GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

Fakultät für
Physik

Bachelorarbeit

Die Dichte-Matrix-Renormierungsgruppe für die tranversale Ising-Quanten-Kette: Ein Demonstrationsprogramm

The Density Matrix Renormalization Group for the transverse Ising quantum chain: A demonstration program

angefertigt von

Thomas Köhler

aus Bremen

am Institut für Theoretische Physik

Bearbeitungszeit: 16. Mai 2011 bis 22. August 2011

Erstgutachter: PD Dr. Andreas Honecker

Zweitgutachter: Prof. Dr. Thomas Pruschke

Inhaltsverzeichnis

1. Einleitung

Ferromagnetismus ist die bekannteste Ausprägungsform des Magnetismus in Festkörpern, da er für die Bildung von Dauermagneten wie z.B. Eisen, Kobald und Nickel verantwortlich ist.
Der Ferro-, wie auch der Antiferromagnetismus, kann durch die Austauschwechselwirkungen einzelner Teilchen dargestellt werden. Da die Dimension des Hamilton-Operators eines so dargestellten Festkörpers zu groß ist, um damit zu arbeiten, wird für die Untersuchung von magnetischen Effekten ein effektives Spinmodell eingeführt. Hierbei wird auf alle anderen Freiheitsgrade der Teilchen verzichtet und nur der Spin betrachtet.
Um dies für einen makroskopischen Festkörper zu tun, steigt allerdings die Dimension des beschreibenden Hamilton-Operators wieder rapide an. Dies verhindert wiederum eine analytische Berechnung der interessanten ferro- und antiferromagnetischen Effekte. Um das magnetische Verhalten von Festkörpern trotzdem vorraussagen zu können, werden Simulationen und Näherungsverfahren eingesetzt.
Für den Grundzustand (T=0) eines ein-dimensionalen Systems hat sich die Dichte-Matrix-Renormierungsgruppe (DMRG) [14][15][7], welche 1992 von Steven White eingeführt wurde, als sehr erfolgreich erwiesen.
Später stellte sich heraus, dass die DMRG sehr stark mit den Matrix-Produkt-Zuständen (MPS) verwandt ist [4]. Allerdings wurde dies erst 2004 vollständig untersucht, da die möglichen Verbesserungen zunächst als zu gering eingestuft wurden [12].
Um das Verständnis der DMRG in MPS-Schreibweise zu erleichtern, ist das Ziel dieser Arbeit ein Programm, welches die berechneten Ergebnisse instantan darstellt. Hierbei steht explizit nicht die Leistungsfähigkeit der Implementation im Vordergrund, sondern der pädagogische Nutzen einer Visualisierung.
Das Applet ist so konzipiert, dass verschiedene ein-dimensionale Spinketten untersucht werden können. Speziell wird die Ising-Quanten-Kette im transversalen Magnetfeld betrachtet, da hierbei ein Phasenübergang beobachtet werden kann. Darüber hinaus gibt es in diesem Modell keine Quantenzahlen, die zur effizienteren Berechnung ausgenutzt werden könnten.

2. Matrix-Produkt-Zustände

Im Folgenden wird die Theorie für die Implementierung der Matrix-Produkt-Zustände (MPS) und der Matrix-Produkt-Operatoren (MPO) beschrieben. Hierbei wird sich stark am Artikel „The density-matrix renormalization group in the age of matrix product states“ von Ulrich Schollwöck [12] orientiert.

2.1. Definition und Konstruktion von MPS

Betrachtet man eine ein-dimensionale Spin-$\frac{1}{2}$-Quantenkette mit L Plätzen und offenen Rändern, die an jedem Platz i den Zustand $\sigma_i = \{\uparrow, \downarrow\}$ annehmen kann, so kann man den Quantenzustand auf dieser Kette als Summe über die Basiszustände darstellen:

$$|\psi\rangle = \sum_{\sigma_1,\ldots,\sigma_L} c_{\sigma_1\ldots\sigma_L} |\sigma_1, \ldots, \sigma_L\rangle$$

Hierbei sind die Koeffizienten $c_{\sigma_1\ldots\sigma_L} \in \mathbb{C}$. Die Anzahl der Koeffizienten $c_{\sigma_1\ldots\sigma_L}$ steigt exponentiell (2^L) mit der Länge der Kette an. MPS stellen jeden dieser Koeffizienten als Matrix-Produkt dar, die Matrizen sind hierbei nur lokal für jeweils einen Platz definiert. Um einen MPS aus einem bekannten Quantenzustand mit Koeffizienten $c_{\sigma_1\ldots\sigma_L} \in \mathbb{C}$ zu erzeugen, müssen im ersten Schritt die 2^L Koeffizienten in eine Matrix c' der Dimension $2 \times 2^{L-1}$ umsortiert werden:

$$c_{\sigma_1\ldots\sigma_L} = c'_{\sigma_1,(\sigma_2\ldots\sigma_L)}$$

Auf die Matrix c' wird dann eine Single-Value-Decomposition (SVD)[2] angewandt:

$$c'_{\sigma_1,(\sigma_2\ldots\sigma_L)} = \sum_{a_1}^{r_1} U_{\sigma_1,a_1} S_{a_1,a_1} (V^\dagger)_{a_1,(\sigma_2\ldots\sigma_L)}$$

Aus S und $V^\dagger$ wird ein neues $\tilde{c}'_{a_1\sigma_2\ldots\sigma_L}$ erzeugt, welches dann wiederum in eine Matrix $c''_{(a_1\sigma_2),(\sigma_3\ldots\sigma_L)}$ umsortiert wird. Darüber hinaus wird die Matrix U, für die $U^\dagger U = \mathbb{1}$ gilt, in zwei A^{σ_1}-Matrizen umsortiert. Die Einträge der A-Matrix ergeben sich zu $A^{\sigma_1}_{1,a_1} = U_{\sigma_1,a_1}$.

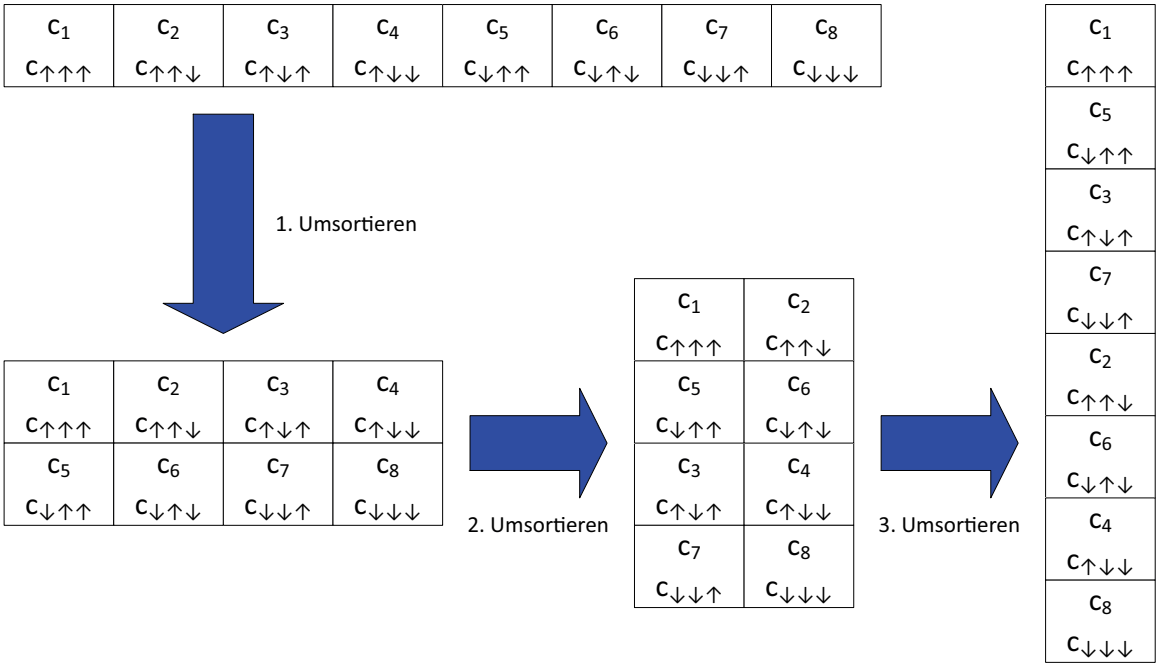

Abb. 2.1.: Umsortieren der Koeffizienten ohne Berücksichtigung dessen, dass ein Teil der enthaltenen Daten ausgekoppelt wird.

Man erhält:

$$c_{\sigma_1,\dots\sigma_L} = \sum_{a_1} A^{\sigma_1}_{1,a_1} c''_{(a_1\sigma_2),(\sigma_3\dots\sigma_L)}$$

Nun wird auf $c''_{(a_1\sigma_2),(\sigma_3\dots\sigma_L)}$ wiederum eine SVD angewandt. S und $V^\dagger$ werden wieder zu einer Matrix zusammengefasst und das entstehende U wieder in zwei A-Matrizen umsortiert. Die Einträge ergeben sich jetzt zu $A^{\sigma_2}_{a_1,a_2} = U_{(a_1\sigma_2),a_2}$.
Dies wird solange wiederholt, bis der Platz $L-1$ erreicht ist. An dieser Stelle ergibt sich dann:

$$\begin{aligned} c_{\sigma_1,\dots\sigma_L} &= \sum_{a_1}\cdots\sum_{a_{l-1}} A^{\sigma_1}_{1,a_1}\cdots A^{\sigma_{l-1}}_{a_{l-2},a_{l-1}} \underbrace{c'^{\cdots\prime}_{(a_{l-1}\sigma_l),(\sigma_{l+1}\dots\sigma_L)}}_{M^{\sigma_L}_{a_{L-1},a_L}} \\ &= A^{\sigma_1}\cdots A^{\sigma_{L-1}} M^{\sigma_L} \end{aligned}$$

Hierbei erfüllen alle A-Matrizen die Bedingung für links-Normierung $\sum_{\sigma_l} A^{\sigma_l\dagger} A^{\sigma_l} = \mathbb{1}$. Analog dazu kann man einen rechts-normierten MPS erzeugen, indem man $V^\dagger$ in B-Matrizen umformt und U und S jeweils weiterverwendet. Alle B-Matrizen erfüllen dann die Bedingung für rechts-Normierung $\sum_{\sigma_l} B^{\sigma_l} B^{\sigma_l\dagger} = \mathbb{1}$. Einen gemischt-normierten Zustand erhält man, wenn man von beiden Seiten beginnt und am Platz l das U und das

S bzw. das S und das $V^\dagger$ zu einer Matrix M umformt. Die Matrix M^{σ_l} erfüllt keine Normierungsbedingung.
Ein so entwickelter MPS ist eine exakte Darstellung des Zustandes. Ein entscheidender Vorzug der MPS besteht aber darin, dass man die maximale Dimension $d_{\max}$ der Matrizen begrenzen kann und trotzdem eine sehr gute Näherung an den exakten Zustand erhält.

2.2. Skalarprodukt

Das Skalarprodukt stellt sich in MPS-Schreibweise folgendermaßen dar:

$$\langle\phi|\psi\rangle = \sum_{\boldsymbol{\sigma}\boldsymbol{\sigma'}} \langle\boldsymbol{\sigma'}|\tilde{M}^{\sigma'_1 *}\cdots\tilde{M}^{\sigma'_L *} M^{\sigma_1}\cdots M^{\sigma_L}|\boldsymbol{\sigma}\rangle$$

Dies lässt sich, da die σ_i orthogonale Basisvektoren sind, zusammenfassen zu:

$$\langle\phi|\psi\rangle = \sum_{\boldsymbol{\sigma}} \tilde{M}^{\sigma_1 *}\cdots\tilde{M}^{\sigma_L *} M^{\sigma_1}\cdots M^{\sigma_L}$$

Um diesen Ausdruck zu berechnen, müsste man 2^L Matrixprodukte summieren, die jeweils aus $2L-1$ Matrixmultiplikationen bestehen. Das bedeutet, dass der Aufwand exponential mit der Länge der Kette steigt. Wenn man den Ausdruck allerdings umschreibt, reduziert sich der Aufwand auf $2L$ Additionen und $4L-2$ Matrixmultiplikationen, was einem polynomial zur Kettenlänge steigenden Aufwand entspricht:

$$\langle\phi|\psi\rangle = \sum_{\sigma_L} \tilde{M}^{\sigma_1 \dagger}\left(\cdots\left(\sum_{\sigma_2}\tilde{M}^{\sigma_2\dagger}\left(\sum_{\sigma_1}\tilde{M}^{\sigma_1\dagger}M^{\sigma_1}\right)M^{\sigma_2}\right)\cdots\right)M^{\sigma_L}$$

2.3. Matrix-Produkt-Operator (MPO)

Jeder Operator kann als Matrix-Produkt-Operator geschrieben werden:

$$\begin{aligned}\hat{O} &= \sum_{\boldsymbol{\sigma},\boldsymbol{\sigma'}} c_{(\sigma_1\ldots\sigma_L)(\sigma'_1\ldots\sigma'_L)}|\boldsymbol{\sigma}\rangle\langle\boldsymbol{\sigma'}| \\ &= \sum_{\boldsymbol{\sigma},\boldsymbol{\sigma'}} c_{(\sigma_1\sigma'_1)\ldots(\sigma_L\sigma'_L)}|\boldsymbol{\sigma}\rangle\langle\boldsymbol{\sigma'}| \\ &= \sum_{\boldsymbol{\sigma},\boldsymbol{\sigma'}} W^{\sigma_1\sigma'_1}\ldots W^{\sigma_L\sigma'_L}|\boldsymbol{\sigma}\rangle\langle\boldsymbol{\sigma'}|\end{aligned}$$

Hierbei nehmen die W-Matrizen die Rolle der M-Matrizen eines MPS ein. Der Unterschied besteht darin, dass sie einen weiteren Index und somit eine weitere Dimension besitzen.

