Lineare Interpolationsverfahren für Zellulare Nichtlineare Netzwerke (CNN)

Diplomarbeit

von
Michael Reinisch
Institut für Angewandte Physik
der Johann Wolfgang Goethe-Universität
in Frankfurt am Main

Frankfurt, 11.07.2005

vom Fachbereich Physik der
Johann Wolfgang Goethe-Universität als Diplomarbeit angenommen.

Gutachter 1: Prof. Dr. Ronald Tetzlaff

Bewertung:

Gutachter 2: Prof. Dr.-Ing. Arild Lacroix

Bewertung:

Für meine Eltern, ohne die mein Studium nicht möglich gewesen wäre.

Inhaltsverzeichnis

Einleitung

Komplexe Systeme zeigen ein Verhalten, das sich im allgemeinen nicht durch eine Betrachtung seiner isolierten Komponenten verstehen läßt, denn diese verhalten sich erst im Zusammenhang komplex. Die Merkmale der Koppelung von nichtlinearen Systemen, die komplexes Verhalten und Chaos ermöglichen, lassen sich deshalb besonders gut anhand von nichtlinearen Netzen studieren.
Von besonderer Bedeutung in dieser Hinsicht sind die von L. O. Chua und L. Yang [5] 1988 vorgeschlagenen Zellularen Nichtlinearen Netzwerke, deren dynamisches Verhalten in zahlreichen Arbeiten [8, 1, 28, 4] analysiert wurde. Insbesondere wird anhand schaltungstechnischer Realisierungen dieser Netzwerke deren unter bestimmten Bedingungen vorkommendes komplexes Verhalten erfolgreich zur Lösung von Problemen der Informationsverarbeitung eingesetzt.
Zur Beschreibung des Verhaltens einer Zelle von CNN wird eine nichtlineare Differentialgleichung – eine sogenannte Zustandsgleichung – zugrundegelegt, die Ein- und Ausgangssignale anderer in einer bestimmten Umgebung liegender Zellen erhält. Durch eine derartige Verknüpfung entsteht ein System lokal gekoppelter Differentialgleichungen mit Lösungen, die üblicherweise nach einer nichtlinearen Abbildung die Ausgangssignale eines CNN darstellen. Eine Betrachtung verschiedener Arbeiten [4, 16, 26, 22] zeigt, daß die Zellzustandsgleichung abhängig von der Problemstellung unterschiedliche Formen aufweisen kann. Im Hinblick auf eine Realisierung adaptiver intelligenter Schaltungen ist es daher von großer Bedeutung eine parametrisierte Darstellung einer möglichst großen Klasse von Gewichtsfunktionen oder Ausgangsfunktionen derartiger Netzwerke zu finden. In [29] ist dargelegt, wie mit CNN, die Polynome als Gewichtsfunktionen enthalten, eine Lösung partieller Differentialgleichungen erfolgen kann. Da jedoch in einigen Fällen ein hoher Polynomgrad – der die Berechnung erheblich erschwert – erforderlich ist, wurde in [17] ein Verfahren entwickelt, daß mit Interpolation von tabellierten Funktionen arbeitet. Diese Interpolation wurde durch eine kubische Spline-Interpolation realisiert und im Simulationssystem SCNN [14] implementiert. Obwohl dies eine wesentliche Verbesserung darstellt, ist der algorithmische Aufwand gerade im Hinblick auf eine Implementierung in einer schaltungstechnischen Realisierung noch relativ hoch. Eine erhebliche Reduzierung des entstehenden Berechnungsaufwands wäre durch die Verwendung linearer Interpolationsverfahren zur Darstellung von nichtlinearen Gewichtsfunktionen oder Zellausgangsfunktionen möglich. Allerdings kann es dabei vorkommen, daß große Tabellen benötigt werden um eine vorgegebene Genauigkeit zu erreichen.
In dieser Arbeit soll ein lineares Interpolationsverfahren für CNN mit adaptiver Bestimmung von Stützstellen implementiert werden; das Ziel ist die Minimierung dessen Anzahl bei vorgegebenen Approximationsfehler. Zum einen soll unter der Annahme, daß die jeweilige Gewichts- oder Ausgangsfunktion bekannt sei, ein im Anwendungsfall zu einer genauen Approximation bei minimaler Stützstellenzahl führendes Verfahren entwickelt werden. Zum anderen soll diese Annahme nicht mehr zugrundegelegt werden, d.h. das eine tabellierte Darstellungsform, beispielsweise der Gewichtsfunktion eines CNN, nur anhand von Ausgangssignalen des Netzwerkes in einem überwachten Parametertraining gefunden werden kann. Auch in diesem Fall soll ein Verfahren

bestimmt werden, dessen Anwendung zu einer minimalen Anzahl von Stützstellen führt. Anhand von drei Anwendungen sollen die einzelnen Verfahren untersucht werden. Die erste Anwendung ist die Bestimmung von Lösungen der Korteweg-de Vries (KdV) Gleichung mit einem CNN. Die KdV Gleichung ist eine nichtlineare partielle Differentialgleichung, die die Ausbreitung von Solitonen beschreibt. Desweiteren soll die Burgers Gleichung und Φ^4 Gleichung untersucht werden. Abschließend soll eine Diskussion und Bewertung der gefundenen Resultate erfolgen.

Teil I

Grundlagen

Kapitel 1

Mathematische Grundlagen

Mathematische Modelle bilden eine wichtige Grundlage zur Beschreibung physikalischer Phänomen. Bei Einsatz von Rechnersystemen – die nur mit diskreten Werten und Zeiten rechnen – sind die Methoden der numerischen Mathematik, wie sie in [27] zu finden sind, von großer Bedeutung. Dabei existieren prinzipiell zwei "Fehlerquellen":

- Die numerischen Methoden sind nur Annäherungen an die tatsächlichen Vorgänge.
- Jeder noch so gute Rechner erlaubt nur eine endliche Darstellungsgenauigkeit von Zahlen, und damit entstehen Rundungsfehler.

Die Fehler lassen sich durch geschickte Anwendung der Methoden eingrenzen, so daß eine hinreichende Genauigkeit erreicht wird.

1.1 Interpolation

Das Ziel einer Interpolation ist, eine Funktion $f(x)$, bei der die Funktionswerte an diskreten Stellen x_i bekannt sind, zwischen diesen geeignet zu approximieren. Dies kann zwei unterschiedliche Gründe haben:

- Die Funktion ist wirklich nur an diesen Stellen bekannt (Meßreihe).
- Es sollen nur einige Werte gespeichert werden (tabellarische Funktion) und die Zwischenwerte nur bei Bedarf berechnet werden.

Unter den verschiedenen Methoden, die zur Bewältigung dieser Aufgabe herangezogen werden, sollen hier nur zwei kurz vorgestellt werden.

1.1.1 Lineare Interpolation

Die lineare Interpolation beruht auf der Annäherung einer Funktion durch Geradenstücke. Dazu gibt man eine bestimmte Anzahl *Stützstellen* vor, an denen die Funktionswerte bekannt sind oder (leicht) berechnet werden können. Die Stützstellen können *äquidistant* – meist bei tabellarischen Funktionen, bzw. systematischen Meßreihen der Fall – oder wie in Abbildung 1.1 *beliebig* sein. Zwischen den Stützstellen wird die Geradengleichung

$$G : \frac{y - y_i}{x - x_i} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{1.1}$$

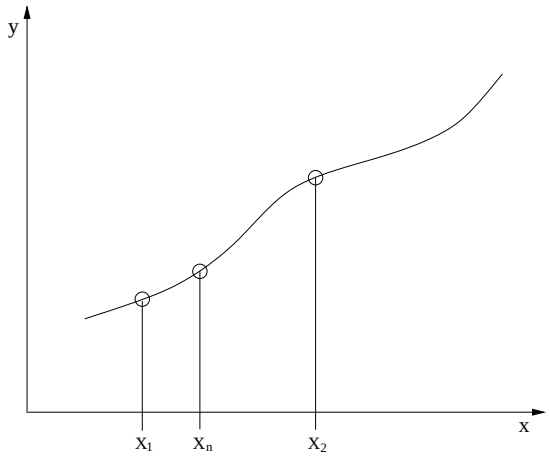

Abbildung 1.1: Funktion mit beliebigen Stützstellen

angesetzt und zur praktischen Berechnung nach

$$y = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i) + y_i \tag{1.2}$$

umgestellt, wobei der Faktor $\frac{y_{i+1}-y_i}{x_{i+1}-x_i}$ auch als Steigung der Geraden bezeichnet wird. Zur einfacheren Umsetzung in Rechner-Programmen wird die Gleichung (1.2) als

$$y = Ay_i + (1 - A)y_{i+1} \tag{1.3}$$

mit $A = \frac{x_{i+1}-x}{x_{i+1}-x_i}$ geschrieben. Um die Unterschiede zwischen linearer Interpolation mit äquidistanten und "beliebigen" Stützstellen zu verdeutlichen sind in Abbildung 1.2 an der Beispielfunktion $f(x) = 10^{-8}(e^{\frac{x}{0.052}} - 1)$ beide Fälle durchgerechnet. Eine Betrachtung zeigt, daß bei gleicher Anzahl die Interpolation mit nicht äquidistanten Stützstellen – bei geeigneter Wahl dieser – viel genauer ist, als die mit äquidistanten Stützstellen. Das Problem ist bei Betrachtung einer bestimmten Funktion die richtigen Stützstellen zu finden, d.h. diejenigen die zu einer maximalen Approximationsgenauigkeit führen. Eine erhebliche Vereinfachung erfolgt gemäß

$$y = \frac{(y_{i+1} - y_i)(x - x_i)}{h} + y_i \tag{1.4}$$

mit $h = x_{i+1} - x_i = x_1 - x_0$, bei äquidistanten Stützstellen. Ein erheblicher Nachteil dabei ist die in der Regel hohe Anzahl von Stützstellen, die benötigt werden, um eine bestimmte Darstellungsgenauigkeit zu erreichen.
Die "Randbehandlung", d.h., was passiert, wenn der gesuchte Wert nicht zwischen zwei Stützstellen liegt, sondern links oder rechts davon, ist ein zusätzliches Problem. Die einfachste – im weiteren auch verwendete – Lösung ist, den Wert mit dem Funktionswert der nächsten Stützstelle (links oder rechts) gleich zu setzen.

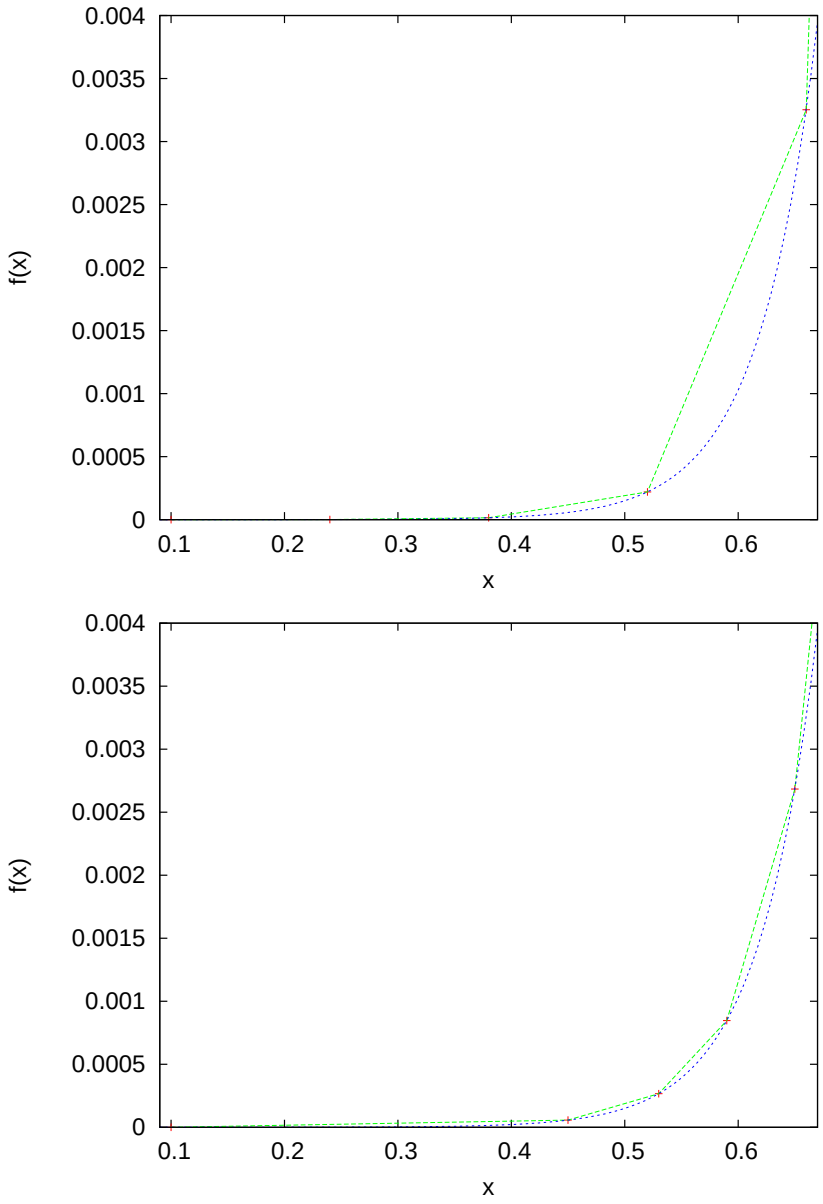

Abbildung 1.2: Lineare Interpolation der Funktion $f(x) = 10^{-8}(e^{\frac{x}{0.052}} - 1)$, oben mit äquidistanten Stützstellen, unten bei beliebiger Wahl der Stützstellen

1.1.2 Spline Interpolation

Bei der Spline Interpolation [27] werden im Gegensatz zur linearen Interpolation keine Geraden, sondern Polynome niedrigen Grades verwendet. Dabei soll die, aus den Polynomstücken zusammengesetzte, Interpolationsfunktion *einfach stetig differenzierbar* sein. Die sogenannte *Spline-Funktion* wird durch Eigenschaften charakterisiert, die sich aus dem Modell der dünnen Latten – im Englischen *Splines* – ergeben. Dazu wird angenommen, daß durch die gegebenen Stützstellen eine dünne, homogene Latte gelegt ist, die in den Stützpunkten gelenkig gelagert ist und dort keinen äußeren Kräften unterliegt. Dann ist die Biegelinie der Latte die gesuchte Spline-Funktion. Nach den Extremalprinzipien wird Deformationsenergie

$$E = \frac{1}{2}\int_{x_0}^{x_N} f''_{spline}(x)^2 dx \tag{1.5}$$

der Latte durch ihre angenommene Form minimiert. Die Spline-Funktion $f_{spline}(x)$ wird durch eine Variationsaufgabe gewonnen und muß daher folgende Nebenbedingungen erfüllen:

- Interpolationseigenschaft $f_{spline}(x_i) = y_i \quad \forall i = 0, 1, \ldots, N$
- $f_{spline}(x)$ stetig differenzierbar an allen inneren Stützstellen $x_i \quad \forall i = 1, 2, \ldots, N-1$
- Zwischen den Stützstellen viermal stetig differenzierbar
- $f_{spline}(x)$ minimiert das Integral $J = \frac{1}{2}\int_{x_0}^{x_N} f''_{spline}(x)^2 dx$

Es müssen drei zusätzliche notwendige Bedingungen erfüllt sein:

1. $f^{(4)}_{spline}(x) = 0 \quad \forall x \neq x_0, x_1, \ldots, x_N$
2. $f''_{spline}(x_i + 0) = f''_{spline}(x_i - 0) \quad \forall i = 1, 2, \ldots, N-1$
3. $f''_{spline}(x_0) = 0$ und $f''_{spline}(x_N) = 0$ *natürliche Randbedingungen* oder $f'_{spline}(x_0) = y'_0$ und $f'_{spline}(x_N) = y'_N$ *vollständige Randbedingungen*

Im folgenden betrachten wir die kubische Spline-Interpolierenden, die durch folgende Formel dargestellt wird:

$$f_{spline_i}(x) = A_i(x - x_i)^3 + B_i(x - x_i)^2 + C_i(x - x_i) + D_i \tag{1.6}$$

Berechnung der kubischen Spline-Interpolierenden

Um die Koeffizienten in (1.6) zu bestimmen, werden die drei oben aufgeführten Bedingungen benutzt. Für das Intervall $[x_i, x_{i+1}]$ wird die Größe $h_i = x_{i+1} - x_i$ eingeführet und die Koeffizienten durch die gegebenen Stützwerte y_i und y_{i+1} und die noch unbekannten zweiten Ableitungen y''_i und y''_{i+1} ausgedrücken; damit folgt

$$\begin{aligned}
A_i &= \frac{1}{6h_i}(y''_{i+1} - y''_i)\,, \\
B_i &= \frac{1}{2}y''_i\,, \\
C_i &= \frac{1}{h_i}(y_{i+1} - y_i) - \frac{1}{6}h_i(y''_{i+1} + 2y''_i) \quad \text{und} \\
D_i &= y_i\,.
\end{aligned}$$

Sind nun auch die zweiten Ableitungen an allen Stützstellen bekannt, dann sind die kubischen Polynome $f_{spline_i}(x)$ eindeutig festgelegt. Durch diese Werte ist die Interpolationseigenschaft, sowie die Stetigkeit der Spline-Funktion und ihrer zweiten Ableitung an den inneren Stützstellen sichergestellt. Es fehlt noch die Stetigkeit der ersten Ableitung. Für die inneren Stützstellen ergibt sich aus dieser Bedingung die Gleichung

$$h_{i-1}y''_{i-1} \quad + \quad 2(h_{i-1}+h_i)y''_i + h_i y''_{i+1} - \frac{6}{h_i}(y_{i+1}-y_i) + \frac{6}{h_{i-1}}(y_i - y_{i-1}) = 0\,. \qquad (1.7)$$

Mit den Randbedingungen wird aus der Gleichung (1.7) ein lineares Gleichungssystem mit $N-1$ Unbekannten, daß eine symmetrische tridiagonale Matrix besitzt, die diagonal dominant ist. Dieses ist eindeutig lösbar und der Rechenaufwand steigt proportional mit N.
In Abbildung 1.3 ist die Spline-Interpolation für die schon bei der linearen Interpolation als Beispiel betrachteten Funktion mit unterschiedlicher Anzahl an Stützstellen durchgerechnet. Man erreicht hier selbst mit äquidistanten Stützstellen eine gute Genauigkeit, auch bei wenig Stützstellen. Der Nachteil ist, daß das Verfahren an sich komplizierter und somit aufwendiger zu implementieren ist.

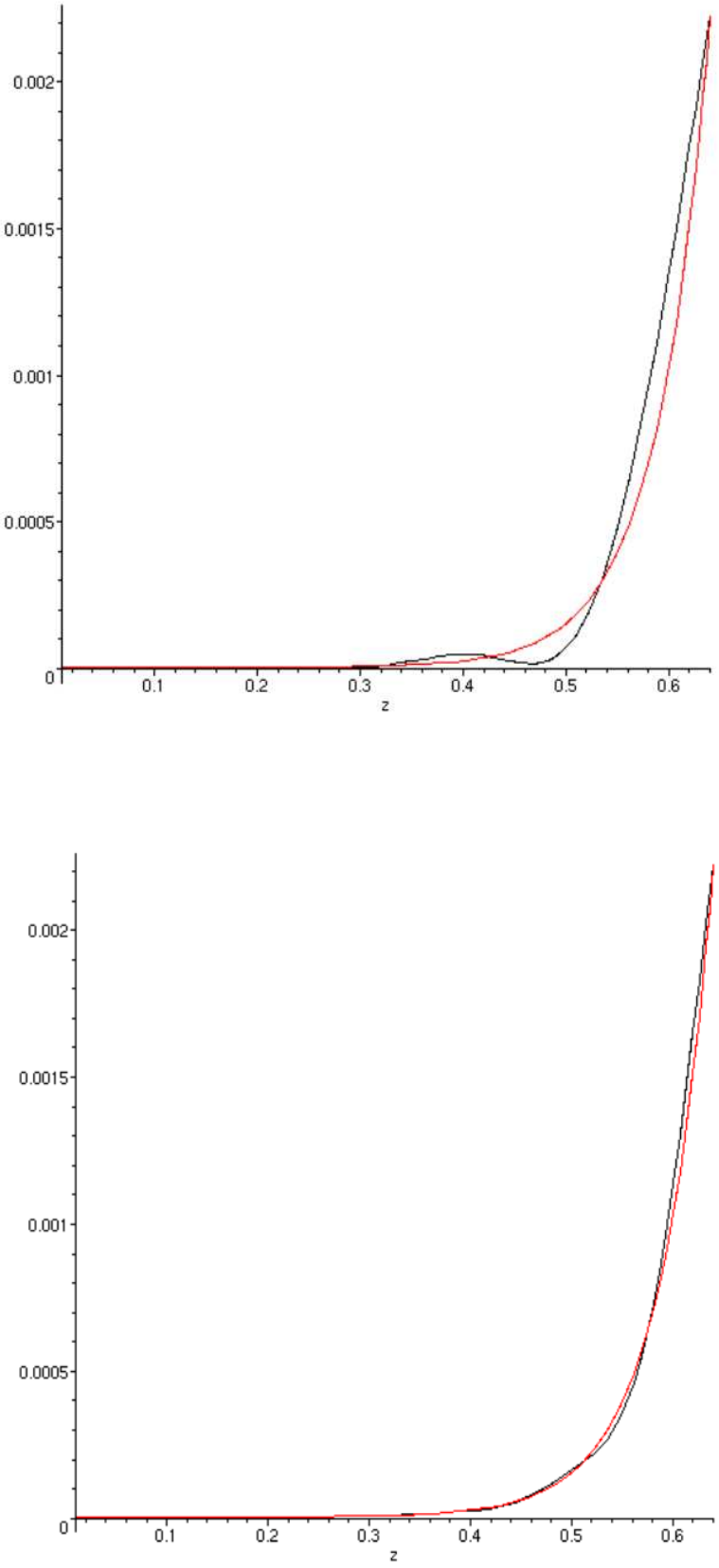

Abbildung 1.3: Spline-Interpolation der Funktion $f(x) = 10^{-8}(e^{\frac{x}{0.052}} - 1)$, oben mit 7, unten mit 11 äquidistanten Stützstellen

1.2 Differentialgleichungen

Viele Probleme in der Physik werden mit Differentialgleichungen beschrieben. Die zeitliche Änderung von Zustandsgrößen wird durch *gewöhnliche Differentialgleichungen*, die nur Ableitungen nach einer Variablen – meist die Zeit – enthalten, dargestellt. Kommen Ableitungen mehrerer Veränderlicher vor, handelt es sich um *partielle Differentialgleichungen*. Bei der im Abschnitt 2.2 beschriebenen CNN Zustandsgleichung handelt es sich um ein System lokal gekoppelter gewöhnlicher Differentialgleichungen. Da bei nichtlinearen Differentialgleichungen häufig analytische Lösungen nicht zugänglich sind, werden numerische Methoden benötigt, die hinreichend genaue Näherungen der Lösungsfunktion liefern. Im Folgenden wird ein Verfahren für gewöhnliche Differentialgleichungen und eine Methode für partielle Differentialgleichungen beschrieben, die später im Zusammenhang mit CNN zur Anwendung kommen.

1.2.1 Methode von Euler

Die *Methode von Euler* ist ein Verfahren zur numerischen Lösung von gewöhnlichen Differentialgleichungen. Eine skalare Differentialgleichung erster Ordnung ist die Gleichung

$$y'(x) \quad = \quad f\left(x, y(x)\right).$$

Unter der Voraussetzung, daß die Anfangsbedingung $y(x_0) = y_0$ bekannt ist, soll die Lösungs-

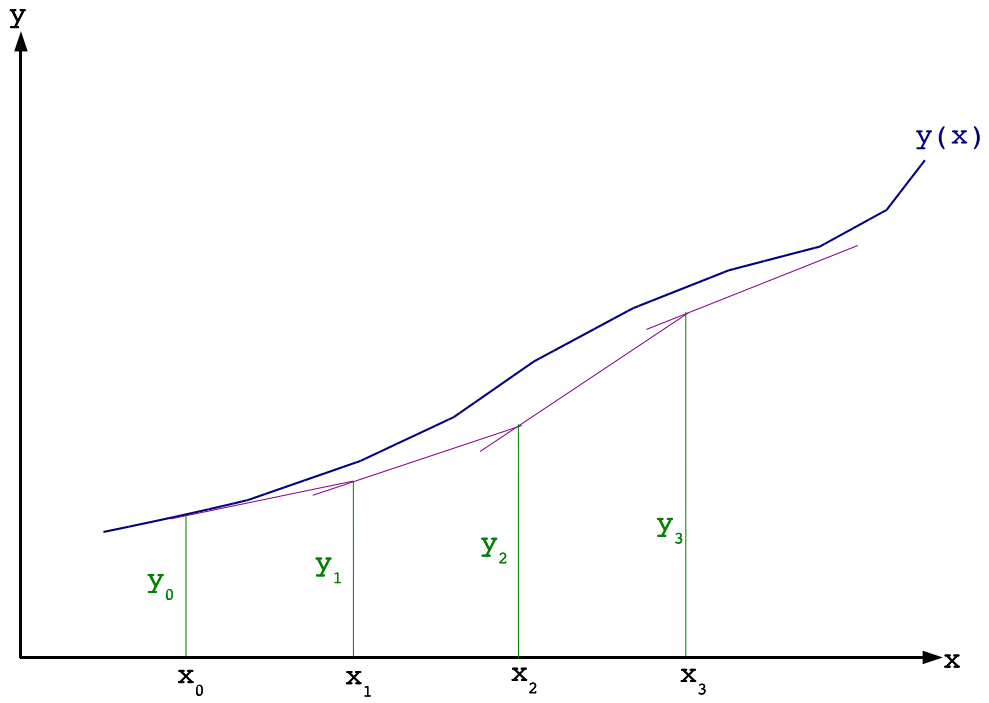

Abbildung 1.4: Methode von Euler [27]

funktion $y(x)$ gefunden werden. Die Grundidee der Methode von Euler ist, da $y'(x_0) = f(x_0, y_0)$ die Steigung der Tangente im Punkt (x_0, y_0) darstellt, die gesuchte Funktion $y(x)$ mit einer Linearisierung durch Tangenten – wie in Abb. 1.4 dargestellt – zu approximieren. Die Geradengleichung (1.2) mit $y'(x_i) = f(x_i, y_i)$ als Steigung läßt sich zur Berechnung des nächsten diskreten Funktionswertes y_{i+1} gemäß

$$\begin{aligned} y_{i+1} &= y'(x_i)(x_{i+1} - x_i) + y_i \\ &\text{mit} \quad h := x_{i+1} - x_i \Rightarrow \\ y_{i+1} &= f(x_i, y_i)h + y_i \end{aligned} \tag{1.8}$$

verwenden. Dabei ist h der Abstand zwischen den diskreten Stützstellen x_i. Die Abweichung der gefundenen Werte y_i zu den exakten Funktionswerten $y(x_i)$ hängt von h ab und wird bei kleineren h besser. Die genaue Wahl von h ist jedoch von der Anwendung abhängig. Da die Berechnung des nächsten Wertes nur von einem vorangegangenen abhängt, spricht man bei der Methode von Euler von einer *Einschrittmethode*.

