Physics Informed Neural Networks zur Lösung des dynamischen Verhaltens chaotischer Systeme

Studienarbeit

des Studienganges Informatik – Intelligente Systeme & Informationstechnik

an der Dualen Hochschule Baden-Württemberg Ravensburg

von

Finley Hogan, Justin Masch und Philipp Rottweiler

14.07.2025

Bearbeitungszeitraum	24 Wochen
Matrikelnummer, Kurs	6486412, TIK22
	4705095, TIT22
	7095401, TIK22
Gutachter der Dualen Hochschule	Prof. Dr. Jürgen Schneider

Eidesstaatliche Erklärung

gemäß Ziffer 1.1.13 der Anlage 1 zu §§ 3, 4 und 5 der Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg vom 29.09.2017.

Wir versichern hiermit, dass wir die Studienarbeit mit dem Thema:

Physics Informed Neural Networks zur Lösung des dynamischen Verhaltens chaotischer Systeme

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versichern zudem, dass alle eingereichten Fassungen übereinstimmen.

.

Philipp Rothweiler

Friedrichshafen, den 14.07.2025

Zusammenfassung

Vor allem im industriellen Bereich werden maschinelle Lernverfahren in Anwendungen eingesetzt, die mit physisch existierenden Objekten interagieren. Zur Verbesserung des Lernprozesses ist es dabei möglich, die der jeweiligen Anwendung zugrundeliegenden physikalischen Gesetzmäßigkeiten in die Struktur eines neuronalen Netzes zu integrieren. Auf diese Weise modifizierte Physics Informed Neural Networks (PINNs) können nicht nur zur Simulation des Verhaltens physikalischer Systeme genutzt werden. Darüber hinaus sind PINNs dazu fähig, aus einer Menge an Messdaten relevante, aber nicht direkt beobachtbare Systemparameter abzuschätzen. Im Vergleich zu alternativen numerischen Lösungsverfahren reduzieren PINNs den Aufwand zur Lösung einer solchen inversen Problemstellung erheblich. Diese Eigenschaft bietet insbesondere für die Analyse chaotischer Systeme ein hohes Anwendungspotential. So könnte anhand von Messdaten festgestellt werden, ob sich ein System in einem chaotischen Parameterbereich befindet. Auf dieser Information aufbauend können Aussagen über die Vorhersagbarkeit des Systems getroffen und etwaig nötige Gegenmaßnahmen eingeleitet werden.

Im Rahmen dieser Studienarbeit ist empirisch zu untersuchen, inwiefern PINNs zur Lösung inverser Probleme für chaotische Systeme geeignet sind. Hierzu ist zunächst ein geeignetes chaotisches System, sowie passende Technologien zur Implementierung eines PINNs zu identifizieren. Daraufhin ist ein für das spezifische System konzipiertes PINN zu entwickeln und dessen korrekte Funktionsweise zu validieren. Zum Abschluss ist die Eignung des implementierten PINNs zur Simulation und Lösung des inversen Problems bei chaotischem Verhalten zu prüfen.

Die Ergebnisse zeigen, dass durch gezielte Architektur- und Trainingsanpassung auch in chaotischen Bereichen eine realitätsnahe Simulation, sowie eine zuverlässige Schätzung der Systemparameter möglich ist. Die Untersuchungen zeigen sowohl das Potenzial als auch bestehende Herausforderungen auf. Die vorliegende Arbeit leistet damit einen empirischen Beitrag zur Einordnung der Fähigkeiten von PINNs im Umfeld chaotischer Systeme.

Abstract

In the industrial sector in particular, machine learning methods are used in applications that interact with physically existing objects. To improve the learning process, it is possible to integrate the physical laws underlying the respective application into the structure of a neural network. Physics Informed Neural Networks (PINNs) can not only be used to simulate the behavior of physical systems. PINNs are also capable of estimating relevant but not directly observable system parameters from a set of measurement data. Compared to alternative numerical solution methods, PINNs considerably reduce the effort required to solve such an inverse problem. This property offers great application potential, especially for the analysis of chaotic systems. For example, measurement data can be used to determine whether a system is in a chaotic parameter range. Based on this information, statements can be made about the predictability of the system and any necessary countermeasures can be initiated.

The aim of this thesis is to empirically investigate the extent to which PINNs are suitable for solving inverse problems for chaotic systems. The first step is to identify a suitable chaotic system and fitting technologies for implementing a PINN. A PINN designed for the specific system must then be developed and validated. Finally, the suitability of the implemented PINN for simulating and solving the inverse problem with chaotic behavior is to be tested.

A partial result of this work is the software for creating a PINN for the Lorenz system.

The results show that a realistic simulation as well as a reliable estimation of the system parameters is possible even in chaotic areas through targeted architecture and training adaptation. The investigations show both the potential and existing challenges. The present work thus makes an empirical contribution to the evaluation of PINNs in the context of solving the inverse problem of chaotic systems.

DHBW
Duale Hochschule
Baden-Württemberg

Inhalt

DHBW
Duale Hochschule
Baden-Württemberg

Abkürzungsverzeichnis

DGL ..... Differentialgleichung
IC ..... Initial Condition
ISTQB ..... International Software Testing Qualifications Board
MSE ..... Mean Squared Error
NN ..... Neuronales Netz
PINNs ..... Physics Informed Neural Networks
ReLU ..... Rectified Linear Unit
RK45 ..... Runge-Kutta Verfahren vierter Ordnung
SiLU ..... Sigmoid-weighted Linear Unit

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

Die interdisziplinäre Verknüpfung verschiedener Wissensgebiete ist die Basis für zahlreiche Innovationen der modernen Technologiegeschichte. Von der Nutzung der Fourier-Transformation als Schlüsselwerkzeug in der Signalverarbeitung[1], über die Verbindung von Kommunikations- und Rechnertechnik zur Vernetzung einzelner Computer,[2] bis hin zur Anwendung der Zahlentheorie bei der Verschlüsselung von Nachrichten.[3] All diese Technologien verbindet, dass der Erfolg der Entwicklung durch die Integration von Wissen begründet ist, das ursprünglich in einem anderen Fachgebiet erforscht und angewendet wurde. Die jeweilige technische Problemstellung wird gelöst, indem Parallelen zwischen der vorliegenden Herausforderung und den Kenntnissen eines anderen wissenschaftlichen Faches identifiziert werden. So können Methoden und Eigenschaften des zusätzlich herangezogenen Wissensgebietes dazu genutzt werden, ein Problem auf einer höheren Abstraktionsebene zu betrachten, um so eine effizientere Lösung zu entwickeln und neue Erkenntnisse zu gewinnen.

Genauso wie die menschliche Fähigkeit, Probleme zu lösen, durch die Berücksichtigung von bereits erforschten Wissensgebieten gesteigert werden kann, sollen auch maschinelle Lernverfahren durch die Einbettung von zusätzlichen Informationen verbessert werden. Ein Ansatz hierzu besteht darin, bekannte physikalische Zusammenhänge in die Struktur eines neuronalen Netzes zu integrieren.[4] Vor allem im industriellen Bereich werden neuronale Netze in Anwendungen eingesetzt, die mit real existierenden Objekten interagieren, und demnach anhand von physikalischen Modellen beschrieben werden können. Beispielhaft hierfür sind der Einsatz von neuronalen Netzen im Bereich der Robotik[5], zur Überwachung von Produktionsprozessen[6] oder in der Regelungstechnik[7] anzuführen. Beim Training eines neuronalen Netzes für eine derartige Anwendung können nun die jeweils zugrundeliegenden physikalischen Gesetzmäßigkeiten in Form

[1] Vgl. Karrenberg, *Universalschlüssel Fourier-Transformation*, 25.
[2] Vgl. Hoffmann, *Einführung in die Informations- und Codierungstheorie*, 66 ff.
[3] Vgl. Beutelspacher, Schwenk, und Wolfenstetter, *Moderne Verfahren der Kryptographie*, 9 ff.
[4] Vgl. Neuer, *Maschinelles Lernen für die Ingenieurwissenschaften*, 185 ff.
[5] Vgl. Mitschke, *Konvolutionäre neuronale Netze in der industriellen Bildverarbeitung und Robotik*, 21.
[6] Vgl. Neuer, *Maschinelles Lernen für die Ingenieurwissenschaften*, 188 ff.
[7] Vgl. Zacher und Reuter, *Regelungstechnik für Ingenieure*, 377 ff.

von analytischen Ausdrücken in den Aufbau des neuronalen Netzes integriert werden. Diese so entstandenen Physics-Informed Neural Networks (PINNs) reichern folglich die mathematische Grundstruktur eines neuronalen Netzes um zusätzliches Wissen an, mit dem Ziel auf diese Weise den Lernprozess eines neuronalen Netzes zu vereinfachen.[8]

1.1 Forschungsstand und Problemstellung

PINNs finden im Wesentlichen in zwei verschiedenen Aufgabenbereichen Anwendung. Zum einen werden PINNs dazu eingesetzt, das Verhalten physikalischer Systeme zu simulieren. Hierbei besteht das Ziel darin, bestimmte Eigenschaften eines Systems für verschiedene Eingangswerte so vorherzusagen, dass diese möglichst exakt mit dem real beobachteten Verhalten des Systems übereinstimmen.[9] Als anschauliches Beispiel hierfür ist die Vorhersage der Auslenkung eines Fadenpendels zu verschiedenen Zeitpunkten bei gegebener Anfangsauslenkung anzuführen. Demgegenüber steht der Einsatz von PINNs zur Lösung von inversen Problemen. Im Gegensatz zur Simulation eines Systems ist die Vorbedingung bei der Inversion eine Menge an Messdaten für ein bestimmtes System. Aus diesen Messdaten sollen bei der Lösung eines inversen Problems relevante, aber nicht direkt beobachtbare Systemparameter abgeschätzt werden.[10] Für das zuvor genannte Anschauungsbeispiel des Fadenpendels, kann etwa die Abschätzung der am Pendel angebrachten Masse aus gemessenen Auslenkungswerten als inverses Problem angeführt werden.

Nach derzeitigem Erkenntnisstand stellen PINNs insofern eine Verbesserung gegenüber klassischen Methoden des maschinellen Lernens dar, dass für das erfolgreiche Training eines PINNs auch Daten von geringer Qualität ausreichend sein können.[11] Insbesondere zeichnen sich PINNs dadurch aus, dass Simulations- oder Inversionsaufgaben mit nur wenigen, lückenhaften oder verrauschten Daten gelöst werden können, wie etwa Steger aufzeigt.[12] Zugleich weisen PINNs im Vergleich zu üblichen maschinellen Lernverfahren bessere Fähigkeiten bei der Generalisierung des

[8] Vgl. Neuer, *Maschinelles Lernen für die Ingenieurwissenschaften*, 186 f.
[9] Vgl. Moseley, „Physics-Informed Machine Learning“, 18.
[10] Vgl. Moseley, 20.
[11] Vgl. Wu, Sicard, und Gadsden, „Physics-informed machine learning“, 16.
[12] Vgl. Steger, „Physics informed neural networks : analysis for a dynamical system - the double pendulum“, 19 ff.

jeweiligen Problems auf. Somit sind PINNs in geringerem Ausmaß von der Verfügbarkeit und Qualität von Trainingsdaten abhängig.[13] Weiterhin ist die Anwendung von PINNs in bestimmten Szenarien auch gegenüber numerischen Methoden vorteilhaft. Während numerische Berechnungen nach Meng et al. zwar besser zur Berechnung exakter Simulationsergebnisse für detailliert beschriebene Systeme geeignet sind, ermöglichen PINNs es, Vorhersageergebnisse mit niedrigerem Implementierungsaufwand und geringerer Rechenkomplexität zu generieren.[14] Zudem ist die Lösung inverser Probleme mithilfe von numerischen Methoden deutlich komplexer als die hierfür nötige Anpassung eines PINNs.[15]

In verschiedenen Arbeiten wird allerdings ebenso angemerkt, dass die Simulation von Systemen, deren Verhalten als chaotisch oder turbulent zu klassifizieren ist, PINNs Schwierigkeiten bereitet.[16] [17] [18]Darüber hinaus ist der Aufwand für die Entwicklung und das Training eines PINNs erheblich höher, wenn hochfrequente Komponenten des Verhaltens eines Systems durch ein PINN abgebildet werden sollen.[19] [20]

Gerade für die Analyse chaotischer Systeme könnten PINNs jedoch einen hohen Grad an Nützlichkeit aufweisen. Nachdem PINNs als effizientes Lösungsschema für inverse Probleme verwendet werden können, liegt es nahe PINNs auf Systemen anzuwenden, die für bestimmte Parameter chaotisches Verhalten aufweisen. Durch die Abschätzung der Systemparameter mithilfe von PINNs wäre es möglich festzustellen, ob sich ein System in einem chaotischen Parameterbereich befindet. Somit könnte bereits anhand weniger Messwerte festgestellt werden, ob das Verhalten eines Systems langfristig vorhersagbar ist oder nicht. Systeme, von denen bekannt ist, dass diese unter bestimmten Bedingungen chaotisches Verhalten zeigen, könnten auf diese Weise effizient auf entsprechende Systemparameter untersucht werden. Diese Information könnte beispielweise in technischen Systemen, bei denen chaotisches Verhalten nicht

---

[13] Vgl. Neuer, *Maschinelles Lernen für die Ingenieurwissenschaften*, 190 f.
[14] Vgl. Meng u. a., „When Physics Meets Machine Learning“, 2.
[15] Vgl. Moseley, „Physics-Informed Machine Learning“, 20 f.
[16] Vgl. Steger, Rohrhofer, und Geiger, „How PINNs Cheat: Predicting Chaotic Motion of a Double Pendulum“, 1.
[17] Vgl. Krishnapriyan u. a., „Characterizing possible failure modes in physics-informed neural networks“, 1.
[18] Vgl. Wang, Sankaran, und Perdikaris, „Respecting Causality Is All You Need for Training Physics-Informed Neural Networks“, 1.
[19] Vgl. Basri u. a., „The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies“, 9.
[20] Vgl. Rahaman u. a., „On the Spectral Bias of Neural Networks“, 8.

erwünscht ist, dazu genutzt werden, die Systemparameter so zu verändern, dass sich das System langfristig vorhersagbar und damit nicht chaotisch verhält.[21] Weiterhin könnten PINNs dazu eingesetzt werden, tatsächlich chaotisches Verhalten von solchen zeitlichen Verläufen zu separieren, die durch einen hohen Anteil an Hintergrundrauschen lediglich unregelmäßig erscheinen.

1.2 Zielsetzung und Forschungsfragen

Ziel dieser Arbeit ist es empirisch zu untersuchen, ob PINNs dazu geeignet sind, inverse Probleme für Systeme zu lösen, die bei einer bestimmten Auswahl an Parametern chaotisches Verhalten aufweisen. Es ist folglich zu prüfen, inwiefern PINNs es erlauben, anhand einer Menge an Messdaten die zugrundeliegenden Parameter eines Systems zu ermitteln, an dem chaotisches Verhalten beobachtet werden kann. Auf diese Weise soll es möglich sein, auf Basis von Messdaten das Verhalten eines Systems entweder als chaotisch oder als nicht chaotisch zu klassifizieren. Voraussetzung dafür ist allerdings, dass bekannt ist, in welchen Parameterintervallen das System chaotisches Verhalten zeigt. Mithilfe des PINNs sollen demzufolge die Parameterwerte, die die beobachteten Messwerte hervorbringen, identifiziert werden. Diese können anschließend dazu genutzt werden, das Verhalten des Systems einzuordnen. Hierzu muss das betrachtete System bereits mithilfe der entsprechenden Methoden zur Analyse chaotischer Systeme untersucht sein.

Eine weitere Forschungsfrage, die es im Rahmen der Arbeit zu beantworten gilt, lautet folglich, ob PINNs die Parameter des Systems mit einem Maß an Genauigkeit bestimmen können, der es erlaubt zwischen chaotischem und nicht chaotischem Verhalten zu differenzieren. Selbst wenn die Systemparameter nicht exakt mithilfe von PINNs bestimmt werden können, wäre so die zuvor beschriebene Identifikation chaotischen Verhaltens möglich.

Wie im zweiten Kapitel deutlich wird, dient die erfolgreiche Simulation eines Systems die Vorbedingung für die Lösung des inversen Problems mithilfe eines PINNs. Daraus ergibt sich die Entwicklung eines PINNs, das die Simulation eines chaotischen Systems für einen kurzen Zeitraum erlaubt, als zusätzliches Teilziel dieser Arbeit.

[21] Vgl. Strogatz, *Nonlinear Dynamics and Chaos*, 342.

Kann ein PINN nicht darauf trainiert werden, das Verhalten eines Systems zu simulieren, so ist auch die realistische Abschätzung der Systemparameter nicht umsetzbar.

Des Weiteren ist es wichtig zu beachten, dass der Fokus der Arbeit auf der Untersuchung der Eigenschaften und Einsatzmöglichkeiten von PINNs liegt. Diese werden zwar konkret auf ihre Fähigkeiten bei der Simulation und Inversion von chaotischen Systemen hin evaluiert. Dennoch ist die Erforschung von Eigenschaften chaotischer Systeme ausdrücklich nicht Bestandteil dieser Arbeit. Zusätzlich ist anzumerken, dass ausschließlich empirische Methoden zur Beantwortung der Forschungsfrage eingesetzt werden. Mathematisch-deduktive Ansätze finden keine Anwendung.

1.3 Aufbau der Arbeit

Zur Realisierung der soeben aufgeführten Ziele sind zunächst grundlegende theoretische Kenntnisse hinsichtlich der Funktionsweise von neuronalen Netzen nötig. Auf dieser Basis werden anschließend die Anpassungen erläutert, die vorzunehmen sind, um ein neuronales Netz in ein Physics Informed Neural Network zu transformieren. Von besonderem Interesse ist es dabei darzulegen, wie ein PINN zur Lösung einer Simulationsaufgabe aufgebaut ist, und welche Veränderungen an der Struktur eines PINNs zur Lösung des inversen Problems erforderlich sind. Zudem sind als zweites Element der für das Verständnis der Arbeit relevanten theoretischen Grundlagen die Charakteristiken chaotischer Systeme einzuführen. Neben einer kurzen Einführung in den Themenbereich der nicht-linearen dynamischen Systeme werden an dieser Stelle Methoden zur Beschreibung chaotischer Systeme vorgestellt. Die Darstellung der gerade genannten theoretischen Grundlagen von PINNs und chaotischen Systemen erfolgt im zweiten Kapitel.

Anschließend wird die Methodik erarbeitet, anhand derer die im vorigen Teilabschnitt dargelegten Projektziele umgesetzt werden sollen. Grundlage des empirischen Vorgehens dieser Arbeit bildet dabei die begründete Identifikation und Auswahl eines geeigneten chaotischen Systems. Ferner sind passende Technologien für die Implementierung eines PINNs zu bestimmen. Die Eignung der ausgewählten Technologien wird anhand der Realisierung von Prototypen nachgewiesen. Im Umkehrschluss können so ungeeignete Technologien ausgeschlossen werden. Der

letzte Teilabschnitt des dritten Kapitels dient dazu, das weitere Vorgehen zu demonstrieren.

Im vierten Kapitel der Arbeit wird die technische Umsetzung der zuvor beschriebenen Methodik dokumentiert. Hierbei ist die Entwicklung und Implementierung der verschiedenen Formen von PINNs genauso darzulegen, wie die Strategie und Umsetzung der Validierung der implementierten PINNs. Anschließend sollen möglichst optimale Hyperparameter für die Lösung von Simulation- und Inversionsproblemen identifiziert werden.

Die auf diese Weise entwickelten PINNs werden daraufhin zunächst zur Vorhersage des kurzfristigen Verhaltens des Systems bei chaotischen Parametern eingesetzt. In einem letzten Schritt ist die Eignung des PINNs zur Lösung des inversen Problems bei chaotischen Parametern zu prüfen. Die Resultate dieser Untersuchungen werden im fünften Kapitel zusammengestellt und ausgewertet.

Zum Abschluss der Studienarbeit werden die erzielten Ergebnisse diskutiert, sowie Möglichkeiten für auf der Arbeit aufbauende, zukünftige Untersuchungen aufgezeigt.

2 Theoretische Grundlagen

Für das Verständnis der restlichen Arbeit ist es notwendig, vorab die beiden Themenblöcke der Physics Informed Neural Networks und der chaotischen Systeme detailliert zu betrachten. Dabei dient der erste Teilabschnitt des Kapitels dazu, die Funktionsweise von PINNs zu erläutern. Grundlage eines PINNs ist die mathematische Struktur eines neuronalen Netzes. Daher sind in einem ersten Schritt die Idee und der Aufbau eines neuronalen Netzes darzulegen. Eine gesonderte Stellung nimmt in diesem Abschnitt die Erläuterung des Lernalgorithmus der Backpropagation ein, der als zentraler Baustein neuronaler Netze zu bewerten ist. Auf der Basis der grundlegenden Kenntnis neuronaler Netze werden anschließend in einem zweiten Abschnitt die Charakteristiken von PINNs beschrieben, anhand derer sich PINNs vom üblichen Aufbau eines neuronalen Netzes unterscheiden. Hierbei ist ebenfalls darzulegen, wie ein PINN jeweils strukturiert sein muss, um entweder eine Simulationsaufgabe oder ein inverses Problem lösen zu können.

Ziel des dritten Kapitelabschnitts ist es, ein Verständnis für die Eigenschaften chaotischer Systeme zu schaffen. Neben der Definition chaotischer Systeme werden hierzu Methoden zur Beschreibung dynamischer Systeme eingeführt. Um über einen Referenzwert für die mithilfe von PINNs ermittelten Lösungen zu verfügen, ist es zudem erforderlich, numerische Lösungsverfahren für dynamische Systeme vorzustellen.

2.1 Neuronale Netze

PINNs bauen auf der Struktur eines neuronalen Netzes auf. Zur Veranschaulichung der Funktionsweise eines PINNs ist es notwendig in einem ersten Schritt den Aufbau eines neuronalen Netzes darzulegen. Dabei werden ausschließlich die für das Verständnis eines PINNs nötigen Grundstrukturen neuronaler Netze vorgestellt. Konkret bedeutet dies eine Einschränkung auf neuronale Netze ohne Rückkopplungsmechanismen, die anhand von überwachten Lernmethoden trainiert werden. Darüber hinaus soll in diesem Abschnitt eine einheitliche mathematische Notation zur Beschreibung neuronaler Netze etabliert werden. Diese kann anschließend zur Erklärung der Funktionsweise eines PINNs genutzt werden.

2.1.1 Grundelemente eines neuronalen Netzes

Grundbaustein eines neuronalen Netzes ist das einzelne Neuron. Das Konzept eines Neurons entspricht dem von Rosenblatt 1957 entwickelten Perceptron[22] und ist in untenstehender Abbildung veranschaulicht. Das Neuron, das durch das Summenzeichens gekennzeichnet ist, erhält eine beliebige Zahl an Eingabewerten $x_1, \dots, x_N$ und generiert einen Ausgabewert a**.** Jeder der Eingabewerte wird jeweils mit einem spezifischen Gewicht $w_1, \dots, w_N$ multipliziert. Die gewichteten Eingabewerte werden im Neuron aufsummiert, wobei ein zusätzlicher Summand – der sogenannte Bias oder Schwellenwert, in der Abbildung als b dargestellt – hinzuaddiert wird. Das Ergebnis dieser linearen Funktion dient schließlich als Eingabe für die nicht lineare Aktivierungsfunktion σ, die die gewichtete Summe aller Eingaben auf einen kleinen Wertebereich abbildet.

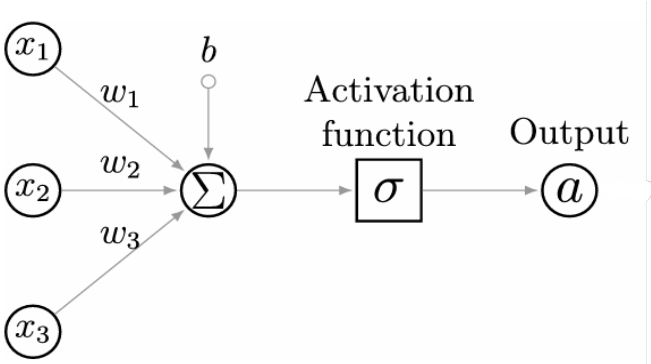

Abbildung 1: Visualisierung eines einzelnen Neurons
[in Anlehnung an: Kollmannsberger et al. (2021), S.21]

Aus mathematischer Sicht ist ein einzelnes Neuron folglich als Verkettung einer linearen und einer nicht linearen Funktion zu betrachten. Zur kompakten Darstellung der nötigen Rechenoperationen bietet es sich an, die vektorielle Schreibweise zu verwenden, die nachfolgend durch Fettdruck kenntlich gemacht wird. Damit kann die Funktionsweise eines Neurons mathematisch präzise anhand von zwei Gleichungen beschrieben werden.[23]

$$z = \left(\sum_{i=0}^{N} w_i \cdot x_i\right) + b = \boldsymbol{w^T} \cdot \boldsymbol{x} + b \qquad (2.1)$$

$$a(x) = \sigma(z) \qquad (2.2)$$

[22] Vgl. Rosenblatt, „The Perceptron“, 386–408.
[23] Vgl. Nielsen, „Neural Networks and Deep Learning“, 41.

Das Resultat der linearen Transformation der Eingangswerte ist explizit in der Variablen z kodiert. Auf diese Weise ist eindeutig gekennzeichnet, dass zwei separate Funktionen sequenziell auf einen Vektor an Eingabewerten angewendet werden.
Von besonderer Bedeutung ist, dass eine nicht lineare Funktion als Aktivierungsfunktion gewählt wird. Nur so ist sichergestellt, dass das neuronale Netz fähig ist, auch nicht lineare Zusammenhänge zu erfassen.[24] Eine Auswahl üblicherweise eingesetzter Aktivierungsfunktionen ist folgender Darstellung zu entnehmen.

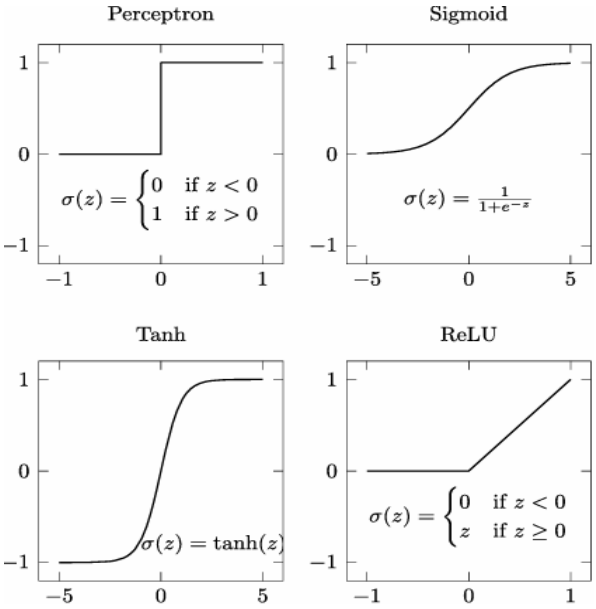

Abbildung 2: Übersicht verschiedener Aktivierungsfunktionen
[entnommen aus: Kollmannsberger et al. (2021), S.21]

Als Inspiration für die Einführung einer Aktivierungsfunktion dient die Funktionsweise biologischer Neuronen, die eingehende Informationen allein nach Erreichen eines bestimmten Schwellenwertes weiterleiten. Mathematisch wird dieses Konzept durch die oben links dargestellte Sprungfunktion abgebildet.[25] Die Sprungfunktion erweist sich allerdings nachteilig für das im folgenden Abschnitt dargestellte Training eines neuronalen Netzes. Hierzu besser geeignet sind die stetigen Funktionen der oben rechts geführten Sigmoid-Funktion und des unten links abgebildeten Tangens-Hyperbolicus. Beide Funktionen behalten jedoch die Eigenschaft der Sprungfunktion bei, den Wertebereich auf ein Intervall zwischen 0 und 1 bzw. -1 und 1 einzuschränken.

