Eigenmoden des Gitter-Laplace-Operators als Hilfsmittel zur Berechnung hadronischer Korrelatoren

BACHELORARBEIT

zur Erlangung des akademischen Grades
Bachelor of Science
im Fach Physik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät I
Institut für Physik
Humboldt-Universität zu Berlin

von
Johannes Roland Siefert

Betreuung:

1. *Dr. Marc Wagner*
2. *Prof. Dr. Michael Müller-Preußker*

eingereicht am: *16.09.2011*

Zusammenfassung

Das *distillation* ist ein Smearing, welches die Eigenmoden des Gitter-Laplace-Operators verwendet. Diese sind im Allgemeinen nicht analytisch bestimmbar und müssen numerisch berechnet werden. Diese Arbeit analysiert das Eigenwertproblem des Laplace-Operators und zeigt, wie dessen Eigenwerte und Eigenvektoren berechnet werden können.

Inhaltsverzeichnis

1 Einleitung

Als wesentlicher Bestandteil des Standardmodells ist die Quantenchromodynamik (QCD) eine Quantenfeldtheorie zur Beschreibung der starken Wechselwirkung mit der Lagrangefunktion

$$\mathcal{L}_{QCD} = \sum_{f=u,d,s,\ldots} \bar{\Psi}^{(f)}(i\gamma^\mu D_\mu - m^{(f)})\Psi^{(f)} - \frac{1}{4}F^a_{\mu\nu}F^{\mu\nu}_a \,. \tag{1}$$

Die QCD beschreibt die Wechselwirkung der Quarks aufgrund ihrer Farbladung unter Austausch von ebenfalls farbgeladenen Gluonen. Rechnungen können nicht mehr analytisch gelöst, sondern müssen mit der Störungstheorie oder durch Computersimulationen berechnet werden. Die Störungstheorie kann nur angewandt werden, wenn die Kopplungskonstante klein ist. Ist die Kopplungskonstante groß, wie im Niederenergiebereich der QCD, dann divergiert die Reihenentwicklung der Kopplungskonstanten und die Störungstheorie führt zu keinen sinnvollen Ergebnissen mehr. In diesem Fall müssen Computersimulationen genutzt werden. Um numerische Verfahren nutzen zu können wird in der so genannten Gitter-QCD die Raumzeit diskretisiert und auf dem so entstandenen Gitter gerechnet [1]. Eine Anwendung ist bspw. die Massenberechnung von Mesonen und Baryonen.

Diese Arbeit stellt eine Vorarbeit zu einer kürzlich eingeführten Smearing-Methode in der Gitter-QCD, der so genannten *distillation* [2], dar. Beim Smearing werden die Erzeuger und Vernichter in der 2-Punktkorrelationsfunktion so verändert, dass in der numerischen Berechnung eine effiziente Massenbestimmung von bspw. Mesonen möglich ist. Bei der *distillation* werden dazu Eigenvektoren des kovarianten Laplace-Operators verwendet.

2 Motivation

Bevor erklärt wird, wie die Eigenvektoren des kovarianten Laplace-Operators berechnet werden, soll noch kurz erläutert werden, wie die Korrelationsfunktion genutzt werden kann, um die Masse eines durch die Operatoren beschriebenen gebundenen Zustands (z.B. eines Mesons) zu bestimmen.

2.1 Vom Korrelator zur Masse

Um die Energie eines Eigenzustands des Hamilton-Operators zu erhalten, berechnet man die Korrelationsfunktion $\Gamma(t)$ zwischen dem Erzeugungsoperator $\mathcal{O}^\dagger(0)$ und dem Vernichtungsoperator $\mathcal{O}(t)$.

$$\Gamma(t) = \langle\Omega|\, \mathcal{O}(t)\mathcal{O}^\dagger(0)\, |\Omega\rangle \tag{2}$$

Ω stellt dabei das Vakuum dar. Fügt man einen vollständigen Satz an Eigenzuständen ein, für die gilt $\hat{H}|n\rangle = E_n|n\rangle$, kann die Korrelationsfunktion umgeformt werden. $\langle n|$ hat dabei die gleichen Quantenzahlen (z.B. Spin, Isospin), wie $\mathcal{O}^\dagger|\Omega\rangle$. Ein Beispiel für den Erzeugungsoperator eines π-Mesons wäre

$\mathcal{O}^\dagger = \bar{u}\gamma_5 d$.

$$\Gamma(t) = \sum_n \langle\Omega|\, \mathcal{O}(t)\, |n\rangle\langle n|\, \mathcal{O}^\dagger(0)\, |\Omega\rangle \tag{3}$$

$$\mathcal{O}(t) = e^{+\hat{H}t}\mathcal{O}(0)e^{-\hat{H}t} \tag{4}$$

$$\Gamma(t) = \sum_n \underbrace{|\langle\Omega|\mathcal{O}(0)|n\rangle|^2}_{c_n} e^{-E_n t} \tag{5}$$

Wobei die Vakuumsenergie zu Null definiert werden kann und somit $\langle\Omega|e^{\hat{H}t} = \langle\Omega|e^{0\cdot t} = \langle\Omega|$ ist. Da E_0 den Grundzustand beschreibt, werden mit steigendem Index n die Energien immer größer. Das bedeutet, dass der Faktor $\exp(-E_n\, t)$ mit größerem E_n stärker exponentiell abnimmt. Für große Zeiten dominiert somit der erste Summand.

$$t \to \infty \tag{6}$$

$$\Gamma(t) = e^{-(E_0)t} \cdot c_0 \tag{7}$$

Berechnen wir nun $\Gamma(t)$ mit Hilfe der Pfadintegraldarstellung, können wir für große Zeiten die Grundzustandsenergie E_0 des entsprechenden Sektors, z.B. des Pions mit den Quantenzahlen $I = 1$, $J = 0$ und $P = -$, aus dem exponentiellen Abfall ablesen.

2.2 Smearing

In der Praxis ist es üblich die effektive Masse zu bestimmen. Für große Zeiten stimmt diese mit der Grundzustandsenergie überein, was aus der Definition der effektiven Masse ersichtlich wird.

$$m_{eff}(t) := -\frac{1}{\Delta t} ln\left(\frac{\Gamma(t+\Delta t)}{\Gamma(t)}\right) \tag{8}$$

$$t \to \infty \qquad = -\frac{1}{\Delta t} ln\left(\frac{c_0 e^{-E_0(t+\Delta t)}}{c_0 e^{-E_0 t}}\right) \tag{9}$$

$$= E_0 \tag{10}$$

Die Konvergenz bei großen Zeiten stellt jedoch ein Problem dar, denn große Zeiten bedeuten große Rechenzeiten, da sich der statistische Fehler aus den Monte-Carlo-Simulationen nur durch mehr Rechenzeit verringern lässt. Eine Möglichkeit die Konvergenz zu beschleunigen, ist die Koeffizienten c_n, auch als Overlap zwischen dem Testzustand $\mathcal{O}^\dagger|\Omega\rangle$ und dem Eigenzustand $|n\rangle$ bezeichnet, zu verändern. Diese werden so verändert, dass alle c_n mit $n > 0$ im Vergleich zu c_0 sehr klein werden. Um dies zu erreichen müssen die Operatoren verändert werden, ohne jedoch deren Quantenzahlen zu verändern. Die Veränderung der Operatoren zu diesem Zweck wird Smearing genannt und umso kleiner diese Overlaps werden, desto besser ist das Smearing.

Eine gängige Methode ist das Jacobi-Smearing. Es basiert auf den Gitter-Laplace-Operator und beginnt mit der einfachsten Darstellung des dreidimensionalen Differentialoperators zweiter Ordnung

$$-\nabla^2_{xy}(t) = 6\delta_{xy} - \sum_{j=1}^{3}\left(U_j(x,t)\delta_{x+\hat{\jmath},y} + U_j^\dagger(x-\hat{\jmath},t)\delta_{x-\hat{\jmath},y}\right)\,, \tag{11}$$

wobei U Links des Eichfeldes sind. Ein einfaches Smearing kann dann wie folgt formuliert werden:

$$J_{\sigma,n_\sigma}(t) = \left(1 + \frac{\sigma\nabla^2(t)}{n_\sigma}\right)^{n_\sigma} \tag{12}$$

σ und n_σ sind frei wählbare und zu optimierende Parameter. Für große n_σ nähert sich J_{σ,n_σ} einer Exponentialfunktion an.

$$\lim_{n_\sigma\to\infty} J_{\sigma,n_\sigma}(t) = \exp\left(\sigma\nabla^2(t)\right) \tag{13}$$

Somit werden die höheren Eigenmoden des Gitter-Laplace-Operators exponentiell unterdrückt und nur eine kleine Anzahl der kleinsten Moden tragen zu J_{σ,n_σ} bei.

