl für das Zeitkriterium \bar{N} und der bleibenden Regelabweichung \bar{e}_∞ über 100 Testtrajektorien mit zufällig gewählten Führungsgrößen aufgetragen. Nach wenigen Iterationen hat das Verfahren einen sehr guten Regler bezüglich der Genauigkeit und Schnelligkeit gelernt.

libriumspunkt der geschlossenen Systemdynamik, ohne die Stabilität über Zyklen herstellen zu müssen. Die Regelgüte der mit NFQCA gelernten Regler ist in Bezug auf die Genauigkeit noch

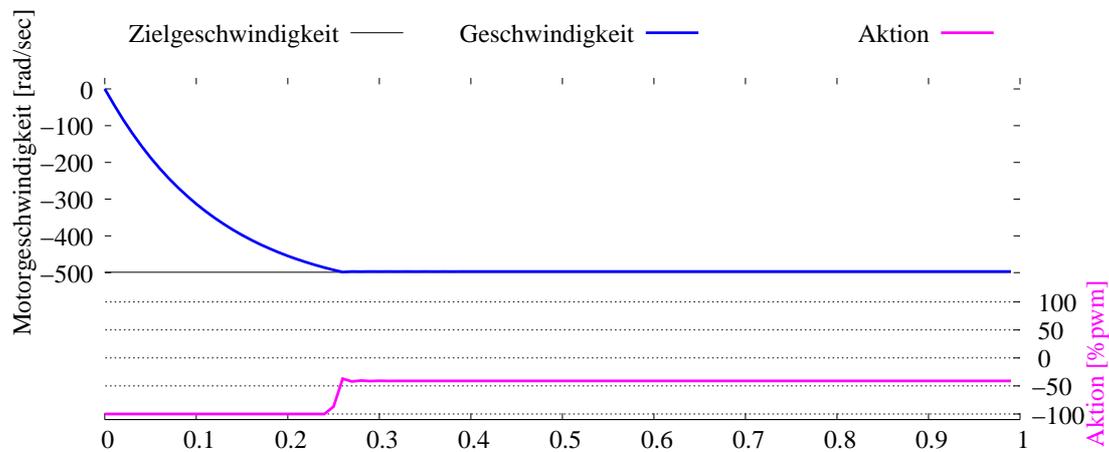


Abbildung 5.4: Eine Regeltrajektorie des mit NFQCA gelernten Reglers auf der Geschwindigkeitsregelung des DC-Motors. Im Unterschied zu der Verwendung diskreter Aktionen, können die für die Regelung benötigten Eingriffsgrößen vom Regler direkt erzeugt werden.

einmal besser als die mit diskreten Aktionen erzielbaren Ergebnisse. Die Regler sind genauer und schneller als die mit dynamischen Ausgabeelementen gelernten Regler. Die Verbesserung im Zeitkriterium wird durch die direkte Wahl der kontinuierlichen Eingriffsgröße begünstigt, die nicht über eine sequentielle Anwendung von diskreten Aktionen in einem Integrationsprozess des DAE erzeugt werden muss. Im Vergleich zu den Versuchen mit diskreten Aktionen, ist die Genauigkeit der Regler noch einmal deutlicher verbessert. Das Zeitkriterium ist in einem ähnlichen Bereich und verschlechtert sich zu den diskreten Aktionen nicht signifikant. In Tabelle 5.3 sind die Ergebnisse mit NFQCA zur Geschwindigkeitsregelung des DC-Motors,

Tabelle 5.3: Auswertung der Folgeregelung für den DC-Motor. Im Vergleich, die Ergebnisse mit diskreten Aktionen aus Kapitel 3 und die entsprechenden mit NFQCA und kontinuierlichen Aktionen erreichten Ergebnissen.

Verfahren	\bar{N}	\bar{e}_∞	\bar{e}_T Profil1	\bar{e}_T Profil2
NFQ 17 Aktionen	9.64	5.64	15.68	7.01
NFQ mit DAE	9.76	1.48	12.01	5.25
NFQCA	9.66	1.21	11.78	3.21

analog zu Kapitel 3, im Vergleich zu den dort erhaltenen Ergebnissen, aufgeführt. Die Vorteile

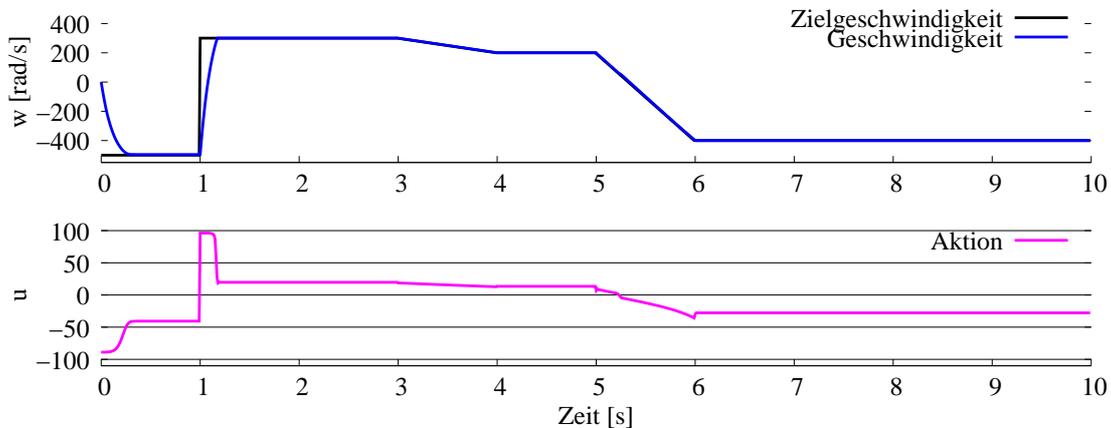


Abbildung 5.5: Die Folgeregelung des DC-Motors durch den mit NFQCA gelernten Regler (Profil2). Die Führungsgröße variiert über 1000 Zeitschritte der Regelung. Zu Beginn sind zwei Sprünge der Führungsgröße enthalten, die schneller variieren als die Systemdynamik des Motor es zulässt. Der Regler muss diese Sprünge schnellstmöglich ausgleichen. Danach wird die Führungsgröße über einen weiten Bereich des Arbeitsbereiches der Regelung hinweg variiert. Der Regler lässt die Motorgeschwindigkeit der Führungsgröße sehr genau folgen. Die vom Regler gewählte Eingriffsgrößen sind kontinuierlich und besitzen eine gewisse Glattheit (zum Vergleich siehe Abbildung 3.17), was für die Regelung von Vorteil ist.

der Verwendung der kontinuierlichen Aktionen zeigen sich vor allem bei der Folgeregelung auf das Profil der Führungsgrößen in Abbildung 5.5. Dies ist dasselbe Profil, wie in Kapitel 3 verwendet wurde. Der Regler, eingelernt in Interaktionsepisoden mit konstanten Führungsgrößen, ist in der Lage, die Folgeregelung auf das gesamte Profil der Führungsgrößen mit hoher Genauigkeit zu regeln. Der erzielte Trajektorienfehler ist durch die Verwendung der kontinuierlichen Aktionen noch einmal deutlich kleiner geworden. Für die Auswertung der weiteren Versuche wird ein zusätzliches Profil von Führungsgrößen definiert (Profil 2). Dieses ändert die Führungsgröße über den gesamten Arbeitsbereich der Regelung und ist in Abbildung 5.6 abgebildet. Die Führungsgröße variiert sinusförmig zwischen -400 und 400 rad/s. Die Regelung beginnt ohne Regelabweichung. Der gelernte Regler ist in der Lage, der Führungsgröße über den gesamten Bereich mit hoher Genauigkeit zu folgen. Die vom Regler beeinflusste Regelgröße folgt der variierenden Führungsgröße über die gesamte Dauer der Trajektorie von 1000 Zeitschritten. Der Regler erzeugt einen mittleren Trajektorienfehler von $\bar{e}_T = 3.21$ auf dem sinusförmigen Profil der Führungsgröße. Dieser Fehler ist höher als der Fehler bei Regelung ohne variierende Führungsgröße. Dies ist ein grundsätzliches Problem der Regelung

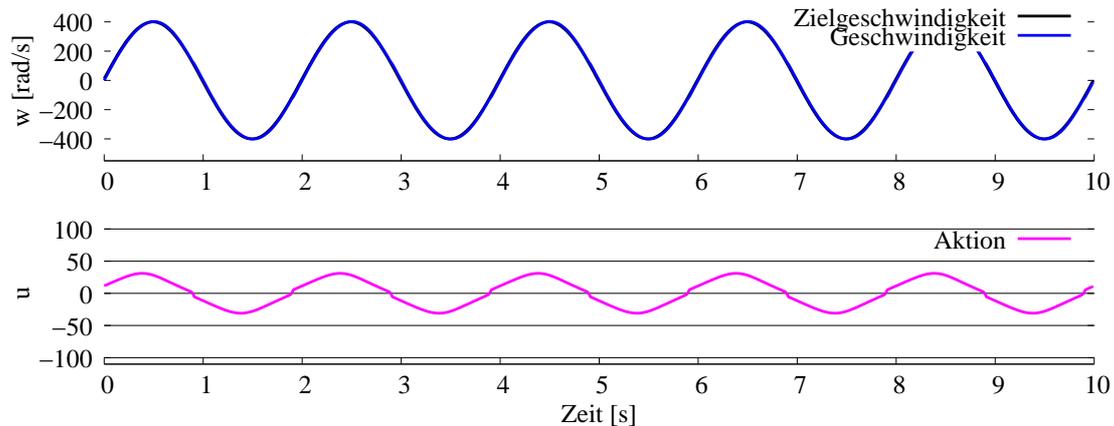


Abbildung 5.6: Die Folgeregelung des DC-Motors durch den mit NFQCA gelernten Regler am Beispiel einer sinusförmig variierenden Führungsgröße (Profil2). Bei diesem Profil sind keine Sprünge in der Führungsgröße vorhanden, wodurch die Dynamik des geschlossenen Systems der Führungsgröße direkt folgen kann.

über reine Rückführung der Regelabweichung, da die Dynamik der Führungsgröße erst durch die Regelabweichung erkannt werden kann. Eine Abhilfe kann hier die Anwendung der Trajektorienregelung schaffen, die an einem anderen Versuch diskutiert wird. Die selbstlernenden Regler mit NFQCA sind somit in der Lage, effizient und in kurzer Zeit ein präzises und schnelles Regelverhalten für die Geschwindigkeitsregelung des DC-Motors mit deren nichtlinearen Systemeigenschaften zu erzeugen. Die Anzahl der Interaktionsepisoden ist mit deutlich unter 100 Episoden, von denen jede nur 1 Sekunde dauert, sehr gering. Bei 100 Interaktionsepisoden ist die benötigte reine Interaktionszeit, mit weniger als 2 Minuten, für die Anwendung auf reale Systeme geeignet. Trotz dieser kurzen Interaktionszeit und den wenigen beobachteten Führungsgrößen kann der gelernte Regler den gesamten Arbeitsbereich der Regelung abdecken. Auch die Folgeregelung mit kontinuierlich variierender Führungsgröße ist mit dem so eingelernten Regler möglich.

5.1.2 Versuche mit variabler externer Last

In vielen regelungstechnischen Anwendungen tritt das Problem auf, dass die Regelung robust gegen extern auftretende und unbekannte Kräfte sein muss. Diese Kräfte entstehen zum Beispiel durch unbekannte Lasten, die bewegt werden müssen. Für den selbstlernenden Regler sind diese Lasten nicht direkt beobachtbar. Die Änderung der Systemdynamik jedoch ist in den Daten vorhanden und kann vom Regler beobachtet werden.

Der DC-Motor wurde bisher mit einer Systemdynamik ohne externe Kräfte betrachtet. Die Regelung bezog sich ausschließlich auf die Systemdynamik des freilaufenden Motors. Analog zu dem Fall mit konstanter externer Last $\tau_{ap} = 0$, kann in den bisherigen Lernversuchen ein beliebiger konstanter Lastfall analog betrachtet werden. In den folgenden Versuchen wird untersucht, wie sich die selbstlernenden Regler auf ein System mit variierender externer Last anwenden lassen. Der DC-Motor besitzt ein maximales Drehmoment von ca. 720mNm. Bei einer externen Belastung von 200mNm sinkt die maximale Geschwindigkeit, die gegen das externe Drehmoment erreicht werden kann, auf ca. 430 rad/s. Für die Versuche wird der zulässige Bereich der Führungsgröße auf eine Geschwindigkeit von -400 bis 400 rad/s beschränkt. Die

Tabelle 5.4: Die für das Lernen unter Last verwendeten Parameter für die Geschwindigkeitsregelung des DC-Motors. Angegeben sind die Parameter, die sich zum Versuch ohne Last ändern.

Zustand x	4 dimensional	ω I u_{t-1} $\tilde{\omega} = \omega_d - \omega$	Geschwindigkeit des Motors Strom in Armaturwicklung letzte Aktion (PWM-Signal) Regelabweichung
Kostenfunktion	\tanh^2	x^d μ C	$(-, -, -, 0)$ $(0, 0, 0, 20)$ 0.01
Q-Funktion	neuronal	Topologie	5-10-10-1 (400 Epochen RProp)
Strategie-Funktion	neuronal	Topologie	4-5-1 (600 Epochen RProp)

extern auftretende Last ist repräsentiert durch ein externes Drehmoment und bewegt sich in einem Bereich von -200 bis 200 mNm. Für die Bestimmung der Regelgüte \bar{e}_T mit externer, variierender Last werden die bereits verwendeten Führungsgrößen-Profile verwendet. Für jedes Profil wird zusätzlich der Verlauf der externen Last über die Zeit definiert.

In einem ersten Schritt soll die Robustheit der ohne externe Last gelernten Regelstrategie bei Auftreten externer Last untersucht werden. Die externe Last kann dabei als auftretende Störung interpretiert werden. Die Bereiche des Zustandsraums, die betreten werden, hat der Regler in der Lernphase nie gesehen. Folglich verschlechtert sich die Regelgüte im Vergleich zu den lastfreien Profilen. Bemerkenswert ist jedoch die Robustheit bei der Ausführung auf nicht gesehene Systemzusammenhänge. Wie in Tabelle 5.5 zu sehen ist, kommt auf beiden Profilen immer noch eine akzeptable Regelgüte zustande. Dies zeigt sich vor allem bei Profil 2, da dieses die genaue Ausregelung im Rahmen der Systemdynamik zulässt. Mit einer mittleren absoluten Abweichung von 7.38 (3.21 im lastfreien Fall) hält sich der Regler im vorgegebenen Toleranzbereich der Regelung.

Lernen unter veränderlicher Last

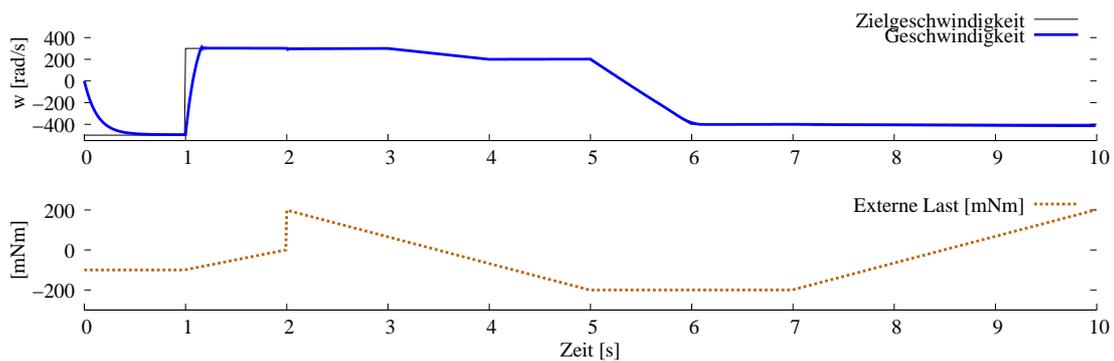
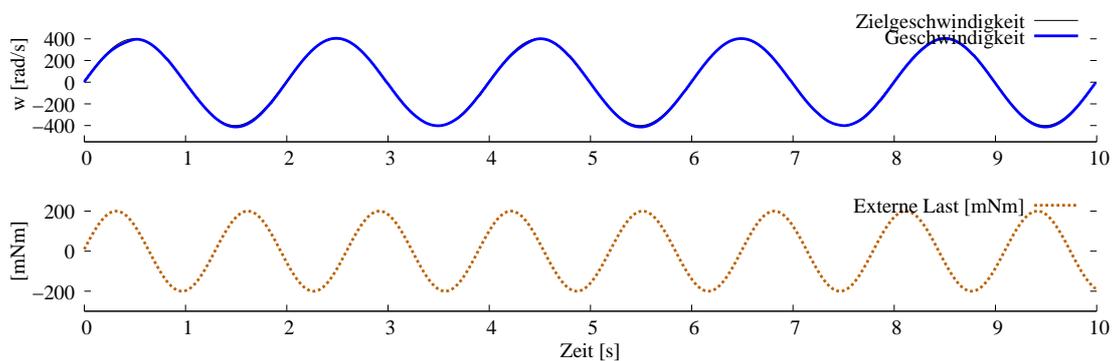
Im Folgenden wird ein selbstlernender Regler mit NFQCA für verschiedene Lastsituationen am System eingelernt. Die externe Last selbst ist nicht messbar, sie ist somit für den Regler nicht direkt im Zustandsvektor beobachtbar. Die Veränderung der Last während einer Interaktionsepisode würde dazu führen, dass die beobachtete Systemdynamik direkt von der Dynamik der Last abhängig ist. Dabei ist es für den Regler nicht möglich, zwischen der Dynamik des Lastwechsels und der Systemdynamik zu unterscheiden. Das Ziel ist ein robuster Regler, der mit verschiedenen Lastsituationen zurecht kommt. Um die Systemdynamik und die Lastdynamik zu trennen, wird während jeder Interaktionsepisode die externe Last konstant gehalten. Ein Lastwechsel findet zwischen den Interaktionsepisoden statt. Für die Lernversuche wird der Regler in jeder Interaktionsepisode aus Geschwindigkeit 0 gestartet. Jede Interaktionsepisode wird mit einer konstanten, zufällig gewählten Zielgeschwindigkeit und einem konstanten, ebenfalls zufällig gewählten Lastwert 100 Schritte lang durchgeführt. Die Ausführung des gelernten Reglers hingegen kann mit veränderlicher Last durchgeführt werden.

Der lernende Regler beobachtet in verschiedenen Lastsituationen unterschiedliches Verhalten der Dynamik des Systems. Die externe Last bestimmt somit einen speziellen Anwendungsfall, für den sich die Systemdynamik unterschiedlich verhält. Zur Modellierung des Zustandes für das Reinforcement-Lernen muss nun sichergestellt werden, dass sich die verschiedenen Lastfälle für den Regler anhand des Zustandes geeignet unterscheiden lassen. Dies macht die Hinzunahme von weiteren Systemvariablen notwendig. Für den DC-Motor wird dies dadurch

Tabelle 5.5: Versuchsauswertung auf Profilen. Mittlere Abweichung über die Trajektorie von 1000 Schritten bei unterschiedlichen Last und Zielwert Profilen.

Regler	\bar{e}_T Profil 1	\bar{e}_T Profil 2
NFQCA ohne Last gelernt	18.09	7.38
NFQCA mit Last gelernt	15.34	5.41

realisiert, dass zusätzlich die im vorigen Zeitschritt angewendete Eingriffsgröße in den Zustand des Reglers übernommen wird. Der Zustand für den lernenden Regler besteht somit aus 4 Systemvariablen ($\omega_t, I_t, \tilde{\omega}_t, u_{t-1}$), der Geschwindigkeit ω_t , dem Strom I_t , dem Fehlersignal $\tilde{\omega}_t = \omega_d - \omega$ und der Eingriffsgröße im letzten Zeitschritt u_{t-1} . Die Kombination aus diesen Systemvariablen sollte es dem Regler ermöglichen, die verschiedenen Lastfälle im Zustandsraum geeignet zu unterscheiden. Für die durchgeführten Versuche wird folglich die Anzahl der Eingabeneuronen bei der neuronalen Q-Funktion und bei der Strategiefunktion jeweils um Eins erhöht (siehe Tabelle 5.4). Die restlichen Parameter des Lernversuches bleiben konstant, wie sie weiter oben aufgeführt wurden. Die erzielbare Regelgüte ist in Tabelle 5.5 für beide Profile aufgetragen. Durch das Lernen am System mit veränderlicher Last lässt sich die Reglergüte deutlich steigern. Der beste Regler wird nach 62 Iterationen mit NFQCA erzeugt. In Abbildung

**Abbildung 5.7:** Der mit NFQCA gelernte Regler auf dem Führungsgröße-Last-Profil 1.**Abbildung 5.8:** Der mit NFQCA gelernte Regler auf dem Führungsgröße-Last-Profil 2.

5.7 und 5.8 sind die Regeltrajektorien des gelernten Reglers auf beiden Führungsgröße-Last-Profilen dargestellt. Die Genauigkeit der Regler erreicht nicht die Güte wie im lastfreien Fall.

Dies ist vor allem der Struktur der Regler zuzuschreiben, da sie erst auf eine Veränderung der Last reagieren können, wenn sich diese in der Systemdynamik auswirkt und damit schon einen Fehler erzeugt hat.

In Abbildung 5.9 wurde der Regler unter zwei konstanten Lastfällen auf das sinusförmige

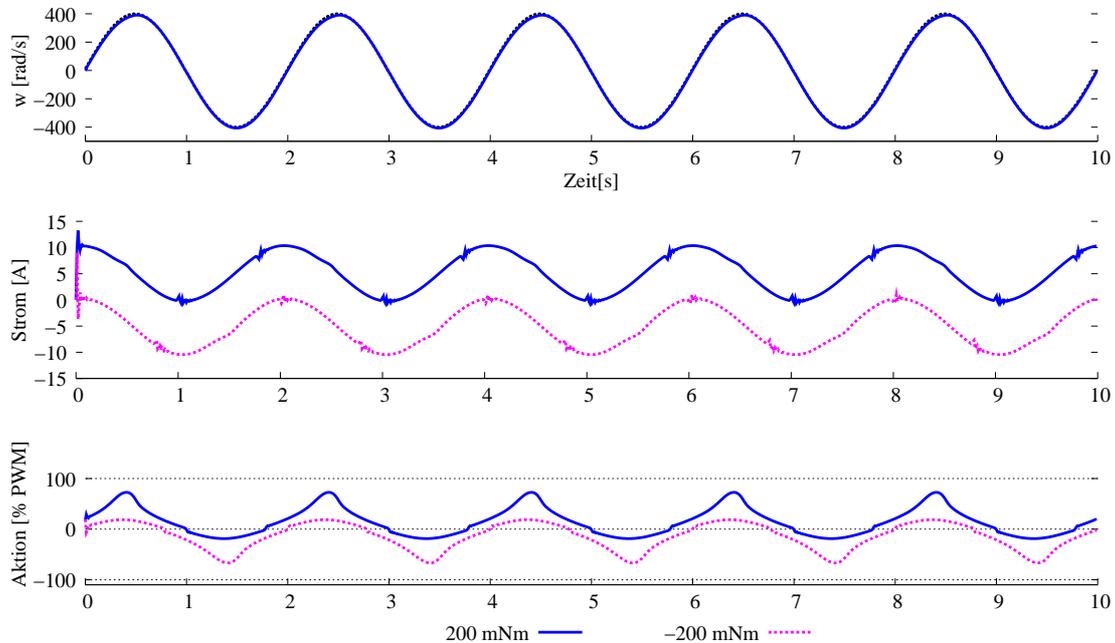


Abbildung 5.9: Die Regeltrajektorie für das sinusförmige Profil der Führungsgröße unter zwei konstanten Lastfällen von 200mNm und -200mNm. Das Führungsprofil und die Regeltrajektorie der Motorgeschwindigkeit sind für den Lastfall 200mNm aufgetragen. Der Regler erzeugt die gleiche Trajektorie für den Lastfall -200mNm. Der Strom und die vom Regler gewählte Aktion sind jeweils für den entsprechenden Lastfall aufgetragen. Die gestrichelte pinke Kurve ist der Lastfall von -200mNm, die blaue durchgezogene Linie der Fall von 200mNm.

Profil der Führungsgröße angewendet. Dargestellt ist jeweils der Verlauf des Stromes und der gewählten Eingriffsgröße im letzten Zeitschritt für den konstanten Lastfall von 200mNm und den konstanten Lastfall von -200mNm. Die Geschwindigkeit des Motors folgt in beiden Fällen der Führungsgröße, womit die Regelabweichung zu jedem Zeitpunkt ebenfalls ähnlich ist. Um die zwei Lastfälle im Zustandsraum zu unterscheiden, kann der Regler diese drei Informationen benutzen. In Abbildung 5.10 sind die zwei Trajektorien über dem gesamten Zustandsraum aufgetragen. Da sie zusammen den Zustandsvektor des MDPs bilden, kann der Regler anhand des Zustands unterscheiden, in welchem Lastfall er sich gerade befindet. Über die Generalisierungsfähigkeit der lernenden Regler ist die Regelgüte auch für nicht gesehene Lastfälle, wie sie in den präsentierten Profilen vorhanden sind, gut. Die Anzahl der benötigten Interaktionsepisoden erhöht sich, liegt jedoch mit 60 Iterationen immer noch unter 4 Minuten reiner Interaktionszeit.

5.1.3 Robustheit gegen Rauschen

Alle in der Realität betrachteten Prozesse besitzen ein gewisse Unsicherheit in ihren erfassbaren Systemvariablen und ihrem Prozessverhalten. Zum Einen kann diese Unsicherheit eine Eigenschaft des betrachteten Prozesses an sich sein, bzw. von nicht erfassten Systemvariablen

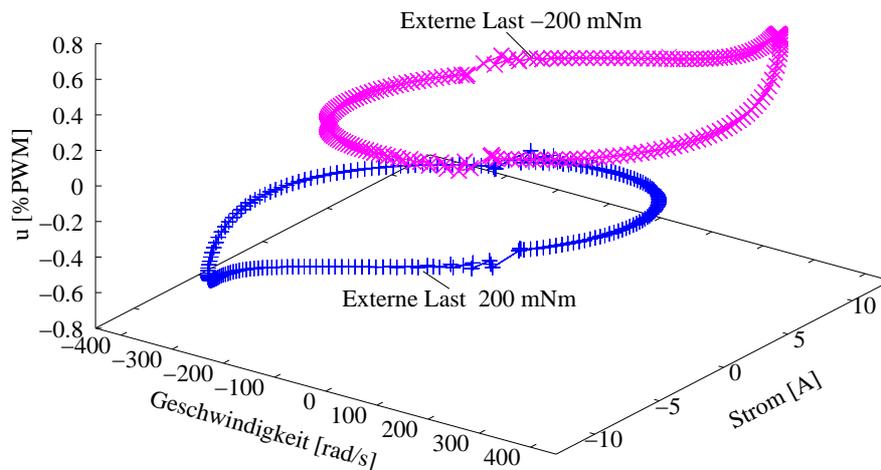


Abbildung 5.10: Der Verlauf der Zustandsvariablen für das sinusförmige Profil der Führungsgröße unter zwei konstanten Lastfällen von 200mNm und -200mNm. Für den Regler sind die Lastfälle im Zustandsraum trennbar.

verursacht werden. Zum Anderen beruhen solche Unsicherheiten oft auf der Qualität der Messeinrichtung des Prozesses und auf deren Messprinzipien. An dieser Stelle werden die Verfahren zur Erstellung selbstlernender Regler deshalb nun auf den Einfluss von Rauschen untersucht, wie es beim Einsatz an realen Systemen vorkommt.

Am Beispiel des NFQCA-Verfahrens wird im Folgenden an der Geschwindigkeitsregelung des DC-Motors untersucht, wie sich verschiedene Arten von Rauschen auf die erzielbare Regelgüte und den Lernverlauf des Verfahrens auswirken. Hierzu werden in das simulierte Modell des DC-Motors verschiedene Arten von Rauschen integriert. Betrachtet werden sowohl Messrauschen als auch Prozessrauschen. Das Messrauschen beeinflusst ausschließlich die Beobachtung der Prozessvariablen des lernenden Reglers, nicht jedoch die Prozessvariablen des Systems und damit die Übergänge im System selbst. Dies repräsentiert ein durch Rauschen beeinflusstes Messsystem, wie es zum Beispiel bei Analog-Digital Wandlern häufig vorkommt. Im Fall des Prozessrauschens werden direkt die Systemübergänge verrauscht. In beiden Fällen wird der Beobachtung bzw. der Systemvariablen selbst in jedem Zeitschritt eine normalverteilte Zufallsvariable aufaddiert. In beiden Fällen soll die Geschwindigkeit des Motors (und somit auch die Regelabweichung) des Systems in jedem Schritt mit Mittelwert 0 und Standardabweichung $\sqrt{100}$, der Strom mit Mittelwert 0 und Standardabweichung $\sqrt{10}$, durch additives Rauschen verändert werden. Für die Versuche werden drei unterschiedliche Fälle untersucht. Das separate Auftreten von Messrauschen ohne Prozessrauschen, das separate Auftreten von Prozessrauschen ohne Messrauschen, sowie ein kombiniertes Rauschen aus beiden Anteilen.

Im ersten Schritt ist die Robustheit des am unverrauschten System eingelernten Reglers mit NFQCA bei der Ausführung am verrauschten System von Interesse. Dieses Szenario ist für die praktische Anwendung der Regler dann relevant, wenn sich nach dem Lernen der Regler die Systemdynamik in Bezug auf das Rauschen ändert. Eine solche Änderung ist beim Einsatz eines an der Simulation gelernten Reglers an einem realen System zu erwarten. Hauptsächlich soll diese Betrachtung jedoch der Robustheit der gelernten Regler dienen. Robustheit bedeutet in diesem Fall, dass sich durch das Rauschen für den Regler Systemzustände ergeben, die während des Lernvorganges noch nie betreten wurden. Ein Beispiel solch eines Systemzustandes sind Kombinationen von Motorströmen und Motorgeschwindigkeiten, die im unverrauschten

Tabelle 5.6: Versuchsauswertung der Regler mit Rauschen. In der linken Spalte sind die Regler und die Bedingungen, wie sie eingelernt wurden, aufgetragen. Die Spaltenüberschriften bezeichnen die Testbedingung und die erzielte Regelgüte für den entsprechenden Regler. Kleinere Werte bedeuten bessere Ergebnisse, da ein geringerer Fehler auf der Folgeregelung erzeugt wird.

Regler	\bar{e}_T Prozessrauschen	\bar{e}_T Messrauschen	\bar{e}_T kombiniert
NFQCA gelernt ohne Rauschen	19.44	22.34	25.23
NFQCA Prozessrauschen	18.90	20.82	23.93
NFQCA Messrauschen	19.79	19.44	23.15
NFQCA kombiniert	19.40	19.87	23.39

System nicht auftreten. Die Betrachtung wird an dem aus den vorigen Kapiteln bekannten Profil 1 der Führungsgrößen ohne externe Last durchgeführt.

Die ohne Rauschen gelernten Regler erreichen bei allen Rauscharten immer noch das Regelungsziel. Die detaillierten Ergebnisse sind weiter unten in Tabelle 5.6 zusammengefasst. Wie zu erwarten ist, wird beim kombinierten Rauschen der höchste Trajektorienfehler von $\bar{e}_T = 25.23$ erzeugt. Unter Prozessrauschen erzeugt der Regler mit $\bar{e}_T = 19.44$ einen deutlich kleineren Trajektorienfehler, als unter kombiniertem Rauschen und auch einen kleineren Trajektorienfehler als unter Messrauschen, mit $\bar{e}_T = 22.34$, erreicht wird. Dies ist in dieser Form auch zu erwarten, da beim Prozessrauschen die durch die beobachteten Regelabweichungen vom Regler erzeugten Eingriffsgrößen eine Korrektur bewirken, wobei sie beim Messrauschen unter Umständen eine Verschlechterung der Regelabweichung erzeugen, wenn die Beobachtung vom aktuellen Wert abweicht. Grundsätzlich kann der gelernte Regler somit mit verschiedenen Arten von nachträglich auftretendem Rauschen immer noch zufriedenstellend umgehen und ist somit in gewissen Grenzen robust. Das hier betrachtete Rauschen ist dabei additives, weißes Rauschen und beinhaltet somit keine systematischen Abweichungen. Im Unterschied zu den klassischen Entwurfsmustern ist die Betrachtung eines systematischen Fehlers bei selbstlernenden Systemen nicht notwendig, da sie direkt durch die Interaktion mit dem System gelernt werden und somit nicht systematischen Modellierungsfehlern unterliegen.

Lernen unter Rauschen

Nachdem die Robustheit der Regler bezüglich nachträglich hinzugefügtem Rauschen im System untersucht wurde, ist nun eine wichtige Frage, ob auch das Lernen unter dem Vorhandensein von Rauschen im System möglich ist. Dies soll mit dem vorgestellten NFQCA-Verfahren an der Geschwindigkeitsregelung des DC-Motors untersucht werden. Im Speziellen ist die Frage, ob das entwickelte Verfahren in der Lage ist, Optimierungsprobleme, basierend auf stochastischen MDPs zu lernen. Für die Lernversuche werden die gleichen Standardparameter verwendet wie für die Versuche ohne Rauschen in Abschnitt 5.1.1. Es wird jeweils ein selbstlernender Regler mit NFQCA für alle drei Rauscharten eingelernt. Der beste Regler in jedem Lernversuch wird auf den anderen zwei Rauscharten getestet. In Tabelle 5.6 sind die detaillierten Ergebnisse der Versuche als Auswertung an dem Profil 1 der Führungsgrößen aufgetragen. Das Verfahren ist in der Lage, für alle drei Rauscharten einen Regler einzulernen und die erzielbare Regelgüte im Vergleich zu dem Regler, der ohne Rauschen eingelernt wurde, deutlich zu steigern. Die Anzahl der benötigten Iterationen des Lernverfahrens, um den Regler einzulernen, erhöhen sich durch das Rauschen. Bei dem Versuch mit Prozessrauschen ist diese Erhöhung nur marginal, nach 40 Iterationen des Lernverfahrens wird schon ein Wert von

$\bar{e}_T = 18.94$ erreicht, was schon eine deutliche Verbesserung darstellt. Durch die Einführung des Rauschens in das System kann das Verfahren mit deutlich mehr Iterationen jedoch noch eine geringe Verbesserung erzielen. So sinkt der Fehler beim Prozessrauschen bei Iteration 217 des Lernversuches auf das beste Ergebnis von $\bar{e}_T = 18.90$. Das Messrauschen verzögert das Lernen etwas mehr. Nach 40 Iterationen ist der Wert bereits unter $\bar{e}_T = 23.4$, kann aber erst nach 93 Iterationen des Lernverfahrens unter den Vergleichswert von $\bar{e}_T = 22.34$ gebracht werden. Bei Iteration 256 wird dann das Minimum von $\bar{e}_T = 19.44$ erreicht. Im Fall des kombinierten Rauschens wird bereits nach 31 Iterationen der Vergleichswert erreicht. Auch hier kann durch deutlich mehr Iterationen noch eine Verbesserung erzielt werden. So wird der beste Wert bei Iteration 225 mit $\bar{e}_T = 23.39$ erreicht. Indem der Regler direkt in Interaktion mit dem verrauschten System eingelernt wird, kann seine Regelgüte somit deutlich gesteigert werden. Ein erfreuliches Ergebnis ist weiterhin, dass sich beim Prozessrauschen und kombinierten Rauschen auch die Güte des Reglers bei Anwendung auf die anderen Rauscharten steigert. Der Regler, der unter Messrauschen eingelernt wurde, kann die Regelgüte bei Ausführung am System mit Messrauschen und kombiniertem Rauschen ebenfalls deutlich steigern, erzeugt auf dem System mit Prozessrauschen jedoch eine etwas schlechtere Güte als der Vergleichsregler, der ohne Rauschen eingelernt wurde. In Abbildung 5.11 ist eine Trajektorie eines Reglers unter

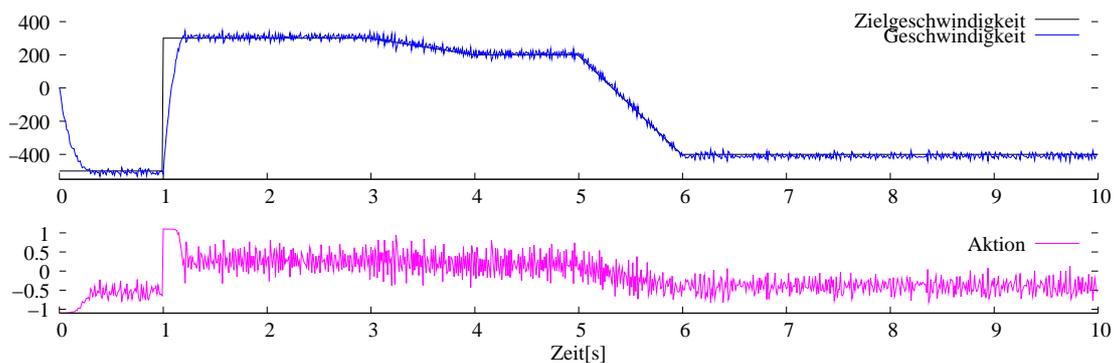


Abbildung 5.11: Der mit NFQCA unter Messrauschen gelernte Regler.

