Klaus Spanderen

Monte-Carlo-Simulationen einer SU(2) Yang-Mills-Theorie mit dynamischen Gluinos

1998

Theoretische Physik

Monte-Carlo-Simulationen einer SU(2) Yang-Mills-Theorie mit dynamischen Gluinos

Inaugural–Dissertation
zur Erlangung des Doktorgrades
der Naturwissenschaften im Fach Physik
der Mathematisch-Naturwissenschaftlichen Fakultät
der Westfälischen Wilhelms–Universität Münster

vorgelegt von
Klaus Spanderen
aus Velen

- 1998 -

Dekan:	Prof. Dr. N. Schmitz
Erster Gutachter:	Prof. Dr. G. Münster
Zweiter Gutachter:	Prof. Dr. M. Stingl
Tag der mündlichen Prüfungen:	
Tag der Promotion:	

Inhaltsverzeichnis

Einleitung

Die von der Poincaré-Gruppe beschriebenen Symmetrien haben weitreichende Konsequenzen für die grundlegenden Eigenschaften von Elementarteilchen. Da diese nach heute gängiger Vorstellung irreduzible Darstellungen der Poincaré-Gruppe tragen, müssen Elementarteilchen eine definierte Ruhemasse und einen Spin bzw. eine Helizität besitzen. Die Poincaré-Gruppe und ihre zugehörige Lie-Algebra sind physikalisch durch die spezielle Relativitätstheorie motiviert. Versuche, zu neuen Theorien zu gelangen, indem man die Poincaré-Algebra durch eine innere Lie-Algebra erweitert, scheitern am no-go Theorem von S. Coleman und J. Mandula [CM67]. Allerdings sind die Voraussetzungen für dieses no-go Theorem unnötig streng gehalten. Schwächt man sie ab, indem man neben den kommutierenden Generatoren für eine Lie-Algebra auch antikommutierende Generatoren zuläßt, so ist es durchaus möglich, die Symmetrien der Poincaré-Gruppe nichttrivial zu erweitern. Das Resultat sind Lie-Superalgebren, unter denen die Supersymmetriealgebra eine besondere Stellung einnimmt, da sie die allgemeinste Lie-Superalgebra ist, die mit den Forderungen einer relativistischen Quantenfeldtheorie konsistent ist [HLS75].

Die erste Lagrange-Dichte, die Supersymmetrie im Rahmen einer vierdimensionalen, relativistischen Quantenfeldtheorie realisiert, wurde von Wess und Zumino vorgestellt [WZ74]. Dieses Modell zeigt die charakteristischen Merkmale supersymmetrischer Feldtheorien. Die beiden fundamentalen Teilchenklassen Bosonen und Fermionen können durch Supersymmetrietransformationen untereinander mischen. Zu jedem Teilchen existiert ein supersymmetrischer Partner mit gleicher Masse aus der jeweils anderen Teilchenklasse. Dies ist in der Natur offensichtlich nicht der Fall. Es besteht aber die Möglichkeit, daß die Supersymmetrie spontan gebrochen ist und deshalb die Massen der supersymmetrischen Partner zu den bekannten Elementarteilchen jenseits der Nachweisgrenzen heutiger Beschleuniger liegen.

Es gibt natürlich handfeste Gründe, warum man sich trotz des negativen experimentellen Befundes seit fast 25 Jahren mit supersymmetrischen Quantenfeldtheorien beschäftigt. U. a. entschärft Supersymmetrie das Hierarchie-Problem von vereinheitlichten Feldtheorien (GUT) durch das non-renormalization Theorem. Im Gegensatz zu den Voraussagen von gewöhnlichen GUT liegt die mittlere Lebensdauer des Protons in supersymmetrischen GUT im Bereich der experimentellen Schranken. Ebenso werden die drei Kopplungskonstanten eines supersymmetrischen Standardmodells nach heutigen Daten bei einer Skala von $m_G \approx 2 \cdot 10^{16}\,\mathrm{GeV}$ gleich stark, wohingegen sich die drei laufenden Kopplungen des Standardmodells nicht in einem Punkt

treffen.

Zur Zeit erfahren supersymmetrische Quantenfeldtheorien eine Renaissance durch die exakten Lösungen vierdimensionaler Eichtheorien von Seiberg und Witten. Diese Lösungen beinhalten wichtige und interessante nichtperturbative Effekte wie Confinement und chirale Symmetriebrechung. Sie basieren auf einer Dualität zwischen „elektrischen“ und „magnetischen“ Freiheitsgraden [SW94a][SW94b]. Ursprünglich wurden diese Dualitäten in N=4 und N=2 supersymmetrischen Theorien gefunden. Mittlerweile konnten die Ergebnisse auch auf einige N=1 Theorien übertragen werden [Sei95]. Einen Weg, diese nichtperturbativen Ergebnisse durch einen unabhängigen Ansatz zu bestätigen, bietet die numerische Simulation im Rahmen der Gitterregularisierung an. Allerdings bricht das Gitter die Supersymmetrie schon allein durch den Umstand, daß die Lorentz-Invarianz verloren geht. Nutzt man zudem noch Wilson-Fermionen, so bricht der Wilson-Term ebenfalls die Supersymmetrie.

Curci und Veneziano schlagen zur Lösung dieses Problems vor, mit einer einfachen Diskretisierung der im Kontinuum supersymmetrischen Wirkung zu starten und die Supersymmetrie im Kontinuumslimes durch ein Fein-„Tuning“ der Parameter wieder zu restaurieren [CV87]. Als einfachsten Testfall für eine nichtabelsche Eichtheorie wählen sie die N=1 SU(2) Super-Yang-Mills-Theorie. Vorteil dieser Theorie ist, daß man neben der Eichkopplung mit der Gluino-Masse lediglich einen Parameter richtig einstellen muß. Allerdings ist die Simulation einer einfachen Version der N=1 SU(2) Super-Yang-Mills-Theorie auf dem Gitter mit einem nicht zu unterschätzenden Aufwand verbunden. Zum einen ist die quenched Approximation nicht geeignet, da sie die Supersymmetrie „hart“ bricht, zum anderen muß man einen einzigen Majorana-Freiheitsgrad simulieren. Ein effizienter Simulationsalgorithmus eröffnete sich erst durch die von Lüscher vorgeschlagenen Multi-Bosonischen Algorithmen [Lüs94]. Ursprünglich war diese Art von Algorithmus nur für eine gerade Anzahl von entarteten Quark-Flavours gedacht. Dieses Konzept wurde von I. Montvay für eine beliebige Anzahl von Flavour-Freiheitsgraden, insbesondere auch für halbzahlige Flavour, erweitert [Mon96]. Damit stand acht Jahre nach dem Vorschlag von Curci und Veneziano ein effizienter Algorithmus für die Simulation einer SU(2) Eichtheorie mit dynamischen Gluinos zur Verfügung.

Im Anschluß rief I. Montvay die DESY-Münster Kollaboration ins Leben, in deren Rahmen auch die vorliegende Arbeit entstand. Die Ziele der Kollaboration sind sowohl physikalischer als auch algorithmischer Natur. Auf der physikalischen Seite steht die Frage im Vordergrund, ob man mit dem Curci-Veneziano-Ansatz die Supersymmetrie auf dem Gitter restaurieren kann. In diesem Fall sollten sich die leichtesten, farblosen Bindungszustände in Supermultipletts anordnen. Insbesondere sollten alle Konstituenten eines Multipletts dieselbe Masse besitzen. Durch eine nichtverschwindende Gluino-Masse wird die Supersymmetrie „weich“ gebrochen. Die dadurch bedingte Massenaufspaltung im leichtesten Supermultiplett kann direkt untersucht werden. Die Gittersimulation ist somit eine geeignete Methode für die Untersuchung der physikalischen Effekte der Supersymmetriebrechung. Hierbei

möchte man den theoretisch vorausgesagten Phasenübergang erster Ordnung bei verschwindender Gluino-Masse mit Hilfe des Ordnungsparameters, in diesem Fall des Fermion-Kondensats, beobachten. Da die Kontinuumstheorie nur im Falle masseloser Gluinos supersymmetrisch ist, liegt für die Weiterentwicklung des Algorithmus der Schwerpunkt auf der Simulation mit sehr kleinen Gluino-Massen. Lösungsansätze hierfür würden sich auch bei QCD-Simulationen nahe des chiralen Limes verwenden lassen. Es zeichnete sich früh ab, daß ein umfangreiches Software-Paket zur Durchführung und Analyse der Monte-Carlo-Simulationen nötig sein würde. Erschwert wurde die Software-Entwicklung durch den Umstand, daß nur ein Parallelrechner die nötige Rechenleistung für dieses Projekt zur Verfügung stellen kann. Um die Flexibilität der Software zu erhöhen, wurde von Anfang an auf ein objektorientiertes Design Wert gelegt. Dabei müssen besondere Techniken entwickelt werden, damit geschwindigkeitskritische Stellen nicht durch die objektorientierte Implementierung in Mitleidenschaft gezogen werden.

Die Arbeit gliedert sich in folgende Abschnitte. Das erste Kapitel gibt einen Überblick über die Poincaré-Superalgebra und über den experimentellen Befund für Supersymmetrie. Anschließend wird der Superraumformalismus eingeführt und es werden aktuelle Vorschläge für effektive N=1 SU(N_c) Wirkungen vorgestellt. Das zweite Kapitel widmet sich der Konstruktion einer einfachen Gitterwirkung für das N=1 SU(N_c) Super-Yang-Mills-Modell. Dessen Simulation durch einen Multi-Bosonischen Algorithmus ist Gegenstand des dritten Kapitels. Im vierten Kapitel wird die Optimierung des Algorithmus durch Präkonditionierung vorgestellt. Für die Simulationsparameter des Multi-Bosonischen Algorithmus benötigt man die genaue Kenntnis der Verteilung des kleinsten Eigenwertes. Diese erlaubt es im Anschluß einen Bogen zu den Zufalls-Matrix-Modellen zu spannen. Die Korrekturverfahren für den Multi-Bosonischen Algorithmus werden im fünften Kapitel besprochen. Der erste Teil befaßt sich dabei mit der Zwei-Schritt-Approximation und den Modifikationen. Wie sich herausstellen wird, kann das ursprüngliche Zwei-Schritt-Verfahren für sehr kleine Gluino-Massen die richtige Verteilung der Konfigurationen nicht mehr mit ausreichender Genauigkeit gewährleisten, so daß eine Meßkorrektur eingeführt werden muß. Zu deren Berechnung wird ein geeignetes hybrides Verfahren, bestehend aus Noisy Correction und exakter Korrektur für die kleinsten Eigenwerte, vorgestellt. Der Inhalt des sechsten Kapitels beschreibt die Massenmessung der Teilchen des leichtesten Supermultipletts auf dem Gitter. Besonderen Umfang wird dabei den verwendeten Smearing-Algorithmen eingeräumt. Das letzte Kapitel beschäftigt sich mit der Bestimmung des kritischen Hopping-Parameters. Im weiteren Verlauf wird das Verhalten der String-Tension und der Fluß der kleinsten Eigenwerte in der Umgebung des kritischen Hopping-Parameters diskutiert. Die Massen der Teilchen aus den leichtesten Supermultipletts und deren evtl. Mischung bilden den Abschluß dieses Kapitels. In den Anhängen C. bis F. wird ein Überblick zum Design und zur Implementation des Software-Paketes gegeben.

Kapitel 1

SU(N_C) Super-Yang-Mills-Theorie

1.1 Poincaré-Superalgebra

1.1.1 Symmetrien des Standardmodells

In der Vergangenheit gelangen wesentliche Fortschritte in der Elementarteilchenphysik durch das Studium von Symmetrien der Natur und den damit verbundenen Gruppen bzw. Lie-Algebren. So lassen sich die elektroschwache und die starke Wechselwirkung durch das direkte Produkt $\mathrm{SU}(3)_C \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$ der Farbgruppe $\mathrm{SU}(3)_C$, der Gruppe $\mathrm{SU}(2)_L$ des schwachen Isospins und der $\mathrm{U}(1)_Y$ der schwachen Hyperladung im Rahmen des Standardmodells beschreiben. Dabei ist die $\mathrm{SU}(2)_C \times \mathrm{U}(1)_Y$-Symmetrie spontan durch die Einführung des Higgs-Feldes gebrochen, so daß sich für Energien deutlich unterhalb der Massen der W- und Z-Bosonen ($m_w = 80.43 \pm 0.08 \frac{GeV}{c^2}$ bzw. $m_z = 91.1863 \pm 0.0019 \frac{GeV}{c^2}$) der ungebrochene Anteil $\mathrm{SU}(3)_C \times \mathrm{U}(1)_{em}$ der vollen Eichsymmetrie manifestiert.

Die Forderung nach einer Charakterisierung von Hadronen bezüglich der Darstellungen einer Symmetriegruppe führte 1961 auf die Flavour-SU(3)-Gruppe. Teilchen, die sich nach der Fundamental-Darstellung der Flavour-SU(3) transformieren, wurden 1964 unabhängig von Gell-Mann und Zweig eingeführt. Der Name *Quarks*, für die zu der Zeit hypothetischen Teilchen, geht auf Gell-Mann zurück. Im Gegensatz zu Raum-Zeit- oder zu Eich-transformationen des Standardmodells ist die Flavour-Symmetrie nur eine Approximation. Zum Beispiel beträgt die Massenaufspaltung im wichtigsten Baryonen-Multiplett, dem Baryonen-Oktett mit $J^P = \frac{1}{2}^+$, zwischen dem Λ-Hyperon und dem Proton ca. 20%.

Wie jede physikalisch sinnvolle Quantenfeldtheorie muß die Lagrange-Dichte des Standardmodells relativistisch kovariant formuliert sein, d. h. die Felder des Standardmodells müssen sich gemäß den Tensordarstellungen der Poincaré-Gruppe transformieren. Die Poincaré-Gruppe kann dabei am einfachsten über ihre Lie-Algebra beschrieben werden. Hierzu werden die vier Generatoren der Translationsgruppe mit P^μ bezeichnet, und die $M^{\rho\sigma} = -M^{\sigma\rho}$ geben die sechs Generatoren der Lorentz-

Gruppe an

$$\begin{aligned} [P^\mu, P^\nu] &= 0 \\ [P^\mu, M^{\rho,\sigma}] &= i(g^{\mu\rho}P^\sigma - g^{\mu\sigma}P^\rho) \\ [M^{\mu\nu}, M^{\rho\sigma}] &= -i(g^{\mu\rho}M^{\nu\sigma} - g^{\mu\sigma}M^{\nu\rho} - g^{\nu\rho}M^{\mu\sigma} + g^{\nu\sigma}M^{\mu\rho}). \end{aligned} \tag{1.1}$$

Die Kenntnis der beiden Casimir-Operatoren der Poincaré-Gruppe

$$\begin{aligned} P^2 &= P_\mu P^\mu \\ W^2 &= W_\mu W^\mu \quad \text{mit} \quad W_\mu = \frac{1}{2}\epsilon_{\mu\nu\rho\sigma}P^\nu M^{\rho\sigma} \ : \text{ Pauli-Lubanski-Vektor} \end{aligned} \tag{1.2}$$

führt auf natürlichem Wege zu den irreduziblen Darstellungen der Poincaré-Gruppe. Alle physikalischen Zustände lassen sich nun nach Eigenwerten dieser beiden Casimir-Operatoren klassifizieren:

- Massive Darstellung

$$\begin{aligned} P^2 &= m^2 > 0, \ P_0 > 0 \\ W^2 &= -m^2 s(s+1) \quad \text{mit} \quad m \in \mathbb{R} \ \wedge \ s = 0, \frac{1}{2}, 1, \frac{3}{2}, ... \end{aligned} \tag{1.3}$$

- Masselose Darstellung

$$\begin{aligned} P^2 &= W^2 = 0 \quad \text{und} \quad P_0 = +|\vec{\mathbf{P}}| \\ \Rightarrow W_\mu &= \lambda P_\mu \quad \text{mit} \quad \lambda = \pm s \ \wedge \ s = 0, \frac{1}{2}, 1, \frac{3}{2}, ... \end{aligned} \tag{1.4}$$

  Im masselosen Fall ist somit die *Helizität* $\lambda = W_0/P_0$ eine Poincaré-Invariante und damit ein zusätzlicher Casimir-Operator.

Es existieren noch weitere irreduzible Darstellungen, so z. B. $P^2 = 0 \ \wedge \ s \in \mathbb{R}$ oder $P^2 < 0$. Die Natur realisiert aber keine Elementarteilchen, welche diese Darstellungen tragen.

Trotz einiger Erfolge des Standardmodells, zu denen ohne Zweifel die Voraussage massiver Vektorbosonen und die Existenz schwacher neutraler Ströme gehören, scheint die Zahl der durch Experimente zu bestimmenden Parameter für eine fundamentale Theorie sehr hoch zu sein. Bestätigen sich zudem die jüngsten Ergebnisse des "Super-Kamiokande"-Detektors zur Neutrino-Oszillation [F98], so wird sich die Anzahl der Parameter weiter erhöhen, da man neben der Kobayashi-Maskawa-Matrix eine weitere Massen-Matrix im Leptonen-Sektor zu berücksichtigen hätte.

Für die Suche nach einer umfassenderen Theorie, welche nicht den Schwächen des Standardmodells unterliegt, ist es nur natürlich, sich weiterhin vom Symmetriegedanken leiten zu lassen. Dabei ist es nicht zwingend notwendig, daß die neuen, postulierten Symmetrien mit den bisherigen Methoden beobachtbar sind. Schon die Vereinheitlichung der elektroschwachen Theorie hat gezeigt, daß es fundamentale

Symmetrien gibt, die schwer zu entdecken sind, weil sie z. B. spontan gebrochen werden.

Ein möglicher Ansatz besteht darin, den Eichfeldsektor des Standardmodells "aufzuräumen", indem man versucht, das direkte Produkt $\mathrm{SU}(3)_C \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$ in einer größeren Symmetriegruppe, z. B. in der SU(5)-Gruppe [1], einzubetten. Einen anderen Ansatzpunkt bietet die Poincaré-Gruppe. Jede Modifikation dieser Gruppe hat natürlich weitreichende Konsequenzen, da jedes Elementarteilchen davon betroffen sein wird und zudem die lokale Poincaré-Invarianz den Ausgangspunkt für die allgemeine Relativitätstheorie bildet.

Versuche, die Poincaré-Algebra nichttrivial mit einer inneren Lie-Algebra zu erweitern, wurden Mitte der sechziger Jahre unternommen. S. Coleman und J. Mandula konnten 1967 im Rahmen eines no-go Theorems zeigen [CM67], daß die Poincaré-Algebra schon die allgemeinste einfache Lie-Algebra zur Beschreibung von Symmetrien der S-Matrix unter den folgenden Bedingungen ist:

- Die S-Matrix basiert auf einer lokalen, relativistischen Quantenfeldtheorie in vier Dimensionen.
- Es gibt eine endliche Zahl von unterschiedlichen Teilchen, welche mit Einteilchenzuständen gegebener Masse assoziiert werden.
- Vakuum und Einteilchenzustände werden durch eine Energielücke getrennt.

Der einzige Weg, dieses no-go Theorem unter physikalisch sinnvollen Randbedingungen zu umschiffen, besteht darin, den Begriff der Lie-Algebra zu erweitern.

Deshalb gelangten Y. Golfand und E. Likhtman zu gradierten Algebren, indem sie zu den kommutierenden Generatoren der Poincaré-Algebra, antikommutierende Generatoren hinzunahmen [GL71]. Speziell die einfache Gradierung mit der zweielementigen Gruppe $\mathbb{Z}_2$ kann benutzt werden, um eine *supersymmetrische* Verallgemeinerung von Lie-Algebren zu konstruieren, die sogenannten Lie-Superalgebren. Unter den Lie-Superalgebren nimmt die Supersymmetriealgebra eine besondere Stellung ein, denn die Autoren R. Haag, J.T. Lopuszanski und M. Sohnius zeigen in [HLS75], daß diese Algebra die allgemeinste Lie-Superalgebra von Symmetrien der S-Matrix ist, die konsistent mit relativistischen Quantenfeldtheorien ist.

[1] Eine gute Einleitung für diesen Zugang zu den *Grand Unified Gauge Theories* bietet [Ros84].

1.1.2 $\mathbb{Z}_2$-Gradierung der Poincaré-Algebra

Unter der Gradierung einer Algebra $\mathcal{G}$ mit einer abelschen Halbgruppe G versteht man die Unterteilung

$$\mathcal{G} = \bigoplus_{i \in G} \mathcal{G}_{(i)} \tag{1.5}$$

von $\mathcal{G}$ in eine direkte Summe von Vektorräumen, so daß

$$x_i \circ y_j \in \mathcal{G}_{(i+j)} \quad \text{mit} \quad i,j \in G \ \wedge \ x_i \in \mathcal{G}_{(i)},\ y_j \in \mathcal{G}_{(j)}. \tag{1.6}$$

Im Fall $G = \mathbb{Z}_2$ ist nun $i,j \in \{0,1\} \ \wedge \ (i+j) = i + j \ mod\ 2$, d. h. nur $\mathcal{G}_{(0)}$ ist bezüglich des Produktes $\circ$ abgeschlossen. Aus einer $\mathbb{Z}_2$-gradierten Algebra kann man eine Lie-Superalgebra konstruieren, indem man dem Produkt $\circ$ die folgenden Eigenschaften auferlegt

$$\begin{aligned}
&\text{Gradierung} && : \ x_i \circ y_j \in \mathcal{G}_{i+j \ mod\ 2} \\
&\text{Supersymmetrie} && : \ x_i \circ y_j = -(-1)^{ij} y_j \circ x_i \\
&\text{Jacobi} - \text{Identität} && : \ 0 = x_k \circ (y_l \circ z_m)(-1)^{km} \\
& && \quad + y_l \circ (z_m \circ x_k)(-1)^{kl} + z_m \circ (x_k \circ y_l)(-1)^{lm}.
\end{aligned} \tag{1.7}$$

Nur $\mathcal{G}_0$ bildet hierbei mit dem Produkt $\circ$ eine Lie-Algebra. Die gesamte Lie-Superalgebra $\mathcal{G}$ ist keine Lie-Algebra im eigentlichen Sinne (Die teilweise benutzten Namen *gradierte* oder $\mathbb{Z}_2 - gradierte\ Lie\text{-}Algebra$ anstelle von Lie-Superalgebra sind deshalb mißverständlich.). Verfügt der Vektorraum $\mathcal{G}$ schon über ein unterliegende Algebra mit einem assoziativen Produkt $(\cdot)$, so kann die Lie-Superalgebra durch

$$x_i \circ y_j = x_i \cdot y_j - (-1)^{ij} y_j \cdot x_i = \begin{cases} x_i \cdot y_j + y_j \cdot x_i & \text{für } i = j = 1 \\ x_i \cdot y_j - y_j \cdot x_i & \text{sonst} \end{cases} \tag{1.8}$$

realisiert werden, d. h. für i=j=1 wird das Produkt $\circ$ durch Antikommutatoren $\{\ ,\ \}$ gebildet, während in den anderen Fällen Kommutatoren $[\ ,\ ]$ dazu verwandt werden.

Auf der Suche nach einer physikalisch sinnvollen Lie- Superalgebra ist man bei $\mathcal{G}_0$, nach dem no-go Theorem von Coleman-Mandula, schon auf die zehndimensionale Poincaré-Algebra (1.1) festgelegt. Für $\mathcal{G}_1$ sollte man sich bezüglich der Dimension zuerst auf $n = 4$ Generatoren $Q_1, .., Q_4$ beschränken. Diese Generatoren werden in der Lie-Superalgebra durch Antikommutatoren verknüpft. Es ist also naheliegend, damit fermionische Freiheitsgrade zu assoziieren. Der physikalischen Intention folgend, sollten die Generatoren einen Spinor bilden; die Anzahl der Komponenten $n = 4$ engt die Wahl im wesentlichen auf Majorana- oder Weyl-Spinoren ein. Die weiteren Rechnungen erfolgen für einen Majorana-Spinor Q_a. Da aber jeder Majorana-Spinor Q_a durch einen Weyl-Spinor Q_A ausgedrückt werden kann

$$Q_a = \begin{pmatrix} Q_A \\ \bar{Q}^{\dot{A}} \end{pmatrix}, \tag{1.9}$$

lassen sich die Ergebnisse direkt übertragen. Als spinorartige Größe sollte sich Q_a bei einer Lorentz-Transformation $M_{\mu\nu}$ gemäß der Spinordarstellung transformieren

und invariant gegenüber Translation P_μ sein. Damit ergeben sich die benötigten Kommutatorrelationen zwischen $\mathcal{G}_0$ und $\mathcal{G}_1$

$$\begin{aligned} \mathcal{G}_0 \circ \mathcal{G}_1 \to \mathcal{G}_1 \; : \quad & [P_\mu, Q_a] = 0, \\ & [M_{\mu\nu}, Q_a] = -(\Sigma_{\mu\nu})_{ab} Q_b, \end{aligned} \tag{1.10}$$

wobei die Generatoren $\Sigma_{\mu\nu}$ der vierdimensionalen Spinordarstellung durch

$$\Sigma_{\mu\nu} = \frac{i}{4}[\gamma_\mu, \gamma_\nu] \tag{1.11}$$

gegeben sind. Für das noch fehlende Produkt $\circ$ zwischen $\mathcal{G}_0$ und $\mathcal{G}_1$ geht man von einem allgemeinen Ansatz

$$\mathcal{G}_1 \circ \mathcal{G}_1 \to \mathcal{G}_0 \; : \quad \{Q_a, Q_b\} = (h^\mu)_{ab} P_\mu + (k^{\mu\nu})_{ab} M_{\mu\nu} \tag{1.12}$$

aus, wobei h^μ und $k^{\mu\nu}$ symmetrisch in a und b, bzw. antisymmetrisch in μ und ν sein sollen. Die Jacobi-Identität in Gleichung (1.7) kann allerdings von einem solchen Ansatz nur erfüllt werden, wenn

$$h^\mu = a\gamma^\mu C \;, a \in \mathbb{R} \quad \wedge \quad k^{\mu\nu} = 0 \tag{1.13}$$

ist, wobei $C = i\gamma^2\gamma^0$ der Ladungskonjugationsoperator ist. Es ist üblich, den Vorfaktor $a = -2$ zu setzen. Nutzt man nun noch die Majorana-Bedingung

$$Q_a = Q_a^C \stackrel{\text{def}}{=} (C\bar{Q}^T)_a, \quad \text{mit } \bar{Q} \stackrel{\text{def}}{=} Q^\dagger\gamma^0 \tag{1.14}$$

aus, so kann der fehlende Antikommutator kompakt als

$$\{Q_a, \bar{Q}_b\} = 2\gamma^\mu_{ab} P_\mu \tag{1.15}$$

geschrieben werden. Die Gleichungen (1.1), (1.10) und (1.15) bilden eine Lie-Superalgebra

$$\begin{aligned} [P^\mu, P^\nu] &= 0 \\ [P^\mu, M^{\rho,\sigma}] &= i(g^{\mu\rho}P^\sigma - g^{\mu\sigma}P^\rho) \\ [M^{\mu\nu}, M^{\rho\sigma}] &= -i(g^{\mu\rho}M^{\nu\sigma} - g^{\mu\sigma}M^{\nu\rho} - g^{\nu\rho}M^{\mu\sigma} + g^{\nu\sigma}M^{\mu\rho}) \\ [P_\mu, Q_a] &= 0 \\ [M_{\mu\nu}, Q_a] &= -(\Sigma_{\mu\nu})_{ab} Q_b \\ \{Q_a, \bar{Q}_b\} &= 2\gamma^\mu_{ab} P_\mu. \end{aligned} \tag{1.16}$$

Nach dem Haag-Lopuszanski-Sohnius-Theorem charakterisiert sie (im wesentlichen) auch schon die allgemeinste Lie-Superalgebra, die konsistent mit den Forderungen einer relativistischen Quantenfeldtheorie ist, und wird deshalb Supersymmetrie-Algebra genannt. Es gibt natürlich noch die Möglichkeit, die Anzahl der Spinorgeneratoren Q_a größer als vier zu wählen. Man gelangt dadurch zu einer erweiterten Supersymmetrie und spricht je nach Anzahl der Spinorgeneratoren von $\mathrm{N} = 2, 3, ...$

Supersymmetrie, im Gegensatz zu N = 1 für die Algebra (1.16).

Die Relation (1.15) erlaubt, Aussagen für die Anzahl der bosonischen bzw. fermionischen Zustände zu treffen. Dazu führt man zuerst den Fermionenzahloperator N_F ein, so daß $(-1)^{N_F}$ den Eigenwert $+1$ für bosonische und -1 für fermionische Zustände hat. Daraus folgt

$$\begin{aligned} \mathrm{Tr}\left((-1)^{N_F}\left\{Q_a,\bar{Q}_b\right\}\right) &= \mathrm{Tr}\left((-1)^{N_F}\left(Q_a\bar{Q}_b+\bar{Q}_bQ_a\right)\right) \\ &= \mathrm{Tr}\left(-Q_a(-1)^{N_F}\bar{Q}_b+Q_a(-1)^{N_F}\bar{Q}_b\right) \\ &= 0. \end{aligned} \tag{1.17}$$

Andererseits gilt mit Gleichung (1.15)

$$0=\mathrm{Tr}\left((-1)^{N_F}\left\{Q_a,\bar{Q}_b\right\}\right)=2\gamma^\mu_{ab}\mathrm{Tr}\left((-1)^{N_F}P_\mu\right), \tag{1.18}$$

so daß sich für den festen Vierimpuls $P_\mu \neq 0$ diese Gleichung auf

$$\mathrm{Tr}(-1)^{N_F}=0 \tag{1.19}$$

reduziert. Nach Gleichung (1.19) enthält somit eine Darstellung der Supersymmetriealgebra gleich viele bosonische wie fermionische Zustände.

1.1.3 Irreduzible Darstellungen

Genauso wie im Falle der Poincaré-Algebra kann nun das mögliche Teilchenspektrum der Supersymmetriealgebra (1.16) durch ihre irreduziblen Darstellungen klassifiziert werden. Dazu ist wieder die Kenntnis der beiden Casimir-Operatoren P^2, C^2 hilfreich

$$P^2=P_\mu P^\mu \quad \wedge \quad C^2=C_{\mu\nu}C^{\mu\nu}, \tag{1.20}$$

wobei $C_{\mu\nu}$ über die Relationen

$$\begin{aligned} C_{\mu\nu} &= Y_\mu P_\nu - Y_\nu P_\mu \\ Y_\mu &= W_\mu + \frac{1}{4}X_\mu \\ X_\mu &= \frac{1}{2}\bar{Q}\gamma_\mu\gamma_5 Q \end{aligned} \tag{1.21}$$

definiert ist. Das Quadrat des Pauli-Lubanski-Vektors W_μ ist selber kein Casimir-Operator mehr, da es nicht mit dem Spinorgenerator Q kommutiert. Somit können Supermultipletts Teilchen mit unterschiedlichem Spin enthalten. An seine Stelle tritt der Superspin Y_μ, der im Ruhesystem die Drehimpulsalgebra

$$[Y^i,Y^i]=im\epsilon^{ijk}Y^k \tag{1.22}$$

erfüllt, wobei m^2 wieder der Eigenwert von P^2 ist. C^2 ist ein Lorentz-Skalar, der im Ruhesystem über die Relation $C^2 = 2m^2Y^2$ mit dem Superspin verknüpft ist. Daraus ergibt sich das Eigenwertspektrum

$$C^2=-2m^4y(y+1) \quad \text{mit} \quad y=0,\frac{1}{2},1,\frac{3}{2},\ldots. \tag{1.23}$$

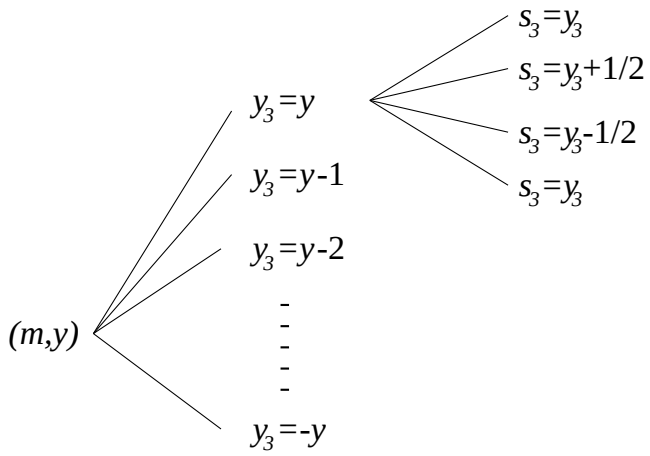


Abbildung 1.1: Graphische Darstellung der massiven Supermultipletts.

Die massiven irreduziblen Darstellungen der Supersymmetriealgebra können somit durch Masse m und Superspin y charakterisiert werden. Innerhalb eines Supermultipletts können die Zustände nach den Eigenwerten my^3 und ms^3 des Superspins Y^3 und des Spins W^3 in z-Richtung klassifiziert werden, da $[W^3, C_{\mu\nu}C^{\mu\nu}] = [W^3, Y^3] = 0$ ist. Erlaubte Eigenwerte für ein Supermultiplett (m, y) sind dabei $-y \leq y^3 \leq y$ und $s^3 \in \{y^3, y^3 + \frac{1}{2}, y^3 - \frac{1}{2}\}$, wobei $s^3 = y^3$ nochmals zweifach entartet ist.

Im masselosen Fall

$$\begin{aligned} P^2 &= W^2 = 0 \quad \text{mit} \quad P_0 = +|\vec{\mathbf{P}}| \\ \Rightarrow\ W_\mu &= \lambda P_\mu \quad \text{mit} \quad \lambda = \pm s \ \wedge\ s \in \left\{ y^3, y^3 \pm \frac{1}{2} \right\} \end{aligned} \tag{1.24}$$

tritt die Helizität λ wieder an die Stelle der z-Komponente des Spins.

1.2 Supersymmetrischer Teilchen-Zoo

1.2.1 Die einfachsten Supermultipletts

Supersymmetrie hat viele Anwendungen in der Physik gefunden, wovon allerdings die Realisierung im Rahmen von Quantenfeldtheorien die wohl wichtigste ist. Hierzu betrachte man zuerst die einfachste Darstellung der Supersymmetriealgebra mit $y = 0$. Im massiven Fall gehört zu diesem *chiralen* Supermultiplett ein Spin $\frac{1}{2}$ Teilchen, ein skalares und ein pseudoskalares Boson mit Spin 0. Für die einfachste Form von SUSY QCD benutzt man z. B. dieses Multiplett, um ein (linkshändiges) Quark und seinen Superpartner zu beschreiben.

Im Rahmen eines minimalen supersymmetrischen Standardmodells, bei dem man die kleinstmögliche Anzahl neuer Teilchen hinzunimmt, wird man alle Materiefelder in masselosen, chiralen Multipletts anordnen. Ihre supersymmetrischen Partner

haben damit allesamt den Spin 0. Man spricht von *Squarks* bzw. *Sleptonen*. Supersymmetrie erfordert dann allerdings, im Gegensatz zum Standardmodell, zwei Higgs-Dubletts, um den Fermionen eine Masse zu geben. Die fermionischen Partner der Higgs-Teilchen werden für gewöhnlich Higgsinos oder kurz Shiggs genannt.

Um die Vektorbosonen des elektroschwachen Sektors und die Gluonen der starken Wechselwirkung einzubauen, benutzt man das *Vektor*-Supermultiplett mit $y = \frac{1}{2}$. Die fermionischen Partner der Eichbosonen haben die Helizität $\lambda = \pm\frac{1}{2}$ und werden im allgemeinen *Gauginos*, und im speziellen *Photinos, Winos, Zinos* und *Gluinos* genannt.

Das Austauschteilchen zur Gravitation, das Graviton, hat die Helizität $\lambda = \pm 2$. Es ist naheliegend, dieses Boson dem *Gravitations*-Multiplett $y = \frac{3}{2}$ zuzuordnen. Dadurch bekommt der supersymmetrische Partner des Gravitons, daß *Gravitino*, die Helizität $\lambda = \pm\frac{3}{2}$.

Multiplett	**Fermionen**			**Bosonen**		
	Teilchen		Spin	Teilchen		Spin
Chiral	Leptonen	$\begin{pmatrix} \nu_L \\ e_L \end{pmatrix}$ e_L^c	$\frac{1}{2}$	Sleptonen	$\begin{pmatrix} \tilde{\nu}_L \\ \tilde{e}_L \end{pmatrix}$ $\tilde{e}_L^c$	0
	Quarks	$\begin{pmatrix} u_L \\ d_L \end{pmatrix}$ u_L^c d_L^c	$\frac{1}{2}$	Squarks	$\begin{pmatrix} \tilde{u}_L \\ \tilde{d}_L \end{pmatrix}$ $\tilde{u}_L^c$ $\tilde{d}_L^c$	0
	Higgsino	$\begin{pmatrix} \tilde{\phi}_1^+ \\ \tilde{\phi}_1^0 \end{pmatrix}$ $\begin{pmatrix} \tilde{\phi}_2^0 \\ \tilde{\phi}_2^- \end{pmatrix}$	$\frac{1}{2}$	Higgs	$\begin{pmatrix} \phi_1^+ \\ \phi_1^0 \end{pmatrix}$ $\begin{pmatrix} \phi_2^0 \\ \phi_2^- \end{pmatrix}$	0
Vektor	Winos	$\begin{pmatrix} \tilde{W}^\pm \\ \tilde{W}^3 \end{pmatrix}$	$\frac{1}{2}$	W-Bosonen	$\begin{pmatrix} W^\pm \\ W^3 \end{pmatrix}$	1
	Photino	$\tilde{B}$		Photon	B	
	Gluinos	$\tilde{G}^a$	$\frac{1}{2}$	Gluonen	G^a	1

Tabelle 1.1: Teilchenspektrum in einem minimalen supersymmetrischen Standardmodell. Alle Elementarteilchen werden als linkshändige (oder chirale) Teilchen aufgefaßt, so daß ein rechtshändiges Teilchen als sein linkshändiges Antiteilchen auftritt, z. B. $e_R = e_L^c$. Supersymmetrische Partner sind durch eine Tilde gekennzeichnet.

1.2.2 Experimenteller Befund

Bis auf das Higgs-Boson sind alle Teilchen des Standardmodells bis heute nachgewiesen worden. Es sollte also gute Gründe geben, warum man Theorien untersuchen will, bei denen das mögliche Teilchenspektrum doppelt so groß ist, während es noch keine Anzeichen für die neu postulierten, supersymmetrischen Partnerteilchen gibt.

Eine der Hauptfragen zum Standardmodell ist der Ursprung der Symmetriegruppe $\mathrm{SU}(3)_C \times \mathrm{SU}(2)_L \times \mathrm{U}(1)_Y$ des Eichsektors. Zu diesen Eichgruppen gehören die drei Kopplungen g_s, g und g', deren Stärke als laufende Kopplungen von der Energieskala abhängt und die durch Zweischleifen- Renormierungsgruppengleichungen zu hohen Energien extrapoliert werden können [LAN95]. Im Bereich von $5 \cdot 10^{13}$ bis 10^{16}GeV werden die Kopplungen ähnlich stark

$$g_s \approx g \approx g' \approx 0.55. \tag{1.25}$$

Allerdings treffen sich nicht alle drei bei einer bestimmten Energie. Es bestehen somit nur noch geringe Hoffnungen, das Standardmodell aus einer vereinheitlichten Eichtheorie, z. B. SU(5), SO(10) oder E(6), mit nur einer Kopplung g_G durch einen einzigen Brechungsschritt abzuleiten. Hingegen zeigt dieselbe Analyse für ein minimales supersymmetrisches Standardmodell, bei dem die SUSY-Partner Massen im Bereich zwischen m_Z und 1TeV besitzen, daß alle Kopplungen bei einer Energieskala von $m_G \approx 2 \cdot 10^{16}$GeV, innerhalb der heutigen experimentellen Unsicherheiten, gleich sind. Supersymmetrie stellt deshalb einen geeigneten Ausgangspunkt für die Formulierung von *Grand Unified Theories* (GUT) dar. Zudem kann die elektroschwache Symmetriebrechung durch Strahlungskorrekturen des Top-Quarks erklärt werden, wenn die Skala der Supersymmetriebrechung über der des elektroschwachen Phasenübergangs liegen sollte. Es gibt aber auch aktuelle Studien zu erweiterten Standardmodellen mit exotischen Fermionen, z. B. vektorartigen Fermionen, die auch ohne Supersymmetrie eine Vereinigung der Kopplungen für $m_G \approx 10^{16}$GeV voraussagen [HR97][RIZ92].

Alle GUT erben aus der Renormierungsgruppenanalyse des Standardmodells das Hierarchieproblem, d. h. den großen Unterschied zwischen der Energieskala m_G, bei der die GUT-Symmetrie gebrochen wird, und der TeV-Skala, der elektroschwachen Symmetriebrechung. Diese beiden Symmetriebrechungen kann man durch zwei Skalarfelder Φ, ϕ mit den Vakuumerwartungswerten

$$< 0|\Phi|0 >= V = O(m_G) \quad \wedge \quad < 0|\phi|0 >= v \simeq 246\mathrm{GeV} \tag{1.26}$$

realisieren. In Analogie zum Higgs-Mechanismus des Standardmodells setzt man für das effektive Potential die Form

$$V_{eff}(\Phi, \phi) = -\frac{1}{2}K\Phi^2 + \frac{1}{4}\Lambda\Phi^4 - \frac{1}{2}\kappa\phi^2 + \frac{1}{4}\lambda\phi^4 + \frac{1}{2}\xi\Phi^2\phi^2 \tag{1.27}$$

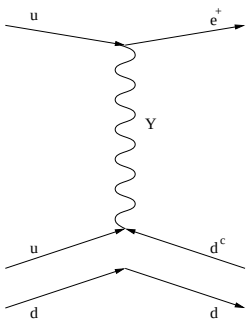
an. Zur Berechnung der GUT-Symmetrie wählt man K und Λ so, daß $V^2 = K/\Lambda$ ist. Für die elektroschwache Brechung erhält man unter der Annahme, daß das

superschwere Higgs-Feld entkoppelt

$$v^2 = (\kappa - \xi V^2)/\lambda, \tag{1.28}$$

d. h., man muß κ mit einer relativen Genauigkeit von 10^{26} einstellen, um genau den gewünschten Vakuumerwartungswert für ϕ zu bekommen. Es mutet für eine physikalische Theorie schon etwas komisch an, wenn man dieses einmal tun muß; im Rahmen einer störungstheoretischen Untersuchung einer GUT ist dieses Feintuning aber in jeder Ordnung erneut nötig. Supersymmetrie entschärft das Hierarchieproblem auf Grund des "non-renormalization"-Theorems [GRS79]. Dieses besagt, daß das effektive Potential (1.27) höchstens durch Wellenfunktionsrenormierung oder endliche Beiträge renormiert werden muß. Diese Renormierungseffekte zerstören aber nicht die Hierarchie der Symmetriebrechungen, d. h., man muß das Feintuning nur einmal durchführen.

Bei der theoretisch attraktiven GUT-Eichgruppe SU(5) werden sowohl in supersymmetrischen als auch in einfachen GUTs. Quarks und Leptonen in einem SU(5)-Multiplett angeordnet. Damit gibt es aber auch Vertizes, die sogenannte *Lepto-Quarks* Eichbosonen X und Y beinhalten, welche Quarks in Leptonen umwandeln können. Das hat weitreichende Konsequenzen für die Stabilität eines Protons, das nun über einen X- bzw. Y-Bosonenaustausch in ein Pion und ein Lepton zerfallen kann. Die wahrscheinlichsten Zerfallskanäle sind hierfür



$$\begin{aligned} p &\rightarrow e^+\pi^0 \\ p &\rightarrow \bar{\nu}_e\pi^+. \end{aligned} \tag{1.29}$$

Ein solcher Prozeß, bei dem die Energien an den äußeren Beinen im Vergleich zum Austauschteilchen klein sind, kann durch eine effektive Lagrange-Funktion beschrieben werden, die aus einer punktförmigen Vier-Fermionen-Wechselwirkung besteht. Dabei ist die Kopplungskonstante in Analogie zur Fermi-Konstante G_F für die Beschreibung des β-Zerfalls proportional zu g_G^2/m_X^2 bzw. g_G^2/m_Y^2. Schon Fermi gab eine Abschätzung für die mittlere Lebensdauer τ bei punktförmigen Strom-Strom Kopplungen an. In diesem Fall lautet sie

$$\tau_p^{-1} = O\left(\frac{1}{16\pi^2}\frac{g_G^4 m_p^5}{m_Y^4}\right). \tag{1.30}$$

Setzt man nun den Wert für g_G aus Gleichung (1.25) ein und schätzt die Massen der Lepto-Quarks durch die GUT-Skala mit $m_Y \approx 10^{15}$GeV ab, so erhält man als erste Abschätzung $\tau_p \approx 5 \cdot 10^{31}$Jahre. Bessere Untersuchungen aus [BL94] ergeben unter Berücksichtigung von Gluon Strahlungskorrekturen den Wertebereich

$$\tau_p = 4.5 \cdot 10^{29\pm1.7}\text{Jahre}. \tag{1.31}$$

Nach den neuesten Ergebnissen des „Super-Kamiokande"-Detektors vom Juni 1998 liegt die untere Schranke der mittleren Lebensdauer des Protons für die Zerfälle

(1.29) bei $1.6 \cdot 10^{33}$Jahren [S98], so daß eine einfache SU(5) Theorie im deutlichen Widerspruch zum Experiment steht.

In einer supersymmetrischen SU(5) GUT kann das Proton natürlich über die gleichen Kanäle zerfallen. Da Supersymmetrie eine „verzögerte“ Vereinheitlichung hin zu $m_G \approx 2 \cdot 10^{16}$ bewirkt, ist das Proton nach Gleichung (1.30) mit $\tau_P \approx 10^{34}$Jahren deutlich stabiler gegen die Zerfälle (1.29). Genaueren Rechnungen zufolge liegt die untere Schranke für diesen Prozeß bei $\tau(p \to e^+\pi^0) \geq 4.1 \cdot 10^{33}$ [MY93]. Allerdings gibt es neben dem konventionellen Lepto-Quark Eichbosonenaustausch auch neue Beiträge durch die Supersymmetrie. Insbesondere dominieren in diesen Modellen Zerfälle unter Beteiligung von Strange-Teilchen, z. B.

$$p \to K^+ \bar{\nu}_\mu. \tag{1.32}$$

Ein Nachweis solcher Zerfälle wäre somit ein starkes Indiz für Supersymmetrie in der Natur. Das aufwendigste Experiment hierfür ist zur Zeit der Soudan 2 Detektor. Nach der Laufzeit von Juni 1997 bis März 1998 kann die untere Schranke für die Lebensdauer des Protons gegenüber diesem Zerfall mit $\tau(p \to K^+\bar{\nu}_\mu) > 4 \cdot 10^{31}$Jahren angegeben werden [A98c], was im Bereich der theoretischen Abschätzungen für ein supersymmetrisches SU(5) Modell liegt, $\tau(p \to K\bar{\nu}_\mu) \geq 6.9 \cdot 10^{31}$Jahre [MY93]. Zudem gibt es noch keine signifikante Häufung von Ereignissen über dem zu erwartenden Untergrund, die als Protonzerfälle nach (1.32) interpretiert werden könnten.

In den vereinheitlichten Modellen werden die drei Kopplungen durch m_G und g_G beschrieben, d. h. man erhält eine Relation zwischen den drei Kopplungen als Voraussage. Daraus kann man nun den schwachen Winkel $sin^2(\theta_w)$ berechnen. Für das minimale SU(5) Modell ist der theoretische Wert schon relativ lange bekannt [ROS84]

$$sin^2\theta_W(m_Z) = 0.214^{+0.006}_{-0.004}. \tag{1.33}$$

Er liegt damit deutlich unterhalb des experimentellen Wertes von [LAN95]

$$sin^2\theta_W(m_Z) = 0.2336 \pm 0.0018, \tag{1.34}$$

während in einer supersymmetrischen SU(5) Theorie der Winkel mit [BL94]

$$sin^2\theta_w(m_Z) = 0.2334 \pm 0.0025 \tag{1.35}$$

erstaunlich gut mit dem Experiment übereinstimmt. Alle Angaben verstehen sich dabei innerhalb des $\overline{\text{MS}}$-Schemas und bei der durch die Masse des Z-Bosons gesetzten Skala m_Z.

Das Higgs-Boson steht ganz oben auf der Liste der gesuchten Elementarteilchen. Resultate des OPAL Detektors am LEP Beschleuniger lassen nur Massen größer als 69.4GeV für dieses Teilchen zu [A98b]. Soll der Higgs-Sektor seinen perturbativen Charakter behalten, so darf die quartische Higgs-Kopplung und damit die Masse des Higgs-Bosons nicht zu groß werden; dem *Higgs Hunter's Guide* zufolge muß

daher auch $m_H \leq O(600\text{GeV})$ gelten [GUN90]. Im minimalen supersymmetrischen Standardmodell reduziert sich diese Grenze bis auf

$$m_H \leq 150\text{GeV} \tag{1.36}$$

für das leichtere der beiden Higgs-Bosonen. Die Daten des OPAL Detektors schließen im Rahmen des minimalen supersymmetrischen Standardmodells Higgs-Massen unterhalb von 59.0GeV schon heute aus [A98a]. Bis Ende 1999 soll LEP in der Lage sein, Higgs-Massen bis 100GeV zu testen. Die nächsten Schritte werden dann von den beiden geplanten Beschleunigern, dem LHC am CERN und TESLA am DESY, gemacht, welche in der Lage sein werden, die Frage nach der Existenz eines relativ leichten Higgs-Bosons zu klären.

Das Standardmodell erzwingt einen Zusammenhang zwischen der Masse des Top-Quarks und der des Higgs-Bosons. Nimmt man an, daß die Higgs-Masse im experimentellen Fenster $60\text{GeV} < m_H < 1\text{TeV}$ liegt, so muß die Masse des Top-Quarks im Bereich

$$m_t = 175 \pm 25\text{GeV} \tag{1.37}$$

liegen [LAN95], was mit den jüngsten Daten der CDF- bzw. D0-Kollaboration übereinstimmt [B98b]

$$m_t = 173.9 \pm 5.2\text{GeV}. \tag{1.38}$$

Im supersymmetrischen Fall hat man einen kleineren erlaubten Bereich für das Higgs-Boson $60\text{GeV} < m_H \leq 150\text{GeV}$. Die erlaubten Massen für das Top-Quark

$$m_t = 160 \pm 17\text{GeV} \tag{1.39}$$

sind trotzdem noch konsistent mit dem Experiment.

Viele Ergebnisse von Präzisionsexperimenten zum Standardmodell geben gleichzeitig untere Massen-Schranken für die supersymmetrischen Partner an, z. B.

- 65GeV für Charginos ($\tilde{W}^{\pm}, \tilde{H}_i^{\pm}$) [GRO96],
- 165GeV für Gluinos [C96],
- 40 GeV für die leichtesten supersymmetrischen-Partner [C96] und
- 65GeV für Selektron und Squark-Massen [A96].

1.3 Supersymmetrische Lagrange-Dichten

Zur Formulierung supersymmetrischer Quantenfeldtheorien benötigt man nun noch unendlichdimensionale, lineare Darstellungen der Supersymmetriealgebra. Aus den zugehörigen Darstellungsräumen stammen dann die gesuchten supersymmetrischen Wirkungen

$$S = \int d^4x \mathcal{L} \,, \tag{1.40}$$

welche invariant gegenüber Supersymmetrietransformationen sind. Der folgende Abschnitt soll eine kurze Übersicht geben, wie man die entsprechenden Lagrange-Dichten konstruieren kann. Dabei wird die Einschränkung auf N = 1 supersymmetrische Theorien gemacht, da Theorien mit N $\geq$ 2 nicht chiral sind. Zudem ist es (fast) unmöglich, in diesen Theorien Supersymmetrie zu brechen [DIN96].

1.3.1 Superfelder

Die Relationen zwischen den Generatoren P_μ und den Weyl-Spinor Supersymmetriegeneratoren Q_A und $\bar{Q}_{\dot{A}}$ lauten nach (1.16) und (1.9) [2]

$$\begin{aligned} [Q_A, P_\mu] = [\bar{Q}_{\dot{A}}, P_\mu] &= [P_\mu, P_\nu] = 0 \\ \{Q_A, Q_B\} &= \{\bar{Q}_{\dot{A}}, \bar{Q}_{\dot{B}}\} = 0 \\ \{Q_A, \bar{Q}_{\dot{B}}\} &= 2\sigma^\mu_{A\dot{B}} P_\mu \end{aligned} \tag{1.41}$$

Um den zusätzlichen fermionischen Generatoren Q_A und $\bar{Q}_{\dot{A}}$ in dieser Lie-Superalgebra Rechnung zu tragen, ist es nützlich, den Parameterraum durch Grassmann-Variablen zu erweitern, z. B. $\mathcal{L} = \mathcal{L}(x^\mu, \theta, \bar{\theta})$ [3]. Wendet man auf ein solches *Superfeld* ein Gruppenelement

$$G(x^\mu, \xi, \bar{\xi}) = \exp\left\{ i(\xi Q + \bar{\xi}\bar{Q} - x^\mu P_\mu) \right\} \tag{1.42}$$

der Supersymmetriegruppe an, so kann man unter Ausnutzung der Baker - Hausdorff Formel die gesuchten Operatoren für die Supersymmetriealgebra

$$\begin{aligned} P_\mu &= i\partial_\mu \\ iQ_A &= \frac{\partial}{\partial\theta^A} - i\sigma^\mu_{A\dot{A}}\bar{\theta}^{\dot{A}}\partial_\mu \\ i\bar{Q}_{\dot{A}} &= -\frac{\partial}{\partial\bar{\theta}^{\dot{A}}} + i\theta^A\sigma^\mu_{A\dot{A}}\partial_\mu \end{aligned} \tag{1.43}$$

identifizieren. Der durch die Superfelder gegebene Darstellungsraum ist im allgemeinen hochgradig reduzibel. Irreduzible Darstellungsräume kann man durch zusätzliche Bedingungen, die kovariant bezüglich der Supersymmetrietransformation sind, erhalten. Ein Beispiel ist

$$\bar{D}_{\dot{A}}\Phi(x^\mu, \theta) = 0 \quad \text{mit} \quad \bar{D}_{\dot{A}} = -\frac{\partial}{\partial\bar{\theta}^{\dot{A}}} - i\theta^A\sigma^\mu_{A\dot{A}}\partial_\mu. \tag{1.44}$$

Die allgemeinste Form für diese Superfelder ist durch

$$\Phi(y, \theta) = \phi(y) + \sqrt{2}\psi(y)\theta + \theta\theta F(x) \tag{1.45}$$

mit $y^\mu = x^\mu + i\theta\sigma^\mu\bar{\theta}$ und $\theta\theta = \theta^A\theta_A = \theta^1\theta_1 + \theta^2\theta_2$ gegeben. Der Teilchengehalt entspricht einem komplexen Skalarfeld ϕ, einem (linkshändigen) Weyl-Spinor ψ und

[2] Die Generatoren $M_{\mu\nu}$ der Lorentz-Transformation bleiben unerwähnt, da von vornherein nur relativistisch kovariante Lagrange-Dichten berücksichtigt werden.

[3] Für $N > 1$ kommen entsprechend mehr Grassmann-Variablen hinzu.

dem skalaren Hilfsfeld F. Damit kann es dem chiralen Multiplett zugeordnet werden, und man spricht an Stelle von $\Phi(y,\theta)$ auch von chiralen Superfeldern. Mit Hilfe der Relationen (1.41) kann man das Verhalten der Felder ϕ, ψ und F unter infinitesimalen Supersymmetrietransformationen nach (1.42) bestimmen. Insbesondere ergibt sich für die Änderung von F

$$\delta F(x) = \frac{i}{\sqrt{2}}\partial_\mu \psi(x)\sigma^\mu \xi \tag{1.46}$$

eine totale Divergenz. Die kovariante Bedingung an ein *Vektor*-Superfeld V lautet $V = V^\dagger$. Fordert man in Analogie zur U(1) Eichtransformation des Vektorpotentials in der QED

$$A_\mu \to A_\mu + \partial_\mu \Lambda, \tag{1.47}$$

daß Superfeldwirkungen invariant unter den Eichtransformationen

$$V \to V + i\left(\Lambda - \Lambda^\dagger\right) \tag{1.48}$$

sind, wobei Λ und damit $\Lambda^\dagger$ chirale Superfelder darstellen, so kann man mit der Wess-Zumino-Eichung ein Vektorsuperfeld in der allgemeinen Form

$$V_{WZ}(x,\theta,\bar\theta) = \theta\sigma^\mu\bar\theta A_\mu(x) + i(\theta\theta)\bar\theta\bar\lambda(x) - i(\bar\theta\bar\theta)\theta\lambda(x) + \frac{1}{2}(\theta\theta)(\bar\theta\bar\theta)D(x) \tag{1.49}$$

schreiben. Der Teilchengehalt besteht somit aus einem Eichfeld $A_\mu(x)$, seinem supersymmetrischen Partner, der durch ein komplexes Weyl-Spinorfeld $\lambda(x)$ beschrieben wird, und einem skalaren Hilfsfeld $D(x)$. Er entspricht damit dem Vektorsupermultiplett. Auch die Änderungen des D-Terms unter den Supersymmetrietransformationen (1.42) können durch eine totale Divergenz beschrieben werden

$$\delta D(x) = \partial_\mu(\lambda(x)\sigma^\mu\bar\xi - \xi\sigma^\mu\bar\lambda(x)). \tag{1.50}$$

Zu einer supersymmetrischen Version des Feldstärketensors gelangt man über die Definition des Feldstärkesuperfelds

$$W_A \stackrel{\text{def}}{=} \bar D^2 D_A V. \tag{1.51}$$

Durch W_A wird ein chirales Superfeld beschrieben, da $\bar D_{\dot B} W_A = 0$ ist. Produkte von chiralen Superfeldern sind wieder chirale Superfelder. Der F-Term entsprechend Gleichung (1.45) von $W^A W_A$ lautet

$$\frac{1}{64}\left(W^A W_A + {W_A}^\dagger {W^A}^\dagger\right)_F = -\frac{1}{4}V^{\mu\nu}V_{\mu\nu} + i\lambda\sigma^\mu\partial_\mu\bar\lambda, \tag{1.52}$$

wobei Terme in $D(x)$, die später innerhalb einer Lagrange-Dichte durch Bewegungsgleichungen eliminiert werden können, weggelassen wurden. Die Größe

$$V_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu$$

entspricht dem Feldstärketensor $F_{\mu\nu}$ einer gewöhnlichen U(1) Eichtheorie.

Im Falle nichtabelscher Eichgruppen benötigt man N Superfelder V^a gemäß den N Generatoren T_a, so daß sich das entsprechende Vektorsuperfeld als

$$V = V^a T^a \qquad \text{mit} \quad [T^a, T^b] = i f^{abc} T^c \tag{1.53}$$

schreiben läßt. In Analogie zur U(1) Eichgruppe definiert man das chirale Feldstärkesuperfeld

$$W^a{}_A \stackrel{\text{def}}{=} \bar{D}^2 D_A V^a - i g f^{abc} \bar{D}^2 (D_A V^b) V^c, \tag{1.54}$$

aus dem man wieder das Pendant $W^{aA} W^a{}_A$ zum Feldstärketensor aufbauen kann. Besonderes Interesse verdient der F-Term dieser chiralen Größe

$$\begin{aligned} \frac{1}{64}\left(W^{aA} W^a{}_A + W^a{}_A{}^\dagger W^{aA\dagger}\right)_F &= -\frac{1}{4} V^a_{\mu\nu} V^{a\mu\nu} + i\lambda^a \sigma^\mu \mathcal{D}_\mu \bar{\lambda}^a + \frac{1}{2} D^a D^a \\ \text{mit} \qquad V^a_{\mu\nu} &= \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g f^{abc} A^b_\mu A^c_\nu \\ \wedge \;\; \mathcal{D}_\mu \bar{\lambda}^{a\dot{A}} &= \partial_\mu \bar{\lambda}^{a\dot{A}} + g f^{abc} A^b_\mu \bar{\lambda}^{c\dot{A}}. \end{aligned} \tag{1.55}$$

Der Term in dem Hilfsfeld $D(x)$ kann nachträglich wieder durch die Bewegungsgleichungen entfernt werden. Die Kopplung des Eichfeldes A_μ an seinen fermionischen supersymmetrischen Partner λ kann mit Hilfe der adjungierten Darstellung

$$\left(R_{adj}(T^b)\right)^{ac} = i f^{abc} \tag{1.56}$$

der Generatoren zur Eichgruppe wieder auf die "gewohnte" Form gebracht werden

$$\mathcal{D}_\mu \bar{\lambda}^{\dot{A}} = \partial_\mu \bar{\lambda}^{\dot{A}} - i g A_\mu \bar{\lambda}^{\dot{A}} \quad \text{mit} \quad A_\mu = A^a_\mu T^a. \tag{1.57}$$

Chirale Materiefelder werden durch das Vektorsuperfeld $\Phi^\dagger e^{-2gV} \Phi$ eichinvariant an ein Eichfeld V angebunden. In physikalischen Anwendungen wird es einige chirale Supermultipletts $\Phi_{(i)}$ geben, die sich evtl. mit unterschiedlichen Darstellungen der Eichgruppe transformieren. Natürlich benutzt man für jedes $\Phi_{(i)}$ die Matrix V, allerdings konstruiert mit der physikalisch motivierten Darstellung $t^a_{(i)}$ zum Feld $\Phi_{(i)}$. So wird z. B. für SUSY-QCD an dieser Stelle die acht 3×3 Matrizen zur **3**- bzw. $\bar{\mathbf{3}}$-Darstellung benutzen und nicht die adjungierte Oktett-Darstellung. Der D-Term des Vektorsuperfelds $\Phi^\dagger e^{-2gV} \Phi$ kann nach kurzer Rechnung bestimmt werden,

$$\begin{aligned} \left(\Phi^\dagger e^{-2gV} \Phi\right)_D &= (\mathcal{D}_\mu \phi)^\dagger (\mathcal{D}^\mu \phi) + i + i\psi \sigma^\mu \mathcal{D}_\mu{}^\dagger \bar{\psi} + F^\dagger F \\ &\quad - i\sqrt{2} g (\phi^\dagger t^a \lambda^a \psi - \bar{\psi} t^a \bar{\lambda}^a \phi) - g \phi^\dagger t^a \mathcal{D}^a \phi \\ \text{mit} \quad \mathcal{D}_\mu \phi &= \partial_\mu \phi - i g t^a A^a_\mu \phi. \end{aligned} \tag{1.58}$$

Mit den eingeführten chiralen Superfeldern und den Vektorsuperfeldern baut man supersymmetrische Wirkungen zusammen. Eine hinreichende Bedingung für die Invarianz der Wirkung unter Supersymmetrietransformationen ist das Verschwinden der Lagrange-Dichte bis auf totale Divergenzen unter Variationen gemäß der Supersymmetrie. Benutzt man somit zum Zusammensetzen der Lagrange-Dichte die F-Terme von chiralen Superfeldern und die D-Terme von Vektorsuperfeldern, so ist

die Wirkung nach den Gleichungen (1.46) und (1.50) qua constructionae supersymmetrisch. Die Forderung nach Renormierbarkeit der resultierenden Theorie schränkt allerdings die möglichen Produktterme zwischen chiralen Materiefeldern stark ein; verträglich sind lediglich Ausdrücke der Form $m_{ij}\Phi_i\Phi_j$ und $g_{ijk}\Phi_i\Phi_j\Phi_k$, so daß die allgemeine Form für eine supersymmetrische Lagrange-Dichte durch

$$\begin{aligned}\mathcal{L} &= -\frac{1}{4}V^a_{\mu\nu}V^{a\mu\nu} + i\lambda^a\sigma^\mu\mathcal{D}_\mu\bar{\lambda}^a + \left(\Phi^\dagger e^{-2gV}\Phi\right)_D \\ &\quad + \left(\frac{1}{2}m_{ij}\Phi_i\Phi_j + \frac{1}{3}g_{ijk}\Phi_i\Phi_j\Phi_k + h.c.\right)_F \end{aligned} \tag{1.59}$$

gegeben ist. Die Art der Kopplung des Eichfeldes an den supersymmetrischen Partner mag auf den ersten Blick etwas überraschend sein. Die Ursache hierfür liegt im Term $\sim f^{abc}A^b_\mu A^c_\nu$, der für nichtabelsche Eichtheorien zum Feldstärketensor $V^a_{\mu\nu}$ hinzukommt. Betrachtet man die Variation von A^c_ν unter einer infinitesimalen Supersymmetrietransformationen nach (1.42)

$$\delta A^c_\nu = i(\xi\sigma^\nu\bar{\lambda}^c(x) - \lambda^c\sigma^\nu\bar{\xi}), \tag{1.60}$$

so erhält man für die Variation in der Wirkung Terme der Form $\sim f^{abc}A^b_\mu\bar{\lambda}^c(x)$. Dies entspricht genau der Kopplung des Eichfeldes an den SUSY-Partner über die Strukturkonstante, d. h. über die adjungierte Darstellung.

Damit ist die Konstruktion supersymmetrischer Lagrange-Dichten abgeschlossen. Aus dem Materieinhalt und der Eichgruppe kann man mit Hilfe von Gleichung (1.59) die entsprechende Lagrange-Dichte bestimmen.

