Implementierung des Clover Terms in CL^{2}QCD für Wilson Fermionen

Institut für Theoretische Physik
der Goethe Universität Frankfurt

Abschlussarbeit

zur Erlangung des akademischen Grades
Bachelor of Science

vorgelegt von

Maximilian Theilig
geboren am 03.11.1993 in Wiesbaden

im November 2016

Erstprüfer:	Prof. Dr. Owe Philipsen
Zweitprüfer:	Jun.-Prof. Dr. Marc Wagner

Zusammenfassung

Ziel dieser Bachelorarbeit ist die Implementierung des *Clover* Terms im Gitter-QCD Programm CL^{2}QCD der Arbeitsgruppe von Professor Owe Philipsen. Dadurch wird der Fehler der Wilson Fermionwirkung von $\mathcal{O}(a)$ zu $\mathcal{O}(a^2)$ reduziert. In dieser Arbeit wird sowohl die Theorie als auch die numerische Umsetzung beschrieben. Am Ende wird eine Möglichkeit vorgestellt, die Implementierung zu testen, und ein Ausblick für weitere mögliche Schritte zur Verbesserung der Diskretisierungsfehler gegeben.

Inhaltsverzeichnis

1. Einleitung

Atomkerne sind aus Protonen und Neutronen aufgebaut. Diese wiederum bestehen aus Quarks. Es gibt 6 verschiedene Quark-Flavours mit unterschiedlichen Massen. Sie wechselwirken über Austauschteilchen, den Gluonen, miteinander. Dies wird durch die **Quantenchromodynamik** (QCD) beschrieben. Sie ist Teil des Standardmodell der Elementarteilchenphysik, das alle bekannten Teilchen und ihre Wechselwirkungen erfolgreich beschreibt.

Eine besondere Eigenschaft der QCD ist die *asymptotische Freiheit*. Die Kraft bzw. die Kopplung zwischen den Teilchen nimmt mit steigender Energie ab. Der Grund hierfür ist die Selbstwechselwirkungen zwischen den Gluonen. Im Bereich hoher Energien lässt sich die QCD störungstheoretisch behandeln, d.h. man entwickelt nach der Kopplungskonstanten. Bei typischen Energieskalen der QCD (~200MeV) ist die Kopplung jedoch von der Größenordnung $\mathcal{O}(1)$ und diese Entwicklung ist nicht möglich. Deshalb benötigt man andere Methoden, um die QCD in diesem Energiebereich zu behandeln.

Eine *first principle* Methode ist, die Raumzeit auf einem hyperkubischem Gitter zu diskretisieren und die Rechnungen mit einem Computer auszuführen. Die dadurch enstehende Theorie nennt man **Gitter-QCD**. Das Gitter bewirkt einen Impuls-*cutoff* bei $1/a$, wobei a den Gitterabstand bezeichnet. Somit wirkt das Gitter als Regulator und die Theorie ist mathematisch wohldefiniert.
Man muss jedoch beachten, dass durch den endlichen Gitterabstand Diskretisierungsfehler entstehen. Für physikalische Ergebnisse muss man die Gitterberechnungen für $a \to 0$ extrapolieren. Ein großes Problem ist jedoch, dass die numerischen Berechnungen sehr aufwändig sind und nur Berechnung mit kleinen Gittern möglich sind. Insbesondere für leichte Quarks benötigt man bei kleinen Gitterabständen sehr große Gitter und die Simulationen benötigen einen großen Rechenaufwand. Da man bei größeren Gitterabständen jedoch Diskretisierungsfehler erhält, die linear mit dem Gitterabstand anwachsen, benötigt man Methoden, um diese Fehler zu minimieren.

In dieser Arbeit geht es darum den linearen Fehler der Wirkung im Gitter-QCD Programm CL^{2}QCD der Arbeitsgruppe von Professor Owe Philipsen zu eliminieren. Zunächst werden ausgehend von der Kontinuums-QCD (Kapitel 2) die wichtigsten Formeln der Gitter-QCD erklärt und die $\mathcal{O}(a)$-Verbesserung in

Kapitel 3 beschrieben. Danach werden in Kapitel 4 die numerischen Methoden vorgestellt. In Kapitel 5 wird zunächst das Programm CL^{2}QCD beschrieben und dann erklärt, wie und an welchen Stellen die Verbesserung implementiert wurde. Danach wird in Kapitel 6 eine Möglichkeit beschrieben die Implementierung zu testen, indem man eine Simulation mit analytischen Rechnungen vergleicht. Die Arbeit endet mit einem Ausblick, in dem die möglichen nächsten Schritte beschrieben werden.

2. Quantenchromodynamik im Kontinuum

In diesem Abschnitt werden die wichtigsten Eigenschaften und Formeln der Kontinuums-Quantenchromodynamik beschrieben, die für den Übergang zur Gitter-QCD benötigt werden.

QCD ist eine SU(3)-Eichtheorie, die die Quarks als Dirac-Spinorfelder $\psi(x)_c$ beschreibt. Hierbei ist $c = 1, 2, 3$ der Farbindex. Die Gluonen werden durch matrixwertige Eichfelder $A_\mu(x)$ beschrieben.
Die fermionische Wirkung im Euklidischen lautet:

$$S_F[\psi, \bar{\psi}, A] = \sum_{f=1}^{N_f} \int \mathrm{d}^4x \, \bar{\psi}^{(f)}(x) \Big(\gamma_\mu(\partial_\mu + iA_\mu(x)) + m^{(f)} \Big) \psi^{(f)}(x). \tag{2.1}$$

Hierbei bezeichnen γ_μ die euklidischen Gamma-Matrizen (siehe Abschnitt A.1). Über den Index μ wird summiert.
Die Wirkung ist invariant unter lokalen SU(3)-Transformationen, beschrieben durch

$$\psi(x) \longrightarrow \psi'(x) = \Omega(x)\psi(x), \quad \bar{\psi}(x) \longrightarrow \bar{\psi}'(x) = \bar{\psi}(x)\Omega(x)^\dagger, \quad \Omega(x) \in SU(3). \tag{2.2}$$

Hierfür müssen die Eichfelder $A_\mu(x)$ das folgende Transformationsverhalten haben:

$$A_\mu(x) \longrightarrow A'_\mu(x) = \Omega(x)A_\mu(x)\Omega(x)^\dagger + i(\partial_\mu\Omega(x))\Omega(x)^\dagger. \tag{2.3}$$

Der gluonische Teil der Wirkung lautet:

$$S_G[A] = \frac{1}{4g^2} \sum_{a=1}^{8} \int \mathrm{d}^4x \;\; F^{(a)}_{\mu\nu}(x) F^{(a)}_{\mu\nu}(x) \tag{2.4}$$

$$F^{(a)}_{\mu\nu} = \partial_\mu A^{(a)}_\nu(x) - \partial_\nu A^{(a)}_\mu(x) - f^{abc} A^{(b)}_\mu(x) A^{(c)}_\nu(x). \tag{2.5}$$

Aufgrund der SU(3)-Symmetrie gibt es $3^2 - 1 = 8$ Gluonfelder.
Die Wirkung der QCD im Kontinuum ist gegeben durch:

$$S_{\mathrm{QCD}}[\psi, \bar{\psi}, A] = S_F[\psi, \bar{\psi}, A] + S_G[A]. \tag{2.6}$$

3. Quantenchromodynamik auf dem Gitter

Als Erstes definiert man sich ein 4-dimensionales Raum-Zeit-Gitter

$$\Lambda = \{n = (n_1, n_2, n_3, n_4) | n_1, n_2, n_3 = 0, 1, ..., N-1; n_4 = 0, 1, ..., N_T - 1\}. \quad (3.1)$$

Hierbei bezeichnet N die Anzahl der Gitterpunkte der Raumrichtungen und N_T die Anzahl der Gitterpunkte in Zeitrichtung. Die Vektoren $n \in \Lambda$ beschreiben Raum-Zeit-Punkte die jeweils den Abstand a voneinander haben. Durch die folgenden formalen Ersetzungen lassen sich die Ausdrücke der Kontinuums-QCD für Fermionen naiv auf das Gitter übertragen:

$$x \in \mathbb{R}^4 \to an,\ n \in \Lambda \quad (3.2)$$

$$\psi(x) \to \psi(n) \quad (3.3)$$

$$\int \mathrm{d}^4 x \to a^4 \sum_{n \in \Lambda} \quad (3.4)$$

$$\partial_\mu f(x) \to \frac{f(n+\hat{\mu}) - f(n-\hat{\mu})}{2a}. \quad (3.5)$$

Hierbei bezeichnet $\hat{\mu}$ den Einheitsvektor in μ-Richtung.

3.1. Naive Diskretisierung von Fermionen

Die Quarks $\psi(n), \bar{\psi}(n)$ werden wie im Kontinuum, durch Spinoren mit Farb- und Flavourquantenzahl beschrieben. Wenn man die freie Fermionen-Wirkung aufs Gitter mit dem symmetrische Ausdruck für die Ableitung benutzt erhält man:

$$S_0[\bar{\psi}, \psi] = a^4 \sum_{n \in \Lambda} \sum_{f=1}^{N_f} \bar{\psi}^{(f)} \left(\sum_{\mu=1}^{4} \gamma_\mu \frac{\psi^{(f)}(n+\hat{\mu}) - \psi^{(f)}(n-\hat{\mu})}{2a} + m\psi^{(f)}(n) \right). \quad (3.6)$$

Der kinetische Term dieser Wirkung ist jedoch nicht invariant unter lokalen SU(3)-Transformation gegeben durch

$$\psi(n) \longrightarrow \psi'(n) = \Omega(n)\psi(n), \quad \bar{\psi}(n) \longrightarrow \bar{\psi}'(n) = \bar{\psi}(n)\Omega(n)^\dagger, \quad \Omega(n) \in SU(3). \quad (3.7)$$

Deshalb führt man Felder $U_\mu(n) \in SU(3)$, sogenannte Link Variablen, mit einem Richtungsindex μ und folgender Transformationsvorschrift ein:

$$U_\mu \longrightarrow U'_\mu = \Omega(n) U_\mu(n) \Omega(n+\hat{\mu})^\dagger. \tag{3.8}$$

Für den zweiten Teil des kinetischen Terms definert man auch Link Variablen mit einem negativen Richtungsindex:

$$U_{-\mu} \equiv U_\mu(n-\hat{\mu})^\dagger,\ U'_\mu(n) = \Omega(n) U_{-\mu}(n) \Omega(n-\hat{\mu})^\dagger. \tag{3.9}$$

