DEUTSCHES ELEKTRONEN-SYNCHROTRON

DESY 95-134
July 1995

A Multiscale View of Propagators in Gauge Fields

M. Bäker
Fachbereich Physik, Universität Hamburg

ISSN 0418-9833

NOTKESTRASSE 85 - 22607 HAMBURG

To be sure that your preprints are promptly included in the
HIGH ENERGY PHYSICS INDEX,
send them to (if possible by air mail):

DESY
Bibliothek
Notkestraße 85
22607 Hamburg
Germany

DESY-IfH
Bibliothek
Platanenallee 6
15738 Zeuthen
Germany

DESY 95-134
July 1995

ISSN 0418-9833

A Multiscale View of Propagators in Gauge Fields

Dissertation
zur Erlangung des Doktorgrades
des Fachbereichs Physik
der Universität Hamburg

vorgelegt von
Martin Bäker
aus Bremen

Hamburg
1995

Gutachter der Dissertation:	Prof. Dr. G. Mack Prof. Dr. H. Joos
Gutachter der Disputation:	Prof. Dr. G. Mack Prof. Dr. K. Fredenhagen
Datum der Disputation:	6. 7. 1995
Sprecher des Fachbereichs Physik und Vorsitzender des Promotionsausschusses:	Prof. Dr. E. Lohrmann

ABSTRACT

Calculating the properties of hadrons from first principles (the Lagrangian of Quantum Chromodynamics) is one of the grand challenges in modern theoretical physics. Large collaborations are trying to do so using high-performance computers. The simulation of the full theory, including dynamical fermions, requires frequent inversion of the Dirac operator, i.e. the solution of a discretized differential equation on large domains. As the Dirac operator has a large condition number because of low-lying eigenvalues, standard methods like Conjugate Gradient and Overrelaxation suffer from critical slowing down: The convergence rate usually decreases proportional to the linear extension of the grid.

For standard differential equations multigrid methods have been very successful in the past. However, the inversion of the Dirac operator is a problem involving disorder because of the gauge field. Non-standard multigrid methods are therefore required that are able to work also when disordered gauge fields are present.

In this thesis, the Iteratively Smoothing Unigrid or ISU is presented and studied in detail. An appropriate definition of smoothness in the presence of disordered gauge fields is given on which the algorithm is based. ISU uses an iterative process to determine the interpolation operators as the eigenvectors to the lowest eigenvalues on blocks of larger and larger sizes. One of the crucial differences to other methods is that this algorithm is a *unigrid*. This means that after each calculation on a coarse grid the change in the field is transported back directly to the finest grid, and blocking is also done always from the finest grid to any of the coarser grids. The advantage of this is that the interpolation operators can be chosen from a much larger space.

The method is possible due to the *principle of indirect elimination* that states that it is easier to calculate the shape of a bad-converging modes than to reduce it directly. The strength of this principle is demonstrated on a simple example: The elimination of convergence problems in the presence of almost-zero-modes created by instanton configurations.

We always used the algorithm in two dimensions, usually with SU(2) gauge fields. The ISU algorithm performs extremely well for the case of the covariant Laplace equation with arbitrarily large disorder. For the Dirac equation, critical slowing down is eliminated in the continuum limit, but when β is kept fixed and the lattice size is increased the critical exponent is estimated to $z \approx 1.6$.

To understand the difference between the two methods, multigrid analyzing tools are developed, the most important of which are the *error-monitoring*, the study of the interpolation operators' *Rayleigh quotients*, and the so-called *κ-criterion*. These methods prove that the problem is due to the many eigenmodes of the Dirac operator that have low eigenvalues and can not be approximated well by the used localized interpolation operators.

That the Laplace equation is solved efficiently is caused by the phenomenon of *Localization*: The lowest modes of the Laplace operator are strongly localized when the disorder of the gauge field is large. We also present an explanation of this phenomenon. For the Dirac operator in two dimension, no localization is found; the reasons for this are not known. It is explained in detail why localization helps to improve the convergence of the algorithm.

After having analyzed the problems of the algorithm for the Dirac equation, we try to invent a cure. It consists of two parts: We change the shapes of the supports of the interpolation operators and we introduce the possibility of having more than one interpolation operator per block-lattice point. These improvements greatly accelerate the algorithm and reduce the critical exponent to $z \approx 0.45 \pm 0.1$ at very high disorder ($\beta = 1$). At $\beta \approx 4$ *critical slowing down completely vanishes*. Despite its complicated structure and the fact that no optimization of the program has been done, the CPU-time needed by the new algorithm is of the same order of magnitude (about six times larger) as that for a standard Conjugate Gradient.

We close the thesis with a look at other methods and some statements about the possible behavior of ISU in four dimensions.

ZUSAMMENFASSUNG

Eine der großen Herausforderungen der theoretischen Physik ist die Berechnung der Eigenschaften der Hadronen aus den fundamentalen Prinzipien, d.h. der Lagrange-Funktion der Quantenchromodynamik. Große Arbeitsgruppen versuchen dies unter Benutzung modernster Computer. Um die volle Theorie unter Einschluß dynamischer Fermionen zu simulieren, ist es notwendig, den Dirac-Operator wiederholt zu invertieren, also eine diskretisierte Differentialgleichung auf großen Gittern zu lösen. Aufgrund niedriger Eigenwerte hat der Dirac-Operator eine große Konditionszahl, was dazu führt, daß Standard-Lösungsverfahren wie "Conjugate Gradient" kritische Verlangsamung erleiden: Die Konvergenzrate nimmt proportional zur Ausdehnung des Gitters ab.

Für die Lösung einfacher Differentialgleichungen haben sich Mehrgittermethoden als sehr erfolgreich erwiesen. Die Invertierung des Dirac-Operators ist allerdings ein Problem, das aufgrund des Eichfeld-Hintergrundes Unordnung enthält. Deshalb ist es notwendig, Mehrgitterverfahren zu verwenden, die auch in ungeordneten Eichfeldern gut funktionieren.

In dieser Arbeit wird das Iteratively Smoothing Unigrid ("Iterativ Glättendes Eingitter", kurz ISU) vorgestellt und detailliert untersucht. Es beruht auf einer Definition des Begriffes "Glattheit", die auch in Gegenwart von Eichfeldern sinnvoll ist, und benützt einen iterativen Prozeß, um die Interpolationsoperatoren als Eigenvektoren zu den niedrigsten Eigenwerten auf Blöcken zunehmender Größe zu bestimmen. Im Unterschied zu anderen Methoden ist dieses Verfahren ein Eingitter (Unigrid): Das Feld wird nach jeder Berechnung auf groben Gittern direkt zum feinsten Gitter zurücktransportiert und auch das Blocken wird immer vom feinen Gitter aus durchgeführt. Der Vorteil einer solchen Methode liegt darin, daß sie eine größere Freiheit in der Wahl der Interpolationsoperatoren erlaubt.

Die Methode wird durch das *Prinzip der indirekten Elimination* ermöglicht, welches besagt, daß es einfacher ist, die Form einer schlecht-konvergierenden Mode zu berechnen als diese direkt zu eliminieren. Die Gültigkeit dieses Prinzip wird am Beispiel der Elimination von Beinahe-Nullmoden demonstriert, die in einem Eichfeld mit Instantonen auftreten.

Der Algorithmus wurde in zwei Dimensionen getestet, zumeist unter Verwendung von SU(2)-Eichfeldern. Für die kovariante Laplace-Gleichung konvergiert das Verfahren exzellent, unabhängig von der Größe der Unordnung. Im Falle der Dirac-Gleichung gibt es keine kritische Verlangsamung für physikalisch skalierte Eichfelder, also im Kontinuum-Limes, doch wenn der Parameter β festgehalten und das Gitter vergrößert wird, ergibt sich ein kritischer Exponent von $z \approx 1.6$.

Um diesen Unterschied zu verstehen, werden Analysemethoden entwickelt, von denen die wichtigsten die *Fehler-Analyse*, die Untersuchung der *Rayleigh-Quotienten* der Interpolationsoperatoren und das sogenannte *κ-Kriterium* sind. Diese Methoden zeigen, daß das Problem in der hohen Anzahl von Eigenmoden mit niedrigen Eigenwerten begründet liegt, die sich nicht durch lokalisierte Interpolationsoperatoren darstellen lassen.

Daß das Verfahren für die Laplace-Gleichung so effektiv ist, wird durch das Phänomen der *Lokalisation* verständlich: Die niedrigsten Eigenmoden des Laplace-Operators sind stark lokalisiert, wenn die Unordnung des Eichfeldes groß genug ist. Eine Erklärung des Phänomens wird präsentiert. Im Falle des zwei-dimensionalen Dirac-Operators hingegen gibt es keine Lokalisation; warum, ist nicht bekannt. Es wird ausführlich erläutert, warum der Algorithmus in Gegenwart von Lokalisation gut konvergiert.

Nachdem die Analyse der Probleme der Dirac-Gleichung abgeschlossen ist, wird ein verbesserter Algorithmus vorgeschlagen. Die Verbesserung besteht aus zwei Teilen: Zum einen wird die Form der Träger der Interpolationsoperatoren verändert, zum anderen wird die Möglichkeit eingeführt, mehr als einen Interpolationsoperator pro Blockgitterpunkt zu benutzen. Diese Verbesserungen führen zu einer starken Beschleunigung des Algorithmus mit einem kritischen Exponenten von $z \approx 0.45 \pm 0.1$ bei $\beta = 1$. Erhöht man β auf 4, so *verschwindet die kritische Verlangsamung vollständig.* Trotz der komplizierten Struktur des Algorithmus und der Tatsache, daß das benutzte Programm in keiner Weise optimiert wurde, ist die benötigte Rechenzeit von derselben Größenordnung (etwa sechsmal langsamer) wie die eines "Conjugate Gradient"-Algorithmus'.

Mit einem kurzen Blick auf andere Mehrgitterverfahren für Propagatoren und einigen Überlegungen zum Verhalten des Verfahrens in vier Dimensionen schließt die Arbeit.

CONTENTS

List of Figures

1 INTRODUCTION

Contemporary high-energy physics faces a unique situation: A vast amount of accurate data on the properties of elementary particles like mass spectra, decay constants etc. exists. On the other hand, theorists have developed the Standard Model which *seems* to be in agreement with the experiments. However, this agreement has only been tested for a few quantities, for which it was excellent, while many experimentally found numbers can not be computed from the theory [1].

This is indeed a curious situation: We have good experimental data and a seemingly good theory, but we can not compare them.

One of the reasons for this is the necessity to use non-perturbative methods. The method which *has been so incredibly accurate for calculating observable in Quantum Electrodynamics*, perturbation theory, fails with a vengeance for most QCD quantities. This is due to the large value of the strong coupling constant which would force us to take too many Feynman diagrams into account. In addition to this, there are phenomena like instantons that are inherently non-perturbative.

So although a promising theory for the behavior of the elementary particles has been known for twenty years, its comparison with experiment proves difficult.

What can be done? One major branch of high-energy physics tries to study the perturbative aspects of QCD in detail, *so that at least their implications for observable quantities can be checked*, thereby raising our confidence in the theory as a whole [1]. This is possible due to the small value of the coupling constant at short distances (asymptotic freedom) and is one of the motivations to build large colliders to probe the *inner structure of the hadrons*.

However, the non-perturbative aspects of QCD are too important to be neglected. Up to now, the only way of describing the theory non-perturbatively is formulating it as a Lattice Gauge Theory [2, 5, 6]. Here the theory is not regularized by subtracting divergences order by order in the Feynman diagrams, but by setting the whole theory on a (Euclidean) space-time lattice which prohibits arbitrarily large energies and momenta and hence provides us with an ultraviolet cutoff. In this formulation, the theory is equivalent to a system of Statistical Mechanics.

Unfortunately, nobody believes that space-time really is a simple regular lattice, as it is usually used in Lattice Gauge Theory. Formulating the theory in such a way will result in lattice artifacts, the simplest example of which is the loss of Lorentz invariance. The ingenious invention of the Renormalization Group [2] serves to remove such artifacts and allows us to take the continuum limit, i.e. the limit of vanishing lattice constant in which continuum physics hopefully is restored again.

Much analytical work has been done for this theory, e.g. [35], but complicated observables like the QCD mass spectrum evade analytical calculations. So, shortly after the invention of Lattice Gauge Theory, it has been proposed to replace the ingeniousness of man by the sheer "number-crunching" power of the computer. The simple idea is to make not only the lattice spacing, but also the volume of the used lattice finite, so that the path integral is an integral over a finite number of variables. Monte-Carlo evaluation techniques then allow to calculate typical configurations of the fundamental fields by importance sampling and so to compute observables for these configurations [21].

However, after the first enthusiasm it became obvious that *these studies are much more costly* than initially expected. By now, doing realistic, large-scale QCD simulations is still one of the Grand Challenge Problems of modern physics, involving large groups of physicists using massive parallel high-performance computers which sometimes are especially customized for the need of QCD simulations. Impressive results have been obtained this way, see [7, 8, 10, 11, 12, 13, 14] for overviews. But *still simulations on really large lattices (like 100^4 lattice points) at physically realistic values* of the fundamental parameters (like the quark mass) leading to high-precision predictions of observables that would enable us to answer the question whether QCD really is able to predict the correct mass *spectrum are missing.*

Of course there are other motivations to do first-principles calculations using Lattice Gauge Theory: Things like a lattice prediction for the strong coupling constant [24], the quark-gluon-plasma [17], glueballs [23], electroweak matrix elements [18], and the study of the thermodynamical properties of

QCD [19] and of the electroweak phase transition in the early universe [20] are realms where predictions from the lattice might have a great impact to physical progress.

Why are these calculations so demanding? The reason is a phenomenon called critical slowing down: As we increase the lattice size and decrease the masses of the quarks used in the simulation, we approach a critical point. In fact, we *must* approach a critical point of the theory, otherwise it would be impossible to take the continuum limit. Systems near critical points exhibit large-scale dynamics which act on a time-scale much larger than the time-scale of the fundamental local dynamics. This is *true for physical systems as well as for computer simulations which use local dynamics as well.* This leads to the fact that the computer time needed for the simulation grows much more rapidly than just with the volume of the system.

Some calculations can be done using the *quenched approximation*, i.e. ignoring the effects of the sea quarks. Results obtained in this way, especially for the meson masses, are in surprisingly good agreement with experiment, indicating that the effect of the virtual quarks might be smaller than expected [22]. But the final goal should still be a calculation using the full theory without approximations; only in this way can we expect to get numbers that are completely reliable.

The state-of-the-art algorithm used for calculating the full theory with the inclusion of sea quarks ("dynamical quarks") is the Hybrid Monte Carlo algorithm [25, 16]. This suffers from several sources of critical slowing down, so computations using this algorithm are the greatest challenge for computer simulations.

Within the Hybrid Monte Carlo algorithm, one step is extremely time-consuming: The solution of the Dirac equation. Although it is far from being the only source of critical slowing down, usually between 90 and 99 % of the computer time is used in this step. Therefore there is a large interest in *finding a Dirac solver that is faster than the commonly used Conjugate Gradient algorithm.*

The Dirac equation is a discretized partial differential equation, so all methods that are applicable to this problem type might be used for solving it. An especially swift way to solve such problems is the *Multigrid method* [58, 59, 60]. For simple differential equations like the Laplace equation, multigrid methods are the fastest solvers known. Not only that they do not show critical slowing down; the absolute value of the convergence rate also is impressive for such problems: In each multigrid iteration the error of the approximate solution is reduced by a factor of order 10.

So it is not surprising that it has been proposed to use multigrid methods for the solution of the Dirac equation [65]. [1] The Dirac equation on the lattice is an especially nasty kind of equation. Three of its properties conspire to prohibit the direct application of all the standard multigrid methods:

- It is disordered.
- It is vector-valued.
- It has many low-lying eigenvalues.

Methods are known to deal with simple disordered problems (the Algebraic Multigrid), but they usually fail when applied to vector equations. The large number of low-lying eigenvalues makes the solution especially difficult because it corresponds to a large number of large-scale eigenmodes that are difficult to handle for a normal multigrid.

Some attempts have been made to find new multigrid methods, applicable also for the Dirac operator. However, *none of these methods has yielded the results hoped for* [66, 68, 72, 74, 76, 77, 78, 79].

The problem of solving the Dirac equation therefore also is a challenge for the multigrid community: *Understanding why this is such a difficult problem to solve and how the difficulties might be overcome*

[1] Similar methods, called Multigrid Monte Carlo, have been studied which might be able to overcome the other sources *of critical slowing down in the simulations. This was indeed the first proposed application of multigrid to Lattice Gauge* Theory [62, 63, 64], but in this thesis we will not investigate it.

could also improve multigrid methods used in other fields dealing with disordered problems, as they occur in solid-state physics.

In addition to these two motivations, one can even think of a third, which is perhaps more philosophical: Multigrid methods work by identifying those modes of the system that are responsible for the slow, large-scale dynamics, which are in most cases the low-energy modes. A working multigrid method might provide us with a better understanding of the Dirac equation in addition to enabling us to solve it [91].

After this brief introduction and motivation, let us give a short

1.1 Overview Over Thesis

Chapter 2 We start by reminding the reader of the basics of Lattice Gauge Theory. After a short recapitulation of the principles we take a look at the Hybrid Monte Carlo algorithm to understand why we have to solve the Dirac equation in this algorithm.

Chapter 3 Then we study the phenomenon of critical slowing down in general, not only for the solution of differential equations, but also for Monte Carlo simulations. Strong emphasis is laid on the fact that critical slowing down is a physical phenomenon and not a computational artifact having nothing to do with the real world.

Chapter 4 After this general motivation we introduce the multigrid method. A brief look at the Algebraic Multigrid explains the need for a new method. We will then present the principle and the details of the Iteratively Smoothing Unigrid algorithm which is the main theme of this thesis. Two points are elaborated in detail: The crucial difference between a true multigrid and a unigrid method and the so-called "principle of indirect elimination". As the ISU algorithm in its standard form involves the solution of eigenvalue equations, we will also answer an objection made to such kind of algorithms.

Chapter 5 By then it is time to put the method to the test: We apply it to the Laplace equation in a gauge field background, where its performance is excellent, and to the Dirac equation, for which the method does not work as well as we hoped for. Finally, we will analyze an important step of the algorithm, the "updating on the last point", and use the results obtained there together with the principle of indirect elimination to show how one can deal with a system which has only a small number of bad-converging modes. Our example will be the Dirac operator at small disorder, but in the presence of instantons. This last method is very general; it could be used for other algorithms as well, not only for the ISU algorithm.

Chapter 6 Before continuing the main line of thought, we will make a short detour and look at the ISU algorithm in a very general light: We will show similarities with wavelet analysis and neural nets, take another look at the problem of critical slowing down and its implications for our algorithm, and finally show ISU's generality by formulating it in the language of Mack's Universal Dynamics.

Chapters 7–9 Having found in chapter 5 that ISU performs drastically different for two seemingly similar disordered problems, we will set out in chapter 7 and chapter 9 to develop analyzing tools to understand this difference. The tools developed here are quite general and neither restricted to the ISU algorithm nor to Lattice Gauge Theory. We will show the latter by briefly looking at the behavior of ISU for another problem, the random resistor network, and understanding why it does not perform well in this case. In between we will take another short side tour to study the phenomenon of localization in Lattice Gauge Theory. We will present an explanation of localization by connecting it to the Anderson-localization problem and discuss some implications for the convergence of our method.

Chapter 10 Were our efforts in vain? This is the question to be answered in the next chapter by trying to use the understanding we have gained to improve the algorithm. Two different improvements are presented. Combined they lead to a version of ISU which shows strongly reduced critical slowing down for the Dirac operator at extremely high disorder with a critical exponent of $z \approx 0.45 \pm 0.1$; *critical slowing down completely vanishes when the disorder parameter is* $\beta \approx 4$. However, the method is quite costly. Comparison with the standard method of Conjugate Gradient is difficult, but we present at least some estimates, showing that ISU is about six times slower than Conjugate Gradient with the version used at present. We also present a generalization of ISU working with dynamical block centers.

Chapter 11 Finally, we present other multigrid approaches to the problem and show why the ISU algorithm is different from these in several respects.

Chapter 12 concludes the thesis with a summary and outlook.

Parts of this thesis have already been published in [80, 81, 82, 83, 84, 85, 90].

2 A Glimpse of Lattice Gauge Theory

Wherein the reader is reminded of the basics of Lattice Gauge Theory and gets an idea why this thesis was written.

2.1 Mathematical Structure of Lattice Gauge Theory

Consider a regular, d-dimensional (hyper-)cubic lattice Λ^0 with lattice constant a, lattice points z and directed links (z, μ). The opposite link is then denoted by $(z+\mu, -\mu)$, where $z+\mu$ means the next neighbour of z in μ-direction. The direction index μ runs from $-d$ to d. Usually, the lattices used will be finite with an extension of L points in each dimension so that the number of degrees of freedom is $n = L^d$.

A lattice gauge theory is defined by a gauge group G which might for example be U(1) or SU(2). Elements of the gauge group act on a vector space V which for the examples above would be $\mathbb{C}$ and $\mathbb{C}^2$, respectively. Unless otherwise stated, we always use the gauge group $SU(2)$ throughout this thesis and work in two dimensions.

We define a (bosonic) *matter field* ξ as a map of Λ^0 to the vector space V [2]

$$\xi : \Lambda^0 \mapsto V \tag{1}$$

$$z \mapsto \xi_z \quad , \tag{2}$$

and a gauge field U by mapping each link to a gauge group element

$$U : (\Lambda^0 \times \{-\mu, \ldots, \mu\}) \mapsto G \tag{3}$$

$$(z, \mu) \mapsto U_{z,\mu} \tag{4}$$

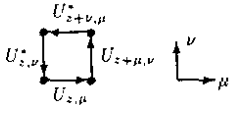
subject to the condition $U_{z,\mu} = U^*_{z+\mu,-\mu}$.

Both the matter field and the gauge field are distributed randomly with the Boltzmann distribution $\exp(-S[U, \xi])$. Here S is the (Euclidean) action of the theory.

The action can be split into two parts: The gauge field part of the action S_W is usually chosen as the Wilson action [2]

$$S_W = \frac{\beta}{4} \sum_P \mathrm{Re}\ tr(1 - U_P) \quad . \tag{5}$$

Here $\beta = 4/g^2$ is the inverse coupling and the sum is over all plaquettes in the lattice. U_P denotes the parallel transport around the plaquette



$$U_P = U^*_{z,\nu} U^*_{z+\nu,\mu} U_{z+\mu,\nu} U_{z,\mu}$$

$$\mu \neq \nu$$

This distribution leads to a correlation between the gauge field matrices with finite correlation length χ for finite β. The case $\beta = 0$ corresponds to a completely random choice of the matrices ($\chi = 0$), *for $\beta = \infty$ all matrices are 1 ($\chi = \infty$). In this sense, β is a disorder parameter, the smaller β* the shorter the correlation length and the larger the disorder. In Lattice Gauge Theory one is interested in the continuum limit, i.e. the limit of $\beta \to \infty$ and $L \to \infty$ (L is the grid length) for a *constant scale parameter $\eta = \sqrt{\beta}/L$. (This is the behavior of the scale parameter in two dimensions* which is the case of most interest in this thesis.)

[2] Fermions should be described as Graßmann variables. However, in the following we will see how to change from this point of view to a *bosonic* description of the fermions, using pseudo-fermion variables.

The action S_m of the matter field is a bilinear:

$$S_m = \langle \xi, \mathbf{D}\xi \rangle \quad . \tag{6}$$

$\mathbf{D}$ is the propagator of the matter field which contains also the coupling to the gauge field.

For a bosonic theory this is the Laplace operator

$$(\Delta\xi)_z = \frac{1}{a^2} \sum_{\mu \neq 0} \left(U^*_{z,\mu} \xi_{z+\mu} - \xi_z \right) \quad . \tag{7}$$

Its form is most easily understood in the so-called stencil notation which directly shows the connections of one point (in the center) to its neighbours [60]

$$\Delta_z = \begin{bmatrix} 0 & U_{z,2} & 0 \\ U_{z,-1} & -4 & U_{z,1} \\ 0 & U_{z,-2} & 0 \end{bmatrix} \quad . \tag{8}$$

It can be seen how the gauge field enters as a parallel transporter to allow forming the differences between matter fields on different points.

For fermions, we have the Dirac operator $\not{\mathbf{D}}$ instead of the Laplacian. We will always use the staggered or Kogut-Susskind discretization [30]

$$(\not{\mathbf{D}}\xi)_z = \frac{1}{a} \sum_{\mu=1}^{d} \eta_{\mu,z} \left(U^*_{z,\mu} \xi_{z+\mu} - U^*_{z,-\mu} \xi_{z-\mu} \right) \quad . \tag{9}$$

Here the $\eta_{\mu,z} = \pm 1$ are the remnants of the Dirac matrices γ^μ in the continuum.

