GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Fakultät für
Physik

Bericht

Spezialisierungspraktikum

angefertigt von

Roland Simon Zimmermann

aus Recklinghausen

am Max-Planck-Institut für Dynamik und Selbstorganisation

Bearbeitungszeit: 1. April 2009 bis 15. Juli 2009

Erstgutachter/in: Prof. Dr. Ulrich Parlitz

Zusammenfassung

Hier werden auf einer halben Seite die Kernaussagen der Arbeit zusammengefasst.

Stichwörter: Physik, Bachelorarbeit

Abstract

Here the key results of the thesis can be presented in about half a page.

Keywords: Physics, Bachelor thesis

Inhaltsverzeichnis

Nomenklatur

Symbole

Symbol	Bedeutung
$\rho(\mathbf{W}))$	Spektralradius von $\mathbf{W}$
$\sigma(\mathbf{W})$	Singulärwert von $\mathbf{W}$
α	Verlustrate des *Leaky Integrators*
N	Anzahl der Einheiten im Reservoir $\mathbf{W}$
$[\cdot\,;\,\cdot]$	Vertikales Aneinanderfügen von Vektoren/Matrizen
ν_{max}	Maximalwert des hinzugefügten Rauschens $\nu(n)$ für die Erhöhung der Stabilität.

1 Einführung

Dieser Bericht gibt eine Übersicht über die im Spezialisierungspraktikum gewonnen Erkenntnisse. In diesem wurde aus dem Bereich des *Machine Learnings* die Technik des *Reservoir Computings* an Hand der Echo State Networks untersucht. Hierbei wurden zuerst die theoretischen Grundlagen betrachtet und die gewonnenen Informationen anschließend auf zwei Anwendungsbeispiele bezogen. Die Wahl der Echo State Networks wurde dadurch bedingt, dass diese eine Vielseite Umsetzung rekurrenter Neuronaler Netzwerke darstellen, wie im Folgenden dargestellt wird.
Alle Programmierbeispiele während des Praktikums wurden in Python mittels numpy und scipy verfasst worden. Die relevanten Auszüge hiervon sind im Anhang zu finden.

2 Überblick über Neural Networks

In den letzten Jahren hat die Technik der Neuronalen Netzwerke erneut stark an Popularität gewonnen. Dies liegt zum einen an der gestiegenen verfügbaren Rechenleistung und zum anderen an der Entwicklung hierfür notwendiger Algorithmen.
Allgemein lassen sich diese Netze in zwei große Gruppen aufteilen: die der Feed Forward Neural Networks und die der Recurrent Neural Networks, welche im Folgenden als FFNN respektive RNN bezeichnet werden. Ein FFNN besteht aus mehreren Ebenen, welche jeweils aus verschiedenen nicht-linearen Einheiten zusammengesetzt sind. Hierbei wird die erste für die Eingabe und die letzte Ebene zur Ausgabe eines Signals genutzt. Die Einheiten zweier benachbarter Ebenen sind mit individuellen Gewichten vollständig in Richtung der Ausgabe verbunden. Dies bedeutet, dass jedes Einheit x_i^n sein Signal an alle Einheiten der folgenden Ebene x_j^{n+1} mit einem individuellen Gewicht $w_{i\to j}^n$ weitergibt. Zwischen den Einheiten innerhalb einer Ebene bestehen keinerlei Verbindungen. Damit ein solches Netzwerk Vorhersagen treffen kann, muss es trainiert werden. Dies wurde durch die Entwicklung des Backpropagation-Algorithmus stark vereinfacht.

RNN haben einen ähnlichen Aufbau, doch können alle Einheiten an alle anderen Einheiten Signale weitergeben und von diesen erhalten. Dies kann die Vorhersage in bestimmten Anwendungsbeispielen wie der Text und Sprachanalyse verbessern. Ein Nachteil ist, dass zum Trainieren nicht mehr der einfachere Backpropagation-Algorithmus Algorithmus genutzt werden kann, sondern eine für RNNs abgewandelte Form genutzt werden muss. Hierfür werden die verschiedenen Zustände die das RNN im Laufe der Signal-Propagation annimmt nacheinander betrachtet und auf diese zeitliche Entwicklung anschließend der Backpropagation-Algorithmus angewendet. Diese Methode ist unter dem Namen Backpropagation through Time (BTT) bekannt. Sie ist rechenaufwendiger und dadurch, dass das Verschwindenden des Gradienten wahrscheinlicher ist, instabiler.