[24] Vgl. Colliot, *Machine Learning for Brain Disorders*, 84.
[25] Vgl. Sonnet, *Neuronale Netze kompakt*, 28.

Diese Charakteristik den Wertebereich eines Neurons zu begrenzen, kann indessen den Lernprozess eines neuronalen Netzes verlangsamen. Diesem Problem soll die in der Abbildung unten rechts visualisierte ReLU-Funktion entgegenwirken, die für negative Eingabewerte den Wert 0 liefert, während positive Eingabewerte unverändert ausgegeben werden.[26]

2.1.2 Mehrschichtige neuronale Netze

Ein neuronales Netz entsteht nun durch die Verknüpfung einzelner Neuronen, die in mehrere Schichten eingeteilt sind. Grundsätzlich ist jedes neuronale Netz in eine Eingabeschicht, eine Ausgabeschicht und eine beliebige Anzahl an dazwischen befindlichen Schichten – den sogenannten versteckten Schichten – untergliedert. In einem vollständig verknüpften neuronalen Netz ist jedes Neuron einer Schicht mit allen Neuronen der vorangehenden Schicht verbunden. Ein Neuron, das Teil eines neuronalen Netzes ist, erhält demnach als Eingabevektor die Ausgabewerte der Neuronen der vorigen Schicht.[27]

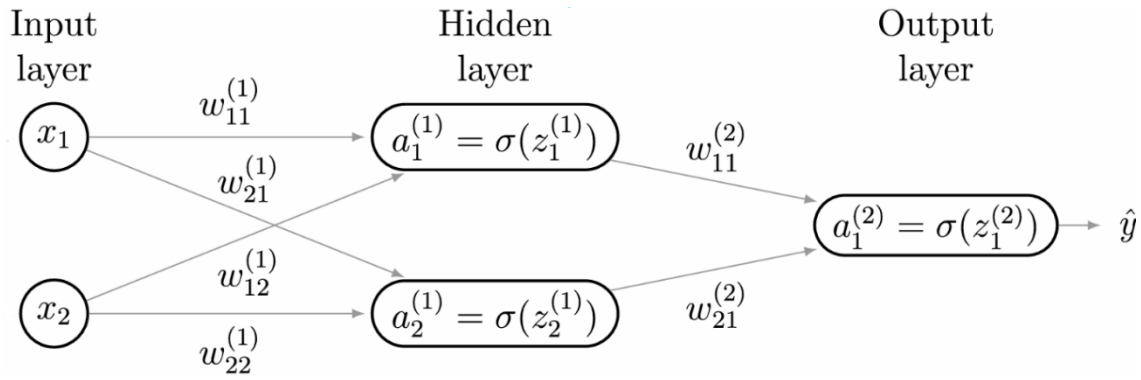

Abbildung 3: Visualisierung eines neuronalen Netzes
[in Anlehnung an: Kollmannsberger et al. (2021), S.23]

Wie zuvor erläutert, ermittelt jedes Neuron aus diesen Eingaben einen Ausgabewert, der wiederum an alle Neuronen der nachfolgenden Schicht weitergeleitet wird. Dieser Prozess wird so lange wiederholt, bis die Werte der Eingabeschicht alle Schichten des Netzes traversiert haben und somit die finalen Ausgangswerte des neuronalen Netzes von der Ausgabeschicht bereitgestellt werden. Die Neuronen der Eingabeschicht nehmen hierbei insofern eine Sonderrolle ein, dass diese lediglich die Eingangswerte des neuronalen Netzes repräsentieren und folglich keine Ausgabewerte anderer

[26] Vgl. Kollmannsberger u. a., *Deep Learning in Computational Mechanics*, 30.
[27] Vgl. Sonnet, *Neuronale Netze kompakt*, 26 f.

Neuronen verarbeiten. In obenstehender Abbildung ist die gerade beschriebene Struktur eines neuronalen Netzes exemplarisch anhand eines neuronalen Netzes mit nur einer versteckten Schicht illustriert.

Es wird ersichtlich, dass zur exakten Beschreibung eines neuronalen Netzes die im vorigen Abschnitt eingeführte mathematische Notation eines Neurons um zusätzliche Informationen ergänzt werden muss. Zum einen ist es nötig, Gewichte und Neuronen mithilfe eines hochgestellten Index so zu markieren, dass diese eindeutig einer Schicht zugeordnet werden können. Zum anderen ist für jedes Gewicht nicht nur der Index des Neurons festzuhalten, mit dessen Ausgabewert das Gewicht multipliziert wird. Darüber hinaus ist anzugeben, welches Neuron der darauffolgenden Schicht die gewichtete Ausgabe als Eingabewert erhält. Hierbei ist zu beachten, dass der Index des Neurons dessen Ausgabewert gewichtet wird, an zweiter Stelle aufgeführt ist. Demzufolge bezeichnet die Schreibweise $w_{jk}^{(l)}$ ein Gewicht der Schicht l, welches den Ausgabewert des Neurons k der vorigen Schicht $l-1$ mit dem Neuron j der Schicht l verknüpft. Werden die Ausgabewerte aller Neuronen einer Schicht ebenfalls in einem Vektor aufgeführt, kann das oben abgebildete neuronale Netz, wie folgt, beschrieben werden. Zunächst ist das Vorgehen zur Berechnung der Ausgabewerte der versteckten Schicht anzugeben, das in untenstehender Gleichung aufgeführt ist.[28]

$$\boldsymbol{z^{(1)}} = \begin{pmatrix} z_1^{(1)} \\ z_2^{(1)} \end{pmatrix} = \begin{pmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \end{pmatrix} = \boldsymbol{W^{(1)} \cdot x + b^{(1)}} \tag{2.3}$$

$$\mathrm{a}^{(1)} = \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \end{pmatrix} = \sigma\left(\mathrm{z}^{(1)}\right) = \sigma\left(\begin{pmatrix} z_1^{(1)} \\ z_2^{(1)} \end{pmatrix}\right) \tag{2.4}$$

Analog dazu wird der Ausgabewert der Ausgabeschicht ermittelt.

$$\boldsymbol{z^{(2)}} = \left(z_1^{(2)}\right) = \begin{pmatrix} w_1^{(2)} \\ w_2^{(2)} \end{pmatrix} \cdot \begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \end{pmatrix} + \left(b_1^{(2)}\right) = \boldsymbol{W^{(2)} \cdot \mathrm{a}^{(1)} + b^{(2)}} \tag{2.5}$$

$$\boldsymbol{a^{(2)}} = \left(a_1^{(2)}\right) = \sigma\left(\boldsymbol{z^{(2)}}\right) = \sigma\left(\left(z_1^{(2)}\right)\right) \tag{2.6}$$

Aus diesem exemplarischen Beispiel lässt sich eine allgemein gültige Gleichung zur Berechnung der Ausgabewerte einer beliebigen Schicht l ableiten.[29]

$$\boldsymbol{a^{(l)}} = \boldsymbol{\sigma\left(z^{(l)}\right)} = \sigma\left(\boldsymbol{W^{(l)} \cdot a^{(l-1)} + b^{(l)}}\right) \tag{2.7}$$

[28] Vgl. Kollmannsberger u. a., *Deep Learning in Computational Mechanics*, 23 f.
[29] Vgl. Nielsen, „Neural Networks and Deep Learning“, 41.

In genereller Form ist die Berechnung der Ausgabe eines neuronalen Netzes schließlich als Verkettung von Ausgabewerten einzelner Schichten zu verstehen, die mithilfe der obigen Gleichung bestimmt werden. Dieser Prozess, bei dem die Eingabewerte des Netzes sukzessive von jeder Schicht des neuronalen Netzes verarbeitet werden, ist unter dem Begriff der Forward Propagation bekannt. Der Einsatz dieser Technik ermöglicht es, jede stetige Funktion beliebig genau zu approximieren.[30]

2.1.3 Optimierung eines neuronalen Netzes

Bisher ist die Struktur eines neuronalen Netzes, sowie das Vorgehen zur Berechnung eines Ausgabewertes bei gegebenen Eingangswerten bekannt. Die Motivation zur Nutzung eines neuronalen Netzes besteht allerdings darin, eine komplexe Problemstellung automatisiert zu lösen, indem das Netz für eine bestimmte Eingabe den gewünschten Ausgabewert ermittelt. Dazu müssen die veränderlichen Parameter des neuronalen Netzes so angepasst werden, dass das Netz fähig ist, den Zusammenhang zwischen Eingabe und angestrebter Ausgabe bestmöglich abzubilden. Hierfür ist es nötig, das bisherige Modell eines neuronalen Netzes um weitere Komponenten und Mechanismen zu ergänzen, die in diesem Teilabschnitt vorgestellt werden.

Um die Fähigkeit eines neuronalen Netzes, Eingabe- auf passende Ausgabewerte abzubilden, zu verbessern, muss die Qualität der Ausgabe des neuronalen Netzes bewertet werden können. Zu diesem Zweck wird die Verlustfunktion L eingeführt. Diese vergleicht die Ausgabe des neuronalen Netzes mit dem Wert, der tatsächlich für die gegebenen Eingabewerte erwartet wird, und bemisst die beobachtete Differenz mit einem numerischen Wert. Die Wahl einer geeigneten Verlustfunktion ist abhängig von der Problemstellung, die das neuronale Netz lösen soll. Für Regressionsprobleme - denen prinzipiell auch die Anwendung von PINNs zuzuordnen ist - wird häufig der mittlere quadratische Fehler als Verlustfunktion eingesetzt. Generell gilt, dass jede differenzierbare Funktion als Verlustfunktion eingesetzt werden kann, wobei die Verlustfunktion mathematisch stets als zusätzliche Verkettung der Ausgabe des neuronalen Netzes zu deuten ist.[31]

[30] Vgl. Colliot, *Machine Learning for Brain Disorders*, 82.
[31] Vgl. Colliot, 87 f.

Die Verlustfunktion bewertet die Qualität eines einzelnen Ausgabewertes des neuronalen Netzes. Für eine generelle Beurteilung des Netzes muss jedoch die Fähigkeit des Netzes gemessen werden, für verschiedene Eingabedaten die erwartete Ausgabe zu bestimmen. Folglich wird zur Messung der Qualität eines neuronalen Netzes die Kostenfunktion C als Mittelwert der Ergebnisse der Verlustfunktion definiert. In untenstehender Gleichung bezeichnet dabei N die Anzahl der verschiedenen Trainingsbeispiele, $\boldsymbol{y_i}$ den Vektor der erwarteten Ausgabewerte für eine zusammengehörige Menge an Eingabedaten und $\boldsymbol{a_i^{(l)}}$ die tatsächlich vom neuronalen Netz ausgegebenen Werte.[32]

$$C = \frac{1}{N} \cdot \sum_{i=0}^{N} L\left(\boldsymbol{y_i}, \boldsymbol{a_i^{(l)}}\right) \tag{2.8}$$

Durch die Definition der Kostenfunktion ist die Fähigkeit eines neuronalen Netzes quantitativ messbar. Somit kann das Vorgehen zur Verbesserung des Netzes als Optimierungsproblem formuliert werden. Ziel der Optimierung ist es, den Wert der Kostenfunktion zu minimieren, indem Gewichte und Biaswerte des neuronalen Netzes in einem iterativen Prozess bestmöglich angepasst werden.[33]

Für die Lösung von Optimierungsaufgaben existieren zahlreiche Algorithmen. Im Umfeld neuronaler Netze werden zumeist Methoden eingesetzt, die auf der Idee des Gradientenverfahren basieren. Hierbei wird das Wissen genutzt, dass der Gradient einer Funktion mehrerer Veränderlicher als Richtung der größten Steigung der Funktion interpretiert werden kann. Demzufolge ergibt die Umkehrung des Vorzeichens des Gradienten die Richtung an, in der der Wert einer Funktion am stärksten abnimmt. Nachdem die Kostenfunktion eines neuronalen Netzes von dessen Gewichten und Biaswerten abhängig ist, enthält der Gradient der Kostenfunktion alle partiellen Ableitungen der Kostenfunktion nach jedem einzelnen dieser Gewichte und Biaswerte. Zur Minimierung der Kostenfunktion werden schließlich die Gewichte des Netzes durch iterative Schritte in Richtung des negativen Gradienten aktualisiert. Zur Präzisierung lässt sich diese Methodik anhand der unten aufgeführten Gleichung beschreiben.[34]

[32] Vgl. Colliot, 87.
[33] Vgl. Kollmannsberger u. a., *Deep Learning in Computational Mechanics*, 25.
[34] Vgl. Colliot, *Machine Learning for Brain Disorders*, 89 f.

$$\boldsymbol{W^{t+1}} \leftarrow \boldsymbol{W^{t}} - \eta \cdot \nabla C(\boldsymbol{W^{t}}) \quad \text{mit } \nabla C(\boldsymbol{W^{t}}) = \begin{pmatrix} \frac{\partial C}{\partial w_{11}^{(1)}} \\ \frac{\partial C}{\partial b_{1}^{(1)}} \\ \cdots \\ \frac{\partial C}{\partial w_{kj}^{(l)}} \\ \frac{\partial C}{\partial b_{k}^{(l)}} \end{pmatrix} \tag{2.9}$$

In dieser Formel bezeichnet $\boldsymbol{W^{t}}$ die aktuelle Belegung aller Gewichte und Biaswerte des neuronalen Netzes, während der Vektor $\boldsymbol{W^{t+1}}$ die aktualisierten Parameter repräsentiert. Zudem wird aus 2.9 ersichtlich, dass der Erfolg der Optimierung von einem zusätzlichen, frei wählbaren Hyperparameter abhängig ist. So legt die durch den Buchstaben η symbolisierte Lernrate die Schrittweite eines einzelnen Optimierungsschrittes fest, und beeinflusst somit maßgeblich Geschwindigkeit und Ergebnis des Algorithmus.

Zur Berechnung des Gradienten der Kostenfunktion wird üblicherweise eine spezielle Variante der automatisierten Differentiation eingesetzt, die unter der Bezeichnung Backpropagation bekannt ist.[35] Zu beachten ist, dass sich der Gradient der Kostenfunktion aus dem Mittelwert des für jedes Trainingsbeispiel separat ermittelten Gradienten der Verlustfunktion ergibt. Zur Berechnung des Gradienten der Kostenfunktion muss also zunächst der Gradient der Verlustfunktion für jedes Trainingsbeispiel bestimmt werden. Der sequenzielle Ablauf aus Forward Propagation, Berechnung der Gradienten und Anpassung der Gewichte für alle Trainingsbeispiele wird im Umfeld des maschinellen Lernens als Epoche bezeichnet.[36]

Grundlegend für das Verständnis der Backpropagation ist die Betrachtung eines neuronalen Netzes als Verkettung einzelner elementarer Funktionen. Die Differentiation der Verlustfunktion kann demzufolge durch Anwendung der Kettenregel aus der multivariablen Analysis gelöst werden. Beispielhaft kann das Vorgehen zur Berechnung der partiellen Ableitung anhand des neuronalen Netzes aus Abbildung 3

[35] Vgl. Baydin u. a., „Automatic differentiation in machine learning“, 12.
[36] Vgl. Kollmannsberger u. a., *Deep Learning in Computational Mechanics*, 30 f.

gezeigt werden. So kann die partielle Ableitung der Gewichte der letzten Schicht nach einer allgemeinen Verlustfunktion L wie folgt ermittelt werden.[37]

$$\frac{\partial L}{\partial \boldsymbol{W}^{(2)}} = \begin{pmatrix} \frac{\partial L}{\partial w_{11}^{(2)}} \\ \frac{\partial L}{\partial w_{21}^{(2)}} \end{pmatrix} = \begin{pmatrix} \frac{\partial L}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{11}^{(2)}} \\ \frac{\partial L}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \cdot \frac{\partial z_1^{(2)}}{\partial w_{21}^{(2)}} \end{pmatrix} = \begin{pmatrix} \frac{\partial L}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \cdot a_1^{(1)} \\ \frac{\partial L}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \cdot a_2^{(1)} \end{pmatrix} = \boldsymbol{v} \cdot (\mathbf{a^1})^{\mathrm{T}}$$

$$\text{mit } \boldsymbol{v} = \begin{pmatrix} \frac{\partial L}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \\ \frac{\partial L}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \end{pmatrix} \tag{2.10}$$

Aus obenstehender Gleichung wird ersichtlich, dass zur Berechnung der partiellen Ableitungen, Werte wiederverwendet werden können, die bereits im Rahmen der Forward Propagation berechnet werden. Genauso können partielle Ableitungen, die für die Gewichte und Biaswerte der Schichten am Ende des Netzwerks berechnet werden, bei der Bestimmung partieller Ableitungen in davorliegenden Schichten eingesetzt werden. Aus diesem Grund ist es möglich, die Berechnung aller partiellen Ableitungen anhand von vier allgemeinen Gleichungen zu beschreiben. Die Verwendung des Hadamard-Produkts ⊙ zeigt dabei in untenstehenden Gleichungen an, dass zwei Vektoren elementweise miteinander multipliziert werden.[38]

$$\frac{\partial L}{\partial \boldsymbol{W}^{(l)}} = \frac{\partial L}{\partial \boldsymbol{a}^{(l)}} \odot \frac{\partial \boldsymbol{a}^{(l)}}{\partial \boldsymbol{z}^{(l)}} \cdot \left(\boldsymbol{a}^{(l-1)}\right)^T \tag{2.11}$$

$$\frac{\partial L}{\partial \boldsymbol{b}^{(l)}} = \frac{\partial L}{\partial \boldsymbol{a}^{(l)}} \odot \frac{\partial \boldsymbol{a}^{(l)}}{\partial \boldsymbol{z}^{(l)}} \tag{2.12}$$

$$\frac{\partial L}{\partial \boldsymbol{a}^{(l-1)}} = \left(\boldsymbol{W}^{(l)}\right)^T \cdot \frac{\partial L}{\partial \boldsymbol{a}^{(l)}} \odot \frac{\partial \boldsymbol{a}^{(l)}}{\partial \boldsymbol{z}^{(l)}} \tag{2.13}$$

$$\frac{\partial L}{\partial \boldsymbol{W}^{(l-1)}} = \frac{\partial L}{\partial \boldsymbol{a}^{(l-1)}} \odot \frac{\partial \boldsymbol{a}^{(l-1)}}{\partial \boldsymbol{z}^{(l-1)}} \cdot \left(\boldsymbol{a}^{(l-2)}\right)^T \tag{2.14}$$

Um nun die partiellen Ableitungen der Verlustfunktion für Gewichte und Biaswerte in tieferliegenden Schichten des neuronalen Netzes zu berechnen, sind zunächst in einem iterativen Prozess mithilfe von Gleichung 2.13 die partiellen Ableitungen für die Ausgabewerte der entsprechenden Schicht zu berechnen. Diese können, wie 2.14

[37] Vgl. Kollmannsberger u. a., 27 f.
[38] Vgl. Nielsen, „Neural Networks and Deep Learning“, 47.

zeigt, wiederum dazu verwendet werden die partiellen Ableitungen der Verlustfunktion für die Gewichte dieser Schicht zu bestimmen. Analog dazu können die partiellen Ableitungen für die Biaswerte der jeweiligen Schicht ermittelt werden.

2.2 Physics Informed Neural Networks

Wie zu Beginn der Arbeit erwähnt, ist es die definierende Charakteristik eines Physics Informed Neural Networks, die einer Anwendung zugrundeliegenden physikalischen Gesetzmäßigkeiten in die Struktur eines neuronalen Netzes zu integrieren. PINNs sind, wie deren Urheber Raissi et al. zeigen, dazu in der Lage, beliebige physikalische Prinzipien in das Training einzubeziehen. Voraussetzung dafür ist, dass die physikalischen Gesetzmäßigkeiten in Form von analytischen Ausdrücken, meist mithilfe von Differentialgleichungen, beschrieben werden können.[39]

Das Ziel beim Training eines PINNs zur Anwendung auf Simulationsprobleme besteht demnach darin, eine Lösung für ein System an gewöhnlichen oder partiellen Differentialgleichungen zu ermitteln. Bei einer inversen Problemstellung ist es hingegen die Aufgabe eines PINNs diejenigen Parameter eines Differentialgleichungssystems zu identifizieren, die die am besten zu einer Menge an Messwerten passende Lösung des Differentialgleichungs-Systems (DGL-System) hervorbringen.[40] In allgemeiner Form kann ein derartiges System an Differentialgleichungen, wie folgt, formuliert werden.[41]

$$\mathcal{D}[\boldsymbol{u}(x);\lambda] = \mathbf{g}(x) \tag{2.15}$$

Hierbei bezeichnet $\mathcal{D}$ einen beliebigen Differentialoperator, $\mathbf{u}(x)$ die Lösungen der Differentialgleichungen, λ eine Menge an Parametern und $\mathbf{g}(x)$ einen allgemeinen Störterm. Für die Darstellung der Funktionsweise eines PINNs ist es jedoch hinreichend ein generisches System an gewöhnlichen Differentialgleichungen erster Ordnung zur betrachten. Die Grundprinzipien eines PINNs, die anhand dieser nachfolgend aufgeführten Vereinfachung deutlich werden, können analog auf beliebige Differentialgleichungen übertragen werden.

[39] Vgl. Raissi, Perdikaris, und Karniadakis, „Physics-informed neural networks“, 696.
[40] Vgl. Raissi, Perdikaris, und Karniadakis, 687.
[41] Vgl. Moseley, „Physics-Informed Machine Learning“, 18.

Allgemein kann ein Differentialgleichungssystem erster Ordnung in untenstehender Form dargestellt werden, bei der die höchste Ableitung auf einer Seite der Gleichung isoliert ist.[42]

$$\frac{d}{dt}\boldsymbol{u} = \begin{pmatrix} \frac{du_1}{dt} \\ \cdots \\ \frac{du_n}{dt} \end{pmatrix} = \mathrm{f}(\boldsymbol{u}, t; \lambda) = \begin{pmatrix} f(u_1, \ldots, u_n, t; \lambda) \\ \cdots \\ f(u_1, \ldots, u_n, t; \lambda) \end{pmatrix} \tag{2.16}$$

Notwendigerweise sind in einem DGL-System erster Ordnung die ersten Ableitungen der unbekannten Funktionen $u_1(t), \ldots, u_n(t)$ enthalten, die in einem Vektor kodiert sind. Der Term $\mathrm{f}(\boldsymbol{u}, t; \lambda)$ beschreibt, dass zusätzlich die Funktion selbst, sowie die unabhängige Variable t in jeder der Differentialgleichungen des Systems auftreten können. Zudem kann in jeder Gleichung eine beliebige Zahl an konstanten Parametern λ vorkommen. Unabhängig davon ob die verschiedenen Elemente der einzelnen Differentialgleichungen linear oder nicht-linear miteinander verknüpft sind, ist jede Funktion $u_i(t)$ eine Lösung der i-ten DGL des Systems, die die jeweilige DGL – in Kombination mit der entsprechenden Ableitung - erfüllt.[43] Damit geht einher, dass die DGL auch für jeden spezifischen Eingabewert $t_i \in D_u$ erfüllt ist, wobei D_u die Definitionsmenge der Funktion $u_i(t)$ bezeichnet.

2.2.1 PINNs zur Lösung von Simulationsproblemen

Zunächst wird der Aufbau eines PINNs zur Lösung einer Simulationsaufgabe veranschaulicht. Dabei kann das in obigem Abschnitt entwickelte Wissen analog auf die zuvor beschriebene Struktur eines neuronalen Netzes angewendet werden. Vorbedingung für die Bestimmung der speziellen Lösung eines DGL-Systems ist zum einen die Angabe von Anfangs- oder Randwerten. Zum anderen müssen alle Parameter des Systems auf einen bestimmten Wert festgesetzt sein.[44] Nachdem somit jeder Wert von λ bekannt ist, kann auf die explizite Angabe von λ zur übersichtlicheren Darstellung nachfolgend verzichtet werden.

Als Eingabewert des neuronalen Netzes wird nun ein beliebiger Wert der unabhängigen Variable t gewählt. Die Ausgabewerte des neuronalen Netzes können hingegen als Approximation der Funktionswerte der unbekannten Funktionen

[42] Vgl. Strogatz, *Nonlinear Dynamics and Chaos*, 15.
[43] Vgl. Papula, *Mathematik für Ingenieure und Naturwissenschaftler Band 2*, 345.
[44] Vgl. Kollmannsberger u. a., *Deep Learning in Computational Mechanics*, 56 f.

$u_1(t_i), \dots, u_n(t_i)$ interpretiert werden. Generell gilt damit folgender Zusammenhang, wobei der Ausdruck $\hat{\boldsymbol{u}}_{\square\square\square\square}$ den Ausgabewerten des Netzes entspricht.[45]

$$\boldsymbol{a}^{(l)} = \hat{\boldsymbol{u}}_{PINN}(t_i) \approx \boldsymbol{u}(t_i) \tag{2.17}$$

Das neuronale Netz muss folglich darauf optimiert werden, die Eingabewerte t so abzubilden, dass die resultierenden Ausgaben die Differentialgleichungen des vorgegebenen DGL-Systems bestmöglich erfüllen. Wie in Abschnitt 2.3 erläutert, dient die Verlustfunktion dazu, die Qualität der Ausgabe eines neuronalen Netzes zu quantitativ zu bewerten. Demnach ist es für die Lösung eines DGL-Systems konstruktiv, die einzelnen Differentialgleichungen des Systems in die Verlustfunktion zu integrieren. Dieser Einbezug eines physikalisch begründeten DGL-Systems in die Verlustfunktion eines neuronalen Netzes ist die Charakteristik eines PINNs, die es vom üblichen Aufbau eines neuronalen Netzes unterscheidet.

Zur Konstruktion einer passenden Verlustfunktion wird die in Gleichung 2.17 vorgestellte Deutung des Ausgabewertes des Netzes als Lösung des DGL-Systems genutzt. Nachdem $\hat{\boldsymbol{u}}_{\square\square\square\square}$ die tatsächliche Lösung des Systems möglichst exakt approximieren soll, kann der Ausgabewert des neuronalen Netzes $\hat{\boldsymbol{u}}_{\square\square\square\square}$ anstelle von $\boldsymbol{u}$ in das DGL-System eingesetzt werden. Falls $\hat{\boldsymbol{u}}_{\square\square\square\square}$ dem gesuchten Wert von $\boldsymbol{u}$ entspricht, sind beide Seiten der in 2.16 formulierten allgemeinen Gleichung eines DGL-Systems identisch. Ausgabewerte des PINNs, die $\boldsymbol{u}$ besser approximieren, führen folglich zu einer geringeren Differenz zwischen linkem und rechtem Term der generischen Gleichung des DGL-Systems. Unter Berücksichtigung dieser Überlegungen kann folgende allgemeine Verlustfunktion für PINNs definiert werden.[46]

$$L_{\text{PHY}} = \left| \frac{d}{dt}\hat{\boldsymbol{u}}_{\text{PINN}} - f(\hat{\boldsymbol{u}}_{\text{PINN}}, t_i) \right|^2 = \sum_{j=0}^{N} \left(\frac{d}{dt}\hat{u}_{PINN,j} - f(\hat{u}_{PINN}, t_i)_j \right)^2 \tag{2.18}$$

Für die Berechnung der Ableitung der Ausgabewerte des PINNs $\hat{\boldsymbol{u}}_{\square\square\square\square}$ nach einem Eingabewert $t_{\square}$ kann dabei die in Abschnitt 2.1.3 erläuterte Methodik der Backpropagation eingesetzt werden. Treten in einem DGL-System Ableitungen höheren Grades auf, so ist der resultierende Ausgabewert iterativ mithilfe von Methoden der automatischen Differentiation zu ermitteln.[47]

[45] Vgl. Neuer, *Maschinelles Lernen für die Ingenieurwissenschaften*, 199.
[46] Vgl. Neuer, 199.
[47] Vgl. Kollmannsberger u. a., *Deep Learning in Computational Mechanics*, 57 f.