1.3.2 Effektive Wirkungen für N=1 SU(N$_c$) Theorien

Die einfachste N=1 supersymmetrische, nichtabelsche Eichtheorie erhält man für die SU(2)-Gruppe, wobei man keine zusätzlichen Materiefelder Φ an das Eichfeld koppelt. Die Lagrange-Dichte einer solchen Theorie entspricht damit

$$\begin{aligned}\mathcal{L} &= -\frac{1}{4}V^a_{\mu\nu}V^{a\mu\nu} + i\lambda^a\sigma^\mu\mathcal{D}_\mu\bar{\lambda}^a \\ \text{mit}\quad \mathcal{D}_\mu\bar{\lambda}^{a\dot{A}} &= \partial_\mu\bar{\lambda}^{a\dot{A}} + gf^{abc}A^b_\mu\bar{\lambda}^{c\dot{A}} \quad\wedge\quad f^{abc} = \varepsilon^{abc}. \end{aligned} \tag{1.61}$$

Man kann den Fermion-Anteil der Lagrange-Dichte mit Gleichung (1.9) wieder durch einen vier-komponentigen Majorana-Spinor Ψ ausdrücken

$$\begin{aligned}\mathcal{L} &= -\frac{1}{4}V^a_{\mu\nu}V^{a\mu\nu} + \frac{i}{2}\bar{\Psi}^a\gamma^\mu\mathcal{D}_\mu\Psi^a \\ \text{mit}\quad \mathcal{D}_\mu\Psi^a &= \partial_\mu\Psi^a + gf^{abc}A^b_\mu\Psi^c. \end{aligned} \tag{1.62}$$

Dies entspricht aber gerade einer gewöhnlichen SU(2) Yang-Mills-Theorie mit einem masselosen Majorana-Fermion in der adjungierten Darstellung. Führt man für den supersymmetrischen Partner des Eichbosons eine Masse $m_{\tilde{g}}$ ungleich Null ein

$$\mathcal{L}_{mass} = m_{\tilde{g}}\left(\lambda^A\lambda_A + \bar{\lambda}^{\dot{A}}\bar{\lambda}_{\dot{A}}\right) = m_{\tilde{g}}\left(\bar{\Psi}\Psi\right), \tag{1.63}$$

so bricht man die Supersymmetrie lediglich weich, d. h., wichtige Eigenschaften wie das non-renormalization Theorem und die Aufhebung von Divergenzen bleiben erhalten [GG82]. Das Ergebnis ist eine QCD-artige Wirkung, die neben den speziellen Majorana-Eigenschaften sich nur durch die adjungierte Darstellung des Fermions unterscheidet. Deshalb spricht man bei dem supersymmetrischen Partner des Eichbosons in Anlehnung an die Terminologie für die starke Wechselwirkung vom Gluino. Im folgenden ist auf Grund des Majorana-Fermions die Flavour-Anzahl N_f gleich $\frac{1}{2}$.

Für den masselosen Fall $m_{\tilde{g}} = 0$ ist die Theorie unter der (starren) chiralen $\mathrm{U}(1)_\lambda$ Transformation

$$\lambda_A \to e^{i\varphi}\lambda_A \ \wedge \ \bar{\lambda}_{\dot{A}} \to e^{-i\varphi}\bar{\lambda}_{\dot{A}} \quad \text{oder} \quad \Psi \to e^{-i\varphi\gamma_5}\Psi \ \wedge \ \bar{\Psi} \to \bar{\Psi}e^{-i\varphi\gamma_5} \tag{1.64}$$

invariant, was in der Sprache der Superfelder der sogenannten R-Symmetrie für die Superraumkoordinaten θ und $\bar{\theta}$

$$\theta_A \to e^{i\varphi}\theta_A \ \wedge \ \bar{\theta}_{\dot{A}} \to e^{-i\varphi}\bar{\theta}_{\dot{A}} \tag{1.65}$$

entspricht. Der zugehörige Noether-Strom $j^5_\mu = \bar{\Psi}\gamma_\mu\gamma_5\Psi$ ist allerdings nur auf klassischem Niveau erhalten (divergenzfrei). In der quantentheoretischen Version besitzt die $\mathrm{U}(1)_\lambda$ Symmetrie eine Anomalie, da der axiale Vektorstrom j^5_μ durch Beträge von Dreiecksgraphen für die Eichgruppe $\mathrm{SU(N_c)}$ nicht frei von Divergenzen ist

$$\partial^\mu j^5_\mu = \frac{N_c g^2}{32\pi^2}\, e^{\mu\nu\rho\sigma} V^\tau_{\mu\nu} V^\tau_{\rho\sigma}. \tag{1.66}$$

Die Herleitung dieser Gleichung verläuft wie in der QCD, allerdings mit $N_f = \frac{1}{2}$ und dem Gruppentheorie-Faktor

$$\mathrm{Tr}\left(T^a T^b\right) = N_c \delta_{ab}, \tag{1.67}$$

wobei T^a und T^b Generatoren der adjungierten $\mathrm{SU(N_c)}$ Darstellung sind. Allerdings läßt dieser anormale Strom eine Z_{2N_c} Symmetrie übrig

$$\varphi_k = \frac{k\pi}{N_c} \quad \text{mit} \quad k = 0, 1, ..., 2N_c - 1. \tag{1.68}$$

In Rahmen eines Modells für die QCD, bei der man die Quarkmassen vernachlässigt, wird die chirale Symmetrie durch ein Fermion-Kondensat ungleich Null spontan gebrochen. Die daraus resultierenden (masselosen) Goldstone-Bosonen entsprechen den Pionen in der vollen QCD. Auch in der supersymmetrischen $\mathrm{SU(N_c)}$ Eichtheorie bricht das nicht verschwindende Gluino-Kondensat

$$< \lambda\lambda > \neq 0 \tag{1.69}$$

die verbliebene Z_{2N_c} Symmetrie weiter auf eine Z_2 Symmetrie $\lambda \to -\lambda$ herunter. Instanton-Rechnungen zeigen, daß es für die $\mathrm{SU}(2)$ Eichgruppe dann noch genau zwei entartete Grundzustände gibt, die sich gerade im Vorzeichen des Gluino-Kondensats

unterschieden [AKM88]. Eine Gluino-Masse ungleich Null bricht die Koexistenz der beiden Grundzustände zugunsten von $<\lambda\lambda>>0$ für $m_{\tilde{g}}>0$ und $<\lambda\lambda><0$ für $m_{\tilde{g}}<0$ auf. Beim Prozeß $m_{\tilde{g}}\to 0$ wird man also einen Phasenübergang erster Ordnung im Parameter $<\lambda\lambda>$ erwarten. Ebenso würde man aus einem Vorzeichenwechsel im Gluino-Kondensat auf ein Verschwinden der Gluino-Masse und damit (im Kontinuumslimes) auf eine nicht weich gebrochene Supersymmetrie schließen können. Darüber hinaus wird noch die mögliche Existenz einer symmetrischen Phase ohne chirale Symmetriebrechung mit $<\lambda\lambda>=0$ kontrovers diskutiert [KS97][SZ97].

Ähnlich wie in der QCD wird man für das Teilchen-Spektrum dieses Modells farblose Bindungszustände zwischen Gluinos und/oder Gluonen erwarten. Im supersymmetrischen Punkt mit Gluino-Masse $m_{\tilde{g}}=0$ sollten die Bindungszustände in Form von supersymmetrischen Multipletts organisiert sein.

Zur Beschreibung der leichtesten Supermultipletts ist es sinnvoll, effektive Feldtheorien, die aus geeigneten farblosen Operatoren zusammengesetzt sind, auszunutzen. Veneziano und Yankielowicz schlugen schon 1982 eine solche effektive N=1 Super-Yang-Mills-Theorie vor [VY82]. Der zum Aufbau der Wirkung benutzte Operator S entspricht dem chiralen Superfeld

$$S \equiv -\frac{\beta(g)}{32g} W^A W_A = \phi(y) + \sqrt{2}\psi(y)\theta + \theta\theta F(x), \tag{1.70}$$

wobei die β-Funktion des Modells gemäß [NSVZ83] exakt bekannt ist. Die Konstituenten des Operators S können daraus berechnet werden

$$\phi = \frac{\beta(g)}{2g}\lambda^A\lambda_A\,, \qquad \sqrt{2}\psi_A = -\frac{\beta(g)}{2g}\left\{-i\lambda_A D + (\sigma^{\mu\nu}\lambda)_A V_{\mu\nu}\right\}$$
$$F = -\frac{\beta(g)}{g}\left\{-\frac{1}{4}V^{\mu\nu}V_{\mu\nu} - \frac{i}{2}\lambda\sigma^\mu\partial_\mu\bar{\lambda} - \frac{i}{8}V^{\mu\nu}\varepsilon_{\mu\nu\rho\sigma}V_{\rho\sigma} + \partial^\mu J^5_\mu + \frac{1}{2}D^2\right\}.$$

Die gewünschten Symmetrien und erwarteten Anomalien legen die Veneziano-Yankielowicz-Wirkung bereits fest

$$\begin{aligned}\mathcal{L}_{eff} &= \frac{1}{\alpha}\left(\sqrt[3]{S^\dagger S}\right)_D + \gamma\left(S\log\frac{S}{\mu^3} - S\right)_F + h.c. \\ &\text{mit} \quad \mu = \mu_0 \exp\frac{8\pi^2}{3N_c g_0^2} \;\wedge\; \alpha,\gamma > 0.\end{aligned} \tag{1.71}$$

Einsetzen von Gleichung (1.70) liefert eine Lagrange-Dichte, die quadratisch in ϕ ist und Vier-Fermionen-Kopplungen aufweist. Ein weiterer Konstituent ist der Operator

$$\begin{aligned}\chi \sim V_{\mu\nu}\sigma^{\mu\nu}\lambda \quad \text{mit} \qquad & \sigma^{\mu\nu} = \frac{i}{4}\left(\sigma^\mu\bar{\sigma}^\nu - \sigma^\nu\bar{\sigma}^\mu\right) \\ \wedge \quad & \sigma^\mu = (\mathbb{1},\vec{\sigma})\,, \quad \bar{\sigma}^\mu = (\mathbb{1},-\vec{\sigma}) = \sigma_\mu.\end{aligned} \tag{1.72}$$

Im niedrigsten Supermultiplett wird man nach der effektiven Wirkung von Veneziano-Yankielowicz somit die folgenden Teilchen, ausgedrückt durch Majorana-Spinoren, erwarten:

- $\bar{\Psi}\gamma_5\Psi \sim$ pseudoskalares Boson mit Masse $m^{0^-}_{\tilde{g}\tilde{g}}$ und Spin 0. In der QCD spricht man für das entsprechende pseudoskalare Meson im $\mathrm{SU}(3)_f$ Singulett vom η' Teilchen, weshalb dieser Zustand im weiteren mit a-η' abgekürzt wird.
- $\bar{\Psi}\Psi \sim$ skalares Boson mit Masse $m^{0^+}_{\tilde{g}\tilde{g}}$ und Spin 0. Hierfür wählt man, wieder in Analogie zur QCD, den Namen a-f_0 Teilchen, da es am ehesten dem $\mathrm{SU}(3)_f$ Singulett Meson f_0 entspricht.
- $V_{\mu\nu}\Sigma^{\mu\nu}\Psi \sim$ Majorana-Fermion mit Spin $\frac{1}{2}$, der sogenannte *Gluino-Glueball*. Die QCD kennt auf Grund der fundamentalen Darstellung keinen vergleichbaren farblosen Operator. Die Masse wird mit $m_{g\tilde{g}}$ gekennzeichnet.

Die zu erwartende Massenaufspaltung in diesem Supermultiplett wird durch ein leichtes Brechen der Supersymmetrie mit einer nicht verschwindenden Gluino-Masse in [EHS97] abgeschätzt

$$\frac{\partial m^{0^-}_{\tilde{g}\tilde{g}}}{m_{\tilde{g}}} \div \frac{\partial m_{g\tilde{g}}}{m_{\tilde{g}}} \div \frac{\partial m^{0^+}_{\tilde{g}\tilde{g}}}{m_{\tilde{g}}} = 5 \div 6 \div 7. \tag{1.73}$$

Allerdings läßt die effektive Veneziano-Yankielowicz-Wirkung einige Fragen offen. Um diese Wirkung zu erreichen, werden die farblosen Operatoren $\sim V^{\mu\nu}V_{\mu\nu}$ und $\sim V^{\mu\nu}\varepsilon_{\mu\nu\rho\sigma}V_{\rho\sigma}$ ausintegriert. Es gibt aber ad hoc keinen Grund zu der Annahme, daß die mit diesen Operatoren verbundenen Gluon-Gluon Grundzustände, die sogenannten *Gluebälle*, schwerer sind als die Bindungszustände des Veneziano-Yankielowicz-Multipletts. Zudem können Gluebälle an a-η' oder a-f_0 Zustände koppeln und zu nicht vernachlässigbaren dynamischen Effekten führen. In Anlehnung zu QCD-Rechnungen wird man auch in diesem Modell erwarten, daß der 0^+ Glueball mit Masse $m^{0^+}_{gg}$ der leichteste Glueball-Zustand sein wird [Wes96].

In einer aktuellen Arbeit von Farrar, Gabadadze und Schwetz (FGS) wird deshalb eine effektive Wirkung bestehend aus zwei chiralen Supermultiplett-Operatoren vorgeschlagen [FGS98]. Als Folge hiervon sagt die effektive Wirkung zwei Multipletts voraus. Die Massen der Teilchen innerhalb eines Multipletts stimmen für die ungebrochene Supersymmetrie überein, allerdings sind die beiden Massen der Multipletts nicht gleich. Das schwerere Multiplett entspricht dem Veneziano-Yankielowicz-Multiplett, das leichtere Supermultiplett besteht aus

- $V^{\mu\nu}V_{\mu\nu} \sim 0^+$ Glueball,
- $V^{\mu\nu}\varepsilon_{\mu\nu\rho\sigma}V_{\rho\sigma} \sim 0^-$ Glueball und
- einem fermionischen Gluino-Glue Grundzustand, das Pendant zum Gluino-Glueball im Veneziano-Yankielowicz-Multiplett.

Die Form der effektiven Wirkung erlaubt ausdrücklich Massenmischung im 0^+ bzw. 0^- Kanal, d. h. Mischung für a-$f_0 \Leftrightarrow 0^+$Glueball und für a-$\eta' \Leftrightarrow 0^-$Glueball. Zudem würde eine Massenmischung den Abstand zwischen den Massen der beiden Multipletts vergrößern. Führt man wieder eine Gluino-Masse $m_{\tilde{g}}$ ein, so wird die Supersymmetrie wieder weich gebrochen, und die beiden Multipletts spalten auf. Man

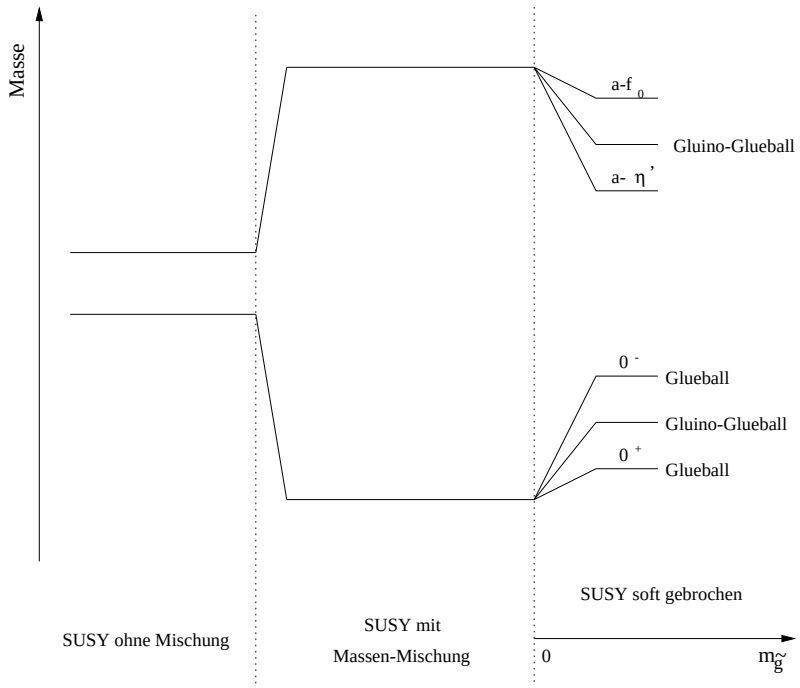


Abbildung 1.2: Graphische Darstellung der Massen gemäß der effektiven FGS-Wirkung in den beiden leichtesten Multipletts beim Übergang von Supersymmetrie zur weich gebrochenen Supersymmetrie. Die Bezeichnungen für die Teilchen im 0^+ bzw. 0^- Kanal sind nur für kleine Mischungswinkel sinnvoll.

beachte, daß sich die Art der Aufspaltung für das schwerere Multiplett quantitativ von der Voraussage (1.73) für die Veneziano-Yankielowicz-Wirkung unterscheidet

$$\frac{\partial m^{0^-}_{\tilde{g}\tilde{g}}}{m_{\tilde{g}}} \div \frac{\partial m_{g\tilde{g}}}{m_{\tilde{g}}} > 1 \quad \wedge \quad \frac{\partial m_{g\tilde{g}}}{m_{\tilde{g}}} \div \frac{\partial m^{0^+}_{\tilde{g}\tilde{g}}}{m_{\tilde{g}}} > 1. \tag{1.74}$$

Kapitel 2

Konstruktion der Gitterwirkung

2.1 Motivation

Nachdem Seiberg und Witten 1994 exakte Lösungen für einige N=2 bzw. N=4 supersymmetrische Eichtheorien vorgestellt haben [SW94a][SW94b][1], ist das Interesse an supersymmetrischen Quantenfeldtheorien sprunghaft gestiegen. Getragen wird dieses Interesse von der Hoffnung, durch diese Theorien einige nichtperturbative Effekte wie Confinement oder chirale Symmetriebrechung zu verstehen. Darüber hinaus konnte Seiberg ein Jahr später auch exakte Resultate für N=1 supersymmetrische Eichtheorien gewinnen [Sei95]. Ausgehend von den exakten Lösungen kann man zudem auch Modelle mit leichter Symmetriebrechung behandeln, indem man die Theorie erst einmal ohne Brechung löst, um dann die Symmetriebrechung im Rahmen einer Störungsentwicklung einzuführen und damit zu einem exakten Ausdruck für die effektive Wirkung zu gelangen [AGDKM96]. Allerdings versagt diese Technik, sobald die Skala der Supersymmetriebrechung größer wird als die dynamische Skala Λ des Modells.

Es ist natürlich interessant, die nichtperturbativen Effekte im Rahmen einer Gitterregularisierung durch numerische Simulationen zu bestätigen und zusätzlich die Aussagen jenseits der exakten Lösungen zu testen bzw. zu erweitern. Dem steht nur ein gewichtiges Problem im Wege. Es gibt noch keine Gittereichtheorie mit exakter Supersymmetrie. Ein Grund hierfür ist das Gitter selbst, das die Lorentz- und damit auch die Supersymmetrie bricht. Benutzt man überdies Wilson-Fermionen zur Beschreibung der Gluinos, so bricht der Wilson-Term sowohl die chirale Symmetrie als auch die Supersymmetrie.

Lediglich die Konstruktion eines zwei- bzw. vierdimensionalen supersymmetrischen Wess-Zumino-Modells gelang auf dem Gitter im Kontext der perfekten Wirkungen [Bie98]. Die Neuberger-Wirkung, welche masselose Quarks auf dem Gitter exakt beschreibt, stellt einen natürlichen Zugang für verbesserte Wirkungen zum masselosen Gluino- (SUSY) Limes dar [Neu98]. Allerdings haben sowohl perfekte Wirkungen

[1] Eine gute Einleitung in die Methoden von Seiberg und Witten bietet [AGH97].

als auch die Neuberger-Wirkung nichtlokale Wechselwirkungsterme, die eine Simulation vor allem auf den leistungsstarken Parallelrechnern sehr erschweren.

Im Gegensatz zu diesen Ansätzen schlagen Curci und Veneziano für den Fall der N=1 Super-Yang-Mills-Theorie vor, mit einer einfachen, nichtsupersymmetrischen Gitterwirkung zu starten und die Supersymmetrie im Kontinuumslimes zu restaurieren. Dies erfordert natürlich ein „Tuning“ der nackten Parameter. Im Falle einer N=1 Super-Yang-Mills-Theorie gilt es lediglich, zwei Parameter geeignet einzustellen, die Eichkopplung und die Gluino-Masse. Diese Prozedur ähnelt damit der Restaurierung der chiralen Symmetrie bei Gitter-QCD-Simulationen mit Wilson-Fermionen. Allerdings benötigt man einen effizienten Monte-Carlo-Algorithmus für die einfache N=1 Gitterwirkung. Eine erste Möglichkeit hierfür bietet die quenched Approximation, d. h., im Rahmen des Update-Prozesses ignoriert man Vakuumpolarisationseffekte, indem man nur die Eichfelder gemäß eines reinen Eichfeld-Updaters verändert. Allerdings lebt Supersymmetrie gerade von der Balance zwischen fermionischen und bosonischen Freiheitsgraden, so daß diese Approximation die Supersymmetrie „hart“ bricht. Man kann nicht hoffen, mit einem solchen Algorithmus den Curci-Veneziano-Ansatz erfolgreich nutzen zu können.

Simulationen mit der vollen Theorie, d. h. mit dynamischen Fermionen, sind notorisch aufwendig und zwar sowohl im Umfang der zu schreibenden Software als auch bei der benötigten Rechenleistung. Es gibt allerdings lohnende physikalische und algorithmische Fragestellungen, die sich im Rahmen der nichtperturbativen Gittersimulationen einer SU(2) Theorie mit Gluinos untersuchen lassen:

- Kann man Supersymmetrie durch den Curci-Veneziano-Ansatz auf dem Gitter restaurieren?

- Obwohl exakte Lösungen vorhanden sind, ist das Verhalten bei niedrigen Energien nicht klar. Besonderes Interesse gilt den Supermultipletts für die leichtesten Bindungszustände und der Massenaufspaltung innerhalb dieser Multipletts bei weich gebrochener Supersymmetrie.

- Ist der Phasenübergang für verschwindende Gluino-Masse erster Ordnung, wie groß ist der Sprung im Gluino-Kondensat und gibt es eine Phase mit Gluino-Kondensat gleich Null?

- Test der exakten Resultate.

- Weiterentwicklung von Algorithmen mit Majorana-Fermionen bzw. für ungerade Quark-Flavour-Anzahl.

- Simulation bei sehr kleinen Gluino-Massen.

Gerade aus den beiden letzten Punkten können auch Impulse für QCD-Simulationen resultieren.

2.2 Majorana-Fermionen auf dem Gitter

Der erste Schritt, um eine Gitterwirkung für eine numerische Simulation zu formulieren, geht zur euklidischen Raum-Zeit. Die Lagrange-Dichte (1.62) geht dadurch in

$$\begin{aligned} \mathcal{L} &= -\frac{1}{4}V^a_{\mu\nu}V^{a\mu\nu} + \frac{1}{2}\bar{\Psi}^a\gamma^\mu\mathcal{D}_\mu\Psi^a \\ \text{mit} \quad V^a_{\mu\nu} &= \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + g f^{abc} A^b_\mu A^c_\nu \\ \wedge\ \mathcal{D}_\mu\Psi^a &= \partial_\mu\Psi^a + g f^{abc} A^b_\mu \Psi^c \end{aligned} \tag{2.1}$$

über. Die Ladungskonjugationsmatrix C erfüllt im Euklidischen die folgenden Eigenschaften

$$C\gamma_\mu C^{-1} = -\gamma_\mu^T\,, \quad -C = C^T = C^{-1} = C^+. \tag{2.2}$$

Für eine Gittertheorie sind allerdings Dirac-Spinoren anstelle der Majorana-Spinoren geeigneter. Aus diesem Grund drückt man zwei Majorana-Spinoren $\Psi^{(i)}$ durch einen Dirac-Spinor ψ aus

$$\Psi^{(1)} = \frac{1}{\sqrt{2}}\left(\psi + C\bar{\psi}^T\right)\,, \quad \Psi^{(2)} = \frac{i}{\sqrt{2}}\left(-\psi + C\bar{\psi}^T\right). \tag{2.3}$$

Einsetzen zeigt sofort, daß $\Psi^{(1)}$ und $\Psi^{(2)}$ die Majorana-Bedingung $\bar{\Psi} = \Psi^T C$ erfüllen. Die Umkehrrelationen lauten

$$\psi = \frac{1}{\sqrt{2}}\left(\Psi^{(1)} + i\Psi^{(2)}\right)\,, \quad \psi_c = C\bar{\psi}^T = \frac{1}{\sqrt{2}}\left(\Psi^{(1)} - i\Psi^{(2)}\right). \tag{2.4}$$

Unter Ausnutzung des von I. Montvay vorgeschlagenen „doubling tricks“ läßt sich daraus die gewünschte Wirkung berechnen [Mon96]. Ausgangspunkt hierfür ist eine gewöhnliche SU(N_c) Yang-Mills-Gitterwirkung, in der die Dirac-Spinoren durch Wilson-Fermionen in der adjungierten Darstellung realisiert sind

$$\begin{aligned} S &= S_f + S_g \\ \text{mit}\ S_f &= \sum_x \left\{ \bar{\psi}^r_x\psi^r_x - K\sum_{rs}\sum_{\mu=1}^{4}\left[\bar{\psi}^r_{x+\hat{\mu}}V_{rs,x\mu}(1+\gamma_\mu)\psi^s_x + \bar{\psi}^r_x V^T_{rs,x\mu}(1-\gamma_\mu)\psi^s_{x+\hat{\mu}}\right]\right\} \\ \wedge\ S_g &= \beta\sum_P\left(1 - \frac{1}{N_c}\mathrm{Re}\,\mathrm{Tr}\,U_P\right). \end{aligned} \tag{2.5}$$

K bezeichnet den Hopping-Parameter, und der irrelevante Wilson-Parameter zum Entfernen der Fermion-Doubler im Kontinuumslimes ist auf $r = 1$ fixiert. S_g entspricht der Wilson-Eichwirkung mit der Notation

$$\begin{aligned} \sum_P &= \sum_x \sum_{1\le\mu\le\nu\le 4} \\ U_P &= U_{x;\mu\nu} = U^{-1}_{x\nu}U^{-1}_{x+\hat{\nu},\mu}U_{x+\hat{\mu},\nu}U_{x\mu}. \end{aligned} \tag{2.6}$$

$U_P = U_{x;\mu\nu}$ wird kurz als Plaquette bezeichnet und ist in Abbildung 2.1 skizziert. Somit hat S_g die angenehme Eigenschaft, daß ein Link lediglich mit der ihn umgebenden „Klammer“-Summe wechselwirkt. Die Eichkopplung β ist mit der Kontinu-

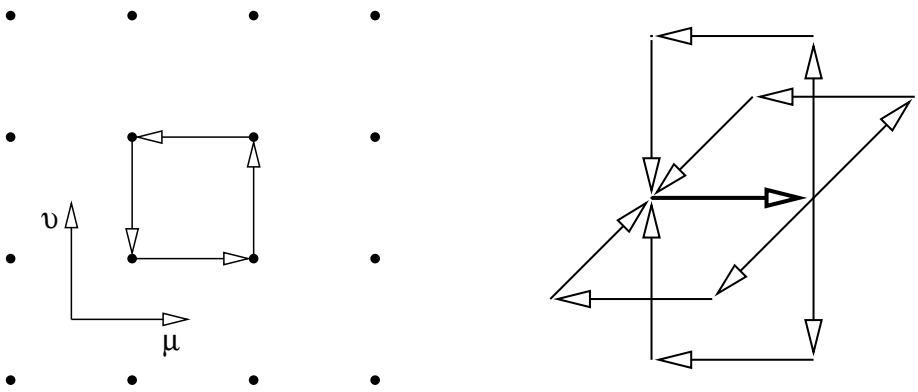


Abbildung 2.1: Einfachste Wechselwirkungen für eine Gittereichtheorie. Das linke Diagramm symbolisiert eine elementare Plaquette, während das rechte Diagramm alle Wechselwirkungsplaquetten für einen speziellen Link zeigt.

umskopplung g über die Beziehung

$$\beta = \frac{2N_c}{g^2} \tag{2.7}$$

verbunden. Eichmatrizen V in der adjungierten Darstellung können aus der fundamentalen Darstellung U über die Relation

$$V_{rs,x\mu} = 2\mathrm{Tr}\left(U^{\dagger}_{x\mu} T_r U_{x\mu} T_s\right) = V^{*}_{rs,x\mu} = V^{-1T}_{rs,x\mu} \tag{2.8}$$

gewonnen werden, wobei die Generatoren T_r der Eichgruppe $\mathrm{SU}(N_c)$ durch die Gleichung $\mathrm{Tr}\,(T_r T_s) = \frac{1}{2}\delta_{rs}$ normiert sind. Im Fall $N_c = 2$ zieht man natürlich mit $T_r = \frac{1}{2}\sigma_r$ die Pauli-Matrizen zur Definition der Generatoren heran (Im Anhang A.4 wird diese Umrechnung auf Ebene der Komponenten besprochen.). Um zu einer kompakteren Schreibweise zu gelangen, führt man die Fermion-Matrix

$$Q_{yr,xs}[U] = \delta_{yx}\delta_{rs} - K \sum_{rs} \sum_{\mu=1}^{4} \left[\delta_{y,x+\hat{\mu}}(1+\gamma_\mu) V_{rs,x\mu} + \delta_{y+\hat{\mu},x}(1-\gamma_\mu) V^{T}_{rs,y\mu}\right] \tag{2.9}$$

ein, die es erlaubt, den fermionischen Anteil als Bilinearform

$$S_f = \sum_{xs,yr} \bar{\psi}^r_y Q_{yr,xs} \psi^s_x \tag{2.10}$$

zu schreiben. Wendet man die Relationen (2.4) hierauf an, so bekommt man eine fermionische Wirkung mit zwei Majorana-Komponenten

$$S_f = \frac{1}{2} \sum_{j=1}^{2} \sum_{xr,ys} \bar{\Psi}^{(j)r}_y Q_{yr,xs} \Psi^{(j)s}_x . \tag{2.11}$$

Das Pfadintegral über den fermionischen Anteil der Wirkung kann nun entweder durch einen Dirac-Spinor oder mit Hilfe zweier Majorana-Spinoren ausgedrückt werden

$$\int \left[d\bar{\psi} d\psi\right] e^{-S_f} = \int \left[d\bar{\psi} d\psi\right] e^{-\bar{\psi} Q \psi} = \det Q = \prod_{j=1}^{2} \int \left[d\Psi^{(j)}\right] e^{-\frac{1}{2}\bar{\Psi}^{(j)} Q \Psi^{(j)}} . \tag{2.12}$$

Die Fermion-Matrix $Q = \gamma_5 Q^\dagger \gamma_5$ selbst ist nicht hermitesch, allerdings gilt dies für

$$\tilde{Q} \stackrel{\text{def}}{=} \gamma_5 Q = \tilde{Q}^\dagger, \tag{2.13}$$

so daß $\det Q = \det \tilde{Q}$ reell sein muß. Darüber hinaus ist jeder Eigenwert von Q und $\tilde{Q}$ (mindestens) zweifach entartet, da diese Matrizen die Relationen

$$CQC^{-1} = Q^T\ , \quad C\gamma_5\tilde{Q}\gamma_5 C^{-1} = \tilde{Q}^T \tag{2.14}$$

erfüllen [MON98b]. Drückt man nun die Fermion-Determinante $\det \tilde{Q}$ mit Hilfe ihrer reellen Eigenwerte $\tilde{\lambda}_i$ aus, so tritt das Vorzeichen

$$\det Q = \det \tilde{Q} = \prod_i \tilde{\lambda}_i^2 \geq 0 \tag{2.15}$$

zutage. Zur Konstruktion der „naiven“ N=1 Super-Yang-Mills-Gittertheorie wird ein Ausdruck für das Pfadintegral mit nur einem Majorana-Fermion gesucht, der sich somit für $\Psi^{(1)} = \Psi$ oder $\Psi^{(2)} = \Psi$ als

$$\int [d\Psi] e^{-\frac{1}{2}\bar{\Psi} Q \Psi} = \pm\sqrt{\det Q} \tag{2.16}$$

schreiben läßt. Legt man sich beim Vorzeichen auf $+1$ fest, so gelangt man zur Curci-Veneziano-Wirkung

$$S_{CV}[U] = \beta \sum_P \left(1 - \frac{1}{N_c}\mathrm{Re\ Tr}\, U_P\right) - \frac{1}{2}\log\det Q[U]. \tag{2.17}$$

Die Festlegung des Vorzeichens ist innerhalb eines Monte-Carlo Update-Zyklus nicht unproblematisch. Für ein gegebenes Eichfeld kann man zwar das Vorzeichen festlegen, es ist aber auf dem Gitter a priori nicht sichergestellt, daß sich das Vorzeichen während eines Monte-Carlo-Updates nicht ändert. Im Kontinuum gibt es, bedingt durch die adjungierte Darstellung und den damit paarweise auftretenden Eigenwerten, keinen Vorzeichenwechsel [HSU98]. Man kann annehmen, daß dies auch für eine Gittertheorie gelten sollte, welche den Kontinuumslimes $a \to 0$ simuliert.

2.3 Der Pfaffian zur Fermion-Matrix

Das Vorzeichen in Gleichung (2.16) kann mit Hilfe eines Pfaffian für eine gegebene Eichkonfiguration bestimmt werden [MON96]. Dazu nutzt man aus, daß für eine komplexe oder reelle antisymmetrische $(2n \times 2n)$-Matrix $M_{\alpha\beta} = -M_{\beta\alpha}$ das $2n$-dimensionale Grassmann-Integral durch den Pfaffian

$$\begin{aligned} Pf(M) &\equiv \int [d\Psi] \exp\left(-\frac{1}{2}\sum_{\alpha\beta} \Psi_\alpha M_{\alpha\beta} \Psi_\beta\right) \\ &= \frac{1}{2^n n!} \sum_{\pi \in S_{2n}} (-1)^\pi M_{\pi(1),\pi(2)} \cdots M_{\pi(2n-1),\pi(2n)} \end{aligned} \tag{2.18}$$

gegeben ist [ROE91]. Hierbei ist S_{2n} die Menge aller bijektiven Abbildungen

$$\pi : \{1, ..., 2n\} \to \{1, ..., 2n\}, \tag{2.19}$$

und das Signum ist durch

$$(-1)^{\pi} \equiv \begin{cases} +1 & \text{für } \pi(1), ..., \pi(2n) \text{ gerade Permutation von } 1, ..., 2n \\ -1 & \text{für } \pi(1), ..., \pi(2n) \text{ ungerade Permutation von } 1, ..., 2n \end{cases} \tag{2.20}$$

definiert. Man kann nun das Pfadintegral aus Gleichung (2.16) als Pfaffian auffassen, indem man die Wirkung mit Hilfe der Majorana-Bedingung

$$-\frac{1}{2}\bar{\Psi}Q\Psi = -\frac{1}{2}\Psi^T CQ\Psi = -\frac{1}{2}\Psi^T M\Psi \tag{2.21}$$

umschreibt und ausnutzt, daß die Matrix $M \equiv CQ$ antisymmetrisch ist, so daß

$$\int [d\Psi] e^{-\frac{1}{2}\bar{\Psi}Q\Psi} = Pf(M) = Pf(CQ) \tag{2.22}$$

gilt. Diese Gleichung ist für einen Monte-Carlo-Algorithmus aber nicht geeignet, da der Aufwand zur Berechnung des Pfaffians mit einer LU-Zerlegung zur Determinantenberechnung vergleichbar ist [KLM98]. Insbesondere muß hierfür eine komplette $(n \times n)$-Matrix im Speicher gehalten werden, was die Berechnung auf kleine Gitter beschränkt.

Da die Fixierung des Vorzeichens auf $+1$ aber ein neuralgischer Punkt für das gesamte Projekt ist, hat I. Montvay ein Programm geschrieben, welches den Pfaffian auf einer gegebene SU(2) Eichkonfiguration gemäß der Fermion-Matrix (2.9) berechnen kann. Selbst bei der mit 1GByte Hauptspeicher ausgestatteten Cray-T90 beträgt die maximale Gittergröße, bei der die Berechnung noch möglich ist, $4^3 \times 8$. Auch die benötigte CPU-Zeit von ca. 30min für diese Gittergröße zeigt, daß man den Pfaffian nicht sinnvoll im Rahmen eines Monte-Carlo-Algorithmus einsetzen kann.

Auch wenn es ein Vorgriff auf die Inhalte der nachfolgenden Kapitel darstellt, soll an dieser Stelle kurz auf die Ergebnisse der Untersuchung des Pfaffians eingegangen werden, da die Fixierung des Vorzeichens auf $+1$ das weitere Vorgehen beeinflußt. Die getesteten Eichkonfigurationen wurden dafür mit dem im nachfolgenden Kapitel vorgestellten Monte-Carlo-Algorithmus bei physikalisch interessanten Parametersätzen (K, β) auf einem massiven Parallelrechner vom Type Cray-T3E generiert. Ebenfalls noch auf der T3E wurden jeweils der kleinste und größte Eigenwert von $Q^\dagger Q$ bestimmt, um dann anschließend die Feldkonfiguration in ein für das T90-Programm verständliches Format zu konvertieren.

Bei allen von I. Montvay untersuchten Konfigurationen war der Pfaffian deutlich größer als Null. Die Abbildung 2.2 zeigt das Ergebnis der umfangreichsten Studie mit 90 Konfigurationen bei $\beta = 2.3,\ K = 0.1925$. Sollte es doch Konfigurationen mit negativem Pfaffian geben, so haben diese Tests aber gezeigt, daß sie sehr selten sein sollten. Somit ist ihr statistisches Gewicht klein.

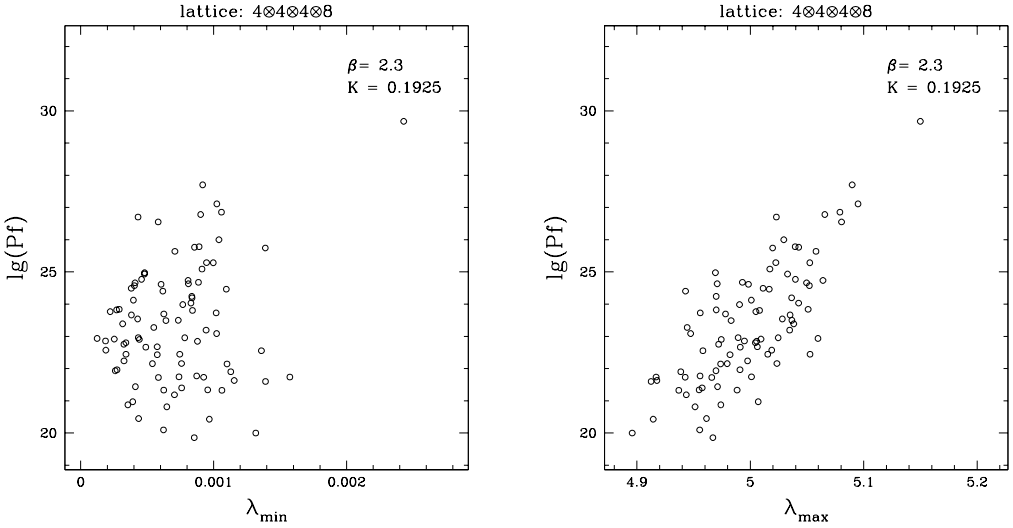


Abbildung 2.2: Die Werte des Pfaffians auf 90 repräsentativen Eichkonfigurationen. Im linken Diagramm sind die Pfaffians gegen den kleinsten Eigenwert, im rechten Diagramm gegen den größten Eigenwert von $Q^\dagger Q$ eingetragen.

Kapitel 3

Monte-Carlo-Algorithmus

3.1 Multi-Bosonischer Algorithmus

Die Curci-Veneziano-Wirkung (2.17) stellt hohe Ansprüche an einen Monte-Carlo-Algorithmus. Die häufig zur Simulation dynamischer Fermionen verwendeten hybriden Monte-Carlo-Algorithmen scheiden schon auf Grund der Flavour-Zahl aus; nur Vielfache von vier Majorana-Flavours können mit diesem Algorithmus simuliert werden. In [DG96] wird der zur Familie der Hybrid Molecular Dynamics Algorithms gehörende R-Algorithmus für die Curci-Veneziano-Wirkung getestet. Aber ebenso wie beim Langevin-Algorithmus muß man für diesen Algorithmus die Extrapolation der Integrationsschrittweite zu $\Delta t = 0$ durchführen, was umständlich ist und viel Rechenzeit benötigt.

Basierend auf den von Lüscher 1993 entwickelten Multi-Bosonischen Fermionenalgorithmen ([LÜS94]), schlug I. Montvay einen Zwei-Schritt-Algorithmus u. a. für die Curci-Veneziano-Wirkung vor [MON96]. Während für QCD-artige Anwendungen noch nicht geklärt ist, ob Multi-Bosonische Algorithmen besser als hybride Monte-Carlo-Algorithmen sind [EB98], kann man für diese Anwendung von einem Vorteil gegenüber den Hybrid Molecular Dynamics Algorithms ausgehen, da die Extrapolation zu $\Delta t = 0$ entfällt.

Der Zwei-Schritt-Algorithmus wurde mit der Zeit fortlaufend optimiert, z. B. durch geeignete Präkonditionierung und einen hybriden Meßkorrekturschritt für kleine Gluino-Massen, das Konzept der Zwei-Schritt Approximation blieb aber unangetastet und hat sich als sehr tragfähig erwiesen. Die portable Implementation des Algorithmus in C++, welche im Rahmen dieser Arbeit entstand, ist auf einer Hardware-Palette vom einfachen Linux-PC über Workstation-Cluster bis hin zu massiven Parallelrechnern lauffähig. Sie ist zur Zeit das numerische „Arbeitspferd“ der DESY-Münster-Kollaboration.

3.1.1 Basiskonzept

Ausgangspunkt bildet die Curci-Veneziano-Wirkung (2.17), wobei die Gleichungen auch sehr einfach auf QCD-artige Wirkungen

$$S_{eff}[U] = S_g[U] - \log\det Q[U] \tag{3.1}$$

mit Wilson-Fermionen übertragen werden können. Allerdings ist $\det Q$ für QCD-Wirkungen nicht notwendigerweise positiv. Lüscher hat ursprünglich den Algorithmus für ein bzgl. des Flavours entartetes Quarks-Paar formuliert, so daß automatisch $\det Q \geq 0$ sichergestellt ist. Im Fall $N_f = 1$ kann diese Bedingung nur für $|K_q| < \frac{1}{8}$ analytisch nachgewiesen werden. Für den hier untersuchten Fall einer N=1 SU(2) Super-Yang-Mills-Theorie gibt es an dieser Stelle keine Probleme, da mit Gleichung (2.15) die Fermion-Determinante immer positiv ist.

Für einen Monte-Carlo-Algorithmus ist die Berücksichtigung der Fermion-Determinante das mit Abstand aufwendigste Problem. Der erste Schritt zur Lösung ist die "Bosonifizierung" dieser Fermion-Determinante durch ein bosonisches Pfadintegral

$$\det(Q^\dagger Q) = \int [d\phi^\dagger d\phi] \exp\left(-\sum_{xy} \phi_y^\dagger \left[Q^\dagger Q\right]^{-1}_{yx} \phi_x\right). \tag{3.2}$$

An dieser Stelle wird gleich das hermitesche Produkt $Q^\dagger Q = \tilde{Q}^2$ betrachtet, da es die im weiteren wichtigen Eigenschaften besitzt, positiv definit und beschränkt zu sein (Zwar ist $\det Q$ positiv, einzelne Eigenwerte können aber durchaus einen negativen Realteil haben.). Natürlich ist der numerische Aufwand, um $[Q^\dagger Q]^{-1}_{yx}$ zu berechnen, enorm, und somit ist dieser Ausdruck innerhalb eines Monte-Carlo-Algorithmus nicht tolerierbar. Eine Möglichkeit ist, die Matrixinversion durch ein geeignetes Polynom P_n vom Grad n zu approximieren

$$\begin{aligned} \det \tilde{Q}^2 = \frac{1}{\det\left(\tilde{Q}^2\right)^{-1}} &\approx \frac{1}{\det P_n(\tilde{Q}^2)} \\ &= \int [d\phi^\dagger d\phi] \exp\left(-\sum_{xy} \phi_y^\dagger P_n(\tilde{Q}^2)_{yx} \phi_x\right). \end{aligned} \tag{3.3}$$

Wenn ϵ und λ der kleinste bzw. größte Eigenwert von $\tilde{Q}^2$ ist, so reicht es aus, wenn das Polynom P_n die Funktion x^{-1} im Intervall $[\epsilon, \lambda]$ hinreichend gut annähert. Die Art der Polynome für die Approximation ist dabei zunächst beliebig, solange

$$\lim_{n\to\infty} P_n(x) = \frac{1}{x} \qquad \forall\, x \in [\epsilon, \lambda] \tag{3.4}$$

gilt. Im Rahmen der N=1 Super-Yang-Mills-Theorie ist man allerdings an einem einzigen Majorana-Fermion interessiert, d. h., man benötigt die Relation für

$$\sqrt{\det Q} = \left[\det(Q^\dagger Q)\right]^{\frac{1}{4}} = \left[\det(\tilde{Q}^2)\right]^{\frac{1}{4}}. \tag{3.5}$$

Somit muß man für diesen Fall die Polynome P_n so wählen, daß gerade

$$\lim_{n\to\infty} P_n(x) = \left[\frac{1}{x}\right]^{\frac{1}{4}} \qquad \forall\, x \in [\epsilon, \lambda] \tag{3.6}$$

approximiert wird. Im allgemeinen Fall von N_f entarteten Quark-Flavour wird man Polynome gemäß der Funktion $x^{-N_f/2}$ wählen. Ein Majorana-Fermion geht wie üblich mit einem halbzahligen Flavour ein, so daß aus dem allgemeinen Fall die Bedingung (3.6) resultiert. In jedem Fall wird man das Polynom durch ein Produkt von Monomen ausdrücken können

$$P_n(x) = c_0 \prod_{i=1}^{n} (x - z_i), \tag{3.7}$$

wobei für gerade n die Wurzeln z_i in komplex konjugierten Paaren auftreten. Schreibt man die Wurzeln als

$$z_k = (\mu_k + i\nu_k)^2 \quad \wedge \quad \bar{z} = (-\mu_k + i\nu_k)^2 \tag{3.8}$$

mit der Konvention $\nu_k > 0$, so kann man das Polynom in seiner offensichtlich positiven Form

$$P_n(x) = c_0 \prod_{k=1}^{n} \left[\left(\sqrt{x} + \mu_k\right)^2 + \nu_k^2 \right] \tag{3.9}$$

schreiben. Der Weg ist nun frei, die gesuchte Fermion-Determinante durch ein *Multi*-Bosonisches Pfadintegral auszudrücken

$$\begin{aligned} \sqrt{\det Q} \quad &= \quad \left[\det(\tilde{Q}^2)\right]^{\frac{1}{4}} \\ &\overset{n\to\infty}{=} \quad \int [d\phi^\dagger d\phi] \exp\left\{ -\sum_{xy} \sum_{j=1}^{n} \phi_x^{(j)\dagger} \left[\left(\tilde{Q} + \mu_j\right)^2_{xy} + \nu_j^2 \delta_{xy} \right] \phi_y^{(j)} \right\}. \end{aligned} \tag{3.10}$$

Die Koeffizienten μ_j und ν_j des Polynoms P_n werden so gewählt, daß die Gleichung (3.6) über das gesamte Eigenwertspektrum von $\tilde{Q}^2$ erfüllt ist. Das gesamte Pfadintegral über die Curci-Veneziano-Wirkung läßt sich damit in einer für Simulationen geeigneten bosonischen Form

$$Z = \int [dU]\, e^{-S_{CV}[U]} \overset{n\to\infty}{=} \int [dU] \int [d\phi^\dagger d\phi] \exp\left\{ -S_{CV}^{(n)}[U, \phi] \right\} \tag{3.11}$$

mit

$$S_{CV}^{(n)}[U, \phi] = \beta \sum_P \left(1 - \frac{1}{N_c} \mathrm{Re}\,\mathrm{Tr}\, U_P \right) + \sum_{xy} \sum_{j=1}^{n} \phi_x^{(j)\dagger} \left[\left(\tilde{Q} + \mu_j\right)^2_{xy} + \nu_j^2 \delta_{xy} \right] \phi_y^{(j)} \tag{3.12}$$

schreiben. Sinngemäß spricht man anstelle von $\phi^{(j)}$ auch von den pseudofermionischen Feldern. Für fermionische Erwartungswerte müssen noch die Majorana-Spinoren Ψ in die bei der Simulation, bedingt durch die Wilson-Fermionen, verwendeten Dirac-Spinoren ψ umgerechnet werden. 2n-Punkt-Erwartungswerte von Dirac-Spinoren können im allgemeinen unter Zuhilfenahme der Gleichung

$$< \psi_{y_1} \bar{\psi}_{x_1} \cdots \psi_{y_n} \bar{\psi}_{x_n} > = \frac{1}{Z} \int [dU] e^{-S_{eff}[U]} \sum_{z_1 \ldots z_n} \epsilon^{z_1 \ldots z_n}_{y_1 \ldots y_n} Q[U]^{-1}_{z_1 x_1} \cdots Q[U]^{-1}_{z_n x_n} \tag{3.13}$$

durch Inversion der Fermion-Matrix berechnet werden [MM94]. Hierbei bezeichnet ϵ den total antisymmetrischen Tensor. Nutzt man die Gleichung (2.3) aus, so kann man daraus die Majorana-wertigen 2n-Punkt-Funktionen mit Hilfe des „doubling tricks“ berechnen, z. B. ergibt sich nach kurzer Rechnung

$$< \Psi_y \bar{\Psi}_x >= \frac{1}{Z} \int [dU]\, e^{-S_{CV}[U]}\, \frac{1}{2} \left\{ Q[U]^{-1}_{yx} + C^{-1} Q[U]^{-1}_{xy} C \right\}. \tag{3.14}$$

Höhere n-Punkt-Funktionen können ebenfalls durch dieses Verfahren bestimmt werden.

3.1.2 Wahl des Approximationspolynoms

Prinzipiell ist die Wahl der Art der Polynome frei. Allerdings sollte man natürlich mit möglichst kurzen Polynomen P_n, also mit kleinem n eine gute Approximation erreichen. Denn zum einen steigt der Aufwand zur Berechnung der bosonischen Wirkung $S^{(n)}_{CV}[U, \phi]$ mit n linear an, zum anderen haben Tests mit Multi-Bosonischen Algorithmen sehr früh gezeigt, daß die Autokorrelationszeit des Algorithmus mit der Anzahl n der verwendeten bosonischen Felder $\phi^{(i)}$ linear wächst [BDFG96]. Dieses Verhalten ist in der subtilen Kopplung der bosonischen Felder untereinander über das Eichfeld begründet.

Lüscher schlug in seiner Originalarbeit Chebyshev-Polynome für die Approximation von $\frac{1}{x}$ vor. Diese sind durch

$$T_0(x) = 1 \quad \wedge \quad T_1(x) = 2x - 1 \tag{3.15}$$

$$xT_r(x) = \frac{1}{4} \left\{ T_{r+1}(x) + T_{r-1}(x) + 2T_r(x) \right\} \ , \quad r \geq 1 \tag{3.16}$$

definiert. Ein mögliches Polynom für die Approximation ist [BJJ95]

$$P_n(x) = \frac{1 + \rho T_{n+1}\left(\frac{x-\epsilon}{1-\epsilon}\right)}{x}, \tag{3.17}$$

wobei die Konstante ρ so gewählt wird, daß der Zähler für $x = 0$ verschwindet. Durch dieses $P_n(x)$ wird x^{-1} im Intervall $[\epsilon, 1]$ mindestens mit der Güte

$$|xP(x) - 1| \leq 2 \left(\frac{1 - \sqrt{\epsilon}}{1 + \sqrt{\epsilon}} \right)^{n+1} \tag{3.18}$$

angenähert. Darüber hinaus gilt $\lim_{n\to\infty} P_n(x) = x^{-1}$ für das gesamte Intervall $]0, 1]$, allerdings nimmt die Konvergenzrate in $]0, \epsilon[$ exponentiell ab. Das Intervall]0,1] ist ausreichend, da man die Fermion-Matrix geeignet reskalieren kann, so daß ihr Spektrum in dieses Intervall fällt. Eine Verallgemeinerung dieses Schemas für die Funktionen $x^{-\alpha}$ mit $\alpha \in \mathbb{R}^+$ führt zu den Gegenbauer-Polynomen [BUN98]. Diese Polynome sind mit $\alpha = \frac{1}{4}$ auch für die Simulation eines einzigen Majorana-Fermions geeignet.

Ein alternativer Weg, um zu Polynomen für beliebiges $\alpha \in \mathbb{R}^+$ innerhalb eines Intervalls $[\epsilon, \lambda]$ zu gelangen, führt über die Minimierung der quadratischen Abweichung [MON98a]

$$\delta = \left[\frac{1}{\lambda - \epsilon} \int_\epsilon^\lambda dx [1 - x^\alpha P_n(x)]^2 \right]^{\frac{1}{2}}. \tag{3.19}$$

Für einen gegebenen Grad n resultieren aus dem Optimierungsprozeß die Koeffizienten $c_{n\nu}(\alpha; \epsilon, \lambda)$ und damit die Wurzeln $r_{nj}(\alpha; \epsilon, \lambda)$ des Polynoms

$$P_n(\alpha; \epsilon, \lambda; x) = \sum_{\nu=0}^{n} c_{n\nu}(\alpha; \epsilon, \lambda)\, x^{n-\nu} = c_0 \prod_{j=1}^{n} [x - r_{nj}(\alpha; \epsilon, \lambda)]. \tag{3.20}$$

Insbesondere können diese analytisch bestimmt werden. Die quadratisch optimierten Polynome erweisen sich im direkten Vergleich mit den Chebyshev-Polynomen für den Spezialfall x^{-1} als vorteilhaft, da bei festem n die Approximationsgüte besser ist [MON98a]. Aus diesem Grund setzt die DESY-Münster-Kollaboration keine Gegenbauer-Polynome, sondern die quadratisch optimierten Polynome ein. Allerdings reicht für größere n die doppelte Genauigkeit der IEEE-Arithmetik nicht mehr zur Bestimmung der Koeffizienten bzw. Wurzeln aus. I. Montvay hat aus diesem Grund für die analytische Bestimmung der Koeffizienten ein Maple-Programm geschrieben, das mit beliebig vorgegebener Genauigkeit rechnen kann[1]. Alternativ zur Koeffizienten- und Produktdarstellung des Polynoms kann es auch im Rahmen eines Rekursionsschemas definiert werden

$$\begin{aligned} P_n(\alpha; \epsilon, \lambda; x) = \sum_{\nu=0}^{n} d_\nu(\alpha; \epsilon, \lambda)\Phi_\nu(x) \quad \text{mit} \quad & \Phi_{r+1}(x) = (x + \beta_r)\Phi_r(x) + \gamma_{r-1}\Phi_{r-1}(x) \\ \wedge \quad & \Phi_0(x) = 1 \ \wedge \ \Phi_1(x) = x + f_{11}. \end{aligned} \tag{3.21}$$

Die Konstanten f_{11}, β_r und γ_{r-1} werden wiederum mit dem Maple-Programm bestimmt. Vorteile des Rekursionsverfahrens sind:

- Es werden keine Wurzeln benötigt, deren Berechnung für $n \geq 400$ teilweise Wochen dauert.
- Die Bestimmung eines Funktionswertes $P_n(x)$ ist numerisch stabil (siehe Abschnitt 5.2.4).
- Da die Koeffizienten $d_\nu(\alpha; \epsilon, \lambda)$ nicht von n abhängen, kann man bei einer beliebigen Ordnung abbrechen und erhält automatisch das optimale Ergebnis für den entsprechende Grad r.

Ein Nachteil ist allerdings, daß man die beiden Werte $\Phi_r(x)$ und $\Phi_{r-1}(x)$ gleichzeitig im Speicher halten muß, um $\Phi_{r+1}(x)$ berechnen zu können. Im Fall $x \sim \tilde{Q}^2$ entspricht dies zwei Fermion-Vektoren, die man zur Berechnung des häufig vorkommenden Ausdrucks $P(\tilde{Q}^2)\psi$ verwalten muß. Dementsprechend benötigt die Rekursionsrelation hierfür ca. $5 - 10\%$ mehr CPU-Zeit als die äquivalente Produktdarstellung.

[1] Das Programm ist im Web erhältlich unter: http://www.desy.de/~montvay/approxima.txt.

In Abbildung (3.1) sind für zwei Polynome, welche in frühen Testrechnungen auf Gittern der Größe $4^3 \times 8$ verwendet wurden, die relativen Abweichungen auf dem Intervall $[\epsilon = 0.002, \lambda = 3.2]$ gegenüber dem Sollwert $x^{-0.25}$ abgetragen. Man sieht das für quadratisch optimierte Polynome typische Verhalten. Über ein breites Intervall ist die Funktion sehr gut angenähert, allerdings wird für kleine Argumente $x = O(\epsilon)$ ein zu kleiner Funktionswert angegeben. Will man auf dem Intervall $[\epsilon_0, \lambda_0]$

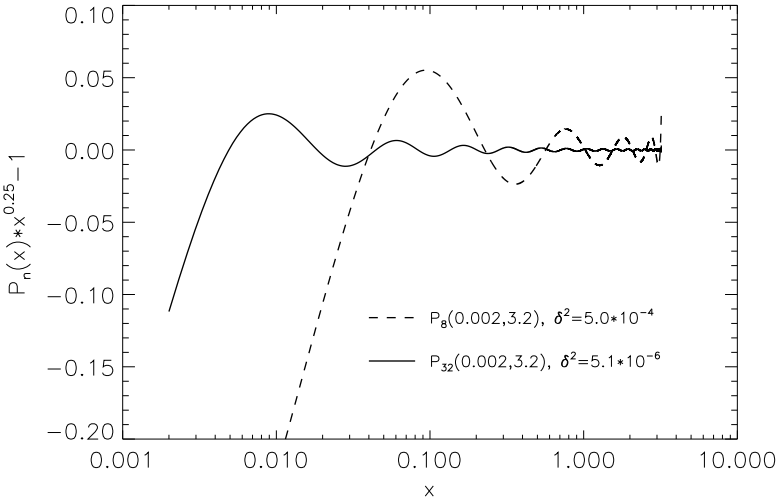


Abbildung 3.1: Relative Abweichung des Polynoms $P_n(\alpha = \frac{1}{4}, \epsilon = 0.002, \lambda = 3.2, x)$ von $x^{-\frac{1}{4}}$ für $n = 8$ bzw. $n = 32$.

die Funktion $x^{-0.25}$ approximieren, so könnte man deshalb in einem naiven Ansatz ϵ sehr viel kleiner als ϵ_0 wählen, um die relativen starken Abweichungen aus dem Approximationsintervall $[\epsilon_0, \lambda_0]$ "herauszuschieben". Allerdings würde man eine solche Strategie mit einem überproportionalen Anstieg der quadratischen Abweichung

$$\delta_n(\alpha, \epsilon, \lambda) \simeq \exp\left\{-C_\alpha n \sqrt{\frac{\epsilon}{\lambda}}\right\} \tag{3.22}$$

über dem gesamten Intervall $[\epsilon_0, \lambda_0]$ erkaufen [MON98a]. Zugleich zeigt das Verhalten (3.22) aber, daß man die Überschwinger der Approximation auf der anderen Seite des Intervalls für $x \simeq \lambda$ sehr wohl aus $[\epsilon_0, \lambda_0]$ herausschieben sollte, indem man λ einige Prozent größer als das gewünschte λ_0 wählt.

Ohne weiteren Korrekturschritt wird man also nicht verhindern können, daß ein Multi-Bosonischer Algorithmus solche Eichkonfigurationen U bevorzugt, die zu kleinen Eigenwerten in $\tilde{Q}[U]$ führen, d. h., diese Eichkonfigurationen werden ein zu großes statistisches Gewicht zugeordnet bekommen. Dies gilt sowohl für quadratisch optimierte Polynome als auch für Chebyshev- bzw. Gegenbauer-Polynome, da letztere ebenfalls für $x \simeq \epsilon$ zu kleine Funktionswerte besitzen. Besonders empfindlich auf diesen systematischen Fehler werden Observablen reagieren, die sensitiv gegenüber den kleinsten Eigenwerten von $\tilde{Q}[U]$ sind, also insbesondere rein fermionische

Erwartungswerte, die nach Gleichung (3.13) direkt über $Q[U]^{-1}$ berechnet werden. Für rein bosonische Erwartungswerte, wie z. B. der Plaquette, wird ein gutmütigeres Verhalten erwartet.

Damit liegen die für Multi-Bosonische Algorithmen typischen Tuning-Probleme vor, denn das Ziel ist natürlich eine möglichst kurze Autokorrelationszeit. Diese kann man am einfachsten durch kurze Polynome erhalten. Erschwerend kommt hinzu, daß das Spektrum von $\tilde{Q}[U]^2$ a priori gar nicht bekannt ist. Im Rahmen einer Monte-Carlo-Teststudie müssen deshalb mit verschiedenen Polynomen ausreichend viele Eichkonfigurationen erzeugt werden, deren kleinste und größte Eigenwerte fortlaufend zur Verfeinerung des Approximationsintervalls $[\epsilon_0, \lambda_0]$ führen. Während der Produktionsläufe muß besonders der kleinste Eigenwert ständig kontrolliert werden, um die Güte der Approximation zu gewährleisten.

3.2 Updater für eine N=1 SU(2) Super-Yang-Mills-Theorie

Auf Grund der analogen Form zu einer QCD-Wirkung wird man auch beim Updater ähnliche Algorithmen verwenden können. Eine Ausnahme bilden die Eichfelder. Ihre Selbstwechselwirkung in der Fundamentaldarstellung und die Kopplung über die adjungierte Darstellung an die pseudofermionischen Felder erlaubt nur lokale Metropolis-Updates.

3.2.1 Pseudofermionische Felder

Ausgangspunkt für die Berechnung der Wechselwirkungsterme des Monte-Carlo-Updaters ist die pseudofermionische Wirkung des lokalen bosonischen Algorithmus in ihrer manifest positiven Formulierung

$$\begin{aligned} S_{CV}^{(n)}[U,\phi] &= S_f^{(n)}[U,\phi] + S_g[U] \\ &= \sum_{xy}\sum_{j=1}^{n} \phi_x^{(j)\dagger}\left[(\tilde{Q}+\mu_j)^2_{xy} + \nu_j^2\delta_{xy}\right]\phi_y^{(j)} + \beta\sum_P\left(1-\frac{1}{N_c}\mathrm{Re}\,\mathrm{Tr}\,U_P\right), \end{aligned} \tag{3.23}$$

wobei die Fermion-Matrizen Q bzw. $\tilde{Q}$ mit der Konvention $\gamma_{-\mu} = -\gamma_\mu$ nach Gleichung (2.9) durch

$$Q_{xy} = \delta_{xy} - K\sum_{\mu=\pm1}^{\pm4}\delta_{x,y+\hat{\mu}}(1+\gamma_\mu)V_{y,\mu} \quad \wedge \quad \tilde{Q} = \gamma_5 Q \tag{3.24}$$

gegeben sind. Im weiteren wird für die Summation über die vier positiven und negativen Raumrichtungen nur noch kurz $\sum_\mu$ ohne Angabe von Grenzen geschrieben. Der erste Schritt sollte das Ausmultiplizieren von $S_f^{(n)}[U,\phi]$ sein, um einen Überblick über die einzelnen Wechselwirkungen zu bekommen. Setzt man dazu die Ausdrücke

$$2\mu_j\tilde{Q}_{xy} = 2\mu_j\delta_{xy} - 2\mu_j K\sum_\mu \delta_{x,y+\hat{\mu}}(\gamma_5+\gamma_5\gamma_\mu)V_{y,\mu}$$

$$
\begin{aligned}
(\tilde{Q}^2)_{xy} &= \delta_{xy} - 2K \sum_{\mu} \delta_{x,y+\hat{\mu}} V_{y,\mu} \\
&\quad + K^2 \sum_{\mu\nu} \delta_{x,y-\hat{\mu}-\hat{\nu}} (1 + \gamma_\mu - \gamma_\nu - \gamma_\mu\gamma_\nu) V_{x+\hat{\mu},-\mu} V_{y,-\nu}
\end{aligned} \tag{3.25}
$$

in den pseudofermionischen Teil der Wirkung

$$
S_f^{(n)}[U,\phi] = \sum_{xy} \sum_{j=1}^{n} \phi_x^{(j)\dagger} \left[(\tilde{Q}^2 + 2\tilde{Q}\mu_j)_{xy} + (\nu_j^2 + \mu_j^2)\delta_{xy} \right] \phi_y^{(j)} \tag{3.26}
$$

ein, so erhält man

$$
\begin{aligned}
S_f^{(n)}[U,\phi] = \sum_{x} \sum_{j=1}^{n} \Bigg\{ & \phi_x^{(j)\dagger} \left[\mu_j^2 + \nu_j^2 + 2\mu_j\gamma_5 + 1 \right] \phi_x^{(j)} \\
& + \phi_x^{(j)\dagger} \left[(-2K) \sum_{\mu} \left(\mu_j(\gamma_5 + \gamma_5\gamma_\mu) + 1 \right) V_{x-\hat{\mu},\mu} \phi_{x-\hat{\mu}}^{(j)} \right] \\
& + \phi_x^{(j)\dagger} \left[K^2 \sum_{\mu\nu} (1 + \gamma_\mu - \gamma_\nu - \gamma_\mu\gamma_\nu) V_{x+\hat{\mu},-\mu} V_{x+\hat{\mu}+\hat{\nu},-\nu} \phi_{x+\hat{\mu}+\hat{\nu}}^{(j)} \right] \Bigg\}.
\end{aligned} \tag{3.27}
$$

Eine Klassifizierung der Wechselwirkungen nach ihrer "Reichweite" liefert:

- lokaler Term:

$$
A^{(j)} = \mu_j^2 + \nu_j^2 + 2\mu_j\gamma_5 + 16K^2 + 1, \tag{3.28}
$$

- nächste Nachbarn:

$$
\begin{aligned}
B_x^{(j)} &= -2K \sum_{\mu} \left(1 + \mu_j(\gamma_5 + \gamma_5\gamma_\mu) \right) V_{x-\hat{\mu},\mu} \phi_{x-\hat{\mu}}^{(j)} \\
&= -2K \sum_{\mu} \left(1 + \mu_j(\gamma_5 - \gamma_5\gamma_\mu) \right) V_{x,\mu}^T \phi_{x+\hat{\mu}}^{(j)},
\end{aligned} \tag{3.29}
$$

- übernächste Nachbarn:

$$
\begin{aligned}
C_x^{(j)} &= K^2 \sum_{\substack{\mu,\nu=\pm 1 \\ \mu \neq \pm\nu}}^{\pm 4} (1 + \gamma_\mu - \gamma_\nu - \gamma_\mu\gamma_\nu) V_{x+\hat{\mu},-\mu} V_{x+\hat{\mu}+\hat{\nu},-\nu} \phi_{x+\hat{\mu}+\hat{\nu}}^{(j)} \\
&= K^2 \sum_{\substack{\mu,\nu=\pm 1 \\ \mu \neq \pm\nu}}^{\pm 4} (1 + \gamma_\mu - \gamma_\nu - \gamma_\mu\gamma_\nu) V_{x,\mu}^T V_{x+\hat{\mu},\nu}^T \phi_{x+\hat{\mu}+\hat{\nu}}^{(j)}.
\end{aligned} \tag{3.30}
$$

Eine kompaktere Schreibweise für den pseudofermionischen Anteil ist somit

$$
S_f^{(n)}[U,\phi] = \sum_{x} \sum_{j=1}^{n} \left[\phi_x^{(j)\dagger} A^{(j)} \phi_x^{(j)} + \phi_x^{(j)\dagger} (B_x^{(j)} + C_x^{(j)}) \right]. \tag{3.31}
$$

Dies ist genau dieselbe Form, auf die man auch QCD-Wirkungen bringt, um damit den Heatbath- bzw. den Overrelaxations-Algorithmus definieren zu können, z. B. [BDFG96].