Kogut-Susskind fermions describe $2^{\lfloor d/2 \rfloor}$ fermions and thereby weaken the famous doubler problem [34]. To achieve this, the fermionic spinor is split and distributed over several grid points. In the language of solid state physics we are using a lattice with a basis which is a hypercube with 2^d points. Each of these points belongs to a different *pseudoflavor*. For this thesis, only the symmetry properties of the staggered grid are of interest; a picture of such a grid can be found on page 39, figure 8. For further details, the reader is referred to the bibliography [30, 68].

Throughout this work we will use the squared Dirac operator $\not{\mathbf{D}}^2$ which has the advantage of being positive definite (and the disadvantage of a larger condition number which is the quotient of highest and lowest eigenvalue). The squared Dirac has the property of totally decoupling the even and odd parts of the lattice; if we color the lattice points in checkerboard fashion, any red point is only coupled to other red points, so that we can restrict our attention to one of the sub-lattices. This will be especially useful because it lifts the degeneracy of the eigenvalues: Usually each eigenvalue of the Dirac operator is degenerated twice (plus additional degeneracies stemming from the gauge group), but we can choose the eigenvectors to live separated on the sub-lattices. For the sake of brevity we will generally speak of the Dirac operator even when we mean the squared Dirac.

Another simplification for gauge group SU(2) can be made by regarding the Dirac operator as being defined not on the field of complex numbers, but on the field of multiples of SU(2)-matrices, so that expansion coefficients etc. belong to this space. This has the advantage that we can always speak of *n degrees of freedom regardless of the gauge group, even when these are actually $2n$ complex numbers.* This makes comparison between U(1) and SU(2) easier and lifts the degeneracy of the eigenmodes of the Dirac operator due to possible rotations in the SU(2) space. Note that this simplification *is possible for* SU(2), *but not for arbitrary gauge groups, and that one has to be careful always to* multiply such matrix-coefficients from the right [40].

After introducing the mathematical framework of the problem, let us now look at the simulation procedure: The goal of any simulation algorithm is to calculate an ensemble of fields with a probability *distribution given by the action in as small an amount of time as possible. This is done by a* Marcov-chain process: One starts with an initial configuration and generates a new configuration from this.

In this way one gets an evolving sequence of configurations. The configurations created in this way are not independent; it takes a certain time, the decorrelation time, to arrive at a truly independent configuration. This is one of the points where critical slowing down can enter into a simulation algorithm.

A crude approximation, the so-called *quenched approximation*, can be made by simply ignoring the matter field part of the action. This means that only the gauge field part of the action is taken into account. Despite the fact that a Monte-Carlo-algorithm for this also suffers from critical slowing down[3], it is much easier to do than the full simulation.

Physically, this approximation means that no sea quarks are taken into account. Although many quantities can be measured in this approximation with surprisingly good agreement to the experimental data, simulations of the full theory are to be preferred. Unfortunately, such simulations are difficult, as we will shortly explain in the next section.

In this thesis, the configurations will usually be calculated using the quenched approximation by a simple heat-bath algorithm [15, 26]. Only in some cases did we use the full theory in a U(1)-gauge field background [29].

2.2 The Hybrid Monte Carlo Algorithm

The standard algorithm for calculating configurations distributed with the correct action of the theory is the Hybrid Monte Carlo (HMC) algorithm. In order to understand how the necessity of solving the Dirac equation arises, let us briefly review this method. A good introduction into the algorithm can be found in [25, 9].

The first step is to get rid of the fermionic Graßmann variables ψ in the path integral which are not computer-friendly. Fortunately, they can be integrated out exactly, yielding the determinant of the fermionic operator which can then be reinserted into the action, but now with *bosonic* (usually called pseudo-fermionic) degrees of freedom:

$$\int \mathcal{D}U\,\mathcal{D}\bar{\psi}\,\mathcal{D}\psi\,\exp\left(-S_W[U]-\bar{\psi}\,\mathbb{D}[U]\psi\right) = \int \mathcal{D}U\,\det\mathbb{D}[U]\,\exp\left(-S_W[U]\right) \tag{10}$$

$$= \int \mathcal{D}U\,\mathcal{D}\bar{\phi}\,\mathcal{D}\phi\,\exp\left(-S_W[U]-\bar{\phi}\,\mathbb{D}[U]^{-1}\phi\right) \quad . \tag{11}$$

In this formula we can already see the crux of the simulation: In trading fermionic variables for pseudo-fermionic, we had to introduce the inverse of the fermion matrix, so that the resulting action is non-local.

To generate configurations according to this action, we complicate the problem further by introducing *canonical momenta* $P_{x,\mu}$ for each gauge field variable $U_{x,\mu}$. The new Euclidean action (or Hamiltonian) becomes

$$H = \frac{1}{2}\sum \mathrm{tr}P_{x,\mu}^2 + S_W[U] + \bar{\phi}\,\mathbb{D}[U]^{-1}\phi \quad . \tag{12}$$

One can easily see that the presence of the canonical momenta in the path integral will not affect the expectation values of any observable $O(U,\bar{\phi},\phi)$.

The pseudo-fermion variables can be updated with a simple heat-bath algorithm. Simultaneously, the canonical momenta of the gauge field are changed, again using a heat-bath algorithm. This step only involves applying the fermion matrix to a vector of random numbers, no inversion is needed here.

This is different for the gauge field updates. They are done next with a molecular dynamics algorithm. This means that we hold the energy fixed and let the gauge field variables and their canonical momenta evolve according to the usual Hamiltonian equations of motion. The pseudo-fermionic variables are regarded as external parameters not to be changed during these steps.

[3] We have not yet explained what critical slowing down is. For the moment it suffices to know that critical slowing down means that the computer time required for a calculation grows faster than the number of degrees of freedom involved. In the next chapter this will be described in detail.

The integration of the equations of motion has to be done numerically, using a small time step size. It is done in a *leapfrog* method, i.e. on the even time steps we calculate the field variables and on the odd time steps the canonical momenta. The updating of the momenta, however, is not simple because from the Hamiltonian equations of motions it follows that

$$\frac{dP}{dt} = -\frac{\partial H}{\partial U} \tag{13}$$

$$= -\frac{\partial S_W[U]}{\partial U} - \bar{\phi}\,\frac{\partial\,\mathbb{D}[U]^{-1}}{\partial U}\,\phi \tag{14}$$

$$= -\frac{\partial S_W[U]}{\partial U} - \bar{\phi}\,\mathbb{D}[U]^{-1}\frac{\partial\,\mathbb{D}[U]}{\partial U}\,\mathbb{D}[U]^{-1}\phi \quad . \tag{15}$$

Here, finally, we see the need for solving the Dirac equation $\mathbb{D}[U]^{-1}\phi$.

The last step of the Hybrid Monte Carlo algorithm is a global Metropolis step. After generating the new configuration we fix it with the usual Metropolis acceptance probability of $\min(1, e^{-\Delta H})$. To calculate the energy change, we have to solve the Dirac equation again.

Hence we can understand that the solution of the Dirac equation is the bottleneck in the Hybrid Monte Carlo algorithm using at least 90% of the computer time of the simulations. This number might grow even larger when the quark masses are chosen to be small because the condition number of the Dirac operator will be increased.

However, the HMC algorithm suffers from other sources of critical slowing down as well. As we saw above, we try to follow the trajectory of the gauge field system in phase space by integrating the equations of motion. If we could do this exactly, the energy of the system would not change during this molecular dynamics step. However, the numerical integration will introduce errors, changing the energy of the system. The algorithm is made exact again through the Metropolis step, but this requires good acceptance rates, otherwise most of the computer time will be wasted by calculating configurations which are finally rejected. To achieve good acceptance rates (of about 60%, say), one has to make the time steps small enough. In fact, the largest contribution to critical slowing down comes from the finite step size of the molecular dynamics. The step size δt has to be scaled as $\delta t \propto L^{-1}$ for fixed values of β and the quark mass m_q, and as $\delta t \propto m_q$ when the quark mass is changed.

In addition, the decorrelation time for the trajectories also grows with decreasing quark masses $t_{decorr} \propto m_q^{-1/2}$. Finally, as we already know, the solution of the Dirac equation with standard algorithms suffers from critical slowing down, determined by the condition number $\kappa \propto m_q$.

Let us finally summarize all sources of critical slowing down in units of the lattice pion mass which is $m_\pi \propto m_q^{1/2}$, also using the relation $L^{-1} \propto m_\pi$. With this we get a behavior of the computer time τ of

$$\tau \propto m_\pi^{-\left(\underbrace{4}_{\text{volume}} + \underbrace{1+2}_{\text{step size}} + \underbrace{1}_{\text{trajectory decorr.}} + \underbrace{2}_{\text{Dirac inversion}}\right)} = m_\pi^{-10} \quad . \tag{16}$$

We see that even a perfect Dirac solver does not cure the problems of the algorithm. The reason why so much effort is put into the writing of a better Dirac solver is simply that not many ideas to overcome the other sources of critical slowing down are around, whereas we know from the solution of other differential equations that usually an algorithm without critical slowing down exists.

So we see that it is a worthwhile task to write a good Dirac solver, which may enable us to do QCD simulations on larger lattices with smaller values of the quark mass and therefore closer to realistic parameter regions.

3 What is Critical Slowing Down?

Wherein we encounter the nemesis of all simulation algorithms and see why it is not just a numerical inconvenience, but a phenomenon with physical origin.

The solution of discretized differential equations or the Monte-Carlo simulation of physical theories is often affected by critical slowing down. In this section, we will explain the meaning of critical slowing down, show how it may arise and be overcome. As the phenomenon is very similar in the cases of solving differential equations and Monte-Carlo simulations, we will explain the phenomenon in general terms. Some scenarios only occur in the latter case which will briefly described for completeness, although they are irrelevant for the main theme. This section is partly based on a paper by Solomon [91].

Before exploring this phenomenon in its generality we would like to explain the appearance of critical slowing down in a simple example, namely the Monte-Carlo simulation of the Ising-model.

Imagine we want to do a Monte-Carlo simulation for the Ising-model at some temperature well below the critical point, that is, in the ferromagnetic region. Our algorithm to do this is the very simple local Metropolis update: We walk through the lattice and propose a spin flip at each point which is then accepted with a probability $\min(1, e^{-\beta\Delta H})$, where ΔH is the change in the energy due to the proposed flip and $\beta = (kT)^{-1}$ the inverse temperature. We always accept the flip when the energy is lowered, but when it is raised the acceptance probability is governed by the temperature T. If T is small, such flips which raise the energy are accepted seldom. Imagine a large island with aligned spins (what a solid states physicist would call a domain) in the up direction. Our Monte Carlo process, which should explore the whole configuration space of the model, should generate from this in a reasonable amount of time a configuration where the large island has aligned spins in the down direction because this configuration is equally likely. If it is not able to do this, measurements of the magnetization for example would give completely wrong results. Unfortunately the local Metropolis algorithm is of such a type: It is very unlikely that a spin in the middle of the island will be flipped *because it is surrounded by up-spins. Therefore the only possibility to change the island* to an island of down spins is to do changes at the boundary, but it will take a long time until these changes can affect the whole island. At very low temperatures the situation may become even worse as nearly all spins in the system are aligned, so the boundary becomes arbitrarily small.

In other words, the island is quite stable under the action of a local algorithm. This is exactly what is meant by critical slowing down: The algorithm will take a long time to generate a new, statistically *independent configuration, and upon enlargening the lattice this time will grow faster than the volume* of the lattice. Not only that one sweep through the lattice is more costly since more points are to be covered, it is also less efficient in generating an independent configuration because the larger the domain, the smaller its volume-surface ratio and the longer the time for changing the domain.

Having understood the problem, it is not too difficult to find a cure, at least in hindsight: The problem is due to large islands (or clusters) of aligned spins that can not be flipped step by step. The *remedy is to search for such clusters of aligned spins and flip all the spins in the cluster at once* [27]. This step has a small energy change per spin, all energy costs arise only at the surface. The new algorithm (called a cluster algorithm) is capable of eliminating critical slowing down completely so that the computer time grows proportionally to the volume of the lattice.

The general wisdom to be extracted from this example may be stated as follows: critical slowing down arises when there are large, collective modes in the system associated with a small change in the *energy. In the Ising case this was the flip of a whole domain, that has negligible energy costs per spin.* We will come back to this general question in section 6.3 where we take another look at the problem of critical slowing down.

In the following we will explain the various physical mechanisms leading to such collective modes having a small energy.

To do this, we will describe the physical model by an energy landscape: With a Monte Carlo *simulation we want to generate configurations of the system according to the Boltzmann distribu-*

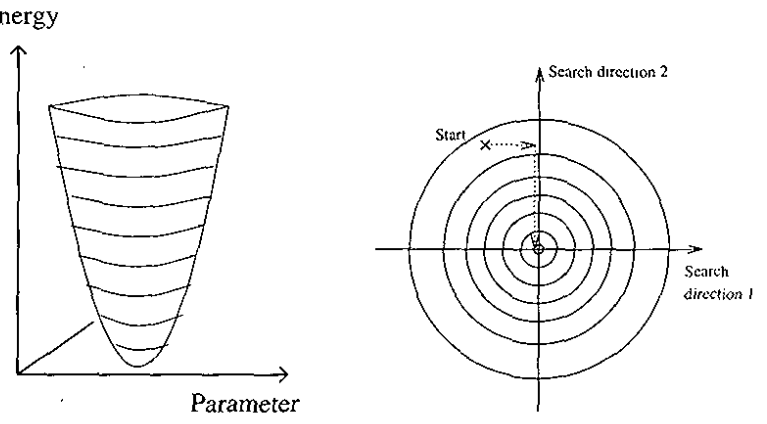


Figure 1: Left: A parabolic energy landscape in a 3D-plot. Right: The contour lines of the same landscape. Two steps in orthogonal search directions will lead to the minimum of the landscape.

tion $e^{-\beta H}$. Here H is the energy functional, depending on all the microscopic degrees of freedom. (In a quantum field theoretic context, H is the analytically continued action.) Obviously the most probable configuration of the system is the ground state, and it is the more probable, the smaller the temperature of the system gets. The solution of differential equations can also be mapped on such a problem at temperature zero: Solving a differential equation is equivalent to minimizing a certain "energy" functional. Therefore the algorithm has to be able to find the minimum of the energy landscape in a reasonable amount of time. For finite temperature, it must explore all regions with low energy effectively.

Practically all algorithm used for this kind of problem are directional methods: They walk through the energy landscape in certain search directions and a proposed step is accepted with a certain probability (Monte Carlo method) or the minimum of the landscape along the search direction is sought. In the case of over-relaxation methods, one steps over the minimum to the other side of the "mountain", see below. The most important ingredient is the determination of good search directions. In local methods, such as local Metropolis or Gauß-Seidel relaxation, these directions are the axes of the parameter space. Only the value at one point is changed at a time, then the algorithm proceeds to the next point (search direction).

Let us now look at different possibilities for the shape of energy landscapes. The most favorable landscape is shown in figure 1 for a two-dimensional parameter space. The contour lines of equal height are circles. Mathematically, this means that the eigenvalues of the energy operator are completely degenerate. In such a case, finding the minimum is easy: any complete set of search directions will find the minimum doing one step in each direction.

However, usually not all energy eigenvalues of the systems are the same—we have low-energy modes and high-energy modes. Starting from the favorable picture, we now want to study how it may be changed to generate critical slowing down. To do this, let us choose one eigenvalue much smaller than the other in our two-dimensional picture. The bowl then becomes a long, narrow valley, see *figure 2. The direction of the valley is the direction of the lowest eigenvector of the energy.*

If the valley is aligned with one of the search directions, we still do not have any difficulties and will reach the minimum in a few steps, but usually it will lie in some other direction. In this case we

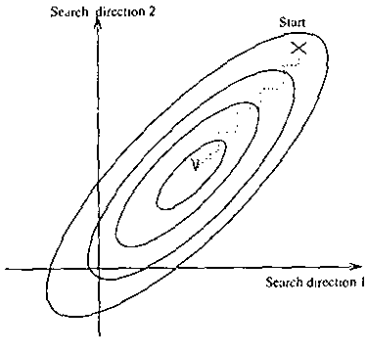


Figure 2: Energy landscape for the long, narrow valley. One of the eigenvalues is much smaller than the other. The direction of the eigenvalues does not coincide with one of the search directions. Many search steps are required to find the minimum.

have a problem: Going along the search directions to the minimum on the search line will lead us in a zig-zag line along the valley; each step is very small, otherwise it would lead us too far up the slope of the other side. (This is the reason, why over-relaxation may be a good idea: The over-relaxation algorithm leads you not to the minimum along the search line, but it lets you walk up the other slope by a certain amount, despite the fact that this may raise the energy. In this way, you can do larger steps through the valley and reach the minimum in fewer steps.) So it is easily understood that the narrower and deeper the valley, the slower the convergence of an algorithm that does not proceed in the right direction. The narrowness of the valley is directly measured by the quotient of the lowest to the highest eigenvalue: The highest eigenvector points to the direction of the steepest slope, the lowest to the direction of smallest slope. Going one step of fixed length in one or the other direction will raise (or lower) the energy by an amount proportional to the eigenvalue and the length of the step. Dividing both numbers therefore measures how narrow the valley is. That the step size along the valley becomes smaller the narrower it is is the reason why the convergence rate of many algorithms (Conjugate Gradient, Gauß-Seidel, etc.) is governed by the quotient of the largest and the smallest eigenvalue, the so-called condition number.

This picture also shows that critical slowing down may occur even when the number of degrees of freedom is as small as two. An example for this (the Laplace equation with periodic boundary conditions on a two-point grid) can be easily constructed. Of course, in this case analytical methods are able to eliminate critical slowing down.

So a good algorithm should proceed along the valley, but this step usually is not a local step. Why is this? The simple answer is that most models we are interested in stem from physical ideas. A physical energy functional usually contains a kinetic energy term. The kinetic energy contains a derivative, i.e. it measures how strongly the configuration fluctuates locally. Hence a low-energy configuration has to possess only small fluctuations and so it must extend over a larger domain. It is indeed often the case (at least for ordered problems) that the lowest mode is a constant and has zero kinetic energy. For the Ising-model we started with the situation is similar, although there is no kinetic energy term in the model: with periodic boundary conditions the lowest energy mode is the one where all spins are aligned.

From this, we also see why critical slowing down is a *physical phenomenon*: Usually, the large-scale, low-energy modes are not only slow under the artificial dynamics of a computer algorithm, but also

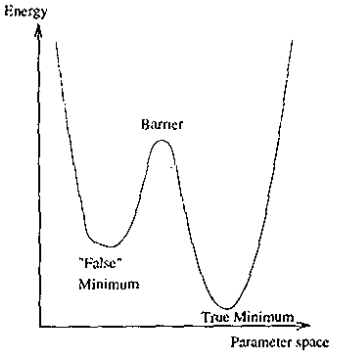


Figure 3: Energy landscape with meta-stable state. There are two local minima, one with a higher value of the energy than the other. Whenever the algorithm is trapped in the "false" minimum, it has to overcome the barrier to get into the global minimum.

under the real-world dynamics. An example for this is a diffusion problem: Local fluctuations in the concentration are straightened out quickly, but a small concentration gradient takes a long time to vanish because the physical dynamics is local. Global moves ("stirring") may serve to accelerate the slow part of the dynamics. In fact, the standard relaxation algorithms to be described in section 4.1.1 can be shown to be mathematically equivalent to a diffusion process.

We want to stress here that what is local depends on the algorithm [67] which determines the search directions: Imagine we have diagonalized the matrix. In this case, local would mean to do an update on each eigenmode separately. As in this basis all modes are completely decoupled, there will be no critical slowing down at all. In the case with the long narrow valley this means that our search proceeds exactly in the direction of the valley, so the shape of the valley does not matter. In fact, there are cases for which this can be done, e.g. the Laplace equation which can be diagonalized using a Fourier transformation.

The next class of problems can be created by completely leveling the valley so that the energy of the associated mode is exactly zero. The ground state is then infinitely degenerated and the system will choose one of them arbitrarily. An example is the famous scenario of spontaneous symmetry breaking. The degenerated zero-mode is called a Goldstone mode of the system. The energy landscape most familiar to a particle physicist is the Mexican hat shape encountered in the Higgs-Model. Here the valley is bent around so that it gets a circular shape. Usually, no local move will be able to take us around the circular valley (except for an infinitely small step at the points where one of the axes is cut by the valley.).

Another case with an exact zero mode is the Dirac equation in the continuum in the presence of instantons. The Atiyah-Singer theorem states that the number of zero-modes in the system is equal to the topological charge [32]. As to any solution of the Dirac-equation an arbitrary multiple of such a zero mode may be added, there is an infinitude of solutions. When the algorithm has found one, it may proceed along the valley and produce new solutions in each step with increasing modulus. In this case it seems as if the algorithm is diverging, not converging. On a lattice, these zero-modes become approximate zero-modes which may spoil the convergence of a solution algorithm. The interesting story of how such a problem was identified can be found in [75]. We will come back to this again in section 5.4.

We see that these two critical slowing down mechanisms are very similar, the Goldstone case is

just the limiting case of the completely flat valley. This is similar for the other two cases, where one can also be created by a special choice of parameters from the other model. These two cases only arise in non-linear systems, not in solving discretized differential equations. Nevertheless, we want to discuss them shortly.

Here the problems are not due to one valley with an inconvenient shape, but to the existence of two valleys, see figure 3. First we consider the case when the minima do not have the same energy. Let us assume that each of the valleys itself has a nearly circular shape, so we can easily find its bottom. Starting at the wrong point and following the slope we may easily become trapped in the higher of the two minima. Getting out may be very hard since usually no move in one of the search directions will lead us out of the minimum. We have found a minimum, but not the true one. This means that we have a different temperature locally than globally: Locally, within the false minimum, everything seems equilibrated, but globally we can see that something is wrong. It is very difficult for the Monte Carlo algorithm to reach the true minimum because climbing over the barrier is very unlikely whenever the barrier height is large compared to the thermal energy β^{-1}.

This scenario is very familiar when a first-order phase transition occurs. Here super-cooling may occur, i.e. pressure and temperature are too low to allow the system to stay in the fluid phase, but the phase is meta-stable and the system does not crystallize. A global change of the system (shaking its container or throwing a sufficiently large seed of condensation in) will lead it over the mountain and into the true minimum. Again we see that critical slowing down is a phenomenon with physical origin.

The last case we want to consider is when the two minima have exactly the same energy. At finite temperature, we are simulating only at one of the minima, the probability of climbing over the mountain being of the order $\exp(-\beta \cdot \text{barrier height})$. This was the scenario we found above for the Ising-Model, when the temperature is low enough: whenever all spins are aligned, it is practically impossible to flip them all and find the other minimum.

We have seen four mechanisms for generating critical slowing down that may be divided into two classes: low-eigenvalue critical slowing down and meta-stability critical slowing down. In practice both phenomena may occur simultaneously: even in a two-dimensional parameter space there is a vast possibility for generating complicated energy landscapes, as each map of a mountain range shows. In higher dimensions the situation becomes even worse.

It is very important to notice that critical slowing down does not exist only at a critical point of the theory. This can be easily seen for the Goldstone mode scenario: The critical point is the point where the Goldstone mode and with it critical slowing down comes into existence, and it will persist for all parameter values below the critical point. For the meta-stable case there is no critical point in the usual sense as it is associated with a first-order phase transition; nevertheless the two minima will coexist for a finite parameter range, not only at the phase transition point itself which is characterized by the equality of the energy minima.

Finally, let us look at the strength of critical slowing down, i.e. at the growth of the computer time, see also section 6.3. Naively, one would expect the time τ needed for generating a new independent configuration to grow proportional to the volume of the grid, i.e. $\tau \propto L^d$, where L is the linear extension of the grid and d the dimension. (For the solution of a differential equation, τ is defined as the number of iterations needed asymptotically to reduce the error by a constant factor, usually chosen to be e.) Instead of this, a behavior like $L^d \kappa^{z/2}$ is usually encountered in the low-eigenvalue scenario[4]. κ is the condition number of the algorithm, i.e. the quotient of the largest and the smallest eigenvalue. The quantity z is called the critical exponent and many algorithms show $z \approx 2$. For the meta-stability case, the computer time may even grow as $L^d e^B$, where B measures the barrier height.

[4]The factor $1/2$ in the exponent is there for the following reason: In many cases, the highest energy eigenvalue is practically constant with varying grid size L, but the lowest eigenvalue is inversely proportional to L^2. This is the case for the classical model problem, the Laplace equation with Dirichlet boundary conditions. The computer time then grows as L^{d+z} without bothersome factors.

In the first case, we speak of polynomial, in the second case of exponential critical slowing down. If both exist simultaneously, the exponential behavior will completely blur the polynomial.

4 Explaining ISU

Wherein the fundamental ideas of multigrid are presented, their limitations explained and the ISU algorithm, the hero of this thesis, sees the light of the day.

4.1 Introduction to Multigrid

Before we describe the new algorithm that is the main theme of this thesis, let us briefly review standard multigrid methods and a special multigrid method which is able do deal with some disordered models, the Algebraic Multigrid or AMG.