Messrauschen zu sehen. Die gelernte Strategie unterliegt dem Rauschen des Systems und ist daher ebenfalls nicht mehr so glatt, wie sie am System ohne Rauschen war. Dennoch wird der Zielwert auf der Folgeregelung erreicht und die Geschwindigkeit um diesen Zielwert herum gehalten. Ein ebenfalls interessantes Ergebnis ergibt die Anwendung der am verrauschten System eingelernten Regler auf die Systemdynamik ohne Rauschen. In diesem Fall erzeugen sie eine ähnliche Regelgüte wie die direkt am System ohne Rauschen eingelernten Regler.

5.1.4 Diskussion

Die Versuche zur Geschwindigkeitsregelung des DC-Motors mit nichtlinearer Endstufe zeigen mehrere für die regelungstechnische Praxis interessante Eigenschaften der selbstlernenden Regler mit NFQ und NFQCA. Mit wenigen Interaktionsepioden und ohne die Integration von Vorwissen über den zu regelnden Prozess ist es möglich, sehr effiziente und robuste Regler zu lernen. Die gelernten Regler zeigen im gesamten Arbeitsbereich der Regelung, auch bei der Folgeregelung auf variierende Führungsgrößen, eine gute Regelgüte. Insbesondere durch die Anwendung des NFQCA-Verfahrens lassen sich Regler mit einer sehr kleinen bleibenden Regelabweichung lernen. Auch die Anwendung auf Systeme mit einem hohen Rauschanteil ist

für die gelernten Regler durch die Art der Modellierung und ihre Eigenschaften möglich. Es wurde gezeigt, dass NFQCA in der Lage ist, auch auf stochastische MDPs mit kontinuierlichen Aktionen zu arbeiten. Am Beispiel der variablen externen Last wurde gezeigt, dass es unter bestimmten Voraussetzungen möglich ist, einen geschlossenen Regler für verschiedene, extern beeinflusste und nicht direkt beobachtbare Arbeitspunkte der Regelung zu erlernen. Die Versuche mit veränderlicher Last und Rauschen zeigen insbesondere die Robustheit der Regler bei Störung der Systemdynamik. Dies ist nicht selbstverständlich, da die Regler im Zustandsraum arbeiten und für diesen Zustandsraum während der Interaktionsepisoden im Vergleich zur Größe des Zustandsraumes wenige Beispiele der Systemdynamik beobachten konnten.

Ein aus der Praxis stammendes Anwendungsbeispiel ist die Geschwindigkeitsregelung der DC-Motoren des omnidirektionalen mobilen Roboters in Abbildung 5.12. Dieser wird im



Abbildung 5.12: Der omnidirektionale mobile Roboter aus dem Middle-Size Wettkampfteam der Brainstormer Tribots. Der Roboter wird angetrieben von 3 Motoren. Die schnelle und robuste Geschwindigkeitsregelung der Motoren ist die zentrale Voraussetzung für alle Verhalten des Roboters.

'Middle Size Team' der Brainstormer Tribots im RoboCup seit mehreren Jahren erfolgreich eingesetzt [38]. Die schnelle und robuste Geschwindigkeitsregelung der Motoren des Roboters ist hierbei eine zentrale Voraussetzung, auf der alle höher angesiedelten Fähigkeiten des Roboters basieren. Der Roboter wird angetrieben durch drei Motoren, die jeweils ein omnidirektionales Rad antreiben. Durch die Geschwindigkeiten der drei Räder ergibt sich direkt die Fahrtrichtung und Geschwindigkeit des Roboters auf dem Untergrund. Die Ansteuerung der drei Motoren erfolgt durch die in 3.1.2 vorgestellte Kombination aus einem Mikrocontroller mit Endstufe, die durch einen PC über Can-Bus kommandiert werden kann. Im Unterschied zu dem in Abbildung 3.3 vorgestellten Grundaufbau kann der Mikrocontroller drei unabhängige DC-Motoren parallel betreiben. Bei diesem Anwendungsbeispiel treten die in den Versuchen zum DC-Motor betrachteten Schwierigkeiten auf. Durch das Zusammenwirken der drei Motoren im Roboter treten dynamisch wechselnde Lastzustände für jeden Motor auf, die der Regler

selbst nicht beobachten kann. Zusätzlich wird durch die Form der verwendeten Allseitenräder ein sehr hohes Rauschen erzeugt. Anstatt einen Regler für jeden DC-Motor einzulernen, wird nur ein einziger Regler eingelernt, der auf jeden Motor separat angewendet wird. Dies wird durch die symmetrischen Eigenschaften des Roboters und die baugleichen DC-Motoren sowie Endstufenkanäle möglich. Aus Sicht des Roboters existiert somit ein Regler, der in jedem Zeitschritt unabhängig für jeden Motor angefragt werden kann. In der Reglersicht existiert nur ein Motor, für den, basierend auf seinem aktuellen Zustand, in verschiedensten Lastsituationen eine geeignete Aktion berechnet werden muss. An dem realen Roboter wurden in 50 Interaktionsepisoden mit 5 Sekunden Länge Daten gesammelt. Dies entspricht 7500 beobachteten Übergängen pro Motor. Da der Regler die Daten parallel an allen drei Motoren sammelt, stehen zum Lernen 22500 Übergänge zur Verfügung, die in etwas weniger als 4 Minuten gesammelt wurden. Dieser Prozess des Datensammelns wurde mit einem rein zufällig initialisierten Regler durchgeführt. Anschließend kann mit rein datenbasiertem NFQ in nur 30 Iterationen ein Regler eingelernt werden, der sehr gute Eigenschaften auf dem Roboter zeigt. In Abbildung 5.13 ist

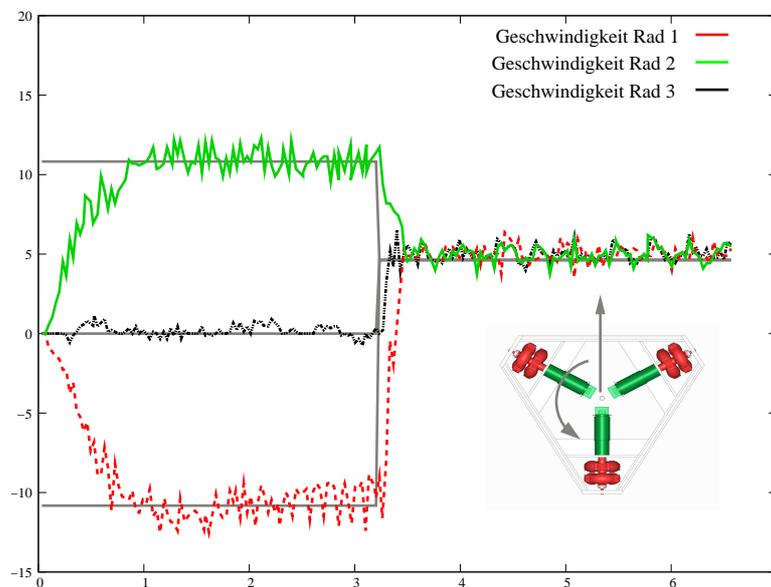


Abbildung 5.13: Eine Regeltrajektorie des gelernten Reglers, angewendet auf die Geschwindigkeitsregelung des omnidirektionalen Roboters. Die spezielle Form der Räder erzeugt ein extrem hohes Maß an Rauschen. Der gelernte Regler ist trotzdem in der Lage, dieses Rauschen auszugleichen und die Geschwindigkeit auf der Führungsgröße zu halten. Das Beispiel zeigt eine Regelung auf 0.5 m/s nach vorne, gefolgt von einer reinen Rotationsbewegung von 2 rad/s.

eine Regeltrajektorie des omnidirektionalen Roboters mit dem gelernten Regler zu sehen. Die globalen Fahrbefehle, die hier zur Demonstration genutzt werden, sind 'fahre vorwärts mit 0.5 $\frac{m}{s}$ ' und anschließend 'drehe mit 2 $\frac{rad}{s}$ '. Zur Berechnung der Zielgeschwindigkeiten der einzelnen Motoren wurde das inverse kinematische Modell des Roboters benutzt. Der neuronale Regler ist in der Lage, die Geschwindigkeit der Motoren schnell und zuverlässig auf ihre Zielgeschwindigkeit zu regeln. Ein sehr erfreuliches Ergebnis ist, dass der Regler sehr zuverlässig in dem großen Bereich der auftretenden Lastzustände arbeitet, die während der Bewegung des realen Roboters entstehen. Auch das, durch die Form der Räder bedingte, hohe Maß an Rauschen kann der selbstlernende Regler zufriedenstellend beherrschen.

Tabelle 5.7: Physikalische Modellparameter des Flugzeugmodelles für die Simulation.

Modellparameter	siehe Literatur [109]	
Eingriffsgröße	$\delta \in [-1.4, 1.4]$	[rad]
Regelintervall	$\Delta_t = 0.05$	[s]

5.2 Autopilot für Verkehrsflugzeug

In diesem Beispiel wird die Verwendung eines selbstlernenden Reglers als Komponente eines Autopilots für die Regelung des Nickwinkels eines Boeing-Verkehrsflugzeuges betrachtet. Diese Regelungsaufgabe hat für die Erstellung selbstlernender Regler die interessante Eigenschaft, dass die betrachtete Systemdynamik des Flugzeuges eine Kombination aus einer schnell veränderlichen Prozessvariablen und einer sehr trägen Prozessvariablen beinhaltet, die sich gegenseitig beeinflussen. Eine komplette Regeltrajektorie dauert bis zu 1000 Zeitschritte und besteht aus einem schnellen Erreichen eines Zielbereiches und einem darauf folgenden langen Einschwingvorgang, der einer genauen Nachführung der Eingriffsgröße bedarf.

Das dynamische Systemverhalten eines Flugzeuges wird im Allgemeinen durch eine Menge von sechs nichtlinearen und komplexen Differenzialgleichungen beschrieben. Unter bestimmten Annahmen können bei diesen die longitudinalen und lateralen Komponenten entkoppelt und linearisiert werden. Die Regelung des Nick-Winkels ist ein longitudinales Problem. Es wird angenommen, dass das Flugzeug sich in einem Gleichgewichtszustand bei konstanter Höhe und Geschwindigkeit befindet. Zusätzlich soll angenommen werden, dass die Änderung des Nick-Winkels die Geschwindigkeit des Flugzeuges zu keinem Zeitpunkt verändere. Diese Annahme ist nicht sonderlich realistisch, vereinfacht die Systemgleichungen zur Simulation jedoch ein wenig. Unter diesen Annahmen kann die Anzahl der Differenzialgleichungen zur Beschreibung des Systemverhaltens für die Simulation auf drei reduziert werden. In Formel 5.1 sind diese drei Differenzialgleichungen angegeben. Zur Vereinfachung der Darstellung wurde auf die genaue Herleitung dieser Differenzialgleichungen unter Verwendung der physikalischen Parameter an dieser Stelle verzichtet. Für eine genauere Beschreibung des Systems sei an [109] verwiesen.

$$\begin{aligned}
 \dot{\alpha} &= -0.313\alpha + 56.7q + 0.232\delta \\
 \dot{q} &= -0.0139\alpha - 0.426q + 0.0203\delta \\
 \dot{\Theta} &= 56.7q
 \end{aligned}
 \tag{5.1}$$

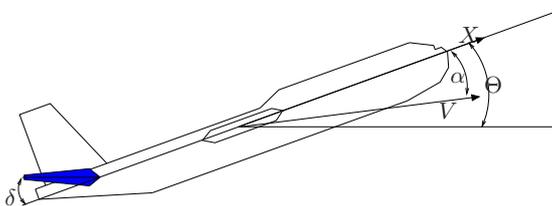


Abbildung 5.14: Das Modell eines Flugzeuges. In Abbildung 5.14 sind die Systemvariablen an einer schematischen Darstellung des Flugzeuges zu sehen. Hierbei bezeichnet Θ den Nickwinkel des Flugzeuges und q dessen Winkelgeschwindigkeit. Der Winkel α bezeichnet die Abweichung des aktuellen Bewegungsvektors des Flugzeuges von seiner Ausrichtung (Angle of Attack). Die Eingriffsgröße des Systems δ ist die Winkelstellung des Höhenruders, die vom Regler direkt in einem Bereich von -1.4 bis 1.4 rad gestellt werden kann. Die eingesetzten Werte entsprechen den technischen Werten einer Boeing Verkehrsmaschine [109]. Der Nickwinkel und seine Winkelgeschwindigkeit sind dabei die sich schnell verändernden Prozessvariablen. Aufgrund dieser

und der Veränderung des Winkels des Höhenruders verändert sich der Angriffswinkel der Flugzeuggeschwindigkeit. Diese Veränderung des Angriffswinkels ist ein sehr langsamer Prozess, der über mehrere hundert Zeitschritte andauert.

5.2.1 Entwurf eines Vergleichsreglers mittels LQR-Methode

Mit Hilfe des LQR-Verfahrens wird ein Vergleichsregler entworfen. Die lineare Systemdynamik lässt sich in Form eines LTI-Systems schreiben (Formel 5.2).

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\Theta} \end{bmatrix} = \begin{bmatrix} -0.313 & 56.7 & 0 \\ -0.0139 & -0.426 & 0 \\ 0 & 56.7 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \Theta \end{bmatrix} + \begin{bmatrix} 0.232 \\ 0.0203 \\ 0 \end{bmatrix} [\delta] \quad (5.2)$$

$$y = [0 \ 0 \ 1] \begin{bmatrix} \alpha \\ q \\ \Theta \end{bmatrix} + [0] [\delta] \quad (5.3)$$

Beim LQR-Reglerentwurf wird eine Regelmatrix K bestimmt, die für das System $\dot{x} = Ax + Bu$ und das Regelgesetz $u = -Kx$ die Kostenfunktion

$$J(x) = \int_0^{\infty} (x^T Q x + u^T R u + 2x^T N u) dt \quad (5.4)$$

minimiert. Über die Matrizen Q , R und N wird eine Balance der Regelgüte betreffend der Schnelligkeit und des Regelaufwandes hergestellt. Für den Entwurf des Vergleichsreglers werden die Werte derart gewählt, dass der Regeleingriff δ in einem realistischen Bereich bleibt, ohne den zulässigen Bereich zu verlassen. Vorgegeben werden müssen weiter, die Güte-Matrix R und die Zustands-Kosten-Matrix Q (mit Gewichtungsfaktor p) (siehe Gleichung 5.5).

$$R = [1] \quad (5.5)$$

$$Q = C^T x C p = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & p \end{bmatrix} \quad (5.6)$$

$$p = 50 \quad (5.7)$$

$$N = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (5.8)$$

Nach Durchführung der LQR-Methode (Matlab) ergibt sich die Regelmatrix K zu: $K = [-0.6435 \ 169.6950 \ 7.0711]$. Da beim Zustandsregler nicht die eigentliche Zielgröße, sondern der Zustand, multipliziert mit der Regelmatrix, mit der Führungsgröße verglichen wird, zeigt ein solcher Regler für Probleme mit beliebiger Führungsgröße ein Verhalten mit stationärem Restfehler. Um diesen zu eliminieren, wird ein Feed-Forward Faktor $\bar{\mathfrak{N}}$ hinzugefügt, der die Differenz zur Führungsgröße ausgleicht. Für das hier verwendete System ergibt sich dieser Faktor zu $\bar{\mathfrak{N}} = 7.0711$ (siehe [109]).

Die generelle Funktionsweise eines Zustandsreglers mit Führungsgröße ist in Abbildung 5.15 dargestellt. Nach Bestimmung der Regelmatrix K und des Feed-Forward Faktors $\bar{\mathfrak{N}}$ ergibt sich das Regelgesetz des Reglers zu $\delta = \bar{\mathfrak{N}}R - Kx$. In Abbildung 5.16 ist eine Regeltrajektorie des LQR-Reglers mit dem Sollwert 0.2 zu sehen. Rot dargestellt ist der Nickwinkel, der dem

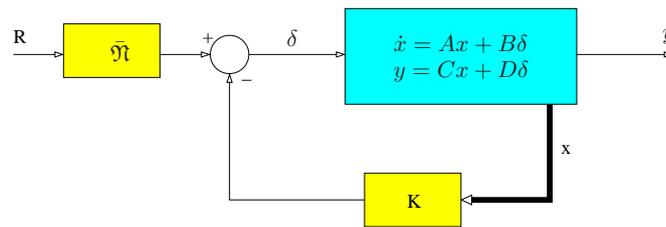


Abbildung 5.15: Die schematische Darstellung der Funktionsweise eines Zustandsreglers mit Regelmatrix K und Führungsgröße R .

Zielwert 0.2 nach ca. 2.5 Sekunden konstant folgt. Die blaue Kurve zeigt die Regler-Ausgabe bzw. den Regeleingriff in Form des Höhenruderausschlages. Nach Erreichen des Zielwertes eilt der Winkel α , der Angriffswinkel der lateralen Flugzeugbewegung im Raum, noch mit starker Verzögerung nach. Dies ist einer der Haupteigenschaften des Systems, die kontinuierliche Aktionen zwingend erforderlich machen, um die Auslenkung des Höhenruders der nacheilenden System-Größe anzupassen. Der LQR-Regler erzeugt eine Reglertrajektorie, die ein deutliches

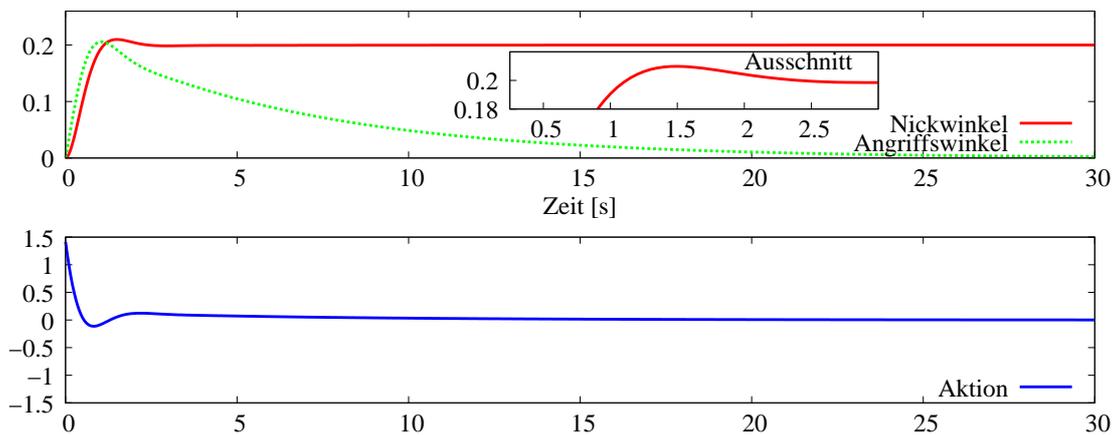


Abbildung 5.16: Die Regeltrajektorie des LQR-Reglers bei einem Sprung des Sollwertes von 0 auf 0.2 rad. Der LQR-Regler erzeugt eine Trajektorie mit deutlichem Überschwingen des Nickwinkels. Nach ca. 2.5 Sekunden wird der Nickwinkel erreicht und gehalten. Nach mehr als 30 Sekunden befindet sich die Systemdynamik in einem Equilibriumspunkt da der Angriffswinkel auf Null zurück geht.

Überschwingen des Nickwinkels über die Führungsgröße aufweist. Bei Anwendung des LQR-Reglers auf ein Profil der Führungsgröße in Abbildung 5.17 wird ebenfalls deutlich, dass der LQR-Regler bei großen Sprüngen der Führungsgröße überschwingt und auf kleinere Sprünge im Verlauf der Führungsgröße nicht mit hohen Ausschlägen reagieren kann. Auffallend ist ebenfalls die Regelabweichung bei der kontinuierlichen Veränderung der Führungsgröße zwischen 7 und 20 Sekunden auf dem gegebenen Profil.

5.2.2 Selbstlernender Regler zur Nickwinkelregelung

Die Zustandsinformation des selbstlernenden Reglers muss den Angriffswinkel α und die Nickwinkelgeschwindigkeit q enthalten. Zudem muss die Regelabweichung in der Zustandsinformation enthalten sein. Betrachtet man die Systemgleichungen in 5.1 genauer, so stellt sich heraus, dass bei dem betrachteten Problem mit der Regelabweichung bereits alle Informatio-

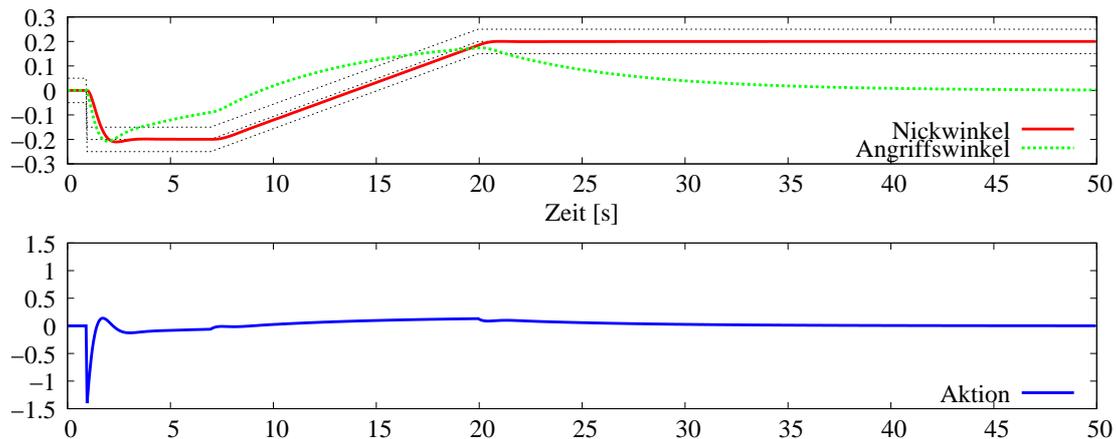


Abbildung 5.17: Die Trajektorie einer Folgeregelung der Nickwinkelregelung des Autopilots mit dem LQR-Regler. Die Führungsgröße und der Toleranzbereich der Regelung ist mit gestrichelten Linien eingezeichnet.

Tabelle 5.8: Die für das Lernen verwendeten Parameter für die Nickwinkelregelung des Flugzeuges.

Zustand x	3 dimensional	α	Angriffswinkel Flugzeuggeschwindigkeit
		q	Nickwinkelgeschwindigkeit
		$\tilde{\theta} = \theta_d - \theta$	Nickwinkel Regelabweichung
Kostenfunktion	\tanh^2	x^d	$(-, -, 0)$
		μ	$(0, 0, 0.06)$
		C	0.01
Aktion u	1 dimensional NFQCA NFQ	δ	Höhenruderausschlag
		$\mathcal{U} =$	$[-1.4, 1.4]$
		$U =$	$\{ \pm 1.4, \pm 0.5, \pm 0.2, \pm 0.1, 0 \}$
Q-Funktion	neuronal	Topologie	4-15-1 (600 Epochen RProp)
Strategie-Funktion	neuronal	Topologie	3-5-1 (400 Epochen RProp)
Exploration	normalverteilt	ϵ	0.30
		Δ	$\sim N(0, 1.0)$
Interaktionsepisode		Länge	300 Schritte
Testepisode		Länge	1000 Schritte (\bar{e}_T)
		Länge	300 Schritte (\bar{N}, \bar{e}_∞)

nen zur Beschreibung des Systems enthalten sind. Der Nickwinkel θ muss nicht explizit in die Zustandsmodellierung aufgenommen werden. Es ergibt sich somit ein dreidimensionaler Zustandsvektor. Die Aktion ist eindimensional und beschreibt direkt den Ausschlag des Höhenruders.

Alle Interaktionsepisoden werden aus dem Zustand $(0,0,0)$ gestartet. Dies entspricht einem Flugzeug, das sich in einer stabilen Fluglage befindet, bei dem der Angriffswinkel der Flugzeuggeschwindigkeit Null ist. Somit bewegt sich das Flugzeug in Richtung seiner Hauptachse. Es wird sowohl ein Regler mit NFQ mit diskreten Aktionen, als auch ein Regler mit NFQCA mit kontinuierlichen Aktionen gelernt. In jeder Interaktionsepisode wird der Regler mit einem zufälligen Zielwert $\theta_d \in [-0.5, 0.5]$ gestartet und für 300 Zeitschritte mit einer Exploration von 30% , bei NFQCA unter einer Normalverteilung von $N(0,1.0)$, auf das System angewendet. Es werden maximal 300 Interaktionsepisoden durchgeführt. Die Versuchsparameter sind in Tabelle 5.8 zusammengefasst. Für beide Regler wird nach jeder Iteration des Lernverfahrens der entstandene Regler auf das aus Abbildung 5.17 bekannte Profil von Führungsgrößen angewendet. Zum Vergleich der Regler wird die mittlere absolute Abweichung \bar{e}_T über die

Folgeregelung auf diesem Profil ermittelt. Für die Bewertung der Regelabweichung nach dem Erreichen des Toleranzbereiches wird jeder Regler zusätzlich auf 100 Testtrajektorien, 300 Schritte lang angewandt. Die Zielwerte werden zu Beginn der Testtrajektorien zufällig aus dem Bereich $\theta_d \in [-0.5, 0.5]$ gezogen. Auf den letzten 150 Zeitschritten wird die Kennzahl \bar{e}_∞ bestimmt und über alle Trajektorien gemittelt. Ebenso wird auf diesen Testtrajektorien das Gütekriterium \bar{N} bestimmt.

Der LQR-Vergleichsregler erzeugt auf dem gegebenen Profil der Führungsgröße einen Trajektorienfehler von 0.00636. Nach nur 26 Iterationen mit diskreten Aktionen mit NFQ bzw. 14 Iterationen mit kontinuierlichen Aktionen und NFQCA sind die selbstlernenden Regler unter diesem Niveau und damit besser als der Vergleichsregler. In Abbildung 5.18 ist eine Regeltrajektorie mit einem konstanten Zielwert von 0.2 des mit NFQCA gelernten Reglers zu sehen. Im Vergleich mit dem LQR-Regler in Abbildung 5.16 kann der NFQCA-Regler das Überschwingen reduzieren und den Zielwert mit maximalen Eingriffsgrößen schneller erreichen. Schon nach 1.2 Sekunden folgt der Regler der Führungsgröße und ist somit mehr als doppelt so schnell als der LQR-Regler. Während der anschließende Phase des Ausschwingens des Angriffswinkels wird die Eingriffsgröße geeignet nachgeführt, bis der Angriffswinkel auf Null ist. In dieser Phase arbeitet der gelernte Regler annähernd gleich wie der LQR-Regler. Eine etwas andere Darstellung dieser Regeltrajektorie ist in Abbildung 5.21 über die ersten 210 Zeitschritte dargestellt. Das beste Regelverhalten zeigen die selbstlernenden Regler mit NFQ

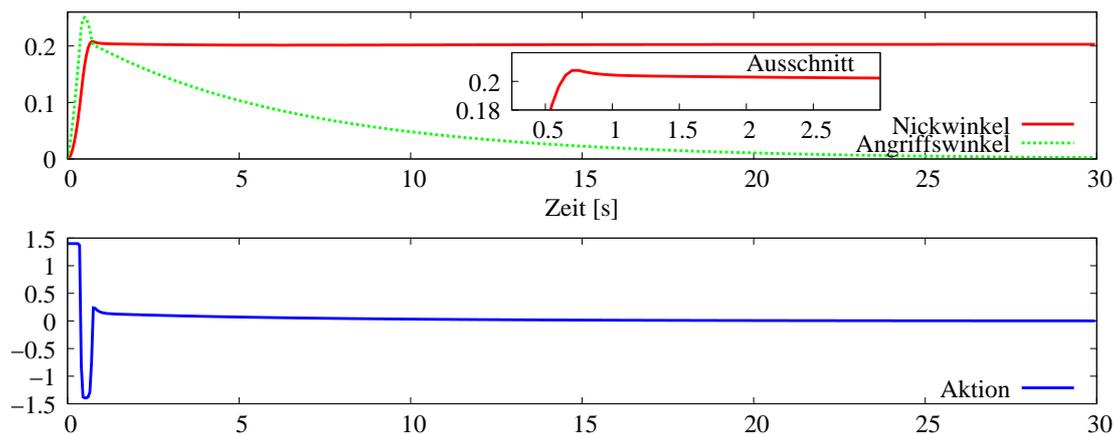


Abbildung 5.18: Die Regeltrajektorie des NFQCA Reglers bei einem Sprung der Führungsgröße von 0 auf 0.2 rad.

in diesen Versuchen schon nach 200 Iterationen des Lernverfahrens. Der beste Regler mit NFQ erzeugt hierbei einen Trajektorienfehler von $\bar{e}_T = 0.0011$. Mit NFQCA wird der beste Regler nach 215 Iterationen des Lernverfahrens gefunden. Dieser erzeugt einen Trajektorienfehler von $\bar{e}_T = 0.00052$. In Tabelle 5.9 sind die Auswertungen der Versuche zusammengefasst.

Die Regelstrategie der selbstlernenden Regler auf dem Profil der Führungsgröße sind in Abbildung 5.19 für den mit NFQCA und in Abbildung 5.20 für den Regler mit NFQ zu sehen. Der mit NFQCA eingelernte Regler in Abbildung 5.19 zeigt eine kontinuierliche Veränderung der Eingriffsgröße und nutzt die ihm zur Verfügung stehenden kontinuierlichen Aktionen aus. Dies ist besonders für die Nachführung der Eingriffsgröße während des Ausschwingens des Angriffswinkels von Vorteil für den Regler. Die während dieser Phase angewendeten Aktionen sind in der Abbildung zur besseren Darstellung als vergrößerter Ausschnitt dargestellt. Im Gegensatz hierzu ist in Abbildung 5.20 die Regeltrajektorie des NFQ Reglers mit diskreten Aktio-

Tabelle 5.9: Versuchsauswertung der Regler zur Nickwinkelregelung eines Flugzeuges.

Regler	\bar{N}	\bar{e}_∞	\bar{e}_T
LQR	27.33	0.00033	0.00636
NFQ	14.82	0.00032	0.00110
NFQCA	14.51	0.00030	0.00052

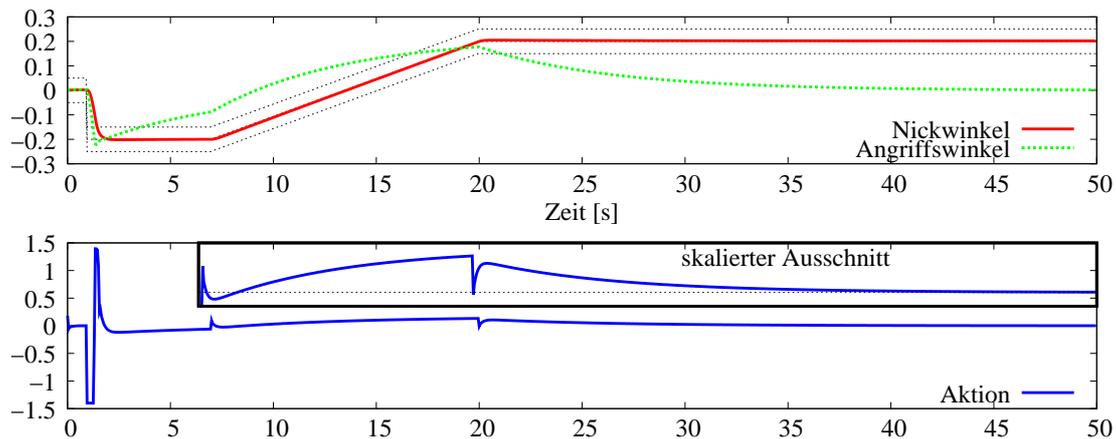


Abbildung 5.19: Die Trajektorie einer Folgeregelung der Nickwinkelregelung des Autopilots mittels NFQCA. Zusätzlich ist ein Ausschnitt der vom Regler ausgegebenen kontinuierlichen Aktion dargestellt, in dem die exakte Nachführung der Aktion zu sehen ist. Die Führungsgröße und der Toleranzbereich der Regelung ist mit gestrichelten Linien eingezeichnet.

nen dargestellt. Dieser kann das Erreichen der Führungsgröße ebenfalls sehr schnell und präzise durchführen. Die exakte Nachführung der Eingriffsgröße muss anschließend jedoch durch die Kombination der ihm zur Verfügung stehenden diskreten Aktionen durchgeführt werden. Dies erzeugt über die gesamten 1000 Schritte der Trajektorie einen schlechteren Trajektorienfehler, als mit kontinuierlichen Aktionen erreicht werden kann.