1.2.2 Differenzenmethode

Die *Methode der finiten Differenzen* wie in [3] – i.A. auch als Differenzenmethode bekannt – ist eine numerische Integrationsmethode für partielle Differentialgleichungen. Im folgenden wird eine partielle Differentialgleichung zweiter Ordnung betrachtet. Das Ziel dieser sogenannten Randwertaufgabe ist eine Funktion $f(x,y)$ zu finden, die die Gleichung

$$A\frac{\partial^2 f(x,y)}{\partial x^2} + 2B\frac{\partial^2 f(x,y)}{\partial x \partial y} + C\frac{\partial^2 f(x,y)}{\partial y^2} + D\frac{\partial f(x,y)}{\partial x} + E\frac{\partial f(x,y)}{\partial y} + Ff(x,y) \quad = \quad H(x,y) \quad (1.9)$$

erfüllt, unter der Voraussetzung, daß die Anfangsbedingungen und die *Randbedingungen* bekannt sind. Die Koeffizienten A, B, C, D, E und F können genauso wie $H(x,y)$ stückweise stetige Funktionen der Variablen x und y sein. Partiellen Differentialgleichungen werden in Analogie zur Klassifikation von Kegelschnittgleichungen, in drei Klassen [27] eingeteilt, d.h. es gilt

- elliptisch, falls $AC - B^2 < 0$,
- hyperbolisch, falls $AC - B^2 < 0$ und
- parabolisch, falls $AC - B^2 = 0$.

Auch die Randbedingungen werden unterschieden:

- $f(x,y) = \varphi(x,y) \quad auf \quad \Gamma$ Dirichlet-Bedingung
- $\frac{\partial f(x,y)}{\partial \vec{n}} = \gamma(x,y) \quad auf \quad \Gamma$ Neumann-Bedingung
- $\frac{\partial f(x,y)}{\partial \vec{n}} + \alpha f(x,y) = \beta(x,y) \quad auf \quad \Gamma$ Cauchy-Bedingung

Hierbei sind $\varphi(x,y)$, $\gamma(x,y)$, $\alpha f(x,y)$ und $\beta(x,y)$ gegebene Funktion auf dem Rand Γ. Die Lösung mit der Differenzenmethode wird in vier Schritten vollzogen:

1. Das vom Rand Γ eingeschlossene Gebiet G wird mit einem regelmäßigen Gitter überzogen. Meist wird ein quadratisches Gitter mit Gitterabstand h verwendet. Die Funktionswerte $f(x_i, y_j)$ an den Gitterpunkten, bzw. den Schnittpunkten von Gitter und Rand, müssen bekannt sein oder berechnet werden.
2. Die Differentialquotienten werden durch Differenzenquotienten ersetzt;
   z.B.: $\frac{\partial^2 f(x_i,y_j)}{\partial x_i^2} \approx \frac{f(x_{i+1},y_j) - 2f(x_i,y_j) + f(x_{i-1},y_j)}{h^2}$.
3. Die Rand- und Anfangsbedingungen sind zu berücksichtigen. Damit lässt sich ein System von Differenzengleichungen aufstellen (pro Gitterpunkt eine Gleichung).
4. Das Gleichungssystem kann nach der unbekannten Funktion $f(x_i, y_j)$ aufgelöst werden und stellt die diskrete Form der gegebenen Randwertaufgabe dar.

Kapitel 2

CNN

Ein Zugang, der zum besseren Verständnis von CNN beiträgt, ist sich mit der historischen Entwicklung von **K**ünstlichen **N**euronalen **N**etzwerken (KNN) – die ihren Namen der Ähnlichkeit mit natürlichen *neuronalen Netzen* verdanken – zu befassen. Bei genauer Betrachtung lassen sich Zusammenhänge zwischen KNN und CNN erkennen. Es soll aber auch auf die Unterschiede hingewiesen werden.

2.1 Neuronale Netze

Bei neuronalen Netzen – wie auch in [2] beschrieben – stellen die Neuronen die kleinste Einheit dar. In Abbildung 2.1 ist der Aufbau eines biologischen Neurons schematisch dargestellt. Das Neuron besteht aus dem *Zellkörper*, auch *Soma* genannt, der den *Zellkern* umschließt. Vom

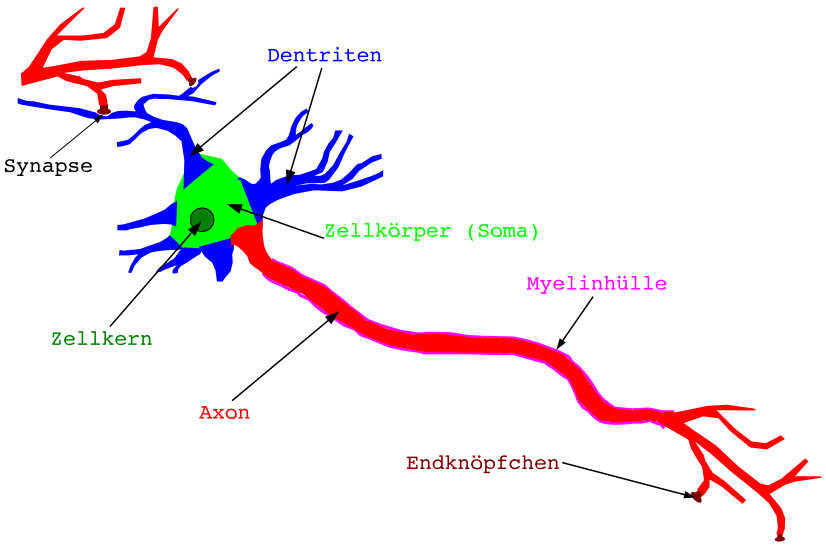

Abbildung 2.1: Aufbau biologischer Neuronen (schematisch)

Zellkörper gehen mehrere kurze, verästelte Zweige, die *Dentriten*, ab. Das *Axon*, das zwischen einigen Millimetern und einem Meter lang sein kann, ist meist von einer *Myelinhülle* umgeben. Am Ende ist das Axon auch stark verästelt und berührt beinahe die Dentriten eines anderen Neurons. Die Lücke zwischen den sogenannten *Endknöpfchen* des Axons und den Dentriten wird *Synapse* genannt. Die Synapse ist zwischen 10 und 50 Nanometer breit. Der Informationsaustausch zwischen den Neuronen erfolgt über die *Neurotransmitter*. Das sind Chemikalien, die von den Endknöpfchen ausgesandt werden und die Polarisation auf der Membran der Dentriten verändern. Die kleinen Änderungen der einzelnen Synapsen summieren sich in einem Neuron auf und die Potentialänderung, die *Aktionspotential* genannt wird, pflanzt sich entlang des Axons fort. Wenn das Aktionspotential das Ende des Axons erreicht hat, bewirkt es eine Ausschüttung von Neurotransmittern an den Endknöpfchen.
Ein vereinfachtes mathematisches Modell für Neuronen sind **Schwellenwertelemente**, die aus gewichteten Eingängen und einem Ausgang bestehen. Überschreitet die Summe der gewichteten Eingänge einen festgelegten Schwellenwert, so schaltet das *Schwellenwertelement* von einem Zustand in einen Anderen. Demgemäß stellt ein Schwellenwertelement die Funktion

$$o = \begin{cases} 1, & \text{falls} \quad \sum_{i=1}^{n} a_i s_i \geq v \\ 0, & \text{sonst} \end{cases} \tag{2.1}$$

mit

- Eingänge s_i
- Gewichte a_i
- Schwellwert v
- Ausgang o

dar. Durch die richtige Wahl der Gewichte und des Schwellwertes lassen sich verschiedene Boolesche Funktionen realisieren. Es ist jedoch nicht die Gesamtheit aller Funktionen mit einem Schwellenwertelement realisierbar, sondern nur linear-seperable Boolesche Funktionen. Dies ändert sich, wenn man Netze von Schwellenwertelementen verwendet. In [2] wird gezeigt, daß alle Booleschen Funktionen durch Schwellenwertelement-Netze aus zwei Schichten berechnet werden können. Die dazu benötigten Parameter (Gewichte und Schwellenwert) sind entweder bekannt oder können durch ein *Training* "gelernt", d.h. bestimmt werden. Für das Training müssen die Eingaben und die erwartete Ausgabe vorliegen. Die Eingaben werden in ein Netz, mit vorerst beliebigen Parametern, eingegeben und die Ausgabe berechnet. Diese wird mit der gewünschten Ausgabe verglichen. Die Differenz der beiden Ausgaben wird als *Fehler* angesehen. Die Parameter müssen nun derart geändert werden, daß der Fehler minimiert wird.
Um ein allgemeineres Modell für neuronale Netze zu erhalten, kann die Graphentheorie [6] herangezogen werden, wie in der folgenden Definition.

Definition 1. *Ein (künstliches)* **neuronales Netz** *ist ein (gerichteter) Graph $U = (C, K)$, dessen Knoten $c \in C$* **Neuronen** *und dessen Kanten $k \in K$* **Verbindungen** *heißen. Die Menge C der Knoten ist unterteilt in die Menge C_{ein} der* **Eingabeneuronen** *, die Menge C_{aus} der* **Ausgabeneuronen** *und die Menge $C_{versteckt}$ der* **versteckten Neuronen**. *Es gilt*

$$\begin{aligned} C &= C_{ein} \cup C_{aus} \cup C_{versteckt}, \quad C_{ein} \neq \emptyset, \\ C_{aus} &\neq \emptyset, \quad C_{versteckt} \cap (C_{ein} \cup C_{aus}) = \emptyset \end{aligned}$$

Jeder Verbindung $(c_i, c_j) \in K$ ist ein **Gewicht** $a_{c_j c_i}$ *zugeordnet und jedem Neuron $c \in C$ drei (reellwertige) Zustandsgrößen: die* **Netzeingabe** net_c*, die* **Aktivierung** akt_c *und die* **Ausgabe**

aus_c . Jedes Eingabeneuron $c \in C_{ein}$ besitzt außerdem eine vierte (reellwertige) Zustandsgröße, die **externe Eingabe** *ext_c. Weiter sind jedem Neuron $c \in C$ drei Funktionen zugeordnet:*

- *die* **Netzeingabefunktion** $f_{net}^{(c)} : \mathcal{R}^{2|pred(c)|+\kappa_1(c)} \rightarrow \mathcal{R}$
- *die* **Aktivierungsfunktion** $f_{akt}^{(c)} : \mathcal{R}^{\kappa_2(c)} \rightarrow \mathcal{R}$
- *die* **Ausgabefunktion** $f_{aus}^{(c)} : \mathcal{R} \rightarrow \mathcal{R}$

mit denen die Netzeingabe net_c, die Aktivierung akt_c und die Ausgabe aus_c des Neurons c berechnet werden. $\kappa_1(c)$ und $\kappa_2(c)$ hängen von der Art und den Parametern der Funktionen ab.[2]

Mit $pred(c)$ ist hier die Menge der Vorgänger von c gemeint. Zur Veranschaulichung ist ein einfaches (künstliches) neuronales Netz in Abb. 2.2 dargestellt. Es besteht aus vier Neuronen

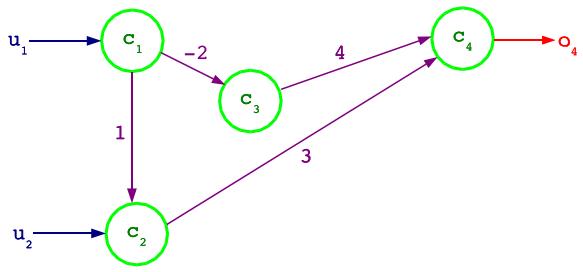

Abbildung 2.2: Graph eines neuronalen Netzes

$C = \{c_1, \ldots, c_4\}$, wobei $c_1, c_2 \in C_{ein}$ *Eingabeneuronen*, $c_4 \in C_{aus}$ ein *Ausgabeneuron* und $c_3 \in C_{versteckt}$ ein *verstecktes Neuron* darstellen. Die Eingabeneuronen erhalten die *externen Eingaben* u_1 und u_2 und die *Ausgabe* des Ausgabeneurons o_4 liefert die Ausgabe des neuronalen Netzes. Die *Verbindungen* und die zugehörigen *Gewichte* sind als Pfeile mit Zahlen dargestellt. Durch diese Art der Darstellung läßt sich die sogenannte **Netzstruktur** – oder Topologie des Netzes – leicht aufzeigen. Anhand der *Netzstruktur* kann man zwei Arten von neuronalen Netzen unterscheiden:

- vorwärtsbetriebene Netze (feedforward networks); der Graph der Netzstruktur enthält **keine** Schleifen
- rückgekoppelte Netze (recurrent networks); der Graph der Netzstruktur enthält Schleifen

Neben der Topologie des Netzes spielt die Art der Berechnung in den einzelnen Neuronen eine entscheidende Rolle für das Verhalten des Netzes. Dazu kann man ein Neuron, wie es in Abb. 2.3 gezeigt ist, als einfachen Prozessor ansehen. Die *Netzeingabefunktion*

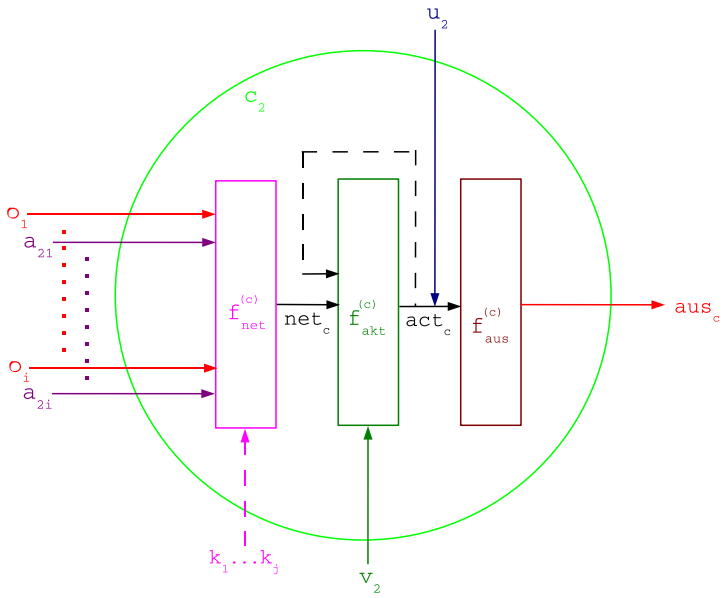

Abbildung 2.3: Aufbau des verallgemeinerten Neurons c_2

$f_{net}^{(c)}(o_1, o_3, \ldots, o_i, w_{21}, w_{23}, \ldots, w_{2i}, k_1, \ldots, k_l)$ berechnet aus den Ausgaben der Vorgänger des Neurons c_2 und den entsprechenden Gewichten die *Netzeingabe* net_c. Die Parameter $k_1, \ldots, k_l$ sind in den meisten KNN optional. Aus der Netzeingabe, einer bestimmten Zahl von Parametern $v_2 = (\theta_1, \ldots, \theta_m)$ – z.B. ein Schwellenwert – und, je nach Art des KNN, der aktuellen Aktivierung des Neurons berechnet die *Aktivierungsfunktion* $f_{akt}^{(c)}(net_c, v_2, akt_c)$ die neue Aktivierung akt_c. Wenn c_2 ein Eingabeneuron ist, wird mit der externen Eingabe u_2 die Anfangs-Aktivierung gesetzt. Die *Ausgabefunktion* $f_{aus}^{(c)}$ dient dazu, die Ausgabe des Neurons aus_c anhand von akt_c in einen gewünschten Wertebereich zu transformieren. Die Berechnung des gesamten Netzes wird in zwei Phasen unterteilt:

- die Eingabephase; sie dient der Initialisierung des Netzes mit den externen Eingaben (Eingabeneuronen) oder willkürlichen Werten, bzw. 0 (restliche Neuronen)
- die Arbeitsphase; hier werden die externen Eingaben abgeschaltet und die Aktivierung, sowie die Ausgabe der Neuronen – gegebenenfalls mehrfach – neu berechnet

Die Berechnung der einzelnen Neuronen kann synchron oder asynchron, in einer festgelegten Reihenfolge, erfolgen.
Wie bei den *Schwellenwertelementen* kann das Verhalten des neuronalen Netzes trainiert werden. Dabei werden zwei Arten von Lernaufgaben unterschieden:

- die feste Lernaufgabe; dem Netz werden verschiedene (externe) Eingaben und die zugehörigne Ausgaben als **Lernmuster** (Lm) präsentiert. Daraufhin sollen die Parameter (Gewichte, Schwellwert, usw.) so optimiert werden, daß auch für beliebige Eingaben die richtige Ausgabe geliefert wird.
- die freie Lernaufgabe; das Netz bekommt nur die Eingaben als *Lernmuster* und soll für "ähnliche Eingaben ähnliche Ausgaben" liefern.

Die gebräuchlichere Art ist die *feste Lernaufgabe*, bei der eine **Fehlerfunktion** verwendet wird, um zu bestimmen, wie gut ein neuronales Netz die Aufgabe löst. Die *Fehlerfunktion* kann, je nach Anwendung, unterschiedlich definiert sein, aber in der Regel verwendet man die Summe der Quadrate der Abweichungen von gewünschter zur tatsächlichen Ausgabe über alle Lernmuster.
Diese sehr allgemeine Definition von (künstlichen) neuronalen Netzen läßt sich auf unterschiedliche Weise spezialisieren. Eine bekannte Form sind die *Hopfield-Netze*, die ursprünglich als physikalische Modelle zur Beschreibung des Magnetismus in Spingläsern eingeführt wurden, aber auch Assoziativspeicher beschreiben.

2.2 Zellulare Nichtlineare Netze

Zellulare Nichtlineare Netze weisen insbesondere folgende Unterschiede zu KNN auf; es gilt:

- Es gibt nur eine Sorte von Neuronen c (von hier ab als *Zellen* bezeichnet), die zugleich Eingabe- und Ausgabeneuronen sind $c \in C_{ein}, c \in C_{aus}$. Bzw. $C = C_{ein} = C_{aus}$ und $C_{versteckt} = \emptyset$.
- Es werden nur lokale **Koppelungen** (Verbindungen) $(c_i, c_j) \in K$ zwischen Zellen, die in einer definierten **Nachbarschaft** $\mathcal{N}$ liegen, zugelassen.
- Die Aktivierung akt_c läßt sich dem *(Zell-)Zustand* s_i zuordnen.
- Die Aktivierungsfunktion $f_{akt}^{(c)}$ kommt in seiner ursprünglichen Form nicht mehr vor. Stattdessen wird eine **Zustandsgleichung** eingeführt.
- Die *externe Eingabe* $ext_c = \vec{u}$ (ab hier nur noch Eingabe genannt) wird mit Gewichten b_k versehen.
- Die Netzeingabe net_c, bzw. Netzeingabefunktion $f_{net}^{(c)}$ ist ein Teil der *Zustandsgleichung* und wird nicht mehr getrennt aufgeführt.
- Es gibt nur einen zusätzlichen Parameter v_i, der *Bias* genannt wird.

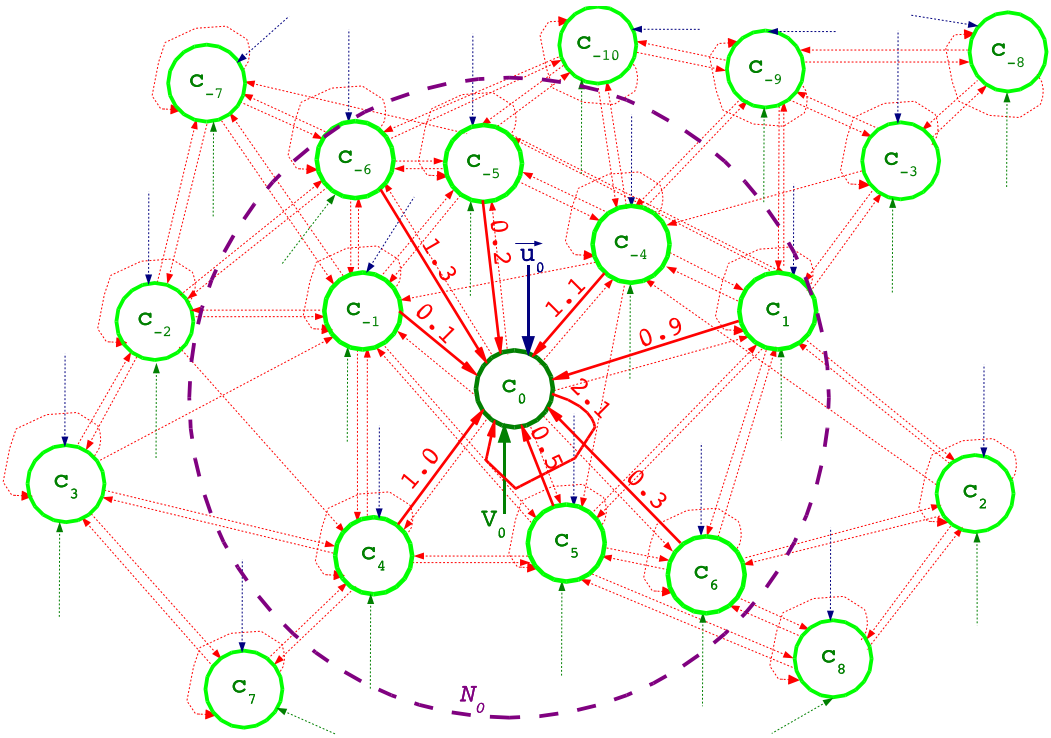

Abbildung 2.4: Beispiel eines KNN, daß sich mit einem CNN vergleichen läßt

In Abb. 2.4 ist ein Beispiel für ein mögliches KNN zu sehen, welches sich mit einem CNN vergleichen läßt. Die **Zelle** c_0 ist zur genaueren Betrachtung hervorgehoben und dient als konkretes Beispiel für die folgenden Ausführungen. Ihr Ausgang ist auch auf sie selbst rückgekoppelt, d.h. es handelt sich um ein rekurrentes Netzwerk. Sie bekommt nur gewichtete Ausgänge von Zellen die in der *Nachbarschaft* $\mathcal{N}_0$ (in der Abb. violett gestrichelt dargestellt) liegen. In Abb. 2.5 ist die Zelle c_0 schematisch dargestellt. Die *Zustandsgleichung* der Zelle c_0 beschreibt ihre zeitliche

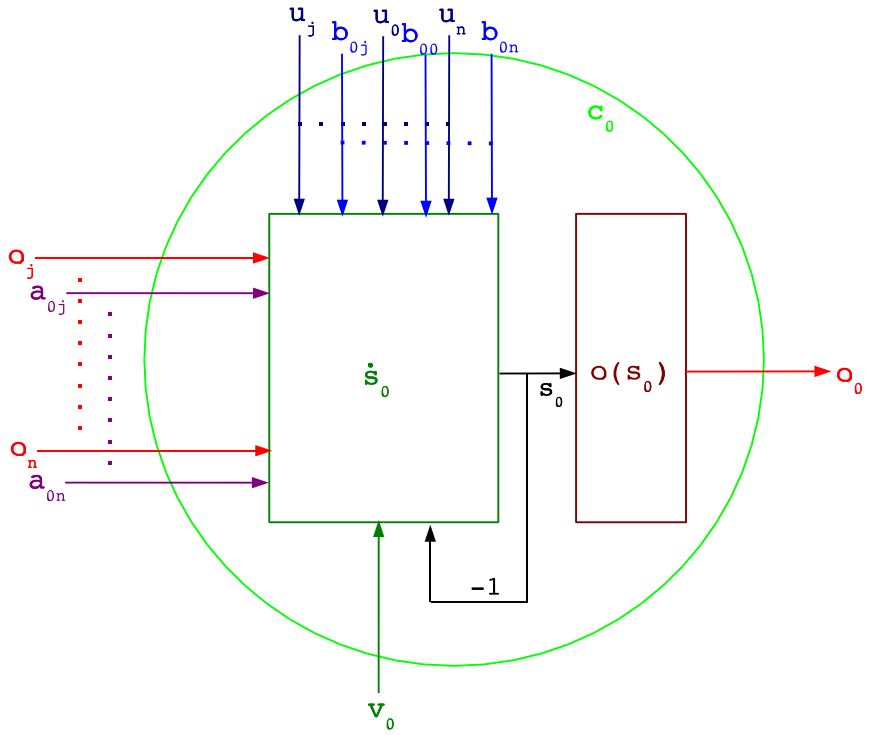

Abbildung 2.5: Eine einzelne Zelle eines CNN

gemäß:

$$\begin{aligned} \dot{s}_0 &= -s_0 + \sum_{j \in \mathcal{N}_0} a_{0j} o(s_j) + \vec{b}_0^T \vec{u}_0 + v_0 \\ &= -s_0 + \sum_{j \in \mathcal{N}_0} a_{0j} o(s_j) + \sum_{j \in \mathcal{N}_0} b_{0j} u_j + v_0 \end{aligned} \tag{2.2}$$

Im Gegensatz zu anderen Netzen kann bei CNN die (externe) Eingabe $\vec{u}_0$ der Zelle, neben der Eingabe u_0 für die eigene Zelle, auch die *Eingaben der Nachbarzellen* enthalten. Deshalb wird auch die Vektorschreibweise verwendet. Der Vektor $\vec{b}_0^T$, bzw. die Parameter b_{0j} gewichten die Eingaben. Die Summe $\sum_{c_j \in \mathcal{N}_0} a_{0j} o(s_j)$ ist vergleichbar mit der Netzeingabefunktion $f_{net}^{(c)}$. Der *Bias* v_0 ist fest eingestellt.
Zur besseren Strukturierung ordnet man die einzelnen Zellen nicht beliebig – wie in Abb. 2.4 – an, sondern plaziert sie z.B. auf einem würfelförmigen Gitter. Es wird nun je nach Anordnung zwischen 1-, 2- oder 3-dimensionalen CNN unterschieden. Häufig werden nur *translationsinvariante* CNN betrachtet, bei denen die Gewichtung zwischen den Zellen nur von der relativen