Neben dem in 2.18 aufgeführten physikalisch begründeten Verlustterm $L_{\square\square\square}$ muss die Verlustfunktion eines PINNs um einen Term erweitert werden, der die Berücksichtigung von Anfangs- oder Randwerten des DGL-Systems einfordert. Ohne diesen zusätzlichen Verlustterm, der üblicherweise unter der Bezeichnung $L_{\square\square}$ geführt wird, kann die spezielle Lösung des DGL-Systems für einen konkreten physikalischen Anwendungsfall nicht bestimmt werden. Die Definition von diesem Verlustterm ist nachstehender Gleichung zu entnehmen.[48]

$$L_{\square\square} = |\hat{\boldsymbol{u}}_{PINN}(t_0) - \boldsymbol{u}(t_0)|^2 = \sum_{i=0}^{N} \left(\hat{u}_{PINN,i}\,(t_0) - u_i(t_0)\right)^2 \qquad (2.19)$$

Dabei bezeichnet $u_{\square}(t_0)$ den Anfangswert, der einer DGL für den spezifischen Eingabewert t_0 zugeordnet ist. Währenddessen benennt $\hat{u}_{PINN,i}\,(t_0)$ den entsprechenden Ausgangswert des PINNs für ebendiesen Eingabewert.

Liegen für ein physikalisches System Messdaten vor, so können diese ebenso in den Verlustterm integriert werden. Wie bereits in Abschnitt 2.1.3 erwähnt, wird hierbei meist der mittlere quadratische Fehler zur Bewertung der Ausgabe des PINNs hinsichtlich der vorliegenden Messdaten eingesetzt.

Sind keine Messdaten vorhanden, so werden ausschließlich zufällig gewählte Werte der unabhängigen Variablen t als Eingabedaten verwendet. Diese sogenannten Kollokationspunkte bilden in diesem Fall die Gesamtmenge an Trainingsbeispielen eines PINNs, die in einem Optimierungsalgorithmus zur Verbesserung des PINNs genutzt werden. Hierbei ist explizit anzumerken, dass es für das Training eines PINNs nicht nötig ist den Wert $u(t)$ vorzugeben, auf den ein Eingabewert t abgebildet werden soll. Diese Zuordnung soll allein durch den Einbezug der entsprechenden Differentialgleichung durch das PINN erfasst werden können.[49] [50]

2.2.2 PINNs zur Lösung inverser Probleme

Der Aufbau eines PINNs zur Bearbeitung inverser Problemstellungen greift das im letzten Abschnitt präsentierte Prinzip zur Bestimmung der Lösung eines DGL-Systems in nahezu identischer Form auf. So werden genauso, wie gerade beschrieben, Werte der unabhängigen Variablen t als Eingabewerte eines neuronalen Netzes verwendet.

[48] Vgl. Moseley, „Physics-Informed Machine Learning“, 33.
[49] Vgl. Kollmannsberger u. a., *Deep Learning in Computational Mechanics*, 56 f.
[50] Vgl. Moseley, „Physics-Informed Machine Learning“, 34.

Damit kann die Ausgabe des neuronalen Netzes wiederum als spezielle Lösung eines gegebenen DGL-Systems aufgefasst werden. Der Unterschied zur Lösung von Simulationsaufgaben ist unterdessen darin begründet, dass bei der Inversion die Parameter λ zumindest teilweise nicht bekannt sind. Stattdessen steht eine Menge an Messdaten zur Verfügung, anhand derer die gesuchten Parameter ermittelt werden sollen. Das PINN wird demzufolge darauf optimiert, diejenigen Parameter zu bestimmen, die in Kombination mit der ebenso in diesem Prozess zu ermittelnden Lösung des DGL-Systems bestmöglich mit den Messwerten übereinstimmen.[51]

Es ist demnach erforderlich, die Parameter des Systems in den Optimierungsprozess zu integrieren. Zur Erläuterung der hierzu eingesetzten Methodik muss die mathematische Beschreibung der Verlustfunktion angepasst werden. Nachdem die Belegung der Parameter λ im Gegensatz zum vorigen Abschnitt nicht vollständig bekannt ist, ist die Abhängigkeit des DGL-Systems von diesen unbekannten Parametern kenntlich zu machen. Damit gilt für den physikalisch begründeten Verlustterm L_{PHY}:

$$L_{\square\square\square} = \left| \frac{d}{dt} \widehat{\boldsymbol{u}}_{PINN} - f(\widehat{\boldsymbol{u}}_{PINN}, t_i; \lambda) \right|^2 \qquad (2.20)$$

Es ist ersichtlich, dass die Ausgabe des PINNs $\widehat{\boldsymbol{u}}_{\square\square\square\square}$ nur indirekt im Rahmen des Optimierungsprozesses von den Parametern des DGL-Systems abhängig ist. Der Subtrahend $f(\widehat{\boldsymbol{u}}_{\square\square\square\square})$ in obenstehender Verlustfunktion, der sich durch Einsetzen des Ausgabewertes in das DGL-System ergibt, wird allerdings offensichtlich direkt von den Parametern des DGL-Systems beeinflusst. Entscheidend bei der Entwicklung eines PINNs für inverse Probleme ist es nun, diese Parameter ebenso als veränderliche Werte zu betrachten, wie die restlichen Gewichte und Biaswerte des neuronalen Netzes. Dementsprechend werden die zu identifizierenden Parameter initial genauso, wie Gewichte und Schwellenwerte, mit einem zufällig gewählten Wert belegt. Nachdem die Verlustfunktion – und damit auch die Kostenfunktion – des PINNs von λ abhängig ist, ist auch die partielle Ableitung der Kostenfunktion nach λ ein Bestandteil des Gradienten der Kostenfunktion. Somit werden bei der Optimierung der Gewichte des PINNs die unbekannten Parameter des DGL-Systems ebenso schrittweise aktualisiert mit dem Ziel die Kostenfunktion zu minimieren.

[51] Vgl. Kollmannsberger u. a., *Deep Learning in Computational Mechanics*, 77.

Die Berechnung der partiellen Ableitung $\frac{\partial \square}{\partial \lambda}$ erfolgt dabei nach den Methoden der automatischen Differentiation, üblicherweise mithilfe des in Kapitel 2.1.3 dargelegten Backpropagation-Algorithmus. Zur vereinfachten Darstellung bezeichnet λ in diesem Kontext nur einen einzelnen Parameter.[52]

Da der Wert des physikalischen Verlustterms, wie gerade gezeigt, direkt von den unbekannten Parametern des DGL-Systems abhängig ist, ist der physikalische Verlustterm allein nicht ausreichend, um die Qualität eines Ausgabewertes des PINNS messen zu können. Zur adäquaten Bestimmung der Qualität eines Ausgabewertes des PINNs - die Grundvoraussetzung für die Verbesserung des PINNs im Trainingsprozess - muss die Verlustfunktion um einen zusätzlichen Term ergänzt werden. Dieser Verlustterm gleicht die von einem PINN ausgegebene Lösung mit tatsächlich gemessenen oder anderweitig ermittelten Messdaten ab und wird daher als datenbasierter Verlustterm $L_{\square\square\square\square}$ bezeichnet. Genauso, wie bei der Integration von Messdaten beim Simulationsproblem, wird ein solcher Verlustterm für gewöhnlich über den mittleren quadratischen Fehler berechnet. Die vollständige Verlustfunktion eines PINNs zur Lösung inverser Probleme setzt sich schließlich aus der Summe der beiden einzelnen Verlustterme $L_{\square\square\square}$ und $L_{\square\square\square\square}$ zusammen. Folglich besteht die Gesamtmenge an Trainingsdaten für derartige PINNs sowohl aus beliebig gewählten Kollokationspunkten als auch aus den gemessenen Daten, wobei jeder Datenpunkt als Paar eines Eingabewert $t_{\square}$ und eines Messwertes $u_{DATA,i}$ zu verstehen ist.[53]

2.2.3 Abgrenzung von anderen Methoden des Physics Informed Machine Learnings

PINNs sind strikt von anderen Methoden des Physics Informed Machine Learning zu unterscheiden. Unter dem Begriff des Physics Informed Machine Learning werden dabei all diejenigen Lerntechniken zusammengefasst, die Vorwissen über die physikalischen Eigenschaften der Eingabedaten nutzen. Grundsätzlich ist dabei zwischen Methoden zu differenzieren, die darauf basieren, die Eingabedaten eines Lernalgorithmus anzureichern, und Techniken, die das vorhandene Vorwissen in den Algorithmus selbst integrieren.[54]

---

[52] Vgl. Raissi, Perdikaris, und Karniadakis, „Physics-informed neural networks“, 693.
[53] Vgl. Moseley, „Physics-Informed Machine Learning“, 34.
[54] Vgl. Neuer, *Maschinelles Lernen für die Ingenieurwissenschaften*, 186 ff.

Bei der physikalisch-informierten Datenanreicherung werden einem Lernalgorithmus zusätzliche Informationen bereitgestellt, die nicht unmittelbar in den Eingabedaten erkennbar sind. Diese Informationen müssen vom Entwickler eines solchen Lernverfahrens aus den physikalischen Zusammenhängen, die den Daten zugrunde liegen, abgeleitet werden. Beispielsweise kann es sinnvoll sein, zeitabhängige Daten um deren Darstellung im Frequenzbereich zu erweitern.[55]

Derartigen Methoden der Datenanreicherung sind Verfahren gegenüberzustellen, die durch die Integration von physikalischem Vorwissen in den Lernalgorithmus gekennzeichnet sind. Hierbei besteht eine Möglichkeit darin, die Struktur des Lernalgorithmus so zu verändern, dass physikalisch begründete Nebenbedingungen während des Lernprozesses stets eingehalten werden müssen.[56] [57] Im Gegensatz zu solchen Verfahren, die die uneingeschränkte Einhaltung der physikalischen Einschränkungen einfordern, wird das physikalische Vorwissen bei Methoden der Physics-Informed Regularization lediglich dazu verwendet, den Lernprozess zu optimieren. Die Einbindung bekannter physikalischen Gesetzmäßigkeiten soll den Lernalgorithmus dabei unterstützen eine Lösung zu ermitteln, die sowohl mit erfassten Beobachtungswerten als auch mit den physikalischen Kenntnissen über ein System übereinstimmt. Dieser Kategorie sind PINNs zuzuordnen.[58] [59]

2.3 Chaotische Systeme

Nachdem im Rahmen dieser Arbeit die Fähigkeiten von PINNs zur Analyse chaotischer Systeme untersucht werden sollen, ist es nötig, in diesem letzten Kapitelabschnitt ein grundlegendes Verständnis für den Themenbereich der chaotischen Systeme zu vermitteln. Hierzu sind zunächst generelle Methoden zur Beschreibung dynamischer Systeme aufzuzeigen. Darauf aufbauend werden die charakteristischen Eigenschaften chaotischer Systeme benannt. Abschließend sind Verfahren zur numerischen Lösung dynamischer Systeme zu erläutern.

[55] Vgl. Neuer, 191.
[56] Vgl. Meng u. a., „When Physics Meets Machine Learning“, 12 f.
[57] Vgl. Moseley, „Physics-Informed Machine Learning“, 22.
[58] Vgl. Wu, Sicard, und Gadsden, „Physics-informed machine learning“, 13.
[59] Vgl. Moseley, „Physics-Informed Machine Learning“, 27 ff.

2.3.1 Dynamische Systeme

Das Themenfeld der Dynamik behandelt die Beschreibung von Systemen, deren Zustand sich in Abhängigkeit von der Zeit ändert. Derartige Systeme lassen sich mathematisch mithilfe von Differentialgleichungen modellieren, wie sie in Gleichung 2.15 notiert sind. Genauso wie zur Darstellung der Funktionsweise von PINNs ist jedoch das generische System von gewöhnlichen Differentialgleichungen aus Gleichung 2.16 auch für die Einführung in chaotische Systeme ausreichend. Dies ist insbesondere dadurch begründet, dass jede DGL der Ordnung n in ein System an n Differentialgleichungen erster Ordnung überführt werden kann. Dabei ist die unabhängige Variable t im Kontext dynamischer Systeme als Kodierung der Zeit zu verstehen. Tritt t explizit in einer der Differentialgleichungen auf, so ist das System durch Einführung einer weiteren Variable $u_{n+1} = \mathrm{t}$ zu vereinfachen. Dadurch kann die Abhängigkeit der Funktion f von der Zeit aufgelöst werden, sodass sich folgende generische Repräsentation dynamischer Systeme ergibt.[60]

$$\frac{d}{dt}\boldsymbol{u} = \begin{pmatrix} \frac{du_1}{dt} \\ \cdots \\ \frac{du_n}{dt} \\ \frac{du_{n+1}}{dt} \end{pmatrix} = \mathrm{f}(\boldsymbol{u}) = \begin{pmatrix} f_1(u_1, \ldots, u_n, u_{n+1}) \\ \cdots \\ f_n(u_1, \ldots, u_n, u_{n+1}) \\ \frac{dt}{dt} = 1 \end{pmatrix} \tag{2.21}$$

Enthält jede der Differentialgleichungen $f_1, \ldots, f_n$ ausschließlich linear verknüpfte Terme, so kann das DGL-System mithilfe einfacher Algorithmen gelöst werden. Sobald jedoch nicht-lineare Terme, wie Potenzen oder trigonometrische Funktionen, im DGL-System vorkommen, ist die analytische Lösung des Systems nur mit großem Aufwand oder gar nicht möglich. Daher werden graphische Methoden dazu eingesetzt, die relevanten Eigenschaften eines solchen Systems qualitativ zu analysieren.[61]

Grundlage der qualitativen Analyse ist das Richtungsfeld des DGL-Systems. Hierbei wird die Struktur des expliziten DGL-Systems ausgenutzt, durch die für einen bestimmten Zustand $u_1, \ldots, u_n$ vorgegeben ist, wie die entsprechende Ableitung zu berechnen ist. Ein Zustand wird dazu als Punkt im sogenannten Phasenraum interpretiert. Jedem dieser Punkte kann nun über die Funktion $f(\boldsymbol{u})$ der Vektor $\frac{\square}{dt}\boldsymbol{u}$ an Ableitungen zugeordnet werden. Dieser gibt für jede Dimension des Phasenraumes

[60] Vgl. Argyris u. a., *Die Erforschung des Chaos*, 27.
[61] Vgl. Strogatz, *Nonlinear Dynamics and Chaos*, 16 f.

die Änderungsrate der jeweiligen Variable an. Setzt man nun an einem beliebigen Punkt im Phasenraum an, so ist es lediglich nötig, den einzelnen Vektoren zu folgen, um eine Lösung des DGL-Systems abschätzen zu können. Eine auf diese Weise ermittelte spezielle Lösung des DGL-Systems wird unter der Bezeichnung Bahnkurve oder Trajektorie geführt. Zur generellen Beschreibung des DGL-Systems werden diejenigen Bahnkurven graphisch veranschaulicht, anhand derer die qualitativen Eigenschaften des Systems vollumfänglich deutlich werden. Diese Darstellungsform wird als Phasenportrait bezeichnet und ist in untenstehender Abbildung beispielhaft für den zweidimensionalen Phasenraum mit $u_1 = x$ und $u_2 = y$ visualisiert.[62] [63]

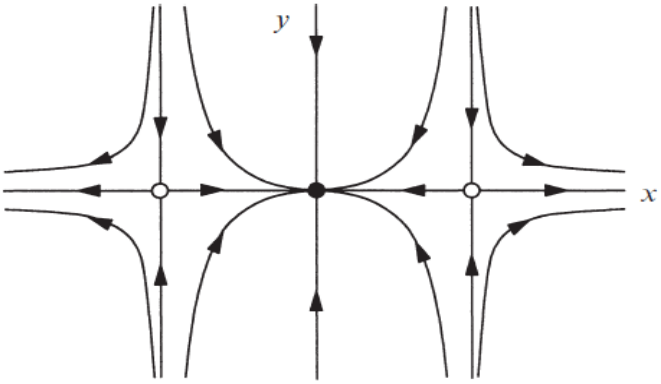

Abbildung 4: Beispielhafte Darstellung eines Phasenportraits

[entnommen aus: Strogatz S. (2015), S.153]

Das Phasenportrait erlaubt es, Aussagen über das langfristige Verhalten des Systems zu treffen. So bewirkt etwa die Wahl eines Startpunkts an einer beliebigen Stelle der y-Achse, dass das System sich langfristig im Gleichgewichtszustand mit den Werten $y = 0,\ x = 0$ befindet. Ein solcher Zustand, auf den die Lösung des DGL-Systems langfristig zuläuft, wird als stabiler Fixpunkt bezeichnet. Demgegenüber stehen die weiß markierten Sattelpunkte, die in einer Dimension identisch zu Fixpunkten fungieren. In der anderen Dimension entfernen sich die Lösungen des DGL-Systems jedoch von diesem Punkt und entwickeln sich entweder in Richtung des anderen Fixpunktes oder in die Unendlichkeit. Wie das Beispiel zeigt, ist es möglich aus dem Phasenportrait Kenntnisse über die Stabilität des Systems abzuleiten. Mittels derselben Methodik kann somit ermittelt werden, ob die Lösungen eines DGL-Systems periodisches oder gar chaotisches Verhalten aufweisen.[64]

[62] Vgl. Argyris u. a., *Die Erforschung des Chaos*, 32.
[63] Vgl. Weitz, *Konkrete Mathematik (nicht nur) für Informatiker*, 738.
[64] Vgl. Strogatz, *Nonlinear Dynamics and Chaos*, 18 f.

Zum Abschluss dieser Einführung in das Themenfeld dynamischer Systeme ist anzumerken, dass die Struktur des Richtungsfeldes eines DGL-Systems von der Belegung der Parameter abhängig ist. Werden die Parameter des DGL-Systems variiert, so ändern sich auch die qualitativen Eigenschaften des Richtungsfeldes. Damit ist es möglich, dass Fixpunkte, Periodizität oder chaotisches Verhalten im selben System durch Variation der Parameter entstehen oder auch verschwinden können. Diejenigen Parameterwerte, an denen eine solche qualitative Änderung des Richtungsfeldes zu beobachten ist, werden Bifurkationspunkte genannt.[65]

2.3.2 Eigenschaften chaotischer Systeme

Der Begriff des chaotischen Systems bezeichnet eine spezielle Ausprägung dynamischer Systeme. Demzufolge gelten die im letzten Abschnitt präsentierten Formalismen, Methoden und Merkmale dynamischer Systeme, genauso für chaotische Systeme. Damit ein dynamisches System chaotisch genannt werden kann, müssen nach Strogatz die nachfolgend aufgeführten Bedingungen erfüllt sein.[66]

Das erste Kriterium, das für die Einordnung als chaotisches System zu erfüllen ist, besteht darin, dass das langfristige Verhalten des Systems durch Aperiodizität gekennzeichnet ist. Die Forderung nach aperiodischem Verhalten schließt nicht nur periodische Lösungen des Systems und die Existenz von Fixpunkten aus. Zusätzlich dürfen Bahnkurven eines Systems, das aperiodisches Verhalten aufweist, nicht ins Unendliche streben, nachdem dies analog zu einem Fixpunkt aufzufassen ist. Bei Betrachtung im Phasenraum bewegen sich Bahnkurven chaotischen Verhaltens demnach stets in einem abgegrenzten Bereich. Zudem muss die Wahrscheinlichkeit für chaotisches Verhalten bei Wahl einer beliebigen Anfangsbedingung positiv sein. Diese Bedingung gewährleistet, dass chaotisches Verhalten nicht nur in Ausnahmefällen zu beobachten ist.[67]

Die zweite Bedingung, der ein chaotisches System genügen muss, liegt in der Einschränkung, dass das DGL-System keine stochastischen Anteile enthalten darf. Das unregelmäßige Verhalten des Systems muss folglich allein aus den deterministischen Gesetzmäßigkeiten resultieren, die zur Beschreibung der Dynamik

[65] Vgl. Strogatz, 45.
[66] Vgl. Strogatz, 331.
[67] Vgl. Strogatz, 331 f.

im DGL-System kodiert sind.[68] Notwendige Voraussetzung für die Entstehung von chaotischem Verhalten ist dabei, die Existenz von nicht-linearen Termen in diesem DGL-System.[69]

Der dritte erforderliche Faktor zur Klassifikation chaotischer Systeme ist die empfindliche Abhängigkeit von den Anfangsbedingungen. Daraus erfolgt, dass sich Bahnkurven im exponentiellen Verhältnis voneinander entfernen, auch wenn deren Ausgangspunkte im Phasenraum in minimaler Entfernung zueinander liegen. Mathematisch kann dieser Zusammenhang demnach wie folgt beschrieben werden.

$$|\delta(t)| \sim |\delta_0| e^{\lambda t} \tag{2.22}$$

Der Ausdruck $\delta(t)$ bezeichnet dabei den Abstand zwischen zwei verschiedenen Bahnkurven in Abhängigkeit von der Zeit t. Initial beträgt dieser Abstand den Wert δ_0. Es wird ersichtlich, dass der Betrag von $\delta(t)$ exponentiell ansteigt, sofern der sogenannte Liapunov Exponent λ positiv ist. Aus dieser Beziehung lässt sich eine Methode zur Abschätzung eines Zeithorizonts $t_{horizon}$ ableiten, in dem die Abweichung der beiden Trajektorien einen festzusetzenden Toleranzbereich a nicht überschreitet.[70]

$$t_{horizon} \sim \frac{1}{\lambda} ln\left(\frac{a}{|\delta_0|}\right) \tag{2.23}$$

Diese hohe Empfindlichkeit hinsichtlich der Anfangsbedingungen ist die entscheidende Ursache dafür, dass das Verhalten chaotischer Systeme langfristig nicht vorhersagbar ist. Nachdem die praktische Messgenauigkeit eines physikalischen Systems limitiert ist, weicht eine gemessene Anfangsbedingung stets von den realen Konditionen ab.[71] Der Messfehler ist folglich als initiale Abweichung zweier Trajektorien zu verstehen, die in obiger Gleichung durch δ_0 repräsentiert ist. Der Zeithorizont $t_{horizon}$ gibt damit die Dauer an, in dem das Verhalten eines chaotischen Systems mit einer Genauigkeit von a vorhergesagt werden kann. Da der Vorhersagehorizont und der Messfehler in logarithmischer Beziehung zueinanderstehen, muss der Messfehler exponentiell reduziert werden, um den

[68] Vgl. Strogatz, 331.
[69] Vgl. Argyris u. a., *Die Erforschung des Chaos*, 27.
[70] Vgl. Strogatz, *Nonlinear Dynamics and Chaos*, 328.
[71] Vgl. Argyris u. a., *Die Erforschung des Chaos*, 22.

Vorhersagezeitraum nur linear zu verlängern. Somit ist die langfristige Vorhersage der Dynamik eines chaotischen Systems nicht möglich.

Es ist dabei explizit hervorzuheben, dass Unvorhersagbarkeit eine inhärente Eigenschaft chaotischer Systeme ist. Anders als bei stochastisch beschriebenen Systemen verhindert nicht der Mangel an Information die Vorhersage des langfristigen Verhaltens. Wie zuvor erläutert, kann ein chaotisches System durch ein vollständig deterministisches DGL-System modelliert werden, sodass die stark begrenzte Vorhersagbarkeit als prinzipielle Eigenschaft eines solchen Systems zu begreifen ist.[72]

2.3.3 Numerische Lösungsverfahren für DGL-Systeme

Zum Abschluss des Kapitels werden numerische Methoden zur Bestimmung der Lösung von DGL-Systemen eingeführt. Wie zuvor gezeigt, werden chaotische Systeme üblicherweise als Systeme von Differentialgleichungen dargestellt. Mithilfe von numerischen Verfahren kann somit zumindest für einen begrenzten Zeitraum die Lösung eines chaotischen Systems abgeschätzt werden.

Verfahren zur numerischen Lösung von DGL-Systemen basieren grundsätzlich auf der Idee, die anhand der in Abschnitt 2.3.1 vorgestellten graphischen Methodik deutlich wird. Numerische Verfahren nutzen also ebenso aus, dass Differentialgleichungen den Zusammenhang zwischen dem aktuellen Zustand eines Systems und der momentanen Änderungsrate einer unbekannten Funktion beschreiben.[73] Eine mathematisch exakte Notation zur Umsetzung dieser Idee liefert das Eulersche Polygonzugverfahren.

Ziel des Verfahrens ist es eine Lösung im Intervall $I = [a; b]$ für ein System von gewöhnlichen Differentialgleichungen erster Ordnung anzunähern. Voraussetzung für die numerische Lösung ist die Angabe von Anfangswerten für jede der Variablen $u_1, \dots, u_n$, die im Vektor $\boldsymbol{u_0}$ gelistet sind. Zudem ist die sogenannte Schrittweite h festzusetzen, die das Intervall gleichmäßig aufteilt. Jede der Differentialgleichungen im DGL-System soll nun näherungsweise gelöst werden, indem die exakten Lösungen $u_1(t), \dots, u_n(t)$ über die jeweilige Kurventangente approximiert werden. Dabei ist die Steigung $m_{\square}$ der entsprechenden Tangenten nach Gleichung 2.16 identisch zum

[72] Vgl. Argyris u. a., 22 f.
[73] Vgl. Weitz, *Konkrete Mathematik (nicht nur) für Informatiker*, 738.

jeweiligen Funktionswert $f_i(u_1, \dots, u_n)$, sodass für den Ordinatenwert der Tangente an der Stelle $t_1 = t_0 + \mathrm{h}$ Folgendes gilt.[74]

$$u_{i,1} = u_{i,0} + (t_1 - t_0) \cdot m_i = u_{i,0} + \mathrm{h} \cdot f_i(u_0) \tag{2.24}$$

Dieselbe Methodik kann nun iterativ zur Berechnung weiterer Werte eingesetzt werden. Allgemein kann die Vorschrift zur Näherung der Funktionswerte eines DGL-Systems, wie folgt, notiert werden.

$$u_{k+1} = u_k + \mathrm{h} \cdot \mathrm{f}(\mathrm{u_k}) \tag{2.25}$$

Zur Verbesserung dieses Verfahrens sind an zusätzlichen Stellen im Intervall zwischen t_0 und t_1 Steigungswerte der Funktion zu ermitteln. Aus diesen Steigungswerten kann anschließend ein gewichteter Mittelwert errechnet werden, um den Zuwachs der einzelnen Funktionen im Intervall genauer abschätzen zu können. Demzufolge liegt derartigen Verfahren derselbe Ansatz wie dem Eulerschen Polygonzugverfahren zugrunde. Lediglich das Vorgehen zur Berechnung der Steigung m_i eines Funktionswertes ist anzupassen.[75] Zur Veranschaulichung solcher verbesserter numerischer Lösungsmethoden ist nachfolgend das Runge-Kutta-Verfahren vierter Ordnung aufgeführt.[76]

$$u_{k+1} = u_k + \frac{1}{6} \cdot (k_1 + 2k_2 + 2k_3 + k_4) \tag{2.26}$$

Die Steigungen der Funktionswerte des DGL-Systems an den unterschiedlichen Stellen im Intervall werden durch $k_1, \dots, k_4$ repräsentiert, die, wie folgt, bestimmt werden.

$$k_1 = \mathrm{h} \cdot f(u_k) \tag{2.27}$$

$$k_2 = \mathrm{h} \cdot f\left(u_k + \frac{k_1}{2}\right) \tag{2.28}$$

$$k_3 = \mathrm{h} \cdot f\left(u_k + \frac{k_2}{2}\right) \tag{2.29}$$

$$k_4 = \mathrm{h} \cdot f(u_k + k_3) \tag{2.30}$$

Der Bezeichnung des Verfahrens entsprechend liegt die Abweichung zwischen dem numerisch ermittelten und dem korrekten Wert in der Größenordnung $\mathcal{O}(h^4)$. Es ist also generell möglich durch Halbierung der Schrittweite die Abschätzung des

[74] Vgl. Papula, *Mathematik für Ingenieure und Naturwissenschaftler Band 2*, 474.
[75] Vgl. Weitz, *Konkrete Mathematik (nicht nur) für Informatiker*, 750 f.
[76] Vgl. Papula, *Mathematik für Ingenieure und Naturwissenschaftler Band 2*, 479.