5.2.3 Diskussion

Der Lernversuch zur Nickwinkelregelung als Komponente eines Autopilots für ein Verkehrsflugzeug ist ein Beispiel für ein System, das eine sehr träge Systemdynamik beinhaltet. Dies bedingt für die selbstlernenden Regler eine sehr hohe Anzahl von Zeitschritten bis das System nach einer Störung in einen stabilen Bereich überführt werden kann. Trotz dieses langen Optimierungs-Horizonts sind die selbstlernenden Regler schon nach wenigen Iterationen des explorativen Lernprozesses in der Lage, die Regelgüte des klassischen LQR-Reglers zu übertreffen. Die selbstlernenden Regler besitzen den Vorteil, dass sie durch den ihnen zugrunde liegenden Optimierungsprozess ein nichtlineares Regelgesetz lernen können. Dieses nichtlineare Regelgesetz wird in diesem Versuch vor allem durch die Beschränkung der Eingriffsgröße notwendig. Die selbstlernenden Regler können diese Randbedingung und ihre Auswirkung auf ein optimales Regelgesetz direkt in ihrem Lernprozess einbeziehen. Beim Entwurf des LQR-Reglers kann diese Randbedingung nur implizit beachtet werden, indem das entworfene Regelgesetz auf die Verletzung der Randbedingung getestet wird.

Ein selbstlernender Regler mit 9 diskreten Aktionen ist unter der Benutzung des NFQ-

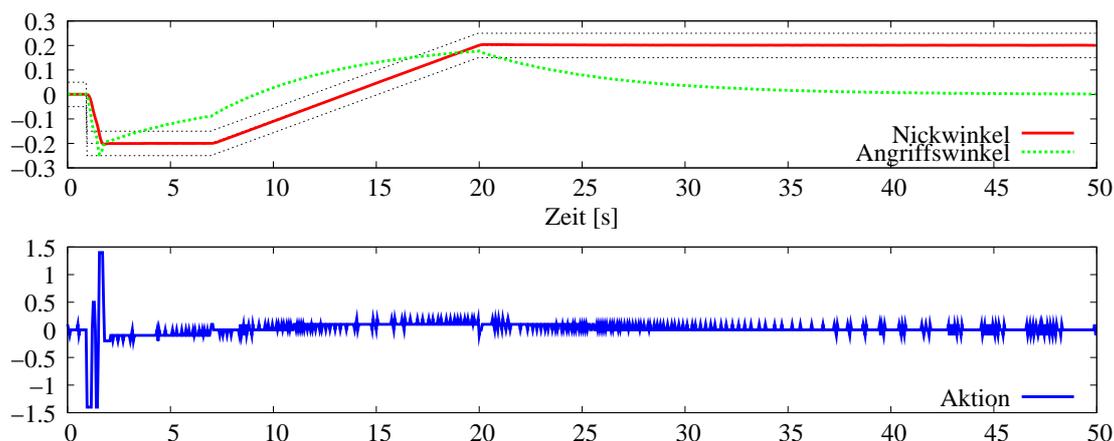


Abbildung 5.20: Die Trajektorie einer Folgeregelung der Nickwinkelregelung des Autopiloten mittels NFQ und diskreten Aktionen. Die Führungsgröße und der Toleranzbereich der Regelung ist mit gestrichelten Linien eingezeichnet.

Verfahrens in der Lage, eine sehr gute Regelgüte zu erzeugen, die dem LQR-Regler deutlich überlegen ist. Durch die Verwendung kontinuierlicher Aktionen mit NFQCA kann diese Regelgüte noch einmal gesteigert werden. Insbesondere erzeugt die Verwendung kontinuierlicher Aktionen während der trägen Veränderung des Anstellwinkels eine sehr präzise Nachführung des Höhenruders mit adäquate Eingriffsgrößen. Im Unterschied dazu muss die benötigte Eingriffsgröße bei der Benutzung diskreter Aktionen durch eine zeitliche Abfolge der vorhandenen Aktionen angenähert werden.

Der Nachteil eines aus diskreten Aktionen zusammengesetzten Regelgesetzes, wie es in Abbildung 5.20 zu sehen ist, lässt sich an diesem Versuch anschaulich diskutieren. Die Veränderung der Stellung des Höhenruders wurde im Versuch als instantaner Eingriff im System betrachtet. Somit sind die Eingriffe des Reglers mit diskreten Aktionen zur Erreichung der Regelgüte zulässig. An einem realen System können diese jedoch zu gravierenden Problemen führen. Die ständige Änderung des Höhenruderausschlages führt zu unerwünschten Vibrationen im gesamten Flugzeug und einem sehr hohen Energieverbrauch. Zusätzlich liegt der Veränderung des Höhenruderausschlages in realen Systemen ein Regler zu Grunde, der die Stellung des Höhenruders als Führungsgröße übernimmt und für dessen Umsetzung sorgt. Typischerweise arbeiten solche unterliegenden Regler mit schnell wechselnden Führungsgrößen, wie sie von den diskreten Aktionen mit NFQ erzeugt werden, nur sehr schlecht. In einer hierarchischen Anordnung von Reglern führt die Verwendung diskreter Aktionen somit eventuell zu einer Verschlechterung der gesamten Regelgüte. Die kontinuierlichen Aktionen zeigen hier einen deutlichen Vorteil. Das von ihnen erzeugte Regelgesetz ist ähnlich glatt wie das des LQR-Reglers, kann jedoch auf bestimmte Zustände mit hohen Ausschlägen des Höhenruders reagieren, um schnell die Führungsgröße zu erreichen. In einer hierarchischen Anordnung von Reglern fügt sich ein selbstlernender Regler mit kontinuierlichen Aktionen somit besser in die Gesamtstruktur ein, womit eine bessere Regelgüte des gesamten Systems zu erwarten ist.

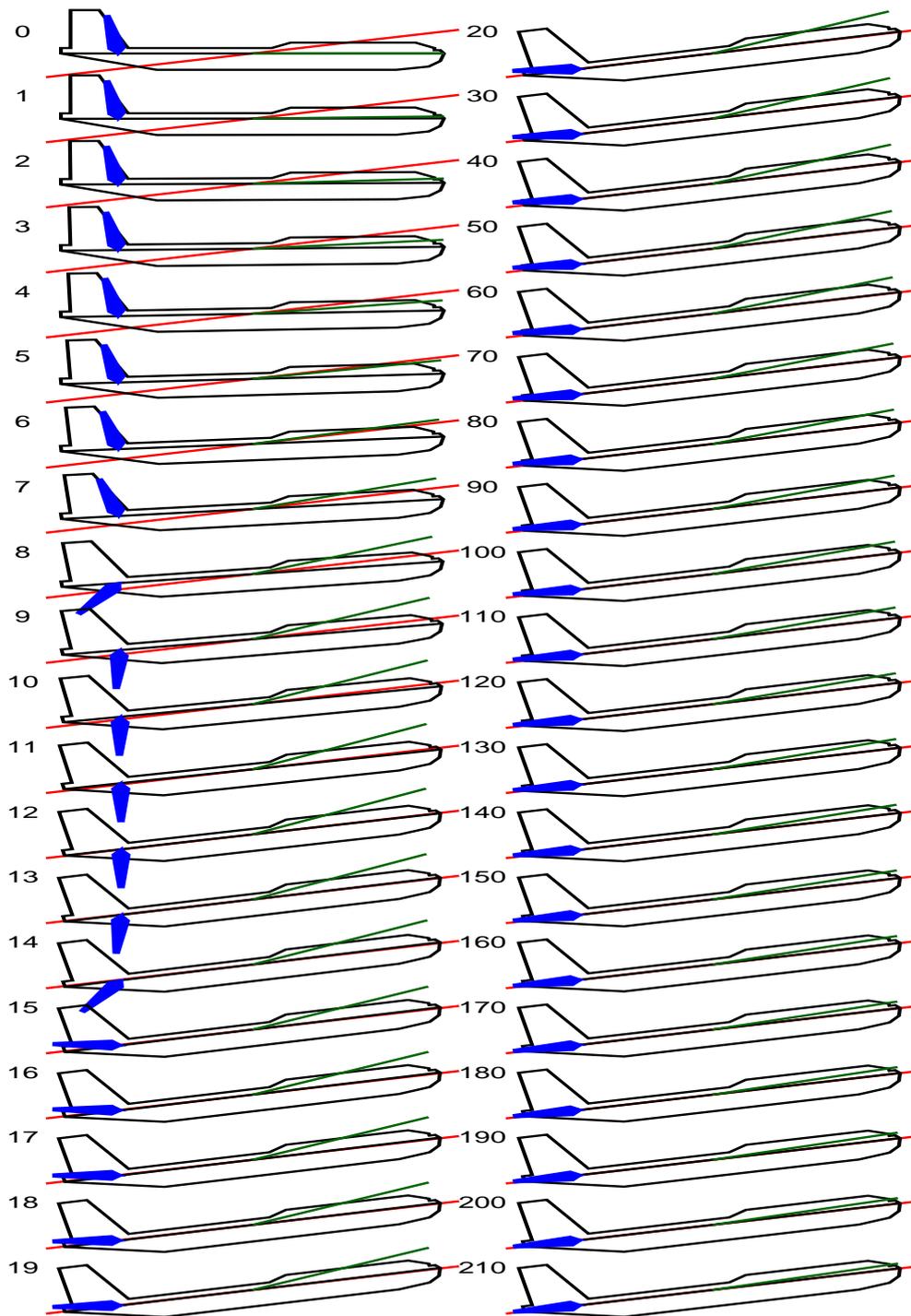


Abbildung 5.21: Die Regeltrajektorie eines mit NFQCA gelernten Reglers auf einen Sprung der Führungsgröße von 0 auf 0.2 rad. Das Regelziel ist das schnelle Erreichen und die genaue Einhaltung des vorgegebenen Nickwinkels des Flugzeuges (rote Linie) durch geeignete Ansteuerung des Höhenruders. Der Angriffswinkel der Geschwindigkeit des Flugzeuges eilt dabei sehr langsam nach, während das Höhenruder genau nachgeführt werden muss, um den Nickwinkel einzuhalten. Der gesamte Regelvorgang dauert über 30 Sekunden bzw. über 600 Regelzyklen, von denen eine Auswahl aus den ersten 210 Zyklen dargestellt ist.

5.3 Aktive Schwingungs-Dämpfung

Bei der aktiven Schwingungs-Dämpfung werden die an einem Prozess auftretenden Schwingungen oder Vibrationen durch das Hinzufügen einer aktiven Komponente eliminiert. Die Regelung dieser aktiven Komponente bietet interessante Fragestellungen für den Entwurf selbstlernender Regler. Die Anwendungsgebiete der aktiven Schwingungs-Dämpfung liegen vor allem im Maschinenbau und der Automobiltechnik. Durch den Einsatz von aktiven Komponenten zur Schwingungsreduktion kann auf konstruktive Elemente verzichtet werden, was eine Reduzierung von zusätzlichem Gewicht und Kosten bedeutet. Speziell in der Automobilindustrie ist die aktive Schwingungsdämpfung aus diesem Grund eine sehr gefragte Technologie. Die Zielsetzung der Schwingungsreduktion durch aktive Komponenten reichen von der Erhöhung des Komforts des Benutzers bis hin zur Einhaltung von sicherheitsrelevanten Zuständen von Maschinen und Fahrzeugen.

Im Rahmen dieser Arbeit wird die aktive Dämpfung einer Busfederung als ein typisches Problem der Automobilindustrie untersucht. Das verwendete Modell ist angelehnt an [109] und wird als gekoppeltes Feder-Masse-System modelliert. Als Vereinfachung wird für diese Modellierung das Problem der Regelung einer aktiven Federung für ein Bus-Fahrwerk auf die unabhängige Modellierung eines Rades beschränkt. In anderen Worten, 1/4 eines Fahrzeuges wird als unabhängiges System betrachtet. Diese Vereinfachung ist im Entwurf solcher Systeme im industriellen Umfeld eine gängige Vorgehensweise, an der die wichtigsten Eigenschaften der Federung untersucht werden können.

Der schematische Aufbau des Modells ist in Abbildung 5.22 dargestellt. Es besteht aus einer Busmasse m_1 , die über ein Feder-Dämpfer Element mit der Aufhängung des Rades verbunden ist. Die Aufhängung des Rades, inklusive des Rades selbst, besitzt die Masse m_2 . Das Verhalten des mit Luft gefüllten Rades wird als weiteres Feder-Dämpfer Element modelliert, das die Aufhängung des Rades mit dem Untergrund bzw. der Straße verbindet. Das Verhalten der Feder-Dämpfer Elemente wird durch ihre Federkonstante k_1 bzw. k_2 und ihre Dämpfungskonstante b_1 bzw. b_2 definiert. Die von der schematischen Darstellung der Modellierung abgeleiteten Prozessvariablen sind die Auslenkung der Busmasse x_1 und die Auslenkung der Masse des Federungsapparates x_2 aus ihrer Ruhelage, sowie deren Geschwindigkeit. Als Eingriffsgröße in das System kann zwischen den Massen m_1 und m_2 eine Kraft u in das System eingebracht werden, welche die aktive Komponente zur Schwingungs-Dämpfung modelliert. Es wird angenommen, dass sich das Rad zu jedem Zeitpunkt mit dem Untergrund in Kontakt befindet und nicht abheben kann. Der Verlauf des Untergrundes wird als Auslenkung w der Straße zu ihrer Anfangslage modelliert und ist eine weitere externe Eingabe an das dynamische System, die vom Regler nicht beeinflusst werden kann.

Für die Beschreibung der Systemgleichungen wird an dieser Stelle von der aus der Motivation hergeleiteten Definition der Prozessvariablen abgewichen. In einem realen System lassen sich die Auslenkungen der Massen und die Geschwindigkeit, mit der sich die Aufhängung des Rades bewegt, nicht messen. Um eine realistische Problemstellung zu gewährleisten werden für die Systemgleichungen deshalb Prozessvariablen gewählt, die sich in einem realen System erfassen lassen. Die Auslenkung des Buskörpers und ihre Geschwindigkeit sollen vorerst für die Berechnungen der Systemgleichungen erhalten bleiben. Die Prozessvariablen x_2 und \dot{x}_2 werden durch zwei neue Variablen y_1 und y_2 ersetzt. Die Variable y_1 sei die Differenz aus den beiden Auslenkungen des Buskörpers und der Aufhängung:

$$y_1 = x_1 - x_2 \quad (5.9)$$

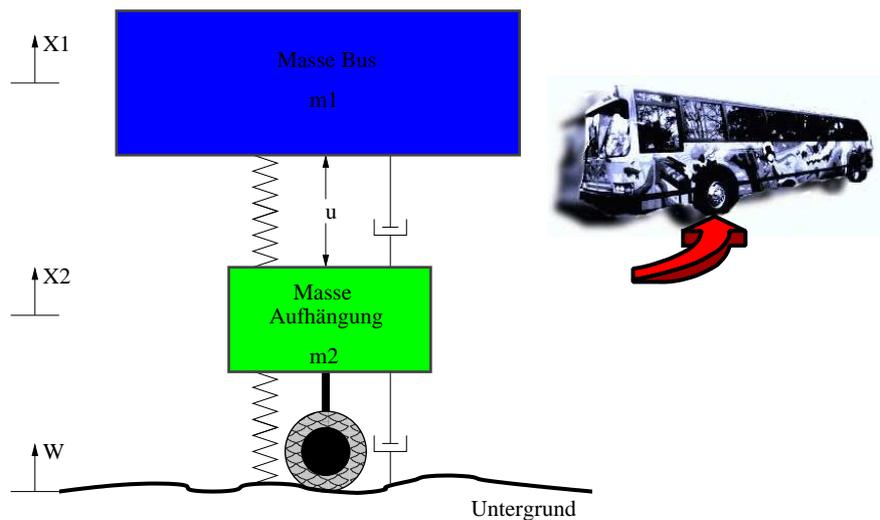


Abbildung 5.22: Schematische Darstellung der aktiven Busfederung.

und soll die eigentliche Ausgabe des dynamischen Systems an den Regler darstellen. Diese lässt sich direkt als die Eintauchtiefe des passiven Federapparates deuten und ist somit auch an einem realen System messbar. Die Variable y_2 ist eine Hilfsvariable für die Systemgleichungen und hat hier keine direkte physikalische Bedeutung. Für die genaue mathematische Herleitung der Systemgleichungen und notwendigen Umformungen sei auf [109] verwiesen.

Die Systemgleichungen des dynamischen Systems der 1/4 Busfederung sind in Formel 5.10 angegeben. Für die durchgeführten Versuche wird das System mittels numerischer Integration mit Runge-Kutta-Verfahren und den in Tabelle 5.10 angegebenen physikalischen Parametern simuliert.

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_1 \\ \dot{y}_1 \\ \dot{y}_2 \end{pmatrix} = A \begin{pmatrix} x_1 \\ \dot{x}_1 \\ y_1 \\ y_2 \end{pmatrix} + B \begin{pmatrix} u \\ w \end{pmatrix} \quad (5.10)$$

mit:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{(b_1 b_2)}{(m_1 m_2)} & 0 & \left(\frac{b_1}{m_1} \left(\frac{b_1}{m_1} + \frac{b_1}{m_2} + \frac{b_2}{m_2}\right) - \frac{k_1}{m_1}\right) & -\frac{b_1}{m_1} \\ \frac{b_2}{m_2} & 0 & -\left(\frac{b_1}{m_1} + \frac{b_1}{m_2} + \frac{b_2}{m_2}\right) & 1 \\ \frac{k_2}{m_2} & 0 & -\left(\frac{k_1}{m_1} + \frac{k_1}{m_2} + \frac{k_2}{m_2}\right) & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} 0 & 0 \\ \frac{1}{m_1} & \frac{b_1 b_2}{m_1 m_2} \\ 0 & -\frac{b_2}{m_2} \\ \left(\frac{1}{m_1} + \frac{1}{m_2}\right) & \frac{-k_2}{m_2} \end{pmatrix}$$

Eine gute Bus-Federung sollte eine gute Straßenlage erzeugen und gleichzeitig den Insassen des Busses ausreichend Komfort gewähren, wenn das Rad über Unebenheiten fährt. In Abbildung 5.23 ist die Systemantwort des Modells auf einen Einheitssprung der Straßenlage mit Höhe 10cm über eine Zeitdauer von 30 Sekunden gezeigt. Dargestellt ist die Auslenkung der

Tabelle 5.10: Physikalische Modellparameter der Busfederung für die Simulation.

Masse Bus	m_1	=	2500	[kg]
Masse Aufhängung	m_2	=	320	[kg]
Federkonstante Federung	k_1	=	80.000	[N/m]
Federkonstante Rad	k_2	=	500.000	[N/m]
Dämpfungskonstante Federung	b_1	=	350	[Ns/m]
Dämpfungskonstante Rad	b_2	=	15.020	[Ns/m]
Eingriffsgröße	u	\in	[-8000, 8000]	[N]
Störgröße	w	$\Delta w \in$	[-0.1, 0.1]	[m]
Regelintervall	Δ_t	=	0.010	[s]

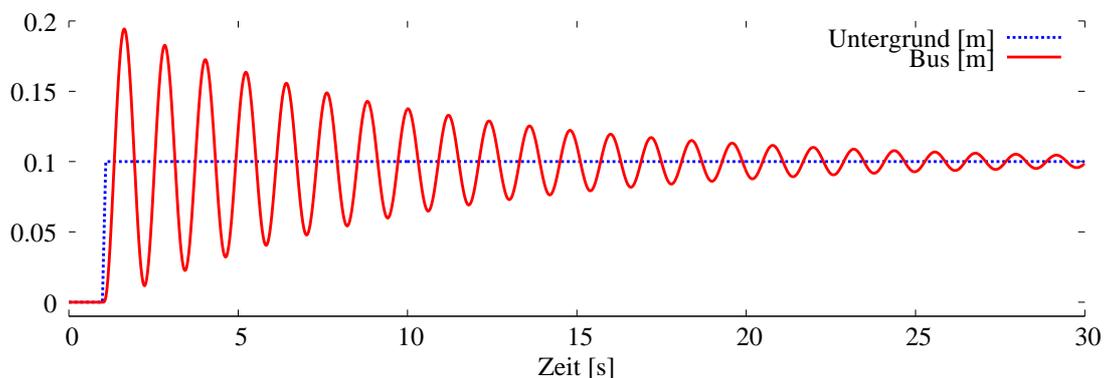


Abbildung 5.23: Die Trajektorie der Systemantwort des offenen dynamischen Systems der Busfederung auf eine Störung von 10cm Auslenkung. Die blaue Kurve ist die Auslenkung der Straße, die als Störung auf das System wirkt und die Schwingung verursacht. Als rote Kurve ist die Auslenkung des Buskörpers aufgetragen, der seine neue Ruhelage nach einer sehr langen Ausschwingzeit relativ zur Auslenkung der Straße findet.

Straße als blaue Kurve, die in der Grafik ebenfalls die neue Ruhelage des Buskörpers angibt, sowie die Auslenkung des Buskörpers als rote Kurve. Ohne aktive Dämpfung ist die Federung nicht in der Lage, die Schwingung dieser Störung geeignet zu dämpfen. Das Systemverhalten des offenen dynamischen Systems ist somit gänzlich ungeeignet, ein komfortables und sicheres Fahrverhalten zu gewährleisten. Das Ziel der Regelung der aktiven Komponente ist es, die entstandenen Schwingungen schnellstmöglich aus dem System zu eliminieren.

Die Regelgröße für dieses System wird ebenfalls auf einer der messbaren Systemvariablen definiert. Dies ist die Eintauchtiefe des Federapparates y_1 . Zusätzlich wird angenommen, dass die Geschwindigkeit der Auslenkung des Buskörpers \dot{x}_1 messbar ist. Dies kann an einem realen System über die Messung der Beschleunigung des Buskörpers und eines geeigneten Integrationsprozesses über Filter realisiert werden. Die Regelung soll sicherstellen, dass y_1 keine großen Amplituden hat und Schwingungen schnell kompensiert werden. Die Wahl der Zustandsinformation ist bewusst so gewählt, dass sie nicht die volle Zustandsinformation des Systems repräsentiert. Die Systemdynamik des Prozesses wird von den in der Zustandsbeschreibung des Reglers enthaltenen Informationen dominiert. Der parasitäre Teil der Systemdynamik, der hauptsächlich die Dynamik des Rades betrifft und der nicht in der Zustandsinformation enthalten ist, stellt eine Unsicherheit dar, gegen die ein Regler für die aktive Schwingungsdämpfung robust sein muss. Aus der Sicht des Reglers liegt, basierend auf diesen Informationen, nun

keine lineare Systemdynamik mehr vor.

PID Vergleichsregler

Für einen einfachen Vergleichsregler wird ein PID-Regler, basierend auf der Systemausgabe $y_1 = x_1 - x_2$, gewählt. Die Werte für die PID-Regelung und die Struktur des Reglers sind der Literatur entnommen [109]. Die PID-Regelung ist nicht das bevorzugte regelungstechnische Konzept, um Probleme dieses Typs zu lösen. Der angegebene Regler dient somit hauptsächlich als Referenz für eine zu erreichende Reglergüte. Die verwendeten Parameter sind: $k_p = 832100$, $k_i = 624075$ und $k_d = 208025$. Im Unterschied zu den selbstlernenden Reglern wird der PID-Regler mit einer 10-fach höheren Regelfrequenz bzw. einer Zykluszeit von 1ms betrieben. Die Ausgabe des PID-Reglers wird auf die zulässigen Werte der Regelung

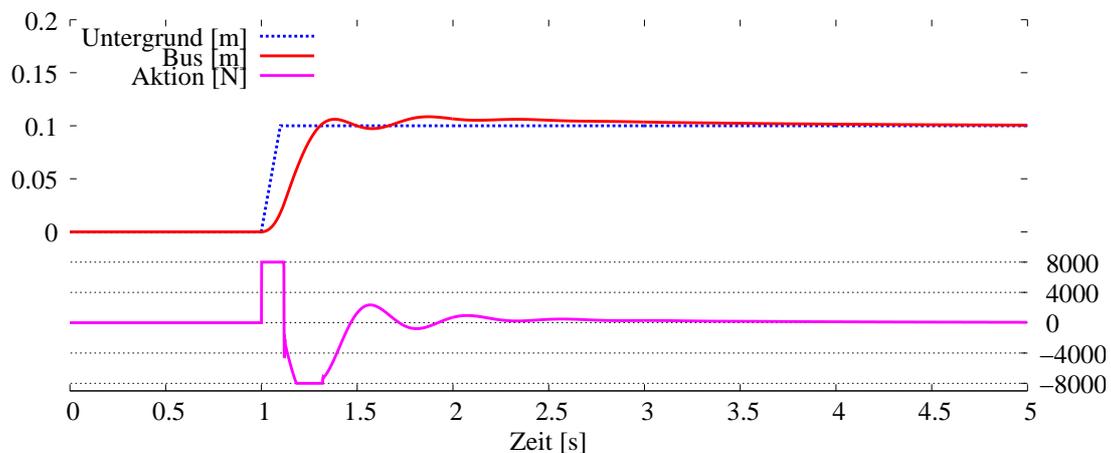


Abbildung 5.24: Systemantwort des PID Reglers auf eine einzelne Störung von 10cm Auslenkung.

von $\pm 8000\text{N}$ begrenzt. In Abbildung 5.24 ist eine Reglertrajektorie des PID-Reglers auf einen Sprung des Untergrundes von 10cm zu sehen. Im Vergleich zu der Systemantwort des offenen dynamische Systems auf diese Störung ist eine deutliche Dämpfung zu erkennen. Der Regler kann jedoch ein Nachschwingen des Buskörpers nicht vollständig verhindern.

5.3.1 Lernender Regler zur Schwingungsdämpfung

Am Beispiel des NFQCA werden der Entwurf und die Leistungsfähigkeit der selbstlernenden Regler für die aktive Schwingungsdämpfung am Problem der Busfederung untersucht. Die Zustandsinformation des Reglers basiert auf der Eintauchtiefe des passiven Dämpfers y_1 und der Geschwindigkeit der Auslenkung des Buskörpers \dot{x}_1 . Im Unterschied zu den anderen betrachteten Problemen existiert bei dieser Regelungsaufgabe keine variierende Führungsgröße. Die Aufgabe ist eine Stabilisierungsaufgabe im Zustandsraum der zwei beobachtbaren Prozessvariablen mit dem Ziel, die Eintauchtiefe des Dämpfers bei auftretenden Störungen durch Unebenheiten der Straße auf Null zu halten. Da dem Regler nicht die gesamte Information des Prozesses zur Verfügung steht, treten aus seiner Sicht sowohl in der Lernphase, als auch in der Anwendungsphase zusätzliche Störungen auf, die durch die nicht modellierten, parasitären Anteile der Systemdynamik verursacht werden. Von besonderem Interesse bei diesem Versuch ist die Art der Anregung auf den Interaktionsepisoden des Reglers während der Lernphase.

Tabelle 5.11: Die für das Lernen verwendeten Parameter für die aktive Schwingungsdämpfung des Busses.

Zustand x	2 dimensional	\dot{x}_1	Auslenkungsgeschwindigkeit Buskörper
		y_1	Eintauchtiefe Dämpfer
Kostenfunktion	\tanh^2	x^d	$(-, 0)$
		μ	$(0, 0.01)$
		C	0.01
Aktion u	1 dimensional NFQCA	u	aktive Dämpfungskraft
		\mathcal{U}	$[-8000, 8000]$
Q-Funktion	neural	Topologie	3-10-1 (400 Epochen RProp)
Strategie-Funktion	neural	Topologie	2-10-1 (600 Epochen RProp)
Exploration		ϵ	0.10
Interaktionsepisode		Länge	500 Schritte
Testepisode		Länge	500 Schritte

Die Störung in Form des externen, nicht beeinflussbaren Straßenverlaufes kann vom Regler nicht von der Dynamik des eigentlich zu regelnden Systems unterschieden werden. Dies bedeutet, dass die Anregungen des Straßenverlaufes als Rauschen in die Datenbasis des Verfahrens eingehen. Ein Lernen an einer realen Straße mit sehr vielen Störungen ist somit schwierig und würde sehr viel Zeit in Anspruch nehmen. Die typische Herangehensweise beim Entwurf der Regler für die aktive Schwingungsdämpfung im Automobilbereich ist die Verwendung eines hydraulischen Teststandes. Auf diesem können bestimmte Profile von Störungen auf das Fahrzeug übertragen werden. Um die Regler einzustellen werden auf dem Teststand einfache Sprungprofile gefahren, um die Einschwingdauer und Dämpfungseigenschaften der Regler zu überprüfen. Die Interaktionsepisoden, die für die selbstlernenden Regler in diesem Versuch benutzt werden, orientieren sich an diesem Vorgehen. In jeder Interaktionsepisode wird der Regler auf ein einfaches Sprungprofil des Straßenverlaufes angewendet. In diesem Sprungprofil ist eine Straße nachempfunden, die zwei Störungen enthält. Nach einer Sekunde enthält sie einen Einheitssprung der Höhe 10cm nach oben, nach weiteren 2 Sekunden wird der Sprung zurück in die Ausgangslage gemacht. Eine Interaktionsepisode hat eine Dauer von 500 Zeitschritten bzw. 5 Sekunden. Die Höhe der externen Anregung auf den Interaktionsepisoden und ihr Verlauf ist bewusst in jeder Interaktionsepisode gleich gewählt. Auf diese Weise beobachtet der Regler während des Lernens nur einen sehr kleinen Teil der möglichen Systemdynamik. Dies ist ein realistisches Szenario, das der praktischen Anwendung des Reglers Rechnung trägt. Von Interesse ist nun, wie gut der Regler, der mit diesen wenigen Anregungen gelernt wurde, auf einem zufälligen Straßenverlauf mit Störungen unterschiedlicher Höhe und Länge arbeitet.

Die für das Lernen des NFQCA-Reglers benutzten Parameter sind in Tabelle 5.11 zusammengestellt. Zum Lernen werden wieder die Standardparameter des Verfahrens benutzt, wobei die Größe der neuronalen Netze in einem etwas anderen Bereich gewählt wurden, ohne sie zu optimieren. Die Kostenfunktion wird auf der Zustandsvariable der Eintauchtiefe des passiven Dämpfers definiert. Die Regelung soll diese zeitoptimal wieder auf Null zurückführen. Die Verwendung der \tanh^2 -Kostenfunktion verhindert bei schwingenden Systemen das Überschwingen über den Zielwert, das von der Bereichsdefinition im Zielbereich toleriert werden würde.

Nach 89 Iterationen des NFQCA-Verfahrens wird die beste Regelgüte erreicht, wobei nach nur 40 Iterationen die Güte des Vergleichsreglers unterschritten wird. Die 89 Iterationen entsprechen einer Interaktionszeit von $89 \cdot 5s = 445$ Sekunden. Dies entspricht weniger als 8 Minuten Interaktionszeit am System, in denen 44500 Übergänge gesammelt wurden, um einen effizienten Regler einzutrainieren.

Tabelle 5.12: Versuchsauswertung der Regler zur Schwingungsdämpfung einer Busfederung an einem Einheitsprung der Höhe 10cm.

Regler	\bar{N}	e_{∞}
ohne	378.9	0.04253
PID	86.10	0.00249
NFQCA	40.00	0.00097

In Abbildung 5.25 ist die Reglertrajektorie des eingelernten Reglers an einem einzigen Einheitsprung der Höhe 10cm zu sehen. Die Dämpfung erfüllt alle Anforderungen der Regelung. Es ist kein Überschwingen des Buskörpers über seine Ruhelage vorhanden und diese wird schnell erreicht. In der Abbildung wurden zur besseren Verständlichkeit die Straßenlage w und

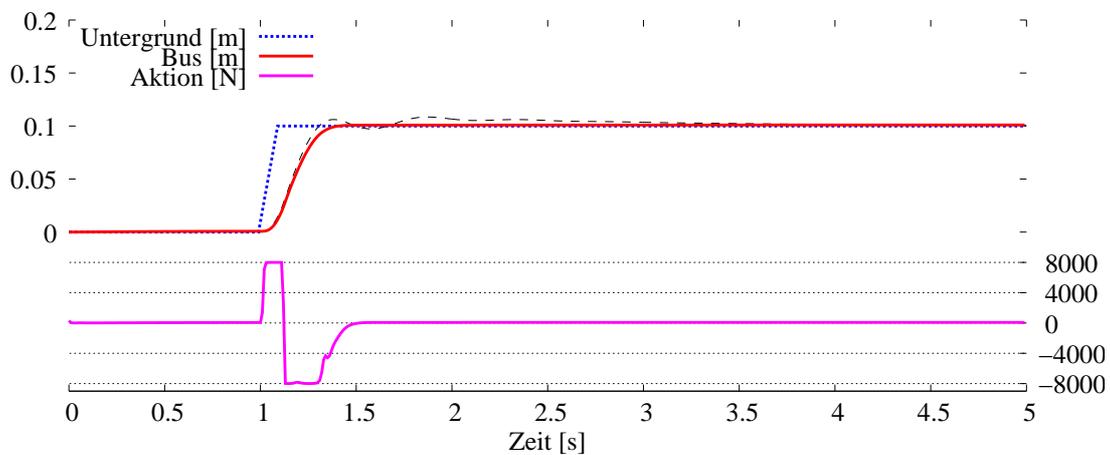


Abbildung 5.25: Systemantwort des geschlossenen Regelkreises, mit gelerntem Regler, auf Störung mit 10cm Auslenkung. Im Unterschied zu der Systemantwort des offenen Systems in Abbildung 5.23 und der Antwort des mit dem PID Regler geschlossenen Systems in Abbildung 5.24, die als Referenz eingezeichnet ist, ist kein Überschwingen des Buskörpers mehr vorhanden. Trotzdem wird die Auslenkung sehr schnell eliminiert.

die Prozessvariable x_1 , die Auslenkung des Buskörpers, verwendet. Es sei noch einmal darauf hingewiesen, dass keine dieser Prozessvariablen als Eingabe der Regelung dient. Diese arbeitet ausschließlich auf der Eintauchtiefe des Dämpfers y_1 und der Geschwindigkeit der Auslenkung des Buskörpers \dot{x}_1 . In Abbildung 5.26 ist die gelernte Strategie des Reglers in seinem Zustandsraum aufgetragen. Zusätzlich ist über der Strategie des Reglers als Funktion seiner Zustandsvariablen die Trajektorie des Reglers aus Abbildung 5.25 dargestellt.

In Tabelle 5.12 ist die Auswertung des gelernten Reglers an einem Einheitsprung der Höhe 10cm, wie in Abbildung 5.25 dargestellt, zu sehen. Die Kennzahl der Reglergüte \bar{N} , wird erfasst durch die Anzahl der Schritte, die sich der Regler außerhalb eines Toleranzbereiches von $\pm 5mm$, bezüglich der Eintauchtiefe des Federbeines, aufhält. Für den PID-Regler wurden die ermittelten Werte auf das Zeitintervall des NFQCA Reglers umgerechnet. Zusätzlich ist als Kennzahl die mittlere bleibende Regelabweichung e_{∞} über die letzten drei Sekunden der Regelung angegeben. Der gelernte Regler ist wesentlich schneller und genauer als der PID Regler.