Mit diesen Link Variablen U_μ und den obigen Transformationsvorschriften erhält man die eichinvariante, naive Fermionwirkung

$$S_F[\psi,\bar{\psi},U] = a^4 \sum_{f=1}^{N_f} \sum_{n\in\Lambda} \Bigg(\bar{\psi}^{(f)}(n) \sum_{\mu=1}^{4} \gamma_\mu \frac{U_\mu(n)\psi^{(f)}(n+\hat{\mu}) - U_{-\mu}(n)\psi^{(f)}(n-\hat{\mu})}{2a} + m^{(f)} \bar{\psi}^{(f)}(n)\psi^{(f)}(n) \Bigg). \tag{3.10}$$

3.2. Wilson's Eichwirkung

Die erste Formulierung einer Gittereichtheorie geht auf Kenneth Wilson zurück:

$$S_G[U] = \frac{2}{g^2} \sum_{n\in\Lambda} \sum_{\mu<\nu} \mathrm{Re}\{\mathrm{Tr}[\mathbb{1} - U_{\mu\nu}(n)]\}. \tag{3.11}$$

Hierbei bezeichnet $U_{\mu\nu}(n)$ ein sogenanntes Plaquette, das als Produkt von 4 Link Variablen definiert ist. Dies ist die kürzeste mögliche geschlossene Schleife auf dem Gitter

$$U_{\mu\nu}(n) = U_\mu(n) U_\mu(n+\hat{\mu}) U_{-\mu}(n+\hat{\mu}+\hat{\nu}) U_{-\nu}(n+\hat{\nu}). \tag{3.12}$$

Durch die Spur ist die Wirkung invariant unter Eichtransformationen (3.8). Definiert man die Eichfelder $A_\mu(n)$ auf dem Gitter durch $U_\mu(n) = \exp(iaA_\mu(n))$ und nimmt den Kontinuumslimes $a \to 0$, findet man, dass diese Gitter-Wirkung in (2.4) über. Der Diskretisierungsfehler ist hierbei $\mathcal{O}(a^2)$ [GL09, S.38f].

3.3. Wilson Fermionen

Die naive Fermionwirkung (3.10) lässt sich aufgrund der Bilinearität in den Fermionfelder in die Form

$$S_F[\psi,\bar{\psi},U] = a^4 \sum_{n,m\in\Lambda} \bar{\psi}(n) D(n|m) \psi(m) \tag{3.13}$$

bringen, wobei hier der naive Dirac Operator auf dem Gitter definiert ist durch

$$D(n|m) = \sum_{\mu=1}^{4} \gamma_\mu \frac{U_\mu(n)\delta_{n+\hat{\mu},m} - U_\mu(n-\hat{\mu})^\dagger \delta_{n-\hat{\mu},m}}{2a} + m\delta_{n,m}. \tag{3.14}$$

Betrachtet man nun dessen Inverse im Impulsraum mit $m = 0$, also den Propagator für masselose Fermionen findet man

$$\tilde{D}(p)^{-1}\Big|_{m=0} = \frac{-ia^{-1}\sum_\mu \gamma_\mu \sin(p_\mu a)}{a^{-2}\sum_\mu \sin^2(p_\mu a)} \xrightarrow{a\to 0} \frac{-i\sum_\mu \gamma_\mu p_\mu}{p^2}. \tag{3.15}$$

Im Gegensatz zum Kontinuum, wo es nur einen Pol bei $p = 0$ gibt, findet man auf dem Gitter 15 weitere unphysikalische Pole, die 15 unphysikalischen Fermionen, sogenannten *doublers*, entsprechen. Um diese ungewollten Pole zumindest im Kontinuumslimes zu entfernen, addierte Wilson einen zusätzlichen Term zum naiven Dirac Operator im Impulsraum. Dieser Term wirkt wie ein zusätzlicher Massenterm, sodass die *doublers* eine effektive Masse von

$$m_{\text{eff}} = m + \frac{2l}{a},\ l \in [1,4] \tag{3.16}$$

haben. Da die Masse der *doublers* im Limes $a \to 0$ divergiert, entkoppeln sie von der Theorie.
Der Wilson Dirac Operator ist dann gegeben durch:

$$D(n|m) = \left(m + \frac{4}{a}\right)\delta_{n,m} - \frac{1}{2a}\sum_{\mu=1}^{4}\Big((1-\gamma_\mu)U_\mu(n)\delta_{n+\hat{\mu},m} + (1+\gamma_\mu)U_\mu(n-\hat{\mu})^\dagger\delta_{n-\hat{\mu},m}\Big). \tag{3.17}$$

Die Wilson Wirkung hat einen Diskretisierungsfehler von $\mathcal{O}(a)$ (siehe Appendix A.2).
Eine weitere Darstellung des Wilson Dirac Operators ist gegeben durch:

$$D = C(\mathbb{1} - \kappa H)\,,\ \kappa = \frac{1}{2(m+4)}\,,\ C = m + 4 \tag{3.18}$$

$$H(n|m) = \sum_{\mu=\pm1}^{\pm4} (1-\gamma_\mu)U_\mu(n)\delta_{n+\hat{\mu},m}. \tag{3.19}$$

Man nennt dies *hopping* Darstellung, da H nur nächste Nachbarn miteinander verbindet. Die Konstante C spielt keine Rolle, da sie durch Neudefinition der Quarkfelder $\psi \to \sqrt{C}\psi$ absorbiert werden kann.

3.4. Das Pfadintegral in der Gitter-QCD

Mit diesen beiden Wirkungen kann man nun formal ein Pfadintegral und Korrelatoren zwischen zwei Operatoren definieren

$$\langle \mathcal{O}_2(t)\mathcal{O}_1(0)\rangle = \frac{1}{Z}\int \mathcal{D}[\psi,\bar{\psi}]D[U]e^{-S_F[\psi,\bar{\psi},U]-S_G[U]}\mathcal{O}_2[\psi,\bar{\psi},U]\mathcal{O}_1[\psi,\bar{\psi},U] \tag{3.20}$$

$$Z = \int \mathcal{D}[\psi,\bar{\psi}]D[U]e^{-S_F[\psi,\bar{\psi},U]-S_G[U]} \tag{3.21}$$

$$\mathcal{D}[\psi,\bar{\psi}] = \prod_{n\epsilon\Lambda}\prod_{f,\alpha,a} d\psi^{(f)}(n)_{\alpha,a}d\bar{\psi}^{(f)}(n)_{\alpha,a}\,,\ \mathcal{D}[U] = \prod_{n\epsilon\Lambda}\prod_{\mu=1}^{4} dU_\mu. \tag{3.22}$$

Diese Pfadintegrale lassen sich in den meisten Fällen nicht analytisch berechnen. Deshalb benutzt man Monte Carlo Simulationen, um die Integrale zu lösen. Diese Methoden werden in Kapitel 4 diskutiert.
Da sich die fermionischen Anteile des Erwartungswertes aufgrund ihrer quadratischen Form im Exponenten ausintegrieren lassen, teilt man den Erwartungswert einer Observablen $\mathcal{O}$ in Eichfeld- und Fermionanteil auf

$$\langle \mathcal{O}\rangle = \langle\langle \mathcal{O}\rangle_F\rangle_G, \tag{3.23}$$

$$\tag{3.24}$$

wobei der fermionische Anteil bzw. der Eichfeldanteil gegeben ist durch

$$\langle \mathcal{A}\rangle_F = \frac{1}{Z_F[U]}\int \mathcal{D}[\psi,\bar{\psi}]e^{-S_F[\psi,\bar{\psi},U]}\mathcal{A}[\psi,\bar{\psi},U]\,,\ Z_F[U] = \int \mathcal{D}[\psi,\bar{\psi}]e^{-S_F[\psi,\bar{\psi},U]}, \tag{3.25}$$

$$\langle \mathcal{B}\rangle_G = \frac{1}{Z}\int \mathcal{D}[U]e^{-S_G[U]}Z_F[U]\mathcal{B}[U]. \tag{3.26}$$

In den beiden folgenden Abschnitten wird beschrieben, wie die Integration über die Fermionfelder bzw. die Link Variablen mathematisch definiert ist.

Fermi-Statistik und Grassmann Zahlen

Um das Pauli-Prinzip zu erfüllen, müssen die Spinoren ψ und $\bar{\psi}$ antikommutierende Variablen sein. Diese Objekte nennt man Grassmann Zahlen. Die definierende Eigenschaft einer Menge von Grassmann Zahlen n_i , $i = 1, ..., n$ ist

$$\eta_i\eta_j = -\eta_j\eta_i\ \forall i,j = 1,...,n. \tag{3.27}$$

Weiterhin kann man eine Grassmann Algebra mit $2n$ Generatoren η_i, η_j , $i, j = 1, ..., n$ einführen. Hierbei antikommutieren alle Generatoren:

$$\eta_i\eta_j = -\eta_j\eta_i,\ \bar{\eta}_i\eta_j = -\eta_j\bar{\eta}_i,\ \bar{\eta}_i\bar{\eta}_j = -\bar{\eta}_j\bar{\eta}_i\ \forall i,j = 1,...,n. \tag{3.28}$$

Für Gauß-Integrale über Grassmann Zahlen ergibt sich die *Matthews-Salam Formel*

$$Z_F = \int \mathrm{d}\eta_n \, \mathrm{d}\bar{\eta}_n \ldots \mathrm{d}\eta_1 \, \mathrm{d}\bar{\eta}_1 \exp\left(\sum_{i,j=1}^{n} \bar{\eta}_i M_{ij} \eta_j \right) = \det[M]. \tag{3.29}$$

Für allgemeine *n-Punkt Funktionen* von Grassmann Zahlen findet man

$$\langle \eta_{i_1} \bar{\eta}_{j_1} \ldots \eta_{i_n} \bar{\eta}_{j_n} \rangle_F = \frac{1}{Z_F} \int \mathrm{d}\eta_n \, \mathrm{d}\bar{\eta}_n \ldots \mathrm{d}\eta_1 \, \mathrm{d}\bar{\eta}_1 \, \eta_{i_1} \bar{\eta}_{j_1} \ldots \eta_{i_n} \bar{\eta}_{j_n} \exp\left(\sum_{i,j=1}^{n} \bar{\eta}_i M_{ij} \eta_j \right) \tag{3.30}$$

$$= (-1)^n \sum_{P(1,2,\ldots,n)} \mathrm{sign}(P) (M^{-1})_{i_1, j_{P_1}} (M^{-1})_{i_2, j_{P_2}} \ldots (M^{-1})_{i_n, j_{P_n}}. \tag{3.31}$$

Detaillierte Ausführungen und Beweise der Aussagen findet man zum Beispiel in [GL09, S.103ff.].