2.3.1. MPO-Anwendung

Beim Anwenden eines MPO auf einen MPS kann der Ausdruck wieder stark zusammengefasst werden:

$$\begin{aligned}\hat{O}|\psi\rangle &= \sum_{\boldsymbol{\sigma},\boldsymbol{\sigma'},\boldsymbol{\sigma''}} (W^{\sigma_1\sigma'_1}\dots W^{\sigma_L\sigma'_L})(M^{\sigma''_1}\cdots M^{\sigma''_L})|\boldsymbol{\sigma}\rangle \underbrace{\langle\boldsymbol{\sigma'}|\boldsymbol{\sigma''}\rangle}_{\delta_{\sigma'\sigma''}} \\ &= \sum_{\boldsymbol{\sigma},\boldsymbol{\sigma'}} (W^{\sigma_1\sigma'_1}\dots W^{\sigma_L\sigma'_L})(M^{\sigma''_1}\cdots M^{\sigma''_L})|\boldsymbol{\sigma}\rangle \\ &= \sum_{\boldsymbol{\sigma}} \tilde{M}^{\sigma_1}\cdots\tilde{M}^{\sigma_L}|\boldsymbol{\sigma}\rangle = |\tilde{\psi}\rangle\end{aligned}$$

Die Einträge der $\tilde{M}$-Matrix sind gegeben durch:

$$\tilde{M}^{\sigma_i} = \sum_{\sigma'_i} W^{\sigma_i\sigma'_i}_{b_{i-1},b_i} M^{\sigma'_i}_{a_{i-1},a_i}$$

$|\tilde{\psi}\rangle$ stellt somit wiederum einen MPS dar. Allerdings ist die Dimension der $\tilde{M}$-Matrizen das Produkt der Dimensionen der MPS-Matrizen und der MPO-Matrizen.
Auf diesem Weg lässt sich der Erwartungswert eines Operators bestimmen:

$$\langle\psi|\hat{O}|\psi\rangle = \langle\psi|\tilde{\psi}\rangle$$

Wiederholtes Anwenden eines MPO auf einen MPS führt dazu, dass die Dimension des entstehenden MPS immer weiter ansteigt.

2.3.2. Hamilton-Operator

Da sich jeder Operator als MPO darstellen lässt, muss dies auch für einen Hamilton-Operator

$$\hat{H} = J\sum_{i=1}^{L-1}\frac{1}{2}\hat{S}_i^+\hat{S}_{i+1}^- + \frac{1}{2}\hat{S}_i^-\hat{S}_{i+1}^+ + J^z\hat{S}_i^z\hat{S}_{i+1}^z - h_z\sum_i^L\hat{S}_i^z - h_x\sum_i^L\hat{S}_i^x$$

gelten. Hierbei stehen J und J^z für die Kopplungskonstanten der Spins und h^z und h^x für Stärke der Magnetfelder in der entspechenden Richtung. Dass sich dieser MPO allerdings explizit, mit einer geringen Dimension, konstruieren lässt, ist der Tensor-Produkt-Struktur des Hamilton-Operators zu verdanken:

$$\hat{H} = J^z \hat{S}_1^z \otimes \hat{S}_2^z \otimes \hat{I}_3 \otimes \hat{I}_4 \ldots + \hat{I}_1 \otimes J^z \hat{S}_2^z \otimes \hat{S}_3^z \otimes \hat{I}_4 \ldots + \ldots$$

Um den Hamilton-Operator als MPO konstruieren zu können, muss dieser zunächst in Operatoren, die jeweils auf einen Platz wirken, zerlegt werden. Hierzu werden die Operatoren

$$\hat{W}^{[i]} = \sum_{\sigma_i, \sigma_i'} W^{\sigma_i \sigma_i'} |\sigma_i\rangle\langle\sigma_i'|$$

eingeführt. Diese haben, für den oben beschriebenen Hamiltonian, folgende Form:

$$\hat{W}^{[i]} = \begin{bmatrix} \hat{I} & 0 & 0 & 0 & 0 \\ \hat{S}^+ & 0 & 0 & 0 & 0 \\ \hat{S}^- & 0 & 0 & 0 & 0 \\ \hat{S}^z & 0 & 0 & 0 & 0 \\ -h_z \hat{S}^z - h_x \hat{S}^x & (J/2)\hat{S}^- & (J/2)\hat{S}^+ & J^z \hat{S}^z & \hat{I} \end{bmatrix}$$

Zwei Ausnahmen bilden die Operatoren für den ersten und den letzten Platz:

$$\hat{W}^{[1]} = \begin{bmatrix} -h_z \hat{S}^z - h_x \hat{S}^x & (J/2)\hat{S}^- & (J/2)\hat{S}^+ & J^z \hat{S}^z & \hat{I} \end{bmatrix} \quad \hat{W}^{[L]} = \begin{bmatrix} \hat{I} \\ \hat{S}^+ \\ \hat{S}^- \\ \hat{S}^z \\ -h_z \hat{S}^z - h_x \hat{S}^x \end{bmatrix}$$

Entsprechend der Konstruktion der $\hat{W}^{[i]}$ können die $W^{\sigma_i, \sigma_i'}$ bestimmt werden:

$$\hat{W}_{j,k}^{\sigma_i \sigma_i'} = \langle \sigma_i | \hat{W}_{j,k}^{[i]} | \sigma_i' \rangle$$

3. Algorithmus

Im Folgenden soll die DMRG in der Matrix-Produkt-Schreibweise beschrieben werden. Hierzu wird zunächst das Grundproblem beschrieben, dann eine effizientere Art um den Erwartungswert zu bestimmen und schließlich ein gesamter Sweep-Step. Auch hier wird sich an Schollwöck [12] orientiert.

3.1. Grundproblem

Das Grundproblem ist das Finden eines Zustands $|\psi\rangle$, der $\frac{\langle\psi|\hat{H}|\psi\rangle}{\langle\psi|\psi\rangle}$ minimiert. Hierzu wird der Erwartungswert $\langle\psi|\hat{H}|\psi\rangle$ minimiert, während seine Normierung $\langle\psi|\psi\rangle$ konstant bleibt.
Um die Minimierung des Erwartungswertes zu erreichen, wird der Variationsansatz verwendet. Hierzu wird, wie in 3.4 beschrieben, das Eigenwert-Problem an einem Platz gelöst; dies entspricht der Single-Site-DMRG. Dass das System auf diese Weise gegen den Grundzustand konvergiert, konnte bisher nicht allgemein gezeigt werden.

3.2. Darstellung der Kette als Blöcke

Wie bei der Single-Site-DMRG wird auch hier die Kette für die Berechnung des Erwartungswertes in drei Blöcke aufgeteilt:

$$\begin{aligned}
\langle\psi|\hat{H}|\psi\rangle &= \sum_{\boldsymbol{\sigma},\boldsymbol{\sigma'}} \langle\boldsymbol{\sigma}|\underbrace{B^{\sigma_L *}\cdots B^{\sigma_{l+1} *} M^{\sigma_l *} A^{\sigma_{l-1} *}\cdots A^{\sigma_1 *}}_{B^{\sigma_L *}\cdots A^{\sigma_1 *}} \hat{H} \underbrace{A^{\sigma'_1}\cdots A^{\sigma'_{l-1}} M^{\sigma'_l} B^{\sigma'_{l+1}}\cdots B^{\sigma'_L}}_{A^{\sigma'_1}\cdots B^{\sigma'_L}} |\boldsymbol{\sigma'}\rangle \\
&= \sum_{\boldsymbol{\sigma},\boldsymbol{\sigma'}} \sum_{\tilde{\boldsymbol{\sigma}},\tilde{\boldsymbol{\sigma}}'} \langle\boldsymbol{\sigma}|B^{\sigma_L *}\cdots A^{\sigma_1 *}|\tilde{\boldsymbol{\sigma}}\rangle W^{\tilde{\sigma}_1,\tilde{\sigma}'_1}\cdots W^{\tilde{\sigma}_L,\tilde{\sigma}'_L} \langle\tilde{\boldsymbol{\sigma}}'|A^{\sigma'_1}\cdots B^{\sigma'_L}|\boldsymbol{\sigma'}\rangle \\
&= \sum_{\boldsymbol{\sigma},\boldsymbol{\sigma'}} \sum_{\tilde{\boldsymbol{\sigma}},\tilde{\boldsymbol{\sigma}}'} \underbrace{\langle\boldsymbol{\sigma}|\tilde{\boldsymbol{\sigma}}\rangle}_{\delta_{\sigma,\tilde{\sigma}}} \sum_{a_i,a'_i,b_i} B^{\sigma_L *}_{b_{L-1},1}\cdots A^{\sigma_1 *}_{1,a_1} W^{\tilde{\sigma}_1,\tilde{\sigma}'_1}_{1,b_1}\cdots W^{\tilde{\sigma}_L,\tilde{\sigma}'_L}_{b_{L-1},1} A^{\sigma'_1}_{1,a'_1}\cdots B^{\sigma'_L}_{a'_{L-1},1} \underbrace{\langle\tilde{\boldsymbol{\sigma}}'|\boldsymbol{\sigma'}\rangle}_{\delta_{\sigma',\tilde{\sigma}'}} \\
&= \sum_{a_{l-1},a_l} \sum_{a'_{l-1},a'_l} \sum_{b_{l-1},b_l} L^{a_{l-1}}_{a'_{l-1},b_{l-1}} \left(\sum_{\sigma_l,\sigma'_l} M^{\sigma_l *}_{a_{l-1},a_l} W^{\sigma_l,\sigma'_l}_{b_{l-1},b_l} M^{\sigma'_l *}_{a'_{l-1},a'_l} \right) R^{a_l}_{a'_l,b_l}
\end{aligned}$$

Hierbei repräsentiert $L^{a_{l-1}}_{a'_{l-1},b_{l-1}}$ die linke Seite, $R^{a_l}_{a'_l,b_l}$ die rechte Seite und $\sum_{\sigma_l,\sigma'_l} M^{\sigma_l *}_{a_{l-1},a_l} W^{\sigma_l,\sigma'_l}_{b_{l-1},b_l} M^{\sigma'_l *}_{a'_{l-1},a'_l}$ den varriierbaren Platz l.

$$L^{a_{l-1}}_{a'_{l-1},b_{l-1}} = \sum_{\{a_i,a'_i,b_i;i<l-1\}} \left(\sum_{\sigma_1,\sigma'_1} A^{\sigma_1 *}_{1,a_1} W^{\sigma_1,\sigma'_1}_{1,b_1} A^{\sigma'_1}_{1,a'_1} \right) \cdots \left(\sum_{\sigma_{l-1},\sigma'_{l-1}} A^{\sigma_{l-1} *}_{a_{l-2},a_{l-1}} W^{\sigma_{l-1},\sigma'_{l-1}}_{b_{l-2},b_{l-1}} A^{\sigma'_{l-1}}_{a'_{l-2},a'_{l-1}} \right)$$

$$R^{a_l}_{a'_l,b_l} = \sum_{\{a_i,a'_i,b_i;i>l\}} \left(\sum_{\sigma_{l+1},\sigma'_{l+1}} B^{\sigma_{l+1} *}_{a_l,a_{l+1}} W^{\sigma_{l+1},\sigma'_{l+1}}_{b_l,b_{l+1}} B^{\sigma'_{l+1}}_{a'_l,a'_{l+1}} \right) \cdots \left(\sum_{\sigma_L,\sigma'_L} A^{\sigma_L *}_{a_{L-1},1} W^{\sigma_L,\sigma'_L}_{b_{L-1},1} A^{\sigma'_L}_{a'_{L-1},1} \right)$$

Das Skalarprodukt $\langle\psi|\psi\rangle$ eines gemischt-normierten MPS lässt sich auf die Spur der nicht normierten Matrizen zusammenfassen:

$$\begin{aligned}
\langle\psi|\psi\rangle &= \sum_{\boldsymbol{\sigma},\boldsymbol{\sigma'}} \langle\boldsymbol{\sigma}| B^{\sigma_L *} \cdots B^{\sigma_{l+1} *} M^{\sigma_l *} \underbrace{A^{\sigma_{l-1} *} \cdots A^{\sigma_1 *} A^{\sigma'_1} \cdots A^{\sigma'_{l-1}}}_{\mathbb{1}} M^{\sigma'_l} B^{\sigma'_{l+1}} \cdots B^{\sigma'_L} |\boldsymbol{\sigma'}\rangle \\
&= \sum_{\sigma_{l+1},\ldots,\sigma_L} \sum_{a_l} \left(M^{\sigma_l *} M^{\sigma_l} \underbrace{B^{\sigma'_{l+1}} \cdots B^{\sigma'_L} B^{\sigma_L *} \cdots B^{\sigma_{l+1} *}}_{\mathbb{1}} \right)_{a_l,a_l} \\
&= \sum_{\sigma_l} \mathrm{Tr}\left(M^{\sigma_l *} M^{\sigma_l} \right)
\end{aligned}$$

3.3. Effektive Berechnung von L und R

Wie bei der DMRG sollen die einzelnen Seiten wachsen und schrumpfen. Um vorhandene Daten weiterverwenden zu können, ist es notwendig, L und R iterativ aufbauen zu können. Da Matrixprodukte effektiver als Summen bestimmt werden können, ist auch ein Umschreiben in Matrixprodukte sinnvoll. Beides wird wie folgt erreicht:

$$\begin{aligned}
L^{a_{l-1}}_{a'_{l-1},b_{l-1}} &= \sum_{a_{l-2},a'_{l-2},b_{l-2}} L^{a_{l-2}}_{a'_{l-2},b_{l-2}} \left(\sum_{\sigma_{l-1},\sigma'_{l-1}} A^{\sigma_{l-1} *}_{a_{l-2},a_{l-1}} W^{\sigma_{l-1},\sigma'_{l-1}}_{b_{l-2},b_{l-1}} A^{\sigma'_{l-1}}_{a'_{l-2},a'_{l-1}} \right) \\
&= \sum_{\sigma_{l-1},\sigma'_{l-1}} \sum_{a_{l-2},a'_{l-2}} A^{\sigma_{l-1} *}_{a_{l-2},a_{l-1}} \left(\sum_{b_{l-2}} L^{a_{l-2}}_{a'_{l-2},b_{l-2}} W^{\sigma_{l-1},\sigma'_{l-1}}_{b_{l-2},b_{l-1}} \right) A^{\sigma'_{l-1}}_{a'_{l-2},a'_{l-1}} \\
&= \sum_{\sigma_{l-1},\sigma'_{l-1}} \sum_{a_{l-2},a'_{l-2}} A^{\sigma_{l-1} *}_{a_{l-2},a_{l-1}} \left(L^{a_{l-2}} W^{\sigma_{l-1},\sigma'_{l-1}} \right)_{a'_{l-2},b_{l-1}} A^{\sigma'_{l-1}}_{a'_{l-2},a'_{l-1}} \\
&= \sum_{\sigma_{l-1},\sigma'_{l-1}} \sum_{a_{l-2}} A^{\sigma_{l-1} *}_{a_{l-2},a_{l-1}} \left(\sum_{a'_{l-2}} \left(\left(L^{a_{l-2}} W^{\sigma_{l-1},\sigma'_{l-1}} \right)^t \right)_{b_{l-1},a'_{l-2}} A^{\sigma'_{l-1}}_{a'_{l-2},a'_{l-1}} \right) \\
&= \sum_{\sigma_{l-1},\sigma'_{l-1}} \sum_{a_{l-2}} A^{\sigma_{l-1} *}_{a_{l-2},a_{l-1}} \left(\left(L^{a_{l-2}} W^{\sigma_{l-1},\sigma'_{l-1}} \right)^t A^{\sigma'_{l-1}} \right)_{b_{l-1},a'_{l-1}}
\end{aligned}$$