3 Echo State Networks

3.1 Überblick

Um die (Leistungs)Probleme der RNN zu umgehen wurden als mögliche Lösung die Echo State Networks von H. Jäger vorgeschlagen [**?**]. TODO: Etwas über die Entwicklung und das Entstehen der ESNs schreiben.

3.1.1 Aufbau

Ein ESN ist eine Spezialform eines RNNs. Hierbei wird eine auf dem ersten Blick eigenartige Entscheidung getroffen: Während des gesamten Trainingsvorganges werden die Verbindungen der einzelnen Einheiten größtenteils nicht verändert. Es wird versucht durch das *Echo* der vorherigen Signale, welche noch im Netzwerk gespeichert sind, diese Signale zu rekonstruieren - hier raus ergibt sich auch der Name. Im Folgenden wird der Aufbau und anschließend die Funktionsweise eines solchen Netzwerkes beschrieben [4].

Allgemein bildet das Netzwerk S ein zeitliches Signal $\vec{u}(n) \in \mathbb{R}^{N_u}$ auf eine zeitlich variable Ausgabe $\vec{y}(n) \in \mathbb{R}^{N_y}$ für die Zeiten $n = 1, ..., T$ ab. Zudem besitzt das System ein sogenanntes Reservoir aus N nicht-linearen Einheiten. Der innere Zustand des Netzwerkes wird durch diese Einheiten beschrieben und als $s(n) \in \mathbb{R}^N$ bezeichnet.
Die Verbindungen der inneren Einheiten untereinander wird durch die Gewichtsmatrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ beschrieben. Das Eingangssignal wird zusammen mit einem *Bias* $b_{in} \in \mathbb{R}$ durch die Matrix $\mathbf{W}_{in} \in \mathbb{R}^{N \times (N_u+1)}$ auf die inneren Einheiten weitergeleitet. Die zeitliche Entwicklung der inneren Zustände berechnet sich nach der Vorschrift

$$\vec{s}(n) = (1 - \alpha) \cdot \vec{x}(n-1)\alpha \cdot f_{in}\left(\mathbf{W}_{in}[b_{in}; \vec{u}(n)] + \mathbf{W}\vec{x}(n-1)\right), \tag{3.1}$$

wobei f_{in} eine beliebige (meistens *sigmoid*-förmige) Transferfunktion ist, und $[\cdot\,;\,\cdot]$

das vertikale Aneinanderfügen von Vektoren beziehungsweise Matrizen bezeichnet. Für diese Zustandsgleichung wurde das Modell eines *Leaky Integrator Neurons* genutzt, wobei $\alpha \in (0, 1]$ die Verlustrate beschreibt. Für $\alpha = 1$ ergibt als Spezialfall ein gewöhnliches Neuron.

Da für manche Anwendungsfälle auch eine direkte Rückkopplung wünschenswert sein kann, kann man das System noch um eine Ausgabe-Rückkopplung erweitert werden. Diese verbindet die Ausgabe erneut mit den inneren Einheiten durch die Matrix $\mathbf{W}_{fb} \in \mathbb{R}^{N \times N_y}$. Somit ergibt sich

$$\vec{s}(n) = (1 - \alpha) \cdot \vec{x}(n-1)\alpha \cdot f_{in}\left(\mathbf{W}_{in}[b_{in}; \vec{u}(n)] + \mathbf{W}\vec{x}(n-1) + \mathbf{W}_{fb}\vec{y}(n)\right) \tag{3.2}$$

als Zustandsgleichung.

An Hand der inneren Zustände lassen sich nun noch die sogenannten erweiterten inneren Zustände $x(n) = [b_{out}; \vec{s}(n); \vec{u}(n)] \in \mathbb{R}^{1+N_u+N}$ definieren, wobei b_{out} ein Bias für die Ausgabe darstellt.