Haar Maß

Für das Maß einer Link Variablen $dU_\mu(n)$ muss man ein neues Integrationsmaß einführen, damit die Integration über die Gruppen-Mannigfaltigkeit SU(3) definiert ist.
Die wichtigste Forderung an das Maß ist, dass es invariant unter Eichtransformation (3.8) ist, damit die Theorie eichinvariant ist:

$$\mathcal{D}[U] = \mathcal{D}[U'] \tag{3.32}$$

$$\mathrm{d}U_\mu(n) = \mathrm{d}U'_\mu(n) = \mathrm{d}(\Omega(n) U_\mu(n) \Omega(n + \hat{\mu})) \,. \tag{3.33}$$

Diese Forderung führt zum sogenannten *Haar Maß*. Es ist invariant unter Rechts- und Linksmultiplikation mit einem Gruppenelement

$$\mathrm{d}U = \mathrm{d}(UV) = \mathrm{d}(WU) \;\; \forall V, W \in \mathrm{SU}(3). \tag{3.34}$$

3.5. Symanzik Improvement und Clover Term

Wie in den vorherigen Abschnitten erläutert, erhält man Diskretisierungsfehler, wenn man die QCD-Wirkung aufs Gitter überträgt. Diese spielen bei Berechnungen auf dem Gitter eine wichtige Rolle, da man numerisch nur mit endlichem Gitterabstand a rechnen kann.
Da die Eichfeld-Wirkung schon von der Ordnung $\mathcal{O}(a^2)$ ist, kümmert man sich zunächst um eine Reduzierung des Fehlers der Wilson-Wirkung von $\mathcal{O}(a)$ zu $\mathcal{O}(a^2)$.
Eine systematische Möglichkeit hierfür ist die Verbesserung nach Symanzik

[Sym83]. Der Ausgangspunkt ist, dass man nahe des Kontinuumslimes die Gitterwirkung durch eine effektive, lokale Wirkung beschreibt:

$$S_{\text{eff}} = \int \mathrm{d}^4x \left(L^{(0)}(x) + aL^{(1)}(x) + a^2 L^{(2)}(x) + \mathcal{O}(a^3) \right). \tag{3.35}$$

Die Terme $L^{(k)}$, mit $k \geq 1$ sind hierbei die Korrekturterme zur Kontinuumswirkung. Diese Terme haben Dimension Länge$^{-(4+k)}$. Korrekturterme sind alle möglichen eichinvarianten, skalaren Operatoren der passenenden Dimension, die alle Symmetrien der Ausgangs-Lagrangedichte erfüllen. Für $L^{(1)}$ lauten sie:

$$L_1^{(1)}(x) = \bar{\psi}(x)\sigma_{\mu\nu}F_{\mu\nu}(x)\psi(x) \tag{3.36}$$

$$L_2^{(1)}(x) = \bar{\psi}(x)\overrightarrow{D}_\mu\overrightarrow{D}_\mu\psi(x) + \bar{\psi}(x)\overleftarrow{D}_\mu\overleftarrow{D}_\mu\psi(x) \tag{3.37}$$

$$L_3^{(1)}(x) = m\ \ \mathrm{Tr}[F_{\mu\nu}(x)F_{\mu\nu}(x)] \tag{3.38}$$

$$L_4^{(1)}(x) = m\ (\bar{\psi}(x)\overrightarrow{D}_\mu\psi(x) - \bar{\psi}(x)\overleftarrow{D}_\mu\psi(x)) \tag{3.39}$$

$$L_5^{(1)}(x) = m^2\ \bar{\psi}(x)\psi(x). \tag{3.40}$$

Benutzt man nun die klassische Feldgleichung $(\gamma_\mu D_\mu + m)\psi = 0$, lassen sich zwei Terme eliminieren.

$$L_1^{(1)} - L_2^{(1)} + 2L_5^{(1)} = 0\ ,\ L_4^{(1)} + 2L_5^{(1)} = 0 \tag{3.41}$$

Man wählt $L_2^{(1)}$ und $L_4^{(1)}$, da diese aufgrund der vorkommenden Ableitungen schwieriger zu implementieren sind. Weitherhin kann man $L_3^{(1)}$ und $L_5^{(1)}$ durch Neudefinition der nackten Parameter m und g einbauen, da diese Terme schon in der ursprünglichen Wirkung vorkommen.
Somit reicht es aus, den sogenannten *Pauli Term* $L_1^{(1)}$ zur Wilson-Wirkung zu addieren:

$$S_{\text{I}} = S_{\text{Wilson}} + c_{\text{sw}}a^5 \sum_{n\in\Lambda}\sum_{\mu,\nu=1}^{4} \bar{\psi}(n)\frac{i}{4}\sigma_{\mu\nu}\hat{F}_{\mu\nu}(n)\psi(n). \tag{3.42}$$

Den Feldstärketensor auf dem Gitter wählt man als Summe von Plaquettes

$$\hat{F}_{\mu\nu}(n) = \frac{1}{8a^2}(Q_{\mu\nu}(n) - Q_{\nu\mu}(n)) \tag{3.43}$$

$$Q_{\mu\nu}(n) \equiv U_{\mu,\nu}(n) + U_{\nu,-\mu}(n) + U_{-\mu,-\nu}(n) + U_{-\nu,\mu}(n) \tag{3.44}$$

$$\sigma_{\mu\nu} = \frac{i}{2}[\gamma_\mu, \gamma_\nu]. \tag{3.45}$$

Da dieser Terme die Form eines Kleeblatts besitzt, nennt man ihn *Clover* Term. Der Dirac Operator in *hopping* Darstellung lautet damit:

$$D(n|m) = \delta_{n,m} - \kappa \sum_{\mu=\pm1}^{\pm4} (1-\gamma_\mu)U_\mu(n)\delta_{n+\hat{\mu},m} + \frac{i}{2}c_{sw}\sigma_{\mu\nu}\hat{F}_{\mu\nu}(n)\delta_{n,m}. \tag{3.46}$$

Der reelle Koeffizient c_{sw} ist nach B. Sheikholeslami und R. Wohlert benannt, da sie die verbesserte Wirkung zuerst aufgeschrieben haben [SW85]. Um die verbesserte Wirkung benutzen zu können, muss noch der Koeffizient c_{sw} bestimmt werden. Für kleine Kopplungen g $\in [0.0, 1.1]$ wurde der Koeffizient auf nicht-pertubative Weise in [JS98] bestimmt:

$$c_{\mathrm{sw}} = \frac{1 - 0.454g^2 - 0.175g^4 + 0.012g^6 + 0.045g^8}{1 - 0.720g^2}. \tag{3.47}$$

4. Numerische Methoden

4.1. Hybrid Monte Carlo Algorithmus

In diesem Abschnitt wird beschrieben, wie man die bei der Berechnung von Vakuumerwartungswerten (z.B. Gleichung (3.20)) auftretenden Pfadintegrale numerisch behandelt.
Die fermionischen Anteile des Pfadintegrals lassen sich ausintegrieren (siehe Abschnitt 3.4) und man erhält die Fermiondeterminante als zusätzlichen Integranden. Für den Erwartungswert einer Observablen A bei $N_f = 2$ entarteten Quarkflavours gilt somit:

$$\langle A \rangle = \frac{1}{Z} \int \mathcal{D}[U] e^{-S_G[U]} \det\big[D^\dagger(m) D(m)\big] A[U]. \tag{4.1}$$

Da diese Integrale nur für kleinste Gitter analytisch zu lösen sind, benutzt man *importance sampling* um das Integral näherungsweise zu berechnen. Dazu benutzt man den Monte Carlo Algorithmus, der eine Markov Kette von N Konfiguration U_n erzeugt, die gemäß

$$\frac{1}{Z} e^{-S_G[U]} \det\big[D^\dagger(m) D(m)\big] \tag{4.2}$$

verteilt sind. Man nähert den Erwartungswert einer Observablen dann durch eine Summe an

$$\langle A \rangle \approx \frac{1}{N} \sum_{n=1}^{N} A[U_n]. \tag{4.3}$$

Man kann zeigen, dass der Fehler sich hierbei verhält wie $\mathcal{O}(1/\sqrt{N})$ [GL09, S.74]. Für $N \to \infty$ wird die Formel exakt.

Die Fermiondeterminante ist reell und positiv definit für eine gerade Anzahl von entarteten Quarks [GL09, S.187] und kann somit als Gewichtsfaktor für die Monte Carlo Simulation benutzt werden. Die Matrix hat jedoch $\mathcal{O}(10^{12})$ Einträge und die Berechnung der Determinanten ist deshalb numerisch zu aufwendig. Deshalb benutzt man sogennante Hybrid Monte Carlo Methoden. Hierfür verwendet man, dass sich die Fermiondeterminante als ein Integral über komplexwertige, bosonische Felder (Pseudofermionen) ϕ ausdrücken lässt:

$$\det\big[D^\dagger(m) D(m)\big] \propto \int \mathcal{D}[\phi^\dagger] \mathcal{D}[\phi] \exp\big(-\phi^\dagger (D^\dagger(m) D(m))^{-1} \phi\big). \tag{4.4}$$

Damit kann man eine effektive Wirkung

$$S_{\text{eff}}[\phi^\dagger, \phi, U] = S_G[U] + \phi^\dagger (D^\dagger(m) D(m))^{-1} \phi \tag{4.5}$$

definieren und erhält als Gewichtsfaktor für die Monte Carlo Simulation

$$\frac{1}{Z} e^{-S_{\text{eff}}[\phi^\dagger, \phi, U]}. \tag{4.6}$$

Die Invertierung des Dirac Operators $D(m)$ ist numerisch sehr aufwändig. Da dies jedoch in jedem Monte Carlo Schritt benötigt wird, ist es für eine effiziente Simulation wichtig, dass neue Konfigurationen mit einer sehr hohen Wahrscheinlichkeit akzeptiert werden. Um dies zu realisieren benutzt man den *molecular dynamics* Algorithmus. Dazu führt man eine Hamiltonfunktion

$$H[P, U] = \frac{1}{2} \operatorname{Tr}\left[P_\mu(x)^2\right] + S_{\text{eff}}[\phi^\dagger, \phi, U] \tag{4.7}$$

ein. Hierbei bezeichnet $P_\mu \in \mathfrak{su}(3)$ den zu den Feldern U_μ konjugierten Impuls. Die Zeitentwicklung des Systems ist dann durch die Hamiltonschen Bewegungsgleichungen gegeben:

$$\dot{P} = -\frac{\partial H}{\partial U} = -\frac{\partial S}{\partial U} \equiv F, \; \dot{U} = \frac{\partial H}{\partial P} = P. \tag{4.8}$$

F bezeichnet hierbei analog zur Mechanik die Kraft (*driving force).* Da die Hamiltonfunktion bzw. die Energie eine Konstante der Bewegung ist, befindet sich der Pfad der Konfigurationen (U, P) im Phasenraum auf einer Hyperfläche mit konstanter Energie. Dadurch wird jede so erzeugte Konfiguration am Ende eines Monte Carlo Schrittes akzeptiert, sofern man die Bewegungsgleichungen exakt integrieren kann.
Zur numerischen Integration der Bewegungsgleichungen benutzt man die *leapfrog* oder *second order minimal* Methode. Da hierbei jedoch Fehler entstehen und die neue Konfigurationen (U', P') sich nach der Zeitentwicklung nicht mehr zwangsläufig auf der Hyperfläche befinden, baut man einen Metropolis Schritt [MRR+53] ein, d.h. eine neue Konfiguration wird nur mit einer Wahrscheinlichkeit von

$$\min(1, \exp(H[U', P'] - H[U, P])) \tag{4.9}$$

akzeptiert. Durch diesen Schritt ist auch *detailed balance* erfüllt.