Overrelaxation: Schreibt man die Wirkung als Bilinearform

$$S_f^{(n)} = \sum_x \sum_{j=1}^{n} \Bigg\{ \left(\phi_x^{(j)\dagger} + V_x^{(j)\dagger} A^{(j)^{-1}}\right) A^{(j)} \left(\phi_x^{(j)} + A^{(j)^{-1}} V_x^{(j)}\right) \tag{3.32}$$
$$+ \text{...Terme unabhängig von } \phi_x \Bigg\} \text{ mit } V_x^{(j)} = \left(B_x^{(j)} + C_x^{(j)}\right),$$

so ergibt sich der Overrelaxations-Algorithmus durch das mikrokanonische Update

$$\phi_x^{(j)} \leftarrow -\phi_x^{(i)} - 2A^{(j)^{-1}} V_x^{(j)}. \tag{3.33}$$

Heatbath: Das Heatbath-Update folgt ebenfalls aus der Schreibweise (3.32). Dazu generiert man einen Gauß'schen Zufallsspinor χ mit Varianz eins und führt das lokale Update durch

$$\phi_x^{(j)} \leftarrow \left(A^{(j)}\right)^{-\frac{1}{2}} \chi - A^{(j)^{-1}} V_x^{(j)}. \tag{3.34}$$

Ein guter Algorithmus entsteht aus nacheinander geschalteter Anwendung beider Updater. Das Mischungsverhältnis muß durch Testläufe und Messen der Korrelationszeiten optimiert werden. Typische Werte sind

$$\text{Heatbath : Overrelaxation} \sim \text{ 1:3 - 1:6.}$$

3.2.2 Eichfelder

Umfangreicher gestalten sich die Wechselwirkungen eines Links mit den Skalarfeldern in $S_f^{(n)}[U,\phi]$. Man betrachte dazu im ersten Schritt die Wechselwirkungen $S_{f,N}^{(n)}[U,\phi]$ für die nächsten Nachbarn

$$\begin{aligned} S_{f,N}^{(n)}[U,\phi] &= \sum_x \sum_{j=1}^{n} \phi_x^{(j)\dagger} B_x^{(j)} \\ &= -2K \sum_x \sum_{j=1}^{n} \sum_{\mu=1}^{4} \Big[\phi_x^{(j)\dagger} V_{x,\mu}^T \left(1 + \mu_j(\gamma_5 - \gamma_5\gamma_\mu)\right) \phi_{x+\hat{\mu}}^{(j)} \\ &\qquad + \phi_x^{(j)\dagger} V_{x-\hat{\mu},\mu} \left(1 + \mu_j(\gamma_5 + \gamma_5\gamma_\mu)\right) \phi_{x-\hat{\mu}}^{(j)} \Big]. \end{aligned}$$

Die Wechselwirkung eines Links $V_{x,\mu}$ mit den Skalarfeldern $\phi_x^{(j)}$ und $\phi_{x+\hat{\mu}}^{(j)}$, die der Link verbindet, ergibt sich als

$$\begin{aligned} \Sigma_{SV,x\mu}^{(j)}[V_{x,\mu}] &= -2K \Big[\phi_x^{(j)\dagger} V_{x,\mu}^T \left(1 + \mu_j(\gamma_5 - \gamma_5\gamma_\mu)\right) \phi_{x+\hat{\mu}}^{(j)} \\ &\qquad + \phi_{x+\hat{\mu}}^{(j)\,\dagger} \left(1 + \mu_j(\gamma_5 + \gamma_5\gamma_\mu)\right) V_{x,\mu} \phi_x^{(j)} \Big] \\ &= -4K \mathrm{Re} \left[\phi_x^{(j)\dagger} V_{x,\mu}^T \left(1 + \mu_j(\gamma_5 - \gamma_5\gamma_\mu)\right) \phi_{x+\hat{\mu}}^{(j)} \right]. \end{aligned} \tag{3.35}$$

Im zweitem Schritt müssen die übernächsten Nachbarwechselwirkungen $S^{(n)}_{f,NN}[U,\phi]$ berücksichtigt werden

$$\begin{aligned} S^{(n)}_{f,NN}[U,\phi] &= \sum_y \sum_{j=1}^{n} \phi^{(j)\dagger}_y C^{(j)}_y \\ &= K^2 \sum_y \sum_{j=1}^{n} \phi^{(j)\dagger}_y \sum_{\substack{\tau,\nu=\pm 1 \\ \tau\neq\nu}}^{\pm 4} (1+\gamma_\tau-\gamma_\nu-\gamma_\tau\gamma_\nu) V^T_{y,\tau} V^T_{y+\hat\tau,\nu} \phi^{(j)}_{y+\hat\tau+\hat\nu} \\ &= K^2 \sum_y \sum_{j=1}^{n} \sum_{\substack{\tau,\nu=\pm 1 \\ \tau\neq\nu}}^{\pm 4} \phi^{(j)\,\dagger}_{y+\hat\tau} (1-\gamma_\tau-\gamma_\nu+\gamma_\tau\gamma_\nu) V_{y,\tau} V^T_{y,\nu} \phi^{(j)}_{y+\hat\nu}. \end{aligned} \tag{3.36}$$

Hieraus folgen für einen bestimmten Link $V_{x,\mu}$ vier Wechselwirkungsbeiträge

$$\begin{aligned} y = x \quad &\wedge \quad \tau = \mu \\ y = x \quad &\wedge \quad \nu = \mu \\ y = x+\hat\mu \quad &\wedge \quad \tau = -\mu \\ y = x+\hat\mu \quad &\wedge \quad \nu = -\mu, \end{aligned} \tag{3.37}$$

welche die nächsten Nachbarwechselwirkungen festlegen

$$\begin{aligned} \Xi^{(j)}_{SV,x\mu}[V_{x,\mu}] &= K^2 \sum_{\nu\neq\pm\mu} \Big[\quad \phi^{(j)\,\dagger}_{x+\hat\mu} (1-\gamma_\mu-\gamma_\nu+\gamma_\mu\gamma_\nu) V_{x,\mu} V^T_{x,\nu} \phi^{(j)}_{x+\hat\nu} \\ &\qquad + \phi^{(j)\dagger}_x (1+\gamma_\mu-\gamma_\nu-\gamma_\mu\gamma_\nu) V^T_{x,\mu} V^T_{x+\hat\mu,\nu} \phi^{(j)}_{x+\hat\mu+\hat\nu} \Big] \\ &\quad + K^2 \sum_{\tau\neq\pm\mu} \Big[\quad \phi^{(j)\,\dagger}_{x+\hat\tau} (1-\gamma_\tau-\gamma_\mu+\gamma_\tau\gamma_\mu) V_{x,\tau} V^T_{x,\mu} \phi^{(j)}_{x+\hat\mu} \\ &\qquad + \phi^{(j)\,\dagger}_{x+\hat\tau+\hat\mu} (1-\gamma_\tau+\gamma_\mu-\gamma_\tau\gamma_\mu) V_{x+\hat\mu,\tau} V_{x,\mu} \phi^{(j)}_{x+\hat\mu+\hat\nu} \Big] \\ &= 2K^2 \sum_{\nu\neq\pm\mu} \mathrm{Re} \Big[\quad \phi^{(j)\,\dagger}_{x+\hat\mu} (1-\gamma_\mu-\gamma_\nu+\gamma_\mu\gamma_\nu) V_{x,\mu} V^T_{x,\nu} \phi^{(j)}_{x+\hat\nu} \\ &\qquad + \phi^{(j)\dagger}_x (1+\gamma_\mu-\gamma_\nu-\gamma_\mu\gamma_\nu) V^T_{x,\mu} V^T_{x+\hat\mu,\nu} \phi^{(j)}_{x+\hat\mu+\hat\nu} \Big]. \end{aligned} \tag{3.38}$$

Es fehlt noch die Selbstwechselwirkung der Eichfelder durch die Wilson-Eichwirkung. Der daraus resultierende Wechselwirkungsterm, die sogenannte „Klammer„-Summe, ist mit

$$\Upsilon_{x\mu} = \frac{\beta}{N_c} \sum_{\substack{\nu=1 \\ \nu\neq\mu}}^{4} \left\{ U^\dagger_{x,\nu} U^\dagger_{x+\hat\nu,\mu} U_{x+\hat\mu,\nu} + U_{x-\hat\nu,\nu} U^\dagger_{x-\hat\nu,\mu} U^\dagger_{x-\hat\nu+\hat\mu,\nu} \right\} \tag{3.39}$$

in Abbildung (2.1) skizziert. Damit sind alle Wechselwirkungsterme klassifiziert, und man kann die Energiedifferenz für einen Testschritt $U_{x\mu} \leftarrow U'_{x\mu}$, $V_{x\mu} \leftarrow \mathrm{Adj}(U'_{x\mu})$ bestimmen

$$\begin{aligned} \delta S &= \mathrm{Tr}\left[\left(U_{x\mu} - U'_{x\mu} \right) \Upsilon_{x\mu} \right] \\ &\quad + \sum_{j=1}^{n} \left(\Xi^{(j)}_{SV x\mu}[V'_{x\mu}] + \Sigma^{(j)}_{SV x\mu}[V'_{x\mu}] - \Xi^{(j)}_{SV x\mu}[V_{x\mu}] - \Sigma^{(j)}_{SV x\mu}[V_{x\mu}] \right). \end{aligned} \tag{3.40}$$

Zu einer effizienten Implementierung des Multihit-Metropolis-Algorithmus gelangt man durch die Beobachtung, daß man $V_{x\mu}$ in den Wechselwirkungstermen $\Xi^{(j)}_{SV,x\mu}[V_{x\mu}]$ und $\Sigma^{(j)}_{SV,x\mu}[V_{x\mu}]$ ausklammern kann, d. h., man kann diese Terme schreiben als

$$\sum_{j=1}^{n} \Xi^{(j)}_{SV,x\mu}[V_{x\mu}] = \mathrm{Tr}\left(V_{x\mu}\Xi_{SVx\mu}\right) \quad \wedge \quad \sum_{j=1}^{n} \Sigma^{(j)}_{SV,x\mu}[V_{x\mu}] = \mathrm{Tr}\left(V_{x\mu}\Sigma_{SVx\mu}\right). \tag{3.41}$$

Entscheidend ist dabei, daß die Wechselwirkungen $\Xi_{SVx\mu}$ und $\Sigma_{SV,x\mu}$ nicht mehr von $V_{x\mu}$ abhängen, so daß die Energiedifferenz für einen Testschritt sehr viel einfacher durch

$$\delta S = \mathrm{Tr}\left[\left(U_{x,\mu} - U'_{x,\mu}\right)\Upsilon_{x\mu}\right] + \mathrm{Tr}\left[\left(V'_{x\mu} - V_{x,\mu}\right)\left(\Xi_{SVx\mu} + \Sigma_{SV,x\mu}\right)\right] \tag{3.42}$$

berechnet werden kann. Im Rahmen eines N-Hit Metropolis-Updates für einen Link $U_{x\mu}$ berechnet man somit zuerst die Terme $\Upsilon_{x\mu}$, $\Xi_{SVx\mu}$ und $\Sigma_{SV,x\mu}$, um dann N Metropolis-Schritte mit Wechselwirkungsenergie (3.42) zu machen. Die Erfahrungen für das SU(2) Super-Yang-Mills-Modell legen nahe, N so groß zu wählen, daß im Mittel 5 Schritte angenommen werden; dann entspricht das N-Hit Metropolis-Update einem effektiven Heatbath-Update. Wählt man N noch größer, so verbraucht man zwar mehr Rechenzeit für das Update, die Eichkonfigurationen werden dadurch aber nicht mehr stärker dekorreliert.

Ein optimales Update erreicht man, indem sich ein Eichfeld-Sweep und ein Skalarfeld-Sweep abwechseln, so daß ein kompletter Sweep aus folgenden Komponenten besteht

1. Skalar-Heatbath $\rightarrow$ (3-6). Skalar-Overrelaxation $\rightarrow$ 1. N-Hit Metropolis

3.2.3 Skalare Hilfsfelder

Die Wechselwirkungsterme (3.39) und (3.30) vermitteln übernächste Nachbarwechselwirkungen, die in dieser Form umständlich zu berechnen sind und deren Implementierung auf einem Parallelrechner unnötige Mühe bereitet. Aus diesen Gründen führt man in gleicher Weise wie von QCD-Implementierungen bekannt, skalare Hilfsfelder ein. Dazu definiert man über die Hopping-Matrix M

$$\tilde{Q}_{xy} = \gamma_5\delta_{xy} - K\tilde{M}_{xy} \text{ mit} \quad \tilde{M}_{xy} = \sum_{\mu} \delta_{x+\hat{\mu},y}(\gamma_5 - \gamma_5\gamma_\mu)V^T_{x,\mu} \tag{3.43}$$

die skalaren Hilfsfelder $h^{(j)}$ und $\theta^{(j)}$ durch

$$h^{(j)}_x = \sum_z \tilde{M}_{xz}\phi^{(j)}_z \quad \wedge \quad \theta^{(j)}_x = \sum_z \tilde{M}_{xz}\gamma_5\phi^{(z)}_z. \tag{3.44}$$

Mit diesen Größen kann der pseudofermionische Teil (3.28) der Wirkung kompakter geschrieben werden

$$\begin{aligned} S^{(n)}_f[U,\phi] \;=\; & \sum_{xy}\sum_{j=1}^{n} \phi^{(j)\dagger}_x \Big\{ \quad \left(\mu_j^2 + \nu_j^2 + 2\mu_j\gamma_5 + 1\right)\delta_{xy}\phi^{(j)}_y \\ & -K\left(2\mu_j h^{(j)}_x + \gamma_5 h^{(j)}_x + \theta^{(j)}_x\right) + K^2\sum_z \tilde{M}_{xz}h^{(j)}_z \Big\}. \end{aligned} \tag{3.45}$$

Insbesondere enthält die Wirkung in dieser Form keinerlei übernächsten, sondern über $\tilde{M}_{xz} h_z^{(j)}$ nur noch nächste Nachbarwechselwirkungen. Damit kann man die im vorhergehenden Abschnitt klassifizierten Wechselwirkungen einfacher formulieren:

Skalar-Updater

- lokaler Term:
$$A^{(j)} = \mu_j^2 + \nu_j^2 + 2\mu_j\gamma_5 + 16K^2 + 1 \tag{3.46}$$
- nächste Nachbarn $\rightarrow$ lokaler Term + nächste Nachbarn:
$$B_x^{(j)} = -2K\left(\mu_j h_x^{(j)} + \sum_\mu V_{x\mu}^T \phi_{x+\hat{\mu}}^{(j)}\right) \tag{3.47}$$
- übernächste Nachbarn $\rightarrow$ lokaler Term + nächste Nachbarn:
$$C_x^{(j)} = -16K^2\phi_x^{(j)} + \sum_\mu (\gamma_5 - \gamma_5\gamma_\mu) V_{x\mu}^T h_{x+\hat{\mu}}^{(j)} \tag{3.48}$$

Eichfeld-Update

- übernächste Nachbarn $\rightarrow$ lokaler Term + nächste Nachbarn:
$$\begin{aligned}(\Xi_{SV,x\mu})_{r_2r_1} = 2K^2\mathrm{Re}\sum_{j=1}^{n}\Big\{ & \quad \phi_{x,r_2}^{(j)\,\dagger}\,(\gamma_5 - \gamma_5\gamma_\mu)\,h_{x+\hat{\mu},r_1}^{(j)} \\ & + \phi_{x+\hat{\mu},r_1}^{(j)\,\dagger}(\gamma_5 + \gamma_5\gamma_\mu)\,h_{x,r_2}^{(j)} \\ & - 2\phi_{x,r_2}^{(j)\,\dagger}(1+\gamma_\mu)\sum_r V_{x\mu,rr_1}^T \phi_{x,r}^{(j)} \\ & - 2\phi_{x+\hat{\mu},r_1}^{(j)\,\dagger}(1-\gamma_\mu)\sum_r V_{x\mu,r_2r}^T \phi_{x+\hat{\mu},r}^{(j)}\Big\}. \end{aligned} \tag{3.49}$$

Diese Wechselwirkungsterme können viel schneller ausgerechnet werden als die äquivalenten Gleichungen ohne Hilfsfelder, da sie im Schnitt mit einer Summation über die Raumrichtungen weniger auskommen. Allerdings muß bei jeder Änderung der Skalar- oder Eichfelder das Hilfsfeld in der Nachbarschaft entsprechend nachgeführt werden. Trotzdem ist die Implementierung mit Hilfsfeldern deutlich effizienter als ohne Hilfsfelder.

Auf einem Distributed-Memory-Parallelrechner wie der T3E bereitet das "Nachführen" der Hilfsfelder besondere Probleme. Das vierdimensionale physikalische Gitter wird bei diesem Rechner auf ein dreidimensionales Prozessor-Gitter verteilt (domain decomposition). Jeder Prozessor ist nur für seinen Anteil verantwortlich und hält auch nur seine Gitterplätze samt Feldern im Speicher. Die Prozessoren können untereinander lediglich mittels einer "Message Passing Software" Nachrichten austauschen. Sie können nicht aus dem Speicher eines anderen Prozessors lesen oder etwas hineinschreiben. Führt ein Prozessor das Update an einer Ecke seines Gebietes durch, so muß er den drei benachbarten Prozessoren die neuen Felddaten schicken

und diese dazu *veranlassen*, ihre Hilfsfelder entsprechend neu zu berechnen. Während dieses Vorgangs sollte der erste Prozessor aus Performance-Gründen natürlich an einer anderen Stelle, unter Wahrung der Detailed Balance, ein Update durchführen. Noch im Laufe dieses Updates werden die Nachbarprozessoren die neuen Werte für die Hilfsfelder zurückschicken und sich wieder ihren eigenen Updates zuwenden. Jeder Schritt dieses Vorgangs muß zu Minimierung der Latenzzeiten asynchron, d. h. ohne Handshake, erfolgen. Ecken stellen einen besonders kniffeligen Spezialfall dar.

Punkte aus Kanten bzw. Oberflächen des lokalen Gitters müssen nur an zwei Nachbarprozessoren bzw. einen Nachbarprozessor verschickt werden. Kanten und Oberflächen werden aus Effizienzgründen nur als Ganzes berechnet und danach verschickt. Der Umfang der zu verschickenden Nutzdaten verringert sich dadurch zwar nicht, allerdings reduziert sich die Anzahl der Nachrichten und der damit verbundene Overhead enorm.

Damit ist die Implementierung der Hilfsfelder deutlich anspruchsvoller als z. B. eine Fermion-Matrix Multiplikation, denn bei dieser müssen nur zu Beginn die Felder an den Rändern ausgetauscht werden, während die eigentliche Berechnung dann ohne Kommunikation erfolgen kann. Es überrascht somit nicht, daß mehr als 50% des Message Passing Codes auf die Hilfsfelder entfällt.

3.2.4 Randbedingungen

Diskretisiert man ein fermionisches Pfadintegral in der euklidischen Raum-Zeit durch ein endliches Gitter, so sollte man für die fermionischen Felder ψ_x periodische Randbedingungen in den Raumrichtungen und antiperiodische Randbedingungen in der Zeitrichtung fordern [MM94].

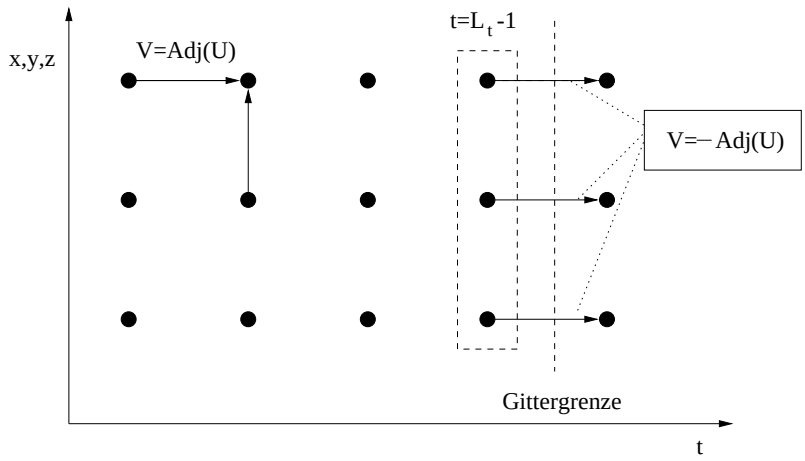


Abbildung 3.2: Implementierung der antiperiodischen Randbedingungen in Zeitrichtung.

Solange die zu den leichtesten fermionischen Bindungszuständen m_f gehörende Kor-

relationslänge

$$\xi_t \sim \frac{1}{m_f} \tag{3.50}$$

klein gegenüber der zeitlichen Ausdehnung des Gitters ist, wird man vernachlässigbare Effekte durch die Wahl der Randbedingungen erwarten. Allerdings brechen die verschiedenen Randbedingungen für das Eichfeld und das Fermionfeld die Supersymmetrie. Das spricht für periodische Randbedingungen in Zeitrichtung für das Fermionfeld. Antiperiodische Randbedingungen sind für die Curci-Veneziano-Wirkung nur innerhalb der Hopping-Matrix

$$\tilde{M}_{xy} = \sum_{\mu} \delta_{x+\hat{\mu},y}(\gamma_5 - \gamma_5\gamma_\mu)V^T_{x,\mu} \tag{3.51}$$

von Relevanz. Man gelangt zu einer effizienten Implementierung, indem man ausnutzt, daß die Information über die Eichfelder redundant in der fundamentalen und in der adjungierten Darstellung gespeichert ist [COL]. Setzt man die Links der letzten Zeitscheibe $T = L_t - 1$ in positiver Zeitrichtung auf

$$V_{x,\mu=t} = -Adj(U_{x,\mu=t}), \tag{3.52}$$

so sind die antiperiodischen Randbedingungen automatisch impliziert, ohne daß die Eichselbstwechselwirkung in der fundamentalen Darstellung davon betroffen ist. Insbesondere entfallen damit lästige if-then-Abfragen innerhalb des Programms bzgl. der Position im Gitter, die sich negativ auf die Performance auswirken würden.

Kapitel 4

Präkonditionierung

4.1 Even-Odd-Zerlegung

Die Länge des benötigten Polynoms und damit die Autokorrelationszeit des Multi-Bosonischen Algorithmus wird nach Gleichung (3.3) entscheidend von der durch den größten und kleinsten Eigenwert bestimmten Konditionszahl

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}} \tag{4.1}$$

der Matrix $\tilde{Q}^2$ beeinflußt. Entscheidender Trick ist deshalb, eine Matrix $\tilde{\tilde{Q}}$ zu finden, welche denselben Wert für die Determinante besitzt

$$\sqrt{\det Q} = \left[\det(\tilde{Q}^2)\right]^{\frac{1}{4}} = \left[\det(\tilde{\tilde{Q}}^2)\right]^{\frac{1}{4}}, \tag{4.2}$$

deren Konditionszahl allerdings kleiner ist. Durch die QCD-ähnliche Struktur der Wirkung bietet sich ebenfalls die sogenannte Even-Odd-Präkonditionierung an, für die man das Gitter in Even- und Odd-Sites zerlegt. Dabei nutzt man aus, daß die Fermion-Matrix nur Selbstkopplung und nächste Nachbarwechselwirkungen vermittelt. Schreibt man dies in Even-Odd-Blockform

$$\psi = \begin{pmatrix} \psi_{even} \\ \psi_{odd} \end{pmatrix} \quad \rightsquigarrow \quad \tilde{Q} = \begin{pmatrix} \gamma_5 & -\gamma_5 K M_{even-odd} \\ -\gamma_5 K M_{odd-even} & \gamma_5 \end{pmatrix}, \tag{4.3}$$

wobei $\psi_{even(odd)}$ dem Feld auf den Even(Odd)-Sites entspricht und wendet die Identität

$$\det \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \det A \det \left(D - C A^{-1} B\right) \tag{4.4}$$

an, so erhält man bereits die gesuchte Relation

$$\det \tilde{Q} = \det\gamma_5 \det \left(\gamma_5 - \gamma_5 K^2 M_{oe} M_{eo}\right) = \det \left(\gamma_5 - \gamma_5 K^2 M_{oe} M_{eo}\right), \tag{4.5}$$

da $\det\gamma_5 = 1$ ist. Die geeigneten Even-Odd-präkonditionierten, hermiteschen Matrizen sind somit durch

$$\tilde{\tilde{M}} = \gamma_5 - \gamma_5 K^2 M_{oe} M_{eo} \quad \wedge \quad \tilde{\tilde{Q}} = \begin{pmatrix} \gamma_5 & 0 \\ 0 & \tilde{\tilde{M}} \end{pmatrix} \tag{4.6}$$

definiert. Sei $v = (v_e, v_o)$ mit $v_o \neq 0$ ein Eigenvektor von Q

$$\begin{pmatrix} 1 & -KM_{eo} \\ -KM_{oe} & 1 \end{pmatrix} \begin{pmatrix} v_e \\ v_o \end{pmatrix} = \lambda \begin{pmatrix} v_e \\ v_o \end{pmatrix} \quad \Rightarrow \quad \begin{array}{l} KM_{eo}v_o = (1-\lambda)v_e \\ KM_{oe}v_e = (1-\lambda)v_o \end{array} . \tag{4.7}$$

Wendet man $\bar{M} = \gamma_5 \tilde{\tilde{M}}$ auf v_o an

$$\begin{aligned} \bar{M}v_o &= (1 - K^2 M_{oe}M_{eo})v_o = v_o - KM_{oe}(1-\lambda)v_e \\ &= v_o - (1-\lambda)^2 v_o = 2\lambda v_o - \lambda^2 v_o \\ &= \lambda(2-\lambda)v_o, \end{aligned} \tag{4.8}$$

so erkennt man, daß v_o ein Eigenvektor von $\bar{M}$ ist. Die Eigenwerte von $\bar{Q} = \gamma_5 \tilde{\tilde{Q}}$ sind demnach mit denen von Q über die Gleichung

$$\bar{\lambda} = 2\lambda - \lambda^2 \tag{4.9}$$

verknüpft, d. h., der zu einem kleinen (reellen) Eigenwert von Q gehörende Eigenwert von $\bar{M}$ ist ungefähr doppelt so groß. Diese Relation gilt in erster Ordnung in λ auch zwischen $\tilde{\tilde{Q}}$ und $\tilde{Q}$, so daß $\tilde{Q}^2$ für Polynomapproximationen von $x^{-0.25}$ besser geeignet sein wird als $\tilde{Q}^2$.

4.2 Konditionszahl der Fermion-Matrix

Um die quantitativen Aussagen mit numerischen Daten zu bestätigen, benötigt man ein effizientes Verfahren zur Bestimmung des kleinsten bzw. des größten Eigenwerts einer Fermion-Matrix, da vor allem der kleinste Eigenwert während eines Produktionslaufes permanent kontrolliert werden muß, um die Güte der Polynomapproximation zu gewährleisten. In ersten Programmversionen wurde das einfache Gradientenverfahren benutzt. Allerdings erwies sich der Algorithmus für zunehmend größer werdende Konditionszahlen als langsam und unzuverlässig. Deshalb wird nun ein konjugiertes Gradientenverfahren eingesetzt.

4.2.1 Konjugierte Gradientenverfahren

Konjugierte Gradientenverfahren gehören zur Familie der Minimierungsalgorithmen [PTVF94]. Die Suche nach dem kleinsten (größten) Eigenwert einer hermiteschen Matrix A kann durch das Ritz-Funktional

$$\mu_A(z) = \frac{< z, Az >}{< z, z >} \tag{4.10}$$

als folgendes Minimierungsproblem formuliert werden

$$\lambda_{min}(A) = \min_z \Big(\mu_A(z)\Big) \quad \wedge \quad \lambda_{max}(A) = \min_z \Big(\mu_{A^{-1}}(z)\Big). \tag{4.11}$$

Der Gradient des Ritz-Funktionals ist schnell berechnet

$$g(z) = \frac{1}{< z, z >} \left[A - \mu_A(z)\right] z. \tag{4.12}$$

Damit sind die Ingredienzen für einen konjugierten Gradientenalgorithmus gegeben. Ausgehend von einem Startvektor z_1 und der Start-/Suchrichtung $p_1 = -g(z_1)$ ist der Algorithmus durch Iteration des folgenden Schemas definiert:

- neuer Ortsvektor:

$$z_{i+1} = z_i + \alpha_i p_i \quad \text{mit } \alpha_i \text{ derart, daß } \mu_A(z_{i+1}) \text{ minimal ist;} \tag{4.13}$$

- konjugierte Suchrichtung:

$$p_{i+1} = g(z_{i+1}) + \beta_i \left(p_i - z_{i+1} \frac{< z_{i+1}, p_i >}{< z_{i+1}, z_{i+1} >} \right) \tag{4.14}$$

Die Iteration wird im allgemeinen abgebrochen, wenn die relative Änderung des Ritz-Funktionals einen Schwellwert unterschreitet. Der Term proportional zu z_{i+1} in der Definition (4.14) gehört nicht zum Standardalgorithmus. Er wird zusammen mit dem Konvergenzbeweis in [KS96] vorgeschlagen, um die durch die Skaleninvarianz des Ritz-Funktionals $\mu_A(\lambda z) = \mu_A(z)$ nahegelegte Orthogonalitätsrelation

$$< z_i, p_i >= 0 \quad \forall\, i \geq 1 \tag{4.15}$$

zu erfüllen. Die Wahl für β_i ist nicht eindeutig festgelegt, u. a. möglich sind:

Fletcher-Reeves:

$$\beta_i = \frac{< g_{i+1}, g_{i+1} >}{< g_i, g_i >}. \tag{4.16}$$

Diese Relation wird oftmals impliziert, wenn nur allgemein von einem konjugierten Gradientenverfahren gesprochen wird.

Polak-Ribiere:[PTVF94]

$$\beta_i = \frac{< g(z_{i+1}) - g(z_i), g(z_{i+1}) >}{< g_i, g_i >}. \tag{4.17}$$

Ad hoc ist nicht klar, welche der beiden Relationen zu einer höheren Konvergenzgeschwindigkeit des Algorithmus führt. Für die in diesem Rahmen untersuchten Fermion-Matrizen hat sich die Polak-Ribiere-Version als 25%-40% effizienter erwiesen.

Erfreulicherweise läßt sich das Minimierungsproblem aus Gleichung (4.13) zur Berechnung von α_i analytisch lösen. Eine längere Rechnung ergibt als notwendige Bedingung für ein Extremum

$$\alpha_i^{(1,2)} = \frac{-st^2 \pm \sqrt{s^2 t^4 + 4t^2 \left(u - v\mu_A(p_i)\right)\left(u - v\mu_A(z_i)\right)}}{2t^2 \left(u - v\mu_A(p_i)\right)} \tag{4.18}$$

$$\text{mit} \qquad v = \frac{< p_i, z_i >}{< z_i, z_i >} \quad \wedge \quad t^2 = \frac{< p_i, p_i >}{< z_i, z_i >}$$

$$\wedge \quad u = \frac{< p_i, Az_i >}{< z_i, z_i >} \quad \wedge \quad s = \mu_A(x_i) - \mu_A(p_i).$$

Das Minuszeichen dieser notwendigen Bedingungen gehört zum Minimum und das Pluszeichen zum Maximum des Ritz-Funktionals. Mit letzterem Vorzeichen kann dementsprechend auch der größte Eigenwert ohne die in Gleichung (4.11) angedeutete Kenntnis von A^{-1} gefunden werden.

Bei der naiven Implementierung des Algorithmus kann es zu numerischen Problemen kommen, da die Länge von z_i nach den Gleichungen (4.15) und (4.13) mit jeder Iteration größer wird. Da das Ritz-Funktional skaleninvariant ist, kann man dieses durch laufende Reskalierung

$$z_i \to \frac{z_i}{\|z_i\|_2} , \quad p_i \to p_i \|z_i\|_2 , \quad g_i \to g_i \|z_i\|_2 , \tag{4.19}$$

(typischerweise alle 25 Schritte) umgehen.

4.2.2 Verteilung des größten und des kleinsten Eigenwertes

Die numerische Studie der größten Eigenwerte von $\tilde{Q}^2[U]$ und $\tilde{\tilde{Q}}^2[U]$ erwies sich als (erfreulich) langweilig. Dieser Wert schwankte für statistische Gleichgewichtskonfigurationen im Schnitt lediglich um ca. $\pm 1\%$, so daß die Festlegung der oberen Approximationsgrenze für das Polynom keine Probleme bereitet. Das Verhältnis der Eigenwerte zwischen der einfachen Fermion-Matrix und der präkonditionierten Version liegt bei ca.

$$\lambda_{max}(\tilde{Q}[U]^2) \div \lambda_{max}(\tilde{\tilde{Q}}[U]^2) \sim 5 \div 3, \tag{4.20}$$

womit schon alles Interessante zu dieser Größe gesagt wäre.

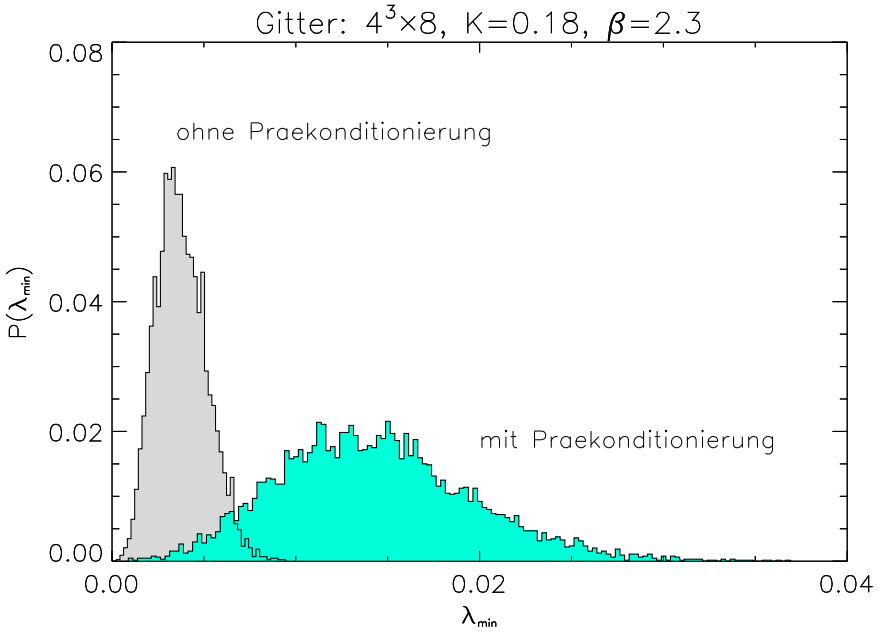


Abbildung 4.1: Verteilung des kleinsten Eigenwertes von $\tilde{Q}^2$ und $\tilde{\tilde{Q}}^2$.

Der kleinste Eigenwert von $\tilde{Q}[U]^2$ bzw. $\tilde{\tilde{Q}}[U]^2$ variiert dagegen typischerweise um mehr als eine Zehnerpotenz für verschiedene Eichkonfigurationen U. Dementsprechend werden die Schwankungen der Konditionszahl vom kleinsten Eigenwert dominiert. Wie zu erwarten und in Abbildung 4.1 zu sehen, ist der kleinste Eigenwert von $\tilde{\tilde{Q}}[U]^2$ ca. um einen Faktor vier größer als derjenige von $\tilde{Q}[U]^2$. Somit ist das Eigenwertspektrum der präkonditionierten Fermion-Matrix kompakter und besser für eine Polynomapproximation geeignet als das Spektrum der naiven Fermion-Matrix.

4.2.3 Zufalls-Matrix-Modelle

Es wird vermutet, daß entsprechend reskalierte Verteilungsfunktionen des kleinsten Eigenwertes von QCD-artigen Fermion-Matrizen universelle Größen sind [BBMS98]. Insbesondere kann man abhängig von der Struktur der Wirkung den entsprechenden Universalitätsklassen Zufalls-Matrix-Modelle zuordnen ([VER94]). Zu diesen Zufalls-Matrix-Modellen existieren analytische Ausdrücke für die Verteilungsfunktionen der kleinsten Eigenwerte auf unendlich großen Gittern. In einer ersten Studie wurden bemerkenswerte Übereinstimmungen zwischen den theoretischen Voraussagen der Zufalls-Matrix-Modelle

$$P(\lambda_{min}) = \sqrt{\frac{\pi}{2}} c\,(c\lambda_{min})^{3/2}\, I_{\frac{3}{2}}(c\lambda_{min}) e^{-\frac{1}{2}(c\lambda_{min})^2} \quad \text{mit } I_{\frac{3}{2}}\text{: Bessel-Funktion} \qquad (4.21)$$

und den numerischen Ergebnissen für das Spektrum des Dirac-Operators mit staggered Fermionen und der SU(2) Eichgruppe in der Fundamentaldarstellung gezeigt [BBJM98]. Insbesondere kann in diesem Fall der Skalierungsfaktor c aus dem Volumen und dem leicht zugänglichen Fermion-Kondensat berechnet werden.

Zu dem hier betrachteten Fall gehört das symplektische Zufalls-Matrix-Ensemble, dessen Verteilungsfunktion für das Quadrat der Fermion-Matrix ebenfalls durch Gleichung (4.21) gegeben ist. Allerdings ist keine Relation bekannt, die den Skalierungsfaktor c durch sekundäre Größen beschreibt, so daß man ihn nur durch einen Fit bestimmen kann.

Wie Abbildung 4.2 zeigt, kann die theoretische Verteilungsfunktion das Histogramm der gemessenen kleinsten Eigenwerte recht gut beschreiben. Das ist um so bemerkenswerter, als das Gitter doch relativ klein ist und der Parametersatz mit $K = 0.18$, $\beta = 2.3$, bei dem die Eigenwertverteilung gemessen wurde, noch recht weit vom chiralen Limes entfernt ist (siehe Abschnitt 7.1). Da das Präkonditionieren für den kleinsten Eigenwert im wesentlichen einer Skalenänderung um den Faktor vier gleich kommt, sollte die Verteilungsfunktion auch hierfür durch die Relation (4.21) beschreibbar sein. Der Vorteil der präkonditionierten Matrizen ist, daß ihre kleinsten Eigenwerte bei den großen Produktionsläufen permanent bestimmt werden, so daß man automatisch über eine sehr gute Statistik verfügt. Die Ergebnisse bestärken auch hier den Universalitätsgedanken, allerdings bleibt der Wermutstropfen, daß die

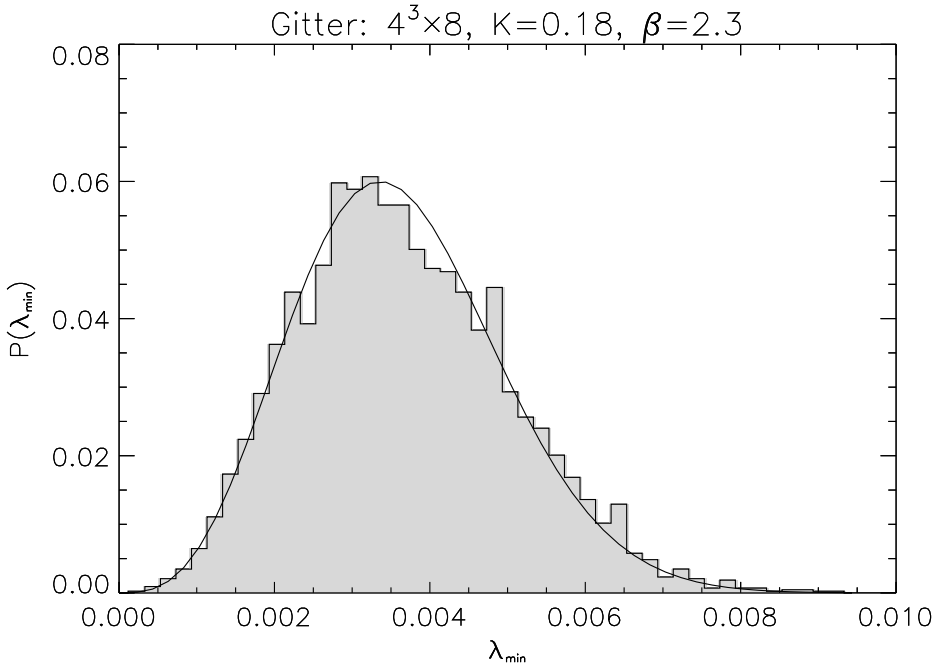


Abbildung 4.2: Verteilung des kleinsten Eigenwertes von $\tilde{Q}^2$. Die durchgezogene Linie entspricht einem Fit mit den theoretischen Voraussagen von Zufalls-Matrix-Modellen.

Skalierungskonstante c nur durch einen Fit [1] bestimmbar ist.

Die Theorie der Zufalls-Matrix-Modelle legt nahe, daß die Skalierungskonstante c proportional zum Gittervolumen V ist. Bedenkt man zudem, daß die Varianz σ^2 von $P(\lambda_{min})$ für typische Werte von c durch die Gauß-Funktion in (4.21) dominiert wird, so gelangt man zur Abschätzung $\sigma_V^2(P(\lambda_{min})) \propto V^{-1}$ oder für das Verhältnis zweier Volumina V_1, V_2 zu

$$\frac{\sigma_{V_1}^2(P(\lambda_{min}))}{\sigma_{V_2}^2(P(\lambda_{min}))} \simeq \frac{V_2}{V_1}. \tag{4.22}$$

Diese Relation ist sehr nützlich, denn oftmals, wenn man einen Lauf für ein großes Gitter vorbereiten will, kennt man die Verteilungsfunktion für den Parametersatz schon von einem kleineren Gitter. Da der Mittelwert von $P(\lambda_{min})$, wie in Abbildung 4.4 zu sehen, fast keine Volumenabhängigkeit zeigt, hat man somit schon eine gute Vorstellung davon, wie die Verteilung des kleinsten Eigenwertes für das große Gitter aussieht und kann von Anfang an mit einem geeigneten Approximationsintervall simulieren. Dabei ist die Abschätzung (4.22) schon für kleine Gitter recht gut erfüllt, wie die Werte zu Abbildung 4.4 zeigen

$$\frac{\sigma_{4^3\times 8}^2}{\sigma_{6^3\times 12}^2} = 5.25 \simeq \frac{6^3 \cdot 12}{4^3 \cdot 8} = 5.06. \tag{4.23}$$

[1] Alle Fits wurden nach der Methode der kleinsten quadratischen Abweichung mit dem Programmpaket ODRPACK durchgeführt.

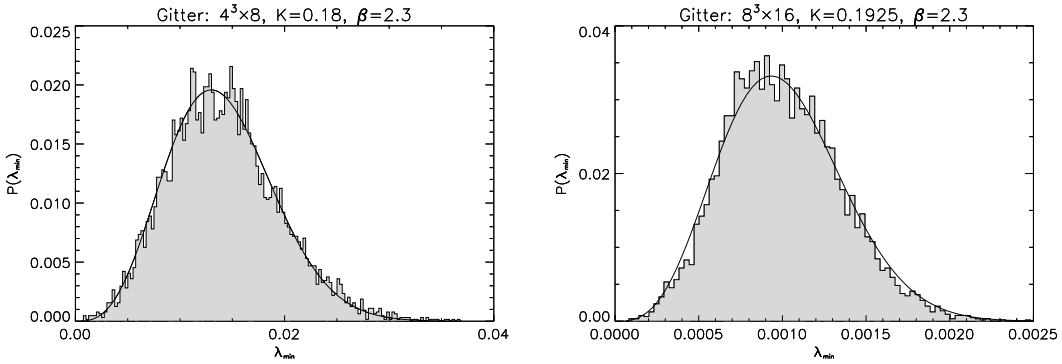


Abbildung 4.3: Verteilung des kleinsten Eigenwertes von $\tilde{\tilde{Q}}^2$. Die durchgezogene Linie entspricht einem Fit mit den theoretischen Voraussagen der Zufalls-Matrix-Modelle.

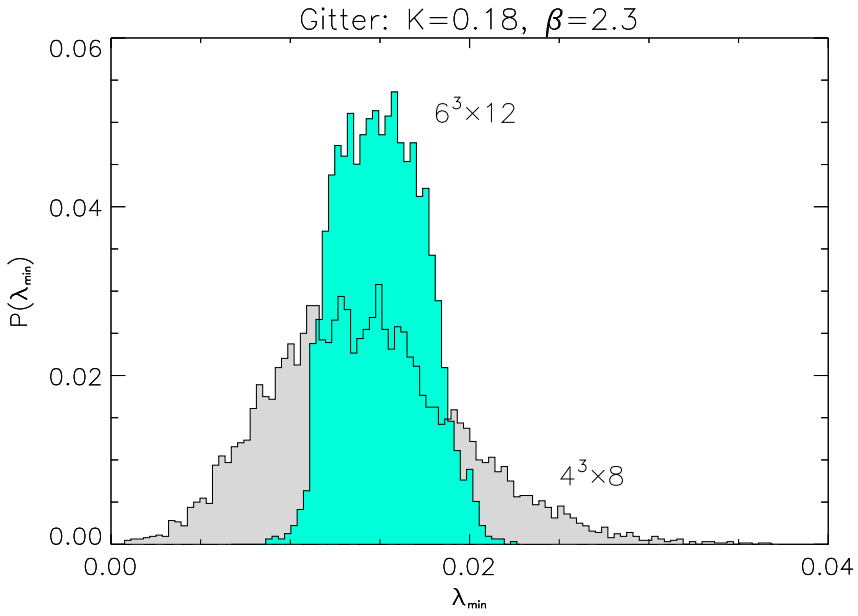


Abbildung 4.4: Verteilung des kleinsten Eigenwertes von $\tilde{\tilde{Q}}^2$.

4.3 Präkonditionierter MBA

Mit dem Wissen um das Verhalten der extremen Eigenwerte kann man nun den Gewinn durch einen präkonditionierten Updater abschätzen. Nach Gleichung (3.22) ist die Polynomlänge n und das Intervall $[\epsilon, \lambda]$ bei gleichbleibender Approximationsgüte über die Relation

$$n \sim \sqrt{\frac{\lambda}{\epsilon}} \tag{4.24}$$

verbunden. Präkonditionierung hebt den kleinsten Eigenwert von $\tilde{Q}[U]^2$ um den Faktor vier an und verringert den größten um ca. 5/3, so daß man mit einem circa 2.5 mal kürzeren Polynom bei gleicher Güte auskommt. Damit verringert sich die Autokorrelationszeit ebenfalls um den Faktor 2.5, und man spart fast die gleiche Zeit beim Update ein, da man entsprechend weniger Skalarfelder zu berücksichtigen hat.

4.3.1 Update der Felder

Allerdings gelangt man durch Präkonditionierung zu Matrizen mit drittnächsten Nachbarkopplungen

$$\tilde{\tilde{M}} = \gamma_5 - \gamma_5 K^2 M_{oe} M_{eo} \quad \wedge \quad \tilde{\tilde{Q}} = \begin{pmatrix} \gamma_5 & 0 \\ 0 & \tilde{\tilde{M}} \end{pmatrix} \quad \text{mit} \quad \det \tilde{Q} = \det \tilde{\tilde{Q}} = \det \tilde{\tilde{M}}. \tag{4.25}$$

Das ist für einen Parallelrechner problematisch. Dieses kann aber rückgängig gemacht werden, siehe z. B. [JEG97]. Dazu schreibt man in einem ersten Schritt das entsprechende Polynom für die Approximation

$$\left[\det \tilde{Q}^2\right]^{\frac{1}{4}} = \left[\det \tilde{\tilde{M}}^2\right]^{\frac{1}{4}} \approx \frac{1}{\det P_n\left(\tilde{\tilde{M}}^2\right)} \tag{4.26}$$

unter Benutzung der Definition $\rho_j = -\mu_j + i\nu_j$ um

$$\begin{aligned} \det P_n(\tilde{\tilde{M}}^2) &= c_0 \prod_{j=1}^{n} \det\left[\left(\tilde{\tilde{M}} + \mu_j\right)^2 + \nu_j^2\right] \\ &= c_0 \prod_{j=1}^{n} \det\left(\tilde{\tilde{M}} - \rho_j\right) \det\left(\tilde{\tilde{M}} - \rho_j^*\right). \end{aligned} \tag{4.27}$$

Nutzt man die Identität (4.4) aus, so gelangt man wieder zu Matrizen, die nur nächste Nachbarwechselwirkungen enthalten

$$\begin{aligned} \det\left(\tilde{\tilde{M}} - \rho_j\right) &= \det\gamma_5 \ \det\left(\gamma_5 - \rho_j - \gamma_5 K^2 M_{oe} M_{eo}\right) \\ &= \det\begin{pmatrix} \gamma_5 & -\gamma_5 K M_{eo} \\ -\gamma_5 K M_{oe} & \gamma_5 - \rho_j \end{pmatrix} \\ &= \det\left(\tilde{M} - P_o \rho_j\right) \quad \text{mit} \quad P_o = \begin{pmatrix} 0 & 0 \\ 0 & \mathbb{1} \end{pmatrix}. \end{aligned} \tag{4.28}$$

Damit läßt sich der pseudofermionische Teil der Wirkung durch

$$S_{f,pre}^{(n)}[U,\phi] = \sum_{j=1}^{n} \phi^{(j)\dagger}\left(\tilde{Q} - P_o\rho_j^*\right)\left(\tilde{Q} - P_o\rho_j\right)\phi^{(j)} \tag{4.29}$$

ausdrücken, und explizites Ausmultiplizieren liefert nach längerer Rechnung

$$S_{f,pre}^{(n)}[U,\phi] = \sum_{j=1}^{n} \phi^{(j)\dagger}\left[\tilde{Q}^2 + \begin{pmatrix} 0 & \rho_j\gamma_5 K M_{eo} \\ \rho_j^*\gamma_5 K M_{oe} & \rho_j\rho_j^* - (\rho_j + \rho_j^*)\gamma_5 \end{pmatrix}\right]\phi^{(j)}. \tag{4.30}$$

Vergleicht man dies mit der nichtpräkonditionierten Version nach (3.26)

$$S_f^{(n)}[U,\phi] = \sum_{j=1}^{n} \phi^{(j)\dagger} \left[\tilde{Q}^2 + \begin{pmatrix} \mu_j^2 + \nu_j^2 + 2\mu_j\gamma_5 & -2\mu_j\gamma_5 K M_{eo} \\ -2\mu_j\gamma_5 K M_{oe} & \mu_j^2 + \nu_j^2 + 2\mu_j\gamma_5 \end{pmatrix} \right] \phi^{(j)}, \quad (4.31)$$

so fällt auf, daß nur moderate Änderungen zur ursprünglichen Version für die Wechselwirkungsterme notwendig sind. Der Term $A^{(j)}$ aus Gleichung (3.28) wird abhängig vom Gitterplatz

$$\begin{aligned} A_{ee} &= 1 + 16K^2 \\ A_{oo} &= 1 + 16K^2 + \rho_j\rho_j^* - (\rho_j + \rho_j^*)\gamma_5, \end{aligned} \quad (4.32)$$

und in anderen Termen muß man je nach Wechselwirkung und Even- oder Odd-Gitterplatz $2\mu_j$ durch $-\rho_j$ bzw. $-\rho_j^*$ ersetzen.

4.3.2 Autokorrelationszeiten

Zur numerischen Bestimmung der integrierten Autokorrelationszeiten über die Autokorrelationsfunktion $\Gamma_f(t)$ einer Observablen f ist die Definition

$$\tau_{int} = \frac{1}{2} + \sum_{t=1}^{\infty} \Gamma_f(t) \quad (4.33)$$

ungeeignet, da die Varianz für den Erwartungswert des naiven Schätzers

$$\hat{\tau}_{int} = \frac{1}{2} + \sum_{t=1}^{N} \hat{\Gamma}_f(t) \quad (4.34)$$

im Limes $N \to \infty$ nicht gegen Null geht, d. h. $\hat{\tau}_{int}$ ist kein erwartungstreuer Schätzer. Der Grund hierfür liegt im schlechten Signal/Rausch-Verhältnis des Schätzers für $t \gg \tau_{exp}$. Die selbstkonsistente „Fenster-Methode“ nach Sokal umgeht dieses Problem, indem sie einen cut-off-Parameter t_{cut} in der Art einführt, daß ein guter Kompromiß zwischen cut-off-bedingtem Bias der Meßgröße $\hat{\tau}_{int}$ und der Varianz dieser Größe getroffen wird [SOK89]

$$t_{cut} \geq c\hat{\tau}_{int}(t_{cut}) \text{ mit } \quad c = 4...10. \quad (4.35)$$

Dieses Verfahren funktioniert allerdings erst dann stabil, wenn die zugrundeliegende Statistik deutlich über $1000\tau_{exp}$ umfaßt.

Im Rahmen des Binning-Verfahrens werden die Meßwerte der Observablen f solange zu größeren Blöcken zusammengefaßt, bis der naive statistische Fehler der Blockmitttelwerte ein Plateau erreicht [FP89]. Der Plateauwert σ gibt den statistischen Fehler der Observable f an; daraus kann die gesuchte Größe wieder berechnet werden

$$\hat{\tau}_{int} = \frac{\sigma^2}{2\sigma_0^2}, \quad (4.36)$$

wobei σ_0 der naive statistische Fehler auf den ungeblockten Daten ist. Wiederum benötigt auch dieser Algorithmus eine Statistik von mehreren Hundert τ_{exp}, um zuverlässige Werte zu liefern.

Beide Algorithmen haben sich als nicht adäquat erwiesen, da bei typischen Produktionsläufen 9 bis 32 unabhängige physikalische Konfigurationen gleichzeitig auf den 512 Prozessoren der T3E simuliert wurden. Die Statistik *pro* Konfiguration umfaßt für die Observable mit der längsten Autokorrelationszeit, der Plaquette, größenordnungsmäßig $50\tau_{ext}$ und damit zuwenig, um die integrierte Autokorrelationszeit zuverlässig mit einem der beiden Verfahren messen zu können. Als praktikabel hat

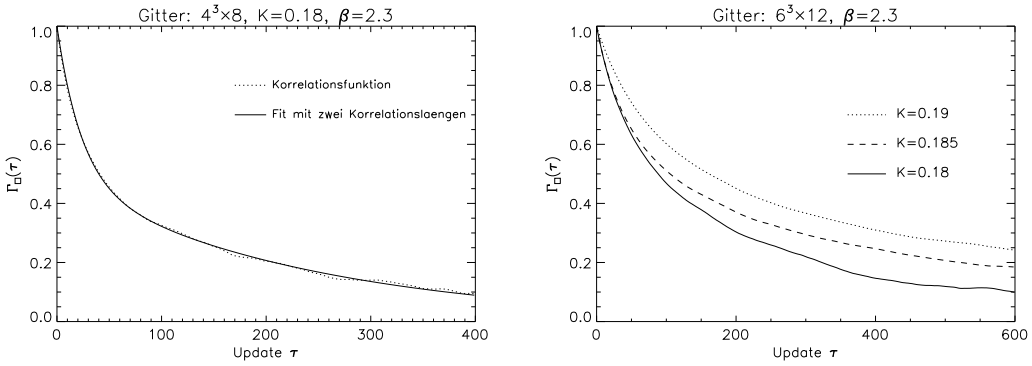


Abbildung 4.5: Das Diagramm links zeigt einen Fit an die Autokorrelationsfunktion der Plaquette durch $\bar{\Gamma}_{\Box}(t) = ae^{-\lambda_1 t} + (1-a)e^{-\lambda_2 t}$ auf dem Intervall $[1, 400]$. Rechts sind die Autokorrelationsfunktionen für verschiedene Polynomlängen bzw. K Werte abgetragen.

sich dagegen erwiesen, die Autokorrelationsfunktion auf jeder physikalischen Konfiguration zu messen, daraus den Mittelwert zu bilden, um daran einen Fit mit

$$\bar{\Gamma}_f(t) = ae^{-\lambda_1 t} + (1-a)e^{-\lambda_2 t} \tag{4.37}$$

auf dem Intervall $[t_1, t_2]$ durchzuführen. Wie Abbildung 4.5 zeigt, funktioniert dieser Fit oftmals schon für $t_1 = 1$ sehr gut. Die integrierte Autokorrelationszeit berechnet sich dann aus

$$\hat{\tau}_{int} = \frac{1}{2} + \sum_{t=1}^{t_1} \hat{\Gamma}(t) + \int_{t_1}^{\infty} \left(ae^{-\lambda_1 t} + (1-a)e^{-\lambda_2 t} \right) dt. \tag{4.38}$$

Hiermit bestimmte Werte für τ_{int} haben sich als stabil gegenüber (sinnvollen) Variationen von t_1 und t_2 erwiesen. Die Fehleranalyse für τ_{int} wird mittels des Jackknife-Prozederes durchgeführt [Ber92], indem man jeweils eine physikalische Konfiguration bei der Mittelwertbildung unbeachtet läßt.
Wie nicht anders erwartet, zeigt der präkonditionierte Algorithmus deutlich kürzere Autokorrelationszeiten als die naive Implementierung, da er die gleiche Approximationsgüte δ^2 mit kleineren Polynomen erreicht. Zudem steigt die CPU-Zeit pro

n	ε	δ^2	präkond.	$\tau_{int}(\Box)$
16	0.0004	0.00085		222(37)
8	0.002	0.00052	$\times$	96(21)

Tabelle 4.1: Autokorrelationszeiten für die Plaquette auf einem $4^3 \times 8$ Gitter bei $(K = 0.18, \beta = 2.3)$.

Sweep fast linear mit der Polynomlänge n an, so daß man insgesamt ca. einen Faktor fünf durch Präkonditionierung gewinnt.

Gleichzeitig bestärken die gemessenen Autokorrelationszeiten für die Produktionsläufe[2] auf $6^3 \times 12$ bei $\beta = 2.3$ und die verschiedenen Hopping-Parameter, daß auch für den in diesem Rahmen betrachteten Multi-Bosonischen Algorithmus mit quadratisch optimierten Polynomen die Hypothese

$$\tau_{int}(\Box) \propto n \tag{4.39}$$

gilt. Bei der Interpretation der Tabelle 4.2 muß man etwas vorsichtig sein, da sich zwei Effekte, zum einen unterschiedliche Polynomlängen und zum anderen verschiedene Hopping-Parameter, überlagern. An dieser Stelle hätte es zuviel CPU-Zeit in Anspruch genommen, extra für eine solche Tabelle den selben K-Wert mit verschiedenen, nicht optimalen Polynomen simulieren zu lassen. Neuer Studien zeigen aber auch, daß es neben n noch andere wichtige Faktoren für die Autokorrelationszeit geben muß.

K	n	ε	δ^2	$\tau_{int}(\Box)$	$\tau_{int}(\Box)/n$
0.18	8	0.008	0.00013	198(16)	24.8(20)
0.185	12	0.002	0.00021	310(18)	25.7(15)
0.19	16	0.0002	0.00019	437(24)	27.3(15)

Tabelle 4.2: Autokorrelationszeiten mit Präkonditionierung für die Plaquette auf einem $6^3 \times 12$ Gitter bei $\beta = 2.3$.

4.4 Präkonditionierung für den CGNE

Ein Algorithmus zur Lösung dünnbesetzter, linearer Gleichungssysteme wird im Rahmen dieses Projektes gemäß Gleichung (3.13) nur für Meßgrößen mit fermionischem Anteil benötigt. Da dieses Problem nicht integraler Bestandteil des Updaters ist, genießt der Algorithmus bei weitem nicht den Stellenwert wie in vielen hybriden Monte-Carlo-Simulationen.

[2]Alle Produktionsläufe nutzten zusätzlich noch einen Korrekturschritt, der allerdings keinen relevanten Einfluß auf die Autokorrelationszeit hat, s. Kapitel 5.

Das Lösen eines linearen Gleichungssystem

$$Az = b \tag{4.40}$$

kann man für eine hermitesche, positiv definite Matrix A wiederum in ein Minimierungsproblem

$$f(z) = \frac{1}{2} < z, Az > -bz \tag{4.41}$$

für die Funktion $f(z)$ umformulieren. Damit steht sofort eine große Klasse iterativer Lösungsalgorithmen zur Verfügung, insbesondere wieder die Methode der konjugierten Gradienten. Ausgehend vom Gradienten

$$g(z) = Az - b \tag{4.42}$$

ist dieser Algorithmus mit dem Startvektor z_1, dem Startresiduum $r_1 = b - Az_1$ und der Start-/Suchrichtung $p_1 = r_1 = -g(z_1)$ durch Iteration des Gleichungssystems

$$\begin{aligned} \alpha_i &= \frac{< r_i, r_i >}{< p_i, Ap_i >}, \qquad z_{i+1} = z_i + \alpha_i p_i, \\ r_{i+1} &= r_i - \alpha_i A p_i, \qquad \beta_i = \frac{< r_{i+1}, r_{i+1} >}{< r_i, r_i >}, \\ p_{i+1} &= r_{i+1} + \beta_i p_i \end{aligned} \tag{4.43}$$

definiert. Dieses Verfahren konvergiert in maximal $rang(A)$ Schritten, im allgemeinen wird aber abgebrochen, sobald die Norm des Residuums $\|r_i\|_2 = \|b - Az_i\|_2$ einen vorgegebenen Schwellwert unterschreitet.

In unserem Fall entspricht die Matrix A gerade der Fermion-Matrix Q, die weder hermitesch noch positiv definit ist. Anwendbar wird der Algorithmus wieder, indem man beide Seiten der Gleichung (4.40) mit $Q^\dagger$ multipliziert

$$Q^\dagger Q z = \tilde{Q}^2 z = Q^\dagger b, \tag{4.44}$$

was auf die Variante „*Conjugate Gradient on Normal Equations*“ (CGNE) führt.

Die Konvergenzrate $\mathcal{R}$ des konjugierten Gradientenverfahrens hängt entscheidend von der Konditionszahl der Matrix A ab [Bor96]

$$\mathcal{R} \stackrel{\text{def}}{=} \lim_{n\to\infty} \frac{\|r_{n+1}\|_2}{\|r_n\|_2} = \frac{\kappa(A) - 1}{\kappa(A) + 1}. \tag{4.45}$$

Dies legt die Verwendung präkonditionierter Matrizen anstelle von $\tilde{Q}^2$ nahe. Wieder kann dafür die Even-Odd-Blockstruktur von $\tilde{Q}$ ausgenutzt werden

$$\tilde{Q} = \tilde{L}\tilde{\tilde{Q}}\tilde{U} = \begin{pmatrix} \gamma_5 & 0 \\ -\gamma_5 K M_{oe} & \gamma_5 \end{pmatrix} \begin{pmatrix} \gamma_5 & 0 \\ 0 & \gamma_5 - \gamma_5 K^2 M_{oe} M_{eo} \end{pmatrix} \begin{pmatrix} \gamma_5 & -\gamma_5 K M_{eo} \\ 0 & \gamma_5 \end{pmatrix},$$

wobei das Inverse zu den Dreiecksmatrizen L und U direkt abgelesen werden kann

$$\tilde{L}^{-1} = \begin{pmatrix} \gamma_5 & 0 \\ \gamma_5 K M_{oe} & \gamma_5 \end{pmatrix} \quad \wedge \quad \tilde{U}^{-1} = \begin{pmatrix} \gamma_5 & \gamma_5 K M_{eo} \\ 0 & \gamma_5 \end{pmatrix}. \tag{4.46}$$

Von der Matrix $\tilde{\tilde{Q}}^2$ ist bereits bekannt, daß sie eine kleinere Konditionszahl als $\tilde{Q}^2$ besitzt. Lediglich die CGNE-Gleichung (4.44) muß noch passend umgeschrieben werden

$$\begin{aligned} Qz = b \quad &\Rightarrow \quad \gamma_5 Q z = \gamma_5 b \quad \Rightarrow \quad \tilde{L}\tilde{\tilde{Q}}\tilde{U}z = \gamma_5 b \\ &\Rightarrow \quad \tilde{\tilde{Q}}^2 \tilde{U} z = \tilde{\tilde{Q}} \tilde{L}^{-1} \gamma_5 b \\ &\Rightarrow \quad z = \tilde{U}^{-1} \left(\tilde{\tilde{Q}}^2 \right)^{-1} \tilde{\tilde{Q}} \tilde{L}^{-1} \gamma_5 b. \end{aligned} \tag{4.47}$$

Die Inversion von $\tilde{\tilde{Q}}^2$ wird mit dem konjugierten Gradientenverfahren durchgeführt, alle anderen Inversen sind bekannt. Der Aufwand für das Präkonditionieren ist somit vergleichbar mit drei Fermion-Matrix Multiplikationen. Wie Abbildung 4.6 zeigt, wird man dafür mit einer viel günstigeren Konvergenzrate entschädigt.