Let the equation to solve live on a lattice Λ^0 with lattice constant a_0. We can write the equation as

$$\mathbf{D}\boldsymbol{\xi} = \mathbf{f} \quad . \tag{17}$$

Here $\boldsymbol{\xi}$ and $\mathbf{f}$ denote functions on the grid, like the matter fields introduced earlier, and the operator $\mathbf{D}$ might be the Laplace or squared Dirac operator, but the method we are going to describe is general enough to deal with any positive definite operator.

However, as we are going to describe standard methods first, for the time being we should think of a simple, ordered equation like the scalar Laplace equation.

4.1.1 Relaxation Methods

A simple method to deal with such an equation is *relaxation* [54, 55]. Here one tries to solve the equation by sweeping through the grid in some sequence (lexicographic or checkerboard sequence for instance) and solves the equation at each point. If we call the approximate solution we might already know (or the initial guess) $\tilde{\xi}$, then this method amounts to

$$\tilde{\xi}_x^{\text{new}} = \frac{1}{D_{xx}} \left(f_x - \sum_{x' \neq x} D_{xx'} \tilde{\xi}_{x'}^{\text{old}} \right) \quad . \tag{18}$$

If we always use the old value on the righthandside of the equation, the method is independent of the sequence in which we sweep through the lattice which is helpful for parallelization or vectorization. This method is called Jacobi-relaxation.

On principal grounds, however, it seems better to use the knowledge one already has gained and therefore to use the new values whenever available. Then we speak of Gauß-Seidel relaxation, and as the method now depends on the sequence we have to discriminate between lexicographic, checkerboard and other Gauß-Seidel variants.

In both cases, it is also possible to *damp* the relaxation, i.e. to choose the new value at a point as a mixture of the old value at this point and the value suggested by the above equation.

We can also write these methods in a matrix-form. To do so, let us split the matrix $\mathbf{D}$ into its diagonal ($\mathbf{E}$), upper diagonal ($\mathbf{U}$), and lower diagonal ($\mathbf{L}$) parts: $\mathbf{D} = \mathbf{E} - \mathbf{U} - \mathbf{L}$. We then get for the lexicographic Gauß-Seidel method

$$\tilde{\boldsymbol{\xi}}^{\text{new}} = (\mathbf{E} - \mathbf{L})^{-1}\mathbf{U}\tilde{\boldsymbol{\xi}}^{\text{old}} + (\mathbf{E} - \mathbf{L})^{-1}\mathbf{f} \quad . \tag{19}$$

The matrix equation for Jacobi-iteration is similar, but for the checkerboard Gauß-Seidel method it is not so simple.

To study convergence, let us rewrite this in a general form:

$$\tilde{\boldsymbol{\xi}}^{\text{new}} = \mathbf{M}\tilde{\boldsymbol{\xi}}^{\text{old}} + \mathbf{N}\mathbf{f} \quad . \tag{20}$$

Here $\mathbf{M}$ is called the *iteration matrix*. The matrix $\mathbf{N}$ plays only a minor roll because it only acts on $\mathbf{f}$, so this part of the equation can be calculated once and for all. The iteration matrix can also be written in the form $\mathbf{M} = \mathbf{1} - \mathbf{B}\mathbf{D}$ with $\mathbf{B}$ depending on the relaxation scheme. (For Jacobi relaxation, it is simply $\mathbf{E}^{-1}$.)

We introduce two important quantities: the *error* $\mathbf{e} = \boldsymbol{\xi} - \tilde{\boldsymbol{\xi}}$ which is the difference between the true and the actual solution and is of course not known, and the *residual* $\mathbf{r} = \mathbf{f} - \mathbf{D}\tilde{\boldsymbol{\xi}}$, the difference between the true and the actual righthandside. With these definitions we can recast the fundamental equation (17) as

$$\mathbf{D}\mathbf{e} = \mathbf{r} \quad , \tag{21}$$

called the error equation. The iteration matrix then only acts on the error:

$$\mathbf{e}^{\text{new}} = \mathbf{M}\mathbf{e}^{\text{old}} \quad . \tag{22}$$

Therefore it is easily understood that the convergence rate of the method is governed by the spectral radius of $\mathbf{M}$ and that the method will converge if it is smaller than one.

For the shown relaxation methods, the spectral radius will indeed be smaller than one. But having in mind the fact that the algorithms are *local*, we should expect the methods to show critical slowing down, as explained in chapter 3. This is in fact the case.

To measure the convergence, we define the *inverse asymptotic convergence rate* τ, already mentioned in chapter 3, as

$$\tau = -\frac{1}{\ln \varrho(\mathbf{M})} \quad , \tag{23}$$

where $\varrho(\mathbf{M})$ is the spectral radius of the iteration matrix. This agrees with the definition given above because the spectral radius is the reduction factor for the worst-converging mode. τ measures the number of iterations needed to reduce the error of the worst-converging mode (which will be the only one alive asymptotically) by a factor of e. For shortness, we will often use the term *convergence time* instead of *inverse asymptotic convergence rate*.

4.1.2 The Multigrid Method

We now expect that the slow-converging modes will be the large-scale modes in the system which in their turn will usually be the eigenmodes of the operator with the lowest eigenvalues. (We will come back to the question whether it is meaningful to speak of the eigenvectors of $\mathbf{D}$ in a later section). Therefore we expect that the iteration matrix effectively damps the strongly oscillating modes while leaving the modes with the larger wavelengths nearly unaffected. That this is the case can be checked by performing a mode analysis. For the case of the Laplace equation, such an analysis can be found in most standard multigrid textbooks, as [58, 59, 60].

The key observation leading to the multigrid method consists of two parts:

- Relaxation methods serve to smoothen the error on the scale of the lattice constant.
- A mode that is smooth on a certain scale can be approximated well by interpolation from a function living on a lattice with this scale as lattice constant.

To apply this observation, we introduce *auxiliary lattices*, also called *block lattices* or *coarse grids*, $\Lambda^1, \Lambda^2, \ldots, \Lambda^N$ with lattice spacings $a_j = L_b^j a_0$, where L_b is the blocking factor and is usually chosen to be 2. Here we assume that the auxiliary layers are also simple cubic lattices, but this is not necessary and the method easily generalizes to more complicated block-lattices, as we will see later. The last lattice Λ^N consists of only one point.

Let $\mathcal{H}^j$ be the space of functions on lattice Λ^j. Then we introduce *grid transfer operators:*

$$\text{the interpolation operator}: \quad \mathcal{A}^{[kj]} : \mathcal{H}^j \mapsto \mathcal{H}^k \quad \text{and} \tag{24}$$

$$\text{the restriction operator}: \quad \mathcal{C}^{[j\,k]} : \mathcal{H}^k \mapsto \mathcal{H}^j \quad , \tag{25}$$

with $k < j$. So operators $\mathcal{A}^{[k\,j]}$ interpolate from a coarser to a finer grid, whereas the restriction operators do the reverse. We do not actually need transfer operators between all layers of the multigrid; there are two different choices for the set of transfer operators needed.

To see this, let us use the observations made so far to define the principles of a multigrid method: We have seen that the error is smoothened by relaxation on the fundamental layer. Therefore it should be possible to write it as (we introduce layer-indices here to distinguish between quantities on different layers)

$$\mathbf{e}^0 = \mathcal{A}^{[0\,1]}\mathbf{e}^1 \quad . \tag{26}$$

Inserting this into the error equation yields

$$\mathbf{D}_0\mathcal{A}^{[0\,1]}\mathbf{e}^1 = \mathbf{r}^0 \tag{27}$$

$$\mathcal{C}^{[1\,0]}\mathbf{D}_0\mathcal{A}^{[0\,1]}\mathbf{e}^1 = \mathcal{C}^{[1\,0]}\mathbf{r}^0 \tag{28}$$

$$\mathbf{D}_1\mathbf{e}^1 = \mathbf{r}^1 \quad , \tag{29}$$

which involves only functions and operators on the block lattice. Here we have defined the blocked operator $\mathbf{D}_1$ as $\mathbf{D}_1 = \mathcal{C}^{[1\,0]}\mathbf{D}_0\mathcal{A}^{[0\,1]}$. We can also choose $\mathcal{C}^{[j\,k]} = \mathcal{A}^{[k\,j]^*}$, where * *denotes the adjoint* operator. We will always use this *Galerkin choice* in the rest of this thesis[5].

Now we can proceed in two different ways. The equation (29) to be worked on lives on layer Λ^1. We can either try to solve it in the same way as we solved the *fundamental equation*, that is using a multigrid method. In this case we would relax on the equation and then block from Λ^1 to the next layer Λ^2 and proceed recursively. After having calculated a solution to the blocked equation on a layer Λ^j, we will correct the solution on the *next-finer* layer: $e^{j-1} \leftarrow e^{j-1} + \mathcal{A}^{[(j-1)\,j]}e^j$. *We might* also do some additional relaxation sweeps here to eliminate high-frequency errors that might have been introduced through the interpolation. The recursion stops because the solution on the last layer, which consists of only one point, is trivial. This method is called a *true multigrid method*.

Or we can just relax on the equation, interpolate the obtained correction back to the fundamental layer, $\tilde{\xi} \leftarrow \tilde{\xi} + \mathcal{A}^{[0\,1]}\mathbf{e}^1$, and then block the fundamental error equation directly to the block lattice Λ^2. This method is called a *unigrid* method.

In the first case, we will only need interpolation operators between adjacent layers, in the second case we need interpolation operators between the fundamental and all other layers, but there will never be a direct transfer between any of the coarser grids.

It is important to notice that we have to restrict the operators to a part of the lattice, otherwise interpolation would be too costly. For the time being we will choose the supports of the interpolation operators to be fixed blocks $[x]$ as shown in figure 4. The operators possess representations as rectangular matrices with elements $\mathcal{A}^{[0\,j]}_{zy}$ with $x \in \Lambda^j$ and $z \in \Lambda^0$. The matrix $\mathcal{A}^{[0\,j]}$ contains all interpolation operators on layer j. The single interpolation operator on block $[x]$, denoted by $\mathcal{A}^{[0\,j]}_{\cdot y}$, corresponds to one column vector of the matrix operator $\mathcal{A}^{[0\,j]}$ and vanishes outside the block $[x]$. It has been found that there is little hope in eliminating critical slowing down for the inversion of the Dirac-operator without overlapping blocks [69]. As we intend to apply our algorithm also to this case, we choose overlapping blocks as shown in the figure.

4.1.3 The Crucial Differences between Multigrid and Unigrid

For the following it is extremely important to understand the crucial differences between a unigrid and a true multigrid method.

[5] With these definitions, a multigrid method is completely defined by two ingredients: The structure of the block lattices and the choice of interpolation operators. Actually, one can even consider the block-lattices as defined through the interpolation operators as the relevant topological structure of these lattices is given completely by the operator on this coarser grid $\mathbf{D}_1$.

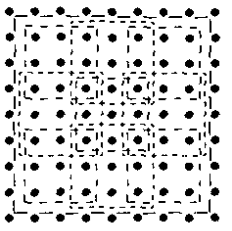


Figure 4: Supports of the interpolation operator for the layers 1 and 2. On layer 1, more than one support is drawn to show the overlap.

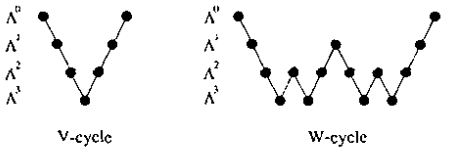


Figure 5: The different multigrid cycles. Each dot represents a relaxation step on the corresponding layer.

First, and this is mostly stressed in the literature, the unigrid method is inferior on CPU-time grounds.

For the true multigrid the recursive structure ensures that the work for relaxation and interpolation on the coarser layers is negligible compared to the relaxation costs on the fundamental layer and the cost of blocking and interpolating to and from the first block layer. So it is possible to do more work on the coarser layers and use so-called *higher cycles*: Here one goes γ times to the next-coarser lattice before going back to the finer grid, where γ is the *cycle index*. The usual method is called a V-cycle, corresponding to $\gamma = 1$, the next-higher cycle is a W-cycle with $\gamma = 2$, see figure 5[6]. As the work on layer Λ^j decreases with a factor of 2^{-dj} and increases with the cycle index as γ^j, for $\gamma < 2^d$ the work is of order $\mathcal{O}(n)$. Such higher cycles are sometimes necessary to improve the convergence.

For the unigrid method, the main computational costs are due to the interpolation, each layer needs $\mathcal{O}(n)$ operations because interpolation always starts from the fine grid. So for a V-cycle, the work will be $\mathcal{O}(n \ln n)$ and higher cycle indices are not allowed.

On pure mathematical grounds, ignoring computational costs, we can formulate each multigrid algorithm as a unigrid by putting together the interpolation operators as $\mathcal{A}^{[0\,j]} = \mathcal{A}^{[0\,1]}\mathcal{A}^{[1\,2]}\ldots\mathcal{A}^{[(j-1)\,j]}$. The unigrid method therefore is more general than the multigrid method because not all interpolation operators can be factorized in this way.

What is the meaning of this greater generality? In a true multigrid method, the interpolation operator $\mathcal{A}^{[0\,1]}$ must be able to represent *all bad-converging (smooth)* modes on the coarser lattice, since viewed from the fundamental layer there are only two steps: relaxation, which reduces the oscillatory part of the error, and the coarse-grid correction from the layer Λ^1. That there is a whole

[6] *One can also be more exact and include the number of relaxation sweeps done before (ν_1) and after (ν_2) the coarse-grid correction step into the cycle description and speak of $V(\nu_1,\nu_2)$-cycles, etc.*

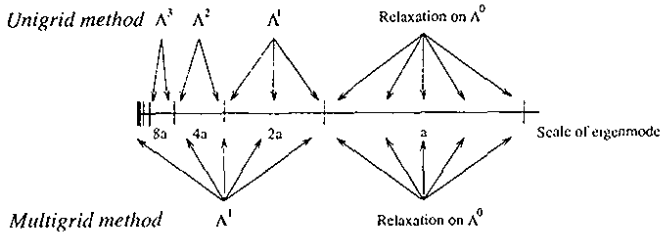


Figure 6: The difference between unigrid and multigrid: In a multigrid method, the layer Λ^1 must *eliminate all* parts of the error with a length-scale larger than the fundamental one; in a unigrid this layer only needs to *smooth the error on its length-scale* a_1, the larger length-scales are dealt with on the coarser grids. This picture shows the case of the one-dimensional Laplace equation; in higher dimensions the number of modes of length scale $\propto a$ will be larger.

multigrid on top of Λ^1 to solve the equation there is not relevant on the fundamental layer. Solving the equation on layer Λ^1 exactly, the so-called two-grid method, would be even better from this point of view.

For a unigrid, all this is not true. Layer Λ^1 only has to address that part of the error that is smooth on scale a_1 and not on larger scales because we only do relaxation on this scale, the blocked equation is not solved. The smoother parts of the error, corresponding to larger length-scales, will be addressed by the other interpolation operators $\mathcal{A}^{[0\,j]}$. So each layer Λ^j of the unigrid has to smoothen the error only on scale a_j, but not on the coarser scales. Figure 6 pictures this difference. Note that in this thesis we always use a full unigrid, where the coarsest layer consists of only one point. One could imagine other unigrid methods, where on a certain coarse lattice all modes on this and the coarser scales are represented, so that it is possible to solve the equation on this lattice and there is no need to proceed to coarser ones.

Let us rephrase this in other terms: The interpolation operators map a space with a larger dimension onto a space with a lower one. Hence they possess a *nullspace*, a space of vectors that are mapped to zero. All vectors in this nullspace must be eliminated by the relaxation method or mapped to vectors not in the nullspace, otherwise these parts of the error can not be reduced. For a multigrid, the nullspace of the interpolation operators $\mathcal{A}^{[0\,1]}$ is important, for a unigrid, the nullspace to be considered is the combined nullspace of all interpolation operators on all layers. This nullspace will be smaller than that of $\mathcal{A}^{[0\,1]}$ whenever the higher interpolation operators do not possess the factorization property explained above.

A good interpolation operator for a unigrid will not necessarily be a good interpolation operator for a multigrid, the requirements for the latter being stronger. Thus it might be easier to construct a unigrid than a multigrid, and not every unigrid can be modified in an easy way to become a multigrid. In fact, as we will see later, the method discussed in this thesis is a genuine unigrid method.

The unigrid method also has the advantage that there are no stability requirements to the blocked operators $\mathbf{D}_j$, see [67]. Stability means the following: Suppose we have a scheme for determining interpolation operators for a fundamental problem operator $\mathbf{D}_0$ that belongs to a class $\mathcal{F}$. If we do a multigrid which is truly recursive, our method is only well-defined if we can apply it also to the blocked operator $\mathbf{D}_1$. This operator will not always belong to the same class $\mathcal{F}$ but to a larger class $\mathcal{F}'$ so we must ensure that our method of determining the interpolation operators is general enough to deal with operators from this larger class. This of course extends to all the higher layers as well, after the second blocking the class may become even larger. So there must exist a class $\mathcal{F}^{\text{stab}}$ with the property

that each operator in this class will be mapped to another operator in it under coarsening, and our method must be well-defined for operators from this class. In other words, there must exist a layer Λ^j with the property that $\mathbf{D}_j$ is stable under coarsening: All operators $\mathbf{D}_k$ with $k > j$ must belong to the same class. A case common in multigrid methods for propagators is that the Dirac operator with couplings that are matrices in the gauge group G gets mapped to an operator whose couplings are *dielectric gauge fields* [39].

In a unigrid, however, all interpolation operators are mapping to the fundamental lattice and no blocking from a coarser layer will be involved. Therefore there is no stability requirement for the blocked operators.

4.1.4 Important Properties of the Interpolation Operators

The interpolation operators have to be able to approximate all bad-converging modes in the system. If a bad-converging mode would not lie in the range of the interpolation operators, it could not be represented on a coarser grid and so the part of the error corresponding to this mode would not be reduced. This is true for both multigrid and unigrid, but for a multigrid the operators $\mathcal{A}^{[0\,1]}$ alone must have this property as explained in the previous subsection and shown in figure 6.

It is not enough to know all the bad-converging modes. As there are $\mathcal{O}(n/2)$ of these modes and each of them will be non-vanishing on (nearly) all grid points, simply reading them in to use this knowledge would have computational costs of order $\mathcal{O}(n^2)$.

What we have to do is to approximate all these modes by *localized objects*, the interpolation operators. We can have many interpolation operators localized on a smaller length scale, but only a few on the very large scales, otherwise the interpolation would be too costly. It is not a priori clear that such objects can be found for each problem operator. This also depends on the relaxation method chosen to smoothen the error.

We can also rephrase these statements in the language introduced in chapter 3: We are looking for *search directions* in the parameter space. The interpolation operators provide these search directions, in which the energy surface is smooth, subject to the restriction that they have to be localized on their scale as well. In the case of a disordered problem, these directions have to be computed, they are not known a priori.

4.2 The Algebraic Multigrid

The basic principle of multigrid algorithms (for ordered elliptical problems) originates from the observation that after doing local relaxation sweeps, the error gets *smooth*. Hence it should be possible to represent the error on a coarser lattice because the intermediate values can be obtained by smooth interpolation. "Good" interpolation operators are known beforehand; for example, one may use linear interpolation. Normally, the smooth modes are the low-lying eigenmodes of the operator. This, however, will not be true for a disordered system. In the case of a Lattice Gauge Theory there even is no a priori definition of smoothness, as we will see in the next section.

The Algebraic Multigrid or AMG algorithm [61] was designed to handle such problems. It is a black-box algorithm: only the problem matrix $\mathbf{D}$ and the righthandside $\mathbf{f}$ are the necessary ingredients to the algorithm. No assumptions about any underlying topological structure (like a regular grid) are made. The choice of block lattices and interpolation operators is only based on the algebraic information given through the matrix, hence the name "algebraic multigrid". The method is applicable to all matrices $\mathbf{D}$ which fulfill the conditions

1. $\mathbf{D}$ is positive definite.
2. $\mathbf{D}$ is diagonally dominant, i.e. $D_{zz} > 0$, $D_{zz'} \leq 0$ for $z \neq z'$, and $\sum_{z'} D_{zz'} \geq 0$ for all z.
3. $\mathbf{D}$ is in some sense sparse. (How many of the elements of the matrix have to be zero is case-dependent.)

The basic idea of the method is to use the key observation that the error gets smooth after relaxation and is not further reduced afterwards as a definition of smoothness: The error is defined to be *"algebraically" smooth*, if it is not reduced significantly by the chosen relaxation method. If we use the *above definitions of the error and the residual*, we can write the error as

$$e_z^{\text{new}} = -\frac{r_z}{D_{zz}} + e_z^{\text{old}} \quad . \tag{30}$$

As the error is not reduced anymore, we have $e_z^{\text{new}} \approx e_z^{\text{old}}$, so the residual is small compared to the error. In the language of section 3 this means that we are in a long narrow valley; the height difference to the true solution is not large, but the distance is. Inserting the definition of the residual, we get

$$D_{zz}e_z \approx -\sum_{z' \neq z} D_{zz'}e_{z'} \quad . \tag{31}$$

This formula is the basis for the definition of the block lattices and the interpolation operators because it allows us to connect the error at one point with the error at another for algebraically smooth errors.

The principle of choosing the block lattice is the following: Define two points z and z' as neighbours when $D_{zz'} \neq 0$. For each point z distinguish between *strong* and *weak* neighbours, where a neighbour z' is strong when $|D_{zz'}| > \alpha \max_{z''} |D_{zz''}|$. The parameter α is usually chosen to be $\alpha = 0.25$. Then there is an algorithm ensuring that each point either is a block lattice point itself or is strongly connected to one.

The interpolation operators can be defined directly from equation (31):

$$\mathcal{A}^{[0\,1]}_{z',z} = -\frac{D_{zz'}}{D_{zz}} \quad . \tag{32}$$

This is the so-called *weighted interpolation*.

The AMG has been applied successfully to many disordered problems. We will see an example in section 9.2. However, the conditions to be fulfilled by $\mathbf{D}$ usually do not allow the method to be applied directly to systems of equations, i.e. where a vector sits at each point of the grid. It is also not obvious how to adapt the weighted interpolation in this cases.

4.3 The Meaning of Smoothness

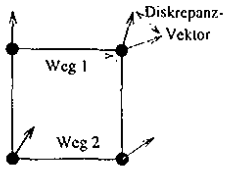
(This and the following section are partly based on joint work with T. Kalkreuter, G. Mack and M. Speh [80, 84].)

In a Lattice Gauge Theory, no a priori notion of smoothness is given: The naive notion of smoothness might be quantified by saying that $\sum_\mu \langle \nabla_\mu \xi, \nabla_\mu \xi \rangle \ll \langle \xi, \xi \rangle$. Here ∇_μ is the ordinary derivative. This definition is obviously not gauge covariant. We could try to make it so by using the covariant derivatives $\tilde{\nabla}_\mu$ instead, but in this case we get

$$\sum_\mu \langle \tilde{\nabla}_\mu \xi, \tilde{\nabla}_\mu \xi \rangle = \langle \xi, -\Delta \xi \rangle \geq \varepsilon_0 \langle \xi, \xi \rangle \quad , \tag{33}$$

where the lowest eigenvalue of the covariant Laplace operator ε_0 is, however, a measure of the disorder of the gauge field and therefore not necessarily small. So there are no smooth modes in this sense *when the gauge field is disordered*.

We can understand this intuitively: Smoothness means that a function changes little when we compare its values at different points. In a gauge field background, this transport has to be *done* using the parallel transporter and so it is path-dependent: What is smooth one way need not be smooth another way:



Unfortunately, this does not mean that there are no bad-converging modes when we try to solve a differential equation in a gauge field background. Some algebraic notion of smoothness should still be appropriate.

We want to define smoothness *on a certain length scale*; this is necessary for a unigrid because different layers have to address modes smooth on different scales. For a multigrid only the fundamental scale would be of interest, every mode smooth on this scale must be eliminated through the coarse-grid correction.

So we propose the following definition of smoothness in a disordered system, assuming that a fundamental differential operator $\mathbf{D}_0$ specifies the problem [80].

Definition 1: A function ξ on Λ^0 is *smooth on length scale* a when

$$\|\mathbf{D}_0 \xi\|^2 \ll a^{-l} \|\xi\|^2 \quad , \tag{34}$$

when the operator $\mathbf{D}_0$ is of dimension [length]$^{-l}$.

This definition implies that the smoothest function is the lowest eigenmode of $\mathbf{D}_0$.[7]
So if the slow-converging modes are the low-lying eigenmodes of the problem operator, as they usually are, we arrive at the basic principle stated above also for the disordered case: The slow-converging modes, which have to be represented on coarser grids, are smooth. Of course we still have to show that the definition is sensible and can be used to construct a good algorithm.