Zur Invertierung des Dirac Operators benutzt man numerisch einen *Krylov solver*, da die Matrix dünnbesetzt ist. Man verwendet entweder *Conjugate Gradient* (CG) oder *Bi-conjugate Gradient Stabilized* (Bi-CGStab).

Ein weiteres Problem ist die Ableitung nach einem SU(3)-Gruppenelement in den Bewgungsgleichungen (4.8). Dazu benutzt man, dass sich jedes Gruppenelement als Exponential schreiben lässt

$$U_\mu = \exp\left(\sum_{i=1}^{8} \omega_\mu^{(i)} T_i\right) \equiv \exp(iQ). \tag{4.10}$$

Hierbei bezeichnen T_i die Generatoren der Algebra $\mathfrak{su}(3)$. Sie sind spurfrei und antihermitesch. $\omega_\mu^{(i)}$ sind 8 reelle Parameter, die das jeweilige Gruppenelement eindeutig definieren. Hierzu sind $P_\mu^{(i)}$ die konjugierten Variablen. Diese lassen sich zu einem Element der Algebra kombinieren

$$P_\mu = \sum_{i=1}^{8} P_\mu^{(i)} T_i. \tag{4.11}$$

Die Ableitung entlang eines Elementes der Algebra lässt sich als

$$\nabla^{(i)} f(U) = \frac{\partial f(U)}{\partial \omega^{(i)}} = \frac{\partial}{\partial \omega} f(e^{i\omega T_i} U)\bigg|_{\omega=0} \tag{4.12}$$

definieren. Damit ergibt sich für die Kraft

$$F[\phi^\dagger, \phi, U] = \sum_{i=1}^{8} T_i \ \nabla^{(i)} S_{\text{eff}}[\phi^\dagger, \phi, U] \in \mathfrak{su}(3). \tag{4.13}$$

Der Algorithmus

Numerisch setzt man den Hybrid Monte Carlo Algorithmus mit *leapfrog* Integrationsschema wie folgt um [GL09, S197]. Ausgehend von einer gegebenen Eichfeldkonfiguration U_0 erzeugt man zunächst Pseudofermionfelder $\phi = D\chi$, wobei χ gemäß $\exp(-\chi^\dagger \chi)$ verteilt ist. Im nächsten Schritt generiert man sich die zu U_0 konjugierten Felder P_0 gemäß der Verteilung $\exp(-\operatorname{Tr}[P^2])$. Nun löst man die Hamiltonschen Bewegungsgleichungen (4.8). Man startet beim *leapfrog* Integrationsschema bei den konjugierten Felder mit einem halben Zeitschritt, führt dann $(n-1)$ volle Schritte aus und endet mit einem halben Zeitschritt. Bei U führt man die Integration in n vollen Zeitschritten aus:

$$P_{\frac{1}{2}} = P_0 - \frac{\varepsilon}{2} F[U, \phi]\bigg|_{U_0} \tag{4.14}$$

$$U_k = \exp\Big(i\varepsilon P_{k-\frac{1}{2}}\Big) U_{k-1}\,, \ P_{k+\frac{1}{2}} = P_{k-\frac{1}{2}} - \varepsilon\, F[U, \phi]\Big|_{U_k} \tag{4.15}$$

$$U_n = \exp\Big(i\varepsilon P_{n-\frac{1}{2}}\Big) U_{n-1}\,, \ P_n = P_{n-\frac{1}{2}} - \frac{\varepsilon}{2} F[U, \phi]\bigg|_{U_n}. \tag{4.16}$$

Zuletzt kommt der Metropolis Schritt 4.9. Man akzeptiert die neue Konfiguration (U', P'), wenn eine Zufallszahl $r \in [0, 1)$ kleiner ist als

$$\exp\Big(\operatorname{Tr}\big[P^2\big] - \operatorname{Tr}\big[P'^2\big] + S_G[U] - S_G[U'] + \phi^\dagger((DD^\dagger)^{-1} - (D'D'^\dagger)^{-1})\phi\Big)\Big). \tag{4.17}$$

4.2. Even-Odd Präkonditionierung

Einer der numerisch aufwändigsten Schritte des in Abschnitt 4.1 beschriebenen Algorithmus ist die Invertierung der Fermionmatrix. Man invertiert die Matrix jedoch nicht explizit, sondern berechnet die Multiplikation mit einer Quelle

$$Mx = b \;\Leftrightarrow\; x = M^{-1}b. \tag{4.18}$$

Um dieses Verfahren zu beschleunigen, benutzt man die even-odd Prekonditionierung. Hierfür teilt man das Raum-Zeit-Gitter in gerade und ungerade Gitterplätze auf. Eine allgmeine Fermionmatrix M kann in Blöcke aufgeteilt werden, die nur auf gerade bzw. ungerade Gitterplätze wirken, und in drei Matrizen zerlegt werden:

$$M = \begin{pmatrix} M_{ee} & M_{eo} \\ M_{oe} & M_{oo} \end{pmatrix} = \begin{pmatrix} 1 & M_{eo}M_{ee}^{-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} M_{ee} - M_{eo}M_{oo}^{-1}M_{oe} & 0 \\ 0 & M_{oo} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ M_{oo}^{-1}M_{oe} & 1 \end{pmatrix} = L\tilde{M}U. \tag{4.19}$$

Hierbei sind L und U obere bzw. untere Dreiecksmatrizen und können somit leicht invertiert werden. Dadurch vereinfacht sich das Ausgangsproblem (4.18) zu

$$L\tilde{M}Ux = b \;\Leftrightarrow\; \tilde{M}Ux = L^{-1}b \;\Leftrightarrow\; \tilde{M}\tilde{x} = \tilde{b} \tag{4.20}$$

mit $\tilde{x} = Ux$ und $\tilde{b} = L^{-1}b$. Somit muss nur das Gleichungssystem

$$(M_{ee} - M_{eo}M_{oo}^{-1}M_{oe})\tilde{x}_e = \tilde{b}_e \tag{4.21}$$

numerisch gelöst werden.

4.3. Householder Triangularisierung

Eine weitere Methode zur Invertierung ist die Householder Triangularisierung. Hierbei bringt man eine komplexe $n \times n$ Matrix A durch $(n-1)$ Transformationsmatrizen R_i auf obere Dreiecksform T:

$$T = R_{n-1}R_{n-2}...R_1A. \tag{4.22}$$

Die Matrizen R sind gegeben durch

$$R = 1 - 2\frac{u \otimes u^\dagger}{\|u\|^2}. \tag{4.23}$$

Für die k-te Transformations Matrix R_k benötigt man die k-te Spaltenvektor v der Matrix $R_{k-1}R_{k-2}...R_1A$. Der für Gleichung (4.23) benötigte Vektor u ist

dann gegeben durch

$$u_l = \begin{cases} 0 & \text{für } l < k \\ v_k - y_k & \text{für } l = k \\ v_l & \text{für } l > k \end{cases}, \tag{4.24}$$

wobei y_k gegeben ist durch

$$y_k = -\frac{v_k}{|a|} r, \qquad r^2 = \sum_{j=k}^{n} |v_j|^2, \qquad r \geq 0. \tag{4.25}$$

R_k sorgt dafür, dass in der k-ten Spalten der Matrix $R_k \cdot R_{k-1} R_{k-2} ... R_1 A$ unterhalb der Diagonalen alle Einträge 0 sind.
Die Inverse der Matrix A ist dann gegeben durch

$$A^{-1} = T^{-1} R_{n-1} R_{n-2} ... R_1. \tag{4.26}$$

Für die Invertierung von T benutzt man, dass die Inverse einer oberen Dreiecksmatrix auch eine obere Dreiecksmatrix ist. Bezeichnet man die Inverse von T mit S und die Matrixelemente mit t_{ij} bzw. s_{ij}, lässt sich die Inverse durch

$$\sum_{j=i}^{k} t_{ij} s_{jk} = \delta_{ik} \tag{4.27}$$

bestimmen. Daraus folgt sofort, dass für die Diagonalelemente $s_{ii} = 1/t_{ii}$ gilt. Für die Elemente oberhalb der Diagonalen ($i < k$) kann man (4.27) in

$$s_{ik} = -s_{ii} \sum_{j=i+1}^{k} t_{ij} s_{jk} \tag{4.28}$$

umschreiben. Diese Gleichung löst man rekursiv, indem man bei $k = n$ beginnt und bis $k = 1$ geht. Für jedes k startet man bei $i = k - 1$ und geht bis $i = 1$.

Berechnung der Determinante

Mithilfe der Householder Triangularisierung lässt sich auf einfache Weise die Determinante einer Matrix berechnen. Die Zerlegung der Matrix in die Transformationsmatrizen R_k und die obere Dreiecksmatrix R lautet:

$$A = R_1 \dots R_{n-1} T. \tag{4.29}$$

Mit dem Determinantenmultiplikationssatz gilt

$$\det[A] = \det[R_1 \dots R_{n-1} T] = \det[R_1] \dots \det[R_{n-1}] \det[T]. \tag{4.30}$$

Die Determinante der Transformationsmatrizen lässt sich mit dem Matrix Determinanten Lemma (siehe Appendix A.3) bestimmen

$$\det[R] = \det\left[1 - 2\frac{u \otimes u^{\dagger}}{\|u\|^2}\right] = -1 \ \forall u \in \mathbb{C}^n. \tag{4.31}$$

Da T eine obere Dreiecksmatrix ist, ergibt sich die Determinante als Produkt der Diagonaleinträge

$$\det[T] = \prod_{i=1}^{n} t_{ii}. \tag{4.32}$$

Damit gilt für die Determinante der Matrix A

$$\det[A] = (-1)^{n-1} \prod_{i=1}^{n} t_{ii}, \tag{4.33}$$

wobei n die Dimension der Matrix ist.