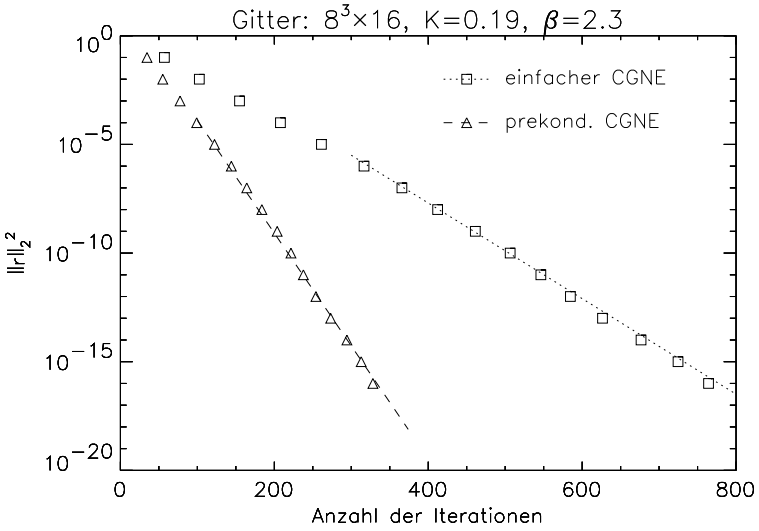


Abbildung 4.6: Vergleich der Konvergenzgeschwindigkeiten zu $(\tilde{Q}^2)^{-1}$ auf einer typischen Eichkonfiguration. Auf der y-Achse ist das Quadrat des Residuums $\|r\|_2$ aufgetragen. Die Linien entsprechen der jeweiligen asymptotischen Konvergenzgeschwindigkeit aus Gleichung (4.45).

Trotzdem sollte man anmerken, daß im Rahmen der Gitter-QCD schon bessere Algorithmen zur Inversion der Fermion-Matrix bekannt sind [BOR96][F97]. Zum einen hat D. Talkenberger allerdings gezeigt, daß der BiCGstab-Algorithmus, im Gegensatz zur QCD, für dieses Modell ineffizienter als der CGNE ist [TAL97] und zum anderen entfallen summa summarum weniger als 10% der gesamten CPU-Zeit auf die Inversion von Fermion-Matrizen. Der teilweise enorme Aufwand für die Programmierung der alternativen Algorithmen bei gleichzeitig ungewissem Ausgang der Effizienzfrage lohnt deshalb kaum. Man kann die eingesparte Zeit besser in die Optimierung der Implementation des Updaters investieren.

Kapitel 5

Korrekturverfahren

5.1 Globaler Metropolis-Korrekturschritt

Um den Rechenaufwand pro unabhängiger Konfiguration möglichst zu minimieren, möchte man natürlich die Zahl der Skalarfelder klein halten. Auf der anderen Seite sollen die Fehler der Approximation durch das Polynom tolerierbar bleiben. Diese beiden Gegensätze können auf einen Nenner gebracht werden, indem man die durch den endlichen Grad des Polynoms bedingten Fehler eines Multi-Bosonischen Sweeps mit einem globalen Metropolis-Test korrigiert. Je besser die Approximation durch das Polynom ist, desto wahrscheinlicher wird der anschließende Metropolis-Schritt das Ergebnis des Multi-Bosonischen Sweeps akzeptieren. Die Hoffnung ist, daß bei relativ hohen Akzeptanzraten die Autokorrelationszeit vom Metropolis-Schritt nicht beeinflußt wird, so daß man trotz kleiner Polynome beliebig genaue Algorithmen mit kurzer Korrelationszeit erhält.

Den Ausgangspunkt bildet der Fehlerterm $\Delta E[U]$ in der Zustandssumme durch die Approximation mit dem Polynom P_n [1]

$$\begin{aligned} Z &\equiv \int [dU]\, e^{-S_{CV}[U]} = \int [dU]\, \left(\det \tilde{Q}^2\right)^{\frac{1}{4}}\, e^{-S_g[U]} \\ &= \int [dU]\, \left(\det \tilde{Q}^2\right)^{\frac{1}{4}} \frac{\det P_n(\tilde{Q}^2)}{\det P_n(\tilde{Q}^2)}\, e^{-S_g[U]} \\ &= \int [dU] \underbrace{\left[\det\left(\tilde{Q}^2 P(\tilde{Q}^2)^4\right)\right]^{\frac{1}{4}}}_{\stackrel{\text{def}}{=} \Delta E[U]} \int [d\phi^\dagger d\phi]\, e^{-S_{CV}^{(n)}[U,\phi]}. \end{aligned} \tag{5.1}$$

Ein exakter Updater muß für den Übergang der Feldkonfiguration $[U] \to [U']$ die Bedingung des detaillierten Gleichgewichtes

$$\frac{P(U \to U')}{P(U' \to U)} = \frac{e^{-S_{CV}[U']}}{e^{-S_{CV}[U]}} = \frac{\Delta E[U']}{\Delta E[U]} \cdot \frac{\int [d\phi^\dagger d\phi]\, e^{-S_{CV}^{(n)}[U',\phi]}}{\int [d\phi^\dagger d\phi]\, e^{-S_{CV}^{(n)}[U,\phi]}} \tag{5.2}$$

[1] Im weiteren werden alle Gleichungen mit der nichtpräkonditionierten Matrix $\tilde{Q}$ geschrieben, alle Aussagen gelten analog für die präkonditionierte Matrix $\tilde{\tilde{Q}}$.

erfüllen. Gelangt man im Rahmen eines Eichfeld-Sweeps, wie es z. B. in Kapitel 3.2.2 vorgestellt wurde, von einer Ausgangskonfiguration $[U]$ zur neuen „Test-Konfiguration“ $[U']$ gemäß des detaillierten Gleichgewichtes

$$\frac{P_\phi(U \to U')}{P_\phi(U' \to U)} = \frac{e^{-S_{CV}^{(n)}[U',\phi]}}{e^{-S_{CV}^{(n)}[U,\phi]}}, \tag{5.3}$$

so können der gesamten Markov-Kette die exakten Übergangsraten gemäß Gleichung (5.2) aufgeprägt werden. Dies geschieht dadurch, daß man die Eichkonfiguration $[U']$ nur mit der Wahrscheinlichkeit

$$P_A([U] \to [U']) = F\left(\frac{\Delta E[U']}{\Delta E[U]}\right) \tag{5.4}$$

akzeptiert, wobei die Abbildung $F : [0 : \infty] \to [0, 1]$ der Bedingung

$$\frac{F(z)}{F(1/z)} = z \tag{5.5}$$

genügen muß. Populäre Möglichkeiten sind

$$F_1(z) = \min(1, z) \quad \vee \quad F_2(z) = \frac{z}{1+z}. \tag{5.6}$$

Die erste Wahl erzeugt die größtmögliche Akzeptanzrate, was aber nicht notwendigerweise zu den kürzesten Korrelationszeiten führen muß. In diesem Fall soll aber der Metropolis-Schritt die Dynamik des Multi-Bosonischen Updaters möglichst wenig stören, so daß mit Ausnahme einiger Testläufe immer $F_1(z)$ verwendet wurde.

5.2 Zwei-Schritt-Approximation

5.2.1 Grundidee

Einer direkten Berechnung des Ausdrucks $\Delta E[U]$ steht aus ökonomischen Gründen die darin enthaltene Determinante entgegen. Zudem taucht die Determinante hier auch noch unterhalb einer vierten Wurzel auf, so daß man sie im Gegensatz zu vielen QCD-Anwendungen nicht durch „Bosonifizierung“ und anschließender Monte-Carlo-Simulation berechnen kann [BDFG96]. I. Montvays Vorschlag ist es, diese Fermion-Determinante genauso zu umgehen, wie man es schon für die Curci-Veneziano-Wirkung getan hat [MON96]. Dazu bestimmt man ein weiteres Polynom R_m vom Grad $m \gg n$, mit dem der Fehlerterm

$$\begin{aligned} \Delta E[U] &= \left(\det \tilde{Q}[U]^2\right)^{\frac{1}{4}} \det P_n(\tilde{Q}[U]^2) \\ &\simeq \frac{1}{\det R_m(\tilde{Q}[U]^2)} = \int [d\eta^\dagger d\eta] \exp\left\{-\eta^\dagger R_m\left(\tilde{Q}[U]^2\right)\eta\right\} \end{aligned} \tag{5.7}$$

approximiert werden kann. Das zweite Polynom R_m sollte dazu die Relation

$$x^{\frac{1}{4}} P_n(x) \simeq \frac{1}{R_m(x)} \;\Rightarrow\; 1 - x^{\frac{1}{4}} P_n(x) R_m(x) \simeq 0 \tag{5.8}$$

möglichst gut erfüllen. Wie man an dieser Gleichung sieht, rührt der Name „Zwei-Schritt-Approximation“ aus der Tatsache, daß $R_m(x)$ ein Korrekturpolynom für die Approximation von P_n an $x^{-0.25}$ ist. An dieser Stelle bieten sich wieder quadratisch optimierte Polynome an, für die der Ausdruck

$$\delta = \left[\frac{1}{\lambda-\epsilon}\int_\epsilon^\lambda dx[1-x^\alpha P_n(x)R_m(x)]^2\right]^{\frac{1}{2}} \text{ mit } \quad \alpha = \frac{1}{4} \tag{5.9}$$

minimal ist. Wenn das Polynom P_n vorgegeben ist, kann R_m für beliebiges α analytisch bestimmt werden. Damit ist dieses Verfahren auch auf andere Wirkungen übertragbar.

Wie Gleichung (5.7) schon andeutet, werden die Determinanten von $R_m(\tilde{Q}^2)$ über bosonische Pfadintegrale, d. h. per Monte-Carlo-Simulation bestimmt. Wichtiges Detail ist dabei, daß man hier nicht mehr auf eine lokale Theorie angewiesen ist, da man nur ein globales Update testen will. Das Integral in Gleichung (5.7) muß deshalb nicht erst in ein Multi-Bosonisches Integral umgewandelt werden, wie z. B. in Gleichung (3.11) geschehen, sondern es kann als bosonisches Integral mit einem einzigen pseudofermionischen Feld η berechnet werden. Insbesondere wächst damit der Speicherbedarf für die Simulation nicht mit dem Grad m des Korrekturpolynoms und man ist nicht darauf angewiesen, das Polynom als Produkt von Monomen darzustellen.

Ein geeigneter, erwartungstreuer Schätzer für $P_A([U]\to[U'])$ kann generiert werden, indem man Zufallsvektoren η gemäß

$$dprob(\eta) = \frac{e^{-\eta^\dagger R_m(\tilde{Q}[U]^2)\eta}}{\int[d\eta^\dagger][d\eta]^{-\eta^\dagger R_m(\tilde{Q}[U]^2)\eta}}[d\eta^\dagger][d\eta] \tag{5.10}$$

erzeugt. Die Akzeptanzwahrscheinlichkeit des Metropolis-Schrittes wird dann durch

$$\begin{aligned} P_A^{(m)}([U]\to[U']) &= \min\left\{1, A^{(m)}(\eta;[U]\to[U'])\right\} \\ \text{mit} \quad A^{(m)}(\eta;[U]\to[U']) &= \exp\left(-\eta^\dagger\left[R_m(\tilde{Q}[U']^2)-R_m(\tilde{Q}[U]^2)\right]\eta\right) \end{aligned} \tag{5.11}$$

berechnet. Da der Übergang $[U]\to[U']$ beliebig oft im Laufe einer Trajektorie vorkommen kann, ergibt sich die Übergangswahrscheinlichkeit durch Mittelung

$$< P_A^{(m)}([U]\to[U']) >_\eta = \frac{\int_{A>1}[d\eta^\dagger][d\eta]e^{-\eta^\dagger R_m(\tilde{Q}[U]^2)\eta} + \int_{A<1}[d\eta^\dagger][d\eta]e^{-\eta^\dagger R_m(\tilde{Q}[U']^2)\eta}}{\int[d\eta^\dagger][d\eta]e^{-\eta^\dagger R_m(\tilde{Q}[U]^2)\eta}}. \tag{5.12}$$

Darüber hinaus ist die Relation

$$A^{(m)}(\eta;[U]\to[U']) = \frac{1}{A^{(m)}(\eta;[U']\to[U])} \tag{5.13}$$

erfüllt, so daß sich ein detailliertes Gleichgewicht gemäß

$$\frac{< P_A^{(m)}([U]\to[U']) >_\eta}{< P_A^{(m)}([U']\to[U]) >_\eta} = \frac{\det R_m(\tilde{Q}[U]^2)}{\det R_m(\tilde{Q}[U']^2)} \overset{m\to\infty}{=} \frac{\Delta E[U']}{\Delta E[U]} = \frac{P_A([U]\to[U'])}{P_A([U']\to[U])} \tag{5.14}$$

einstellt. Dabei ist es prinzipiell egal, wie groß die Anzahl $N_{noisy} \geq 1$ der „Noisy Estimator“ $A_i^{(m)}(\eta_i;[U] \to [U'])$ ist, die man zur Bewertung eines Testschrittes $[U] \to [U']$ heranzieht.

Problematisch an diesem Korrekturschritt ist noch die Ermittlung von Zufallsvektoren nach der Verteilung (5.10). Zwar kann man relativ leicht normalverteilte Zufallsvektoren $\bar{\eta}$ gemäß

$$dprob(\bar{\eta}) = \frac{e^{-\bar{\eta}^\dagger \bar{\eta}}}{\int [d\bar{\eta}^\dagger][d\bar{\eta}] e^{-\bar{\eta}^\dagger \bar{\eta}}} [d\bar{\eta}^\dagger][d\bar{\eta}] \tag{5.15}$$

berechnen, die Umrechnung nach

$$\eta = R_m(\tilde{Q}[U]^2)^{-\frac{1}{2}} \bar{\eta} \tag{5.16}$$

ist aber in dieser Form zeitaufwendig. Der bisherigen Intention folgend, sollte man auch dieses Problem durch eine Approximation an $R_m(x)^{-1/2}$ mit einem Polynom T_l durch die Approximationskette

$$T_l(x) \simeq R_m(x)^{-1/2} \simeq x^{1/8} P_n(x)^{-1/2} \simeq x^{1/8} S_l(P_n(x)) \tag{5.17}$$

lösen. Hierbei ist S_l ein Hilfspolynom zur Approximation der Wurzelfunktion im Intervall $[\lambda^{-1/4}, \epsilon^{-1/4}]$, das mit Gleichung (3.19) für $\alpha = -1/2$ bestimmt werden kann. Das gesuchte Polynom T_l wird wiederum als Minimum der quadratischen Abweichung

$$\delta = \left[\frac{1}{\lambda - \epsilon} \int_\epsilon^\lambda dx \left[x^{-1/8} S_l(P_n(x)) - T_l(x) \right]^2 \right]^{\frac{1}{2}} \tag{5.18}$$

definiert. Um entlang der gesamten Approximationskette (5.17) dieselbe Approximationsgüte gewährleisten zu können, sollte $l = m$ gewählt werden. Ein praktikabeler Ausdruck zur Berechnung der Akzeptanzwahrscheinlichkeit aus einem normalverteilten Zufallsvektor $\bar{\eta}$ ist somit

$$\begin{aligned} A^{(m)}(\bar{\eta};[U] \to [U']) = \\ \exp\left\{ -\bar{\eta}^\dagger T_m(\tilde{Q}[U]^2) \left[R_m(\tilde{Q}[U']^2) - R_m(\tilde{Q}[U]^2) \right] T_m(\tilde{Q}[U]^2) \bar{\eta} \right\}. \end{aligned} \tag{5.19}$$

Man kann erwarten, daß dieser Ausdruck gegenüber den Fehlern durch die Polynomapproximation $T_m(x)$ von $R_m(x)^{-1/2}$ numerisch stabil ist, da sowohl der Minuend $R_m(\tilde{Q}[U']^2)$ als auch der Subtrahend $R_m(\tilde{Q}[U]^2$ gleichermaßen davon betroffen sind. Die Auswertung von (5.19) ist allerdings numerisch aufwendig, da dazu $3m$ Fermion-Matrix-Multiplikationen mit $\tilde{Q}[U]^2$ durchgeführt werden müssen. Der Aufwand ist mit einem Sweep der Eich- und Skalarfelder vergleichbar.

5.2.2 Modifikation und Optimierung

Der Korrekturschritt (5.19) wurde bis Mitte 1997 auf der Cray-T3E sowohl beim naiven Updater als auch im Rahmen des präkonditionierten Updaters eingesetzt. Seine

Ablösung beruhte nicht auf Problemen innerhalb der Monte-Carlo-Simulationen, sondern auf dem Umstand, daß die Generierung geeigneter Polynome $R_m(x), S_m(x)$ und $T_m(x)$ für die anstehenden Parametersätze (K, β) wochenlange Laufzeiten des Maple-Programms auf Workstations nötig gemacht hätten. Ein sinnvolle Optimierung der Approximationsintervalle und Polynomlängen ist bei solchen Wartezeiten aber nicht möglich, weshalb ein neues Verfahren implementiert wurde, dessen Polynome in deutlich kürzerer Zeit generiert werden können [MON].

Ausgangspunkt für die neue Korrekturmethode bildet die Darstellung (4.27)

$$R_m(\tilde{Q}^2) = c_0 \prod_{j=1}^{m} \left(\tilde{Q} - \rho_j\right) \left(\tilde{Q} - \rho_j^*\right)$$

unter Berücksichtigung von $\rho_j = -\mu_j + i\nu_j$. Daraus definiert man sich das Wurzelpolynom

$$R_{\sqrt{}}(\tilde{Q}) = \sqrt{c_0} \prod_{j=1}^{m} (\tilde{Q} - \rho_j), \tag{5.20}$$

um damit das Polynom R_m in einer positiven Form

$$R_m(\tilde{Q}^2) = R_{\sqrt{}}(\tilde{Q})^\dagger R_{\sqrt{}}(\tilde{Q}) \tag{5.21}$$

schreiben zu können. Der gesuchte Zufallsvektor η gemäß der Verteilung (5.10) läßt sich nun unter Vermeidung der inversen Wurzel aus R_m aus einem normalverteilten Vektor $\bar{\eta}$ gewinnen

$$\eta = R_{\sqrt{}}(\tilde{Q})^{-1} \bar{\eta}. \tag{5.22}$$

Die Inversion von $R_{\sqrt{}}(\tilde{Q})$ könnte man direkt mit einem iterativen Lösungsverfahren bestimmen. Alternativ dazu kann man Gleichung (5.21) zu

$$R_{\sqrt{}}(\tilde{Q})^{-1} = \frac{R_{\sqrt{}}(\tilde{Q})^\dagger}{R_m(\tilde{Q}^2)} = R_m(\tilde{Q}^2)^{-1} R_{\sqrt{}}(\tilde{Q})^\dagger \tag{5.23}$$

umformen und gelangt zu einem äquivalenten Ausdruck

$$\begin{aligned} &A^{(m)}(\bar{\eta}; [U] \to [U']) = \\ &\exp\left\{-\bar{\eta}^\dagger \left[1 - R_{\sqrt{}}(\tilde{Q}[U]) R_m(\tilde{Q}[U]^2)^{-1} R_m(\tilde{Q}[U']^2) R_m(\tilde{Q}[U]^2)^{-1} R_{\sqrt{}}(\tilde{Q}[U])^\dagger\right] \bar{\eta}\right\} \end{aligned} \tag{5.24}$$

zur Relation (5.19). Die Inversion von $R_m(\tilde{Q}[U]^2)$ kann mit einem konjugierten Gradientenverfahren durchgeführt werden, da $R_m(\tilde{Q}[U]^2)$ hermitesch ist. Darüber hinaus besitzt $R_m(\tilde{Q}[U]^2)$ eine Konditionszahl nahe bei eins, da hiermit lediglich die Approximation von P_n an $x^{-0.25}$ nachgebessert wird. Somit weist dieses Verfahren eine gute Konvergenzrate auf.

Wiederum besteht die Möglichkeit, $R_m(x)^{-1}$ durch ein Polynom $T_l(x)$ auf dem Intervall $[\epsilon, \lambda]$ zu approximieren. Allerdings fallen die Fehler durch diese Approximation

nur im Subtrahenden von Gleichung (5.24) an, so daß man mit numerischen Ungenauigkeiten rechnen muß, die deutlich größer sind als es die minimierte Abweichung δ_{T_m} zu $T_l(x)$ erwarten lassen würde. Man benötigt also Polynome T_l mit $\delta_{T_l} \ll \delta_{R_m}$, d. h. $l > m$.

Im Rahmen dieses Projektes wird ein mittlerer Fehler von 0.3% in der Akzeptanzwahrscheinlichkeit $A^{m,l}(\eta;[U] \to [U'])$ durch die Polynomapproximation $T_l(x)$ an $R_m(x)^{-1}$ toleriert. A priori ist nicht klar, wieviel größer l als m gewählt werden muß, um dieses zu gewährleisten, sondern es muß in Testläufen bestimmt werden. Dazu bieten sich die beiden folgenden Testverfahren an:

Update-Test: Bei diesem Verfahren bestimmt man für typische Eichkonfigurationen $[U]$ und Updates $[U']$ den Ausdruck (5.24) zum einen mit dem Polynom $T_l(x) \simeq R_m(x)^{-1}$ und zum anderen mit dem Ergebnis eines konjugierten Gradientenverfahrens bei maximaler Präzision für $R_m(x)^{-1}$. Die Differenz beider Ergebnisse ergibt einen Schätzer für den Fehler.

„Standardtest“: Der Update-Test ist für den „täglichen Gebrauch“, insbesondere für eine permanente Qualitätssicherung während langer Produktionsläufe, zu umständlich. Eine Möglichkeit, sich den problematischen Teil mit dem konjugierten Gradientenverfahren zu sparen, besteht darin, auf typischen Eichkonfigurationen $[U]$ gar kein Update durchzuführen und die Abweichung der Akzeptanzwahrscheinlichkeit $A^{m,l}(\eta;[U] \to [U])$ von eins als Maß für den mittleren Fehler aufzufassen. Wie Abbildung 5.1 zeigt, modelliert dieses viel einfachere Testverfahren den wirklichen Fehler recht gut.

Erfahrungswerte mit diesen Testverfahren legen nahe, daß für gewünschte mittlere Genauigkeiten von 0.3% die Verhältnisse der Polynomlängen im Bereich

$$m \div l = 2 \div 3 \sim 3 \div 4 \tag{5.25}$$

liegen sollten, wobei 3/4 eher für große $m \geq 140$ gilt, während 2/3 für $m \approx 40$ benötigt wird. Damit ist die Polynomapproximation durch $T_l(x)$ dem konjugierten Gradientenverfahren in jedem Fall überlegen, denn für eine vergleichbare Qualität benötigt letzteres ca. 3 bis 5 Iterationen, d. h. $3m$ bis $5m$ Matrixmultiplikationen mit $\tilde{Q}^2$ im Gegensatz zu den $\frac{4}{3}m \sim \frac{3}{2}m$ Multiplikationen, die zur Berechnung des Approximationspolynoms nötig sind. Präkonditionierte Verfahren bringen in diesem speziellen Fall auch keine Vorteile, da allein die eigentliche Präkonditionierung aufwendiger sein dürfte als die Berechnung von T_l. Als prägnante Schreibweise zur Charakterisierung der drei an einem Lauf beteiligten Polynome hat sich

$$\begin{aligned} P_{\epsilon,\lambda}(n,m,l) \text{ mit } \quad [\epsilon,\lambda] &: \text{ Approximationsintervall} \\ n &: \text{ primäres Polynom } P_n(x) \simeq x^{-0.25} \\ m &: \text{ sekundäres Polynom } R_m, P_n(x)R_m(x) \simeq x^{-0.25} \\ l &: \text{ Polynom } T_l(x) \simeq R_m(x)^{-1} \end{aligned} \tag{5.26}$$

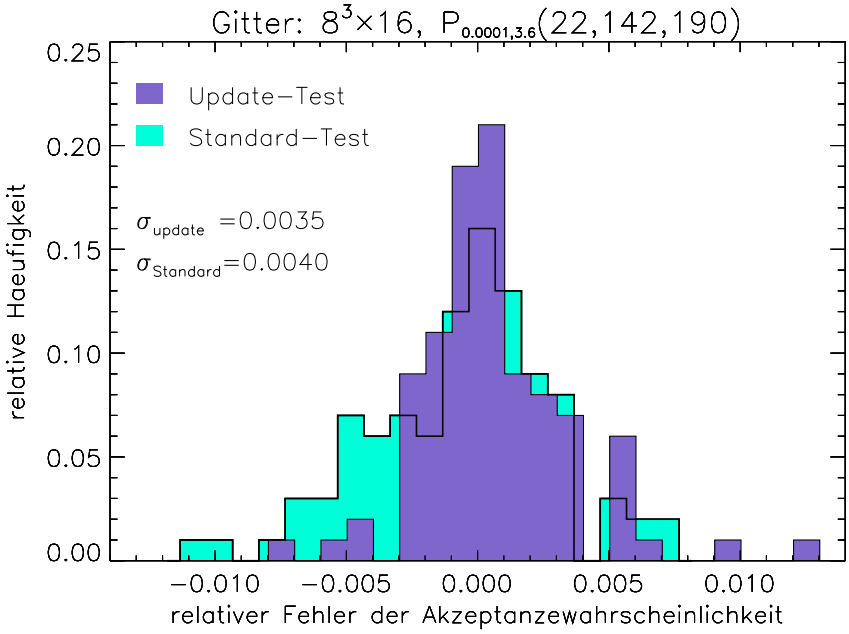


Abbildung 5.1: Vergleich der Testverfahren zur Bestimmung des Fehlers durch die Polynominversion $T_l(x) \simeq R_m(x)^{-1}$ im Rausch-Korrekturschritt.

durchgesetzt. Verbleibt noch die Frage nach der Effizienz der beiden Korrekturrelationen (5.19) und (5.24). Zur Auswertung der ursprünglichen Version sind $3m$ Multiplikationen mit $\tilde{Q}^2$, d. h. $6m$ Fermion-Matrix Multiplikationen nötig. Das zweite Verfahren benötigt dafür im ungünstigsten Fall mit $l = \frac{3}{2}m$ ebenso viele Multiplikationen, wird aber für lange Polynome R_m mit $m \geq 140$ einen leichten Vorteil von ca. 5% herausarbeiten können, da für diese Polynomlängen $l \simeq \frac{4}{3}m$ sinnvoll ist; allerdings immer zum Preis eines „Nullpunkt-Rauschens“ $A^{m,l}(\eta; [U] \to [U]) \simeq 0.3\%$.

5.2.3 Einfluß auf die Korrelationslänge

Die Zwei-Schritt-Approximation durch $P_n(x)$ und $R_m(x)$ ist in der Hoffnung eingeführt worden, daß man die Genauigkeit von $P_n(x)R_m(x)$ zum Preis der Korrelationslänge $\tau_{int} \propto n$ bekommt und zusätzlich nur für n Skalarfelder Speicherplatz benötigt. Diese Hoffnung bewahrheitet sich, wie man in Abbildung 5.2 sieht, falls man den Grad n des ersten Polynoms groß genug wählt, damit die Akzeptanzrate grob gesprochen über 80% liegt. Insbesondere hängt die Korrelationslänge dann auch nicht mehr von der Anzahl der Noisy Correction Steps pro Sweep ab, so daß man natürlich nur einen Schritt durchführen sollte.

Um zu den gewünschten Akzeptanzraten zu gelangen, hat sich in umfangreichen Tests die „Faustregel“

$$n \div m = 1 \div 5 \sim 1 \div 6 \tag{5.27}$$

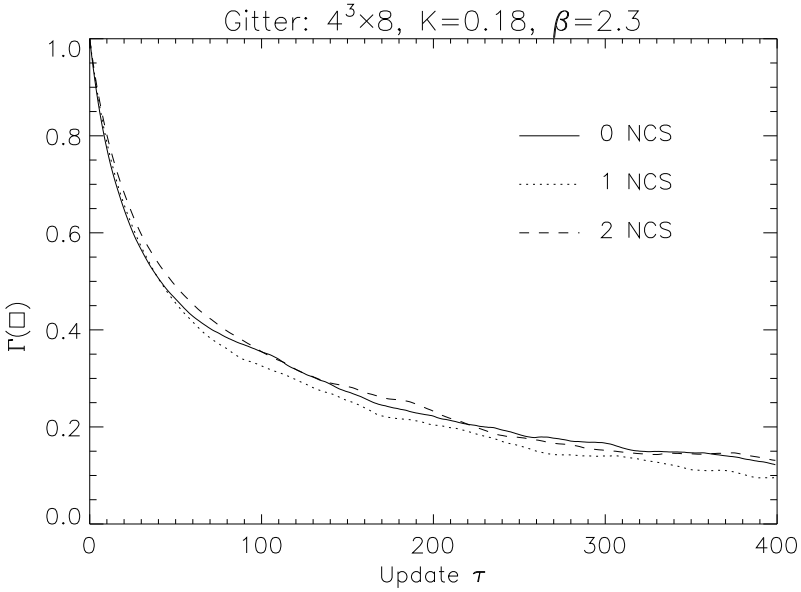


Abbildung 5.2: Autokorrelationsfunktion der Plaquette für verschiedene Anzahlen von Noisy Correction Steps (NCS). Die Akzeptanzrate liegt bei circa $85 \sim 87\%$.

für das Verhältnis der Polynomlängen bewährt. Damit gewinnt man durch das Zwei-Schritt-Verfahren ungefähr einen Faktor fünf in der Autokorrelationszeit bei gleichbleibender Approximationsgüte δ, allerdings dauert ein Sweep ungefähr doppelt so lang, so daß man insgesamt noch einen Gewinn um 250% einfahren kann.

Wählt man n größer als in Relation (5.27) angedeutet, so beraubt man den Korrekturschritt seiner Wirkung zum Preis von unnötig langen Korrelationszeiten $\propto n$. Wählt man n zu klein, so gewinnt der Korrekturschritt an Einfluß auf die Dynamik des Updaters. Dies äußert sich in extrem langen Korrelationsmoden, die man z. B. beim Binning von primären Größen am unangenehmen „Kriechen“ der Standardabweichung in Abhängigkeit von der Blockgröße erkennen kann.

Bei Tests mit Akzeptanzraten um 30% zeigt sich sogar eine Abhängigkeit der integrierten Autokorrelationszeit von der Anzahl der Noisy Correction Steps und der Wahl der Akzeptanzfunktion in der Art, daß mehr Schritte zu kürzeren Korrelationszeiten führen und daß $F_2(z)$ aus (5.6) durchaus günstiger sein konnte. Generell zeigt sich allerdings auch, daß alle Szenarien, bei denen der Korrekturschritt signifikant in die Dynamik eingreift, ökonomisch nicht sinnvoll sind.

5.2.4 32-Bit- versus 64-Bit-Arithmetik

Die numerische Berechnung von Polynomen kann abhängig von der Darstellung über die Wurzeln (3.20) oder durch das Rekursionsschema (3.21) problematisch sein, wenn Rundungsfehler das Ergebnis verfälschen. Die Modellierung dieser Rundungsfehler in Abhängigkeit von der Rechengenauigkeit wird durch den Umstand erleichtert, daß

alle gängigen Rechner-Plattformen dem IEEE-754 Standard für die Zahlendarstellung und für die Fehlertoleranzen bei numerischen Grundrechenarten entsprechen. Dieser Standard unterscheidet grundsätzlich zwischen den beiden Gleitkommatypen *float* (32-Bit) und *double* (64-Bit).

Gerade im Bereich der Simulation von Gittereichtheorien gibt es mit der Quadrics und den QCDSP-Maschinen speziell konstruierte Rechner, die lediglich den 32-Bit-Typ hardwaremäßig unterstützen. Benötigt man mehr Genauigkeit, so muß man auf eine um Größenordnungen langsamere Software-Implementation des Typs *double* zurückgreifen (Zwar wird der Nachfolger der Quadrics, die APEmille, auch 64-Bit-Hardware besitzen, trotzdem werden 32-Bit-Operationen immer noch doppelt so schnell sein, wie ihre 64-Bit-Pendants [B98a].).

Alle kommerziellen RISC- bzw. CISC-Prozessoren besitzen hingegen komplette 64-Bit-Gleitkommaregistersätze, die bei numerischen Operationen genauso schnell arbeiten, wie die analogen 32-Bit-Operationen. Allerdings kann auch hier die Verwendung von 32-Bit-Typen von Vorteil sein, denn im Gegensatz zu kommerziellen Programmen zeichnen sich Simulationen für Gittereichtheorien durch große, nichtlokale Datenmengen aus. Damit stellen sie hohe Ansprüche an die Cache-Strategie und den Speicherbus des Rechners.

Unglücklicherweise besitzt gerade die Cray-T3E im Gegensatz zum üblichen Design von Rechnern mit dem ALPHA-Mikroprozessor keinen dritten Cache. Man hat also auf der einen Seite einen sehr schnellen Prozessor, dem eine schlechte Speicheranbindung gegenüber steht. Es verwundert demnach nicht, daß oftmals der Speicherbus auf einer Cray-T3E die geschwindigkeitsbestimmende Komponente ist (siehe Abbildung D.2). Man kann das Potential des Prozessors besser nutzen, indem man sich auf 32-Bit-Genauigkeit beschränkt, damit im gleichen Zeitraum die doppelte Menge von Zahlen über den begrenzten Speicherbus zum Prozessor gelangen kann.

Schon frühzeitig zeichnete es sich auch in diesem Projekt ab, daß die Performance maßgeblich vom Speicherbus beeinflußt wird. Aus diesem Grund wurde das Simulationsprogramm von Anfang an darauf ausgelegt, mit beiden Gleitkommatypen arbeiten zu können. Ist dies bei einem gewöhnlichen numerischen Projekt dieser Größe schon eine nicht zu unterschätzende Aufgabe, so kommt bei diesem Projekt erschwerend hinzu, daß es Message Passing Code enthalten muß. Die Ein- und Aussprungadressen sämtlicher Kommunikationspuffer sind natürlich vom Zahlentyp und dem entsprechenden Speicherverbrauch abhängig. Die nötigen Anpassungen für die gewählten Typen sollten aber aus Effizienzgründen beim Übersetzen vom Compiler selbst durchgeführt werden, und nicht permanent zur Laufzeit erfolgen. Dies kann mit den C++ - spezifischen Möglichkeiten zur *template-meta*-Programmierung und dem *Trait*-Klassen-Konzept realisiert werden (siehe Anhang E).

In der Tat wird der Updater auf einer T3E-900 um $20\% \sim 30\%$ schneller, indem man nur den numerischen Typ von *double* auf *float* wechselt und alle Programmteile

neu übersetzt. 32-Bit-Arithmetik ist für dieses Projekt auf der T3E attraktiv. Auf einem handelsüblichen PC, der über 1MByte Cache außerhalb des Prozessor verfügt, fällt dieser Unterschied mit $\sim 4\%$ deutlich sanfter aus.

Berechnet man den Wert des Polynoms durch die Produktdarstellung

$$P_n(x) = c_0 \prod_{j=1}^{n} [x - z_j],$$

so hängt die numerische Stabilität entscheidend von der Reihenfolge der Wurzeln z_j ab. Die verschiedenen Ordnungsschemas wurden auf ihre numerische Qualität hin untersucht [BEFJ98]. Für den QCD-Fall $P_n(x) \simeq x^{-1}$ kristallisieren sich zwei Verfahren als am besten geeignet heraus, daß „Bit-reversal“ Verfahren und das „optimized ordering“ Schema [MON98a]. Zur Generierung der Polynome für dieses Projekt wird das letztere Verfahren benutzt.

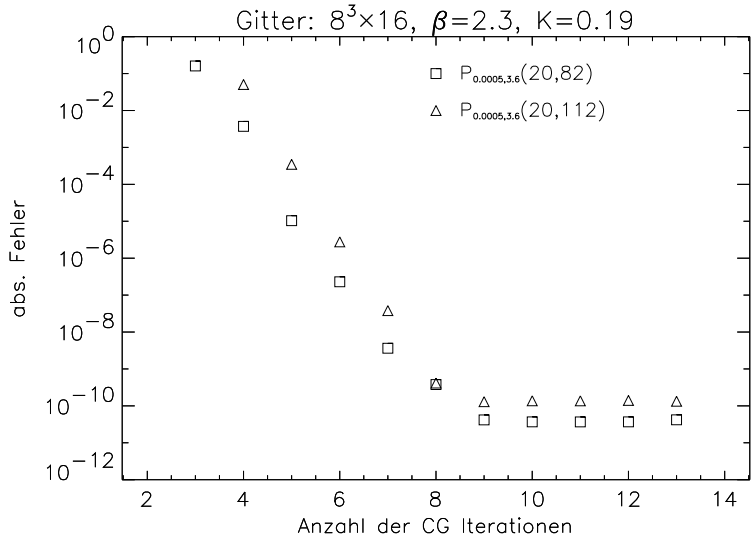


Abbildung 5.3: Absoluter Fehler in der Akzeptanzwahrscheinlichkeit des Korrekturschrittes beim Standardtest auf einer typischen Eichkonfiguration mit doppelter Genauigkeit.

Um die numerischen Fehler im Korrekturschritt bei 64-Bit-Arithmetik zu schätzen, wurde auf thermalisierten Konfigurationen der Standardtest für die Größe

$$A^{(m)}(\eta; [U] \to [U])$$

angewandt. Damit der Fehler des endlichen dritten Polynoms T_l das Ergebnis nicht überlagert, wurde das Inverse $R_m(x)^{-1}$ mit einem konjugierten Gradientenverfahren berechnet. Wie in Abbildung 5.3 zu sehen, erhält man mit 64-Bit-Arithmetik das richtige Ergebnis $A^{(m)}(\eta; [U] \to [U]) = 1$ auch für Polynome R_m mit $m \geq 100$ auf $9 \sim 10$ Stellen, wenn man ausreichend viele Iterationen des Gradientenverfahrens

durchführt, das $R_m(x)^{-1}$ berechnen soll. Insbesondere besitzt die 64-Bit-Arithmetik damit genügend Reserven für die Berechnung deutlich längerer Polynome in der Produktdarstellung.

Anders sieht die Lage bei 32-Bit-Genauigkeit aus, wie Abbildung 5.4 zeigt. Die numerischen Fehler sind vom Hopping-Parameter K abhängig, und zwar solcher Art, daß oberhalb von $K \geq 0.14$ die Produktdarstellung mit 32-Bit-Arithmetik nicht tolerierbare Fehler erzeugt. Zwar kann das „optimized ordering“ Schema den Fehler gegenüber der naiven Anordnung der Monome reduzieren, allerdings nicht im ausreichenden Maße. Zudem stellt sich der numerische Fehler nicht als zusätzliches Rauschen dar, das sich durch häufiges Messen mit verschiedenen Zufallsvektoren η_i herausmitteln würde, sondern auch auf einer einzelnen Konfiguration $[U]$ gilt signifikant

$$< A^{(m)}(\eta; [U] \to [U]) >_\eta \not\approx 1. \tag{5.28}$$

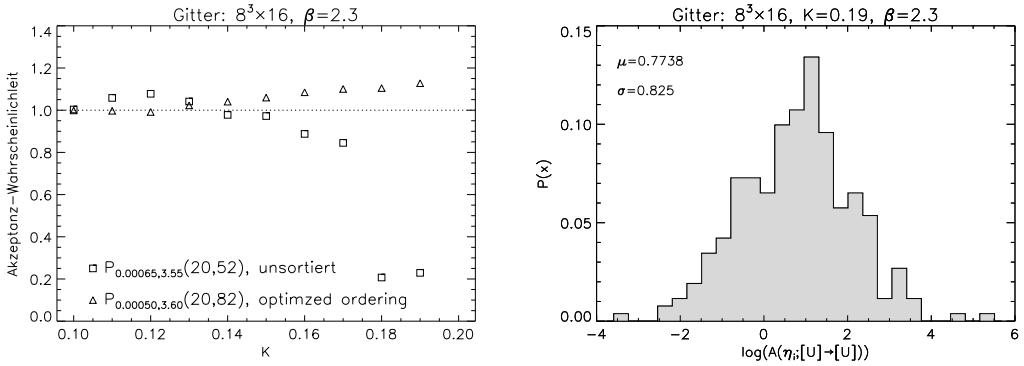


Abbildung 5.4: Standardtest für den Korrekturschritt auf einer typischen Eichkonfiguration mit einfacher Genauigkeit.

Ein optimistisches Bild bei 32-Bit-Arithmetik ergab sich, wenn man das dritte Polynom $T_l(x) \simeq R_m(x)^{-1}$ durch die Rekursionsrelation (3.21) bestimmt. Es überrascht vor allem, daß die Resultate des Standardtests damit besser sind als wenn man $R_m(x)^{-1}$ durch ein konjugiertes Gradientenverfahren mit maximaler Genauigkeit berechnet. Der Grund wird darin zu suchen sein, daß die Matrix, die das Verfahren invertieren muß, selbst wieder durch ein Polynom in Produktdarstellung gegeben ist.

Um die Tauglichkeit der Rekursionsrelation für Produktionsläufe zu testen, wurde ein noch etwas differenzierteres Testverfahren durchlaufen. Grundlage bilden die längsten in dieser Arbeit verwendeten Polynome $P_{0.0001,3.6}(22, 142, 190)$ beim physikalisch interessanten Hopping-Parameter $K = 0.1925$. Für 300 unkorrelierte Konfigurationen wurde nach je einem Update die Größe $A^{(m)}(\eta; [U] \to [U'])$ mit 64-Bit-Genauigkeit gemessen. Dieselben Messungen wurden dann bei 32-Bit-Arithmetik mit Produkt- bzw. Rekursionsdarstellung des dritten Polynoms wiederholt, um dann die

Ergebnisse mit den 64-Bit-Referenzen zu vergleichen. Gemäß Abbildung 5.5 können demnach selbst bei 32-Bit-Arithmetik mit diesen Polynomen in geeigneten Darstellungen Korrekturschritte bis zu Konditionszahlen

$$\kappa(\tilde{\tilde{Q}}^2) \approx \frac{\lambda}{\epsilon} = 3.6 \cdot 10^4 \tag{5.29}$$

mit tolerierbaren Fehlern durchgeführt werden.

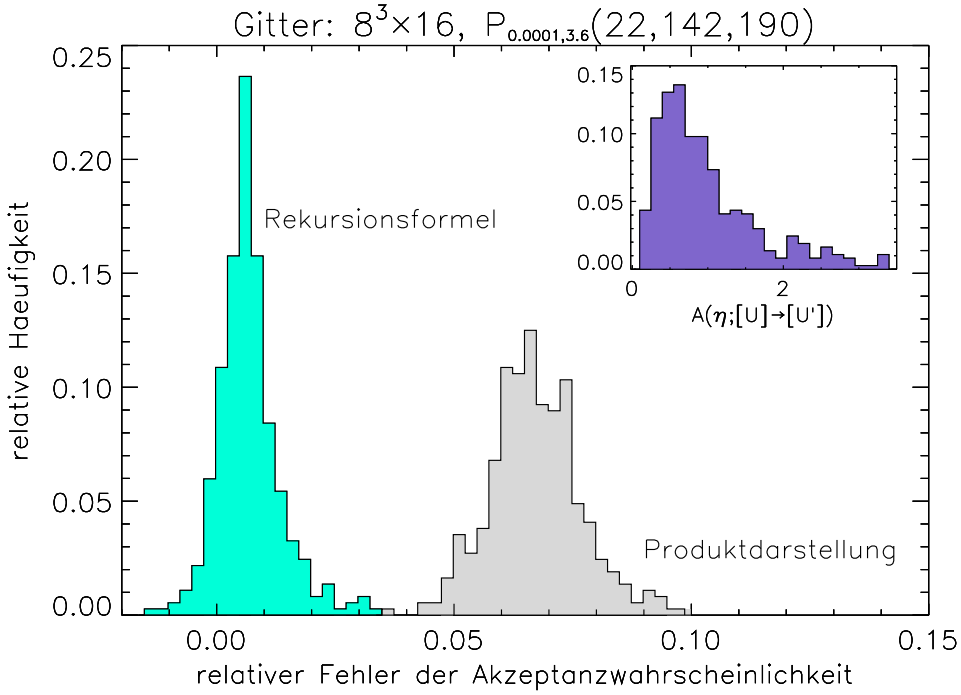


Abbildung 5.5: Relativer Fehler in der Akzeptanzwahrscheinlichkeit des Korrekturschrittes für 300 unkorrelierte Konfigurationen bei einfacher Rechengenauigkeit in Abhängigkeit vom Berechnungsverfahren für das dritte Polynom. Die Simulationsparameter waren $K = 0.1925, \beta = 2.3$, das Update bestand aus einem Heatbath-, drei Overrelaxation- und einem Metropolis-Sweep mit 20 Hits auf jedem Link.

5.2.5 Grenzen des Verfahrens

Die Zwei-Schritt-Approximation stellt natürlich einen Fortschritt gegenüber dem ursprünglichen Algorithmus dar. Allerdings kann auch mit dem Zwei-Schritt-Verfahren nicht verhindert werden, daß es systematische Fehler bei der Behandlung der kleinen Eigenwerte von $\tilde{Q}[U]^2$ gibt.[2] Vergleicht man dazu die Güte der Zwei-Schritt-Approximationen von typischen Produktionspolynomen $P_n(x)R_m(x)$ am unteren Ende des Spektrums in Abbildung 5.6 mit den Verteilungen der kleinsten Eigenwerte aus Kapitel 4.2.3, so wird klar, daß es immer Ausreißer geben wird, für die

[2]Es sei nochmals darauf hingewiesen, daß alle folgenden Aussagen ebenfalls für die präkonditionierte Matrix $\tilde{\tilde{Q}}[U]^2$ gelten.

diese Approximation nicht akzeptierbare Fehler erzeugt.

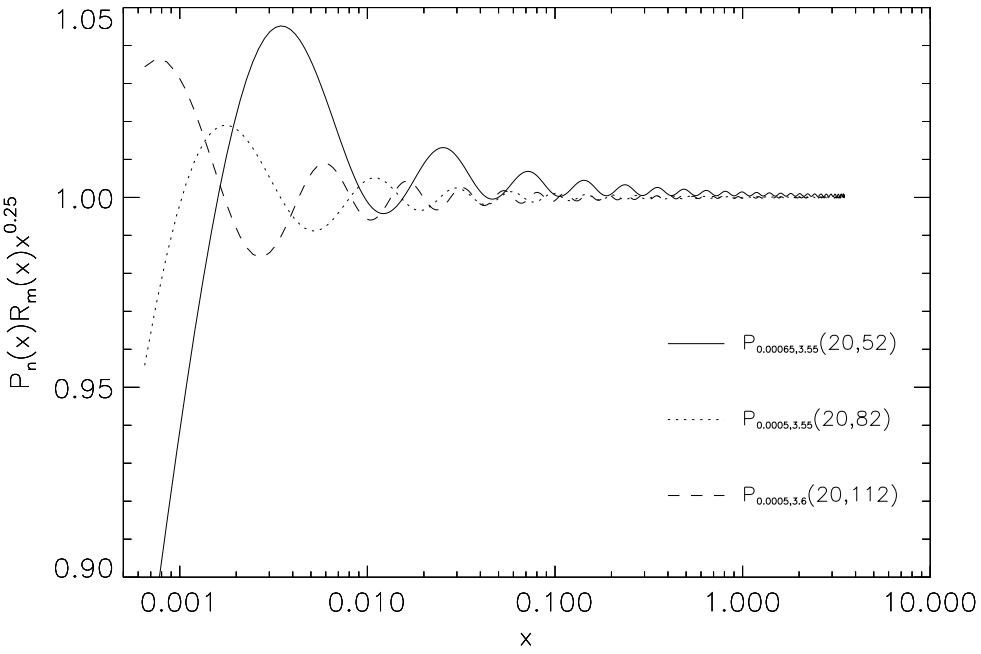


Abbildung 5.6: Gütefaktor $P_n(x)R_m(x)x^{1/4}$ der Zwei-Schritt-Approximation.

Kennt man alle Eigenwerte λ_i der Matrix $\tilde{Q}[U]^2$, so kann der Fehler des Zwei-Schritt-Verfahrens durch einen Korrekturfaktor ausgedrückt werden

$$\left[\det\left(\tilde{Q}^2\right)\right]^{\frac{1}{4}} \overset{k\to\infty}{=} \frac{1}{\det[P_n(\tilde{Q}^2)]\det[R_m(\tilde{Q}^2)]} \underbrace{\prod_{i=1}^{k} P_n(\lambda_i)R_m(\lambda_i)\lambda_i^{\frac{1}{4}}}_{\text{Korrekturfaktor}}. \tag{5.30}$$

Allerdings ist es nicht praktikabel, alle Eigenwerte λ_i der Matrix $\tilde{Q}[U]^2$ zu bestimmen; für die kleinsten $O(10 \sim 100)$ Eigenwerte hingegen sind effiziente Algorithmen bekannt. Es ist nach Abbildung 5.6 auch ausreichend, wenn man das Produkt in Gleichung (5.30) für ein endliches k abbricht, so daß nur diejenigen Eigenwerte berücksichtigt werden, die kleiner als $\sim 40\epsilon$ sind.

5.3 Korrektur für die k kleinsten Eigenwerte

Die Idee ist nun, durch den Zwei-Schritt-Algorithmus unabhängige Konfigurationen generieren zu lassen, von denen man anschließend die Korrekturfaktoren A_i berechnet. Diese fließen als Meßkorrektur in die Erwartungswerte aller Observablen W ein

$$\hat{W} = \frac{\sum_i A_i w_i}{\sum_i A_i}. \tag{5.31}$$

Man sollte sich von der Meßkorrektur aber nicht dazu verleiten lassen, zu kurze Polynome P_n, R_m zu benutzen. Als Folge davon würden die Korrekturfaktoren A_i deutlich von eins verschieden sein und damit ein zusätzliches Rauschen auf das Signal w_i aufmodulieren. Im Extremfall könnte w_i völlig in diesem Rauschen untergehen.

5.3.1 Modifizierter Kalkreuter-Simma-Algorithmus

Der einfachste sequentielle Algorithmus zur Berechnung der k kleinsten Eigenwerte einer hermiteschen Matrix A besteht im wesentlichen aus dem konjugierten Gradientenverfahren zur Berechnung des kleinsten Eigenwertes λ_{min} und des dazugehörigen Eigenvektors v_{min} (siehe Abschnitt 4.2.1). Hat man zunächst λ_{min} und v_{min} gefunden, so definiert man sich den Projektor

$$P^{\perp} = \mathbb{1} - v_{min} < v_{min}, \cdot >, \tag{5.32}$$

um damit den zweitkleinsten Eigenwert zu berechnen, indem man den kleinsten Eigenwert von $P^{\perp}AP^{\perp}$ bestimmt. Man kann dieses Verfahren beliebig oft wiederholen und erhält den k kleinsten Eigenwert durch Bestimmung des kleinsten Eigenwertes von

$$A_k = P_k^{\perp} A P_k^{\perp} \quad \text{mit} \quad P_k^{\perp} = \mathbb{1} - \sum_{i=1}^{k-1} v_i < v_i, \cdot > . \tag{5.33}$$

Allerdings zeigte sich für die Matrizen $\tilde{Q}^2$ und $\bar{\tilde{Q}}^2$, daß dieses Verfahren schon nach wenigen Eigenwerten numerisch instabil wird. Der Hauptgrund liegt in den Projektoren $P_k^{\perp}$, welche, bedingt durch die Eigenvektoren, immer nur mit endlicher Genauigkeit bekannt sind.

Man kann diesen sequentiellen Algorithmus numerisch stabilisieren und beschleunigen, indem man ihn in ein iteratives Schema einbettet, bei dem nach jedem sequentiellen Durchlauf die $(k \times k)$-Matrix

$$M_{ij} =< v_i, A v_j > \tag{5.34}$$

diagonalisiert wird und die resultierenden Eigenvektoren zur Berechnung besserer Schätzer für die Eigenvektoren v_i von M benutzt werden, wie Kalkreuter und Simma in [KS96] vorschlagen.

Dazu werden beim ersten Durchlauf die k Eigenvektoren v_i jeweils nur sehr ungenau mit wenigen Iterationsschritten bestimmt, um daraus die Matrix M zu berechnen. Diese Matrix ist sehr klein und ihre Diagonalisierung performance-unkritisch. In der Implementation für die T3E wird das mit Hilfe eines QR-Algorithmus erledigt. Die resultierenden Eigenvektoren $\xi^{(i)}$ werden normiert und so umsortiert, daß die zugehörigen Eigenwerte eine monoton wachsende Folge in i bilden. Verbesserte Schätzer v_i' für die Eigenvektoren von A können damit aus

$$v_i' = \sum_{j=1}^{k} \xi_j^{(i)} v_j \quad \text{mit} \quad i = 1, ...k \tag{5.35}$$

gewonnen werden, wobei mit $\xi_j^{(i)}$ die j-te Komponente des i-ten normierten Eigenvektors von M gemeint ist. Nach diesem Diagonalisierungsschritt setzt man die konjugierten Gradientenverfahren mit den neuen Startvektoren v_i' weiter fort, um dann nach einigen weiteren Schritten wieder einen Diagonalisierungsschritt einzufügen.

Allerdings leidet die Effizienz des konjugierten Gradientenverfahrens unter diesen Unterbrechungen, da jedes Mal eine neue, gute Suchrichtung gefunden werden muß. Deshalb sollte die Anzahl der Suchiterationen zwischen den Diagonalisierungsschritten sorgfältig ausbalanciert werden. Insbesondere sollte die Anzahl nicht konstant sein, denn am Anfang profitiert dieser Algorithmus besonders von den zwischenzeitlichen Diagonalisierungen, während zum Ende hin der „Feinschliff" vom konjugierten Gradientenverfahren geleistet werden muß. Dieses braucht jedes Mal etliche Iterationen, bis eine optimale Suchrichtung gefunden wird. Für dieses Projekt hat sich bewährt, zwischen dem i und dem $i+1$ Diagonalisierungsschritt $5 + 10 \cdot i$ Suchiterationen des konjugierten Gradientenverfahrens durchzuführen. Zudem hat sich für die Bestimmung der Suchrichtung wieder die Polak-Ribiere-Variante gegenüber der in [KS96] vorgeschlagenen Fletcher-Reeves-Methode bewährt. Die Suchiteration wird wie üblich abgebrochen, wenn das Residuum einen gewissen Schwellwert unterschreitet.

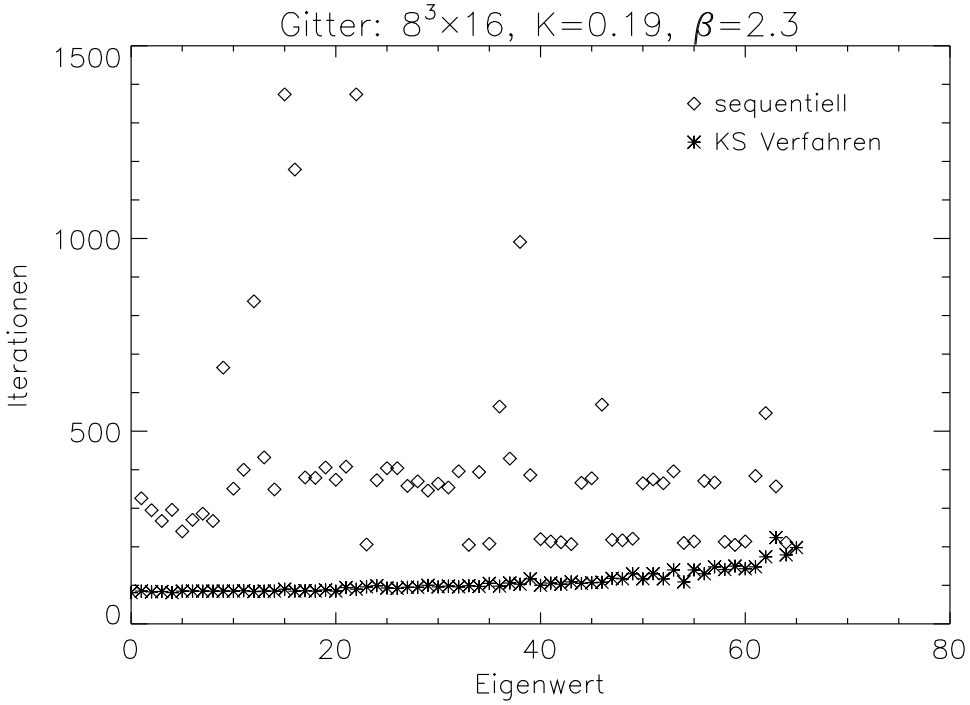


Abbildung 5.7: Vergleich des modifizierten Kalkreuter-Simma-Verfahrens mit dem naiven sequentiellen Algorithmus bei der Bestimmung der 64 kleinsten Eigenwerte von $\tilde{Q}^2$ auf einer typischen Eichkonfiguration. Abbruchkriterium für das Gradientenverfahren war 10^{-8}.

Die Anwendung des Projektionsoperators $P_k^{\perp}$ kann schon für moderate k-Werte aufwendiger sein als die Multiplikation mit der Fermion-Matrix $\tilde{Q}^2$. Da $P_k^{\perp}$ aber näherungsweise Eigenräume von A herausprojiziert, muß man ihn nicht bei jeder Suchiteration berücksichtigen. Denn ist der schon bestimmte Eigenraum einmal aus der Suchrichtungen herausprojiziert, kann das konjugierte Gradientenverfahren nur durch numerische Fehler wieder in diesen Raum zurückfallen. Tests zeigen, daß es ausreicht, wenn der Projektionsoperator nur bei den Diagonalisierungsschritten und bei jeder 25. Gradienteniteration angewandt wird.

Wie die Abbildung 5.7 zeigt, hat sich die Implementierung und Optimierung des Kalkreuter-Simma-Algorithmus gelohnt. Es ist interessant zu sehen, daß die Anzahl der Suchiterationen pro Eigenwert im Mittel um so kleiner wird, je mehr Eigenwerte der Algorithmus insgesamt berechnet. Will man mehr als 16 Eigenwerte berechnen, kann es von Vorteil sein, den Algorithmus mit 5% mehr Eigenwerten laufen zu lassen, um dann abzubrechen, wenn die gesuchten Eigenwerte das Abbruchkriterium erfüllen.

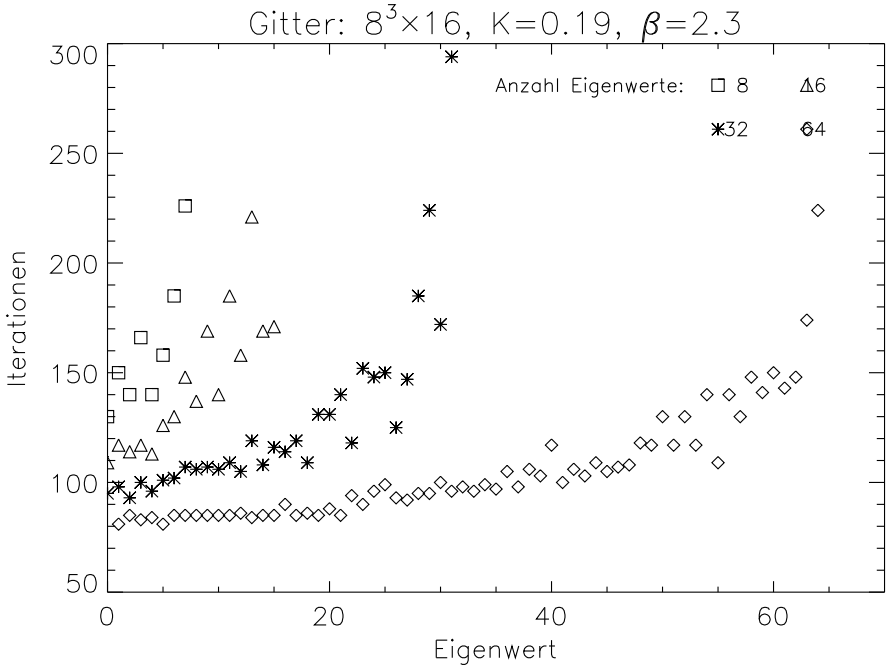


Abbildung 5.8: Konvergenzgeschwindigkeit des modifizierten Kalkreuter-Simma-Verfahrens in Relation zur Anzahl der berechneten, kleinsten Eigenwerte von $\tilde{\tilde{Q}}^2$ auf einer typischen Eichkonfiguration. Abbruchkriterium für das Gradientenverfahren war 10^{-8}.

5.3.2 Spektrale Dichte für die kleinsten Eigenwerte

Eine realistische Anzahl von Eigenwerten, die man mit dem Kalkreuter-Simma-Algorithmus noch berechnen kann, ist $32 \sim 64$. Begrenzender Faktor ist zum einen

die CPU-Zeit und zum anderen auf größeren Gittern der Hauptspeicher, der zur Bearbeitung einer noch größeren Anzahl von Eigenvektoren nicht ausreichen würde.

In einer Teststudie wurden von 167 unabhängigen Konfigurationen, zu deren Generierung die Polynome $P_{0.0005,3.7}(20, 82, 112)$ benutzt wurden, die 32 kleinsten Eigenwerte von $\tilde{\tilde{Q}}^2$ bestimmt. Mit einer geeignet diskretisierten Version der spektralen Eigenwertdichte

$$\rho(\lambda) =< \sum_i \delta(\lambda - \lambda_i) > \tag{5.36}$$

kann man einen Schätzer für die kumulierte Eigenwertdichte durch Aufintegration von $\hat{\rho}(\lambda)$ erhalten. Das Ergebnis für diese Stichprobe ist in Abbildung 5.9 zu sehen. Vergleicht man dieses mit der Güte der zugrundeliegenden Polynomapproximation aus Abbildung 5.6, so zeigt sich, daß 32 Eigenwerte nicht für eine verläßliche Korrektur ausreichen. Im Mittel können die 24 kleinsten Eigenwerte nur die Approximationsfehler bis $x = 0.007$ beheben. Die kumulierte Eigenwertdichte steigt zudem viel zu stark an, um alle benötigten Eigenwerte bis zur Grenze $x \approx 40\epsilon = 0.02$, ab der man die Polynomapproximation nicht mehr korrigieren will, berechnen zu können. Ein weiteres Indiz für die nur unzureichende Berechnung der Korrekturfaktoren ist der Umstand, daß diese noch stark von k abhängen, und zwar unterscheiden sich in diesem Fall die Faktoren aus 32 Eigenwerten noch um ca. 6% von denen aus 64 Eigenwerten.

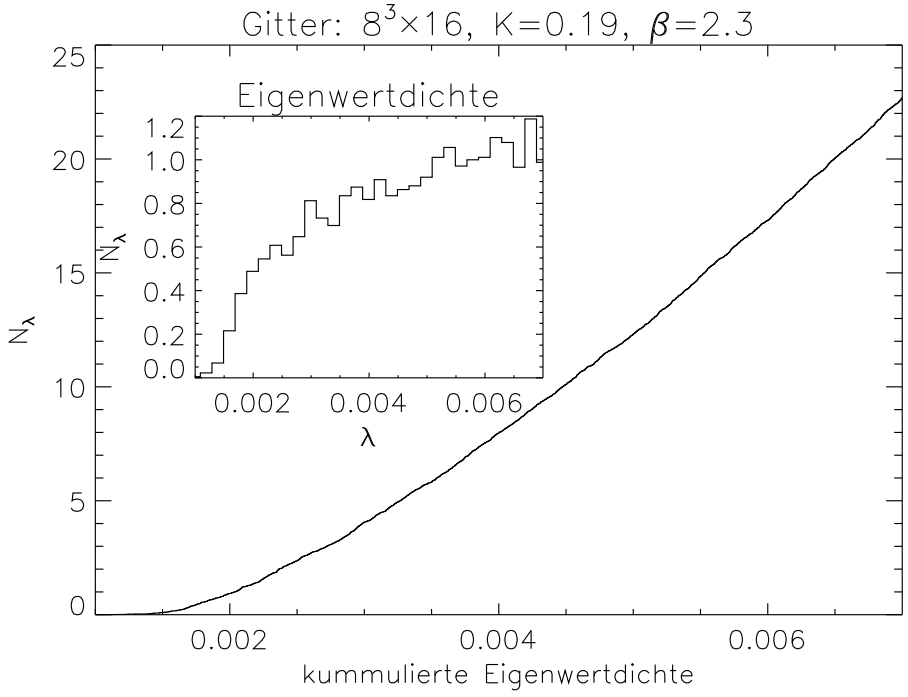


Abbildung 5.9: Mittlere Eigenwertdichte von 167 Eichkonfigurationen.