Position der betrachteten Zelle zu den Zellen in der Nachbarschaft abhängt. Die Zellen, die am Rand des Netzwerkes liegen, müssen in diesem Fall gesondert betrachtet werden. In Abb. 2.6 ist

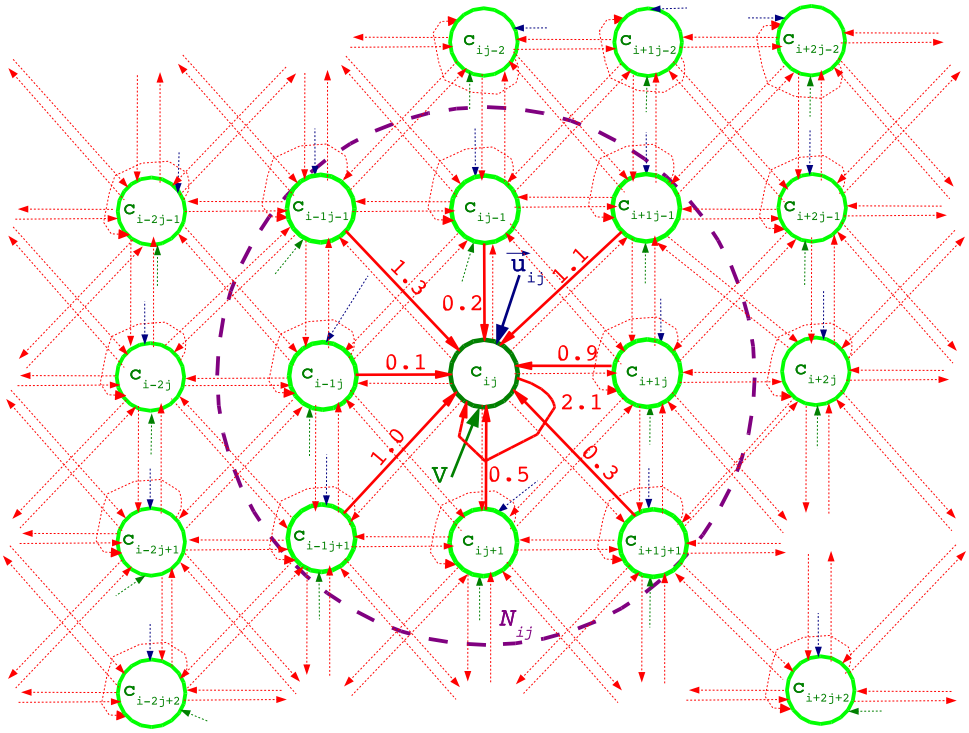

Abbildung 2.6: Ausschnitt eines 2-dimensionalen translationsinvarianten CNN

ein Ausschnitt eines 2-dimensionalen translationsinvarianten CNN zu sehen, bei dem die Zelle c_{ij} zur genaueren Betrachtung hervorgehoben ist. Diese Zelle kann an einer beliebigen Stelle – außer auf dem Rand – im CNN sitzen, da die gewichteten Verbindungen und Eingänge nur von der relativen Position der Zellen abhängen. Das heißt, daß z.B. die Verbindung von der Zelle c_{i-1j-1} "links oben" immer mit 1.3 gewichtet wird. Entsprechendes gilt für die anderen Nachbarzellen und die Rückkopplung auf die Zelle selbst. Auch der Bias v ist für alle Zellen gleich. Der Nachbarschaftsradius $r = 1$ bedeutet, nur die nächsten Nachbarn haben Kopplungen zur Zelle c_{ij}. Daraus folgt für die Nachbarschaft $\mathcal{N}_{ij} = \{k, l = 1, 2, 3, \ldots : max\{|k-i|, |l-j|\} \leq r\}$. Im Gegensatz zum in Abb. 2.4 gezeigten allgemeinen Beispiel, wo jede Zelle ihre eigene Zustandsgleichung haben kann, lassen sich alle Zellen (außer am Rand) durch eine Zustandsgleichung beschreiben:

$$\dot{s}_{ij} = -s_{ij} + \sum_{k,l \in \mathcal{N}_{ij}} a_{kl} o(s_{kl}) + \sum_{k,l \in \mathcal{N}_{ij}} b_{kl} u_{kl} + v \tag{2.3}$$

2.2.1 Standard CNN

In den von Chua und Yang [5] 1988 vorgestellten CNN's wurden die Zellen ursprünglich, wie in Abb. 2.7 dargestellt, durch ein nichtlineares *RC-Netzwerk* realisiert. Die Schaltung einer Zelle besteht dann aus

- einem Kondensator C_s,

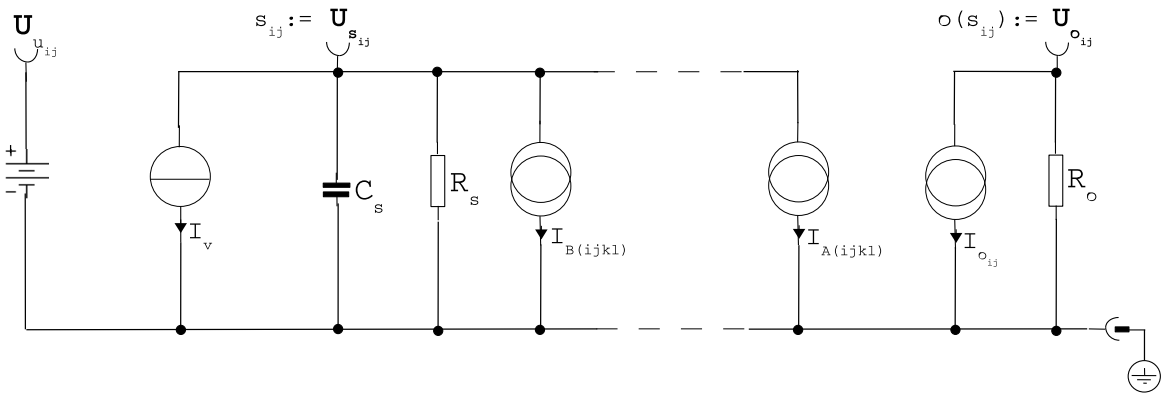

Abbildung 2.7: Ersatzschaltbild einer als RC-Netzwerk realisierten Zelle

- zwei Widerständen R_s und R_o,
- einer unabhängigen Stromquelle I_v,
- mehreren linearen spannungsgeregelten Stromquellen $I_{A(ijkl)} = a_{ijkl}U_{okl}$, bzw. $I_{B(ijkl)} = b_{ijkl}U_{ukl}$,
- einer unabhängigen Spannungsquelle,
- und einer **nichtlinearen** spannungsgeregelten Stromquelle $I_{o_{ij}} = o(U_{s_{ij}})$.

Die Spannung $U_{s_{ij}}$ wird der Zustand s_{ij} der Zelle c_{ij} genannt. Der Widerstand R_o stellt den Ausgangswiderstand der Schaltung dar und die Spannung $U_{o_{ij}}$ ist dann die Ausgangsspannung der Zelle c_{ij}. Die Spannungen U_{okl} sind dementsprechend die Ausgangsspannungen der Nachbarzellen mit $k, l \in \mathcal{N}_{ij}$. Analog sind die Spannungen $U_{u_{ij}} =: u_{ij}$ und $U_{u_{kl}} =: u_{kl}$ die Eingangsspannungen der Zelle c_{ij}, bzw. der Nachbarzellen $c_{kl} \in \mathcal{N}_{ij}$. Die Stromquelle I_v wird als Bias v_{ij} bezeichnet. Die Zustandsänderung $\dot{s}_{ij}$ wird durch die Gleichung

$$\dot{s}_{ij} = -\frac{1}{C_s R_s} s_{ij} + \sum_{k,l \in \mathcal{N}_{ij}} a_{kl} o(s_{kl}) + \sum_{k,l \in \mathcal{N}_{ij}} b_{kl} u_{kl} + v_{ij} \tag{2.4}$$

beschrieben. Als *Ausgangsfunktion* $o(s_{ij})$ der Zelle c_{ij} wird meist die in Abb. 2.8 dargestellte stückweise lineare Funktion

$$o(s_{ij}) = \frac{1}{2}\left(|s_{ij}+1| - |s_{ij}-1|\right) \tag{2.5}$$

verwendet. Ein so definiertes CNN – beschrieben durch die Gleichungen (2.4) und (2.5) – wird als *Standard CNN* bezeichnet. Bei translationsinvarianten CNN lassen sich die Kopplungen der Zellen a_{kl} und b_{kl} für eine strukturierte Darstellung als "Matrix" anordnen. Dabei ist die räumliche Anordnung der Gewichte erkennbar; zum Beispiel bei einem Nachbarschaftsradius von $r = 1$ folgt

$$\mathbf{A} := \begin{pmatrix} a_{-1-1} & a_{0-1} & a_{+1-1} \\ a_{-10} & a_{00} & a_{+10} \\ a_{-1+1} & a_{0+1} & a_{+1+1} \end{pmatrix} \text{, bzw. } \mathbf{B} := \begin{pmatrix} b_{-1-1} & b_{0-1} & b_{+1-1} \\ b_{-10} & b_{00} & b_{+10} \\ b_{-1+1} & b_{0+1} & b_{+1+1} \end{pmatrix}.$$

Diese werden nun auch **Templates** genannt, das bedeutet **A** heißt *Feedback-Template* und **B** wird *Feedforward-Template* genannt. In diesem Fall laßt sich durch die Angaben der beiden Templates

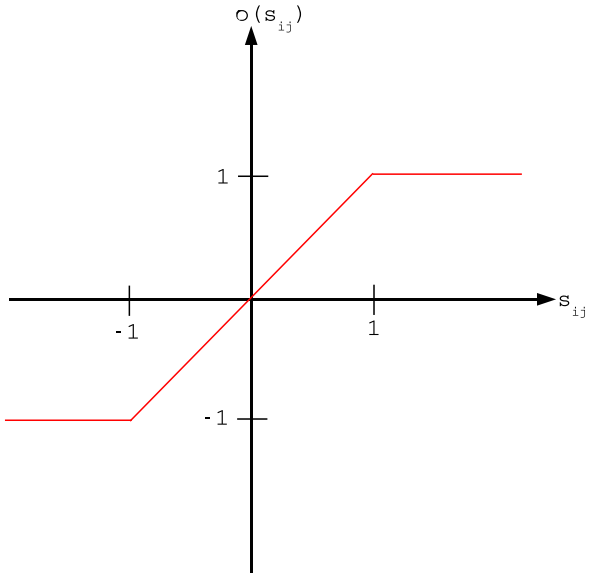

Abbildung 2.8: Die stückweise lineare Ausgangsfunktion des Standard CNN

A, **B** und des Bias v das Verhalten des CNN beschreiben, wenn noch eine Bedingung für die Randzellen angegeben wird. Die gebräuchlichsten **Randbedingungen** sind:

- Dirichletsche Randbedingungen; hier werden die Zustände der Randzellen auf einen festen Wert gesetzt
- von Neumannsche Randbedingungen; dazu werden die die Zustände der Randzellen auf den Zustandswert ihrer direkten Nachbarzelle gesetzt
- Ringförmig geschlossene CNN; es existieren keine "echten" Randzellen, da jede linke Randzelle mit einer rechten Randzelle gekoppelt wird

2.2.2 Erweiterte Definition von CNN

In den bisherigen Betrachtungen von CNN waren die Zellkopllungen immer linear gewichtet. Die Nichtlinearität kommt ausschließlich durch die Ausgangsfunktion zustande. Die im folgenden dargestellte erweiterte Definitionsweise läßt aber auch nichtlineare Kopplungen zu, die für einige Anwendungen [11, 24, 12] sinnvoll und notwendig sind. Die *Kopplungsfunktionen* $a_{ij}\left(o(s_{kl})\right)$ und $b_{ij}\left(u_{kl}\right)$ können auf unterschiedlichste Weise repräsentiert werden[16, 26, 22]. Die Zustandsgleichung (2.4) wird dann allgemeiner zu

$$\dot{s}_{ij} \quad = \quad -s_{ij} + \sum_{k,l \in \mathcal{N}_{ij}} a_{ij}\left(o(s_{kl})\right) + \sum_{k,l \in \mathcal{N}_{ij}} b_{ij}\left(u_{kl}\right) + v_{ij} \tag{2.6}$$

mit $\tau := 1/R_s C_s = 1$.
Da sich Funktionen als Polynom darstellen lassen, sind polynomiale Kopplungsfunktionen [22]

$$a_{ij}\left(o(s_{kl})\right) = \sum_{m \geq 0}^{P} p_{ijkl} o(s_{kl})^m \quad \text{, bzw.} \quad b_{ij}\left(u_{kl}\right) = \sum_{n \geq 0}^{P} q_{ijkl} u_{kl}^n \tag{2.7}$$

ein naheliegender Ansatz, mit dem Polynomgrad P und den Koeffizienten p_{ijkl} und q_{ijkl}. Mit tabellierten Funktionen lassen sich nicht nur polynomialen Gewichte, sondern auch andere Funktionen, die sonst als Reihen entwickelt werden müssten, als Kopplung verwenden. Die Interpolation zwischen den Stützstellen (tabellierte Werte) kann linear oder durch Spline-Interpolation[17] erfolgen. Oft wird hierbei als Ausgangsfunktion $o(s_{ij})$ die *Identität* gewählt, so daß die Nichtlinearität ausschließlich durch die Kopplungsfunktionen zustande kommt. Diese Art von CNN wird auch zur Modellierung von partielle Differentialgleichungen häufig verwendet. In dieser Arbeit werden eindimensionale autonome CNN's mit

$$\dot{s}_i \quad = \quad -s_i + \sum_{k \in \mathcal{N}_i} a_i \left(o(s_k) \right) + v_i \tag{2.8}$$

zugrundegelegt; d.h. es werden gemäß $u_i = 0$ keine Eingänge berücksichtigt.

Mehrschichtige CNN

Eine andere Form bilden sogenannte *mehrschichtige CNN*. Hier wird die 3. räumliche Dimension dadurch repräsentiert, indem mehrere 2-dimensionale CNN, wie in der Abb. 2.9 zu sehen, übereinander geschichtet werden. Zu den Kopplungen innerhalb einer Schicht aus Zellen kommen

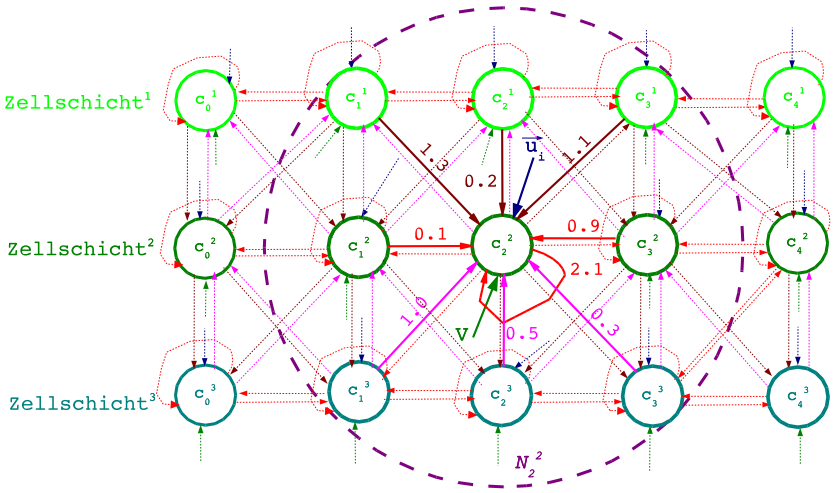

Abbildung 2.9: Mehrschichtiges CNN mit drei 1-dimensionalen Zellschichten

nun noch Verbindungen zwischen den Zellschichten. Die Nachbarschaft $\mathcal{N}_i^m$ (im Beispiel $\mathcal{N}_2^2$) gilt nicht nur in der Zellschicht L^m (bzw. L^2) sondern auch für die Kopplungen zwischen den Zellschichten. Die Eingaben $\vec{u}_i$ sind häufig davon ausgenommen, d.h. sie wirken nicht Schichtübergreifend. Für die Gewichte wird eine neue Schreibweise eingeführt, um die Wirkung zwischen Zellschichten deutlich zu machen. Die Richtung der Kopplung geht dabei immer von m' nach m und ein Gewicht wird $a_i^{m'm}$, bzw. $a_i^{m'm}(o(s_i))$ für Gewichtsfunktionen, geschrieben. Die allge-

meine Zustandsgleichung ein CNN mit N Schichten besitzt demgemäß die Form

$$\dot{s}_i^m = -s_i^m + \sum_{m'=1}^{N} \left(\sum_{c_k^m \in \mathcal{N}_i^m} a_i^{m'm} \left(o(s_k) \right) \right) + \sum_{c_k^m \in \mathcal{N}_i^m} b_i \left(u_k \right) + v_i \; . \tag{2.9}$$

Diese Art von CNN wird in den unterschiedlichsten aktuellen Anwendungen [22, 25, 18] verwendet.
Schließlich gab Chua in der umfassenden Monographie [4] eine weitere verallgemeinerte Definition an, die als Grundlage zahlreicher aktueller Arbeiten dient; sie lautet

Definition 2 (CNN). *Ein CNN [4] ist eine beliebige räumliche Anordnung von lokal gekoppelten Zellen; wobei jede Zelle ein dynamisches System beschreibt, das Eingänge, Ausgänge und einen Zustand, der sich nach vorgeschriebenen Gesetzen entwickelt, besitzt.*

Kapitel 3

Partielle Differentialgleichungen

Um eine partielle Differentialgleichungen auf ein CNN abzubilden, muß diese mit der in Abschnitt 1.2.2 beschriebenen Differenzenmethode räumlich diskretisiert werden. Hierbei werden nur die ersten beiden Schritte durchgeführt. Die so entstandenen Systeme lokal gekoppelter Differentialgleichungen werden nun mit der Zustandsgleichung (2.6) verglichen, wobei der Term $H(x, y)$ aus der Gleichung (1.9) den Eingängen $\vec{u}$ und die restlichen Werte den Zellzuständen $\vec{s}$ zugewiesen werden. Durch Koeffizientenvergleich lassen sich die Templates ablesen.

3.1 Korteweg-de Vries-Gleichung (KdV) Gleichung

Die sogenannte Korteweg-de Vries-Gleichung [7] beschreibt ein Wellenphänomen, welches 1834 J. Scott Russel [9] das erste Mal beobachtet wurde. Um den Bug eines Bootes, das sich zunächst in einem engen, flachen Kanal bewegte, und dann abrupt gestoppt wurde, sammelte sich ein Wasserberg an der sich mit unverminderter Geschwindigkeit durch den Kanal ohne Änderung der Form und Geschwindigkeit fortbewegte. Russel hatte eine Solitärwelle beobachtet. Zarbusky und Kruskal beschäftigten sich in den 1960er mit dem sogenannten Fermi-Pasta-Ulam (FPU) Modell für die Leitfähigkeit in Festkörpern. Hierbei beobachteten sie solitäre Wellen, die die Eigenschaft hatten, daß sie ohne Formänderung durcheinander hindurch propagierten. Sie prägten darauf hin für Wellen mit einem solchen Verhalten den Begriff Solitonen. Zur Beschreibung dieser Phänomene konnten sie eine Gleichung herleiten, die einer Form der schon 1895 von Korteweg und de Vries aufgestellten nichtlinearen Differentialgleichung entsprach. Die Korteweg-de Vries-Gleichung [7] ist eine nichtlineare partielle Differentialgleichung, die als Lösung solitone Wellen besitzt. Die KdV Gleichung kann in verschiedenen Formen auftreten, die durch lineare Transformationen äquivalent ineinander überführt werden können [19]. Eine mögliche Darstellung ist folgende Gleichung

$$\frac{\partial f(x,t)}{\partial t} = 2\frac{\partial^3 f(x,t)}{\partial x^3} - \frac{\partial f^2(x,t)}{\partial x} \tag{3.1}$$

wie sie in [11] und [23] verwendet wird. Die folgende Form

$$\frac{\partial f(x,t)}{\partial t} = -\frac{\partial^3 f(x,t)}{\partial x^3} + 3\frac{\partial f^2(x,t)}{\partial x} \tag{3.2}$$

ist weiter verbreitet, z.B. [7], [13] und [19]. Die räumliche Diskretisierung der Gleichung (3.1) ergibt:

$$\begin{aligned}\frac{\partial f(x_i,t)}{\partial t} &\approx \frac{f(x_{i+2},t)-2f(x_{i+1},t)+2f(x_{i-1},t)-f(x_{i-2},t)}{h^3} \\ &\quad -\frac{f^2(x_{i-1},t)-f^2(x_{i+1},t)}{2h}\end{aligned} \tag{3.3}$$

Mit $h=1$ und etwas umsortiert sieht die Gleichung so aus:

$$\begin{aligned}\frac{\partial f(x_i,t)}{\partial t} &\approx f(x_{i+2},t)-2f(x_{i+1},t)+\frac{1}{2}f^2(x_{i+1},t)+0 \\ &\quad +2f(x_{i-1},t)-\frac{1}{2}f^2(x_{i-1},t)-f(x_{i-2},t)\end{aligned} \tag{3.4}$$

Die Zustandsgleichung (2.8) mit polynomialen Gewichtsfunktionen $a_i\left(o(s_k)\right)=\sum_{m\geq 0}^{P} p_{ik}o(s_k)^m$ wird verwendet, um (3.4) darzustellen. Dazu muß als Ausgangsfunktion die Identität $o(s_i)=s_i$ gewählt werden, $v_i=0$ und der Term $-s_i$ fällt weg ($\tau=0$), so daß die Gleichung

$$\dot{s}_i = \sum_{k\in\mathcal{N}_i}\sum_{n=1}^{P} p_{ikn}s_k^n \tag{3.5}$$

resultiert. Durch Vergleich (3.4) lassen sich die Werte p_{ikn} ablesen.

3.2 Burgers Gleichung

Eine weitere partielle Differentialgleichung ist die Burgers Gleichung [7]

$$\frac{\partial f(x,t)}{\partial t} = \frac{\partial f(x,t)^2}{\partial x}+\frac{\partial^2 f(x,t)}{\partial x^2}\,, \tag{3.6}$$

die zur Modellierung eindimensionaler Flußprozesse, Turbulenzen, Schockwellen und Massentransport verwendet wird. Analog zur KdV wird die räumliche Diskretisierung vorgenommen und aus Gl. (3.6) wird

$$\frac{\partial f(x_i,t)}{\partial t} = \frac{f(x_{i-1},t)^2-f(x_{i+1},t)^2}{2h}+\frac{f(x_{i-1},t)-2f(x_i,t)+f(x_{i+1},t)}{h^2}\,, \tag{3.7}$$

mit dem Diskretisierungsabstand h. Durch Vergleichen mit (3.5) erhält man auch hier die Werte p_{ikn}.

3.3 Φ^4 Gleichung

Die Φ^4 Gleichung[7] ist durch

$$\frac{\partial^2 f(x,t)}{\partial t^2} = \frac{\partial^2 f(x,t)}{\partial x^2}+f(x,t)^3-f(x,t) \tag{3.8}$$

gegeben. Sie gehört zu den nichtlinearen Klein-Gordon Gleichungen, deren Lösungen stark von der Wahl der Anfangsbedingungen abhängen. Wegen der zweiten Ableitung nach der Zeit läßt sich

diese nicht direkt auf ein CNN übertragen. Durch die Substitution $w(x,t) = \frac{\partial f(x,t)}{\partial t}$ resultieren die folgende Gleichungen

$$w(x,t) = \frac{\partial f(x,t)}{\partial t} \tag{3.9}$$

$$\frac{\partial w(x,t)}{\partial t} = \frac{\partial^2 f(x,t)}{\partial x^2} + f(x,t)^3 - f(x,t)\ . \tag{3.10}$$

Diese lassen sich, nach der räumlichen Diskretisierung, auf ein CNN mit zwei Zellschichten (siehe Abschnitt 2.2.2) übertragen. Dabei werden die einzelnen Terme aus (3.9) und (3.10) den Kopplungen zwischen den Schichten zugeordnet.

Teil II

Verfahren

In den beiden folgenden Verfahren werden immer tabellierte Funktionen mit linearer Interpolation verwendet. Die auf diese Weise dargestellten Funktionen können in CNN sowohl als Ausgangs-Funktion, als auch als Gewichtsfunktion in der **A**- oder **B**-Template vorkommen. Der Unterschied zu polynomialen Templates ist, daß beliebige nichtlineare Funktionen – auch solche, die nicht mit Polynomen approximiert werden können – darstellbar sind. Ein Nachteil der herkömmlichen Vorgehensweise, Tabellen mit äquidistanten Stützstellen anzulegen, ist, daß für eine ausreichend hohe Approximationsgenauigkeit oft sehr große Tabellen benötigt werden. Mit nicht äquidistanten Abständen lassen sich bei gleichbleibender Genauigkeit eine erhebliche Anzahl an Stützstellen einsparen. Dazu muß die Lage der Stützstellen jedoch genau an die Funktion angepaßt werden. Die optimale Lage der Stützstellen automatisch zu bestimmen ist Ziel der beiden neuen im folgenden beschriebenen Verfahren, die dies auf ganz unterschiedliche Weise bewerkstelligen.
Das sogenannte **T**able **M**inimising **A**lgorithm (TMA [11]) Verfahren basiert auf einer Darstellung mit äquidistanten Stützstellen mit einer hohen Anzahl und minimiert diese, bis in der Regel nur noch nicht äquidistante Stützstellen übrig bleiben. Das **G**elernte **L**ineare **I**nterpolation (GLI) Verfahren beginnt mit zwei Stützstellen (Randwerte einer Funktion); dabei wird versucht die dazwischen liegenden nicht äquidistante Stützstellen mit einem Trainingsalgorithmus zu finden.