Zu beachten ist hier, dass L^{a_0} ein Skalar mit Wert 1 ist.
Analog lässt sich $R^{a_l}_{a'_l,b_l}$, mit $R^{a_L}=1$, iterativ und mit Matrixmultiplikationen bestimmen:

$$R^{a_l}_{a'_l,b_l} = \sum_{\sigma_{l+1},\sigma'_{l+1}} \sum_{a_{l+1}} B^{\sigma_{l+1}*}_{a_l,a_{l+1}} \left(W^{\sigma_{l+1},\sigma'_{l+1}} \left(B^{\sigma'_{l+1}} R^{a_{l+1}} \right)^t \right)_{b_l,a'_l}$$

3.4. Optimierung eines Platzes

Um die optimalen Matrizen für einen Platz l zu finden, muss die Lagrange-Gleichung

$$L = \langle\psi|\hat{H}|\psi\rangle - \lambda\left(\langle\psi|\psi\rangle - 1\right)$$

nach $M^{\sigma_l}_{a_{l-1},a_l}$ abgeleitet und null gesetzt werden. Hieraus erhält man dann:

$$\begin{aligned} 0 \stackrel{!}{=} \frac{\partial L}{\partial M^{\sigma_L}_{a_{l-1},a_l}} &= \frac{\partial}{\partial M^{\sigma_L}_{a_{l-1},a_l}} \left(\langle\psi|\hat{H}|\psi\rangle\right) - \lambda \frac{\partial}{\partial M^{\sigma_L}_{a_{l-1},a_l}} \left(\langle\psi||\psi\rangle\right) \\ &= \sum_{\sigma'_l} \sum_{a'_{l-1},a'_l} \sum_{b_{l-1},b_l} L^{a_{l-1}}_{a'_{l-1},b_{l-1}} W^{\sigma_l,\sigma'_l}_{b_{l-1},b_l} R^{a_l}_{a'_l,b_l} M^{\sigma'_l}_{a'_{l-1},a'_l} - \lambda \sum_{a'_{l-1},a'_l} M^{\sigma'_l}_{a'_{l-1},a'_l} \end{aligned}$$

Führt man $H_{\mathrm{Eff}(\sigma_l a_{l-1} a_l)(\sigma'_l a'_{l-1} a'_l)} = \sum_{b_{l-1},b_l} L^{a_{l-1}}_{a'_{l-1},b_{l-1}} W^{\sigma_l,\sigma'_l}_{b_{l-1},b_l} R^{a_l}_{a'_l,b_l}$ und $\nu_{\sigma_l a_{l-1} a_l} = M^{\sigma_l}_{a_{l-1},a_l}$ ein, so erhält man ein Eigenwert-Problem:

$$H_{\mathrm{Eff}}\nu - \lambda\nu = 0$$

Dieses kann, auf Grund der geringen Dimension von H_{Eff}, exakt gelöst werden. Der kleinste Eigenwert entspricht dem optimierten Wert für die Grundzustandsenergie E_0 und der dazugehörige Eigenvektor ν_0 liefert die optimierten Matrizen M^{σ_L}.

3.5. Iteratives Optimieren

Um nun den Zustand $|\psi\rangle$ iterativ an allen Plätzen zu verbessern und gleichzeitig viele Daten wiederverwenden zu können, wird folgendermaßen vorgegangen:

1. **Initialisierung**
   Zunächst werden die Matrizen des Zustands mit beliebigen Werten gefüllt und

anschließend rechts-normiert. Während des Rechts-Normierens werden die jeweiligen R-Matrizen erzeugt. Darüber hinaus wird der Hamilton-MPO erzeugt. Der folgende Sweep beginnt nun am Platz $l = 1$ und der MPS hat folgende Form:

$$M^{\sigma_1} B^{\sigma_2} \cdots B^{\sigma_L}$$

2. **Sweep-Schritt nach rechts**
   Als erstes wird H_{Eff} am Platz l, wie im vorherigen Abschnitt beschrieben, gebildet und das Eigenwert-Problem gelöst. Anschließend wird die Matrix M^{σ_l} durch den umgeformten kleinsten Eigenvektor ersetzt (1). Um die links-normierte Seite zu vergrößern, wird die SVD auf das aktualisierte $\tilde{M}^{\sigma_l}$ angewandt und das U als neues A^{σ_l} gespeichert, während das $SV^\dagger$ an die Matrix $B^{\sigma_{l+1}}$ multipliziert wird, womit die rechts-normierte Seite verkleinert wird (2). Im Folgenden ist der Vorgang veranschaulicht:

$$\begin{aligned} A^{\sigma_1} \cdots A^{\sigma_{l-1}} M^{\sigma_l} B^{\sigma_{l+1}} \cdots B^{\sigma_L} &\xrightarrow{(1)} A^{\sigma_1} \cdots A^{\sigma_{l-1}} \tilde{M}^{\sigma_l} B^{\sigma_{l+1}} \cdots B^{\sigma_L} \\ &\xrightarrow{(2)} A^{\sigma_1} \cdots A^{\sigma_l} M^{\sigma_{l+1}} B^{\sigma_{l+2}} \cdots B^{\sigma_L} \end{aligned}$$

   Aus den Singulärwerten (Diagonaleinträge der Matrix S) kann die Dichtematrix $\rho = S^2$ und die „von Neumann“-Entanglement-Entropie $S_{\text{vN}}(|\psi\rangle) = -\text{Tr}\rho \log_2 \rho$ berechnet werden. Anschließend wird zum nächsten Platz gesprungen ($l \to l + 1$).

3. **Sweep nach rechts**
   Um einen kompletten Sweep durchzuführen, wird Schritt 2 wiederholt bis $l = L$ erreicht ist. Der MPS ist anschließend links-normiert und hat folgende Form:

$$A^{\sigma_1} \cdots A^{\sigma_{L-1}} M^{\sigma_L}$$

4. **Sweep nach links**
   Analog zu Schritt 3 wird der Sweep nach links durchgeführt. Er beginnt bei $l = L$ und endet bei $l = 1$.

5. **Sweepen**
   Schließlich werden die Schritte 3 und 4 wiederholt, bis ein Abbruchkriterium (z.B. Konvergenz, max. Anzahl an Sweeps) erreicht ist.

4. Implementation

4.1. DMRG

In diesem Abschnitt werden die einzelnen Klassen beschrieben, die zu dieser Implementation der DMRG gehören. Obwohl es sich bei dem Applet jederzeit um eine Spin-$\frac{1}{2}$-Kette handelt, sind alle Klassen mit einer variablen Anzahl an möglichen lokalen Zuständen implementiert.

4.1.1. Matrix

Die Klasse „Matrix“ implementiert über einen reinen Speicher von 2-dimensional angeordneten Werten hinaus auch Wrapper für folgende jlapack-Funktionen[1][3]:

- DGEMM - Matrixmultiplikation (*mult*), Matrixaddition (*add*), Matrixsubtraktion (*sub*) und Transponieren (*trans*)
- DGESVD - Single value decomposition (*svd*)
- DSYEV - Exakte Diagonalisierung (*Eigen*)

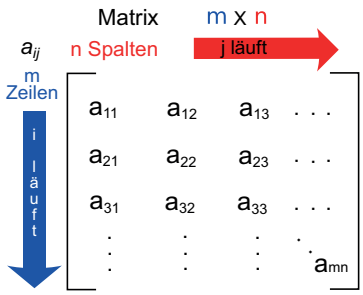

Abb. 4.1.: Anordnung der Matrixelemente[1]

Darüber hinaus ist es möglich, zwei Matrizen horizontal oder vertikal aneinander zu heften (*merge*). Die nötigen Daten um sie wieder zu trennen (*split*), werden innerhalb der Klasse gespeichert, wobei zu beachten ist, dass immer die letzte Heftung aufgetrennt wird.

4.1.2. MatrixMatrix

Diese Klasse bietet die Möglichkeit, dass eine Matrix Matrizen enthält, wie es durch die in 2.3.2 eingeführten $\hat{W}^{[i]}$ notwendig ist. Da auf einer solchen Matrix Operationen wie

[1] Author: HB, Lizenz: Creative Common by-sa, Quelle: `http://commons.wikimedia.org/w/index.php?title=File:Matrice.svg&oldid=39390047&uselang=de`

die SVD oder die exakte Diagonalisierung nicht sinnvoll wären, sind diese auch nicht implementiert.

4.1.3. Mps

In dieser Klasse sind drei zwei-dimensionale Matrix-Arrays enthalten. Eines enthält die M-, A- bzw. B-Matrizen. Die anderen beiden enthalten die R- und L-Matrizen.
Zum Erstellen eines Mps stehen zwei Konstruktoren zur Verfügung. Einer füllt die M-Matrizen zufällig. Der andere Konstruktor erzeugt einen vorgegebenen Zustand. Dieser muss als ein-dimensionales Array, welches die Koeffizienten enthält, vorgegeben werden. Beide Konstruktoren links-normieren anschließend den MPS. Bei diesem Vorgang werden auch die R-Matrizen gefüllt, weshalb es notwendig ist einen zugehörigen MPO anzugeben.

Listing 4.1: Implementation der Berechnung der R-Matrizen in der Klasse Mps

```
356   public void BuildR(Mpo H, int N)
357   {
358     Matrix Eins = new Matrix(1,1); Eins.SetContent(0,0,1);
359     for (int a=0; a<this.A[N+1][0].GetSize()[0];a++)
360     {
361       this.R[N][a] = new Matrix(H.W[N+1][0][0].GetSize()[0], this.A[N
            +1][0].GetSize()[0]);
362       for (int j=0; j<this.states; j++)
363         for (int k=0; k<this.states; k++)
364           for (int l=0; l<this.A[N+1][0].GetSize()[1]; l++)
365             if (N>this.N-2)
366               for (int m=0; m<this.A[N+1][j].GetSize()[0]; m++)
367                 for (int o=0; o<H.W[N+1][j][k].GetSize()[0]; o++)
368                   this.R[N][a].SetContent(o,m, this.R[N][a].GetContent(o,m
                        )+this.A[N+1][j].GetContent(0,m)*H.W[N+1][j][k].
                        GetContent(0,o)*this.A[N+1][k].GetContent(0,a));
369             else
370               this.R[N][a] = new Matrix(this.R[N][a].add(H.W[N+1][j][k].
                      mult(this.R[N+1][l].trans()).mult(this.A[N+1][k].trans())
                      .mult(this.A[N+1][j].GetContent(l,a)).trans()));
371     }
372   }
```

Das Übergeben des MPO ist ebenfalls notwendig, wenn ein „Sweep-Step“ gemacht wird, da auch hier die L- und R-Matrizen neu gefüllt werden.

Listing 4.2: Implementation des einzelnen Sweep-Steps nach links in der Klasse Mps

```
275   public Matrix DoLeftSweepStep(int j, Mpo op)
276   {
277     Matrix AAllStates = new Matrix(this.A[j][0]);
278     for (int k=1; k<this.states; k++)
279       AAllStates.merge(this.A[j][k], false);
280     Matrix u=new Matrix();
281     Matrix vt=new Matrix();
282     Matrix s=AAllStates.svd(u, vt);
283     for (int i=vt.Merges.length;i>=0;i  )
284       this.A[j][i]=vt.split();
285     for (int i=0; i<this.states; i++)
286       this.A[j 1][i]=(this.A[j 1][i].mult(u).mult(s));
287     this.BuildR(op, j 1);
288     return s;
289   }
```

Auch das in 2.2 beschriebene Skalarprodukt ist in dieser Klasse implementiert:

Listing 4.3: Implementation des Skalarprodukts in der Klasse Mps

```
331   public double ScalarProd(Mps b)
332   {
333     if (this.N != b.N)
334       return 0.0;
335     Matrix out = new Matrix();
336     Matrix pre_out = new Matrix();
337     pre_out = this.A[0][0].trans().mult(b.A[0][0]);
338     for (int s=1; s<this.states;s++)
339       pre_out = pre_out.add(this.A[0][s].trans().mult(b.A[0][s]));
340     out = new Matrix(pre_out);
341     for (int i=1;i<this.N;i++)
342     {
343       pre_out = this.A[i][0].trans().mult(out).mult(b.A[i][0]);
344       for ( int s=1; s<this.states;s++)
345         pre_out = pre_out.add(this.A[i][s].trans().mult(out).mult(b.A[i][s
              ]));
346       out = new Matrix(pre_out);
347     }
348     return out.GetContent(0,0);
349   }
```

4.1.4. Mpo

Um die MPO-Matrizen zu speichern, verwendet die Klasse Mpo ein drei-dimensionales Matrizen-Array. Beim Erzeugen einer Instanz dieser Klasse gibt es die Möglichkeit, ein „MatrixMatrix“-Array mit alles $\hat{W}^{[i]}$ zu übergeben. Alternativ kann auch nur eine einzelne „MatrixMatrix“ übergeben werden; hierbei wird $\hat{W}^{[1]}$ aus der letzten Zeile und $\hat{W}^{[L]}$ aus der ersten Spalte erzeugt.
Die Anwendung eines Mpo-Objektes auf ein Mps-Objekt ist, wie in 2.3.1 beschrieben, durch die Funktion *mult* implementiert:

Listing 4.4: Implementation der Anwendung eines MPO auf ein MPS

```
136   public Mps mult(Mps b)
137   {
138     Mps out = new Mps(this.N, b.d*this.W[0][0][0].GetSize()[1], b.d_max*
          this.W[this.N/2][0][0].GetSize()[1], this.states);
139     for (int l=0; l<this.N; l++)
140       for (int i=0; i<this.states; i++)
141         for (int j=0; j<this.states; j++)
142           for (int m=0; m<b.A[l][i].GetSize()[0];m++)
143             for (int o=0; o<b.A[l][i].GetSize()[1];o++)
144               for (int q=0; q<this.W[l][i][j].GetSize()[0];q++)
145                 for (int r=0; r<this.W[l][i][j].GetSize()[1];r++)
146                   out.A[l][i].SetContent(r*b.A[l][i].GetSize()[1]+o, q*b.A
                        [l][i].GetSize()[0]+m, out.A[l][i].GetContent(r*b.A[l
                        ][i].GetSize()[1]+o, q*b.A[l][i].GetSize()[0]+m)+this
                        .W[l][i][j].GetContent(r, q)*b.A[l][j].GetContent(o,
                        m));
147     return out;
148   }
```

4.1.5. Dmrg

Die Klasse „Dmrg“ beinhaltet ein Mps-Objekt und ein Mpo-Objekt und stellt sowohl den Algorithmus, wie er in Kapitel 3 beschrieben ist, als auch die Schnittstelle zur GUI (Graphical-User-Interface) dar. Die Kernaufgaben übernehmen hierbei vier Funktionen, die im Folgenden dargestellt werden:

- **DoNextStep**
  Diese Funktion ist die einzige, neben dem Konstruktor, die von außen aufgerufen wird. Sie kümmert sich darum, dass immer in die richtige Richtung gesweept wird.