As a first step we can see that the notion of *algebraic smoothness* as introduced in the previous *section implies smoothness in our sense. Remember that a mode was called algebraically smooth, if it* is not reduced efficiently by relaxation which implies that the residual is much smaller than the error:

$$r_z^0 = \sum_{z'} D_{0,zz'} e_{z'}^0 \ll e_{z'}^0 \Longrightarrow \|\mathbf{D}_0 \mathbf{e}^0\|^2 \ll \|\mathbf{e}^0\|^2 \quad . \tag{35}$$

The crucial step in the setup of the algorithm is the choice of the grid transfer operators $\mathcal{A}^{[0\,1]}$ *and* $\mathcal{C}^{[1\,0]}$. These operators should be smooth in our sense because we want to use them to represent a smooth error on a coarser grid. Since our definition of smoothness depends on the problem matrix $\mathbf{D}$, they are not given a priori. Instead, we have to *compute* these operators.

We now look for smooth interpolation operators in the specified sense which fulfill (approximately) the eigenvalue equation restricted to the block $[x]$

$$\mathbf{D}_0|_{[x]} \mathcal{A}^{[0\,1]}_{z,x} = \varepsilon_0(x)\, \mathcal{A}^{[0\,1]}_{z,x} \quad . \tag{36}$$

Here $\mathbf{D}_0|_{[x]}$ denotes the restriction of $\mathbf{D}_0$ to the block $[x]$ and $\varepsilon_0(x)$ is the lowest eigenvalue on the block. As the interpolation operator must vanish outside the block, we impose Dirichlet boundary *conditions*.

From this idea a restriction to our algorithm follows: It will only work for positive definite operators. This is the reason why we will always work with the squared Dirac operator.

[7] It has recently been remarked by Sokal [67] that the operator $\mathbf{D}$ does not possess eigenvectors in a strict sense. We will come back to this question shortly.

The crucial assumption of our algorithm is that the solution $\mathcal{A}^{[0,j]}_{,x}$ of this equation is smooth on length scale a_j. This is true for the scalar Laplace-operator, where the solution is half a sine wave on the block.

What happens when this is not true? It might be that the bad-converging modes are not associated with the lowest eigenmodes on the blocks. Is the method doomed to fail in this case? Fortunately, we can change our definition of smoothness and return to the AMG definition given above, but suitably adapted to our case. To calculate a smooth mode on the length scale of a block $[x]$, we can use the algorithm to try to solve the equation

$$\mathbf{D}_0|_{[x]}\mathcal{A}^{[0,j]}_{,x} = 0 \quad . \tag{37}$$

Of course the solution to this equation is zero, but the algorithm will not arrive at this solution immediately. Instead, it will converge to the shape of a bad-converging mode. If the algorithm we are using is able to deal with the bad-converging modes on the smaller scales, the approximate solution we will get after some iterations of the algorithm will be smooth on the scale of the block $[x]$, exactly as we wanted it to be.

This is an example of the following

principle of indirect elimination: It is easier to calculate the shape of a bad-converging mode for a certain algorithm than to reduce it directly using this algorithm.

To see this, consider the case where there is only one bad-converging mode and all others are reduced efficiently by the algorithm. By *trying* to solve the equation $\mathbf{D}\xi = 0$, after a few iterations the approximate solution $\tilde{\xi}$ will have the shape of the bad-converging mode which can then be used to eliminate this mode from the error in the fundamental equation. We will see a direct application of this principle in section 5.4.

Before we present an algorithm which will use these ideas, we have to answer an objection, that has been raised against this and similar proposals:

4.3.1 Does D *Possess Eigenvectors?*

Recently, Sokal has pointed out [67] that the operator $\mathbf{D}$ does not possess eigenvalues in a strict sense because it maps a space on its dual and there does not exist a natural scalar product on the two spaces. One can see this easily for the Laplace equation as it occurs in electrodynamics: $\Delta\phi = 4\pi\varrho$. Here it is not meaningful to say that ϕ is an eigenvector of the Laplace operator since it maps potentials onto charge densities. In simpler words: As the Laplace operator is not dimensionless, applying it to some quantity results in a quantity of different physical dimension. The Laplace operator does provide us with a bilinear form, however. The quantity $\langle\phi, \Delta\phi\rangle$ is meaningful, because the scalar product between a potential and a charge density is the energy of the electrical field.

One might argue that a vector space and its dual are isomorphic, but there are many maps between the two spaces and only when it is clear which of these is the one physically meaningful can we use this fact to identify the two spaces in a unique way.

If all this is true, why is it so common to speak of the eigenvectors of the problem operator in the multigrid literature?

The reason is that the multigrid method provides an identification between the two spaces through the relaxation method: As we saw in equation (22), during relaxation we map the old error (which lives in the same vector space as the solution vector) to the new error, using the iteration matrix $\mathbf{M} = \mathbf{1} - \mathbf{BD}$, where $\mathbf{B}$ depends on the relaxation scheme. The matrix $\mathbf{B}$ can be used to define a mapping between the two vector spaces. In the case of the Jacobi relaxation and for an operator $\mathbf{D}$ with constant diagonal, the matrix $\mathbf{B}$ will be a multiple of the unit matrix *in the site-wise basis* therefore it seems as if the identification of the spaces is natural. However, for other relaxation methods or problem operators this might be a problem. We encountered this problem already in chapter 3, where we saw that the choice of a basis corresponds to a choice of search directions in the parameter space and that the question whether critical slowing down occurs depends on this choice.

Is the ISU algorithm invalidated by this observation? The answer is no for several reasons: First of all, we might not use the eigenvalue method, but the other idea, where the righthandside is chosen to be zero. In this case there is no need to think about eigenvectors of $\mathbf{D}$. Secondly, we could adjust the eigenvalue equation and compute the lowest eigenvectors of $\mathbf{BD}$ using inverse iteration. Finally, we can assume that the site-wise basis is a natural basis of our problem. Then we can check whether the relaxation algorithm really reduces the low-lying eigenmodes of the operator $\mathbf{D}$ which can now be defined using the mapping given through the basis. If this behaves as expected, we have shown that the site-wise basis is appropriate for this relaxation scheme. This is the point of view we will take in most of this thesis. However, one should always keep in mind that the meaning of eigenmodes depends on the choice of the relaxation scheme.

4.4 The ISU Algorithm

After this digression, we now have to answer the crucial question: *"Where do we get the smooth operators?"* Consider for instance the eigenvalue equation for $\mathcal{A}^{[0,N]}$ (the considerations for the alternative choice of the interpolation operators described above are similar).

$$\mathbf{D}_0\,\mathcal{A}^{[0,N]} = \varepsilon_0\,\mathcal{A}^{[0,N]} \quad , \tag{38}$$

involving the full unrestricted operator $\mathbf{D}_0$ because the last lattice consists of only one point. If we want to solve this equation with inverse iteration (i.e. by computing $\mathbf{D}_0^{-n}\mathcal{A}^{[0,N]}_{,\text{start}}$ for large n and an arbitrary starting vector), we will have to solve an equation which seems to be exactly as difficult as our starting point, equation (17).

But this is not so. The worst-converging mode of our starting equation is the mode to the lowest eigenvalue of $\mathbf{D}_0$, but now we want to *compute* this mode, so it does not contribute to the error of the eigenvalue equation. (We have to do a simple normalization step after each iteration.) Consequently the mode to the second-lowest eigenvalue of $\mathbf{D}_0$ is the one that converges worst and if we could handle this (and all higher modes as well), we could also handle the lowest mode in our inhomogeneous equation (17) by solving first equation (38). This is again an illustration of the principle of indirect elimination that it is easier to calculate the shape of a bad-converging mode than to reduce it directly.

The basic idea of our algorithm is that the higher modes are smooth on shorter length-scales. This means that it should be possible to construct them out of pieces which are smooth on these length scales and have supports on parts of the lattice. So the next-lowest modes ξ^{low} are representable by linear combination of the interpolation operators $\mathcal{A}^{[0,N-1]}_{,x}$:

$$\xi^{\text{low}}_x = \sum_{x\in\Lambda^{N-1}} c_x \mathcal{A}^{[0,N-1]}_{,x} \quad . \tag{39}$$

If this is true we see that the calculation of $\mathcal{A}^{[0,N-1]}$ is similar. Again the worst-converging mode is the mode we aim at, the next-higher modes can be represented on smaller blocks. Hence their calculation is simpler. Finally we arrive at the calculation of $\mathcal{A}^{[0,1]}_{,x}$, having to solve an equation on a 3×3-lattice. This is easily done. Because of the Dirichlet boundary conditions there is no low-lying mode here.

The algorithm proceeds in the direction opposite to this explanation: We start with the smallest blocks and after knowing interpolation operators on these blocks we proceed to the coarser scales, always taking the already known interpolation operators into account.

Remark: It might happen that the lowest eigenvalue is much larger than the difference between it and the next eigenvalue. In this case, many inverse iterations have to be done to resolve the two corresponding modes. A possible remedy for this problem is to calculate estimates $\tilde{\varepsilon}_0$ of the lowest eigenvalue as we proceed and to invert not $\mathbf{D}_0$ but $\mathbf{D}_0 - \tilde{\varepsilon}_0 \cdot 1$. We will study this question again shortly in section 5.3.

We identify the site $x \in \Lambda^j$ with the block $[x]$ in Λ^0 having x at the center. Equation (36) is an eigenvalue equation on $[x]$ for the vector $\mathcal{A}^{[0,j]}_{,x}$. It can be solved via inverse iteration by our unigrid

method, using the already calculated interpolation operators $\mathcal{A}^{[0\,k]}_{\cdot,y}$ with supports inside the block. With this we arrive at the following

Algorithm for calculating smooth interpolation operators (Eigenmode version):

For $1 \leq j \leq N$ **do**

For all $x \in \Lambda^j$ **do**

1. Choose initial value $\mathcal{A}^{[0\,j]}_{x;\mathrm{start}}$.
2. Relax *on the fundamental lattice on block* $[x]$.
3. **For** all $1 \leq k < j$ **do:**
   Calculate the residual $\mathcal{R}^{[0\,j]}_{\cdot,x} = \mathbf{D}_0|_{[x]}\mathcal{A}^{[0\,j]}_{\cdot,x} - \mathcal{A}^{[0\,j]}_{\cdot,x,\mathrm{start}}$.
   Block the residual to layer Λ^k: $\mathcal{R}^{[0\,j]'}_{\cdot,x} = \mathcal{A}^{[0\,k]}\mathcal{R}^{[0\,j]}_{\cdot,x}$.
   Calculate $\mathbf{D}_k|_{[x]} = C^{[k\,0]}\mathbf{D}|_{[x]}\mathcal{A}^{[0\,k]}$.
   Determine approximate solution of $D_k|_{[x]}\delta\mathcal{A}^{[0\,j]'}_{\cdot,x} = \mathcal{R}^{[0\,j]'}_{\cdot,x}$ by relaxation on $[x]\cap\Lambda^k$.
   Correct $\mathcal{A}^{[0\,j]}_{\cdot,x}$: $\mathcal{A}^{[0\,j]}_{z,x} \longleftarrow \mathcal{A}^{[0\,j]}_{z,x} + \sum_{y\in\Lambda^k}\mathcal{A}^{[0\,k]}_{z,y}\delta\mathcal{A}^{[0\,j]'}_{y,x}$.
4. Normalize the interpolation operator.
5. **If** approximate solution $\mathcal{A}^{[0\,j]}_{\cdot,x}$ good solution of the eigenvalue equation **then do next** x
   If approximate solution $\mathcal{A}^{[0\,j]}_{\cdot,x}$ good solution of the inhomogeneous equation
   then set $\mathcal{A}^{[0\,j]}_{x,\mathrm{start}} = \mathcal{A}^{[0\,j]}_{\cdot,x}$.
6. **go to** 2.

We call this method *Iteratively Smoothing Unigrid* or ISU because it is a unigrid method which computes smooth operators by means of an iterative method (and not directly from the given operator as in the AMG-algorithm).

Instead of solving the eigenvalue equation, one could use the alternative definition and try to solve an equation with righthandside zero. In this case the approximate solution is equal to the error, so trying to solve the equation $\mathbf{D}\mathcal{A}^{[0\,j]}_{\cdot,x} = 0$ means that we apply the iteration matrix $\mathbf{M}$ several times to the initial guess and so project directly onto the eigenvector of the iteration matrix with the largest eigenvalue which is the worst-converging mode. So this method restricts the iteration matrix to larger and larger parts of the grids and projects onto the worst-converging mode on each scale.

An idea that is similar in spirit has been discussed in [72, section 4.6]. Brandt proposes to do relaxations on the fundamental lattice with arbitrary starting vectors to determine "typical shapes of a slow-to-converge error". Unfortunately, this idea suffers from a severe disease: The number of modes *that converge badly under simple relaxation is huge (about half of the number of grid points).* What one will get by this procedure is a mixture of low-lying eigenmodes with contributions depending on their eigenvalues. The time needed to arrive at a function that consists only of the lowest eigenmodes will be proportional to the lattice size, so the method will not work without critical slowing down.

The difference to our method is that this is a *unigrid* method: On each length scale we need not represent all modes that converge badly on this and on all higher scales; only the modes that belong to the scale corresponding to a certain lattice constant have to be dealt with. The next-coarser length-scale will then take care of the modes corresponding to this scale and *in their computation the* smaller scales are already taken into account.

Our first implementation of ISU uses the eigenvalue method, but we will come back to the other idea in section 10.2.

Now we could try to use ISU to define a true multigrid algorithm: Just replace the eigenvalue equation for $\mathcal{A}^{[0\,j]}_{\cdot,x}$ by an equation for $\mathcal{A}^{j}_{\cdot,x}$ which interpolates between adjacent layers and use $\mathbf{D}_{j-1}$ as the operator for this equation to get operators which are smooth with respect to the blocked

differential operator. But this will usually not work: Formulate the new true multigrid algorithm in unigrid language. It involves operators $\mathcal{A}^{[0\,j]} = \mathcal{A}^{[0\,1]}\mathcal{A}^{[1\,2]}\ldots\mathcal{A}^{[(j-1)\,j]}$. But the product of operators which are smooth on different length scales is smooth only on the shorter of these length scales. So we will never get a transfer operator that is smooth on large scales which is an example for what was said above: A good interpolation operator in a unigrid algorithm will not necessarily be a good interpolation operator for a multigrid.

This tells us that our algorithm really is a unigrid algorithm, not just a multigrid in disguise. It is therefore impossible to apply the usual two-grid-analysis to prove convergence. Furthermore, we cannot stop the algorithm on a layer $j < N$ because in this case the modes on the larger scales would not be handled appropriately.

From the above description it is clear that the work involved in calculating good interpolation operators is larger than the time needed for the solution of the equation (17) itself. The following table shows the computational costs of the algorithm, compared to a true multigrid and to a local relaxation. As always, L denotes the grid length and d is the dimension.

Algorithm	CPU-time	Storage space
Local relaxation (with periodic boundary conditions [69])	$L^d(\delta m^2)^{-z/2}(z\approx 2)$	L^d
True multigrid and AMG	L^d	L^d
ISU-algorithm	$L^d \ln^2 L$	$L^d \ln L$

5 Performance of ISU

Wherein the ISU algorithm succeeds in eliminating critical slowing down for a model problem, but fails for the Dirac equation; and wherein some subtle points are made which enable us to deal with instanton problems.

5.1 ISU for Bosons

We studied this algorithm for the 2-dimensional bosonic model problem

$$(-\Delta - \varepsilon_0 + \delta m^2)\, \xi = \mathrm{f} \quad . \tag{40}$$

Here $-\Delta$ is the negative bosonic Laplace operator in an SU(2) gauge field as described in equation 8. Its lowest eigenvalue is large, so there would be no need to apply a multigrid method. By subtraction of this eigenvalue ε_0 we can make the problem critical and directly control criticality by tuning δm^2, the lowest eigenvalue of the full problem. We will use this tuning of criticality throughout the thesis. We measured the inverse asymptotic convergence rate τ defined as

$$\tau = -\lim_{m\to\infty} \frac{1}{\ln \varrho_m} \quad , \tag{41}$$

where ϱ_m is the quotient of the error norms before and after iteration number m. This quantity cannot be measured directly because the error is unknown. Instead we measured this quantity by substituting the residuals for the errors. We checked that measuring the residuals instead of the errors does not affect τ much by calculating a solution to a very good approximation and then comparing both definitions for several cases. The differences usually amounted only to a few percent and in all cases the true value was smaller than the value from the residual computation.

That finite-size effects do not spoil criticality can be easily seen by applying a standard iterative method to the problem. We choose Gauss-Seidel relaxation as a test method. It suffers from severe critical slowing down when applied to the model problem. The critical exponent z was found to be 2 independent of the grid size, as we had to expect from the works of Kalkreuter [69]. The behavior of the conjugate gradient algorithm should be similar since its convergence only depends on the condition number (quotient of largest and smallest eigenvalues) for ordered systems [56] as well as for disordered ones [71].

Figure 7, top, shows the inverse asymptotic convergence rate of the ISU-algorithm as a function of δm^2 for grid sizes 32^2–128^2 at $\beta = 1.0$ in an SU(2)-gauge field background. Absence of critical slowing down can be seen clearly, and the absolute value of τ is quite small. ($\tau = 1$ corresponds to a reduction of the residual by a factor of ϵ in one multigrid sweep. The standard Gauss-Seidel relaxation reaches values of $\tau \approx 2 \cdot 10^5$ at $\delta m^2 = 10^{-5}$.) The sweeps are V-cycles with one pre- and one post-relaxation step. The results do not vary appreciably by changing β as can be seen from the bottom part of the figure which shows τ for three different values of β on a 64^2-grid. We did 30–50 configurations for each data point.

Remark: To conclude that critical slowing down is absent, it is necessary to study the dependence of the convergence time on δm^2 not only for fixed but also for varying grid sizes. The reason is that our method ensures correct treatment of the *lowest* mode with eigenvalue δm^2. But the eigenvalue of the second-lowest mode depends on the grid-size, so the grid-size has to be large enough to make also this eigenvalue fairly low.

Because of the large work involved in calculating the interpolation operators it has also to be verified that the number of inverse iterations needed for the calculation of the interpolation operators does not grow with the grid size. We found that six multigrid sweeps with one pre- and no postrelaxation step are sufficient for this on every layer *regardless of the lattice size*; doing more sweeps and thereby

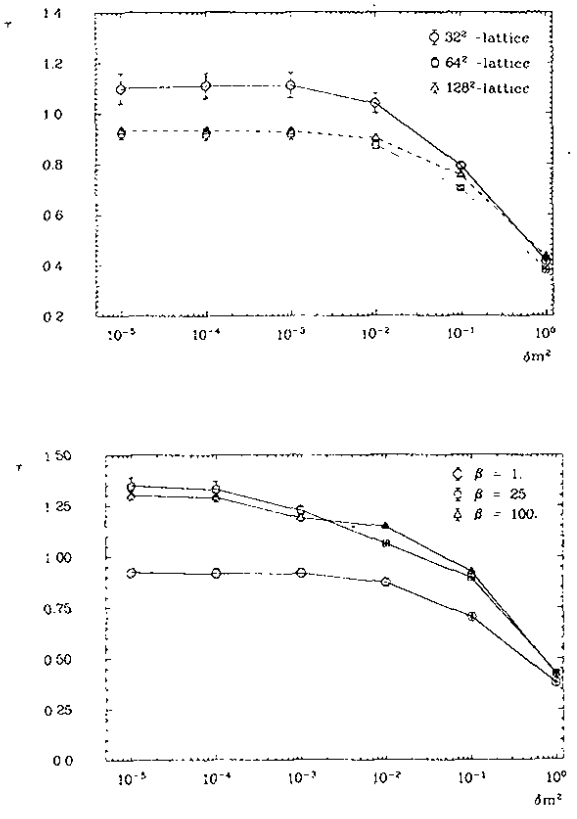


Figure 7: Performance of the ISU-algorithm for the model problem of equation (17) in a SU(2)-field.
Top: The inverse asymptotic convergence rate τ is shown for different grid sizes as function of the critical parameter δm^2 at $\beta = 1.0$.
Bottom: τ as a function of δm^2 is shown for different β on grids with size 64^2.
Lines are only drawn to guide the eye.

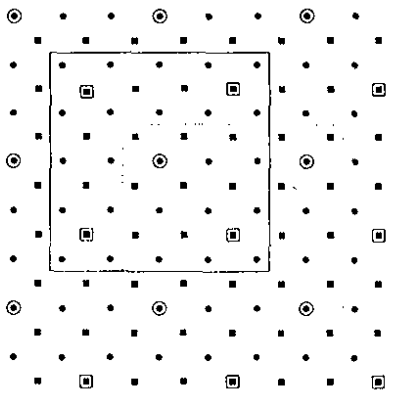


Figure 8: The symmetry of the staggered grid. The block-grid points are a distance of 3 apart. Only one of the two sub-lattices is shown.

calculating the operators more exactly did not improve the convergence of equation (40). There also is a dependence on δm^2: The smaller δm^2, the *less* inverse iterations are necessary, the number of inverse iterations needed slightly decreases with increasing criticality. (This effect only is appreciably for large values of $\delta m^2 \approx 1.0$, where the criticality is so small that standard algorithms can be used for the solution.) This can be understood easily: the inverse iteration has to resolve the lowest from the second-lowest eigenmode and therefore it works the better, the smaller the ratio $\delta m^2/(\varepsilon_1 - \delta m^2)$ becomes, where ε_1 is the second-lowest eigenvalue [56].

5.2 *ISU for Fermions*

After this success let us now try the algorithm at a more difficult problem: The Dirac equation. Again we use an SU(2) gauge field on a square two-dimensional lattice. The equation to solve is the squared Dirac equation, see section 2.1

$$(\not{D}^2 - \varepsilon_0 + \delta m^2)\xi = \mathrm{f} \quad . \tag{42}$$

Again we subtract the lowest eigenvalue to tune criticality. This is not truly necessary for the Dirac operator because its smallest eigenvalue is quite low, but in this way we can control the criticality parameter exactly which is important in measuring critical slowing down.

We choose a block-factor of 3 instead of 2, so that the block-lattice preserves the symmetry of the fine lattice, see figure 8. This is advantageous for a true multigrid algorithm [68]; for a unigrid, it is not really necessary as there is no blocking starting on a coarse grid. Another disadvantage of this blocking procedure is that the effective operator has 17 couplings, whereas the fundamental squared Dirac operator only has nine. (One to the point itself and eight to the eight neighbours of the same relative pseudoflavor that are taxi-driver distance 2 apart.) This increases the work in calculating these operators which is not negligible due to the unigrid-nature of the algorithm. In section 10.1 we will present an improved blocking scheme having among others the advantage that the effective operator has the same complexity as the fundamental one.

The interpolation operator on the last point which is the lowest eigenmode of the full operator now is degenerate: There is one mode on each of the sub-lattices. If we restrict our attention to one sub-lattice, we have one interpolation operator on the last layer as in the bosonic case. In the continuum limit, there are actually four sub-lattices not seeing each other, so that one should choose

Config. Nr.	*Grid size*	δm^2	β	*Nr. of runs*	τ
1	18^2	$2 \cdot 10^{-3}$	1	50	25.6 ± 0.8
2		$2 \cdot 10^{-4}$	1	50	41.5 ± 2.23
3		$2 \cdot 10^{-3}$	9	50	3.65 ± 0.15
4		$2 \cdot 10^{-4}$	9	50	4.07 ± 0.21
5	54^2	$2 \cdot 10^{-4}$	1	30	249 ± 10
6		$2/9 \cdot 10^{-4}$	9	20	8.3 ± 0.4
7		$2/9 \cdot 10^{-3}$	9	50	6.67 ± 0.14
8	162^2	$2/9 \cdot 10^{-4}$	9	5	48.9 ± 4.6
9		$2/81 \cdot 10^{-3}$	81	11	7.5 ± 0.7

Table 1: Convergence time of the ISU algorithm for SU(2) fermions for various grid sizes and β-values. Configurations 1, 7, and 9 correspond to physical scaling, as do 2 and 6. Absence of critical slowing down for these configurations is obvious. For fixed values of β and the critical parameter δm^2 (from configurations 2,5 and 6,8) critical slowing down is strong. The lowest eigenvalue of the Dirac operator has been subtracted to tune criticality, so δm^2 is directly the lowest eigenvalue.

four such operators. For simplicity, we used the following trick: We chose the righthandside of the equation to be a delta-function at one point of the grid, so that the solution lives on only one sub-lattice. In this case, one operator on the last point is sufficient: At small values of β the lowest mode on the sub-lattice is not degenerate and the operator will equal this, at large values the solution will be concentrated one one of the four pseudoflavor-sub-lattices and again the degeneracy will cause no problems. That this is a correct approach can be seen from the fact that the method converges well for large β-values, if the method were flawed, we would expect severe critical slowing down because there would exist one bad-converging mode that is not approximated well. For more general righthandsides, one should choose one interpolation operator on each of the sub-lattices.