Kapitel 4

Lineare Interpolation von Gewichtsfunktionen in Zellularen Nichtlinearen Netzwerken

4.1 Der Table Minimising Algorithm (TMA)

Das Verfahren mit dem TMA besteht im Prinzip aus zwei Schritten:

1. Eine Tabelle mit N äquidistante Stützstellen wird angelegt. Dabei müssen die Stützstellen so *dicht* liegen, daß bei linearer Interpolation eine ausreichend hohe erreicht wird. Die Funktionswerte an den Stützstellen müssen bekannt sein oder berechnet werden. Die lineare Interpolation würde jetzt schon funktionieren, aber die Tabelle wäre viel zu groß.

2. Mit dem TMA werden die Stützstellen so weit wie möglich reduziert, so daß am Ende nicht äquidistante Stützstellen übrig bleiben.

Zur Reduzierung der Stützstellen wird ein Greedy-Algorithmus – wie in [20] beschrieben – verwendet. Solche Algorithmen finden oft nur suboptimale Lösungen; sind aber wesentlich schneller als Algorithmen die immer ein optimale Lösung suchen.

4.1.1 Greedy-Algorithmus

Zur Erläuterung betrachten wir als Beispiel den in Abb. 4.1 dargestellten Fall. Der hierbei verwendete Greedy-Algorithmus benötigt als Eingabe neben den äquidistanten Stützstellen $x_i \forall i = 1 \ldots N$ und den zugehörigen Funktionswerten $y(x_i) \forall i = 1 \ldots N$, eine Angabe des maximal zulässigen Fehlers ϵ und eine Größe R, die eine Art Schrittweite darstellt. Der Algorithmus läuft iterativ von einer gefundenen Stützstelle x_i, zu Beginn die erste x_1, zur nächsten x_{1+1}, bzw. am Ende zur letzten Stützstelle x_N. In jedem Iterationsschritt wird zuerst anhand einer Stützstelle x_{i+R} im Abstand R geprüft, welcher Fehler $e = |y(k) - g(k)| \quad \forall k = i+R-1, i+R-2, \ldots, i+1$ resultiert, wenn die beiden Stützstellen mit einer Geraden

$$\begin{aligned} g(k) &= \frac{y_{i+R} - y_i}{x_{i+R} - x_i} x_k - \frac{y_{i+R} - y_i}{x_{i+R} - x_i} x_i + y_i \\ mit & \qquad k = i + R - 1, i + R - 2, \ldots, i + 1 \end{aligned} \tag{4.1}$$

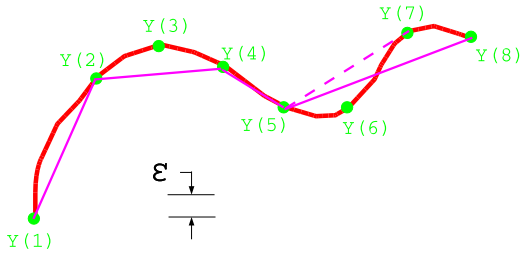

Abbildung 4.1: Anwendung des Greedy-Algorithmus am Beispiel

wie Gleichung 1.1 verbunden werden. Dann wird rückwärts jede Stützstelle zwischen R und x_i geprüft bis die Bedingung $e < \epsilon$ erfüllt ist. Ist dies der Fall wird Überprüfung abgebrochen und die restlichen Stützstellen werden **nicht** mehr berücksichtigt. Die letzte Stützstelle x_k, bei der der Fehler e noch kleiner ϵ war, wird zur aktuellen betrachteten Stützstelle x_i und der Vorgang wird wiederholt. Am Schluß wird die letzte Stützstelle x_N genommen, auch wenn R über diese hinaus reichen würde. Es wird also die Stützstelle gesucht, die am weitesten von der aktuellen entfernt ist und noch einen Fehler kleiner als ϵ erzeugt. Dadurch werden möglichst viele Stützstellen der äquidistanten Darstellungsform nicht mehr berücksichtigt. Die gefundene Lösung ist aber nicht minimal, da nicht alle Kombinationen geprüft werden, da wie oben beschrieben die Suche abgebrochen wird, sobald die Bedingung $e < \epsilon$ erfüllt wird. Es kann vorkommen, daß im nächsten Schritt mehr Stützstellen "eingespart" werden könnten. An der Stelle $x_i = 2$ im Beispiel (Abb.4.1), d.h. $y(2)$; bricht der Algorithmus die Prüfung bei $y(4)$ ab, da $e < \epsilon$ erfüllt ist und die nächste Stützstelle liegt zwangsläufig bei $y(5)$. Im nächsten Schritt – $y(3)$ – wäre als nächste Stützstelle $y(6)$ möglich. Es könntene $y(4)$ und $y(5)$ eingespart werden, d.h. es würde so eine Stützstelle weniger benötigt. Die Reduktion der Stützstellen ist jedoch für die in dieser Arbeit gezeigten Fälle (siehe Abschnitt III) ausreichend.

4.1.2 Algorithmische Komplexität

Um Algorithmen vergleichen zu können, kann die algorithmische Komplexität[6] näher betrachtet werden. Dazu wird meist der (Zeit-)Aufwand $T(N)$ in Abhängigkeit der Größe der Eingaben N betrachtet. Da dies erst für große N von Bedeutung ist, wird die *asymptotische Notation*[6] benutzt. Die sogenannte O-Notation beschreibt eine asymptotische *obere Grenze* und ist wie folgt definiert:

$$O\left(f_g(N)\right) \quad = \quad \{f_T(N) | \exists d, N_0 > 0 : 0 \leq f_T(N) \leq d f_g(N) \forall N \geq N_0\}.$$

Das heißt $f_g(N)$ ist eine asymptotische obere Grenze, weil ab N_0 alle Funktionswerte von $f_T(N)$ kleiner als das Produkt der *Konstante* d und der Funktion $f_g(N)$ sind.

Komplexität des Greedy-Algorithmus

Wir benutzen die Analogie zu einem *gewichteten Graphen* um die Komplexität des Greedy-Algorithmus zu analysieren. Die Stützstellen korrespondieren mit den Knoten $c_i \in C$ und die Geraden mit den Kanten $(c_i, c_j) \in K$. Den Knoten, der der ersten Stützstelle x_1 entspricht, wird mit c_{quelle} und der letzte (c_N) mit c_{senke} bezeichnet. Von jedem Knoten c_i – außer von c_{senke} selbst – existiert nun eine endliche Anzahl von *Pfaden* $P(c_{quelle}, c_{senke})$ nach c_{senke}. Wenn der maximale absolute Fehler e einer Geraden größer als ϵ ist, wird das zugehörige Gewicht w_{ji} auf ∞ gesetzt, sonst auf 1. Dadurch hat sich das Problem auf die Analyse des bekannten Problems "Kürzester Wege" in einem gewichteten Graphen [21] reduziert. Um die optimale Lösung zu erhalten müssen alle möglichen Pfade von c_{quelle} nach c_{senke} verfolgt werden. Von jedem der N Knoten c_i hat man $N - i$ mögliche beginnende Wege – bzw. Pfade $P(i, N)$ – um c_{senke} zu erreichen. Die Länge dieser Pfade läßt sich mit

$$L_{P_i}(N) := \quad L(P(i,N)) = \quad \sum_{k=0}^{N-2} (N-i) - k \tag{4.2}$$

berechnen. Für die Gesamtrechenzeit $T(N)$ muß man die Pfadlängen aufaddieren:

$$\begin{aligned} T(N) &= \sum_{i=1}^{N-1} L_{P_{i(N)}} = \sum_{i=1}^{N-1} \sum_{k=0}^{N-2} (N-i) - k = N^2 - 2N + 1 \\ \Rightarrow T(N) &= O(N^2) \end{aligned} \tag{4.3}$$

Beim Greedy-Algorithmus existieren zwar auch $N - i$ mögliche Pfade von jedem Knoten c_i, von denen jedoch nur ein Teil betrachtet wird, da sich die weitere Richtung nach jedem Schritt neu entscheidet. Im ungünstigsten Fall ergibt sich

$$\begin{aligned} T(N) &= \sum_{i=1}^{N-1} N - i = \frac{N^2 - N}{2} \\ \Rightarrow T(N) &= O(N^2), \end{aligned} \tag{4.4}$$

aber im besten Fall nur $T(N) = N - 1$ und im Durchschnitt ergibt sich die Größenordnung $T(N) = O(N)$. Dadurch ist der Greedy-Algorithmus, auch wenn er nicht immer die optimale Lösung findet, schneller als Algorithmen, die immer nach der optimalen Lösung suchen. Der benötigte Speicherplatz ist $2N$, da für jede Stützstelle der Wert und der zugehörige Funktionswert gespeichert werden müssen.

Komplexität der Spline-Interpolation

Bei der Spline-Interpolation wird neben den N Werten und den zugehörigen Funktionswerten auch die zweite Ableitung benötigt. D.h., es wird zusätzlicher Speicherplatz ($3N$) benötigt oder die zweiten Ableitungen müssen jedes mal neu berechnet werden, was eine Erhöhung der Rechenzeit um N entspricht. Die Berechnung der tridiagonalen Matrix läßt sich in $O(N)$ durchführen. Die Spline-Funktion ist auch in dieser Zeit berechenbar. In der Summe erhält ergibt sich eine Rechenzeit von $T(N) = O(N)$, die mit der des Greedy-Algorithmus vergleichbar ist. Der bedeutende Unterschied ist, daß der TMA nur einmal angewendet werden muß um die Stützstellen zu reduzieren; dann wird mit der reduzierten Tabelle gearbeitet. Die Spline-Interpolation muß jedesmal durchgeführt werden, wenn ein Funktionswert interpoliert werden muß, der nicht in der Tabelle steht.

4.1.3 Parametereinstellungen

Die Reduktion der Stützstellen, die mit dem TMA erzielt wird, ist stark von den beiden Parametern ϵ und R abhängig. Um gute Ergebnisse zu erzielen müssen sie auf einander abgestimmt werden. Um die Abhängigkeit der Reduzierung von ϵ und R zu verdeutlichen wurde als Beispiel das Polynom

$$y(x) \quad = \quad 1.0x + 50.0x^2 - 17.0x^3 \tag{4.5}$$

untersucht. Ausgangspunkt waren 100 äquidistante Stützstellen im Intervall $[0, 5]$. Die Resul-

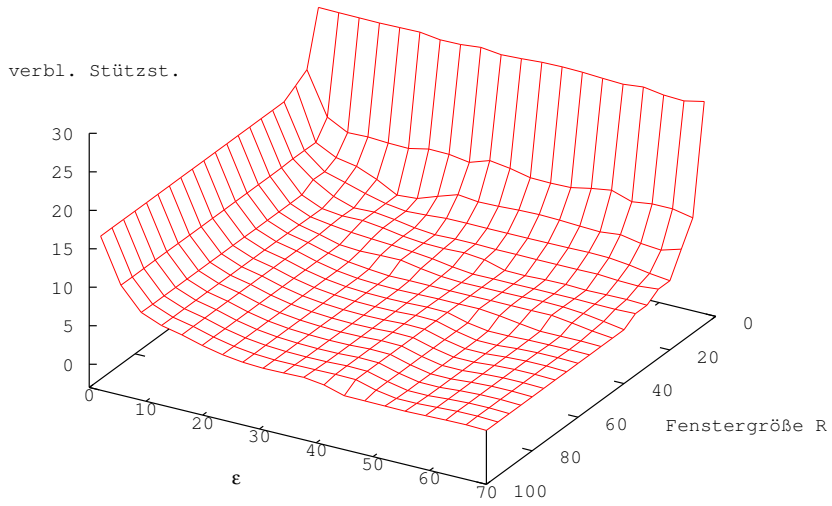

Abbildung 4.2: Verbleibende Stützstellen in Abhängigkeit von ϵ und R

tate sind in Abb. 4.2 dargestellt. Bei großem ϵ ist die Anzahl der verbleibenden Stützstellen proportional zu R, da alle Stützstellen dazwischen nicht berücksichtigt werden und äquidistante Stützstellen im Abstand R zurückbleiben. Für praktische Anwendungen wird eher ein kleines ϵ gefordert. In diesem Fall werden tatsächlich **nicht** äquidistante Stützstellen erreicht und das optimale R muß mittels Optimierung bestimmt werden. Ein typisches Resultat des TMA mit $\epsilon = 20$ und $R = 50$ ist in Abb. 4.3 zu sehen. Auf der linken Seite sind durch lineare Interpolation verbundene Stützstellen und die Funktion aus Gleichung (4.5) zu sehen. Rechts sind die korrespondierenden Fehlerwerte $e = |y(k) - g(k)|$ für jede der ursrünglichen Stützstellen angegeben. Der absolute Fehler $e =$ wächst zwischen den übrig gebliebenen Funktionswerten $y(x)$ bis zum maximal zulässigen Wert ϵ und fällt dann wieder. In der Nähe von $x = 0.5$ wird der Wert $e = 0.0$, obwohl hier keine Stützstelle vorhanden ist. Dies läßt sich darauf zurückführen, daß die approximierende Gerade die eigentliche Funktion kreuzt (siehe rechteckige Markierung links in Abb. 4.3).

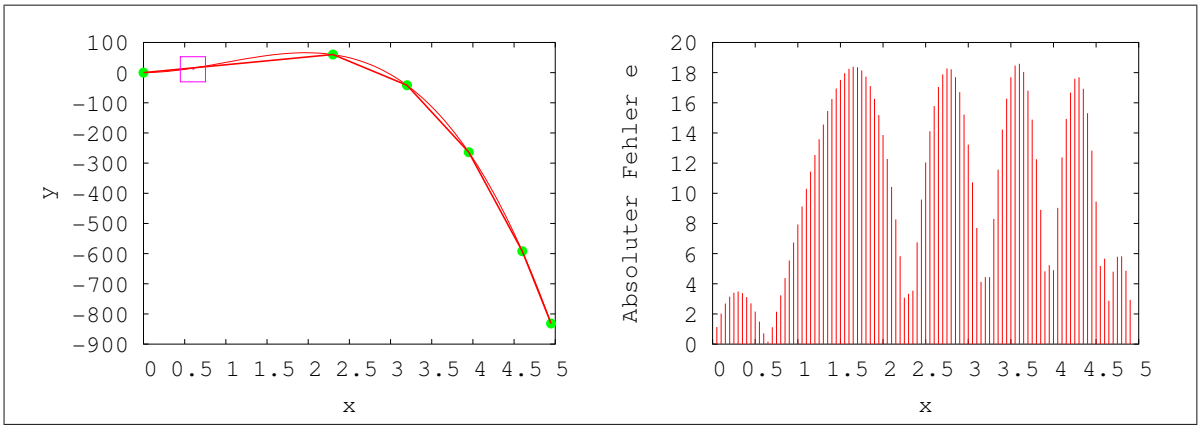

Abbildung 4.3: Ergebnis des TMA für Gl. (4.5) (linke Seite); Approximationsfehler e gegen x (rechte Seite)

4.2 Lernverfahren

Die Idee des Gelernte Interpolation (GLI) Verfahrens ist, mit wenig Stützstellen (mindestens zwei) zu beginnen und so lange es nötig ist Stützstellen einzufügen. Beim Einfügen soll auch die optimale Position der Stützstelle bestimmt werden.
Als Fehlermaß wird beim GLI Verfahren die Ausgabe des CNN mit der vorgegebenen Lösung verglichen. Es werden folgende Schritte nacheinander durchgeführt:

- Die beiden Ränder der Tabelle müssen festgelegt und die zugehörigen Funktionswerte bestimmte werden. Dadurch sind die ersten zwei Stützstellen festgelegt; sie sollten sich auch nicht mehr ändern.
- Die nächsten Schritte werden iterativ angewandt, bis das gewünschte Ergebnis erreicht ist:
  1. Es wird zwischen die vorhandenen Stützstellen (willkürlich) eine neue Stützstelle eingefügt.
  2. Die Stützstellen werden mit einem Training optimiert. Das heißt, ihre Position und ihr Funktionswert werden so angepaßt, daß das Verhalten des CNN optimal (der Fehler minimal) wird.

  Diese beiden Schritte werden so oft wiederholt, bis der entstehende Fehler, entsprechenden den Vorgaben, klein genug ist.

Teil III

Ergebnisse

Kapitel 5

Korteweg-de Vries-Gleichung

Die KdV-Gleichung wird als Modell für verschiedene physikalische Fragestellungen verwendet. Hierbei sind vor allem die solitären und solitonen Lösungen von besonderem Interesse. Für die KdV-Gleichung existieren sowohl numerische als auch analytische Lösungen [13, 19]. Für zwei Solitonen ist eine analytische Lösung durch

$$f(x,t) = \frac{2\left(w_1^2E_1 + w_2^2E_2 + 2(w_2-w_1)^2E_1E_2 + \frac{(w_2-w_1)^2}{(w_1+w_2)^2}\left(w_2^2E_1 + w_1^2E_2\right)E_1E_2\right)}{1 + E_1 + E_2 + \frac{(w_2-w_1)^2}{(w_1+w_2)^2}E_1E_2} \quad (5.1)$$

mit $E_1 = e^{w_1(x-x_{01}-w_1^2t)}$ und $E_2 = e^{w_2(x-x_{02}-w_2^2t)}$ gegeben. Mit den Parameter $x_{01} = 10$,

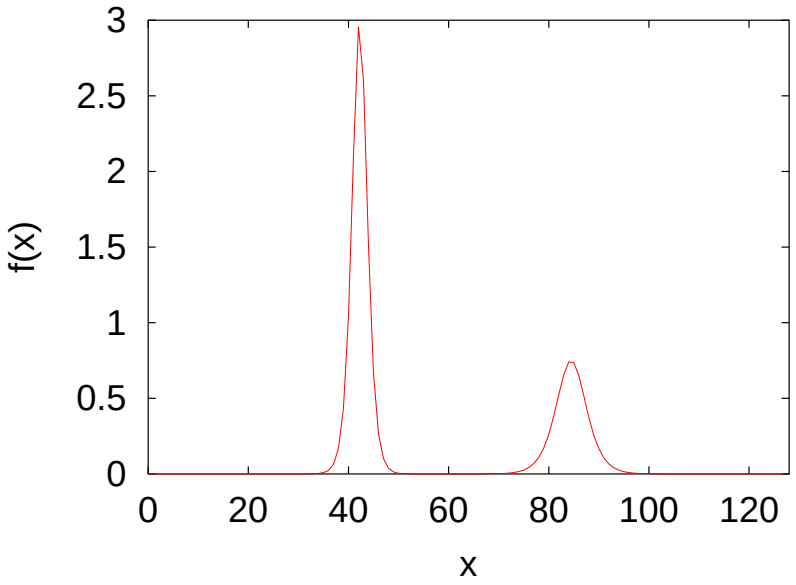

Abbildung 5.1: Lösung von (5.1) an der Stelle $t = 0$

$x_{02} = 20$, $w_1^2/2 = 2$ und $w_2^2/2 = 1$ ergibt sich die in Abb. 5.1 gezeigte Zwei-Solitonenlösung. Zur

nummerischen Lösung der KdV-Gleichung wird in den folgenden Simulationen diese als Anfangsbedingung, die in ein eindimensionales CNN mit 128 Zellen geladen wird, verwendet. Weiterhin kommt das Euler-Verfahren als numerische Integrationsmethode mit einer Integrationsschrittweite von $\Delta t = 0,0001$ zur Anwendung.
Da CNN zeitkontinuierlich und räumlich diskret sind, ist eine räumliche Diskretisierung der KdV-Gleichung notwendig. Hierfür wird die nach der im Abschnitt 3.1 beschriebenen Methode zur Ermittelung der in Tabelle 5.1 gezeigten Template angewandt. Zur Modellierung des Ver-

$1.0s$	$-2.0s + 0.5s^2$	0.0	$2.0s - 0.5s^2$	$-1.0s$

Tabelle 5.1: KdV Template entsprechend zu (3.4)

haltens der KdV, wurde die oben gezeigte Anfangsbedingung in ein autonomes eindimensionales CNN mit 128 Zellen geladen und mit der Template Tab. 5.1 berechnet. Das Ergebnis ist in Abbildung 5.2 zu sehen. Die beiden solitone Wellen, die durch die Zustände s der Zellen c an der

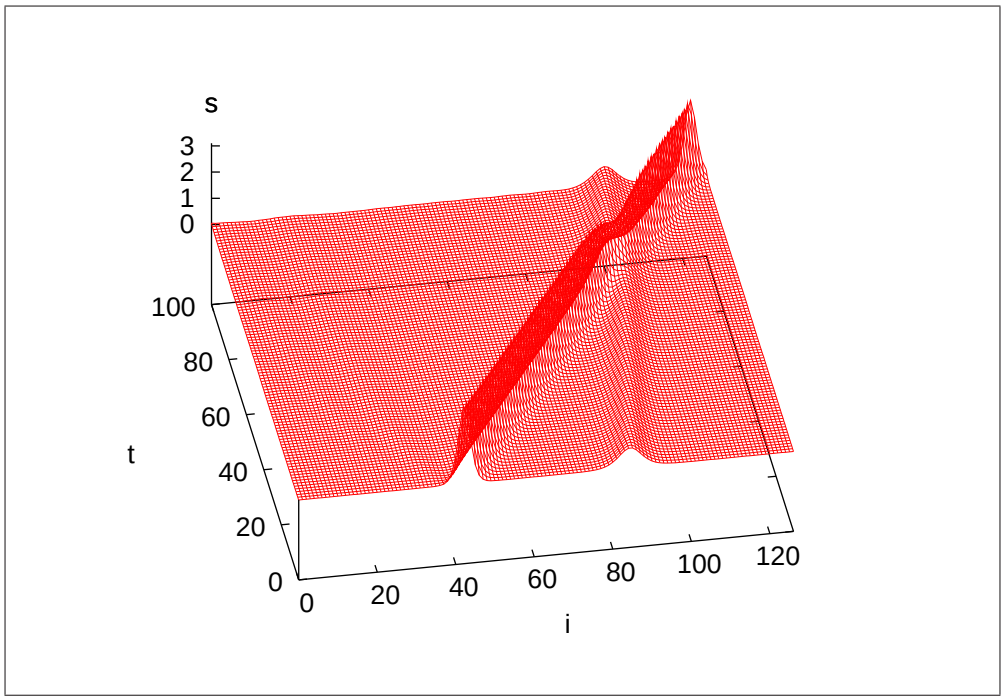

Abbildung 5.2: Lösung der KdV mit polynomialen Gewichtsfunktionen entsprechend zu (3.3)

Position i repräsentiert werden, pflanzen sich mit der Zeit t fort. Nach der Kollision bewegen sie sich mit unveränderter Amplitude und Geschwindigkeit weiter. Das Resultat dieser Lösung der KdV-Gleichung mit CNN's mit polynomialen Kopplungsfunktionen dient im Folgenden als Referenz zum Vergleich der Lösungen mit tabellierten Kopplungsfunktionen. Im folgenden wird der Zustand der Zellen in dieser Referenzösung als s_{ref} bezeichnet.
Die Kopplungsfunktionen lassen sich unter Verwendung linearer Interpolation mittels Tabellen repräsentieren. Die Genauigkeit dieser Darstellung ist hierbei ausschließlich durch die Anzahl der Tabelleneinträge in einem betrachteten Funktionswertebereich gegeben. Die in der Template Tab. 5.1 gegebenen polynomialen Gewichtsfunktionen werden jeweils mit 10000 äquidistanten

Stützstellen im Intervall $[-50, 50]$ tabelliert. Das so gewonnene Ergebnis ist in Abb. 5.3 oben

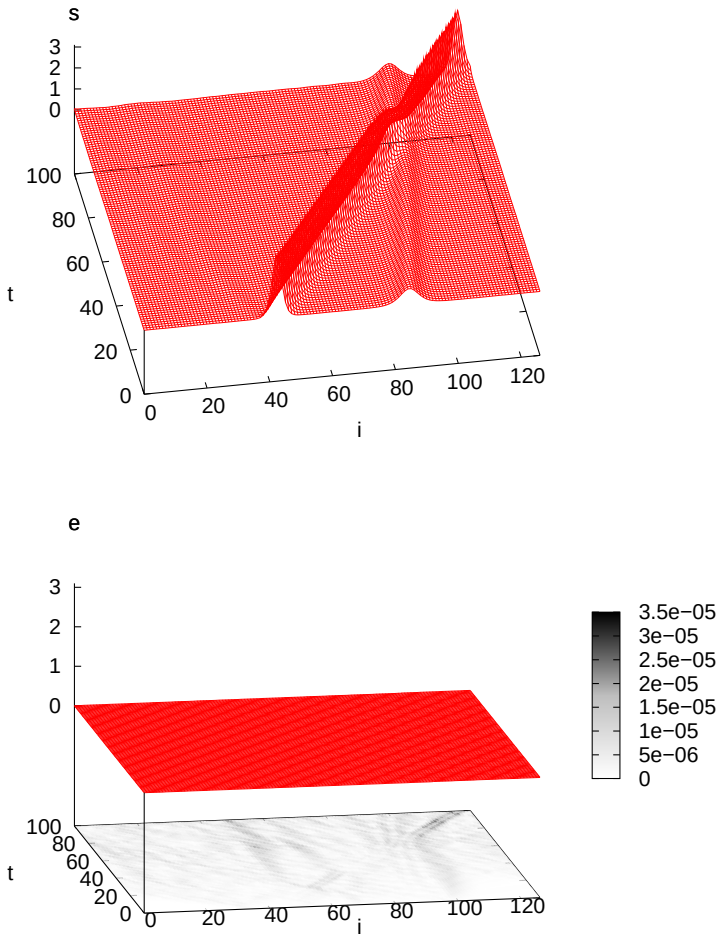

Abbildung 5.3: oben: Lösung der KdV mit tabellierten Gewichtsfunktionen mit äquidistanten Stützstellen; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 5.2.

und die Abweichung $e = |s - s_{ref}|$ zur der mit polynomialen Gewichtsfunktionen unten zu sehen. Wegen der hohen Anzahl an Stützstellen ist e in der Größenordnung von 10^{-5}.
Durch Anwendung des TMA läßt sich die Anzahl der tabellierten Stützstellen der Gewichtsfunktionen unter Berücksichtigung eines akzeptablen Fehlers reduzieren. Die Stützstellen in den resultierenden Tabellen sind nichtäquidistant. In den hier gezeigtem Fall wurden für die Parameter des TMA $\epsilon = 3.0$ und $R = 20$ gewählt, was eine Reduzierung der Stützstellen auf ein Fünftel bewirkt. In Abb. 5.4 ist oben die Lösung mit nichtäquidistant tabellierten Gewichtsfunktionen und unten die Abweichung $e = |s - s_{ref}|$ zur Referenz zu sehen.