Es existieren verschiedene weitere Techniken solch ein Smearing durchzuführen. Eine neue Variante aus dem Jahr 2008 wird in dem Paper [2] beschrieben und wird als so genanntes *distillation* bezeichnet. Hierbei werden die Eigenvektoren des kovarianten Laplace-Operators verwendet. Es wird dabei ein so genannter Smearing-Operator $\Box(t)$ auf einer Zeitschicht t definiert, der aus dem Produkt einer $N \times M$-Matrix mit deren Adjungierten gebildet wird.

$$\Box(t) = V(t)V^\dagger(t) \tag{14}$$

Die Spalten der Matrix $V(t)$ bestehen aus den den M kleinsten Eigenwerten zugehörigen Eigenvektoren des kovarianten Laplace-Operators in der Zeitschicht t. Besteht $V(t)$ aus allen Eigenvektoren, dann reduziert sich der Smearing-Operator zum Einheitsoperator und es wird nicht geschmiert. Im Folgenden soll nun erläutert werden, wie die Eigenvektoren des kovarianten Laplace-Operators berechnet werden können.

3 Eigenwertproblem des Laplace-Operators

Bevor dem kovarianten Laplace-Operator nachgegangen wird, soll der freie Laplace-Operator untersucht werden. Dessen Eigenwertproblem lässt sich analytisch lösen und die Ergebnisse können später als Test wiederverwendet werden. Anschließend wird der kovariante Laplace-Operator der SU(3) Eichtheorie betrachtet, da dieser für das *distillation* verwendet wird.

Im n-dimensionalen, Euklidischen Raum ist der Laplace-Operator die Summe der zweiten partiellen Ableitungen nach den Raumkoordinaten

$$\Delta = \sum_{j=1}^{n} \frac{\partial^2}{\partial x_j^2} \quad . \tag{15}$$

Der Laplace-Operator taucht in vielen physikalischen Problemstellungen auf. Ein Beispiel ist die stationäre Schrödingergleichung, die das Verhalten quantenmechanischer Systeme beschreibt

$$-\frac{\hbar^2}{2m}(\partial_x^2 + \partial_y^2 + \partial_z^2)\Psi(x,y,z) = E\Psi(x,y,z) \quad . \tag{16}$$

Dabei wurde die gekürzte Schreibweise $\frac{\partial^2}{\partial x^2} = \partial_x^2$ verwendet. Es handelt sich bei der stationären Schrödingergleichung um ein Eigenwertproblem, welches sich allgemein als Eigenwertproblem des Laplace-Operators formulieren lässt.

$$\Delta f(x,y,z) = \lambda\, f(x,y,z) \tag{17}$$

Der Einfachheit halber wird zunächst der eindimensionale Fall betrachtet.

3.1 analytische Lösung des Eigenwertproblems im Kontinuum

Unser Eigenwertproblem hat nun die Form

$$\partial_x^2 f(x) = \lambda f(x) \quad . \tag{18}$$

Die Gleichung lässt sich mit dem e-Ansatz lösen, wobei wir für die Funktion $f(x) = Ae^{qx} + Be^{-qx}$ ansetzten. Wir erhalten dann für die Eigenwerte $\lambda = q^2$. Ist λ positiv, sind die Sinushyperbolikus- und Kosinushyperbolikusfunktionen Lösungen, ist λ negativ, sind die Sinus- und Kosinusfuktionen Lösungen. An dieser Stellen sollen, in Analogie zu den später interessanten QCD Gitterrechnungen, periodische Randbedingungen eingeführt werden, womit $f(x+L) = f(x)$ gilt und somit $\lambda > 0$ ausgeschlossen wird. Die Lösung nimmt somit die Form

$$f(x) = Ae^{\sqrt{\lambda}x} + Be^{-\sqrt{\lambda}x} \tag{19}$$

$$0 > \lambda = -k^2 \tag{20}$$

$$f(x) = Ae^{ikx} + Be^{-ikx} \tag{21}$$

an. Verwenden wir die Randbedingungen, erhalten wir für k:

$$k_n = \frac{2\pi}{L} n \qquad n = 0, \pm 1, \pm 2, ... \tag{22}$$

k_n ist somit für $n > 0$ zweifach entartet, mit den Eigenfunktionen

$$f_1(x) = \sin\left(\frac{x \cdot 2\pi}{L} n\right) \qquad f_2(x) = \cos\left(\frac{x \cdot 2\pi}{L} n\right) . \tag{23}$$

3.2 Der Laplace-Operator auf dem Gitter in 1D

Diskretisieren wir unseren Ortsparameter x, dann können wir die zweite Ableitung mit Hilfe der zwei benachbarten Punkte beschreiben:

$$\partial_x^2 f(x_0) \qquad \overset{a\to 0}{\longleftarrow} \qquad \frac{1}{a^2}\left(f(x_0 - a) - 2f(x_0) + f(x_0 + a)\right) \tag{24}$$

Eine Diskretisierung bedeutet, dass sich unsere Funktion als Vektor mit $N = L/a$ Einträgen schreiben lässt, mit a als den Gitterabstand. Dies ermöglicht uns eine zweite Ableitung in Form einer Matrix A_{jk} zu definieren, die nicht nur auf einen Punkt $f(x_0)$ sondern auf die gesamte in Form eines Vektor diskretisierte Funktion wirkt.

$$A_{jk} = \frac{1}{a^2}(\delta_{j,k+1} - 2\delta_{j,k} + \delta_{j,k-1}) \tag{25}$$

Wir können jetzt unser Eigenwertproblem

$$A_{jk}v_k = \lambda v_j \tag{26}$$

erneut lösen. Als Ansatz wird wieder der e-Ansatz gewählt.

$$v_k = e^{i\alpha k} \tag{27}$$

Die Periodizität erfordert dabei

$$\alpha = \frac{2\pi}{N}n\,, \qquad n = 0, \pm 1, \pm 2, ... \pm N/2 - 1, N/2\,, \tag{28}$$

wobei N die Anzahl an Gitterpunkten ist. Die linke Seite des Eigenwertproblems wird dann zu

$$A_{jk}v_k = \frac{1}{a^2}e^{i\alpha j} \cdot \underbrace{(\, e^{-i\alpha} - 2 + e^{i\alpha}\,)}_{-2+2\cos(\alpha) = -4\sin^2(\alpha/2)} \tag{29}$$

und die rechte Seite zu

$$\lambda v_j = \lambda e^{i\alpha j} \quad . \tag{30}$$

Vergleichen wir beide Seiten miteinander, können wir λ extrahieren.

$$\lambda e^{i\alpha j} = \frac{1}{a^2}e^{i\alpha j}(-4\sin^2(\alpha/2)) \tag{31}$$

$$\lambda = -\frac{4}{a^2}\sin^2(\alpha/2) \tag{32}$$

$$\lambda = -\frac{4}{a^2}\sin^2(\frac{\pi}{N}n) \qquad \longrightarrow \qquad \lambda_{min} = -\frac{4}{a^2} \tag{33}$$

nur für kleine α : (34)

$$\lambda = -\frac{4}{a^2}\left(\frac{\alpha}{2}\right)^2 = -\left(\frac{\alpha}{a}\right)^2 \tag{35}$$

$$\lambda = -\left(\frac{2\pi n}{aN}\right)^2\,, \qquad a \cdot N = L \tag{36}$$

$$k = \frac{2\pi}{L}n \tag{37}$$

Die Kontinuumslösung erhalten wir, wenn wir bei festem L $a \to 0$ und somit $N \to \infty$ streben lassen. Es ist zu beachten, dass die Lösungen des Gitters nur für kleine n mit der Kontinuumslösung übereinstimmen und nach oben durch den Gitterabstand beschränkt sind. Als physikalische Anwendung sei hier das Problem der Fermionenverdopplung genannt.