Aus diesen erweiterten inneren Zuständen kann nun die Ausgabe $\vec{y}(n)$ konstruiert werden. Dies kann entweder im Sinne einer Linearkombination durch die Ausgangsmatrix $\mathbf{W}_{out} \in \mathbb{R}^{(1+N_u+N)\times N_y}$ oder durch andere nicht lineare Klassifizierer wie beispielsweise einer Support Vector Machine (SVM) durchgeführt werden. Hier wird nur der Fall einer Linearkombination betrachtet, da sich für die anderen Methoden ein analoges Verfahren ergibt. Hierdurch erhält man die Ausgabe

$$\vec{y}(n) = f_{out}\left(\mathbf{W}_{out}\vec{x}(n) = \mathbf{W}_{out}[b_{out}; \vec{s}(n); \vec{u}(n)]\right), \tag{3.3}$$

wobei f_{out} nun die Transferfunktion der Ausgabe steht.

Während die Matrix $\mathbf{W}_{out}$ durch den Trainingsvorgang bestimmt wird, werden die Matrizen $\mathbf{W}_{in}$ und $\mathbf{W}$ a priori generiert und festgelegt. Hierbei hat sich für das Generieren der Eingangsmatrix eine Zufallswahl zwischen -1 und 1 als geschickt herausgestellt. Falls ein Feedback gewünscht ist, also Gleichung ??? genutzt wird, wird $\mathbf{W}_{fb}$ gleichartig konstruiert. Auf das Generieren der inneren Matrix $\mathbf{W}$ wird später genauer eingegangen.

3.1.2 Training

Nachdem der Aufbau des Netzwerkes definiert ist, ergibt sich nun die Frage, wie man den Trainingsvorgang durchführt.
Hierfür wird für die Zeiten $n = 0, ..., T_0$ das ESN mit dem Signal $\vec{u}(n)$ betrieben, wobei T_0 die *transiente Zeit* beschreibt. Anschließend wird das System für Zeiten $n < T$ weiter betrieben und die erweiterten Zustände $\vec{x}(n)$ als Spalten in der *Zustandsmatrix* $\mathbf{X} \in \mathbb{R}^{(1+N_u+N)\times T}$ gesammelt. Analog dazu werden die gewünschten Ausgaben $\vec{y}(n)$ nach dem Anwenden der Inversen f_{out}^{-1} der Ausgabe-Transferfunktion f_{out} auch als Spalten in der *Ausgabematrix* $Y \in \mathbb{R}^{N_y \times T}$ gesammelt.
Nun wird eine Lösung der Gleichung

$$\mathbf{Y} = \mathbf{W}_{out}\mathbf{X} \tag{3.4}$$

für $\mathbf{W}_{out}$ gesucht. Hierfür stehen mehrere Verfahren zur Verfügung, von denen zwei prominente erwähnt sein sollen. Zum einen kann die Lösung durch eine *Tikhonov Regularisierung* mit der Regularisierung $\beta \cdot ||W_i||^2$ mit der Konstante β erhalten werden. Das Verfahren

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}^T \left(\mathbf{X}\mathbf{X}T + \beta I\right)^{-1} \tag{3.5}$$

ist sehr leistungsstark, aber auch teilweise numerisch instabil. Das Verfahren kann allerdings bei geeigneter Wahl von β die besten Ergebnisse erzielen.

Zum anderen kann zur Lösung die *Moore-Penrose-Pseudoinverse* $\mathbf{X}'$ genutzt werden, sodass

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}' \tag{3.6}$$

folgt. Dieses Verfahren ist zwar sehr rechenaufwendig aber dafür numerisch stabil. Nichts desto trotz, kann allerdings auf Grund des Fehlens einer Regularisierung leicht der Effekt des Overfittings auftreten. Dieser kann umgangen werden, wenn in der Zustandsgleichung ??? eine leichte normalverteilte Störung $\vec{\nu}(n)$ der Größenordnung 0.01 bis 0.0001 addiert wird. Dieser Ansatz beruht auf Empirie, da eine mathematische Begründung hierfür noch nicht vollständig gelungen ist.

Nachdem das Training beschrieben wurde, ergibt sich somit der Folgende Funkti-

onsablauf für das Betreiben eines ESN:

1. Zufälliges Generieren der Matrizen $\mathbf{W}_{in}, \mathbf{W}_{fb}$ und Konstruktion der Matrix $\mathbf{W}$

2. Einspeisen des Signals $u(n)$ und Konstruktion der Zustandsmatrix $\mathbf{X}$ und der Ausgabematrix $\mathbf{Y}$

3. Berechnung der Ausgabematrix $\mathbf{W}_{out}$

4. Einspeisen des Signals $u(n)$ für Vorhersagen des Signales $y(n)$ für $n > T$

3.2 Theorie

Um die mathematischen Eigenschaften beschreiben zu können, sind zuerst zwei Definitionen nötig [6].