5. Implementierung in CL^{2}QCD

In diesem Kapitel werden die grundlegenden Eigenschaften des Gitter-QCD Programms CL^{2}QCD erklärt und beschrieben, wie die verbesserte Wirkung (3.42) implementiert wurde.

5.1. Einführung in CL^{2}QCD

CL^{2}QCD[1] ist ein Gitter-QCD Programm, das auf OpenCL[2] basiert. Dadurch läuft es gleichzeitig sowohl auf CPUs als auch auf GPUs. Das Programm wurde von Dr. Christopher Pinke, Dr. Matthias Bach und anderen entwickelt [PPSB15]. Da GPUs von ihrer Konstruktion her dafür ausgelegt sind, viele Prozesse parallel auszuführen, eignen sie sich hervorragend zur Simulation von Gittern.
CL^{2}QCD hat folgende Funktionen eingebaut:

- Mehrere Fermion Diskretisierungen
  - Wilson
  - Twisted-Mass
  - Staggered
- Simulation reiner Eichtheorien (Heat Bath, Monte Carlo)
- Invertierungs- und Integrationsalgorithmen
- Pseudo-Zufallszahlengenerator RANLUX
- Even-Odd Präkonditionierung
- und vieles mehr...

Das OpenCL Programm besteht aus zwei Teilen, dem host Programm und den OpenCL-Funktionen, sogenannten Kernels. Die Struktur des Programms ist in Abbildung 5.1 dargestellt.
Das **host** Programm ist in C++11 geschrieben. Es besteht wiederum aus drei Teilen. In *meta* befinden sich die verschiedenen Parameter der Simulationen,

[1] CL^{2}QCD ist auf Github erhältlich: `https://github.com/CL2QCD/cl2qcd`
[2] Website des Projekts: `https://www.khronos.org/opencl/`

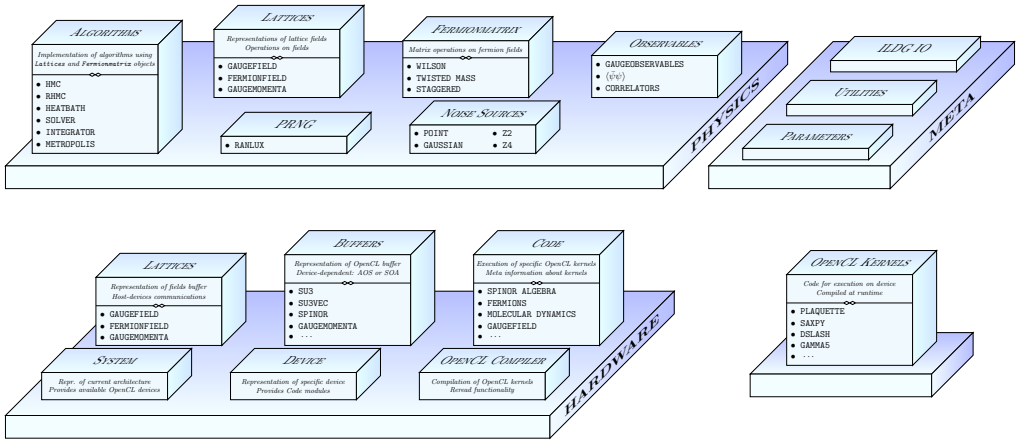

Abbildung 5.1.: Aktuelle Struktur des Codes.
Mit freundlicher Genehmigung von M. Bach, O. Philipsen, C. Pinke, and A. Sciarra “CL2QCD - Lattice QCD based on OpenCL”, in The XXXII International Symposium on Lattice Field Theory, Lattice2014, 2014 [PPSB15].

die Input/Output-Funktionalitäten und die Funktionen zur Kontrolle des Programms.
Das Packet *physics* besteht aus *fermionmatrix*, *lattices*, *observables* und *algorithms*. Hier befinden sich die verschiedenen Fermionmatrizen, die für die Krylov Solver benötigt werden. Sie sind sowohl für die verschiedenen Fermiondiskretisierungen als auch mit oder ohne even-odd Präkonditionierung implementiert. *Lattices* enthält die verschiedenen Gitterfelder, wie Spinor- und Eichfelder, als Klassen. Aktuell sind im Code Eich-, Wilson- und Staggeredobservablen implementiert. In *algorithms* sind die Rountinen geschrieben, die mit diesen Objekten arbeiten. So befinden sich hier zum Beispiel die Berechnung der Kraft und die Methoden zur Invertierung.
In *hardware* werden diese Objekte aus *physics* mit der Hardware zusammengeführt. Dazu gehört, dass die Speicherung der verschiedenen Gitterfelder an die jeweilige Hardware angepasst wird, um effizienter zu arbeiten. Weiterhin werden die OpenCL-Kernels und ihre Eingabeparameter initialisiert.
Alle mathematischen Operationen und tatsächlichen Simulationen werden von den **OpenCL-Kernels** ausgeführt. Das bedeutet, dass sowohl einfachste Rechnungen, wie Matrixmultiplikationen, als auch komplexe Operationen, wie die Berechnung der Kraft des Hybrid Monte Carlo Algorithmus, hier ausgeführt werden. Hierbei ist zu beachten, dass es einen Unterschied gibt zwischen den Kernels und einfachen Funktionen. Nur die Kernels können vom host Part aus direkt aufgerufen werden. Die Funktionen dienen dazu, den Code übersichtlicher zu machen und zu strukturieren. So befinden sich in den Dateien, die mit *operations_* beginnen, die Funktionen für die grundlegenden Rechenoperatio-

nen der verschiedenen Objekte.
Für weitere Information über die Struktur des Programmes und die Optimierung für die jeweilige Hardware siehe [PPSB15], [Pin14] und [Bac15].

5.2. Implementierung des Clover Terms

Im Folgenden wird beschrieben, wie und an welchen Stellen der Clover Term in CL^{2}QCD implementiert wurde. Die Änderungen sind in der *branch clover*[3] auf Github zu finden.
Zunächst werden die relevanten Gleichungen vorgestellt. Die folgenden Ausführungen orientieren sich an [JS98]. Der Ausgangspunkt für die Implementierung des *Clover* Terms ist die effektive Wirkung. Das bedeutet, die fermionsichen Pfadintegrale sind ausintegriert und man drückt die Fermiondeterminante durch Pseudofermionen aus (vgl. 4.1)

$$Z = \int \mathcal{D}[U]\mathcal{D}[\phi^\dagger]\mathcal{D}[\phi] e^{-S_{\text{eff}}}, \; S_{\text{eff}} = S_G[U] + \phi^\dagger Q^{-2}\phi. \tag{5.1}$$

Die modifizierte Fermionmatrix Q ist gegeben durch

$$Q(n|m) = c_0\gamma_5 D(n|m) = c_0\gamma_5(\delta_{n,m} - \kappa \sum_{\mu=\pm1}^{\pm4} (1-\gamma_\mu)U_\mu(n)\delta_{n+\hat{\mu},m} + \frac{i}{2}c_{sw}\kappa\sigma_{\mu\nu}\hat{F}_{\mu\nu}(n)\delta_{n,m}) \tag{5.2}$$

mit $\kappa = (2m+8)^{-1}$, wie in Abschnitt 3.3, und $c_0 = (1+8\kappa)^{-1}$.
Zur effizienteren Invertierung der Matrix benutzt man die in Abschnitt 4.2 beschriebene even-odd Präkonditionierung

$$Q \equiv c_0\gamma_5 \begin{pmatrix} 1+T_{ee} & M_{eo} \\ M_{oe} & 1+T_{oo} \end{pmatrix}. \tag{5.3}$$

Die nicht-Diagonalblöcke M_{eo} und M_{oe} verbinden jeweils gerade mit ungeraden bzw. ungerade mit geraden Gitterplätzen und entsprechen den Wilson *hopping* Matrizen aus Abschnitt 3.3. Die Matrizen T_{ee}, T_{oo} entsprechen dem *Clover* Term und verbinden jeweils gerade bzw. ungerade Gitterplätze. Sie sind gegeben durch

$$T(n|m) = \frac{i}{2}c_{sw}\kappa\sigma_{\mu\nu}\hat{F}_{\mu\nu}(n)\delta_{nm}. \tag{5.4}$$

Die Präkonditionierung erhält man, indem man die Determinante als

$$\det[Q] \propto \det[1+T_{ee}]\det\big[\hat{Q}\big] \tag{5.5}$$

$$\hat{Q} = \hat{c}_0\gamma_5(1 + T_{oo} - M_{oe}(1+T_{ee})^{-1}M_{eo}) \tag{5.6}$$

[3] `https://github.com/CL2QCD/cl2qcd/tree/clover`

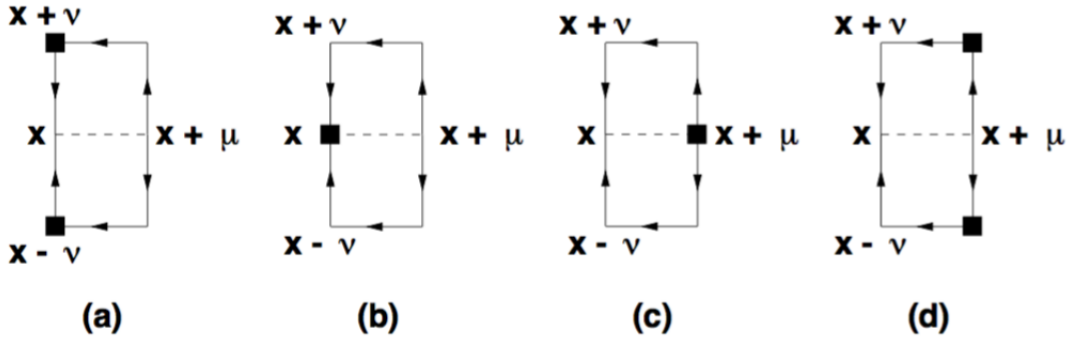

Abbildung 5.2.: Diagramme, die Beiträge zu $F_\mu^{(\mathrm{sw})}$ liefern.

mit $\hat{c}_0 = (1+64\kappa^2)^{-1}$ schreibt. Damit lässt sich die effektive Wirkung als Summe von drei Wirkungen schreiben

$$S_{\mathrm{eff}}^{(\mathrm{eo})} = S_G[U] + S_{\mathrm{det}}[U] + S_b[U, \phi^\dagger, \phi] \tag{5.7}$$

$$S_G[U] = \frac{2}{g^2} \sum_{n\in\Lambda} \sum_{\mu<\nu} \mathrm{Re}\{\mathrm{Tr}[\mathbb{1} - U_{\mu\nu}(n)]\} \tag{5.8}$$

$$S_{\mathrm{det}}[U] = -2\,\mathrm{Tr}[\log[1 + T_{ee}]] \tag{5.9}$$

$$S_{\mathrm{b}}[U, \phi^\dagger, \phi] = \phi^\dagger \hat{Q}^{-2} \phi, \tag{5.10}$$

wobei für $S_{\mathrm{det}}[U]$ die Relation $\det[\mathbb{1} - M] = \exp(\mathrm{Tr}[\log[\mathbb{1} - M]])$ benutzt wurde. Für einen Beweis siehe zum Beispiel [GL09, S.334f.].