3.3 Der Laplace-Operator auf dem Gitter in 3D

Im 3-dimensionalen Fall sieht die Diskretisierung wie folgt aus:

$$\begin{aligned}
&(\partial_x^2 + \partial_y^2 + \partial_z^2)f(x_0, y_0, z_0) \\
&\overset{a\to 0}{\longleftarrow} \\
&\frac{1}{a^2}\left(f(x_0 - a, y_0, z_0) - 2f(x_0, y_0, z_0) + f(x_0 + a, y_0, z_0)\right) \\
+\;&\frac{1}{a^2}\left(f(x_0, y_0 - a, z_0) - 2f(x_0, y_0, z_0) + f(x_0, y_0 + a, z_0)\right) \\
+\;&\frac{1}{a^2}\left(f(x_0, y_0, z_0 - a) - 2f(x_0, y_0, z_0) + f(x_0, y_0, z_0 + a)\right)
\end{aligned} \tag{38}$$

Da es sich im Wesentlichen drei mal um das gleiche Problem handelt, kann das Eigenwertproblem durch den Produktansatz in drei unabhängige Probleme aufgeteilt werden.

$$\Delta f(x_0, y_0, z_0) = \lambda f(x_0, y_0, z_0) \tag{39}$$
$$= -k^2 f(x_0, y_0, z_0) \tag{40}$$
$$(\partial_x^2 + \partial_y^2 + \partial_z^2) f(x_0, y_0, z_0) = -(k_x^2 + k_y^2 + k_z^2) f(x_0, y_0, z_0) \tag{41}$$

Die Lösung für λ ist somit die bereits in Gl.(33) ermittelte, nur dass diese drei mal auftaucht mit unterschiedlichen Variablen n.

$$\lambda = -\left(\frac{4}{a^2}\sin^2(\frac{\pi}{N}n_x) + \frac{4}{a^2}\sin^2(\frac{\pi}{N}n_y) + \frac{4}{a^2}\sin^2(\frac{\pi}{N}n_z)\right) \tag{42}$$
$$n_x, n_y, n_z = 0, \pm 1, \pm 2, ..., \pm N/2 - 1, N/2 \tag{43}$$

Betrachten wir den Fall für kleine Argumente in den Sinusfunktionen, dann können wir die Lösung noch weiter zusammenfassen.

$$\lambda = -\left(\frac{2\pi}{aN}\right)^2 \left(n_x^2 + n_y^2 + n_z^2\right) \tag{44}$$
$$\lambda = -\left(\frac{2\pi}{aN}\right)^2 \left(n^2\right) \tag{45}$$

Mit $n^2 = n_x^2 + n_y^2 + n_z^2$ ist λ somit mehrfach entartet und einige Eigenwerte existieren mehrfach. Dies trifft auch auf die Kontinuumslösung in 3D zu, da mit $L = aN$ Gl.(45) ebenfalls Lösung im Kontinuum ist.

$$\lambda_0 = 0: \quad \times 1 \tag{46}$$
$$\text{z.B. } n_x = 0, n_y = 0, n_z = 0 \tag{47}$$
$$\lambda_1 = -\left(\frac{2\pi}{aN}\right)^2 \cdot 1: \quad \times 6 \tag{48}$$
$$\text{z.B. } n_x = 1, n_y = 0, n_z = 0 \tag{49}$$
$$\lambda_2 = -\left(\frac{2\pi}{aN}\right)^2 \cdot 2: \quad \times 12 \tag{50}$$
$$\text{z.B. } n_x = 1, n_y = -1, n_z = 0 \tag{51}$$
$$\lambda_3 = -\left(\frac{2\pi}{aN}\right)^2 \cdot 3: \quad \times 8 \tag{52}$$
$$\text{z.B. } n_x = 1, n_y = -1, n_z = -1 \tag{53}$$
$$\lambda_4 = -\left(\frac{2\pi}{aN}\right)^2 \cdot 4: \quad \times 6 \tag{54}$$
$$\text{z.B. } n_x = -2, n_y = 0, n_z = 0 \tag{55}$$

Dies kann beliebig mit $n_x, n_y, n_z = 0, \pm 1, \pm 2, ..., \pm N/2 - 1, N/2$ fortgesetzt werden.

3.4 Der kovariante Laplace-Operator

Schaltet man ein Eichfeld ein, transformiert sich das Quarkfeld unter Eichtransformation zu $\Psi \to g\Psi$. Das Quarkfeld mit dem darauf angewendeten Laplace-

Operator $\Delta\Psi$ hat aber ein anderes Transformationsverhalten. Aus diesem Grund wird der kovariante Laplace-Operator D^2 verwendet, welcher sich zu $D^2 \rightarrow gD^2g^\dagger$ eichtransformiert. Die kovariante Ableitung ist dabei wie folgt definiert [3]:

$$D_j = \partial_j + igA_j \tag{56}$$

Am Gitter wird das Eichfeld A_μ zu so genannten Links U_μ.

$$U(x_0, x_0 \pm a) = e^{+igA_\mu(x \pm \frac{a}{2})a} \tag{57}$$

Der kovariante Laplace-Operator hat dann auf dem Gitter die Form

$$D^2 f(x_0) = \frac{1}{a^2}\left(U(x_0, x_0 - a)f(x_0 - a) - 2f(x_0) + U(x_0, x_0 + a)f(x_0 + a)\right) \quad . \tag{58}$$

Die Eigenwerte des kovarianten Laplace-Operators sind nicht mehr analytisch zu ermitteln, weshalb numerische Methoden ([4], [5]) eingesetzt werden müssen. Eine mögliche Methode ist das Arnoldi-Verfahren, welches in Kapitel 4 näher erläutert wird.

Das Feld, auf das wir den kovarianten Laplace-Operator anwenden wollen, besitzt jedoch nicht nur einen komplexen Wert an jedem Punkt, sondern, aufgrund der SU(3) Eichgruppe, drei. Wir betrachten nämlich alle drei Farbfelder an einem Punkt, sodass unsere Funktion $f_a(x_0)$ noch einen Farbindex $a = 1, 2, 3$ erhält. Wir fassen die drei Farbfelder zu einer Funktion $\Psi(x_0) = (f_1(x_0), f_2(x_0), f_3(x_0))$ zusammen. Unsere Funktion hat nun die Form eines Vektors, worauf die Links die Form einer 3×3-Matrix haben müssen. Die Farbfelder koppeln aneinander, weshalb nicht nur die Diagonalelemente der Links von Null verschieden sind. Setzt man die Diagonalelemente Eins und den Rest Null, hat man wieder den Laplace-Operator, wobei alle Eigenwerte dreifach vorkommen, da wir die Eigenwerte von drei unabhängigen Feldern berechnen. Der kovariante Laplace-Operator in 3D verhält sich analog zu dem hier geschilderten Fall in 1D.

4 Lösen des Eigenwertproblems durch die implicitly restarted Arnoldi Methode (IRAM)

Die implicitly restarted Arnoldi Methode (IRAM) ist eine gekürzte Form des implicitly shifted QR-Algorithmus, mit einer zusätzlichen Arnoldi-Faktorisierung. Die IRAM dient der Eigenwert- und Eigenvektorbestimmung von dünn besetzten, komplexwertigen Matrizen und ist somit geeignet die Eigenwerte und -vektoren des kovarianten Laplace-Operators zu berechnen.

Das Arnoldi-Verfahren dient dabei zur Verkleinerung der $n \times n$-Matrix A, von der $k < m < n$ Eigenwerte bestimmt werden sollen, auf eine $m \times m$-Matrix H_m. Die Eigenwerte von H_m sind aufgrund der speziellen Form und der kleineren Größe der Matrix leichter zu bestimmen und u.U. identisch mit den Eigenwerten von A.

Die QR-Iteration dient eigentlich der Umformung einer Matrix durch Ähnlichkeitstransformationen in eine Dreiecksform, um anschließend aus den Diagonalelementen die Eigenwerte abzulesen. Dazu kann zuvor noch ein so genannter Shift angewendet werden, der die Diagonalelemente der Matrix mit einer

bestimmten Zahl subtrahiert und somit die Konvergenz der QR-Iteration beschleunigt.

Die IRAM kombiniert nun diese beiden Verfahren in einem iterativen Prozess, da die Form der Matrix H_m sehr gut von der QR-Iteration ausgenutzt werden kann. Die QR-Iteration wird dabei aber bei jedem Iterationsschritt nicht bis zum Ende durchgeführt, sondern frühzeitig abgebrochen. Anschließend wird die Matrix H_m auf eine $k \times k$-Matrix reduziert und bevor die Iteration von neuem beginnt mit der Arnoldi-Faktorisierung von A wieder zu einer neuen $m \times m$-Matrix H'_m vergrößert.