Alternativ zu diesem Verfahren hat I. Montvay den „subspace iteration“ Algorithmus

von B. Bunk in einer Fortran90 Version implementiert [BUN97]. Dieser Algorithmus ist in der Effizienz dem optimierten Kalkreuter-Simma-Algorithmus allerdings um ca. 30% unterlegen.

5.4 Hybrid-Korrekturschritt

I. Montvay hat einen anderen Zugang für die Berechnung der Korrekturfaktoren vorgeschlagen. Wiederum soll ein noch längeres, quadratisch optimiertes Polynom V_c mit $c > l > m > n$ die Fehler der anderen Polynome durch einen Korrekturfaktor $A[U]$ ausmerzen. Startpunkt für diese Überlegungen bildet die Definition des Erwartungswertes für eine Observable $W[U]$

$$< W[U] >_U = \frac{\int [dU] e^{-S_g[U]} \det(\tilde{Q}[U]^2)^{\frac{1}{4}} W[U]}{\int [dU] e^{-S_g[U]} \det(\tilde{Q}[U]^2)^{\frac{1}{4}}}, \tag{5.37}$$

wobei sich die Fermion-Determinante für ein unendlich langes Polynom

$$V_c(x) \stackrel{c\to\infty}{=} \left\{ x^{\frac{1}{4}} P_n(x) R_m(x) \right\}^{-1} \tag{5.38}$$

bei endlichem n, m exakt durch bosonische Determinanten ausdrücken läßt

$$(\det \tilde{Q}[U]^2)^{\frac{1}{4}} \stackrel{c\to\infty}{=} \frac{1}{\det P_n(\tilde{Q}[U]^2) \det R_m(\tilde{Q}[U]^2) \det V_c(\tilde{Q}[U]^2)} \tag{5.39}$$

$$= \frac{A[U]}{\det P_n(\tilde{Q}[U]^2) \det R_m(\tilde{Q}[U]^2)}. \tag{5.40}$$

Prinzipiell ist das Approximationsintervall von $V_c(x)$ nicht auf das der beiden anderen Polynome beschränkt; insbesondere ist $]0, \lambda]$ möglich, so daß man auch Eigenwerte korrigieren kann, die gar nicht mehr im ursprünglichen Intervall $[\epsilon, \lambda]$ liegen. Die Berechnung des Korrekturfaktors kann wie gehabt über einen „Noisy Estimator" des bosonischen Pfadintegrals

$$\begin{aligned} A[U] &= \frac{1}{\det V_c(\tilde{Q}[U]^2)} \\ &= \frac{\int [d\eta^\dagger][d\eta] e^{-\eta^\dagger \eta} \exp\left\{ \eta^\dagger [1 - V_c(\tilde{Q}[U]^2)] \eta \right\}}{\int [d\eta^\dagger][d\eta] e^{-\eta^\dagger \eta}} \\ &= < \exp\left\{ \eta^\dagger [1 - V_c(\tilde{Q}[U]^2)] \eta \right\} >_\eta \end{aligned} \tag{5.41}$$

erfolgen, d. h. im Rahmen einer Monte-Carlo-Simulation, für die lediglich normalverteilte Zufallsvektoren η_i benötigt werden.

Die Funktion $V_c(x)$ ist über einen weiten Bereich sehr glatt, allerdings wird sie für kleine x sehr große Funktionswerte annehmen. Eine reine Monte-Carlo-Integration ist für solch eine Funktion eine eigentlich eher ungeeignete Methode, und man muß,

wie Tests zeigten, mit einer großen Streuung von $A[U]_i$ innerhalb einer Stichprobe $\{\eta_i\}$ rechnen. Dieses Rauschen überträgt sich via der Definition des gewichteten Mittelwertes (5.31) auch auf die eigentlichen Meßgrößen.

Hinzu kommt, daß dieses Verfahren mit Ausnahmekonfigurationen (siehe Abschnitt 7.1.5), d. h. mit Konfigurationen, bei denen Eigenwerte im Bereich 10^{-7} bis 10^{-12} liegen, nicht zurecht kommt, denn man kann schwerlich Polynome V_c mit sinnvollem, endlichem c erzeugen, die für solche Werte noch tolerierbare Fehler zeigen. Darüber hinaus wird der Schätzer $A[U]_i$ auf diesen Konfigurationen enorme Schwankungen aufweisen. Gerade bei fermionischen Größen ist man aber auf einen fehlerfreien Korrekturfaktor für Ausnahmekonfigurationen angewiesen, da diese Größen auf solchen Konfigurationen typischerweise Werte annehmen, die um etliche Größenordnungen vom Mittelwert abweichen. Werden solche Ausreißer durch einen verrauschten Korrekturfaktor nicht energisch genug unterdrückt, können sie die Ergebnisse eines ganzen Laufes in Mitleidenschaft ziehen.

Der Trick besteht nun darin, beide Korrekturverfahren zu einem hybriden Korrekturverfahren zu kombinieren. Dabei übernimmt die Korrektur für die kleinsten Eigenwerte den Part von $V_c(x)$, der für eine Monte-Carlo-Simulation nur schwer zugänglich ist, während diese für das Gros der Eigenwerte, bei denen die Funktion $V_c(x)$ sehr glatt ist, die Korrektur vornimmt

$$A[U] \simeq \underbrace{\prod_{\lambda<\lambda_G} P_n(\lambda) R_m(\lambda) \lambda^{\frac{1}{4}}}_{\substack{\text{Korrekturfaktor durch} \\ \text{die kleinsten Eigenwerte}}} \cdot \underbrace{\prod_{\lambda>\lambda_G} V_c(\lambda)^{-1}}_{\substack{\text{Korrekturfaktor durch} \\ \text{Monte-Carlo-Simulation}}} \quad . \tag{5.42}$$

Mit der Grenze λ_G definiert man diejenige Stelle, ab der die Funktion $V_c(x)$ glatt genug ist, um von einem Monte-Carlo-Algorithmus integriert zu werden. Für ein relativ kleines c kann es auch diejenige Stelle sein, ab der die Approximation eine gewisse Mindestqualität erreicht.

Anzahl Eigenwerte	λ_G	Korrekturfaktoren		
		Eigenwerte	Noisy Estimator	Gesamt
0	$\leq 2.0 \cdot 10^{-3}$	1.0000	0.9482(85)	0.9482(85)
16	$6.0 \cdot 10^{-3}$	0.9361	1.0276(76)	0.9618(76)
32	$8.5 \cdot 10^{-3}$	0.9089	1.0540(68)	0.9579(68)

Tabelle 5.1: Gewichtungsfaktoren aus dem hybriden Korrekturalgorithmus bei verschiedenen λ_G für eine typische Konfiguration auf $8^3 \times 16$ bei $K = 0.19, \beta = 2.3$ mit $P_{0.0005,3.55}(20, 82, 112)$ und $V_{180}(\lambda_{min} = 0)$.

Die Implementation des Algorithmus sieht dementsprechend vor, daß in einem ersten Schritt alle Eigenwerte und Eigenvektoren bis λ_G mit dem Kalkreuter-Simma-Algorithmus bestimmt werden (Die Eigenwerte werden jeweils in Vierer- oder Achter-Gruppen berechnet, bis λ_G überschritten wird.). In der anschließenden Monte-Carlo-Simulation werden die Eigenräume mittels des Projektors $P_k^{\perp}$ aus den Zufallsvektoren η_i herausprojiziert. Zusätzlich muß noch während der Berechnung von $V_c(x)$

nach jeder 25. Multiplikation mit $\tilde{Q}^2$ der Eigenraum $P_k^{\perp}$ aus den Zwischenergebnissen herausprojiziert werden, um ein „Zurückfallen“ durch numerische Ungenauigkeiten zu verhindern.

Anzahl Eigenwerte	σ_A	Zeitdauer für 10 Noisy Estimator & Eigenwerte
0	0.415	154 sek.
2	0.173	173 sek.
4	0.134	191 sek.
8	0.127	224 sek.

Tabelle 5.2: Streuung der Noisy Estimator in Abhängigkeit von der Anzahl der zuvor berechneten Eigenwerte für eine typische Konfiguration auf $6^3 \times 12$ bei $K = 0.195, \beta = 2.3$ mit $P_{0.00003,3.7}(22, 66, 102)$ und $V_{400}(\lambda_{min} = 0)$.

Wie in Tabelle 5.1 an einem Beispiel gezeigt, ist dieser Algorithmus gegen das Verschieben von λ_G stabil, solange $V_c(\lambda_G)$ noch eine sinnvolle Approximation erlaubt. Zudem erzeugen die kleinsten Eigenwerte einen entscheidenden Anteil des Rauschens im Noisy Estimator. Aus diesem Grund ist es generell günstiger, erst die $4 \sim 8$ kleinsten Eigenwerte zu berechnen und dann die Monte-Carlo-Simulation durchzuführen.

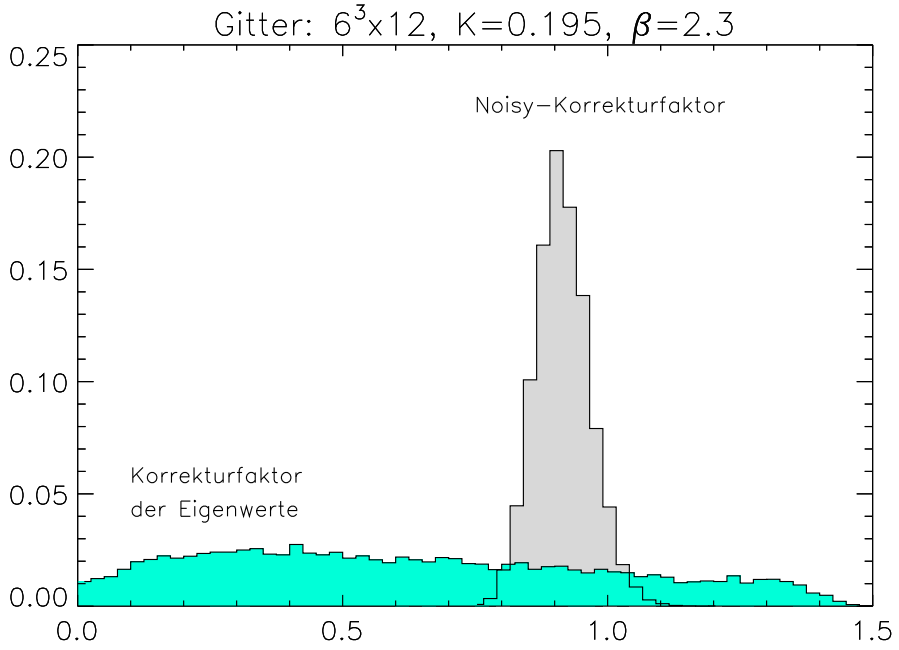


Abbildung 5.10: Verteilung der Korrekturfaktoren aus den Eigenwerten $\lambda_G < 0.0008$ und dem nachgeschaltetem Noisy Correction Step mit $V_{400}(\lambda_{min} = 0)$ basierend auf 13000 Konfigurationen. Zur Erzeugung der Konfigurationen wurden die Polynome $P_{0.00003,3.7}(22, 66, 102)$ verwendet.

In Abbildung 5.10 sind die Resultate für die 13000 Korrekturfaktoren eines Produktionslaufes dargestellt. Die Polynome waren bei diesem Lauf, gemessen am Approximationsintervall, sehr kurz, so daß die Korrekturfaktoren deutlich von eins abweichen. Man erkennt, daß der Korrekturfaktor der Eigenwerte die grobe Arbeit übernimmt, während die Monte-Carlo-Simulation den Feinschliff anbringt. Insbesondere konnten hierbei sämtliche Ausnahmekonfigurationen durch den hybriden Algorithmus mit kleinen Fehlern korrigiert werden.

Wie zu erwarten, weisen fermionische Observablen, die durch Inversion der Fermion-Matrix berechnet werden, auf Konfigurationen, die durch den Korrekturfaktor stark unterdrückt werden, Meßwerte auf, welche deutlich vom Mittelwert abweichen. Ursache hierfür sind die durch die Polynomapproximation $P_n(x)R_m(x)$ bedingten zu kleinen Eigenwerte.

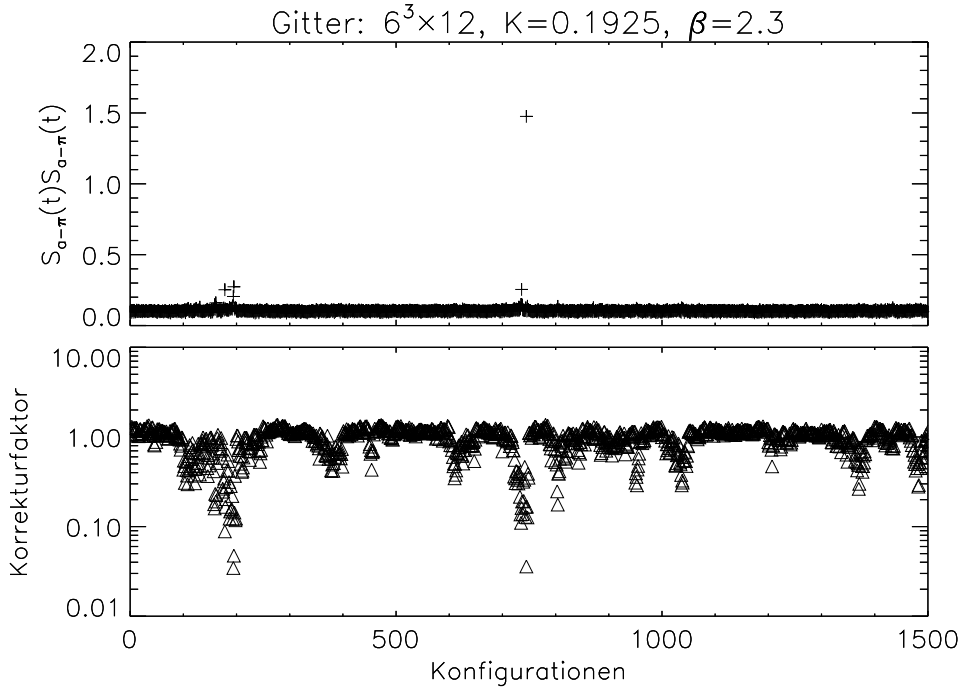


Abbildung 5.11: Korrekturfaktoren von 1500 Konfigurationen $P_{0.00003,3.7}(22, 66, 102)$ und der Wert des jeweiligen a-π Propagators Tr $(\gamma_5 Q^{-1}_{yx} \gamma_5 Q^{-1}_{xy})$ für den Zeitscheibenabstand $\triangle t = 0$. Polynome: $P_{0.00003,3.7}(22, 66, 102)$ und $V_{400}(\lambda_{min} = 0)$.

Kapitel 6

Massenbestimmung auf dem Gitter

6.1 Zeitscheibenkorrelationsfunktion

Mit dem vorgestellten Monte-Carlo-Algorithmus ist man nun in der Lage, Konfigurationen mit sehr großen Konditionszahlen der Fermion-Matrix, d. h. mit kleinen Gluino-Massen und damit (hoffentlich) nahe dem supersymmetrischen Limes zu erzeugen. Auf diesen Konfigurationen möchte man natürlich die Massen derjenigen Teilchen bestimmen, die nach den theoretischen Arbeiten (s. Kapitel 1.3.2) die chiralen Multipletts der effektiven Wirkung aufbauen, denn die Restaurierung der Supersymmetrie auf dem Gitter erfordert, daß die Massen innerhalb der Multipletts gleich werden. Ein weiterer wichtiger Punkt ist, die Massenaufspaltung innerhalb der Multipletts für weich gebrochene Supersymmetrie zu untersuchen.

Massen werden in Gittersimulationen durch das asymptotische Verhalten der Korrelationsfunktionen

$$\Gamma_{S(t)S(t+\Delta t)} =< S(t+\Delta t)^{\dagger} S(t) > \tag{6.1}$$

zwischen den Zeitscheibenerwartungswerten

$$S(t,\mathbf{p}) = \frac{1}{L^{3/2}} \sum_{x} e^{-i\mathbf{p}\mathbf{x}} \phi(\mathbf{x},t) \tag{6.2}$$

für den Fall $S(t) \stackrel{\text{def}}{=} S(t,\mathbf{p}=0)$ in der Euklidischen Raum-Zeit bestimmt

$$\begin{aligned} \Gamma_{S(t)S(t+\Delta t)} \stackrel{L_t \to \infty}{=} \sum_{n} \Big\{ & \quad | < n|S(t)|0 > |^2 e^{-E_n \Delta t} \\ & \pm \; | < 0|S(t)^{\dagger}|n > |^2 e^{-E_n (L_t - \Delta t)} \Big\}. \end{aligned} \tag{6.3}$$

Dabei ist $\{|n>\}$ ein kompletter Satz von Energieeigenfunktionen. Das Vorzeichen bezieht sich auf periodische $(+)$ bzw. antiperiodische $(-)$ Randbedingungen für den Operator $\phi(\mathbf{x},t)$ in Zeitrichtung. Für Abstände $\Delta t \approx L_t/2$ wird die Zeitscheibenkorrelationsfunktion vom leichtesten Zustand $|1>$ mit der Energie E_1 dominiert, den

der Operator $\phi(\mathbf{x},t)$ aus dem Vakuum erzeugen kann

$$\Gamma_{S(t)S(t+\Delta t)} \overset{\substack{L_t\to\infty \\ \Delta t\to L_t/2}}{\sim} \begin{cases} c_0 + c_1 \cosh\left[m_1\left(\frac{L_t}{2}-\Delta t\right)\right] & \text{periodische Randbed.} \\ c_1 \sinh\left[m_1\left(\frac{L_t}{2}-\Delta t\right)\right] & \text{antiperiodische Randbed.} \end{cases} \tag{6.4}$$

Die Konstante c_0 ist nur dann von Null verschieden, wenn $< S(t) > \neq 0$ gilt. Man kann demnach die Masse des leichtesten Zustands durch einen Fit der Zeitscheibenkorrelationsfunktion in einem Intervall $\Delta t \in [t_1, t_2]$ bestimmen. Um den statistischen Fehler zu ermitteln, wird wieder das Jackknife-Verfahren bemüht. Allerdings ist das Ergebnis nur eine effektive Masse $m_1(L_t, t_1, t_2)$, welche erst nach dem Grenzübergang $L_t \to \infty$ und für das Fit-Intervall $\Delta t \in [L_t/2-1, L_t/2+1]$ der wirklichen Masse m_1 entspricht. Vergrößert man das Fit-Intervall, so kann man versuchen, die nächst höhere Masse m_2 abzuschätzen

$$\Gamma_{S(t)S(t+\Delta t)} \overset{\substack{L_t\to\infty \\ \Delta t\to L_t/2}}{\sim} \begin{cases} c_0 + c_1 \cosh\left[m_1\left(\frac{L_t}{2}-\Delta t\right)\right] + c_2 \cosh\left[m_2\left(\frac{L_t}{2}-\Delta t\right)\right] \\ c_1 \sinh\left[m_1\left(\frac{L_t}{2}-\Delta t\right)\right] + c_2 \sinh\left[m_2\left(\frac{L_t}{2}-\Delta t\right)\right]. \end{cases} \tag{6.5}$$

Allerdings reicht oftmals die unmittelbare Umgebung von $L_t/2$ für einen Fit nicht aus, da gerade in diesem Bereich das Signal/Rausch-Verhältnis am ungünstigsten ist. Teilweise kann es sogar nicht einmal sinnvoll sein, daß $L_t/2$ im Fit-Intervall liegt. Trotzdem muß man evtl. einen Fit gemäß Relation (6.4) durchführen, da für noch mehr Parameter nicht genügend verläßliche Punkte von $\Gamma_{S(t)S(t+\Delta t)}$ zur Verfügung stehen. Damit wird die effektive Masse $m_1(L_t, t_1, t_2)$ Anteile von m_2 oder noch höheren Massen enthalten. Dieser Einfluß kann minimiert werden, indem man für den gewünschten Teilchenkanal einen Operator $S(t)$ wählt, der das Vakuum besonders gut auf den Zustand $|1>$ abbildet, im optimalen Fall

$$< n|S(t)|0 > \sim \delta_{n,0}. \tag{6.6}$$

Insbesondere wird für einen solchen Operator das Verhältnis $c_2/c_1 = 0$, und die effektive Masse $m_1(L_t \to \infty, t_1, t_2)$ stimmt für alle t_1, t_2 mit der wirklichen Masse überein. Die Konstruktion dieses Operators erfordert allerdings die Kenntnis der Wellenfunktion des Grundzustandes in der Ortsdarstellung, deren Berechnung sehr aufwendig ist [CRE92]. Zudem verlangt sie die Kenntnis der Masse.

Aus diesem Grund wurden eine ganze Reihe von sogenannten Smearing-Algorithmen entworfen, die es erlauben, in kurzer Zeit geeignete Operatoren $S(t)$ zu berechnen. Das beste Verfahren ist grundsätzlich dasjenige, welches die kleinsten Massen $m_1(L_t, t_1, t_2)$ ergibt.

6.2 Glueball Spektrum

6.2.1 Observablen

Der leichteste Glueball sollte derjenige mit Spin $J = 0$ und positiver Parität sein. Um die Masse zu bestimmen, könnte man im Rahmen eines naiven Ansatzes die

Plaquette-Plaquette-Korrelationsfunktion studieren. Allerdings sollte der Operator zu einer irreduziblen Darstellung der kubischen Gruppe auf dem Gitter gehören und im Kontinuumslimes an $J = 0$ koppeln. Ein solcher Operator ist z. B.

$$S_{0^+}(t) = \mathrm{Re} \sum_{\mathbf{x}} \left(U_{x,12} + U_{x,23} + U_{x,31} \right). \tag{6.7}$$

Der 2^+ Glueball hat keine unmittelbare physikalische Relevanz für die Restauration der Supersymmetrie. Er wurde trotzdem mit ins Programm aufgenommen, da die Implementation kaum mehr Aufwand bedeutete, wenn eine 0^+ Routine vorhanden ist. Geeignete Observablen für diesen Zustand sind

$$S_{2^+}(t) = \mathrm{Re} \sum_{\mathbf{x}} (U_{x,13} - U_{x,23}) \quad \vee \quad S_{2^+}(t) = \frac{1}{\sqrt{3}} \mathrm{Re} \sum_{\mathbf{x}} (U_{x,13} + U_{x,23} - 2U_{x,12}). \tag{6.8}$$

Die Autokorrelationszeiten von $\Gamma_{S(t)S(t+\Delta t)}$ sind von derselben Größenordnung wie die der Plaquette. Zudem sind die naiven Operatoren für die Zeitscheibenkorrelationsfunktion lokal, so daß man auch bei relativ großen Glueball-Massen mit einer schlechten Projektion auf den Grundzustand rechnen muß. Dies kann man verbessern, indem man die Links vor der Messung eichinvariant „verschmiert".

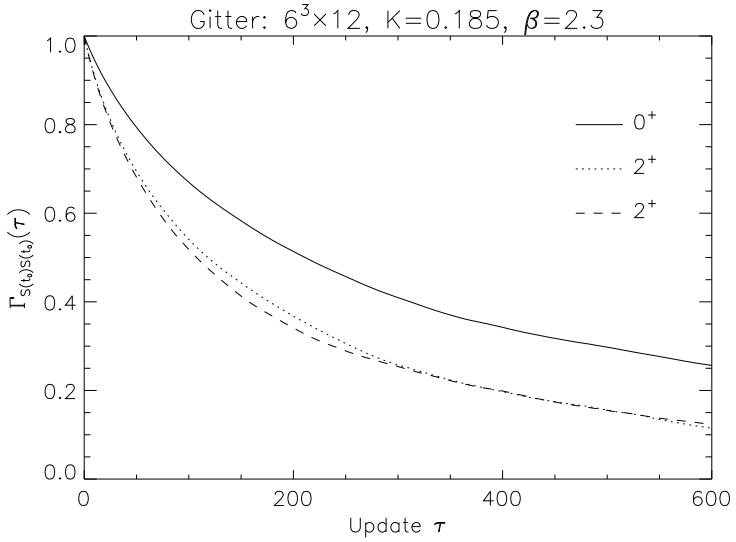


Abbildung 6.1: Autokorrelationsfunktion für den Zeitscheibenkorrelator an der Stelle $\triangle t = 0$ für den 0^+ Zustand und für die beiden 2^+ Observablen.

6.2.2 APE-Smearing

Beim APE-Smearing wird, wie in Abbildung 6.2 angedeutet, jeder räumliche Link durch die Summe aus sich selber und den vier „Klammern" in den beiden Raumrichtungen quer zur Link-Richtung ersetzt [A87]. Die Klammern gehen dabei mit

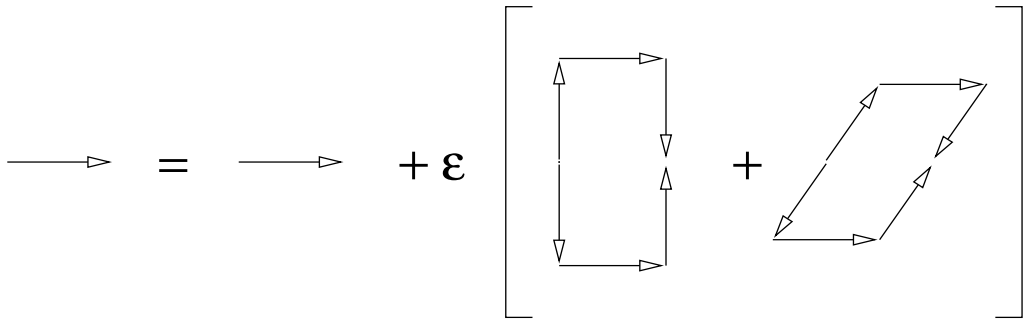


Abbildung 6.2: Graphische Darstellung des iterativen APE-Smearings.

einem Gewichtsfaktor ϵ in die Summe ein. Anschließend wird der Link wieder in die Gruppe der SU(N_c)-Matrizen zurückprojiziert

$$U_{x,\mu} \leftarrow U_{x,\mu} + \epsilon \sum_{\substack{\nu=1,2,3 \\ \nu \neq \mu}} \left(U^{\dagger}_{x+\hat{\mu},\nu} U_{x+\hat{\nu},\mu} U_{x,\nu} + U_{x+\hat{\mu}-\hat{\nu},\nu} U_{x-\hat{\nu},\mu} U^{\dagger}_{x-\hat{\nu},\nu} \right) . \tag{6.9}$$

Diese Prozedur wird N mal für das gesamte Gitter wiederholt. Für moderate Werte von ϵ entspricht das Verfahren dem numerischen Lösen einer Diffusionsgleichung, dementsprechend ist der Diffusions- oder besser Smearing-Radius durch

$$R_s = N\epsilon \tag{6.10}$$

gegeben. Es gibt zahlreiche Modifikationen dieses Verfahrens [GPB91], von denen aber keines für den Allgemeinfall deutliche Vorteile bietet. Das ursprüngliche Verfahren nutzt zudem nur Wechselwirkungen aus, die man auch schon beim Updater benötigt, was diesen Algorithmus attraktiv für einen Parallelrechner macht.

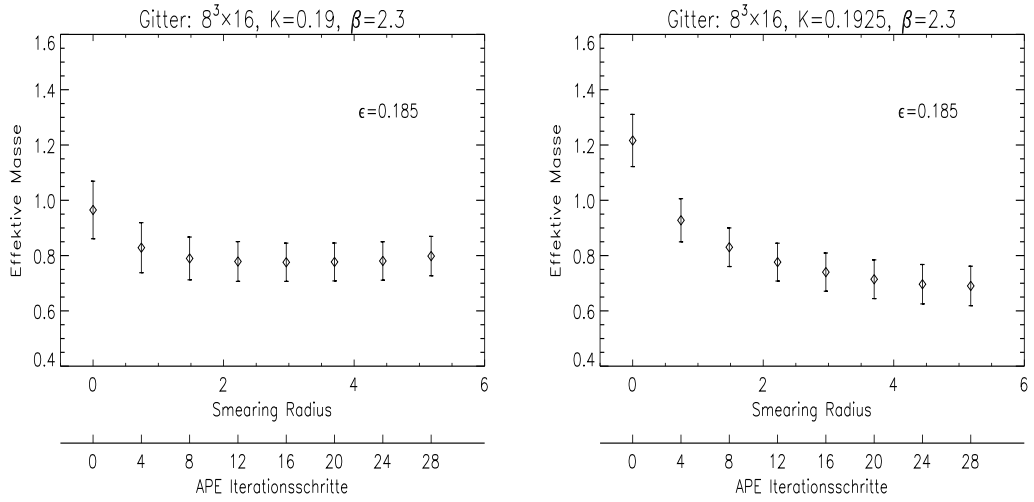


Abbildung 6.3: Effektive Massen $m^{0^+}_{gg}(L_t = 16, t_1 = 1, t_2 = 2)$ für verschiedene APE Smearing-Radien aus dem Zweipunkt-Fit der Zeitscheibenkorrelationsfunktion.

Der Smearing-Radius, bei dem das APE-Smearing die kleinsten Massen ergibt, ist ein grober Schätzer für die räumliche Ausdehnung des Grundzustandes. Erreicht man das Optimum, wie im rechten Diagramm aus 6.3, erst mit Smearing-Radien jenseits der halben räumlichen Gitterausdehnung, so ist dies ein Zeichen dafür, daß der Grundzustand durch das endliche Gitter bereits „eingequetscht“ wird.

6.2.3 Teper-Blocking

Das rekursive Teper-Blocking berechnet die Links $U^N_{x,\mu}$ für die N-te Rekursion aus den Links $U^{N-1}_{x,\mu}$ des $(N-1)$-ten Rekursionsschrittes gemäß der Abbildung 6.4. Nach der Berechnung müssen die neuen Links $U^N_{x,\mu}$ wieder in die Gruppe der SU(N_c)-Matrizen zurückprojiziert werden

$$U^N_{x,\mu} \leftarrow U^{N-1}_{x+2^{N-1}\hat{\mu},\mu} U^{N-1}_{x,\mu} + \sum_{\substack{\pm\nu,\ \nu\neq 4 \\ \nu\neq\mu}} U^{N-1\dagger}_{x+2^N\hat{\mu},\mu} U^{N-1}_{x+2^{N-1}\hat{\nu}+2^{N-1}\hat{\mu},\mu} U^{N-1}_{x+2^{N-1}\hat{\nu},\mu} U^{N-1}_{x,\nu}. \quad (6.11)$$

Die effektive Länge der geschmierten Links nimmt mit jedem Rekursionsschritt um den Faktor zwei zu. Die Länge $2^{N_{opt}}$, die sich aus der Anzahl N_{opt} der Blocking-Schritte ergibt, die man benötigt, um eine minimale Masse zu erhalten, gibt eine grobe Abschätzung für den Durchmesser des Grundzustandes an.

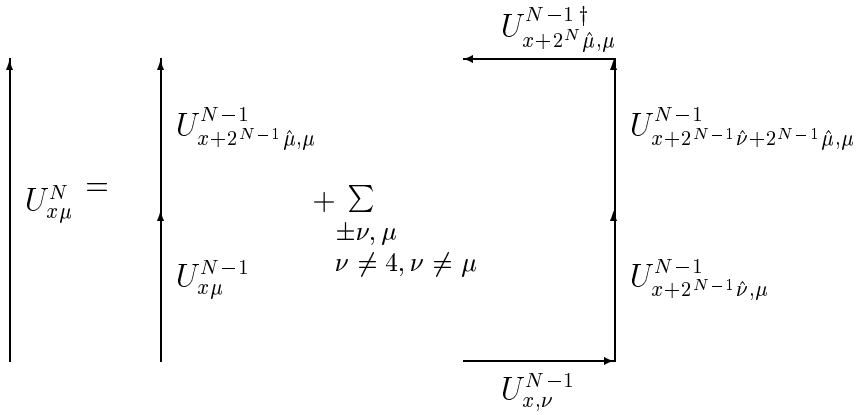


Abbildung 6.4: Graphische Darstellung des Teper-Blocking Verfahrens. Die geschmierten Links werden mit jedem Iterationsschritt um den Faktor zwei länger.

Im Rahmen dieser Arbeit kann kein eindeutiges Votum für das eine oder andere Verfahren abgegeben werden. Die Glueball-Massen aus beiden Verfahren stimmen im allgemeinen innerhalb der Fehler überein. Das APE-Smearing zeigt auf kleinen Gittern leichte Vorteile, da der Smearing-Radius differenzierter justiert werden kann. Hingegen unterschreiten auf größeren Gittern die statistischen Fehler der Massen aus dem Teper-Blocking die Fehler aus dem APE-Smearing (vgl. Abbildung 6.5 und 6.3). Ein Nachteil des Teper-Blockings für Parallelrechner mit verteiltem Speicher ist der globale Zugriffe auf Links vom Abstand 2^{N-1}. Diese Funktionalität zu implementieren ist deutlich anspruchsvoller als die Programmierung des APE-Smearings.

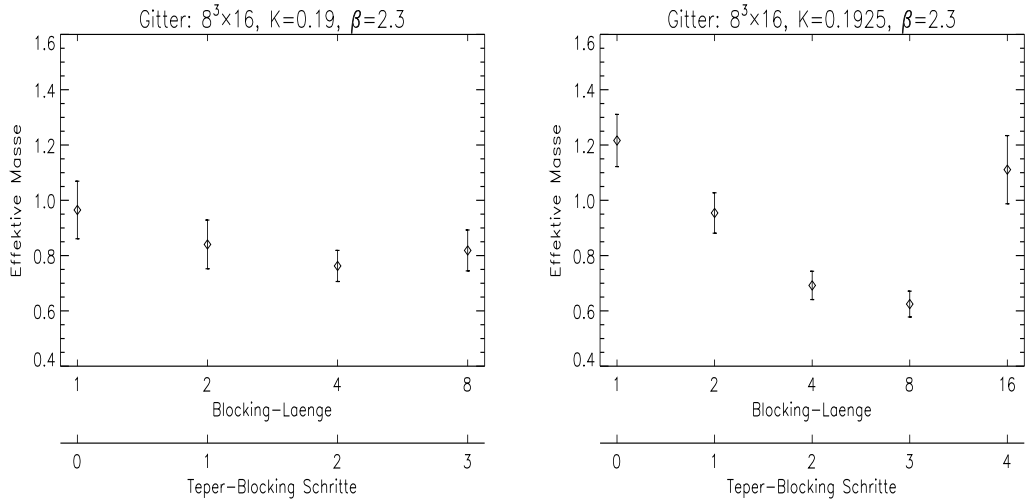


Abbildung 6.5: Effektive Massen $m_{gg}^{0^+}(L_t = 16, t_1 = 1, t_2 = 2)$ für verschiedene Blocking-Längen aus dem Zweipunkt-Fit der Zeitscheibenkorrelationsfunktion.

6.3 Gluinobälle

6.3.1 Korrelationsfunktionen

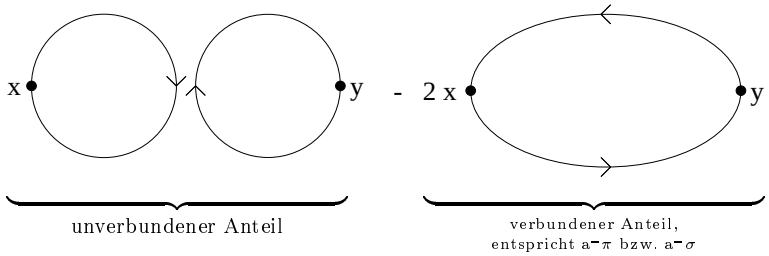
Das Veneziano-Yankielowicz-Multiplett enthält neben dem Gluino-Glueball zwei Gluinobälle mit Spin 0^- bzw. 0^+

$$a\text{-}\eta' \sim \bar{\Psi}\gamma_5\Psi \quad \wedge \quad a\text{-}f_0 \sim \bar{\Psi}\Psi. \tag{6.12}$$

Die zugehörigen rein fermionischen Korrelationsfunktionen können aus Gleichung (3.13) und $C\Gamma^T C^{-1} = -\Gamma$ mit $\Gamma \in \{\mathbb{1}, \gamma_5\}$ für punktförmige Quellen bzw. Senken berechnet werden

$$\Gamma_{\tilde{g}\tilde{g}}(x,y) = < \mathrm{Tr}_{sc}\{\Gamma Q_{xx}^{-1}\}\mathrm{Tr}_{sc}\{\Gamma Q_{yy}^{-1}\} - 2\,\mathrm{Tr}_{sc}\{\Gamma Q_{xy}^{-1}\Gamma Q_{yx}^{-1}\} > . \tag{6.13}$$

Mit Tr_{sc} ist dabei die Spur über Dirac- und Farbindex gemeint. Dies ist bis auf den Faktor zwei, der durch den Majorana-Charakter des Fermions bedingt ist, dieselbe Korrelationsfunktion wie in der QCD. Einem Flavour-Singulett Meson entsprechend, besteht sie aus einem verbundenen und einem unverbundenen Anteil.



Der verbundene Anteil entspricht in der QCD dem aus u- und $\bar{d}$-Quark aufgebauten Pion ($\Gamma = \gamma_5$) bzw. Sigma-Teilchen ($\Gamma = \mathbb{1}$). Da es in diesem Modell nur eine Sorte

von Gluinos gibt, handelt es sich hierbei nicht um physikalische Zustände. Allerdings erwartet man, daß genauso wie in der QCD die Masse dieses unphysikalischen a-π im chiralen Limes gegen Null geht [CV87]. Damit ist sie ein guter Indikator für den gesuchten Übergang $m_{\tilde{g}} \to 0$.

Die Berechnung eines Zeitscheibenerwartungswertes $S(t)$ des verbundenen Anteils ist für die Quelle zu aufwendig, da für jeden Gitterpunkt der Zeitscheibe eine Inversion von Q durchgeführt werden müßte; für die praktische Berechnung beschränkt man sich an dieser Stelle auf wenige, zufällig ausgewählte Punktquellen. Die Senke hingegen wird über die jeweilige Zeitscheibe gemittelt, um den $\mathbf{p} = 0$ Anteil heraus zu projizieren.

Die unverbundenen Anteile werden mit der „volume source technique“ berechnet [KFM94]. Dieses Verfahren nutzt an entscheidender Stelle das Elitzur-Theorem aus, wonach Erwartungswerte nichteichinvarianter Größen, also insbesondere Q^{-1}_{xy} mit $x \neq y$, unter allgemeinen Bedingungen verschwinden [Eli75]. Damit schreibt man den unverbundenen Erwartungswert als

$$< Q^{-1}_{xx} >=< Q^{-1}_{xx} > + \sum_{y \neq x} < Q^{-1}_{xy} >= \sum_{y} < Q^{-1}_{xy} > . \tag{6.14}$$

Den letzten Term kann man auf allen Gitterpositionen x für einen Satz von Dirac- und Farbindizes (α, c) durch eine einzige Fermion-Inversion gewinnen, indem man den Quellenvektor $b_{x\alpha c} = 1 \; \forall \; x \in V$ wählt und das Gleichungssystem $Qx = b$ löst.

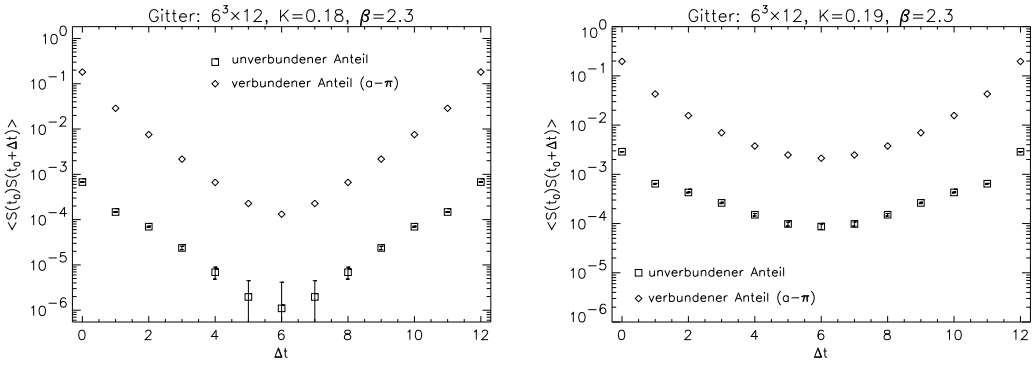


Abbildung 6.6: Verbundener und unverbundener Anteil zur Zeitscheibenkorrelationsfunktion des a-η' Teilchens (bei periodischen Randbedingungen).

Bei allen im Rahmen dieser Arbeit untersuchten Parametersätzen (K, β) wurde der Propagator des a-η' Zustands klar vom a-π Anteil dominiert, wie z. B. in Abbildung 6.6 zu sehen. Beim a-f_0 Zustand hingegen waren beide Anteile stets von derselben Größenordnung. Man wird erwarten, daß der a-η' Zustand auch im supersymmetrischen Limes eine endliche Masse behält, d. h. daß der Einfluß des unverbundenen

Anteils mit Annäherung an den kritischen Punkt zunimmt.

Auf Grund der verschiedenen Berechnungsmethoden unterscheidet sich das Signal/Rausch-Verhältnis beider Anteile der Korrelationsfunktion recht deutlich. Auch wenn man nur mit einer Punktquelle pro Konfiguration für den verbundenen Anteil arbeitet, ist das Rauschen auf diesem Part deutlich geringer als beim unverbundenen Anteil. Für das a-η' ist das nicht tragisch, allerdings wird der a-f_0 Zustand von beiden Anteilen gleichermaßen bestimmt. Durch das schlechte Signal der „volume source technique“ ist deshalb die Masse des a-f_0 Zustands mit größeren Fehlern behaftet als die des a-η'.

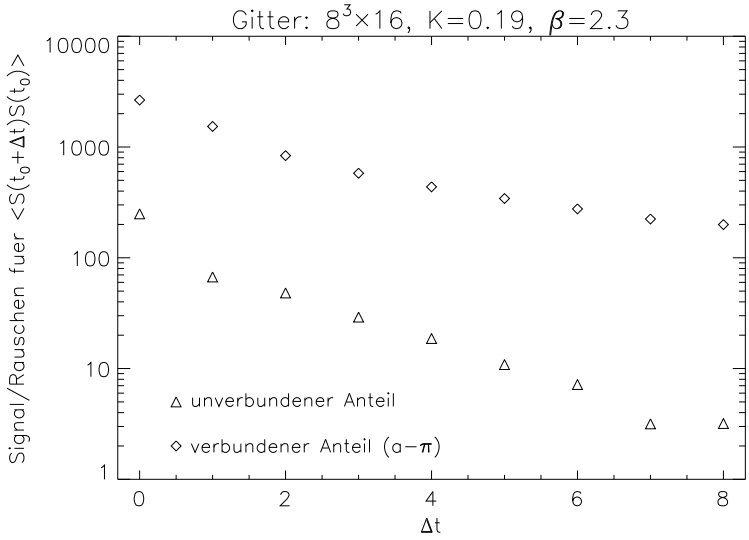


Abbildung 6.7: Das Verhältnis von Signal zu Rauschen der beiden Anteile zum a-η' Propagator.

6.3.2 Jacobi-Smearing

Natürlich ist die Wahl von punktförmigen Quellen bzw. Senken für den verbundenen Anteil des Gluinoballs nicht optimal. Es gibt für fermionische Bindungszustände zahlreiche Smearing-Algorithmen, wovon die populärsten, eichkovarianten Verfahren das „Wuppertal-“ bzw. das Jacobi-Smearing sind [GLM90][C93].

In diesem Fall wurde das Jacobi-Smearing eingesetzt, da es bei gleichen Smearing-Radien die zugehörigen Quellen bzw. Senken mit weniger CPU-Zeit berechnen kann. Beide Verfahren nutzen “shell model“ Wellenfunktionen für die Approximation des Grundzustandes, bei denen die punktförmigen Quellen für die Gluinos (Quarks) im Ursprung $\mathbf{y}$ durch einen skalaren Propagator verschmiert werden. Die verschmierten Quellen $J_{\mathbf{x}}^{(\mathbf{y})}$ werden dazu aus einer Lösung der dreidimensionalen Klein-Gordon Gleichung

$$K_{\mathbf{x}',\mathbf{x}} J_{\mathbf{x}}^{(\mathbf{y})} = \delta_{\mathbf{x}',\mathbf{y}}, \tag{6.15}$$

mit dem eichkovarianten Klein-Gordon-Operator

$$K_{\mathbf{x}',\mathbf{x}} = \delta_{\mathbf{x}',\mathbf{x}} - \kappa_S \sum_{\mu=1}^{3} \left\{ \delta_{\mathbf{x}',\mathbf{x}+\hat{\mu}} V_{x\mu} + \delta_{\mathbf{x}'+\hat{\mu},\mathbf{x}} V^T_{x'\mu} \right\} \tag{6.16}$$

gewonnen. Durch die Form von $K_{\mathbf{x}',\mathbf{x}}$ ist ein skalares Smearing gewährleistet, d. h., es werden keine Dirac- oder Farbindizes miteinander gemischt. Diese Gleichung kann durch das iterative Jacobi-Verfahren gelöst werden [GL91], das in diesem Fall die Form

$$J^{(\mathbf{y})}_{\mathbf{x}} = \sum_{n=0}^{N} (\delta_{\mathbf{x},\mathbf{x}'} - K_{\mathbf{x},\mathbf{x}'})^n \delta_{\mathbf{x}',\mathbf{y}} \tag{6.17}$$

annimmt, wenn man mit dem Ausgangsvektor $J = 0$ die Jacobi-Iteration startet. Unterhalb eines kritischen Wertes für κ_S wird diese Reihe konvergieren, und man erhält die skalare Wellenfunktion. Oberhalb dieser Grenze divergiert die Reihe zwar, aber für geeignetes, endliches N ergibt sie aber immer noch eine akzeptable Wellenfunktion. Das Verschmieren der Quelle erfordert damit nicht viel Rechenzeit. Sollen allerdings auch die Senken verschmiert werden, so muß man den Smearing-Algorithmus auf jede Zeitscheibe anwenden, was vom Aufwand her ungefähr der Invertierung der Fermion-Matrix entspricht.

Im freien Fall $U = 1$ entspricht $J^{(\mathbf{y})}_{\mathbf{x}}$ einer Gauß-Funktion in $|\mathbf{x} - \mathbf{y}|$. Der Smearing-Radius R_S kann allgemein durch die Varianz

$$R_S^2 = \frac{\sum_{\mathbf{x}} |\mathbf{x}|^2 |J^{(\mathbf{0})}_{\mathbf{x}}|^2}{\sum_{\mathbf{x}} |J^{(\mathbf{0})}_{\mathbf{x}}|^2} \tag{6.18}$$

abgeschätzt werden. Die Parameter (N, κ_S) des Jacobi-Smearings müssen für unterschiedliche (K, β) jeweils neu optimiert werden, um die beste Überlappung mit dem Grundzustand zu erreichen. Insbesondere ist der optimale Smearing-Parameter κ_S im allgemeinen nicht gleich dem Hopping-Parameter K. Man beobachtet, daß für ein nicht zu kleines N der Smearing-Radius R_S über die Qualität des Smearings entscheidet und weniger die jeweilige Kombination (N, κ_S). Wie in Abbildung 6.8 zu sehen ist, steigt $R_S(\kappa_S)$ in einem sehr kleinen Bereich rapide an. Es bedurfte immer etwas „Fingerspitzengefühls“ (und viel Rechenzeit), um κ_S so einzustellen, daß man den optimalen Smearing-Radius traf.

Abbildung 6.9 zeigt, wie das Jacobi-Smearing die Projektion auf den Grundzustand beim a-π verbessern kann. Hierbei wurden sowohl die Quellen als auch die Senken gleichermaßen verschmiert. Ohne Smearing läßt selbst die effektive Masse $m_{a\text{-}\pi}(12, 4, 5)$, also diejenige Masse, welche auf dem Fit-Intervall $\Delta t \in [4, 5]$ bestimmt wird, keine gute Extrapolation auf die wirkliche Masse zu. Hingegen ist beim optimalen Smearing-Radius $R_S^2 = 1.02(2)$ schon die effektive Masse $m_{a\text{-}\pi}(12, 2, 3)$ für die Extrapolation auf die physikalische Masse geeignet. Für Smearing-Radien über $R_S^2 = 1.02(2)$ nahm die Qualität der Projektion auf den Grundzustand wieder ab.

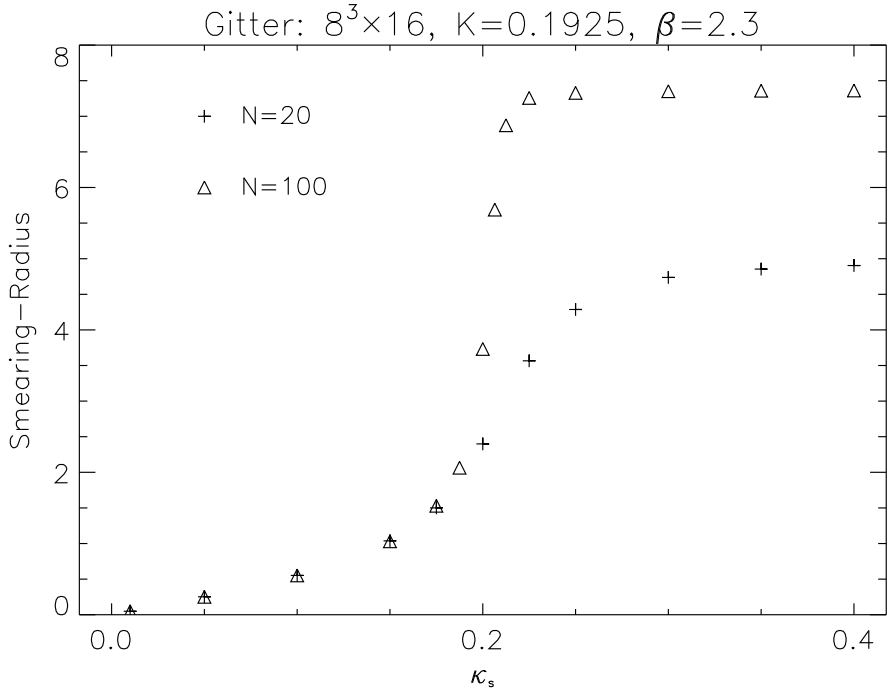


Abbildung 6.8: Radien des Jacobi-Smearings gemittelt über 32 Eichkonfigurationen. Die Fehler der Erwartungswerte sind kleiner als die Symbole.

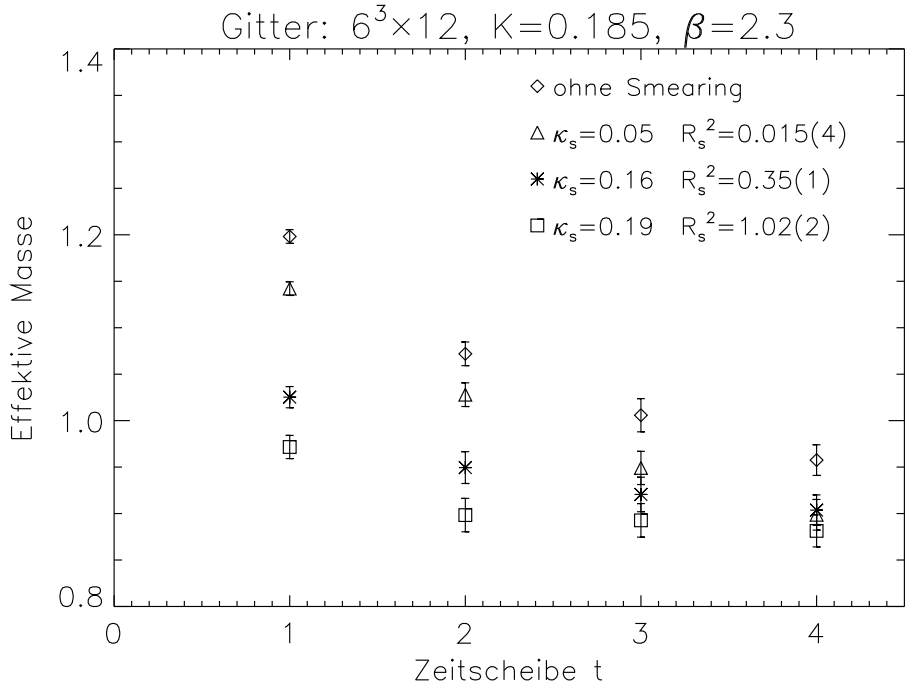


Abbildung 6.9: Effektive Massen $m_{a\text{-}\pi}(12, t, t+1)$ des a-π Zustands aus den Zweipunkt Fits für verschiedene Smearing-Radien.

In der QCD wird das Wuppertal-Smearing auch zur Bestimmung der η' Masse eingesetzt [VKG97]. Prinzipiell könnte hier eine modifizierte Version der a-π Meßroutine für das Jacobi-Smearing auf die Zustände a-η' und a-f_0 angewandt werden. A priori ist aber nicht klar, ob dieses Verfahren für die unverbundenen Anteile ein besseres Signal als die „volume source technique“ liefert.

6.4 Gluino-Glueball

6.4.1 Korrelationsfunktion

Die adjungierte Darstellung erlaubt es, farblose Bindungszustände aus den Gluinos und dem Feldstärketensor aufzubauen. Das Resultat ist ein Spin $\frac{1}{2}$ Majorana-Fermion, das schon von Veneziano und Yankielowicz zum Aufbau ihrer effektiven Wirkung benutzt wurde.

Um die niedrigste Masse in diesem Kanal zu bestimmen, benutzt man eine Korrelationsfunktion, die aus zwei Plaquetten besteht, welche durch eine Valenz-Gluino Linie verbunden sind

$$\Gamma_{g\tilde{g}}(x,y) = \mathrm{Tr}\left(\chi^a_x Q^{-1}_{x\alpha a, y\beta b}\chi^b_y\right), \tag{6.19}$$

wobei man aus den Plaquette-Erwartungswerten

$$\bar{U}_x = U_{x,12} + U_{x,13} + U_{x,23} \tag{6.20}$$

durch die Pauli-Matrizen σ^a ein Fermion-Feld in der adjungierten Darstellung

$$\chi^a_x = \frac{1}{2i}\mathrm{Tr}\left(\sigma^a \bar{U}_x\right) \tag{6.21}$$

konstruiert. Da die Korrelationsfunktion nur eine Gluino-Linie besitzt, hängt sie von den gewählten Randbedingungen für die Fermionen in zeitlicher Richtung ab. Man kann durch das Einschieben einer γ_4-Matrix in die Korrelationsfunktion (6.19) auch bei antiperiodischen Randbedingungen in Zeitrichtung für die Fermionen zu einer periodischen Korrelationsfunktion gelangen [MM94] (s. Abbildung 6.10 und 6.11).

Es zeigt sich, daß die resultierende Projektion auf den Grundzustand für beide Korrelationsfunktionen gleich gut ist, d. h., die effektiven Massen enthalten innerhalb der statistischen Fehler jeweils gleichstarke Beimischungen von den nächst größeren Massen. Noch nicht klar ist zur Zeit, ob beide Korrelationsfunktionen ein unabhängiges Signal liefern oder ob die Ergebnisse aus den beiden Funktionen stark korreliert sind. Alle Massen für den Gluino-Glueball in dieser Arbeit stammen von der ursprünglichen Version der Korrelationsfunktion ohne γ_4-Matrix.

6.4.2 Smearing-Algorithmen

Um die Projektion auf den Grundzustand und das Signal/Rausch-Verhältnis zu verbessern, muß auch für diesen Zustand Smearing benutzt werden. Bedingt durch

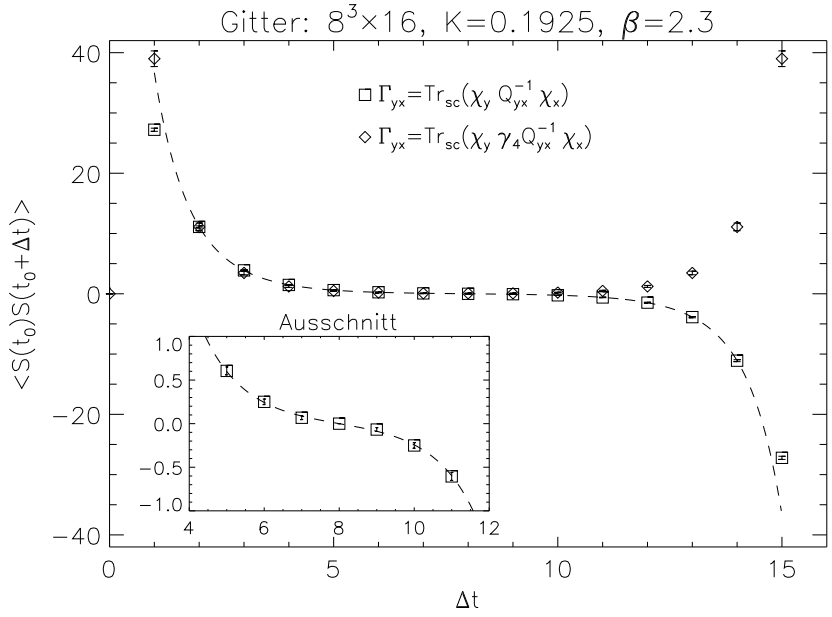


Abbildung 6.10: Die Zeitscheibenkorrelationsfunktion des Gluino-Glueballs bei antiperiodischen Randbedingungen in Zeitrichtung und optimiertem APE- und Jacobi-Smearing. Die Linie entspricht einem Fit für $\Delta t \in [2, 14]$ für die beiden kleinsten Massen.

den hybriden Charakter des Teilchens stehen hier viele Varianten zur Verfügung. Die Links, auf denen die Plaquetten berechnet werden, können entweder mit APE-Smearing oder mit Teper-Blocking verschmiert werden. Es hat sich herausgestellt, daß Teper-Blocking für diesen Zweck deutlich schlechter geeignet ist als APE-Smearing. Das trifft für das Signal/Rausch-Verhältnis und für die Projektion auf den Grundzustand zu. APE-Smearing besitzt den Vorteil, daß man den Smearing-Radius sehr viel differenzierter einstellen kann, als es das diskrete Blocking-Schema des Teper-Algorithmus zuläßt.

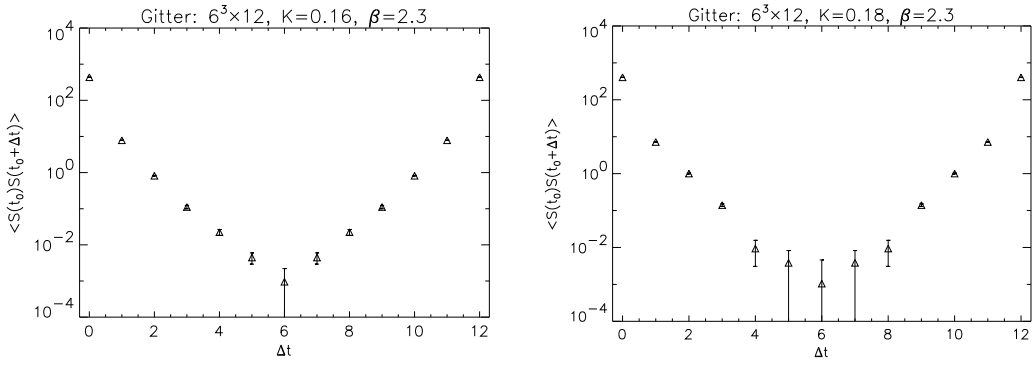


Abbildung 6.11: Die Zeitscheibenkorrelationsfunktion des Gluino-Glueballs bei periodischen Randbedingungen und Teper-Blocking der Länge 2.

Es hat sich als vorteilhaft erwiesen, sowohl an den Quellen als auch an den Senken des Gluino-Propagators Jacobi-Smearing zu verwenden. Die Qualität des Signals und der Projektion sind sehr empfindlich von der optimalen Wahl der beiden APE-Parameter (N_{ape}, ϵ) und der beiden Jacobi-Parameter (N_{jacobi}, κ_S) abhängig. Dieses vierdimensionale Optimierungsproblem macht das Messen der Gluino-Glueball Masse sehr umständlich. In der Tat wurde oftmals mehr CPU-Zeit zur Optimierung der Parameter verbraucht als dann zur eigentlichen Massenmessung auf allen zur Verfügung stehenden Konfigurationen. Auf der anderen Seite hat erst die Kombination dieser beiden Smearing-Verfahren eine aussagekräftige Bestimmung der Massen möglich gemacht.

Kapitel 7

Das chirale Supermultiplett

7.1 Kritischer Hopping-Parameter

7.1.1 Phasendiagramm

Eine Gluino-Masse $m_{\tilde{g}} \neq 0$, wie sie durch die Wilson-Wirkung eingeführt wird, bricht sowohl die chirale Symmetrie als auch die Supersymmetrie. Stellt man allerdings den Hopping-Parameter K auf den kritischen Wert K^{cr}_{chiral} ein, an dem die Gluino-Masse verschwindet, so kann man die chirale Symmetrie wieder herstellen. Im Kontinuumslimes muß damit auch die Restauration der Supersymmetrie einhergehen, d. h. in diesem Fall gilt $K^{cr}_{chiral} = K^{cr}_{susy}$. Zum Kontinuumslimes gelangt man in nichtabelschen Eichtheorien durch $\beta \to \infty$. Der Wert von K^{cr} ist eine Funktion von β mit $lim_{\beta\to\infty} = 1/8$.

Für die reine SU(2) Eichtheorie hat sich aber gezeigt, daß man schon bei moderaten Werten von $\beta = 2.0 \sim 3.0$ die Kontinuumsphysik extrahieren kann. Andererseits wird die Monte-Carlo-Simulation mit steigendem β immer aufwendiger, so daß alle bisherigen Simulationen mit dynamischen Gluinos beim Kompromißwert $\beta = 2.3$ durchgeführt wurden. Man wird auch für diesen Wert von β hoffen können, daß die beiden kritischen Hopping-Parameter, wenn sie schon nicht aufeinanderfallen sollten, sehr nahe beieinander liegen.

Wie im Abschnitt 1.3.2 beschrieben, besitzt die SU(2) Super-Yang-Mills-Theorie für eine verschwindende Gluino-Masse zwei entartete Vakuumzustände mit unterschiedlichen Erwartungswerten für das Gluino-Kondensat $<\lambda\lambda>$. Die Koexistenz von verschiedenen Vakuumzuständen erfordert im Kontinuum einen Phasenübergang erster Ordnung an der Stelle $K = K^{cr}_{susy}$. In der Parameterebene (K, β) sollte davon bei endlichem Gitterabstand mindestens ein „cross-over“ Gebiet übrig bleiben. Trotzdem ist natürlich auch ein „richtiger“ Phasenübergang erster Ordnung auf unendlich großen Gittern für ein endliches β denkbar. Daraus resultiert für den einfachsten Fall ein Phasendiagramm, wie es in Abbildung 7.1 angedeutet ist.

Darüber hinaus wird die Existenz einer möglichen weiteren Phase mit verschwinden-

dem Fermion-Kondensat $<\lambda\lambda>$ diskutiert. [KS97], die zwischen den beiden Phasen I und II liegen soll. Auf dem Gitter würde das natürlich zu weiteren Komplikationen bei der Simulation führen, da die verschiedenen Phasen metastabile Bereiche in der (K, β) Parameterebene haben könnten.

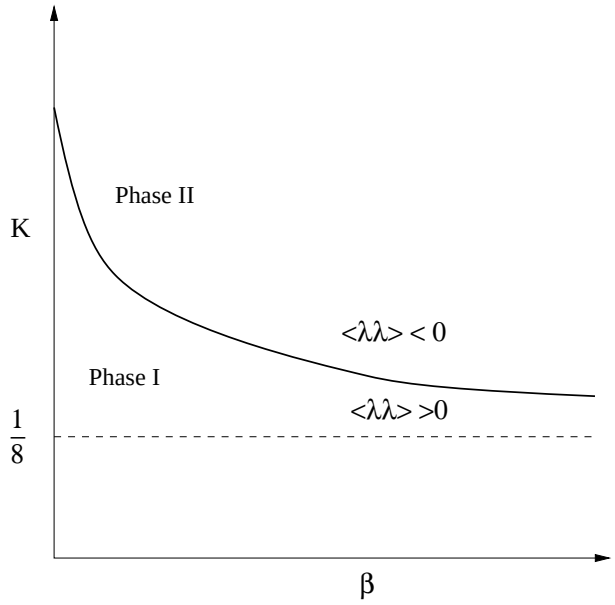


Abbildung 7.1: Einfachstes Phasendiagramm für eine SU(2) Eichtheorie mit Gluinos in der adjungierten Darstellung. Auf der Linie zwischen den beiden Phasen kann ein Phasenübergang erster Ordnung oder aber ein cross over Bereich lokalisiert sein.

Im Kontinuumslimes sollte die Masse des a-π, das als Goldstone-Boson der chiralen Symmetrie aufgefaßt werden kann, gegen Null gehen [VY82]. Gleichzeitig muß die Masse des a-η' Zustands bei der Restaurierung der Supersymmetrie ungleich Null bleiben, da es zu einem massiven Supersymmetrie-Multiplett gehört. Mit zunehmender Annäherung an den kritischen Hopping-Parameter muß deshalb die unverbundene Korrelationsfunktion des a-η' einen größer werdenden Anteil an der Masse gewinnen.