Funktionswertes in der vierten Potenz zu verbessern. Verschiedene Adaptionen des Runge-Kutta-Verfahrens, die auf dem gerade vorgestellten Prinzip aufbauen, erhöhen die Genauigkeit der numerischen Abschätzung über diese Größenordnung hinaus. Hierbei kann etwa algorithmisch die Wahl einer geeigneten Schrittweite ermittelt werden.[77] Eine detaillierte Vorstellung dieser Algorithmen kann im Rahmen dieser Arbeit nicht geleistet werden.

[77] Vgl. Mathews und Fink, *Numerical methods using MATLAB*, 497.

3 Methodik

Basierend auf dem Verständnis für den Aufbau eines PINNs und den grundlegenden Kenntnissen chaotischer Systeme ist in diesem Kapitel die Methodik darzulegen, die zur Umsetzung der zu Beginn formulierten Ziele anzuwenden ist. Wie zu Beginn erwähnt, sind die in dieser Arbeit vorgestellten Untersuchungen nach einem empirischen Ansatz durchzuführen. Folglich ist es notwendig, ein geeignetes chaotisches System zu identifizieren, das experimentell analysiert werden kann. Hierzu werden in einem ersten Teilabschnitt des Kapitels Kriterien benannt, anhand derer die Eignung eines Systems überprüft werden kann. Anschließend wird ein spezifisches System, das diese Kriterien vollständig erfüllt, ausgewählt und präsentiert. Dazu sind die in Kapitel 2.3 dokumentierten Methoden zur Beschreibung eines dynamischen Systems einzusetzen. In einem zweiten Schritt sind diejenigen Technologien zu bestimmen, mit deren Hilfe ein PINN für das identifizierte dynamische System entwickelt werden kann. Die Tauglichkeit einer Technologie für die Implementierung eines PINNs ist dabei durch Entwicklung passender Prototypen nachzuweisen. Der letzte Abschnitt des Kapitels dient dazu, die konkrete Vorgehensweise festzulegen, durch die überprüft werden soll, inwiefern PINNs zur Lösung inverser Probleme für chaotische Systeme eingesetzt werden können.

3.1 Identifikation eines geeigneten chaotischen Systems

Für die Auswahl eines passenden Systems sind vorab Kriterien festzuhalten, die ein dynamisches System erfüllen muss, um für weitere Untersuchungen als geeignet zu gelten. Es ist hierbei ausreichend ein einziges System zu bestimmen, das den erforderlichen Bedingungen entspricht. Somit sind die Kriterien als uneingeschränkt einzuhaltende Anforderungen zu verstehen, sodass jedes System, das diese Eigenschaften aufweisen kann, für den Zweck dieser Arbeit als geeignet betrachtet werden kann.

3.1.1 Auswahlkriterien chaotischer Systeme

Damit ein System für die Untersuchung durch PINNs passend ist, darf dieses nicht ausschließlich chaotisches Verhalten zeigen. Zusätzlich müssen klar definierte Parameterbereiche und Anfangsbedingungen existieren, für die die Dynamik des Systems nicht aperiodisch ist. Hintergrund dieser Anforderung ist, dass es möglich sein muss, die korrekte Implementierung eines PINNs für das gewählte System zu

prüfen. Weist das System nur chaotisches Verhalten auf, so ist die Ursache bei einem etwaigen Scheitern bei der Lösung des Systems kaum zu lokalisieren. Ob der Misserfolg auf eine fehlerhafte Implementierung des PINNs zurückzuführen ist, oder ob PINNs grundsätzlich nicht zu einer Lösung fähig sind, wäre in diesem Szenario schwerlich feststellbar.

Ein weiteres Kriterium besteht darin, dass ein geeignetes System über möglichst wenige frei wählbare Parameter verfügen soll. Dieses Kriterium leitet sich direkt aus der Zielsetzung der Arbeit ab, nach der zu prüfen ist, inwiefern inverse Probleme für chaotische Systeme durch PINNs gelöst werden können. Zur Beantwortung dieser Fragestellung wäre es folglich hinderlich, wenn das zur Untersuchung eingesetzte System es nicht ermöglicht, Parameter unabhängig voneinander zu bestimmen. Für eine strukturierte Analyse der Eigenschaften von PINNs muss es hingegen möglich sein, ein PINN darauf zu trainieren, nur einen einzelnen Parameter des Systems zu ermitteln.

Die letzten beiden Kriterien ergeben sich daraus, dass der Fokus der Arbeit auf der Untersuchung der Fähigkeiten von PINNs liegt und ausdrücklich nicht auf der Analyse chaotischer Systeme. Demzufolge sollte das zu analysierende System zum einen möglichst einfach strukturiert sein. Ein passendes DGL-System sollte wenige Variablen umfassen, in expliziter Form dargestellt werden können, und sich exklusiv aus gewöhnlichen Differentialgleichungen zusammensetzen. Zum anderen muss ein geeignetes System bereits umfassend erforscht und dokumentiert sein. Durch andere Methoden zur Analyse dynamischer Systeme, wie sie etwa in Kapitel 2.3 beschrieben sind, müssen qualitative Unterschiede des Verhaltens in verschiedenen Parameterbereichen bekannt sein. Weiterhin müssen die Bifurkationspunkte, an denen diese qualitative Änderung der Dynamik des Systems eintritt, identifiziert sein.

3.1.2 Lorenz-System

Ein System, das diese Bedingungen vollständig erfüllt, ist das sogenannte Lorenz-System. Dieses System aus drei Differentialgleichungen wurde ursprünglich als vereinfachte Modellierung von atmosphärischen Luftbewegungen von Edward Lorenz entwickelt wurde. Von besonderer Bedeutung ist allerdings, dass Lorenz an diesem System historisch erstmalig chaotisches Verhalten beobachtet hat.[78] Ausgehend von

[78] Vgl. Gleick, *Chaos*, 21 ff.

der untenstehenden Darstellung des DGL-Systems[79] ist nachfolgend zu zeigen, dass das Lorenz-System die im vorigen Teilabschnitt genannten Kriterien realisiert.

$$\begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \\ \frac{dz}{dt} \end{pmatrix} = \begin{pmatrix} \sigma(y-x) \\ rx - y - xz \\ xy - bz \end{pmatrix} \tag{3.1}$$

Nachdem das Lorenz-System das als erstes entdeckte chaotische System ist, ist das System und dessen Verhalten umfassend untersucht und in zahlreichen Publikationen und Lehrbüchern dokumentiert. Zudem ist anhand der obigen Gleichung ersichtlich, dass sich das System lediglich aus drei gewöhnlichen Differentialgleichungen zusammensetzt, die in expliziter Darstellung notiert werden können. In den einzelnen Differentialgleichungen sind dabei zwei nicht-lineare Terme enthalten, die als notwendige Bedingung für die Entstehung von chaotischem Verhalten erforderlich sind. Die beiden nicht-linearen Terme $x \cdot z$ und $x \cdot y$ ergeben sich aus der Multiplikation der Funktionswerte von $x(t)$, $y(t)$ und $z(t)$, und können somit ohne großen Aufwand berechnet werden. Die beiden Kriterien, die die weitreichende Erforschung und Dokumentation des Systems, sowie eine einfache mathematische Struktur einfordern, sind damit als erfüllt zu betrachten.

Weiterhin ist festzuhalten, dass in der obenstehenden Gleichung σ, r und b als Parameter des Systems zu deuten sind. Während die Variation von σ und b keine qualitative Änderung des dynamischen Verhaltens des Systems hervorbringt, ist es der Parameter r, der angepasst werden muss, um die Dynamik des Systems entscheidend zu verändern.[80] Infolgedessen kann für die Lösung des inversen Problems dieser Parameter isoliert untersucht werden. Die restlichen Parameter σ und b werden hierbei jeweils auf einen konstanten Wert festgelegt, da diese ohnehin keinen Einfluss auf die Änderung der Dynamik nehmen. Aus der physikalischen Anwendung des Systems leiten sich hierbei $\sigma = 10$ und $b = \frac{8}{3}$ als geeignete Belegungen der Parameter ab. Durch diese Eigenschaft des Lorenz-Systems ist auch

[79] Strogatz, *Nonlinear Dynamics and Chaos*, 309.
[80] Vgl. Strogatz, 337.

das Kriterium verwirklicht, nach dem ein geeignetes System möglichst wenige Parameter zur freien Belegung offenlässt.

Anhand der Unterschiede der Dynamik des Systems, die durch Variation des Parameters r entstehen, ist der Wertebereich von r in separate Intervalle zu untergliedern. Innerhalb dieser Intervalle von r erfolgt keine qualitative Änderung im Verhalten des Systems. Die Grenze eines Intervalls entspricht folglich einem Bifurkationspunkt, an dem sich das Verhalten des Systems entscheidend ändert. Für $r < 1$ ist zu beobachten, dass sich jede Bahnkurve unabhängig von deren Anfangsbedingung langfristig dem Ursprung annähert. Im Intervall $I_1 = [0; 1)$ ist folglich kein periodisches oder gar chaotisches Verhalten zu beobachten, weshalb der Ursprung in diesem Parameterbereich als globaler Fixpunkt zu deuten ist.[81] Wird der Parameter r über die obere Grenze des Intervalls I_1 hinaus erhöht, so verliert der Fixpunkt im Ursprung seine Stabilität, während sich zwei weitere Fixpunkte außerhalb des Ursprungs entwickeln. Im Parameterbereich $r \in I_2 = [1; 13{,}96)$ oszillieren die Trajektorien des Systems zunächst um diese Fixpunkte herum, bevor diese einen der Fixpunkte erreichen. Dieses Verhalten im Phasenraum entspricht einer gedämpften harmonischen Schwingung bei zeitlicher Betrachtung der Funktionen des DGL-Systems. Überschreitet r die Grenze des Intervalls I_2, ist das Verhalten des Lorenz-Systems als aperiodisch zu beschreiben. Über einen längeren Zeitraum geht diese Aperiodizität jedoch in einen stabilen Zustand in einem der beiden Fixpunkte über. Ab einem Wert von $r = 24.74$ erfüllt die Dynamik des Lorenz-System schließlich alle der in Kapitel 2.2.2 geführten Eigenschaften eines chaotischen Systems.[82]

Anhand dieser Betrachtung der Dynamik des Lorenz-Systems in verschiedenen Parameterbereichen wird deutlich, dass das Lorenz-System nicht ausschließlich chaotisches Verhalten aufweist. Damit ist auch dieses Eignungskriterium durch das Lorenz-System umgesetzt.

Folglich erfüllt das Lorenz-System alle Bedingungen, die ein chaotisches System für diese Arbeit aufbringen muss. Demnach dient das Lorenz-System als Basis der empirischen Untersuchungen dieser Arbeit.

[81] Vgl. Strogatz, 323.
[82] Vgl. Strogatz, 338.

3.2 Technologieauswahl

Dieser Teilabschnitt dient dazu geeignete Technologien für die Entwicklung eines PINNs zu identifizieren. Als Grundlage der Implementierung eines PINNs ist eine Programmiersprache zu identifizieren, die für das Umfeld des maschinellen Lernens passend ist. Für die gewählte Programmiersprache ist nachzuweisen, dass diese, mitsamt der für diese Sprache entwickelten Programmbibliotheken, die Implementierung eines PINNs erlaubt.

Die naheliegende Wahl einer Technologie zur Implementierung einer Anwendung im Bereich des maschinellen Lernens ist die Programmiersprache Python. Python bietet nicht nur unterstützende syntaktische Elemente, die sowohl objektorientierte als auch funktionale und strukturierte Programmierung ermöglichen. Darüber hinaus zeichnet sich Python durch die Abstraktion von Hardware und Betriebssystem aus, indem etwa Speicherplatz, der von nicht mehr benötigten Variablen belegt ist, automatisiert freigegeben wird. Insbesondere im Umfeld neuronaler Netze erweist sich diese Charakteristik der Programmiersprache als hilfreich. Wie in Kapitel zwei gezeigt, erfordert die Optimierung eines neuronalen Netzes die Berechnung zahlreicher Matrixoperationen, weshalb diese auf hierfür spezialisierten Prozessorarchitekturen durchgeführt wird. Python ermöglicht die Nutzung solcher Hardware-Komponenten, ohne dass der Anwender über deren Implementierungsdetails informiert sein muss.[83] Weiterhin vereinfacht eine Vielzahl an Programmbibliotheken und Frameworks, die explizit für den Einsatz in Kombination mit Python konzipiert sind, die Implementierung maschineller Lernverfahren. Genauso existieren Software-Module, durch die mathematische Operationen effizient ausgeführt werden können. Die Einbettung von extern entwickelten Programmpakten wird in Python über ein integriertes Paketverwaltungssystem koordiniert, sodass selbst geschriebener Programmcode leicht um zusätzliche Funktionalitäten erweitert werden kann.[84]

[83] Vgl. Hope, *Einfuhrung in TensorFlow*, 292.
[84] Vgl. Steyer, *Programmierung in Python*, 3 f.

Um nachweisen zu können, dass Python auch für die Entwicklung eines PINNs eine geeignete Programmiersprache darstellt, ist ein prototypisches PINN in Python zu entwickeln. Gelingt die Entwicklung eines repräsentativen Modells eines PINNs, so ist sichergestellt, dass die Implementierung eines PINNs in dieser Programmiersprache möglich ist. Hierfür ist zunächst ein möglichst einfach strukturiertes PINN zu modellieren. Beim Aufbau eines solchen Prototyps ist zum einen darauf zu achten, dass alle charakteristischen Merkmale eines PINNs berücksichtigt sind. Zugleich ist die Struktur des Modells so zu gestalten, dass die korrekte Umsetzung der Funktionsweise des PINNs leicht überprüfbar ist. Aus diesen Vorüberlegungen resultiert die Architektur des in folgender Abbildung veranschaulichten Prototyps eines PINNs.

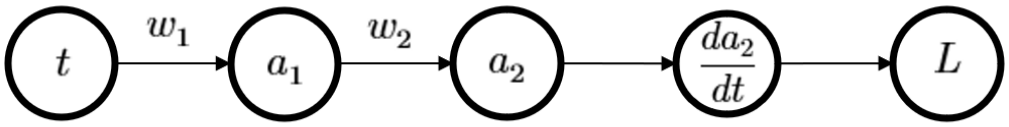

Abbildung 5: Prototyp eines PINNs

Während t als Eingabewert des PINNs aus einer zeitlichen Domäne zu verstehen ist, werden die Ausgabewerte der einzelnen Neuronen des PINNs nach Gleichung 2.1 und 2.2. berechnet. Diese können in identischer Form in Python-Programmcode überführt werden. Es wird ebenso ersichtlich, dass die Verlustfunktion L direkt von der Ableitung des Ausgangswertes des neuronalen Netzes nach dessen Eingabewert abhängig ist, so wie in Kapitel 2.2 für physikalisch informierte Verlustterme beschrieben. Diese Ableitung nach dem Eingabewert kann anhand des Backpropagation-Algorithmus bestimmt werden, der in den Gleichungen 2.11 – 2.14 formuliert ist und in Kapitel 2.1 vorgestellt wird. Da obenstehendes PINN jeweils nur ein Neuron in jeder Schicht enthält, reduziert sich die allgemeine Formulierung der Backpropagation für neuronale Netze auf die Anwendung der Kettenregel für Funktionen einer Variablen. Die Berechnung dieser Ableitung des Ausgabewertes nach der Eingabe des PINNs ist untenstehender Gleichung zu entnehmen.

$$\frac{da_2}{dt} = \frac{da_2}{dz_2} \cdot \frac{dz_2}{da_1} \cdot \frac{da_1}{dz_1} \cdot \frac{dz_1}{dt} \tag{3.2}$$

Nachdem $z_i = w_i \cdot a_{i-1} + b_i$ gilt, können die Ableitungen $\frac{dz_i}{da_{i-1}}$ mit dem konkreten Wert $w_{\square}$ belegt werden. Ableitungen der Form $\frac{a_{\square}}{z_i}$ sind hingegen von der Wahl der Aktivierungsfunktion abhängig und daher weiterhin in allgemeiner Form darzustellen. Allerdings ist hierbei zu berücksichtigen, dass jeder der Ausgabewerte in Abhängigkeit zu seinem Gewicht steht, wodurch die Ableitung stets zumindest teilweise von diesem Gewicht, sowie von den Gewichten vorheriger Ausgabewerte beeinflusst wird. Dieser Zusammenhang soll durch die Notation $\frac{da_{\square}}{dz_i}(w_i)$ deutlich werden. Durch Einsetzen in die obige Gleichung, ergibt sich folgender Zusammenhang, der wiederum in Python analog umgesetzt werden kann.

$$\frac{da_2}{dt} = \frac{da_2}{dz_2}(w_2, w_1) \cdot w_2 \cdot \frac{da_1}{dz_1}(w_1) \cdot w_1 \tag{3.3}$$

Der auf diese Weise berechnete Wert der Ableitung des Ausgangswertes nach dem Eingangswert eines neuronalen Netzes wird nun in die Verlustfunktion integriert. Zur Optimierung der Vorhersage eines PINNs müssen anschließend die Gewichte des neuronalen Netzes angepasst werden. Wie in Kapitel 2.1 dargelegt, ist hierzu die Berechnung der Ableitung der Verlustfunktion nach den einzelnen Gewichten notwendig. Hierbei ist jedoch zu beachten, dass die Anwendung der Kettenregel zur Bestimmung dieser Ableitung nicht ausreichend ist. In Gleichung 3.3 ist zu erkennen, dass jeweils mehrere Faktoren des in die Verlustfunktion eingesetzten Terms einzelnen Gewichten abhängig sind. Für die Differenzierung nach diesen Gewichten ist folglich der Einsatz der Produktregel nötig. Veranschaulicht wird dies exemplarisch anhand der Berechnung der Ableitung der Verlustfunktion nach dem Gewicht w_2.

$$\frac{dL}{dw_2} = \frac{dL}{\left(\frac{da_2}{dt}\right)} \cdot \left(\frac{d}{dw_2}\left(\frac{da_2}{dz_2}(w_2, w_1)\right) \cdot w_2 \cdot \frac{da_1}{dz_1}(w_1) \cdot w_1 + \frac{da_2}{dz_2}(w_2) \cdot \frac{da_1}{dz_1}(w_1) \cdot w_1\right) \tag{3.4}$$

Zwar kann ein solcher Term ebenso durch entsprechenden Python Code berechnet werden. Gleichwohl ist festzuhalten, dass die Berechnung von nur einem Element des Gradienten selbst für das einfach strukturierte Modell eines PINNs im Vergleich zum neuronalen Netz erheblich komplexer ist. Dieses Problem wirkt sich noch stärker auf die Berechnung der Ableitung der Verlustfunktion nach Gewichten aus, die sich in Schichten nahe am Eingangswert des Netzes befinden. So tritt w_1 in Gleichung 3.3 in drei verschiedenen Faktoren auf, was dazu führt, dass die Ableitung nach w_1 bereits

einen zusätzlichen Summanden im Verhältnis zu Gleichung 3.4, umfasst. Auf diese Weise zeigt der ausschließlich in Python – unter Ausschluss weiterer Programmbibliotheken – entwickelte Prototyp auf, dass eine solche Lösung bei weitem nicht in einem Grad skalierbar ist, wie es zur Implementierung eines für diese Arbeit geeigneten PINNs nötig ist.

Als Alternative zur Umsetzung in Python ohne Verwendung extern bereitgestellter Programmbibliotheken, liegt es nahe eines der für das maschinelle Lernen in Python entwickelten Frameworks für die Realisierung eines PINNs zu nutzen. Ein weitverbreitetes Framework, das bei der Implementierung neuronaler Netze Anwendung findet, ist TensorFlow. Grundprinzip von TensorFlow ist die Darstellung der Berechnung von Ausgabewerten als Graph. Nachdem die Struktur neuronaler Netze ebenfalls als derartiger Berechnungsgraph interpretiert werden kann, eignet sich TensorFlow für die Durchführung der mathematischen Operationen beim Training eines neuronalen Netzes. Insbesondere erleichtert die Strukturierung der Berechnung von Ausgabewerten in Form eines Graphen, die Durchführung von Algorithmen zur automatischen Differentiation, wie etwa der Backpropagation. Darüber hinaus ermöglicht der modulare Aufbau des Frameworks es, einzelne Komponenten eines neuronalen Netzes, wie zum Beispiel die Aktivierungsfunktion oder das Optimierungsverfahren, problemlos auszutauschen.[85] [86]

Um die Eignung von TensorFlow für die Implementierung eines PINNs zu belegen, wird der in Abbildung 5 visualisierte Prototyp eines PINNs mithilfe von TensorFlow umgesetzt. Dabei wird das zuvor dargelegte Problem der Berechnung des Gradienten der Kostenfunktion eines PINNs durch den Einsatz der TensorFlow-Komponente GradientTape gelöst. Die Korrektheit des mithilfe dieser Komponente berechneten Ergebnisses wird durch den Abgleich mit dem Resultat von Gleichung 3.4 für eine konkrete Belegung der Variablen sichergestellt. Aufgrund dieser erfolgreichen Überprüfung ist festzuhalten, dass TensorFlow für die Implementierung eines PINNs geeignet ist. Zum Abschluss des Prozesses der Technologieauswahl ist festzuhalten, dass im weiteren Verlauf dieser Arbeit die Programmiersprache Python in Kombination mit dem Framework TensorFlow zur Realisierung von PINNs eingesetzt wird.

[85] Vgl. Nelli, *Python Data Analytics*, 298 f.
[86] Vgl. Hope, *Einfuhrung in TensorFlow*, 5 f.

3.3 Empirische Untersuchung

Nachdem sowohl ein geeignetes DGL-System als auch passende Technologien ausgewählt sind, ist das weitere Vorgehen zur Beantwortung der zu Beginn formulierten Forschungsfragen zu dokumentieren.

Zunächst wird die grundlegende Struktur eines PINNs zur Lösung von Simulationsproblemen und inversen Problemen für den spezifischen Anwendungsfall des Lorenz-Systems implementiert. Anschließend sind die entwickelten PINNs auf ihre korrekte Umsetzung hin zu überprüfen. Zur Validierung eines PINNs wird dabei das Verfahren des vergleichenden Testens nach ISTQB-Standard angewandt. Beim vergleichenden Test wird das zu testende Software-System einer alternativen Variante des Systems, die jedoch zur Erfüllung derselben Aufgabe dient, gegenübergestellt. Voraussetzung für diese Methodik ist, dass die Funktionsweise der alternativen Ausführung des Systems bereits erfolgreich geprüft ist. Ist diese Bedingung gegeben, werden beiden Systemen dieselben Eingabedaten zur Verfügung gestellt. Daraufhin können die von den verschiedenen Systemen auf diese Eingabe hin generierten Ausgabedaten miteinander abgeglichen werden. Auf diese Weise ist es möglich, etwaige Verbesserungen der Vorhersagegenauigkeit erfassen und einem Referenzwert gegenüberstellen zu können.[87] Im konkreten Anwendungsfall ist ein PINN als zu prüfendes System zu betrachten. Währenddessen sind die numerische Berechnung von Lösungen nach dem Runge-Kutta Verfahren oder auch neuronale Netze ohne physikalisch informierten Verlustterm als Alternativsysteme zur Bestimmung eines Referenzwertes zu verwenden.

Aufbauend auf der Untersuchung des Lorenz-Systems in Kapitel 3.1 werden verschiedene Parameterbereiche des Systems, die zur Validierung geeignet sind, bestimmt. Zum einen ist das Parameterintervall $r \in I_1 = [0; 1)$ zur Validierung von PINNs einzusetzen, in dem der Ursprung als Fixpunkt des Systems fungiert. Die Dynamik des Lorenz-Systems ist in diesem Bereich leicht zu beschreiben, sodass dieses Intervall für eine erste, grundsätzliche Validierung zweckmäßig ist. Zur weiteren Validierung werden PINNs im Parameterbereich $r \in I_2 = [1; 13{,}96)$ trainiert. In diesem Intervall ist das Lorenz-System durch periodisches zeitliches Verhalten gekennzeichnet ist. Da periodisches Verhalten deutlich schwerer zu erfassen ist,

[87] Vgl. Röttger, Dietrich, und Röttger, *Basiswissen KI-Testen*, 206 f.

können auf diese Weise bei erfolgreicher Validierung zugleich Hyperparameter des Systems optimiert werden.

Schließlich ist das validierte PINN für die Lösung von Simulationsproblemen und inversen Problemen im chaotischen Parameterintervall des Lorenz-Systems einzusetzen. Damit auch für chaotisches Verhalten eine solche Referenzlösung bestimmt werden kann, ist es nötig den Vorhersagehorizont des Lorenz-Systems für numerische Lösungsmethoden zu bestimmen. Das in dieser Arbeit für die numerische Erzeugung einer Lösung des Lorenz-Systems eingesetzte Runge-Kutta Verfahren vierter Ordnung ist hierbei mit einer solchen Schrittweite konfiguriert, dass der Fehler im ersten Berechnungsschritt auf $|\delta_0| = 10^{-9}$ begrenzt ist. Werden zusätzlich der numerisch für das Lorenz-System ermittelte Wert für $\lambda = 0.9$, sowie ein Toleranzwert von $a = 10^{-2}$ festgesetzt, ergibt sich nach Gleichung 2.23 folgender Wert für den Vorhersagehorizont.

$$t_{horizon} \sim \frac{1}{0.9} ln\left(\frac{10^{-2}}{10^{-9}}\right) \approx 17.91 \tag{3.5}$$

Zudem ist anzumerken, dass in dieser Arbeit ausschließlich der in Kapitel 2.2 vorgestellte Aufbau eines PINNs eingesetzt wird. Ansätze, die die Struktur eines PINNs grundlegend verändern oder um andere Methoden des maschinellen Lernens ergänzen, werden in dieser Arbeit nicht betrachtet.[88] Genauso finden weitere Verfahren des Physics Informed Machine Learnings, wie in Kapitel 2.2.3 präsentiert, keine Anwendung bei den nachfolgenden Untersuchungen. Diese Einschränkung auf den unveränderten Aufbau eines PINNs ist dadurch begründet, dass das Ziel dieser Arbeit die Untersuchung der Eigenschaften von PINNs ist. Werden PINNs mit weiteren Verfahren verknüpft, so ist es nicht mehr möglich, den Anteil des PINNs am Resultat der Arbeit eindeutig zu benennen.

[88] Vgl. Moseley, „Physics-Informed Machine Learning“, 35 ff.

4 Technische Umsetzung

In diesem Kapitel wird die technische Realisierung der Physik-informierten neuronalen Netze (PINNs) vorgestellt. Ziel ist es, die konkreten Schritte zur Implementierung verschiedener PINN-Varianten darzulegen. Anschließend werden die Architektur, sowie grundlegende Hyperparameter im Trainingsprozess aufgelistet, welche die Grundlage aller Vorhersagen der umgesetzten PINNs bilden. Zuletzt liegt der Fokus auf der Validierung der implementierten PINN-Varianten, die jeweiligen Umsetzungen bezüglich der Lösung des Simulationsproblems und des inversen Problems miteinander verglichen werden. Zudem werden beide Probleme sowohl für Parameter aus dem Bereich des Fixpunktes und dem periodischen Bereich untersucht. Dieses Kapitel legt das Fundament für die erfolgreiche Implementierung der Systeme und schafft damit die Voraussetzung für die anschließende Untersuchung im chaotischen Bereich.

4.1 Verlustterm für das Lorenz-System

Dieses Unterkapitel befasst sich mit der konkreten Umsetzung unterschiedlicher PINN-Implementierungen. Zunächst wird auf Basis der in Kapitel 2 beschriebenen physikalischen Verlustfunktion, die spezifische Belegung des Terms aus Gleichung 2.18 für das Lorenz-System dargelegt. Anschließend werden zwei verschiedene Ansätze behandelt: Das IC-PINN (Initial-Condition PINN) sowie das datengetriebene PINN. Für beide Varianten wird die Modellarchitektur und die konkrete mathematische Beschreibung der Verlustfunktion für das Lorenz-System erläutert.