Definition 1 (Kompatibler Zustand)**.** *Sei* $S : X \times U \to X$ *ein* ESN *mit der Zustandsgleichung* $\vec{x}_{n+1} = F(\vec{x}_n, \vec{u}_{n+1})$*. Eine Folge von Zuständen* $(\vec{x}_n)_n$ *ist kompatibel mit der Eingangsfolge* $(\vec{u}_n)_n$*, wenn* $\vec{x}_{n+1} = F(\vec{x}_n, \vec{u}_{n+1}), \forall n \leq 0$ *erfüllt ist.*

Definition 2 (Echo State Eigenschaft (ESP))**.** *Ein* ESN $S : X \times U \to X$ *besitzt die Echo State Eigenschaft genau dann wenn eine Nullfolge* $(\delta_n)_{n\geq 0}$ *existiert, sodass für alle Zustandsfolgen* $(\vec{x}_n)_n, (\vec{x'}_n)_n$ *die kompatibel mit der Eingangsfolge* $(\vec{u}_n)_n$ *sind gilt, dass* $\forall n \geq 0 ||x_n - x'_n|| < \delta_n$

Dies bedeutet, dass nachdem das Netzwerk lang genug betrieben worden ist, der Zustand nicht mehr von dem beliebig gewähltem Anfangszustand abhängt. Diese Eigenschaft ist für notwendig, damit das ESN Vorhersagen treffen kann [2].

Nun stellt sich die Frage, wann ein Netzwerk diese Eigenschaft besitzt. Es wird schnell klar, dass dies nur durch die Gewichtsmatrix $\mathbf{W}$ bestimmt wird. Betrachtet man die Zustandsgleichung des Netzwerkes, so lässt sich auf Grund des *Banachschen Fixpunktsatzes* erkennen, dass die *ESP* vorliegt, sobald $||\vec{x}_{n+1} - \vec{x'}_{n+1}|| < ||\vec{x}_n - \vec{x'}_n||$ für zwei kompatible Zustände $\vec{x}_n \neq \vec{x'}_n$ erfüllt ist [1]. Hier raus ergibt sich, dass die *ESP* vorliegt, wenn

$$|1 - \alpha(1 - \sigma(\mathbf{W}))| < 1 \tag{3.7}$$

erfüllt ist, wobei $\sigma(mathbfW)_{max}$ der größte Singulärwert ist [3].
Weitergehend ist bekannt, dass für Systeme bei denen der Spektralradius $\rho(\mathbf{W}) > 1$ ist diese Eigenschaft nicht vorliegen kann, sofern $\vec{u}_n = 0$ möglich ist [3] [1].

Hier raus folge lange Zeit die falsche Annahme, dass für Systeme mit $\rho(\mathbf{W}) < 1$ die Eigenschaft garantiert ist. Wie allerdings gezeigt werden konnte, ist dies nicht der Fall. Stattdessen konnte gezeigt werden, dass eine hinreichende Bedingung durch

$$\rho(\alpha|\mathbf{W}| + (1 - \alpha)\mathbf{I}) < 1 \tag{3.8}$$

gegeben ist [6] - wobei als Betrag der Matrix hier das elementweise Betragsnehmen gemeint ist. Dies ist äquivalent dazu, dass $\rho(|\mathbf{W}|) < 1$ gilt. Diese Bedingung ist weniger einschränkend als 3.7 [6].
Darauf basierend kann nun eine Methode angegeben werden, um die Gewichtsmatrix $\mathbf{W}$ zu konstruieren:

1. Generiere zufällige Matrix $\mathbf{W}$ mit $|\mathbf{W}| = \mathbf{W}$
2. Skaliere $\mathbf{W}$, sodass 3.8 erfüllt ist.
3. Wechsel zufällig das Vorzeichen von ungefähr der Hälfte aller Einträge.

Statt dieser Vorschrift wurde zuvor oftmals $\mathbf{W}$ zufälliger generiert und anschließend nur $\rho(\mathbf{W})$ skaliert, was mit unter zu instabilen Systemen geführt hat.

4 Anwendungen

Um den Aufwand für die spätere Benutzung zu reduzieren ist zuerst eine allgemeine Implementation des ESNs geschrieben worden.