Kraft des HMC

Die für den *molecular dynamics* Algorithmus benötigte Kraft lässt sich als Summe von vier Beiträgen schreiben. Im folgenden Abschnitt werden die Gittervektoren mit x bezeichnet:

$$F_\mu(x) = V_\mu(x) + F_\mu^{(\mathrm{w})}(x) + F_\mu^{(\mathrm{sw})}(x) + F_\mu^{(\mathrm{det})}(x). \tag{5.11}$$

Hierbei bezeichnet $V_\mu(x)$ die zur Eichwirkung gehörende Kraft und $F_\mu^{(\mathrm{w})}(x)$ die Kraft der Wilson Fermionwirkung. Die zusätzlichen Kräfte des *Clover* Terms lauten:

$$F_\mu^{(\mathrm{sw})}(x) = -\frac{\hat{c}_0 c_{sw} \kappa}{8} (\text{Diagramme in Abb. 5.2}), \tag{5.12}$$

$$F_\mu^{(\mathrm{det})}(x) = -\frac{c_{sw}\kappa}{4} (\text{Diagramme in Abb. 5.3}). \tag{5.13}$$

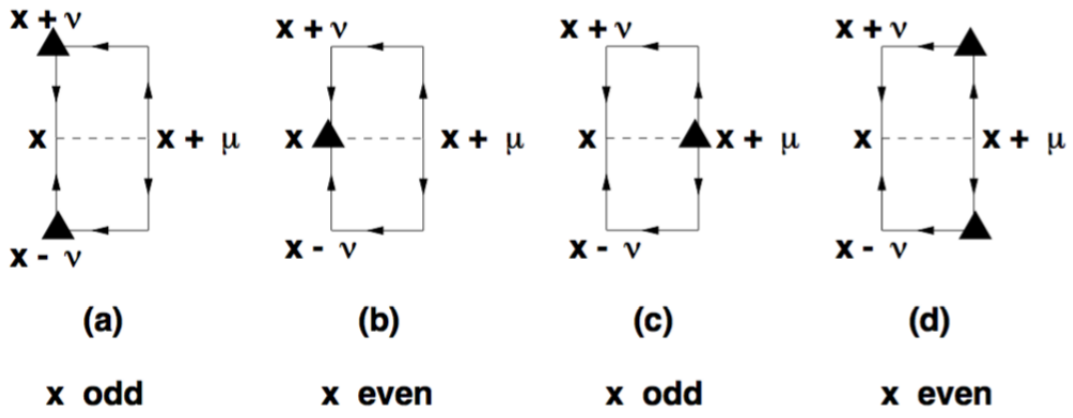

Abbildung 5.3.: Diagramme, die Beiträge zu $F_\mu^{(\mathrm{det})}$ liefern.

Wie in [JS98] beschrieben, haben die Diagramme eine einfache Interpretation. Man startet am Punkt $x+\mu$ und geht entlang der Richtung der Pfeile bis zum Punkt x. Entlang des Weges multipliziert man die den Pfeilen entsprechenden Link Variablen und fügt an den Boxen bzw. Dreiecken man folgende Ausdrücke ein:

$$(\blacksquare)_x = \mathrm{Tr}_{\mathrm{Dirac}}[i\gamma_5\sigma_{\mu\nu}Y(x)\otimes X^\dagger(x) + i\gamma_5\sigma_{\mu\nu}X(x)\otimes Y^\dagger(x)] \tag{5.14}$$

$$(\blacktriangle)_x = \mathrm{Tr}_{\mathrm{Dirac}}[i\sigma_{\mu\nu}(1+T_{ee})^{-1}]. \tag{5.15}$$

Die even-odd präkonditionierten Spinorfelder X und Y sind gegeben durch

$$X = \begin{pmatrix} -(1+T_{ee})^{-1}M_{eo}X_o \\ X_o \end{pmatrix}, \qquad Y = \begin{pmatrix} -(1+T_{ee})^{-1}M_{eo}Y_o \\ Y_o \end{pmatrix}, \tag{5.16}$$

wobei $X_o = \hat{Q}^{-2}\phi_0$ und $Y_o = \hat{Q}^{-1}\phi_0$ die auf den ungeraden Gitterplätzen definierten Felder sind.

Die Beiträge der vier Pfade, die nach unten beginnen, d.h. durch den Punkt $x-\nu$ gehen, erhalten ein relatives Minuszeichen. Schließlich addiert man alle möglichen Beiträge für $\nu \neq \mu$ auf. Damit lautet zum Beispiel der Beitrag von Diagramm a) für $F_\mu^{(\mathrm{sw})}(x)$:

$$\sum_{\nu\neq\mu}\Big\{U_\nu(x+\mu)U_\mu^\dagger(x+\nu)(\blacksquare)_{x+\nu}U_\nu^\dagger(x) - U_\nu^\dagger(x+\mu-\nu)U_\mu^\dagger(x-\nu)(\blacksquare)_{x-\nu}U_\nu(x)\Big\}. \tag{5.17}$$

In CL^{2}QCD sind die Kernels zur Berechnung der beiden neuen Kräfte $F_\mu^{(\mathrm{sw})}(x)$ und $F_\mu^{(\mathrm{det})}(x)$ in den Dateien *force_fermion_clover1_eo.cl* bzw. *force_fermion_clover2_eo.cl* zu finden. In diesen Dateien werden $(\blacksquare)_x$ und

$(\blacktriangle)_x$ sowie die Diagramme durch Funktionen berechnet, die als Eingabeparameter zwei Richtungen haben. In den eigentlichen Kernels werden die Diagramme und die verschiedenen Richtungen explizit aufsummiert. Bei der Implementierung von $F_\mu^{(\mathrm{sw})}(x)$ ist zu beachten, dass die Unterscheidung zwischen geraden und ungeraden Feldern im host Teil gemacht wird. Das für $F_\mu^{(\mathrm{det})}(x)$ benötigte Feld von 12×12 Matrizen $(1+T_{ee})^{-1}$ wird als zwei 6×6 Matrixfelder übergeben, da die Matrix blockdiagonal ist (siehe nächster Abschnitt). Für $(\blacksquare)_x$ benötigt man die Dirac-Spur des Kroneckerproduktes von zwei Spinoren. Diese Funktion befindet sich in *operations_su3vec.cl*.

In der Funktion *calc_fermion_force* in *physics/algorithms/fermion_force.cpp* werden die Kernels, die die konjugierten Impulse updaten, aufgerufen und ihre Eingabeparameter berechnet. Da man für Clover auch Wilson benötigt (vgl. (5.11)), ist Clover hierbei eine Art Erweiterung. Mit mehreren Abfragen ist jedoch gewährleistet, dass die Operation für Clover nur dann ausgeführt werden, wenn man explizit Clover als Wirkung eingestellt hat. In der Funktion werden zunächst Spinorfelder $X_o = \hat{Q}^{-2}\phi_0$ und $Y_o = \hat{Q}^{-1}\phi_0$ durch einen Krylov Solver berechnet. Danach werden aus diesen Spinorfeldern die eigentlichen Eingabeparameter X und Y für die Kernels berechnet. Die beiden 6×6 Matrixfelder für $F_\mu^{(\mathrm{det})}(x)$ werden hier ausgehend von der jeweiligen Eichfeldkonfiguration initialisiert. Dies ist realisiert durch die Methode *setField* der Klasse *Matrix6x6Field*. Mit diesem Input wird nun die Funktion *fermion_force* zweimal (even und odd) aufgerufen.

Man beachte, dass in CL^{2}QCD die Faktoren c_0 bzw. $\hat{c}_0$ nicht vorkommen, da die Implementierung des Clover Terms auf der Wilson Diskretisierung aufbaut und dort diese Faktoren nicht benutzt werden. Wie in Abschnitt 3.3, beschrieben kann man Faktoren in der Wirkung durch eine Neudefinierung der Felder absorbieren. Das bedeutet, dass derartige Faktoren die Physik nicht verändern. Man muss jedoch darauf achten, dass die Implementierung einheitlich ist.

Fermionmatrix

Sowohl für $\hat{Q}$ als auch für die $F_\mu^{(\mathrm{det})}(x)$ benötigt man die Fermionmatrix $1+T$ sowie ihre Inverse. Hierbei wird, wenn möglich, die Fermionmatrix aufgrund ihrer Größe nicht explizit berechnet und gespeichert, sondern die Kernels berechnen die Multiplikation mit einem gegeben Spinorfeld.
In *fermionmatrix_eo_clover.cl* ist dies für $1+T$ realisiert. Hierbei ist die Funktion, die die Multiplikation des Feldstärketensors mit einem Spinor berechnet, in einer eigenen Funktion ausgelagert. Die Multiplikation mit den Matrizen $\sigma_{\mu\nu}$ findet man in *operations_spinor.cl*.

Für die Multiplikation der Inversen mit einem Spinor berechnet man zu-

nächst explizit den Pauli Term in *fermionmatrix_eo_clover_explicit.cl*. Mit den im Appendix A.1 definierten Gamma Matrizen hat er eine blockdiagonale Form [Lü]:

$$\sum_{\mu,\nu=0}^{3} \frac{i}{4}\sigma_{\mu\nu}\hat{F}_{\mu\nu} = \sum_{k=1}^{3} \frac{i}{16}\begin{pmatrix} \sigma_k(E_k - B_k) & 0 \\ 0 & -\sigma_k(E_k + B_k) \end{pmatrix}. \tag{5.18}$$

Hierbei bezeichnen σ_k die Pauli Matrizen und die elektrischen sowie magnetischen Komponenten des Feldstärketensors sind gegebend durch

$$E_k = 8\hat{F}_{0k}, \qquad B_k = \sum_{i,j=1}^{3} 4\,\varepsilon_{\mathrm{klj}}\hat{F}_{lj}. \tag{5.19}$$

Da auch die inverse Fermionmatrix blockdiagonal ist, reicht es aus, hierbei mit 6×6 Matrizen zu arbeiten. In *operations_matrix6x6.cl* befinden sich die Funktionen, die man hierfür benötigt. Die Invertierung ist in *fermionmatrix_eo_clover_inverse.cl* durch die in Abschnitt 4.3 beschriebene Householder Triangularisierung realisiert. In dieser Datei befindet sich auch der Kernel, der die Multiplikation der inversen Fermionmatrix mit einem Spinorfeld berechnet.
In *fermionmatrix_eo_clover _explicit_inverse.cl* berechnen die Kernels die Inverse explizit und speichern sie als zwei 6×6 Matrixfelder für alle Gitterplätze.