4.1 Arnoldi-Verfahren

Das Arnoldi-Verfahren berechnet von einer gegebenen Matrix A eine orthonormale Basis auf dem zugeordneten Krylowraum $\mathcal{K}_m$ [5]. Der Krylowraum setzt sich dabei folgendermaßen zusammen:

$$\mathcal{K}_m(A, v) \equiv \text{span}\{v, Av, A^2v, ..., A^{m-1}v\} \tag{59}$$

v ist dabei ein nicht notwendig normierter Startvektor. Ist der Startvektor v eine Linearkombination von Vektoren, die einen invarianten Unterraum von A aufspannen (z.B. die ersten k Eigenvektoren), dann ist der Krylow-Raum ein invarianter Unterraum für A.

Ein möglicher Algorithmus des Arnoldi-Verfahrens wäre folgender nach [5] (siehe auch [6]):

1. *Wähle einen Startvektor* $\vec{v}_1$ *mit* $|\vec{v}_1| = 1$
2. *For* $j = 1, 2, ..., m$, *Do*
3. *Berechne* $h_{ij} = (A\,\vec{v}_j, \vec{v}_i)$ *for* $i = 1, 2, ..., j$
4. *Berechne* $\vec{w}_j := A\,\vec{v}_j - \sum_{i=1}^{j} h_{ij}\vec{v}_i$
5. $h_{j+1,j} = |\vec{w}_j|$
6. *If* $h_{j+1,j} = 0$ *then Stop*
7. $\vec{v}_{j+1} = \vec{w}_j / h_{j+1,j}$
8. *EndDo*

Bei jedem Schritt wird hierbei das Produkt von der $n \times n$-Matrix A mit dem vorher berechneten Arnoldi-Vektor $\vec{v}_j$ gebildet und nach dem Gram-Schmidtschen Orthonormalisierungsverfahren orthonormiert (Zeile 4). Wir erhalten also eine orthonormale Basis die wir in einer $n \times m$-Matrix $V_m = (\vec{v}_1, \vec{v}_2, ..., \vec{v}_m)$ zusammenfassen können. Zudem bilden die h_{ij} die von Null verschiedenen Werte einer so genannten $(m+1) \times m$-Hessenbergmatrix $\bar{H}_m$, wobei wir die Matrix $\bar{H}_m$ mit fehlender letzter Zeile H_m nennen. Eine Hessenbergmatrix hat die Form einer oberen Dreiecksmatrix mit einer zusätzlichen Unterdiagonalen. Aus Zeile 4 des Algorithmus folgt Gl.(60) und aus Zeile 7 folgt Gl.(61). Fügt man den letzten

Summanden der Summe hinzu, erhält man Gl.(62).

$$A\,\vec{v}_j = \sum_{i=1}^{j} h_{ij}\vec{v}_i + \vec{w}_j\,, \qquad j = 1, 2, ..., m \tag{60}$$

$$A\,\vec{v}_j = \sum_{i=1}^{j} h_{ij}\vec{v}_i + h_{j+1,j}\,\vec{v}_{j+1} \tag{61}$$

$$A\,\vec{v}_j = \sum_{i=1}^{j+1} h_{ij}\vec{v}_i \tag{62}$$

Diese Gleichungen lassen sich auch in Matrixschreibweise umformulieren. Gl.(60) wird somit zu Gl.(63) und Gl.(62) wird zu Gl.(64).

$$A\,V_m = V_m H_m + \vec{w}_m \vec{e}_m^T \tag{63}$$

$$A\,V_m = V_{m+1}\bar{H}_m \tag{64}$$

Da $\vec{w}_m \vec{e}_m^T = h_{m+1,m}\vec{v}_{m+1}\vec{e}_m^T$ gilt für kleine $h_{m+1,m}$

$$V_m^\dagger A\,V_m \approx H_m\,. \tag{65}$$

Wichtig bei dem Arnoldi-Verfahren ist, dass die Eigenwerte der Hessenbergmatrix H_m gute Approximationen für die Eigenwerte einer gegebenen Matrix A sind. Dies gilt für kleine $h_{m+1,m}$ und für Eigenwerte am Rand des Spektrums von A. Für $h_{m+1,m} = 0$ sind die Eigenwerte sogar identisch, da dann in Gl.(65) Gleichheit herrscht und V_m eine orthonormale Matrix ist. Je nach Wahl von $m \leq n$ besitzt H_m nur m der n Eigenwerte von A. Die Eigenvektoren y der Matrix H_m bilden zusammen mit V_m die Eigenvektoren x der Matrix A, wobei gilt $x = V_m y$ [4]. Typischer Weise wird die Gl. (65) unter Gleichheit als Arnoldi-Faktorisierung bezeichnet.

4.2 QR-Iteration

Gemäß der QR-Zerlegung lässt sich jede quadratische Matrix A in ein Produkt aus einer unitären Matrix Q und einer oberen Dreiecksmatrix R zerlegen.

$$A = QR \tag{66}$$

Diese Zerlegung erfolgt üblicher Weise mit der Householdertransformation, kann aber auch durch ein anderes Verfahren, wie dem Gram-Schmidtschen Orthogonalisierungsverfahren, realisiert werden. Dreht man das Produkt in Gl.(66) um, erhält man eine andere Matrix A', wobei sich zeigen lässt, dass diese aus einer Ähnlichkeitstransformation von A hervorgeht.

$$R = Q^\dagger A \tag{67}$$

$$A' = RQ \tag{68}$$

$$A' = Q^\dagger A Q \tag{69}$$

Das bedeutet, dass A' manche Eigenschaften von A behält, so z.B. die Eigenwerte. Durch Multiplizieren der Eigenvektoren von A' mit Q erhält man auch

die Eigenvektoren von A. Ist A eine Tridiagonalmatrix oder liegt in Hessenbergform vor, hat A' die gleiche Form. Es kann nur passieren, dass A' mehr Nullen besitzt. Die oben angegebene Umformung kann nun mit A' wiederholt werden, sodass

$$A' = Q'R' \tag{70}$$
$$A'' = R'Q' \tag{71}$$
$$A'' = Q'^{\dagger}A'Q' \quad . \tag{72}$$

Ziel der QR-Iteration ist es nun, die Umformungen so oft zu wieder holen, bis unsere Matrix in Dreiecksform vorliegt. Lt. [7] ex. ein Satz, der die Konvergenz der Reihe

$$A_1 = A\,, \quad A_2 = Q_1^{\dagger}A_1Q_1\,, \quad ...\,, \quad A_{k+1} = Q_k^{\dagger}A_kQ_k\,, ... \tag{73}$$

zu einer Dreiecksmatrix erklärt.
Doch nur in der einfachsten Variante wird A selbst Zerlegt. Im Allgemeinen wird dafür ein Polynom $p(A)$ verwendet, welches beim Einfachen-Shift die Form $p(A) = A - \mu I$ annimmt. Es wird somit die Diagonale von A mit einer geeigneten Zahl "verschoben". Die Idee dabei ist, dass bei geeigneter Wahl des Shifts μ schneller eine Konvergenz erreicht wird, wobei μ möglichst nah an einem Eigenwert von A liegen muss. Die Eigenvektoren bleiben bei solch einem Shift die gleichen.

$$Ax = \lambda x \tag{74}$$
$$Ax - \mu x = (\lambda - \mu)x \tag{75}$$
$$(A - \mu I)x = (\lambda - \mu)x \tag{76}$$

Der in ARPACK verwendete Algorithmus einer shifted QR-Iteration folgt lt. [6] folgendem Schema:

Input: (A, V, H) mit $AV = VH$, $V^{\dagger}V = I$ und H sei eine obere Hessenbergmatrix;
For $j = 1, 2, 3, ...$ bis zur Konvergenz,
    Wähle eine Verschiebung $\mu \leftarrow \mu_j$
    Bilde Faktoren $QR = \mathrm{qr}(H - \mu I)$ mit Householdertransformation;
    Setze $H \leftarrow Q^{\dagger}HQ$; $V \leftarrow VQ$;
End_For

qr() ist dabei eine Funktion, welche die QR-Zerlegung durchführt. Das angegebene Schema verwendet bereits die Hessenbergmatrix H aus der Arnoldi-Faktorisierung für die Zerlegung, da deren Eigenwerte (fast) gleich sind.