- **DoRightStep**
  Hier wird ein kompletter Sweep-Step nach rechts vorgenommen. Dazu muss zunächst der in 3.4 eingeführte effektive Hamiltonien H_{Eff} erstellt werden (Zeile 125). Dieser muss dann diagonalisiert werden (Zeile 126). Anschließend wird der Eigenvektor in die MPS-Matrix des bearbeiteten Platzes eingeordnet (Zeilen 128 bis 135). Mit dem Rechts-Normieren in Zeile 137 ist der eigentliche Sweep-Step abgeschlossen. Zuletzt werden dann noch die Ergebnisse der verschiedenen Operationen gespeichert (Zeile 138 bis 142), damit sie später ausgelesen werden können.

Listing 4.5: Kompletter Sweep-Step nach rechts

```
123   public void DoRightStep(int N)
124   {
125     Matrix H = this.BuildHeff(N);
126     double[] EigenValue = H.Eigen();
127     Matrix[] M = new Matrix[this.state.states];
128     for (int i=0; i<this.state.states; i++)
129     {
130       M[i] = new Matrix(this.state.A[N][0].GetSize()[1], this.state.A
              [N][0].GetSize()[0]);
131       for (int m=0; m<this.state.A[N][0].GetSize()[0]; m++)
132         for (int o=0; o<this.state.A[N][0].GetSize()[1]; o++)
133           M[i].SetContent(o, m, H.GetContent(0, (i*this.state.A[N
                ][0].GetSize()[0]*this.state.A[N][0].GetSize()[1])+(m*
                this.state.A[N][0].GetSize()[1])+o));
134       this.state.A[N][i] = new Matrix(M[i]);
135     }
136     this.CalcSxSz();
137     Matrix s = this.state.DoRightSweepStep(N, this.operator);
138     s = s.mult(s);
139     this.CurrentDm = new double[s.GetSize()[0]];
140     for (int i = 0; i<this.CurrentDm.length; i++)
141       this.CurrentDm[i] = s.GetContent(i,i);
142     this.CurrentEnergy = EigenValue[0];
143   }
```

- **DoLeftStep**
  Diese Funktion entspricht der Funktion DoRightStep mit der Ausnahme, dass links-normalisiert wird.

- **BuildHeff**
  Um H_{Eff} zu bilden, wird zunächst ein Zwischenspeicher $LWR^{[\sigma][\sigma'][a_l][a_{l-1}]}$ erstellt. Hierzu kann man mehrere Summen in Matrixprodukte umformulieren, wie in 3.3 gezeigt. Anschließend werden dann alle Werte einzeln in H_{Eff} einsortiert.

Listing 4.6: Erzeugen von H_{Eff}

```
227   public Matrix BuildHeff(int N)
228   {
229     int Size = this.state.states*this.state.A[N][0].GetSize()[0]*this
          .state.A[N][0].GetSize()[1];
230     Matrix H = new Matrix(Size, Size);
231     Matrix[][][][] LWR = new Matrix[this.state.states][this.state.
          states][this.state.A[N][0].GetSize()[0]][this.state.A[N][0].
          GetSize()[1]];
232     for (int i=0; i<this.state.states; i++)
233       for (int j=0; j<this.state.states; j++)
234         for (int m=0; m<this.state.A[N][0].GetSize()[0]; m++)
235           for (int o=0; o<this.state.A[N][0].GetSize()[1]; o++)
236             if (N-1 < 0)
237               LWR[i][j][m][o] = new Matrix((this.operator.W[N][i][j].
                    mult(this.state.R[N][o].trans())));
238             else
239               LWR[i][j][m][o] = new Matrix(this.state.L[N-1][m].mult(
                    this.operator.W[N][i][j].mult(this.state.R[N][o].
                    trans())));
240
241     for (int i=0; i<this.state.states; i++)
242       for (int j=0; j<this.state.states; j++)
243         for (int m=0; m<this.state.A[N][0].GetSize()[0]; m++)
244           for (int n=0; n<this.state.A[N][0].GetSize()[0]; n++)
245             for (int o=0; o<this.state.A[N][0].GetSize()[1]; o++)
246               for (int p=0; p<this.state.A[N][0].GetSize()[1]; p++)
247                 H.SetContent((j*this.state.A[N][0].GetSize()[0]*this.
                      state.A[N][0].GetSize()[1])+(n*this.state.A[N][0].
                      GetSize()[1])+p, (i*this.state.A[N][0].GetSize()
                      [0]*this.state.A[N][0].GetSize()[1])+(m*this.state
                      .A[N][0].GetSize()[1])+o, LWR[i][j][m][o].
                      GetContent(p,n));
248     return H;
249   }
```

4.2. Applet

In diesem Abschnitt werden die Klassen und Bibliotheken beschrieben, die zum Erstellen der GUI notwendig sind.

4.2.1. DmrgCollector

Diese Klasse, die von der jchart2D-Klasse „ADataCollector“ abgeleitet ist, fügt den einzelnen Charts Datenpunkte hinzu. Damit dies, inklusive der Berechnungen, parallel zur Bedienung des Applets möglich ist, implementiert diese Klasse das Interface „Runnable“[2], wodurch diese Klasse in einem eigenen Thread ausgeführt wird.
Die gesamten Berechnungen werden aus der Funktion „collectData“ heraus gestartet bzw. in dieser durchgeführt. Zunächst wird überprüft, ob das Sweep-Limit erreicht wurde:

Listing 4.7: Stop-Kriterium

```
24      // Maximale Anzahl an Sweeps erreicht? ==> Stop
25      if (this.visuRef.DmrgRef.CurrentSweep == this.visuRef.Sweeps)
26        this.visuRef.stopData();
```

Anschließend wir der eigentliche Sweep ausgeführt und der neu berechnete Energiepunkt direkt in den Energie-Chart eingefügt:

Listing 4.8: Ausführen des Sweeps und Einfügen des neuen Energiepunktes

```
28      // Erzeuge neue Daten
29      int y = this.visuRef.DmrgRef.DoNextStep();
30
31      // Erzeuge den neuen Energie Punkt
32      TracePoint2D Point = new TracePoint2D(y, this.visuRef.DmrgRef.
            CurrentEnergy);
33      this.visuRef.getEnergyTrace().addPoint(Point);
```

Im Folgenden wird die Spur der Dichtematrix aktualisiert:

Listing 4.9: Aktualisierung der Spur der Dichtematrix

```
35      // Aktualisiere die Spur der Dichtematrix
36      this.visuRef.getDmTrace().removeAllPoints();
37      for (int i = 0; i<this.visuRef.DmrgRef.CurrentDm.length; i++)
```

[2]Für nähere Informationen: `http://download.oracle.com/javase/1.5.0/docs/api/java/lang/Runnable.html`

```
38        this.visuRef.getDmTrace().addPoint(i, this.visuRef.DmrgRef.CurrentDm
              [i]);
```

Danach werden die Erwartungswerte bestimmt. Hierzu wird eine Operator-Klasse verwendet, die lediglich Single-Site-Operatoren unterstützt.

Listing 4.10: Erwartungswerte aktualisieren

```
40    // Aktualisieren der Erwartungswerte für Sx und Sz
41    for (int i=0; i<this.visuRef.Site; i++)
42    {
43      this.visuRef.getErwSzTrace().addPoint(i, this.visuRef.DmrgRef.
            CurrentSzExV[i]);
44      this.visuRef.getErwSxTrace().addPoint(i, this.visuRef.DmrgRef.
            CurrentSxExV[i]);
45    }
```

Anschließend wird die Entanglement-Entropie berechnet.

Listing 4.11: Berechnung der Entanglement-Entropie

```
47    // Bestimme die entanglement entropy
48    double enen = 0.0;
49    for (int i = 0; i<this.visuRef.DmrgRef.CurrentDm.length; i++)
50      enen = enen + this.visuRef.DmrgRef.CurrentDm[i]*(Math.log(this.
            visuRef.DmrgRef.CurrentDm[i])/Math.log(2.0));
51    this.visuRef.getEeTrace().addPoint(y+0.5*(this.visuRef.DmrgRef.
          Direction==false?1:1), enen);
```

Der Rest der Funktion erledigt die Textausgabe und die Positionierung des Positionsbalkens.

4.2.2. visu

Die Klasse *visu* beinhaltet die *main*-Funktion und die gesamte grafische Oberfläche. Das bedeutet, dass alle Objekte instanziiert werden, die später die Ausgabe übernehmen. Darüber hinaus werden in dieser Klasse alle Matrizen und Operatoren definiert und gespeichert. Dies wiederum führt dazu, dass sämtliche Einstellmöglichkeiten in dieser Klasse zu finden sind.

5. Benutzung des Applets

In diesem Kapitel soll die Benutzung des Applets beschrieben werden. Hierzu wird auf alle Einstellmöglichkeiten eingegangen. Einige Testfälle werden dann im folgenden Kapitel erörtert.

5.1. Screenshots

Um einmal ein komplettes Bild des Applets zu zeigen, sind hier zwei Screenshots abgebildet. Im Folgenden wird aufgrund der schlechten Lesbarkeit auf Screenshots verzichtet. Stattdessen werden die jeweiligen Diagramme direkt eingebunden. Dies ist eine von vielen Fähigkeiten der verwendeten Diagramm-Bibliothek jchart2D.

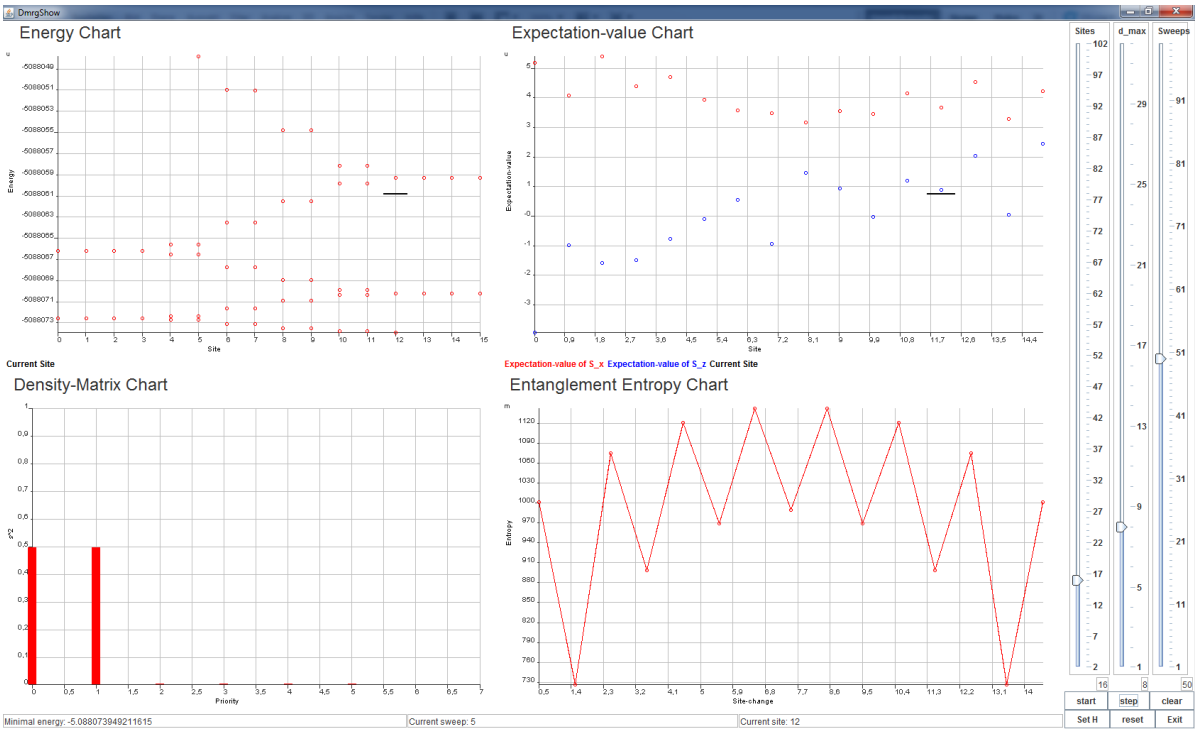

Abb. 5.1.: Screenshot des Applets mit $H = \frac{1}{2}(S^+S^- + S^-S^+) + 0.1S^zS^z$

Abb. 5.2.: Screenshot des Applets mit eingestelltem $H = \frac{1}{2}(S^+S^- + S^-S^+) + 0.1S^zS^z$

5.2. Hauptfenster

Im Hauptfenster des Applets (s. Abb. 5.1) werden vier Diagramme dargestellt. Diese geben, von oben links nach unten rechts, den jeweiligen Wert der Grundzustandsenergie, die Erwartungswerte von S^x und S^z, die Einträge der Dichtematrix sowie die Entanglement-Entropie an. Da die jchart-2d-Bibliothek sehr viele Einstellmöglichkeiten bietet, sei hier auf die Dokumentation[1] verwiesen. Die ersten beiden Diagramme enthalten außerdem einen schwarzen Balken, der anzeigt, an welchem Platz die Kette gerade optimiert wird. Diese Information und der geringste bisher errechnete Grundzustandsenergiewert können ebenfalls am unteren Fensterrand abgelesen werden.
Die rechte Seite dient der Steuerung des Applets. Hier können, von links nach rechts, die Länge der Kette, die maximale Größe der MPS-Matrizen und die maximale Anzahl der Sweeps eingestellt werden. Der jeweilige Wert kann unterhalb der Slider abgelesen werden.
Rechts unten befinden sich sechs Buttons. Mit dem Ersten (*start* bzw. *stop*) startet bzw. unterbricht man die Berechnung, mit dem Zweiten (*step*) kann man die Berechnung um einen Platz voranbringen. Der letzte Button in der oberen Reihe dient dem Löschen der Diagramme. Die untere Reihe beginnt mit dem Button *Set H*, dieser öffnet ein weiteres Fenster, auf das weiter unten eingegangen wird. Der zweite Button (*reset*) löscht die bisherige Berechnung. Mit dem *exit*-Button kann das Applet, außerhalb des Browsers, beendet werden.