Damit ergeben sich neben der direkten Massenmessung des chiralen Multipletts drei Verfahren, um für gegebene Eichkopplung β zum kritischen Hopping-Parameter zu extrapolieren:

- In Analogie zur quenched-QCD könnte die Masse des a-π in der Umgebung von $K^{cr}_{chiral} \approx K^{cr}_{susy}$ wie $m^2_{a\text{-}\pi} \propto K^{-1} - K^{cr\,-1}$ abfallen.
- Die Masse und damit die Korrelationsfunktion des a-η' muß in der Nähe von K^{cr}_{susy} zunehmend vom unverbundenen Anteil dominiert werden.
- Die Verteilung des Gluino-Kondensats könnte bei K^{cr}_{susy} aufspalten und zwar in die beiden Erwartungswerte der Phasen I und II. Zumindest sollte ein cross over in K mit steiler Änderung des Kondensats beobachtbar sein.

Die letzte Möglichkeit wird die genaueste sein, da man erfahrungsgemäß den zu den zwei Phasen gehörenden „double peak“ im Ordnungsparameter bei einem Phasenübergang erster Ordnung nur in unmittelbarer Umgebung vom kritischen Punkt beobachten kann.

7.1.2 Das Verschwinden der a-π Masse

Das Verhalten des a-π für verschiedene Hopping-Parameter ist für die quenched Approximation schon untersucht worden. Für $\beta = 2.3$ auf einem $8^3 \times 16$ Gitter ist als kritischer Hopping-Parameter $K^{cr}_{chiral} = 0.2151(3)$ gemessen worden[KM97], während auf einem $16^3 \times 32$ Gitter bei $\beta = 2.6$ ein Wert von $K^{cr}_{chiral} = 0.18752(9)$ angegeben wird [DGHV98].

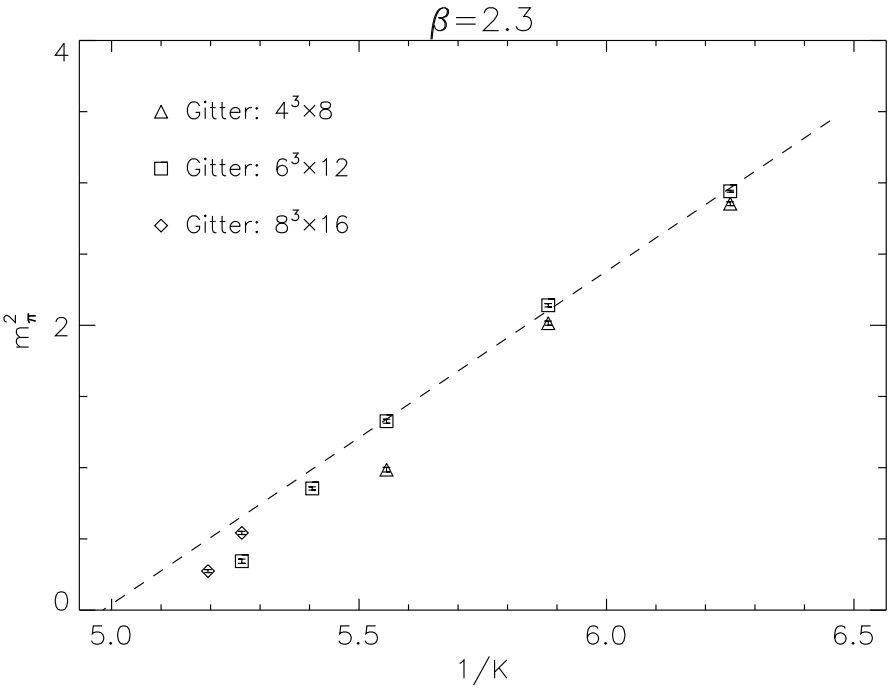


Abbildung 7.2: Das Quadrat der effektiven a-π Masse $m_{a\text{-}\pi}(L_t, \frac{L_t}{2} - 2, \frac{L_t}{2})$ als Funktion des inversen Hopping-Parameters. Die Linie ist ein Fit zur Bestimmung von K^{cr}_{chiral}.

Im direkten Vergleich zu den quenched Rechnungen funktioniert der Fit-Ansatz $m^2_{a\text{-}\pi} \propto K^{-1} - K^{cr\,-1}$ für die Resultate aus den Simulationen mit dynamischen Gluinos, wie in Abbildung 7.2 zu sehen, insbesondere für $K \geq 0.185$, nicht mehr so gut. Zum einen kann der Grund hierfür in finite-size-Effekten liegen, wie die Massen von unterschiedlichen Gittergrößen bei gleichem Hopping-Parameter suggerieren (Dabei beruhen diese Abweichungen zum Teil auf der schlechteren Projektion auf den Grundzustand für Gitter mit kleineren L_t. Dieser „unechte“ finite-size-Effekt könnte durch Jacobi-Smearing, wie in Abschnitt 6.3.2 vorgestellt, minimiert werden.). Zum

anderen können es aber auch Effekte der dynamischen Gluinos sein, die den Ansatz $m^2_{a\text{-}\pi} \propto K^{-1} - K^{cr\,-1}$ für kleine Gluino-Massen unbrauchbar machen.

In jedem Fall sorgen die dynamischen Gluinos bei gleichem Hopping-Parameter für kleinere a-π Massen und senken damit den Wert für K^{cr}_{chiral} deutlich unter den Wert aus der quenched Approximation. Da aber letztlich nicht klar ist, welche Funktion man in der vollen Theorie für den Fit an $m^2_{a\text{-}\pi}(K)$ bei kleinen a-π Massen benutzen sollte, kann man lediglich ein relativ großes Intervall für den kritischen Hopping-Parameter angeben $K^{cr}_{chiral} \approx 0.195 \sim 0.20$.

7.1.3 Verbundener und unverbundener Anteil von a-η'

Der unverbundene Anteil des a-η' Propagators war bei allen untersuchten Hopping-Parametern $K \in [0.16, 0.1925]$ immer mindestens eine Größenordnung kleiner als der verbundene Anteil. Es ist allerdings ein deutlicher Trend zu beobachten, wonach der unverbundene Anteil mit steigendem K größer werdende Anteile am Gesamtpropagator bekommt. Um eine Extrapolation in Bereichen zu erlauben, in denen der unverbundene Anteil dominiert, werden verschiedene Fit-Versuche für das Verhältnis der beiden Korrelationsfunktionen bei $\Delta t = 0$ in K^{-1} unternommen.

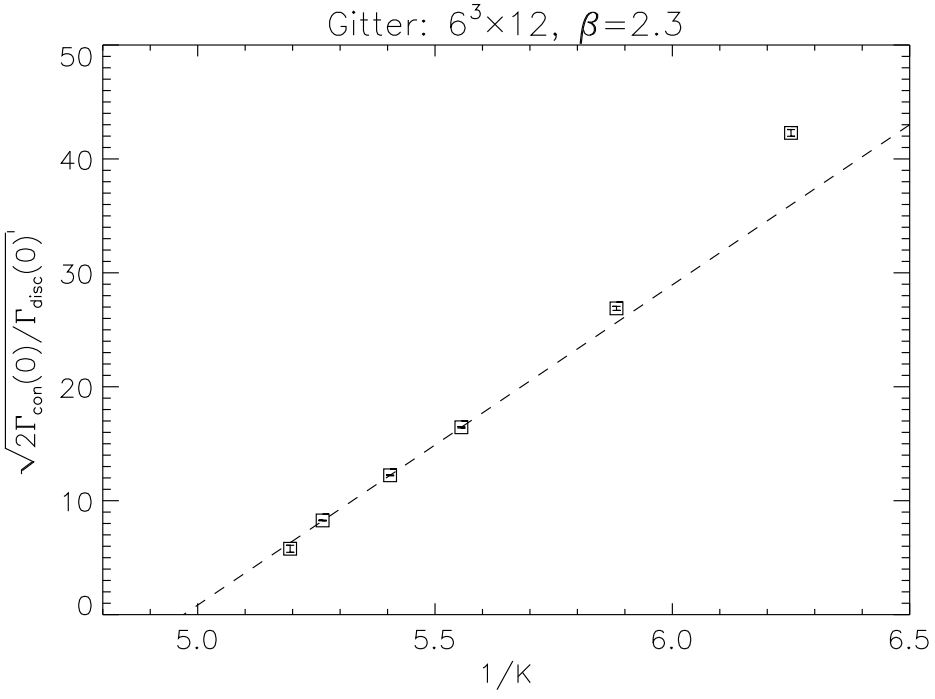


Abbildung 7.3: Vergleich des verbundenen und des unverbundenen Anteils der Zeitscheibenkorrelationsfunktion von a-η' an der Stelle $\Delta t = 0$. Die Linie entspricht dem Fit zur Bestimmung von K^{cr}_{susy} nach Gleichung 7.1.

Ein Fit, wie er zur Bestimmung kritischer Exponenten üblich ist

$$\frac{2\,\Gamma_{con}(\Delta t=0)}{\Gamma_{dis.}(\Delta t=0)} = c_0\left(\frac{1}{K}-\frac{1}{K^{cr}}\right)^{\nu}, \tag{7.1}$$

eignet sich dazu, die Meßwerte zu extrapolieren; für das Intervall $K \in [0.18, 0.19]$ ergibt er $\nu = 2.04(5)$. Wie in Abbildung 7.3 zu sehen, resultiert aus der damit getroffenen Annahme $\nu = 2$ eine plausible Extrapolation für dieses Verhältnis hin zu größeren K-Werten, wohlwissend, daß es keine physikalische Begründung für die Wahl der Fit-Funktion gibt. In diesem Sinne sollte die eingetragene Linie in der Abbildung auch als „Hilfslinie für das Auge“ verstanden werden.

In der Nähe der supersymmetrischen Theorie sollte der unverbundene Anteil zunehmend die a-η' Masse erzeugen, d. h., man wird erwarten, daß dieses Verhältnis mindestens Ordnung eins wird. Dem Fit zur Folge würde man demnach $K^{cr}_{susy} \approx 0.20$ erwarten. Allerdings legt der Wert bei $K = 0.1925$ einen kleineren Wert für K^{cr}_{susy} nahe, weil er unterhalb der Fit-Linie liegt.

7.1.4 Fermion-Kondensat

Das renormierte Fermion-Kondensat $<\lambda\lambda>(\mu)$ ist durch das nackte Fermion-Kondensat $<\lambda\lambda>(a)$ beim Gitterabstand a über die Relation

$$<\lambda\lambda>(\mu) = Z(\mu a)\Big[<\lambda\lambda>(a) - b_0\Big] \tag{7.2}$$

verbunden, wobei $Z(\mu a)$ eine multiplikative Renormierungskonstante ist und b_0 eine additive Konstante (Eine aktuelle Studie dieser Größe für die QCD mit Herleitung der Relation (7.2) findet man in [GRTV98].). Es existiert eine erste grobe Abschätzung über die Stärke des Sprungs im nackten Kondensat für die SU(2) Eichtheorie mit Gluinos, wonach man mit einer Aufspaltung der Größenordnung eins zu rechnen hat [Mon98c].

Bestimmen kann man das Fermion-Kondensat auf dem Gitter nach Gleichung (3.14) mit $C^{-1} = C^{\dagger}$ durch $<\Psi_x\bar{\Psi}_x> = Q^{-1}_{xx}$. Allerdings würde die Bestimmung von Q^{-1}_{xx} durch das konjugierte Gradientenverfahren an allen Gitterpunkten x selbst auf kleinen Gittern zuviel CPU-Zeit in Anspruch nehmen. Man wird sich hier mit einer kleinen Anzahl N von zufällig ausgewählten Punkten begnügen müssen. Damit ist die Streuung des Schätzers für das Fermion-Kondensat mit einer „natürlichen“ Linienbreite σ_{nat}, und einer Meßstreuung

$$\sigma_{mess}(N) = \frac{\sigma_{mess}(1)}{\sqrt{N}}, \tag{7.3}$$

die aus der unvollständigen Messung von $\frac{1}{V}\sum_x Q^{-1}_{xx}[U]$ resultiert, bei kleinen N modellierbar. Die Gesamtstreuung ergibt sich im Fall, daß noch keine Aufspaltung vorliegt, durch Addition der beiden Gauß-Verteilungen

$$\sigma = \sqrt{\sigma^2_{nat} + \frac{\sigma_{mess}(1)^2}{N}}. \tag{7.4}$$

Durch Messung bei verschiedenen N-Werten kann man die beiden Streuungen bestimmen. Für das $8^3 \times 16$ Gitter bei $(K = 0.1925, \beta = 2.3)$ ergeben sich die typischen Werte

$$\sigma_{nat} \approx 0.05 \quad \wedge \quad \sigma_{mess}(N = 1) \approx 0.35. \tag{7.5}$$

Je nach gewünschter Ortsauflösung muß man demnach genügend Inversionen von $Q^{-1}_{xx}[U]$ durchführen.

Das Fermion-Kondensat ist natürlich auch über die Ergebnisse der „volume source technique“ zugänglich. Allerdings ist das gesuchte Signal $\frac{1}{V}\sum_x Q^{-1}_{xx}[U]$ hierbei vom Rauschen der nichteichinvarianten Terme $Q^{-1}_{xy}[U]$ mit $x \neq y$ überlagert, so daß man eine Vergrößerung der „natürlichen“ Linienbreite erwarten wird. Typische Werte für das $8^3 \times 16$ Gitter bei $(K = 0.1925, \beta = 2.3)$ lagen im Bereich $\sigma \approx 0.15$.

Eine dritte Möglichkeit eröffnen Noisy Estimators mit denen man direkt aus den pseudofermionischen Feldern $\phi^{(n)}$ Erwartungswerte rein fermionischer Größen gewinnen kann. In diesem Fall kann das Fermion-Kondensat direkt durch

$$\begin{aligned} <\Psi_x \bar{\Psi}_x> \quad &\stackrel{n\to\infty}{=} \quad 2\sum_{i=1}^{n} <\phi_x^{(i)\dagger}\gamma_5\hat{\phi}_x^{(i)} + \hat{\phi}_x^{(i)\dagger}\gamma_5\phi_x^{(i)}> \\ \text{mit} \quad \hat{\phi}_x^{(i)} \quad &= \quad \sum_y \left(\tilde{Q} + \mu_i\right)_{xy} \phi_y^{(i)} \end{aligned} \tag{7.6}$$

ohne Matrixinversion gemessen werden [Mon96]. Allerdings hängt die Qualität dieses Schätzers von der Approximationsgüte des ersten Polynoms P_n ab. Durch den Zwei-Schritt-Algorithmus mit nachgeschaltetem hybriden Korrekturschritt wird aber gerade diese erste Approximation oftmals noch recht ungenau sein. Deshalb ist es nicht verwunderlich, daß dieser Erwartungswert oftmals mehrere Standardabweichungen von den Erwartungswerten der beiden anderen Methoden abweicht. Zudem macht der Noisy Estimator in diesem Fall seinem Namen alle Ehre, denn die Streuung war deutlich größer als die der beiden anderen Observablen. Damit ist dieser Noisy Estimator nicht geeignet, den Sprung im Fermion-Kondensat zu detektieren.

Da in allen Produktionsläufen jeweils die a-η' und a-f_0 Massen bestimmt werden, stehen automatisch die Messungen zum Fermion-Kondensat durch „volume source technique“ bzw. für mindestens einen Gitterpunkt auch das Ergebnis durch Inversion $Q^{-1}_{xx}[U]$ zur Verfügung. Die Verteilungen der Fermion-Kondensate aus diesen Messungen werden laufend mit Hilfe des χ^2-Anpassungstests auf Abweichungen von der Normalverteilung hin untersucht [PTVF94].

In Abbildung 7.4 ist exemplarisch das Ergebnis eines χ^2-Anpassungstests zu sehen. Ein Signifikanzniveau von 16% für dieses Beispiel gibt keinen Anlaß an der Nullhypothese zu zweifeln, daß die Verteilung einer Gauß-Funktion entspricht. Dasselbe gilt auch für alle untersuchten Hopping-Parameter kleiner als $K = 0.1925$. Allerdings ist ein großes Gitter für die Untersuchung des Phasenübergangs nicht so gut geeignet, da die Tunnelwahrscheinlichkeit zwischen den Phasen mit dem Gittervolumen kleiner wird. Zudem zeigt das Fermion-Kondensat nur sehr geringe finite-size-Effekte,

so daß kleinere Gitter für die Suche nach dem Phasenübergang besser geeignet sind.

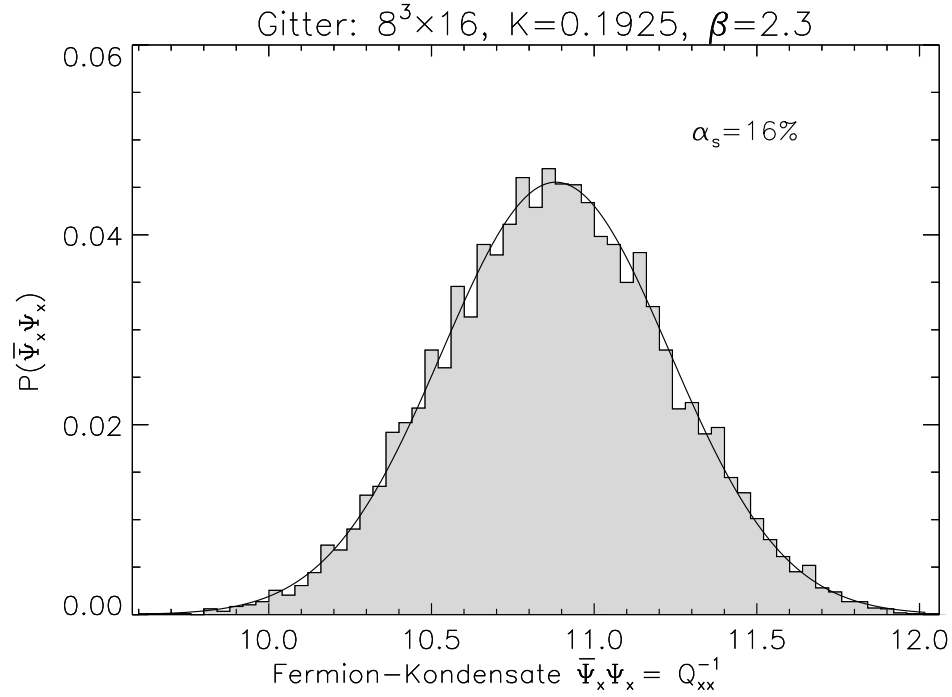


Abbildung 7.4: Verteilung des Fermion-Kondensats $\bar{\Psi}_x\Psi_x = Q^{-1}_{xx}[U]$. Pro Eichkonfiguration wurden der Wert für einen zufällig ausgewählten Gitterplatz x berechnet. Die durchgezogene Linie ist ein Fit mit einer Gauß-Kurve. α_s bezeichnet das Signifikanzniveau des χ^2-Anpassungstests.

Deshalb wurde die Suche nach dem Phasenübergang auf einem $6^3 \times 12$ Gitter bei $(K = 0.195, \beta = 2.3)$ fortgesetzt. Dieser Produktionslauf war von vornherein nicht mehr darauf ausgelegt, noch irgendwelche Massen zu bestimmen, was bei dieser Gittergröße nach Abbildung 7.2 sowieso nicht mehr sinnvoll wäre. Es wurden bei periodischen Randbedingungen für das Gluino in Zeitrichtung 832 unabhängige Konfigurationen erzeugt und deren Korrekturfaktoren mit dem hybriden Algorithmus bestimmt. In einem ersten Schritt wurde für einen zufällig ausgewählten Gitterpunkt je Konfiguration das Fermion-Kondensat berechnet und die Resultate unter Berücksichtigung des jeweiligen Korrekturfaktors in einem Histogramm zusammengefaßt. Zum ersten Mal gab der χ^2-Anpassungstest Anlaß an einer Normalverteilung zu zweifeln.

Um die Ortsauflösung des Histogramms zu verbessern, wurde im Anschluß daran auf jeder Konfiguration an 320 verschiedenen Gitterpositionen das Fermion-Kondensat gemessen. Wie Abbildung 7.5 zeigt, hat der erste Eindruck nicht getrogen. Die Verteilung der Fermion-Kondensate ist nicht mehr durch eine Gauß-Kurve erklärbar, allerdings bietet ein Fit mit zwei Gauß-Kurven bei einem Signifikanzniveau von 46%

eine plausible Erklärung für die gemessene Verteilung. Die unterschiedliche Höhe der beiden Gauß-Kurven deutet an, daß man das Gebiet der Koexistenz zweier Phasen noch nicht genau getroffen hat.

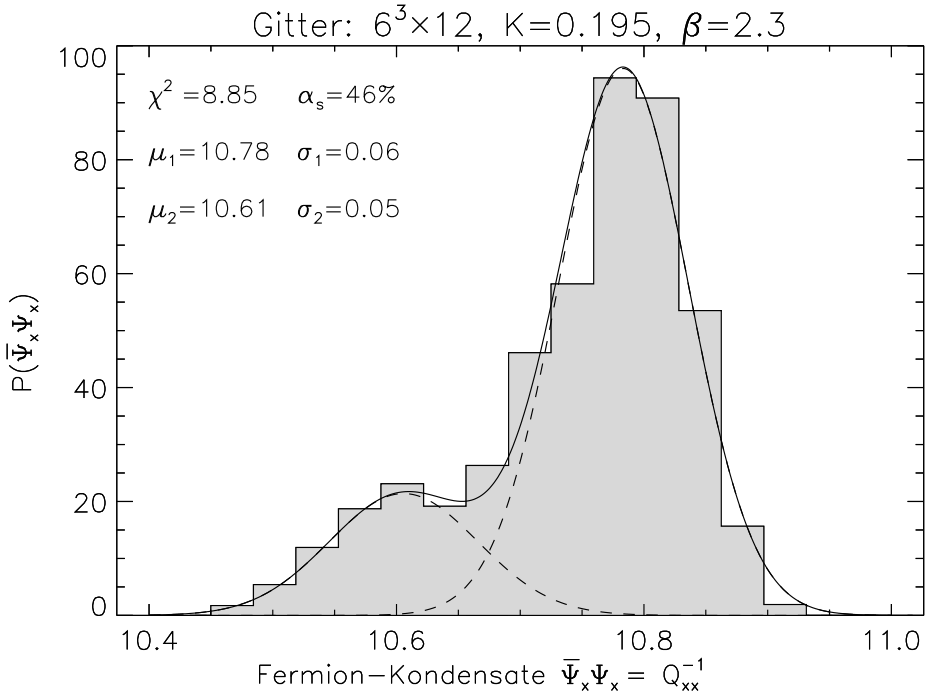


Abbildung 7.5: Verteilung der Fermion-Kondensate $Q_{xx}^{-1} = \bar{\Psi}_x \Psi_x$. Die durchgezogene Linie ist ein Fit mit zwei Gauß-Kurven. α_s bezeichnet das Signifikanzniveau des χ^2-Anpassungstests.

Um die Abweichung von der Normalverteilung auch bei zeitlich antiperiodischen Randbedingungen und mit längeren Polynomen zu untersuchen, hat J. Westphalen einen unabhängigen Lauf bei gleicher Gittergröße und Parametern (K, β) durchgeführt. Die ursprünglichen Konfigurationen wurden mit relativ kurzen Polynomen erzeugt $P_{0.00003,3.7}(22, 66, 102)$. Dies äußert sich schon darin, daß der Mittelwert für die Korrekturfaktoren bei lediglich 0.53 lag. Auch beim zweiten Lauf zeigte sich eine signifikante Abweichung von der Normalverteilung. Der Verlauf entspricht dem in Abbildung 7.5, was auch zu erhoffen war, da das Fermion-Kondensat wenig von finite-size-Effekten und damit von den Randbedingungen beeinflußt wird.

Eine Vorstudie wurde für den Hopping-Parameter $K = 0.1975$ auf einem $4^3 \times 8$ Gitter unternommen. Zum einen ist die Verteilung der kleinsten Eigenwerte für diesen Fall gutmütiger als noch bei $K = 0.195$ auf $6^3 \times 12$, obwohl das Gitter kleiner und der Hopping-Parameter größer geworden sind. Zum anderen konnte im Rahmen dieser Vorstudie keine signifikante Abweichung des Fermion-Kondensates von der Normalverteilung festgestellt werden. Beides spricht dafür, daß man mit diesem Hopping-Parameter schon jenseits des kritischen Bereiches liegt, d. h. $K_{susy}^{cr} = 0.196(1)$. Der

interessante Bereich oberhalb von 0.195 wird zur Zeit von R. Kirchner und J. Westphalen näher untersucht.

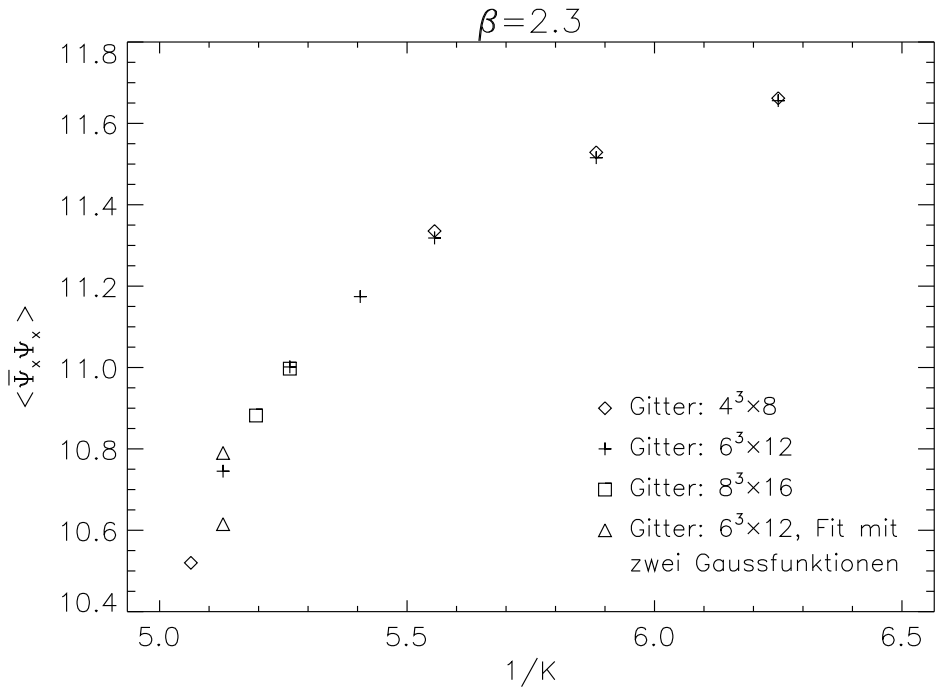


Abbildung 7.6: Erwartungswerte des Fermion-Kondensates. Die Fehler sind kleiner als die Symbole.

Alle gemessenen Fermion-Kondensate bei $\beta = 2.3$ sind in Abbildung 7.6 gegen $1/K$ eingetragen, um einen direkten Vergleich mit der nackten Gluino-Masse

$$m_0 = \frac{1}{2a}\left(\frac{1}{K} - \frac{1}{K^{cr}}\right) \tag{7.7}$$

zu ermöglichen. Sollte sich, wie man nach Abbildung 7.5 vermuten kann, die Koexistenz zweier Phasen bei $K = 0.195$ andeuten, so steht der Verlauf des Fermion-Kondensates im Einklang mit dem Phasendiagramm in Abbildung 7.1.

Bei der Plaquette-Verteilung konnte für alle getesteten Parameter keine signifikante Aufspaltung festgestellt werden (siehe z. B. Abbildung 7.7). Dies wird man nach der Argumentation von [KS97] auch nicht erwarten.

7.1.5 Ausnahmekonfigurationen

Mit stetiger Annäherung an den kritischen Hopping-Parameter, insbesondere bei den Läufen $K = 0.1925$ und $K = 0.195$ auf $6^3 \times 12$ Gittern, traten vermehrt Ausnahmekonfigurationen auf, d. h. Konfigurationen, deren kleinster Eigenwert von $\tilde{Q}[U]^2$

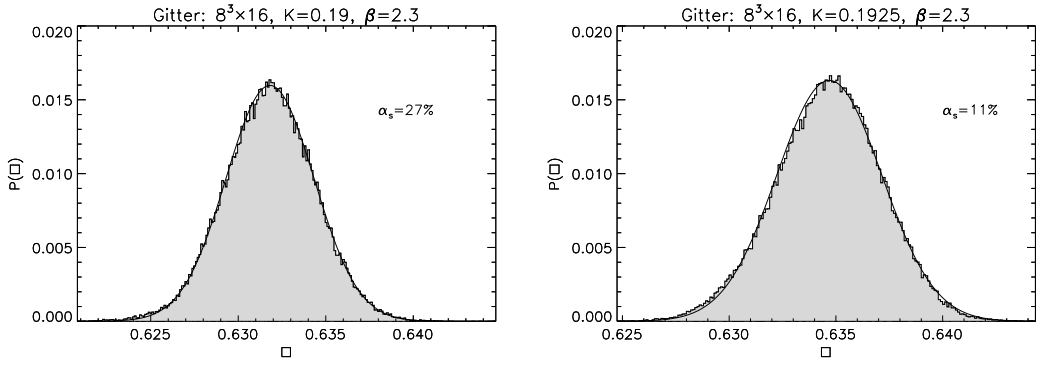


Abbildung 7.7: Häufigkeitsverteilung der Plaquette. Die durchgezogenen Linien stammen von Fits an eine Gauß-Kurve. α_s bezeichnet das Signifikanzniveau des χ^2-Anpassungstests. Für diesen Anpassungstest wurden nur Plaquette-Werte von unabhängigen Konfigurationen berücksichtigt, d. h., die Teststatistik ist sehr viel geringer, als es das Histogramm vermuten läßt.

circa vier bis sieben Größenordnungen kleiner waren als der Mittelwert der kleinsten Eigenwerte. Eigentlich sind solche Konfigurationen durch den Faktor

$$\left[\det \tilde{Q}[U]^2\right]^{\frac{1}{4}} \tag{7.8}$$

im Pfadintegral so stark unterdrückt, daß sie selbst in umfangreichen Produktionsläufen kaum auftreten sollten. Bedingt durch die Abweichung der Polynomapproximation $P_n(x)$ für $x \leq \epsilon$ bekommen diese Konfigurationen aber durch den Updater ein zu großes Gewicht (siehe Abschnitt 5.2.5). Desweiteren begünstigt der Umstand, daß im allgemeinen nur noch ein Noisy Correction Step mit dem zweiten Polynom $R_m(x)$ durchgeführt wird, die Übergangswahrscheinlichkeit von einer normalen zu einer Ausnahmekonfigurationen.

Abbildung 7.8 zeigt den Vergleich zwischen dem Spektrum der 64 kleinsten Eigenwerte für eine gewöhnliche Konfiguration und für eine Ausnahmekonfiguration. Alle untersuchten Ausnahmekonfigurationen zeigen hierbei dieselbe Struktur. Zwei gleiche Eigenwerte (Alle Eigenwerte von $\tilde{Q}[U]^2$ sind gemäß (2.14) zweifach entartet.) liegen etliche Größenordnungen unter dem „Bulk“ von Eigenwerten, deren Werte mit einer gewöhnlichen Konfiguration vergleichbar sind. Inwieweit diese Ausnahme-Eigenwerte mit dem Tunneln zwischen verschiedenen topologischen Ladungssektoren zusammenhängen, wird zur Zeit von R. Kirchner untersucht.

Damit stellen die Spektren von Ausnahmekonfigurationen aber kein Problem für den hybriden Meßkorrekturalgorithmus dar. In der Regel müssen einfach nur zwei bis vier Eigenwerte mehr als üblich berechnet und korrigiert werden. Im Gegensatz zu vielen quenched Simulationen ist man nicht darauf angewiesen, Ausnahmekonfigurationen gesondert zu behandeln [BDE98]. Allerdings sind die Konditionszahlen von $\tilde{Q}[U]^2$ bei Ausnahmekonfigurationen für jeden Inversionsalgorithmus problematisch. Aus

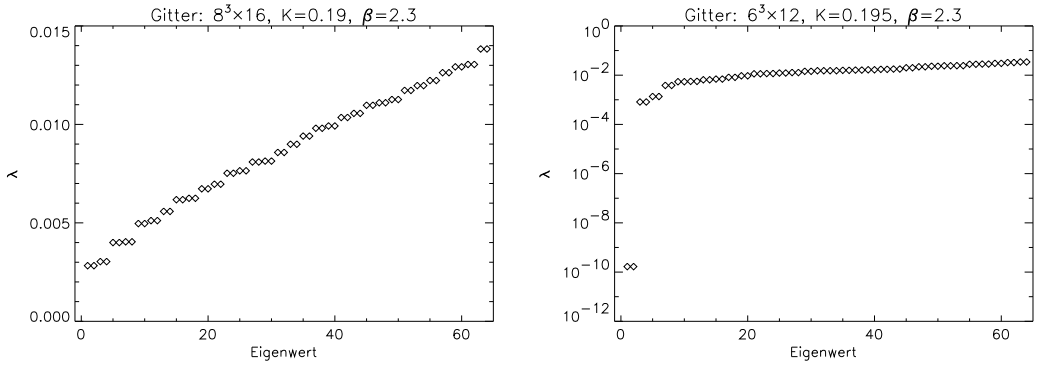


Abbildung 7.8: Die 64 kleinsten Eigenwerte für eine gewöhnliche Konfiguration und für eine Ausnahmekonfiguration.

diesem Grund sollte bei der Berechnung von fermionischen Erwartungswerten ähnlich wie bei der Meßkorrektur vorgegangen werden, d. h., die Inversion wird auf dem zu den Ausnahmeeigenwerten gehörenden Eigenraum „von Hand“ ausgeführt und der verbleibende, numerisch gutmütige Rest wird wie gehabt durch ein konjugiertes Gradientenverfahren gelöst.

7.1.6 Fluß des kleinsten Eigenwertes

Am kritischen Punkt sollte mit dem Verschwinden der Gluino-Masse auch der kleinste Eigenwert von $\tilde{Q}[U]^2$ auf einem endlichen Gitter in ein Minimum laufen. Für die Extrapolation zum Phasenübergang thermalisiert man zuerst einen Satz von Konfigurationen $\{U\}$ für einen bestimmten Hopping-Parameter K_0, der nahe am vermuteten kritischen Hopping-Parameter K^{cr} liegt. Man faßt nun die Fermion-Matrix als eine Funktion des Hopping-Parameters auf und berechnet jeweils für den gewünschten Hopping-Parameter K den kleinsten Eigenwert von $\tilde{Q}_K[U]^2$ für jede Konfiguration aus dem Satz $\{U\}$. Zwar sind die Konfigurationen $\{U\}$ für den jeweiligen Hopping-Parameter K nicht repräsentativ, allerdings immer noch besser geeignet als Konfigurationen, die mittels der quenched Approximation generiert wurden. Man gelangt somit zu einer Aussage für die „pseudo-quenched“ Fermion-Masse, die verläßlicher ist als reine quenched Rechnungen, da man zumindest teilweise die Dynamik der Fermionen berücksichtigt hat. Die quenched Konfigurationen liefern ein deutlich zu großes K^{cr}, und auch bei dieser pseudo-quenched Methode muß man ein ähnliches Verhalten erwarten.

Als ein geeigneter Ausgangspunkt für diese Extrapolation erscheinen die Konfigurationen vom $8^3 \times 16$ Gitter bei $K = 0.1925$. In Abbildung 7.9 ist das Ergebnis für 64 typische Eichkonfigurationen festgehalten. Die kleinsten Eigenwerte sind ab einem Wert von $K = 0.20 \sim 0.2025$ mit einer verschwindenden Fermion-Masse verträglich. Interessanterweise bleibt die so bestimmte pseudo-quenched Masse dann für größere K weiterhin mit Null verträglich, wie es auch in [EHN98] vorgeschlagen wird.

Mit dieser Extrapolationsmethode kann man den Fluß der kleinsten Eigenwerte bei Annäherung an den kritischen Punkt verfolgen (s. Abbildung 7.10). Dieser Fluß in $1/K$ zeigt deutlich, daß jenseits von $K \geq 0.2025$ die Verteilung des kleinsten Eigenwerts qualitativ anders wird.

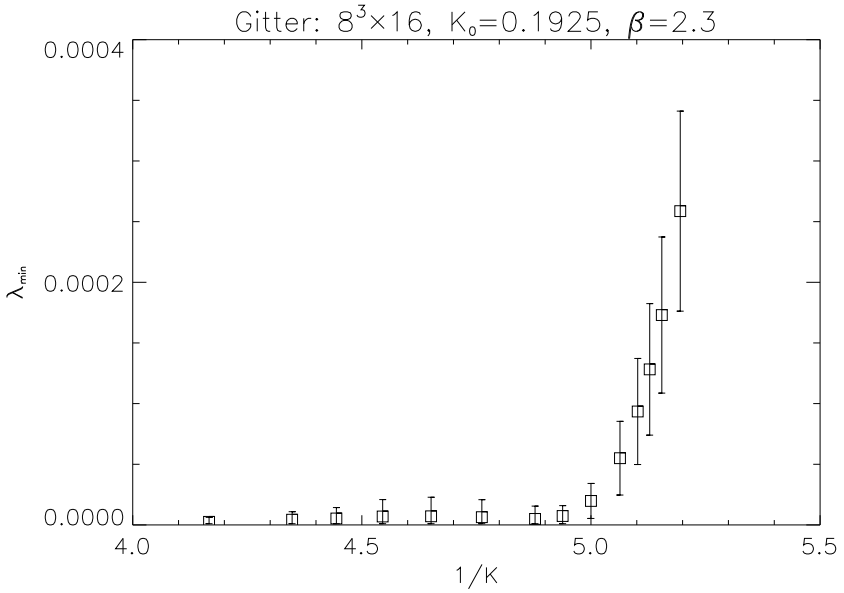


Abbildung 7.9: Mittelwert und Streuung des kleinsten Eigenwertes von $\tilde{Q}_K[U]^2$ für 64 bei $K_0 = 0.1925$ thermalisierte Konfigurationen.

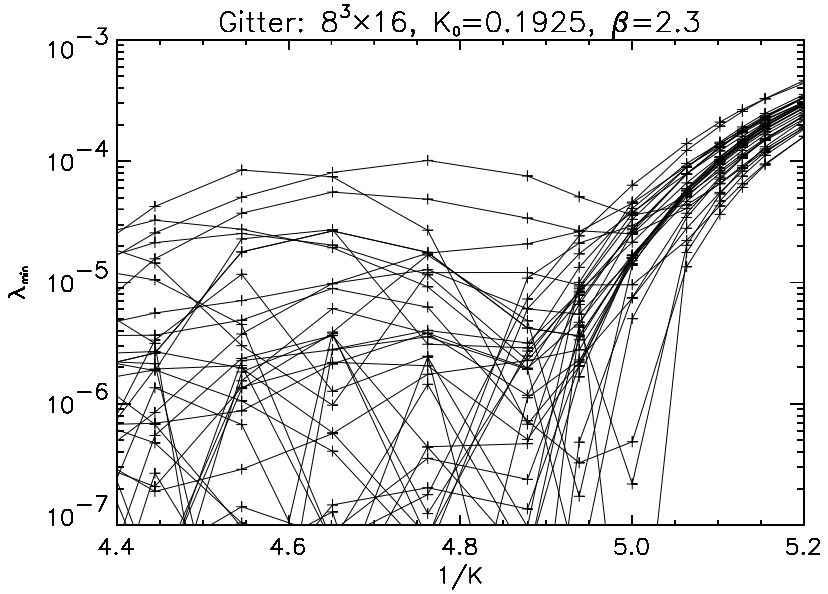


Abbildung 7.10: Fluß des kleinsten Eigenwertes von $\tilde{Q}_K[U]^2$ für 32 Konfigurationen, die bei $K_0 = 0.1925$ thermalisiert wurden.

7.1.7 Statisches Potential

Die Energie des Eichfeldes zwischen zwei statischen Farbladungen in der fundamentalen Darstellung (Quark-Antiquark-Paar) wird durch das statische Potential

$$V(R) = -\lim_{T\to\infty} \frac{1}{T} \log W(R,T) \tag{7.9}$$

beschrieben. Dabei sind die Wilson-Loops $W(R,T)$ als Erwartungswerte geschlossener $R \times T$ Rechteckschleifen

$$\begin{aligned} W(R,T) &= < W_{x,1}(R,T) + W_{x,2}(R,T) + W_{x,3}(R,T) > \qquad (7.10)\\ \text{mit} \quad W_{x,\mu}(R,T) &= \mathrm{Tr}\left(\prod_{i=0}^{T-1} U^{\dagger}_{x+i\hat{4},4} \prod_{i=0}^{R-1} U^{\dagger}_{x+i\hat{\mu}+T\hat{4},\mu} \prod_{i=T-1}^{0} U_{x+R\hat{\mu}+i\hat{4},4} \prod_{i=R-1}^{0} U_{x+i\hat{\mu},\mu} \right) \end{aligned}$$

definiert. Zieht man das Quark-Antiquark Paar sehr weit auseinander, so kann aus ihrem Potential die String-Tension σ mit

$$\sigma = \lim_{R\to\infty} \frac{1}{R} V(R) \tag{7.11}$$

bestimmt werden. Ist die String-Tension ungleich Null, so wächst das Potential für große Abstände linear in R an $V(R) \overset{R\to\infty}{\sim} \sigma R$. Man spricht in diesem Fall vom statischen Confinement (siehe Abbildung 7.11). Auf der anderen Seite deutet eine verschwindende String-Tension Screening-Effekte von fundamentalen Farbladungen durch die dynamischen Gluinos an, welche in zwei Dimensionen für dieses Modell im Falle von masselosen Gluinos theoretisch vorausgesagt wurden [GKMS96]. Wird die Supersymmetrie hingegen durch eine Gluino-Masse gebrochen, verschwindet auch im zweidimensionalen Modell das Screening, und es gibt wieder ein statisches Confinement für Farbladungen in der fundamentalen Darstellung.

Die String-Tension kann im Rahmen einer Gittersimulation nur durch eine Extrapolation zu $R,T \to \infty$ bestimmt werden. Dies kann zum einen durch einen Fit an den asymptotischen Teil des Potentials für große R,T geschehen. Zum anderen kennt man aus den Stark-Kopplungsentwicklungen in der reinen Eichtheorie näherungsweise das Verhalten der Wilson-Loops $W(R,T)$ mit endlicher Fläche

$$W(R,T) = C \exp\left(-\sigma T R + \mu(R+T)\right). \tag{7.12}$$

Um den konstanten Faktor und die „Peripherie“-Terme $\mu(R+T)$ zu eliminieren, definiert man die Creutz-Ratios

$$\chi(R,T) = -\log\left(\frac{W(R,T)W(R-1,T-1)}{W(R,T-1)W(R-1,T)}\right). \tag{7.13}$$

Würden Wilson-Loops exakt durch Gleichung (7.12) beschrieben, so wäre $\chi(R,T)$ schon bei endlichen R,T gleich der String-Tension. Allerdings kommen in der vollen Theorie zur rechten Seite der Gleichung (7.12) noch Terme hinzu. Man kann in

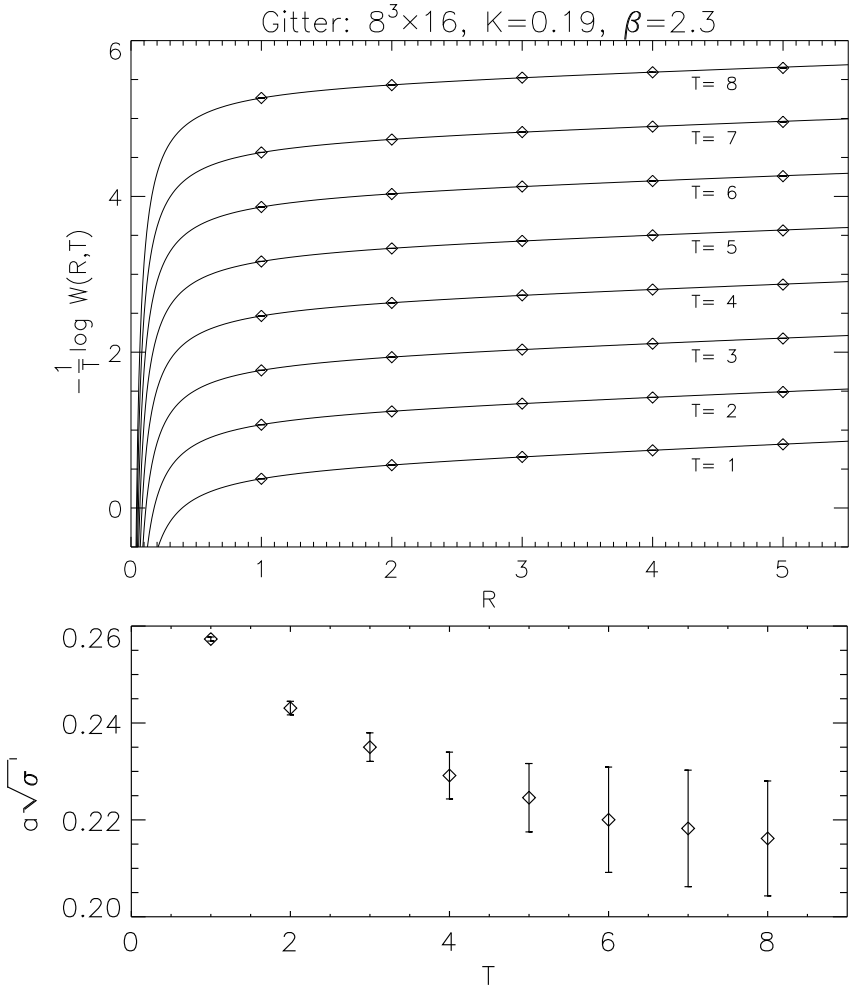


Abbildung 7.11: Statisches Potential $V_T(R) \stackrel{\text{def}}{=} -\frac{1}{T}\log W(R,T)$ für verschiedene Werte von T. Die durchgezogene Linie entspricht einem durch die Stark-Kopplungsentwicklung motivierten Fit mit $V(R) = V_0 + \sigma R - a/R$ im Intervall $R \in [1,6]$. Zur besseren Übersicht wurde auf die Meßwerte eine T-abhängige Konstante addiert.

der Praxis aus den Creutz-Ratios erst durch eine Extrapolation zu großen R, T hin auf die String-Tension schließen. Nach Gleichung (7.9) wird ein Wilson-Loop $W(R, T_f - T_i)$ für den Fall, daß die Zeiten T_f und T_i weit genug auseinander liegen, durch den niederenergetischen Gluon-Zustand in Gegenwart eines Quark-Antiquark-Paares im Abstand R dominiert

$$W(R, T_f - T_i) \sim e^{-(T_f - T_i)V(L)}. \tag{7.14}$$

Auch in diesem Fall kann man wieder die Überlappung mit dem Grundzustand durch Smearing verbessern [BSS95]. Ziel ist es, das Signal/Rausch-Verhältnis zu erhöhen und gleichzeitig das asymptotische Verhalten für kleinere R, T extrahieren zu können. Natürlich bieten sich an dieser Stelle wieder die beiden eichinvarianten Algorithmen, das Teper-Blocking und das APE-Smearing, an. In Tests zeigte sich, daß bei einer sorgfältig Optimierung der Parameter (N_{ape}, ϵ) das APE-Smearing

dem Teper-Blocking überlegen ist. Wie in Abbildung 7.11 an Hand der Creutz-Ratios zu sehen, verbessert Smearing sowohl das Signal/Rausch-Verhältnis als auch die Projektion auf den Grundzustand deutlich. Dasselbe gilt auch für den Fall, daß man die String-Tension durch Fits an das Potential gewinnt.

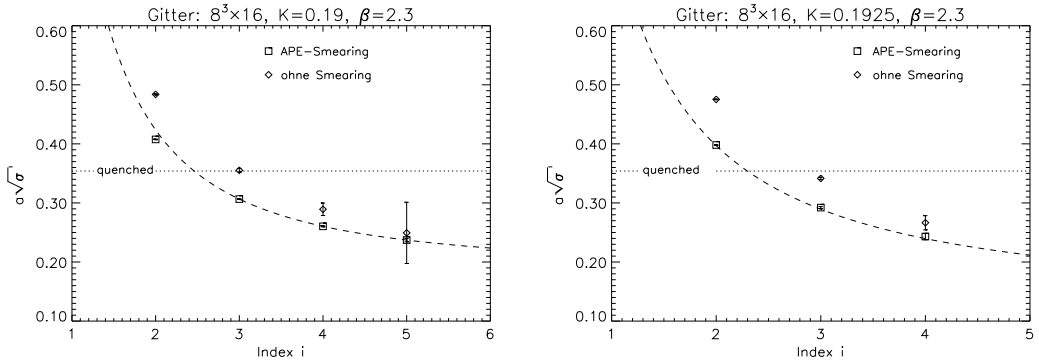


Abbildung 7.12: Die Wurzel der String-Tension σ in Abhängigkeit des Indexes des zugrunde liegenden Creutz-Ratios $\chi(i,i)$. Der Wert für reine SU(2) Eichtheorie $a\sqrt{\sigma} = 0.3690(30)$ stammt aus [MT87].

Trotz Smearing ist eine seriöse Extrapolation zu $R, T \to \infty$ auf kleineren Gittern als $8^3 \times 16$ nicht möglich. Basierend auf den mit dem APE-Algorithmus verschmierten Konfigurationen wird die String-Tension auf drei verschiedene Weisen berechnet:

- durch die Creutz-Ratios,
- mit einem Potential-Fit, wie er in Abbildung 7.11 zu sehen ist und
- durch ein differenzierteres Zwei-Fit-Verfahren, das S. Luckmann auf denselben Daten durchgeführt hat [BSS95].

Das Zwei-Fit-Verfahren ist nach den Daten aus Tabelle 7.1 am besten geeignet. Einer genaueren Bestimmung der String-Tension bei $K = 0.1925$ stehen weniger die statistischen Fehler im Wege als vielmehr die deutlich zu Tage tretenden finite-size-Effekte, die das nutzbare Fit-Intervall verkleinern. Unter anderem wird deshalb zur Zeit eine Simulation für den gleichen Hopping-Parameter auf einem $12^3 \times 24$ Gitter durchgeführt.

Abschließend läßt sich sagen, daß mit den zur Verfügung stehenden Daten die Frage nach einem möglichen Verschwinden der String-Tension im supersymmetrischen Limes nicht zu klären ist. Bei endlicher Gluino-Masse hingegen erhält man die für Confinement typische Potentialform, wie sie z. B. in Abbildung 7.11 zu sehen ist.

7.2 Massen des Supermultipletts

Gelingt die Restauration der Supersymmetrie auf dem Gitter nach dem von Curci und Veneziano vorgeschlagenen Schema, so muß die Entartung der aus den leich-

K	Creutz Ratio	Potential Fit	Zwei-Fit (Daten von S.L.)
0.19	0.189(19)	0.216(12)	0.21(1)
0.1925	0.123(33)	–	0.10(3)

Tabelle 7.1: Wurzel der String-Tension in Einheiten des Gitterabstandes a auf dem $8^3 \times 16$ Gitter. Alle Fehler wurden durch das Jackknife-Verfahren bestimmt. Der einfache Potential-Fit liefert bei $K = 0.1925$ keine konsistenten Daten mehr.

testen Bindungszuständen aufgebauten Supermultipletts im Rahmen einer Gittersimulation beobachtbar sein. Dem Veneziano-Yankielowicz-Vorschlag für eine effektive Theorie folgend, besteht ein mögliches Multiplett aus den Zuständen a-η', a-f_0 und dem Gluino-Glueball. Durch die Existenz eines leichten, rein gluonischen 0^+ Zustandes motiviert, fügen Farrar, Gabadadze und Schwetz (FGS) ein weiteres chirales Supermultiplett zur effektiven Wirkung hinzu, das aus einem Gluino-Glueball und den beiden Gluebällen $0^+, 0^-$ besteht. Diese effektive Wirkung erlaubt zudem ausdrücklich Massenmischung zwischen a-$f_0 \Leftrightarrow 0^+$Glueball und für a-$\eta' \Leftrightarrow 0^-$Glueball. Zwar ist es kein notwendiges Kriterium für die effektive FGS-Wirkung, allerdings würde die Beobachtung eines Mischungswinkels ungleich Null ein starkes Indiz für diese Form der Wirkung sein. Kann die Restauration der Supersymmetrie beobachtet werden, so ist natürlich die Umkehrrichtung von besonderem Interesse, um die Massenaufspaltung bei einer durch die endliche Gluino-Masse weich gebrochenen Supersymmetrie zu untersuchen.

7.2.1 Erste Teststudie: $4^3 \times 8$ Gitter

Die Bestimmung der Massen von Bindungszuständen auf einem so kleinem Gitter wie $4^3 \times 8$ birgt Risiken. Zum einen benötigt man sehr gute Smearing-Algorithmen, da der Grundzustand durch die kleine zeitliche Ausdehnung des Gitters nicht sehr gut aus der Zeitscheibenkorrelationsfunktion herausprojiziert werden kann. Zum anderen muß man nach der Faustregel $L \geq 6m^{-1}$ schon für Massen m, die kleiner als 1.5 sind, mit deutlichen finite-size-Effekten rechnen.

Gerade bei den $4^3 \times 8$ Gittern, die in der Anfangszeit des Projekts benutzt wurden, ist als Smearing-Algorithmus nur das recht grobe Teper-Blocking eingesetzt worden, so daß die erste Bedingung für die Simulation auf solch kleinen Gittern kaum erfüllt wird. Zudem liegt in der reinen SU(2) Eichtheorie die Masse des 0^+ Glueballs schon bei $m_{gg}^{0^+} = 1.1(1)$ [CMT87], so daß man (auch) für kleine Hopping-Parameter mit starken finite-size-Effekten rechnen muß.

Behält man diese Argumente im Auge, so verwundert es nicht, daß auf dem $4^3 \times 8$ Gitter mit den in Tabelle 7.2 aufgelisteten Meßwerten für die Massen des Veneziano-Yankielowicz-Multipletts noch keine Entartung beobachtbar ist. Desweiteren liegt der größte Hopping-Parameter mit $K = 0.18$ noch recht weit vom vermuteten kriti-

Zustand	Smearing-Verfahren	Fit-Intervall	Masse beim Hopping-Parameter		
			0.16	0.17	0.18
0^+ Glueball	Level 2 Teper-Blocking	$[1,2]$	1.02(12)	1.00(10)	1.19(14)
Gluino-Glue	Level 1 Teper-Blocking	$[1,2]$	2.61(12)	2.47(10)	2.31(10)
a-η'	-	$[2,3]$	1.70(3)	1.44(3)	1.02(2)
a-f_0	-	$[1,3]$	2.33(63)	1.93(15)	1.64(14)

Tabelle 7.2: Massen des Veneziano-Yankielowicz-Multipletts und des 0^+ Glueballs. Alle Fehler wurden mit dem Jackknife-Verfahren bestimmt.

schen Wert $K^{cr}_{susy} = 0.196(1)$ entfernt. Allerdings steht der Simulation von Werten $K > 0.18$ auf diesem Gitter die a-η' Masse im Wege. Sie würde deutlich unter eins fallen. Man müßte mit nicht tolerierbaren finite-size-Effekten rechnen.

Betrachtet man den Verlauf der Massen in Abbildung 7.13, so läßt sich „mit dem Auge“ nicht erahnen, wie zumindest die Massen von a-η', a-f_0 und des Gluino-Glueballs im kritischen Bereich zusammentreffen könnten. Der interessante Teil der Dynamik, die zur Restauration der Supersymmetrie führen könnte, liegt jenseits der mit einem $4^3 \times 8$ Gitter simulierbaren Hopping-Parameter.

Trotzdem zeichnen sich einige wichtige Effekte ab. Die Masse des 0^+ Glueballs wird im untersuchten Bereich nicht durch die Dynamik der Gluinos beeinflußt; innerhalb der statistischen Fehler stimmt sie mit der Werten der reinen SU(2) Eichtheorie überein. Die a-η' Masse wird bis auf 1% vom verbundenen Anteil bestimmt, d. h., im wesentlichen gilt $m^{0^-}_{\tilde{g}\tilde{g}} = m_{a\text{-}\pi}$. Anders sieht es beim a-f_0 Zustand aus. Die Massen zum verbundenen Anteil, der dem unphysikalischen a-σ Teilchen entspricht, liegen im Bereich 2.7 bis 3.0. Der unverbundene Anteil liefert hier einen signifikanten Beitrag zur Zeitscheibenkorrelationsfunktion, allerdings auch ein überproportionales Rauschen. Teilweise über 90% des statistischen Fehlers vom a-f_0 Zustand stammen vom unverbundenen Anteil.

7.2.2 Ergebnisse für das $6^3 \times 12$ Gitter

Da praktisch jeder Lauf auf dem $4^3 \times 8$ Gitter allein schon wegen der kleinen 0^+ Glueball-Masse von finite-size-Effekten betroffen war, wurden auf dem $6^3 \times 12$ nochmals Simulationen für die Hopping-Parameter $K = 0.16, 0.17, 0.18$ durchgeführt. Darüber hinaus konnten auf diesem Gitter die Massen für $K = 0.185, 0.19$ bestimmt werden. Die beiden Produktionsläufe mit $K = 0.1925, 0.195$ dienten nur noch der Suche nach dem Phasenübergang, da die finite-size-Effekte bei einer Massenbestimmung nicht mehr tolerierbar sind.

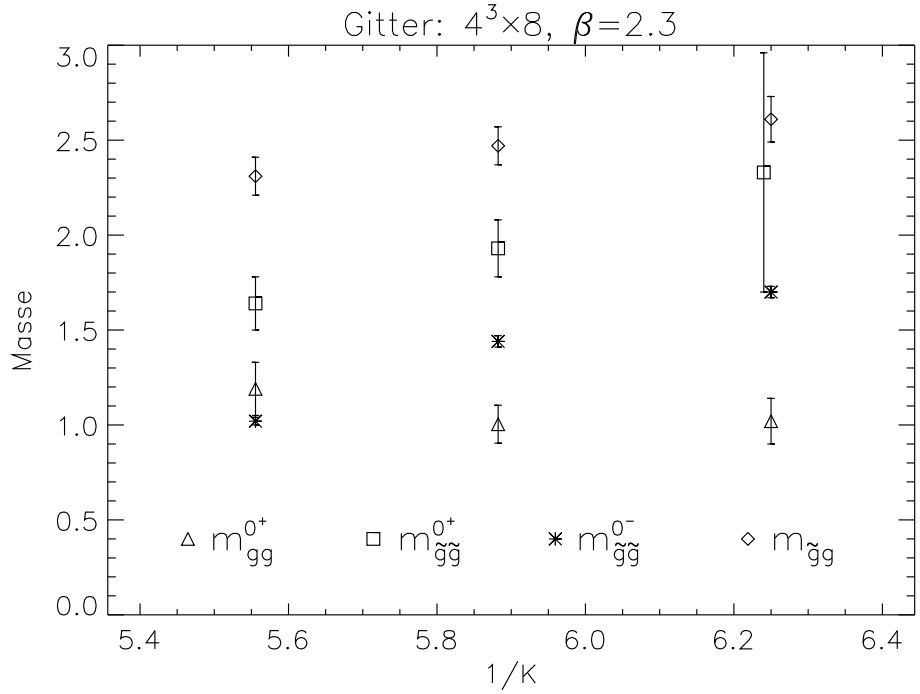


Abbildung 7.13: Massen des chiralen Multipletts für verschiedene Werte des Hopping-Parameters auf einem $4^3 \times 8$ Gitter.

Die integrierten Autokorrelationszeiten der verschiedenen Zeitscheibenerwartungswerte streben mit Annäherung an den kritischen Hopping-Parameter immer weiter auseinander. Besonders lange Korrelationszeiten resultieren dabei aus der Beteiligung der Plaquette an einer Observablen. Wie in Abbildung 7.14 dargestellt, reichen für die rein fermionische Zeitscheibenkorrelationsfunktion des a-η' Zustands schon ca. 50 Updates für eine Dekorrelation aus, während die rein bosonische Observable für den 0^+ Glueball mit einer Autokorrelationszeit von ca. 400 aufwartet (Allerdings ist die kurze Korrelationszeit für a-η' teilweise auch dadurch bedingt, daß bei jeder Messung die Quelle zufällig über das Gitter verteilt wird.).

Die Smearing-Methoden zur Bestimmung der Massen sind für die Hopping-Parameter $K = 0.16, 0.17, 0.18$ mit denen des $4^3 \times 8$ Gitters identisch. Wiederum waren die optimalen Blocking-Level beim Teper-Algorithmus 2 für den 0^+ Glueball bzw. 1 für den Gluino-Glue Zustand. Allerdings sind, wie in Tabelle 7.3 zu sehen, die Massen für den Gluino-Glueball auf dem $6^3 \times 12$ Gitter doch deutlich kleiner als noch auf dem $4^3 \times 8$ Gitter, was andeutet, daß die Projektion auf den Grundzustand durch das Teper-Blocking allein nicht ausreichend ist. Ebenso läßt das Signal/Rausch-Verhältnis keine Vergrößerung des Fit-Intervalls zu.

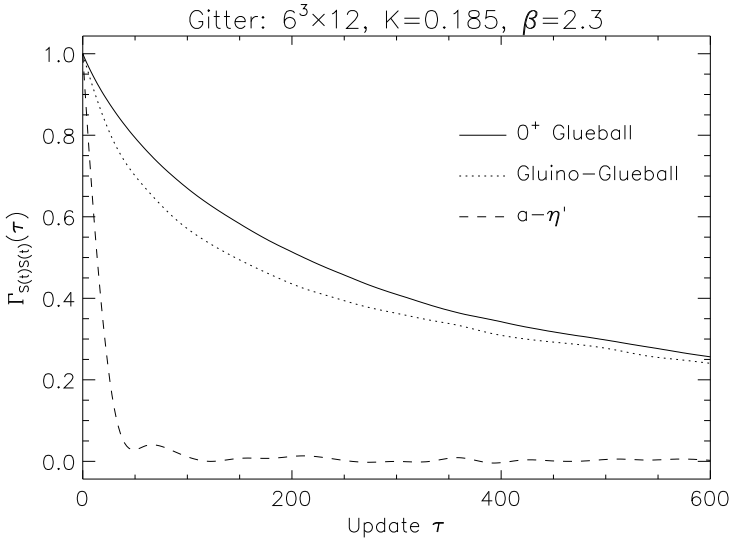


Abbildung 7.14: Korrelationsfunktionen für die grundlegenden Zeitscheibenkorrelationsfunktionen bei $\triangle t = 0$.

Diese Beobachtung war der Auslöser dafür, neue Smearing-Verfahren auszuprobieren. Schließlich bewährte sich die Kombination aus APE/Jacobi-Smearing, wie sie in Abschnitt 6.4.2 vorgestellt wurde. Neben einem deutlich besseren Signal/Rausch-Verhältnis, das Fits im Intervall $\Delta t \in [3,4]$ möglich macht, ist die Projektion auf den Grundzustand besser. Dieses äußert sich bei optimierten Smearing-Parametern (N_{ape}, ϵ) und (N_{jacobi}, κ_S) darin, daß die effektive Masse $m_{\tilde{g}g}(L_t, 1, 2)$ im allgemeinen nur noch $10\% \sim 15\%$ größer ist als $m_{\tilde{g}g}(L_t, 3, 4)$.

Zustand	Fit-Intervall	Masse beim Hopping-Parameter				
		0.16	0.17	0.18	0.185	0.19
0^+ Glueball	$[1,2]$	0.93(11)	0.86(13)	0.95(10)	0.85(6)	$0.83(15)^\dagger$
Gluino-Glue	$[1,2]$	2.28(3)	2.07(4)	1.93(5)	$1.394(77)^*$	$1.05(20)^*$
a-η'	$[4,5]$	1.68(2)	1.46(2)	1.155(11)	0.9414(78)	0.594(14)
a-f_0	$[1,3]$	2.16(47)	1.88(27)	1.49(13)	$1.11(17)^\ddagger$	–

Tabelle 7.3: Massen des Veneziano-Yankielowicz-Multipletts und des 0^+ Glueballs. Wenn möglich, wurde ein besseres Fit-Intervall gewählt, $\dagger \doteq [2,4]$, $* \doteq [3,4]$ und $\ddagger \doteq [2,3]$. Für $K = 0.19$ konnte die a-f_0 Masse nicht bestimmt werden, da daß Signal/Rausch-Verhältnis zu ungünstig war.

Betrachtet man die in Abbildung 7.15 eingetragenen Massen, so fällt auf, daß sie für $K = 0.185$ näher zusammenrücken. Vor allem die Gluino-Glue-Masse wird über-

proportional kleiner, was natürlich zum Teil auf das bessere Smearing-Verfahren zurückzuführen ist. Für $K = 0.19$ reicht das $6^3 \times 12$ Gitter schon wieder nicht mehr aus. Es ist zu erwarten, daß die Masse des a-η' Zustands hier deutliche finite-size-Effekte zeigt. Schon bei $K = 0.18$ war diese Masse auf $4^3 \times 8$ im Vergleich zum $6^3 \times 12$ Gitter um 15% zu klein. Desweiteren verschlechterte sich das Signal/Rausch-Verhältnis des Glueballs von $K = 0.185$ nach $K = 0.19$ merklich. Das ist ein Hinweis darauf, daß sich die räumliche Ausdehnung des Grundzustandes vergrößert hat. Es macht auf diesem Gitter keinen Sinn, über Blocking-Level zwei hinauszugehen, so daß auch für den Glueball das Gitter zu klein geworden ist.