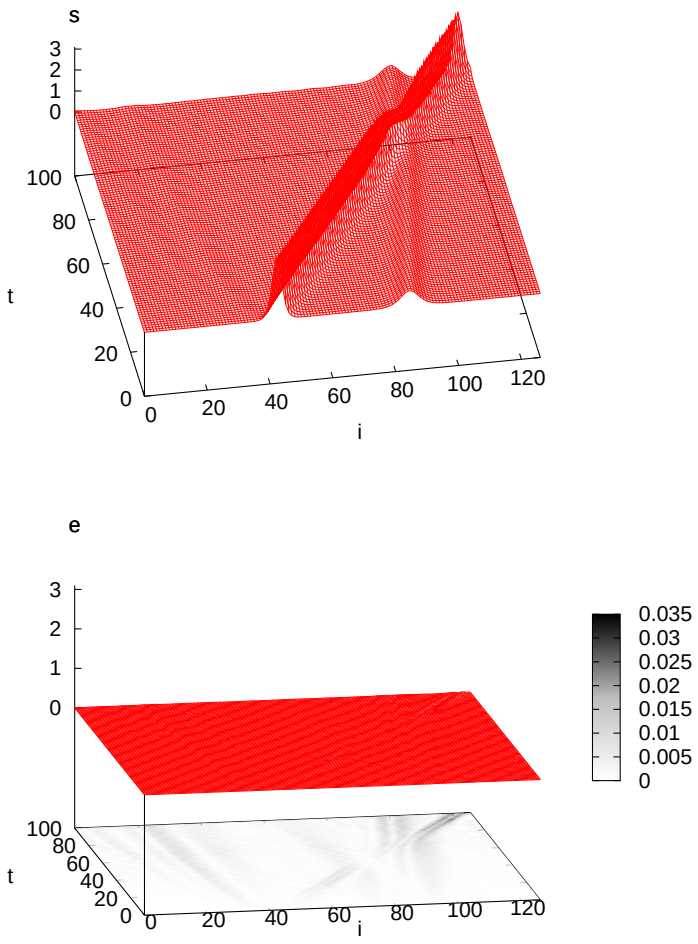

Abbildung 5.4: oben: Lösung der KdV mit tabellierten Gewichtsfunktion und Anwendung des TMA zu Reduzierung der Anzahl der Tabelleneinträge; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 5.2.

Kapitel 6

Burgers Gleichung

Eine analytische Lösung der Burgers Gleichung für zwei Solitonen ist durch

$$f(x,t) \quad = \quad \frac{w_1 \exp(-w_1(x - x_{01} - w_1 t)) + w_2 \exp(-w_2(x - x_{02} - w_2 t))}{1 + \exp(-w_1(x - x_{01} - w_1 t)) + \exp(-w_2(x - x_{02} - w_2 t))} \,. \tag{6.1}$$

gegeben. Die Parameter w_1, w_2 lassen sich als Geschwindigkeiten und x_{01}, x_{02} als Anfangspositionen der beiden solitären Wellen auffassen. Die Wellen verschmelzen zum Zeitpunkt

$$t \quad = \quad \frac{x_{02} - x_{01}}{w_2 - w_1}$$

zu einer einzigen solitären Welle. Mit $x_{01} = 25$, $x_{02} = 75$ und $w_1 = 2$, $w_2 = -1$ ergibt sich die in

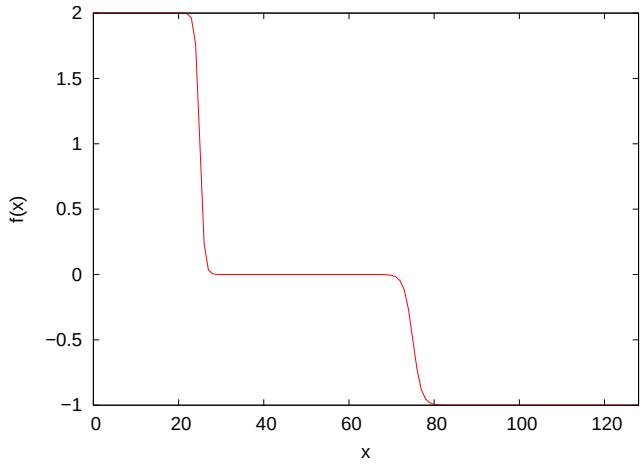

Abbildung 6.1: Lösung von (6.1) an der Stelle $t = 0$

Abb. 6.1 gezeigte Lösung. Analog zu dem für die KdV beschriebenen Verfahren kann die Template Tab. 6.1 mit in diesem Fall (3.7) für $h = 1$ erzeugt werden. Die Berechnung der in Abb.6.2 gezeigten Lösung erfolgte mit einem Runge-Kutta-Verfahren vierter Ordnung mit $\Delta t = 0,005$ mittels eines eindimensionalen autonomen CNN's mit 128 Zellen. Die beiden solitären Wellen, die sich mit der Zeit t ausbreiten, werden durch die Zustände s der Zellen an den Positionen

$1.0s + 0.5s^2$	$-2.0s$	$1.0s - 0.5s^2$

Tabelle 6.1: Template entsprechend disktretisierter Burgers Gleichung (3.7)

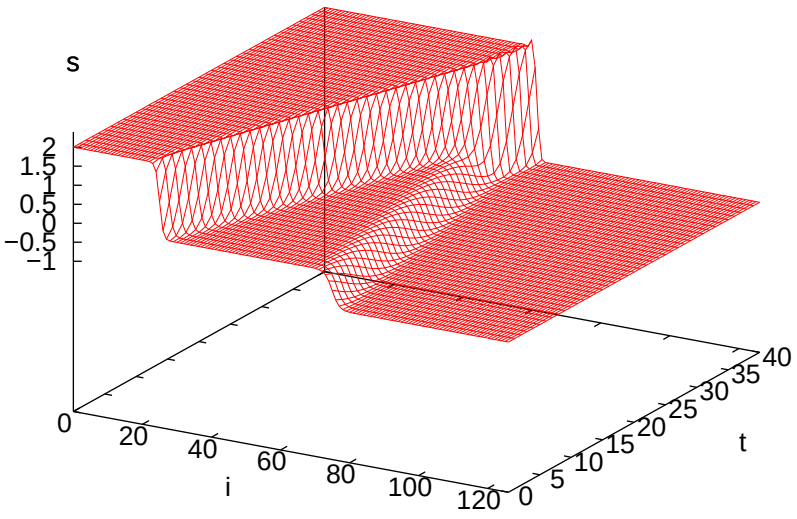

Abbildung 6.2: Lösung der Burgers Gleichung mit einem eindimensionalen autonomen CNN und der Template Tab. 6.1

i repräsentiert. Diese Lösung dient im folgendem immer als Referenz und die Zustände werden mit s_{ref} bezeichnet.
Die polynomialen Gewichtsfunktionen aus Tab. 6.1 lassen sich mit 650 äquidistanten Stützstellen darstellen. Das Resultat der Berechnung mit solchen Gewichtsfunktionen ist in Abb. 6.3 oben zu sehen. Darunter ist die Abweichung $e = |s - s_{ref}|$ zur Referenzlösung aus Abb. 6.2 zu sehen. Daß die Abweichung e an den Kanten der Wellenfronten am stärksten ausgeprägt ist, läßt sich mit kleine Abweichungen der Geschwindigkeit der propagierenden Wellenfronten in zwei zu vergleichenden Fällen erklären. Insgesamt bleibt die Differenz e jedoch in einem akzeptablen Rahmen.

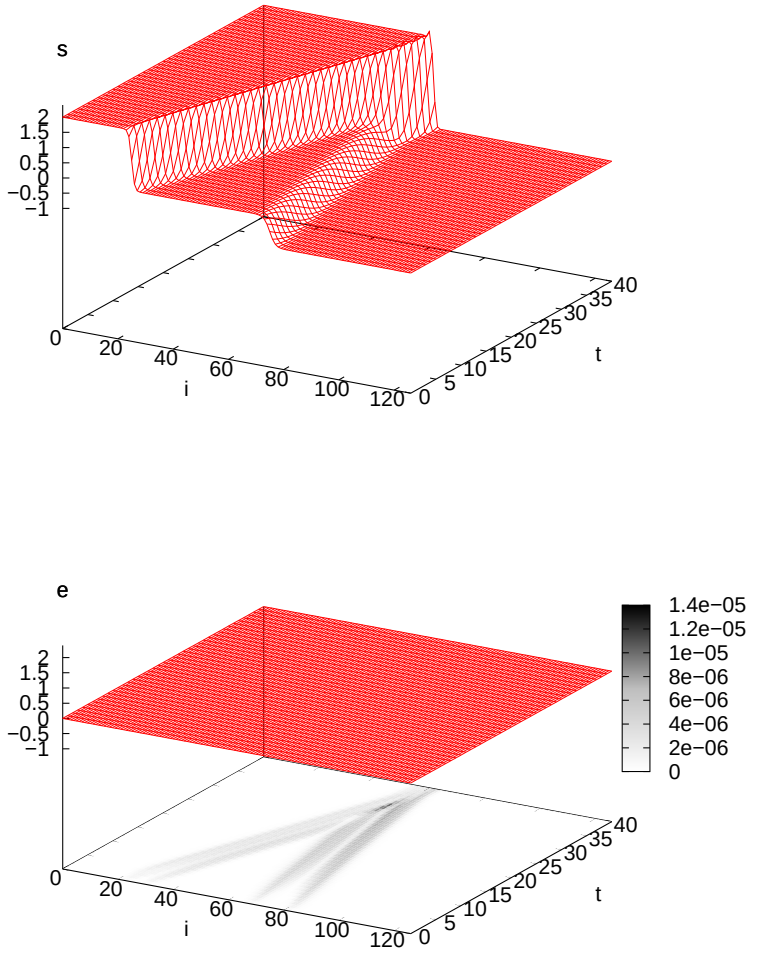

Abbildung 6.3: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen mit 650 äquidistanten Stützstellen; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

Durch die Anwendung des TMA läßt sich die Anzahl der Stützstellen bei vorgegebenem Approximationsfehler reduzieren. Der Grad der Reduzierung hängt von den beiden frei wählbaren Parametern des TMA, ϵ und R ab. Dabei korrespondiert ϵ mit dem vorgegebenen Approximationsfehler und wird möglichst klein gewählt. Die Fenstergröße R muß dem Anwendungsfall angemessen gewählt werden. Bei der Burges Gleichung ergibt sich für die nichtlineare Gewichts-

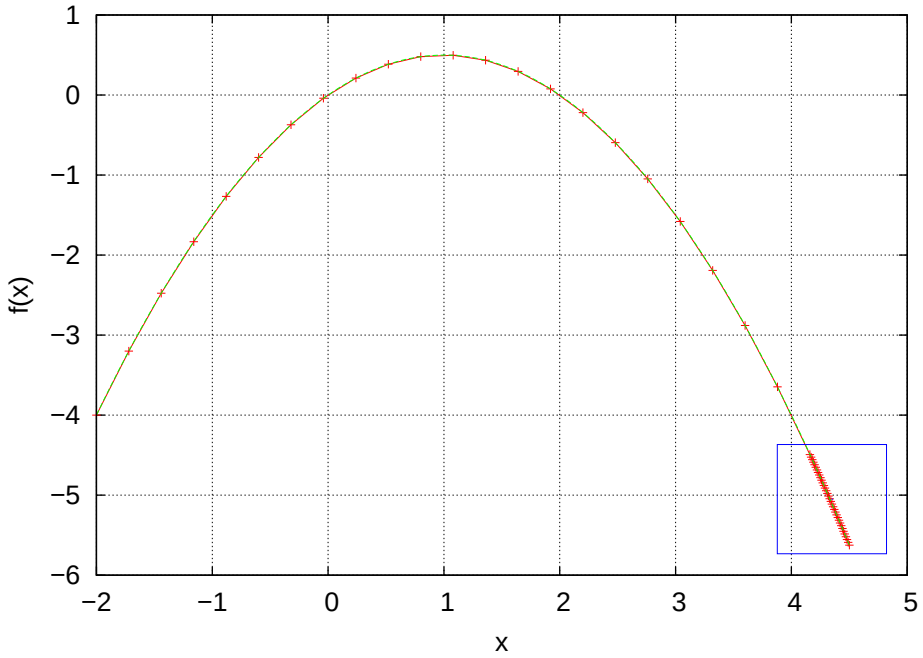

Abbildung 6.4: nichtlineare Gewichtsfunktion $f(x) = x - 0.5x^2$, nichtäquidistante Stützstellen für den Fall $\epsilon = 0,01$ und $R = 50$ und im Rechteck Dichte der äquidistanten Stützstellen

funktion $f(x) = x - 0.5x^2$ der in Abb. 6.4 gezeigte Funktionsverlauf. Weiterhin sind die nichtäquidistanten Stützstellen für den Fall $\epsilon = 0,01$ und $R = 50$ eingezeichnet. Zum Vergleich zu dem Fall mit äquidistanten Stützstellen ist im rechts hervorgehobenen Ausschnitt die Dichte der äquidistanten Stützstellen gezeigt. Es ist zu erkennen, daß die nichtäquidistanten Stützstellen um den Scheitelpunkt der Funktion, aufgrund der stärkeren Krümmung, dichter liegen. In Abb. 6.5 oben ist die Lösung der Burgers Gleichung mit nichtäquidistanten Stützstellen nach Anwendung des TMA mit $\epsilon = 0,1$ und $R = 20$ in den Gewichtsfunktionen gezeigt. Wie im Fall mit äquidistanten Stützstellen ist auch hier im darunter dargestellten Fehlerbild ein Unterschied in der Geschwindigkeit der Wellenfronten zu erkennen.

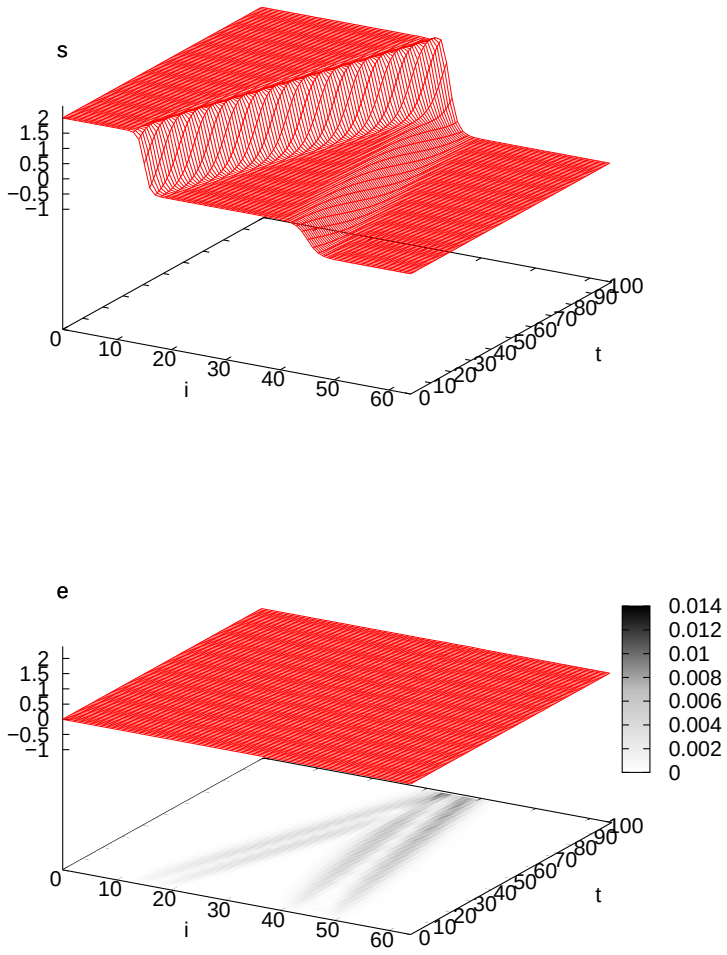

Abbildung 6.5: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,1$ und $R = 20$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

Weitere untersuchte Fälle sind in Tabelle Tab. 7.1 zusammengefaßt. Hierbei wurde sowohl der Parameter ϵ, als auch der Parameter R systematisch variiert. Die in der Tabelle Tab. 7.1 genannten Parametereinstellungen führen zu den Lösung, die in den Abb. 6.5 bis Abb. 6.15 gezeigt sind. Die jeweils zugehörigen Fehler e sind darunter aufgeführt und zur Verdeutlichung durch Grauwerte codiert dargestellt. Auch hier zeigt sich ein geringer Unterschied in der Ausbreitungsgeschwindigkeiten der Wellenfronten, der mit kleinerem ϵ ebenfalls kleiner wird. Der im jeweiligen Fehlerbild angebende maximale Fehler entsteht an der Stelle, an der die Wellen zu einer einzigen solitären Welle verschmelzen. Der Einfluß des Parameters R auf die Ergebnisse variiert für unterschiedliche ϵ. In einigen Fällen führt eine Vergrößerung von R nicht zu einer stärkeren

ϵ	R	Reduktion der Stützstellen in %	dargestellt in Abb.
$0,1$	20	≈ 94	6.5
$0,1$	50	≈ 97	6.6
$0,01$	20	≈ 94	6.7
$0,01$	50	≈ 93	6.8
$0,01$	100	≈ 88	6.9
$0,01$	200	≈ 80	6.10
$0,005$	20	≈ 95	6.11
$0,001$	20	≈ 89	6.12
$0,001$	50	≈ 87	6.13
$0,0001$	20	≈ 68	6.14
$0,0001$	50	≈ 67	6.15

Tabelle 6.2: Parameterkombinationen, mit denen der TMA angewandt wurde

Reduzierung der Anzahl der Stützstellen. Dieses Verhalten ist jedoch nicht systematisch, wie in Tab 7.1 für ein $\epsilon = 0,1$ gezeigt werden konnte.

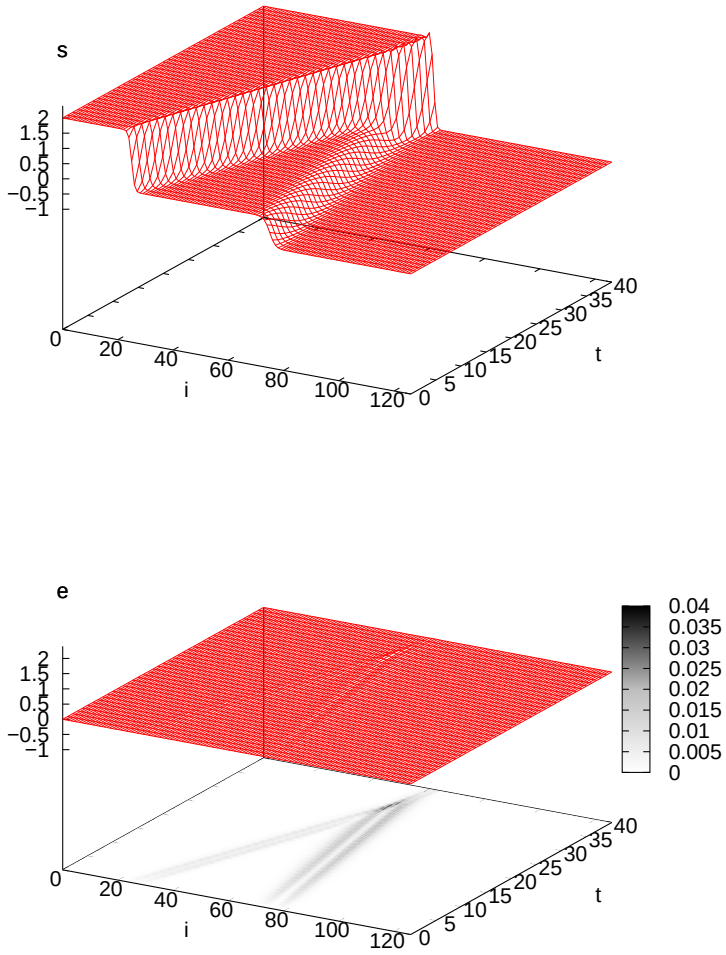

Abbildung 6.6: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,1$ und $R = 50$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

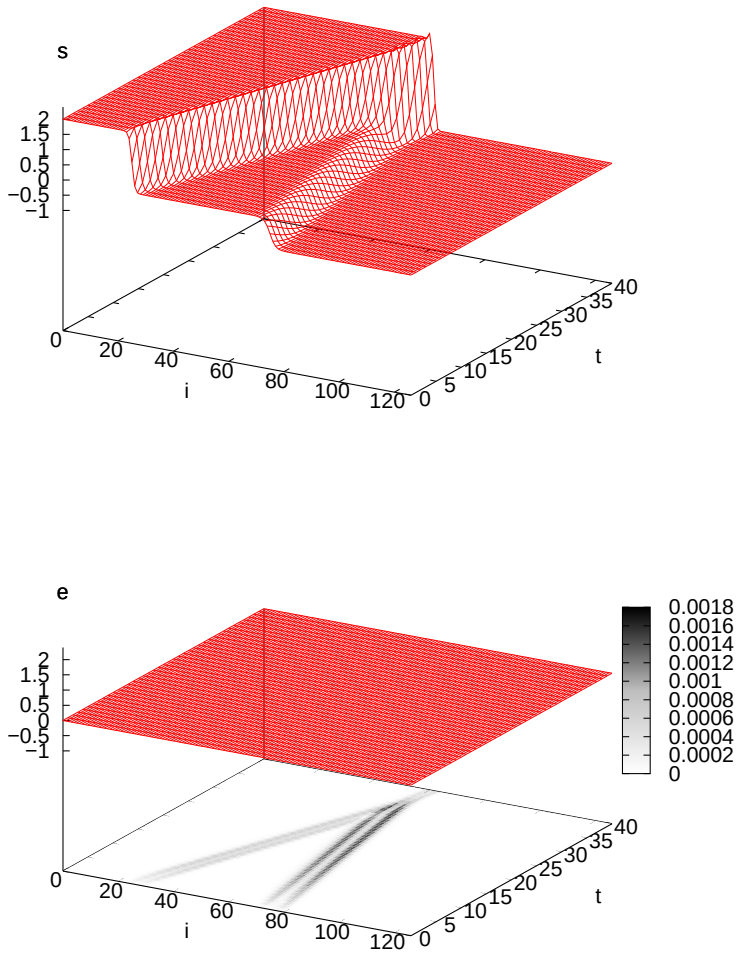

Abbildung 6.7: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,01$ und $R = 20$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

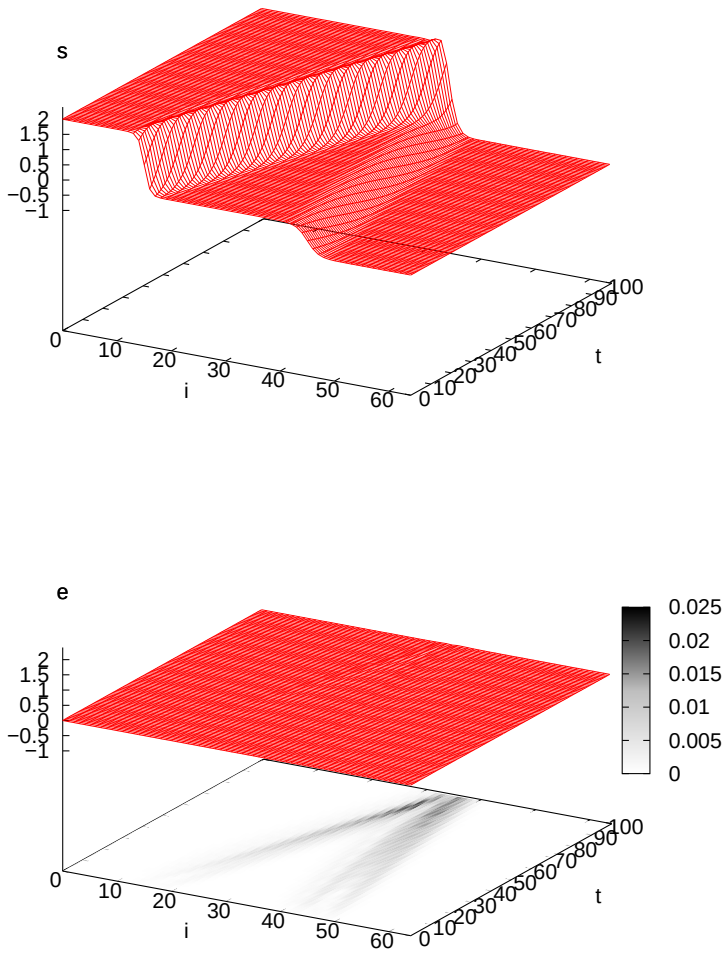

Abbildung 6.8: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,01$ und $R = 50$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

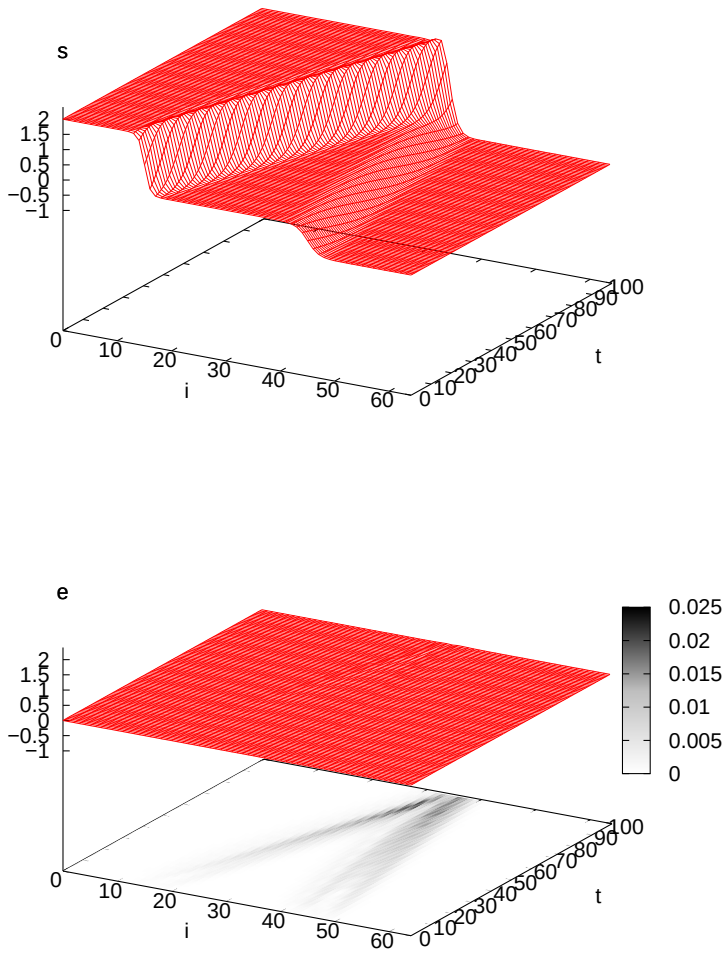

Abbildung 6.9: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,01$ und $R = 100$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