Die Strategie des Reglers wurde auf Interaktionsepisoden mit zwei Störungen der Höhe

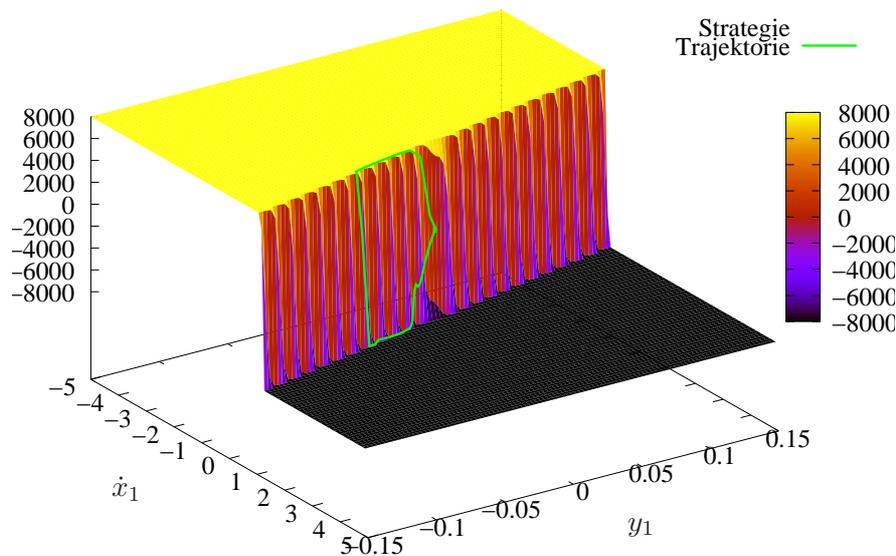


Abbildung 5.26: Die gelernte Strategie des NFQCA-Reglers. Zu sehen ist die Strategie, geplottet über dem Zustandsraum. Zusätzlich eingezeichnet ist die Trajektorie des Systems aus Abbildung 5.25

10cm und -10cm eingelernt. Dennoch ist er in der Lage, auch bei anderen Anregungen bzw. Straßenverläufen das System geeignet zu dämpfen. In Abbildung 5.27 ist die Regeltrajektorie des gelernten Reglers auf eine Folge von verschiedenen Anregungen mit unterschiedlicher Höhe zu sehen. Wieder ist nur die Auslenkung des Buskörpers und der Straßenverlauf aufgezeichnet. Die für die Regelung des Systems maßgeblichen Größen sind von der absoluten Höhe des Straßenverlaufs unabhängig. Der Regler hat mit einer nur sehr begrenzten Anregung in der Lernphase gelernt, verschiedene Störungen bzw. Anregungen durch den Straßenverlauf zu dämpfen.

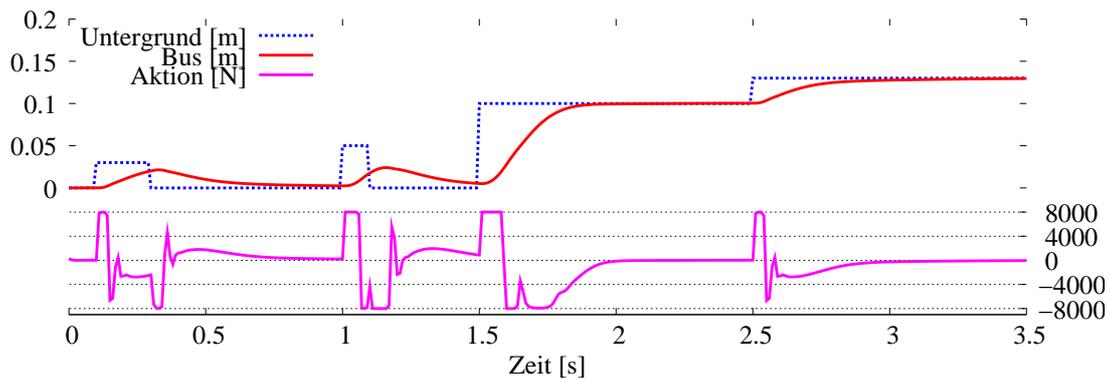


Abbildung 5.27: Der gelernte Regler angewandt auf eine Folge von Störungen. In der Lernphase wurde nur die Anregung 10cm benutzt. Trotzdem ist der Regler in der Lage auch andere Anregungen geeignet zu dämpfen.

Für die vergleichenden Auswertung der Regler wurde ein zufälliges Straßenprofil mit einer Länge von 50 Sekunden erzeugt, das aus einer zufälligen Abfolge von rechteckförmigen Auslenkungen der Straße, mit zufälliger Höhe und zufälliger Länge, besteht. Dieses enthält im Vergleich zum Profil in Abbildung 5.27 deutlich mehr Störungen, die in einer zufälligen Abfol-

Tabelle 5.13: Versuchsauswertung der Regler zur Schwingungsdämpfung einer Busfederung an einem zufälligen Straßenverlauf über 50 Sekunden.

Regler	e^{∞} Trajektorie
ohne	0.08912
PID	0.01027
NFQCA	0.01006

ge auch direkt hintereinander auftreten können. Die ersten 30 Sekunden des Straßenprofils sind in Abbildung 5.28 mit der Regeltrajektorie des NFQCA Reglers dargestellt. Das verwendete Profil entspricht dem Vorgehen bei der praktischen Anwendung, die Regler an einem auf realen Straßen aufgenommenen Profil zu testen. Zum Vergleich ist in Abbildung 5.28 zusätzlich

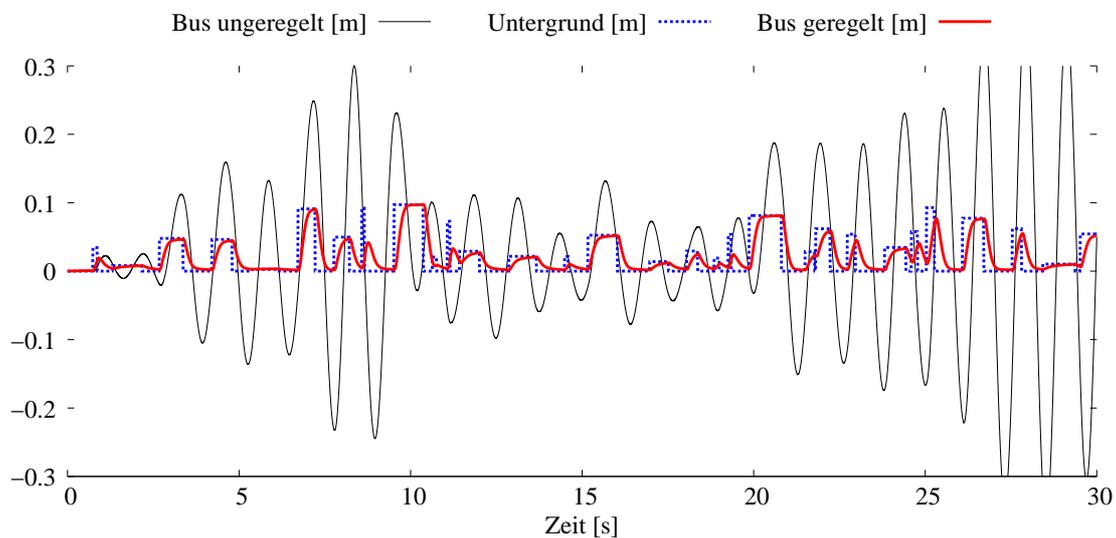


Abbildung 5.28: Der gelernte Regler angewandt auf ein Straßenprofil mit zufälligen Störungen. Der gelernte Regler dämpft die Störungen und lässt kein Schwingen des Buskörpers zu. Im Vergleich ist die Antwort des unregulierten Systems auf der Anregung zu sehen. Trotz der zufällige Abfolge der Störungen und ihre Wechselwirkung untereinander funktioniert der gelernte Regler zur aktiven Schwingungsdämpfung sehr gut.

die Trajektorie des unregulierten Systems mit aufgetragen. Das unregelte System schwingt sich schnell auf und wird somit nach wenigen Sekunden unkontrollierbar. Durch die aktive Schwingungsdämpfung durch den NFQCA Regler wird das Schwingen des Systems gänzlich verhindert. Somit wird sichergestellt, dass nicht nur der Komfort der Insassen ausreichend ist, sondern auch eine stabile Fahrdynamik des Fahrzeuges erhalten bleibt. Auf dem Profil des zufälligen Straßenverlaufs über 50 Sekunden erzeugt der NFQCA Regler eine mittlere absolute Abweichung von $\bar{e}_T = 1.006\text{cm}$ über die gesamte Trajektorie. Er reduziert das Schwingen des Buskörpers und zeigt kein Überschwingen mehr. Der PID Regler kann mit $\bar{e}_T = 1.027\text{cm}$ ein ähnlich gutes Ergebnis auf dieser Trajektorie erzeugen, zeigt jedoch ein deutliches Überschwingen. In Tabelle 5.13 sind die Auswertungen der Regler an diesem Profil zusammengefasst.

5.3.2 Diskussion

An einem für die aktive Schwingungsdämpfung typischen Problem konnte gezeigt werden, dass ein selbstlernender Regler mit NFQCA in der Lage ist, mit nur wenigen Interaktionen und beispielhaften Anregungsmustern während der Lernphase einen effektiven Dämpfungsregler zu lernen. Trotz der durch die im Regler nicht repräsentierten parasitären Anteile der Systemdynamik kann das schwingende System effektiv gedämpft werden. Der gelernte Regler ist in der Lage, auch auf Profilen mit anderen Anregungen als in der Lernphase diese effektiven Dämpfungseigenschaften zu zeigen. Für die Interaktionsepisoden wurde eine praktikable Möglichkeit demonstriert, bei der der Regler mit weniger als 4 Minuten reiner Interaktionszeit mit dem System, unter wenigen Anregungen, eine effiziente Regelstrategie, für beliebige Anregungen lernt. Dies macht ihn zu einer interessanten Alternative für reale Problemstellungen dieses Typs.

Die aktive Schwingungsdämpfung ist sowohl im Maschinenbau als auch in der Automobilindustrie ein sehr aktuelles Forschungsgebiet. Im Rahmen eines Industrieprojektes wurde eine Machbarkeitsstudie durchgeführt, welche die Anwendung des gezeigten Reglerkonzeptes auf eine aktive Schwingungsdämpfung eines Serienfahrzeuges untersucht. In diesem Projekt liegt das Fahrzeug als komplexe Finite-Elemente-Simulation einer kompletten Fahrzeugkarosserie mit Fahrwerk vor. Das Ziel der Studie ist die Erstellung eines selbstlernenden Reglers zur aktiven Schwingungsdämpfung der auftretenden Torsionsschwingungen, die konstruktionsbedingt an diesem Fahrzeugtyp vermehrt auftreten. Um diese zu eliminieren, wird eine aktive, in ihrer Länge verstellbare, Verstrebung ins Fahrzeugheck eingebaut, die durch einen Regler geeignet angesteuert werden muss. Als Zustandsvariablen des Reglers dienen simulierte Sensoren an verschiedenen Stellen in der Karosserie, die als Beschleunigungsaufnehmer Informationen über Auslenkungen und Vibrationen an diesen Stellen liefern. Das für das Lernen benutzte Vorgehen ist ähnlich zu dem hier vorgestellten Versuch der Schwingungsdämpfung. Im Unterschied zu den Versuchen hier, wird das gesamte Fahrzeug betrachtet. Über einen simulierten hydraulischen Teststand wird während der Interaktionsepisoden ein einfaches Anregungsprofil auf die vier Räder des Fahrzeuges gegeben. In der Studie konnte gezeigt werden, dass mit den in dieser Arbeit vorgestellten selbstlernenden Reglern in einer sehr geringen Interaktionszeit von nur wenigen Minuten ein effizienter Regler zur aktiven Schwingungsdämpfung an diesem Problem gelernt werden kann. Die gelernten Regler sind in ihrer Regelgüte vergleichbar mit den vom Industriepartner zur Verfügung gestellten Referenzreglern. Auf einer Teststrecke, basierend auf realen Daten einer Autobahn, kann mit den gelernten Reglern sogar eine deutliche Verbesserung der Dämpfung, im Vergleich zu den Referenzreglern, erzielt werden.

5.4 Elektromagnetisches Schweben einer Metallkugel

Das kontaklose Positionieren eines Körpers mit magnetischen Eigenschaften in einem durch magnetische Felder erzeugten Schwebezustand ist eine Technologie, die ein weites Feld von Anwendungen besitzt. Beispiele für den Einsatz solcher Systeme finden sich unter anderem bei reibungsfreien Lagern, Magnetschwebbahnen und der kontaklosen Positionierung von Messobjekten. Dabei stellt die exakte und schnelle Positionierung des schwebenden Objektes eine zentrale Forderung für den Einsatz solcher Systeme dar. Da solche Systeme typischer Weise als offenes dynamisches System extrem instabil sind und die elektromechanischen Eigenschaften einen hohen Grad an Nichtlinearität bedingen, ist der Einsatz von leistungsfähigen Reglern notwendig. Der Entwurf eines geeigneten Reglers ist für den Entwickler eine sehr komplexe

und anspruchsvolle Aufgabe, da ein linearer Regler typischer Weise nur für einen sehr kleinen Bereich um einen Arbeitspunkt Gültigkeit besitzt. Es existieren verschiedenste Technologien zur Umsetzung und Anwendung des magnetischen Schwebens. Diese können sehr komplexe Lösungen und spezialisierte Komponenten enthalten. Ein standardisierter Aufbau zur Entwicklung und zur Untersuchung von Reglern in diesem Bereich ist ein gravitationsbasiertes eindimensionales magnetisches Schwebesystem [115]. Dieses besteht aus einem Elektromagneten, der anziehende Kräfte auf eine unter ihm liegende Stahlkugel ausüben kann. Am Beispiel des NFQCA wird ein selbstlernender Regler für die spannungsbasierte Positionierung der Stahlkugel in diesem Versuchsaufbau entwickelt. Von besonderem Interesse in dieser Untersuchung ist die erzielbare Genauigkeit über den gesamten Arbeitsbereich der Regelungsaufgabe unter dem hohen Grad an Nichtlinearität des Systems.

Der in der Simulation verwendete Versuchsaufbau ist in Abbildung 5.29 dargestellt [115]. Er besteht aus einer Stahlkugel mit der Masse M , auf die ein darüber hängender Elektromagnet eine anziehende Kraft ausüben kann. Die Eingriffsgröße des Systems ist die Spannung u , die am Elektromagneten angelegt wird. Die Charakteristik der Spule und des erzeugten elektrischen Feldes ist definiert durch die Parameter R , x_∞ , L_∞ und ξ . Die Position und Geschwindigkeit der Kugel kann berührungslos über einen Lasersensor unterhalb einer Stahlplatte gemessen werden. Die Systemvariablen sind gegeben durch die Position d der Kugel als Länge des Luft-

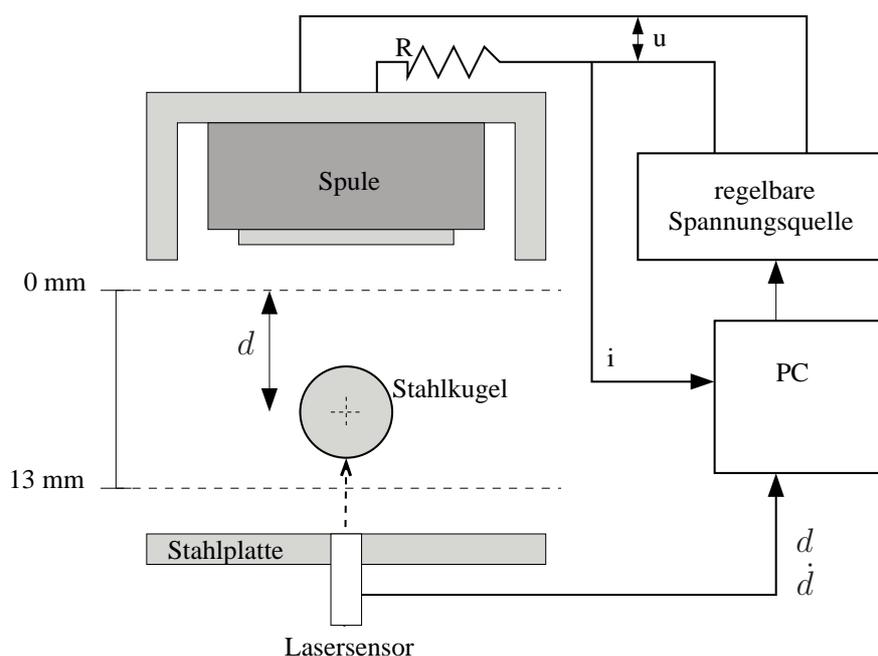


Abbildung 5.29: Der experimentelle Aufbau eines Magnetschweberversuchs für eine Metallkugel. Durch eine Spule mit Metallkern kann ein elektromagnetisches Feld erzeugt werden, das mit der Stahlkugel interagiert. Die Position der Kugel wird mit Hilfe eines Abstandssensors berührungslos gemessen. Die Regelung soll eine möglichst genaue und schnelle Positionierung der Kugel ermöglichen.

spalts zwischen Kugel und Elektromagnet, der Geschwindigkeit der Kugel \dot{d} und dem Strom i , der durch die Spule fließt. Durch Definition des Zustandsvektors des dynamischen Systems als $x = [x_1, x_2, x_3]^T = [d, \dot{d}, i]^T$, ergeben sich die Systemgleichungen für die Simulation des

Tabelle 5.14: Physikalische Modellparameter des elektromagnetischen Aufbaus der schwebenden Stahlkugel für die Simulation.

Masse Stahlkugel	M	=	0.8	[kg]
Elektrischer Widerstand	R	=	11.68	[Ω]
Spulenparameter	x_∞	=	0.007	[m]
Spulenparameter	L_∞	=	0.8052	[H]
Spulenparameter	ξ	=	0.001599	[Hm]
Eingriffsgröße	u	\in	[-60, 60]	[V]
Regelintervall	Δ_t	=	0.004	[s]

Prozesses zu:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{pmatrix} = \begin{pmatrix} x_2 \\ \alpha(x) \\ \beta(x) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \gamma(x) \end{pmatrix} u \quad (5.11)$$

$$\alpha(x) = g - \frac{\xi x_3^2}{2M(x_\infty + x_1)^2} \quad (5.12)$$

$$\beta(x) = \frac{x_3(\xi x_2 - R(x_\infty + x_1)^2)}{\xi(x_\infty + x_1) + L_\infty(x_\infty + x_1)^2} \quad (5.13)$$

$$\gamma(x) = \frac{x_\infty + x_1}{\xi + L_\infty(x_\infty + x_1)} \quad (5.14)$$

Für die Versuche wird das dynamische System durch numerische Integration mit dem Runge-Kutta-Verfahren mit den in Tabelle 5.14 gegebenen Parametern simuliert. Für die Simulation wird der Weg der Kugel auf den physikalisch erlaubten Bereich zwischen 0 und 13mm begrenzt. Bei 13mm liegt die Kugel am Boden der Stahlplatte auf, bei 0mm hängt sie oben an der Spule. In beiden Fällen wird die Geschwindigkeit in der Simulation auf Null gesetzt, der Strom entwickelt sich entsprechend weiter. Vor jedem Lauf wird eine Initialisierungsphase durchgeführt, bei der die Kugel bei 13mm, also auf der Stahlplatte liegend, gestartet wird. Für 0.5 Sekunden wird eine Spannung von 15V angelegt. Diese Initialisierungsphase stellt sicher, dass sich der Strom in einem zulässigen Bereich befindet und die magnetische Sättigung initial erreicht ist. Diese Initialisierungsphase stellt ausschließlich sicher, dass die Simulation in einem physikalisch relevanten Bereich arbeitet. Die Kugel bleibt in dem sich einstellenden Gleichgewichtszustand auf der Stahlplatte liegen und wird nicht angehoben. Dies muss von den Reglern erreicht werden, die am Anschluss an diese Initialisierungsphase gestartet werden.

Der Entwurf eines nichtlinearen Reglers ist ein in der Literatur oft beschriebenes schwieriges Problem und Gegenstand der aktuellen Forschung im Bereich der nichtlinearen Systeme. Das hier betrachtete Beispielsystem wurde von Yang et al. in mehreren Publikationen mit verschiedenen Reglerstrukturen untersucht [114–117]. Die klassischen Reglerkonzepte für das magnetische Schweben reichen dabei von der Linearisierung der Systemdynamik bis zum Entwurf adaptiver nichtlinearer Regler. Das Hauptaugenmerk beim Entwurf dieser Regler ist die Robustheit gegen die beim Entwurf unbekanntenen physikalischen Parameter des Systems. Die entwickelten Regler bedürfen typischer Weise einer sehr sorgfältigen und zeitaufwändigen Entwurfsphase und besitzen komplexe Reglerstrukturen. Die Regelungsstrategien sind dabei hochdynamisch und finden für Systeme mit nicht genau bekannten physikalischen Parametern kei-

Tabelle 5.15: Die für das Lernen verwendeten Parameter.

Zustand x	4 dimensional	d \dot{d} I $\tilde{d} = d_d - d$	Position Kugel Geschwindigkeit Kugel Strom in Spule Regelabweichung
Kostenfunktion	\tanh^2	x^d μ C	$(-, -, -, 0)$ $(0, 0, 0, 0.002)$ 0.01
Aktion	1 dimensional	u	Spannung an Spule
Q-Funktion	neuronal	Topologie	5-15-20-1 (600 Epochen RProp)
Strategie-Funktion	neuronal	Topologie	4-15-1 (1000 Epochen RProp)
Exploration		ϵ Δ	0.15 $\sim N(0,1)$

ne glatte Regelungsstrategie.¹ Durch die Anwendung eines selbstlernenden Reglers entfällt das Problem der unbekanntenen physikalischen Parameter, da der Regler direkt mit dem System interagiert und kein Entwurfsprozess notwendig ist.

5.4.1 Selbstlernender Regler zum Schweben der Stahlkugel

Für die Modellierung des selbstlernenden Reglers wird zusätzlich zu den Systemvariablen die Regelabweichung in den Zustandsvektor integriert. Der Zustandsvektor des Reglers ergibt sich somit zu $x = (d, \dot{d}, I, e)$ mit $e = d_d - d$. Als Aktionsraum steht dem Regler die Ausgabe einer Spannung zwischen $-60V$ und $60V$ zur Verfügung.

Die Interaktionsepisoden mit dem System werden jeweils mit einer Länge von 160 Zeitschritten ausgeführt. Da das System nicht die beliebige Wahl von Startzuständen zulässt, wird jede Episode im gleichen Zustand gestartet, bei dem die Kugel auf der unteren Stahlplatte aufliegt. Für jede Interaktionsepisode wird nun zu Beginn ein zufälliger Zielwert aus dem Arbeitsbereich gewählt. Da die Regelung auf diese Weise sich immer nur von einer Richtung dem Zielwert nähert, wird nach 80 Zeitschritten ein neuer zufälliger Zielwert gewählt, um die Regelung robuster gegen den Wechsel der Führungsgröße zu machen.

Die Evaluation der Regler wird auf 50 zufällig gewählten Zielpositionen durchgeführt, auf die der Regler 200 Schritte angewandt wird. Für die Bewertung der Zeitoptimalität wird ein Zielbereich mit der Breite $\mu = 0.4mm$ um die Zielposition definiert und die Schritte \bar{N} außerhalb dieses Zielbereiches betrachtet. Die Genauigkeit der Regelung auf der Testmenge der 50 Zielpositionen wird über die durchschnittliche absolute Abweichung der Regeltrajektorie \bar{e}_∞ auf den jeweils letzten 150 Schritten der Regelung berechnet.

Das Vorgehen bei der Exploration während der Interaktionsepisoden ist bei diesem System eine wichtige Komponente des Lernverfahrens. Die in der Arbeit bisher diskutierten Explorationsstrategien einer rein zufälligen Wahl aus dem Aktionsbereich oder der rein normalverteilten Wahl um die aktuelle Strategie können das vorliegende System nicht zufriedenstellend explorieren. Dies liegt an den Eigenschaften des dynamischen Verhalten des Prozesses. Zu Beginn der Trajektorie werden konstante, maximale Spannungen benötigt, um die Kugel in Bewegung zu setzen. Danach werden sehr genaue Aktionen erforderlich, welche die Kugel möglichst nahe an die Zielposition überführen und dort halten. Bei reiner normalverteilter Exploration um

¹In der angegebenen Literatur finden sich keine Kennzahlen über die Güte der entworfenen Regler. Da eine Implementation der Ansätze sehr komplex ist und zentrale Parameter der Regler nicht publiziert sind ist ein Vergleich mit den publizierten Verfahren an dieser Stelle leider nicht möglich.

die aktuelle Strategie wird dieses initiale Verhalten zu selten beobachtet und nur ein uninteressanter Bereich des Zustandsraumes exploriert, der nicht zur Lösung beiträgt. Im Gegensatz dazu wird mit der rein zufälligen Wahl der Aktion während der Exploration das initiale Verhalten zum Anheben der Kugel sehr schnell gelernt, es lässt sich jedoch nur eine unzureichende Genauigkeit im gesamten Arbeitsbereich der Regelung erzielen. In diesem speziellen Fall wird deshalb eine Mischung aus beiden Explorationsstrategien benutzt. In 15% der Schritte in einer Interaktionsepisode wird exploriert. Dabei wird in 50% der Explorationschritte eine rein zufällige Wahl der Aktion aus dem Aktionsraum durchgeführt. Ansonsten wird mit einer sehr kleinen Varianz, normalverteilt um die aktuelle Strategie, zufällig exploriert. Dieses Vorgehen erzeugt ein für dieses System geeignete Explorationsstrategie, die es zulässt, sowohl die initiale Sequenz an konstanten Aktionen, als auch die genaue Wahl der Aktionen für die Regelung zu finden. Das System ist hier exemplarisch für andere Systeme zu sehen, die ähnliche Eigenschaften besitzen. Es zeigt, dass sich durch geeignete Wahl der Explorationsstrategie die Lernfähigkeit der Methoden anpassen lassen, ohne quantitatives Wissen über die Dynamik des Systems explizit vorzugeben. Die für den Lernversuch verwendeten Parameter sind in Tabel-

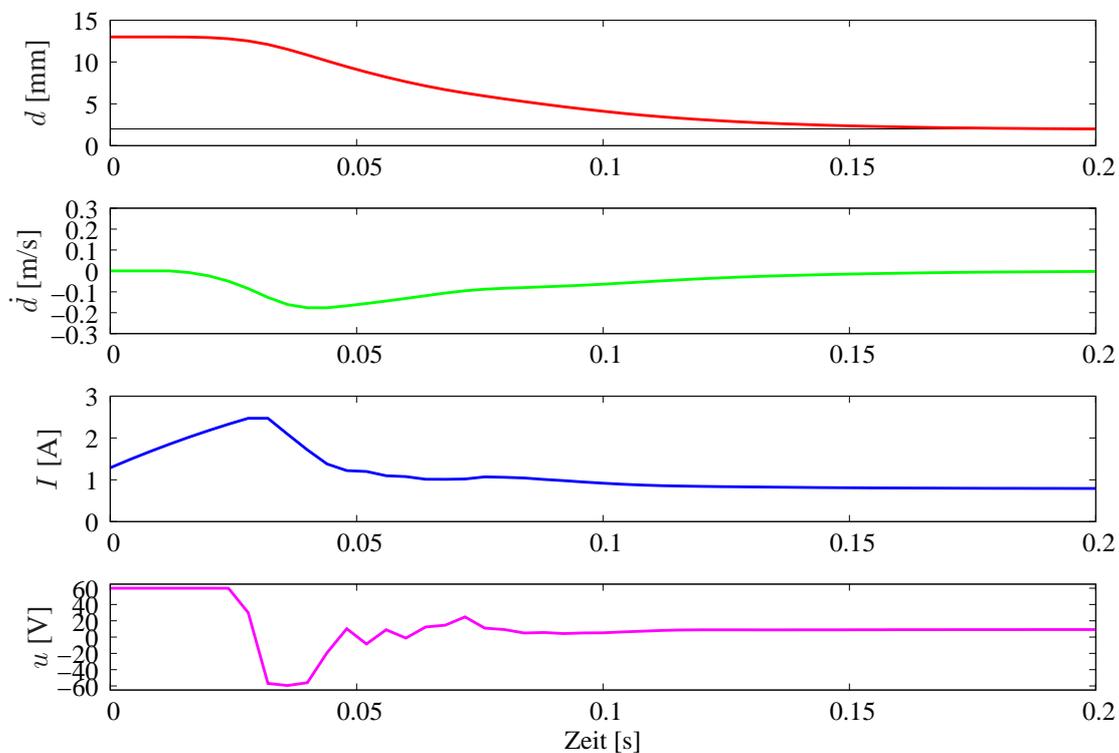


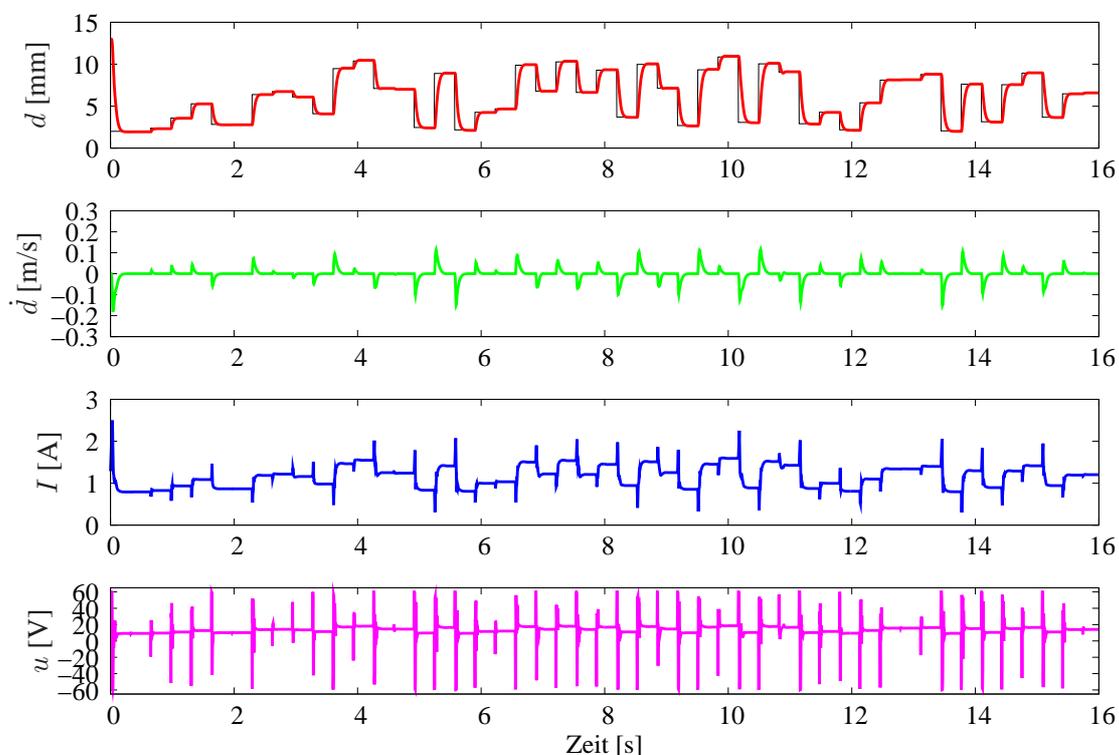
Abbildung 5.30: Die Regelung auf eine Zielposition, ausgehend vom Startzustand. Der Spannungsverlauf zu Beginn der Regeltrajektorie muss einem bestimmten Profil folgen um die Kugel in Bewegung zu setzen. Dies hat der selbstlernende Regler aus der Systemdynamik gelernt.

le 5.15 angegeben. Eine Interaktionsepisode dauert in Echtzeit 0.64 Sekunden. Nach weniger als 30 Trainings- und Interaktionsepisoden erfüllt der gelernte Regler zum ersten Mal die Anforderungen der Regelung. Alle Testtrajektorien werden in den Zielbereich geregelt und dort gehalten. Dies entspricht bis zu diesem Zeitpunkt einer gesamten Interaktionszeit von nur 19.2 Sekunden bzw. einer gesammelten Menge von 4800 Übergängen. Der gelernte Regler ist zu dieser Zeit noch nicht optimal und erzeugt eine mittlere Abweichung von $\bar{e}_\infty = 0.0704\text{mm}$

Tabelle 5.16: Zusammenfassung der Versuche eines selbstlernenden Reglers mit NFQCA zum Schweben einer Metallkugel.

Regler	\bar{N}	\bar{e}_∞
NFQCA	26.74	0.0203mm

über die 50 zufälligen Zielpositionen. Die schlechteste Trajektorie erzeugt dabei einen Fehler von 0.1mm und die beste einen Fehler von 0.01mm. Nach 86 Iterationen wird das beste Ergebnis des Reglers erzielt. Diese erreicht eine mittlere Genauigkeit von $\bar{e}_\infty = 0.0203\text{mm}$. Die beste Trajektorie der 50 Testtrajektorien hat dabei eine absolute mittlere Abweichung von numerisch Null, die schlechteste eine Abweichung von 0.0409mm. Der Regler benötigt im Durchschnitt 26.74 Schritte, um die Position der Kugel in den definierten Zielbereich zu bringen. In Abbildung 5.30 ist die Regeltrajektorie des gelernten Reglers von der initialen Lage

**Abbildung 5.31:** Der mit NFQCA gelernte Regler an einer Folgeregelung der schwebenden Kugel auf zufällige Sprungfolgen von Zielwerten.

der Kugel auf eine Zielposition von 3.59mm in den ersten 200ms der Regelung dargestellt. Zu Beginn der Regelung liegt die Kugel auf der Stahlplatte auf. Der Regler hat gelernt, dass er für mindestens 6 Zeitschritte die Spannung auf einem sehr hohen Wert halten muss, damit sich die Kugel in Bewegung setzt. Während dieser Zeit steigt der Strom in der Spule stark an. Hat sich die Kugel in Bewegung gesetzt, wird die Spannung kurzzeitig vom Regler invertiert, um die Kugel wieder abzufangen. Bemerkenswert ist die konstante Eingriffsgröße, die der Regler nach Erreichen des Zielwertes auf das System anwendet und die von den klassischen Reglern aufgrund der Parameterunsicherheit beim Entwurf nicht erzeugt werden kann. Nach weniger

als 200ms hat der Regler die Kugel in der vorgegebenen Zielposition ohne ein Überschwingen stabilisiert. In Abbildung 5.31 ist der Regler auf einer Folgeregelung mit mehreren Zielwerten

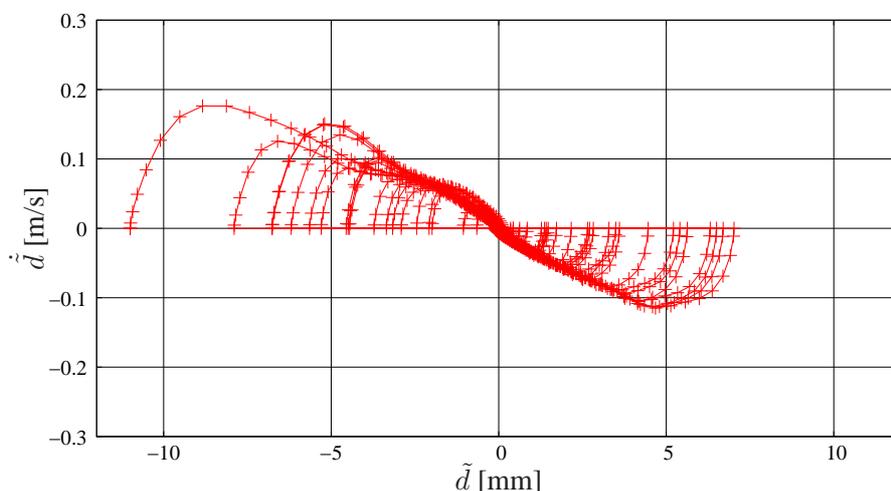


Abbildung 5.32: Die Darstellung der Regelabweichung und der Fehlergeschwindigkeit als Phasendiagramm der Folgeregelung auf eine trajektorie mit zufälligen Sprungfolgen der Führungsgröße. Bei Auftreten einer Abweichung wird der Fehler schnell ausgeregelt und das System im Zielzustand stabilisiert.

zu sehen. Das Profil der Führungsgröße besteht aus Zielwerten, die alle 82 Zeitschritte bzw. 328ms zufällig aus dem Arbeitsbereich der Regelung neu ausgewählt werden. Bei Auftreten einer Regelabweichung korrigiert der Regler die Position der Kugel sehr schnell und findet für alle Zielwerte eine genaue Aktion, um die Kugel an dieser Stelle zu stabilisieren. In Abbildung 5.32 ist die Regeltrajektorie aus Abbildung 5.31 als Phasenplot der Regelabweichung e und ihrer Geschwindigkeit \dot{e} dargestellt. Die Ableitung der Regelabweichung \dot{e} wird dabei durch $\dot{e} = -\dot{d}$ angenommen. Dies ist bei konstanter Führungsgröße korrekt, bei einer variierenden Führungsgröße hingegen wird \dot{e} , während der Wert variiert, unterschätzt. Am Phasenplot wird die gelernte Regelstrategie des selbstlernenden Reglers deutlich. Ist der Fehler positiv, die Kugel also höher als der aktuelle Zielwert, so kann der Regler die Kugel fallen lassen und die Trajektorien im Phasenplot laufen nach einer sehr schnellen Annäherungsphase auf einer Geraden bis zum stabilen Punkt im Ursprung des Phasendiagramms. Dies erinnert stark an die Vorgehensweise eines 'Sliding Mode'-Reglers [95], bei dem eine solche Gerade im Phasenraum durch den Entwurf des Reglers vorgegeben wird. Ist der Fehler negativ, die Kugel muss somit vom Magneten angehoben werden, so ergibt sich im Phasenraum keine eindeutige Gerade mehr. Durch die nichtlineare Natur des Systems kann der Regler die Kugel in verschiedenen Bereichen des Zustandsraumes unterschiedlich schnell in den Nullpunkt des Phasenraumes überführen. An diesem Beispiel wird die Problematik des Entwurfs eines nichtlinearen Reglers mit klassischen Methoden deutlich. Diese Hyperebenen im Phasenraum müssen bei diesem Entwurf beschrieben und gefunden werden und werden meist durch einfache Geraden angenähert. Im Gegensatz dazu ist der selbstlernende Regler ohne eine a priori Information dazu in der Lage, eine Strategie zu finden, die für das System annähernd optimal ist.