After some test-runs, we found that the convergence for small values of β was not as good as in the bosonic case. To improve convergence we introduced an additional block layer Λ^{add} consisting of only four points, one of each pseudoflavor. These operators do not extend over the whole lattice, one row and one column of the lattice is missing. This additional layer is not consistent with the symmetry as given by the block-factor three, it is more similar to the second-last layer of a block-factor two multigrid. All results presented here and in the following were obtained with the algorithm including this additional layer.

We also increased the cycle parameters to improve the convergence and used $V(4,4)$-cycles.

We measured the convergence times τ defined in the previous section on lattices of size 18^2, 54^2, and 162^2 (grid sizes $2 \cdot 3^N$) with different values for β.

We can clearly distinguish two trends in the results shown in table 1:

1. For physically scaled gauge fields, i.e. choosing $\beta \propto L^2$ and $\delta m^2 \propto L^{-2}$, there is no critical slowing down, but "critical speeding up". This is expected because when β is increased the Dirac operator gets closer and closer to the Laplace operator for which convergence is excellent. (We checked that ISU for the Laplace equation converged well ($\tau \approx 1.5$) for block-factor 3.) The interpolation operators were always calculated with the same number of 20 inverse iterations, regardless of the lattice size. So in this case, the algorithm works fine. As in increasing the lattice size even further the difference to the Laplace operator will vanish, it seems not necessary to check that the number of iterations for the interpolation operators really does not increase with the lattice size, for we checked this thoroughly in the last section.

2. For fixed parameters β and δm^2 the convergence time increases drastically with the grid size. From configurations 2,5 and 6,8 one can estimate the critical exponent to be $z \approx 1.6$.

These results are not what we hoped for. That the convergence increases in the continuum limit is also true for other algorithms like the Parallel-Transported Multigrid, see section 11, which are simpler and less costly than the ISU algorithm. Therefore we are not content.

So our intention must be to understand the differences between the Dirac- and the Laplace-case. The PTMG, for example, will probably fail when applied to the Laplace equation presented in the previous section, because it does not take the gauge field disorder into account. As ISU works in one case, but fails in the other, there must be deeper lying reasons.

In chapters 7 and 9 we will develop analyzing tools to understand these reasons. They are not only interesting for the ISU algorithm and the Dirac equation, but for the study of other algorithms or problems as well. We will show an example of another physical model to be investigated in section 9.2.

5.3 Fine Points on Last Points

The updating on the last point is one of the crucial ingredients to the ISU algorithm. ISU shows severe critical slowing down when the updating on the last point is replaced by an exact solution on the second-last block-lattice, as should be expected, for the lowest mode cannot be approximated on that layer without problems. In the case of the scalar Laplace equation, four sine-shaped interpolation operators are not able to represent a constant function well.

Let us explore the updating on the last point in greater detail. To do so, we imagine an idealized situation to prove the following rather trivial

> **Theorem:** We have an algorithm consisting of two parts. The first part is able to eliminate completely all components of the error except one single mode $\mathbf{e}$ which need not be an eigenmode of the problem-operator. The second updating then consists of an updating on the last point using an interpolation operator $\mathcal{A}$ (we omit superscripts here) and Galerkin choice for the other operators.
>
> We can split the bad-converging mode $\mathbf{e}$ into two D-orthogonal parts[8]:
>
> $$\mathbf{e} = \mathcal{A}\mathbf{e}' + \mathbf{f} \quad , \text{ with } \langle \mathcal{A}\mathbf{e}', \mathbf{D}\mathbf{f}\rangle = 0 \quad . \tag{43}$$
>
> Then the iteration matrix $\mathbf{M}$ of the full algorithm consisting of both steps has the (squared) energy norm (with respect to $\mathbf{D}$)
>
> $$\|\mathbf{M}\|_{\mathbf{D}}^2 = \frac{\langle \mathbf{f}, \mathbf{D}\mathbf{f}\rangle}{\langle \mathbf{e}, \mathbf{D}\mathbf{e}\rangle} \quad . \tag{44}$$

Proof: The energy norm is defined as

$$\|\mathbf{M}\|_{\mathbf{D}}^2 = \sup_{\boldsymbol{\xi}} \frac{\langle \mathbf{M}\boldsymbol{\xi}, \mathbf{D}\mathbf{M}\boldsymbol{\xi}\rangle}{\langle \boldsymbol{\xi}, \mathbf{D}\boldsymbol{\xi}\rangle} \quad . \tag{45}$$

We can consider the algorithm as a special kind of two-grid algorithm: In the smoothing part all modes except one are eliminated, in the last point updating the equation is blocked and solved exactly (because the block lattice consists of only one point). Therefore the iteration matrix $\mathbf{M}$ can be written as a usual two-grid iteration matrix, see [59]. Here $\mathbf{S}$ is the iteration matrix of the first part of the algorithm and $\mathbf{D}'$ is the blocked operator. We get

$$\mathbf{M} = (1 - \mathcal{A}\mathbf{D}'^{-1}\mathcal{A}^*\mathbf{D})\mathbf{S} \quad . \tag{46}$$

[8] We have been very careful here to strictly obey the warnings given by Sokal, see 4.3.1. Therefore we always use the norm and the scalar product involving the problem operator $\mathbf{D}$.

As $\mathbf{S}$ eliminates all parts of an arbitrary error except the mode $\mathbf{e}$, it is clear that the supremum in the definition will be reached for $\boldsymbol{\xi} = \mathbf{e}$. $\mathbf{S}$ does not affect $\mathbf{e}$. So we get

$$\begin{aligned}
\mathbf{M}\mathbf{e} &= (1 - \mathcal{A}\mathbf{D}'^{-1}\mathcal{A}^*\mathbf{D})\mathbf{e} && (47)\\
&= \mathbf{e} - \mathcal{A}\mathbf{D}'^{-1}\mathcal{A}^*\mathbf{D}\mathbf{e} && (48)\\
&= \mathcal{A}\mathbf{e}' + \mathbf{f} - \mathcal{A}\mathbf{D}'^{-1}\mathcal{A}^*\mathbf{D}\mathcal{A}\mathbf{e}' - \mathcal{A}\mathbf{D}'^{-1}\mathcal{A}^*\mathbf{D}\mathbf{f} && (49)\\
&= \mathcal{A}\mathbf{e}' + \mathbf{f} - \mathcal{A}\mathbf{D}'^{-1}\mathbf{D}'\mathbf{e}' - \mathcal{A}\mathbf{D}'^{-1}\mathcal{A}^*\mathbf{D}\mathbf{f} && (50)\\
&= (1 - \mathcal{A}\mathbf{D}'^{-1}\mathcal{A}^*\mathbf{D})\mathbf{f} && (51)\\
&= \mathbf{f} \quad , && (52)
\end{aligned}$$

where we have made use of the Galerkin definition of the blocked operator, the split of the mode $\mathbf{e}$ and the fact that $\mathcal{A}\mathbf{D}'^{-1}\mathcal{A}^*\mathbf{D}$ is the $\mathbf{D}$-orthogonal projection onto the range of $\mathcal{A}$ (see e.g. [67]), so that because of the orthogonality of $\mathbf{f}$ the operator $1 - \mathcal{A}\mathbf{D}'^{-1}\mathcal{A}^*\mathbf{D}$ acts trivially on $\mathbf{f}$.

Thus we have

$$\|\mathbf{M}\|_{\mathbf{D}}^2 = \frac{\langle \mathbf{M}\mathbf{e}, \mathbf{D}\mathbf{M}\mathbf{e}\rangle}{\langle \mathbf{e}, \mathbf{D}\mathbf{e}\rangle} = \frac{\langle \mathbf{f}, \mathbf{D}\mathbf{f}\rangle}{\langle \mathbf{e}, \mathbf{D}\mathbf{e}\rangle} \tag{53}$$

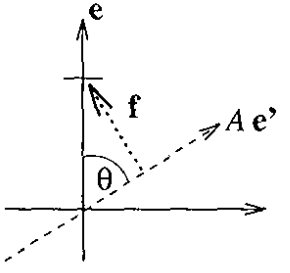
q.e.d.

We can prove this result also more intuitively: The vector $\mathbf{f}$ is nothing but the error after the coarse-grid correction. The convergence rate is governed by the quotient of the energy norms before and after this correction step.

It is also useful to look at this geometrically as we will use a similar idea later in section 9.5. The angle θ between the vector $\mathbf{e}$ and $\mathcal{A}\mathbf{e}'$ with respect to the scalar product defined by $\mathbf{D}$ is given by

$$\cos\theta = \frac{\langle \mathbf{e}, \mathbf{D}\mathcal{A}\mathbf{e}'\rangle}{\langle \mathbf{e}, \mathbf{D}\mathbf{e}\rangle^{1/2}\langle \mathcal{A}\mathbf{e}', \mathbf{D}\mathcal{A}\mathbf{e}'\rangle^{1/2}} \quad . \tag{54}$$

The reduction works by first projecting $\mathbf{e}$ onto the direction given by $\mathcal{A}$ and then taking the $\mathbf{D}$-orthogonal part of this. This orthogonal part is the vector $\mathbf{f}$: it is all that remains after the coarse-grid correction step. The length of this vector is given by $\|\mathbf{f}\|_{\mathbf{D}} = \|\mathbf{e}\|_{\mathbf{D}} \sin\theta$. The reduction factor, which is equal to the norm of the iteration matrix, is $\sin\theta$:



Using Pythagoras' theorem we get

$$\sin^2\theta = 1 - \cos^2\theta = 1 - \frac{\langle \mathbf{e}, \mathbf{D}\mathcal{A}\mathbf{e}'\rangle^2}{\langle \mathbf{e}, \mathbf{D}\mathbf{e}\rangle\langle \mathcal{A}\mathbf{e}', \mathbf{D}\mathcal{A}\mathbf{e}'\rangle} \quad , \tag{55}$$

and inserting the split of the vector $\mathbf{e}$ and again using the orthogonality property, we finally arrive at

$$\sin^2\theta = 1 - \frac{(\langle \mathcal{A}\mathbf{e}', \mathbf{D}\mathcal{A}\mathbf{e}'\rangle + \langle \mathbf{f}, \mathbf{D}\mathcal{A}\mathbf{e}'\rangle)^2}{\langle \mathbf{e}, \mathbf{D}\mathbf{e}\rangle\langle \mathcal{A}\mathbf{e}', \mathbf{D}\mathcal{A}\mathbf{e}'\rangle} \tag{56}$$

$$= \frac{\langle \mathbf{e}, \mathbf{De}\rangle - \langle \mathcal{A}\mathbf{e}', \mathbf{D}\mathcal{A}\mathbf{e}'\rangle}{\langle \mathbf{e}, \mathbf{De}\rangle} \tag{57}$$

$$= \frac{\langle \mathbf{f}, \mathbf{Df}\rangle}{\langle \mathbf{e}, \mathbf{De}\rangle} . \tag{58}$$

What are the implications of this theorem? First it must be understood that the energy norm of the iteration matrix will be equal to the spectral radius provided the matrix $\mathbf{S}$ is $\mathbf{D}$-symmetric, i.e. $\mathbf{S}^*\mathbf{D} = \mathbf{DS}$. This is true when the operator $\mathbf{D}$ does not mix the mode $\mathbf{e}$ with the other modes. We can regard $\mathbf{e}$ as an eigenmode of $\mathbf{D}$ because the matrix $\mathbf{S}$ provides us with an identification of the mode $\mathbf{e}$ with the corresponding mode in the dual space. Hence the energy norm directly tells us about the convergence rate of the algorithm.

Choosing $\mathbf{e}$ as the zero-mode, the theorem shows how important the correct interpolation of this mode is. The closer the range of the interpolation operator $\mathcal{A}$ is to the zero-mode and the smaller the energy norm of the residual part $\mathbf{f}$ the better the convergence will be. (In the limiting case where the two are identical, the difference vector is zero and the error of the zero-mode is eliminated perfectly, as expected.) It is not only important that the zero-mode is approximated well by the interpolation operator on the last point, the convergence will also be better when the difference vector between zero-mode and the mode used for the interpolation has a small energy norm and is as smooth as possible. This is another reason why inverse iteration is a good method to find the zero-mode: even if the approximation is not perfect, the difference vector will have the largest contributions from other low-lying modes and therefore have a small energy norm.

Thus we can expect that the convergence rate with respect only to the last point updating will not depend strongly on the distribution of the eigenvalues: If the lowest eigenvalue is much smaller than the second-lowest, the inverse iteration will converge fast and $\mathcal{A}$ will be close to the zero-mode after a few iterations. If, however, the lowest eigenvalue is larger than the difference between lowest and second-lowest eigenvalue, the interpolation operator $\mathcal{A}$ will contain contributions also from the higher modes. In this case, the difference vector $\mathbf{f}$ will not be small. Its energy-norm, however, will be comparable to the energy-norm of the lowest eigenvector and the convergence will still be good. This explains our findings from section 5.1 that there is only an appreciable dependence of the number of iterations for the interpolation operator on δm^2, when δm^2 is of order 1, even if the prerequisites of the theorem are not exactly met.

The theorem might also serve to explain a finding by Kalkreuter [69]. He found that it is possible to eliminate critical slowing down in a two-grid algorithm at $\beta = \infty$ even with interpolation operators that are not able to represent the zero-mode (which is a constant in this case) exactly, but only approximately. In the light of our theorem, this could be understood if the difference vector has a small energy norm. This, however, has not been tested so far.

Thus we have seen the importance of the correct treatment of the zero-mode. One might, however, think of another method to perform the updating on the last point than calculating a good approximation to the zero-mode. Such a method was proposed by Kalkreuter [69].

He uses a simple rescaling of the approximate solution $\tilde{\xi}$ in addition to a multigrid or a relaxation algorithm to improve the convergence. This method eliminates critical slowing down in the simplest model problem, the Laplace equation on a two-point grid. This rescaling amounts to using the approximate solution $\tilde{\xi}$ itself as interpolation operator $\mathcal{A}$.

The motivation for this updating can be found in the following argument: Imagine we want to solve the equation $(\mathbf{D} + \delta m^2)\xi = \mathbf{f}$ for smaller and smaller values of the lowest eigenvalue δm^2 (let $\mathbf{D}$ already be a critical operator) but with fixed righthandside. The solution ξ will then become smoother and smoother as δm^2 is decreased. To see this, we can diagonalize the operator $\mathbf{D}$. Then we get $\xi = (\mathbf{D}^{-1} + \delta m^2)\mathbf{f}$. The smaller the lowest eigenvalue of $(\mathbf{D} + \delta m^2)$ becomes, the larger the contribution of the corresponding zero-mode to ξ gets. Therefore ξ is smooth and can itself be used as an interpolation operator. (If we wanted to be strict about the eigenvectors of $\mathbf{D}$ which do not really exist, we could insert the matrix $\mathbf{B}$ at the appropriate places, as described in section 4.3.1.)

Kalkreuter used this method *in addition* to a usual multigrid or relaxation method. He found that there is no strong improvement for a multigrid algorithm, but for standard local relaxation the asymptotical critical slowing down (i.e. critical slowing down for fixed grid size and infinitely many iterations) was eliminated. This is what we expect for a method that treats the zero-mode of the problem correctly, as we saw in section 5.1. We also expect that for increasing grid sizes critical slowing down is still present because the second-lowest mode is crucial for this; this in agreement with Kalkreuter's results.

Can we use this method of updating on the last point for our algorithm which would be cheaper than calculating the lowest eigenvector directly? In fact we could do so, but only when the lowest eigenvalue of the operator is small enough.

We tried this method for the Laplace equation both with and without a gauge field. We also tried it for a true multigrid with linear interpolation operators, again for the scalar Laplace equation. In all cases, the righthandside $\mathbf{f}$ of the equation was chosen to be a δ-function. (The righthandside is important now since the smoothness of the solution depends on it.) For the scalar Laplace equation, the convergence was good for large values of the critical parameter δm^2, where relaxation itself is a good solver, and for very small values of δm^2. For these small values, the solution becomes a very smooth function which is nearly constant. For intermediate values of the mass, however, the convergence time increased. In the case of the Laplace equation with gauge field, we sometimes observed convergence times $\tau \approx 10$ on a 32^2-lattice at $\beta = 1$ and $\delta m^2 = 10^{-4}$, instead of the usual $\tau \approx 1$. So we see that this method is not always a good alternative to the usual last-point updating. Kalkreuther's rescaling method can only be successful when the approximate solution is smooth; the usual multigrid philosophy however assumes that it is the *error* that is smooth. As we saw, these two notions agree only for small critical parameters.

If we are interested in doing production runs for a problem with a small critical parameter, the method might be useful, but as we want to study the behavior of the ISU-algorithm as a function of the critical parameter Kalkreuter's method would add an additional effect that complicates the results. We therefore chose always to use the inverse iteration method even on the last point.

5.4 Killing Instantons

Many algorithms for solving the Dirac equations become problematic in the presence of instantons. The Atiyah-Singer theorem states that in this case the spectrum in the continuum will possess exact zero-modes [32]; these become modes with extremely small eigenvalues on the lattice [33]. In this case, the condition number of the Dirac operator becomes very large and the problem is ill-posed. In [75] this problem is investigated in detail for the Parallel Transported Multigrid.

Figure 9 shows the lower part of the spectrum of the squared Dirac operator on an 18^2-lattice at $\beta = 10$ for different instanton charges. Clearly the Atiyah-Singer theorem is nicely reflected on the lattice.

In this section, we want to use the principle of indirect elimination to show how an algorithm which converges well in the absence of instantons can be adapted to a case with instantons.

We will choose the ISU algorithm on small lattices and at quite large values of β for a U(1) gauge field as an example, but the method could be applied to a Conjugate Gradient algorithm as well.

The idea is very simple: If m zero-modes are present, add m interpolation operators to the algorithm which live on the whole lattice. This means that we add a block-lattice consisting of one point and calculate as many interpolation operators on this layer as are needed to eliminate the bad-converging modes as discussed in the last section.

The idea for this method arose when we observed that the ISU algorithm converged well for instanton charges 0 or ± 1 at large β but badly for larger instanton charges. The reason is that the algorithm in its standard form contains one interpolation operator on the last point (which approximates the zero-mode) and so it is able to eliminate one zero-mode, but not more. (The instanton modes seem not to be approximable on the other layers.) Therefore a simple generalization seems possible.

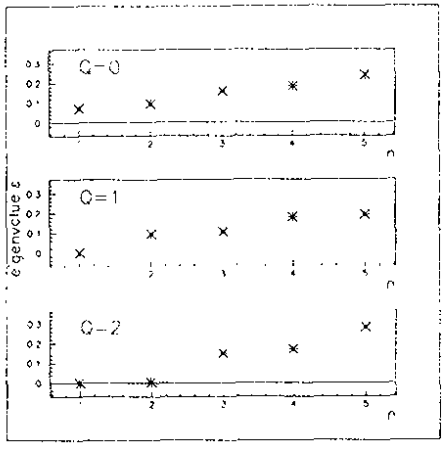


Figure 9: The Atiyah-Singer theorem on the lattice: Shown are the five lowest eigenvalues of the staggered squared Dirac operator in a U(1) gauge field at $\beta = 10$ on an 18^2-lattice for different topological charges Q.

There are two possible ways of doing this, either using the eigenvalue approach or the solution of the equation (37). Let us discuss the second version, the first is similar.

One tries to solve the equation $\mathbf{D}A^{[0,N]} = 0$ with the given algorithm. As the algorithm eliminates all other modes quickly, the approximate solution will converge against a linear combination of the zero-modes. Then we start the procedure again, but now orthogonalizing the approximate solution to the interpolation operator we already know successively for all m bad-converging modes. (As the instanton charge can be easily measured, one usually knows beforehand how many operators are needed; if one does not for some reason, a dynamical approach can be chosen: Simply proceed calculating the next interpolation operator until the convergence rate of the trivial equation becomes good enough.)

The overall work for this procedure is proportional to the square of the number of bad-converging modes, as is the work of actually applying the interpolation operators to eliminate them. (The number gets squared there because of the need to calculate an effective operator on the last point layer. This would not be true for exact eigenmodes, but as we calculate them only approximately it seems better to calculate the full effective operator and not only its diagonal elements. However, the effective operator only needs to be calculated once for each configuration.) This restricts the method to cases where the number of bad-converging modes is not too large, which usually is the case for instanton charges.

Figure 10 shows the performance of the usual ISU method compared to the improved version for the Dirac operator in a U(1) gauge field with different instanton charges. The improved version of ISU used a number of interpolation operators on the last point equal to the instanton charge which equals the number of bad-converging modes. The data were generated on an 18^2-lattice at $\beta = 10$. For this high value of β, the convergence in the absence of instantons is good, as can be seen from the value at $Q = 1$. The standard method works well for instanton charge 0 or 1, as explained above, and its sensitivity to higher instanton charges is striking. The improved method shows no dependence on the instanton charge, the convergence is good in all cases. Note also that the standard deviation

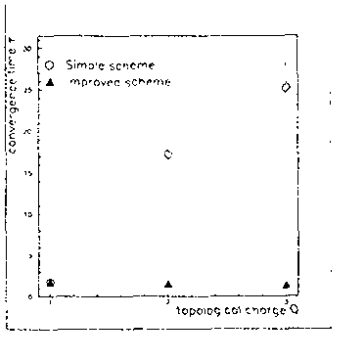


Figure 10: Performance of the standard and the improved ISU algorithm for the eliminating of instanton modes. The data were generated on an 18^2-lattice at $\beta = 10$ with a U(1) gauge field. The improved algorithm uses Q interpolation operators on the last point to eliminate the almost-zero modes. (The number of configurations evaluated for the different topological charges was 217, 113, and 22.)

is much higher for the usual method because it is affected by fluctuations in the eigenvalues of the bad-converging modes.

Clearly, the improved method is superior—the cost of calculating the instanton modes is about 10 iterations for each instanton plus the cost of the orthogonalization, whereas the saving in the solution of the final equations is of the order of hundreds of iterations depending on how much we want to reduce the residual.

6 ISU IN DIFFERENT GUISES

Wherein we take a small side tour and see that the ISU algorithm can be connected with many fashionable ideas.

6.1 Connections to Wavelets

In the last few years, the new method of *wavelet transforms* has gained considerable attention from various fields. This method has much in common with multigrid methods, at least in general spirit. In this section we want to show some relations between the ISU algorithm and the wavelet transform technique.

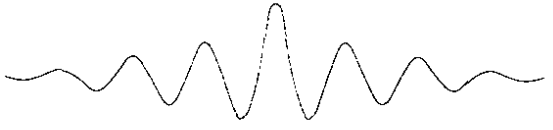
Up to now, not many introductions to the field exist [86, 87]. Hence we give a short overview here.

6.1.1 Wavelets—the Essentials

The wavelet transform is a generalization of the usual Fourier transformation. Fourier transformations suffer from a major drawback: To define the Fourier transform of a function $f(t)$ (usually thought of as a signal), it must be known over the whole time domain $-\infty < t < \infty$. The Fourier analysis forces us to choose either the frequency picture $\tilde{f}(\omega)$, where all information about the time dependence is obscured, or the time picture $f(t)$, yielding no information about the frequencies. This can be seen clearly for a function like this:

Because of the sudden on- and offset of the function, a Fourier transformation would contain contributions from all frequencies, despite the obvious fact that the signal is either zero or a simple sine wave.

The cure for this problem is to define a transformation using a time window which decomposes the time axis into several parts. For the *integral wavelet-transform* a single function $\psi(t)$, the *wavelet*, is chosen to perform this task. Of course it is desirable to be able to choose different sizes for the time window (which will give rise to different sized frequency windows through the uncertainty relation) and different locations; for this reason, the wavelet can be scaled and translated. A typical wavelet might look like this:



Out of this one function (also called the "mother wavelet"), we create all wavelets we need through dilation and translation. We can define the *wavelet transform* as

$$(W_\psi f)(b,a) = \frac{1}{\sqrt{a}} \int dt\, f(t) \psi^* \left(\frac{t-b}{a} \right) \quad . \tag{59}$$

a is the dilatation scale of the transformation and b controls the translation.

Of course such a wavelet transform is over-complete. One can restrict the attention to *dyadic wavelet transforms*, using only a restricted set of scaled and dilated wavelets:

$$\psi_j^{(i)}(t) = \frac{1}{\sqrt{2^i}} \psi \left(\frac{t - 2^i j}{2^i} \right) \quad . \tag{60}$$

Here 2^i is the dilatation scale and $2^i j$ is the amount of translation.

What are the properties of a "good" wavelet?