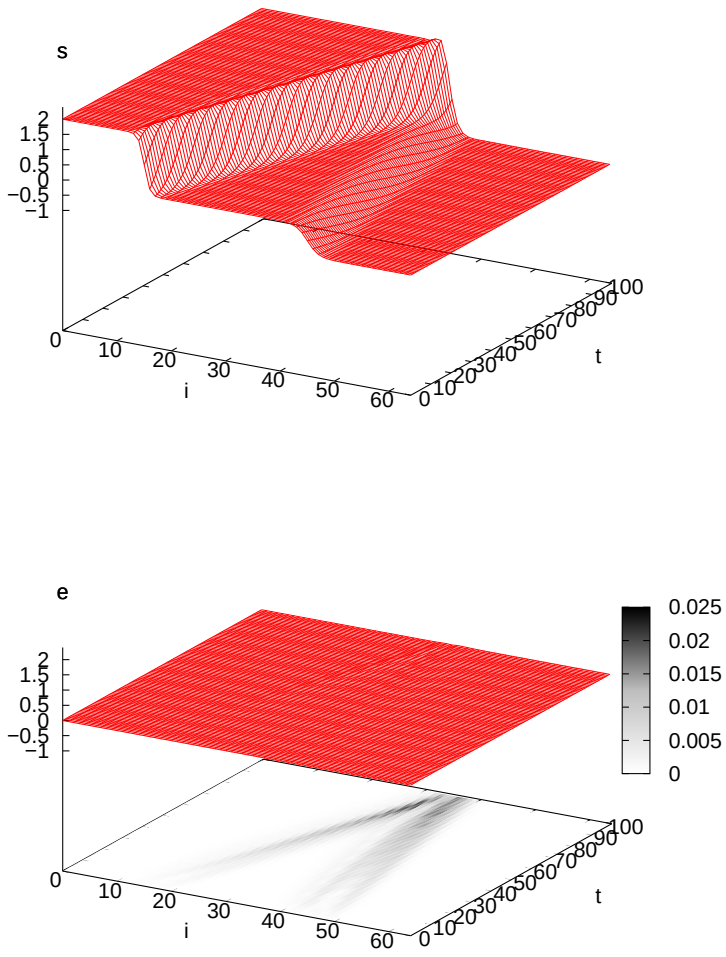

Abbildung 6.10: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,01$ und $R = 200$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

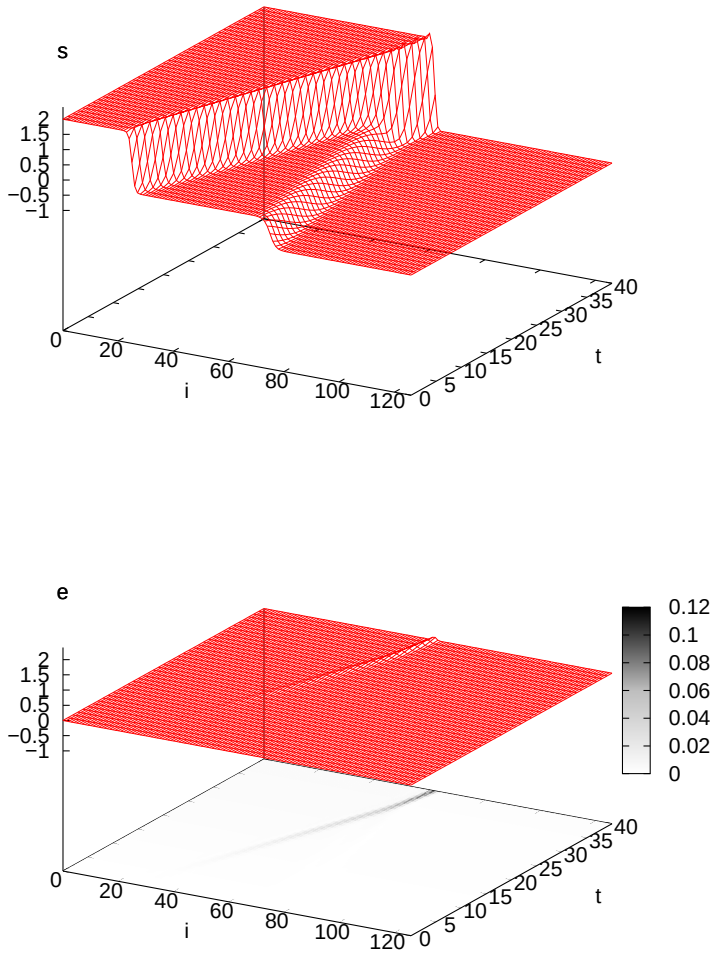

Abbildung 6.11: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,005$ und $R = 20$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

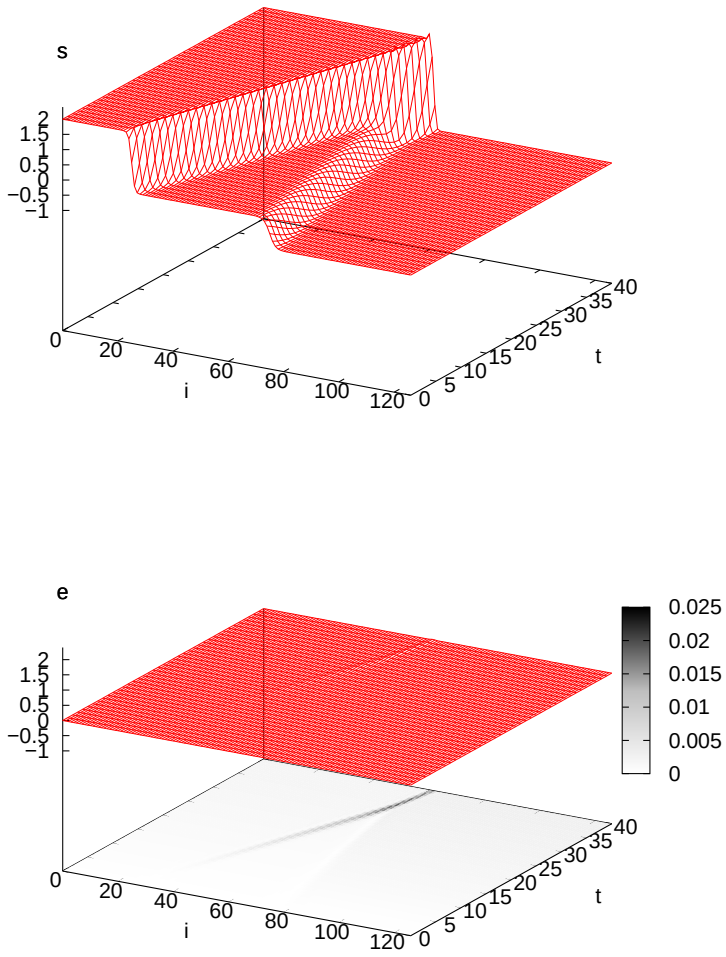

Abbildung 6.12: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,001$ und $R = 20$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

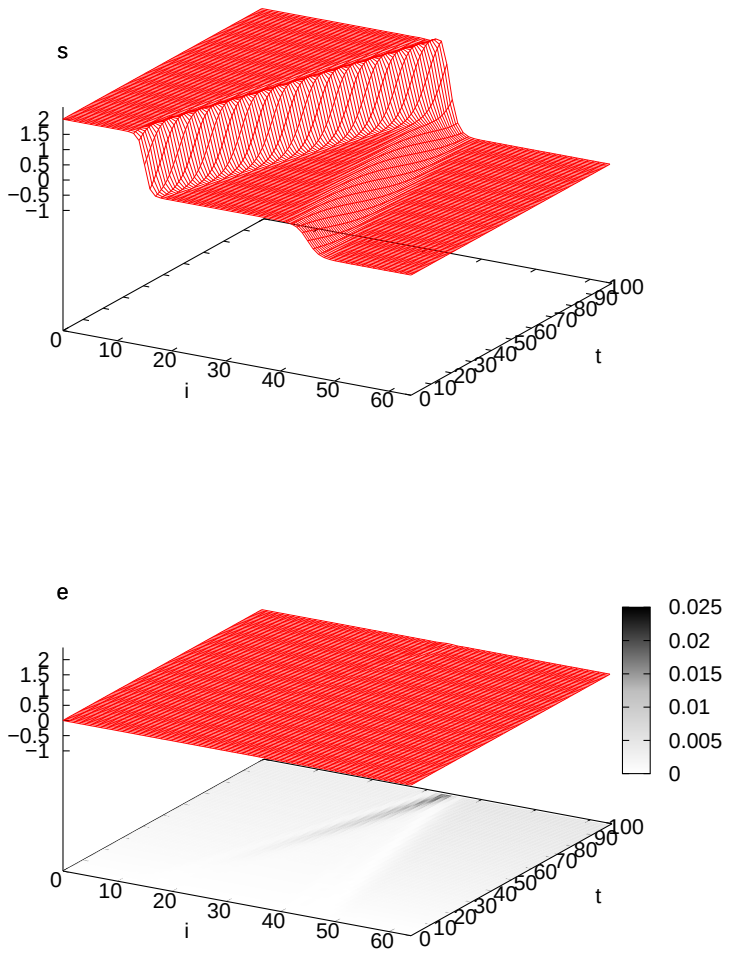

Abbildung 6.13: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,001$ und $R = 50$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

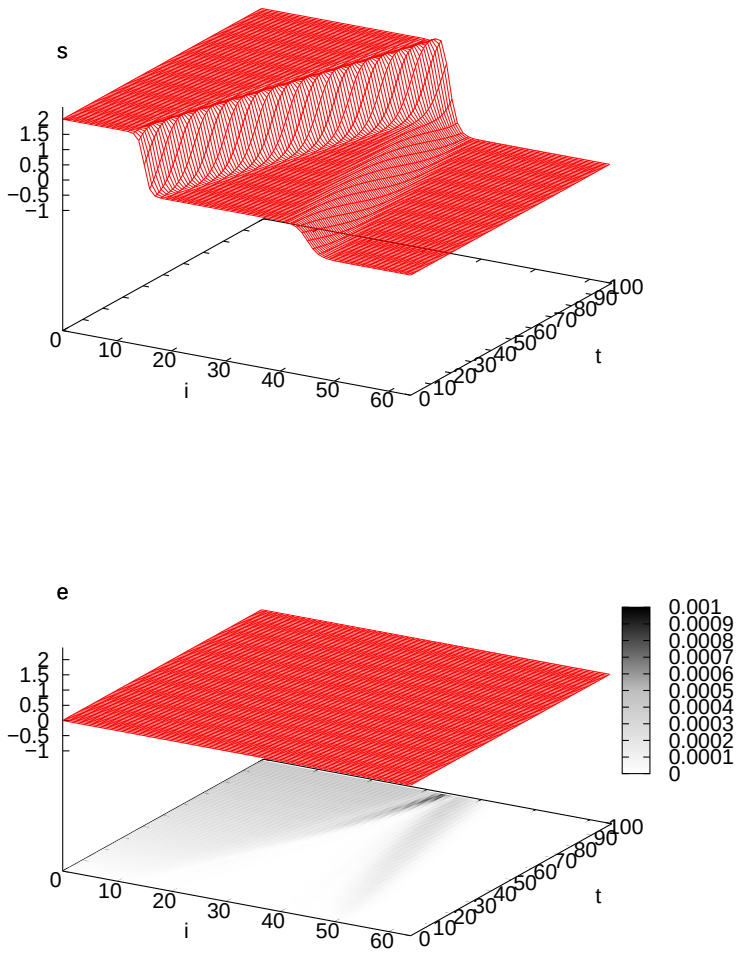

Abbildung 6.14: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,0001$ und $R = 20$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

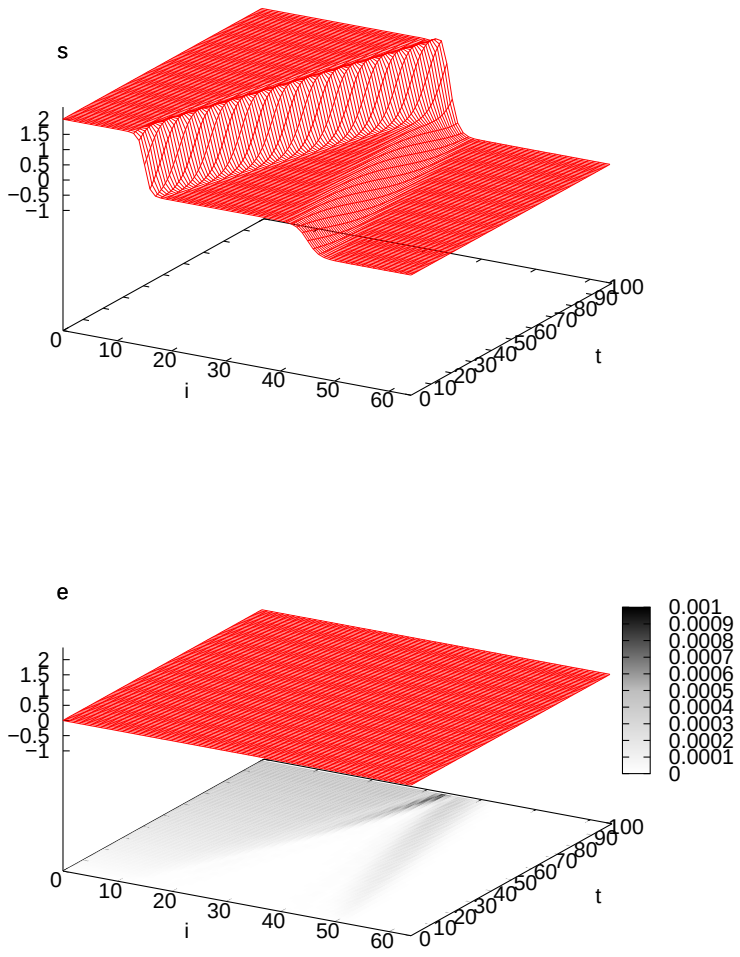

Abbildung 6.15: oben: Lösung der Burgers Gleichung mit tabellierten Gewichtsfunktionen nach Anwendung des TMA mit den Parametern $\epsilon = 0,0001$ und $R = 50$; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

6.1 Anwendung des GLI

Eine weitere Methode zur Erzeugung nichtäquidistanter Stützstellen tabellierter Gewichtsfunktionen ist das überwachte Parametertrainining. Hierbei wird im allgemeinen von einer Tabelle mit wenigen beliebig gewählten Stützstellen als Anfangsbedingung für den GLI-Algorithmus ausgegangen. Die Optimierung besteht darin die bereits vorhanden Stützstellen zu variieren und bei Bedarf neue hinzuzufügen. Dies wird solange wiederholt bis die gewünschte Approximationsgenauigkeit erreicht wird. In Abb. 6.16 oben ist die Lösung der Burgers Gleichung mit durch den

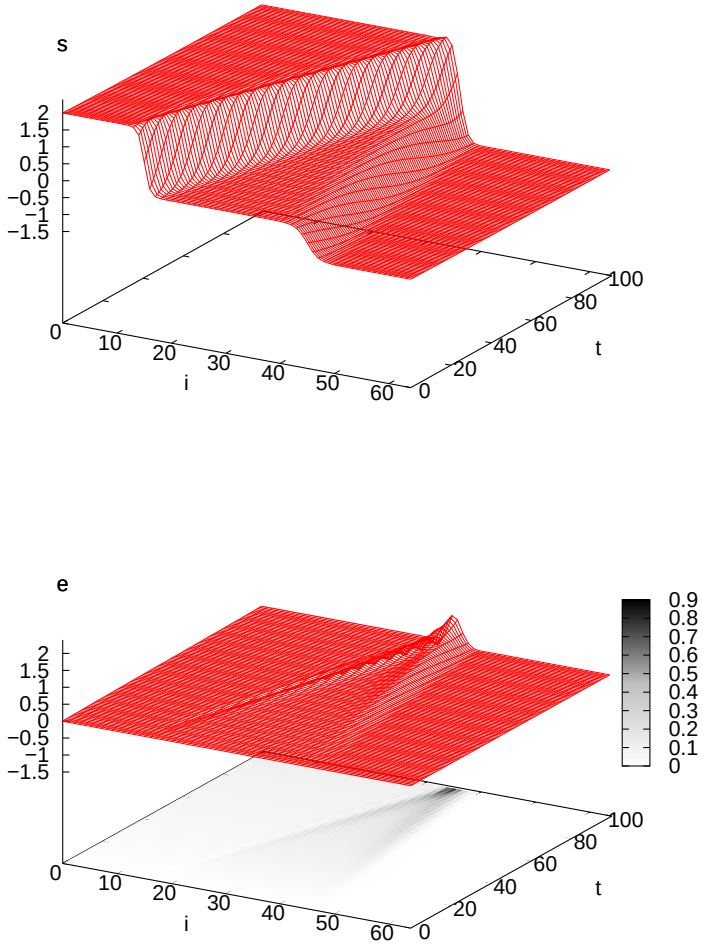

Abbildung 6.16: oben: Lösung der Burgers Gleichung mit durch GLI-Algorithmus gewonnenen tabellierte Gewichtsfunktionen mit nichtäquidistanten Stützstellen; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.2

GLI-Algorithmus gewonnenen mit nichtäquidistanten Stützstellen tabellierten Gewichtsfunktionen dargestellt. In Abb. 6.16 unten ist der zugehörige Fehler gezeigt. Dabei werden die solitären

x_{01}	x_{02}	w_1	w_2	dargestellt in Abb.	Referenz in Abb.
15	45	2.0	-1.0	6.16	6.2
20	40	1.0	-1.0	6.18	6.17
25	40	0.5	-0.75	6.20	6.19
15	30	2.0	0.5	6.22	6.21
25	35	1.0	0.5	6.24	6.23

Tabelle 6.3: Einstellungen für unterschiedliche Anfangsbedingungen

Wellen durch die Zustände s der Zellen mit dem Index i repräsentiert. Die Wellen propagieren mit der Zeit t. Auch hier entsteht durch unterschiedliche Geschwindigkeit dieser Lösung zur Referenz eine Abweichung e. Im weiteren wurden unterschiedliche Anfangsbedingung, deren Geschwindigkeiten w_1, bzw. w_2 und Anfangspositionen x_{01}, bzw. x_{02} in Tab. 6.3 angegeben sind, zur Untersuchung des GLI-Algorithmus herangezogen. Hierbei wurde jeweils ein Training der tabellierten Gewichtsfunktionen durchgeführt. Die Resultate und die Referenzlösungen sind in den Abb. 6.16 bis Abb. 6.24, wie in Tab. 6.3 aufgeführt, gezeigt.

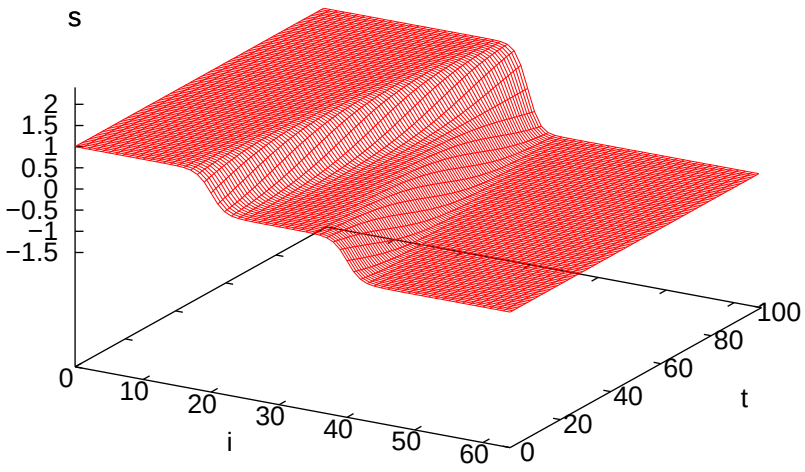

Abbildung 6.17: Lösung der Burgers Gleichung mit einem eindimensionalen autonomen CNN mit polynomialen Gewichtsfunktionen als Referenz

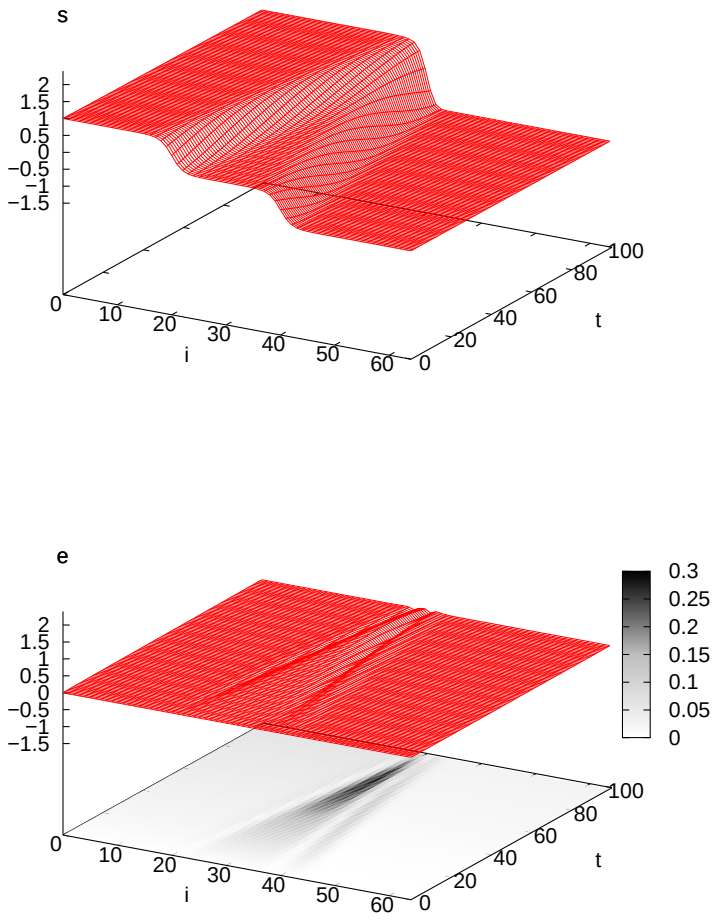

Abbildung 6.18: oben: Lösung der Burgers Gleichung mit durch GLI-Algorithmus gewonnenen tabellierten Gewichtsfunktionen mit nichtäquidistanten Stützstellen; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.17

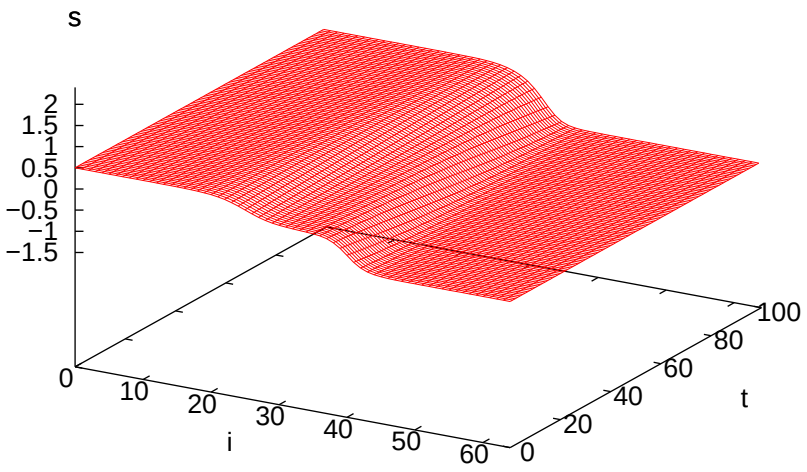

Abbildung 6.19: Lösung der Burgers Gleichung mit einem eindimensionalen autonomen CNN mit polynomialen Gewichtsfunktionen

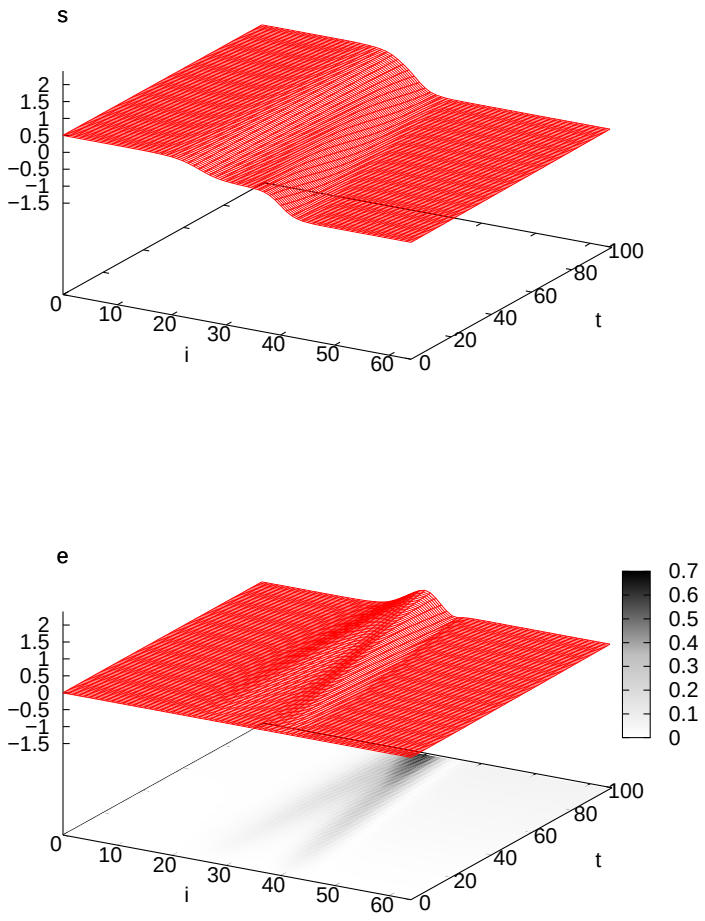

Abbildung 6.20: oben: Lösung der Burgers Gleichung mit durch GLI-Algorithmus gewonnenen tabellierten Gewichtsfunktionen mit nichtäquidistanten Stützstellen; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.19