4.3 Implicitly restarted Arnoldi method (IRAM)

Die IRAM in ARPACK hat lt. [6] folgenden Ablauf:

- *Start:* Bilde eine Arnoldi-Faktorisierung der Länge $m \leq n$ von der gegebenen $n \times n$-Matrix A, $AV_m = V_mH_m + f_me_m^T$, mit dem Startvektor v_1.

- *Iteration:* Wiederhole, bis zur Konvergenz der k gesuchten Eigenwerte λ_j, $j = 1, ..., k$. Eine Konvergenz ist dann erreicht, wenn im Rahmen einer bestimmten Tolleranz Gl.(65) erfüllt wird, also $h_{m+1,m}$ einen bestimmten Wert unterschreitet.

  1. Berechne mit geringer Genauigkeit die Eigenwerte $\{\lambda_j : j = 1, 2, ..., m\}$ von der nun kleineren Matrix H_m mit einer beliebigen Methode. Sortiere diese Eigenwerte je nach Benutzervorgabe in ein gesuchtes Set von $k < m$ Eigenwerten $\{\lambda_j : j = 1, 2, ..., k\}$ und ein nicht gesuchtes Set von $p = m - k$ Eigenwerten $\{\lambda_j : j = k + 1, k + 2, ..., m\}$. Ein Beispiel für ein gesuchtes Set könnten die k betragsmäßig kleinsten Eigenwerte sein.
  2. Führe $p = m-k$ Schritte der QR-Iteration (siehe Schema unter **QR-Iteration**) auf H_m mit den nicht gesuchten Eigenwerten $\{\lambda_j : j = k+1, k+2, ..., m\}$ als Verschiebung durch (auch 'Multishift' genannt), um $H_m Q_m = Q_m H_m^+$ zu erhalten. Q_m ist eine orthonormale $m \times m$-Matrix. Hätten wir die QR-Iteration vollständig durchgeführt, wäre H_m^+ eine Dreiecksmatrix. Da wir die Iteration aber bereits nach p Schritten abgebrochen haben, ist H_m^+ wieder eine Hessenbergmatrix. Da H_m^+ aus einer Ähnlichkeitstransformation von H_m gewonnen wird, sind die Eigenwerte beider Matrizen gleich. Wir ersetzen H_m in der Arnoldi-Faktorisierung mit $Q_m H_m^+ Q_m^{-1}$:

$$A\,V_m = V_m Q_m H_m^+ Q_m^{-1} + f_m e_m^T \tag{77}$$

  3. *'Restart':* Q_k sei die Matrix, die aus den ersten k Spalten der Matrix Q_m besteht, ist somit ebenfalls orthonormal und hat die Größe $m \times k$. Wir verkürzen die Arnoldi-Faktorisierung von der Länge m auf die Länge k, in dem wir Gl.(77) von rechts mit der Matrix Q_k multiplizieren.

$$\begin{aligned} A\,V_m Q_k &= V_m\,(Q_m H_m^+ Q_m^{-1}) Q_k + (f_m e_m^T) Q_k \\ A\,V_m Q_k &= V_m\,(Q_k Q_k^{-1})\,(Q_m H_m^+ Q_m^{-1}) Q_k + (f_m e_m^T) Q_k \\ A\,V_m Q_k &= V_m\,Q_k\,H_k^+ + f_k^+ e_k^T \end{aligned} \tag{78}$$

     H_k^+ ist die vordere Untermatrix von H_m^+ der Ordnung k. Setze anschließend $V_k \leftarrow V_m Q_k$.

$$A\,V_k = V_k\,H_k^+ + f_k^+ e_k^T \tag{79}$$

     Gl.(79) ist jetzt eine Arnoldi-Faktorisierung der Länge $k < m$.
  4. Verlängere die Arnoldi-Faktorisierung der Länge k auf eine Faktorisierung der Länge m.

Nach der ersten Arnoldi-Faktorisierung werden die Eigenwerte von H_m bestimmt, um diese für die Verschiebung in der QR-Iteration zu verwenden. Es muss sich dabei nicht notwendiger Weise um Eigenwerte handeln, es können auch andere passende Werte verwendet werden.
Bei 2. wird der implicitly shifted QR-Algorithmus angewand, welcher jedoch nach dem $p$ten Schritt abgebrochen wird. Somit kann man sagen, dass die IRAM

eine gekürzte Form des implicitly shifted QR Algorithmus ist. Der Shift des QR-Algorithmus bewirkt, dass die Information des beliebig gewählten Startvektors langsam verloren geht.
3. und 4. beschreiben den sogenannten 'Restart', was nichts anderes macht, als die Arnoldi-Faktorisierung bei k abzuschneiden und die letzten $k+1, ..., m$ Faktoren neu zu berechnen. Der Algorithmus ist schematisch in [6] auf Seite 54 dargestellt.

4.4 ARPACK Software

ARPACK ist eine in Fortran77 geschriebene Programmbibliothek zur Lösung von Eigenwertproblemen großer Matrizen. Es ist besonders dazu geeignet Eigenwerte und -vektoren von großen schwach besetzten Matrizen A zu berechnen. Dazu verwendet ARPACK die zuvor vorgestellte spezielle Variante des Arnoldi-Verfahrens, die implicitly restarted Arnoldi method (IRAM), welche sich bei symmetrischen Matrizen auf das entsprechende Lanczos-Verfahren (IRLM) reduziert. Für die Eigenwertberechnung wird lediglich die Wirkung der Matrix auf einen Vektor benötigt, also das Matrix-Vektor-Produkt $b = Ax$. Diese Software berechnet k Eigenwerte, welche, je nach Benutzereingabe, bspw. den größten/kleinsten Betrag oder größten/kleinsten Realteil besitzen. Die IRAM verwendet dabei eine Faktorisierung der Matrix A, deren Teile leichter handhabbar sind, die wichtigen Eigenschaften wie Eigenwerte aber beibehalten. Zu dem liegt bei der eigentlichen Eigenwertberechnung der IRAM die QR-Zerlegung zu Grunde.

5 Programmtests

Das im Rahmen dieser Arbeit geschriebene Programm `eigenv_Laplace.cc` soll die Basis für extensive Tests der *distillation*-Methode sein und u.U. als Grundlage für zahlreiche zukünftige Spektrumsberechnungen dienen. Es existieren verschiedene Testverfahren, um die berechneten Größen zu überprüfen. Zum Einen kann das Eichfeld auf das Einheitseichfeld gesetzt werden, wodurch wir nur noch die Eigenwerte und -vektoren des Laplace-Operators berechnen, die uns analytisch bekannt sind. Zum Anderen können wir eine Eichtransformation durchführen, die die Eigenwerte des Problems nicht verändern darf.

5.1 Vorbereitungen für den kovarianten Laplace-Operator

Für die Eigenwertberechnung werden wir eine Programmbibliothek verwenden, die die Eigenwerte von Matrizen berechnet. Daher könnten wir unseren Operator in eine Matrixform bringen. Das Programm, dass die Eigenwerte berechnen soll, verwendet eine Funktion, in der die Matrix auf einen eingehenden Vektor angewandt werden muss. Somit ist es nicht zwingend erforderlich den Operator als Matrix zu formulieren. Für die Anwendung des Operators auf den Vektor, interpretieren wir jeweils drei hintereinander liegende Zahlen des Vektors als die drei Werte der Farbfelder an dem gleichen Punkt. Da das 3-dimensionale Gitter auf dem Vektor abgebildet werden muss, schreiben wir uns eine Funktion, mit der wir auf die entsprechenden Stellen des Vektors zugreifen, wenn wir die

Raumkoordinaten x, y und z angeben.

$$\begin{aligned} xx &= (x + L)\%L\ ; \\ yy &= (y + L)\%L\ ; \\ zz &= (z + L)\%L\ ; \\ \texttt{return}\ &((xx \cdot L + yy) \cdot L + zz) \cdot 3\ ; \end{aligned} \tag{80}$$

Die Definitionen der Variablen xx, yy und zz resultieren aus den periodischen Randbedingungen. Im Prinzip schneiden wir unser Gitter entlang der z-Achse in Streifen und legen sie hintereinander auf den Vektor.
Wenden wir nun den kovarianten Laplace-Operator auf den Vektor bzw. das Gitter an, durchlaufen wir alle Gitterpunkte, also alle x-y-z-Kombinationen und berechnen dort nach Gl.(58), nur in alle drei Raumrichtungen, die zweite kovariante Ableitung.