5.4.2 Die Annäherung der Trajektorienregelung

Die Führungsgröße und Regelabweichungen für den Regler sind ausschließlich auf der Position der Kugel definiert worden. In Abbildung 5.33 ist die Regeltrajektorie des gelernten Reglers auf einem sinusförmigen Verlauf der Führungsgröße zu sehen. Diese dynamische Änderung der Führungsgröße kann durch den so definierten Regler nicht voll kompensiert werden, da die Rückführung des Fehlersignales die Dynamik der Führungsgröße nicht beinhaltet. Dadurch wird eine systematische Regelabweichung erzeugt, da der Regler in jedem Zeitschritt das Ziel hat, den aktuellen Fehler im nächsten Zeitschritt auszuregeln. In anderen Worten, der Regler nimmt den Regeleingriff so vor, dass die Position der Kugel im nächsten Zeitschritt den aktuellen Fehler ausgleicht und mit Geschwindigkeit Null dort verbleibt. Im nächsten Zeitschritt kann der Fehler durch die Dynamik der Führungsgröße jedoch nicht kompensiert werden, da sich diese geändert hat. Dies ist ein genereller Nachteil, der sich bei Reglern mit Rückführung der reinen Regelabweichung ergibt. Eine Verbesserung des Regelungsverhaltens ergibt sich

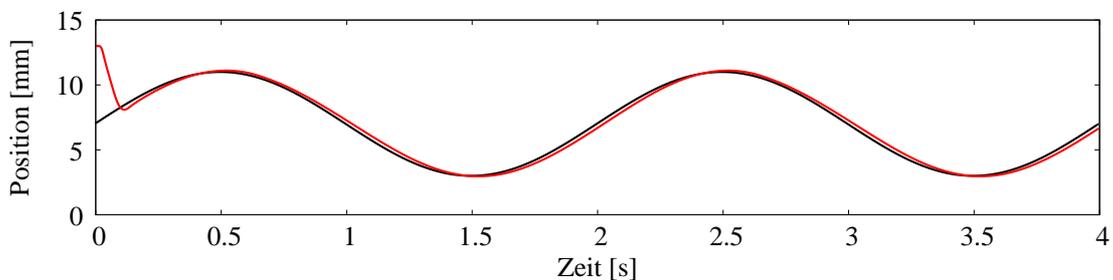


Abbildung 5.33: Der Verlauf der Regeltrajektorie auf einer Folgeregelung mit kontinuierlichem Verlauf der Führungsgröße. Deutlich ergibt sich die Regelabweichung im Vergleich zur Vorgabe.

bei Verwendung einer Trajektorienregelung des Systems. Hierbei werden nicht nur die aktuelle Regelabweichung e des Systems, sondern auch ihre Ableitungen \dot{e} , \ddot{e} , \dots , je nach Ordnung der Systemdynamik, berücksichtigt. Bei der vorgestellten Modellierung der selbstlernenden Regler kann diese Fehlerdynamik beim Lernen berücksichtigt werden. Das intelligente Fehlerglied muss dabei die geeigneten Systemübergänge erzeugen, die es dem Regler erlauben, zwischen der Dynamik des Systems und der Dynamik der Führungsgröße zu unterscheiden. Dies ist jedoch ein aufwändiger Prozess und verlangsamt den Lernvorgang, da deutlich mehr Übergänge am System gesammelt werden müssen um die Fehlerdynamik in verschiedenen Zuständen zu explorieren.

Für die Regelung der schwebenden Stahlkugel soll hier eine weitere Möglichkeit vorgestellt werden, die auf der Struktur der selbstlernenden Regler basiert. Der ohne die dynamische Änderung der Führungsgröße eingelernte Regler soll so angepasst werden, dass er auf der Aufgabe der Trajektorienregelung ein besseres Ergebnis erzielt. Die Information, auf welcher der Regler gelernt hat, seine Entscheidung zu treffen, basiert unter anderem auf der aktuellen Geschwindigkeit der Kugel und der aktuellen Regelabweichung. Die Grundidee ist es nun, die vom Regler gelernten Zusammenhänge im System auszunutzen. Dies wird dadurch erreicht, dass die Geschwindigkeit der Kugel im Zustandsvektor des Reglers mit der Veränderung der Führungsgröße angepasst wird. Der neue Zustandsvektor ergibt sich dann zu $x = (d, \dot{d} - \dot{d}_d, I, e)$. Dem Regler wird somit die Information über die Veränderung der Führungsgröße in der beobachteten Geschwindigkeit der Kugel kodiert übergeben.

In Abbildung 5.34 ist die Regeltrajektorie des gelernten Reglers bei Ausführung mit dem ver-

änderten Zustandsvektor auf dem sinusförmigen Verlauf der Führungsgröße zu sehen. Deutlich ist der reduzierte Trajektorienfehler zu sehen. Dies zeigt sich vor allem beim initialen Erreichen der vorgegebenen Trajektorie. Im Vergleich zu Abbildung 5.33 wird die Trajektorie ohne Überschwingen erreicht. Der systematische Fehler wird durch diese Korrektur des Zustandes für den gelernten Regler ausgeglichen. Die Geschwindigkeit der Führungsgröße wurde dabei durch einfache Differenzenbildung in zwei aufeinanderfolgenden Zeitschritten gebildet. Dies erlaubt es, beliebige Profile der Führungsgröße vorzugeben, ohne deren Ableitung zu kennen. Im Vergleich zu Methoden des nichtlinearen Reglerentwurfes werden bei der Trajektorienregelung keine weiteren Annahmen über die Systemdynamik gemacht.

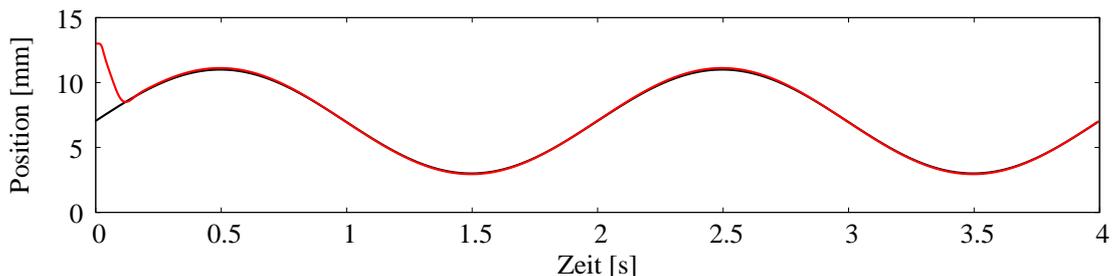


Abbildung 5.34: Der Verlauf der Regeltrajektorie (rot) auf einer Folgeregelung mit kontinuierlichem Verlauf der Führungsgröße (schwarz). Durch die Verwendung der Trajektorienregelung kann die Abweichung von der Führungsgröße über die Zeit deutlich reduziert werden.

5.4.3 Diskussion

Am Beispiel eines spannungsgeregelten elektromagnetischen Systems zum Schweben einer Stahlkugel wurde die erzielbare Regelgüte der selbstlernenden Regler für Systeme mit hohen Nichtlinearitäten untersucht. Der mit NFQCA gelernte Regler kommt mit einer mittleren Abweichung von etwas mehr als zwei Hundertstel Millimeter im ausgeregelten Zustand der Messgenauigkeit eines realen Abstandssensors schon sehr nahe. Die zum Lernen benötigte Interaktionszeit von unter 2 Minuten ist ebenfalls in einem realistischen Bereich, um den Regler für Probleme diesen Typs in realen Anwendungen einzusetzen. Der Regler lernt trotz der vorhandenen Nichtlinearitäten des Systems in dieser kurzen Zeit und zeigt auf einer Folgeregelung mit zufällig springendem Zielwert eine zufriedenstellende Qualität. Um die Folgeregelung bei kontinuierlich variierender Führungsgröße zu verbessern, wurde an diesem System exemplarisch eine Annäherung der Trajektorienregelung demonstriert, die für viele andere Systeme ebenfalls anwendbar ist. Durch nachträgliche Veränderung der Zustandsinformation zeigt der Regler auf einer Trajektorienregelung ein deutlich besseres Ergebnis als mit der reinen Folgeregelung.

5.5 Die Regelung einer Heizspirale

Die bisherigen Problemstellungen betrachten Störungen, die von externen Variablen abhängen und für den Regler selbst nicht beobachtbar sind. Es existieren jedoch auch Problemtypen, deren Systemdynamik von externen Umgebungsvariablen abhängt, die er durch geeignete Messeinrichtungen erfassen kann. Ein Beispiel eines solchen Systems mit externen Prozessvariablen ist die Regelung einer Heizspirale in einer Lüftungs- oder Klimaanlage, die Kretschmar in [51]

vorstellt. Systeme dieser Art werden in der Literatur als HVAC-Probleme (Heating, Ventilation and Air Conditioning) geführt. Sie repräsentieren eine Klasse schwieriger Regelungsprobleme mit starken Nichtlinearitäten, die in der Regelungstechnik oft betrachtet wurden. HVAC-Systeme haben einen hohen Grad an nichtlinearem Systemverhalten und variieren in einem großen Bereich ihrer Dynamik ihre Arbeitspunkte.

Das von Kretchmar vorgestellte Modell ist eine zeitdiskrete Repräsentation, das anhand von Daten einer realen Anlage erzeugt wurde. In Abbildung 5.35 ist eine schematische Darstellung des Modells zu sehen. Das Modell des Systems besitzt drei interne Prozessvariablen, drei ex-

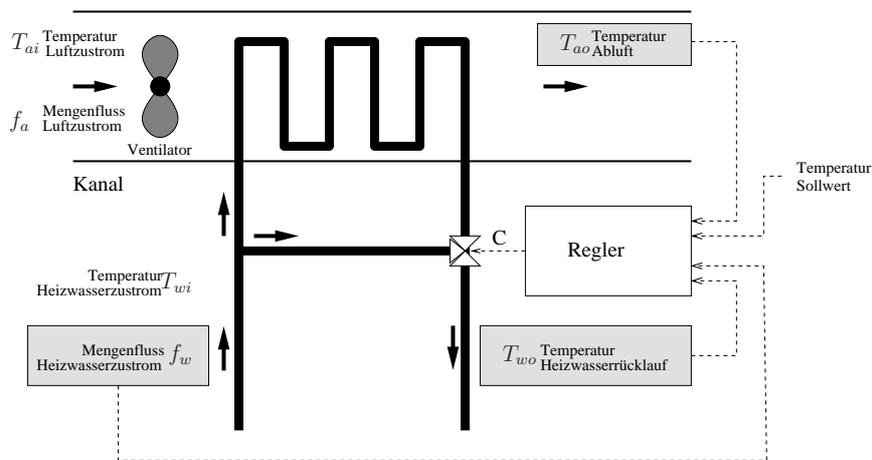


Abbildung 5.35: Das Modell einer Heizspirale in einer Klimaanlage. Die ausströmende Luft soll auf eine bestimmte Temperatur geregelt werden. Dazu kann ein Ventil Heizwasser mit einer bestimmten Temperatur aus einem Boiler durch eine Heizspirale im Luftstrom leiten.

terne Systemvariablen und eine Eingriffsgröße. Die drei externen Prozessvariablen beschreiben die Umgebungsbedingungen, in denen das System arbeitet. Dies ist die Temperatur des Luftstroms von außen, T_{ai} , die Temperatur des Heizwassers am Eingang der Heizspirale, T_{wi} und der Massenfluss des Luftstroms durch den Lüftungskanal, f_a . Die beiden Temperaturen werden von externen Bedingungen der Umgebung, der Außentemperatur und der Boilertemperatur vorgegeben. Der Luftmassenstrom wird durch Ventilatoren im Lüftungssystem erzeugt. Diese Variablen beeinflussen den Prozess, werden jedoch weder vom Regler noch vom Prozess selbst beeinflusst.

Die einzige Eingriffsgröße des Prozesses ist die Öffnung des Ventils der Heizspirale $u \in [670, 1400]$. Über dieses Ventil kann mehr oder weniger Heizwasser durch die Heizspirale im Luftstrom geleitet werden. Indirekt beeinflusst wird somit auch die Temperatur des Luftstromes am Ausgang des Lüftungskanals, da die Flussrate des Heizwassers in der Heizspirale die Menge an Energie bestimmt, die an den Luftstrom abgegeben werden kann. Die drei internen Prozessvariablen sind somit die Flussrate des Heizwassers in der Heizspirale, f_w , die Temperatur des Heizwassers nach dem Durchlaufen der Heizspirale, T_{wo} , und zuletzt die Temperatur der Luft, die den Lüftungskanal verlässt, T_{ao} . Der Regler soll die Prozessvariable T_{ao} so nahe wie möglich an dem Referenzsignal des Sollwerts T_{ao}^d halten.

Die Systemgleichungen des zeitdiskreten Modells ergeben sich zu ²:

$$\begin{aligned}
 F_w &= 6.7210^{-10}u^3 - 2.3010^{-6}u^2 + 2.1810^{-3}u - 0.2823 & (5.15) \\
 T_{wo} &= T_{wo} + 0.649f_w T_{wi} - 0.649f_w T_{wo} \\
 &\quad - 0.012T_{wi} - 0.012T_{wo} + 0.023T_{ai} + 0.104f_w T_{ai} \\
 &\quad - 0.052f_w T_{wi} - 0.052f_w T_{wo} + 0.028f_a T_{ai} \\
 &\quad - 0.014f_a T_{wi} - 0.014f_a T_{wo} \\
 T_{ao} &= T_{ao} + 0.197 * f_a * T_{ai} - 0.197 * f_a * T_{ao} \\
 &\quad + 0.016 * T_{wi} + 0.016 * T_{wo} - 0.032 * T_{ai} \\
 &\quad + 0.077 * f_w * T_{wi} + 0.077 * f_w * T_{wo} - 0.015 * f_w * T_{ai} \\
 &\quad + 0.022 * f_a * T_{wi} + 0.022 * f_a * T_{wo} - 0.045 * f_a * T_{ai} \\
 &\quad + 0.206 * T_{ai} - 0.206 * T_{ai}
 \end{aligned}$$

Im ersten Schritt wird das System unter konstanten Umgebungsbedingungen betrachtet. In diesem Fall seien diese gegeben durch die drei externen Prozessvariablen $T_{wi} = 78.0C$, $T_{ai} = 12.0C$ und $f_a = 0.8kg/s$. Hierdurch ergibt sich ein interessantes regelungstechnisches Problem mit nichtlinearer Prozessdynamik, für das in der HVAC-Industrie zumeist PI-Regler (Proportional Integral) verwendet werden.

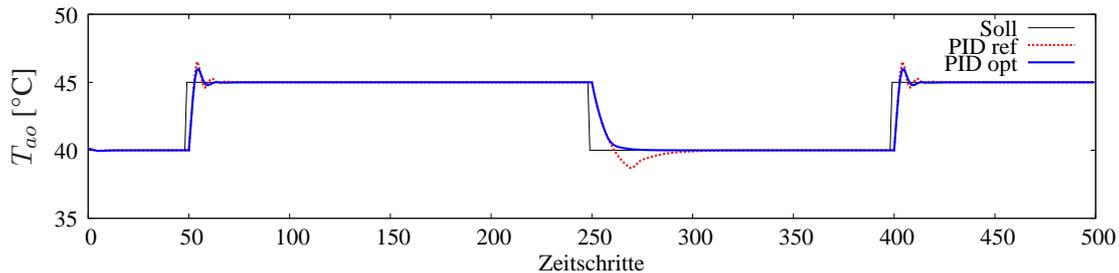


Abbildung 5.36: Die Regeltrajektorien der PI Regler.

Auf einer nichtlinearen Systemdynamik können diese jedoch nicht auf beliebige Arbeitspunkte genau eingestellt werden. In [51] wird für das obige System ein PI Regler für einen Arbeitspunkt bei einer Zieltemperatur von $45^{\circ}C$ angegeben ($K_p = 135$, $K_i = 13$). Die Ausgabe des Reglers wird auf den zulässigen Bereich der Eingriffsgröße $u \in [670, 1400]$ beschränkt. In Abbildung 5.36 ist die Anwendung des Reglers auf eine Regeltrajektorie am nichtlinearen System aufgetragen. Die erzielten Werte in dieser Arbeit stimmen mit den in [51] publizierten Werten überein. Durch die Verwendung eines PI Reglers mit 'Anti Windup' [108] auf dem Integralteil und der Suche nach optimaleren Parametern durch einen evolutionären Ansatz, konnte in dieser Arbeit ein deutlich besserer PI Regler gefunden werden ($K_p = 96.692$, $K_i = 18.198$, Regelgesetz mit 'Anti Windup' auf Integralteil). Dieser ist in Abbildung 5.36 ebenfalls dargestellt. Obwohl der optimierte PI-Regler deutlich besser ist als der in der Literatur publizierte Regler, arbeitet er nicht zufriedenstellend.

²Die Systemgleichungen sind aus [51] entnommen. Die Darstellung dort enthält jedoch zwei Fehler, diese wurden anhand von [3] korrigiert. Die Korrektur wurde durch das Reproduzieren der publizierten Ergebnisse validiert.

Tabelle 5.17: Die für das Lernen verwendeten Parameter für die selbstlernenden Regler mit NFQCA an der Heizspirale.

Zustand x	4 dimensional	f_w T_{wo} T_{ao} $\tilde{T}_{ao} = T_{ao}^d - T_{ao}$	Flussrate des Heizwassers Temperatur Heizwasserabfluss Temperatur Abluft Regelabweichung
Kostenfunktion	\tanh^2	x^d μ C	$(-, -, -, 0)$ $(0, 0, 0, 1.0)$ 0.01
Aktion u	1 dimensional NFQCA	u \mathcal{U}	Ventilöffnung [670 , 1400]
Q-Funktion	neuronal	Topologie	5-10-10-1 (600 Epochen RProp)
Strategie-Funktion	neuronal	Topologie	4-9-1 (1000 Epochen RProp)
Exploration		ϵ Δ	0.20 $\sim N(0,200)$

Tabelle 5.18: Versuchsauswertung der Regler des Heizspirale.

Regler	\bar{e}_T
PID ref	0.17972
PID opt	0.13854
NFQCA	0.12979

5.5.1 Selbstlernender Regler zur Regelung der Heizspirale

Es wird ein selbstlernender Regler mit NFQCA auf das System mit konstanten Umweltbedingungen angewandt. Der Zustandsvektor des Reglers besteht aus den internen Systemvariablen und der aktuellen Regelabweichung. Hierzu werden in einem explorativen Lernprozess Interaktionsepisoden mit 100 Zeitschritten, jeweils mit einem zufälligen Sollwert $T_{ao}^d \in [40, 45]$, durchgeführt. Die für den selbstlernenden Regler verwendeten Parameter sind in Tabelle 5.17 angegeben. Nach jeder Iteration des Lernverfahrens wird der Regler an dem in Abbildung 5.36 dargestellten Verlauf der Führungsgröße über 500 Zeitschritte getestet. Auf dieser Trajektorie wird die Kennzahl \bar{e}_T für das Gütekriterium der mittleren Regelabweichung ermittelt. Durch die Anwendung eines selbstlernenden Reglers mit NFQCA kann die Regelgüte deutlich gesteigert werden. Kretchmar betrachtet in [51] die Kombination eines PI Reglers mit einem auf Neuronalen Netzen basierenden Reinforcement Lernverfahren und berichtet von einer Verbesserung um 1.25%, im Vergleich zu seinem PI-Regler ('PID ref') durch die lernende Komponente. Im Vergleich dazu kann die mittlere absolute Regelabweichung auf der Testtrajektorie, durch den in nur 49 Iterationen gelernten NFQCA Regler um 27.78% gesteigert werden. Im Vergleich zum optimierten PI-Regler weist der mit NFQCA gelernte Regler immer noch eine Verbesserung von 6.23% auf. Die Regeltrajektorie des selbstlernenden Reglers ist in Abbildung 5.37 zu sehen. Der Regler erreicht die Solltemperatur durch das Beeinflussen des Ventils schnell und ohne das Überschwingen, das selbst der optimierte PI Regler nicht verhindern kann. Die Ergebnisse sind in Tabelle 5.18 zusammengefasst.

5.5.2 Lernen unter veränderlichen Umgebungsvariablen

In der praktischen Anwendung der Regler ist die Annahme über konstante Umgebungsbedingungen wenig realistisch. Angelehnt an Anderson et al. [3] wird nun der Fall von sich verändernden Umgebungsvariablen T_{ai} , T_{wi} und f_a betrachtet. Die externen Prozessvariablen wer-

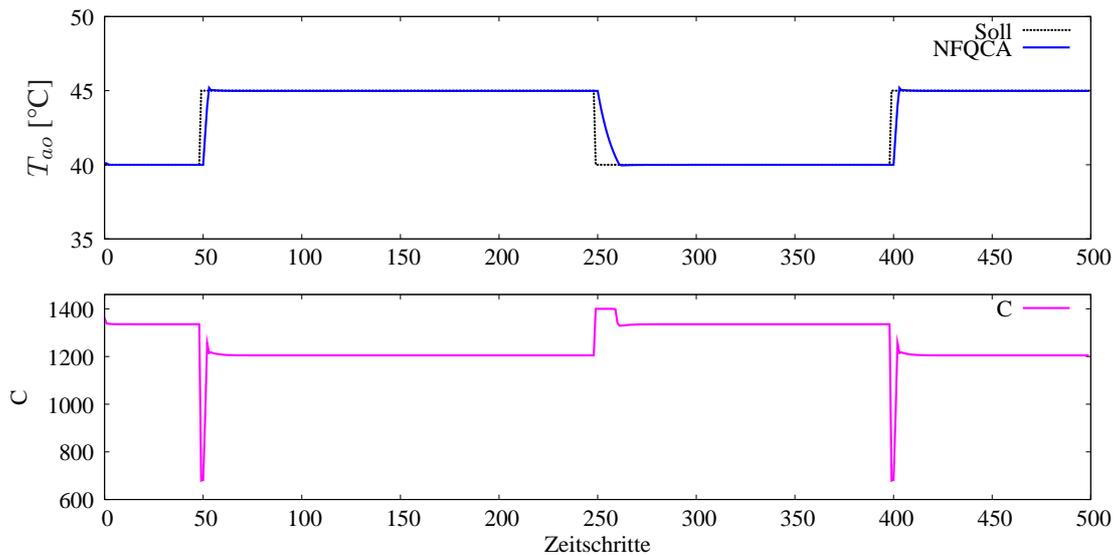


Abbildung 5.37: Die Regeltrajektorien des NFQCA Reglers. Das nichtlineare, kontinuierliche Regelgesetz, das durch das Strategienetz realisiert wird reduziert das Überschwingen deutlich und erreicht den Zielwert sehr schnell. Im unteren Teil der Grafik ist der Verlauf der vom Regler gewählten Eingriffsgröße auf der Trajektorie zu sehen.

den über eine zufällige Bewegung in einem 'Random Walk' verändert. Diese Veränderung wird alle 30 Zeitschritte durchgeführt, wobei die drei externen Variablen dabei in einem Bereich von $4 \leq T_{ai} \leq 12^\circ\text{C}$, $73 \leq T_{wi} \leq 81^\circ\text{C}$ und $0.7 \leq f_a \leq 0.9\text{kg/s}$ gehalten werden. Für die Beurteilung der Regelgüte werden die Regler auf 100 Trajektorien angewandt und die mittlere Regelabweichung \bar{e}_T über alle Trajektorien ermittelt. Der verbesserte PI-Regler zeigt dabei wieder eine bessere Güte als der in der Literatur publizierte Regler. In Abbildung 5.38 ist eine Regeltrajektorie des verbesserten PI-Reglers unter zufälliger Veränderung der Umgebungsvariablen zu sehen. Bei jeder Veränderung der Umgebungsvariablen tritt eine deutliche Regelabweichung auf, die vom Regler erst nach mehreren Zeitschritten ausgeregelt werden kann. Im Gegensatz zu den bisher betrachteten Problemen sind die externen Variablen in diesem Versuch für den Regler beobachtbar. Sie können somit direkt in den Zustandsvektor des selbstlernenden Reglers integriert werden. Dieser besitzt somit 7 Dimensionen und ergibt sich zu $x = (F_w, T_{wo}, T_{ao}, T_{wi}, T_{ai}, f_a, \tilde{T}_{ao})$. Die für den selbstlernenden Regler verwendeten Parameter sind in Tabelle 5.19 zusammengefasst. Für die Anwendung des selbstlernenden Reglers unter veränderlichen Umgebungsbedingungen wird wiederum ein explorativer Lernprozess verwendet. Jede Interaktionsepisode wird mit einem zufällig gewählten Sollwert durchgeführt. Die Umgebungsvariablen ändern sich während der Interaktion zufällig alle 30 Zeitschritte. Die Veränderung der Umgebungsbedingungen erscheint für den Regler somit als zufällige Störung während des Lernvorganges. Die Regler werden an derselben Trajektorie bewertet, wie sie bei der Untersuchung der Regler unter konstanten Umgebungsbedingungen verwendet wurde. Die Umweltbedingungen selbst ändern sich jedoch bei den Testtrajektorien ebenfalls alle 30 Zeitschritte in der beschriebenen Weise. Trotz des größeren Zustandsvektors und der Veränderung der Umgebungsbedingungen während der Interaktion erzielt der Regler bereits nach 60 Iterationen des explorativen Lernprozesses sein bestes Ergebnis. Dieses ist mit $\bar{e}_T = 0.1467$ um 17.02% besser als der optimierte PI Regler bzw. um 28.68% besser als der ursprüngliche PI-

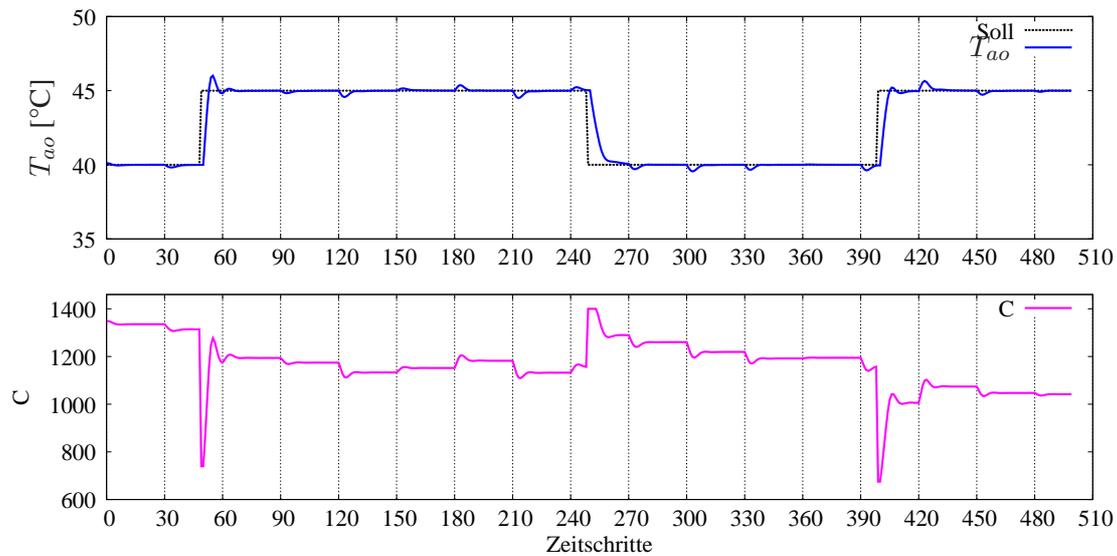


Abbildung 5.38: Die Regeltrajektorie des optimierten PI Reglers auf einer Trajektorie mit zufällig veränderten Umgebungsbedingungen.

Referenzregler auf dieser Regelungsaufgabe. Die Ergebnisse sind in Tabelle 5.20 zusammengefasst. In Abbildung 5.39 ist die Regeltrajektorie des gelernten Reglers auf dem definierten Profil dargestellt. Im Vergleich zu Abbildung 5.38 ist die deutliche Verbesserung des Regelverhaltens zu sehen. Auf eine Veränderung der Umweltbedingungen reagiert der Regler schnell und bringt die Regelgröße wieder an den Sollwert heran.

5.5.3 Diskussion

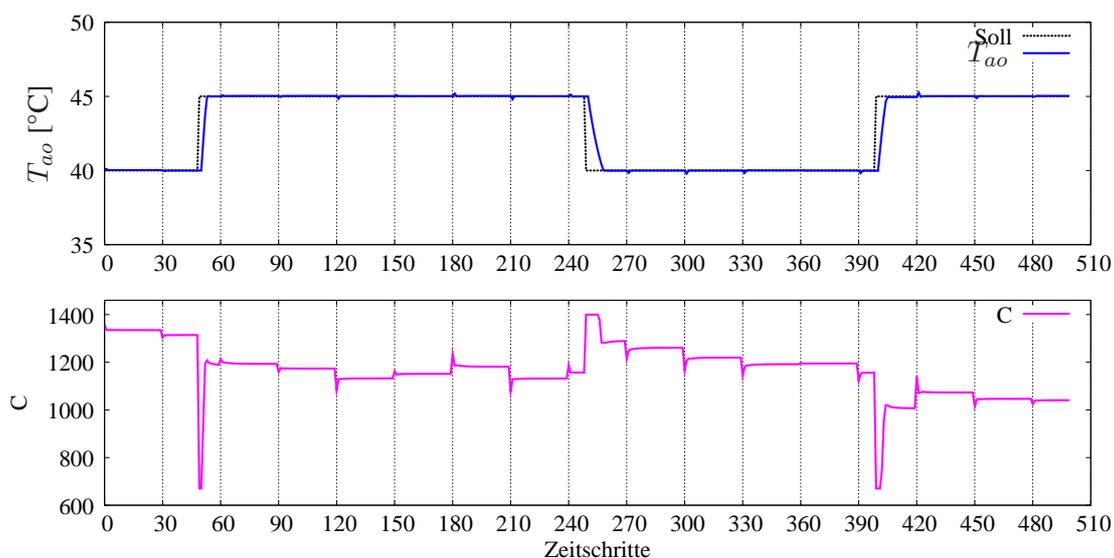
In diesem Versuch wurde ein nichtlinearer Prozess betrachtet, der von externen Umgebungsvariablen abhängig ist. Die Dynamik dieser Umgebungsvariablen kann der Regler selbst nicht beeinflussen. Daher wurde ihr Verhalten als zufälliger Verlauf über die Zeit angenommen, die für den Regler als Störung der Systemdynamik erscheinen. Im Unterschied zu anderen Lernver-

Tabelle 5.19: Die für das Lernen verwendeten Parameter für die selbstlernenden Regler mit NFQCA an der Heizspirale.

Zustand	7 dimensional	f_w	Flussrate des Heizwassers
		T_{wo}	Temperatur Heizwasserabfluss
		T_{ao}	Temperatur Abluft
		T_{ai}	Temperatur Zuluft
		T_{wi}	Temperatur Heizwasserzuluft
		f_a	Massenfluß Zuluft
		$\tilde{T}_{ao} = T_{ao}^d - T_{ao}$	Regelabweichung
Kostenfunktion	\tanh^2	x^d	(-, -, -, -, -, 0)
		μ	(0, 0, 0, 0, 0, 1.0)
		C	0.01
Aktion	1 dimensional	u	Ventilöffnung
Q-Funktion	neuronal	Topologie	8-10-10-1 (600 Epochen RProp)
Strategie-Funktion	neuronal	Topologie	4-15-1 (1000 Epochen RProp)
Exploration		ϵ	0.20
		ν	N(0,200)

Tabelle 5.20: Versuchsauswertung der Regler des Heizspirale unter zufälliger Veränderung der Umgebungsvariablen.

Regler	\bar{e}_T
PID ref	0.2057
PID opt	0.1768
NFQCA	0.1467

**Abbildung 5.39:** Die Regeltrajektorien des NFQCA Reglers auf einer Trajektorie mit zufällig veränderten Umgebungsbedingungen.

suchen wurde angenommen, dass diese externen Umgebungsbedingungen in Form von messbaren Größen dem Regler zur Verfügung stehen. In diesem Fall ist dies durch den Einsatz geeigneter Messeinrichtungen möglich. Der selbstlernende Regler mit NFQCA kann schon nach wenigen Interaktionsepisoden mit dem System eine Regelgüte erreichen, die den optimierten PI-Regler übertrifft. Ähnlich zu der Betrachtung der externen Last am DC-Motor kann der Regler die unterschiedlichen Umgebungsbedingungen im Zustandsraum unterscheiden. Wie in Abbildung 5.40 zu sehen, bleibt der Lernverlauf dabei stabil und schnell, obwohl sich die Umgebungsbedingungen während der Lerniterationen ändern.