4.1.1 Physikalisch begründeter Verlustterm des Lorenz-Systems

Zur Integration des Lorenz-Systems in das Training eines PINNs muss die allgemein formulierte physikalisch begründete Verlustfunktion zur Anwendung für das Lorenz-System modifiziert werden. Dabei gilt folgender Zusammenhang zwischen dem spezifischen Ergebnis und der allgemeinen Form.

$$\boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}; \quad \lambda = \{\sigma, r, b\} \tag{4.1}$$

Diese Belegung wird in Gleichung 2.18 eingesetzt und ergibt dadurch den physikalisch begründeten Verlustterm für das Lorenz-System. Die Berechnung des physikalisch

begründeten Verlustterms erfolgt, indem das PINN zunächst eine Vorhersage für x, y, z des Lorenz-Systems für eine gegebene Eingabezeit t durchgeführt.

Die Modellvorhersagen werden dementsprechend im Folgenden als $\hat{x}(t), \hat{y}(t), \hat{z}(t)$ notiert. Berechnet man anschließend die Gradienten der Modellvorhersagen in Bezug auf die Eingabezeitpunkte t, so lassen sich diese mit den durch das Lorenz-System definierten Ableitungen vergleichen. Für die Berechnung der Gradienten wird gemäß der Implementierung in TensorFlow die Komponente GradientTape verwendet, deren Eignung für die Anwendung für PINNs bereits in Kapitel 3.2 nachgewiesen wurde. GradientTape ist ein Mechanismus, der die automatische Differenzierung nutzt, um die Ableitungen der Modellvorhersagen bezüglich der Eingabevariablen effizient zu berechnen. Während eines Vorwärtsdurchlaufs zeichnet GradientTape die komplette Rechenkette von Eingabe bis zur Ausgabe des neuronalen Netzes, um die spätere Ableitungsberechnung zu ermöglichen. Diese Aufzeichnung lässt sich durch die Analogie eines Bands veranschaulichen. Die Rechenschritte, die bei der Vorhersage durch das PINN-Modell durchgeführt werden, werden auf einem Band festgehalten. Anschließend wird dieses Band in einem Rückwärtsdurchlauf in umgekehrter Reihenfolge „abgespielt“, wobei berechnet wird, wie sich Änderungen der Eingabe *t* auf die Ausgabegrößen auswirken. Dies erlaubt die recheneffiziente Berechnung der Ableitungen.[89]

Die berechneten Gradienten bilden die Grundlage für die Berechnung des physikalisch begründeten Verlustterms. Mithilfe der bekannten Parameter des Lorenz-Systems und des Eingabevektors kann nun die tatsächliche Lösung der Ableitungen des Systems bestimmt werden. Der physikalisch informierte Verlustterm ergibt sich schließlich durch die Berechnung des Mean Squared Error (MSE) zwischen den Gradienten des PINNs und den Ableitungen des zugrundeliegenden physikalischen Systems:

$$L_{\text{PHY}} = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\frac{d\hat{x}}{dt}(t_i) - \frac{dx}{dt}(t_i)\right)^2 + \left(\frac{d\hat{y}}{dt}(t_i) - \frac{dy}{dt}(t_i)\right)^2 + \left(\frac{d\hat{z}}{dt}(t_i) - \frac{dz}{dt}(t_i)\right)^2\right] \quad (4.1)$$

Hierbei stellt N die Anzahl der zu untersuchenden Zeitpunkte dar, während $\hat{x}(t_i), \hat{y}(t_i), \hat{z}(t_i)$ die vom Netzwerk geschätzten Ableitungen nach der Zeit beschreiben

[89] Vgl. „Gradient Tape“.

und $x(t_i), y(t_i), z(t_i)$ die physikalisch bekannten bzw. erwarteten Ableitungen des Systems darstellen.

4.1.2 Initial-Condition Verlustterm des Lorenz-Systems

Zur Berücksichtigung bekannter Anfangswerte des zu vorhersagenden Systems im Trainingsprozess eines PINNs, wird zusätzlich der IC-Loss definiert. Dieser Verlustterm stellt sicher, dass das Modell neben den physikalischen Gleichungen auch die Anfangswerte des betrachteten Systems beachtet.

Berechnet wird der Initial-Condition Loss ähnlich zum physikalisch begründeten Verlustterm, indem das Modell die Anfangswerte durch eine Modellvorhersage approximiert. Die vorhergesagten Anfangswerte werden anschließend mit den tatsächlichen, während des Trainingsprozesses bekannten Anfangswerten verglichen, in dem der Fehler des IC-Loss über den Mean Squared Error (MSE) ermittelt wird. Für das Lorenz-System ergibt sich folgender Verlustterm zur Einhaltung der Anfangsbedingungen.

$$L_{IC} = |\hat{\boldsymbol{u}}_{PINN}(t_0) - \boldsymbol{u_0}|^2 = (\hat{x}(t_0) - x_0)^2 + (\hat{y}(t_0) - y_0)^2 + (\hat{z}(t_0) - z_0)^2 \qquad (4.2)$$

Dabei stellen $\hat{\boldsymbol{u}}(t_0)$ die vom Modell vorhergesagten und $\boldsymbol{u_0}$ die bekannten Anfangswerte dar.

4.1.3 Datenbasierter Verlustterm des Lorenz-Systems

Analog zum IC-Loss kann zur Unterstützung des Trainingsprozesses neben dem physikalisch begründeten Verlustterm zusätzlich eine datenbasierter Term in die Verlustfunktion eingeführt werden. Dies erlaubt dem Modell, neben den physikalischen Gesetzen auch Beobachtungsdaten in den Trainingsprozess miteinzubeziehen. Beobachtungsdaten können beispielsweise aus Messdaten einer physikalischen Experimentdurchführung oder der maschinellen Simulation von Daten durch numerische Verfahren entstammen. Das PINN kann sich somit explizit an vorhandene Beobachtungsdaten anpassen, ähnlich zum Prozess eines neuronalen Netzes ohne physikalischen Verlustterm, welches ausschließlich Anpassung an Daten die Kostenfunktion minimiert.

Für jeden Zeitpunkt in einem betrachteten Zeitintervall erhält das PINN als Eingabe die jeweiligen Zustandsgrößen für x, y und z. Sollten die Beobachtungsdaten maschinell generiert sein (so wie es häufig der Fall im Rahmen dieser Arbeit ist), so werden die Datenpunkte durch ein additives Gaußsches Rauschen transformiert, um die

Trainingsdaten an realistische Messungen anzunähern. Diese Datenpunkte repräsentieren schließlich die verrauschten Beobachtungen des Systems. Das PINN lernt somit neben der Physik des Systems, auch die Annäherung an die gemessenen oder simulierten Daten.

Der Verlustterm des Data-Loss ergibt sich durch den Mean Squared Error (MSE) zwischen der Modellvorhersage des PINNs und den gegebenen Beobachtungsdaten:

$$L_{DATA} = \frac{1}{N}\sum_{i=1}^{N} |\hat{u}\left(t_{data}^{(i)}\right) - \breve{u}_{data}^{(i)}|^2 \tag{4.3}$$

$$L_{DATA} = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\hat{x}(t_i) - x_{data}^{(i)}\right)^2 + \left(\hat{y}(t_i) - y_{data}^{(i)}\right)^2 + \left(\hat{z}(t_i) - z_{data}^{(i)}\right)^2\right] \tag{4.4}$$

Die Bezeichnungen $\hat{x}(t_i)$, $\hat{y}(t_i)$, $\hat{z}(t_i)$ stellen die vorhergesagten Ausgaben des Modells zum Zeitpunkt t_i dar, während $x_{\text{data}}^{(i)}$, $y_{\text{data}}^{(i)}$, $z_{\text{data}}^{(i)}$ die jeweils zugehörigen Beobachtungsdaten sind. Der MSE wird über alle N Datenpunkte berechnet.

Durch diese Verlustkomponente in der Gesamtverlustfunktion wird das PINN gezwungen, empirische Ergebnisse zu berücksichtigen und ist somit robuster gegen unregelmäßige und verrauschte Daten.

4.2 PINN Architektur und Trainingsprozess

Sowohl für die Lösung des Simulationsproblems als auch zur Lösung des inversen Problems wird das PINN wie folgt konfiguriert.

Hidden Layer	6
Neuronen pro Schicht	30
Aktivierungsfunktion	Silu
Gewichtsinitialisierung	Glorot Uniform

Tabelle 1: Konfiguration des PINNs

Die PINN-Architektur besteht aus einem vollständig verbundenen Feedforward-Netzwerk mit 6 verdeckten Schichten (Hidden Layers) und jeweils 30 Neuronen pro Schicht. Als Aktivierungsfunktion kommt die nichtlineare SiLU-Funktion (Sigmoid-weighted Linear Unit) zum Einsatz. Weiterhin erfolgt die Initialisierung der Gewichte nach dem Glorot-Uniform-Verfahren. Die gewählte Aktivierungsfunktion und Gewichtsinitialisierung gewährleisten die Stabilität beim Training des PINNs.

Die Hyperparameter während des Trainingsprozess sind standardmäßig wie folgt definiert:

Lernrate	0,01
Abklingrate	0,09
Optimierungsverfahren	Adam
Epochen	20000
Kollokationspunkte	1024
Alpha	0,5

Tabelle 2: Initiale Belegung der Hyperparameter des PINNs

Die Lernrate beträgt 0.01 und steuert die Schrittweite, mit der die Gewichte des Netzwerks während der Optimierung angepasst werden. Um eine stabilere Konvergenz während des Trainings zu erzielen, kommt eine Abklingrate von 0.09 zum Einsatz, die die Lernrate im Trainingsverlauf schrittweise reduziert. Als Optimierungsverfahren wird der Adam-Optimierer verwendet – ein adaptives Gradientenverfahren, das sich besonders bei der Lösung nichtlinearer Probleme im Kontext von PINNs bewährt hat.[90] Das Modell durchläuft 20.000 Trainingsepochen, wobei in jeder Epoche die Verlustfunktion anhand von 1024 Kollokationspunkten aktualisiert wird. Zusätzlich wird ein Gewichtungsfaktor $\alpha = 0.5$ eingesetzt, der das Verhältnis zwischen dem IC- bzw. Data-Loss und dem physikalisch begründeten Physics Loss innerhalb der Gesamtverlustfunktion bestimmt.
Diese Eigenschaften bilden die Grundlage für den Aufbau der in den Untersuchungen verwendeten PINNs. Sollten Änderungen an dieser Architektur vorgenommen werden, so wird dies explizit vor der Durchführung einer Untersuchung deutlich gemacht.

4.3 Verlustfunktionen für das Lorenz-System

Ziel der Bearbeitung eines Simulationsproblems mit einem PINN ist es, eine Approximation der Lösung eines physikalischen Systems zu ermitteln. Im Gegensatz zum inversen Problem werden hier keine Systemparameter der zugrundeliegenden physikalischen Gleichungen geschätzt, sondern möglichst genaue Lösungen unter Einhaltung der physikalischen Gesetze und weiterer Verlustterme ermittelt. Zur

[90] Vgl. Kingma und Ba, „Adam“, 1.

Formulierung der Verlustfunktion für das Simulationsproblem, das auch als Forward-Problem bezeichnet werden kann, stehen zwei mögliche Varianten zur Verfügung.

Variante 1: IC-Loss in Kombination mit physikalisch begründetem Verlustterm

Diese Form der Verlustfunktion kombiniert den Verlustterm für das Einhalten der Anfangswerte und der physikalischen Gesetzmäßigkeiten. Die Verlustfunktion wird wie folgt beschrieben durch:

$$\mathcal{L} = \alpha \cdot L_{IC} + (1 - \alpha) \cdot L_{PHY} \tag{4.5}$$

Dabei ist L_{IC} der IC-Loss, der den Fehler zwischen den vorhergesagten Anfangszuständen des Systems und den bekannten Anfangswerten misst, während $L_{\square\square\square}$ misst, wie gut die Modellvorhersagen die zugrundeliegenden physikalischen Gleichungen erfüllt. In vollständiger Form unter Verwendung der Definitionen der vorigen Abschnitte lautet die Formel für die Verlustfunktion somit:

$$\mathcal{L} = \alpha \cdot \left(\frac{1}{N_{\text{IC}}} \sum_{i=1}^{N_{\text{IC}}} \left[\left(\hat{x}\left(t_i^{\text{IC}}\right) - x_0 \right)^2 + \left(\hat{y}\left(t_i^{\text{IC}}\right) - y_0 \right)^2 + \left(\hat{z}\left(t_i^{\text{IC}}\right) - z_0 \right)^2 \right] \right) + (1 - \alpha)$$

$$\cdot \frac{1}{N_{\text{Phy}}} \sum_{i=1}^{N_{\text{Phy}}} \left[\left(\frac{d\hat{x}}{dt}(t_i) - \frac{dx}{dt}(t_i) \right)^2 + \left(\frac{d\hat{y}}{dt}(t_i) - \frac{dy}{dt}(t_i) \right)^2 + \left(\frac{d\hat{z}}{dt}(t_i) - \frac{dz}{dt}(t_i) \right)^2 \right] \tag{4.6}$$

Variante 2: Data-Loss in Kombination mit physikalisch begründetem Verlustterm:

Diese Form der Verlustfunktion kombiniert den Verlustterm für das Einhalten von Beobachtungsdaten mit dem Verlustterm zur Einhaltung des physikalischen Systems. Die Verlustfunktion wird wie folgt beschrieben:

$$\mathcal{L} = \alpha \cdot L_{DATA} + (1 - \alpha) \cdot L_{PHY} \tag{4.7}$$

Dabei misst L_{DATA} den Fehler zwischen den vorhergesagten Ergebnissen und den verfügbaren Beobachtungsdaten. L_{PHY} ist identisch zur vorhin gezeigten ersten Variante. Unter Einsatz der voriger Definitionen lautet die Formel für die Verlustfunktion:

$$\mathcal{L} = \alpha \cdot \left(\frac{1}{N_{\text{Data}}} \sum_{i=1}^{N_{\text{Data}}} \left[\left(\hat{x}(t_i) - x_{\text{data}}^{(i)} \right)^2 + \left(\hat{y}(t_i) - y_{\text{data}}^{(i)} \right)^2 + \left(\hat{z}(t_i) - z_{\text{data}}^{(i)} \right)^2 \right] \right) + (1 - \alpha)$$

$$\cdot \frac{1}{N_{\text{Phy}}} \sum_{i=1}^{N_{\text{Phy}}} \left[\left(\frac{d\hat{x}}{dt}(t_i) - \frac{dx}{dt}(t_i) \right)^2 + \left(\frac{d\hat{y}}{dt}(t_i) - \frac{dy}{dt}(t_i) \right)^2 + \left(\frac{d\hat{z}}{dt}(t_i) - \frac{dz}{dt}(t_i) \right)^2 \right] \quad (4.8)$$

In beiden Fällen sorgt der Parameter $a \in [0; 1]$ für die Gewichtung zwischen dem Physikalischen Verlustterm und einem zweiten Verlustterm (entweder IC, oder Data Loss). Je nach Problemstellung kann somit entschieden werden, welcher Verlustterm bevorzugt werden soll.

4.4 PINN Implementierung für das inverse Problem

Im Gegensatz zum Simulationsproblem, bei dem die Lösung des Systems für gegebene Systemparameter bestimmt wird, besteht das Ziel beim inversen Problem darin, unbekannte Systemparameter anhand gegebener Beobachtungsdaten zu bestimmen. Das PINN wird hierbei so trainiert, dass neben der Approximation der Lösung des Systems, ebenso die unbekannten Systemparameter in den Trainingsprozess integriert werden. So kann das PINN die Systemparameter im Verlauf der Optimierung mitlernen.

Für das inverse Problem wird die Verlustfunktion ausschließlich auf Basis der Kombination des Data-Loss mit dem Physics-Informed Loss berechnet. Der Initial-Condition Loss wird nicht berücksichtigt, da für die Schätzung der Parameter Beobachtungsdaten benötigt werden. Die Gesamtverlustfunktion lautet demnach:

$$\mathcal{L} = \alpha \cdot L_{DATA} + (1 - \alpha) \cdot L_{PHY} \quad (4.9)$$

$$\mathcal{L} = \alpha \cdot \left(\frac{1}{N_{\text{Data}}} \sum_{i=1}^{N_{\text{Data}}} \left[\left(\hat{x}(t_i) - x_{\text{data}}^{(i)} \right)^2 + \left(\hat{y}(t_i) - y_{\text{data}}^{(i)} \right)^2 + \left(\hat{z}(t_i) - z_{\text{data}}^{(i)} \right)^2 \right] \right) + (1 - \alpha)$$

$$\cdot \frac{1}{N_{\text{Phy}}} \sum_{i=1}^{N_{\text{Phy}}} \left[\left(\frac{d\hat{x}}{dt}(t_i) - \frac{dx}{dt}(t_i) \right)^2 + \left(\frac{d\hat{y}}{dt}(t_i) - \frac{dy}{dt}(t_i) \right)^2 + \left(\frac{d\hat{z}}{dt}(t_i) - \frac{dz}{dt}(t_i) \right)^2 \right] \quad (4.10)$$

Hierbei ist es ebenso möglich die Gewichtung der Verlustterme durch den Parameter α zu bestimmen. Durch die Optimierung der Verlustfunktion und der Integration der Paramater in das PINN ist eine Schätzung für unbekannte Parameter für vorliegende Messdaten möglich.

4.5 Validierung der Implementierung für nicht chaotisches Verhalten

In diesem Kapitel wird die technische Umsetzung des entwickelten PINNs untersucht. Ziel ist es zu untersuchen, ob das implementierte PINN in der Lage ist sowohl das Forward als auch das inverse Problem zu lösen. Zunächst wird anhand verschiedener Anfangswerte und unterschiedlichen Belegungen des Parameters r gezeigt, dass das PINN mittels IC-Loss dazu in der Lage ist, Vorhersagen für das Forward-Problem zu treffen. Anschließend erfolgt ein Vergleich zwischen dem implementierten datengetriebenen PINN, einem neuronalen Netz ohne physikalischen Verlustterm und einem neuronalen Netz mit L2-Regulierung. Alle drei Modelle werden unter verschiedenen Einschränkungen in den Daten evaluiert: Verrauschte, lückenhafte und nur teilweise vorhandene Daten. Abschließend wird demonstriert, dass das PINN ebenfalls zur Lösung des inversen Problems unter unterschiedlichen Bedingungen eingesetzt werden kann.

4.5.1 Validierung des IC-PINNs für Simulationsprobleme

Zur Validierung des entwickelten IC-basierten PINN wird zunächst ein exemplarisches Vorhersageproblem betrachtet. Das PINN soll in der Lage sein die Anfangswerte $(x_0, y_0, z_0) = (0{,}1{,}0)$ mit dem Parameter $\mathrm{r} = 0.5$ für einen Zeitraum $t_{\max} = 15$ für das Lorenz-System vorherzusagen. Die Wahl der Anfangsbedingungen und des Parameters wurde so getroffen, dass das resultierende Verhalten zwar nicht chaotisch ist, jedoch visuell ansprechend und dynamisch genug erscheint, um eine anschauliche und nicht-triviale Validierung des PINNs zu ermöglichen. Als Referenzwert der zur vorhergesagten Lösung dient eine mithilfe Runge-Kutta-Verfahrens (RK45) numerisch bestimmte Lösung.

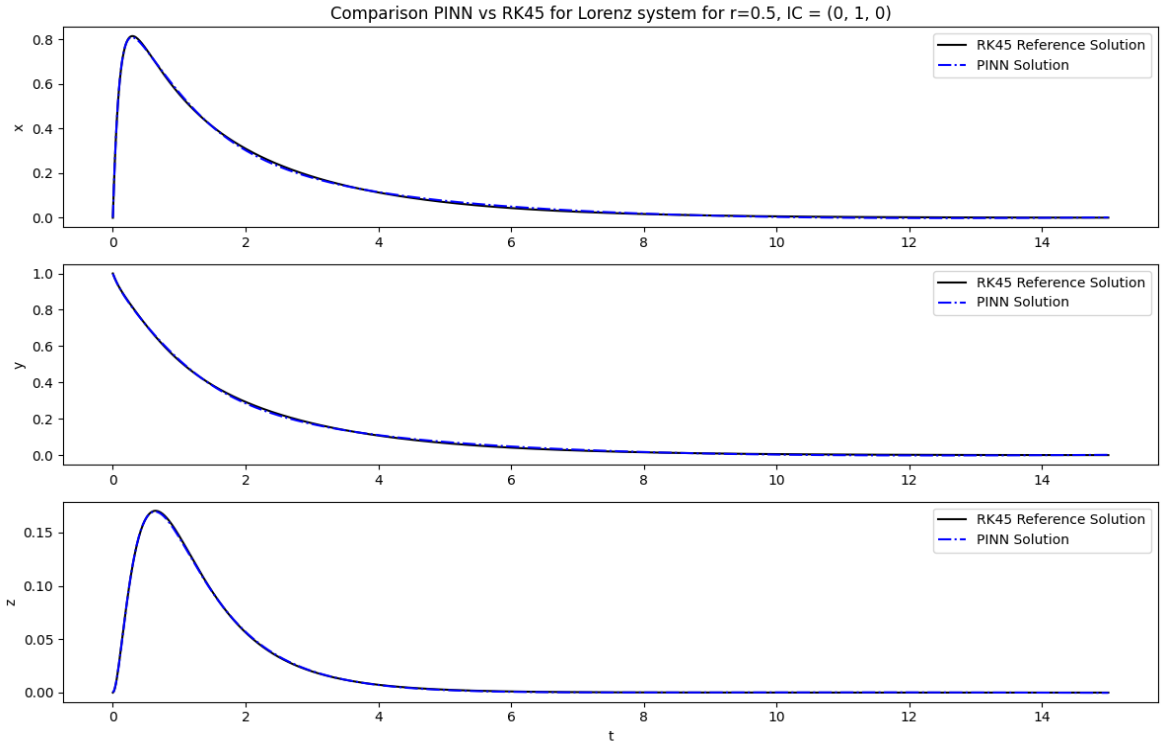

Abbildung 6: IC-PINN und Referenzlösung für nicht-chaotisches Verhalten

Abbildung 10 zeigt den zeitlichen Verlauf (horizontale Achse) der Zustandsgrößen $x(t), y(t), z(t)$ (vertikale Achse) für die Referenzlösung in schwarzer Farbe, während die Vorhersage des IC-PINN mittels gestrichelter, blauer Linien dargestellt wird. Es ist deutlich zu erkennen, dass beide Kurven nahezu deckungsgleich verlaufen. Der berechnete mittlere quadratische Fehler (MSE) zwischen der PINN-Lösung und der Referenzlösung liegt dabei ungefähr bei $MSE \approx 10^{-4}$, also nahezu bei einer minimalen Abweichung. Dies zeigt, dass die Vorhersage des IC-PINN sehr nah an der tatsächlichen, zu vorhersagenden Lösung des Lorenz-System liegt.

Um die Aussagekraft dieses Ergebnisses über den Einzelfall hinaus zu erweitern, kann das IC-PINN auf unterschiedliche Anfangs- und Parameterwerte r getestet und verglichen werden. Zur besseren Übersichtlichkeit sind die Ergebnisse dieser Untersuchungen kompakt in Form einer Heatmap dargestellt. Die Heatmap visualisiert den jeweils berechneten MSE zwischen PINN-Vorhersage und Referenzlösung für verschiedene Anfangswerte und r-Werte.

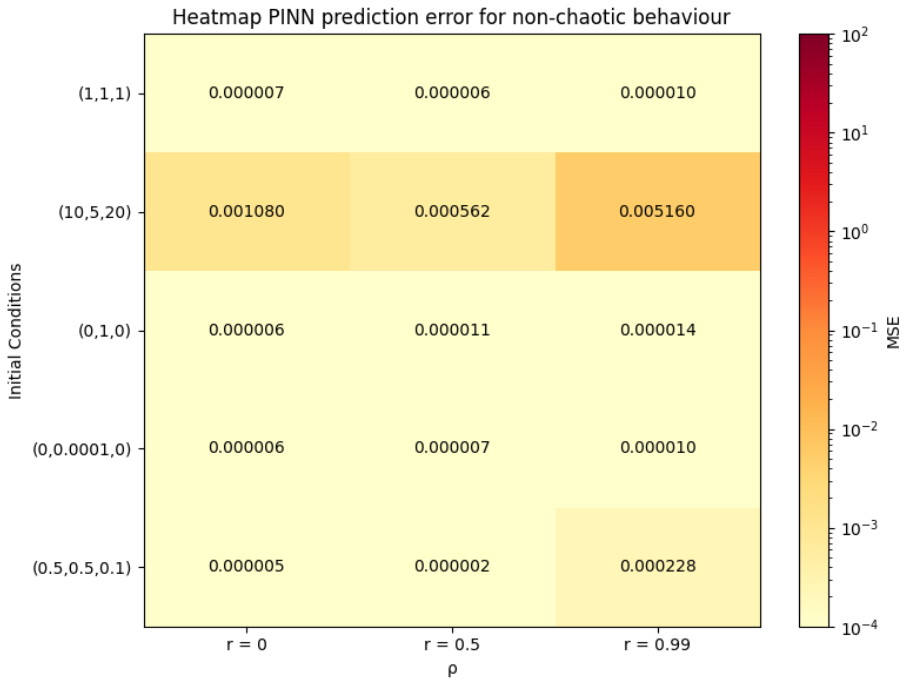

Abbildung 7: Heatmap für den Vergleich des Fehlers der PINN-Vorhersagen für verschiedene Anfangswerte und Parameter im nicht-chaotischen Bereich

In der Heatmap (Abbildung 7) entspricht jeder Eintrag einer bestimmten Kombination aus Anfangswerten und r-Werten. Die Farbgebung der einzelnen Rechtecke beschreibt den gemessenen Fehler der Vorhersage. Helle Farbtöne repräsentieren einen geringeren Fehler (gute Übereinstimmung zwischen Vorhersage und Referenzlösung) und dunklere Farbtöne einen höheren Fehler. Für die Validierung des IC-PINN für die Vorhersage von nicht-chaotischen Verhalten wurden die Anfangsbedingungen $[(1,1,1),(10,5,20),(0,1,0),(0,0.0001,0),(0.5,0.5,0.1)]$ und die Werte $[0,0.5,0.99]$ für r gewählt.

Für alle Kombinationen ist der MSE überwiegend gering. Zusammenfassend belegt die Analyse, dass das implementierte IC-PINN das Forward-Problem für nicht-chaotisches Verhalten präzise und mit hoher Genauigkeit vorhersagen kann.

4.5.2 Validierung des datengetriebenen PINNs für Simulationsprobleme

Um die Korrektheit der Implementierung datengetriebener PINNs zu evaluieren, werden systematische Untersuchungen unter verschiedenen Bedingungen durchgeführt. Ziel ist es zu überprüfen, inwiefern datengetriebene PINNs in der Lage sind das Forward-Problem zu lösen und somit das zeitliche Verhalten des zugrundeliegenden Systems zu simulieren. Zum Vergleich für die Bewertung des

datengetriebenen PINNs, dient die standardmäßige Implementierung eines neuronalen Netzes ohne physikalische Informationen, sowie ein neuronales Netz mit L2-Regularisierung verglichen. Ein L2-reguliertes neuronales Netzwerk ist eine Erweiterung des Standard neuronalen Netzes und dient der Verminderung von Overfitting (Überanpassung der Vorhersagen an die Daten), indem ein Strafterm in die Verlustfunktion integriert wird, um große einzelne Gewichtungen und hohe Modellkomplexität zu verringern.