The major requirement is that the wavelet function is "admissible" which means that the Fourier transform of the wavelet decreases sufficiently fast when the frequency approaches zero. A simple consequence of this is that the wavelet has zero mean (because its frequency-zero component vanishes). If this is true, the signal can be reconstructed from the wavelet transform without difficulties. Another important requirement is that the wavelet function must be localized to yield a good time-window. It should also be chosen such that the signal can be recovered exactly from the wavelet transform

To see the connection to the multigrid method, we must go a step further and take a look at the *multi-resolution analysis*. Here the idea is to decompose the signal into parts that are smooth on certain scales, similarly to our unigrid analysis. To do this, one has to complete the wavelets parentship and must introduce the "wavelets father" ϕ, also called the scaling function which differs from the wavelet in that its mean is not zero, but one: $\int dt\, \phi(t) = 1$. This father function can again be translated and dilated to yield a family of functions $\phi_{ij}(t) = 2^{j/2}\phi(2^j t - i)$, these should be orthogonal to the wavelets $\psi_{j'}^{(i')}$ for $j' > j$ and for all i, i'.

In the language of signal processing, the father wavelet and its children are low-pass filters, letting through information about the smooth scales, whereas the mother wavelet generates high-pass filters, which contain the information about the fluctuations of the functions.

One of the mathematical advantages of this method is that *any* function can be decomposed using both wavelet parts, whereas the integral wavelet transform defined above is only meaningful if the function decays fast enough at infinity.

The decomposition of a function $f(t)$ now becomes

$$f(t) = \sum_{i=-\infty}^{+\infty} \langle \phi_{0i}, f \rangle \phi(t-i) + \sum_{j=0}^{+\infty} \sum_{i=-\infty}^{+\infty} \langle \psi_j^{(i)}, f \rangle \psi_j^{(i)}(t) \quad . \tag{61}$$

The first sum approximates the smooth part of f at the largest scale, whereas the second adds finer and finer details on the scale 2^{-j}. As a trivial example, take the constant function $f(t) = 1$. For this we have $\langle \phi_{0i}, f \rangle = 1$ and $\langle \psi_j^{(i)}, f \rangle = 0$, clearly showing that there are no non-smooth parts on any scale.

The multi-resolution analysis can now be used to reconstruct the function on any desired scale by splitting off the high-frequency part, using the wavelets $\psi_j^{(i)}$, and keeping the remainder on the larger scales. As only the ideas seem important here, we do not elaborate further on this.

6.1.2 Wavelets and ISU

By now it should be intuitively clear that multigrid methods and the wavelet analysis have much in common even if their mathematical goals are quite different: In a wavelet analysis we try to decompose a given function into its parts living on several scales, in a multigrid algorithm we try to eliminate the contributions to the error on all scales. The common feature of both methods is their multiscale nature.

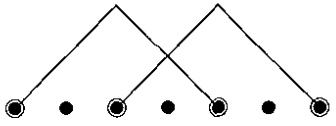
In the multigrid algorithm, the interpolation operator plays the role of the father wavelet: It is used to transport the information about the smooth parts of the mode to the next scale. Both the scaling function and the interpolation operator have to be localized objects on the relevant scales.

It seems as if there is nothing that can replace the mother wavelet in the multigrid method since no other function is involved. But in fact, the role of the mother wavelet is played by the *relaxation*. As our goal is not to be able to *reconstruct* the error function through the multigrid process but to *eliminate* it on all scales, we need not store its high-frequency parts. So the two processes can be compared like this: In a wavelet multi-resolution analysis we split off the high-frequency part on a certain scale and transport the remaining low-frequency part to the next scale [9], whereas in multigrid

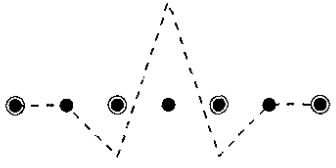
[9] This is the same approach taken in the renormalization group analysis: Here one splits the fundamental quantum

we eliminate the high-frequency part and transport the rest to the coarser grid.

Let us study this in a little bit more detail: What kind of functions are to be eliminated by the relaxation? Obviously all those that can not be represented on the coarse grid or, in other words, all modes lying in the *nullspace* of the interpolation operators, see section 4.1.3. Consider a simple example: A multigrid method for the one-dimensional scalar Laplace equation. (For a unigrid, the pictures will be similar.) We take full weighting operators that look like this:



In the wavelet context, this are our scaling functions. We now construct functions that are in the nullspace of these scaling functions: To do so we have to ensure that each operator will yield zero when applied to these functions. In addition, we want the functions to be localized to look like wavelets. This is what we get from this simple prescription:



The resemblance to a wavelet is indeed striking. So we just tried to find a localized basis for the nullspace of the interpolation operators and arrived at a wavelet-like function. Observe that our function is indeed a good wavelet: It fulfills the admissibility condition (the sum over all values is zero) and it is localized. It is also possible to reconstruct any vector from the wavelet and its known representation on the coarse grid because the range of the interpolation operators and their nullspace together are the whole vector space. Let us stress here that all this is actually well-known, the nullspace analysis can be found in [60, chapter 5], except that the word "wavelet" is never used there.

What was said so far is true for a multigrid. For a unigrid, the usual difference occurs: The function transported to the coarser scale only contains the information on this scale, not the information on all the larger scales as well. Up to now this method has no counterpart in wavelet analysis, which is also true for another feature of our ISU algorithm: ISU is intended to deal with problems *without translational invariance*. Where then is the scaling and translating which was so essential for the definition of the wavelet transform?

The answer is that although the interpolation operators themselves are not translationally invariant, they are solutions of equations which evolve from each other through scaling and translation. The basic equation always involves the problem operator restricted to a certain domain, namely the block $\{x\}$. But the blocks *can* be generated from one "mother block" by translation and dilation, at least for the usual definition of the algorithm where the blocks are fixed. So the algorithm uses translation and dilatation in a more abstract sense.

6.2 ISU as a Neural Net

(This section and section 6.4 are based on a joint paper with G. Mack and M. Speh [90].)

In this section we will reformulate the ISU-algorithm in the language of neural nets, see [90] for further details. As many good introductions to this subject exist, for example [88, 89], we do not repeat more than the barest fundamentals here.

field into its smooth part, called the *block-spin*, and a remainder, the fluctuation field. The fundamental field can be reconstructed from these two parts [3, 63].

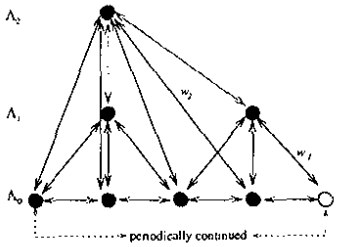


Figure 11: Topology of the ISU algorithm considered as a neural net for the case of a three-grid.

A neural net consists of several layers of so-called neurons. The information transport from one neuron to the next is governed by the connection strength between them which can be regarded as directed link with a numerical value. The link between points that are not to be connected gets the value zero. Each neuron takes the values presented to it by its input connections and computes from them and its internal state (which is another number) an output value which is then passed to other neurons of the network. The network is given an input pattern into a special layer of neurons, the input layer. This pattern is then propagated through the network.

Neural networks start with a learning phase in which they are fed with test patterns for which the desired output is known. The connection strengths of the network are then adapted in such a way as to return this output value until the network is stable, i.e. has learned to do its task. There are many standard learning algorithms, however, none of them seems to be suitable for the problem of calculating good interpolation operators. The computer time needed for, e.g., the back-propagation method grows much too fast with the number of layers in the network and therefore would result in critical slowing down.

Let us now describe the ISU algorithm in this language. This could be done formally by comparing the abstract definition of a neural network (as given in [88]) with our description of the algorithm, but as this comparison is rather straightforward we simply present the general idea.

We can identify the different layers in the multigrid with the layers of the network and the neurons with the (block-)grid points. The connection strengths between them correspond to the interpolation operators for neurons on different layers and the operator D_k if both neurons are on layer Λ^k. Note that in contrast to normal neural networks, the values of the connection strengths and states are not real numbers, but multiples of SU(2)-matrices. The topology of the network is shown in figure 11.

Our algorithm starts by learning the right connection strengths between the layers, i.e. by computing the interpolation operators in the described iterative way. Having learned good connection strengths on smaller scales (lower levels) this new knowledge is the used on the coarser layers. This is in fact a problem of cognition: We are looking for shapes into which the low-lying modes of the system can be decomposed. The reason that the algorithm is not a standard neural net is that we do not know the shapes we want to decompose; we only know that they are smooth.

After this learning phase is completed, the network will return a solution of the equation (an improved guess) when an initial guess is presented to its input layer.

6.3 ISU and Solomon's Micro's and Macro's

In a recent paper [91] Solomon proposed to look at the problem of critical slowing down not only as a nuisance to be overcome but also as a concept to quantify our understanding of the large-scale dynamics of physical systems. He calls the large-scale modes the *macro's* of the system, whereas the fundamental, short-scale degrees of freedom are called the *micro's*. The basic question is how

Type of understanding	CPU-time	example
analytical	--	free theory
no volume-factor	1	see [72]
no critical slowing down	L^d	Ordered differential equation with multigrid.
ISU-like	$L^d \ln^2 L$	ISU for the Laplace-equation (with or without gauge field background)
Polynomial critical slowing down	$L^d L^z$	Local relaxation algorithms
exponential critical slowing down	$L^d e^B$	Monte-Carlo simulation near first-order phase transition
not computable	∞	Quantum gravity (summing over topologies in four dimensions)

Table 2: Different grades of algorithmical understanding and the corresponding CPU times.

the macro-dynamic arises from the micro-dynamics. Having answered this, we can try to develop an algorithm with reduced critical slowing down.

The general idea behind this has already been encountered in chapter 3, when we looked at the example of the Ising model: Understanding the structure of the large-scale, slow-converging modes we were able to create an algorithm with reduced critical slowing down. Solomon proposes to reverse this connection: If we know an algorithm with no or reduced critical slowing down, this means that we understand (some of) the large-scale properties of the model. Therefore we can use the amount of critical slowing down as a measure of how good we understand the slow modes of the system.

The best understanding of course is the analytical understanding; the worst case is when the problem is not even computable as in quantum gravity. Table 2 shows the whole spectrum of possibilities.

We can see again—as we already did in section 4.1.4—that the simple knowledge of all bad-converging modes in the algorithm is not sufficient for a true understanding of the large-scale structure. The computer time for a method relying on this knowledge would be $\tau \propto (L^d)^2$, so we can now quantify how bad such an algorithm would be.

However, we believe that two caveats must be stated here. Both of them are due to the fact that the definition is algorithm-dependent:

1. It might happen that the connection between slow modes in the algorithm and large-scale modes is not appropriate. The definition of a slow mode depends on the (local) algorithm chosen, so the identification of the two different concepts will only be correct if the algorithm itself represents the correct physical dynamics. This is exactly the point made by Sokal [67], as we saw in section 4.3.1. In other words, the algorithm usually chooses a basis in which it works, and only if this basis is natural on physical grounds the two notions will agree.

2. What happens if we work with a black-box algorithm, i.e. an algorithm suited to deal with many different problems? Having no critical slowing down in applying this algorithm does not mean that we have a true understanding of the problem. The algorithm correctly identifies the large-scale modes, but it might be difficult to extract this knowledge and transfer it to the physicist. To do this, visualization might be an important tool, as Solomon himself pointed out. We will see an example for this in section 8.1 and section 9.2.

Let us now take a look at the ISU algorithm in this light: In the case of bosons we have eliminated critical slowing down completely. Nevertheless our understanding is not as good as it were for a true multigrid algorithm without critical slowing down and the difference is the additional factor of $\ln^2 L$. This is sensible because in a unigrid the fundamental length scale can never be ignored. In a true

multigrid each scale contains all relevant information for this and all the larger scales, but this is not true for a unigrid. Here each layer only represents the degrees of freedom on this scale. The intermediate scales do not contain the whole macro-dynamics for this and all larger scales. We see that Solomon's ideas correctly reflect the differences between multigrid and unigrid methods.

The ISU algorithm is intended to automatize the identification of the bad-converging modes, especially if we do not use the eigenvalue version but the one based directly on the principle of indirect elimination. The algorithm calculates the shapes of the slow modes and uses them directly to reduce critical slowing down; there is no need to understand where these modes come from. It is only necessary that the modes exist on different length scales and are nested, so that the number of bad-converging modes which live on the coarser scales gets smaller the coarser the scale becomes. This is exactly in the spirit of Solomon because on larger scales there should be less degrees of freedom (macro's in his language).

6.4 *ISU and Universal Dynamics*

In [92] Mack proposed a general framework for the formulation of complex systems. The ISU algorithm can be viewed as a special example of Mack's Universal Dynamics for a problem with a linear structure. In this sense the algorithm is a general problem solver.

Let us shortly review the basics of Universal Dynamics:

One pictures complex adaptive systems as composed of agents with relations between them which determine their interactions. These relations are assumed to be composable which leads to a generalization of gauge theories.

Mathematically, one pictures agents as objects X of a category, and relations as arrows $f : X \mapsto Y$ of a category. The basic postulates of mathematical category theory are

1. There are identity arrows $\iota : X \mapsto X$. $(\iota \circ f = f = f \circ \iota)$.

2. Arrows can be composed.

$$f : X \mapsto Y, g : Y \mapsto Z \quad \text{defines} \quad g \circ f : X \mapsto Y$$

Categories can be objects of new categories. This makes the setup suitable for the analysis of structures on multiple scales.

Often one has the additional property that a relation f of X to Y defines a uniquely specified possible relation f^* of Y to X. In this case we speak of a *$*$-category*. f^* is called the adjoint of f. It is supposed to be unique but may be absent in a particular category.

Finally there is the important notion of Locality: In a local category, certain arrows called links are declared as fundamental. Other arrows can be composed from them: from links $b_i : X_{i-1} \mapsto X_i$ one gets paths; these define arrows $C : X_0 \mapsto X_n, C = b_n \circ \cdots \circ b_2 \circ b_1$. The gauge theory aspect resides in the possible path dependence of the arrow between given objects X_0 and X_n.

According to hypothesis, the state of a system at a time t is described by a local $*$-category K_t. By definition, a universal dynamics (of first order) is a rule which determines the state K_{t+1} of the system after one time step, given any local category K_t whatsoever as initial state. Here K_t and K_{t+1} are regarded as the same category in different states, not as different categories.

Universal dynamics of first order consists of local moves of the following kinds which are defined for every local ($*$-) category:

1. *death* of an object (or arrow)

2. *replication of* an object (or arrow)

3. *fusion* of indistinguishables (inverse of previous move)

4. *recovery of missing adjoint* in a *-category.

5. *declaration as fundamental* of a composite arrow

There are actually two kinds of replication of an object, $2a$ with, $2b$ without replication of its attached arrows.

The ISU-algorithm can be formulated completely in the language of Universal Dynamics with one exception: The original category consists only of Λ^0 with objects $f(z)$ labeled by z and connection strengths specified by $\mathbf{D}$. The other sites x of the multigrid can be thought of as objects grown by replication of those objects already present. The updating steps consist in composition of arrows, declaring such composites as fundamental, and adding them to links already present. This last step is the exception mentioned above because linearity is used to add corrections to already calculated terms, e.g., in the updating of the field or the interpolation operators. It is also used in the calculation of effective operators.

We see that the ISU algorithm is a general problem solving strategy that will hopefully be applicable to many different kinds of models.

7 Analyzing ISU—Part I

Wherein we try to understand why ISU did not fulfill our hopes, take a look at how it actually works, and make a first attempt at an explanation which is, however, not correct.

7.1 The Need for New Analyzing Tools

As we have seen in chapter 5, the ISU algorithm performs very differently for the Laplace and the Dirac equation. The understanding of this difference is the key to an improvement to our algorithm which shows no critical slowing down even for the Dirac case. Therefore one would like to use multigrid analyzing tools to understand this behavior.

However, the standard methods to analyze multigrid algorithms is to perform a smoothing analysis or an idealized twogrid-algorithm. *In a smoothing analysis one studies the behavior of the chosen* smoother for the given problem, usually by analytical calculations. The twogrid analysis consists of solving the blocked equation on the first block lattice exactly and, thereby, studying whether the interpolation operators $\mathcal{A}^{[0\,1]}$ are appropriate.

Both methods are not directly applicable for ISU. Smoothing analysis for this problem probably can not be done analytically without great effort. So we do a numerical study of the smoothing algorithm in the next section. The twogrid analysis can not be used for a unigrid, as we saw in section 4.1.3, because all interpolation operators are crucial, not only $\mathcal{A}^{[0\,1]}$.

In this and the following two chapters we try to understand the problems of ISU by developing and using analyzing tools that can replace the standard methods. These tools are not restricted to the problem of gauge theories or to the ISU algorithm; they are in fact quite general. Most of these tools involve the calculation of eigenmodes of the fundamental operator—as there is no translational invariance, Fourier decomposition is of no use for this problem. Furthermore, in this chapter, as well as in chapter 9 we explain some analysis methods which do not lead to conclusive answers. The reason for discussing them nevertheless is that studying these methods and the reasons for their failure increases the understanding of the algorithm.

7.2 Monitoring ISU

The first step in analyzing the behavior of the ISU-algorithm is to convince ourselves that it works as expected. The basic idea stated that the bad-converging modes of a standard relaxation scheme *should be the modes corresponding to the lowest eigenvalues of the operator* $\mathbf{D}$. This is not always true as has been pointed out in section 4.3.1, so we have to check whether it is the case for the Laplacian and the Dirac operator[10].

To this end, we calculated all eigenmodes of the operator (on a rather small grid of size 16^2 and 18^2, resp.) and expanded every function on the grid into these eigenmodes. Note that owing to the hermiticity of the operator the modes are orthogonal, so the expansion is unique. In the trivial case ($\beta = \infty$) this corresponds to a Fourier analysis. (Due to this, we will call the eigenmodes with high eigenvalues the high-frequency-modes, etc. We will also speak of low and high eigenmodes.)

Now we calculated a solution of the propagator equation to a high accuracy and then started the iteration anew knowing the error which could then be expanded into the eigenmodes. We always started with an initial guess whose error has constant expansion coefficients, so we could see directly which modes were causing trouble and which were harmless. We call this method *monitoring the error*.

In more mathematical terms this means that we apply the iteration matrix of the problem to an initial guess. As the eigenfunctions of the operator $\mathbf{D}$ are not eigenfunctions of the iteration matrix

[10] Remember that by this we always mean the squared Dirac operator because only for this we can use the ISU algorithm.

itself the modes get mixed. For a full analysis of the iteration matrix we should therefore do this for n linearly independent initial guesses. However we may expect that the asymptotic shape of the error will not depend to strongly on the initial guess because the bad-converging modes will always make the strongest contribution. This is in fact similar to inverse iteration: if we apply the iteration matrix again and again to the initial guess it will project onto the space of the eigenvectors with the largest eigenvalues (not the smallest because we do direct, not inverse iteration).

A similar method for the Monte-Carlo simulation of a two-dimensional XY-model has been developed in [28], where the authors studied the behavior of the Fourier modes (which are the eigenmodes of the system) for a local and a global updating procedure.

Let us first study the performance of the simple Gauß-Seidel relaxation. This is the replacement of the standard smoothing analysis mentioned above. Figure 12 shows the development of the error on a 16^2-lattice at $\beta = 1$ for the Laplacian case for a mass-value $\delta m^2 = 10^{-4}$. Clearly the higher components of the error are eliminated quickly, and the lower the eigenmode the slower its reduction through the relaxation. *This justifies our intuition we got from the study of ordered systems in section 4.1.* The picture for the Dirac operator is nearly identical to this.

Next we study the complete ISU algorithm. Let us first consider the case of the Laplacian where it performs well. Again we take $\beta = 1$ and work on a 16^2-lattice with $\delta m^2 = 10^{-4}$ performing a complete $V(1,1)$-cycle, see figure 13. The algorithm exactly does what we expect, eliminating lower and lower frequency parts as it proceeds to coarser and coarser grids. The updating on the last point in the middle of the cycle brings the zero-mode component of the error exactly to zero. Note that relaxing on intermediate grids increases the error of the low-frequency parts, so successively the error "is transferred to the zero-mode" where it is then eliminated.

Finally, we can see that the algorithm is not so effective in eliminating the low-frequency parts as is the usual relaxation in reducing the high frequencies. This agrees with the slightly worse convergence time observed for small values of δm^2 compared to the rate for large δm^2 where the relaxation is an effective solver because the problem is not critical.

Now let us examine the fermionic case where the algorithm did not work so well. We take $\beta = 1$ and $\delta m^2 = 2 \cdot 10^{-4}$ on an 18^2-lattice with a $V(3,3)$-cycle. Figure 14 shows the result of the monitoring. Obviously the problem is due to the modes of the lowest frequencies. These are not reduced appropriately. In fact, seeing this was the motivation to introduce the additional layer into the unigrid, but even with the additional layer the problem is only softened, not cured.

7.3 The Eigenvalue Spectra

The first thing we may look at to understand the different behavior of the two cases are the eigenvalue spectra of the Laplace and the Dirac operator. For relaxation and Conjugate Gradient algorithms, the convergence rates are determined by the ratio of the highest and the lowest eigenvalue.

Figure 15 shows the spectra in both cases. To be able to show a comparison between quenched and *unquenched gauge fields for the Dirac operator (see section 2.1), we used U(1)-gauge fields in this case* for which a Hybrid Monte Carlo program was available [29]. (The picture for SU(2) is similar.) It can be seen clearly that the Laplace operator has fewer low-lying eigenvalues and it would be tempting to take this as an explanation for the better convergence of the algorithm.

This idea is strengthened by looking at the eigenvalues of the Dirac operator in an unquenched gauge field. It is expected [38] that its spectrum has fewer low-lying eigenvalues because the fermion determinant in the path integral suppresses such configurations.[11] The convergence of our algorithm is in fact much better in the unquenched case: we have $\tau = 29.7$ for quenched and $\tau = 9.9$ for unquenched U(1)-configurations on grid sizes 18^2. (20 configurations for each case where generated.) So we might draw the conclusion that the higher the number of low-lying eigenvalues the worse the convergence of ISU.

[11] One has to be careful here since the quenched and unquenched configurations at the same value of β are not physically equivalent. For a truly physical comparison, renormalization effects would have to be taken into account.

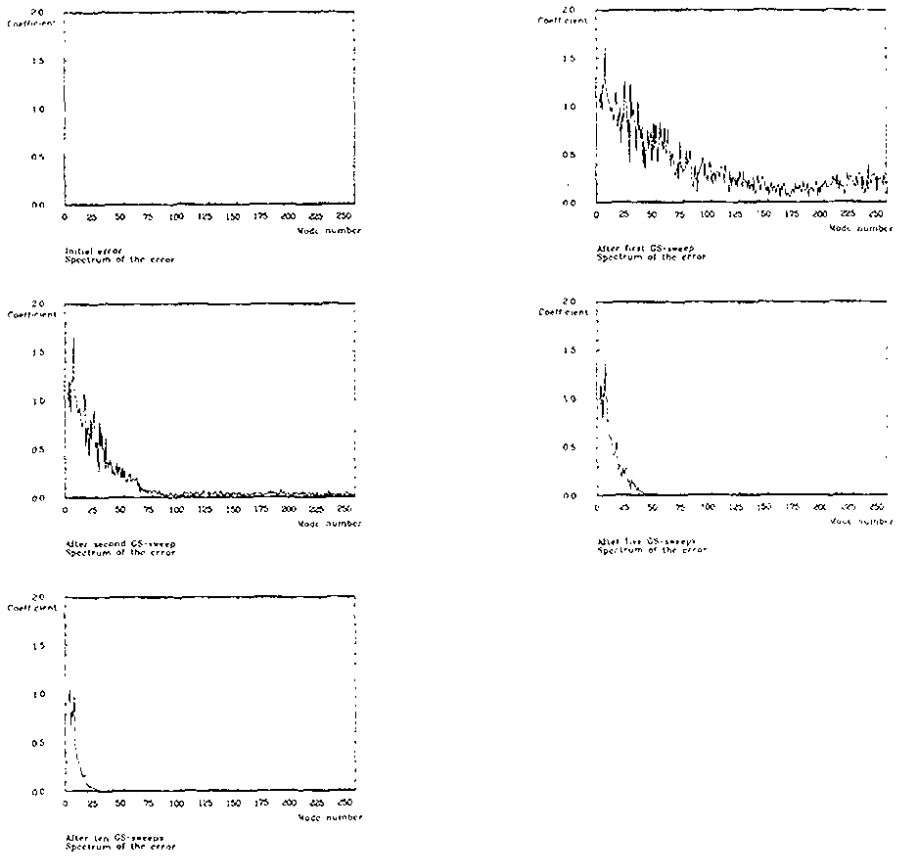


Figure 12: Monitoring Gauß-Seidel sweeps for the Laplace equation.
Lattice size is 16^2 at $\beta = 1.0$ and $\delta m^2 = 10^{-4}$. The expansion coefficients of the error are shown before, after one, two, five, and ten iterations.