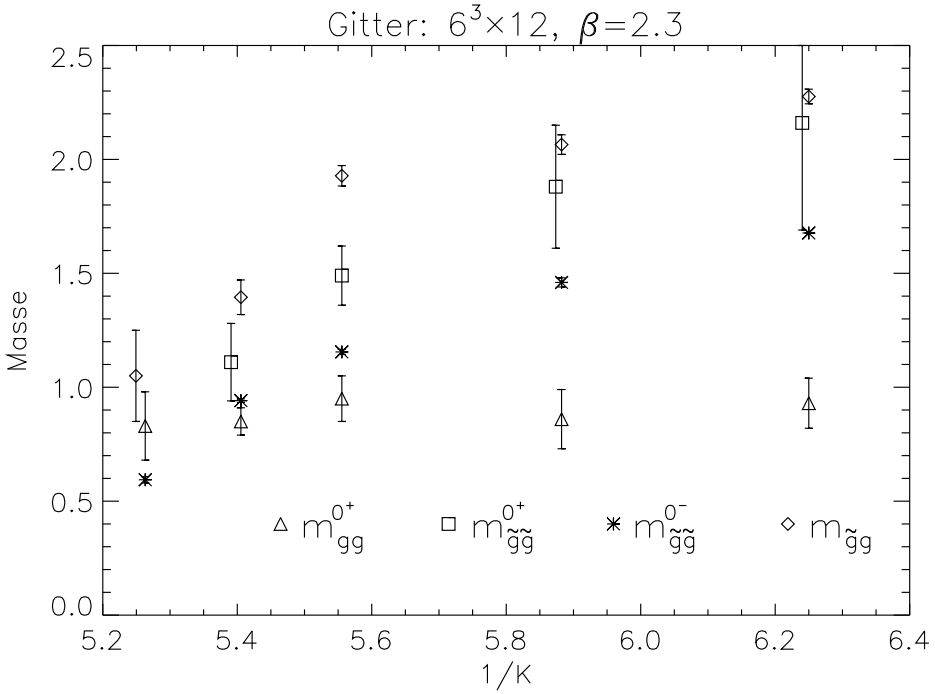


Abbildung 7.15: Massen des chiralen Multipletts für verschiedene Werte des Hopping-Parameters auf einem $6^3 \times 12$ Gitter.

7.2.3 Ergebnisse für das $8^3 \times 16$ Gitter

Die umfangreichsten Produktionsläufe zur Massenbestimmung wurden bei den Hopping-Parametern $K = 0.19$ und $K = 0.1925$ auf $8^3 \times 16$ Gittern durchgeführt. Dabei wurden sorgfältig ausgewählte Polynome benutzt, so daß eine explizite Meßkorrektur noch nicht berücksichtigt werden mußte.

Wie in Abbildung 7.16 zu sehen, bewährt sich auf diesem Gitter das kombinierte APE/Jacobi-Smearing für den Gluino-Glueball, da die effektive Masse nur noch wenig vom gewählten Fit-Intervall abhängt. Da für den a-η' Zustand kein Smearing eingesetzt wird, muß man für eine zuverlässige Bestimmung der Masse entweder Fit-Intervalle nahe bei $L_t/2$ wählen oder aber einen Fit für zwei Massen gemäß

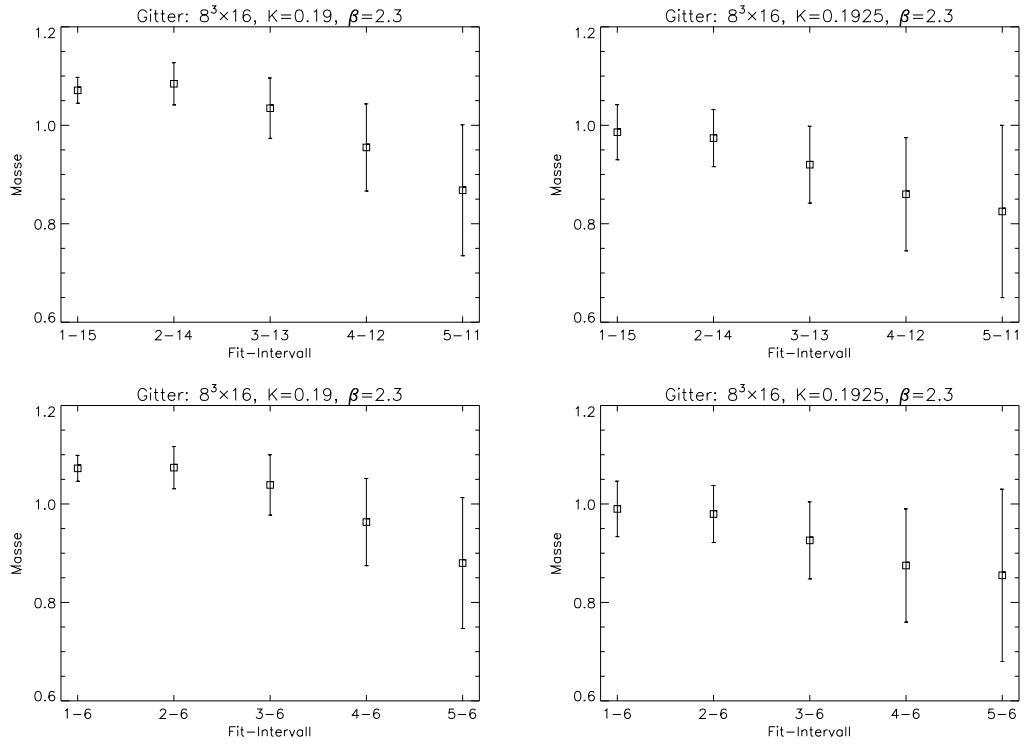


Abbildung 7.16: Fit mit einer Masse an die Zeitscheibenkorrelationsfunktion des Gluino-Glueballs in Abhängigkeit vom Fit-Intervall.

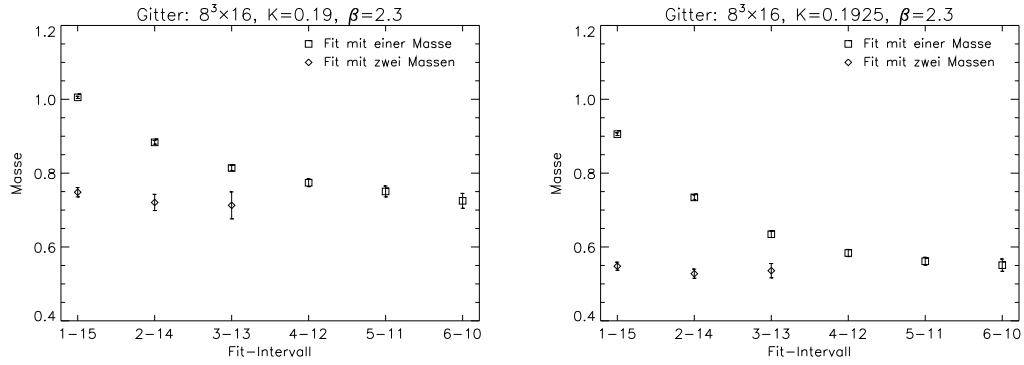


Abbildung 7.17: Die Masse des a-η' Teilchens aus der Zeitscheibenkorrelationsfunktion in Abhängigkeit vom Fit-Intervall.

Gleichung 6.5 durchführen. Allerdings ist auch diese Masse bei $K = 0.1925$ mit $m_{a\text{-}\eta'}(16,6,10) = 0.551(17)$ zu klein für das $8^3 \times 16$ Gitter. Die wirkliche Masse wird erfahrungsgemäß größer sein.

Problematisch bleibt der a-f_0 Zustand auf Grund des großen statistischen Fehlers des unverbundenen Anteils. Für dieses Teilchen kann nur die effektive $m_{a\text{-}f_0}(16,2,4)$ Masse bestimmt werden. Da hierbei ebenfalls noch kein Smearing eingesetzt wird, beinhaltet die effektive Masse deutliche Beiträge des ersten angeregten Zustands aus diesem Kanal. Zum Vergleich: Beim a-η' Zustand sorgen diese Beiträge dafür, daß $m_{a\text{-}\eta'}(16,2,4)$ ca. $10\% - 20\%$ größer ist als $m_{a\text{-}\eta'}(16,6,10)$. Beim Glueball hingegen geben sowohl das APE-Smearing als auch das Teper-Blocking, wie schon in den Abbildungen 6.3 bzw. 6.5 gezeigt wurde, wenig Anlaß zur Sorge. Lediglich die optimalen Smearing-Radien bei $K = 0.1925$ deuten auf finite-size-Effekte für das $8^3 \times 16$ Gitter hin.

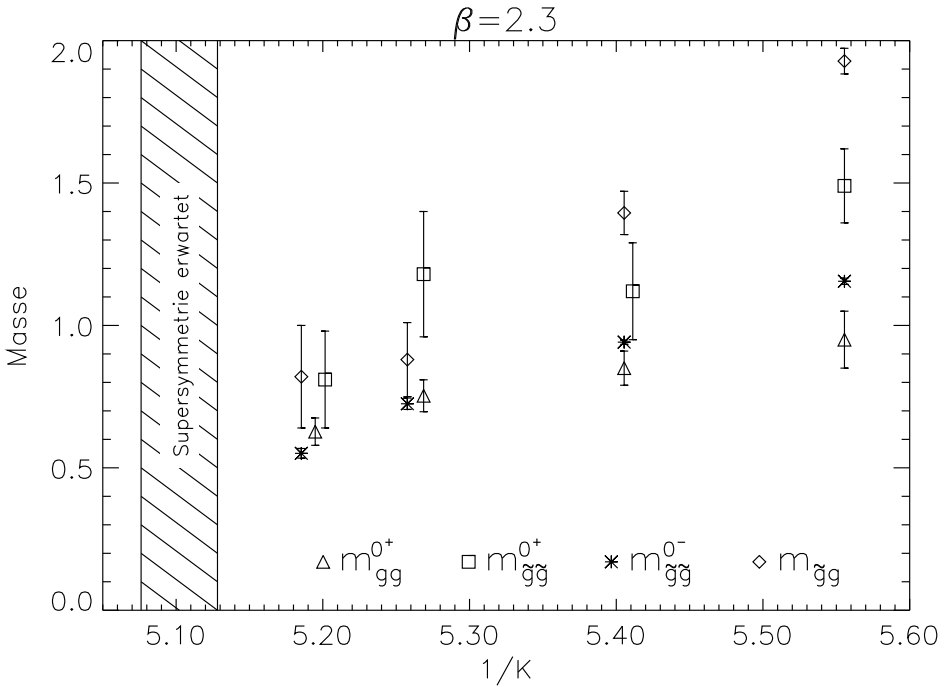


Abbildung 7.18: Massen des chiralen Multipletts und des 0^+ Glueballs für $K = 0.18, 0.185$ von einem $6^3 \times 12$ Gitter und für $K = 0.19, 0.1925$ von einem $8^3 \times 16$ Gitter.

Betrachtet man den Verlauf der Massen in Abbildung 7.18 bzw. Tabelle 7.4, so setzt sich der Trend vom $6^3 \times 12$ Gitter hin zu ähnlicheren Massen der Bindungszustände des Veneziano-Yankielowicz-Multipletts fort. Desweiteren fällt die Glueball-Masse sichtbar vom Wert der reinen SU(2) Eichtheorie ab. Ob der Glueball leichter bleibt als das Veneziano-Yankielowicz-Multiplett, wie es die effektive FGS-Wirkung voraussagt, läßt sich nicht sagen. Bei $K = 0.1925$ ist die Masse des a-η' Zustands zwar

signifikant kleiner, seine wirkliche Masse wird aber auf Grund der finite-size-Effekte höher liegen.

Eine Massenentartung läßt sich aber mit den zur Verfügung stehenden Daten noch nicht wirklich beobachten. Ein Grund hierfür ist mit Sicherheit, daß für den a-f_0 Zustand noch kein Smearing eingesetzt wird. Man kann erwarten, daß die eingetragenen Massen aus dem Fit-Intervall $[2, 4]$ zu groß sind und die wirkliche Masse besser zu den beiden anderen Konstituenten des Veneziano-Yankielowicz-Multipletts paßt. Wie in Abschnitt 6.3.2 gezeigt, funktioniert das Jacobi-Smearing für die a-π Masse sehr gut. Es ist zu hoffen, daß diese Aussage auch auf den a-f_0 Zustand übertragbar ist.

Über die beiden anderen Zustände des leichtesten chiralen Multipletts der FGS-Wirkung, dem 0^- Glueball und einem weiteren Spin $\frac{1}{2}$ Gluino-Glue Zustand, läßt sich zur Zeit noch wenig sagen. Der Vorschlag zu dieser Wirkung ist noch sehr jung. Mit der Arbeit an der Software zur Messung des 0^- Glueballs konnte S. Luckmann gerade erst beginnen. Ein Zwei-Massen-Fit für den Gluino-Glue Kanal ist auch noch weit davon entfernt, stabil und aussagekräftig zu sein.

Zur Zeit wird gerade Statistik auf einem $12^3 \times 24$ Gitter bei $K = 0.1925$ gesammelt. Dieser Lauf verspricht qualitativ bessere Aussagen zu ermöglichen, denn für ihn ist deutlich mehr Rechenzeit als für den vergleichbaren Lauf des $8^3 \times 16$ Gitters geplant. Zudem werden natürlich die finite-size-Effekte kleiner sein und die Smearing-Parameter können besser eingestellt werden, da die Grundzustandswellenfunktionen nicht mehr durch die Gitterränder „eingequetscht“ sind.

Zustand	Smearing-Verfahren	Fit-Intervall	Masse beim Hopping-Parameter	
			0.19	0.1925
0^+ Glueball	Level 3 Teper-Blocking	$[1, 2]$	0.753(56)	0.627(48)
Gluino-Glue	APE/ Jacobi	$[5, 6]$	0.87(13)	0.82(18)
a-η'	-	$[6, 10]$	0.725(20)	0.551(17)
a-f_0	-	$[2, 4]$	1.20(22)	0.81(17)
a-η'^*	-	$[2, 14]$	0.721(22) 1.387(66)	0.527(13) 1.282(26)

Tabelle 7.4: Massen des Veneziano-Yankielowicz-Multipletts und des 0^+ Glueballs. Die mit einem Stern gekennzeichnete Zeile enthält die Masse des Grundzustandes und die Masse des ersten angeregten Zustandes aus einem Zwei-Massen-Fit für das a-η' Teilchen.

7.2.4 Massenmischung im 0^+ Kanal

Eine wichtiges Unterscheidungskriterium für eine effektive Wirkung vom FGS-Typ gegenüber der Veneziano-Yankielowicz-Version ist, daß Massenmischung sowohl im 0^+ Kanal zwischen a-$f_0 \Leftrightarrow 0^+$Glueball als auch im 0^- Kanal zwischen den Zuständen a-$\eta' \Leftrightarrow 0^-$Glueball erlaubt ist.

Um den Mischungswinkel in einem Kanal zwischen den beiden Zuständen a und b mit den Zeitscheibenkorrelationsfunktionen $S_a(t)$ und $S_b(t)$ berechnen zu können, führt man die Kreuzkorrelationsmatrix

$$\Lambda_{ab}(\Delta t) = \begin{pmatrix} \Gamma^c_{aa}(\Delta t) & \omega\Gamma^c_{ab}(\Delta t) \\ \omega\Gamma^c_{ba}(\Delta t) & \omega^2\Gamma^c_{bb}(\Delta t) \end{pmatrix} \tag{7.15}$$

mit den Kreuzkorrelationsfunktionen

$$\begin{aligned} \Gamma^c_{ij}(\Delta t) &= < S_i(t_0)S_j(t_0+\Delta t) > - < S_i(t_0) >< S_j(t_0+\Delta t) > \\ \text{mit} &\quad i \in \{a,b\} \ \wedge \ j \in \{a,b\} \end{aligned} \tag{7.16}$$

ein. Die Konstante ω ist hierbei zunächst beliebig, kann aber später dazu ausgenutzt werden, die Beträge von höheren Zuständen auszulöschen. Diagonalisiert man diese Korrelationsmatrix, so werden ihre Eigenwerte bei großen Abständen Δt von den beiden leichtesten Zuständen des Kanals dominiert [C93]

$$\lambda_0(\Delta t, \omega) = f_0(\omega)e^{-E_0\Delta t}\left\{1 + O(E_1 - E_0)\right\} \tag{7.17}$$

$$\lambda_1(\Delta t, \omega) = f_1(\omega)e^{-E_1\Delta t}\left\{1 + O(\min(E_1 - E_0, E_2 - E_1))\right\}, \tag{7.18}$$

wobei ein unendlich großes Gitter vorausgesetzt wurde. Die Kreuzkorrelationsmatrix ist hermitesch und ihre beiden Eigenvektoren $\vec{v}_0(\Delta t)$, $\vec{v}_1(\Delta t)$ stehen senkrecht aufeinander. Damit kann man den Mischungswinkel

$$\theta_{ab}(\Delta t) = \prec \left[\vec{v}_0(\Delta t), \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right] \tag{7.19}$$

definieren. Für große Δt muß eine Plateau-Region beobachtbar sein, in der dieser Winkel einen konstanten, von ω unabhängigen Wert annimmt. Alle nötigen Meßwerte zur Bestimmung des Mischungswinkels im 0^+ Kanal liegen mit den Zeitscheibenkorrelationsfunktionen zum Veneziano-Yankielowicz-Multiplett und zum 0^+ Glueball schon vor.

Sowohl für $K = 0.19$ und $K = 0.1925$ kann kein signifikantes Signal für einen Mischungswinkel ungleich Null, unabhängig davon, wie man den Parameter ω wählt, auf einem 8^3x16 Gitter festgestellt werden, d. h. es gibt bei diesen Hopping-Parametern keine Mischung zwischen dem 0^+ Glueball und dem a-f_0 Zustand. Natürlich liegt man zum einen mit diesen Hopping-Parametern noch nicht im Bereich, in dem man die Restaurierung der Supersymmetrie erwartet. Zu anderen ist Massenmischung kein notwendiges Kriterium für die effektive FGS-Wirkung, d. h., der Befund spricht nicht *gegen* diesen Typ von effektiver Wirkung.

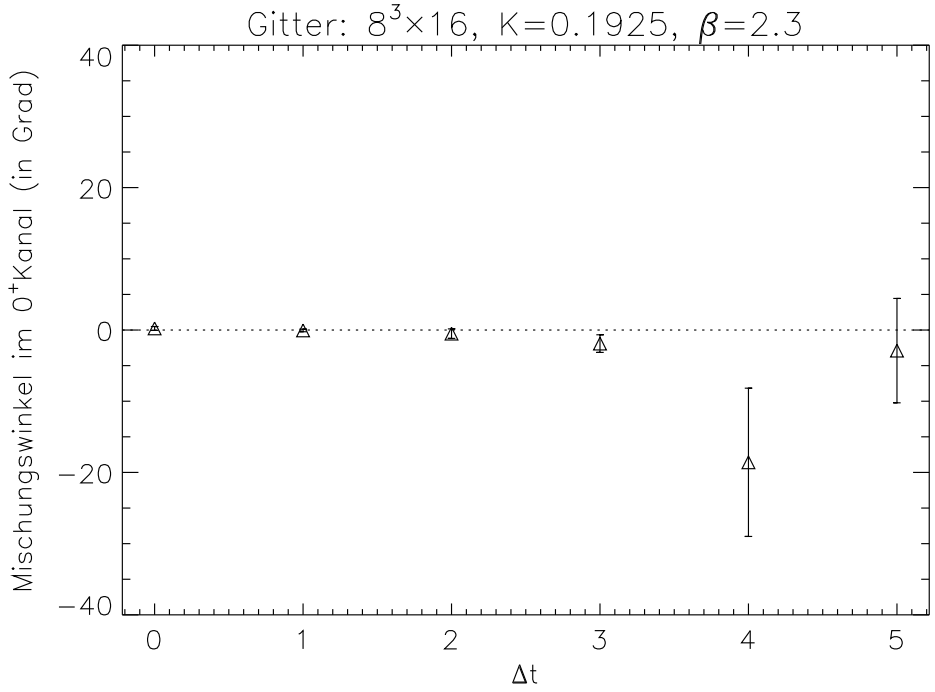


Abbildung 7.19: Mischungswinkel zwischen dem 0^+ Glueball und dem a-f_0 Zustand in Abhängigkeit vom Zeitscheibenabstand Δt für die Kreuzkorrelationsmatrix $\Lambda_{ab}(\Delta t)$. Der Parameter ω wurde so gewählt, daß die statistischen Fehler minimal sind. Zur Fehleranalyse diente wieder das Jackknife-Verfahren.

Kapitel 8

Zusammenfassung

Ziel dieser Arbeit war die numerische Simulation einer SU(2) Eichtheorie mit dynamischen Gluinos. Dabei galt es sowohl physikalische als auch algorithmische Fragestellungen zu untersuchen.

Basis für die numerischen Untersuchungen bildete eine effiziente Implementierung des Multi-Bosonischen Algorithmus in der von I. Montvay vorgeschlagenen Zwei-Schritt-Variante. Die Anzahl der numerischen Operationen pro Update wurde durch die Einführung von skalaren Hilfsfeldern um einen Faktor fünf reduziert. Allerdings führen diese Felder, bedingt durch ihre komplexere Wechselwirkung, zu einem Mehraufwand bei der Kommunikation zwischen den Prozessoren auf einem Parallelrechner. Da der gesamte Algorithmus eine große Anzahl von Parametern besitzt, ist die Optimierung für gegebene Kopplungskonstanten eine nichttriviale Aufgabe. Insbesondere kann man nicht wie z. B. beim Hybrid Monte-Carlo-Algorithmus auf jahrelange Erfahrungen zurückgreifen, sondern betritt vielfach "Neuland". Es kristallisierten sich aber mit der Zeit hilfreiche Regeln für die Optimierung heraus. Es war in jedem Fall sinnvoll, daß erste Polynom so groß zu wählen, daß die Akzeptanzrate des Noisy Correction Steps nicht unter 80% fällt. Dadurch wurde gewährleistet, daß der Noisy Correction Step nicht signifikant in die Dynamik des Updaters eingriff. Die integrierte Autokorrelationszeit wurde hauptsächlich vom ersten Polynom bestimmt, und zwar derart, daß sie ungefähr linear mit dem Grad des ersten Polynoms wuchs. Insbesondere war die integrierte Autokorrelationszeit im wesentlichen unabhängig von der Anzahl der Noisy Correction Steps. Da diese einen entscheidenden Anteil an der Ausführungszeit hatten, wurde pro Sweep jeweils nur ein Noisy Correction Step durchgeführt. Die Anzahl der Hits pro Link beim Multi-Hit-Metropolis Update sollte so eingestellt werden, daß im Mittel pro Link und Sweep fünf Testschritte angenommen werden. Ein optimales Mischungsverhältnis zwischen den Skalarfeld- und Eichfeld-Algorithmen bestand typischerweise aus einem Heatbath-Sweep, drei bis sechs Overrelaxation-Sweeps und einem Multi-Hit-Metropolis Sweep.

Even-Odd-Präkonditionierung wurde ausgenutzt, um die Konditionszahl des Quadrats der Fermion-Matrix im Updater zu reduzieren. Damit konnte automatisch das Approximationspolynom bei gleichbleibender Approximationsgüte verkleinert

werden. In den betrachteten Fällen konnte hierdurch circa ein Faktor fünf an eingesparter CPU-Zeit gegenüber der ursprünglichen Variante erzielt werden. Die genaue Kenntnis der Verteilung des kleinsten Eigenwertes ist nötig, um die untere Approximationsgrenze des Polynoms richtig festzulegen. Auf keinen Fall sollten mehr als 1% der kleinsten Eigenwerte unter diese Approximationsgrenze fallen. Theoretische Voraussagen von Zufalls-Matrix-Modellen erwiesen sich als geeignet, die Verteilung der kleinsten Eigenwerte zu beschreiben. Zur Bestimmung des kleinsten Eigenwertes hat sich die Minimierung des Ritz-Funktionals durch die Polak-Ribiere-Variante des konjugierten Gradientenverfahrens bewährt. Das früher verwendete Gradientenverfahren zeigte für Konditionszahlen über 10^5 numerische Instabilitäten. Die Wahl der oberen Schranke war unproblematisch, da der größte Eigenwert um weniger als 1% schwankte. Es war ausreichend, die obere Approximationsgrenze des Polynoms 5% über den Mittelwert des größten Eigenwertes zu legen.

Die Berechnung der Polynome für den ursprünglichen Noisy Correction Step wurde mit zunehmender Annäherung an den kritischen Hopping-Parameter zu langwierig. Aus diesem Grund wurde der auf den Wurzelpolynomen basierende Korrekturschritt implementiert. Um numerische Instabilitäten zu unterdrücken, sollte bei diesem Verfahren der Grad des dritten Polynoms um ungefähr 30% größer sein als der des zweiten. Nutzt man die Rekursionsrelation zur Bestimmung der Polynome aus, so liefert dieser Korrekturschritt selbst bei einfacher Genauigkeit noch zuverlässige Ergebnisse bis zu Konditionszahlen von $3 \cdot 10^4$ bzw. für Polynome vom Grad 200. Bei sehr kleinen Gluino-Massen zeigte sich, daß es sinnvoll ist, der Zwei-Schritt Polynomapproximation eine Meßkorrektur zur Seite zu stellen. Um insbesondere auch Ausnahmekonfigurationen richtig bewerten zu können, wurde ein hybrides Verfahren entwickelt, bei dem alle Eigenwerte unter einer definierten Grenze mit Hilfe des Kalkreuter-Simma-Algorithmus exakt berechnet und korrigiert werden. Der Korrekturfaktor für den verbleibenden Rest des Spektrums wird mit einem Noisy Estimator und einem viertem, sehr genauen Polynom bestimmt. Durch die Meßkorrektur arbeitet der Gesamtalgorithmus problemlos bis zu mittleren Konditionszahlen von 10^5 bis 10^6 und stellt damit eine interessante Variante für volle QCD-Simulationen bei kleinen Quarkmassen dar.

Bei der Suche nach dem kritischen Hopping-Parameter bei $\beta = 2.3$ traten für $K = 0.195$ Anzeichen des erwarteten Phasenübergangs erster Ordnung für die Supersymmetriebrechung mittels einer nichtverschwindenden Gluino-Masse zutage. In der Verteilung des Ordnungsparameters, in diesem Fall des Fermion-Kondensats, zeigte sich eine Aufspaltung, wie sie für die Existenz zweier entarteter Vakua typisch ist. Im Rahmen einer Teststudie für $K = 0.1975$ konnte diese Aufspaltung nicht mehr nachgewiesen werden. Sollte sich dies bestätigen, so wird man den kritischen Hopping-Parameter im Bereich $K^{cr}_{susy} = 0.196(1)$ erwarten. Das ist konsistent mit den Abschätzungen, die man aus dem Verschwinden der a-π Masse bzw. dem Vergleich von verbundenem zu unverbundenem Anteil des a-η' Zustandes gewinnt. Darüber, ob das statische Potential an diesem Punkt verschwindet, wie es in theoretischen Arbeiten in zwei Dimensionen beobachtet wird, kann trotz Smearing noch

keine Aussage getroffen werden, da die finite-size-Effekte auf den bisherigen Gittern zu groß sind.

Neben dem Fermion-Kondensat sind die Massen der farblosen Bindungszustände, aus denen die leichtesten Supermultipletts aufgebaut sind, die physikalisch interessantesten Größen. Dazu gehören die Konstituenten des Veneziano-Yankielowicz-Multipletts, der Spin $\frac{1}{2}$ Gluino-Glueball und die beiden Bosonen a-η' und a-f_0. Zusätzlich wurde noch die Masse des 0^+ Glueballs und der beiden unphysikalischen Zustände a-π und a-σ gemessen. Zur Bestimmung der Massen der einzelnen Zustände war es wichtig, jeweils ein geeignetes Smearing-Verfahren anzuwenden. Für den 0^+ Glueball wurden sowohl APE-Smearing als auch Teper-Blocking ausgenutzt. Es kann keine klare Aussage getroffen werden, welches Verfahren geeigneter ist. Dies mußte von Fall zu Fall entschieden werden. Das Signal/Rausch-Verhältnis des Gluino-Glueballs wurde durch eine Kombination APE/Jacobi-Smearing optimiert – allerdings mit dem Nachteil behaftet, daß man nun vier Parameter justieren mußte. Teper-Blocking erwies sich als nicht adäquat, da der Smearing-Radius durch den diskreten Charakter des Verfahrens nicht fein genug eingestellt werden konnte. Die Bestimmung der a-π Masse profitierte ebenfalls vom Jacobi-Smearing.

Nach den entmutigenden Resultaten für diese Massen auf dem $4^3 \times 8$ Gitter bei $(\beta = 2.3;\ K = 0.16, 0.17, 0.18)$ kommen sich die Massen auf den größeren Gittern mit wachsenden Hopping-Parametern deutlich näher. Für $K = 0.19$ stimmen die Massen des Gluino-Glueball, a-η' und des 0^+ Glueballs innerhalb der Fehlertoleranzen bereits überein. Allerdings kann eine weitere Annäherung der Massen nicht mehr beobachtet werden, da für $K = 0.1925$ schon deutliche finite-size-Effekte auf dem $8^3 \times 16$ Gitter für den a-η' Zustand und den 0^+ Glueball zu spüren sind. Es ist aber bemerkenswert, wie die vier Massen sich von $K = 0.18$, wo noch über 100% Unterschiede in den Massen gemessen werden, bis zu $K = 0.1925$ angleichen. Desweiteren deuten die Meßwerte an, daß der 0^+ Glueball mindestens genauso leicht wie die drei Zustände des Veneziano-Yankielowicz-Multipletts bleibt. Der Glueball liefert einen wichtigen Betrag zur effektiven Wirkung und sollte nicht ausintegriert werden. Es kann, abgesehen vom leichten 0^+ Glueball, kein weiteres Indiz dafür gefunden werden, daß die nötige Erweiterung der effektiven Veneziano-Yankielowicz-Wirkung die von Farrar, Gabadadze und Schwetz (FGS) geforderte Form hat. Die effektive FGS-Wirkung erlaubt ausdrücklich Massenmischung im 0^+ bzw. 0^- Kanal. Diese wird aber bis $K = 0.1925$ für den 0^+ Kanal nicht beobachtet.

Zur Zeit sammelt ein Produktionslauf Statistik bei $(K = 0.1925, \beta = 2.3)$ auf einem $12^3 \times 24$ Gitter. Für diesen Lauf ist sehr viel Rechenzeit reserviert, so daß die resultierenden Massen nicht nur weniger finite-size-Effekte zeigen werden, sondern auch einen geringeren statistischen Fehler besitzen sollten. Die Beobachtung einer sich anbahnenden Massenentartung sollte sich weiter verbessern. Da sich das Jacobi-Smearing schon für das a-π bewährt hat, liegt es nahe, zusätzlich diesen Smearing-Algorithmus auf den a-f_0 Zustand anzuwenden, um zu verläßlicheren Werten für diese Masse zu gelangen. Eine Meßroutine zur Bestimmung der 0^- Glueball-Masse

ist ebenfalls in Arbeit, die es erlauben wird, die FGS-Wirkung besser zu untersuchen.

Anhang A

Notation und Konventionen

A.1 Pauli-Matrizen

Zusammen mit der Einheitsmatrix $1_{(2\times 2)}$ bilden die Pauli-Matrizen eine Basis für die hermiteschen 2×2 Matrizen

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{A.1}$$

Diese drei Matrizen σ_1, σ_2 und σ_3 erfüllen die Relationen

$$\begin{aligned} \sigma_i\sigma_j + \sigma_j\sigma_i &= 2\delta_{ij} \\ \sigma_i\sigma_j - \sigma_j\sigma_i &= 2i\epsilon_{ijk}\sigma_k \\ \sigma_1\sigma_2 &= i\sigma_3 \quad \text{und zyklische Vertauschungen.} \end{aligned} \tag{A.2}$$

Jedes Element der speziellen unitären Gruppe $\Lambda \in SU(2)$ kann nun mit Hilfe der Generatoren

$$T_a = \frac{\sigma_a}{2}, \quad a = 1, 2, 3 \tag{A.3}$$

in der Form

$$\Lambda = e^{iT_a\omega^a}, \quad \omega^a \in \mathbb{R} \tag{A.4}$$

dargestellt werden, wobei die Parametrisierung durch ω^a innerhalb einer hinreichend kleinen Umgebung von $\mathbb{1}$ eindeutig ist. Weiterhin defininert man

$$\begin{aligned} \sigma^\mu = (\mathbb{1}, \vec{\sigma}) \qquad & \bar{\sigma}^\mu = (\mathbb{1}, -\vec{\sigma}) = \sigma_\mu \\ \sigma^{\mu\nu} = \frac{i}{4}\left(\sigma^\mu\bar{\sigma}^\nu - \sigma^\nu\bar{\sigma}^\mu\right) \qquad & \bar{\sigma}^{\mu\nu} = \frac{i}{4}\left(\bar{\sigma}^\mu\sigma^\nu - \bar{\sigma}^\nu\sigma^\mu\right). \end{aligned} \tag{A.5}$$

A.2 γ-Matrizen im Euklidischen

Zwischen den γ-Matrizen im Euklidischen und denen in der Minkowski-Metrik besteht der Zusammenhang

$$\begin{aligned} \gamma_{1,2,3}^{Euklidisch} &= -i\gamma_{1,2,3}^{Minkowski} \\ \gamma_4^{Euklidisch} &= -i\gamma_4^{Minkowski} = \gamma_0^{Minkowski}. \end{aligned} \tag{A.6}$$

Die γ-Matrizen im Euklidischen werden im weiteren einfach mit γ_μ, $(\mu = 1,2,3,4)$ bezeichnet. Die euklidischen γ-Matrizen nehmen in der Weyl- (chiralen-) Darstellung die folgende hermitesche 2×2 Blockmatrix-Form an

$$\gamma_{1,2,3} = \begin{pmatrix} 0 & -i\sigma_{1,2,3} \\ i\sigma_{1,2,3} & 0 \end{pmatrix}, \quad \gamma_4 = \begin{pmatrix} 0 & \mathbb{1} \\ \mathbb{1} & 0 \end{pmatrix}. \tag{A.7}$$

Die γ-Matrizen genügen den Antikommutationsrelationen

$$\{\gamma_\mu, \gamma_\nu\} = 2\delta_{\mu\nu}\mathbb{1} \tag{A.8}$$

und bilden somit eine Clifford-Algebra. Desweiteren definiert man

$$\gamma_5 = \gamma_1\gamma_2\gamma_3\gamma_4 = \gamma_5^\dagger = \begin{pmatrix} \mathbb{1} & 0 \\ 0 & -\mathbb{1} \end{pmatrix} \tag{A.9}$$

und gelangt zu den Projektoren auf die links- bzw. rechtshändigen Komponenten eines Fermion-Feldes

$$P_L = \frac{1}{2}(1 - \gamma_5), \quad P_R = \frac{1}{2}(1 + \gamma_5). \tag{A.10}$$

Die Matrix γ_5 erfüllt die nützlichen Eigenschaften

$$\begin{aligned} \{\gamma_5, \gamma_\mu\} &= 0 \\ \gamma_5\gamma_5 &= \mathbb{1} \\ \gamma_5^\dagger &= \gamma_5. \end{aligned} \tag{A.11}$$

Bezüglich der Ladungskonjugation gelten die folgenden Gleichungen

$$\begin{aligned} C\gamma_\mu C^{-1} &= -\gamma_\mu^T \\ C^{-1}\gamma_\mu^T C &= -\gamma_\mu \\ C\gamma_5 C^{-1} &= \gamma_5^T. \end{aligned} \tag{A.12}$$

Die Generatoren $\Sigma_{\mu\nu}$ der vierdimensionalen Spinordarstellung sind durch die folgende Relation gegeben

$$\Sigma_{\mu\nu}^{Minkowski} = \frac{i}{4}\left[\gamma_\mu, \gamma_\nu\right]. \tag{A.13}$$

A.3 Gitternotation

Ein Gitter Λ ist durch die Menge der Punkte

$$\Lambda = \{x = (\mathbf{x}, t) = (x_1, x_2, x_3, x_4) \in \mathbb{Z}^4; 0 \leq x_\mu < L_\mu, \mu = 1, \cdots, 4\} \tag{A.14}$$

definiert. Die Summation über dieses Gitter wird kompakt durch

$$\sum_x = \prod_{\mu=1}^{4} \sum_{x_\mu=0}^{L_\mu - 1} \tag{A.15}$$

symbolisiert. Im allgemeinen ist die Ausdehnung des Gitters in allen räumlichen Richtungen gleich und ist mit $L_s = L_1 = L_2 = L_3$ gegeben, während $T = L_4$ der Länge des Gitters in Zeitrichtung entspricht. Das räumliche Volumen des Gitters ist $V = L_s^3$, und die Gesamtzahl aller Gitterpunkte berechnet sich aus $\Omega = VT$. Die vier orthogonalen Richtungen sind $\mu = 1, \cdots, 4$. Der Einheitsvektor in Richtung μ wird durch $\hat{\mu}$ benannt. Die Summe über die vier positiven Richtungen ist mit

$$\sum_{\mu=1}^{4} \tag{A.16}$$

gegeben. Soll die Summation über negative und positive Richtungen erfolgen, wird nur kurz

$$\sum_{\mu} = \sum_{\mu=\pm 1}^{\pm 4} = \sum_{\mu=\pm 1,\pm 2,\pm 3,\pm 4} \tag{A.17}$$

geschrieben. Die Gitterableitungen einer Funktion ϕ_x sind durch Vorwärts- bzw. Rückwärtsdifferenzieren

$$\begin{aligned} \triangle_\mu^f \phi_x &= \phi_{x+\hat{\mu}} - \phi_x \\ \triangle_\mu^b \phi_x &= \phi_x - \phi_{x-\hat{\mu}} \end{aligned} \tag{A.18}$$

definiert. Durch die Relation $\triangle_\mu^b = -(\triangle_\mu^f)^\dagger$ ergibt sich eine kompakte Formulierung für den Laplace-Operator

$$\begin{aligned} \triangle\phi_x &= \triangle_\mu^f \triangle_\mu^b \phi_x = \triangle_\mu^b \triangle_\mu^f \phi_x \\ &= \sum_{\mu=1}^{4} (\phi_{x+\hat{\mu}} + \phi_{x-\hat{\mu}} - 2\phi_x). \end{aligned} \tag{A.19}$$

Elementare Paralleltransporter zu einer Eichgruppe G werden auf dem Gitter mit den Verbindungen (oder auch "Links") zwischen den Gitterpunkten assoziiert. Entsprechend wird ein Paralleltransporter $U(x+\hat{\mu}, x) = U_{x\mu} \in G$ für zwei benachbarte Punkte x und $x+\hat{\mu}$ auch "Link-Variable" genannt.

Die kleinste geschlossene Kurve, aufgebaut aus Link-Variablen, wird Plaquette genannt. Die positive Orientierung einer Plaquette ist durch

$$\begin{aligned} U_P &= U_{x;\mu\nu} \\ &= U(x, x+\hat{\nu}) U(x+\hat{\nu}, x+\hat{\nu}+\hat{\mu}) U(x+\hat{\mu}+\hat{\nu}, x+\hat{\mu}) U(x+\hat{\mu}, x) \\ &= U_{x\nu}^{-1} U_{x+\hat{\nu},\mu}^{-1} U_{x+\hat{\mu},\nu} U_{x\mu} \end{aligned} \tag{A.20}$$

definiert. Unter der Summe über alle Plaquetten P vesteht man diejenige Summation über das gesamte Gitter, bei der jede Plaquette nur mit einer, der positiven Orientierung, beiträgt

$$\sum_P = \sum_x \sum_{1 \le \mu \le \nu \le 4} . \tag{A.21}$$

In analoger Weise definiert man die Summation über die räumlichen

$$\sum_{P_s} = \sum_{x} \sum_{1 \le \mu \le \nu \le 3} \tag{A.22}$$

und über die zeitartigen Plaquetten

$$\sum_{P_t} = \sum_{x} \sum_{1 \le \mu \le 3, \nu = 4} . \tag{A.23}$$

A.4 Darstellungen der SU(2) Gruppe

In den fermionischen Wechselwirkungstermen wird die adjungierte Darstellung der Eichgruppe SU(2) gebraucht. Da aber gerade die Darstellung der SU(2) Gruppe insbesondere in Programmen notorische Quelle für Mißverständnisse ist, soll an dieser Stelle auf die Darstellungen und Umrechnungsformeln im Simulationsprogramm eingegangen werden.

Jede SU(2) Matrix U läßt sich mit Hilfe der Pauli-Matrizen $\{\sigma_1, \sigma_2, \sigma_3\}$ über die Relation

$$U = x_0 \mathbb{1} + i \sum_{i=1}^{3} x_i \sigma_i = \begin{pmatrix} x_0 + i x_3 & x_2 + i x_1 \\ -x_2 + i x_1 & x_0 - i x_3 \end{pmatrix} \tag{A.24}$$

darstellen, wobei die Parameter x_i den Bedingungen

$$\det U = \sum_{i=0}^{3} x_i^2 = 1 \quad \wedge \quad x_i \in \mathbb{R} \tag{A.25}$$

genügen müssen. Die fundamentale Darstellung U und die adjungierte Darstellung V sind durch die Beziehung

$$V_{rs} = \frac{1}{2} \mathrm{Tr} \left(U^\dagger \sigma_r U \sigma_s \right) \tag{A.26}$$

verbunden. Links in der fundamentalen Darstellung werden im Simulationsprogramm in Objekten vom Typ *SU2_Link* abgelegt, Links in der adjungierten Darstellung werden mit der Klasse *SU2Adj_Link* assoziiert. Für beiden Klassen entspricht der Indexzugriff über *operator[](const int i)* dem eines gewöhnlichen reellen C-Feldes der Länge vier bzw. neun. Das Speicherlayout wird dabei auf

x_0	x_3	x_2	x_1

für einen Link in der fundamentalen Darstellung festgelegt.

Anhang B

Produktionsläufe

Die folgenden Tabellen enthalten eine Übersicht zu den langen Produktionsläufen. Dabei werde die Bezeichnungen

- $[\epsilon, \lambda]$: Approximationsintervall
- $n_1, ..., n_4$: Ordnung der Approximationspolynome
- Update: Charakterisierung der Updatesequenz durch

$$N_{Heatbath} \div N_{Overrelaxation} \div N_{Multi-Hit-\text{Metropolis}}.$$

- NKS: Anzahl der Noisy-Korrekturschritte

verwendet.

B.1 $4^3 \times 8$ Gitter

K	$[\epsilon, \lambda]$	n_1	n_2	n_3	n_4	Nr. Updates	Update	NKS
0.16	[0.008,3.2]	8	32	32	-	96200	$1 \div 6 \div 10$	5
0.17	[0.008,3.2]	8	32	32	-	241500	$1 \div 6 \div 10$	5
0.18	[0.002,3.2]	8	32	32	-	342800	$1 \div 6 \div 10$	5
0.1975	[0.00003,3.7]	22	66	102	-	134400	$1 \div 3 \div 10$	1

Tabelle B.1: Produktionsläufe auf $4^3 \times 8$ Gittern bei $\beta = 2.3$. Für $K = 0.1975$ wurde die präkonditionierte Version des Updaters bei antisymmetrischen Randbedingungen eingesetzt, alle anderen Läufe nutzten noch den Originalalgorithmus bei symmetrischen Randbedingungen.

B.2 $6^3 \times 12$ Gitter

K	$[\epsilon, \lambda]$	n_1	n_2	n_3	n_4	Nr. Updates	Update	NKS
0.16	[0.008,3.2]	8	32	32	-	374400	$1 \div 6 \div 10$	6
0.17	[0.008,3.2]	8	32	32	-	332800	$1 \div 6 \div 10$	6
0.18	[0.008,3.2]	8	32	32	-	540800	$1 \div 6 \div 10$	6
0.185	[0.002,3.4]	12	32	48	-	384000	$1 \div 10 \div 10$	1
0.19	[0.0002,3.5]	16	60	96	-	712800	$1 \div 6 \div 10$	2
0.1925	[0.00003,3.7]	22	66	102	400	1280000	$1 \div 3 \div 20$	1
0.195	[0.00003,3.7]	22	66	102	400	1224000	$1 \div 3 \div 20$	1

Tabelle B.2: Produktionsläufe zu der Gittergöße $6^3 \times 12$ bei $\beta = 2.3$.

B.3 $8^3 \times 16$ Gitter

K	$[\epsilon, \lambda]$	n_1	n_2	n_3	n_4	Nr. Updates	Update	NKS
0.19	[0.00065,3.55]	20	82	112	-	1038400	$1 \div 4 \div 20$	1
0.1925	[0.0001,3.6]	22	142	190	-	870400	$1 \div 4 \div 20$	1

Tabelle B.3: Produktionsläufe zu der Gittergöße $8^3 \times 16$ bei $\beta = 2.3$.

Appendix C

The Paradigm of Object Orientation

C.1 Introduction

The contents of this appendix is based on a 'deja vu event' probably every parallel software developer has had. Our collaboration developed a numerical code for the simulation of the SU(2) gauge theory with dynamical fermions for a vector-computer. The collaboration first used a Cray T90 vector computer and standard workstations for this simulation, but it occurred that it was not fast enough to solve our problems. Therefore we decided to use the Cray T3E, a parallel, distributed memory computer. A short view at the old code showed that it was simply not suited for a parallel computer. It had to be reprogrammed completely and in order to fulfill future needs its functionality had to be extended. This was my job.

Because we had to start from scratch anyway, we decided to take care about the three big R's of software-engineering: reusability, reliability and readability. Experts in software developing today say that object-oriented design may help to fulfill these needs [Boo97]. It must be said that these experts usually do not write dynamic fermion simulations, but word processors or spreadsheets . At the beginning I did not know whether this was applicable to numerical problems [SWWW97][PT97].

In this appendix the evolution from a procedural way of solving a problem to an object oriented approach is shown. First the new ideas of object oriented programming are presented [1]. The second part deals with the advantages of this programming technique in relation to specific problems in lattice gauge theories simulation codes. The last two parts give an overview on the hardware independent implementation of a lattice gauge theory using object oriented technologies. Emphasis is laid on objects for the purpose of parallelizing. Due to this one will see that the essential code for local algorithms can be programmed architectural independent. This means that the code for a parallel computer is identical to that for a single processor machine.

At first let's have a look on a common SU(N) lattice gauge theory. Values which can

[1] Of course good books are available on this topic e.g. [Boo97][GHJV94] and [BN94].

be found in nearly every simulation code are e.g. the simple plaquette and Wilson action

$$\begin{aligned} U_P &\equiv U^{\dagger}_{x,x+\nu} U^{\dagger}_{x+\nu,x+\mu+\nu}, U_{x+\mu+\nu,x+\mu} U_{x+\mu,x} \\ S[U] &= \sum_P \beta \left\{ 1 - \frac{1}{N} \mathrm{Re}(\mathrm{Tr} U_P) \right\}. \end{aligned} \tag{C.1}$$

As seen these values are obviously valid for any choice of N. And of course they do not depend on the representation in computer memory. Everybody tries to keep up the general formulas as long as possible. The only attributes typically used are the behaviours of a SU(N) group element which are the existence of multiplication that all elements possess an inverse and the capability to compute the trace. SU(N) matrices are intuitively treated as *things*, on which several functions are defined. No one cares about a particular representation.

This holds true as long as one is not sitting in front of a computer keyboard, because ordinary computer programs do care on it. Furthermore they are valid only for one choice of N. As a fatal result of this approach the whole program depends on a single design issue, which is how to store SU(N) matrices. Often this makes it impossible to share code with other programmers which might have chosen another possible representation. If someone wants to use foreign code, he probably has to change a lot of the original code. As pointed out before it is not useful to program in terms of a special implementation of a SU(N) matrix. The reusability of a code is improved, if it's kept independent from a certain implementation. Because of this it might be sensible to use *things*, which have a hidden internal implementation and a standardised communication interface to the outside. Because the internal implementation is invisible to the outside it can be changed easily – the rest of the program does not depend on this single design issue.

The word *thing* might sound a little naive, from now on the term *object* will be used. These objects combine data that represent state with functions which access or modify the data [Boo97]. Usually there are many possibilities to store the internal data which correspond to a single status. For example several different possibilities to represent a proper SU(2) matrix in the computer memory exist. It does not matter how a proper state is realized in the object. Obviously there must be a place where the implementation details and the behaviour of an object are defined. This duty is done by the class. For example one could model the attributes of the SU(2) group with a class:

- **state:** a certain SU(2) matrix
- **interface:** multiplicate(), invert(), trace()

} **an Object**

- **class:** SU(2) Group
  - defines representation in computer memory
  - defines multiplication(), invert() and trace(), representation dependent

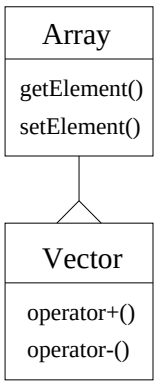
An object of the SU(2) class type has a state which is the value of the given SU(2) matrix. The class offers a well-defined interface to the outside. The class itself defines the representation of a SU(2) matrix in the computer memory. It also defines the behaviour of an instance that is the multiplication, inversion and the tracing. Of course these parts of the program, and only these, depend on the internal representation. If at this single point high speed assembler code is inserted the whole program will benefit from this without further modifications.

Because smaller units are easier to manage, it is desirable that a program can be broken up into small independent units. None of these units should depend on details of another unit [Boo97]. Internal data should not be accessible from the outside, because one wants to avoid foreign functions to depend on representation details of an object. A good design is characterized by the fact that it is unnecessary to peek inside the object. In object oriented languages it is also common to provide member data and member functions of the object with access permission, which is called data encapsulation. This can be compared with the read and write permission in a UNIX-file system. For example C++ allows the following specifications [BN94]:

- **private:** that means only accessible for the member functions of the class.
- **public:** accessible to anyone
- **protected:** visible only to child classes.

In large scale software projects it became usual to organize classes in hierarchies. These class hierarchies are composed of only a few different kinds of subsystems in various arrangements. Instead of rewriting common functionality over and over again, it is possible to inherit a class. Here inheriting means that the child class of a given parent class inherits all of its functionality.

In the following example array is a common class for numerical data, it only provides two numerical functions, which are getElement and setElement. Say one wants to write a vector class which has the functionality of an array and some more operations like plus or multiply. For this it is beneficial to reuse code which has been originally written for the array class. As a result of inheritance only the new functions have to be implemented. The vector class inherits the basic functionality of a so called container from the class array. This means the vector extends the functionality of an array. As a result a vector is always usable as an array.



Professional software developers happily adopted the so called Unified Modelling Language (UML) in order to present complicated class hierarchies in an easy, understandable way [FS97]. Later on more basic elements of UML will be introduced which makes it easier to understand the shown class hierarchies.

There is another type of inheritance, the so called polymorphism. In this case only an abstract interface is provided by the parent class. The parent class itself does not implement these functions, this is the job of the child classes. Polymorphism is the promise of the child classes to implement the abstract interface of the parent class. Necessary code will not be reused, it must be included in the child classes.

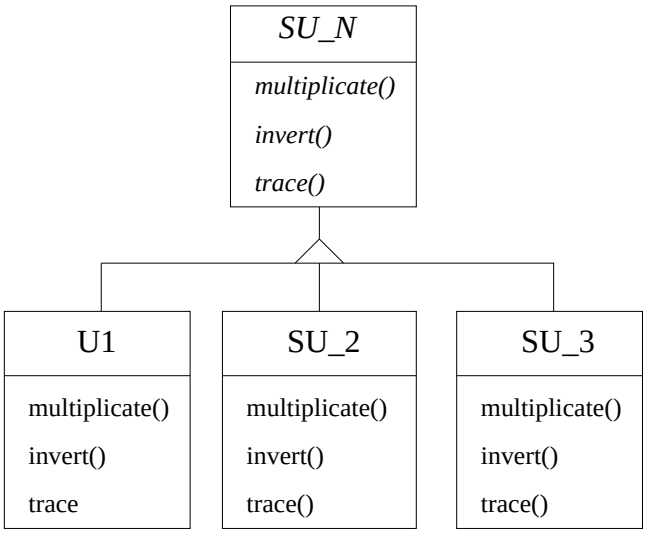


Figure C.1: Possible class hierarchy of the link objects.

The parent class is an abstract class for SU(N) matrices. The italic font signals that this is only an abstract class which provides an abstract interface. None of these functions are implemented, they must be provided by the child classes. Object oriented languages do allow the writing of functions in terms of the abstract class. This function determines at runtime which proper object and corresponding member functions really are to be used and it can be used by all derived classes. This is one of the main advantages of Object Oriented Programming.

Polymorphism introduces a whole new level of flexibility and makes it possible to e.g. write a function plaquette in terms of SU(N), to compile and then to store it in a library. Later one could write an arbitrary gauge group class, which would inherit the abstract SU(N) interface. The gauge group functions are linked dynamically at runtime when needed. Therefore the plaquette function would then be usable for this new class. On the other hand runtime linking introduces a run-time penalty but it will show later on that there are techniques to avoid this.

C.2 The Three Big R's

Even though the three big *R's* originally derive from commercial needs of software developing they also make sense from a numerical programmers point of view.

Readability: In real life the majority of numerical programmers seldomly attach

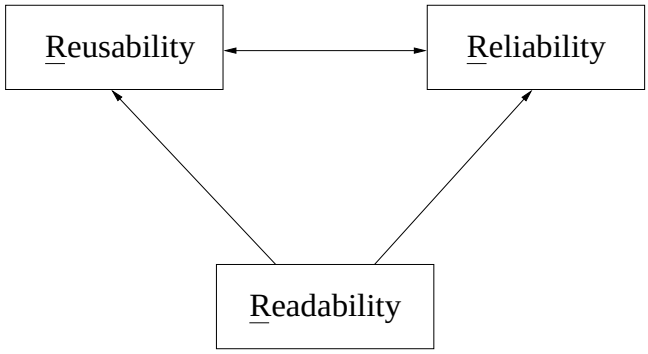


Figure C.2: Relationship between the R's.

great importance to remarks. Due to this the readability of programs often is vital for the reusability and reliability of the code. Sadly a lot of numerical programmers also seldomly attach great importance to readability. This might come from the "fire-and-forget-style" most programs have.

C++ comes with additional features which enhance readability especially for numerical codes. This includes operator overloading. It allows to redefine the basic numerical operations for own classes. E.g. one could let the operator '*' perform matrix multiplication between SU(2) matrices.

C++:

```
return   link[     site       ][dvr].invert()
       * link[ nup[site][dvr] ][dir].invert()
       * link[ nup[site][dir] ][dvr]
       * link[      site      ][dir];
```

Fortran:

```
!
!    U^+ V
!
           plq1(ist,0)=                              &
              Conjg(link(0,ist,nu))* link(0,ist,mu) &
            + Conjg(link(1,ist,nu))* link(1,ist,mu)
           plq1(ist,1)=                                 &
                 -link(1,ist,nu) *      link(0,ist,mu)  &
            +     link(0,ist,nu) *      link(1,ist,mu)
!
!    U V
!
           plq2(ist,0)=                                      &
                    plq1(ist,0)  * link(0,nup(ist,mu),nu) &
              - Conjg(plq1(ist,1)) * link(1,nup(ist,mu),nu)
           plq2(ist,1)=                                      &
                    plq1(ist,1)  * link(0,nup(ist,mu),nu) &
              + Conjg(plq1(ist,0)) * link(1,nup(ist,mu),nu)
!
!    U V^+
!
           plq1(ist,0)=                                        &
                    plq2(ist,0)  * Conjg(link(0,nup(ist,nu),mu)) &
              + Conjg(plq2(ist,1)) *       link(1,nup(ist,nu),mu)
           plq1(ist,1)=                                        &
                    plq2(ist,1)  * Conjg(link(0,nup(ist,nu),mu)) &
              - Conjg(plq2(ist,0)) *       link(1,nup(ist,nu),mu)
```

Figure C.3: Comparison between the new C++ and the old Fortran implementation of a plaquette computation.

A common expression for dynamical fermion simulation is

$$\chi_x = (\gamma_5 - \gamma_5\gamma_\mu)\psi_x. \tag{C.2}$$

In order to reach a natural programming style and enhance readability it is possible in C++ to define objects like spinor and γ-matrices and to overload the related operators.

$$\chi_x = (\gamma_5 - \gamma_5\gamma_\mu)\psi_x \;\rightarrow\; \texttt{chi[x]=g5*psi[x] - g5*g<mu>*psi[x]}$$

The corresponding Fortran code which was used in the old program looks rather cryptic

$\chi_x = (\gamma_5 - \gamma_5\gamma_\mu)\psi_x \rightarrow$

```
Do 10 idir=0,3
  iu=gamin(idir,mu+1)
  g5=Real(gaval(idir,5))
  gu5=gaval(idir,mu+5+1)
  chi(x,idir)=psi(x,idir)*g5 - psi(x,iu)*gu5
10  Continue
```

Due to performance enhancement the compiler should perform γ-matrix multiplications at compile-time. In C++ this can be achieved with the help of the template meta programming [Vel95b]. This technique drastically improves runtime performance, but it is only intended for C++ experts.

Another useful feature of C++ is function overloading. For example two algorithms have to be provided for a square root algorithm: one for double, one for complex numbers. In C++ both functions can be named *sqrt()*. Depending on the function argument the compiler will call the proper code. Fortran on the other hand uses *Dsqrt* and *Csqrt* to distinguish between the functions.

Reusability: To reach reusability, different techniques can be used. One of these is abstraction or polymorphism. It is used to express similarities among objects, to keep clients unaware of the specific type and of the classes that implement these objects. It supplies a so called "is usable as" commonality [Boo97]. Software experts highly recommend to program to an interface and not to a specific implementation [GHJV94], e.g. see figure C.4.

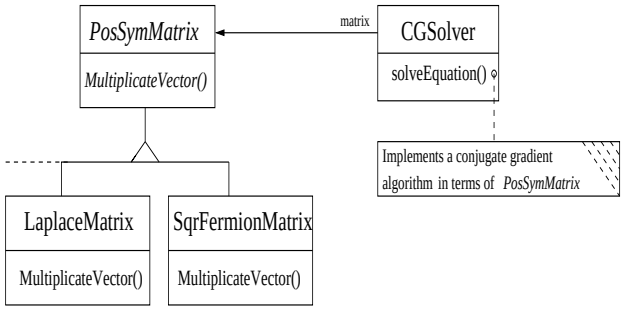


Figure C.4: Implementation of a matrix inversion via polymorphism.

Another technique is inheritance. It favours extensibility, new objects and their behaviours can be defined as incremental modifications and extensions of existing objects. You have to know the class for inheritance, so this is called white-box reuse [GHJV94]. It is used for "is a" relations [Boo97]. The disadvantage of this technique is that it often leads to fat interfaces and fat objects [Cop92].

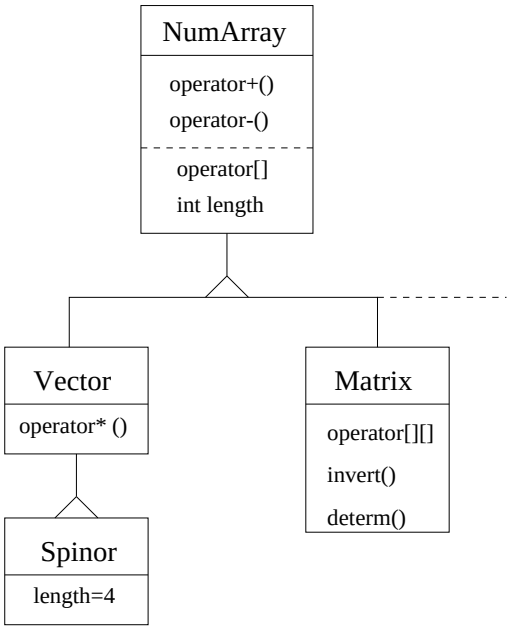


Figure C.5: UML Diagram for different array classes using inheritance to reuse code and to share common behaviour.

With composition new functionality is obtained by assembling objects. It is a black box reuse, because no internal details are visible. This might lead to runtime inefficiencies, so it requires carefully designed interfaces. Nevertheless to avoid the fat interface problem object composition should be prefered to class inheritance.

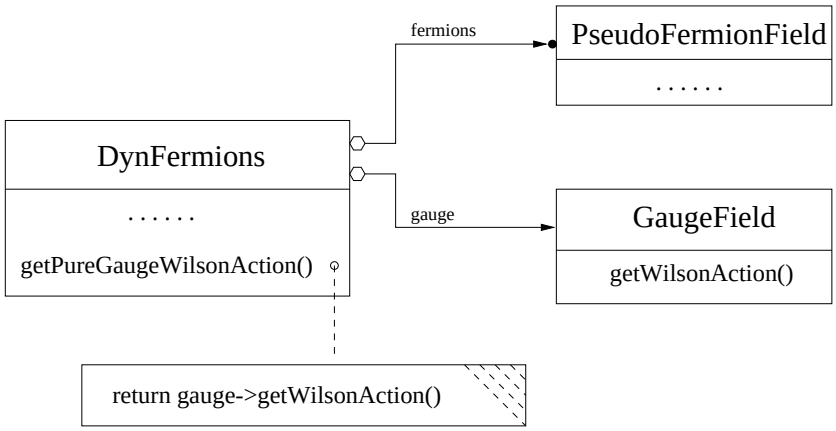


Figure C.6: A example for functionality enhancement via object composition.

C++ makes it possible to write algorithms with generics or templates [Str97]. To improve runtime performance it is beneficial to avoid an abstract SU(N) base class in order to obtain a static linking in all functions. One therefore needs a code template for the function plaquette for which the proper gauge type is supplied at the point of use. This can be treated as a slot machine in which the special gauge group is inserted and afterwards with the help of the code template code is produced. This technique is not strictly object oriented and it is not needed in languages that do not have compile-time type checking.

```
template <class GaugeType>
GaugeType GaugeField<GaugeType>::Plaquette(int site,
                                           int dir, int dvr)
{
  return   link[     site       ][dvr].invert()
         * link[ nup[site][dvr] ][dir].invert()
         * link[ nup[site][dir] ][dvr]
         * link[      site      ][dir];
}
```

In this example plaquette is a member function of the class GaugeField. The class GaugeField itself is only a code template. By supplying a gauge group class the compiler is able to generate the whole code. The class GaugeField also contains the capability to perform a Teper blocking. By simply providing the gauge group class, the compiler is able to generate code for this rather complicated operation. This keeps the gauge group dependent code tiny. For example a highly efficient SU(2) class consists of approx. 500 lines of code. Just throw it in the slot machine and the compiler will do the work for you. In the SU(2) Super-Yang-Millsproject the whole pure gauge theory depend code consists of approx. 8000 lines which are completely independ of the chosen gauge group.

Reliability: Software Design should generate the illusion of simplicity [Boo97]. That means that complex behaviours should be hidden behind a simple facade or an interface. Object oriented programming helps to provide the illusion by using polymorphism and encapsulation.

Object oriented languages provide a new method for error reporting called exception handling. In procedural languages errors are often indicated by flags but they are not necessarily noticed by the program. In contrast to that in object oriented languages an error is indicated by throwing an exception. This exception has to be caught by another part of the program. If it is not caught, the program stops. As a result an error is always detected.

Sometimes this method is called the “goto of the 90s” related to the old basic style “on error goto” command. In addition in numerical programs strong typing helps to avoid side effects in subroutines and allows to divide the problem in small, independent and reliable units.

Appendix D

High Performance C++

D.1 Overview

Today there exist several languages for object oriented programming. Smalltalk is often used in so called rapid prototyping environments. It makes it possible to begin with a simple working system right from the start and build up complex structures from it. This is often a better approach than designing a complex system from scratch. Due to its pure object oriented style and the build in garbage collector it is definitely to slow for numerical applications.

C++'s practical compromise is the missing of a garbage collector. Under normal circumstances a disadvantage, it makes the language suitable for numerical simulation. The programmer is able to build highly specified garbage collectors for small numerical objects like SU(2) matrices.

A new and popular language is Java. It is a mixture of C++ and Smalltalk, because Java has the syntax of C++, which makes it easy to learn for a C++ programmer. Like Smalltalk it has garbage collection and no pointers. The runtime environment includes a Smalltalk-like virtual machine.

There were several reasons, which led to the decision of using C++ in our project. First and most important: C++ is the only object oriented (OO) language available for high performance computers. It is the only high efficient OO-language. Communication and multithreading libraries are usable with C++. Its strong typing enhances the reliability for numerical programs. On the other hand there are some disadvantages which have to be taken into consideration. C++ was never designed - it mutated as one mistake after another became obvious. The portability problems of C++ are comparable to some problems Fortran90 had at the beginning of the 90s. At the time of writing a draft ANSI C++ standard has been released and great strides have been made in this area. Different optimization capabilities may also lead to problems. Compilers like the KAI C++ Compiler have an object oriented optimizer which generates high efficient code. On the other hand the same code delivers rather poor performance with the Sun C++ compiler. Furthermore tem-

plates are a relatively new feature of C++, so not every compiler is supporting it. In the earlier version of the ANSI standard the instantiation of templates was not defined. Every compiler manufacturer had its own standard and the code became manufacturer dependant. Packages/modules are also not supported by the language. Nevertheless it is my opinion that all in all the advantages clearly dominate.

The efficiency of OO-numerical problems stands and falls with a highly efficient vector class. A problem that arises almost always while overloading numerical operators is the generation of temporary objects. This problem is not only limited to C++ but also to Fortran90. I have tried to overload operators in Fortran90 several times, but the performance was always disappointing. The main problem is that the evaluation of numerical expression generates temporary vectors

$$\vec{a} = \underbrace{\underbrace{\vec{x} + \vec{y}}_{\vec{t_1}} + \vec{z}}_{\vec{t_2}=\vec{t_1}+\vec{z}}. \tag{D.1}$$

T1 is generated in a function, which means on the stack. If the function is left this temporary object has to be copied away from the stack using the so called copy constructor. That means the copy constructor is used to generate a temporary copy from a temporary object. This does not make sense. Furthermore the compiler may generate hidden temporary vectors using the copy constructor.