4.1 Vorhersage eines Rössler-Systems

Eine erste Anwendung um die Möglichkeiten eines ESN zu demonstrieren besteht in der Vorhersage des Verhaltens eines *Rössler-Systems*

$$\begin{aligned} \dot{x} &= -(y + z) \\ \dot{y} &= x + 0.25 \cdot y \\ \dot{z} &= 0.4 + (x - 8.5) \cdot z. \end{aligned} \tag{4.1}$$

Hierbei werden zwei verschiedene Aufgaben betrachtet: (a) die Vorhersage von $x(n+50)$ durch die Kenntnis von $x(n)$ und (b) die Kreuz-Vorhersage $y(n)$ durch $x(n)$. Für beide Aufgaben wurde das System für jeweils 10000 Schritte mit einer Abtastrate $\Delta t = 0.05$ simuliert, wobei die Hälfte der Daten für das Training und der Rest für die Testergebnisse verwendet worden ist.

4.1.1 Vorhersage $x(n) \rightarrow x(n + 48)$

Hierfür ist durch eine GridSearch ein geeignetes System gefunden werden, Parameter in der Tabelle 4.1 dargestellt sind. Als Maß für die Optimierung wurde der *Mean Square Error (MSE)* benutzt.

Variable	Wert
N	300
α	0.10
$\rho(\|\mathbf{W}\|)$	0.035
ν_{max}	0.01

Tab. 4.1: Auflistung der für (a) benutzten Parameter des ESN.

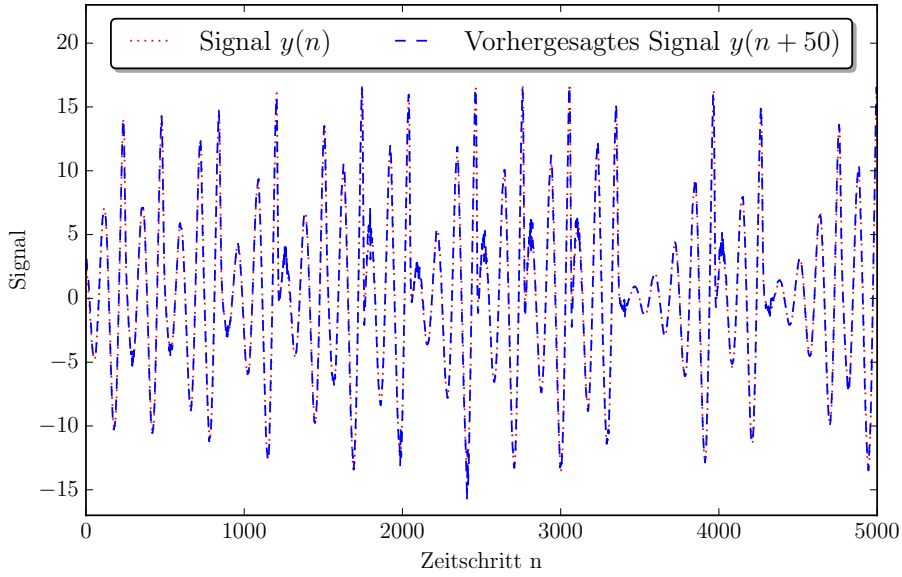

Abb. 4.1: Graphische Darstellung des Signals $y(n)$ und der Vorhersage $y(n+50)$.

Hierbei wurde für den Trainingsvorgang ein Fehler $MSE = 0.575$ und für den Testvorgang von $MSE = 0.270$ bestimmt. Die grafische Darstellung der Vorhersage $v(n)$ und des ermittelten Fehlers sind in Abbildung 4.1 zu finden.

4.1.2 Kreuz-Vorhersage $x(n) \rightarrow y(n)$

Auch hierfür konnte sehr schnell durch eine GridSearch ein geeignetes System gefunden werden. Als Maß wurde ebenfalls der *MSE* benutzt. Die gefundenen Parameter sind in der Tabelle 4.2 dargestellt.

Variable	Wert
N	500
α	0.20
$\rho(\lvert\mathbf{W}\rvert)$	0.01
ν_{max}	0.01

Tab. 4.2: Auflistung der für (b) benutzten Parameter des ESN.

Für das Auslesen wurde eine Linearkombination genutzt, für die die Lösung $\mathbf{W}_{out}$ mittels der Pseudoinversen bestimmt werden konnte.