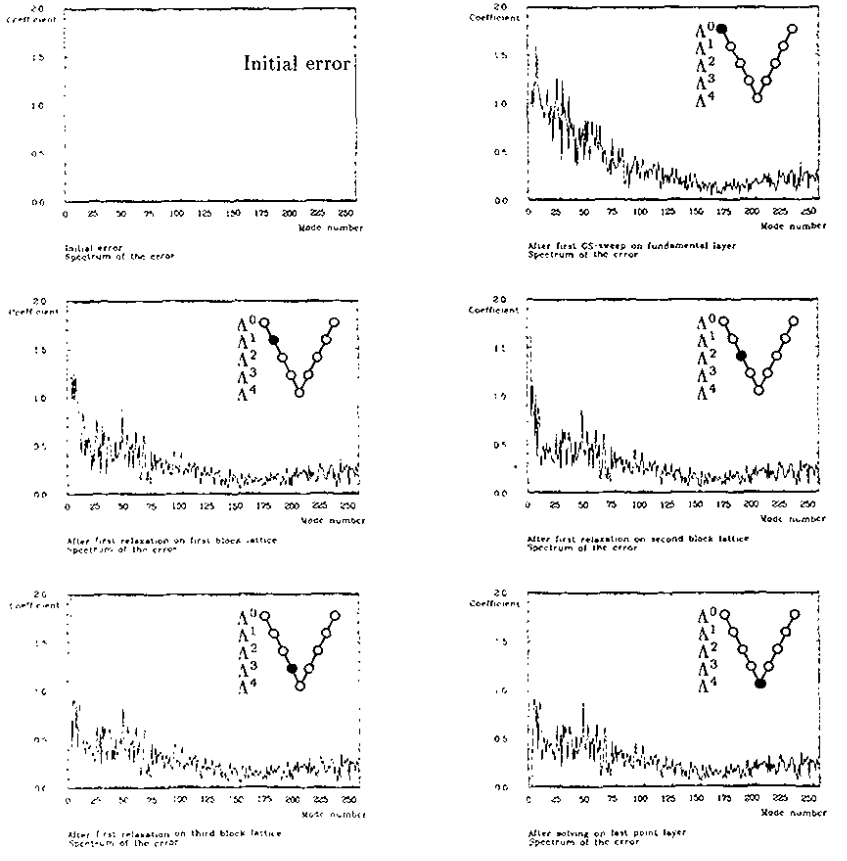


Figure 13: Monitoring ISU for the Laplace equation.
The expansion coefficient for the error is shown. The cycle used was a $V(1,1)$-cycle. The parameters are the same as in the previous figure: Lattice size is 16^2 at $\beta = 1.0$ and $\delta m^2 = 10^{-4}$. The inlay in the upper right of each picture shows the position in the V-cycle.

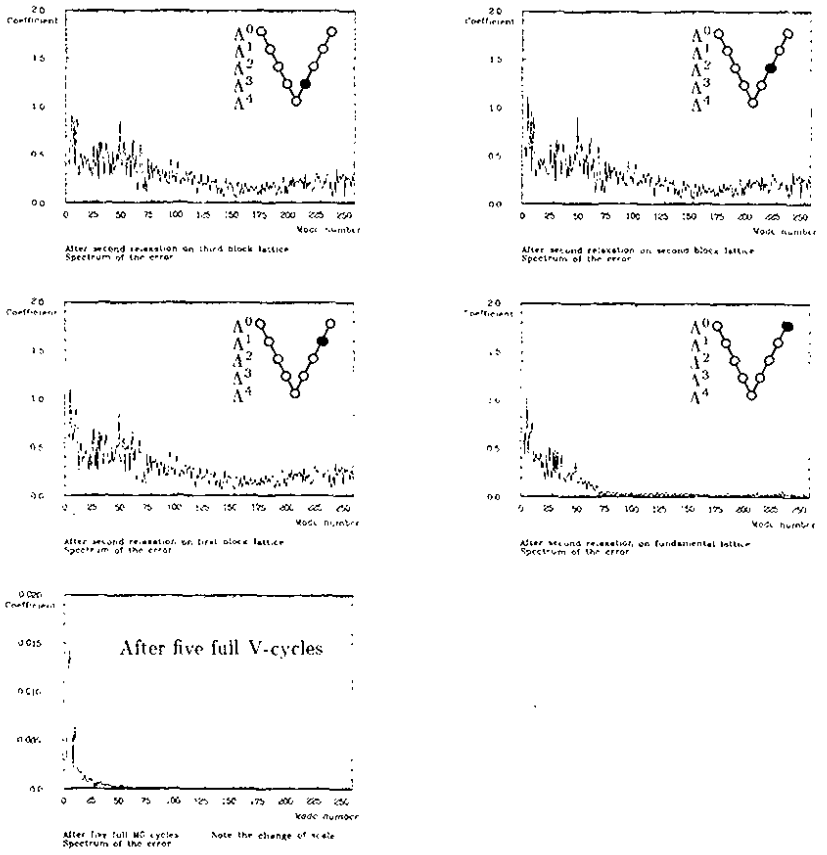


Figure 13: (continued) Monitoring ISU for the Laplace equation.
Note the change of scale by a factor 100 in the last picture.

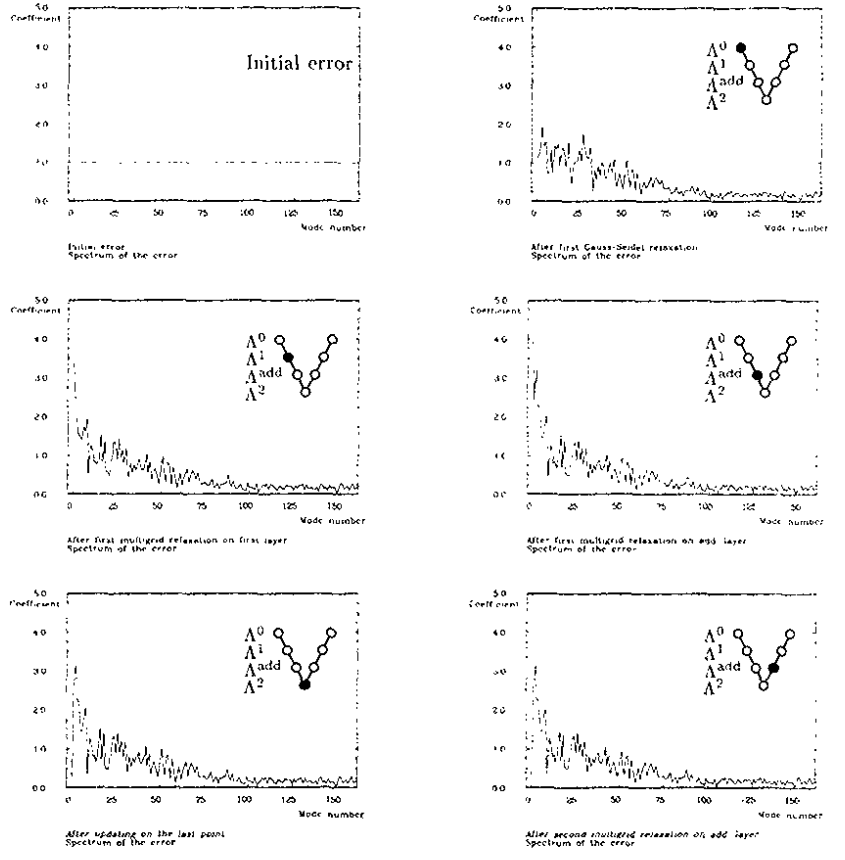


Figure 14: Monitoring ISU for fermions.
The expansion coefficient for the error is shown. The cycle used was a $V(3,3)$-cycle. The lattice size is 18^2 at $\beta = 1.0$ and $\delta m^2 = 2 \cdot 10^{-4}$. The inlay in the upper right of each picture shows the position in the V-cycle.

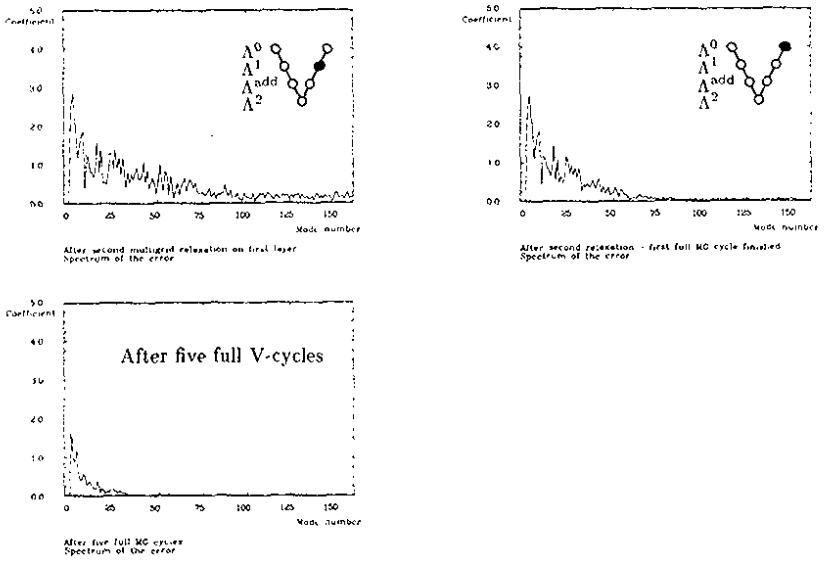


Figure 14: (continued) Monitoring ISU for fermions.

But this is not true. We can easily see this by looking at the simplest problem we can solve with a multigrid method: The one dimensional Laplace operator with Dirichlet boundary conditions. There the eigenvalues are $\varepsilon^j = 4\sin^2\frac{j\pi}{2L}$. For large enough grid lengths L this is approximately $\varepsilon^j \propto j^2$ which gives many low-lying eigenvalues. Nevertheless, any multigrid algorithm (ISU included) will easily solve this problem with an excellent convergence rate and no critical slowing down.

The reason that the unquenched case converges so much better than the quenched case is that ISU does not eliminate critical slowing down here. The algorithm has problems due to the low-lying eigenmodes, and the lower these modes are the clearer these problems can be seen. The lowest eigenmode is treated correctly, so it is mainly the second-lowest eigenmode that gives the biggest problems. As the algorithm does not eliminate critical slowing down its convergence will probably be determined by a kind of condition number as for usual relaxation methods. We must calculate this condition number now with the second-lowest mode because the lowest modes gives not rise to any problems. Therefore the smaller the second-lowest eigenvalue the worse the convergence.

So we see that looking at the eigenvalues alone will not give us sufficient understanding why the algorithm converges differently for bosons and fermions. On the other hand we know that the problems are in fact due to the low-lying eigenmodes of the Dirac operator. So let us now look more closely at the eigenmodes themselves.

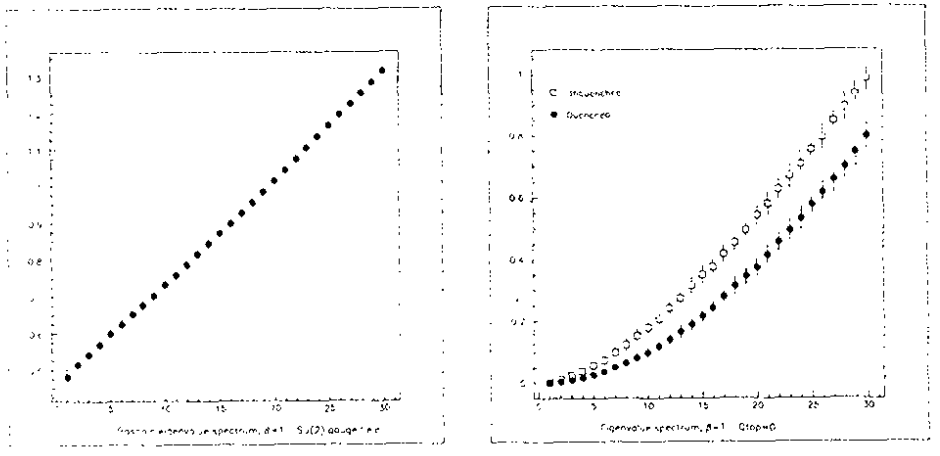


Figure 15: Spectrum of the Laplace (left) and of the Dirac operator (right). The data are averaged over twenty configurations each. For the Laplace operator, they were generated on a 16^2, for the Dirac operator on an 18^2-lattice, both times at $\beta = 1$. For the Dirac operator, the spectrum of the unquenched operator with U(1) gauge fields is also shown.

8 Localization in Lattice Gauge Theory

Wherein we look at the lowest eigenmodes of the Laplace operator, get surprised by their remarkable shape and, after having understood this, are surprised again by the Dirac operator.

8.1 Localization for the 2-dimensional Laplace Equation

At the end of the previous chapter we found that it would be helpful for the understanding of the ISU algorithm to study the shape of the lowest eigenmodes.

If we look only at the norm of the eigenmodes we can see immediately that the lowest and the highest eigenmodes will look identical because of the following

Theorem: Let $\mathbf{D}$ be a $(n \times n)$-matrix with the following properties:

- $\mathbf{D}$ has a constant diagonal θ,
- For all z, z' with $z \neq z'$ and $z + z'$ even: $D_{z,z'} = 0$.

Then the following statement holds:

If ξ^k is an eigenvector of $\mathbf{D}$ to the eigenvalue λ^k ($k \leq n/2$), then the vector ξ^{n-k} with components $(-)^z \xi_z^k$ is also an eigenvector to $\lambda^{n-k} = 2\theta - \lambda^k$.

The proof of this theorem can be found in the appendix. It is also true if the matrix elements are matrices themselves and also applies to the Laplace operator if its matrix elements are ordered in checkerboard fashion (then $(-)^z$ will be $+1$ for all even and -1 for all odd points) but not to the (squared) Dirac operator.

Before we proceed we want to explain intuitively why this theorem holds for the Laplacian: The covariant Laplace operator measures the "naive smoothness" of a mode as explained in section 4.3. The smoothest mode, i.e. the mode with the lowest energy, will be the one where the "discrepancy" will be distributed evenly on all sites of a plaquette involved, so that it is as small as possible everywhere. By simply flipping the value on every even site (rotating it by 180°) we can produce a mode with an extremely large discrepancy vector. In fact, the largest possible discrepancy can be reached when the gauge field is trivial: In this case the discrepancy vector can take the value of twice the value of the mode because the mode flips between $+1$ and -1. This also explains the connection between the eigenvalues of corresponding modes: if the lowest eigenvalue is raised due to the disorder, the highest eigenvalue gets smaller as the discrepancy vector shrinks by the same amount.

That the discrepancy is smaller on some plaquettes than on others raises the suspicion that it might be advantageous for the modes to concentrate themselves only on a part of the grid when disorder is present, and this is indeed what actually happens: The modes become localized.

Figure 16 shows the norm of the lowest eigenmode of the covariant Laplace operator on a 64^2-lattice at $\beta = 1.0$ and on a 32^2-lattice at $\beta = 5.0$ with periodic boundary conditions. For the smaller β-value the localization can be seen clearly, for larger β the localization is in a more extended domain. These pictures show the typical shapes of the low modes, but there are also configurations with more than one localization peak. An example is shown in figure 24, page 73. These modes with multiple peaks are exceptional, the majority of modes having only one peak.

This poses the question whether all modes are localized. To answer this we calculated all modes on a smaller grid. It was found that only a few of the low modes show localization. It might be possible that on very large grids (with very low values of β) all of the modes are localized as localization increases with the disorder. But to study this a large computational effort would be needed. (The simple storage of all eigenvectors on a 128^2-lattice would need about 1 Gigabyte.) Furthermore, the

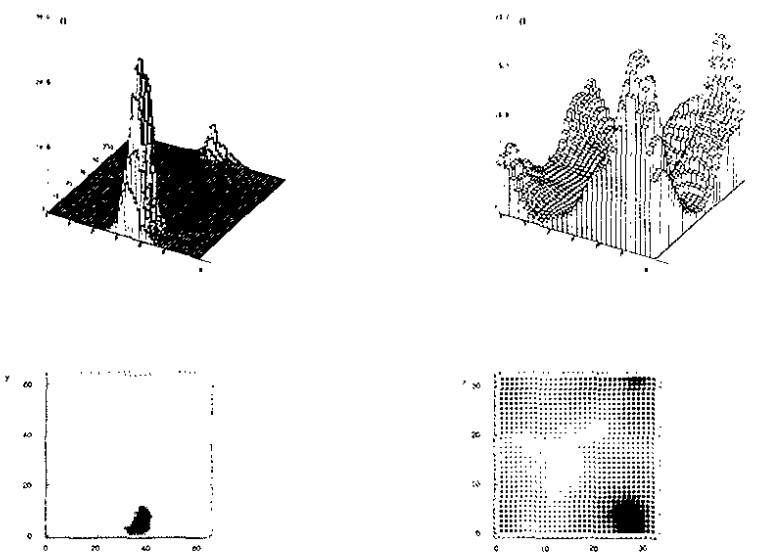


Figure 16: Norm of the lowest eigenmode of the covariant Laplace operator in arbitrary units.
Left: at $\beta = 1.0$ on a 64^2-lattice.
Right: at $\beta = 5.0$ on a 32^2-lattice.
Pictures at the top are 3D-plots, bottom pictures are density plots with dark regions corresponding to high values.

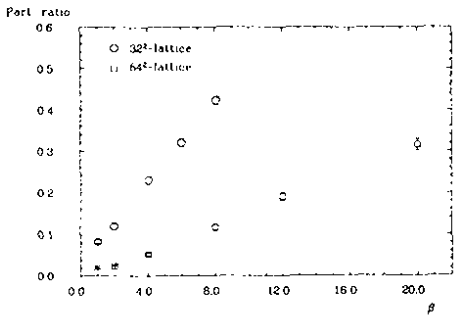


Figure 17: Participation ratio as a function of the disorder. The configurations were generated on a 32^2- and on a 64^2-lattice.

explanation given in the next section suggests that probably only a fraction of the modes will be localized in all cases.

The decay of the modes is exponential. This was shown by plotting the norm of the mode logarithmically.

It is expected that the sharpness of localization will increase with the disorder, i.e. with decreasing β. To study this effect we may look at the *participation ratio* defined as [41]

$$a = \frac{1}{N} \frac{\left(\sum_{i=1}^{N} |\xi_i|^2\right)^2}{\sum_{i=1}^{N} |\xi_i|^4} \quad , \tag{62}$$

where N denotes the number of grid points. This quantity measures the fraction of the lattice over which the peak of the localized state ξ is spread. Figure 17 shows the participation ratio of the lowest eigenmode as function of β calculated on 32^2- and 64^2-lattices. 100 runs for the smaller lattice size and 50 runs for the larger lattices were done for each value of β. The dependence on the disorder is obvious. The absolute size of the localized state, measured by Na, does not depend on the grid size if the grid is larger than the localization peak.

Fig 18 shows the participation ratio versus the scale parameter $\eta = \sqrt{\beta}/L$ where L is the grid length. The participation ratio should only depend on this parameter, so the values for the different grid sizes 32^2 and 64^2 should lie on the same curve, as in fact they do. For very small values of β one expects deviations from the scaling behavior because the absolute size of the mode (in lattice units) depends for large enough grids only on β: as this sizes approaches a constant when β goes to zero, the participation ratio should then scale with the lattice volume. One would therefore expect in our case that the participation ratio is about four times larger for the smaller grid size at fixed β which is approximately the case. Some deviation can be explained by the fact that on the one hand the occurrence of multiple localization peaks is more probable on the larger grid and on the other hand localization might be stronger on the larger grid as each grids samples a larger range of gauge field fluctuations.

We want to remark here that a similar phenomenon of localization also has been found for the four-dimensional Laplace equation in an SU(2)-gauge-field [70].

8.2 Explaining the Localization

How can we understand this phenomenon of localization in a gauge field background?

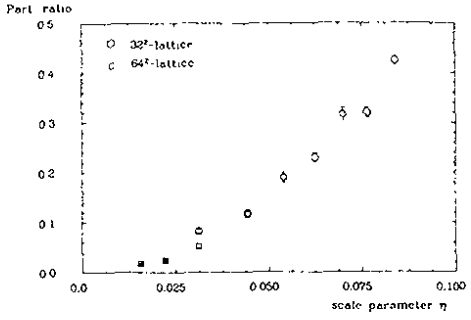


Figure 18: Participation ratio as a function of the scale parameter $\eta = \sqrt{\beta}/L$, where L is the grid length, using the data of figure 17.

We will try to relate it to Anderson-localization [42, 43, 44]. Anderson examined a tight-binding-model of an electron in a random potential $V(z)$, leading to a Schrödinger equation

$$(-\Delta_{\text{scalar}} + V(z))\psi = E\psi \quad . \tag{63}$$

Localization of all modes may occur dependent on the disorder and the dimension. In one dimension, arbitrarily small disorder leads to localization, in two dimensions there is a phase transition between weak and strong localization as one increases the disorder, in higher dimensions a transition between non-localized and localized states occurs. A theoretical understanding of this was achieved by the Abrahams *et al.*[48] and later by Fröhlich and Spencer [45]. As the Schrödinger operator for this model is again the Laplacian (without a gauge field), we may expect a similarity between our localization problem and Anderson localization, however, there are crucial differences: First, our operator is not fully random because the equilibrated gauge field possesses correlations. Second, our operator shows *off-diagonal disorder*: It is not the potential that varies from site to site but the couplings between the sites. And third, the couplings do not vary in *strength*, but in orientation in color space resulting in a frustrated system.

Nevertheless, in the following we will try to stress similarities between the two models, giving us an -at least intuitive--picture of what happens.

We will use a slightly involved argument first which is based on renormalization. After having identified the quantity responsible for the localization, we will give an intuitive explanation of its importance.

We have to look for a quantity in our model that may play the role of the random potential $V(z)$ in the Anderson problem. There we can extract the value of the potential by calculating the difference between the diagonal element and the norm of the couplings because this is zero for the Laplace operator without potential, see equation (8). Doing this for our model problem of course only gives $\delta m^2 - \varepsilon_0$, independent of the lattice site. However, we already know that the lowest eigenvalue of the Laplace operator is a measure of the disorder. This eigenvalue can be associated with the whole grid, it is similar to a mass term. If we divide the grid into several regions, the eigenvalues on these regions (using for example Dirichlet boundary conditions) would differ analogous to the differing potential values in the Anderson problem.

A quantitative formulation of this idea is the following: To arrive at a varying quantity, we may look at a blocked version of our operator: the simplest blocking procedure one can think of is the "taxi-driver" blocking [4]. Here four points are blocked to one and all block-quantities are calculated

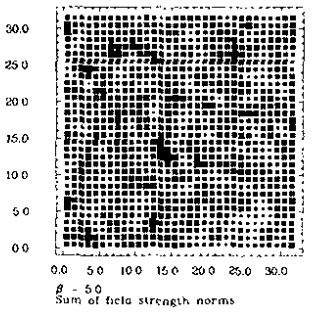
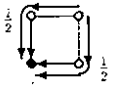


Figure 19: Sum of squared field strengths $W(z)$ corresponding to the gauge field configuration of figure 16, right. The localization clearly lies where $W(z)$ is small.

by parallel-transporting the fine-grid-quantities to one of the four points (e.g. the lower left point) via the shortest possible route. (For the upper right point there are two routes, each weighted with a factor 1/2.)



If we use a blocking-operator of this type and calculate the blocked operator $\mathbf{D}' = \mathcal{C}\mathbf{D}\mathcal{A}$ we still have an operator with fluctuating bonds and a constant diagonal. (Here $\mathcal{C}$ and $\mathcal{A}$ are the blocking and interpolation operators.) But now the bonds are fluctuating in strength, so we have to separate the kinetic and the potential part of the diagonal elements by calculating $V(x) = \|\mathbf{D}'(x,x)\| - \sum_{y \neq x} \|\mathbf{D}'(x,y)\|$ as explained above. This quantity *is* fluctuating on the block lattice, and so we arrived at a situation much more similar to Anderson-localization.

The blocking procedure has a certain arbitrariness (e.g. in the choice of the blocks), but a simple calculation shows that it is mainly the quantity $\|F_{\mu\nu}(z)\|$ that enters into $V(x)$. (This has to be expected considering that the quantity involved has to be gauge covariant.) Let us now look at the field strength norm, or, more conclusive, at the quantity

$$W(z) = \sum_{\mu \neq \nu} F_{\mu\nu}(z) F^{\mu\nu}(z) \quad . \tag{64}$$

Figure 19 shows this quantity for the same configuration as in figure 16, right, and one can see clearly that the localization is in a region where $W(z)$ is small, as should be expected from our argument. This is true for *all* configurations we looked at: *The localization center is always in a region with low field strength sum.* In this way one can also understand the occurrence of modes with multiple localization peaks: These modes exist when $W(z)$ has several minima which are equally good as localization centers.

We can support this idea further by looking at the eigenmodes of the following operator which we called Anderson-Laplace-operator: $\mathbf{D}_{AL} = -\Delta_{\text{scalar}} + W(z)$, where Δ_{scalar} is the Laplace operator

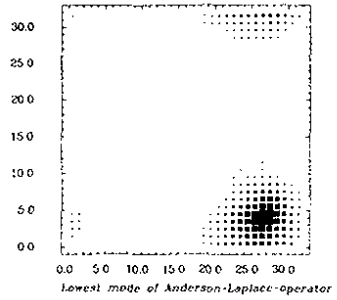


Figure 20: Lowest mode of the Anderson-Laplace-operator D_{AL} for the same configuration as figure 16, right, using $W(z)$ from figure 19 as potential.

without gauge field. So now we are looking at a true Anderson-localization problem except that the *random potential is not independent at different sites due to the finite correlation length of the gauge* field. Figure 20 shows the lowest mode of this operator. If compared to figure 16, right, one sees that the center of localization sits at the same place. This, however, is not true for all configurations. For *example, it may happen that the lowest mode of the Anderson-Laplace operator is localized at the* same place as the second-lowest mode of the covariant Laplacian. Nevertheless the tendency is clear.