5.3. Hamilton-Operator-Einstellung

In dem Fenster, das erscheint, wenn man im Hauptfenster *Set H* drückt, kann man den Hamilton-Operator konfigurieren (s. Abb. 5.2). Hierbei kann ausgewählt werden, ob die

[1] jchart2d-Dokumentation: `http://jchart2d.sourceforge.net/docs.shtml`

Operatoren S^+ und S^- oder die Operatoren S^x und S^y verwendet werden. Diese beiden Möglichkeiten, die formal identisch sind, werden im Quellcode unterschiedlich dargestellt; darüber hinaus können in letzterem Fall die beiden Terme unterschiedlich gewichtet werden. Die Auswahl erfolgt mittels der Radio-Buttons am Anfang der jeweiligen Zeile. Im folgenden Kapitel soll folgende Notation gelten:

$$H = \frac{J}{2}(S^+S^- + S^-S^+) + J^zS^zS^z + h^zS^z + h^xS^x$$

beziehungsweise:

$$H = J^xS^xS^x + J^yS^yS^y + J^zS^zS^z + h^zS^z + h^xS^x$$

6. Testfälle

Im Folgenden werden die Ergebnisse des Applets für verschiedener Modelle mit zum Teil bekannten analytischen Lösungen verglichen. Um die Werte reproduzieren zu können, werden die jeweiligen Einstellung mit angegeben. Die aus dem Applet direkt erzeugten Ergebnisse sind im Anhang A abgebildet.

6.1. Ising-Modell (ohne Magnetfeld)

Das Ising-Modell ist ein nach Ernst Ising benanntes Modell zur Beschreibung von Ferro- und Antiferromagnetismus in Festkörpern. Es beschreibt die Wechselwirkung benachbarter Spins. Diese Spins stellen die Ursache für den Anti- bzw. Ferromagnetismus dar. Es wird ausgenutzt, dass für die magnetischen Momente keine Raumrichtung ausgezeichnet ist und daher eine beliebige Richtung gewählt werden kann[13]. Üblicherweise wird die z-Achse ausgewählt[8]. Im isotropen, d. h. magnetfeldfreien, ein-dimensionalen Fall existiert kein Phasenübergang, wie die exakte Lösung zeigt[5]. Im zweidimensionalen Fall existieren dahingegen Phasenübergänge[9][16]. Für höhere Dimensionen gibt es bisher keine exakten Lösungen, allerdings existiert ein Beweis für die Existenz von Phasenübergängen[10].

Der Hamilton-Operator $\hat{H}$ für das Ising-Modell ohne Magnetfeld mit konstanter Nächster-Nachbar-Wechselwirkung und offenen Randbedingungen lautet:

$$\hat{H} = J^z \sum_{i=1}^{N-1} \hat{S}_i^z \hat{S}_{i+1}^z$$

Die Grundzustandsenergie beträgt für eine Kette mit N Plätzen:

$$E_0 = -\frac{(N-1) \cdot |J^z|}{4}$$

Dies kann, beispielhaft für eine Kette mit 8 Plätzen, in Abb. A.1 betrachtet werden.[1]

6.1.1. Antiferromagnet

Ist $J > 0$ so beschreibt der Hamilton-Operator einen Antiferromagneten. Das bedeutet, dass sich alle Spins im Grundzustand entgegengesetzt zueinander ausrichten.

$$|\psi\rangle = |\uparrow\downarrow\uparrow\ldots\rangle \quad \text{oder} \quad |\psi\rangle = |\downarrow\uparrow\downarrow\ldots\rangle$$

Dies wird als Néelzustand bezeichnet. Der Grundzustand kann eine beliebige Linearkombination dieser beiden Zustände sein, in der DMRG wird allerdings immer einer dieser beiden ausgewählt. Die Auswahl einer der beiden Möglichkeiten erfolgt in der DMRG nicht vorhersehbar. Es kann auch sprunghaft zwischen beiden gewechselt werden. Der Erwartungswert von $\hat{S}_i^z$ am Platz i lautet:

$$\langle \hat{S}_i^z \rangle = \pm\frac{1}{2}$$

Da beide Erwartungswerte abwechselnd auftreten, lautet der Erwartungswert für $\hat{S}^z$:

$$\langle \hat{S}^z \rangle = \begin{cases} 0, & \text{falls } N \text{ gerade} \\ \pm\frac{1}{2}, & \text{falls } N \text{ ungerade} \end{cases}$$

Ein Beispiel hierfür kann in den Abb. A.2 - A.3 betrachtet werden.[2]

6.1.2. Ferromagnet

Ist $J < 0$ handelt es sich um einen Ferromagneten. Bei diesem sind die Spins im Grundzustand parallel zueinander ausgerichtet:

$$|\psi\rangle = |\uparrow\uparrow\uparrow\ldots\rangle \quad \text{oder} \quad |\psi\rangle = |\downarrow\downarrow\downarrow\ldots\rangle$$

Auch hier ist der Grundzustand eine Linearkombination der beiden Zustände, wobei die DMRG wiederum unvorhersehbar einen auswählt. Der Erwartungswert von $\hat{S}_i^z$ entspricht dem im antiferromagnetischen Fall. Der Erwartungswert von $\hat{S}^z$ lautet:

$$\langle \hat{S}^z \rangle = \pm\frac{N}{2}$$

[1] Die verwendeten Einstellungen lauten: $J = 0$, $J^z = 1$, $h^z = 0$, $h^x = 0$.
[2] Die verwendeten Einstellungen lauten: $J = 0$, $J^z = 1$, $h^z = 0$, $h^x = 0$.

Ein Beispiel für den hier angeführten Wert kann in der Abb. A.4 betrachtet werden.[3]

6.2. Ising-Modell (mit longitudinalem Magnetfeld)

Um den Hamilton-Operator $\hat{H}$ des Ising-Modells um ein longitudinales Magnetfeld zu erweitern, wird ein zusätzlicher Term in z-Richtung hinzugefügt:

$$\hat{H} = J^z \sum_i \hat{S}_i^z \hat{S}_{i+1}^z + h^z \hat{S}_i^z$$

Hierbei gibt h^z die Stärke des Magnetfeldes an. Der Antiferromagnet zeigt einen Phasenübergang zum Ferromagneten bei $h_c^z = J^z$. Der Ferromagnet hingegen weist keinen Phasenübergang auf.

6.2.1. Antiferromagnet

Gilt $h^z \ll J^z$, so gibt das Vorzeichen des Magnetfelds h^z im antiferromagnetischen Fall für eine Kette mit einer ungeraden Anzahl an Plätzen die Ausrichtung der äußeren beiden Spins vor.

$$|\psi\rangle = \begin{cases} |\uparrow\downarrow\uparrow\cdots\downarrow\uparrow\rangle, & \text{falls } h^z < 0 \\ |\downarrow\uparrow\downarrow\cdots\uparrow\downarrow\rangle, & \text{falls } h^z > 0 \end{cases}$$

Siehe hierzu Abb. A.5 und Abb. A.6.[4]
Für Ketten mit gerader Anzahl an Plätzen beträgt die Grundzustandsenergie:

$$E_0 = \begin{cases} -\frac{N-1}{4}|J^z|, & \text{falls } h^z \leq \frac{J^z}{2}\text{, Antiferromagnet: } |\uparrow\downarrow\uparrow\cdots\uparrow\downarrow\rangle \\ -\frac{N-3}{4}|J^z|-|h^z|, & \text{falls } \frac{J^z}{2} < h^z \leq J^z\text{, defekter Antiferromagnet: } |\uparrow\downarrow\uparrow\cdots\uparrow\uparrow\rangle \\ \frac{N-1}{4}|J^z|-\frac{N}{2}|h^z|, & \text{falls } J^z < h^z\text{, Ferromagnet: } |\uparrow\uparrow\uparrow\cdots\uparrow\uparrow\rangle \end{cases}$$

Im Fall einer ungeraden Anzahl an Kettenplätzen beträgt die Grundzustandsenergie:

$$E_0 = \begin{cases} -\frac{N-1}{4}|J^z|-\frac{|h^z|}{2}, & \text{falls } h^z \leq J^z\text{, Antiferromagnet: } |\uparrow\downarrow\uparrow\cdots\uparrow\rangle \\ \frac{N-1}{4}|J^z|-\frac{N}{2}|h^z|, & \text{falls } J^z < h^z\text{, Ferromagnet: } |\uparrow\uparrow\uparrow\cdots\uparrow\uparrow\rangle \end{cases}$$

Diese Fallunterscheidungen sind damit zu erklären, dass ab einer Feldstärke $h^z \leq h_c^z$ das externe Magnetfeld stärker als die inneren Wechselwirkungen ist und es somit energetisch günstiger ist, dem Magnetfeld zu folgen. Der mittlere Fall für Ketten mit gerader

[3] Die verwendeten Einstellungen lauten: $J = 0$, $J^z = -1$, $h^z = 0$, $h^x = 0$.
[4] Die verwendeten Einstellungen lauten: $J = 0$, $J^z = 1$, $h^z = -0.1$ bzw. $h^z = 0.1$, $h^x = 0$.

Anzahl an Plätzen ist den offenen Randbedingungen geschuldet, da hierdurch zunächst nur der Rand-Spin am Magnetfeld ausgerichtet wird. Dies ist daran zu erkennen, dass es sich nur um eine kleine Veränderung handelt. In Abb. 6.1 sind diese zwei bzw. drei Bereiche gut zu erkennen.

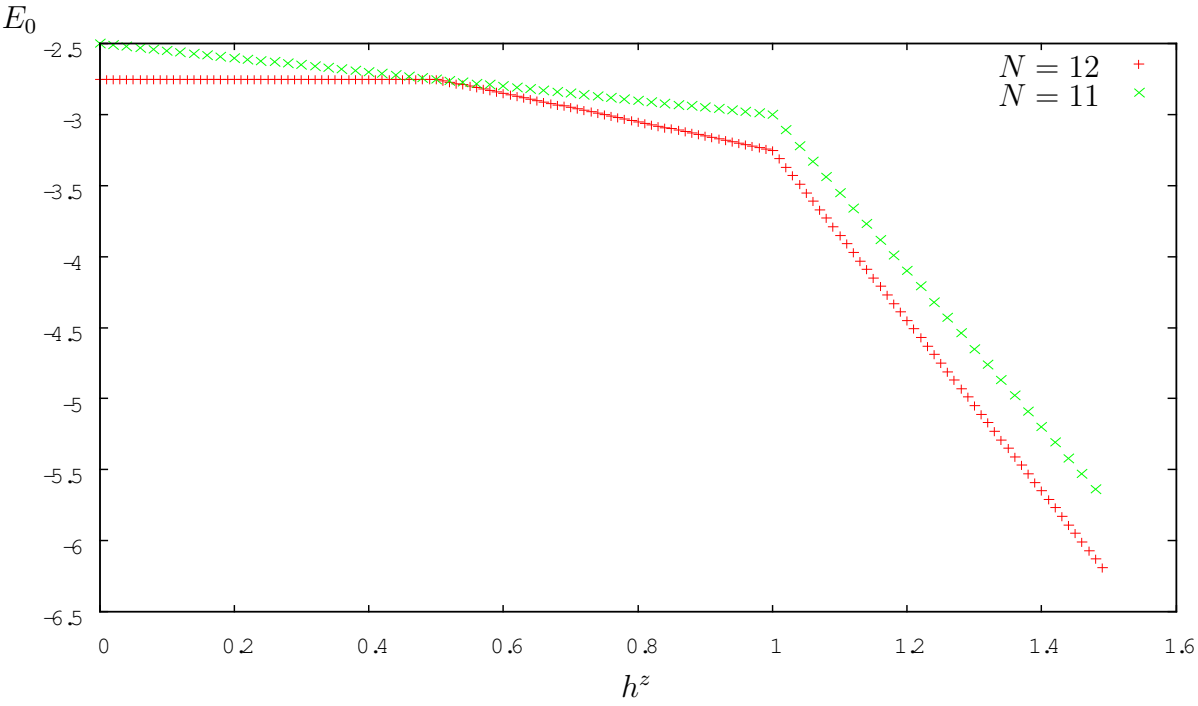

Abb. 6.1.: Grundzustandsenergie in Abhängigkeit zum angelegten Magnetfeld für eine Kette mit gerader bzw. ungerader Anzahl an Plätzen.

Der Erwartungswert $\langle S^z \rangle$ ist für Ketten mit gerader Anzahl an Plätzen wiederum dreigeteilt. Auch hier ist dies den offenen Randbedingungen geschuldet. Dass der Erwartungswert von S^z auch für Ketten mit ungerader Anzahl an Plätzen drei unterschiedliche Werte annehmen kann, folgt aus der Wahlfreiheit des Erwartungswertes ohne Magnetfeld, wie sie in 6.1.1 beschrieben ist.