In *physics/fermionmatrix/fermionmatrix.cpp* wird die für die Kraft benötigte Matrix $\hat{Q}$ aufgebaut. Benutzt man die in Abschnitt 4.2 beschriebene Aufteilung der Fermionmatrix gilt für Clover

$$\hat{Q} = 1 + T_{ee} - M_{eo}(1 + T_{oo})^{-1}M_{oe}. \tag{5.20}$$

Hierbei ist zu beachten, dass es eine andere mögliche Aufteilung der Matrix gibt und die Multiplikation mit γ_5 im Programm an verschiedenen Stellen realisiert sein kann. In den jeweiligen Funktionen wird abhängig von der Diskretisierung die jeweilige Matrix aufgebaut.

Wirkung

Im Metropolis Schritt (4.9) muss die Wirkung explizit berechnet werden. Für S_{b} berechnet man sie mit Hilfe eines *Krylov Solvers* $\hat{Q}^{-1}\phi$ und bildet von diesem Spinorfeld die quadrierte 2-Norm.
Für $S_{\mathrm{det}} = -2\,\mathrm{Tr}[\log(1 + T_{ee})] = -\log(\det[(1 + T_{ee})^2])$ wurde in *clover_action _S_det.cl* ein neuer Kernel implementiert, der für jeden geraden Gitterplatz die Wirkung berechnet und dann die Beiträge aufsummiert. Die Determinante wurde mithilfe der Householder Triangularisierung berechnet (siehe 4.3). Hierbei wurde verwendet, dass die Determinante von $(1 + T_{ee})^2$ reell und positiv

definit ist.
Dieser Kernel wird in *metropolis.cpp* in der Funktion *calc_s _fermion* aufgerufen, die den fermionischen Teil der Wirkung nach einer vollen Trajektorie des *molecular dynamics* Algorithmus berechnet. Der Kernel für S_{det} wird durch die Funktion *S_det* angesteuert. Diese Funktion befindet sich in *physics/lattices/matrix6x6Field.cpp* und ist ein Freund der dort definierten Klasse.

6. Test der Implementierung

6.1. Testgetriebene Entwicklung

Die Arbeit an CL^{2}QCD folgt dem Prinzip der testgetriebenen Entwicklung. Das bedeutet, dass man parallel zum eigentlichen Programm Tests der Implementierungen schreibt. Dazu werden alle Kernels auch mit **Mathematica** geschrieben, um so Vergleichswerte zu berechnen. Die Mathematica Notebooks befinden sich im Ordner *tools/mathematica*. Mit **Boost**[1] sind diese Tests in das Programm implementiert.
In den Notebooks werden die Operationen für nur einen Gitterplatz ausgeführt und danach mit der Anzahl der Gitterplätze multipliziert, um das Ergebnis mit dem Kernel vergleichen zu können. Damit dieses Verfahren einen Sinn ergibt, müssen die Felder, die man den Kernels als Eingabeparameter übergibt, für alle Gitterplätze gleich sein. In CL^{2}QCD sind für die verschiedenen Gitterobjekte meist mehrere *fillTypes* implementiert, um die Kernels mit verschiedenen Eingabewerten zu testen.
Im Folgenden werden nun die Besonderheiten der Tests für die neu geschriebenen Kernels erklärt. Das Problem der Tests ist der Gitter-Feldstärketensor. Er ist immer 0, wenn man alle Eichfelder unabhängig von der Richtung gleich wählt. Damit ist kein vernünftiger Test möglich, da der Pauli Term so immer 0 ist. Deshalb wurde in *hardware/code/GaugefieldTester.cpp* ein neuer *fillType* mit dem Namen *ascendingInTDirNonTrivialInSpatial* geschrieben. Hierbei sind die Eichfelder in Zeitrichtung Matrizen mit aufsteigenden Einträgen und in den räumlichen Richtungen nicht-triviale Matrizen. Damit ist es möglich die neuen Fermionmatrizen zu testen. Die Tests befinden sich in *hardware/code/fermions_test.cpp*. Desweiteren wurden Tests für die beiden Kräfte des Clover Terms in *hardware/code/molecular_dynamics_test.cpp* geschrieben. $F_\mu^{(\mathrm{det})}$ benötigt zwei 6×6 Matrixfelder als Input. Die hierfür benötigten Funktionen befinden sich in *hardware/code/Matrix6x6FieldTester.cpp*. In *hardware/code/matrix6x6Field_test.cpp* werden die Kernels für die Wirkung S_{det} und die Invertierung von $1 + T$ getestet.

[1] `http://www.boost.org`

6.2. Physikalischer Test

In diesem Abschnitt wird ein physikalischer Test der Implementierung beschrieben. Das Verfahren wurde aus [JL97] entnommen. Die Grundidee ist, Simulationen mit kleinen Parameter durchzuführen und so Erwartungswerte zu erhalten, die man auch analytisch durch Entwicklung nach diesen Parametern berechnen kann.
Man will den Clover Term alleine testen. Dazu definiert man den Parameter $\bar{c}_{\mathrm{sw}} = \kappa c_{\mathrm{sw}}$, der nur im Clover Term vorkommt. Nimmt man nun den Limes $\kappa \to 0$ aber hält $\bar{c}_{\mathrm{sw}}$ bei einem bestimmten Wert fest, trägt der Wilson *hopping* Term nicht mehr bei und der fermionische Anteil der Wirkung besteht nur aus dem Clover Term. Dafür wurde in CL^{2}QCD eine *command line option* implementiert. Bei der Ausführung der Datei *hmc* muss man die Option *use_only_clover* auf 1 bzw. true setzen.
Der folgende Test wird bei $\beta = 0$ durchgeführt. Damit ergibt sich für die effektive Wirkung (vgl. (5.7)):

$$S_{\mathrm{eff}} = -2 \sum_{n \in \Lambda} \mathrm{Tr}[\log(1 + T(n))]. \tag{6.1}$$

Der Vorteil ist nun, dass man den Erwartungswert eines Plaquettes $P \equiv \frac{1}{6} \mathrm{Tr}\left[U_P + U_P^\dagger\right]$ in $\bar{c}_{\mathrm{sw}}$ entwickeln kann

$$\langle P \rangle \simeq \bar{c}_{\mathrm{sw}}^2 \langle P \,\mathrm{Tr}\left[\sigma \hat{F}\right]^2 \rangle_0 - \frac{\bar{c}_{\mathrm{sw}}^4}{2} \langle P \,\mathrm{Tr}\left[\sigma \hat{F}\right]^4 \rangle_0 \tag{6.2}$$

$$+ \frac{\bar{c}_{\mathrm{sw}}^4}{2} \langle P \,\mathrm{Tr}\left[\sigma \hat{F}\right]^2 \mathrm{Tr}\left[\sigma \hat{F}\right]^2 \rangle_0 - \bar{c}_{\mathrm{sw}}^4 \langle P \,\mathrm{Tr}\left[\sigma \hat{F}\right]^2 \rangle_0 \langle \mathrm{Tr}\left[\sigma \hat{F}\right]^2 \rangle_0 \tag{6.3}$$

mit $\sigma \hat{F} = \frac{1}{2} \sigma_{\mu\nu} \hat{F}_{\mu\nu}$ und $\langle O \rangle_0 = \int \mathcal{D}U O[U]$.
Die analytische Berechnung führt zu:

$$\langle P \rangle \simeq -\frac{1}{12} \bar{c}_{\mathrm{sw}}^2 (1 + \frac{475}{384} \bar{c}_{\mathrm{sw}}^2). \tag{6.4}$$

Nun berechnet man den Erwartungswert mit einer Hybrid Monte Carlo Simulation für verschiedene Werte von $\bar{c}_{\mathrm{sw}}^2 \in (0, 0.1]$ und vergleicht die Ergebnisse mit (6.4). In Abbildung 6.1 sind die Ergebnisse der Autoren von [JL97] dargestellt.

Hierbei stellt die gestrichelte Linie nur die führende Ordnung der Entwicklung und die durchgezogende Linie die nächst höhere Ordnung dar. Man erkennt, dass die Messwerte sehr gut mit den analytischen Berechnung übereinstimmen.

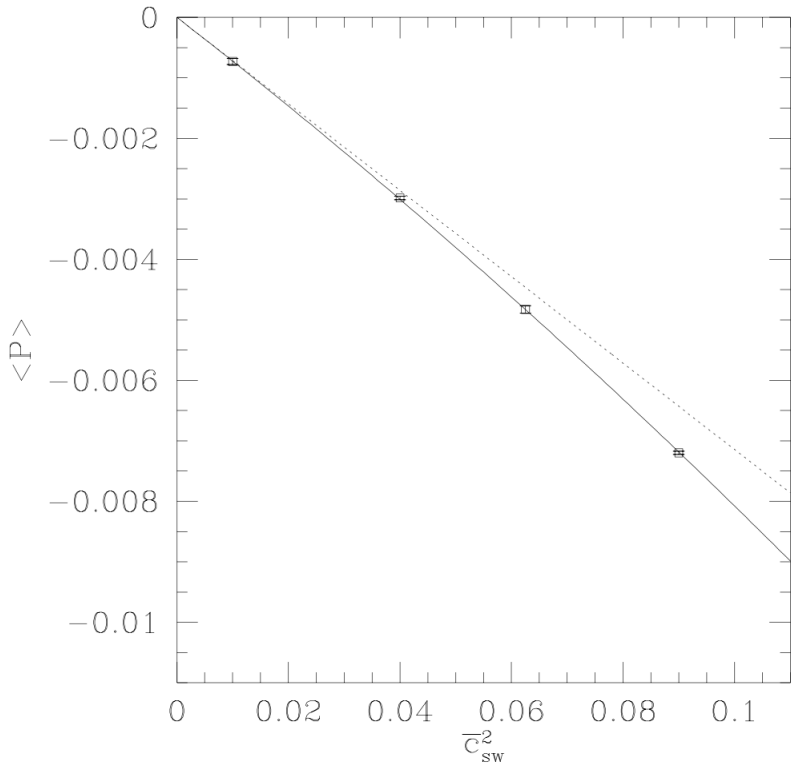

Abbildung 6.1.: Analytisch vs. HMC, entnommen aus [JL97].