A popular method to avoid unnecessary copying is reference counting [Cop92]. But due to the additional level of indirection reference counting it is efficient only for large vectors and not suited for typical vector length of dynamical fermion simulations. I found temporary base classes [SX96] and expression templates very useful to write a high efficient vector class [Vel95a]. These are technical solutions only available in C++. It would lead to far in this context to explain them in detail (For further informations see also [Rob96][VJ97].). The basic ideas are

- **Temporary Base Class Idiom**
  - introduce an own class TmpVector for temporary vectors.
  - TmpVector construct/destructed shallowly.
  - operator+(Vector &) returns a TmpVector.
  - operator+(TmpVector) is implemented as TmpVector+=Vector.
  - Disadvantage: four times more operators have to be overloaded.
- **Expression Templates**
  - avoid temporary objects in the first place by automatically transform `vector u,v,w; u=v+w;` at compile time (more or less) into

    ```
    for (int i(0); i < u.length(); ++i)
      u[i]=v[i] + w[i];
    ```

    using template meta programming (or compile time programs).

On one hand it is desirable to implement a class for handling gamma matrices. On the other hand it is obvious that the gamma matrix multiplication has to be done at compile time rather than at compile time. Otherwise a fermion matrix multiplication would proceed at a snail's pace. Two techniques exist to achieve this.

- **Lazy Evaluation for** $\gamma_\mu\psi$
  - delay computation until the result is needed.
  - processing expressions like $\chi + \gamma_\mu\psi$ in a single task.
- **Expression** $\gamma_\mu\gamma_\nu$
  - force the compiler to perform this multiplication at compile time using template meta programming.

D.2 Benchmark Results

Instead of writing vector classes from scratch one should have a look on existing high efficiency vector classes. Available libraries on the commercial market are math.h++ and ObjectSuite. Freely available are Blitz++ (Expression Templates), MV++ (Reference Counting) and NumArray.h (Temporary Base Class).

An often asked question is: Is C++ fast enough for numerical computations or can C++ be as fast as Fortran90? To test the different vector classes Y. Xylander and I used a Monte-Carlo simulation of the two dimensional σ-model. The update algorithm is a single Metropolis Monte-Carlo step. This is a worst case test for a vector class because the vector length is three. The administration overhead caused by the introduction of a class can be huge compared to the performed operation. Generally the difference between the class libraries are small for larger vector sizes.

F90	Blitz++	NumArray.h	MV++	math.h++
4.71s	4.93s	5.21s	40.1s	69.1s

Table D.1: Runtimes of various vector classes for the simulation of the 2d O(3) symmetric non-linear σ-model on the T3E-512/600 with the Cray C++ compiler.

As one can see in tabular D.1 Blitz++ and TBC reach comparable speed to Fortran90 on a T3E-512/600. MV++ uses reference counting which is not suitable for small vector sizes. The only commercial library prized 1000$ surprisingly is the slowest one.

There are C++ Compiler available which have two separate optimizers. One acts on register level and does not differ from a Fortran90 optimizer. The second operates on the object level. With this optimizer C++ is sometimes faster than comparable

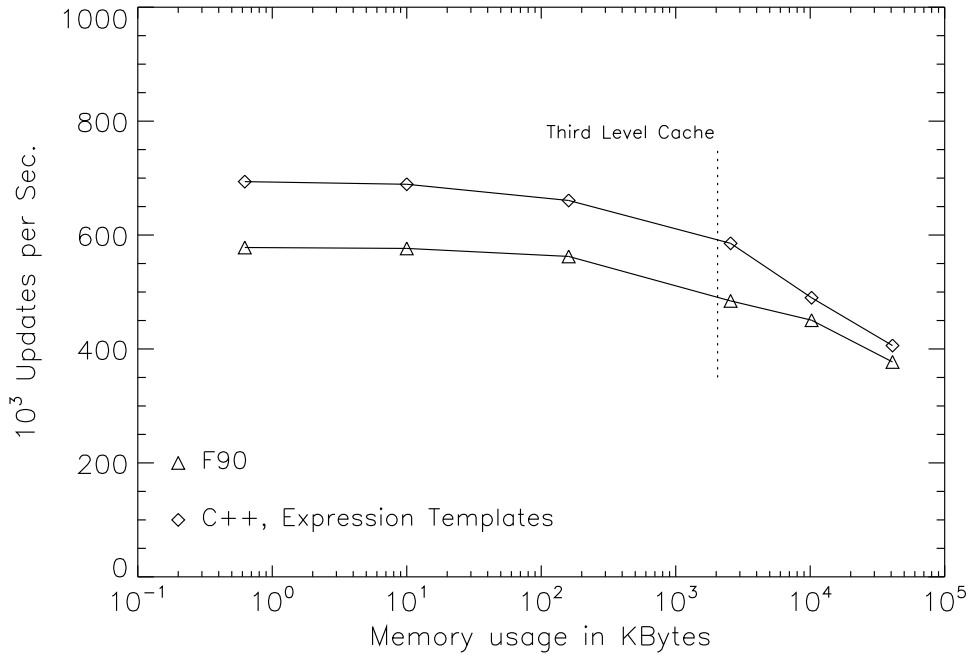


Figure D.1: Efficency of Monte-Carlo simulations of the 2d O(3) symmetric non linear σ-model with different lattice sizes on a SUN Ultra-Sparc.

Fortran90 code as one can see in figure D.1 for the same benchmark on a SUN workstation with different lattice sizes using the KAI C++ compiler. A similar result was also discovered by S. Booth from the UKQCD-Collaboration and T. Veldhuizen. For testing purposes they implemented their QCD-kernel routine in C++ using the Blitz++ library. On an IBM Power PC the C++ code was 20% faster than their old Fortran90 code [Vel97].

Because the SUN is not a typical numbercruncher the benchmark was also tested on a T3E for different lattice sizes. Since the Cray C++ Compiler has no additional object optimizer Fortran90 has a small margin to C++ on this machine as can be seen in figure D.2. However due to the small second level cache of 96KBytes and the missing third level cache the performance breaks down by a factor of two and more if the memory size for the fields becomes to large for the cache. Of course fields in dynamic fermion simulations do not fit into this cache. With this in mind performance differences are very small.

Similar results can be obtained by the simulation of a pure SU(2) gauge theory with the standard heatbath and overrelaxation Monte-Carlo algorithms [KP85]. To get optimal performance I used a combined technique composed of expression templates and temporary base classes. Again, C++ is a little faster than Fortran90 on a SUN workstation using the KAI C++ compiler.

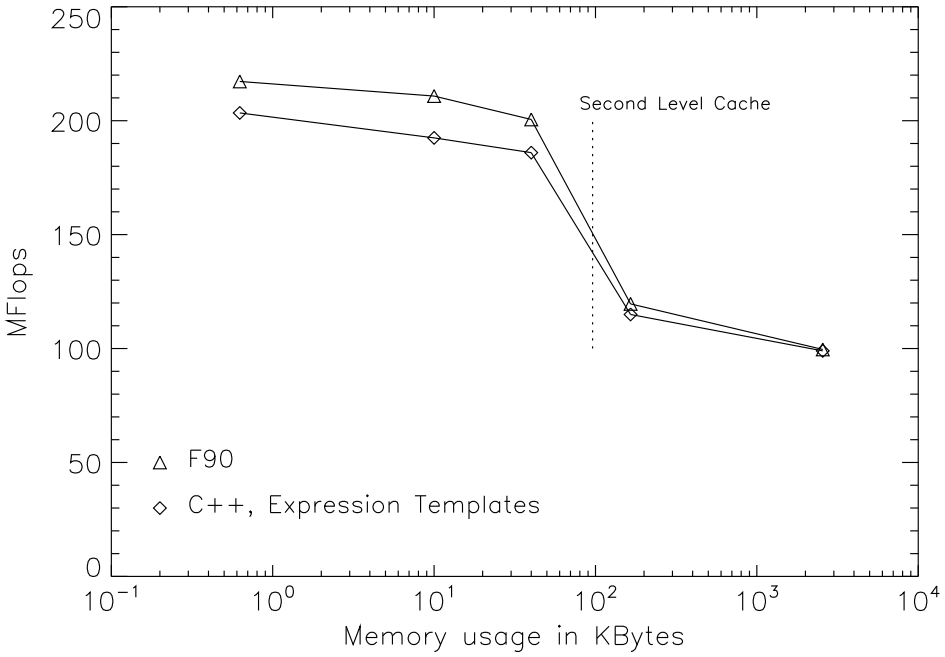


Figure D.2: Efficency of Monte-Carlo simulations of the 2d O(3) symmetric non linear σ-model with different lattice sizes on a T3E-512/600.

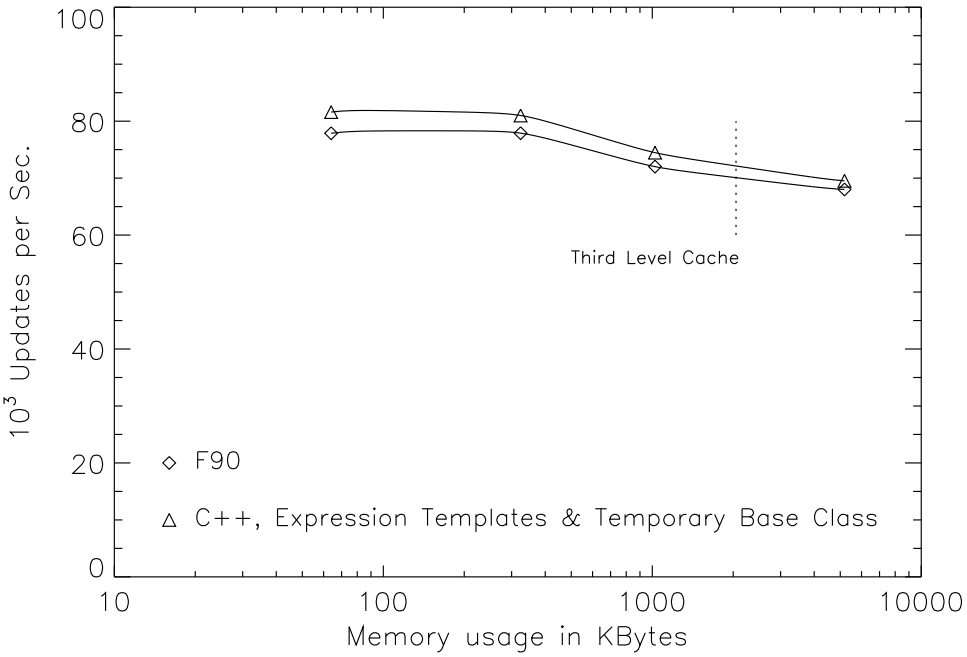


Figure D.3: Performance of a pure SU(2) lattice gauge field simulation on a SUN Ultra-Sparc in relation to the lattice size.

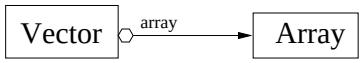
Appendix E

Hardware Independent Design

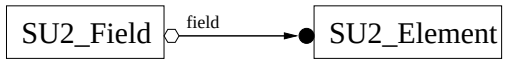
E.1 Algorithm Encapsulation

This chapter will describe how to implement local algorithms hardware independently. To achieve this more complex class hierarchies will be needed and thus more UML idioms.

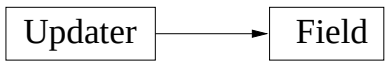
- aggregation



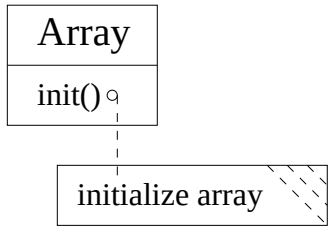
- more than one



- acquaintance



- pseudo code annotation



Often one object holds a reference to another object. For example a vector may delegate its container functionality to an array. The diamond indicates that both objects have the same lifetime. If a vector will be deleted the array will also be deleted to avoid memory leaks. A dot indicates that an object has more than one reference to another class. For example, a *SU2_Field* has many *SU2_Elements* and the elements have the same lifetime as *SU2_Field*. If an object keeps a reference to another object but both objects have different lifetimes this will be indicated by a simple arrowheaded line.

The implementation of algorithms is central to many computer programs especially to Monte Carlo simulations of quantum field theories. Often the first idea is, to let the code for an update algorithm be a member function of a lattice field object. But this binds the algorithm staticly to a certain field object. It is impossible to reuse this code for any other field object. Same algorithms are often suitable for

different clients (clients is used in terms of field objects). I found the so called strategy pattern very useful to define a family of algorithms and encapsulate each one [GHJV94]. The UML diagram shows the class hierarchy of this pattern. An

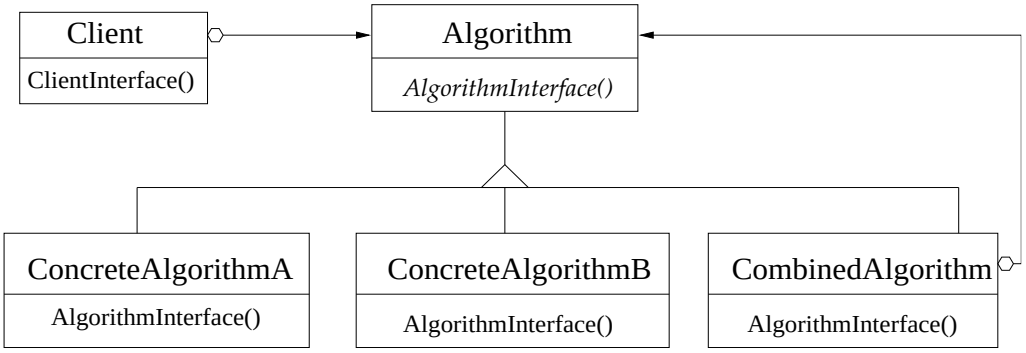


Figure E.1: UML diagram for a simple algorithm object.

abstract algorithm object is defined which provides one abstract member function at the point. No algorithm will be implemented by this class. It is only an abstract interface. Concrete algorithm classes inherit this interface and implement different algorithms. For example, let client be a *SU2_Field* class, algorithm A is a heatbath updater and algorithm B is an overrelaxation step. The optimal update algorithm is a combined algorithm which performs both update techniques. Let *CombinedAlgorithm* be a class which only contains a list of abstract algorithm references. If *CombinedAlgorithm* receives the request to perform an update it delegates this request to all members of the list. The client on which the algorithm should be applied is often a function argument of *AlgorithmInterface()*. This is a simple case of the more complex algorithm object pattern [VDW95].

E.2 Iterator Concept

Iterators are very common in object oriented class libraries. For example the algorithms of the Standard Template Library are based on iterators [GS96]. The main idea in the scope of local Monte Carlo algorithms is that one wants to write flexible, hardware independent algorithms but an integer is not smart enough or not flexible enough to control the loops of such an algorithm over the lattice. Instead of using a simple integer to perform this loop it is controlled by a special so called 'iterator object'.

The interface of this object is defined by an abstract iterator class. A 'lattice' iterator provides the following member functions: first(), next(), isDone() and getLocation(). The detailed balance of local Monte Carlo updaters does not depend on the sequence in which the iterator runs through the lattice. For example, it is possible to use one algorithm with an iterator which first visits the even sites and then odd sites or an iterator which runs through the lattice one after the other site. This can be an internal implementation detail of the iterator. The only thing the

iterator should ensure is that every site is visited exactly once during a sweep.

Furthermore iterators are imaginable which perform the domain decomposition on massive parallel systems. These iterators have to take care about the necessary message passing calls to keep up the consistency of the field in the distributed memory of the parallel computer. This technique separates message passing code from the proper algorithm. As a result it is possible to use the same code for an algorithm on a single CPU machine and on a parallel computer.

```
for (iterator->first(); !iterator->isDone(); iterator->next())
  {
    const int site(iterator->getLocation());

    /* perform local algorithm */
  }
```

The biggest advantage is that these algorithms can be programmed for a single CPU machine and can later on be used on a parallel machine. Here the complexity of the message passing code is passed to the iterator. Therefore different iterators are needed for different interaction types. This leads to a rather natural approach of solving the parallelization problem. Because the two problems of implementing the algorithm and the parallelization are solved separatly. Each part of the solution can further be reused. The iterator for the Ising Model could be used to implement the fermion matrix multiplication because both applications base on next neighbourhood interactions

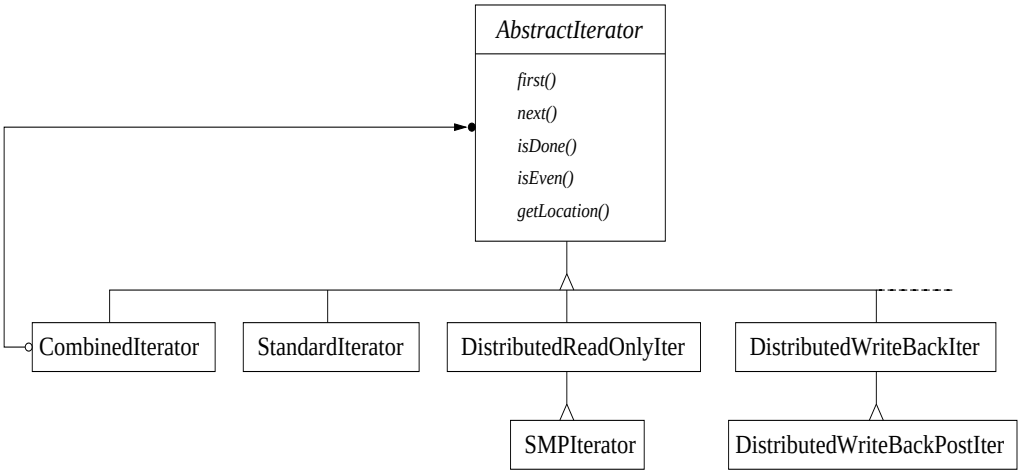


Figure E.2: UML diagram for the iterator class hierarchy.

In fact I successfully tested this iterator with a simple Ising model simulation. For that reason it is obvious that it will also work with other next-neighbourhood interactions, for example, the distribution of SU(2)-Adjoint-Fields. Another feature are the combined iterators. Often it is necessary to change multiple fields within a single

updater. By combining the related iterators to a single one and letting the loop over the lattice being carried out by it, the detailed balance of the whole algorithm is secured. The combined iterator will then delegate the corresponding requests to its clients.

It is very simple to write iterators that perform domain decomposition on SMP machines with global shared memory. Parallelization can be done easily with standard POSIX threads. In fact less than 700 lines of code were necessary to implement an SMP iterator. Unfortunateley I had no access to a high end SMP computer. As can be seen in figure E.3 it's not sporting to compare a standard dual PentiumPro PC with a T3E. Nevertheless it was surprising to see that a lattice gauge theory simulation does not scale properly on the PC. On the other hand this proves that the T3E has a very good scaling behaviour.

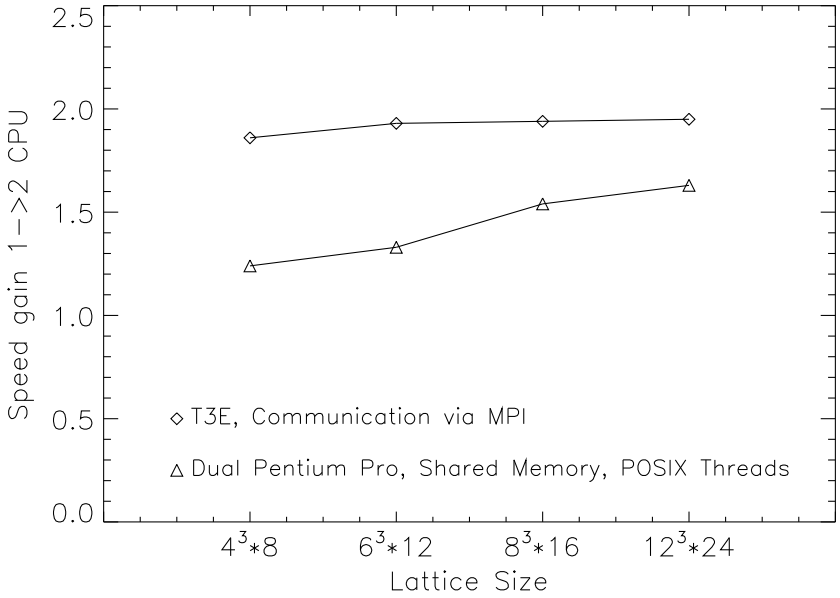


Figure E.3: Performance scaling for a pure SU(2) gauge simulation.

E.3 Abstract Factory Pattern

A problem for hardware independent design is the place where the iterators are being constructed. If they are directly constructed by the algorithm either single CPU iterators or distributed iterators have to be generated. As a result the algorithm becomes hardware-architecture dependent which is not desirable. This statical binding due to object construction typically is prevented by the so called abstract factory pattern [GHJV94]. The algorithms can not directly construct an iterator but have to ask the central object factory for an iterator. Depending on the kind of architecture the factory provides either single CPU, MPP or SMP iterators. There is

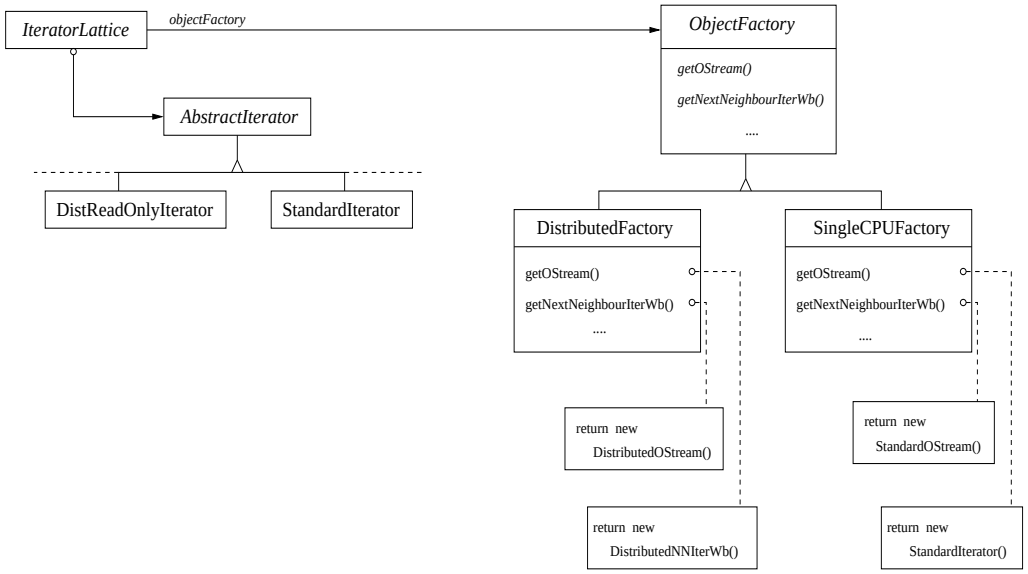


Figure E.4: UML diagram for the *object factory pattern*.

a single instance in the whole program which handles the hardware-dependencies. Another part of the program which should be handled by the object factory are the I/O-functions. A parallel computer for example, reads a lattice gauge configuration in a completely other way as a single CPU machine does. The obvious solution is to define an abstract I/O system and to develop two implementations, one for parallel machines and one for single CPU machines which simply can be derived from the standard C++ I/O stream. To keep the simulation code hardware independent I/O streams have to be constructed with the help of the object factory too.

E.4 Package Structure

Usually bigger object oriented programs break up into packages which are only loosely connected. Unfortunately C++ does not support the decomposition in modules as for example JAVA does. The package structure of the simulation looks like this:

The main ingredients are algorithms which act on fields. They make up 90% of the code. With the abstract iterator and I/O concept they are hardware independent. The hardware dependent objects, like iterators or I/O streams, should not be created by objects of these packages, but by the central object factory.

Depending on the hardware on which the program is running, the object factory generates the suitable objects. The only hardware dependent components are the iterators and the I/O system. The question might arise why a dedicated I/O system for SMP is missing. The answer is that it is not needed. Only the algorithms really use more than one thread. When the algorithm is completed all threads are joined

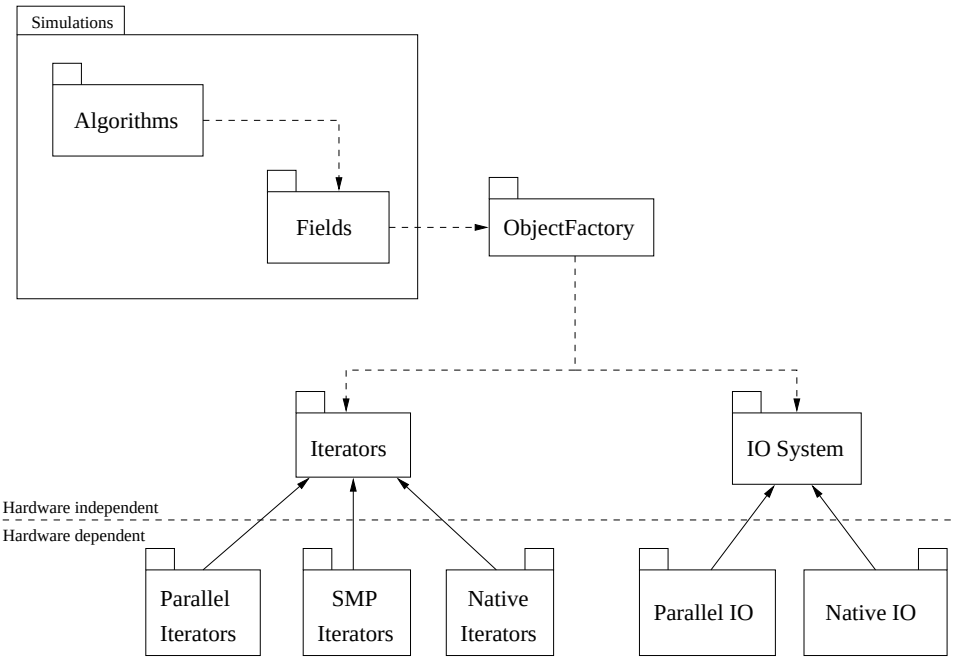


Figure E.5: UML packet structure diagram, showing the hardware dependent and independent part of the project.

to a single one and this one uses native I/O routines.

Appendix F

Case Study: SU(2)-Simulation with Gluinos

F.1 Component software

A simulation of the SU(2) Yang-Millstheory with dynamical gluinos contains many different types of fields. Since these fields have a lot of properties in common, it seems natural to organize them in a global class hierarchy. This is shown in diagram F.1. The parent class of all derived field classes is the *IteratorLattice*. It sets the lattice size and topology. In addition it demands that all field classes must have at least two iterators, one for read only access to the field and one for write access to the field.

Then the tree splits up into branches for *GaugeField*s which live on links and for *BosonicField*s which are located on the lattice sites. Both classes solely are code templates. In order to instantiate an object of one of these branches, one has to specify the proper field type. This is done with the help of trait classes [MYE95]. The main purpose of *GaugeField* and *BosonicField* is to manage the memory allocation of the fields and to allow I/O.

In addition *GaugeField* supports basic functionality like computing a plaquette as well as functions for field access to allow measure algorithms like *WilsonLoops* or *TeperBlocking* to act on the gauge field. In the further extend of this branch the *SU_N* object is derived. Among initialization routines this object comes with a general Metropolis updater and a cooling algorithm for SU(N) gauge fields. Again this object is solely a code template. In order to build a usable object out of this one has to specify the field type. This is tested with the existing three link type objects for the U(1), SU(2) and SU(3) gauge group. In the other branch the classes for the physical fields are directly derived from the class *BosonicField*. The many different subclasses benefit from the functionality of the parent class, code does not have to be reprogrammed over and over again.

All measure and update algorithms on fields stand and fall with the iterator pro-

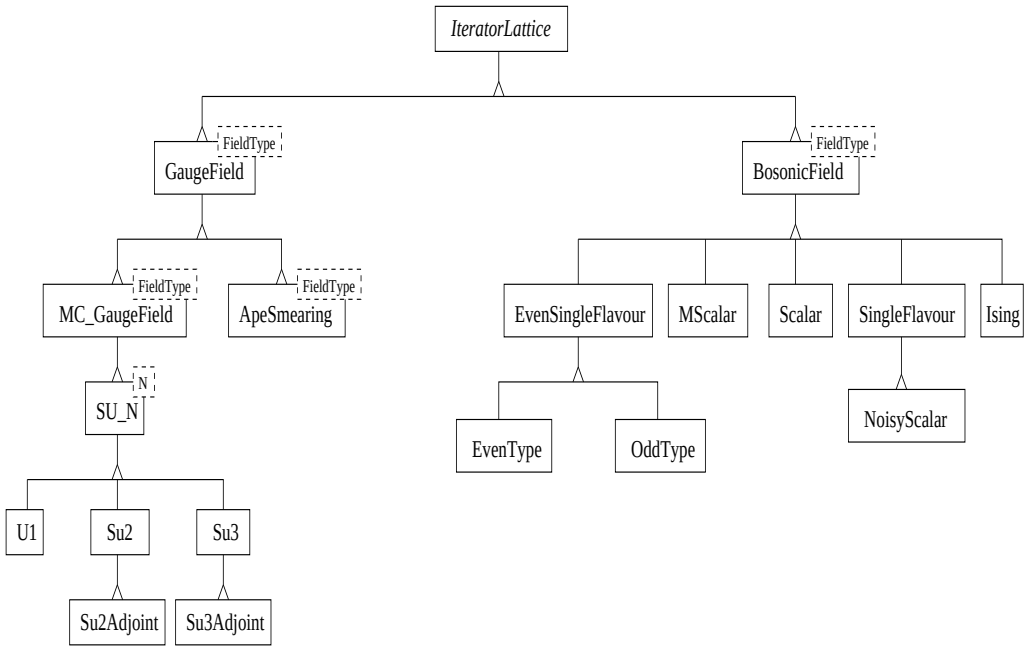


Figure F.1: Class hierarchy for different field objects.

vided by the field object. As shown in the last chapter iterators are constructed by an object factory in order to achieve hardware independency of the code. It is the responsibility of the object factory to project a simple request for an iterator to the related construction process which is suited for the given hardware.

UML does not only offer diagrams for statical class hierarchies but also sequence diagrams for presenting dynamical processes. Diagram F.3 shows the sequence diagram for the construction of a gauge field iterator on a single CPU machine while diagram F.2 does the same for a multi processor machine. One can be see, the complexity of the construction process varies greatly between a single and a multi processor machine. In contrast to the process on a single CPU machine where the object factory is capable of constructing the iterator on the fly, it needs the assistance of an object from the type *DistributedGaugeField* on a parallel computer. Because the *DistributedGaugeField* controls all message passing actions on the gauge field it has to know the exact positions of the links in memory. Therefore it interviews the *GaugeField* at the time of construction about its memory layout.

The iterator *DistributedIteratorWb* for a parallel machine does not contain any message passing, it solely delegates the necessary requests to secure the persistence of the gauge field in the distributed memory of the computer. In this context persistence means that each processor has the same local copy of each physical link in its own memory. Since message passing functionality tends to be defective self tests on every level have to be performed. Especially while deleting an object from the type *DistributedGaugeField* it has to be assured that the object is cleanly separated from the message passing system.

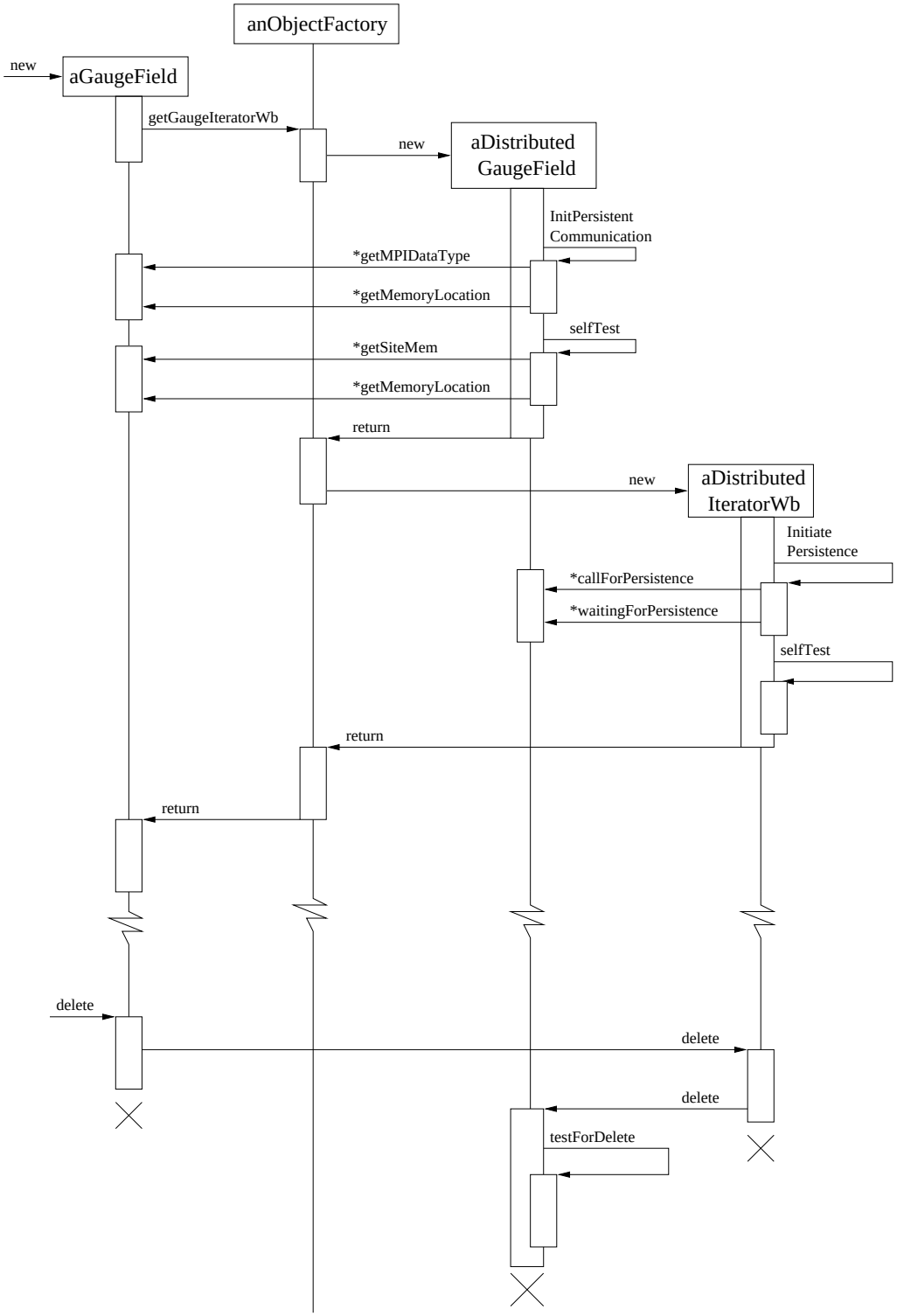


Figure F.2: UML sequence diagram for the life cycle of a gauge field iterators on a parallel computer.

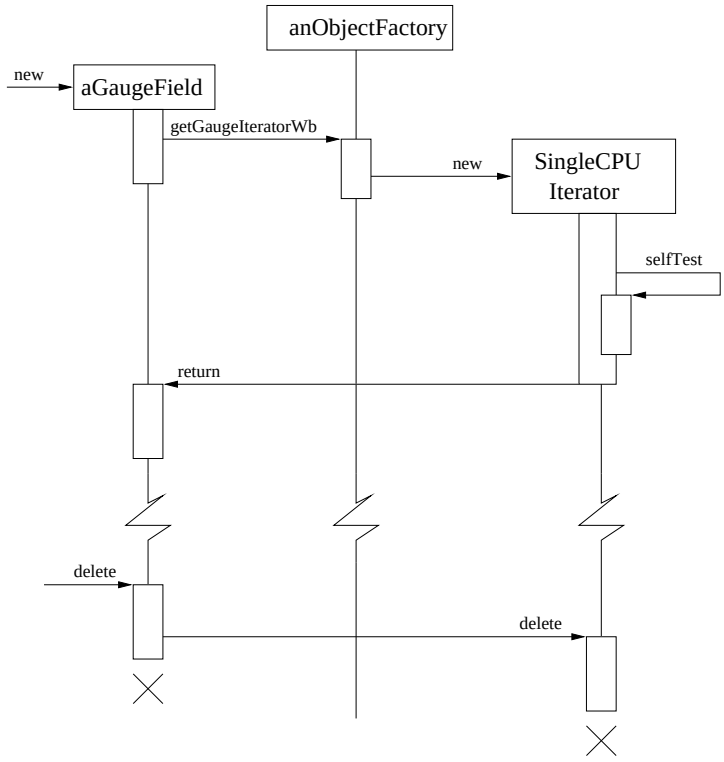


Figure F.3: UML sequence diagram for the life cycle of a gauge field iterator on a computer with a single CPU.

The code for this project splits up into independent modules which are arranged in a top down layer model as can be seen in figure F.4. The foundation consists of the classes which implement base functionality for different hardware architectures and the classes which are necessary for efficient numeric with C++. On top of these lies the object factory which serves as a mediator between the different hardware types. Above this level all code which strictly uses the object factory becomes hardware independent. It is now possible for the next level to assemble field objects. The last two levels are the most interesting ones because here is where true physics happens. Deeper knowledge especially on the message passing library is not necessary (as long as no new interaction types like e.g. clover terms are introduced). A special rôle plays the analysis part of the project. It was necessary to develop it for single CPU machines only and it provides completely different functionality as the rest of the program. Therefore it is only loosely connected to the rest of the program.

F.2 Hardware review

One has to think about the proper hardware architecture to use. As can be seen in diagram E.3 in the last chapter PC's based on symmetrical multi processor technology are not suited. The obvious idea is to use a workstation cluster connected via standard network hardware like Ethernet or FDDI as a low cost solution. An ex-

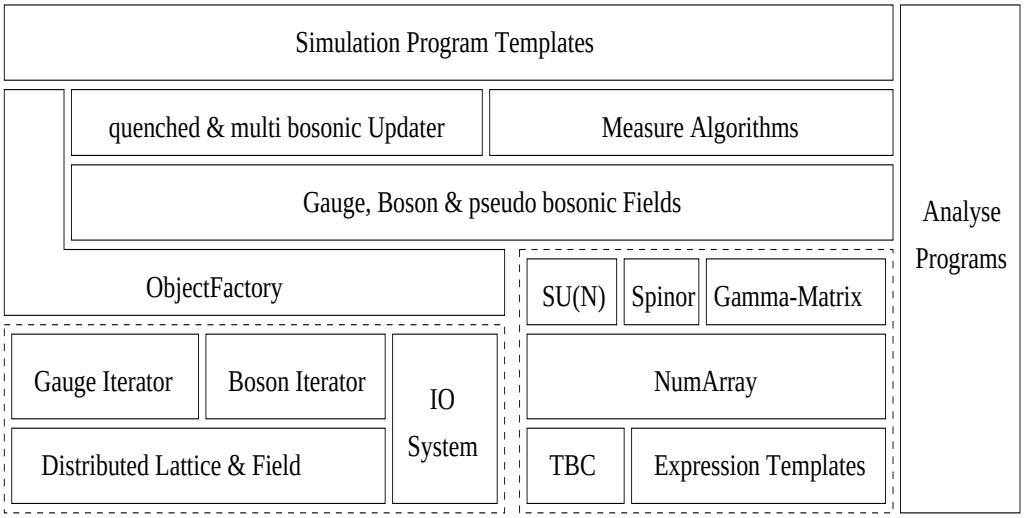


Figure F.4: The single components of the simulation software.

ample for a highly cost efficient supercomputer is the recently assembled AVALON machine at LANL [SCB98] which consists of 70 Alpha PCs linked with Fast Ethernet.

The first questions that arise when this type of hardware is taken into consideration for the simulation of dynamical fermions are: is the bandwidth sufficient and are the latency times short enough. When simulating a very large lattice with only few workstations the bandwidth of standard network technology might be just large enough. What makes this technology useless for this problem is the latency time (typically about 1ms). This is only partially caused by the hardware but also by the large protocol overhead of the operating system. Due to the utilization of scalar support fields in the existing case (see section 3.2.3) a large amount of very small packages has to be send between the nodes. As a result small latency times are 'mission - critical'. Keeping this in mind, it is not surprising that the software for the dynamical gluino simulation in fact runs faster on one workstation than on a standard linked cluster. As shown in the diagram F.5 the T3E outperforms state of the art network technology by a factor of thirty in bandwidth and by a factor of 100 in latency times.

Nevertheless it is important for a good scaling with the amount of nodes that computation and communication may overlap. To minimize message passing overhead communication has to be done with boundary faces instead of boundary sites. The scaling behaviour of parallel computers can be modelled by the Amdahl relation

$$S_P(N) = \frac{1}{s + \frac{1-s}{N}}. \tag{F.1}$$

Here $S_P(N)$ describes the speed gain for N nodes and s is the non-parallelizable portion of the problem. As diagram F.6 shows the pure gauge updater and the local

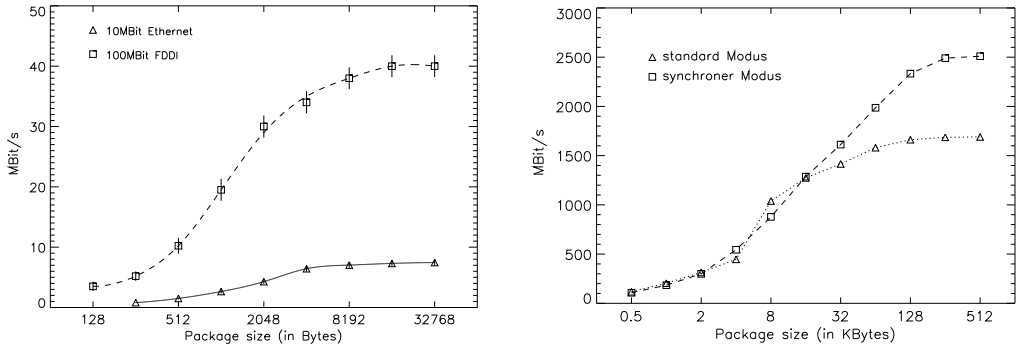


Figure F.5: Bandwidth on a standard linked cluster measured with PVM (left) and on a T3E-512/600 measured with the Cray implementation of MPI (right).

bosonical updater are well represented by the Amdahl law. Consequently the non-parallelizable portion of the pure gauge updater runs up to 0.7% and 1.8% for the local bosoincal updater. The reason for the worse scaling behaviour of the dynamical gluino updater is once again the scalar support field.

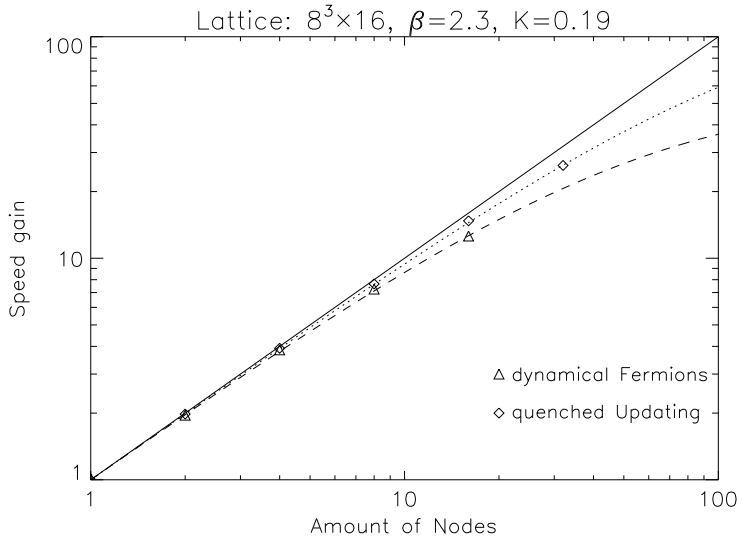


Figure F.6: Scaling behaviour of a T3E-256/900 for pure SU(2) and for MBA Updater.

Another suitable architecture are vector computers like the Cray-T90. In fact at the beginning of the project, earlier versions of the non preconditioned updater written in Fortran90 were running on that machine. Therefore it was possible to directly compare the T90 with the T3E and its predecessor the T3D. Though one vector processor of the T90 was 3.6 times faster than one node of the T3E-256/900 the T90 is not that suited due to the fact that one CPU hour is 25 times more expensive

than on the T3E [1] . On the other hand the implementation of such a large scale project on a parallel computer using message passing software is a complicated task.

Algorithm	T3D	T3E	T90
Update	54.0s	13.4s	3.7s
Matrix Inversion	18.2s	4.30s	1.40s

Table F.1: CPU time on T3D, T3E-256/900 and on a T90. The parallel machines ran the C++ code while the vector machine used the old optimized Fortran90 code. Performance measurements in MFlops on the T3D/E are not available due to problems with the Cray performance tool 'Apprentice' and C++ code.

[1] For details see KFA accounting informations.

Literaturverzeichnis

[A87] M. Albanese et al. Glueball Masses and String Tension in Lattice QCD. *Phys. Lett.* **B192** (1987) 163.

[A96] S. Aid et al. A Search for Selectrons and Squarks at HERA. *hep-ex/9605002, Phys. Lett.* **B380** (1996) 461.

[A98a] K. Ackerstaff et al. A Search for Neutral Higgs Bosons in the MSSM and Models with Two Scalar Field Doublets. *hep-ex/9803019, Eur. Phys. J.* **C5** (1998) 19.

[A98b] K. Ackerstaff et al. Search for the Standard Modell Higgs Boson in e^+e^- Collisions at $\sqrt{s} = 161 - 172\,\mathrm{GeV}$. *hep-ex/9709003, Eur. Phys. J.* **C1** (1998) 425.

[A98c] W.W.M. Allison et al. Search for the Proton decay mode $p \to K^+\nu$ in Soudan 2. *hep-ex/9803030, Phys. Lett.* **B427** (1998) 217.

[AGDKM96] L. Alvarez-Gaume, J. Distler, C. Kounnas, and M. Marino. Large Softly broken N=2 QCD. *hep-th/9604004, Int. J. Mod. Phys.* **A11** (1996) 4745.

[AGH97] L. Alvarez-Gaume and S.F. Hassan. Introduction to S-Duality in N=2 Supersymmetric Gauge Theory. (A pedagogical review of the work of Seiberg and Witten). *hep-th/9701069, Fortsch. Phys.* **45** (1997) 159.

[AKM88] D. Amati, K. Konishi, Y. Meurice, G.C. Rossi, and G. Veneziano. Non-perturbative aspects in supersymmetric gauge theories. *Phys. Rep.* **162** (1988) 169.

[B98a] A. Bartoloni et al. Progress and status of APEmille. *Nucl. Phys.* **B63**(991) (1998).

[B98b] S.R. Blusk et al. Measurment of the Top Quark Mass. *hep-ex/9805035* (1998).

[BBJM98] M.E. Berbenni-Bitsch, A.D. Jackson, S. Meyer, A. Schäfer, J.J.M. Verbaarschot, and T. Wettig. Random-matrix universality in the small-eigenvalue spectrum of the lattice Dirac operator. *hep-lat/9709102, Nucl. Phys. Proc. Suppl.* **B63** (1998) 820.

[BBMS98] M.E. Berbenni-Bitsch, S. Meyer, A. Schäfer, J.J.M. Verbaarschot, and T. Wettig. Microscopic universality in the spectrum of the lattice Dirac operator. *hep-lat/9704018, Phys. Rev. Lett.* **80** (1998) 1146.

[BDE98] W. Bardeen, A. Duncan, E. Eichten, G. Hockney, and H. Thacker. Resolving Exceptional Configurations. *Nucl. Phys. Proc. Suppl.* **B63** (1998) 141.

[BdFG96] A. Borrelli, Ph. de Forcrand, and A. Galli. Non-hermitian Exact Local Bosonic Algorithm for Dynamical Quarks. *Nucl. Phys.* **B477** (1996) 809.

[BEFJ98] B. Bunk, S. Elser, R. Frezzotti, and K. Jansen. Ordering monomial factors of polynomials in the product representation. *hep-lat/9805026* (1998).

[Ber92] B.A. Berg. Double Jackknife Bias Corrected Estimators. *Comp. Phys. Commun.* **69** (1992) 7.

[Bie98] W. Bietenholz. Excat Supersymmetry on the Lattice. *hep-lat/9807010* (1998).

[BJJ95] B. Bunk, K. Jansen, B. Jegerlehner, M. Lüscher, H. Simma, and R. Sommer. A new simulation algorithm for lattice QCD with dynamical quarks. *hep-lat/9411016, Nucl. Phys. Proc. Suppl.* **B42** (1995) 49.

[BL94] D. Bailin and A. Love. *Supersymmetric Gauge Field Theory and String Theory.* Institute of Physics Publishing, 1994.

[BN94] J.J. Barton and L.R. Nackman. *Scientific and Engineering C++.* Addison Wesley Pusblishing Company, Inc., 1994.

[Boo97] G. Booch. *Object-Oriented Analysis and Design with Applications.* The Benjamin/Cummings Publishing Company, Inc., 1997.

[Bor96] A. Boriçi. Krylov Supspace Methods in Lattice QCD. Technical report, Swiss Center for Scientific Computing, 1996.

[BSS95] G.S. Bali, K. Schilling, and C. Schlichter. Observing Long Colour Flux Tubes in SU(2) Lattice Gauge Theory. *hep-lat/9409005, Phys. Rev.* **D51** (1995) 5165.

[Bun97] B. Bunk. Computing the lowest eigenvalues of the Fermion matrix by subspace iterations. *hep-lat/9608109, Nucl. Phys. Proc. Suppl.* **B53** (1997) 987.

[Bun98] B. Bunk. Fractional Inversion in Krylov Space. *hep-lat/9805030, Nucl. Phys. Proc. Suppl.* **B63** (1998) 952.

[C93] S. Collins et al. Gauge Invariant Smearing and Matrix Correlators using Wilson Fermions at beta=6.2. *hep-lat/9211039, Phys. Rev.* **D47** (1993) 5128.

[C96] J.S. Conway et al. Recent CDF Results. *Nucl. Phys. Proc. Supp.* **B52A** (1996) 8.

[CM67] S. Coleman and J. Mandula. All Possible Symmetries of the S Matrix. *Phys. Rev.* **159** (1967) 1251.

[CMT87] B. Carpenter, C. Michael, and M. Teper. 0^+ and 2^+ Glueball Masses from Large Lattices in SU(2) Lattice Gauge Theory. *Phys. Lett.* **B198** (1987) 511.

[Col] DESY-Münster Collaboration. private communication.

[Cop92] J.O. Coplien. *Advanced C++ Programming Styles and Idioms.* Addison Wesley Pusblishing Company, Inc., 1992.

[Cre92] M. Creutz. *Quantum Fields on the Computer.* World Scientific, 1992.

[CV87] G. Curci and G. Veneziano. Supersymmetry and the Lattice: a Reconciliation. *Nucl. Phys.* **B292** (1987) 555.

[DG96] A. Donini and M. Guagnelli. Hybird Molecular Dynamics for Lattice Supersymmetry. *hep-lat/9605010, Phys. Lett.* **B383** (1996) 301.

[DGHV98] A. Donini, M. Guagnelli, P. Hernandez, and A. Vladikas. Towards N=1 Super-Yang-Mills on the Lattice. *hep-lat/9710065, Nucl. Phys.* **B523** (1998) 529.

[Din96] M. Dine. Supersymmetry Phenomenology With a Broad Brush. *hep-ph/9612389* (1996).

[EB98] S. Elser and B. Bunk. Local bosonic versus HMC — a CPU cost comparison. *hep-lat/9709121, Nucl. Phys. Proc. Suppl.* **B63** (1998) 940.

[EHN98] R.G. Edwards, U.M. Heller, and R. Narayanan. Spectral flow, condensate and topology in lattice QCD. *hep-lat/9802016* (1998).

[EHS97] N. Evans, S.D.H. Hsu, and M. Schwetz. Lattice Tests of Supersymmetric Yang-Mills Theory? *hep-th/9707260* (1997).

[Eli75] S. Elitzur. Impossibility of Spontaneously Breaking Local Symmetries. *Phys. Rev.* **D12** (1975) 3978.

[F97] S. Fischer et al. A Parallel SSOR Preconditioner for Lattice QCD. *hep-lat/9608066, Nucl. Phys. Proc. Suppl.* **B53** (1997) 990.

[F98] Y. Fukuda et al. Study of the atmospheric neutrino flux in the multi-GeV energy range. *hep-ex/9805006* (1998).

[FGS98] G.R. Farrar, G. Gabadadze, and M. Schwetz. The Spectrum of Softly Broken N=1 Supersymmetric Yang-Mills Theory. *hep-th/9806204* (1998).

[FP89] H. Flyvbjerg and H.G. Petersen. Error Estimates on Averages of Correlated Data. *J. Chem. Phys.* **91** (1989) 461.

[FS97] M. Fowler and K. Scott. *Applying the Standard Object Modeling Language.* Addison Wesley Pusblishing Company, Inc., 1997.

[GG82] L. Girardello and M.T. Grisaru. Soft Breaking of Supersymmetry. *Nucl. Phys.* **B194** (1982) 65.

[GHJV94] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software.* Addison Wesley Pusblishing Company, Inc., 1994.

[GKMS96] D.J. Gross, I.R. Klebanov, A.V. Matytsin, and A.V. Smilga. Screening vs. Confinement in 1+1 Dimensions. *hep-th/9511104, Nucl. Phys.* **B461** (1996) 109.

[GL71] Y. Golfand and E. Likhtman. Extension of the Algebra of Poincare Group Generators and Violation of P Invariance. *JETP Lett.* **13** (1971) 323.

[GL91] G.H. Golub and C. Loan. *Matrix Computations.* The Johns Hopkins University Press, 1991.

[GLM90] S. Güsken, U. Löw, K.-H. Mütter, R. Sommer, A. Patel, and K. Schilling. A Study of Smearing Techniques for Hadron Correlation Functions. *Nucl. Phys. Proc. Supp.* **B17** (1990) 361.

[GPB91] R. Gupta, A. Patel, C.F. Bailie, G.W. Kilcup, and S.R. Sharpe. Exploring glueball wave functions on the lattice. *Phys. Rev.* **D43** (1991) 2301.

[Gro96] E. Gross. Search for Supersymmetry at $\sqrt{s} = 130 - 160\,\mathrm{GeV}$ at LEP. *Nucl. Phys. Proc. Supp.* **B52A** (1996) 3.

[GRS79] M. Grisaru, M. Rocek, and W. Siegel. Improved Methods for Supergraphs. *Nucl. Phys.* **B159** (1979) 429.

[GRTV98] L. Giusti, F. Rapuano, M. Talevi, and A. Vladikas. The QCD Chiral Condensate from the Lattice. *hep-lat/9807014* (1998).

[GS96] G. Glass and B. Schuchert. *The STL Primer.* Prentice Hall PTR, 1996.

[Gun90] J. Gunion. *The Higgs Hunter's Guide.* Addison-Wesley, Redwood City, 1990.

[HLS75] R. Haag, J.T. Lopuszanski, and M. Sohnius. All Possible Generators of Supersymmetries of the S Matrix. *Nucl. Phys.* **B88** (1975) 257.

[HR97] J.L. Hewett and T.G. Rizzo. Don't Stop Thinking About Leptoquarks: Constructing New Models. *SLAC-PUB-7549, submitted to Physical Rev. D* (1997).

[Hsu98] S.D.H. Hsu. Gaugino Determinant in Supersymmetric Yang-Mills Theory. *hep-th/9704149, Mod. Phys. Lett.* **A13** (1998) 673.

[Jeg97] B. Jegerlehner. Improvements of the local bosonic algorithm. *hep-lat/9612013, Nucl. Phys. Proc. Suppl.* **B53** (1997) 959.

[KFM94] Y. Kuramashi, M. Fukugita, H. Mino, M. Okawa, and U. Ukawa. η' Meson Mass in Lattice QCD. *Rev. Lett.* **72** (1994) 3448.

[KLM98] R. Kirchner, S. Luckmann, I. Montvay, K. Spanderen, and J. Westphalen. Numerical Simulation of Dynamical Gluinos: First Results. *to appear in the Lattice 98 Proceedings* (1998).

[KM97] G. Koutsoumbas and I. Montvay. Gluinos on the Lattice: Quenched Calculations. *hep-lat/9612003, Phys. Lett.* **B398** (1997) 130.

[KP85] A.D. Kennedy and B.J. Pendleton. Improved Heatbath Method for Monte Carlo Calculations in Lattice Gauge Theories. *Phys. Lett.* **B156** (1985) 393.

[KS96] T. Kalkreuter and H. Simma. An Accelerated Conjugate Gradient Algorithm to Compute Low-Lying Eigenvalues — a Study for the Dirac Operator in SU(2) Lattice QCD. *hep-lat/9507023, Comput. Phys. Commun.* **93** (1996) 33.

[KS97] A. Kovner and M. Shifman. Chirally Symmetric Phase of Supersymmetric Gluodynamics. *hep-th/9702174, Phys. Rev.* **D56** (1997) 2396.

[Lan95] P. Langacker. *Precision Tests of the Standard Electroweak Model.* World Scientific, 1995.

[Lüs94] M. Lüscher. A New Approach to the Problem of Dynamical Quarks in Numerical Simulations of Lattice QCD. *hep-lat/9311007, Nucl. Phys.* **B418** (1994) 637.

[MM94] I. Montvay and G. Münster. *Quantum Fields on a Lattice.* Cambridge University Press, Cambridge UK, 1994.

[Mon] I. Montvay. private communication.

[Mon96] I. Montvay. An Algorithm for Gluinos on the Lattice. *hep-lat/9510042, Nucl. Phys.* **B466** (1996) 259.

[Mon98a] I. Montvay. Quadratically optimized polynomials for fermion simulations. *hep-lat/9707005, Comput. Phys. Commun.* **109** (1998) 144.

[Mon98b] I. Montvay. SUSY on the Lattice. *hep-lat/9709080, Nucl. Phys. Proc. Suppl.* **B63** (1998) 108.

[Mon98c] I. Montvay. SYM on the Lattice. In *Theory of Elementary Particles*, edited by H. Dorn, D Lüst, and G. Weigt, page 361, 1998.

[MT87] C. Michael and M. Teper. Universality and scaling in SU(2) Lattice Gauge Theory. *Phys. Lett.* **B199** (1987) 95.

[MY93] H. Murayama and T. Yanagida. Nucleon decay in the minimal supersymmetric SU(5) grand unification. *Nucl. Phys.* **B402** (93) 46.

[Mye95] N.C. Myers. Traits: a new and useful template technique. *C++ Report* (June 1995) 102.

[Neu98] H. Neuberger. More about exactly massless quarks on the lattice. *hep-lat/9801031, Phys. Lett.* **B427** (1998) 353.

[NSVZ83] V.A. Novikov, M.A. Shifman, A.I. Vainshtein, and V.I. Zakharov. Exact Gell-Mann-Low Function of Supersymmetric Yang-Mills Theories From Instanton Calculus. *Nucl. Phys.* **B229** (1983) 381.

[PT97] W.H. Press and S.A. Teukolsky. Numerical Recipes: Does this Paradigm Have a Future? *Computers in Physics* (Sep/Oct 1997) 416.

[PTVF94] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, Melbourne, 1994.

[Riz92] T.G. Rizzo. Desert grand unified theories and new light degress of freedom. *Phys. Rev.* **D45** (1992) 3903.

[Rob96] A.D. Robison. C++ Gets Faster for Scientific Computing. *Computers in Physics* (Oct 1996) 458.

[Roe91] G. Roepstorff. *Pfadintegrale in der Quantenphysik.* Vieweg, 1991.

[Ros84] G.G. Ross. *Grand Unified Theories.* Benjamin/Cummings, Menlo Park, California, 1984.

[S98] M. SHIOZAWA et al. Search for Proton Decay via $p \to e^+\pi^0$ in a Large Water Cherenkov Detector. *hep-ex/9806014, to appear in Phys. Rev. Lett.* (1998).

[SCB98] T. STERLING, T. CWIK, D. BECKER, J. SALMON, M. WARREN, and B. NITZBERG. An assessment of Beowulf-class computing for NASA requirements: Initial findings from the first NASA workshop on Beowulf-class clustered computing. *http://loki-www.lanl.gov/papers, Proceedings, IEEE Aerospace Conference.* (1998).

[SEI95] N. SEIBERG. Electric - Magnetic Duality in Supersymmetric Nonabelian Gauge Theories. *hep-th/9411149, Nucl. Phys.* **B435** (1995) 129.

[SOK89] A.D. SOKAL. Monte Carlo Methods in Statistical Mechanics: Fundations and New Algorithms. *Cours de Troisième Cycle de la Physique en Suisse Romande, Lausanne* (1989).

[STR97] B. STROUSTRUP. *C++ Programming Language.* Addison Wesley Pusblishing Company, Inc., third edition edition, 1997.

[SW94a] N. SEIBERG and E. WITTEN. Electric - Magnetic Duality, Monopole Condensation and Confinement in N=2 Supersymmetric Yang-Mills Theory. *hep-th/9407087, Nucl. Phys.* **B426** (1994) 19.

[SW94b] N. SEIBERG and E. WITTEN. Monopoles, Duality and Chiral Symmetry Breaking in N=2 Supersymmetric QCD. *hep-th/9408099, Nucl. Phys.* **B431** (1994) 484.

[SWWW97] M.M. STEINER, W. WENZEL, J.W. WILKINS, and K.G. WILSON. Object-Oriented Techniques in Large Scientific Computing Projects: Experience with C++. *Computers in Physics* (Sep/Oct 1997) 467.

[SX96] K. SPANDEREN and Y. XYLANDER. Effiziente Numerik mit C++. *iX, Magazin für professionelle Informationstechnik* **11** (1996) 166.

[SZ97] M. SCHWETZ and M. ZABZINE. Gaugino Condensate And Veneziano-Yankielowicz Effective Lagrangian. *hep-th/9710125* (1997).

[TAL97] D. TALKENBERGER. *Monte-Carlo Simulationen von Modellen der Elementarteilchenphysik mit dynamischen Fermionen.* PhD thesis, Universität Münster, 1997.

[VDW95] R. VAN DER WAL. Algorithmic Objects. *C++ Report* (May 1995) 23.

[VEL95a] T. VELDHUIZEN. Expression Templates. *C++ Report* (June 1995) 26.

[Vel95b] T. Veldhuizen. Using C++ template metaprograms. *C++ Report* (May 1995) 36.

[Vel97] T. Veldhuize. Scientific Computing: C++ versus Fortran. *Dr. Dobb's Journal* (Nov 1997).

[Ver94] J. Verbaarschot. Spectrum of the QCD Dirac Operator and Chiral Random Matrix Theory. *Phys. Rev. Lett.* **72** (1994) 2531.

[VJ97] T. Veldhuizen and M. Jernigan. Will C++ be faster than Fortran. talk given at ISCOPE'97, http://monet.uwaterloo.ca/tveldhui/papers/iscope97.ps, 1997.

[VKG97] L. Venkataraman, G. Kilcup, and J. Grandy. The Staggered η' with Smeared Operators. *Nucl. Phys.* **B53** (1997) 259.

[VY82] G. Veneziano and S. Yankielowicz. An Effective Lagrangiean for the Pure N=1 Supersymmetric Yang-Mills Theory. *Phys. Lett.* **B113** (1982) 231.

[Wes96] G.B. West. Theorem on the Lightest Glueball State. *hep-lat/9603316, Phys. Rev. Lett.* **77** (1996) 2622.

[WZ74] J. Wess and B. Zumino. Supergauge Tranformations in Four Dimensions. *Nucl. Phys.* **B70** (1974) 39.

Am Schluß dieser Arbeit möchte ich mich bedanken

- bei Herrn Prof. Dr. Gernot Münster, für die gute Betreuung und für seine Anregung einer Teilnahme an einem gemeinsamen Projekt mit einigen Mitgliedern der DESY-Theoriegruppe.
- bei Herrn Dr. István Montvay für sein Interesse am Fortgang meiner Arbeit und für seine stets vorhandene Bereitschaft zu ausgiebigen Gesprächen.
- bei den Mitgliedern der DESY-Münster-SUSY Kollaboration, insbesondere bei Dirk Talkenberger, für die gute Zusammenarbeit.
- bei York Xylander für die ungezählten Software-Meetings über objektorientiertes Programmieren auf der wohl „bequemsten“ Treppe der Welt (Hallo Venessa, Du bist noch wach?).
- bei Bodo Junglas, für viele Diskussionen über objektorientiertes Design und für sein unerschöpfliches Reservoir an C++ Tips & Hacks.
- bei Büro-Jesus Jens Küster, der leider trotz einer intensiven, dreijährigen Beschallung nicht zum richtigen Musik-Glauben konvertierte.
- bei Bernd Nottelmann, durch den man beim Mittags-Kaffee immer wieder auf den neuesten Stand der Linux-Entwicklung gebracht wird.
- bei Klaus Pinn, der mich ab und zu auch mal beim Dart gewinnen ließ und viele wertvolle Tips zu Monte-Carlo-Simulationen gab.
- bei Fabian Weis für die Anwendungen des „Fabian-Filters“ auf die Anhänge C bis F.
- Elisabeth und Toni Junker für die Erkenntnis, daß man „Autositz“ und nicht „Auto-Sitz“ schreibt.
- bei Silke Luckmann, Christian Demmer und Johannes Göttker-Schnetmann für die vielen sinnvollen und „unsinnvollen“ Diskussionen zur Physik, zu richtigen Computern und zu Rechenschiebern mit intuitiver Benutzeroberfläche (Modula2, ist das nicht ein Müsliriegel?).