5.6 Zusammenfassung

Im vorliegenden Kapitel wurden die Leistungsfähigkeit der in dieser Arbeit entworfenen selbstlernenden Regler an ausgewählten Problemstellungen der regelungstechnischen Praxis untersucht. Die auf dem Batchlernen basierenden Verfahren sind in der Lage, für verschiedene Problemstellungen mit nur sehr wenigen Interaktionen mit dem zu regelnden System eine sehr gute Regelgüte zu erreichen. Dabei können sie durch ihre optimierende Eigenschaft ohne Vorwissen über den Prozess mit einem sehr geringen Entwurfsaufwand auf verschiedenste Problemstellungen angewandt werden. Im Speziellen war es für die vorgestellten Problemstellungen in keinem Fall notwendig, die Parameter der Verfahren durch eine intensive Suche auf das Problem anzupassen. Die Verfahren sind in weiten Bereichen robust gegen die Variation der Parameter

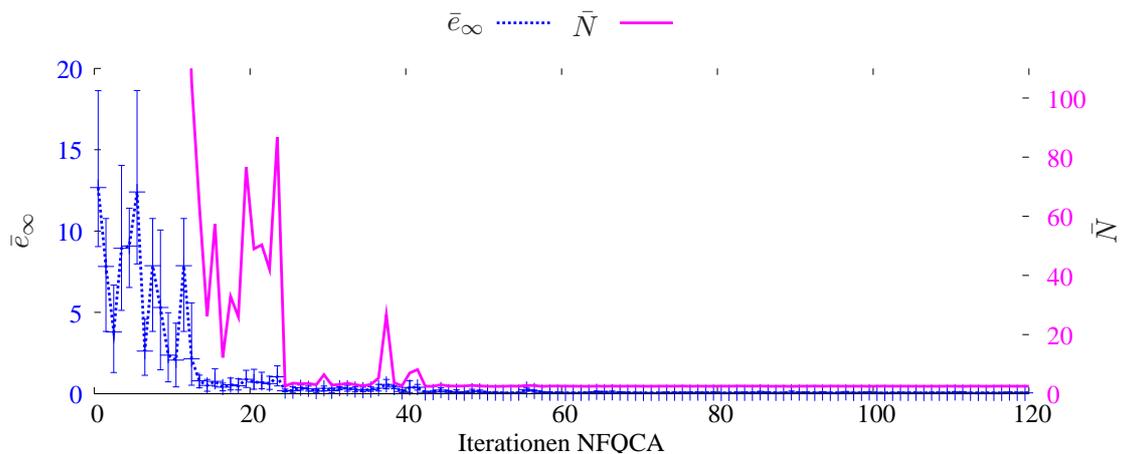


Abbildung 5.40: Der Verlauf der Gütekriterien \bar{N} und \bar{e}_∞ über die Iterationen des NFQCA-Verfahrens auf der Regelungsaufgabe der Heizspirale mit veränderlichen Umgebungsbedingungen.

wie Netztopologien, Anzahl der Epochs für das überwachte Lernen und Explorationsraten. Die Wahl der Parameter wurde in den meisten Fällen schon nach zwei bis drei Versuchen grob festgelegt und nicht weiter optimiert.

Die Verfahren sind in der Lage, die Folgeregelung für verschiedene Problemstellungen aus nur wenigen Interaktionsepisoden mit wenigen Beispielen von Führungsgrößen zu erlernen. Dabei wird während einer Interaktionsepisode die Führungsgröße möglichst lange konstant gehalten, um möglichst viele Informationen über die Systemdynamik zu erhalten. Trotzdem ist es anschließend möglich, den Regler auf Folgeregelungen mit stark variierender und kontinuierlich verlaufender Führungsgröße anzuwenden. Die Folgeregelung als solche zeigt bei der reinen Rückführung der Regelabweichung auf Profilen mit variierenden Führungsgrößen keine optimale Güte. Dies liegt daran, dass die Regler nur auf das Fehlersignal reagieren können, nachdem schon eine Regelabweichung entstanden ist. Dies ist eine Eigenschaft, die auch die klassischen Reglerkonzepte betrifft und keine spezielle Eigenschaft der selbstlernenden Regler. Es wurde gezeigt, dass es durch nachträgliche Anpassung der Zustandsinformation möglich ist, eine echte Trajektorienregelung anzunähern. Diese zeigt deutlich bessere Ergebnisse als die reine Folgeregelung. Für diese Anpassung muss der Regler nicht neu gelernt werden, vielmehr wird die bereits gelernte Information über die Systemdynamik geschickt ausgenutzt, was in den meisten Fällen ausreicht eine sehr gute Trajektorienregelung zu erzeugen.

Wie dargestellt, sind die gelernten Regler in einem gewissen Bereich robust gegen eine veränderte Systemdynamik durch externe Einflüsse, auftretendes Rauschen und Störungen. Ein sehr wichtiges Ergebnis ist, dass das Lernen der Regler direkt unter diesen Einflüssen möglich ist und die Strategie der Regler diese Einflüsse in gewissen Grenzen kompensieren kann. So wurde an einem Beispiel gezeigt, dass sich Regler unter hohem Rauschen des Prozesses und der Messeinrichtungen mit immer noch wenigen Interaktionsepisoden lernen lassen. Dies ist speziell für die praktische Anwendung der Regler auf reale Systeme ein relevantes Ergebnis, da diese immer einem gewissen Rauschen unterworfen sind. Für den Entwurf von Robusten Reglern ist besonders das Ergebnis der lastunabhängigen Regelung des DC-Motors von Interesse. Bei diesem Versuch wurde gezeigt, dass die selbstlernenden Regler bei einer geeigneten Wahl der Zustandsvariablen in der Lage sind, verschiedene Arbeitspunkte der Regelung im Zustandsraum zu unterscheiden, ohne direkt die externen Einflüsse zu beobachten. Des Weiteren sind

die lernenden Regler robust gegen das Auftreten einzelner externer Störungen während des Interaktionsprozesses. Dies kommt häufig bei der aktiven Schwingungsreduktion vor. An diesem Beispiel wurde gezeigt, dass durch die Repräsentation von einzelnen, künstlichen Anregungen während der Interaktionsepisoden ein sehr effektiv arbeitender Regler gelernt werden kann. Die Erfahrungen mit den Verfahren zeigen, dass diese auch auf realen Straßenprofile sehr gute Ergebnisse zeigen. Speziell im Fall von messbaren externen Störungen in Form von externen Prozessvariablen können die Regler diese sehr effizient kompensieren, was an der Regelung der Heizspirale deutlich wurde.

Die mit NFQCA erzielbare Regelgüte in Bezug auf die Genauigkeit ist bei allen Versuchen äußerst hoch. Dies wurde speziell an der Regelung der schwebenden Stahlkugel deutlich, bei der für zufällige Führungsgrößen eine mittlere Genauigkeit von wenigen Hundertstel Millimeter erreicht wurde. Durch die Verwendung der \tanh^2 -Kostenfunktion kombinieren die Regler dabei ein annähernd zeitoptimales Verhalten mit dieser präziser Regelung. Die diskutierten Probleme und die präsentierten Ergebnisse bestätigen die Anwendbarkeit der vorgestellten Verfahren für praktische Probleme der Regelungstechnik. Die benötigte Interaktionszeit mit den zu regelnden Systemen ist bei allen Versuchen äußerst gering. In sehr kurzer Zeit lassen sich sehr effektive Regler erstellen, die in ihrer Qualität für den industriellen Einsatz an realen Prozessen geeignet erscheinen. Erste Industrieprojekte, basierend auf dieser Arbeit, bestätigen diese Annahme.

Einbettung in den wissenschaftlichen Kontext

Im vorliegenden Kapitel werden die in dieser Arbeit vorgestellten Verfahren im Kontext der selbstlernenden Regler, dem Stand der Wissenschaft folgend, in aktuelle Forschungsarbeiten eingeordnet. Für einen Überblick in das Gebiet des Reinforcement-Lernens sei auf [47, 101] verwiesen. Einen Überblick über aktuelle Verfahren im Reinforcement-Lernen gibt [44].

6.1 Batch-Verfahren im Reinforcement-Lernen

Die Verwendung des Reinforcement-Lernens in diskreten Zustandsräumen, basierend auf tabellenbasierten Lernverfahren, schränkt die Relevanz der Verfahren für den praktischen Einsatz stark ein. Aus diesem Grund wurden Verfahren entwickelt, die durch den Einsatz von Funktionsapproximatoren auch für kontinuierliche Zustandsräume anwendbar sind [58, 100, 111].

Neuronale Netze sind als leistungsfähige Funktionsapproximatoren im Reinforcement-Lernen ein erfolgversprechender Ansatz, da sie in der Lage sind, allgemeine nichtlineare Funktionen darzustellen und gute Generalisierungseigenschaften besitzen. Es existieren viele Beispiele für den erfolgreichen Einsatz und die Vorteile von neuronalen Netzen im Reinforcement Lernen [20–22, 58, 105]. Es gibt jedoch annähernd ebensoviele Arbeiten, die von Problemen beim Einsatz von neuronalen Netzen berichten [18]. Im Gegensatz zu anderen Funktionsapproximatoren wie z.B. 'Tile-Coding'-Verfahren, zu denen der CMAC [2] gehört, muss die Struktur des Netzes nicht speziell auf die Form des Zustandsraumes oder die Systemdynamik angepasst werden. Die erzielbare Güte der Lernverfahren ist, wie in der Arbeit diskutiert, in einem akzeptablen Bereich robust gegen die verwendete Topologie der neuronalen Netze. Um die Anwendung der selbstlernenden Regler für den Benutzer weiter zu vereinfachen, existieren in der Literatur verschiedene Verfahren, die in der Lage sind, die Topologie eines Netzes selbstständig aufzubauen. Ein vielversprechendes Verfahren hierzu ist das 'Cascade Correlation'-Verfahren das Precup in [85] auf die Verwendung in Reinforcement-Lernverfahren untersucht. Durch die Abwandlung von Treadgold in [107] kann das Verfahren direkt auf das überwachte Lernen mit RProp angepasst und in NFQ und NFQCA integriert werden. Erste Ergebnisse weisen darauf hin, dass sich mit diesem Verfahren, startend von einem Q-Netz und Strategie-Netz mit nur einem Neuron in der verborgenen Schicht, eine ähnlich gute Regelungsgüte erzielen lässt, wie sie hier präsentiert wurde. Erkauft werden muss dieser Vorteil jedoch mit einer erhöhten Anzahl von benötigten Lerniterationen. Zudem neigt das Verfahren zu einem 'Overfitting'-Effekt, der mit einer festen Topologie nicht beobachtet wird.

Reinforcement-Lernverfahren wie Q-Lernen [101] und Sarsa [88] sind direkte Verfahren, um die kumulierten Kosten eines Agenten in einem sequentiellen Entscheidungsproblem zu minimieren. Sie werden als direkt bezeichnet, da sie ausschließlich auf einem Übergang, den sie in einem Zeitschritt im System beobachten, arbeiten. Somit benötigen sie eine hohe Anzahl von Interaktionen mit dem zu regelnden Prozess, um eine geeignete Regelstrategie zu lernen. In Verbindung mit einem Funktionsapproximator ergibt sich zudem eine sehr hohe Zahl an Iterationen des Lernverfahrens, um eine geeignete Wertfunktion einzulernen, was die Anzahl der Interaktionen mit dem System weiter erhöht. Die Batch-Verfahren im Reinforcement-Lernen sind explizit für die Reduktion der benötigten Interaktionen mit dem zu regelnden Prozess entworfen worden. Dies wird durch die Speicherung der beobachteten Übergänge im Agenten ermöglicht. Damit sind sie deutlich speicherintensiver und benötigen mehr Rechenzeit für eine Iteration des Lernverfahrens als die direkten Verfahren. Angesichts der Möglichkeiten heutiger Rechnersysteme ist dies, im Gegensatz zur Interaktionszeit an einem realen System, keine Einschränkung für ihren Einsatz für praktische Problemstellungen.

Die 'Batch-Verfahren' basieren auf dem 'Experience Replay' von Lin (1992) [58]. Dieser entwickelt das Verfahren, um die direkten Methoden des RL schneller und effizienter zu machen. Obwohl er ein neuronales Netz benutzt, ist in seinem Verfahren die Idee der 'Fitted'-Methoden noch nicht integriert. Die Aktualisierung der Wertfunktion wird auf der Menge der gesammelten Übergänge durchgeführt, jedoch einzeln in den Funktionsapproximator eingelesen. Trotz dieser einfachen Idee zeigt das Verfahren in aktuellen Untersuchungen von Kalyanakrishnan und Stone (2007) [48], in denen sie es mit einem aktuellen Batch-Verfahren vergleichen, sehr gute Eigenschaften hinsichtlich des asymptotischen Verhaltens des Lernverfahrens.

Die Vorteile der Verwendung der gespeicherten Übergänge für die Verbesserung des Approximators wurde 1995 von Gordon in Form eines Wertiterationsverfahrens als 'Fitted Value Iteration' publiziert [32, 34]. Diese basieren auf parametrisierten, linearen Funktionsapproximatoren, die er intensiv auf Konvergenz im Dynamischen Programmieren untersucht [33].

In den Arbeiten von Ormonitei und Sen wurden die Ideen von Gordon übernommen [65] und auf weitere Funktionsapproximatoren erweitert. Im Speziellen betrachten sie den Fall des kernelbasierten RL und formulieren das Problem der Bestimmung der Q-Funktion in eine Sequenz von kernelbasierten Regressionsproblemen um.

Das von Lagoudakis, Parr und Bartlett 2003 veröffentlichte 'Least Square Policy Iteration' (LSPI) [54], ist ein modellfreies Batch-Verfahren, bei dem die Strategie-Funktion als Linearkombination von Basisfunktionen über dem Zustandsraum dargestellt wird. Bei k solcher Funktionen berechnet der LSPI-Batch-Aktualisierungsschritt ihre Gewichte direkt durch eine lineare Operation mit einer $k \times k$ -Matrix, die, basierend auf der Datenbasis, gefüllt wird. Während die Konvergenz des LSPI-Verfahrens garantiert ist, hängt die Qualität der Lösung von der Auswahl der speziellen Basisfunktionen und der Verteilung der gespeicherten Übergänge ab. Dies sind beides kritische Punkte beim Entwurf selbstlernender Regler, die ohne Kenntnis der Systemdynamik gelernt werden.

Das 'Fitted Q-Iteration' Verfahren von Ernst et al. (2005) [27] ist ebenfalls ein Batch-Verfahren zum modellfreien Reinforcement-Lernen. Sie leiten Konvergenzkriterien des Verfahrens für eine spezielle Klasse von Funktionsapproximatoren, den kernelbasierten Verfahren, her. Für den speziellen Fall der Funktionsapproximation in einer Baumstruktur untersuchen sie das Verfahren empirisch an klassischen RL-Benchmarks, wie 'Acrobot' und 'Mountain Car'.

Das 'Neural Fitted Q-Iteration'-Verfahren von Riedmiller (2005) [79], das dieser Arbeit zugrunde liegt, kann als Spezialform des 'Fitted Q-Iteration' von Ernst et al. betrachtet werden.

Im Unterschied zu Ernst schlägt Riedmiller die Verwendung von neuronalen Netzen als Funktionsapproximator für die Q-Funktion vor. Die bei den direkten Verfahren des RL bestehenden Probleme bei der Verwendung neuronaler Netze werden durch die 'Fitted'-Idee im Batch-Lernen effizient behoben, da die neuronalen Netze mit epochenbasiertem Lernen unter Verwendung des RProp-Verfahrens gelernt werden können. Durch die Untersuchung des Verfahrens an Standardbenchmarks wird die Leistungsfähigkeit im Vergleich zu anderen bestehenden Verfahren demonstriert. In [84] und [41] wurde die Anwendung des NFQ-Verfahrens auf reale Systeme untersucht. Durch die geringe Anzahl von Interaktionsepisoden am realen System, können in kurzer Zeit sehr effiziente Strategien gelernt werden, die sich für den praktischen Einsatz eignen. Dies stellt eine zentrale Voraussetzung selbstlernende Regler für praktische Problemstellungen dar. Durch die Verwendung neuronaler Netze als Funktionsapproximator für die Q-Funktion gibt es zum NFQ-Verfahren jedoch keine Konvergenzaussagen [5]. Dennoch erweist es sich in empirischen Untersuchungen in dieser Arbeit als sehr robustes Verfahren, das für viele Problemstellungen die annähernd optimale Lösung findet.

Die Idee der 'Fitted Q-Iteration'-Verfahren wurde auch auf andere Funktionsapproximatoren angewandt. In [48] berichten Kalyanakrishnan und Stone (2007) von einer effizienten Steigerung der Lerngeschwindigkeit und von guten Ergebnissen mit Fitted Q-Iteration in einer hochdimensionalen Multiagenten-Aufgabe. Als Approximator für die Q-Funktion wird ein CMAC verwendet, dessen Parameter (Auflösung und Basisfunktionen) auf das Problem angepasst werden müssen. Ein ähnlicher Ansatz wurde auch in [106] präsentiert. Andere erfolgreiche Realisierungen der Batch-Verfahren finden sich in annähernd allen Bereichen des Reinforcement-Lernens. Beispiele sind in Gabel (2008) [30] für die Planung von Ressourcenauslastung oder in [6] für das 'Residual Gradient'-Verfahren gegeben.

6.2 Reinforcement-Lernverfahren mit kontinuierlichen Aktionen

Während viele der veröffentlichten Arbeiten die Erweiterung des Reinforcement-Lernens auf kontinuierliche Zustandsräume betrachten, wurde der Betrachtung kontinuierlicher Aktionsräume relativ wenig Beachtung geschenkt. Bestimmte Probleme mit kontinuierlichen Eingriffsgrößen können durch eine feine Diskretisierung der Aktionsvariablen mit 'Tile Coding'-Verfahren zumindest näherungsweise gelöst werden [89,92]. Für typische regelungstechnische Probleme, die einer hoch präzisen Lösung bedürfen, werden jedoch andere Herangehensweisen benötigt. Dies ist vor allem für Probleme der Fall, bei denen eine Aktion, die sich nur wenig von der optimalen Aktion unterscheidet, zu einer schlechten Güte der Strategie führt. Dies wurde in Kapitel 4 anhand des Unterwasserfahrzeuges mit NFQ untersucht. Wird ein Funktionsapproximator mit lokaler Generalisierung verwendet, haben die Verfahren zusätzlich den Nachteil, dass in einem Zustand sehr viele Aktionen ausgeführt werden müssen, um ihre Auswirkungen zu erfassen. Eine Reihe von Verfahren lösen dieses Problem durch eine Interpolation zwischen diskreten Aktionskandidaten auf der Basis ihrer Wertfunktion. Die Leistungsfähigkeit dieser Verfahren hängt sehr stark von Annahmen über die Form der Wertfunktion ab, die für stark nichtlineare Regelungsprobleme nicht immer erfüllt sind. Eines dieser Verfahren ist das von Hasselt und Wiering (2007) in [110] vorgestellte 'Continuous Actor Critic Learning Automata' (CACL) Verfahren. Ihre Ergebnisse evaluieren sie anhand eines eingliedrigen Wagen-Stab Systems am einfachen Fall des Balancierens ohne die genaue Stabilisierung in einem Zielbereich. In Millan et al. (2002) [60] wird ein weiteres dieser Verfahren am Beispiel einer Navi-

gationsaufgabe eines mobilen Roboters vorgestellt. Anhand der Navigationsaufgabe wird der Grad an benötigtem Vorwissen deutlich, den der Benutzer als Teil der Lösung in das Verfahren einbringen muss. Das 'Wire Fitt'-Verfahren von Baird und Klopff (1993) [11], das von Gaskett et al. 1999 [31] weiter angepasst wurde, versucht dieses Problem durch ein adaptives Interpolations-Schema zu lösen.

Eine weitere wichtige Eigenschaft der Verfahren bei der Verwendung kontinuierlicher Aktionen ist die effiziente Bestimmung der gierigen Strategie. Dies ist speziell für regelungstechnische Problemstellungen mit einer hohen Regelfrequenz und den damit einhergehenden Echtzeitanforderungen wichtig. Das in 4.2 vorgestellte NFQG ist ein Beispiel eines Verfahrens, das diese Anforderung nicht erfüllt. Die gradientenbasierte Suche in der Q-Funktion ermöglicht einerseits die Wahl beliebiger reellwertiger Aktionen, die Suche wird jedoch iterativ durchgeführt und benötigt hohe Rechenzeit. Groß, Stephan and Krabbes stellen (1998) in [37] ein modellfreies Lernverfahren für kontinuierliche Aktionen vor, das ähnliche Eigenschaften bei der Bestimmung der gierigen Strategie aufweist. Durch eine Vektorquantisierung mit dem 'Neural Gas'-Ansatz werden ähnliche Zustände zusammengefasst. Ein dynamisches neuronales Feld kodiert die Werte der Aktionen über diesen Zuständen, so dass die Bestimmung der gierigen Strategie die iterative Auswertung der Dynamik des neuronalen Feldes notwendig macht. Diese Verfahren werden auch als Kritiker-Verfahren bezeichnet, da sie eine Wertfunktion berechnen, die jedes Zustands-Aktions-Paar bezüglich ihrer Nützlichkeit bewertet. Hagen et al. [42] stellen (2000) ein Verfahren vor um die Auswertung der Q-Funktion mit kontinuierlichen Aktionen effizienter berechenbar zu machen. Bei diesem Verfahren, das auf einer Arbeit von Bradke [19] basiert, werden strikte Annahmen über die Form der Q-Funktion, des Approximators und der Kostenfunktion getroffen, wodurch sich die gierige Auswertung der Q-Funktion über eine lineare Operation berechnen lässt. Allerdings muss die Kostenfunktion hier eine ganz bestimmte Form haben, die für komplexere Systeme mit unbekannter Systemdynamik schwer zu finden ist.

Im Gegensatz dazu wurden reine Aktor-Verfahren entwickelt, die keine Wertfunktion berechnen, sondern ausschließlich eine parameterisierbare Familie von Strategien besitzen. Durch direktes Anwenden der Strategie auf das System kann der Gradient ihrer Regelgüte bezüglich der Parameter der Strategie berechnet werden. Die Parameter der Strategie werden dann in Richtung der Verbesserung aktualisiert. Diese Verfahren ermöglichen eine effiziente Berechnung der Strategie, da sie in Form von parameterisierten Funktionen direkt vorliegt und können direkt kontinuierliche Aktionen verwenden. Diese Strategie-Gradienten-Verfahren ('Policy Gradient') unterscheiden sich untereinander vor allem durch die Bestimmung des Gradienten. Bei der Verwendung von finiten Differenzen, z.B. in Ng (2000) [63], wird dieser Gradient durch möglichst kleine Änderungen der Strategie-Parameter und durch Interaktion mit der Strategie auf vielen Trajektorien am System bestimmt. Durch die Verwendung des REINFORCE-Verfahrens von Williams (1992) [113] kann die Anzahl der benötigten Trajektorien zur Berechnung eines Strategie-Gradienten reduziert werden [70]. Ein Nachteil der Verfahren ist, dass bei Änderung der Strategie jeweils ein neuer Gradient, unabhängig von vorigen Schätzungen, bestimmt werden muss. In diesem Sinne wird ein großer Teil der Information, die über das System gesammelt wird, nicht zum weiteren Lernen verwendet. Desweiteren sind die verwendeten parametrisierten Strategien sehr stark auf das betrachtete Problem zugeschnitten und die Verfahren benötigen oft initial eine funktionierende Strategie die das Problem näherungsweise löst. Für den Entwurf selbstlernender Regler muss hier ein hoher Grad an Vorwissen verwendet werden. Ein Vorteil der reinen Aktor-Verfahren ist, dass sie direkt eine annähernd optimale Strategie suchen, wobei reine Kritiker-Verfahren die optimale Wertfunktion annähern, deren

gierige Auswertung nicht zwingend auch die Annäherung der optimalen Strategie ist.

Die Aktor-Kritiker-Verfahren werden eingesetzt, um die Stärken der reinen Aktor- und reinen Kritiker-Verfahren zu kombinieren. Die Kernidee der Aktor-Kritiker-Verfahren [12, 50] ist die Repräsentation der Strategie in einem unabhängigen Funktionsapproximator im Aktor, die von der Repräsentation der Wertfunktion im Kritiker unabhängig ist. Die Parameter der Strategie im Aktor werden aufgrund der Information im Kritiker in einem Verbesserungsschritt angepasst. Durch diese explizite Repräsentation der Strategie kann die Aktionswahl auch für kontinuierliche Aktionen effizient durchgeführt werden. Die Repräsentation der Strategie wird durch parametrisierbare Funktionen, z.B. durch neuronale Netze [110], Gaußsche Verfahren [49] oder sequentielle 'Monte Carlo'-Verfahren [57] durchgeführt. In [69] führen Peters et al. (2005) das 'Natural Actor-Critic'-Verfahren ein, das von Sutton et. al (2007) in [17] erweitert wird. Das Verfahren ermöglicht die Verwendung der Aktor-Kritiker-Architektur für die Bestimmung des Strategie-Gradienten in 'Policy Gradient'-Verfahren durch Berechnung einer expliziten Wertfunktion. Im Gegensatz zu NFQCA wird in beiden Verfahren ein linearer Funktionsapproximator vorausgesetzt, der durch die Verwendung von Basisfunktionen auf nichtlineare Zusammenhänge des zu regelnden Systems angepasst werden muss. Das NFQCA vereint die Vorteile der 'Policy Gradient'-Verfahren mit dem modellfreien Lernen von reinen Kritiker-Systemen in Form einer Aktor-Kritiker-Architektur. Die Anpassung der Gewichte der neuronalen Strategie basiert auf dem berechneten Gradienten in der Q-Funktion und ist somit im weiteren Sinne ein 'Policy Gradient'-Verfahren. Die aktuelle Schätzung der Pfadkosten wird jedoch in Form einer Q-Funktion repräsentiert und nicht wie bei den 'Policy Gradient'-Verfahren durch eine rein zustandsbasierte Wertfunktion. Dadurch wird die modellfreie Aktualisierung der Wertfunktion in jedem Zustand, basierend auf den Übergängen der Datenbasis, möglich. Diese Übergänge können unter beliebigen und unterschiedlichen Strategien gesammelt worden sein. Im Gegensatz dazu müssen die 'Policy Gradient'-Verfahren die Kostenschätzung für einen Zustand durch aufwändige Verfahren aus Übergängen, basierend auf der aktuellen Strategie, aktualisieren. Die zusätzlich repräsentierte Strategie in NFQCA approximiert die gierige Auswertung der aktuellen Q-Funktion, die über den Zustandsraum generalisiert. Das neuronale Strategie-Netz bestimmt die Strategie, unter welcher der Agent mit dem System interagiert und geht durch die Aktualisierung der Q-Funktion in die iterative Aktualisierung der Q-Funktion ein. Somit wird ebenfalls vordringlich die optimale Strategie angenähert und nicht die optimale Wertfunktion. Dies erzeugt ein sehr stabiles und schnelles Lernverfahren, das nur von wenigen Parametern abhängt und rechtfertigt die redundante Speicherung der Strategie-Information in der Q-Funktion und im Strategie-Netz.

Doya geht (2000) in [24] noch einen Schritt weiter und betrachtet neben kontinuierlichen Zustands- und Aktionsräumen ebenfalls eine in der Zeit kontinuierliche Regelung. Für die Modellierung der Probleme werden die Hamilton- und Jacobimatrix des Systems betrachtet und die Systemgleichungen als gegeben vorausgesetzt. Somit wird, im Gegensatz zu den in dieser Arbeit entwickelten Verfahren, ein Entwurfsaufwand für den Benutzer erzeugt, der nahe an den des klassischen Reglerentwurfs kommt.

In [5] untersuchen Munos et al. (2007) die theoretischen Eigenschaften von gefitteten Aktor-Kritiker-Verfahren in kontinuierlichen Aktions- und Zustandsräumen. Ein für diese Arbeit interessantes Ergebnis aus ihren Betrachtungen ist die Feststellung, dass es in der Praxis sehr schwer ist, ein Lernverfahren auf Basis dieses Ansatzes zu realisieren. Sie vermuten, dass ein Strategie-Suchschritt, wie er in Aktor-Kritikersystemen verwendet wird, eine Möglichkeit darstellt, ein funktionierendes Verfahren zu erzeugen. Mit der vorliegenden Arbeit wurde parallel zu den Arbeiten von Munos et al., mit NFQCA ein solches Verfahren entworfen und seine Leis-

tungsfähigkeit empirisch an praktischen Problemstellungen der Regelungstechnik untersucht.

In [26, 73] wird die Verwendung von Gaußschen Prozessen (GP) im Reinforcement-Lernen vorgeschlagen. Dies wird im aktuellen wissenschaftlichen Kontext als mögliche interessante Alternative zur Verwendung neuronaler Netze betrachtet. Durch die spezielle Form der Approximation der Q-Funktion kann das Minimum der Q-Funktion für kontinuierliche Aktionen berechnet werden. Eine Anwendung der Gaußschen Prozesse mit kontinuierlichen Aktionen wird von Rottmann et al. (2007) in [87] an der Höhenregelung eines Modellluftschiffes vorgestellt. In [23] wird die Verwendung von Gaußschen Prozessen an einem Motorpendel mit NFQ verglichen. In ihrem Verfahren verwenden sie die 'Hint to Goal'-Heuristik von Riedmiller und kombinieren diese mit einer zu der in 3.4 ähnlichen Kostenfunktion. In ihrem Vergleich zeigt sich, dass das NFQ-Verfahren sehr effizient lernt und nahe an das für den Versuch bestimmbare theoretische Optimum herankommt, während der mit GP gelernte Regler hinter den erzielten Ergebnissen zurückbleibt.

6.2.1 NFQCA als 'Adaptive Critic Design'

Die Verfahren des 'Adaptive Critic Designs (ACD)' [71, 72] umfassen verschiedene Ansätze, die sich basierend auf der Einordnung des 'Approximate Dynamic Programming (ADP)' von Werbos ergeben [93]. Unter diese fallen das 'Heuristic Dynamic Programming (HDP)', wie auch andere abgewandelte Verfahren wie das 'Dual Heuristic Programming (DHP)' und 'Globalized Dual Heuristic Programming (GDHP)'. Während die HDP-Verfahren die Kostenfunktion $J(t)$ approximieren, wird in den DHP-Verfahren direkt die partielle Ableitung der Kostenfunktion nach der Eingabe bzw. in den GDHP-Verfahren beides parallel approximiert. Diese Kostenfunktion als Realisierung eines Kritikers hat in der aktionsabhängigen ('Actions Dependent (AD)') Form der Verfahren ebenfalls die aktuelle Ausgabe des Aktors als Eingabe, womit sich das klassische Q-Lernen unter diese Verfahren einordnen lässt. In dieser Design-Hierarchie wird mit dem 'Action Dependent Heuristic Dynamic Programming (ADHDP)' [71, 72, 94] ein sehr interessantes Verfahren beschrieben, das für die vorliegende Arbeit und deren Einordnung in den wissenschaftlichen Kontext eine hohe Relevanz besitzt. Das ADHDP-Verfahren benötigt weder für das Lernen des Kritikers noch für das Lernen des Aktors ein Prozessmodell. Als Repräsentation des Kritikers wird ein neuronales Netz verwendet, das als Eingabe den aktuellen Zustand des MDPs und die vom Kritiker verwendete Aktion in diesem Zustand besitzt. Das Ziel des Kritikers ist die Minimierung des Fehlermaßes $\sum_t (J(x_t, u_t) - \gamma J(x_{t+1}, u_{t+1}) - c_t)^2$ über die Zeit. Der Kritiker wird als neuronales Netz $\mathcal{J}(x, u, w_c)$ mit Gewichten w_c approximiert. Eine typische Lernregel für die Einstellung der Gewichte mit Gradientenabstieg durch Backpropagation bei Observierung eines Überganges am MDP ergibt sich dann zu $\Delta w_c = -\eta (J(x_t, u_t) - \gamma J(x_{t+1}, u_{t+1})) \frac{\partial J(x_t, u_t)}{\partial w_c}$ mit einer geeigneten Lernrate η . Der Aktor ist ebenfalls ein neuronales Netz $\mathcal{U}(x, w_u)$, dessen Ausgang direkt als Eingang des Kritikers dient. Für die Einstellung der Gewichte w_u des Aktors wird $\frac{\partial J}{\partial J} = I$ durch das gesamte Netzwerk zurück propagiert um den Gradient $\frac{\partial J}{\partial u}$ bzw. $\frac{\partial J}{\partial w_u}$ zu berechnen, mit dem sich schließlich die Gewichte des Aktors einlernen lassen.

Das in dieser Arbeit vorgestellte NFQCA-Verfahren lässt sich somit direkt als Batch- bzw. Fitted-Erweiterung des ADHDP-Verfahrens einordnen. Diese Erweiterung wird in der vorliegenden Arbeit auf Basis des klassischen Q-Lernens und dessen Fitted-Version mit neuronalen Netzen, dem NFQ-Verfahren, motiviert.

Als direktes Verfahren lernt das ADHDP während der Ausführung der aktuellen Strategie, repräsentiert durch das Aktor-Netz, am zu regelnden Prozess. Die Information, die für die An-

passung des Aktors und des Kritikers zur Verfügung steht, beschränkt sich somit auf einen beobachteten Übergang am MDP, der unter der aktuellen Strategie gesammelt wurde. Diese direkte Anwendung des Lernverfahrens (in der Literatur als 'online'- oder 'on policy'-Verfahren beschrieben) erlaubt die Adaption des Aktors auf Änderungen der Prozessdynamik während des Lernvorganges [97,98]. Die direkte Anwendung des ADHDP-Verfahrens bedingt jedoch eine schlechte Konvergenz und Lerngeschwindigkeit [104]. Dies wird vor allem durch die bereits in Kapitel 2 diskutierten Nachteile der direkten Verfahren mit globalen Funktionsapproximatoren bedingt [35]. Des Weiteren ist das Verfahren sehr anfällig für die Einstellung der Lernraten des Kritiker- und Aktor-Netzes.

Mit NFQCA wurde ein Verfahren entwickelt, das durch die Idee der Fitted- bzw. Batch-Verfahren mit nur wenigen Interaktions-Episoden einen annähernd optimalen Regler lernt und ohne schwierig einzustellende Lernparameter auskommt. Die verwendeten neuronalen Netze zur Approximation des Kritikers und des Aktors, sowie die verwendeten Gradienten für die Anpassung der Gewichte, entsprechen dem Vorgehen des ADHDP-Verfahrens. Das Einlernen der Netz-Gewichte basiert jedoch auf einem epochenbasierten Lernprozess unter Benutzung eines leistungsfähigen überwachten Lernverfahrens für neuronale Netze. Die Herleitung über das Q-Lernen betont die Eigenschaft des NFQCA-Verfahrens den Kritiker auf den gesamten für den Prozess interessanten Arbeitsbereich der Regelung einzulernen. Im Gegensatz dazu kann bei ADHDP die Anpassung des Kritikers auf die optimalen Pfadkosten nur in einem sehr kleinen Bereich um die aktuelle Strategie erwartet werden. Dies macht die durch das NFQCA-Verfahren erzeugte Regel-Strategie robuster gegen externe Störungen während ihrer Anwendung. Insbesondere erlaubt das NFQCA-Verfahren die Verwendung sehr hoher Explorationsraten, die diese Robustheit der gelernten Regler weiter erhöht.