5.2 Test mit Laplace-Operator

Um die Rechnung für den Laplace-Operator durch zu führen, müssen wir sämtliche Links Eins setzen.
Da die drei Farbfelder jetzt nicht mehr aneinander koppeln, betrachten wir gleichzeitig drei identische entkoppelte Eigenwertprobleme. Somit erhalten wir jeden Eigenwert drei mal. Die bekannten Entartungesgrade (siehe Gl.(46) - (55)) werden nochmal mit drei multipliziert.

```
-1.09229881854063e-16     1.94526247861650e-17
-1.22142398726614e-15     3.93507208683561e-17
-1.82579794999063e-15    -9.24919957077177e-17
-1.38196601125010e+00    -6.19909366978576e-17
-1.38196601125010e+00     3.82354973729776e-18
-1.38196601125010e+00     3.18793906072923e-17
-1.38196601125011e+00     2.63783642254242e-17
-1.38196601125011e+00     5.55111512312585e-17
-1.38196601125011e+00    -3.18755438710738e-17
-1.38196601125011e+00    -3.43471860111619e-18
-1.38196601125011e+00    -4.51029324744412e-17
-1.38196601125011e+00    -7.27526111910542e-17
-1.38196601125011e+00    -6.93943290877084e-18
-1.38196601125011e+00    -5.98477977821352e-17
-1.38196601125011e+00     6.93912813067179e-18
-1.38196601125011e+00    -8.30349574566943e-17
-1.38196601125011e+00    -3.81638743803147e-17
-1.38196601125012e+00     5.75690880615747e-18
-1.38196601125012e+00     7.77508391667269e-18
-1.38196601125012e+00    -2.42861393951722e-17
-1.38196601125012e+00    -3.81625394856181e-17
-2.76393202250021e+00     3.69827531345559e-17
```

Listing 1: die 22 betragsmäßig kleinsten Eigenwerte des Laplace-Operators auf einem Gitter mit $L = 5$

In Liste (1) sieht man das Ergebnis des Programms für die ersten 22 Eigenwerte bei einer Gittergröße von $L = 5$. Die Eigenwerte sind von oben nach unten aufgelistet, wobei links der Realteil und rechts der entsprechende Imaginärteil steht. Die ersten drei Werte sind nahezu Null und somit die dreifache Entartung des ersten Eigenwerts. $-1,38$ taucht 18 mal auf, was die dreifache Entartung

der nächsten 6 gleichen Eigenwerte entspricht. Berechnet man mehr Eigenwerte, dann sieht man, dass der nächste Eigenwert von $-2,76\ 36$ mal auftaucht, also der nächste Eigenwert mit der 12-fachen Entartung ist und anschließend der Eigenwert $-3,62\ 18$ mal auftritt und somit der vierte Eigenwert mit einer Entartung von 6 ist. Die Anzahl der Entartungen stimmen demnach mit den Erwartungen überein.

Nun muss noch überprüft werden, ob die Zahlenwerte korrekt sind. Die analytische Lösung hat gezeigt, dass alle Eigenwerte reell und negativ bzw. Null sind. Dies spiegelt die Rechnung ebenfalls wider, da die Imaginärteile extrem klein sind. Die Realteile der gleichen Eigenwerte unterscheiden sich erst in der 14. Nachkommastelle, was als erste Abschätzung für die Genauigkeit der Rechnung genutzt werden kann.
Sehen wir uns die Werte konkret an, dann sehen wir, dass

$$\lambda_1 = -\left(\frac{4}{a^2}\sin^2(\frac{\pi}{N}\cdot 1)\right) = -1,38196601125011 \tag{81}$$

$$\lambda_2 = -\left(\frac{4}{a^2}\sin^2(\frac{\pi}{N}\cdot 2)\right) = -2,76393202250021 \tag{82}$$

und somit die berechneten Werte mit der analytischen Lösung (Gl.(42)) übereinstimmen.
Die Eigenwerte des Laplace-Operators wurden somit auf eine Genauigkeit von 13 Nachkommastellen korrekt berechnet.

5.3 Testen der Eigenwerte durch Eichtransformation

Für das Eigenwertproblem des kovarianten Laplace-Operators existiert i.A. keine analytische Lösung, deshalb muss eine andere Methode zum Testen verwendet werden. Dazu wenden wir auf unseren kovarianten Laplace-Operator eine beliebige Eichtransformation g an.

$$\sum_x D^2(y,x)\Psi(x) = \lambda\Psi(y) \tag{83}$$

Eichtransformation:

$$D^2(y,x) \quad \rightarrow \quad g(y)D^2(y,x)g^\dagger(x) = D'^2(y,x) \tag{84}$$

Für D'^2 können wir nun das gleiche Problem formulieren, wie in Gl.(83) für D^2.

$$\sum_x D'^2(y,x)\Psi'(x) = \lambda'\Psi'(y) \tag{85}$$

$$\sum_x g(y)D^2(y,x)g^\dagger(x)\Psi'(x) = \lambda'\Psi'(y) \tag{86}$$

$$\sum_x D^2(y,x)g^\dagger(x)\Psi'(x) = \lambda' g^\dagger(y)\Psi'(y) \tag{87}$$

Wenn wir nun das Problem in Gl.(85) gelöst haben, dann haben wir mit $g^\dagger(x)\Psi'(x) = \Psi(x)$ und $g^\dagger(y)\Psi'(y) = \Psi(y)$ das Problem in Gl.(87) und damit auch Gl.(83) gelöst. Dabei ist klar, dass $\lambda' = \lambda$ ist und somit die Eigenwerte unter einer Eichtransformation des kovarianten Laplace-Operators invariant sind. Wenn wir

die Eigenwerte nach einer Eichtransformation des kovarianten Laplace-Operators erneut berechnen, dürfen sich diese also nicht verändern.

Die Listen (2) und (3) sollen dies bestätigen. Liste (2) zeigt die Eigenwerte des kovarianten Laplace-Operators mit einem Test-Eichfeld, welches zufällige Links enthält, auf einer Gitterlänge von $L = 4$. Liste (3) zeigt die Eigenwerte nach der Eichtransformation. Man sieht, dass die Eigenwerte bis zur 13. Nachkommastelle übereinstimmen. Dies ist auch die Rechengenauigkeit, die wir in Abschnitt 5.2 ermittelt haben.

```
−1.27104747608827e+00    1.52950661074386e−17
−1.38476667185735e+00    1.37780070168388e−17
−1.46363536906659e+00   −2.90845665499169e−17
−1.51387878340712e+00    6.96684784508170e−18
−1.61885754459129e+00   −3.67207537924523e−17
−1.64179664812256e+00    1.51818279222626e−17
−1.77143096368542e+00   −4.74471257319985e−17
−1.81324655304078e+00    1.34546148451049e−16
−1.92499811454916e+00   −2.17229134642526e−17
−1.97695977896400e+00   −1.25189779436674e−17
−2.05133089686205e+00    1.36557146673142e−17
−2.12899607259574e+00    1.99074157651582e−17
−2.16408041488254e+00   −3.09210283268285e−17
−2.22244531093147e+00    7.52625306742143e−18
−2.28022614880702e+00   −2.52752959560414e−17
−2.34147146841391e+00   −6.82235667699491e−17
−2.42806553358132e+00   −7.35850324829795e−17
−2.52583772514888e+00   −5.29949451134421e−17
−2.57950641684425e+00   −6.30047427516106e−17
−2.60599860679130e+00    6.86187323771561e−17
−2.69344604159013e+00   −2.64363687828814e−18
−2.74701709988201e+00    4.67291247325606e−17
```

Listing 2: die 22 betragsmäßig kleinsten Eigenwerte des kovarianten Laplace-Operators auf einem Gitter mit $L = 4$ und einem Test-Eichfeld

```
−1.27104747608825e+00   −6.55039325790423e−18
−1.38476667185734e+00   −4.69707535797765e−17
−1.46363536906658e+00    2.94154861719881e−17
−1.51387878340712e+00   −8.48556172953974e−17
−1.61885754459128e+00   −2.83492352514394e−17
−1.64179664812257e+00   −1.02172468103264e−16
−1.77143096368542e+00    1.96011000229739e−17
−1.81324655304077e+00    4.78669702399440e−17
−1.92499811454914e+00   −2.36317545044273e−17
−1.97695977896397e+00    5.71897154934243e−17
−2.05133089686204e+00    1.75510282895345e−17
−2.12899607259572e+00   −9.45960629138978e−18
−2.16408041488251e+00    6.45938452017174e−18
−2.22244531093146e+00    3.88097928230964e−17
−2.28022614880701e+00   −3.53857889504383e−17
−2.34147146841388e+00    2.66142393146409e−19
−2.42806553358130e+00   −1.49619423616356e−17
−2.52583772514888e+00   −1.19694917148766e−17
−2.57950641684426e+00   −5.33025318155917e−18
−2.60599860679130e+00   −1.30098316820798e−17
−2.69344604159010e+00   −3.52598711939959e−17
−2.74701709988200e+00    1.47946588562671e−17
```