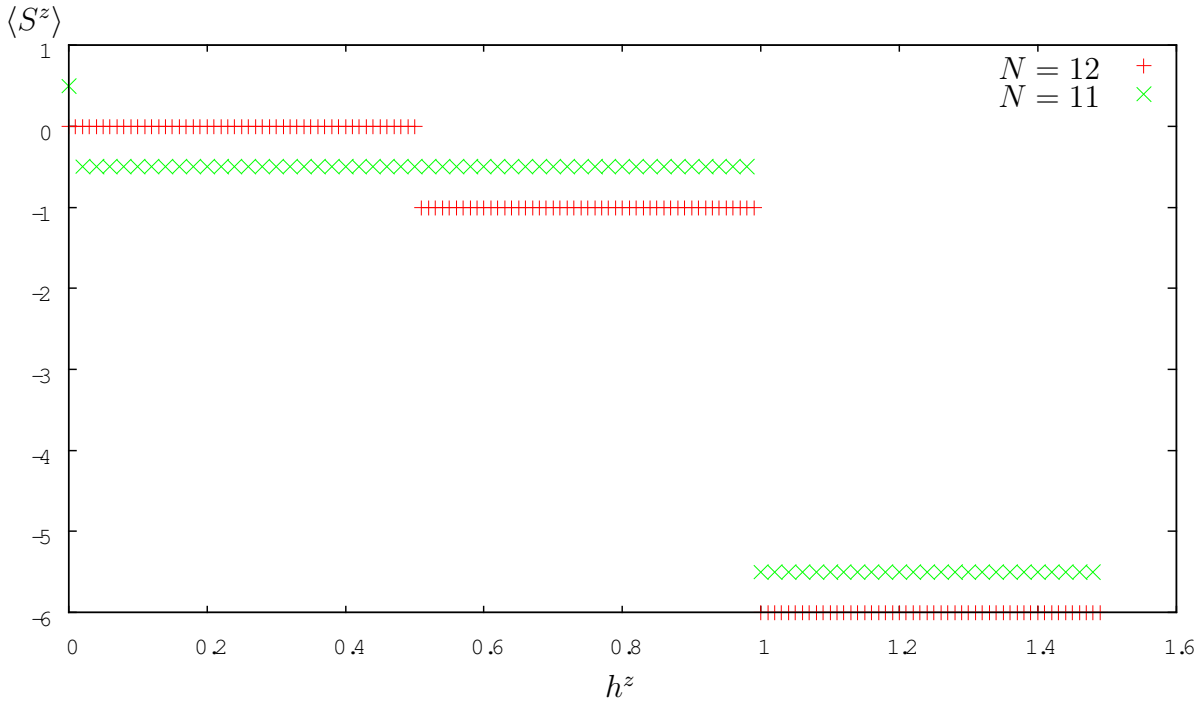

Abb. 6.2.: Erwartungswert von S^z in Abhängigkeit zum angelegten Magnetfeld für eine Kette mit gerader bzw. ungerader Anzahl an Plätzen.

6.2.2. Ferromagnet

In diesem Fall gibt das Vorzeichen von h^z immer die Ausrichtung der Spins an.

$$|\psi\rangle = \begin{cases} |\uparrow\uparrow\uparrow\rangle, & \text{falls } h^z < 0 \\ |\downarrow\downarrow\downarrow\rangle, & \text{falls } h^z > 0 \end{cases}$$

Der Erwartungswert von S^z wird ebenfalls durch h^z festgelegt.

$$\langle\psi|\hat{S}^z|\psi\rangle = \begin{cases} \frac{N}{2}, & \text{falls } h^z < 0 \\ -\frac{N}{2}, & \text{falls } h^z > 0 \end{cases}$$

Die Grundzustandsenergie beträgt:

$$E_0 = -\frac{(N-1)\cdot|J|}{4} - |h^z|\frac{N}{2}$$

Da sich die Abbildungen nur minimal von denen im ferromagnetischen Fall ohne Magnetfeld unterscheiden, wird an dieser Stelle auf weitere Abbildungen verzichtet.

6.3. Ising-Modell (mit transversalem Magnetfeld)

Der Hamilton-Operator für das Ising-Modell mit transversalem Magnetfeld lautet:

$$\hat{H} = J^x \sum_i \hat{S}_i^x \hat{S}_{i+1}^x + h^z \hat{S}_i^z$$

Dieses System wurde exakt gelöst[11] und hat einen Phasenübergang bei $h^z = J^x/2$. Um diesen Phasenübergang veranschaulichen zu können, werden systemgrößenbereinigte Werte für die Grundzustandsenergie und die Erwartungswerte verwendet (finite-size-scaling). Um diese zu bestimmen, werden zunächst E_0/N, $\langle S^x \rangle/N$ und $\langle S^z \rangle/N$ gegen $1/N$ für verschiedene h^z aufgetragen. Hieraus kann dann, mittels linearer Regression, ein größenbereinigtes $E_0(h^z)$, $\langle S^x \rangle(h^z)$ und $\langle S^x \rangle(h^z)$ bestimmt werden. Diese größenbereinigten Werte werden in Abb. 6.4 - 6.5 gegen h^z aufgetragen. Hierbei liefert Pfeuty[11] analytische Lösungen für die Grundzustandsenergie und den Erwartungswert von S^z in Abhängigkeit von $\lambda = J/2h^z$:

$$\frac{E_0}{J^x N}(\lambda) = \frac{2}{\pi}(1+\lambda) E\left(\frac{\pi}{2}, \Theta\right)$$

Hierbei ist $\Theta^2 = \frac{4\lambda}{(1+\lambda)^2}$ und $E\left(\frac{\pi}{2}, \Theta\right)$ ein elliptisches Integral zweiter Art.

$$\langle \hat{S}^z \rangle(\lambda) = \frac{1}{2} G(0, \lambda) \quad \text{mit } G(n, \lambda) = L(n, \lambda) + \lambda L(n+1, \lambda)$$

Hier ist $G(n, \lambda) = L(n, \lambda) + \lambda L(n+1, \lambda)$ wobei

$$L(n, \lambda) = 1/\pi \int_0^\pi dk \Lambda_k^{-1} cos(kn)$$

mit $\Lambda_k^2 = 1 + \lambda^2 + \lambda \cos(k)$ proportional zu einem elliptischen Integral erster Art ist. In den Abbildungen ist zu erkennen, dass die numerischen Ergebnisse den analytischen entsprechen.

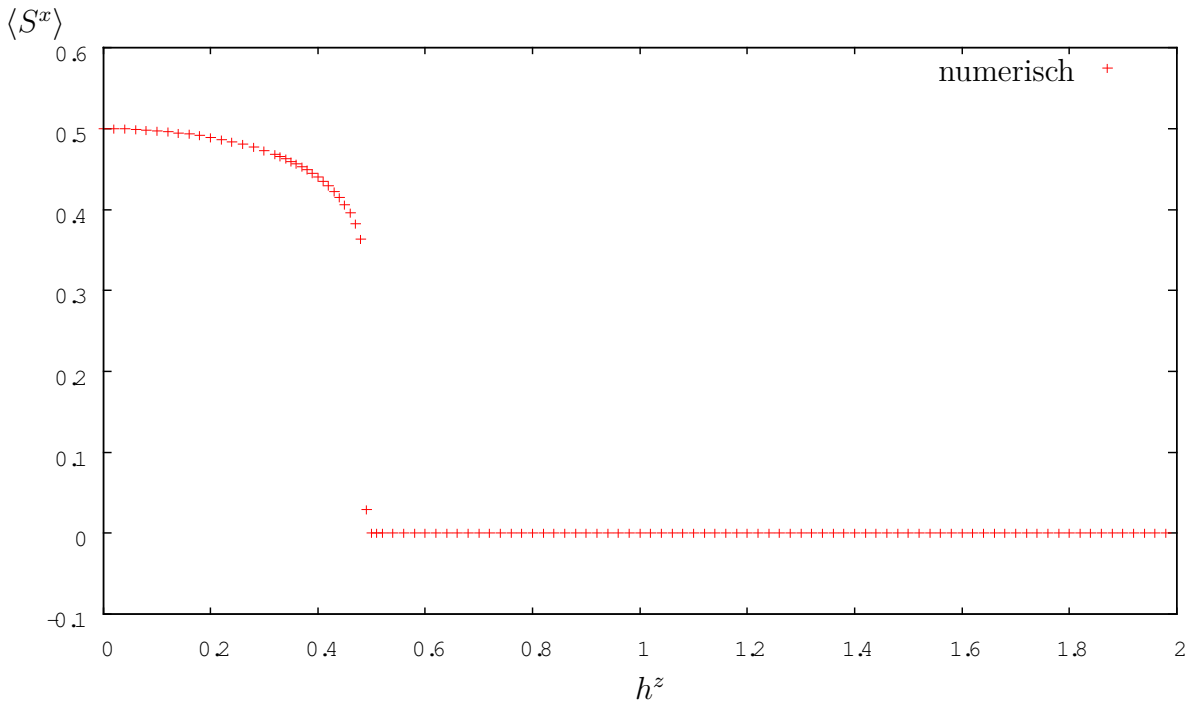

Abb. 6.3.: Größenbereinigte Erwartungswerte von S^x in Abhängigkeit zum angelegten Magnetfeld.

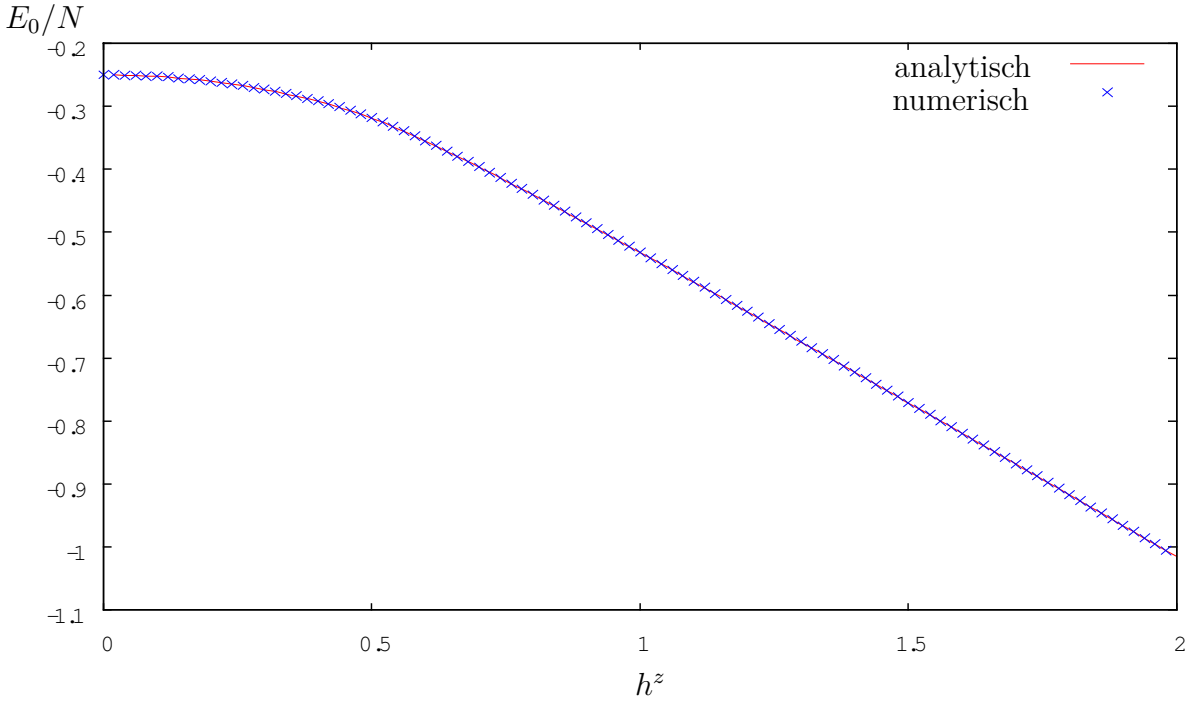

Abb. 6.4.: Größenbereinigte Grundzustandsenergie in Abhängigkeit zum angelegten Magnetfeld.

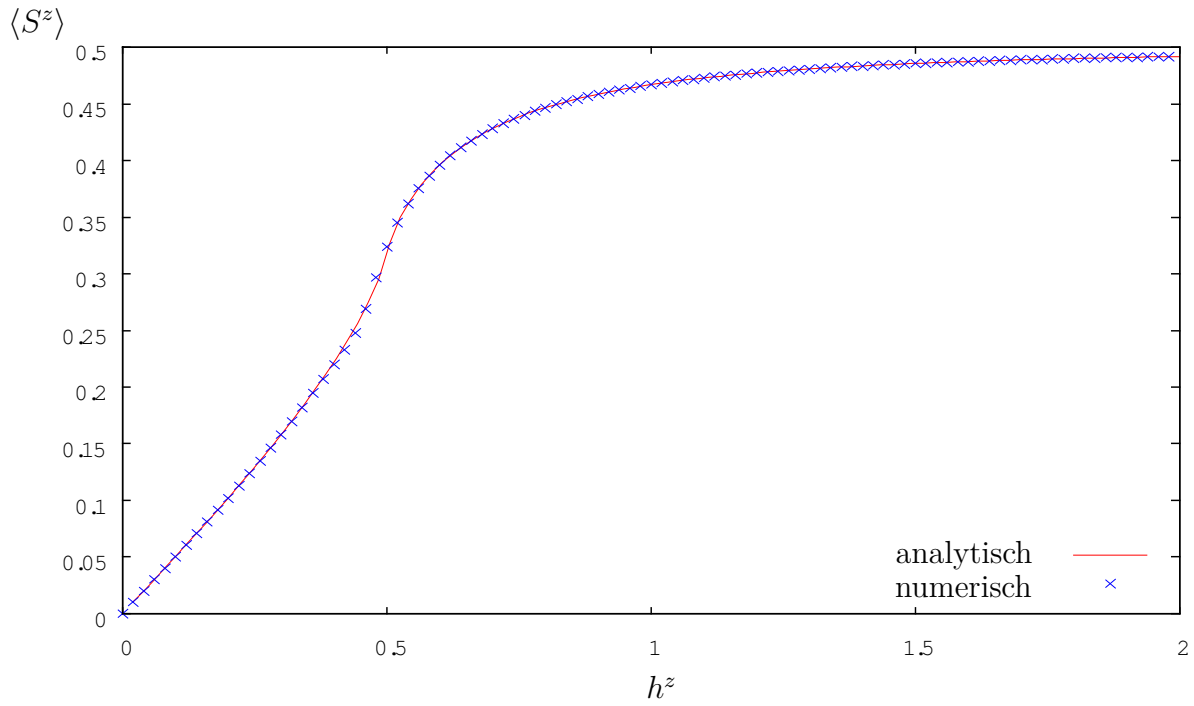

Abb. 6.5.: Größenbereinigte Erwartungswerte von S^z in Abhängigkeit zum angelegten Magnetfeld.