7. Zusammenfassung und Ausblick

Im Rahmen dieser Bachelorarbeit wurde der Clover Term in CL^{2}QCD implementiert und alle neuen Funktionen mit Mathematica getestet.

Im Folgenden werden die möglichen, nächsten Schritte erläutert.
Zunächst sollte man die im vorherigen Abschnitt erläuterten physikalischen Test durchführen, um zu gewährleisten, dass nicht nur die einzelnen Kernels, sondern auch der gesamte Hybrid Monte Carlo Algorithmus mit eingebautem Clover Term richtig funktioniert.

Um alle $\mathcal{O}(a)$-Fehler der Wirkung zu eliminieren, muss nun der Koeffizient c_{sw} richtig bestimmt werden. Dazu kann man *Gitter Störungstheorie* [Woh87] benutzen. Da die Störungstheorie jedoch nur bis zu einer bestimmten Ordnung in der Kopplungskonstanten gilt, führt dieses Verfahren nur zu einer unvollständigen Eliminierung der Fehler. Eine andere, nicht-pertubative Möglichkeit ist, den Koeffizienten durch Simulationen zu bestimmen. Dieses Verfahren ist dann exakt bis zu einer bestimmten Ordnung im Gitterabstand. Auf der anderen Seite sind diese nicht-pertubativen Methoden entweder sehr kompliziert oder mit hohem numerischen Aufwand verbunden. Ein Beispiel hierfür findet man in [JS98].

Für eine vollständige Eliminierung aller linearen Fehler in Gitter-QCD Simulation muss man auch die Felder optimieren. Hierfür definiert man analog zur effektiven Wirkung (3.35) effektive Felder als Potenzreihe $\phi_{\text{eff}} = \phi_0 + a\phi_1 + a^2\phi_2 + \ldots$ und implementiert den Korrekturterm ϕ_1.

Nachdem man alle notwendigen Korrekturterme implementiert und die jeweiligen Koeffizienten bestimmt hat, wäre es interessant, die Fehler mit denen der Wilson Formulierung zu vergleichen. Weiterhin sollte man den numerischen Mehraufwand bestimmen und so den praktischen Nutzen für Simulationen abschätzen.

A. Appendix

A.1. Gamma Matrizen

Die euklidischen Gamma Matrizen γ_μ, $\mu = 0, 1, 2, 3$ sind durch die Anti-Kommutationsrelation

$$\{\gamma_\mu, \gamma_\nu\} = 2\delta_{\mu\nu}\mathbb{1} \tag{A.1}$$

definiert. Hierbei beichnet $\delta_{\mu\nu}$ das Kronecker Delta und $\mathbb{1}$ die 4×4 Einheitsmatrix.
Eine explizite Darstellung ist gegeben durch:

$$\gamma_0 = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 \end{pmatrix}, \quad \gamma_1 = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & -i & 0 \\ 0 & +i & 0 & 0 \\ +i & 0 & 0 & 0 \end{pmatrix}, \tag{A.2}$$

$$\gamma_2 = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & +1 & 0 \\ 0 & +1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}, \quad \gamma_3 = \begin{pmatrix} 0 & 0 & -i & 0 \\ 0 & 0 & 0 & +i \\ +i & 0 & 0 & 0 \\ 0 & -i & 0 & 0 \end{pmatrix}. \tag{A.3}$$

Die Darstellung nennt man chiral, weil $\gamma_5 = \gamma_0\gamma_1\gamma_2\gamma_3$, der Chiralitätsoperator, diagonal ist:

$$\gamma_5 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{A.4}$$

A.2. Diskretisierungsfehler der Wilson Fermionwirkung

Die naive Fermionwirkung (3.10) hat einen Diskretisierungsfehler von $\mathcal{O}(a^2)$. Im Folgenden wird gezeigt, dass der Wilson-Term proportional zur Gitterkonstanten a ist und somit die Wilson Fermionen einen Diskretisierungsfehler von $\mathcal{O}(a)$ haben.

Hierfür entwickeln wir den Wilson Term für kleine a. Der Dirac Operator im Ortsraum ist gegeben durch:

$$D(n|m) = -a \sum_{\mu=1}^{4} \frac{U_\mu(n)\delta_{n+\hat{\mu},m} - 2\delta_{n,m} + U_{-\mu}(n)\delta_{n-\hat{\mu},m}}{2a^2}. \tag{A.5}$$

Die Entwicklung der Link-Variablen lautet:

$$U_\mu(n) = \mathbb{1} + iaA_\mu(n) + \mathcal{O}\big(a^2\big), \quad U_{-\mu}(n) = \mathbb{1} - iaA_\mu(n-\hat{\mu}) + \mathcal{O}\big(a^2\big). \tag{A.6}$$

Damit ergibt sich für diesen Teil der Wirkung:

$$S = a^4 \sum_{n\in\Lambda} \bar{\psi}(n) D(n|m) \psi(n) \tag{A.7}$$

$$= a^4 \sum_{n\in\Lambda} (-) a \sum_{\mu=1}^{4} \bar{\psi}(n) \frac{(\mathbb{1} + iaA_\mu(n))\psi(n+\hat{\mu}) - 2\psi(n) + (\mathbb{1} - iaA_\mu(n-\hat{\mu}))\psi(n-\hat{\mu})}{2a^2} \tag{A.8}$$

$$= a^4 \sum_{n\in\Lambda} (-) \frac{a}{2} \sum_{\mu=1}^{4} \bar{\psi}(n) \frac{\psi(n+\hat{\mu}) - 2\psi(n) + \psi(n-\hat{\mu})}{a^2} + \mathcal{O}(a). \tag{A.9}$$

Daraus folgt, dass der Wilson Term eine Diskretisierung von $-(a/2)\partial_\mu\partial_\mu$ ist. Dies bedeutet, dass der Wilson Term einen Diskretisierungsfehler von $\mathcal{O}(a)$ bewirkt.

A.3. Matrix Determinanten Lemma

In diesem Abschnitt soll das sogenannte Matrix Determinanten Lemma gezeigt werden. Es lautet:

$$\det\big[\mathbb{1} + vv^\dagger\big] = 1 + v^\dagger v \;\; \forall v \in \mathbb{C}^n. \tag{A.10}$$

Für den Beweis der Aussage benutzt man die Identität

$$\begin{pmatrix} \mathbb{1} & 0 \\ v^\dagger & 1 \end{pmatrix} \begin{pmatrix} \mathbb{1} + vv^\dagger & v \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbb{1} & 0 \\ -v^\dagger & 1 \end{pmatrix} = \begin{pmatrix} \mathbb{1} & v \\ 0 & 1 + v^\dagger v \end{pmatrix}, \tag{A.11}$$

wobei hierbei zu beachten ist, dass es sich um Matrizen der Größe $(n+1)\times(n+1)$ handelt.
Berechnet man nun auf beiden Seite die Determinante und benutzt den Determinantenmultiplikationssatz folgt

$$1 \cdot \det\big[\mathbb{1} + vv^\dagger\big] \cdot 1 = 1 + v^\dagger v. \tag{A.12}$$

Damit ist die Aussage gezeigt.

Literaturverzeichnis

[Bac15] Matthias Bach. *Energy-and cost-efficient Lattice-QCD computations using graphics processing unit*. PhD thesis, Goethe-Universität Frankfurt/Main, 2015.

[DR90] Thomas A. DeGrand and Pietro Rossi. Conditioning Techniques for Dynamical Fermions. *Comput. Phys. Commun.*, 60:211–214, 1990.

[GL09] C. Gattringer and C. Lang. *Quantum Chromodynamics on the Lattice: An Introductory Presentation*. Lecture Notes in Physics. Springer Berlin Heidelberg, 2009.

[HB06] Martin Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Mathematische Leitfäden. Teubner, 2006.

[JL97] K. Jansen and C. Liu. Implementation of Symanzik's improvement program for simulations of dynamical Wilson fermions in lattice QCD. *Comput. Phys. Commun.*, 99:221–234, 1997.

[JS98] K. Jansen and R. Sommer. O(α) improvement of lattice QCD with two flavors of Wilson quarks. *Nucl. Phys.*, B530:185–203, 1998. [Erratum: Nucl. Phys.B643,517(2002)].

[JU09] K. Jansen and C. Urbach. tmLQCD: A Program suite to simulate Wilson Twisted mass Lattice QCD. *Comput. Phys. Commun.*, 180:2717–2738, 2009.

[LSSW96] Martin Lüscher, Stefan Sint, Rainer Sommer, and Peter Weisz. Chiral symmetry and o(a) improvement in lattice qcd. *Nuclear Physics B*, 478(1):365 – 397, 1996.

[Lü] M. Lüscher. OpenQCD Dokumentation. `http://luscher.web.cern.ch/luscher/openQCD/`.

[Mat12] H. Matsufuru. Lattice QCD simulations with clover quarks. 2012.

[MM97] I. Montvay and G. Münster. *Quantum Fields on a Lattice*. Cambridge Monographs on Mathematical Physics. Cambridge University Press, 1997.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[Pin14] Christopher Pinke. *Lattice QCD at Finite Temperature with Wilson Fermions*. PhD thesis, Goethe-Universität Frankfurt/Main: Fachbereich Physik, 2014.

[PPSB15] Owe Philipsen, Christopher Pinke, Alessandro Sciarra, and Matthias Bach. CL&2QCD - Lattice QCD based on OpenCL. In *Proceedings, GPU Computing in High-Energy Physics (GPUHEP2014): Pisa, Italy, September 10-12, 2014*, pages 169–174, 2015.

[SW85] B. Sheikholeslami and R. Wohlert. Improved Continuum Limit Lattice Action for QCD with Wilson Fermions. *Nucl. Phys.*, B259:572, 1985.

[Sym83] K. Symanzik. Continuum Limit and Improved Action in Lattice Theories. 1. Principles and phi**4 Theory. *Nucl. Phys.*, B226:187–204, 1983.

[vH16] B. von Harrach. Skript: Einführung in die Numerik. `http://numerical.solutions`, WS 2015/2016.

[Woh87] R. Wohlert. Improved Continuum Limit Lattice Action For Quarks. unpublished, 1987.

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Abschlussarbeit selbstständig und nur unter Verwendung der von mir angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Datum: _____ Unterschrift: _____