Listing 3: die 22 betragsmäßig kleinsten Eigenwerte des kovarianten Laplace-Operators auf einem Gitter mit $L = 4$ nach einer zufälligen Eichtransformation

5.4 Test der Eigenvektoren

Nachdem wir die Eigenwerte überprüft haben und bestätigen konnten, dass diese korrekt berechnet wurden, können jetzt die Eigenvektoren überprüft werden. Dazu setzen wir die berechneten Eigenvektoren in die Eigenwertgleichung (18) ein, berechnen beide Seiten separat und vergleichen diese miteinander. Beide Seiten stimmen mit einer Genauigkeit von 13 Nachkommastellen überein.

```
left side:
 -0.001255241622713  -0.000757314931795
 -0.055918976256505  +0.002771172263881
 +0.038208569555342  -0.107362107041221

right side:
 -0.001255241622713  -0.000757314931795
 -0.055918976256506  +0.002771172263881
 +0.038208569555343  -0.107362107041223
```

Listing 4: Berechnung der einzelnen Seiten des EW-Problems mit einem Test-Eichfeld auf einem Gitter mit $L = 4$, an der Stelle $(t, x, y, z) = (0, 2, 1, 0)$, die drei Zeilen entsprechen den drei Farbkomponenten

5.5 Laufzeitverhalten bei unterschiedlichen Gittergrößen

Die Programmbibliothek ARPACK stellt eine Laufzeit proportional zur Matrixgröße n mal der Anzahl zu berechnender Eigenwerte k in Aussicht [6]. Die Laufzeiten wurden mit komplexen Matrizen der Größe $n = 4^3 \times 3 = 192$ bis $n = 32^3 \times 3 = 98\,304$ getestet[1]. Die in der Abb.(1) dargestellten Laufzeiten beziehen sich auf Rechnungen mit je 48 Zeitschichten, wobei immer 21 Eigenwerte mit Eigenvektoren berechnet wurden. Die vorhergesagte lineare Abhängigkeit

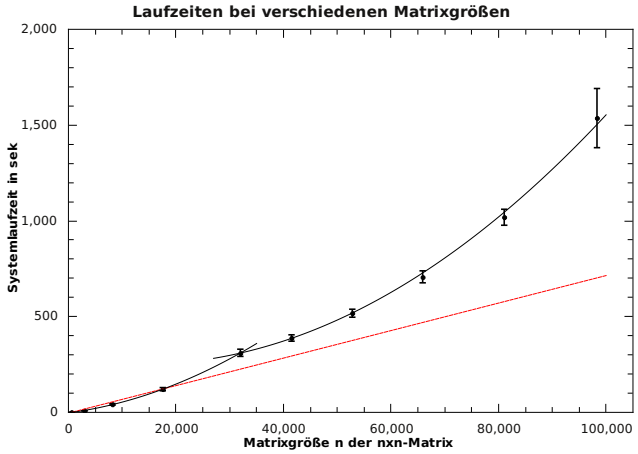

Abbildung 1: Laufzeitverhalten des Programms `eigenv_Laplace`, zur Bestimmung der EW und EV des kovarianten Laplace-Operators, bei unterschiedlichen Matrixgrößen

[1] Die Rechnungen wurden auf einem Rechner mit 4 CPUs mit je 2,66GHz und einem Arbeitsspeicher von 4GB durchgeführt.

konnte nicht bestätigt werden (rote Linie), die Laufzeiten verhalten sich in Teilen eher quadratisch. Ein Grund dafür könnte die Struktur der Matrix sein, da in [6] keine Aussagen darüber getroffen wird, unter welchen Bedingungen die Laufzeit des Programms linear ansteigt. Ein anderer Grund könnte in der Hardware, z.B. dem Speicher, liegen, da für große Matrizen auf eine große Menge an Speicher zugegriffen werden muss. Für die Analyse des Laufzeitverhaltens wurde das Einheitseichfeld verwendet. Bevor die Eigenwerte und -vektoren berechnet wurden, wurde eine zufällige Eichtransformation mit dem Eichfeld durchgeführt.

Die Abhängigkeit von der Anzahl k der zu berechnenden Eigenwerte und -vektoren wurde mit einer Matrix der Größe $n = 8^3 \times 3 = 1536$ und 48 Zeitschichten durchgeführt. In Abb.(2) ist die Abhängigkeit des Rechenaufwands von der Anzahl zu berechnender Eigenwerte abgetragen, welche zwischen 14 und 94 variiert wurde. Die Standardabweichung ist recht groß, was an der zufäl-

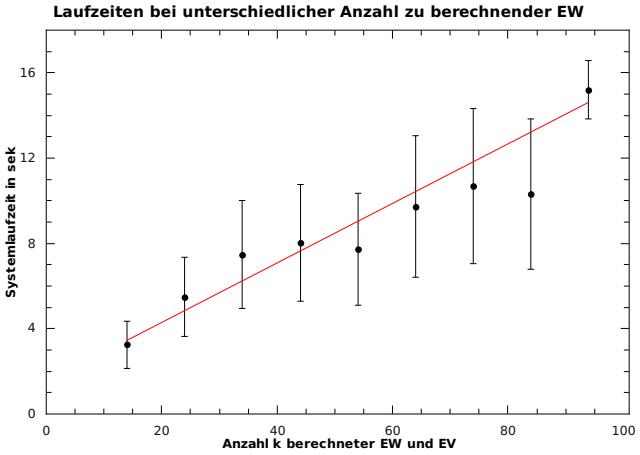

Abbildung 2: Laufzeitverhalten des Programms `eigenv_Laplace`, zur Bestimmung der EW und EV des kovarianten Laplace-Operators, bei unterschiedlichen Anzahlen zu berechnender EW und EV

ligen Eichtransformation liegen kann. Der Rechenaufwand ist somit stark von den Einträgen der Matrix abhängig. Im Groben kann ein linearer Anstieg angenommen werden. Die selbstgeschriebene Funktion `av` ist nicht von der Anzahl zu berechnender Eigenwerte abhängig und somit nicht für die Rechenzeit in Abhängigkeit von k verantwortlich. Die Proportionalität zu k resultiert aus der k-Abhängigkeit von `ncv`, da mit steigendem k mehr Arnoldi-Faktorisierungen durchgeführt werden.

6 Ausblick

Die Eigenvektoren des Gitter-Laplace-Operators können nun effizient und genau bestimmt werden. Um den Smearing-Operator zu bestimmen, muss eine gewisse Anzahl von Eigenvektoren zu allen Zeiten bestimmt werden. Pro Zeitschicht werden die Eigenvektoren spaltenweise in einer Matrix angeordnet. Diese wird mit dem Adjungierten multipliziert. Somit erhält man für jede Zeitschicht den

Smearing-Operator $\Box(t)$.

$$\Box_{\alpha\beta}(t) = \sum_{k=1}^{M} v_{\alpha}^{(k)}(t) v_{\beta}^{(k)\dagger}(t) \tag{88}$$

Dieser wird mit dem Erzeugungsoperator $\mathcal{O}^{\dagger}(0)$ und dem Vernichtungsoperator $\mathcal{O}(t)$ zu einer Korrelationsfunktion verrechnet. Wie viele Eigenvektoren im Smearing-Operator verarbeitet werden müssen, muss getestet werden. Es gilt jedoch, dass schon wenige Eigenvektoren ausreichen, um die Konvergenz erheblich zu verbessern.

Eine Verwertung der hier erzielten Ergebnisse ist bereits in Planung, um, angelehnt an [2], die Masse eines Mesons (z.B. π-Meson) zu bestimmen.