6.4. Heisenberg-Modell

Das Heisenberg-Modell verallgemeinert die im Ising-Modell auf eine Raumrichtung festgelegten Spins in alle Raumrichtungen. Da die Anzahl der Freiheitsgrade hierdurch ansteigt, ist die Berechnung deutlich aufwendiger. Daher wird in den meisten Fälle hiermit die exakte Lösbarkeit geopfert. Mit dem Bethe-Ansatz[6] kann das ein-dimensionale Heisenberg-Modell allerdings gelöst werden. Der Hamilton-Operator einer Kette im Heisenberg-Modell lautet:

$$\hat{H} = J \sum_i \left(\hat{S}_i^x \hat{S}_{i+1}^x + \hat{S}_i^y \hat{S}_{i+1}^y + \hat{S}_i^z \hat{S}_{i+1}^z \right)$$

Für den ferromagnetischen Fall ist die Grundzustandsenergie identisch zum Ising-Modell (ohne Magnetfeld). Der Erwartungswert von S_i^z befindet sich im Intervall $[-0.5, 0.5]$. Die Grundzustandsenergie pro Platz bzw. pro Bindung ist für den antiferromagnetische Fall im folgenden Plot abgebildet:

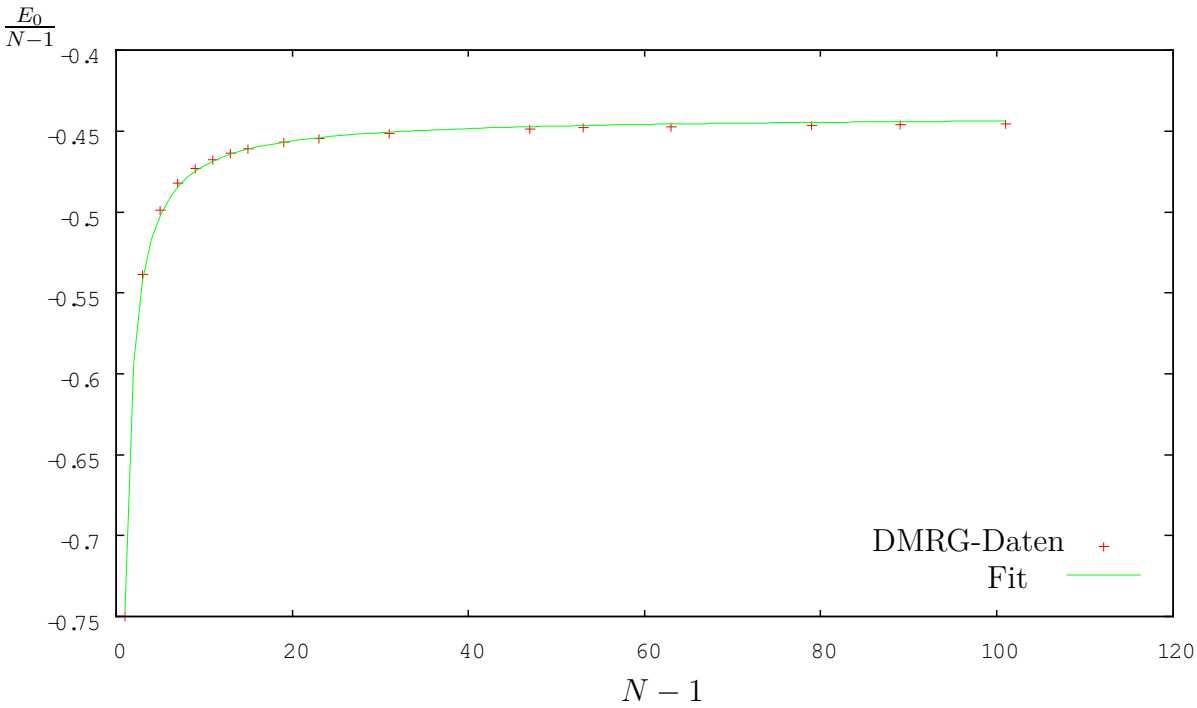

Abb. 6.6.: Abhängigkeit der Grundzustandsenergie von der Kettenlänge.

Der Fit liefert eine Grundzustandenergie von -0.4407 ± 0.00057 pro Bindung für eine unendlich lange Kette. Mit dem Bethe-Ansatz erhält man $\frac{1}{4} - ln(2) = -0.443147$. Der berechnete Wert liegt somit relativ nah am analytischen Wert.

7. Fazit

Die Ergebnisse der Tests zeigen, dass das Applet die erwarteten Werte liefert. Darüber hinaus bietet es aber noch Potenzial für weitere Modelle, wie zum Beispiel das Heisenberg-Modell mit Magnetfeld, welches in dieser Arbeit nicht behandelt wurde. Obwohl die DMRG ein sehr effizienter Algorithmus ist, hat die Erzeugung der Daten für den Abschnitt 6.3 viel Rechenzeit benötigt, da am Phasenübergang sehr lange Ketten untersucht werden mussten, damit das Ergebnis nicht durch die offenen Ränder beherrscht wird.
Die während der Entwicklung des Applets entstandenen und beantworteten Fragen haben zu einem deutlich verbessertem Verständnis der Materie, insbesondere der DMRG und der MPS geführt.
Es gibt sicherlich noch viele Möglichkeiten die Effizienz des Applets zu steigern, indem z.B. bereits erzeugte Daten wiederverwendet werden oder unabhängige Berechnungen in verschiedene Threads verlagert werden. Da der Kern dieser Arbeit allerdings darin besteht, die DMRG zu veranschaulichen, wurde vorerst darauf verzichtet. Die Auswahl der Programmiersprache Java unterstreicht diesen Aspekt.
Abschließend lässt sich feststellen, dass das Ziel der Arbeit erreicht wurde. Das fertige Applet ist unter `www.theorie.physik.uni-goettingen.de/~thomas.koehler/applet.html` zu finden. Der Großteil dieser Arbeit (inkl. dem Quellcode des Applets) sind unter `www.theorie.physik.uni-goettingen.de/~thomas.koehler` in einem Wiki aufbereitet. Hierin können die Plots aus Anhang A außerdem direkt reproduziert werden.

A. Plots

Für alle Erwartungswert-Diagramme gilt: S^x ist in rot und S^z ist in blau dargestellt.

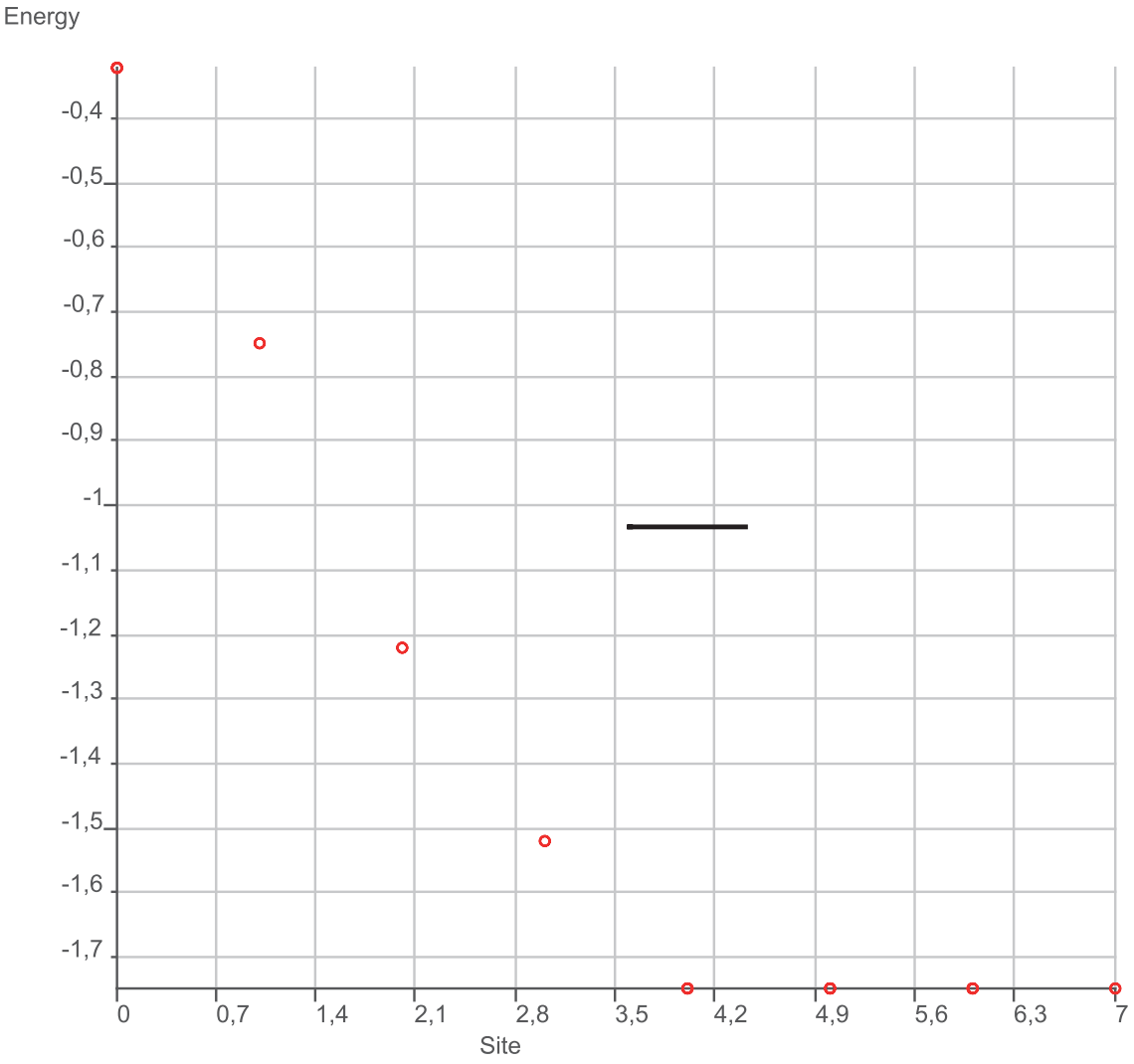

Abb. A.1.: Grundzustandsenergieverlauf (Minimum: -1,75) für eine Ising-Kette ($N = 8$, $d_{\max} = 8$, $H = \sum_i S_i^z S_{i+1}^z$)

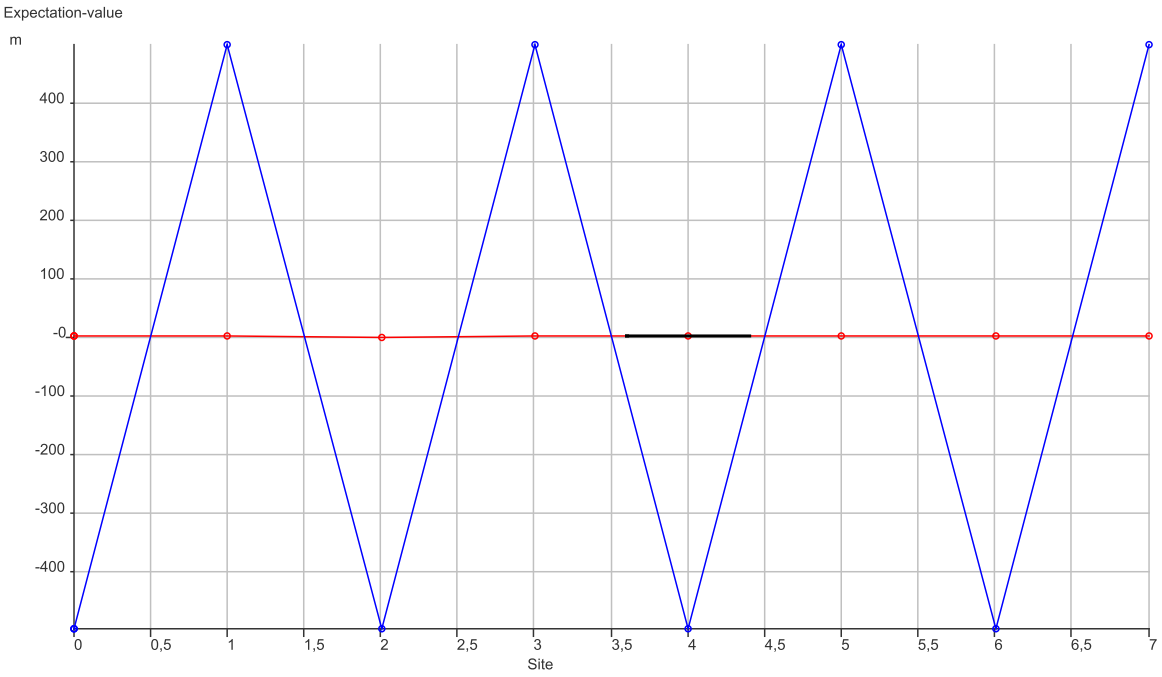

Abb. A.2.: Erwartungswerte für eine Ising-Kette ($N = 8$, $d_{\max} = 8$, $H = \sum_i S_i^z S_{i+1}^z$) nach einem Sweep.

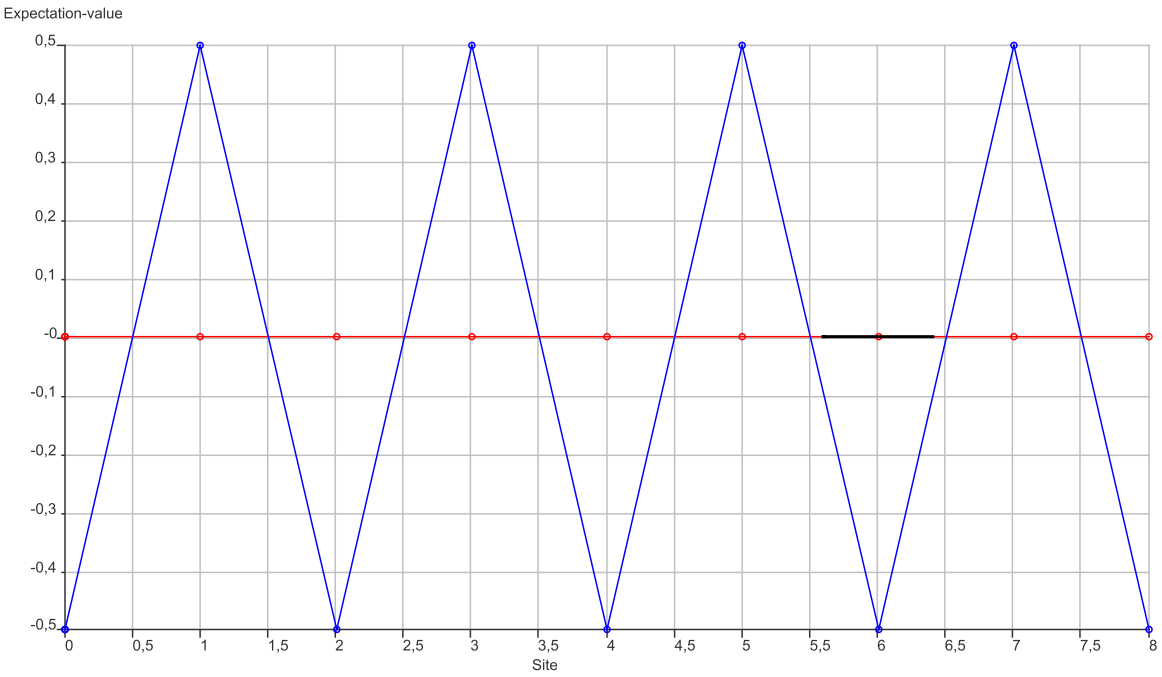

Abb. A.3.: Erwartungswerte für eine Ising-Kette ($N = 9$, $d_{\max} = 8$, $H = \sum_i S_i^z S_{i+1}^z$) nach einem Sweep.