In der praktischen Anwendung benötigt das ADHDP-Verfahren eine Diskontierung der Kosten ($0 < \gamma < 1$) und eine für den betrachteten Prozess speziell entworfene Kostenfunktion. Die Wahl des Diskontierungs-Faktors und der Kostenfunktion beeinflusst den Lernprozess stark. Beide müssen sorgfältig auf die betrachtete Regelungsaufgabe abgestimmt werden [71, 93]. Durch den Entwurf des NFQCA-Verfahrens, mit den in dieser Arbeit vorgestellten Verfahren zur Korrektur des systematischen Fehlers in der Q-Funktion und der präzisen zeitoptimalen Regelung, werden diese Beschränkungen aufgehoben. Es ist somit möglich, ohne Diskontierung, mit einer nicht auf die Prozessdynamik abgestimmten Kostenfunktion und unter Verwendung nur weniger Interaktionsepisoden einen annähernd zeitoptimalen Regler einzulernen.

6.3 Reinforcement-Lernverfahren in regelungstechnischen Anwendungen

Obwohl in der Literatur zum Reinforcement-Lernen oft der Begriff der kontinuierlichen Regelung benutzt wird (z.B. [66]), beschränken sich die Beispiele zumeist auf einfache Aufgabenstellungen mit diskreten Aktionen ohne externen Sollwert. An die selbstlernenden Regler in dieser Arbeit hingegen wurden bestimmte Anforderungen gestellt, die in der praktischen Anwendung essentiell sind. Dies sind:

- Die Regelung ist ein Problemtyp ohne festen Endzeitpunkt, es existiert somit kein Terminalzustand, in dem der Regler seine Arbeit einfach einstellen kann.
- Sie hat die Stabilisierung des geschlossenen dynamischen Systems zum Ziel, bei welcher der Toleranzbereich und die Genauigkeit der bleibenden Regelabweichung in einem für

die Praxis akzeptablen Bereich liegt.

- Sie stellt eine intelligente Reglerkomponente des Regelkreises dar, welche den zu regelnden Prozess, basierend auf einer externen Führungsgröße, geeignet beeinflusst.
- Die selbstlernenden Regler sind eine geschlossene Lösung, für die nur wenige Parameter zu bestimmen sind und deren Struktur nicht an das Regelungsproblem angepasst werden muss. Sie sind für allgemeine, insbesondere nichtlineare, dynamische Systeme geeignet und benötigen kein spezielles Wissen über die zugrunde liegende Systemdynamik.

Zusätzlich besitzen die in der Praxis betrachteten Prozesse meist kontinuierliche Eingriffsgrößen, die für eine genaue und optimale Regelung des Prozesses geeignet ausgenutzt werden müssen. In dieser Arbeit wurden mehrere regelungstechnische Probleme vorgestellt, die diese Eigenschaften besitzen und an denen die vorgestellten Verfahren zur Erzeugung selbstlernender Regler untersucht wurden.

In der Literatur des Reinforcement-Lernens erfüllen wenige der betrachteten Problemstellungen diese Anforderungen. Die meistbenutzten Standardbenchmarks für dynamische Systeme im RL werden im Folgenden bezüglich dieser Anforderungen diskutiert. Ein oft betrachtetes Problem ist der von Moore (1991) eingeführte 'Mountain Car'-Versuch [62]. Bei dieser Problemstellung muss ein Wagen mit begrenzter Kraft auf einen Hügel gesteuert werden, wobei die Kraft nicht ausreicht, um ihn direkt bergauf zu steuern. Die optimale Strategie besteht aus einer Sequenz von maximalen Aktionen, die den Wagen zuerst auf einen gegenüberliegenden Hügel steuert und anschließend auf den Berg befördert. Auf der einen Seite ist der Versuch für explorative Lernversuche interessant, auf der anderen Seite erfüllt er keine der obigen Anforderungen, die in dieser Arbeit an einen regelungstechnischen Versuch gestellt werden. Das Regelungsziel ist das Überschreiten einer bestimmten Position auf dem Berg, unabhängig von der Geschwindigkeit, worauf ein Terminalzustand erreicht wird und die Regelung beendet ist. Trotzdem wird diese Problemstellung in vielen Arbeiten als typisches Regelungsproblem bezeichnet. So stellt Pachowicz (2003) in [66] einen Vergleich mehrerer RL-Verfahren für die Regelung vor, die er an diesem Benchmark evaluiert.

Ein anderer, oft verwendeter Versuchsaufbau für die Evaluation von RL-Verfahren in regelungstechnischen Anwendungen, ist das Wagen-Stab-Problem, wie es von Lagoudakis et. al. in [55] verwendet wird. Ein Wagen kann dabei auf einer Linearführung bewegt werden. Auf dem Wagen ist mit einem passiven Gelenk ein Stab befestigt, dessen Winkel gemessen werden kann. Die Aufgabe ist das Balancieren dieses Stabes durch die Anwendung einer Kraft auf den Wagen. Dieses Problem besitzt keinen Terminalzustand im Zielbereich und kann somit nicht mehr als SKP-Problem formuliert werden. In den meisten Arbeiten wird jedoch ein Regler gelernt, der das Umfallen der Stäbe verhindert, was nicht der gewünschten Stabilisierung an einem vorgegebenen Punkt im Zustandsraum entspricht [4, 94]. Eine Verfeinerung des Regelungsproblems, das die echte Stabilisierung des geschlossenen Systems in einem kleinen Toleranzbereich, definiert auf die Wagenposition und den Winkel des Stabs, betrachtet, sind in der Literatur selten [74, 78, 79]. Selbst bei dieser Verfeinerung wird immer noch ein Regler betrachtet, der nur eine Festwertregelung auf einen einzigen Sollwert durchführt.

Ein vom Wagen-Stab-Problem abgeleiteter Versuchsaufbau ist das Motor-Pendel, bei dem ein Stab direkt an einem Motor befestigt ist [23, 67, 68]. Ähnlich wie beim 'Mountain Car', reicht die Motorkraft zumeist nicht aus, um den Stab direkt aufzuschwingen. Die Aufgabe wird zumeist als Aufschwingen des Stabes mit anschließender Stabilisierung seines Winkels um seine aufrechte Position betrachtet. Es wird dabei wiederum nur eine Festwertregelung mit einem

einzigsten Zielwert betrachtet. Wie in [23] deutlich wird, reichen jedoch eine Sequenz von maximalen Aktionen aus, um das Pendel aufzuschwingen und in einem Bereich um die aufrechte Position des Stabes zu stabilisieren.

Ein für die Regelungstechnik interessantes Benchmarkproblem wird in [97] zur Evaluation mehrerer RL-Verfahren verwendet. Angelehnt an den 'Mountain Car'-Versuch muss eine Masse durch die Anwendung einer Kraft auf einer Schräge positioniert werden. Zusätzlich werden die Verfahren auf ihr Verhalten bei einem Wechsel der Zielposition untersucht. Dieser Wechsel der Zielposition wird jedoch direkt durch einen Wechsel der Kostenfunktion und somit der Änderung des gesamten Lernproblems durchgeführt. Die gelernten Regler müssen sich auf neue Zielwerte durch Lernen einstellen und können somit eine Regelabweichung durch eine schnelle Sollwertveränderung erst nach mehreren Iterationen des Lernverfahrens ausgleichen.

Im Unterschied zu diesen Standardbenchmarks sind die verwendeten Versuche in dieser Arbeit an praxisnahen Problemstellungen orientiert. Neben der zeitoptimalen, genauen Regelung unter externer Sollwertvorgabe und verschiedenen Störungen wurde dabei auch auf die Modellierung der Aktoren der Systeme Wert gelegt. Ihre Dynamik spielt für praktische Problemstellungen eine wichtige Rolle, da bei diesen keine instantane Änderung einer virtuellen Kraft auf einen Körper möglich ist.

Die Anwendung der neuronalen Netze in regelungstechnischen Problemstellungen ist in der praktischen Anwendung zum Stand der Technik geworden [46, 61, 75]. Einen aktuellen Überblick der verwendeten Methoden und Vorgehensweisen gibt Sarangapani in [90]. Die neuronalen Netze dienen dabei als Module für komplexe Reglerstrukturen, die als Vorfilter, Vorsteuerung oder zur expliziten Behandlung von unmodellierten Nichtlinearitäten in der Prozessdynamik dienen. Auf diese Weise können praktische Problemstellungen aus der Regelungstechnik mit kontinuierlichen Eingriffsgrößen unter externen Führungsgrößen gelöst werden. Ein Beispiel in Kombination mit Reinforcement-Lernen wird von Schoknecht und Riedmiller in [91] an der Regelung eines Verbrennungsmotors gegeben. Diese Verfahren benötigen jedoch typischerweise einen gewissen Entwurfsaufwand und spezielle Reglerstrukturen, die unter Verwendung von explizitem Prozesswissen in die Reglersynthese eingehen.

Perkins und Barto [67, 68] integrieren mit den Lyapunov-Funktionen [36] eine generelle Entwurfsmethode der nichtlinearen Regler in das Reinforcement-Lernen. Das Reinforcement-Lernen dient ausschließlich dazu, in jedem Zeitschritt einen aus mehreren durch die Lyapunov-Kandidaten vorgegebenen Regler auszuwählen. Auf diese Weise können mehrere suboptimale Lyapunov-Kandidaten zu einem annähernd optimalen Regelverhalten kombiniert werden. Der Vorteil ist die sehr genaue Regelung des Prozesses, der durch die Lyapunov-Kandidaten möglich wird. Der Nachteil besteht darin, dass diese durch eine Reglersynthese vorgegeben werden müssen, die eine Analyse des Prozesses notwendig macht. Ihren Ansatz evaluieren sie an einem Motor-Pendel. Trotz des hohen Grades an Vorwissen werden mehr als 1000 Interaktionsepisoden benötigt, um eine annähernd optimale Strategie zu lernen. Dies ist erstaunlich, da in einem ähnlichen Lernversuch mit NFQ ohne dieses Vorwissen schon mit weniger als 100 Interaktionsepisoden eine annähernd optimale Strategie gelernt werden konnte [23]. Die Kombination des Reinforcement-Lernens mit klassischen Reglern wird in mehreren Arbeiten vorgestellt. Kretchmar et al. publizieren 2001 [52, 53] einen Ansatz, der die robuste Regelung mit Verfahren des Reinforcement-Lernens betrachtet. Sie verwenden ein Reinforcement-Lernverfahren mit neuronalem Aktor-Kritikernetz und kombinieren dies mit einem robusten, klassisch erzeugten LTI Regler. Dabei kann das Systemverhalten, das durch die Linearisierung des Systems vernachlässigt wurde, durch die nichtlinearen und adaptiven Eigenschaften des neuronalen Regleranteils kompensiert werden. Den Fokus ihrer Arbeit legen sie dabei auf die

Stabilität des geschlossenen dynamischen Systems, die durch den klassisch erzeugten Regler vorgegeben wird. Der Lernprozess der neuronalen Netze wird dabei geeignet eingeschränkt, um diese Stabilität beizubehalten. Der Entwurf des klassischen Reglers macht jedoch wiederum die Analyse des Prozesses sowie eine klassische Reglersynthese notwendig. Die Betrachtung von explizit robusten Reglern mittels Reinforcement-Lernen bezieht sich hierbei auf die Robustheit gegen Modellierungsfehler, die beim klassischen Reglerentwurf wichtig ist. Diese betrachtet auch Doya (2001) in [25]. Unter Benutzung eines modellbasierten Ansatzes werden in seinem Verfahren die Modellierungsfehler und Störungen in den Lernvorgang einbezogen. Dies macht wiederum die Analyse des Prozesses und die Abschätzung der Modellierungsfehler notwendig.

Interessante Problemstellungen für die Anwendung des Reinforcement-Lernens auf regelungstechnische Probleme ergeben sich auch bei Prozessen mit unbekanntem Wirkungszusammenhang von hochdimensionalen Zustandsvariablen, Eingriffsgrößen und komplexen Optimierungskriterien. Erfolgreiche Beispiele sind hier die von Bagnell und Schneider (2001) [7] vorgestellte Steuerung eines Modellhelikopters in verschiedenen Fluglagen, die von Ng et al. (2004) [64] und Abbeel et al. (2007) [1] auf Kunstflug-Manöver erweitert wird, oder die von Groß (2000) [96] vorgestellte Regelung eines industriellen Verbrennungsprozesses. In [70] zeigen Peters und Schaal (2005) eine beeindruckende Demonstration eines Roboterarms, der mit Hilfe von 'Policy Gradient'-Verfahren lernt, ausgehend von einer vom Benutzer demonstrierten Trajektorie, einen Baseball zu treffen. Weitere Anwendungen des RL für reale Problemstellungen kommen aus den verschiedensten Bereichen. So werden in [59] und [99] Studien zur optimierten Behandlung von Patienten oder in [39] eine Pfadplanung direkt im Zustands-Aktionsraum eines omnidirektionalen Roboters gelernt.

6.4 Zusammenfassung

Das Reinforcement-Lernen ist von seiner Struktur her eine natürliche Formulierung der optimierenden Regelung dynamischer Prozesse [102]. Die Regelung von dynamischen Systemen wird in der Literatur als weiter Begriff von verschiedenen Problemstellungen behandelt. So wird er sowohl für Benchmarkprobleme wie dem 'Mountain Car' als auch für reine Ablaufsteuerungen mit mächtigen Aktionen verwendet, wie sie Bernstein et al. in [14] für die Ablaufsteuerung in einem mobilen Roboter vorstellt. Die in dieser Arbeit betrachtete Problemstellung der selbstlernenden Regler für technische Prozesse mit externer Führungsgröße wird in der RL-Literatur selten betrachtet. Beispiele sind die von Riedmiller in [75] und [91] oder von Kretchmar in [53] veröffentlichten Arbeiten. Im Unterschied zu den hier betrachteten Verfahren wird in diesen Arbeiten jedoch die Struktur der Regler grundsätzlich auf die Regelungsaufgabe angepasst und zusätzliches Prozesswissen verwendet.

Die dateneffizienten Batch-Verfahren sind im RL eine leistungsfähige Weiterentwicklung, die vor allem auf reine Kritiker-Verfahren angewendet wurden. Im NFQ-Verfahren werden dabei die Probleme des neuronalen Funktionsapproximators, die bei direkten Verfahren bestehen, effizient behoben. Hierdurch kann die Leistungsfähigkeit der neuronalen Netze für komplexe, nichtlineare Systeme besser ausgenutzt werden. Durch das entworfene NFQCA in Form eines Aktor-Kritiker-Verfahrens können die Vorteile der allgemeinen Funktionsrepräsentation der neuronalen Netze auf ein Verfahren mit kontinuierlichen Aktionen übertragen werden. Da das NFQCA dadurch nicht auf Linearkombinationen von Basisfunktionen basiert, entfallen kritische Parameter für den Entwurf. Durch die Verwendung einer neuronalen Q-Funktion können

dabei alle gesammelten Übergänge des Systems effizient ausgenutzt werden, um die Wertfunktion zu aktualisieren.

Zusammenfassung und Ausblick

7.1 Überblick

Der zentrale Fokus der vorliegenden Arbeit liegt auf der Untersuchung einer intelligenten Reglerkomponente für den Einsatz in geschlossenen Regelkreisen. Bei dem hier betrachteten Vorgang der Regelung wird der Istwert der Regelgröße einer Regelstrecke erfasst und mit einem Sollwert (bzw. einer Führungsgröße) verglichen. Weichen beide voneinander ab, z.B. durch Einwirkung einer Störgröße oder durch schnelle Veränderung des Sollwertes, dann muss die Reglerkomponente geeignete Eingriffsgrößen für den Prozess erzeugen, um den Istwert an den Sollwert heranzuführen. Die Intelligenz der Reglerkomponente besteht in ihrer Fähigkeit, aus vorangegangenen Regelungsvorgängen gesammelte Erfahrung über den zu regelnden Prozess geeignet auszunutzen, um selbstständig eine geeignete Regelstrategie zu erzeugen. Die Regelstrategie basiert dabei auf einem Optimierungsprozess, für den vom Benutzer ausschließlich das Regelungsziel vorgegeben wird und der eine zeitoptimale und präzise Regelung des Prozesses erlaubt. Da dieser Optimierungsprozess in Form eines sequentiellen Entscheidungsproblems, auf der Basis der Marovschen Entscheidungsprozesse modelliert wird, beobachtet der Regler in jedem Zeitschritt nicht nur die Regelgröße, sondern den gesamten Zustandsvektor des dynamischen Systems, das den zu regelnden Prozess beschreibt. Darüber hinaus ist wenig Wissen über den Prozess selbst ausreichend, um einen solchen selbstlernenden Regler für ein gegebenes regelungstechnisches Problem anzuwenden. Explizit ist die Analyse der Systemgleichungen des dynamischen Systems und die Schätzung der physikalischen Parameter des Prozesses nicht erforderlich. Dies eliminiert eine große Fehlerquelle, die der klassischen Synthese von Regelgesetzen zugrunde liegt und reduziert das vom Benutzer benötigte Expertenwissen um den zu regelnden Prozess und seinen Aufwand für den Entwurf drastisch. Für die Praxistauglichkeit muss die Anzahl der benötigten Regelungsvorgänge für das Erlernen des Reglers bzw. die dafür aufgewendete Zeit am betrachteten Prozess möglichst klein sein, um keine langen Standzeiten des Prozesses zu verursachen. Hierzu eignen sich die Batch-Verfahren des Reinforcement-Lernens, zu denen das in dieser Arbeit verwendete NFQ-Verfahren zählt.

In Kapitel 2 sind die Grundlagen des Reinforcement-Lernens und deren dateneffiziente Weiterentwicklung auf kontinuierliche Zustandsräume mit diskreten Aktionen dargestellt. Das NFQ-Verfahren als modellfreies Lernverfahren basiert dabei auf einer neuronalen Approximation der Q-Funktion. Durch die Kombination des Batch-Lernens mit neuronalen Netzen ist es in der Lage für viele komplexe Problemstellungen mit nur wenigen Interaktionsepisoden, ei-

ne annähernd optimale Strategie zu lernen. Das NFQ wird dabei in Form eines explorativen Lernprozesses angewendet, bei dem das Verfahren ohne Wissen über den zu regelnden Prozess beginnt. Durch Interaktion mit dem System unter der aktuellen Strategie wird Erfahrung gesammelt, die in Form von Übergängen einer Datenbasis hinzugefügt wird. Nach jeder dieser Interaktionsepisoden wird eine Iteration des Lernverfahrens ausgeführt, die eine verbesserte Schätzung der Q-Funktion und somit der Strategie ergibt. Diese Schritte werden abwechselnd ausgeführt, um eine annähernd optimale Strategie zu lernen. Wie diskutiert wurde, ist das Lernverfahren dabei sehr robust gegen seine Parameter und benötigt keine aufwändige Suche nach geeigneten Parametern, um einen erfolgreichen Lernprozess zu implementieren.

Basierend auf diesen Grundlagen wird in Kapitel 3 die Verwendung des NFQ-Verfahrens für die Erstellung selbstlernender Regler unter der Zielsetzung der Arbeit untersucht. An der einfacheren Problemstellung der Festwertregelung mit einem festen Sollwert wird der Problemtyp der Regelung in NFQ modelliert. Dabei wird das auftretende Problem der Degenerierung der Q-Funktion am Beispiel der Geschwindigkeitsregelung eines DC-Motors diskutiert und mit der 'Hint to Goal'-Heuristik eine bestehende Methode vorgestellt um dieses zu lösen. Diese benötigt jedoch für den hier betrachteten Fall der Regelung auf einen Sollwert ein fundiertes Wissen über die Systemdynamik und der zu lernenden Strategie in bestimmten Punkten im Zustandsraum. Das neu entwickelte Verfahren zur Korrektur des systematischen Fehlers in der Q-Funktion (Q_{min} -Verfahren) benötigt dieses Vorwissen nicht und ist damit auf allgemeinere Systeme anwendbar. Zusätzlich konnte in den empirischen Versuchen am DC-Motor gezeigt werden, dass der Lernverlauf mit dem entwickelten Verfahren deutlich stabiler ist, ohne einen Mehraufwand in Form von zusätzlichen Interaktionen mit dem Prozess zu erzeugen. Das entwickelte Verfahren ist, im Gegensatz zur bestehenden Vorgehensweise, vollständig parameterlos. Dies ist ein weiterer Vorteil, welcher der Zielsetzung der Arbeit entspricht.

Die erreichbare Genauigkeit ist für den praktischen Einsatz der selbstlernenden Regler eine zentrale Voraussetzung. Mit der vorgestellten zeitoptimalen Regelung unter der Bereichsdefinition existiert ein Verfahren, das ausschließlich auf der Regelabweichung im Zustandsvektor arbeitet und das ansonsten kein Wissen aus der Systemdynamik verwendet. Somit ist es als Optimierungskriterium einfach für beliebige Systeme zu spezifizieren. Die daraus resultierende Kostenfunktion ist jedoch nicht dazu geeignet, eine präzise Regelung des Prozesses zu gewährleisten. Durch die Weiterentwicklung zur verfeinerten \tanh^2 -Kostenfunktion, werden die Vorteile der Bereichsdefinition mit einer präzisen Regelung vereint. Diese ist eine essentielle Voraussetzung für die genaue Regelung und erzeugt zusätzlich eine für die Approximation mit neuronalen Netzen geeignetere Form der Q-Funktion. Wie in den Untersuchungen an der Geschwindigkeitsregelung des DC-Motors gezeigt wurde, ergibt sich eine drastische Verbesserung der Genauigkeit der Regelung. Zusätzlich werden weniger Iterationen des Lernverfahrens benötigt, um die optimale Strategie für das Problem zu lernen, was die Vorteile für die Approximation durch die verfeinerte Kostenfunktion zeigt. Mit dem Q_{min} -Verfahren und der \tanh^2 -Kostenfunktion sind die selbstlernenden Regler in der Lage, eine präzise und zeitoptimale Regelung zu lernen, ohne dass spezielles Wissen über die Systemdynamik vorliegt.

Durch die Integration der Regelabweichung in der Modellierung des MDP wird das vollständige Problem der Regelung unter externer Führungsgröße möglich. Wie am Beispiel des DC-Motors gezeigt wurde, ist der selbstlernende Regler basierend auf dieser Modellierung in der Lage, die Geschwindigkeit des Motors auf beliebige Sollwerte zu regeln. Hierzu werden nur wenige Iterationen des Lernverfahrens und somit auch nur wenige Interaktionsepisoden am System mehr benötigt als bei der Festwertregelung. Die Genauigkeit der Regelung ist auch bei der Anwendung auf eine Regeltrajektorie mit variierender Führungsgröße zufriedenstellend.

Ein grundsätzliches Problem in der Anwendung der selbstlernenden Regler ist die Verwendung diskreter Aktionen in NFQ, welche die bei der Regelung benötigten kontinuierlichen Eingriffsgrößen des Prozesses nur annähern können. Dieses Problem kann mit den vorgestellten dynamischen Ausgabeelementen für praktische Anwendungen handhabbar gemacht werden. Durch die Erhöhung der erreichbaren Eingriffsgrößen für den Prozess kann die Güte der Regelung deutlich gesteigert werden. Die Weiterentwicklung der DAE zu intelligenten DAEs durch die Integration klassischer Regelgesetze ist eine Möglichkeit die bleibende Regelabweichung, welche die vorgestellten Verfahren aufgrund ihrer numerischen Ungenauigkeiten grundsätzlich besitzen, ganz zu eliminieren. Die Verwendung der intelligenten DAE-Strukturen bedingt jedoch wiederum einen Entwurfsaufwand über der Vorwissen über die Prozessdynamik voraussetzt.

Die Auswirkungen diskreter Aktionen in regelungstechnischen Prozessen mit kontinuierlichen Eingriffsgrößen wird in Kapitel 4 genauer betrachtet und die Leistungsfähigkeit der diskreten Aktionen am Beispiel der Regelung eines Prozesses mit stark nichtlinearer Dynamik untersucht. Eine reine Erhöhung der Anzahl der verwendeten diskreten Aktionen führt nicht zum erwünschten Ergebnis und verlangsamt den Lernprozess. Bereits die Verwendung einer einfachen gradientenbasierten Such-Strategie, die mit dem NFQG-Verfahren vorgestellt wurde, erzeugt auf dem untersuchten Problem sehr gute Ergebnisse mit kontinuierlichen Aktionen. Der Nachteil des Verfahrens ist jedoch eine sehr hohe Rechenzeit, die besonders bei der Auswertung der Q-Funktion für die Erzeugung der Strategie für reale Systeme nicht adäquat ist. Eine vorgestellte Möglichkeit zur Behebung dieses Problems ist die explizite Repräsentation der Strategie als neuronales Netz, wie sie im NFQGN-Verfahren vorgestellt wurde. Durch die Erweiterung des NFQ-Verfahrens auf kontinuierliche Aktionen, mit dem NFQCA-Verfahren, wird ein robustes und einfach zu handhabendes Verfahren entworfen, das in Form eines Aktor-Kritiker-Ansatzes eine Regelstrategie mit kontinuierlichen Aktionen lernbar macht. Die empirischen Untersuchungen im Vergleich zu NFQ mit diskreten Aktionen zeigen, dass sich eine annähernd optimale, kontinuierliche Strategie berechnen lässt und sich dadurch die benötigten Iterationen des Verfahrens nicht erhöhen. Diese Strategie ist sowohl zeitoptimal als auch präzise und übertrifft die mit NFQ erzielbaren Ergebnisse sowohl im Kriterium der Zeitoptimalität, als auch im Kriterium der Genauigkeit deutlich. Dabei werden keine zusätzlichen Informationen über den Prozess benötigt und keine Strukturänderungen des Reglers wie z.B. bei den DAE notwendig. Zusätzlich entfällt die Modellierungsentscheidung, welche die Auswahl der verwendeten diskreten Aktionen in NFQ bedingt. Das NFQCA behält durch die Verwendung einer Q-Funktion dabei die Eigenschaft bei, mit einem rein datenbasierten Lernprozess auf einer beliebigen Datenmenge arbeiten zu können. Diese kann unter beliebigen Strategien gesammelt worden sein, wobei nun auch kontinuierlich arbeitende Regelgesetze als Datenquelle in Betracht kommen. So ist es nun möglich mit einem suboptimalen, klassischen Regelgesetz mit zufälliger Exploration, Übergänge an einem Prozess zu sammeln um anschließend mit dem NFQCA-Verfahren, basierend auf diesen Daten, einen optimalen Regler zu lernen. Im Unterschied dazu muss bei NFQ ein Regelgesetz mit diskreten Aktionen entworfen werden, das sich weitaus schwieriger entwickeln lässt.

Die generelle Leistungsfähigkeit der entwickelten selbstlernenden Regler zeigt sich in Kapitel 5 an ihrer Anwendung auf praktische Problemstellungen der Regelungstechnik. Speziell das NFQCA-Verfahren unter Verwendung des Q_{min} -Verfahrens und der \tanh^2 -Kostenfunktion erfüllt die Zielsetzung der Arbeit vollständig. Wie in den Versuchen gezeigt, reichen wenige Minuten Interaktionszeit an einem Prozess, um komplexe und nichtlineare Regelungsvorgänge annähernd optimal zu lösen. Wie die Einordnung in den wissenschaftlichen Kontext in Kapitel

6 zeigt, wird das hier betrachtete Problem der präzisen Regelung unter Vorgabe einer externen Führungsgröße in einem geschlossenen Ansatz ohne die Verwendung zusätzlicher Reglerkomponenten im Reinforcement-Lernen selten betrachtet.

7.1.1 Entwurfsaufwand selbstlernender Regler

Wie beschrieben, ist die Reduzierung des vom Benutzer benötigten Expertenwissens für den Einsatz der selbstlernenden Regler eine zentrale Zielsetzung dieser Arbeit. Ein gewisses Maß an Vorwissen ist auch bei den selbstlernenden Reglern notwendig. Für die Anwendung der selbstlernenden Regler muss diesem ein Zustandsvektor zur Verfügung gestellt werden, der die Modellierung des Regelungsproblems als MDP zulässt. In diesem Zustandsvektor muss sich das Optimierungskriterium über die Kostenfunktion berechnen lassen. Wie dargestellt, wird dies für typische regelungstechnische Problemstellungen durch die Integration der Regelabweichung evtl. in mehreren Variablen erreicht. Die Prozessdynamik an sich muss durch diesen Zustandsvektor ausreichend repräsentierbar sein, was bedeutet, dass sich die Dynamik des betrachteten Systems grundsätzlich mit einer Funktion auf diesem Zustandsvektor berechnen lässt. Diese Funktion selbst muss nicht bekannt sein und der Zustandsvektor muss keine minimale Beschreibung des Prozesses darstellen. Insbesondere kann er redundante oder irrelevante Systemvariablen beinhalten. Eine solche Repräsentation kann für die meisten Systeme ohne fundiertes Vorwissen gefunden werden. Das Regelungsziel selbst kann durch die Verwendung der \tanh^2 -Kostenfunktion oder die Bereichsdefinition direkt auf dem Zustandsraum und nur in den Variablen der Regelabweichung über den gewünschten Toleranzbereich der Regelung definiert werden. Es wird somit kein weiteres Wissen über den Prozess benötigt. Durch die Verwendung des Q_{min} -Verfahrens zur Korrektur des systematischen Fehlers in der Q-Funktion, wird das Lernverfahren ermöglicht, ohne weitere Information in den Prozess integrieren zu müssen. Bei Verwendung der diskreten Aktionen mit NFQ müssen die verwendeten diskreten Aktionen definiert werden mit denen sich das System erwartungsgemäß beeinflussen lässt. Für Systeme, die sich mit einem Bang-Bang-Verhalten regeln lassen, sind diese Aktionen meist bekannt. Im Allgemeinen kann diese Auswahl die erzielbare Regelgüte jedoch stark beeinflussen. Durch die Verwendung des NFQCA müssen dem Verfahren ausschließlich die physikalisch gegebenen Beschränkungen der Eingriffsgrößen mitgeteilt werden, die typischerweise bekannt sind.

Die weiteren Parameter des Lernverfahrens sind, wie in der Arbeit dargestellt, in weiten Bereichen robust gegen ihre Wahl. Zu wählen sind vor allem die Topologie der neuronalen Netze und die Anzahl der Epochen mit denen sie in jeder Iteration eingelernt werden. Dies kann eine grobe Suche nach geeigneten Parametern notwendig machen, die jedoch nicht exakt eingestellt werden müssen. Im Gesamten sind mehrere Verfahren auf einer Meta-Ebene denkbar, die diese Parameterwahl für einen unerfahrenen Benutzer handhabbar machen.

7.2 Ausblick

7.2.1 MDPs mit externen Aktionen

In Abschnitt 5.1.2 und 5.3 wurde jeweils ein Beispiel für einen Prozess gegeben, dessen Dynamik von externen Prozessvariablen abhängt, die jedoch selbst weder vom Regler noch von der Prozessdynamik beeinflusst werden. In beiden Fällen sind die externen Systemvariablen nicht beobachtbar und werden als Rauschen bzw. als Systemstörungen aufgefasst, gegen die

der lernende Regler robust ist. Für viele regelungstechnische Problemstellungen der Verfahrenstechnik existieren jedoch Prozesse, die verschiedenen messbaren, externen Einflüssen unterliegen. Ein solches Beispiel wurde mit der Regelung der Heizspirale in Abschnitt 5.5 gegeben, bei der diese messbaren Einflüsse als Zustände des selbstlernenden Reglers modelliert wurden. Eine interessante Frage ist nun, wie diese externen, messbaren Prozessvariablen in den Lernprozess eingebunden werden können, ohne ihre Veränderung als Rauschen der Systemdynamik zu betrachten. Eine vielversprechende Alternative ist es, diese nicht als Zustandsvariablen, sondern als vom Regler nicht veränderbare Aktionen in den MDP zu integrieren. Die externen, messbaren Prozessvariablen werden hierzu in einem Vektor α zusammengefasst. In jedem Zeitschritt erhält der Agent den Zustand des Systems x_t , bestehend aus den internen Prozessvariablen bzw. der Regelabweichung und den Messwerten der externen Prozessvariablen α_t . Basierend auf diesen Werten berechnet er durch seine Strategie $\Pi(x_t, \alpha_t, w_\pi)$ eine Eingriffsgröße u_t . In NFQCA werden die Übergangsdaten nun in der Form $(x_t, \alpha_t, u_t, x_{t+1})$ gespeichert. Die Lernregel für die Q-Funktion ergibt sich angepasst zu $Q(x_t, \alpha_t, u_t) = c(x_t, u_t, x_{t+1}) + \min_u Q(x_{t+1}, \alpha_t, u)$. Für die Aktualisierung des Aktors ist die partielle Ableitung der Q-Funktion nach der Aktion u an der Stelle (x, α, u) nun maßgebend. Mit diesem partiellen Gradienten können dann wieder die Gewichte des Strategienetzes w_π angepasst werden.

In weiteren Arbeiten wird die Leistungsfähigkeit dieser Modellierung auf verschiedene Probleme untersucht. Von Interesse ist vor allem die Generalisierungsleistung der Strategie für unbekannte Umgebungssituationen, die sich in einem weiten Bereich sehr dynamisch ändert. Die so erstellten Regler sind für verschiedene technische Anwendungen von großem Interesse, da sich die Arbeitspunkte der Regelung, basierend auf externen Messungen, besser repräsentieren lassen.

7.2.2 Erweiterung der Lernverfahren

Weitere interessante Ansätze für weiterführende Arbeiten ergeben sich in der Erweiterung des Verfahrens auf Abwandlungen des Q-Lernens. Zwei der vielversprechendsten Ansätze im Kontext der selbstlernenden Regler sind hierbei das 'Advantage Learning' und das 'Residual Gradient'-Verfahren.