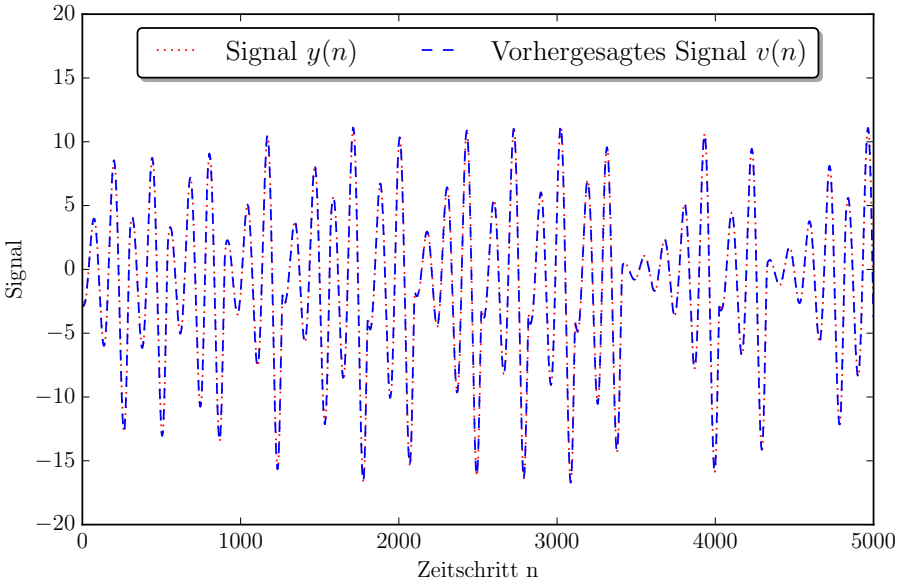

Abb. 4.2: Graphische Darstellung des Signals $y(n)$ und der Vorhersage $v(n)$.

Hierbei wurde für den Trainingsvorgang ein Fehler $MSE_{train} = 0.011$ und für den Testvorgang von $MSE_{test} = 0.091$ bestimmt. Die grafische Darstellung der Vorhersage $v(n)$ und des ermittelten Fehlers sind in Abbildungen 4.2, 4.3 zu finden. Im Vergleich zu anderen Publikationen [5] ergibt sich ein ähnlicher Fehler. Leider kann hierbei nur mit den Graphen verglichen werden, da keine Fehler angegeben sind.

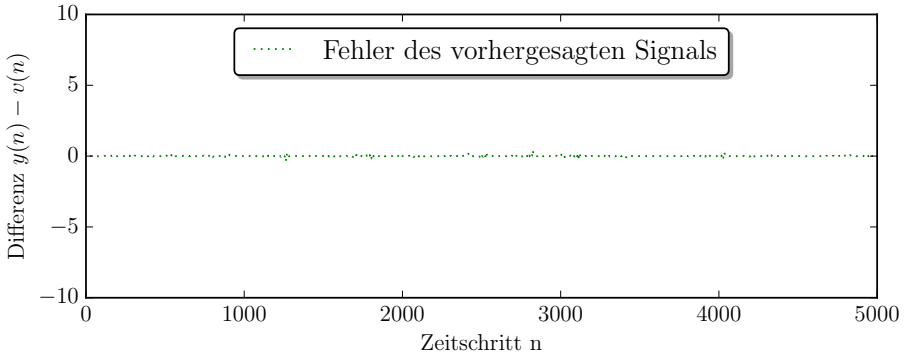

Abb. 4.3: Graphische Darstellung des des Fehlers $y(n) - v(n)$.

4.2 Vorhersage eines Mackey-Glass-Systems

Als nächste Anwendung wird ein *Mackey-Glass-System*

$$\dot{x} = \beta \frac{x_\tau}{1 + x_\tau^n} - \gamma x \tag{4.2}$$

betrachtet, wobei $\tau = 17, n = 10, \beta = 0.2$ gewählt wurden. Dies ist eine beliebte Aufgabe im Bereich der Vorhersagen chaotischer Zeitreihen.

Variable	Wert
N	1000
α	0.20
$\rho(\mathbf{W})$	0.35
ν_{max}	0.001

Tab. 4.3: Auflistung der benutzten Parameter des ESN.