Für die Evaluierung werden verschiedene Anfangsbedingungen für ein festes r ($r = 0{,}5$) mit jeweils drei unterschiedlichen Szenarien betrachtet. Die Charakteristiken dieser Szenarien beruhen auf der mangelnden Qualität von echten Beobachtungsdaten aus der Realität. Dies macht die Evaluierung praxisorientierter, da die untersuchten, nachfolgend aufgelisteten Szenarien Anwendung in der Realität finden:

- Verrauschte Daten
- Lückenhafte Daten
- Teilweise vorhandene Daten

Beim Szenario der verrauschten Daten erhalten die modellierten Netze die zu lernenden Daten in verrauschter Form. Hierbei werden einzelne Datenpunkte der Referenzlösung entnommen und mit einem gewissen Rauschfaktor mulitpliziert. Der Versuch unter Verwendung lückenhafter Daten nimmt die zuvor verrauschten Daten, entfernt jedoch Datenpunkte in einem gewissen Definitionsbereich. Soll die Vorhersage im Zeitintervall $t \in [0{,}15]$ erfolgen, so enthalten die verrauschten Beobachtungsdaten in bestimmten Szenarien gezielt eine Datenlücke. Diese Lücke ist beispielsweise im Teilintervall $t \in [4{,}6]$ definiert, sodass in diesem Abschnitt keine Beobachtungsdaten zur Verfügung stehen. Formal lässt sich die Menge der Zeitpunkte der verfügbaren Daten dann als $t_{data} = [0{,}4] \cup [6{,}15]$ beschreiben. In diesem Fall fehlen also absichtlich Daten im Intervall $[4{,}6]$, während außerhalb dieses Bereichs verrauschte Beobachtungsdaten aller Zustandsgrößen $x(t), y(t), z(t)$ gegeben sind. Die Modelle werden dadurch vor die Herausforderung gestellt, auch innerhalb des fehlenden Intervalls konsistente Vorhersagen zu liefern. Das dritte Szenario zeichnet sich dadurch aus, dass Beobachtungsdaten nur für eine der drei Zustandsgrößen verfügbar sind. In der Realität kann es vorkommen, dass Beobachtungsdaten nicht für

alle Zustandsgrößen gemessen werden, eine Vorhersage der nicht gemessenen Daten jedoch dennoch möglich sein soll.

Folgend wird die Vorhersage der Modelle anhand von Diagrammen der drei Szenarien am Beispiel der Anfangsbedingung $(x_0, y_0, z_0) = (1{,}1{,}1)$ dargestellt und anschließend erläutert. Eine Abbildung stellt jeweils eines der genannten Szenarien dar, indem auf der horizontalen Achse jeweils die Zeit und auf der vertikalen Achse eine der Zustandsgrößen $x(t), y(t)$ oder $z(t)$ dargestellt wird. Die Vorhersagen der drei Modelle werden jeweils im Vergleich zur numerischen Referenzlösung, sowie den im Training verwendeten Beobachtungsdaten ausgegeben.

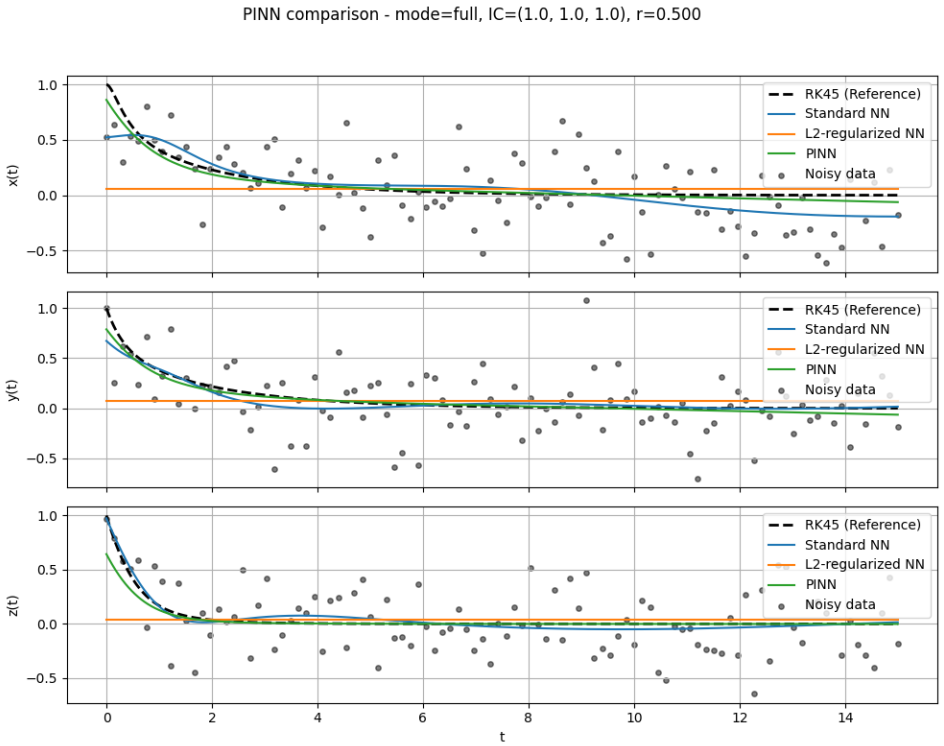

Abbildung 8: Modellvergleich mit verrauschten Daten für datengetriebene PINNs

Im ersten Szenario (Abbildung 8) liegen für die gesamte Zeitspanne Beobachtungsdaten zu allen drei Zustandsgrößen vor, jedoch sind diese mit Rauschen überlagert. Die grafische Darstellung zeigt, dass das PINN die Referenzlösung sehr gut approximieren kann. Der Verlauf der PINN-Vorhersage folgt dem Verlauf der Referenzlösung eng, ohne stark von den verrauschten Daten beeinflusst zu werden. Im Gegensatz dazu fällt auf, dass sich das Standard-NN ebenfalls gut an der Referenzlösung orientiert, sich allerdings leicht von den verrauschten Daten beeinflussen lässt. Besonders bei der Zustandsgröße $x(t)$, die jeweils am Anfang und

am Ende eine hohe Dichte an verrauschten Datenpunkten aufweist, führt dies zu Schwankungen und somit Abweichungen der Vorhersage zur Referenzlösung. Dies ist ein klassisches Symptom des Overfitting, welches bei dem Modell des Standard-NN zu erwarten ist. Das L2-regularisierte neuronale Netz zeigt in diesem Fall eine sehr flache Vorhersage, die sich kaum der Dynamik des Systems anpasst. Offensichtlich führt die Regularisierung hier zu einer zu starken Einschränkung der Modellkapazität, sodass wesentliche Aspekte der Lösung gar nicht erst gelernt werden.

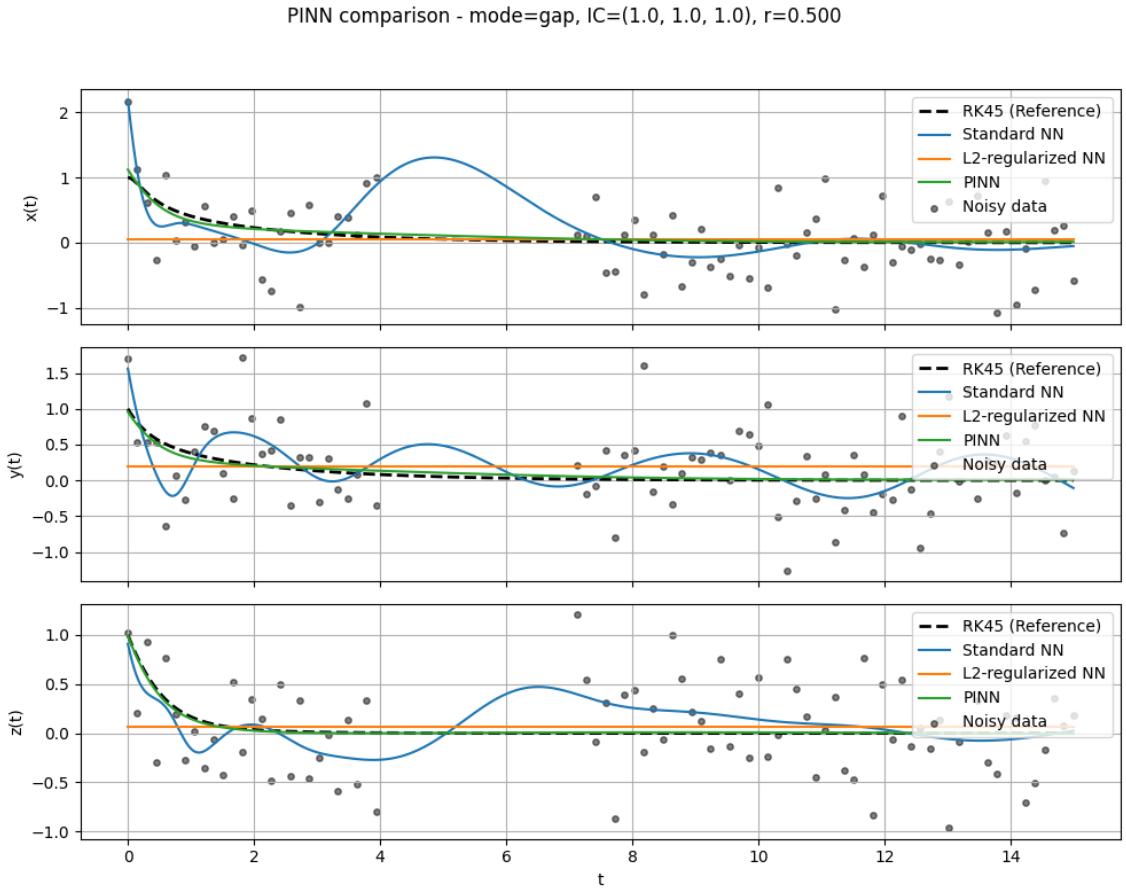

Abbildung 9: Modellvergleich mit lückenhaften Daten für datengetriebene PINNs

Im zweiten Szenario (Abbildung 9) wird die Datengrundlage zusätzlich erschwert: Im Zeitintervall zwischen $t = 4$ und $t = 7$ gibt es keine Beobachtungsdaten. Somit entsteht eine Lücke innerhalb der Trainingsdaten. Auch in diesem Fall zeigt sich das datengetriebene PINN gegenüber den anderen Modellen als überlegen. Die Vorhersage ist über den gesamten Zeitbereich hinweg konsistent und bleibt auch im lückenhaften Bereich sehr nahe an der Referenzlösung. Hier wird deutlich, dass das PINN die zugrundeliegende Dynamik erlernt hat und sie auch in datenfreien Intervallen korrekt vorhersagen kann. Demgegenüber verschlechtert sich die Qualität der

Vorhersage des Standard-NN. Das Modell bestätigt hiermit die Annahme des Overfitting aus dem vorherigen Beispiel. Während es im Bereich vor und nach der Lücke bereits fehlerbehaftete Ergebnisse liefert, ist die Vorhersage innerhalb der Lücke wesentlich von der Referenzlösung abliegend. Innerhalb der Lücke spannt die Vorhersage einen deutlich erkennbaren „Bogen“ um die Datenlücke herum - es ist nicht in der Lage die Lösung des Systems zu generalisieren. Das L2-regulierte NN zeigt auch hier keine nennenswerte Verbesserung und bleibt insgesamt unterhalb der Modellgüte des PINNs.

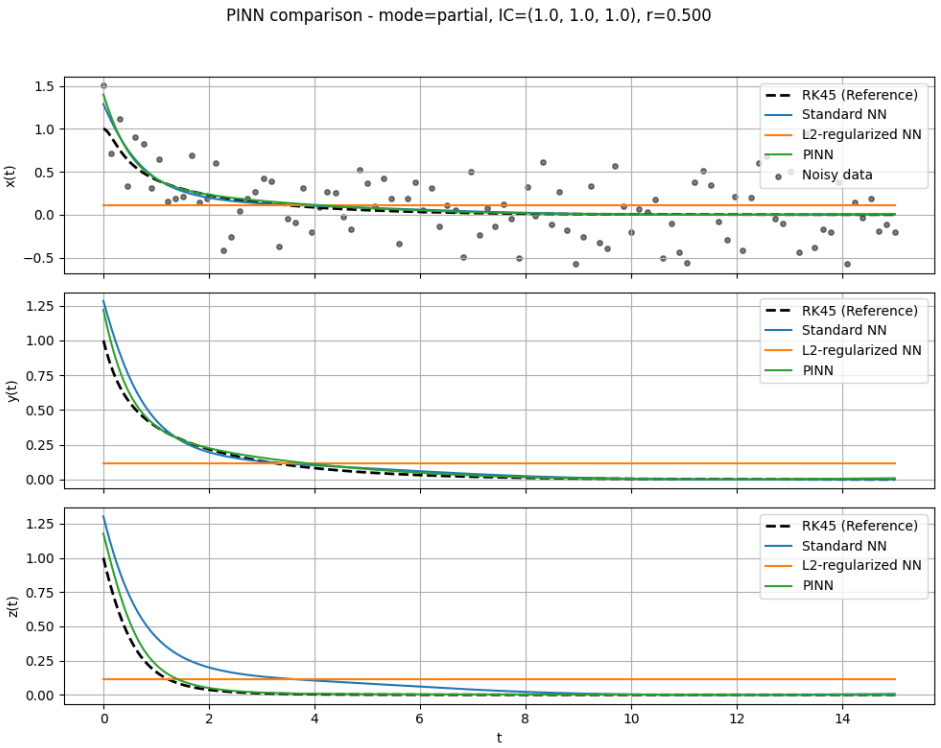

Abbildung 10: Modellvergleich mit teilweise vorhandenen Daten für datengetriebene PINNs

Für das letzte Szenario (Abbildung 10) wird ersichtlich, dass alle drei Modelle im Verhältnis zu den vorherigen Anwendungsfällen größere Schwierigkeit haben, korrekte Vorhersagen zu treffen. In diesem Szenario werden nur für die Zustandsgröße $x(t)$ Beobachtungsdaten bereitgestellt. Die Daten zu $y(t)$ und $z(t)$ fehlen vollständig, dennoch soll ihr Verlauf ausschließlich durch die vorhandenen Daten vorhersagbar sein. Die Ergebnisse zeigen, dass das PINN auch in diesem Fall sinnvolle Vorhersagen liefert. Es ist jedoch anzumerken, dass vor allem für die Zustandsgrößen $y(t)$ und $z(t)$ die Vorhersage leicht von der Referenzlösung abweicht. Das Standard-Neuronale Netz (NN) hingegen liefert für $x(t)$ eine gute Vorhersage, die Qualität des

Ergebnis lässt jedoch für die weiteren Zustandsgrößen leicht nach. Auch hier sind die Ergebnisse akzeptabel. Es ist anzumerken, dass das Standard-NN in diesem Beispiel recht akzeptabel abschneidet, für andere Anfangsbedingung hingegen größere Probleme bei der korrekte Vorhersage aufweist. Ähnliches gilt für das L2-regulierte Netz, das keine nennenswerte Verbesserung bietet, analog zu den zuvor durchgeführten Experimenten.

Die präsentierten Ergebnisse zeigen die Stärken von PINNs. Während klassische Modelle stark auf die Verfügbarkeit und Qualität der Trainingsdaten angewiesen sind, sind PINNs dazu in der Lage durch die Einbeziehung der physikalischen Gesetze, sehr gute Vorhersagen zu erreichen. Die physikalische Komponente der PINNs erhöht die Robustheit gegenüber Rauschen und führt zu einer verbesserten Generalisierungsfähigkeit. Die genannten Erkenntnisse sind nicht nur des Experiments für die Anfangsbedingung $(x_0, y_0, z_0) = (1,1,1)$ zuzuordnen, für folgende Anfangsbedingungen konnten die Ergebnisse ebenfalls nachgewiesen werden:

- $(10, 5, 20)$
- $(0, 1, 0)$
- $(0, 0.0001, 0)$
- $(0.5, 0.5, 0.1)$

Die vollständige Analyse für alle Fälle kann im begleitenden Jupyter Notebook *data_pinn_investigation.ipynb* eingesehen werden. Aus Gründen der Übersichtlichkeit wurden die Visualisierungen in diesem Kapitel auf das zentrale Beispiel beschränkt.

4.5.3 Inverses Problem für das Lorenz-System

In diesem Abschnitt wird das implementierte PINN auf die Lösung des inversen Problems für das Lorenz-System evaluiert. Hierbei wird der Systemparameter r des Lorenz-System als zu lernender Parameter betrachtet. Wie in Kapitel 3.3 erläutert beeinflusst r das Verhaltens des Lorenz-System maßgeblich, sodass hierbei die weiteren Parameter $\sigma = 10$ und $\beta = 8/3$ konstant gehalten werden. Falls die Inversion mit einem Parameter erfolgreich gelöst werden kann, ist das PINN dazu in der Lage, mehr als nur einen inversen Parameter zu bestimmen. Standardweise liegt der Wert für r initial bei $r = 1.0$. Es ist wichtig anzumerken, dass der Startwert einen großen Einfluss auf das Erlernen des korrekten Systemparameters haben kann.

Zur Validierung dieser Fähigkeit wird zunächst das Beispiel mit dem Anfangszustand $(x_0, y_0, z_0) = (10.0, -5.0, 20.0)$ und dem wahren Parameterwert $r = 1.0$ untersucht. Der tatsächlich wahre Wert von r (Ground-Truth Wert) ist lediglich dem Entwickler des PINNs, jedoch nicht dem Modell bekannt. Die Beobachtungsdaten, die für das Trainieren des PINNs für das inverse Problem nötig sind, sind wiederum mit einem Rauschfaktor synthetisch generiert.

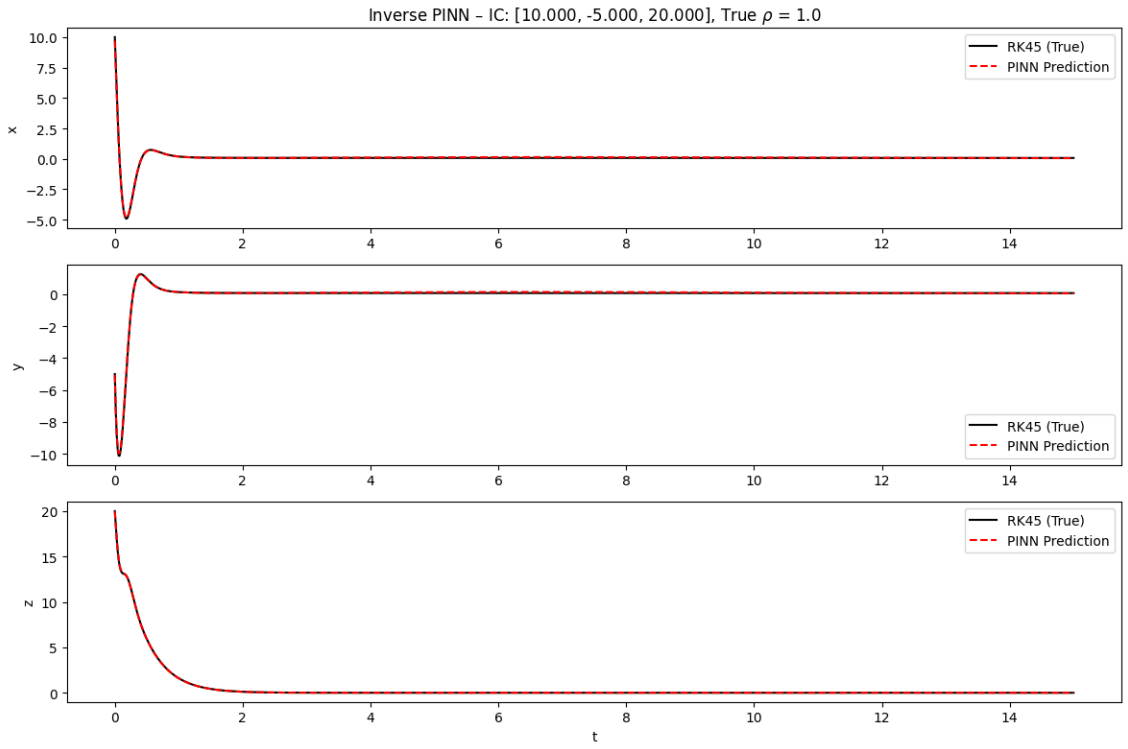

Abbildung 11: Lösung des inversen Problems

Das trainierte PINN ist fähig, sowohl den Lösungsverlauf des Systems im Zeitbereich (für alle drei Zustandsvariablen) als auch den zugrundeliegenden Parameter r mit hoher Genauigkeit zu rekonstruieren (siehe Abbildung 11). Während $r_{true} = 1.0$ ist, liegt die Vorhersage des PINNS bei $\mathrm{r}_{pred} \approx 1.0056$ – und somit nah am tatsächlichen Systemparameter. Dies zeigt exemplarisch, dass das inverse Problem im nicht-chaotischen Bereich erfolgreich durch das entwickelte PINN gelöst werden kann.

Um die Erkenntnisse des gezeigten Beispiels zu prüfen, werden folgend, systematisch weitere Kombinationen aus Anfangswerten und Belegungen des Parameters r untersucht. Konkret werden folgende Anfangsbedingungen getestet:

- $(10, 5, 20)$
- $(0, 1, 0)$
- $(0, 0.0001, 0)$

- $(0.5, 0.5, 0.1)$

Für jede dieser Anfangsbedingungen wird das inverse Problem für die Werte von r gelöst, wobei gilt $r \in [0.0, 0.5, 1.0]$. Die Ergebnisse der Parameterrekonstruktion sind in einem Streudiagramm in Abbildung 16 zusammengefasst. Dabei wird der wahre Wert von r auf der horizontalen Achse und der durch das PINN geschätzte Wert auf der vertikalen-Achse dargestellt. Jede Punktfarbe repräsentiert eine unterschiedliche Anfangsbedingung, die zusätzlich in der Legende gekennzeichnet ist. Die gestrichelte Diagonale zeigt die Idealhypothese an, bei der die Schätzung exakt dem wahren Wert entspricht. Je näher also die Vorhersage an der Diagonalen liegt, desto näher ist der geschätzte Parameter am wahren Wert r.

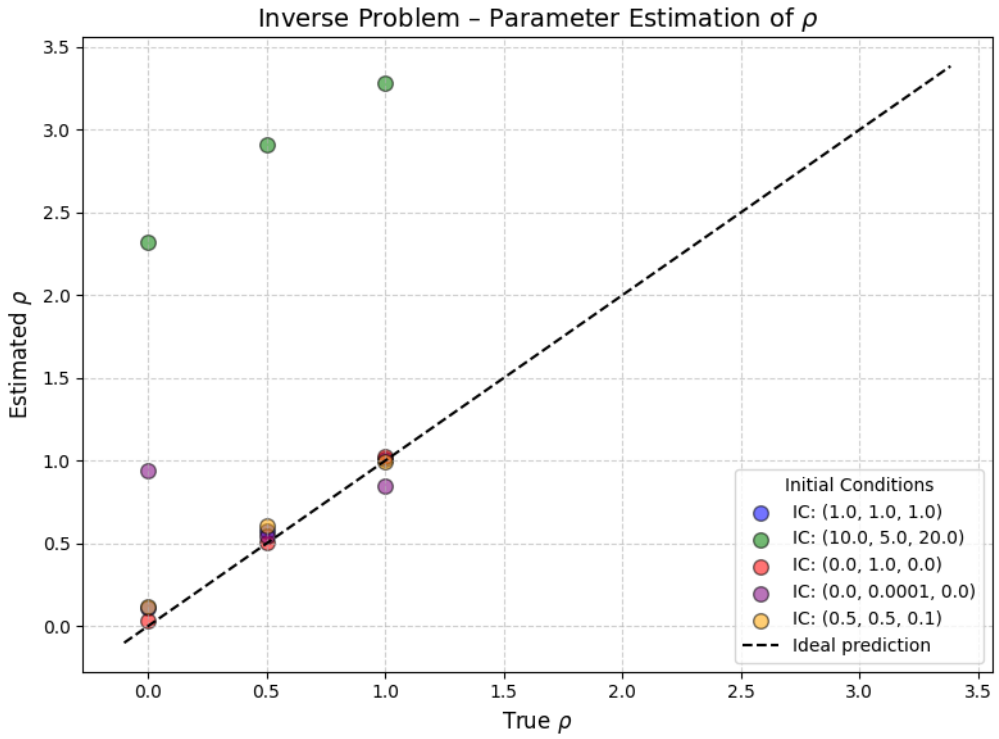

Abbildung 12: Bestimmung von r für verschiedene Anfangswerte und Belegungen von r

Insgesamt lässt sich eine gute Übereinstimmung der Schätzwerte mit den tatsächlichen Parametern beobachten (siehe Abbildung 12). Für die Untersuchungen der Anfangswerte $(x_0, y_0, z_0) = (0.0, 0.0001, 0.0)$ existieren vereinzelt kleinere Abweichungen, während die Parameterschätzung für die Anfangswerte $(x_0, y_0, z_0) = (10.0, -5.0, 20.0)$ deutlich vom wahren Wert abweicht. Eine schlechte Vorhersage der Parameter kann dabei auf verschiedene Gründe zurückgeführt werden. Entweder ist die Vorhersage durch das PINN nicht möglich, der mitgelernte Trainingsparameter

verliert im Verlauf des Trainings zunehmend an Bedeutung (Gewichtung) und „verschwindet“ im System oder das PINN identifiziert eine alternative Parametrisierung, die zwar zu einer korrekten Vorhersagekurve führt, jedoch auf einem anderen Parameterwert basiert. Es ist deshalb von großer Bedeutung, den Verlauf der zeitlichen Vorhersage auch für das inverses Problem zu untersuchen, bevor Rückschlüsse auf die Qualität und Aussagekraft der geschätzten Parameterwerte gezogen werden.

4.6 Validierung der Implementierung für periodisches Verhalten

In diesem Kapitel wird die Lösung des inversen Problems für Systemparameter betrachtet, die periodisches Verhalten hervorbringen. Dabei ist es von Bedeutung zu erwähnen, dass die Darstellung des Simulationsproblems nicht berücksichtigt wird, da das inverse Problem auf dem Forward-Problem basiert und dessen Ergebnisse implizit nutzt. Liefert das PINN für das inverse Problem gute Vorhersagen der zu bestimmenden Parameter und insbesondere der vorherzusagenden Dynamik, so kann zugleich auf die erfolgreiche Lösbarkeit des PINNs im Rahmen des Forward-Problems geschlossen werden. Zusätzlich liegt der Fokus dieser Arbeit, auf die Untersuchung der Lösbarkeit von PINNs für inverse Probleme. Im Folgenden werden daher ausschließlich die Ergebnisse und Vorhersagen des inversen Problems im periodischen Bereich vorgestellt und analysiert. Um die Genauigkeit der Modellierung periodischen Verhaltens zu verbessern, werden die Hyperparameter des PINNs im Vergleich zur vorherigen Modellierung angepasst.

Lernrate	0.01
Abklingrate	0.8
Optimierungsverfahren	Adam
Epochen	20000
Kollokationspunkte	1024
Alpha	0.5

Tabelle 3: Belegung der Hyperparameter für periodisches Verhalten

Insbesondere wird die Abklingrate von 0.09 auf 0.8 erhöht, was zu einer deutlich verbesserten Vorhersagequalität im periodischen Bereich führt. Eine systematische und umfassende Untersuchung der Hyperparameteroptimierung wurde jedoch nicht

durchgeführt aufgrund des immensen Umfangs einer derartigen Untersuchung. Zur Validierung der Fähigkeiten des PINNs bei der Inversion im periodischen Parameterbereich wird untenstehenden Fallbeispiel betrachtet. Hierbei wird das PINN auf den Anfangszustand $(x_0, y_0, z_0) = (1.0, 1.0, 1.0)$ trainiert, wobei der tatsächliche Wert des zulernenden Parameters bei $r = 5.0$ liegt.