A Programmdokumentation

Dies ist die Dokumentation des Programms *eigenv_Laplace*, ein in C++ geschriebenes Programm zur Berechnung der Eigenwerte und -vektoren des kovarianten Laplace-Operators. Dabei wird eine in Fortran77 geschriebene Programmbibliothek namens ARPACK verwendet, welche die implicitly restarted Arnoldi Methode (IRAM) verwendet. *eigenv_Laplace* ist eingebettet in einen von Marc Wagner[2], Carsten Urbach[3] und weiteren geschriebenen Code namens *ContractionCode*, wovon viele Funktionen genutzt und einige hinzugefügt wurden. Das Programm wurde im Rahmen einer Bachelorarbeit von Johannes R. Siefert[4] entwickelt zum Thema: "Eigenmoden des kovarianten Laplace-Operators als Hilfsmittel zur Berechnung hadronischer Korrelatoren"

A.1 Aufgabe des Programms

Das Programm berechnet die ersten betragsmäßig kleinsten Eigenwerte und die zugehörigen Eigenvektoren des kovarianten Laplace-Operators, welcher der Laplace-Operator unter Beachtung eines Eichfeldes ist. Der Laplace-Operator ist die zweite räumliche Ableitung und kann folgender Maßen dargestellt werden:

$$\Delta f(x,y,z,t) = \nabla^2 f(x,y,z,t) \tag{89}$$
$$= \partial_{xx} f(x,y,z,t) + \partial_{yy} f(x,y,z,t) + \partial_{zz} f(x,y,z,t) \tag{90}$$

Da es sich um eine Rechnung auf dem Gitter handelt, werden die Raum- und Zeitschritte diskretisiert. Wenn wir nur eine der zweiten Ableitungen betrachten, dann können wir diese wie folgt darstellen:

$$\nabla^2 f(x_0) \to \frac{1}{a^2}\left(f(x_0-a) - 2f(x_0) + f(x_0+a)\right) \tag{91}$$

Mit a als Schrittweite des Gitters. Aufgrund des Eichfeldes, erhält die Funktion an den Punkten, die nicht genau am Ort der Ableitung liegen, einen Vorfaktor $U(x_0, x_0 \pm a)$. Dieser Vorfaktor U wird als Linker bezeichnet und ist spezifisch für die Nachbarpunkte eines Punktes. Das soll bedeuten, dass $U(x_0, x_0+a) \neq U(x_0+a, x_0)$. Stattdessen gilt beim Vertauschen der Argumente $U(x_0, x_0+a) = U^\dagger(x_0+a, x_0)$. Der kovariante Laplace-Operator hat somit die Form

$$D^2 f(x_0) = \frac{1}{a^2}\left(U(x_0, x_0-a) f(x_0-a) - 2f(x_0) + U(x_0, x_0+a) f(x_0+a)\right) \quad . \tag{92}$$

Werden die Linker auf Eins gesetzt, erhalten wir wieder den Laplace-Operator.

A.2 Struktur des Programms

Im Prinzip setzt sich *eigenv_Laplace* aus drei Dateien zusammen:
`eigenv_Laplace.cc` beinhaltet neben der main-Methode eine Funktion `av`, die von der zweiten Datei `eigenv_Laplace.h` benötigt wird. `eigenv_Laplace.h` führt einige Vorbereitungen durch und verwendet ARPACK zur Bestimmung der Eigenwerte und -vektoren. `DD_phi.cc` wird von `av` benötigt.

[2]Email: mcwagner@physik.hu-berlin.de
[3]Email: urbach@hiskp.uni-bonn.de
[4]Email: siefert@physik.hu-berlin.de

- `eigenv_Laplace.cc`:
  - läd das Eichfeld, bzw. setzt das Einheitseichfeld
  - reserviert Speicher für zwei Farbfelder
  - reserviert Speicher für die gesuchten Eigenwerte und -vektoren
  - führt `eigenv_Laplace.h` aus und erhält die berechneten EW und EV
  - speichert die Daten
  - ggf. Eichtransformation des Operators und erneute Berechnung
  - `av`: erhält input-Vektor, wendet mit Hilfe von `DD_phi.cc` den Operator an und gibt den Ergebnis-Vektor zurück
- `eigenv_Laplace.h`:
  - setzt die Parameter und reserviert Speicher
  - berechnet alle EW und EV mit *znaupd* von ARPACK und `av`
  - extrahiert die gesuchten EW und EV mit *zneupd* von ARPACK
  - speichert die Daten auf die reservierten Plätze
- `DD_phi.cc`:
  - berechnet psi=D^2phi auf dem gesamten Gitter

A.3 Parameter setzen in `eigenv_Laplace.cc`

Es können verschiedenartige Eichfelder geladen werden. Es bestehen bereits Muster für drei verschiedene Größen. Um die gewünschte Art von Eichfeld zu laden, muss das Kommentarzeichen vor dem entsprechenden `#define` entfernt werden. Werden mehrere Felder definiert, wird nur die letzte Definition akzeptiert. `__04X04_TEST__` ist dabei das kleinste Feld mit räumlichen und zeitlichen Ausmaßen von 4 und dient eher Testzwecken. `__16X32_B3_9_M0_0075__` hat räumliche Ausmaße von 16 in jede Raumrichtung und 32 Zeitschichten. `__24X48_B3_90_K0_160856_M0_0064__` läd ein Eichfeld für ein Gitter der Größe $24^3 \times 48$. In den Definitionen gibt `path[]` den Pfad des Eichfeldes an und `gauge_field_id` die Endung der Datei.
Weitere Parameter sind:
`gauge_field_on`:

> Gibt an, ob ein Eichfeld geladen oder ob das Einheitseichfeld genutzt werden soll. 'true' bedeutet, dass ein Eichfeld geladen wird, 'false' bedeutet, dass alle Linker auf Eins gesetzt werden.

`gauge_trafo_on`:

> Gibt die Möglichkeit an, ob die Rechnung nach einer Eichtransformation wiederholt werden soll. 'true' wiederholt die Rechnung nach der Eichtransformation und speichert die zusätzlichen Daten in einem anderen Ordner ab. 'false' unterlässt eine zweite Rechnung.

`time_slice`:

Gibt an, auf welcher Zeitschicht die EW und EV bestimmt werden sollen. Dabei ist zu beachten, dass $0 \leq time\_slice < T$ erfüllt ist.

`nev`:

Gibt die Anzahl an zu berechnenden EW an. Diese Zahl darf nicht zu groß sein, die obere Grenze muss ausgetestet werden.

A.4 Parameter setzen in `eigenv_Laplace.h`

Im Quellcode von `eigenv_Laplace.h` wurden alle gesetzten Parameter kommentiert. Trotzdem sollen die wichtigsten Einstellungen noch einmal kurz erklärt werden.

`n`:

Gibt die Ordnung der Matrix an, deren EW und EV berechnet werden sollen. Ist auf L*L*L*3 gesetzt, wobei die 3 aus den drei Farbfeldern resultiert.

`which[3]`:

Definiert die Art der gesuchten EW. Ist auf 'SM' ("smalest Magnitude") gesetzt und extrahiert somit die EW mit den kleinsten Beträgen. Weitere Optionen wären: 'LM' (größter Betrag), 'LA' (größter Realanteil), 'SA' (kleinster Realanteil), 'LI' (größter Imaginarteil), 'SI' (kleinster Imaginärteil)

`tol`:

Definiert die Genauigkeit der berechneten EW und EV. Ist auf 0 gesetzt, was als Maschienengenauigkeit umgesetzt wird.

`rvec`:

Gibt an, ob EV mitberechnet werden sollen oder nicht. Es ex. zwei Funktionen, eine mit 0 (ohne EV) und eine mit 1 (mit EV). Das vorliegende Programm ruft nur die Funktion mit EV-Berechnung auf.

Literatur

[1] H.J. Rothe. *Lattice gauge theories: an introduction*. World Scientific lecture notes in physics. World Scientific, 2005.

[2] Michael Peardon et al. A novel quark-field creation operator construction for hadronic physics in lattice QCD. *Phys. Rev.*, D80:054506, 2009.

[3] Michael E. Peskin and Dan V. Schroeder. *An Introduction To Quantum Field Theory (Frontiers in Physics)*. Westview Press, October 1995.

[4] Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[5] Yousef Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. Society for Industrial and Applied Mathematics, 2 edition, April 2003.

[6] C. Yang R. B. Lehoucq, D. C. Sorensen. ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods. 1997.

[7] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 3 edition, September 2007.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ort, Datum

Unterschrift