Für dieses System wurde nun die Vorhersage $x(n) \to x(n + 48)$ betrachtet, wobei die Parameter aus Tabelle 4.3 genutzt wurden. Eine Darstellung der Ergebnisse ist in Abbildung 4.4 zu sehen. Hierbei wurde der *MSE* für die gesamte Trainingsphase und die ersten 1000 Zeitpunkte der Testphase zu $MSE_{train} = 0.0046$ und $MSE_{test} = 0.0002$.

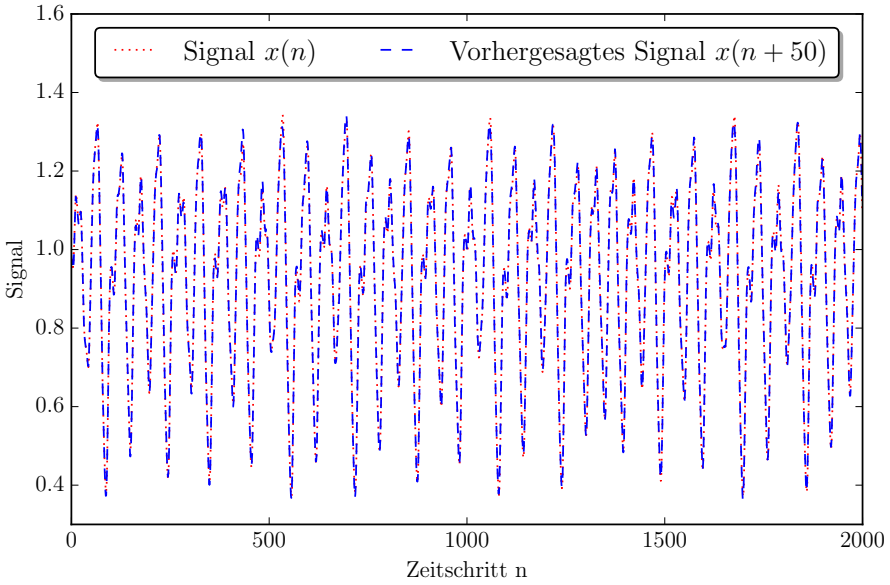

Abb. 4.4: Graphische Darstellung des des Fehlers $y(n) - v(n)$.

5 Fazit

6 Ausblick

Für das Schreiben des Berichtes für dieses Spezialisierungspraktikums hat sich die Wahl der LaTeX-Umgebung als gut geeignet herausgestellt. Die optisch Darstellung von mathematischen Ausdrücken sowie des erklärenden Textes ist hierbei intuitiver umzusetzen als bei den bekannten Alternativen.

Das Programmieren der Anwendungsbeispiele wurde mit der Sprache Python und den bekannten Frameworks Numpy und Scipy umgesetzt. Diese Entscheidung wurde hauptsächlich durch die große Flexibilität dieser Sprache und der ausreichenden Leistung der Bibliotheken beeinflusst. Die hierbei entstandenen Grafiken wurden mit der Pyplot Bibliothek umgesetzt.

Während des Praktikums wurde für die Literaturrecherche der Dienst Google Scholar benutzt. Dies liegt daran, dass er nicht nur eine umfassende Datenbank besitzt, sondern auch, dass das Erhalten der indizierten Werke sehr einfach ist.
Die benutzte Literatur wurde über BibTeX direkt in den LaTeX-Quellcode des Berichts eingebunden und zitiert.

Dieses gesamte Vorgehen hat sich im Laufe des Praktikums als gut erwiesen und wird somit auch in der anschließenden Bachelorarbeit weiterverfolgt werden.

Literaturverzeichnis

[1] H. Jäger. The “echo state“ approach to analysing and training recurrent neural networks - with an erratum note. *GMD Report*, 148, 2001/2010.

[2] H. Jäger. A tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the “echo state network“ approach. *GMD Report*, 159:48 ff., 2002.

[3] H. Jäger, M. Lukoševičiusa, D. Popovici, and U. Siewert. Optimization and applications of echo state networks with leaky- integrator neurons. *Neural Networks*, 20:335–352, 2007.

[4] M. Lukoševičiusa and H. Jäger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[5] U. Parlitz and A. Hornstein. Dynamical prediction of chaotic time series. *Chaos and Complexity Letters*, 1(2):135–144, 2005.

[6] I. Yildiz, H. Jäger, and S. Kiebel. Re-visiting the echo state property. *Neural Networks*, 35:1–9, 2012.