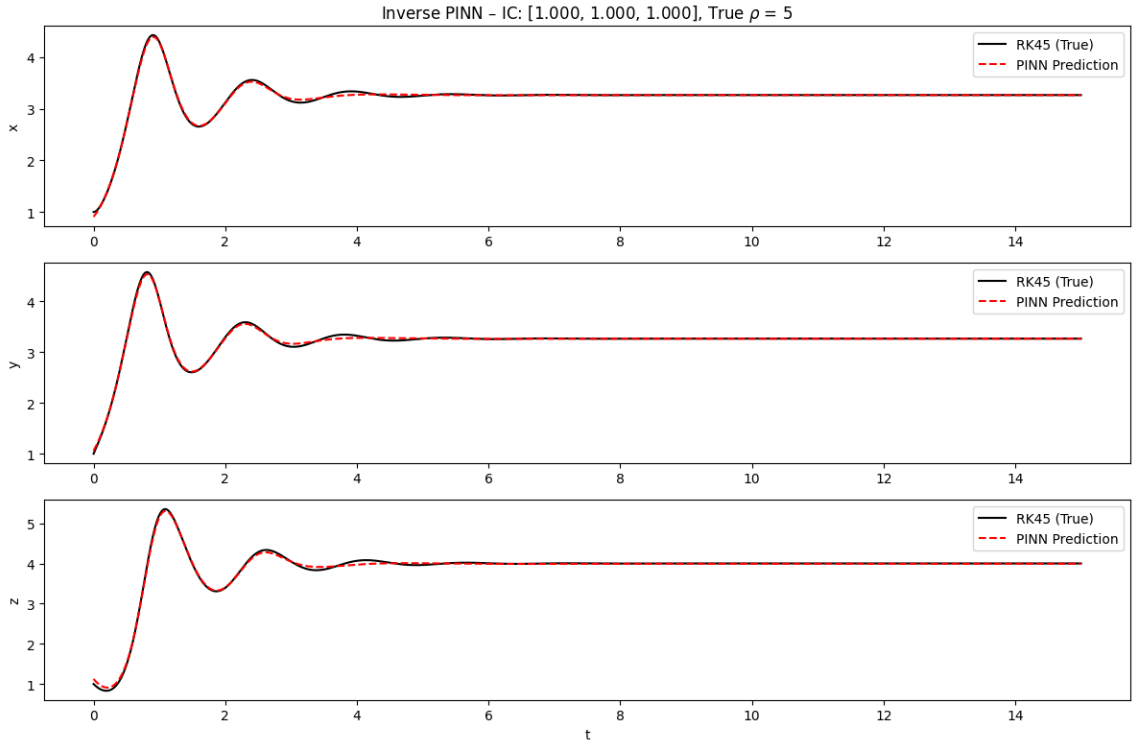

Abbildung 13: Modellvorhersage für periodisches Verhalten zur Lösung des inversen Problems

Wie in Abbildung 13 erkennbar ist, ist das trainierte PINN fähig, sowohl das Systemverhalten des Systems als auch den zugrundeliegenden Parameter r präzise zu rekonstruieren. Die Vorhersage für $r = 0.5$ beträgt $r_{pred} \approx 5{,}0003$ und liegt somit äußerst nah am tatsächlichen Wert. Durch die visuelle Überlagerung von Referenz- und Vorhersagekurve wird bestätigt, dass das System ebenfalls das Systemverhalten sehr gut vorhersagen kann.

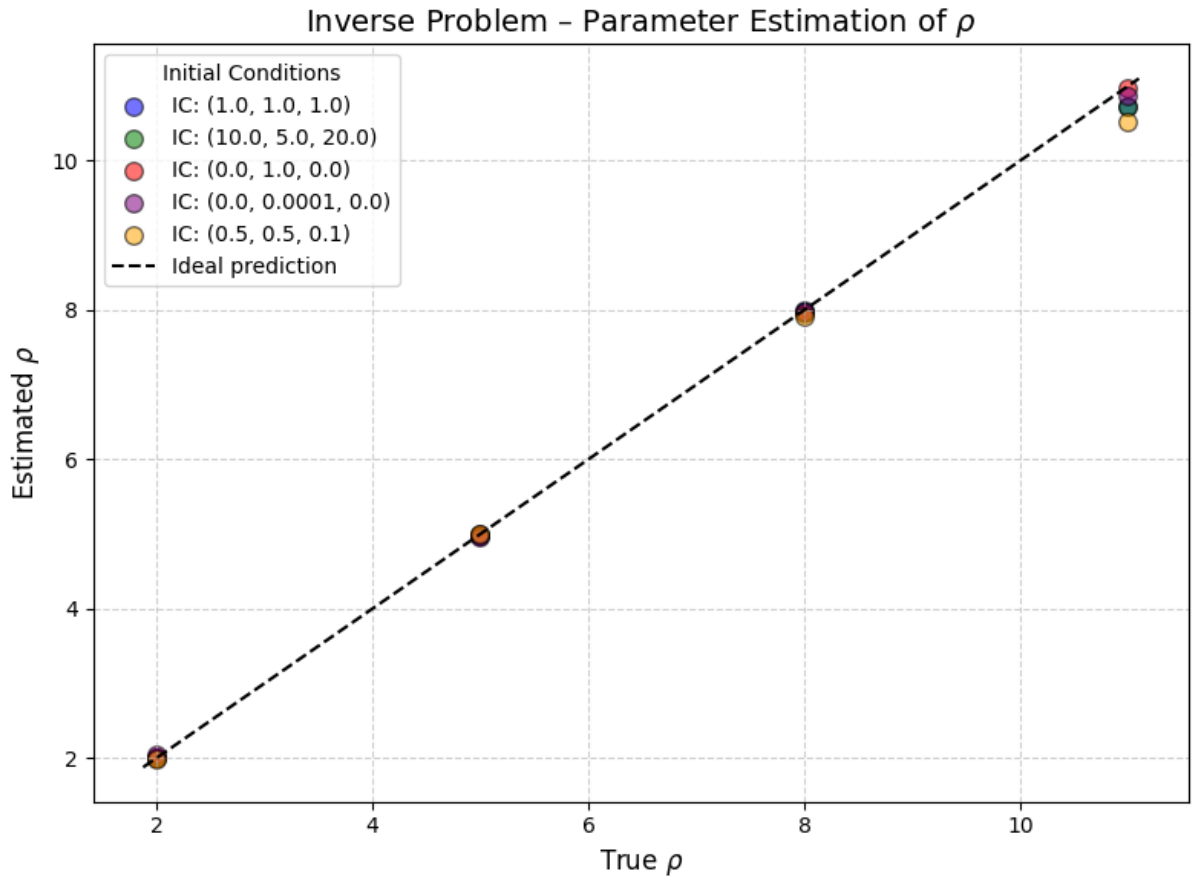

Abbildung 14: Periodische Vorhersagen für r für verschiedene Anfangswerte und Belegungen von r

Insbesondere ist festzuhalten, dass die Vorhersagequalität bezüglich Parameter und Vorhersagekurve im untersuchten periodischen Fall höher ist, als in einigen der zuvor betrachteten nicht-periodischen Szenarien. Dies zeigt sich deutlich in Abbildung 18, da die geschätzten r-Werte in der zusammenfasssenden Übersicht sehr nahe an der Idealgeraden liegen. Die Punkte im Streudiagramm weisen kaum Streuung auf, was eine präzise Vorhersage der Parameter aufzeigt.

Insgesamt lässt sich festhalten, dass das implementierte PINN im periodischen Bereich die äußerst robuste Rekonstruktion der Systemdynamik und des Modellparameters r ermöglicht. Dieses Ergebnis ist insofern bemerkenswert, als dass man im periodischen Bereich, der Vorstufe zum chaotischen Bereich, tendenziell eine höhere Unsicherheit verglichen mit nicht-chaotischen Verhalten erwarten würde. Die hohe Genauigkeit in der Parameterschätzung und die präzise Vorhersage der Zustandsgrößen weisen somit die valide Implementierung des PINNs im periodischen Bereich für das inverse Problem nach.

5 Ergebnisse

In diesem Kapitel wird untersucht, inwiefern PINNs schließlich dazu in der Lage sind, das Verhalten chaotischer Systeme vorherzusagen. Aufbauend auf der in den vorherigen Kapiteln beschriebenen Implementierung der PINNs und den optimierten Hyperparametern aus den Untersuchungen für die Vorhersage von periodischem Verhalten, wird zunächst ermittelt, in welcher Qualität die aktuelle Konfiguration des PINNs die Vorhersage von chaotischen Parametern ermöglicht. Ziel dieser Analyse ist es, eine Ausgangsbasis zu schaffen, die die Fähigkeiten der validierten Modelle für die Vorhersage von chaotischem Verhalten darlegt. Im Anschluss daran wird systematisch analysiert, weshalb die Vorhersagegenauigkeit in chaotischen Bereichen eingeschränkt ist. Abschließend werden potenzielle Ansätze zur Verbesserung der Vorhersageleistung untersucht, beispielsweise durch die Anpassung der PIINN-Architektur, Modifikation im Trainingsprozess oder mittels Methodiken zur Vermeidung von trivialen Lösungen. Ziel ist es, Wege und Ergebnisse der Untersuchungen aufzuzeigen, wie PINNS robuster gegenüber chaotisches Verhalten gemacht werden können.

5.1 Ergebnisse der Simulation im chaotischen Bereich

Inhalt dieses Abschnittes ist die Untersuchung, ob sich die für das periodische Systemverhalten optimierten Hyperparameterkonfigurationen auch auf chaotische Dynamiken übertragen lassen. Ziel ist es eine Ausgangsbasis zu schaffen, anhand derer die Vorhersagequalität der implementierten PINN-Modelle im chaotischen Bereich bewertet werden kann.

Dazu werden unterschiedliche Vorhersagen durchgeführt, bei denen jeweils unterschiedliche Anfangsbedingungen gewählt werden.

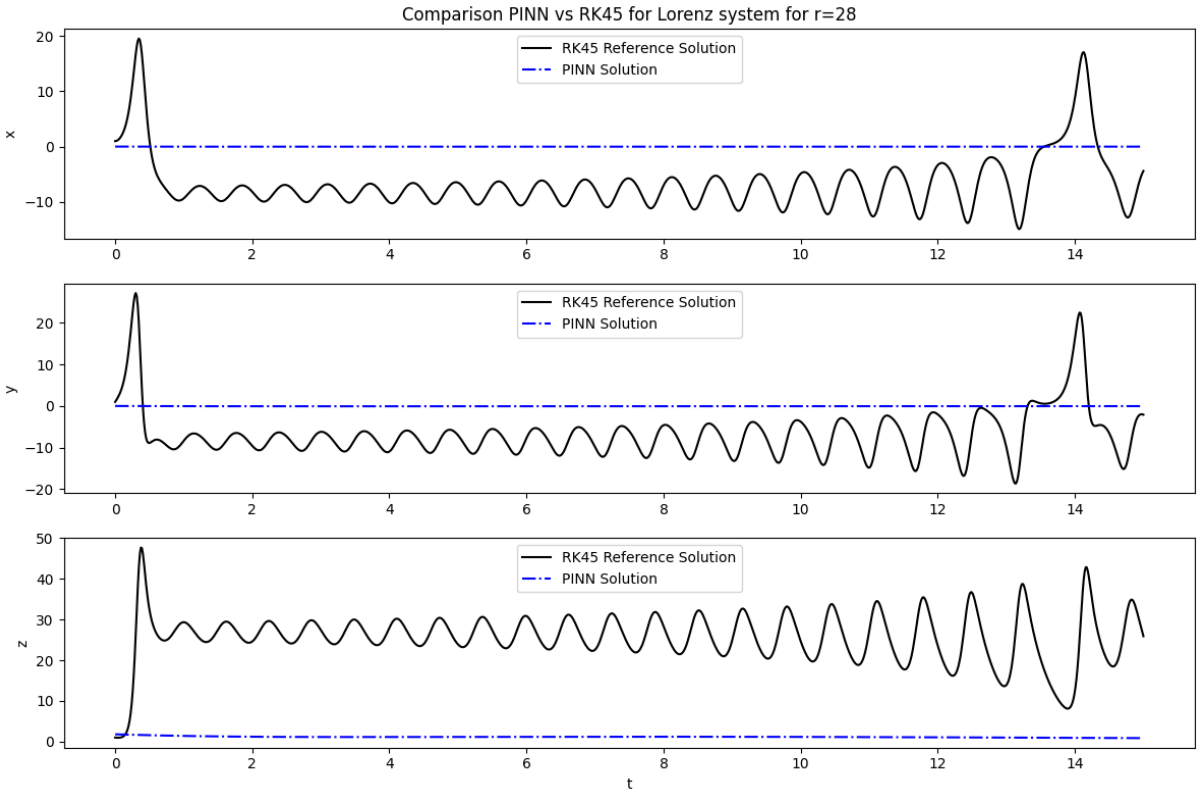

Abbildung 15: PINN-Vorhersage für chaotisches Verhalten bei periodischer Konfiguration (1.0, 1.0, 1.0)

Im Fallbeispiel 1 (Abbildung 15) wird mit der Anfangsbedingung $(x_0, y_0, z_0) = (1.0, 1.0, 1.0)$ und $r = 28$. gearbeitet. Es ist zu erkennen, dass die Vorhersage stark von der Referenzlösung abweicht.

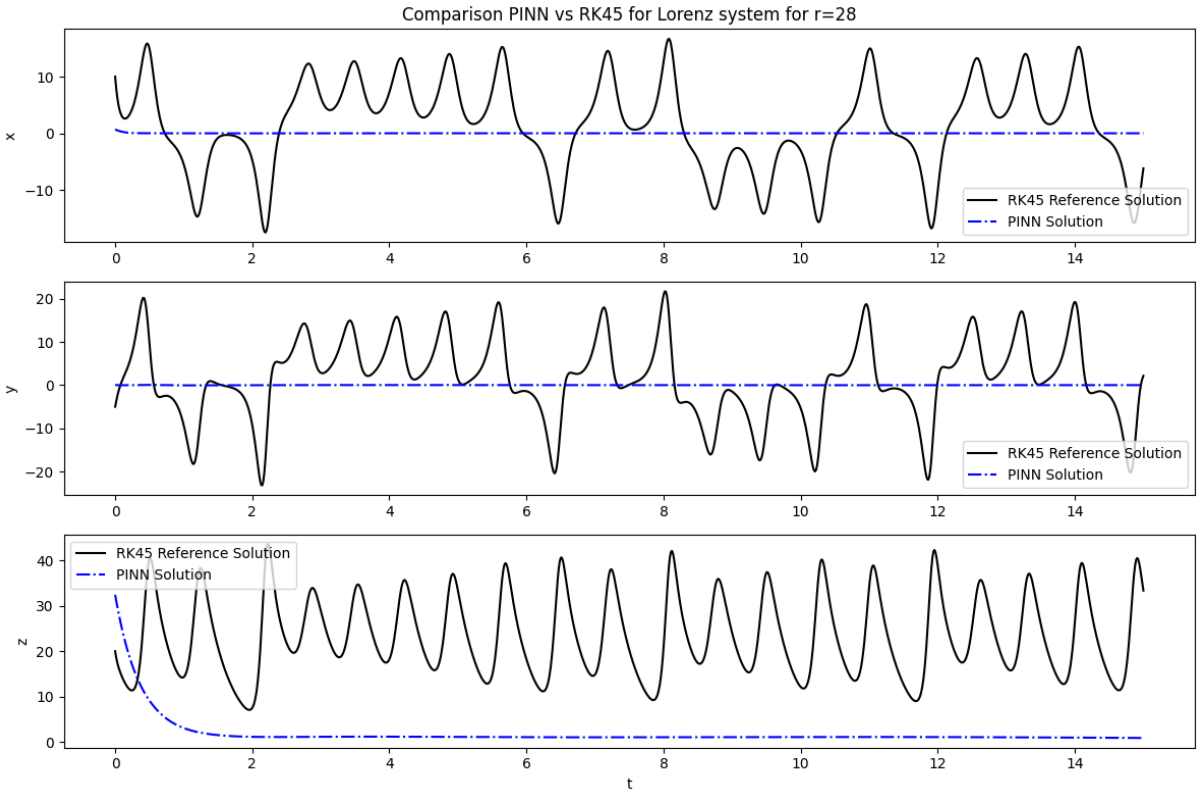

Abbildung 16: PINN-Vorhersage für chaotisches Verhalten bei periodischer Konfiguration (10.0, -5.0, 20.0)

Fallbeispiel 2 (Abbildung 16) nutzt Anfangsbedingung $(10.0, -5.0, 20.0)$ ebenfalls mit $r = 28$. Vergleichbar mit dem ersten Beispiel ist die Vorhersage auch für weitere Anfangsbedingungen mangelhaft.

In Beiden Fällen zeigen sich deutliche Probleme in der Vorhersagefähigkeit der Modelle: Der zu rekonstruierende Lösungsverlauf weicht stark von der tatsächlichen Systemdynamik ab. Die Erkenntnisse für das Forward-Problem in Bezug auf chaotisches Verhalten lassen sich analog auf das inverse Problem übertragen.

Ein zentrales Problem für die Vorhersage ist darauf zurückzuführen, dass die PINN-Modelle während des Trainings in einem lokalen Minimum der Verlustfunktion verharren, anstatt das gesuchte globale Minimum zu finden. Dies führt dazu, dass das Netzwerk in frühem Stadium des Trainingsprozess kaum mehr Optimierungsschritte vornimmt und nicht in der Lage ist die Dynamik des Systems zu erlernen. PINNs haben die negative Eigenschaft, zur Bestimmung der trivialen Lösung zu tendieren. Hierbei werden mathematisch einfache, aber physikalisch nichtssagende Approximationen für die Lösung der Vorhersage bevorzugt. Die triviale Lösung in diesem Kontext bezeichnet eine Vorhersage, bei der das Modell einen möglichst simplen Verlauf lernt,

der die Verlustfunktion zu einem gewissen Grad minimiert, jedoch nicht in der Lage ist die zugrundeliegende Dynamik des Systems korrekt abzubilden. Ein einfaches Beispiel für die Visualisierung des Problems der trivialen Lösung ist die Vorhersage des Verlaufs der trigonometrischen Funktion $y = sin(x)$. Beispielsweise kann das System die Lösung der horizontalen Geraden $y = 0$ bevorzugen, da diese Lösung die physikalischen Bedingungen der zugrundeliegenden Gleichungen erfüllt und einen niedrigen Wert des Verlustterms erzeugt, obwohl die Lösung selbst aus physikalischer Sicht vollkommen ungeeignet ist.

Dieses Verhalten macht deutlich, dass die direkte Übertragung der anhand des periodischen Verhaltens optimierten Modelle auf chaotische Systeme nicht möglich ist. Die Ergebnisse dieser Experimente bilden somit die Grundlage für weiterführende Untersuchungen zur Verbesserung der Vorhersagen und zur gezielten Vermeidung von trivialen Lösungen.

5.2 Analyse der Konvergenzprobleme

In diesem Unterkapitel wird systematisch untersucht, wie die Konvergenzprobleme der PINNs im chaotischen Bereich verbessert werden können. Der Fokus liegt dabei auf datengetriebene PINNS, da die IC-basierten PINNs starke Probleme bei der Vorhersage im chaotischen Bereich aufweisen und für das inverse Probleme ohnehin datenbasierte Modelle erforderlich sind. Zunächst wird die Auswirkung verschiedener Hyperparamater bezüglich der Verbesserung der Vorhersage und Konvergenz analysiert. Dabei werden gezielt die Anzahl der Kollokationspunkte, die Lernrate sowie die Abklingrate angepasst. Zusätzlich wird der Einfluss des Gewichtungsfaktors α, der das Verhältnis zwischen datenbasiertem und physikalischem Verlust steuert, untersucht. Auch architektonische Veränderungen der Modelle wie die Erhöhung der Anzahl an Hidden Layers und Neuronen pro Schicht werden erprobt, um die Vorhersagequalität des Netzwerks zu verbessern. Im nächsten Schritt wird über die reine Hyperparameteroptimierung hinaus ein dynamisches Gewichtungsverfahren für des Gewichtungsfaktor α implementiert, welches als zusätzlicher Parameter ins Training des PINNs integriert wird. Ergänzend dazu werden Methoden, die speziell auf die Vermeidung von trivialen Lösungen ausgerichtet sind umgesetzt und untersucht. Zum Abschluss werden die Ergebnisse dieser Optimierungsmaßnahmen

zusammengeführt und die finale Leistungsverbesserung anhand konkreter Beispiele demonstriert.

5.2.1 Hyperparameteroptimierung zur Verbesserung der Vorhersage chaotischer Dynamik

In diesem Abschnitt wird untersucht, inwieweit die gezielte Anpassung von Hyperparametern und Trainingskonfigurationen dazu beitragen, die Vorhersageleistung von PINNs im Kontext chaotischer Systemdynamiken in Relation zur gezeigten Ausgangsbasis zu verbessern. Der Begriff der Basiskonfiguration des PINNs, auf der die nachfolgenden Anpassungen aufbauen, bezieht sich auf die Hyperparameter der zuvor dargestellten Validierung bei der Vorhersage von periodischem Verhalten.

In einem ersten Schritt werden die Kollokationspunkte als Faktor für die Vorhersagegenauigkeit betrachtet. Die Erhöhung der Anzahl der Kollokationspunkte führt zu einer besseren Trainingsgenauigkeit und somit zu einem geringeren Wert der Gesamtverlustfunktion. Dennoch ist das Modell nicht in der Lage die chaotischen Lösungsverläufe korrekt vorherzusagen. Das PINN-Modell leidet weiterhin am Problem der trivialen Lösungen und bevorzugt konstante Ausgaben, die den Verlust zu einem gewissen Grad minimieren, jedoch die physikalischen Gesetzmäßigkeiten nicht einhalten.

Untersucht man die Lernrate des Modells, durch Variation im Intervall von $[0.0095, 0.011]$, so zeigt sich, dass keine der getesteten Lernraten zur Verbesserung der Vorhersageleistung für chaotisches Verhalten führt. Es ist wichtig anzumerken, dass eine pauschale Aussage über eine höhere bzw. niedrigere Lernrate nicht möglich ist, sondern dass die Effektivität der gewählten Rate immer vom vorliegenden Problem abhängig ist. Ein ähnliches Bild ergibt sich bei der Untersuchung der Abklingrate. Hier wurden die Werte 0.095; 0.09; 0.8 und 0.7 verglichen. Auch wenn sich für das periodische Verhalten insbesondere ein Wert von 0.8 als vorteilhaft herausgestellt hatte, zeigen die Unterschiede im Training chaotischer Systeme nur geringe Auswirkungen auf die Modellleistung. Aufgrund der bewährten Stabilität der Abklingrate bei einem Wert von 0.8 bei der Analyse des periodischen Parameterbereichs, wird dieser Wert im weiteren Verlauf beibehalten.

Letzte Untersuchungen zeigen, dass die Anpassung der Netzarchitektur durch die Erhöhung der Anzahl der Hidden Layer von 6 auf 10 und Anzahl der Neuronen pro Schicht von 30 auf 50 die Trainingsgenauigkeit verbessern. Jedoch bleibt die Fähigkeit der Vorhersage von chaotischer Dynamik begrenzt.

Folglich zeigen die Ergebnisse, dass klassische Hyperparameteranpassungen allein nicht ausreichen, um chaotisches Verhalten mittels PINNs zu modellieren. Trotz verbesserter Genauigkeit im Training bzw. der Reduzierung des Verlustes manifestiert sich die Vorhersagequalität in Form von trivialen Lösungen, was die Notwendigkeit weiterführender Optimierungsstrategien begründet.

5.2.2 Dynamische Gewichtung von datenbasiertem und physikalischem Verlustterm

Um die triviale Lösung zu vermeiden, soll der Gewichtungsfaktor α dynamisch bestimmt und dessen Einfluss auf die Vorhersagequalität untersucht werden. Wie bereits erläutert, beschreibt α das Verhältnis der Gewichtungen des physikalischen und des datenbasierten Anteils der zu optimierenden Verlustfunktion. Anstatt α statisch festzulegen, wird dieser Hyperparameter als trainierbarer Parameter direkt in den Trainingsprozess des PINN-Modells integriert. Somit lässt sich der Gewichtungsfaktor dynamisch durch das Modell erlernen, abhängig vom aktuellen Trainingszustand des Modells. Als Vorteil dieser Methode ist zu nennen, dass der Gewichtungsfaktor dynamisch an die Herausforderungen im Optimierungsprozess angepasst werden kann, wodurch der gesamte Trainingsvorgang verbessert wird. Zudem ermöglicht die dynamische Bestimmung von α eine für das vorliegende Problem – gekennzeichnet durch Anfangsbedingungen und Belegung von r - eine Ausrichtung von α, um einen Richtwert für die Wahl von α für vorliegende Probleme zu bestimmen.

Der in Kapitel 4 beschriebene Gesamtverlust für vorliegende Probleme ergibt sich formal zu:

$$\mathcal{L} = \alpha \cdot L_{DATA} + (1 - \alpha) \cdot L_{PHY} \qquad (5.1)$$

Dabei wird $\alpha \in [0; 1]$ während des Trainings mitgelernt- und optimiert. Beobachtungen während der Untersuchungen zeigen, dass α bereits bei niedriger Anzahl an Trainingsepochen rapide gegen den Wert "1" (das Maximum) konvergiert. Dies bedeutet, dass das Netzwerk den Datenverlust nahezu vollständig priorisiert und den physikalischen Verlustterm vernachlässigt. Dieses Verhalten lässt sich darauf

zurückführen, dass neuronale Netzwerke zu Beginn des Trainings keine guten Approximationen der zu lernenden Trainingsdaten liefern. Da vor allem der physikalische Verlust im Vergleich zum Datenverlust zu Beginn des Trainings wesentlich geringer ist, drängt das Optimierungsverfahren, α zu maximieren, um den dominanten Fehlerteil stärker im Training zu gewichten. Dies führt zu dem Problem, dass der physikalische Verlustterm nicht mehr berücksichtigt wird, obwohl dieser essenziell ist, um die physikalischen Gesetzmäßigkeiten generalisieren zu können.
Interessanterweise zeigen Trainingsprozesse, in dem nur der datengetriebene Anteil berücksichtigt wird, im Bereich des chaotischen Regimes gute Approximationen der Referenzlösung. Insbesondere bei PINNs mit vielen Kollokationspunkten, ermöglicht es das PINN die zugrundeliegenden dynamischen Strukturen im chaotischen Bereich vorherzusagen.

5.2.3 Methoden zur Vermeidung trivialer Lösungen

Leiteritz und Pflüger schlagen in ihrer Arbeit zwei Methoden vor, um das Auftreten trivialer Lösungen in Bezug auf PINNs zu vermeiden. Folgend soll untersucht werden, ob diese Methoden die Konvergenzprobleme des PINN-Modells im chaotischen Bereich verbessern können. Die beiden Autoren machten dabei folgende Beobachtungen:[91] Sobald sich die Optimierung des PINNs gegen eine triviale Lösung tendiert, macht sich dies im physikalischen Verlustterm der Gesamtverlustfunktion durch eine starke Änderung des Gradienten bemerkbar. Um diese stark-ansteigenden, instabilen Übergänge zu unterbinden, wird ein zusätzlicher Term in die Gesamtverlustfunktion eingeführt, der einen hohen Anstieg des Gradienten des physikalischen Verlustterms bestraft.

$$\mathcal{L}_{\square\square\square} = \boldsymbol{max} \parallel \nabla \mathcal{L}_{phy}(x;\theta) \parallel \tag{5.2}$$

Dabei bezeichnet $\mathcal{L}_{pen}$ den zusätzlichen Bestrafungsterm, welcher den größten Steigungswert über alle Kollokationspunkte identifiziert, basierend auf dem Gradienten des physikalischen Verlusts. Die Parameter x und θ bezeichnen dabei die Eingabe und die Parameter des PINNs. Es ergibt sich folgende Gesamtverlustfunktion, die es zu minimieren gilt:

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{Data}} + (1-\alpha) \cdot \mathcal{L}_{\text{Phy}} + \mathcal{L}_{pen} \tag{5.3}$$

[91] Vgl. Leiteritz und Pflüger, „How to Avoid Trivial Solutions in Physics-Informed Neural Networks“, 3.

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\text{Data}} + (1-\alpha) \cdot \mathcal{L}_{\text{Phy}} + \boldsymbol{max} \parallel \nabla \mathcal{L}_{phy}(x;\theta) \parallel \quad (5.4)$$

Durch die Bestrafung starker Gradienten soll die Wahrscheinlichkeit reduziert werden, dass das Netzwerk in lokale Minima konvergiert.