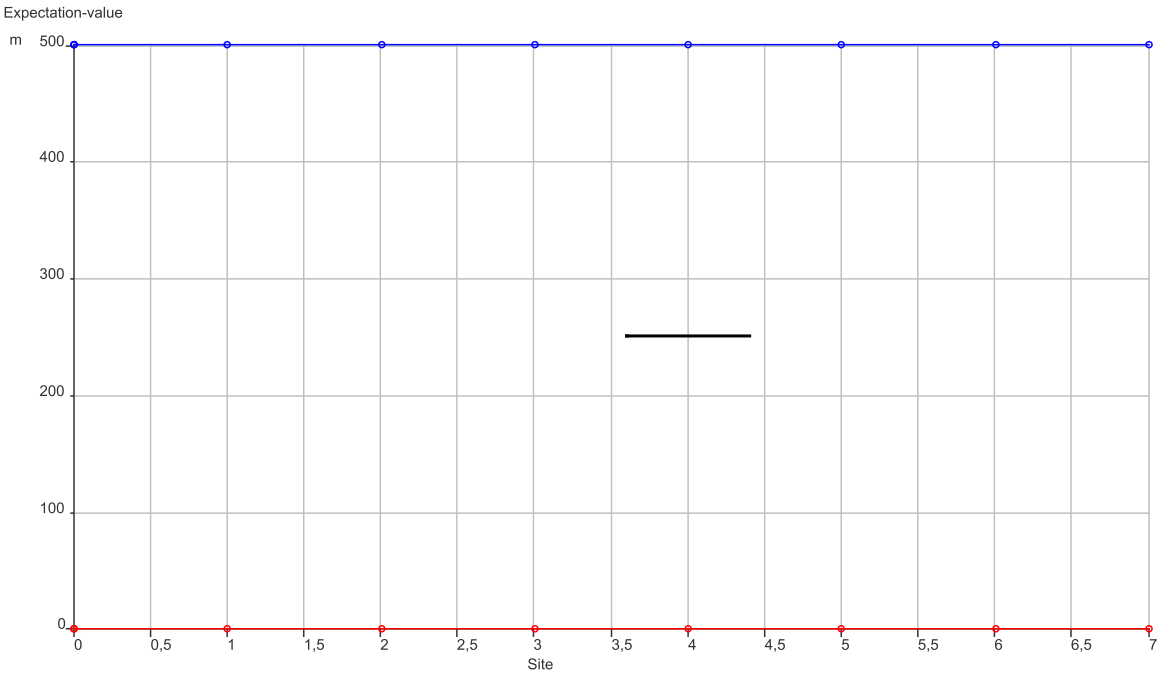

Abb. A.4.: Erwartungswerte für eine Ising-Kette ($N = 8$, $d_{\max} = 8$, $H = -\sum_i S_i^z S_{i+1}^z$) nach einem Sweep.

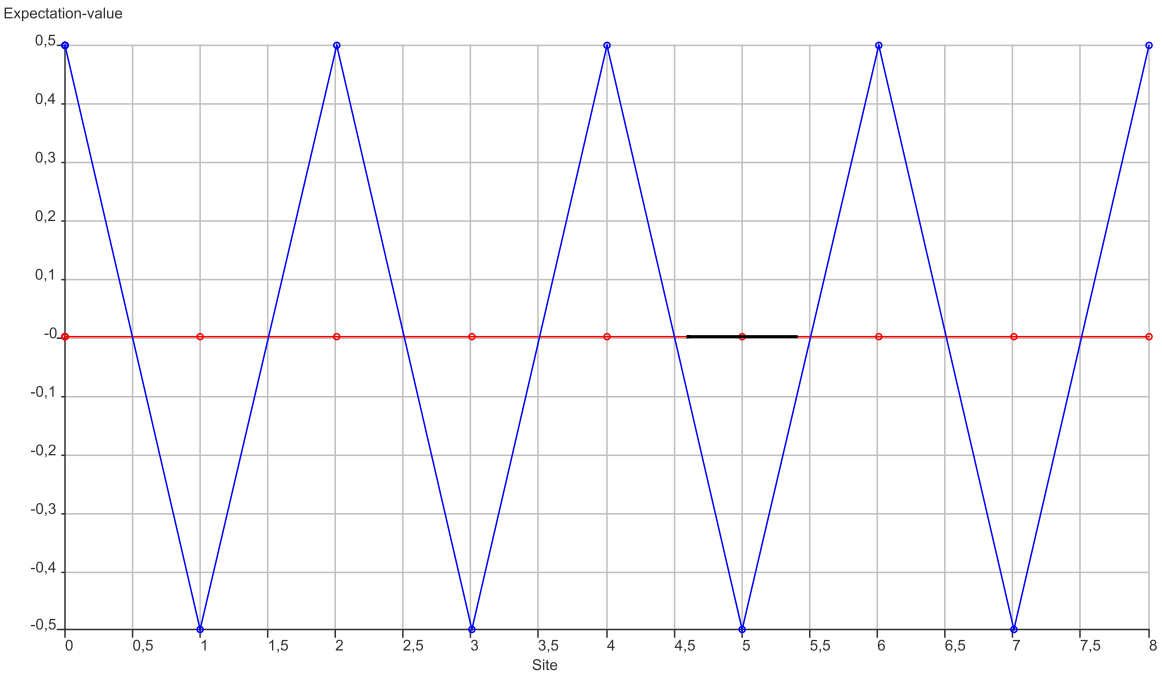

Abb. A.5.: Erwartungswerte für eine Ising-Kette für eine Ising-Kette ($N = 9$, $d_{\max} = 8$, $H = -\sum_i S_i^z S_{i+1}^z - 0.1 \sum_i S^z$) nach einem Sweep.

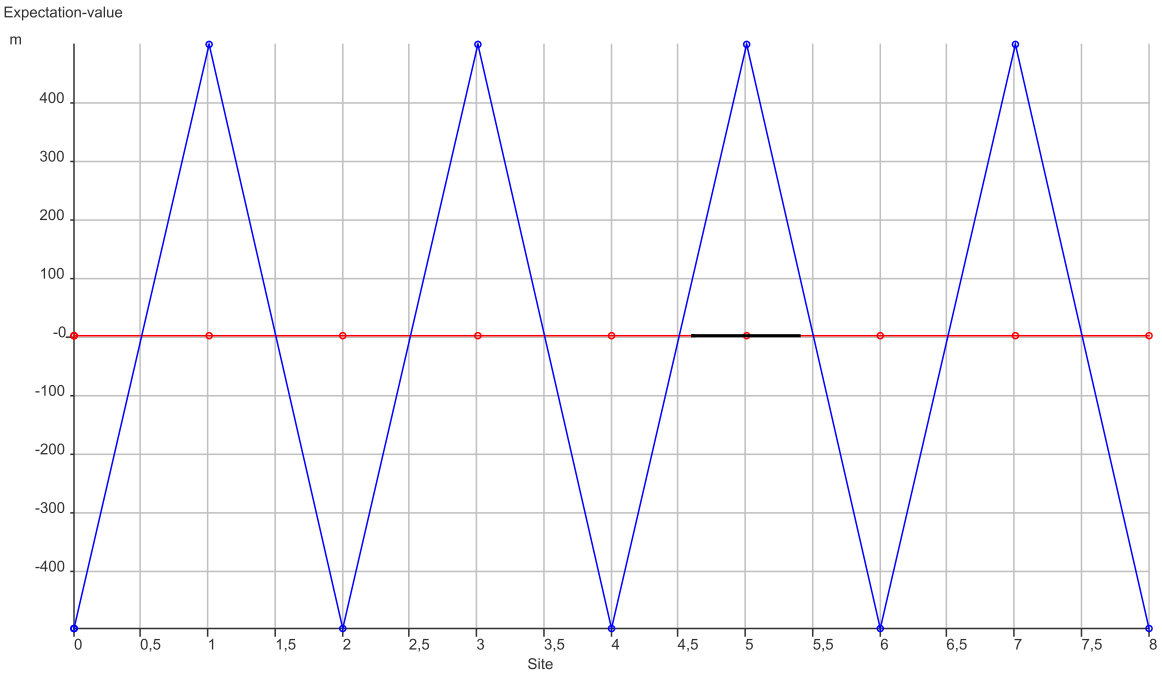

Abb. A.6.: Erwartungswerte für eine Ising-Kette für eine Ising-Kette ($N = 9$, $d_{\max} = 8$, $H = -\sum_i S_i^z S_{i+1}^z + 0.1 \sum_i S^z$) nach einem Sweep.

Literaturverzeichnis

[1] Anderson, E. ; Bai, Z. ; Bischof, C. ; Blackford, S. ; Demmel, J. ; Dongarra, J. ; Du Croz, J. ; Greenbaum, A. ; Hammarling, S. ; McKenney, A. ; Sorensen, D.: *LAPACK Users' Guide*. Third. Philadelphia, PA : Society for Industrial and Applied Mathematics, 1999. – ISBN 0-89871-447-8 (paperback)

[2] Deuflhard, P. ; Hohmann, A.: *Numerische Mathematik I: eine algorithmisch orientierte Einführung*. De Gruyter, 2002 (De Gruyter Lehrbuch v. 1). – URL `http://books.google.com/books?id=6pcxZMeSpj8C`. – ISBN 9783110171822

[3] Doolin, David M. ; Dongarra, Jack ; Seymour, Keith: JLAPACK - compiling LAPACK Fortran to Java. In: *Sci. Program.* 7 (1999), April, S. 111–138. – URL `http://portal.acm.org/citation.cfm?id=1239860.1239868`. – ISSN 1058-9244

[4] Dukelsky, J. ; Martín-Delgado, M. A. ; Nishino, T. ; Sierra, G.: Equivalence of the variational matrix product method and the density matrix renormalization group applied to spin chains. In: *EPL (Europhysics Letters)* 43 (1998), Nr. 4, S. 457. – URL `http://stacks.iop.org/0295-_5075/43/i=4/a=457`

[5] Ising, Ernst: Beitrag zur Theorie des Ferromagnetismus. In: *Zeitschrift für Physik A Hadrons and Nuclei* 31 (1925), S. 253–258. – URL `http://dx.doi.org/10.1007/BF02980577`. – 10.1007/BF02980577. – ISSN 0939-7922

[6] Karbach, Michael ; Hu, Kun ; Müller, Gerhard: Introduction to the Bethe Ansatz II. In: *Comput. Phys.* 12 (1998), November, S. 565–573. – URL `http://portal.acm.org/citation.cfm?id=307043.307062`. – ISSN 0894-1866

[7] Noack, Reinhard ; White, Steven: The Density Matrix Renormalization Group. In: Peschel, Ingo (Hrsg.) ; Kaulke, Matthias (Hrsg.) ; Wang, Xiaoqun (Hrsg.) ; Hallberg, Karen (Hrsg.): *Density-Matrix Renormalization* Bd. 528. Springer Berlin / Heidelberg, 1999, S. 27–66. – URL `http://dx.doi.org/10.1007/BFb0106064`. – 10.1007/BFb0106064

[8] Nolting, W.: *Grundkurs Theoretische Physik. 6. Statistische Physik.* Springer, 2005 (Springer-Lehrbuch Series). – URL `http://books.google.com/books?id=Uw2QaRQxLdYC`. – ISBN 9783540205050

[9] Onsager, L.: Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition. In: *Physical Review* 65 (1944), Februar, S. 117–149

[10] Peierls, R.: On Ising's model of ferromagnetism. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 32 (1936), Nr. 03, S. 477–481. – URL `http://dx.doi.org/10.1017/S0305004100019174`

[11] Pfeuty, P.: The one-dimensional Ising model with a transverse field. In: *Annals of Physics* 57 (1970), M$\hat{}$barz, S. 79–90

[12] Schollwöck, Ulrich: The density-matrix renormalization group in the age of matrix product states. In: *Annals of Physics* 326 (2011), Nr. 1, S. 96 – 192. – URL `http://www.sciencedirect.com/science/article/pii/S0003491610001752`. – January 2011 Special Issue. – ISSN 0003-4916

[13] Schwabl, F.: *Statistische Mechanik.* Springer-Verlag Berlin Heidelberg, 2006 (Springer-Lehrbuch). – URL `http://books.google.com/books?id=kuHbIADUYmoC`. – ISBN 9783540310952

[14] White, Steven R.: Density matrix formulation for quantum renormalization groups. In: *Phys. Rev. Lett.* 69 (1992), Nov, Nr. 19, S. 2863–2866

[15] White, Steven R.: Density-matrix algorithms for quantum renormalization groups. In: *Phys. Rev. B* 48 (1993), Oct, Nr. 14, S. 10345–10356

[16] Yang, C. N.: The Spontaneous Magnetization of a Two-Dimensional Ising Model. In: *Physical Review* 85 (1952), Mar, S. 808–816

Danksagung

Vielen Dank an Andreas Honecker und Piet Dargel für die stets offenen Türen. Außerdem danke ich Thomas Pruschke dafür, dass er sich als Zweitgutachter zur Verfügung gestellt hat.
Darüber hinaus geht mein Dank an das Theoretische Institut der Fakultät für Physik, das den Web-Space für die Online-Version und das Applet zur Verfügung stellt.

Erklärung nach §13(8) der Prüfungsordnung für den Bachelor-Studiengang Physik und den Master-Studiengang Physik an der Universität Göttingen:

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe.
Darüberhinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, im Rahmen einer nichtbestandenen Prüfung an dieser oder einer anderen Hochschule eingereicht wurde.

Göttingen, den 21. August 2011

(Thomas Köhler)