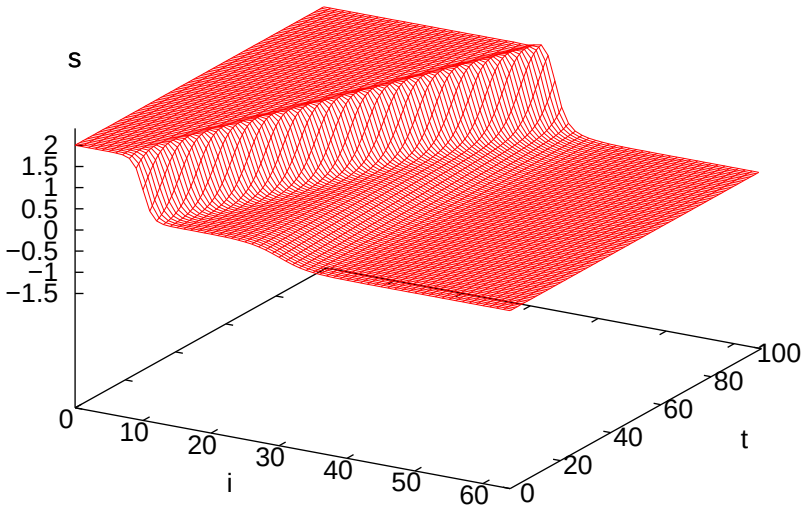

Abbildung 6.21: Lösung der Burgers Gleichung mit einem eindimensionalen autonomen CNN mit polynomialen Gewichtsfunktionen

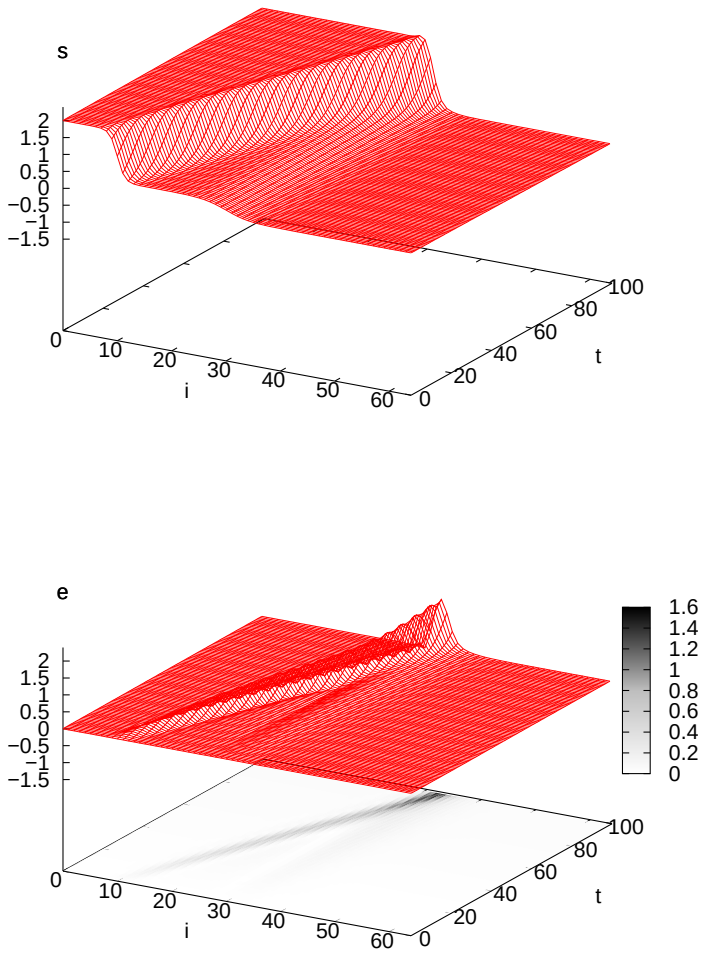

Abbildung 6.22: oben: Lösung der Burgers Gleichung mit durch GLI-Algorithmus gewonnenen tabellierten Gewichtsfunktionen mit nichtäquidistanten Stützstellen; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.21

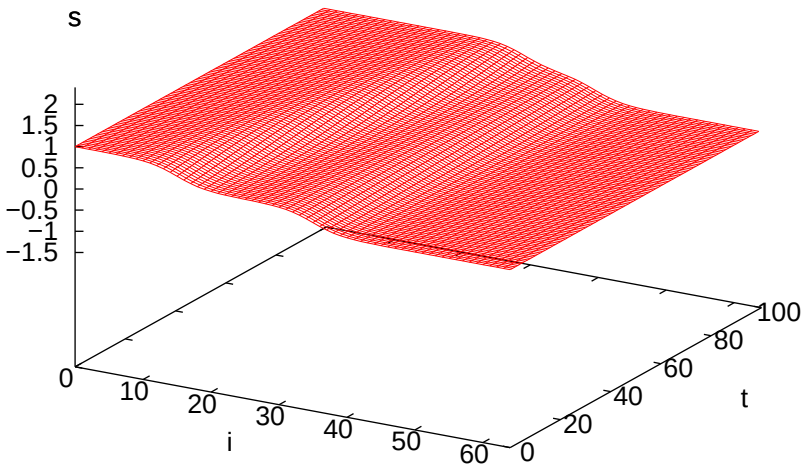

Abbildung 6.23: Lösung der Burgers Gleichung mit einem eindimensionalen autonomen CNN mit polynomialen Gewichtsfunktionen

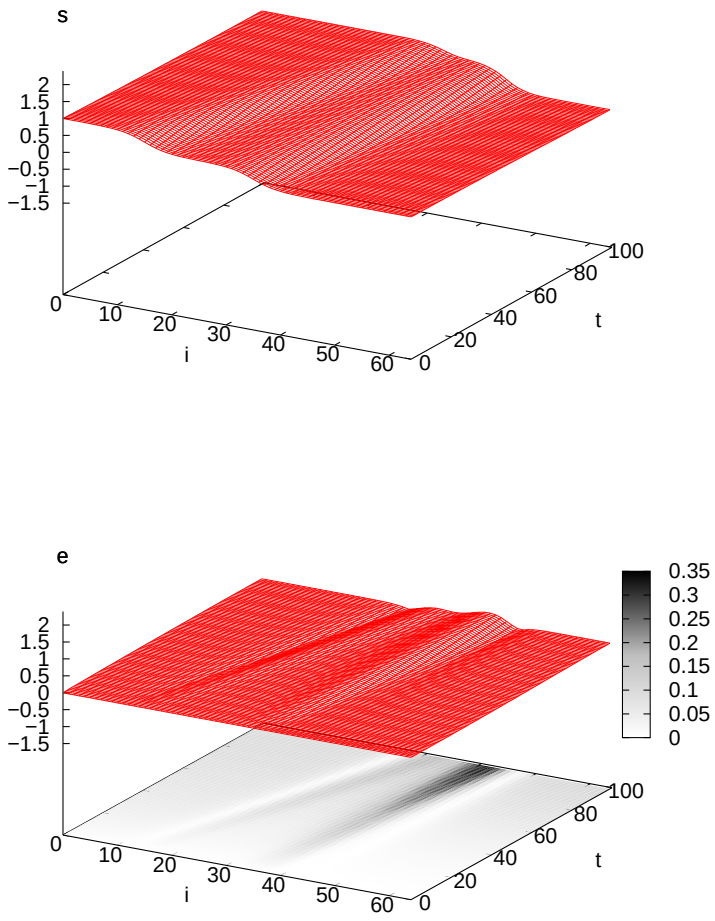

Abbildung 6.24: oben: Lösung der Burgers Gleichung mit durch GLI-Algorithmus gewonnenen tabellierten Gewichtsfunktionen mit nichtäquidistanten Stützstellen; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 6.23

Kapitel 7

Φ^4 Gleichung

Eine nichtlineare partielle Differentialgleichung, die zur Klasse der nichtlinearen Klein-Gordon-Gleichungen gehört, ist die Φ^4 Gleichung. Die *Kink-Antikink* Kollision ist eine solitäre Lösung der Φ^4 Gleichung, deren Anfangsbedingung eine Linearkombination der Kink-Wellenfunktion

$$f_+(x,t) = +\tanh\left(\frac{x - w_+ t - x_{0+}}{\sqrt{2(1-w_+^2)}}\right) \tag{7.1}$$

und der Antikink-Wellenfunktion

$$f_-(x,t) = -\tanh\left(\frac{x - w_- t - x_{0-}}{\sqrt{2(1-w_-^2)}}\right) \tag{7.2}$$

mit den Geschwindigkeiten w_+, w_- und den Anfangspositionen x_{0+}, x_{0-} ist. Die Lösungen lassen sich, sobald die Wechselwirkung zwischen Kink und Antikink einsetzen, nur noch numerisch berechnen. Im betrachteten Fall wird $x_{0+} = 24$, $x_{0-} = 40$ und $w_+ = 0.1$, $w_- = -0.25$ gewählt, so daß

$$f(x,0) = \begin{cases} f_+(x,0) & \text{für} \quad x < \frac{x_{0+}+x_{0-}}{2} \\ f_-(x,0) & \text{für} \quad x \geq \frac{x_{0+}+x_{0-}}{2} \end{cases} \tag{7.3}$$

erfüllt ist. Durch diese Wahl von x_{0+} und x_{0-} wird sichergestellt, daß für $t = 0$ an der Stelle $\frac{x_{0+}+x_{0-}}{2}$ die Funktionswerte identisch sind und die ersten Ortsableitungen praktisch gleich Null sind. Da die Φ^4 Gleichung von zweiter zeitlicher Ordnung ist, muß als Anfangsbedingung $\frac{\partial f(x,0)}{\partial t}$ angegeben werden. Mit

$$\frac{\partial f(x,0)}{\partial t} = \begin{cases} f'_+(x,0) & \text{für} \quad x < \frac{x_{0+}+x_{0-}}{2} \\ f'_-(x,0) & \text{für} \quad x \geq \frac{x_{0+}+x_{0-}}{2} \end{cases} \tag{7.4}$$

ergibt sich

$$f'_+(x,0) = \frac{w_+ \mathrm{sech}^2\left(\frac{-(x-x_{0+})}{\sqrt{2(1-w_+^2)}}\right)}{\sqrt{2(1-w_+^2)}}$$

und

$$f'_-(x,0) \quad = \quad \frac{w_- \mathrm{sech}^2\left(\frac{-(x-x_{0-})}{\sqrt{2(1-w_-^2)}}\right)}{\sqrt{2(1-w_-^2)}} \; .$$

Die Berechnung der Lösungen erfolgte mit einem Runge-Kutta-Verfahren vierter Ordnung mit der Integrationsschrittweite $\Delta t = 0,05$. In Abb. 7.1 ist die Lösung der Φ^4 Gleichung mit polynomialen Gewichtsfunktionen gezeigt, die im folgenden als Referenz dient. Es ist zu erkennen,

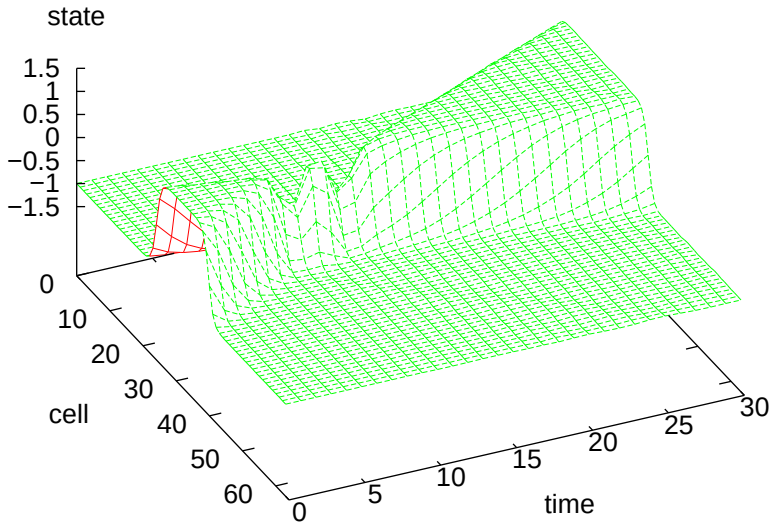

Abbildung 7.1: Lösung der Φ^4 Gleichung mit polynomialen Gewichtsfunktionen

daß während der Wechselwirkung ein Schwingen stattfindet. Hierbei entstehen zwei Wellentäler und ein Wellenberg in der zeitlichen Entwicklung t. Anschließend breiten sich die Kink- und Antikink-Wellen in entgegengesetzte Richtung aus.

Um eine ausreichend genaue Approximation der mit polynomiellen CNN erhaltenen Lösung zu erreichen wurde die lineare Interpolation mit 1000 äquidistanten Stützstellen angewendet. Das

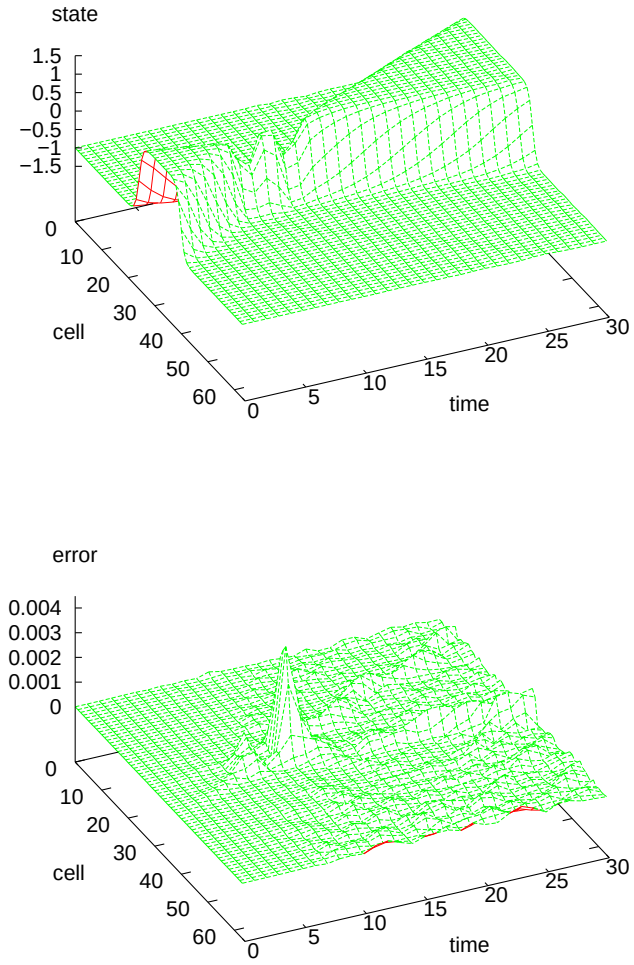

Abbildung 7.2: oben: Lösung der Φ^4 Gleichung mit tabellierten Gewichtsfunktionen mit 1000 äquidistanten Stützstellen; unten: Abweichung dieser zur polynomiellen Lösung aus Abb. 7.1

Resultat und das Fehlerbild sind in Abb. 7.2 zu sehen. Im Fehlerbild ist deutlich eine Erhöhung zu erkennen, die daher resultiert, daß der Wellenberg in t der Lösung mit tabellierten Gewichtsfunktionen mit dem Wellental in t der Referenzlösung zusammenfällt. Im weiteren zeitlichen Verlauf sind die Abweichungen e zwischen dem Fall der tabellierten- und der polynomialen Gewichtsfunktionen nicht so ausgeprägt.

Durch die Anwendung des TMA läßt sich die Anzahl der Stützstellen bei vorgegebenem Approximationsfehler reduzieren. Der Grad der Reduzierung hängt von den beiden frei wählbaren Parametern des TMA, ϵ und R ab. In Abb. 7.3 sind die erhaltenen Ergebnisse dargestellt.

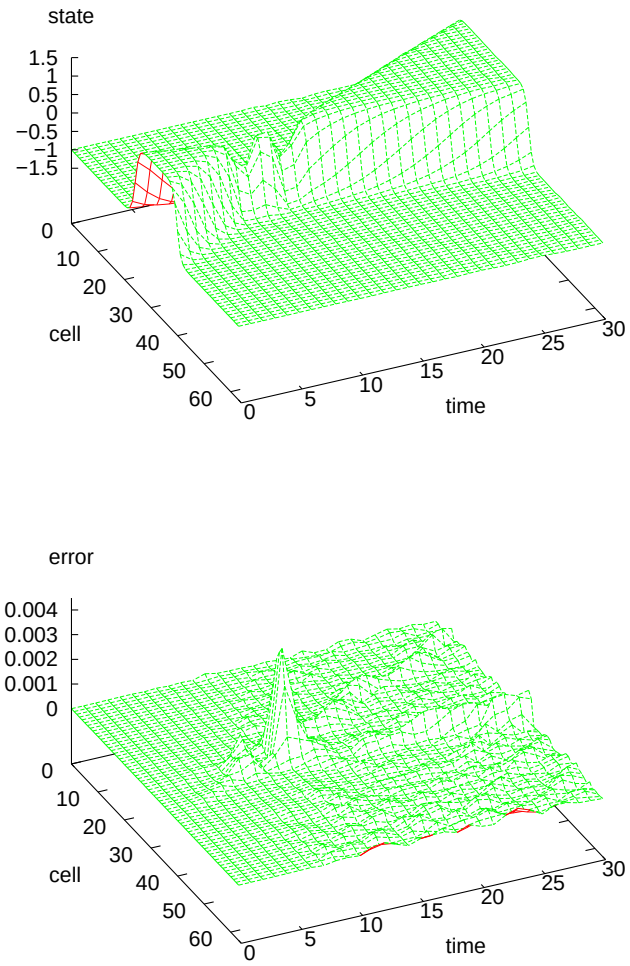

Abbildung 7.3: oben: Lösung der Φ^4 Gleichung tabellierten Gewichtsfunktionen nach Anwendung des TMA; unten: Abweichung dieser zur polynomialen Lösung aus Abb. 7.1

ϵ	R	Reduktion der Stützstellen in %	dargestellt in Abb.
$0,000001$	20	≈ 23	7.3
$0,00001$	20	≈ 30	7.4
$0,0001$	20	≈ 86	7.5

Tabelle 7.1: Parameterkombinationen, mit denen der TMA angewandt wurde

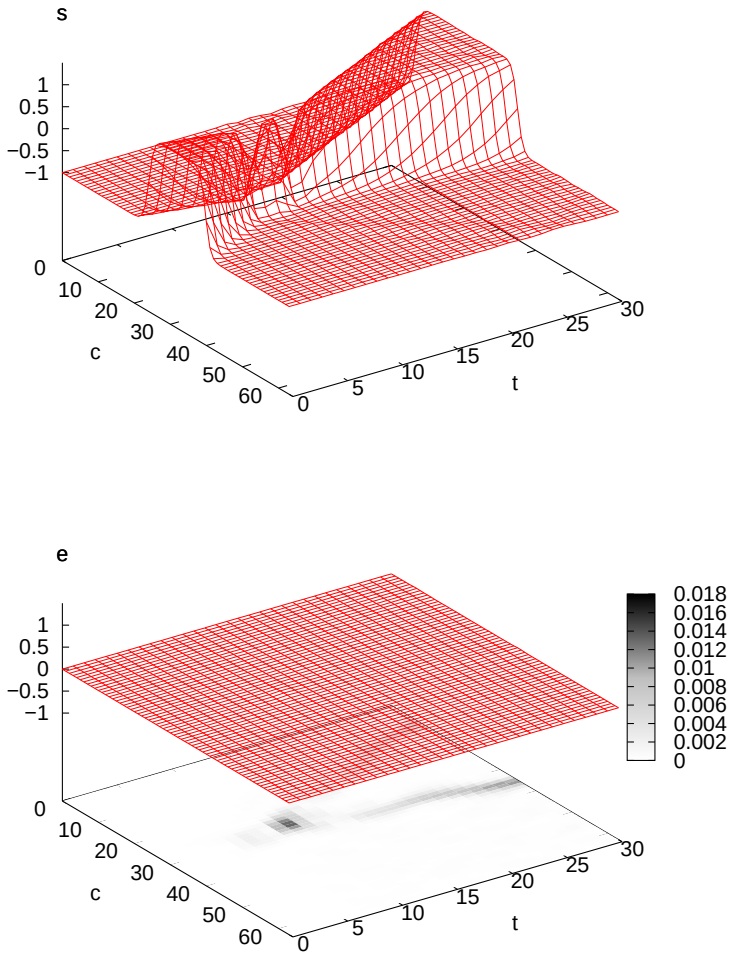

Abbildung 7.4: oben: Lösung der Φ^4 Gleichung tabellierten Gewichtsfunktionen nach Anwendung des TMA; unten: Abweichung dieser zur polynomialen Lösung aus Abb. 7.1

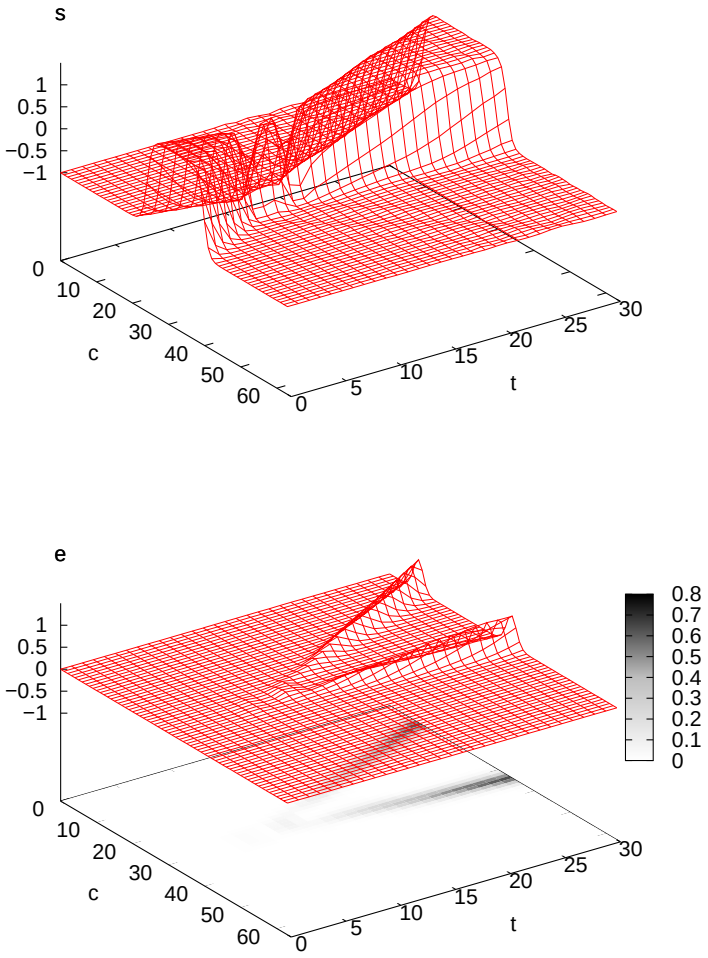

Abbildung 7.5: oben: Lösung der Φ^4 Gleichung tabellierten Gewichtsfunktionen nach Anwendung des TMA; unten: Abweichung dieser zur polynomialen Lösung aus Abb. 7.1

Teil IV

Schlußbemerkungen

Kapitel 8

Zusammenfassung

Im Rahmen dieser Arbeit wurde die Darstellung nichtlinearer Gewichtsfunktionen oder Ausgangsfunktionen eines CNN mittels linearer Interpolation untersucht. Ein lineares Interpolationsverfahren läßt sich im Vergleich zu anderen Methoden relativ einfach mit einem geringen Berechnungsaufwand implementieren und erscheint daher für schaltungstechnische Realisierungen derartiger Netzwerke besonders geeignet. Zur Repräsentation nichtlinearer Funktionen von CNN werden sowohl tabellierte Funktionen mit äquidistanten Stützstellen, als auch solche mit nichtäquidistanten Stützstellen herangezogen. Es wurden zwei Verfahren zur Bestimmung nichtäquidistanter Stützstellen entwickelt und ihre Genauigkeit bei numerischen Berechnung nichtlinearer partieller Differentialgleichungen mit CNN untersucht. Ausgehend von einer Diskretisierung der Ortskoordinaten unterschiedlicher partieller Differentialgleichungen ergaben sich CNN mit polynomialen Gewichtsfunktionen, deren genaue Approximation unter Verwendung einer linearen Interpolation mit minimaler Stützstellenzahl im Vordergrund der Untersuchungen stand. Bekannterweise lassen sich polynomiale Gewichtsfunktionen durch äquidistante Stützstellen darstellen. Für eine hinreichend große Approximationsgenauigkeit ist bei äquidistanten Stützstellen in einigen Fällen jedoch eine relativ hohe Anzahl Stützstellen erforderlich. Um die Anzahl der Stützstellen zu reduzieren, bzw. von vornherein mit weniger Stützstellen auszukommen, wurden in der vorliegenden Arbeit zwei Verfahren – der sogenannte "Tabel Minimising Algorithm" (TMA) und das Gelernte Interpolations Verfahren (GLI) – implementiert und untersucht. Während bei dem TMA von vorgegebenen äquidistanten Stützstellen, die eine definierte Gewichtsfunktion darstellen, ausgegangen wurde um eine nichtäquidistante Darstellung mit reduzierter Anzahl an Stützstellen zu erzeugen, besteht bei dem GLI-Verfahren die Aufgabenstellung mittels linearer Interpolation eine Darstellung, beispielsweise von Gewichtsfunktionen, nur anhand von Ausgangssignalen von CNN in einem überwachten Parametertraining zu bestimmen.
Zur Untersuchung der beiden entwickelten Algorithmen wurden Lösungen der Korteweg-de Vries (KDV) Gleichung, der Burgers Gleichung und der Φ^4-Gleichung zugrundegelegt. Es konnte gezeigt werden, daß mittels des TMA in allen untersuchten Fällen, eine relevante Stützstellenreduzierung bei Vorgabe der maximal zulässigen Abweichungen durchgeführt werden konnte. Allerdings hat es sich dabei gezeigt, daß die Wahl der Parameter ϵ und R des TMA einen bedeutenden Einfluß auf die Güte der Ergebnisse hat; deshalb sind zur optimalen Einstellung der Parameter ϵ und R des TMA jeweils Voruntersuchungen nötig. Der zweite hier untersuchte Ansatz basiert auf einem überwachten Parametertraining. Dabei wird mit dem implementierten Verfahren ein möglicher Parameterraum abgesucht und zum einen die Anzahl der Stützstellen, zum anderen aber auch die Abzissenwerte der Stützstellen der darzustellenden Funktion variiert.