Darüber hinaus adressieren Leiteritz und Pflüger die Bedeutung der Auswahl geeigneter Kollokationspunkte. Es wird empfohlen die Kollokationspunkte regelmäßig über die zeitliche Domäne zu verteilen, anstatt diese zufällig auszuwählen. Dies verbessert die Genauigkeit von PINNs insbesondere bei Modellen mit wenigen Kollokationspunkten.

Beide genannten Methoden zielen darauf ab, die Optimierung von PINNs robuster gegenüber typischen Konvergenzproblemen zu machen. Die Umsetzung dieser Methoden – also die Bestrafung starker Gradienten im physikalischen Verlust sowie die Nutzung regelmäßiger Verteilung der Kollokationspunkte in der Domäne, zeigt bei der Vorhersage chaotischer Dynamiken jedoch keine signifikante Verbesserung der Ergebnisse. Trotz theoretischer Vorteile konnte weder das Verhindern trivialer Lösungen noch eine stabilere Approximation der Lösung erreicht werden. Die Beobachtungen zeigen, dass PINNs weiterhin in lokal optimalen, physikalisch aber irrelevanten Lösungen stecken bleiben.

5.3 Anwendung der Analyseergebnisse

Durch die systematische Untersuchung verschiedener Maßnahmen zur Verbesserung der Vorhersagen des PINNs im chaotischen Parameterbereich, ist es nun möglich, alle zur Verbesserung beitragenden Methoden kombiniert anzuwenden.

Unter Anwendung dieser Resultate soll die Lösung des inversen Problems bei der Wahl des Parameters $r = 28$ untersucht werden. Für diese Belegung des Systemparameters werden zusätzlich unterschiedliche Anfangsbedingungen untersucht, um zu testen, wie sich verschiedene Anfangsbedingungen auf die Vorhersagequalität auswirken. Zudem liefert die die Untersuchung verschiedener Anfangszustände Hinweise auf die Robustheit und mögliche Generalisierbarkeit der Ergebnisse. Der Parameter r, der für die Lösung des inversen Problems als lernbare Größe in das Modell integriert wird, ist in folgenden Untersuchungen initial auf den Wert 1.0 gesetzt. Damit befindet sich der Startwert für r weit entfernt vom vorherzusagenden Wert $r_{true} = 28$.

Zur Verbesserung der Vorhersagequalität werden die optimalen Konfigurationen des PINNs für die Vorhersage des periodischen Verhaltens übernommen und auf Basis der in den vorigen Abschnitten aufgezeigten Untersuchungen angepasst. Die Konfiguration des PINNs, sowie die Belegung der Hyperparameter für das Training im chaotischen Bereich ist untenstehender Tabelle zu entnehmen.

Hidden Layer	10
Neuronen pro Schicht	50
Aktivierungsfunktion	Silu
Gewichtsinitialisierung	Glorot Uniform
Lernrate	0,01
Abklingrate	0,8
Optimierungsverfahren	Adam
Epochen	20000
Kollokationspunkte	4000
Alpha	0,99

Tabelle 4: Optimierte Konfiguration und Hyperparameterbelegung des PINNs

Dabei sind folgende Änderungen der Konfiguration des PINNs und der Hyperparameterbelegung besonders hervorzuheben.

- Die Anzahl der Hidden Layer wurde von 6 auf 10 erhöht
- Die Anzahl der Neuronen pro Schicht wurde von 30 auf 50 angehoben
- Die Anzahl der Kollokationspunkte beträgt nun 4000 statt 1024
- Der Gewichtungsfaktor α für die Gewichtung des Datenverlustterms wurde auf 0.99 gesetzt, da vorherige Untersuchungen ein stark datengetriebenes Training als vorteilhaft gezeigt hatten
- Weitere Hyperparameter orientieren sich an den vorherigen Untersuchungen zum periodischen Systemverhalten

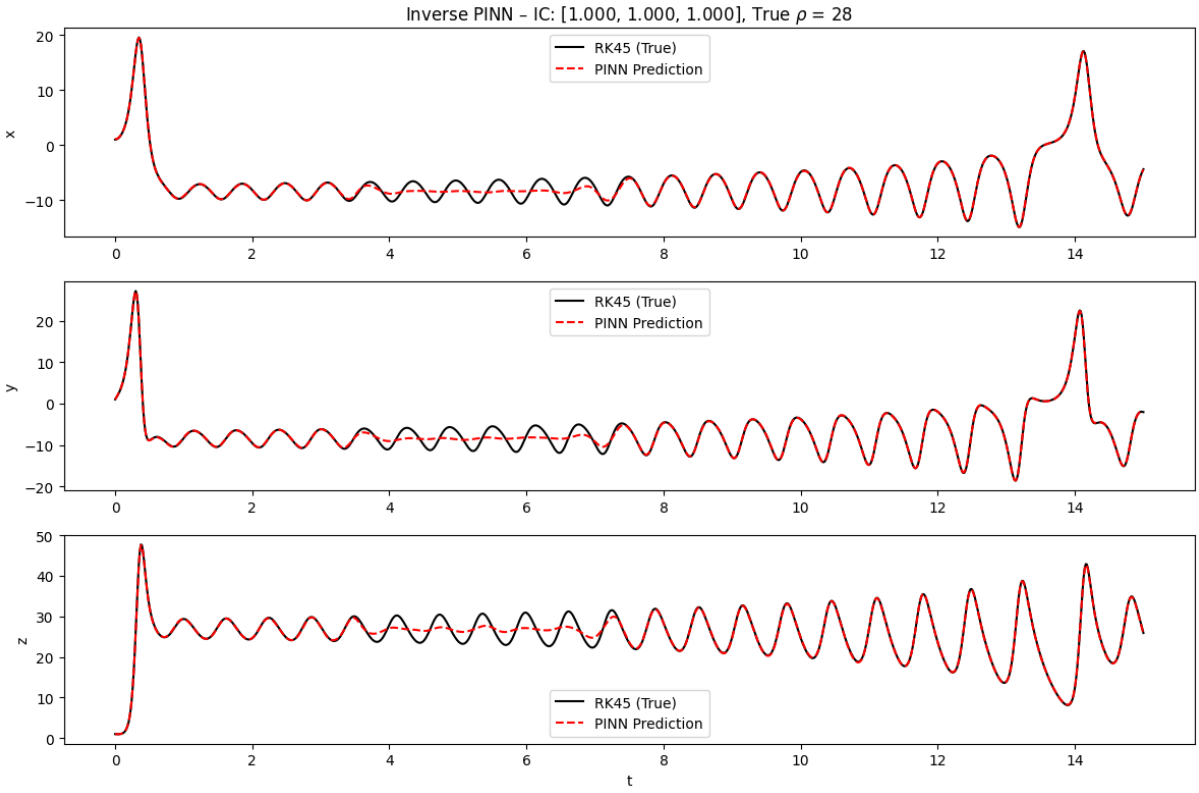

Abbildung 17: PINN-Vorhersage für chaotisches Verhalten für Anfangswerte (1.0, 1.0, 1.0)

In Abbildung 17 wird exemplarisch die Systemdynamik des chaotischen Lorenz-Systems für die Anfangswerte $(x_0, y_0, z_0) = (1.0, 1.0, 1.0)$ gezeigt. Auf der horizontalen Achse ist die der zeitliche Verlauf dargestellt, auf der horizontalen Achse die drei Zustandsgrößen des Lorenzsystems. Die schwarze Kurve repräsentiert die Referenzlösung, während die rote Kurve die PINN-Vorhersage darstellt. Es zeigt sich, dass die Dynamik weitgehend korrekt rekonstruiert werden kann, wobei im mittleren Zeitbereich leichte Abweichungen erkennbar sind. Zwischen $t = 4$ und $t = 7$ ist eine kurzzeitige Tendenz zur trivialen Lösung erkennbar, die jedoch nicht dauerhaft anhält, sodass das Modell anschließend wieder korrekte Vorhersagen liefert. Die Abschätzung des Parameterwertes durch das PINN liegt bei $r_{pred} = 28{,}0057$ und ist damit nahezu perfekt an der vorherzusagenden Lösung von $r = 28$.

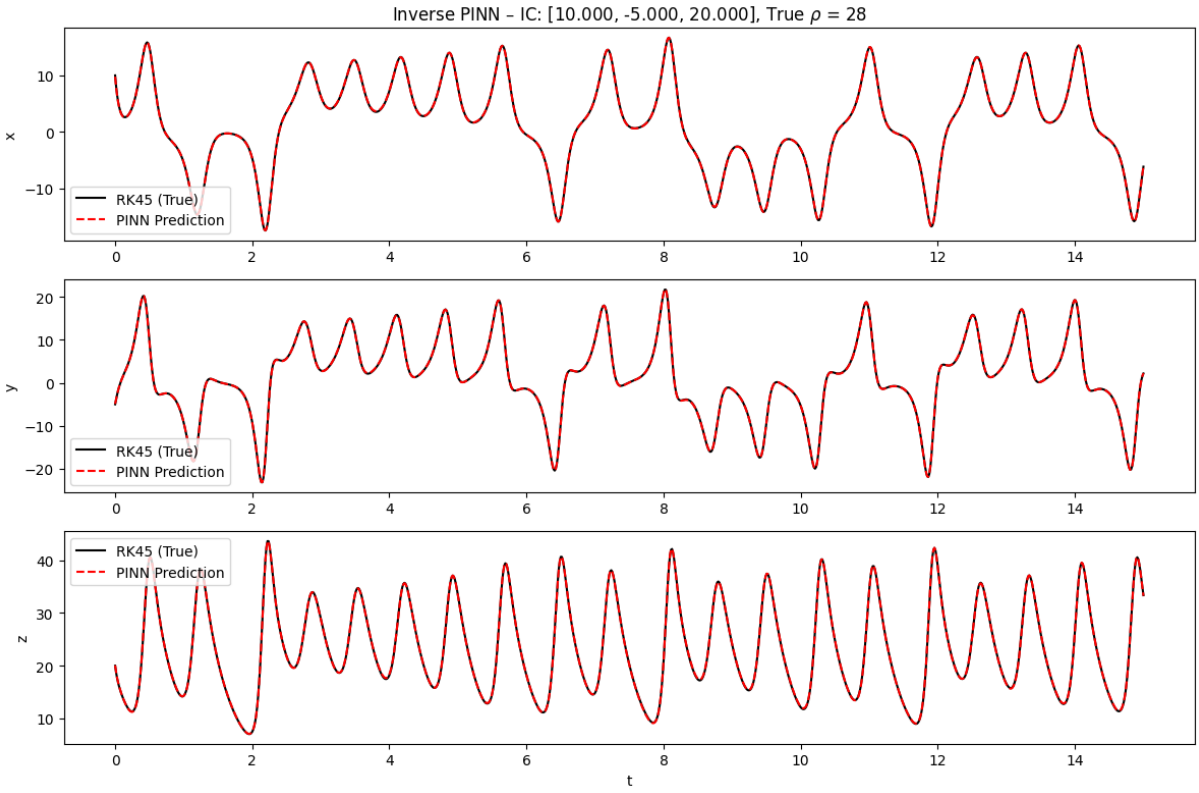

Abbildung 18: PINN-Vorhersage für chaotisches Verhalten für Anfangswerte (10.0, -5.0, 20.0)

Abbildung 18 zeigt das Ergebnis für die Wahl der Anfangsbedingungen $(x_0, y_0, z_0) = (10.0, -5.0, 20.0)$. Hier fällt die Übereinstimmung der Vorhersage und Referenz im Vergleich zum ersten Fallbeispiels noch besser aus. Eine Abweichung der Vorhersage von der Referenzlösung ist auf den ersten Blick nicht erkennbar. Ebenso überzeugt die Parameterschätzung die mit einem Wert von bei $r_{pred} = 27{,}8838$ nah am tatsächlichen Parameter liegt.

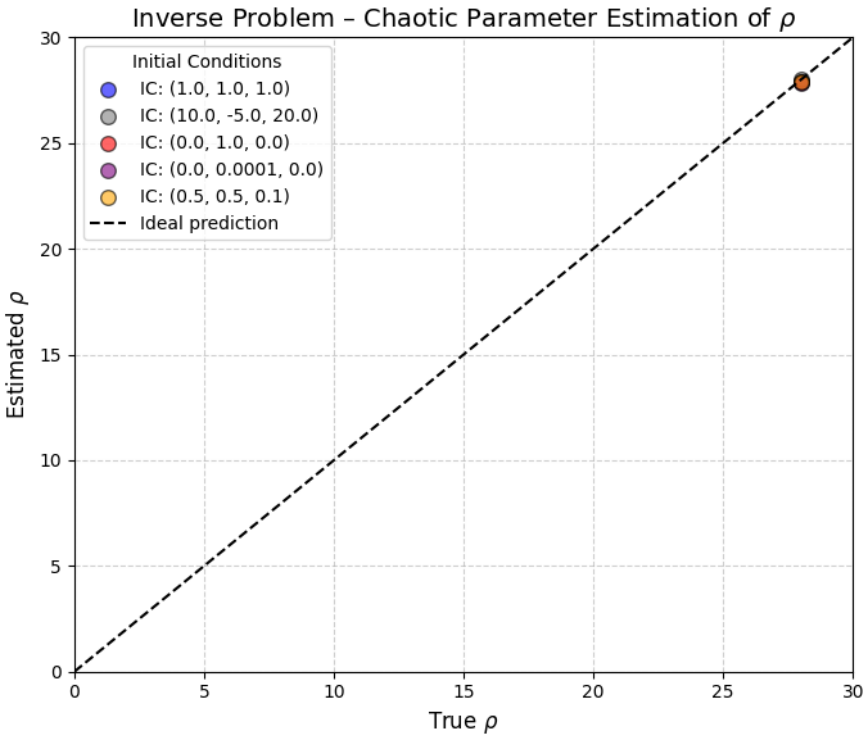

Abbildung 19: Vergleich Parameterbestimmung chaotisches Verhalten für verschiedene Anfangswerte

In Abbildung 19 werden die geschätzten r-Werte für alle untersuchten Anfangsbedingungen gegenüber dem wahren Wert für $r = 28$ aufgetragen. Alle Punkte überlappen und liegen nahezu exakt auf der Idealgeraden, was darauf hinweist, dass das Netzwerk in der Lage ist, die zugrunde liegenden Parameter unabhängig von der gewählten Anfangsbedingung im chaotischen Bereich präzise zu rekonstruieren. Diese Ergebnisse zeigen, dass die vorgenommenen Optimierungen signifikant zur Verbesserung der Modellgüte beigetragen haben und die PINN-Architektur in der Lage ist, inverse Probleme auch im komplexeren chaotischen Bereich erfolgreich zu lösen.

6 Fazit und Ausblick

Ziel dieser Arbeit war es, die Fähigkeiten von PINNs zur Lösung inverser Probleme für Systeme mit chaotischem Verhalten zu untersuchen. PINNs sind dazu in der Lage physikalisches Wissen explizit in den Lernprozess eines neuronalen Netzes zu integrieren. Darauf aufbauend, wurden im Rahmen der Untersuchungen unterschiedliche PINN-Modelle entwickelt, die sowohl das Simulationsproblem als auch das inverse Problem eines dynamischen, chaotischen Systems lösen können. Als primäres Untersuchungsobjekte diente das Lorenz-System, welches als klassisches Beispiel eines chaotischen Systems etabliert ist. Dabei wurde zuerst im nicht-chaotischen und periodischen Bereich gezeigt, dass mithilfe von IC-PINNs und datengetriebenen PINNs sowohl das Vorwärts als auch das inverse Problem unter verschiedenen Bedingungen mit einer hohen Genauigkeit gelöst werden kann. Dies diente der Validierung der PINN-Implementierungen, die für die weiteren Untersuchungen, speziell im chaotischen Bereich, herangezogen wurden.

Anschließend wurden die auf Basis der Vorhersage der periodischen Dynamik optimierten Hyperparameter zur Lösung des Systems im chaotischen Parameterbereich eingesetzt. Das so konfigurierte PINN-Modell ist allerdings nicht fähig das chaotische Verhalten des Lorenz-System vorherzusagen. Während des Trainingsprozesses konvergiert die Vorhersage des PINNs stets in einem lokalen Minimum, welches im Vergleich zur Referenzlösung stark vom tatsächlichen zeitlichen Verlauf abweicht. Ein bekanntes Problem von PINNs ist die Bevorzugung von sogenannten Null-Lösungen, bzw. trivialen Lösungen. PINN-Modelle neigen dazu, in einem lokalen Minimum zu verharren, und somit eine Lösung zu präferieren, die einen geringen Verlust bezüglich der Optimierung der Verlustfunktion aufweisen, jedoch die zugrundeliegenden, physikalischen Gegebenheiten nicht korrekt vorhersagen. Deshalb galt es zu untersuchen, wie die Konvergenz in eine Null-Lösung verhindert werden kann, um somit das Training und das Modellergebnis des PINNs zu verbessern. Dafür wurden gezielte Optimierungen der Architektur und des Trainingsprozesses des PINNs durchgeführt. Des Weiteren wurden Methodiken für die Vermeidung von trivialen Lösungen recherchiert, umgesetzt und ihre Auswirkungen dokumentiert. Dabei ist festzuhalten, dass eine dynamische Gewichtung der Verlustterme dazu führt, dass bei der Vorhersage der chaotischen Dynamik der

datenbasierte Verlustterm gegenüber dem physikalischen Verlustterm priorisiert wird. Mit diesem datengetriebenen Ansatz kann die Vorhersage des PINNs im chaotischen Bereich stark verbessert werden.

Anhand der gesammelten Erkenntnisse der Untersuchungen ist es schließlich möglich, das inverse Problem für den chaotischen Bereich des Lorenz-Systems zu lösen. Das auf diese Weise optimierte PINN ist für unterschiedliche Anfangsbedingungen dazu in der Lage, sowohl die Dynamik des chaotischen Verlaufs vorherzusagen als auch den zu bestimmenden Systemparameter r präzise abzuschätzen. Die abschließende Gegenüberstellung aller untersuchten Anfangswerte verdeutlicht, dass die durch das PINN bestimmte Approximationen des Parameters r für alle untersuchten Bedingungen sehr nahe am vorzuhersagenden Wert liegen. Daraus lässt sich ableiten, dass PINNs, unter gezielter Anpassung der Hyperparameter und Architektur, durchaus fähig sind, inverse Probleme selbst in chaotischen Systemen erfolgreich zu lösen.

Zusammenfassend lässt sich festhalten, dass Physics Informed Neural Networks ein vielversprechendes Werkzeug zur Lösung inverser Probleme in nicht-linearen Differentialgleichungssystemen mit chaotischen Verhalten darstellen. Die in dieser Arbeit entwickelten und evaluierten PINN-Varianten zeigen, dass es möglich ist, chaotische Dynamiken über kurze Zeiträume hinweg realitätsnah zu simulieren und zusätzlich Systemparameter zu erlernen bzw. mit hoher Genauigkeit zu rekonstruieren. Im Hinblick auf die eingangs formulierten Forschungsfragen und Ziele ist festzuhalten, dass PINNs es tatsächlich ermöglichen, auf Basis von Beobachtungsdaten die zugrundeliegenden Parameter eines gemessenen Verhaltens zu ermitteln, auch wenn sich diese im einem chaotischen Parameterintervall befinden. Die Ergebnisse belegen, dass eine zuverlässige Parameteridentifikation selbst unter chaotischen Bedingungen gelingt. Demnach ist es möglich, das Systemverhalten basierend auf den geschätzten Parametern entweder als chaotisch oder nicht chaotisch zu klassifizieren. Das Teilziel, nach dem zu untersuchen war, ob PINNs die zeitliche Dynamik eines chaotischen Systems abbilden können, ist somit durch die erfolgreiche Vorhersage der zugrundeliegenden Systemparameter ebenfalls gegeben. Die Ergebnisse legen insgesamt nahe, dass der erfolgreiche Einsatz von PINNs im chaotischen Kontext realisierbar ist, dies jedoch eine sorgfältige Wahl der Architektur des PINNs und des Trainingsprozesses erfordert. Zudem sind die genannten

Hyperparameter empfindlich und auf die konkrete Problemstellung der Lösung des Lorenz-Systems zugeschnitten. Die vorliegende Arbeit liefert somit einen empirischen Beitrag zur Einordnung von PINNs als effektives Werkzeug zur Lösung des inversen Problems für chaotische Systeme. Die Untersuchungen zeigen sowohl das methodische Potenzial als auch bestehende Herausforderungen auf und bilden eine fundierte Grundlage für weiterführende Arbeiten im Bereich des physikbasierten Machine-Learnings.

7 Literaturverzeichnis

Argyris, John, Gunter Faust, Maria Haase, und Rudolf Friedrich. *Die Erforschung des Chaos*. Berlin, Heidelberg: Springer, 2017. https://doi.org/10.1007/978-3-662-54546-1.

Basri, Ronen, David Jacobs, Yoni Kasten, und Shira Kritchman. „The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies". arXiv, 2. Dezember 2019. https://doi.org/10.48550/arXiv.1906.00425.

Baydin, Atilim Gunes, Barak A. Pearlmutter, Alexey Andreyevich Radul, und Jeffrey Mark Siskind. „Automatic differentiation in machine learning: a survey". arXiv, 5. Februar 2018. https://doi.org/10.48550/arXiv.1502.05767.

Beutelspacher, Albrecht, Jörg Schwenk, und Klaus-Dieter Wolfenstetter. *Moderne Verfahren der Kryptographie*. Wiesbaden: Vieweg+Teubner, 2006. https://doi.org/10.1007/978-3-8348-9103-7.

Colliot, Olivier, Hrsg. *Machine Learning for Brain Disorders*. Neuromethods 197. New York: Springer Nature, 2023.

Gleick, James. *Chaos: Making a New Science*. London: Vintage, 1998.

„Gradient Tape". Zugegriffen 12. Juli 2025. https://www.tensorflow.org/api_docs/python/tf/GradientTape.

Hoffmann, Dirk W. *Einführung in die Informations- und Codierungstheorie*. Berlin, Heidelberg: Springer, 2023. https://doi.org/10.1007/978-3-662-68524-2.

Hope Tom, Yehezkel S. Resheff und Itay Lieder. *Einführung in TensorFlow*. O'Reilly, 2018.

Karrenberg, Ulrich. *Universalschlüssel Fourier-Transformation: Innovative Anwendungen in Wissenschaft und Technik*. Wiesbaden: Springer Fachmedien, 2025. https://doi.org/10.1007/978-3-658-47439-3.

Kingma, Diederik P., und Jimmy Ba. „Adam: A Method for Stochastic Optimization". arXiv, 30. Januar 2017. https://doi.org/10.48550/arXiv.1412.6980.

Kollmannsberger, Stefan, Davide D´Angella, Moritz Jokeit, und Leon Herrmann. *Deep Learning in Computational Mechanics: An Introductory Course*. Studies in Computational Intelligence, volume 977. Cham: Springer, 2021.

Krishnapriyan, Aditi S., Amir Gholami, Shandian Zhe, Robert M. Kirby, und Michael W. Mahoney. „Characterizing possible failure modes in physics-informed neural networks“. arXiv, 11. November 2021. https://doi.org/10.48550/arXiv.2109.01050.

Leiteritz, Raphael, und Dirk Pflüger. „How to Avoid Trivial Solutions in Physics-Informed Neural Networks“. arXiv, 10. Dezember 2021. https://doi.org/10.48550/arXiv.2112.05620.

Mathews, John H., und Kurtis D. Fink. *Numerical methods using MATLAB*. 4th ed. Upper Saddle River, N.J: Pearson, 2004.

Meng, Chuizheng, Sam Griesemer, Defu Cao, Sungyong Seo, und Yan Liu. „When Physics Meets Machine Learning: A Survey of Physics-Informed Machine Learning“. *Machine Learning for Computational Science and Engineering* 1, Nr. 1 (7. Mai 2025): 20. https://doi.org/10.1007/s44379-025-00016-0.

Mitschke, Norbert. *Konvolutionäre neuronale Netze in der industriellen Bildverarbeitung und Robotik*. Forschungsberichte aus der Industriellen Informationstechnik. Erscheinungsort nicht ermittelbar: KIT Scientific Publishing, 2022.

Moseley, B. „Physics-Informed Machine Learning: From Concepts to Real-World Applications“. Http://purl.org/dc/dcmitype/Text, University of Oxford, 2022. https://ora.ox.ac.uk/objects/uuid:b790477c-771f-4926-99c6-d2f9d248cb23.

Nelli, Fabio. *Python Data Analytics: With Pandas, NumPy, and Matplotlib*. Third edition. New York: Apress, 2023.

Neuer, Marcus J. *Maschinelles Lernen für die Ingenieurwissenschaften: Einführung in physikalisch informierte, erklärbare Lernverfahren für KI in technischen Anwendungen*. Berlin, Heidelberg: Springer, 2024. https://doi.org/10.1007/978-3-662-68216-6.

Nielsen, Michael A. „Neural Networks and Deep Learning“, 2015. http://neuralnetworksanddeeplearning.com.

Papula, Lothar. *Mathematik für Ingenieure und Naturwissenschaftler Band 2: Ein Lehr- und Arbeitsbuch für das Grundstudium*. Wiesbaden: Springer Fachmedien, 2015. https://doi.org/10.1007/978-3-658-07790-7.

Rahaman, Nasim, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, und Aaron Courville. „On the Spectral Bias of Neural Networks“. In *Proceedings of the 36th International Conference on Machine Learning*, 5301–10. PMLR, 2019. https://proceedings.mlr.press/v97/rahaman19a.html.

Raissi, M., P. Perdikaris, und G. E. Karniadakis. „Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations“. *Journal of Computational Physics* 378 (1. Februar 2019): 686–707. https://doi.org/10.1016/j.jcp.2018.10.045.

Rosenblatt, F. „The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.“ *Psychological Review* 65, Nr. 6 (1958): 386–408. https://doi.org/10.1037/h0042519.

Röttger Nils, Verena Dietrich und Gerhard Runze. *Basiswissen KI-Testen*, 2024.

Sonnet, Daniel. *Neuronale Netze kompakt: Vom Perceptron zum Deep Learning*. 1. Aufl. 2022. IT kompakt. Wiesbaden: Springer Fachmedien Wiesbaden, 2022. https://doi.org/10.1007/978-3-658-29081-8.

Steger, Sophie. „Physics informed neural networks : analysis for a dynamical system - the double pendulum“. TU Graz, 2022.

Steger, Sophie, Franz M Rohrhofer, und Bernhard C Geiger. „How PINNs Cheat: Predicting Chaotic Motion of a Double Pendulum“, o. J.

Steyer, Ralph. *Programmierung in Python: Ein kompakter Einstieg für die Praxis*. Wiesbaden: Springer Fachmedien, 2024. https://doi.org/10.1007/978-3-658-44286-6.

Strogatz, Steven H. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Boca Raton London New York: Westview Press, 2014.

Wang, Sifan, Shyam Sankaran, und Paris Perdikaris. „Respecting Causality Is All You Need for Training Physics-Informed Neural Networks“. arXiv, 14. März 2022. https://doi.org/10.48550/arXiv.2203.07404.

Weitz, Edmund. *Konkrete Mathematik (nicht nur) für Informatiker: Mit vielen Grafiken und Algorithmen in Python*. Berlin, Heidelberg: Springer, 2021. https://doi.org/10.1007/978-3-662-62618-4.

Wu, Yuandi, Brett Sicard, und Stephen Andrew Gadsden. „Physics-informed machine learning: A comprehensive review on applications in anomaly detection and condition monitoring“. *Expert Systems with Applications* 255 (1. Dezember 2024): 124678. https://doi.org/10.1016/j.eswa.2024.124678.

Zacher, Serge, und Manfred Reuter. *Regelungstechnik für Ingenieure*. Wiesbaden: Vieweg+Teubner, 2011. https://doi.org/10.1007/978-3-8348-9837-1.