Neural Fitted Advantage Learning mit Kontinuierlichen Aktionen (NFACA)

Ein in der Literatur oft beschriebenes Problem beim Q-Lernen ist, dass speziell bei sehr kleinen Zeitschritten Δt des MDPs der Q-Wert zweier Aktionen in einem Zustand sich nur sehr gering unterscheidet. Dies ist insbesondere in Verbindung mit der Benutzung von Funktionsapproximatoren ein Problem, da der zu lernende Unterschied der Q-Werte von dem numerischen Approximationsfehler überdeckt wird. Das 'Advantage Updating'-Verfahren [9, 10] ist ein Reinforcement-Lernverfahren, bei dem zwei verschiedene Arten von Information gespeichert werden. Für jeden Zustand x wird die Wertfunktion $J(x)$ gespeichert, die die erwarteten Pfadkosten, beginnend im Zustand x bei Anwendung der optimalen Strategie repräsentiert. Desweiteren wird für jeden Zustand x und jede Aktion u ein 'Advantage'-Wert $A(x, u)$ gespeichert. Dieser repräsentiert eine Schätzung, zu welchem Grad die Wahl der Aktion u in diesem Zustand die erwarteten kumulierten Pfadkosten gegenüber der optimalen Aktion erhöht. Das 'Advantage Learning' ist eine Weiterentwicklung des 'Advantage Updating' Verfahrens. Im Unterschied zu diesem werden nicht mehr zwei getrennte Funktionen gespeichert, sondern

nur noch eine 'Advantage'-Funktion $A(x, u)$. Die Lernregel, abgeleitet aus der Bellmann Gleichung, für das Advantage-Lernen lautet:

$$\begin{aligned}
 A^{k+1}(x_p, u_p) &= (1 - \alpha)A^k(x_p, u_p) \\
 &+ \alpha \left[\frac{1}{h\Delta_t} \left(c_p + \gamma^{\Delta_t} \min_u A^k(x'_p, u) \right) \right. \\
 &\left. + \left(1 - \frac{1}{h\Delta_t} \right) \min_u A^k(x_p, u) \right] \quad (7.1)
 \end{aligned}$$

In der gezeigten Form ist das Verfahren eine Erweiterung des Q-Lernens. Im Grenzfall bei $h\Delta_t = 1$ ergibt sich das normale Q-Update. Für $h\Delta_t < 1$ ist die gelernte 'Advantage'-Funktion eine skalierte Version der Q-Funktion. Intuitiv wird in einem Zustand der Q-Wert für suboptimale Aktionen durch den Faktor $h\Delta_t$ skaliert und somit der Abstand zwischen optimalen und suboptimalen Aktionen in der gelernten Funktion vergrößert.

Das 'Advantage Learning' lässt sich in das NFQ- und NFQCA-Verfahren sehr einfach integrieren. Das neuronale Q-Netz wird ersetzt durch ein neuronales Advantage-Netz, das faktisch die gleichen Eingänge und Topologie aufweist. Für die Aktualisierung der Advantage Werte wird die in 7.1 aufgeführte Lernregel angewandt. Die Aktualisierung der Strategie in NFQCA kann ohne Anpassungen auf der neuronalen 'Advantage'-Funktion durchgeführt werden.

Residuelles NFQCA

Die residuelle Form von Reinforcement Lernverfahren wurde in [8] von Baird vorgestellt. Die Verfahren reduzieren direkt den Bellmann-Fehler in der Wertfunktion und können für verschiedene Reinforcement-Lernverfahren hergeleitet werden. Ihr Vorteil ist, dass sie bei der Aktualisierung der Wertfunktion den Funktionswert nicht nur an einem Zustand verändern, sondern direkt versuchen den Funktionswert am Folgezustand mit anzupassen. Dadurch ergeben sich in der Praxis für viele Lernverfahren eine bessere Approximation der Wertfunktion und eine implizite Glättung der Funktionswerte. Der zu minimierende Bellmann(-äquivalente)-Fehler ergibt sich in der residuellen Form des NFQCA für einen Übergang p für die aktuelle Q-Funktion und Strategie zu:

$$E_B(Q(w_q), \Pi(w_\pi), p) = \frac{1}{2} (c_p + Q(x'_p, \Pi(x'_p, w_\pi), w_q) - Q(x_p, u_p, w_q))^2 \quad (7.2)$$

Somit ergibt sich die partielle Ableitung des Bellmann-Fehlers bezüglich der Gewichte der aktuellen Q-Funktion zu:

$$\begin{aligned}
 \frac{\partial}{\partial w_q} E_B &= (c_p + Q(x'_p, \Pi(x'_p, w_\pi), w_q) - Q(x_p, u_p, w_q)) \\
 &\quad \left(\frac{\partial}{\partial w_q} Q(x'_p, \Pi(x'_p, w_\pi), w_q) - \frac{\partial}{\partial w_q} Q(x_p, u_p, w_q) \right) \quad (7.3)
 \end{aligned}$$

Dieser Gradient kann als zusätzliche Glättung beim Lernen der Q-Funktion benutzt werden.

Bei Verwendung von Aktor- und Kritikernetz in NFQCA kann auch für die Lernregel des Aktors eine residuelle Form hergeleitet werden. Hierzu wird der Bellmann Fehler um eine zusätzliche Komponente erweitert.

$$\begin{aligned}
 E_{BExt}(Q(w_q), \Pi(w_\pi), p) &= \frac{1}{2} (c_p + Q(x'_p, \Pi(x'_p, w_\pi), w_q) - Q(x_p, u_p, w_q))^2 \\
 &+ Q(x'_p, \Pi(x'_p, w_\pi), w_q) \quad (7.4)
 \end{aligned}$$

Diese Erweiterung des zu minimierenden Fehlers garantiert die Anpassung der Strategie an die Nebenbedingung $\Pi(x, w_\pi) = \underset{u}{\operatorname{argmin}} Q(x, u, w_q)$. Diese Anpassung ändert an der Bestimmung des Gradienten in 7.3 nichts. Zur Herleitung einer Lernregel für das Strategienetz wird die partielle Ableitung des Fehlers bezüglich den Gewichten der Strategie gebildet:

$$\begin{aligned}
\frac{\partial}{\partial w_\pi} E_{BEExt} &= (c_p + Q(x'_p, \Pi(x'_p, w_\pi), w_q) - Q(x_p, u_p, w_q)) \\
&= \frac{\partial}{\partial w_\pi} Q(x'_p, \Pi(x'_p, w_\pi), w_q) + \frac{\partial}{\partial w_\pi} Q(x'_p, \Pi(x'_p, w_\pi), w_q) \\
&= (1 + c_p + Q(x'_p, \Pi(x'_p, w_\pi), w_q) - Q(x_p, u_p, w_q)) \\
&\quad \frac{\partial Q(x'_p, \Pi(x'_p, w_\pi), w_q)}{\partial \Pi(x'_p, w_\pi)} \frac{\partial \Pi(x'_p, w_\pi)}{\partial w_\pi}
\end{aligned} \tag{7.5}$$

Mit den Methoden des residuellen Reinforcement-Lernens kann nun ein auf NFQCA basierendes residuelles Verfahren entwickelt werden. Durch die Verwendung in einem Batch-Verfahren kann die Gewichtung des Bellmann-Fehlers und des Approximationsfehlers gegeneinander mit einem zu RProp ähnlichen Vorgehen durchgeführt werden [6].

Ein Problem beim reinen Advantage-Lernen ist, dass sich bei einer schlechten Schätzung der aktuellen Strategie und der aktuellen Advantage-Funktion durch die Skalierung eine sehr zerklüftete Wertfunktion herausbildet. Daher wird das beschriebene Advantage-Learning in der Literatur häufig mit einem residuellen Gradienten kombiniert [43]. Für beide Erweiterungen des NFQCA liegen erste Ergebnisse vor. Ein Problem ist hierbei die zusätzliche Einführung von mehreren Lernparametern für die Verfahren. Die bisherigen Ergebnisse zeigen, dass diese zusätzlichen Parameter schwer einzustellen sind und eine ausgiebige Suche nach guten Parametern erfordern. In zukünftigen Arbeiten werden diese anhand der in dieser Arbeit vorgestellten Ergebnisse evaluiert.

7.2.3 Erweiterung des Lernprozesses

Der in der Arbeit benutzte explorative Lernprozess, kombiniert mit einem Batch-Verfahren des RL, stellt eine leistungsfähige Möglichkeit dar, ohne Vorwissen über den Prozess eine annähernd optimale Strategie zu lernen. Die Speicherung der Übergänge im Agenten ist dabei ein speicherintensiver Vorgang, der durch die vorhandenen Möglichkeiten moderner Rechner keine Einschränkung für die Verfahren darstellt. Die Rechenzeit für eine Iteration des NFQ- bzw. NFQCA-Verfahrens steigt jedoch linear mit der Anzahl der gespeicherten Übergänge in der Datenbasis. Durch den explorativen Lernprozess werden bestimmte Bereiche im Zustandsraum sehr häufig besucht. Dies ist zum Beispiel für Zustände nahe eines oft gewählten Startzustandes zu erwarten. Zur Reduzierung der Rechenzeit können geeignete Verfahren gefunden werden, um die Verteilung der in der Datenbasis vorhandenen Übergänge im Zustandsraum zu schätzen. Basierend auf dieser Verteilungsschätzung kann dann eine Auswahl der Übergänge in der Datenbasis durchgeführt werden, die möglichst kleiner ist als die gesamte Datenbasis, aber annähernd gleichen Informationsgehalt besitzt. Mit dem gleichen Argument ergeben sich in der Datenbasis unterrepräsentierte Bereiche des Zustandsraumes. Werden diese durch eine Verteilungsschätzung identifiziert, so kann die Explorationsstrategie angepasst werden, um diese Bereiche zu erkunden. Dieses Vorgehen wird motiviert durch das Paradigma des aktiven Lernens. In diesem Fall entscheidet das Verfahren selbst, aus welchen Bereichen des Zustandsraumes neue Daten gesammelt werden und welche der gesammelten Daten für das Lernen ver-

wendet werden sollen. Erste Versuche wurden basierend auf 'Parzen Windows' und einfachen Histogrammen durchgeführt. Es zeigt sich, dass sich in vielen Fällen die Explorationsstrategie deutlich verbessern lässt, wenn einfache Explorationsschemata scheitern. Die Histogramme bedürfen allerdings der manuellen Einstellung ihrer Diskretisierung des Zustandsraumes und führen so schwierig einzustellende Parameter ein. Das Parzen-Window-Verfahren ist hier einfacher einzustellen, benötigt jedoch eine zu hohe Rechenzeit. Die Anwendung von weiterentwickelten Verfahren zur Dichteschätzung der Daten oder unüberwacht lernende Verfahren bieten hier interessante Alternativen, die eine weitere Untersuchung als lohnenswert erscheinen lassen.

7.3 Fazit

Die vorgestellten Verfahren erzeugen zeitoptimale, präzise Regler aus einfacher Interaktion mit dem System. Speziell mit dem NFQCA steht ein Verfahren zur Verfügung, das für den Einsatz in praktischen Problemstellungen der Regelungstechnik qualitativ hochwertige Regelgesetze mit kontinuierlicher Ausgabe erzeugt. In der praktischen Anwendung in der Automatisierungs- und Prozesstechnik erlaubt dies die Anpassung von Komponenten auf neue Aufgabenstellungen und Umgebungsbedingungen in kürzester Zeit. Diese Anpassungen sind auch von unerfahrenen Benutzern durchzuführen, die kein Expertenwissen über die veränderte Systemdynamik besitzen. Im Speziellen entfällt der Aufwand für die Analyse der Systemgleichungen, die Identifizierung der physikalischen Parameter und die Reglersynthese der klassischen Regelungstechnik. Durch die Repräsentation der Regelstrategie in einem neuronalen Netz kann im Gegensatz zum klassischen Reglerentwurf die für den Prozess und die gegebenen Optimierungskriterien optimale Regelstrategie gefunden werden. So können auch die Ressourcen von Prozessen mit sehr komplexen Zusammenhängen durch eine optimale Regelstrategie ausgenutzt werden, die vom Benutzer in einem klassischen Reglerentwurf nicht vorherzusehen ist. In diesem Sinn lassen sich die Vorteile neu entwickelter Aktuatoren, Materialien oder Sensoren in einem Prozess voll ausschöpfen, ohne dass der Stand der erreichten Analysetechniken des klassischen Reglerentwurfs die verwendete Lösung beeinträchtigt.

Die Vision unserer Arbeitsgruppe ist eine vollständig autonom lernenden Regeleinrichtung für allgemeine technische Prozesse. Diese stellt eine Recheneinrichtung mit mehreren Ausgabesignalen für Aktuatoren sowie Eingabesignalen für Sensoren und Führungsgrößen dar, die sich mit beliebigen Prozessen verbinden lässt. Durch einfache Vorgabe eines Regelungszieles und Interaktion mit dem Prozess lernt diese Regeleinrichtung selbstständig die optimale Regelung des angeschlossenen Prozesses. Dies würde in der praktischen Anwendung viele schwierig einzustellende und hoch spezialisiert einzusetzende Regeleinrichtungen ersetzen, die hohe Umrüstzeiten und wenig Flexibilität bieten. Aus Sicht des Autors wird diese Vision schon in wenigen Jahren erreichbar sein. Die vorliegende Arbeit ist hierzu ein wichtiger Baustein.

Obwohl sich die Arbeit auf die Anwendung der selbstlernenden Regler auf technische Prozesse beschränkt, sind die vorgestellten Verfahren in vielen Bereichen des Reinforcement-Lernens anwendbar. Dies gilt insbesondere für die kontinuierlichen Aktionen, die in vielen Anwendungsbereichen benötigt werden. Im Sinne des autonomen Lernens autonomer Systeme, stellen sie einen Baustein dar, der zur Erzeugung komplexer Verhalten, z.B. für autonome mobile Roboter, dienen kann.

Literaturverzeichnis

- [1] P. Abbeel, A. Coates, M. Quigley und A. Y. Ng. *An application of reinforcement learning to aerobatic helicopter flight*. In *In Advances in Neural Information Processing Systems 19*, S. 2007. MIT Press (2007).
- [2] J. S. Albus. *Brains, Behavior, and Robotics*. BYTE Books, Peterborough (1981).
- [3] C. W. Anderson, D. Hittle, A. Katz und R. M. Kretchmar. *Synthesis of Reinforcement Learning, Neural Networks, and PI Control Applied to a Simulated Heating Coil*. *Journal of Artificial Intelligence in Engineering*, 11(4): 423–431 (1997).
- [4] C. W. Anderson und Z. Hong. *Reinforcement learning with modular neural networks for control*. In *In IEEE International Workshop on Neural Networks Application to Control and Image Processing*, S. 90–93 (1994).
- [5] A. Antos, R. Munos und C. Szepesvári. *Fitted Q-iteration in continuous action-space MDPs*. In J. Platt, D. Koller, Y. Singer und S. Roweis, Hg., *Advances in Neural Information Processing Systems 20*, S. 9–16. MIT Press, Cambridge, MA (2008).
- [6] A. Antos, C. Szepesvári und R. Munos. *Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path*. In *In COLT-19*, S. 574–588. Springer-Verlag (2006).
- [7] J. A. Bagnell und J. G. Schneider. *Autonomous helicopter control using reinforcement learning policy search methods*. In *In International Conference on Robotics and Automation*, S. 1615–1620. IEEE Press (2001).
- [8] L. Baird. *Residual algorithms: Reinforcement learning with function approximation*. In *In Proceedings of the Twelfth International Conference on Machine Learning*, S. 30–37. Morgan Kaufmann (1995).
- [9] L. C. I. Baird. *Advantage Updating*. Techn. Ber. WL-TR-93-1146, Wright-Patterson Air Force Base Ohio: Wright Laboratory (1993). Available from the Defense Technical Information Center, Cameron Station, Alexandria, VA 22304-6145).
- [10] L. C. I. Baird. *Reinforcement Learning in Continuous Time: Advantage Updating*. In *In Proceedings of the International Conference on Neural Networks, Orlando, FL, June (1994)*.

- [11] L. C. I. Baird und A. H. Klopff. *Reinforcement Learning with High-Dimensional, Continuous Actions*. Techn. Ber. WL-TR-93-1147, Wright Laboratory, Wright-Patterson Air Force Base (1993).
- [12] A. G. Barto, R. S. Sutton und C. W. Anderson. *Neuron like adaptive elements that can solve difficult learning problems*. IEEE Transactions on Systems, Man and Cybernetics, 13: 834–847 (1983).
- [13] R. Bellman. *Dynamic Programming*. Princeton Univ Press (1957). ISBN 069107951X.
- [14] D. S. Bernstein und S. Zilberstein. *Reinforcement Learning for Weakly Coupled MDPs and an Application to Planetary Rover Control*. In *In UAI* (2001).
- [15] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific (1995). ISBN 1886529132.
- [16] D. P. Bertsekas. *Tsitsiklis. Neuro-dynamic programming: an overview*. In *In Proceedings of the 34th IEEE Conference on Decision and Control*, S. 560–564 (1995).
- [17] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh und M. Lee. *Incremental Natural Actor-Critic Algorithms*. In *Twenty First Annual Conference on Neural Information Processing Systems*, S. 105–112. Vancouver, Canada (2007).
- [18] J. A. Boyan und A. W. Moore. *Generalization in reinforcement learning: Safely approximating the value function*. In *Advances in Neural Information Processing Systems 7*, S. 369–376. MIT Press (1995).
- [19] S. J. Bradtke. *Reinforcement learning applied to linear quadratic regulation*. In *In Advances in Neural Information Processing Systems 5*, S. 295–302. Morgan Kaufmann (1993).
- [20] S. Buck, M. Beetz und T. Schmitt. *Approximating the Value Function for Continuous Space Reinforcement Learning in Robot Control*. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2002*, S. 1062–1067 (2002).
- [21] R. Coulom. *Feedforward neural networks in reinforcement learning applied to high-dimensional motor control*. In *In 13th International Conference on Algorithmic Learning Theory*, S. 402–413. Springer (2002).
- [22] R. Coulom. *High-accuracy value-function approximation with neural networks applied to the acrobat*. In *European Symposium on Artificial Neural Networks (ESANN)*, S. 7–12 (2004).
- [23] M. P. Deisenroth, C. E. Rasmussen und J. Peters. *Gaussian Process Dynamic Programming* (2008). To appear in: Neurocomputing.
- [24] K. Doya. *Reinforcement learning in continuous time and space*. Neural Computation, 12: 219–245 (2000).
- [25] K. Doya. *Robust Reinforcement Learning*. In *Advances in Neural Information Processing Systems 13*, S. 1061–1067. MIT Press (2001).

-
- [26] Y. Engel, S. Mannor und R. Meir. *Reinforcement learning with Gaussian processes*. In *In Proc. of the 22nd International Conference on Machine Learning*, S. 201–208 (2005).
- [27] D. Ernst, P. Geurts, L. Wehenkel und L. Littman. *Tree-based batch mode reinforcement learning*. *Journal of Machine Learning Research*, 6: 2005 (2005).
- [28] O. Föllinger. *Regelungstechnik Einführung in die Methoden und ihre Anwendung*. Hüthig, 8. Aufl. (1994). ISBN 3-7785-2336-8.
- [29] T. Gabel, R. Hafner, S. Lange, M. Lauer und M. Riedmiller. *Bridging the Gap: Learning in the RoboCup Simulation and Midsize League*. In *Proceedings of the 7th Portuguese Conference on Automatic Control (Controlo 2006)* (2006).
- [30] T. Gabel und M. Riedmiller. *Evaluation of Batch-Mode Reinforcement Learning Methods for Solving DEC-MDPs with Changing Action Sets*. S. 82–95 (2008).
- [31] C. Gaskett, D. Wettergreen und E. Zelinsky. *Q-learning in continuous state and action spaces*. In *In Australian Joint Conference on Artificial Intelligence*, S. 417–428. Springer-Verlag (1999).
- [32] G. J. Gordon. *Stable Function Approximation in Dynamic Programming*. In *Proceedings of IMCL '95* (1995).
- [33] G. J. Gordon. *Stable fitted reinforcement learning*. In *Advances in Neural Information Processing Systems 8*, S. 1052–1058. MIT Press (1996).
- [34] G. J. Gordon. *Approximate Solutions to Markov Decision Processes*. Techn. Ber. (1999).
- [35] J. Govindhasamy, S. McLoone, G. Irwin, J. French und R. Doyle. *Reinforcement Learning for Online Control and Optimisation*. In *IEE Control Engineering Book Series*, S. 293–326. , 70. Aufl. (2005).
- [36] J. Grizzle und J.-M. Kang. *Discrete-time control design with positive semi-definite Lyapunov functions*. *Systems and Control Letters*, 43(4): 87–292(6) (2001).
- [37] H.-M. Gross, V. Stephan und M. Krabbes. *A neural field approach to topological reinforcement learning in continuous action spaces*. In *In Proc. of WCCI-IJCNN'98, Anchorage*, S. 1992–1997. IEEE Press (1998).
- [38] R. Hafner, S. Lange, M. Lauer und M. Riedmiller. *Brainstormers Tribots Team Description*. In G. Lakemeyer, E. Sklar, D. Sorrenti und T. Takahashi, Hg., *RoboCup 2006: Robot Soccer World Cup X*. LNCS. Springer (2006).
- [39] R. Hafner und M. Riedmiller. *Reinforcement Learning on an omnidirectional mobile robot*. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*. Las Vegas (2003).
- [40] R. Hafner und M. Riedmiller. *Case study: Control of a Real World System in CLSquare*. NIPS: Reinforcement Learning Benchmarks and Bake-offs II (2005).
- [41] R. Hafner und M. Riedmiller. *Neural Reinforcement Learning Controllers for a Real Robot Application*. In *2007 IEEE International Conference on Robotics and Automation*. IEEE, Rome, Italy (2007).

- [42] S. Hagen und B. Kröse. *Q-Learning for systems with continuous state and action spaces.*. In *BENELEARN 2000, 10th Belgian-Dutch Conference on Machine Learning* (2000).
- [43] M. E. Harmon und L. C. B. Iii. *Residual Advantage Learning Applied to a Differential Game.* In *In Proc. of the International Conference on Neural Networks, Washington D.C* (1995).
- [44] V. Heidrich-Meisner, M. Lauer, C. Igel und M. Riedmiller. *Reinforcement Learning in a Nutshell.* In M. Verleysen, Hg., *15th European Symposium on Artificial Neural Networks (ESANN 2007)*, S. 277–288. Evere, Belgien: d-side publications (2007).
- [45] I. W. Hunter und M. Korenberg. *The identification of nonlinear biological systems: Wiener and Hammerstein cascade models.* *Biol. Cybern.*, (55): 135–144 (1986).
- [46] G. W. Irwin, K. Warwick und K. J. Hunt. *Neural Network Applications in Control.* IET (1995). ISBN 0852968523.
- [47] L. P. Kaelbling, M. L. Littman und A. W. Moore. *Reinforcement learning: a survey.* *Journal of Artificial Intelligence Research*, 4: 237–285 (1996).
- [48] S. Kalyanakrishnan und P. Stone. *Batch Reinforcement Learning in a Complex Domain.* In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems* (2007).
- [49] H. Kimura und S. Kobayashi. *Reinforcement learning for continuous action using stochastic gradient ascent.* In *5th. Intl. Conf. on Intelligent Autonomous Systems*, S. 288–295 (1998).
- [50] V. R. Konda, John und N. Tsitsiklis. *Actor-critic algorithms.* In *SIAM Journal on Control and Optimization*, S. 1008–1014. MIT Press (2000).
- [51] R. M. Kretchmar. *A Synthesis of Reinforcement Learning and Robust Control Theory.* Dissertation, Colorado State University, Fort Collins, CO (2000).
- [52] R. M. Kretchmar, P. M. Young und C. W. Anderson. *Robust Reinforcement Learning Control.* In *ACC2001: 2001 American Control Conference.* Washington DC (2001).
- [53] R. M. Kretchmar, P. M. Young, C. W. Anderson und M. L. Delnero. *Robust Reinforcement Learning Control with Static and Dynamic Stability.* *International Journal of Robust and Nonlinear Control*, 11: 1469–1500 (2001).
- [54] M. G. Lagoudakis, R. Parr und L. Bartlett. *Least-squares policy iteration.* *Journal of Machine Learning Research*, 4: 2003 (2003).
- [55] M. G. Lagoudakis, R. Parr und M. L. Littman. *Least-squares methods in reinforcement learning for control.* In *In Proceedings of the Second Hellenic Conference on AI*, S. 249–260. Springer-Verlag (2002).
- [56] M. Lauer und R. Hafner. *Roboter lernen Fußball spielen.* *Physik in unserer Zeit*, (2): 82–87 (2007).

-
- [57] A. Lazaric, M. Restelli und A. Bonarini. *Reinforcement Learning in Continuous Action Spaces through Sequential Monte Carlo Methods*. In *NIPS*.
- [58] L.-J. Lin. *Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching*. *Mach. Learn.*, 8(3-4): 293–321 (1992). ISSN 0885-6125.
- [59] J. D. Martín-Guerrero, E. Soria-Olivas, M. Martínez-Sober, A. J. Serrano-López, R. Magdalena-Benedito und J. Gómez-Sanchis. *Use of Reinforcement Learning in Two Real Applications*. S. 191–204 (2008).
- [60] J. D. R. Millán, D. Posenato und E. Dedieu. *Continuous-Action Q-Learning*. *Machine Learning*, 49(2-3): 247–265 (2002). ISSN 0885-6125.
- [61] T. W. Miller, R. S. Sutton und P. J. Werbos. *Neural Networks for Control*. MIT Press (1995). ISBN 026263161X.
- [62] A. W. Moore. *Variable Resolution Dynamic Programming*. In *ML*, S. 333–337 (1991).
- [63] A. Y. Ng und M. Jordan. *Pegasus: A policy search method for large mdps and pomdps*. In *In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, S. 406–415 (2000).
- [64] A. Y. Ng, H. J. Kim, M. I. Jordan und S. Sastry. *Inverted autonomous helicopter flight via reinforcement learning*. In *In International Symposium on Experimental Robotics*. MIT Press (2004).
- [65] D. Ormoneit und S. Sen. *Kernel-based reinforcement learning*. In *Machine Learning*, S. 161–178 (2002).
- [66] P. Pachowicz. *Competitive reinforcement learning in continuous control tasks*. In *In Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (2003).
- [67] T. J. Perkins und A. G. Barto. *Lyapunov-constrained action sets for reinforcement learning*. In *Proceedings of the Eighteenth International Conference on Machine Learning*, S. 409–416. Morgan Kaufmann (2001).
- [68] T. J. Perkins und A. G. Barto. *Lyapunov design for safe reinforcement learning*. *Journal of Machine Learning Research*, 3: 803–832 (2002).
- [69] J. Peters und S. Schaal. *Natural actor-critic*. In *Proceedings of the Sixteenth European Conference on Machine Learning*, S. 280–291. springer (2005).
- [70] J. Peters und S. Schaal. *Policy gradient methods for robotics*. In *In Proceedings of the IEEE international conference on intelligent robotics systems (iros)*, S. 2219–2225 (2006).
- [71] D. Prokhorov und D. Wunsch. *Adaptive Critic Designs*. *IEEE Transactions on Neural Networks*, 8: 997–1007 (1997).
- [72] D. V. Prokhorov. *Adaptive critic designs and their applications*. Dissertation, Lubbock, TX, USA (1997).
- [73] C. E. Rasmussen und M. Kuss. *Gaussian Processes in Reinforcement Learning*. In *Advances in Neural Information Processing Systems 16*, S. 751–759. MIT Press (2004).

- [74] M. Riedmiller. *Application of sequential reinforcement learning to control dynamic systems*. In *In IEEE International Conference on Neural Networks (ICNN '96)* (1996).
- [75] M. Riedmiller. *Selbstaendig lernende neuronale Steuerungen*. Phd thesis, Universitaet Karlsruhe (1996).
- [76] M. Riedmiller. *Generating continuous control signals for reinforcement controllers using dynamic output elements*. In *Proc. of the European Symposium on Neural Networks, ESANN '97*. Brugge, Belgium (1997).
- [77] M. Riedmiller. *Reinforcement learning without an explicit terminal state*. In *Proc. of the International Joint Conference on Neural Networks, IJCNN '98*. Anchorage, Alaska (1998).
- [78] M. Riedmiller. *Concepts and facilities of a neural reinforcement learning control architecture for technical process control..* Journal of Neuro Computing and Applications, 8: 323–338 (2000).
- [79] M. Riedmiller. *Neural Fitted Q Iteration - First experiences with a data efficient neural Reinforcement Learning Method*. In *Proc. of the European Conference on Machine Learning, ECML 2005*. Porto, Portugal (2005).
- [80] M. Riedmiller und H. Braun. *RPROP: A Fast Adaptive Learning Algorithm*. In E. Gelenbe, Hg., *International Symposium on Computer and Information Science VII*, S. 279 – 286. Antalya, Turkey (1992).
- [81] M. Riedmiller und H. Braun. *A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm*. In H. Ruspini, Hg., *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, S. 586 – 591. San Francisco (1993).
- [82] M. Riedmiller und H. Braun. *RPROP: A Fast and Robust Backpropagation Learning Strategy*. In M. Jabri, Hg., *Fourth Australian Conference on Neural Networks*, S. 169 – 172. Melbourne (1993).
- [83] M. Riedmiller, T. Gabel, R. Hafner, S. Lange und M. Lauer. *Die Brainstormers: Entwurfsprinzipien lernfähiger autonomer Roboter*. Informatik-Spektrum, 29(3): 175–190 (2006).
- [84] M. Riedmiller, R. Hafner, S. Lange und M. Lauer. *Learning to dribble on a real robot by success and failure*. In *IEEE International Conference on Robotics and Automation (ICRA 2008)*, S. 2207–2208 (2008).
- [85] F. Rivest und D. Precup. *Combining TD-learning with cascade-correlation networks*. In *Proceedings of the 20th International Conference on Machine Learning*. AAAI Press, Washington, DC. Menlo Park, CA (2003).
- [86] R. Rojas. *Systemtheorie der neuronalen Netze*. Springer (1994).
- [87] A. Rottmann, C. Plagemann, P. Hilgers und W. Burgard. *Autonomous blimp control using model-free reinforcement learning in a continuous state and action space*. In *IROS*, S. 1895–1900 (2007).

- [88] G. A. Rummery und M. Niranjan. *On-line Q-learning using connectionist systems*. Techn. Ber. CUED/F-INFENG/TR 166, Cambridge University Engineering Department (1994).
- [89] J. C. Santamaría, R. S. Sutton und A. Ram. *Experiments with reinforcement learning in problems with continuous state and action spaces*. *Adapt. Behav.*, 6(2): 163–217 (1997). ISSN 1059-7123.
- [90] J. Sarangapani. *Neural Network Control of Nonlinear Discrete-time Systems*. CRC Press (2006). ISBN 0824726774.
- [91] R. Schoknecht und M. Riedmiller. *Using reinforcement learning for engine control*. In *In Ninth International Conference on Arti Neural Networks (ICANN '99)* (1999).
- [92] E. A. Sherstov und P. Stone. *Function approximation via tile coding: Automating parameter choice*. In *of Lecture Notes in Artificial Intelligence*, S. 194–205. Springer Verlag (2005).
- [93] J. Si, A. G. Barto, W. B. Powell und D. Wunsch. *Handbook of learning and approximate dynamic programming*. Wiley-IEEE (2004). ISBN 047166054X.
- [94] Y.-T. W. Si J. *On-Line Learning Control by Association and Reinforcement*. *IEEE Transactions on Neural Networks*, 12(2): 264–276 (2001).
- [95] J. E. Slotine und W. Li. *Applied Nonlinear Control*. Prentice Hall (1991).
- [96] V. Stephan, K. Debes, H.-M. Gross, F. Wintrich und H. Wintrich. *Reinforcement Learning Based Control Scheme for a Visually-guided Combustion Process*. In *Proc. 15th European Meeting on Cybernetics and Systems Research (EMCSR 2000)*, S. 555–558. Austrian Society for Cybernetic Studies, Vienna (2000).
- [97] V. Stephan, M. Saupe, M. Bischoff, H. Reindanz und H.-M. Gross. *Is Reinforcement-Learning Able to Solve Real-World Challenges?*. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN/ICONIP 2003)*, S. 346–349. Istanbul (2003).
- [98] V. Stephan, F. Wintrich, A. König und K. Debes. *Application of Action Dependent Heuristic Dynamic Programming to Control an Industrial Waste Incineration Plant*. In *Fortschrittberichte VDI*, Nr. 743 in 10, S. 262–270. VDI-Verlag (2004).
- [99] D. Surmeli, G. Kohler und H.-M. Gross. *Medically Motivated Testbed for Reinforcement Learning in Neural Architectures*. In *Proceedings of the 5. International Workshop Fuzzy-Neuro Systems 98 (FNS 98)* (1998).
- [100] R. S. Sutton. *Generalization in reinforcement learning: Successful examples using sparse coarse coding*. In *Advances in Neural Information Processing Systems 8*, S. 1038–1044. MIT Press (1996).
- [101] R. S. Sutton und A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press (1998). ISBN 0262193981.
- [102] R. S. Sutton, A. G. Barto und R. J. Williams. *Reinforcement learning is direct adaptive optimal control*. In *In Proceedings of the American Control Conference*, S. 2143–2146 (1991).

- [103] R. S. Sutton, M. L. Littman, D. Precup und P. Stone. *NIPS Workshop: The First Annual Reinforcement Learning Competition* (2006).
- [104] R. S. Sutton und S. Whitehead. *Online learning with random representation*. In *Proceedings of the 10th International Conference on Machine Learning*, S. 314–321. Morgan Kaufmann (1993).
- [105] G. Tesauro. *Practical issues in temporal difference learning*. In *Machine Learning*, S. 257–277 (1992).
- [106] S. Timmer und M. Riedmiller. *Fitted Q Iteration with CMACs*. In *International Symposium on Approximate Dynamic Programming and Reinforcement Learning. ADPRL07*, S. 1–8. IEEE, Honolulu (2007).
- [107] N. K. Treadgold und T. D. Gedeon. *A cascade network algorithm employing progressive rprop*. In *In Proceedings International Work-Conference on Artificial and Natural Neural Networks (IWANN'97). Canary Islands*, S. 733–742 (1997).
- [108] A. G. Tsirikis und M. Morari. *Controller design with actuator constraints*. In *Proceedings of the 31 st Conference on Decision and Control* (1992).
- [109] University of Michigan, www.engin.umich.edu/group/ctm. *CTM: Digital Control Tutorial* (1996). (online).
- [110] H. Van Hasselt und M. A. Wiering. *Reinforcement Learning in Continuous Action Spaces*. In *Dynamic Programming and Reinforcement Learning (ADPRL 2007)* (2007).
- [111] C. J. Watkins. *Learning from Delayed Rewards*. Phd thesis, Cambridge University (1989).
- [112] C. J. Watkins und P. Dayan. *Q-learning*. *Machine Learning*, 8(3): 279–292 (1992).
- [113] R. J. Williams. *Simple statistical gradient-following algorithms for connectionist reinforcement learning*. In *Machine Learning*, S. 229–256 (1992).
- [114] Z.-J. Yang, K. Kunitoshi, S. Kanae und K. Wada. *Adaptive Robust Output Feedback Control of a Magnetic Levitation System by K-Filter Approach*. *IEEE Transactions on Industrial Electronics*, 55(1): 390–399 (2008).
- [115] Z.-J. Yang und M. Minashima. *Robust Nonlinear Control of a Feedback Linearizable Voltage-Controlled Magnetic Levitation System*. *Transactions of the Institute of Electrical Engineers of Japan*, S. 1203–1211 (2001).
- [116] Z.-J. Yang und M. Tateishi. *Adaptive Robust Nonlinear Control of a Magnetic Levitation System*. *Automatica*, 37(7): 1125–1131 (2001).
- [117] Z.-J. Yang, H. Tsubakihara, S. Kanae und K. Wada. *Robust Nonlinear Control of a Voltage-Controlled Magnetic Levitation System Using Disturbance Observer*. *Transactions of IEE of Japan*, 127-C(12): 2118–2125 (2007).
- [118] A. Zell. *Simulation neuronaler Netze*. Addison-Wesley (1994).