Dabei kann das Optimierungsverfahren unter Umständen das globale Optimum verfehlen.
Die vorherigen Ergebnisse verdeutlichen, daß eine lineare Interpolation im Zusammenhang mit den vorgestellten Verfahren – TMA und GLI – zur Darstellung von nichtlinearen Gewichts- oder Ausgangsfunktionen bei CNN besonders geeignet sind. Die Anwendung beider Verfahren führte zu einer hohen Approximationsgenauigkeit von Lösungen der betrachteten partiellen Differentialgleichungen. Mit dem GLI-Verfahren besteht sogar die Möglichkeit eine genaue Repräsentation der jeweiligen Gewichts- oder Ausgangsfuntionen ohne Kenntnis von deren Definitionsweise zu ermitteln.

Danksagung

Herrn Prof. Dr. Ronald Tetzlaff danke ich für die Betreuung und die Durchsicht dieser Arbeit.
Herrn Dipl. Phys. Gunter Geis und Herrn Dipl. Phys. Frank Gollas möchte ich für zahlreiche fachliche Diskussionen und konstruktive Anregungen danken. Bei meinen Eltern möchte ich mich für die fortwährende Unterstützung meines gesamten Bildungsweges bedanken.

Teil V

Anhang

Anhang A

Abkürzungen und Fachbegriffe

A.1 Abkürzungen

Folgende Abkürzungen wurden im Rahmen dieser Diplomarbeit verwendet:

CNN	:	Zellulare Nichtlineare Netzwerke / engl.: **C**ellular **N**eural **N**etworks
GLI	:	**G**e**L**ernte **I**nterpolation: Verfahren zur Erzeugung von nicht äquidistanten Stützstellen
KdV	:	**K**orteweg-**d**e **V**ries (Gleichung)
KNN	:	**K**ünstliche **N**euronale **N**etzwerke
Lm	:	Lernmuster
TMA	:	Minimierungsalgorithmus vom Typ Greedy / engl.: **T**able **M**inimizing **A**lgorithm

A.2 Fachbegriffe

Definition 1 (Zellulares Nichtlineares Netzwerk)**.** *ein CNN ist eine beliebige räumliche Anordnung von lokal gekoppelten Zellen; wobei jede Zelle ein dynamisches System beschreibt, das Eingänge, Ausgänge und einen Zustand, der sich nach vorgeschriebenen Gesetzen entwickelt, besitzt.[4]*

Definition 2 (Chaos)**.** *Unter Chaos versteht man in der Mathematik oder Physik Systemzustände, die in der Chaostheorie untersucht werden.*
Beispielsweise wiederholt sich die Bewegung eines chaotischen Systems wie eines doppelten Pendels unter gewissen Voraussetzungen (Anfangsbedingungen) niemals, sondern verändert sich stetig, so dass sein Verhalten zufällig und ungeordnet erscheint. Man spricht von chaotischen Systemen, wenn man nicht vorhersagen kann, wie sich ein System entwickeln wird. Formal ist ein deterministisches System dann chaotisch, wenn unter nur minimalen Veränderungen der Anfangsbedingungen die zeitliche Entwicklung des Systems nicht berechenbar ist und zu deutlich unterschiedlichen Endzuständen führt. Andernfalls (kleine Ursache, kleine Wirkung oder große Ursache, große Wirkung) spricht man von stetigem Verhalten. Chaotische Systeme, wie z.B. das Wetter, sind instabil, kleine Änderungen können enorm (insbesondere exponentiell) verstärkt werden, große Änderungen können gedämpft werden.

Im mathematischen Sinne wird ein Prozess als chaotisch bezeichnet, wenn Systemvariablen des Prozesses zu einem beliebigen Zeitpunkt mit exponentieller Geschwindigkeit divergieren.[30]

Definition 3 (Differentialgleichung). *Eine Gleichung, in der als Variable Funktionen einer oder mehrerer Veränderlicher sowie Ableitungen dieser Funktion auftreten, bezeichnet man als Differentialgleichung. Sind die Variablen Funktionen mehrerer Veränderlicher, so spricht man von einer* **partiellen Differentialgleichung**; *kommen nur Funktionen einer Veränderlichen vor, so spricht man von einer* **gewöhnlichen Differentialgleichung**.*[15]*

Definition 4 (Polynom). *Allgemein ist ein Polynom in einer Variablen vom* **Grad** n *ein Term der Form* $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$ *mit* $a_n \neq 0$. *Die* **Koeffizienten** $a_0, a_1, \ldots, a_n$ *sind dabei reelle Zahlen.[15]*

Definition 5 ((lineare) Interpolation). *Von einer Funktion f seien die Werte an den Stellen x_1 und x_2 (Stützstellen, Grundpunkte) bekannt, man möchte aber $f(x)$ für eine Zwischenstelle $x_1 \leq x \leq x_2$ näherungsweise bestimmen. Dazu denkt man sich die Punkte $(x_1, f(x_1))$ und $(x_2, f(x_2))$ durch eine Strecke verbunden und die Kurve zwischen x_1 und x_2 durch diese Strecke ersetzt.[15] Den Wert y kann man auf der Strecke ablesen oder mit der Geradengleichung (siehe 4.1.1) berechnen.*

Definition 6 ((algorithmische) Komplexität (engl. complexity)). *Die Komplexität einer berechenbaren Funktion ist der zu ihrer Berechnung erforderliche Aufwand an Betriebsmitteln wie Speicherplatz, Rechenzeit, benötigte Geräte usw.*
Damit die Untersuchung des Rechenaufwands unabhängig von speziellen Computern ist, wird ein formales Berechnungsmodell zugrunde gelegt. Übliche Modelle sind hierbei z.B. Turingmaschinen, Registermaschinen und kombinatorische Schaltwerke. Man unterscheidet zunächst zwischen der Komplexität eines Algorithmus und der Komplexität einer Funktion. Die Komplexität eines Algorithmus ist der erforderliche Rechenaufwand bei einer konkreten Realisierung des Algorithmus innerhalb des Berechnungsmodells. Die Komplexität einer Funktion ist die Komplexität des bestmöglichen Algorithmus aus der Menge aller Algorithmen, die die Funktion berechnen.[10]

Definition 7 (Approximation). *bezeichnet im mathematischen Sinn eine Näherung.[30]*

Definition 8 (Solitone). *Solitäre Wellen, die die Eigenschaft haben, daß sie ohne Formänderung durcheinander hindurch propagierten werden Solitone genannt.*

Definition 9 (solitäre Wellen). *Solitäre Wellen sind Wellen, die ihre Geschwindigkeit und Form beibehalten. Die Geschwindigkeit ist von der Amplitude abhängig.*

Definition 10 (numerisch). *Darstellungen aus einem Zeichenvorrat, wenn dieser aus den Ziffern eines Zahlensystems ... besteht, heißen* **numerisch**.*[15]*

Definition 11 (äquidistant). *gleich weit voneinander entfernt, gleiche Abstände aufweisend.[15]*

Definition 12 (Geradengleichungen). *Jede Gleichung der Gestalt*

$$ax + by + c = 0$$

mit reellen Zahlen a, b, c ($a \neq 0$ oder $b \neq 0$) beschreibt eine Gerade in der Ebene. Man nennt diese Gleichung die allgemeine Form der Geradengleichung *(auch* allgemeine Gleichung 1. Grades*). Besondere Geradengleichungen sind die Achsenabschnittsform, die Hessesche Normalform, die Normalform, die Punkt-Steigungs-Form und die* **Zweipunkteform**.*[15]*

Definition 13 (Zweipunkteform der Geradengleichung). *Eine Gerade G in der Ebene (nicht parallel zur y-Achse) durch die beide Punkte $P_1(x_1; y_1)$, $P_2(x_2; y_2)$ mit $x_1 \neq x_2$ läßt sich durch die* Zweipunkteform *der Geradengleichung beschreiben.*

$$G : \frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

.[15]

Definition 14 (stetig). *Eine Funktion heißt stetig an einer Stelle x_0 ihres Definitionsbereichs, wenn es zu jeder Umgebung von $f(x_0)$ eine Umgebung von x_0 gibt, so daß für jedes x aus dieser Umgebung der Funktionswert $f(x)$ in der Umgebung von $f(x_0)$ liegt.[15] D.h., die Funktion $f(x)$ besitzt keine Sprungstellen.*

Definition 15 (differenzierbar). *Eine Funktion heißt n mal differenzierbar, wenn die Ableitung $\frac{d^n}{dx^n} f(x)$ existiert.*

Definition 16 (stetig differenzierbar). *Eine Funktion heißt nfach stetig differenzierbar, wenn die Ableitung $\frac{d^n}{dx^n} f(x)$ stetig ist.*

Definition 17 (Tangente). *Eine Gerade, welche eine Kurve in einem Punkt P berührt, heißt Tangente an diese Kurve in P.[15]*

Definition 18 (Diskretisierung). *Als Diskretisierung bezeichnet man in der Mathematik die Gewinnung algebraischer (diskreter) Daten aus kontinuierlicher Information;*

- *in Bezug auf Differentialgleichungen die Gewinnung von Näherungslösungen für ein kontinuierliches Problem durch Lösen eines endlich-dimensionalen Ersatzproblems;*
- *bei der Finite-Elemente-Methode die Zerlegung eines Kontinuums in Elemente, die durch Knoten miteinander verbunden sind.*

[30]

Definition 19 (Gebiet). *Bezeichnung für eine Punktmenge, in der man zwei Punkte stets so verbinden kann, daß die Verbindungslinie ganz in dieser Punktmenge verläuft. Von besonderer Bedeutung sind Gebietseinteilungen der Ebene. Eine fundamentale Eigenschaft der Ebene ist durch den* **Jordanschen Kurvensatz** *(nach C. Jordan, 1838-1922) gegeben:* Eine einfach geschlossene Kurve teilt die Ebene in zwei Gebiete auf, das Innere der Kurve und das Äußere. *Man kann also drei disjunkte Punktmengen unterscheiden: die Kurve k selbst, das innere Gebiet k_i und das äußere Gebiet k_a. Die Kurve k wird auch als* **Grenze** *oder* **Rand** *der Gebiete bezeichnet.[15]*

Definition 20 (Normale). *Gerade, die in einem vorgegebenen Punkt einer Kurve rechtwinklig zur Tangente in diesem Punkt ist.[15]*

Definition 21 (Schwellenwertelement). *Ein* **Schwellenwertelement** *ist eine Verarbeitungseinheit für reelle Zahlen mit n Eingängen $x_1, \ldots, x_n$ und einem Ausgang y. Der Einheit als Ganzer ist ein* **Schwellwert** *θ und jedem Eingang x_i ein* **Gewicht** *w_i zugeordnet. Ein Schwellenwertelement berechnet die Funktion*

$$y = \begin{cases} 1, & \textit{falls} \quad \sum_{i=1}^{n} w_i x_i \geq \theta \\ 0, & \textit{sonst} \end{cases}$$

.[2]

Definition 22 (Boolesche Funktion). *Eine Boolesche Funktion ist eine Abbildung* $f : \{0,1\}^n \to \{0,1\}$*, wobei* $\{0,1\}^n$ *die Menge aller n-Tupel über* $\{0,1\}$ *ist. Oft auftretende Boolesche Funktionen sind die NOT-Funktion, die UND-Funktion und die ODER-Funktion. Jede Boolesche Funktion läßt sich mit Hilfe der Funktionen NOT, UND und ODER darstellen.[10]*

Definition 23 (Gerichteter Graph). *Ein gerichteter* **Graph** *ist ein Paar* $G = (V, E)$ *bestehend aus einer (endlichen) Menge* V *von* **Knoten** *(vertices, nodes) und einer (endlichen) Menge* $E \subseteq V \times V$ *von* **Kanten** *(edges). Wir sagen, daß eine Kante* $e = (v, u) \in E$ *vom Knoten* v *auf den Knoten* u **gerichtet** *sei.[2]*

Definition 24 (Hopfield-Netze). *Als Hopfield-Netz (nach John Hopfield) bezeichnet man eine besondere Form eines künstlichen neuronalen Netzes. Es handelt sich dabei um ein Netz mit Rückkopplung. Das Hopfield-Netz wird daher auch gelegentlich als Autoassoziationsnetz bezeichnet.*
Bei einem Hopfield-Netz existiert nur eine Schicht, die gleichzeitig als Ein- und Ausgabeschicht fungiert. Jedes Neuron ist mit jedem, ausgenommen sich selbst, verbunden.[30]

Definition 25 (Assoziativspeicher). *Bezeichnung für einen Speicher mit wahlfreiem Zugriff, bei dem nicht auf eine durch eine Adresse spezifizierte Speicherzelle erfolgt, sondern auf alle Speicherzellen gleichzeitig, die eine vorgegebene Inhaltsspezifikation erfüllen. Der Inhalt des Suchregisters wird an den durch das Maskenregister mit* 1 *markierten Bitposition in einem Zeittakt gleichzeitig mit den entsprechenden Bits aller Speicherzellen verglichen. Die im Vergleichsindikator mit* 1 *belegten Bits zeigen die Speicherzellen an, die die Inhaltsspezifikation erfüllen.*
Wegen der komplexen Vergleichslogik sind Assoziativspeicher relativ teuer. Sie werden daher im allgemeinen nur für Spezialzwecke eingesetzt (z.B. in der Speicherverwaltung).[10]

Definition 26 (Greedy-Algorithmus). *Ein Algorithmus, der immer die beste direkte oder lokale Lösung, beim finden einer Antwort, benutzt. Greedy-Algorithmen finden für einige Optimierungsprobleme die gesamte oder globale optimale Lösung, aber für einige Fälle anderere Probleme finden sie nur suboptimale Lösungen.[20]*

Definition 27 (asymptotische Notation). *Landau-Symbole werden in der Mathematik und in der Informatik verwendet, um das asymptotische Verhalten von Funktionen und Folgen zu beschreiben. In der Informatik werden sie insbesondere in der Komplexitätstheorie verwendet, um verschiedene Probleme und Algorithmen danach zu vergleichen, wie "schwierig" oder aufwendig sie zu berechnen sind.*
Groß O und klein o sind die am häufigsten verwendeten Landau-Symbole; darüber hinaus gibt es noch Ω*,* ω *und* Θ*.*

Notation	*Definition*	*Mathematische Definition*
$f(x) \in O(g(x))$	*asymptotische obere Schranke*	$0 \leq \limsup_{x \to a} \lvert\frac{f(x)}{g(x)}\rvert < \infty$
$f(x) \in o(g(x))$	*asymptotische vernachlässigbar*	$0 = \lim_{x \to a} \lvert\frac{f(x)}{g(x)}\rvert$
$f(x) \in \Omega(g(x))$	*asymptotische untere Schranke*	$0 < \liminf_{x \to a} \lvert\frac{f(x)}{g(x)}\rvert \leq \infty$
$f(x) \in \omega(g(x))$	*asymptotische dominant*	$\lim_{x \to a} \lvert\frac{f(x)}{g(x)}\rvert = \infty$
$f(x) \in \Theta(g(x))$	*asymptotische scharfe Schranke*	$0 < \liminf_{x \to a} \lvert\frac{f(x)}{g(x)}\rvert \leq \limsup_{x \to a} \lvert\frac{f(x)}{g(x)}\rvert < \infty$

[30]

Anhang B

Symbolverzeichnis

x_i	Stützstelle, oder x-Koordinate eines diskreten Punktes, an der Stelle i
y_i	Funktionswert an der Stützstelle x_i, oder an der diskreten der Stelle i
G	Gerade oder Geradengleichung
h	Abstand zwischen den Stützstellen
$f_{spline}8x)$	Spline-Funktion
Γ	stückweise stetig differenzierbare Randkurve eines beschränkten Gebietes G (siehe Anhang **B** Definiton 19)
$\vec{n}$	Normalenvektor, steht senkrecht auf einer Kurve (siehe Anhang A Definiton 20)
s_i	Eingang eines Schwellenwertelementes oder Zustand der Zelle c_i
a_i	Gewicht eines Schwellenwertelementes, bzw. des rückgeckoppelten Zustands in einem eindimensionalen CNN
v, v_i bzw. v_{ij}	Bias, bzw. Schwellwert
o	Ausgang eines Schwellenwertelementes
$U = (C, K)$	Gerichteter Graph (siehe Anhang A Definiton 23)
C	Menge aller Neuronen, bzw. Zellen
K	Menge aller Kanten in einem gerichteten Graphen
c, c_i bzw. c_{ij}	einzelnes Neuron, bzw. einzelne Zelle
$a_{c_j c_i}$	Gewicht zwischen c_i und c_j (gerichteter Graph)
$pred(c_j) = \{c_i \in C \vert (c_i, c_j) \in K\}$	Menge der Vorgänger des Konten $c_j \in C$ in einem gerichteten Graphen
$\mathcal{N}$ bzw. $\mathcal{N}_{ij} = \{c_{kl} \in U : max\{\vert k - i\vert, \vert l - j\vert\} \leq r\}$	Nachbarschaft
$\vec{u}$	Eingänge
$\dot{s}_{ij} := \frac{d}{dt} s_{ij}(c_{ij}, t)$	Folgezustand, bzw. Zustandsänderung der Zelle c_{ij}
s_{ij}	Zustand der Zelle c_{ij}

a_{kl}	Element der A-Template
$o(s_{ij})$	Ausgangsfunktion der Zelle c_{ij}
b_{kl}	Element der B-Template
v_{ij}	Bias oder Schwellenwert der Zelle c_{ij}
C_s	Kondensator
R_s und R_o	Wiederstände
$\tau := 1/R_sC_s$	Vorfaktor in der CNN Zustandsgleichung, meist $= 1$
$\mathbf{M} := \begin{pmatrix} m_{-1-1} & m_{0-1} & m_{+1-1} \\ m_{-10} & m_{00} & m_{+10} \\ m_{-1+1} & m_{0+1} & m_{+1+1} \end{pmatrix}$	Indizierung einer 3×3 Matrix, bzw. Template
$\mathbf{A}$	A-Template
$\mathbf{B}$	B-Template
$a(o(s_{kl}))$	Gewichtsfunktion in der A-Template
$b(u_{kl})$	Gewichtsfunktion in der B-Template
P	Maximaler Polynomgrad bei Gewichtsfunktionen
s_i^m	Zustand der Zelle c_i in der Schicht m
L^m	Zellschicht m
$a_i^{m'm}(o(s_i))$	Gewichtsfunktion, die den Einfluß von Zellen in Schicht m' auf Zellen in Schicht m definiert (Elemente der A-Template)
$\mathcal{N}_i := \{i-r, i-r+1, \ldots, i+r\}$	Nachbarschaft bei eindimensionalen CNN
ϵ	vorgegebener Fehlerwert, bzw. obere Fehlerschranke
R	Parameter für Greedy-Algorithmus (siehe Kapitel 4.1.1)
N	(maximale) Anzahl Zellen, Stützstellen oder Eingaben, bzw. Schichten
$O\left(f_g(N)\right) = \{f_T(N) \vert \exists d, N_0 > 0 : 0 \leq f_T(N) \leq df_g(N) \forall N \geq N_0\}$	asymptotische obere Grenze
$T(N)$	abstrakte Rechenzeit in Abhängigkeit von der Eingabe N
$e_i := \vert f_1(x_i) - f_2(x_i)\vert$	absoluter Fehler (vertikaler Abstand zwischen zwei Funktionswerten), an der Stelle i
$e := \vert s - s_{ref}\vert$	Differenz zwischen Zuständen einer Lösung und der zugehörigen Referenz
w_i	Geschwindigkeit der solitären Welle i
Δt	zeitliche Integrationsschrittweite

Literaturverzeichnis

[1] Arena, P., R. Caponetto, L. Fortuna und D. Porto: *Nonlinear Noninteger Order Circuits an Systems – An Introduction*, Band 38 der Reihe *World Scientific Series on Nonlinear Science*. World Scientific, A Auflage, 2002.

[2] Borgelt, Christian, Frank Klawonn, Rudolf Kruse und Detlef Nauck: *Neuro-Fuzzy-Systeme*. Friedr. Vieweg & Sohn, Wiesbaden, 3. Auflage, Oktober 2003.

[3] Bronstein, I.N., K.A. Semendjajew, G. Musiol und H. Mühlig: *Taschenbuch der Mathematik*. Verlag Harrideutsch, Thun, Neubearbeitung 3. Auflage, 1996.

[4] Chua, Leon O.: *CNN: A Paradigm for Complexity*, Band A 31 der Reihe *World Scientific Series on Nonlinear Science*. University of California, 1998.

[5] Chua, Leon O. und Lin Yang: *Cellular Neural Networks: Theory*. IEEE Transactions on Circuits and Systems, 35(10), Oktober 1988.

[6] Cormen, Thomas H., Charles E. Leiserson und Ronald L. Rivest: *Introducing to Algorithms*. MIT Press, Cambridge Massachusetts, 23. Auflage, 1999.

[7] Dodd, R. K., J. C. Eilbeck, J. D. Gibbon H. C. und Morris: *Solitons and Nonlinear Wave Equations*. Academic Press Inc. Ltd., London, 1984.

[8] Dogaru, Radu: *Universality and Emergent Computation in Cellular Neural Networks*, Band 43 der Reihe *World Scientific Series on Nonlinear Science*. World Scientific, A Auflage, MAi 2003.

[9] Drazin, P. G.: *Solitons*. 85. London Matematical Society Lecture Notes, London, New York, New Rochelle, Melburne and Sydney, 1983.

[10] Engesser, Dipl.-Math. Hermann (Herausgeber): *Duden Informatik*. Dudenverlag, Mannheim, 2. vollständig überarbeitete und erweiterte Auflage, 1993. Bearbeitet von Prof. Dr. Volker Claus und Dr. Andreas Schwill.

[11] Geis, Gunter, Michael Reinisch, Ronald Tetzlaff und Frank Puffer: *Linear interpolation of nonlinearities in Cellular Neural Networks*. In: *Proceedings of the* 8^{th} *IEEE International Biannual Workshop of CNNs and their Applications*, Seiten 393–398, Budapest, Juli 2004.

[12] Gollas, Frank und Ronald Tetzlaff: *Modeling Complex Systems by Reaction-Diffusion Cellular Nonlinear Networks with Polynomial Weight-Functions*. In: *SPIE'05*, Seville, Mai 2005. Johann Wolfgang Goethe Universität.

[13] Hirota, Ryogo: *Exact Solution of the Korteweg-de Vries Equation for Multiple Collisions of Solitons.* Physical Review Letters, 27(18), November 1971.

[14] Kunz, Roland: *Simulation Zellularer Neuronaler Netzwerke.* Institut für Angewandte Physik der Johann Wolfgang Goethe Universität, Frankfurt am Main, 1996.

[15] Lexikonredaktion, Meyers (Herausgeber): *Der kleine Duden Mathematik.* Dudenverlag, Mannheim, 2. Auflage, 1996. Bearbeitet von Dipl.-Math. Hermann Engesser.

[16] Lian, G., R. Domínguez-Castro, S. Espejo und A. Rodríquez-Vázquez: *Design of a large-complexity analog I/O CNN UC.* In: Roska, T., C. Beccari, M. Biex, P.P. Civalleri und M. Gilli (Herausgeber): *Design Automation Day on Cellular Visual Microprocessor*, Stresa, 1999.

[17] Loncar, Andrej: *Kennlinienverfahren zur Modellierung nichtlinearer Systeme mit Zellularen Neuronalen Netzwerken.* Institut für Angewandte Physik der Johann Wolfgang Goethe Universität, Frankfurt am Main, März 1999.

[18] Niederhöfer, Christian und Ronald Tetzlaff: *Recent Results on the Prediction of EEG Signals in Epilepsy by Discrete-Time Cellular Neural Networks (DTCNN).* In: *ISCAS 2005*, Kobe, Mai 2005.

[19] Niehl, Elke: *Zur numerischen und analytischen Behandlung der Korteweg-de Vries Gleichung.* Mathematisch-Naturwissenschaftliche Fakultät der Rheinisch-Westfälische Technische Hochschule, Aachen, 1991.

[20] NIST: *Dictionary of Algorithms and Data Structures.* Homepage, National Institute of Standards and Technology, http://www.nist.gov/dads/, 2004.

[21] Ottmann, Thomas und Peter Widmayer: *Algorithmen und Datenstrukturen.* Spektrum Akademischer Verlag, Heidelberg, 3. überarbeitete Auflage, 1996.

[22] Puffer, Frank: *Neuronale Modellierung mehrdimensionaler nichtlinearer Systeme: Lernverfahren und Anwendungen.* 2000.

[23] Puffer, Frank, Ronald Tetzlaff und D. Wolf: *A Learning Algorithm for Cellular Neural Networks (CNN) solving nonlinear Partial Differential Equations.* In: *ISSSE'95*, Seiten 105–405, San Francisco, 1995.

[24] Puffer, Frank, Ronald Tetzlaff und D. Wolf: *A Learning Algorithm for the Dynamics of CNN with Nonlinear Templates Part II: Continuous-Time Case.* In: *CNNA'96*, 1996.

[25] Reinisch, Michael, Gunter Geis und Ronald Tetzlaff: *System Identification by Cellular Neural Networks (CNN): Linear Interpolation of Nonlinear Weight Functions.* In: *SPIE*, Seville, Mai 2005.

[26] Rodríguez-Vásquez, S. Espejo, R. Domínguez-Castro, J.L. Huentas und Sánhez-Sinencio: *Current-Mode Techniques for the Implementation of Continuos- and Discrete-Time Cellular Neural Network.* In: Tetzlaff, Prof. Dr. Ronald (Herausgeber): *Proceedings of the 7^{th} IEEE International Workshop of CNNs and their Applications*, Seiten 132–146. world Scientific, 2002.

[27] Schwarz, Hans Rudolf: *Numerische Mathematik.* B.G. Teubner, Stuttgart, 4. Auflage, 1997.

[28] Slavova, Angela: *Cellular Neural Networks: Dynamics and Modelling*, Band 16 der Reihe *Mathematical Modelling: Theoriy and Applications*. Kluwer Academic Publisher, 2003.

[29] Tetzlaff, Ronald: *Stochastische und neuronale Modellierung nichtlinearer Systeme*. Habilitationsschrift. Institut für Angewandte Physik, Frankfurt am Main, 1998.

[30] Wikipedia: *Wikipedia die freie Enzyklopädie*. Homepage, Wikimedia Foundation Inc., http://de.wikipedia.org/wiki/Hauptseite, 2005.

Tabellenverzeichnis

Abbildungsverzeichnis