Coming back to the argument from the previous section, we can understand the importance of *the quantity $W(z)$ even more intuitively: It measures directly how large the discrepancy vectors in* a certain region are. The smaller these vectors, the smoother the gauge field and therefore the lower the energy of a mode placed here.

The connection to the Anderson problem also explains why even for arbitrarily disordered gauge fields not all modes are localized: We could expect this only when $W(z)$ were itself arbitrarily disordered. But this will not happen, even at $\beta = 0$ $W(z)$ will only have a finite distribution width.

8.3 Localization for the Dirac Operator

Having understood the phenomenon of localization for the Laplace operator, let us now turn our attention to fermions. Similar mechanisms might be expected to hold.

Figure 21 shows the lowest eigenmode of the squared Dirac operator on a 54^2-lattice at $\beta = 1$. Clearly, there is no localization. The participation ratio at these parameter values was found to be 0.18 ± 0.04, where we have calculated 30 configurations. Comparison with values at other parameters seems to imply that this number scales physically, but we do not have enough data to be sure about this. So localization is absent for the lowest mode of the Dirac operator.

However, this is only true in two dimensions. The lowest modes of the Dirac operator in four dimensions have also been studied [70, 36] both for Wilson and Kogut-Susskind fermions. There it has been found that the lowest eigenmodes are strongly localized; in fact so strongly that the localization can already be seen on very small lattices.

It is not clear why this is so. Judging from the connection to the Anderson-localization problem *one would expect the contrary: In the Anderson case localization gets weaker when the dimension is* increased.

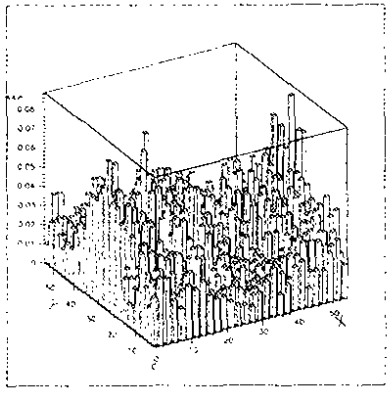


Figure 21: Lowest eigenmode of the squared Dirac operator on a 54^2-lattice at $\beta = 1$.

However, it has to be kept in mind that we are dealing with a special kind of off-diagonal disorder. The (squared) Dirac operator has a larger stencil than the Laplace operator, so instead of the quantity $W(z)$ as defined in the previous section one might have to look at a quantity which averages over a larger region of the grid, e.g. by taking all Plaquettes up to a size of 2×2 into account. This might lead to a stronger averaging of the gauge field fluctuations and so to a more ordered Anderson potential.

Up to now it is not known whether the localization in four dimensions is governed by the quantity $W(z)$ or an analogous measure, but this will be studied in the future [37].

That $W(z)$ *is* a meaningful quantity for the Dirac operator in two dimensions can be proven by looking at the *highest* modes. Remember that for the Dirac operator high and low modes do not look identical. Figure 22 shows the highest mode of the Dirac operator on lattices of size 64^2 and 128^2 at $\beta = 1$. The localization is obvious; for the larger grid size, several distinct localization centers occur.

Figure 23 shows the participation ratio of the highest modes as a function of β and the scale parameter η. The data were generated on lattices of sizes 32^2 and 64^2, generating 50 configurations on the smaller and 20 on the larger lattice size. All modes exhibit strong localization. The plot showing the dependence on the scale parameter seems to indicate that the participation ratio shows some deviation from the correct scaling behavior because the data lie on different curves for small η which merge as η becomes larger. This agrees with what was said above for the scaling behavior at small β: the quotient of the participation ratios for the different lattice sizes is slightly larger than four.

To prove that $W(z)$ is a meaningful quantity at least for the high modes, we did the same steps as in the previous section: We compared the maxima of $W(z)$ with the position of the localization centers. Again, the agreement was striking, as was the agreement with the highest mode of the Anderson-Laplace operator defined in the previous section. So $W(z)$ is a useful quantity even in this context despite the fact that it seems to fail for the lowest modes, at least in two dimensions.

Finally, let us remark that the localization picture is the same for gauge group U(1) in two dimensions, i.e. the highest modes are localized, but the lowest are not for unquenched as well as for quenched configurations.

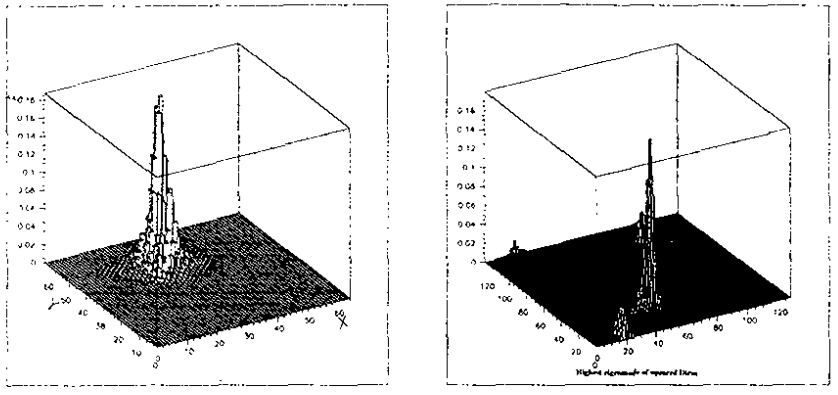


Figure 22: Highest eigenmode of the squared Dirac operator at $\beta = 1$ on a 64^2-lattice (left) and a 128^2-lattice (right).

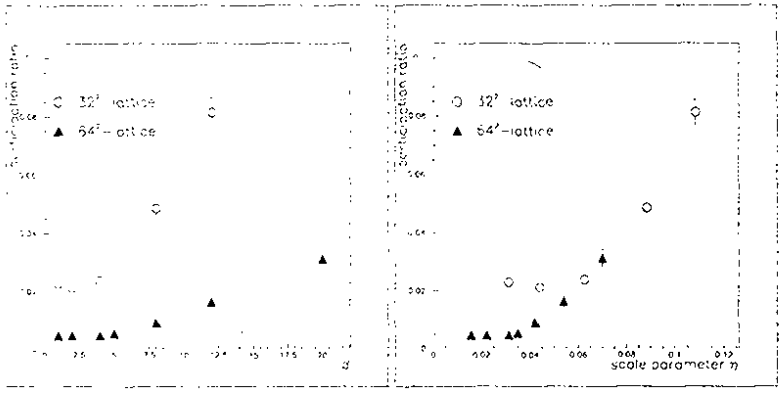


Figure 23: Participation ratio of the highest eigenmode of the Dirac operator for two different lattice sizes at various values of β.
Left: Participation ratio as function of β
Right: Participation ratio as function of the scale parameter $\eta = \sqrt{\beta}/L$.

Model	Loc. of low modes	Loc. of high modes	Scaling	Relation to Anderson Loc.
2D Laplace	yes	yes	yes	yes
4D Laplace	yes	yes	*unknown*	*unknown*
2D Dirac	no	yes	yes	yes for high modes
4D Dirac	yes	unknown	unknown	unknown

Table 3: Overview of localization phenomena in Lattice Gauge Theory from this thesis and [70, 36].

8.4 Further Questions

Our study of localization for fermions has been rather cursory because in this thesis the main point of interest are the implications for the performance of the ISU algorithm. Nevertheless it would be interesting to understand the phenomenon of localization further to fill the gaps in table 3 and to answer the following questions:

- *What happens if we approach the continuum limit, i.e. on larger grids?*
- What are the physical implications of localization?
- Why are the lowest modes of the two-dimensional Dirac operator not localized?

Some of these questions would require the study of localization on large grids. To do so, it might be possible to adapt the *transfer matrix method* [47] which is state of the art in the investigation of Anderson localization in solid state physics. The basic idea is to study a system which is extremely long in one of its directions (a long, thin bar), where the length might take values of 10^6 lattice constants. One then reformulates the basic equation in a way that has the form of a transfer matrix, i.e. a matrix which acts on the values in slice t to yield the values in slice $t+1$. The eigenvalues of the product of all transfer matrices between the slices can then be calculated step by step, always multiplying the next slice to the already known matrix. The great advantage of this approach is that one needs to store only the $(d-1)$-dimensional sub-lattice of the actual slice which permits the extremely large lattices required to get reliable information about localization. This method has first been applied analytically by Ishii [49] for the one-dimensional problem where strict theorems about the products of random matrices hold. Numerical studies of this and the related Green's function method have been done later for higher dimensions [47] after the breakthrough of finding scaling relations for the Anderson localization problem which was achieved by Abrahams *et al.*[48].

It would be worthwhile to adapt these methods to our case, as it has been found for the Anderson localization problem that results achieved on small grids are not overly reliable due to possible large fluctuations.

If localization of the lowest modes of the Dirac operator is a physical phenomenon in four dimensions, there might be interesting implications. For example, it could happen that the ground state of the Dirac operator in the continuum limit becomes highly (or even infinitely) degenerated because it could consist of several localized maxima which are so widely separated that they are independent of each other. This would imply a large value of the spectral density near zero which is linked to the chiral condensate [31].

9 Analyzing ISU—Part II

Wherein we continue our analysis of ISU, see how localization helps its convergence, make a short digression to another problem to test our analyzing tools, and finally construct some new and helpful multigrid analysis methods.

9.1 Localization and the ISU Algorithm

As we have seen, the lowest eigenmodes of the covariant Laplace operator are strongly localized. As we also found that for the Laplace equation the convergence of the ISU algorithm was fairly good, it is natural to ask whether there is a connection between the two facts: Does localization help the algorithm to converge? This is highly plausible because if a mode consists of a few localized parts it seems clear that we can patch it together from operators which are restricted to a part of the grid. We will now look at this argument in greater detail.

Figure 24 shows the lowest mode of an—exceptional—gauge field configuration on a 32^2-lattice at $\beta = 1.0$. This mode consists of several localization peaks, dropping to a small value between the peaks. So there is near-degeneracy of the lowest mode and a smooth error may consist of any linear combination of these peaks.

To remove such an error effectively, each of the peaks must be treated correctly, so for each peak there must be an interpolation operator resembling it. The four interpolation operators on layer Λ^{N-1} shown in figure 25 clearly fulfill this requirement. The reason for this lies in the boundary conditions: Whenever a boundary cuts a localization peak, the restricted operator can not have a mode with this peak as low-lying mode and will single out a different peak (or different peaks). This is also true for the more typical configurations as they are shown in figure 16, only there it will be the second-lowest mode which is singled out, not another part of the (nearly) degenerated lowest mode.

But what happens if the lowest mode is very strongly localized as in figure 16, left? In this case, all interpolation operators on layer Λ^{N-1} will probably resemble this mode, so how are the higher modes represented? This is done by the operators on Λ^{N-2}, which is possible because the localization length has to be very small compared to the grid size (otherwise the peak would be cut by one of the boundaries), so these operators are also able to cover the low-lying modes. For the configuration

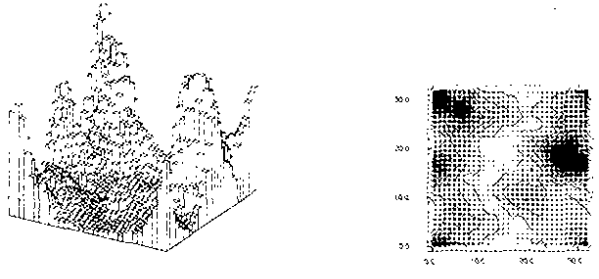


Figure 24: Norm of the lowest mode of a gauge field configuration with multiple localization peak on a 32^2-lattice at $\beta = 1.0$ as 3D-plot and density-plot.

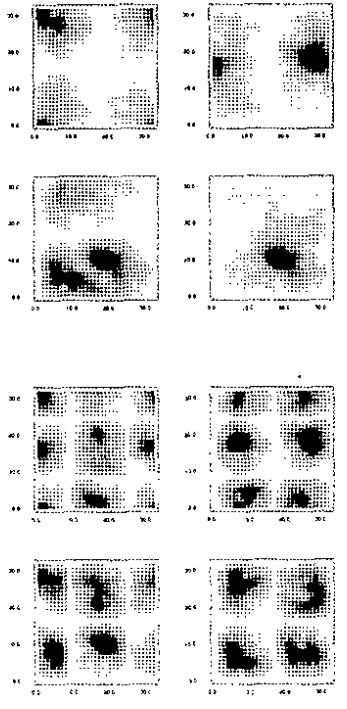


Figure 25: Norm of the interpolation operators as calculated with the ISU-algorithm for the same configuration as figure 24
Top: Operators on layer Λ^{N-1}, each having a 31^2-support.
Bottom: Operators on layer Λ^{N-2} with support-sizes 15^2. Of the 16 operators on this layer, four with non-overlapping blocks are shown in each picture.

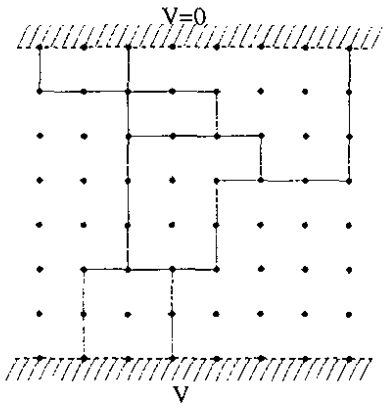


Figure 26: A bond percolation cluster. The solid lines belong to the backbone of the cluster, the dotted lines are irrelevant side lines.

of figure 24 the shape of these operators is mainly determined by the boundaries, as the localization *length is similar to the support length.*

These explanations are true whenever the disorder is strong and therefore the localization is also strong. It has to be stressed here that this does *not* mean that the algorithm only works in the strong-coupling regime (at small β). Critical slowing down does not appear for any value of β and the grid size. The algorithm requires that the low modes are approximated well by the interpolation operators. For strong disorder this is true because of the given arguments, for weak or no disorder this is also true since then the low modes and the interpolation operators are smooth in the usual sense. One can see this in the free field case, where the low-lying modes and the interpolation operators are sine waves with a long wavelength. (The lowest mode is a constant, but this mode is calculated as interpolation operator on the last point lattice A^N and so the interpolation operator $\mathcal{A}^{[0,N]}$ is also constant.)

So we have understood how localization helps the algorithm to converge well. As the lowest modes of the two-dimensional Dirac operator are not localized, this might be part of the reason why the algorithm does not perform well.

9.2 ISU and the Random Resistor Network

Before we continue the analysis of the problems in using ISU for the Dirac-operator, let us make a short digression to another problem: The random resistor network [46]. This model of a disordered solid is an example of a *bond-percolation cluster.*

Consider an infinitely extended cubic lattice. A bond-percolation cluster is defined by setting bonds (i.e. links between nearest neighbours) with a certain probability p. Imagine that at each set bond we have an electrical resistor of unit strength whereas on each unset bond there is no connection (the resistance is infinity). A typical percolation cluster on a small grid is shown in figure 26. Such a system is called a *random resistor network.*

It is easily shown by a duality argument [46], that at a critical bond probability of $p_c = 0.5$ there will exist an infinitely extended cluster for almost all configurations. (This is only true for the two-dimensional square lattice in which we are interested here.)

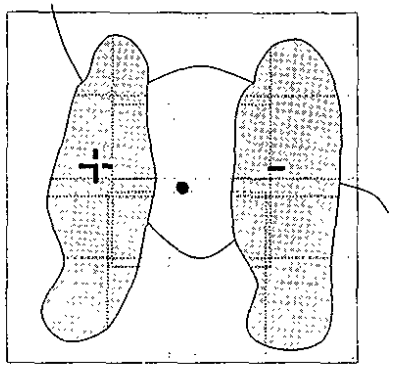


Figure 27: A bad-converging mode of the random resistor network problem. The mode has different signs on two loosely connected regions. None of the single parts of the regions is covered by the support of an interpolation operator.

The disorder of such a percolation cluster is very strong: Cutting a single bond may result in the whole cluster falling apart. The random resistor network is an interesting model for disordered systems and may serve as a testing ground for algorithms dealing with them. Some numerical studies can be found in [53]

Imagine now that we apply a voltage to two points on the cluster that are a large distance apart or to the upper and lower edge of the lattice in the case of a finite lattice. We then may ask for the voltage on each of the nodes of the grid. It is clear that all branches that are "loose ends" not connecting *several parts of the cluster can be cut off without changing the voltage distribution. The remaining* part of the cluster is called its "backbone". In the figure the backbone is marked with solid lines, the loose ends with dotted lines. A backbone cluster can be created by a so-called "snip"-algorithm [51].

Let us now find the equation for the voltage at each point: Let $I_{j,\mu}$ be the current on the link between a point j of the cluster and $j+\mu$, its nearest neighbour in the μ-th direction, and V_j the voltage at point j. Using Ohm's and Kirchhoff's Laws we get for all points within the cluster

$$0 = \sum_\mu I_{j,\mu} = \sum_\mu{}' \frac{1}{R}(V_j - V_{j+\mu}) \Longrightarrow \sum_\mu{}'(V_j - V_{j+\mu}) = 0 \quad . \tag{65}$$

where the sum $\sum_\mu{}'$ is only over those directions in which a bond is set, and R is the resistance of a *single bond. This equation is a simple Laplace equation with Neumann boundary conditions on the* cluster. So it is not surprising that the solution of this equation will be affected by critical slowing down when the cluster gets large.

As has been shown by Edwards et al.[50], the Algebraic Multigrid shows no critical slowing down for this problem. As we are intending our ISU algorithm to be more generally applicable than the AMG, it seemed worthwhile to try the algorithm at this problem as well. For simplicity, we used a very simple implementation with *fixed block-centers and supports*, chosen identically to the bosonic case described in section 4.1.2. It is of course clear that the AMG will be superior to ISU in any case because of its true multigrid character, but we hoped that ISU would perform without critical slowing down and be as good as it could.

However, this was not true. We tried lattice sizes from 16^2 up to 256^2 with statistics of several dozen runs each. The results we got for the solution of the Laplace equation itself gave an estimate

of the critical exponent of $z \approx 0.75$. But as we have to calculate good interpolation operators before solving the equation itself, this in fact means that the algorithm performs worse than this: More and more iterations would be needed to calculate the interpolation operators. We did not study this in detail because the failure of the algorithm is obvious.

What we did study were the reasons for the failure. One might guess that the problem lies in the fixed blocks and block-centers. To find out whether this is really true, we used some of the methods developed above. Let us shortly sketch what was done to show the usefulness of the described tools:

The first thing we did was an error-monitoring, as described in section 7.2. Which modes are the ones responsible for the bad convergence? It was found that the picture was similar to the Dirac case: some of the lowest modes were not reduced efficiently. Different from the Dirac case, usually there was one mode which converged much worse than the others. What are the characteristics of these modes?

Typically, the bad-converging modes were large on two loosely interconnected regions of the network, with a positive sign on the one and a negative sign on the other region, as sketched in figure 27. Following Coniglio [52], we call these parts "blobs". In the interior of each part there are many connections, but the part itself only has few connections to the other part of the network. The problem with these modes arises when the shape of the two blobs is such that one of them (or both) is not covered by the support of an interpolation operator. In this case the "+"-part or the "−"-part of the mode can not be reduced efficiently because each of these parts corresponds to a low-energy mode on the blob. No interpolation operator has the shape of the low-energy-mode of the blobs: the smaller ones since they do not cover a whole blob, the larger ones because they will have the same sign on both blobs, as this has a slightly smaller energy. This also explains why it is not the lowest or second-lowest mode that is bad-converging but a higher one.

This analysis teaches us that it would indeed be helpful to choose the blocks dynamically; in this case each of the blobs would have its own interpolation operator and so the two parts of the mode could be moved independently. It might be interesting to do this using an AMG-type algorithm for the determination of the block-centers and to compare the two algorithms. It would also be nice to check that the AMG block-centers indeed agree with the intuitive topological picture of the random resistor network which describes the network as consisting of blobs and nodes connected by links. Such an analysis would be in the spirit of Solomon's work [91], see section 6.3. It would show (or disprove) that the AMG-algorithm has no critical slowing down because it correctly represents the relevant degrees of freedom. We will sketch an algorithm to determine dynamical block centers in section 10.5.

9.3 *The Rayleigh Quotients of the Interpolation Operators*

After this excursion, let us now return to the Dirac operator. In the following we want to develop further tools for the analysis of bad-converging modes to get a fuller understanding of the problem.

To see whether it is possible to approximate an eigenmode by interpolation operators as they are calculated using ISU, we can do the following:

Let the low-lying mode to be approximated be χ. The interpolation operators $\mathcal{A}^{[0,j]}$ can be considered as vectors on the fine grid $\xi^i = \mathcal{A}^{[0,j]}_{,x}$, where i enumerates j and x in some way.

For simplicity, let us first consider the case of non-overlapping supports of the interpolation operators and let us fix the layer A^j used. To approximate the mode χ, each ξ^i has to be as similar to χ as possible. As χ is an eigenmode, ξ^i also has to be an eigenmode "locally", or, more strictly speaking, its Rayleigh quotient

$$\mathcal{R}(\xi^i) = \frac{\langle \xi^i, \mathrm{D}\xi^i \rangle}{\langle \xi^i, \xi^i \rangle} \tag{66}$$

must be constant where it is defined (i.e. within the support of ξ^i) and (at least approximately) smaller than the eigenvalue of χ. It is not necessary that the Rayleigh quotient is equal to the eigenvalue because when it is smaller, combining the mode linearly with the interpolation operators will be partly possible, the remaining (orthogonal) part then having a higher eigenvalue.

Note that this statement is of course no more true when the interpolation operators are not required to be eigenfunctions on an extended domain. Obviously we can approximate every function we like with L^d delta-functions.

Let us explore this mathematically: Let the mode χ have no part orthogonal to the interpolation operators: $\chi = \sum_i \alpha_i \xi^i$ with $\sum_i |\alpha_i|^2 = 1$. Then we have

$$\langle \chi, \mathrm{D}\chi \rangle = \sum_{ij} \alpha_i^* \alpha_j \langle \xi^i, \mathrm{D}\xi^j \rangle \quad . \tag{67}$$

If the ξ^i were orthogonal eigenfunctions to the full operator and had only a restricted support (which are demands not to be fulfilled simultaneously), the term in brackets would be a Kronecker delta and the eigenvalue of each of the ξ^i were equal to that of χ. But the ξ^i are eigenmodes only locally, therefore $\mathrm{D}\xi^i$ will be a vector which is nonzero outside the support of the ξ^i itself and the term in the above formula will not be a Kronecker delta even for non-overlapping blocks. However, if we have Dirichlet boundary conditions and large blocks, the value of $\mathrm{D}\xi^i$ will be quite small so $\langle \xi^i, \mathrm{D}\xi^j \rangle$ will also be small whenever $i \neq j$ and the Rayleigh quotient will be at least approximately equal to the eigenvalue of the mode.

We now have to consider the case of overlapping blocks. Here, the argument is similar but slightly more involved. Even in this case, there exists a point for each interpolation operator (namely the block-center) where all other interpolation operators vanish. Because of the Dirichlet boundary conditions used for the interpolation operators, they will be small also in the vicinity of the block-center of another interpolation operator, since otherwise they would have large energy costs from the boundary and could not correspond to low eigenmodes. So near its block-center the kernel has to approximate the eigenmode nearly without help from the other interpolation operators. Hence its Rayleigh quotient has to be small there, and, as ISU calculates interpolation operators as eigenmodes of the restricted operator, its Rayleigh quotient has to be small everywhere.

So the criterion involves a delicate balance between the number of interpolation operators involved and the size of the Rayleigh quotients: Many interpolation operators allow for larger Rayleigh quotients; if there are only a few of them, their quotients have to be small.

This can be easily checked for the scalar Laplace equation with periodic boundary conditions in one dimension using the usual linear interpolation operators: The lowest mode of the operator is the constant which can be perfectly approximated by triangular shaped operators. If we choose the triangles small, we have many of them and they have large Rayleigh quotients, if we have a few of them with a large support, their Rayleigh quotients are very small because they are smooth.

So let us now compare the Rayleigh quotients of the interpolation kernels with the eigenvalues of the operator. To approximate an eigenmode χ^i, there has to exist at least one interpolation operator with Rayleigh quotient equal to this mode. (Actually, there should be more than one if the interpolation operators are restricted to a part of the grid.) We can now ask the question:

> How many interpolation operators exist on a layer A^j with a Rayleigh quotient smaller than the m-th eigenvalue of the operator D?

We call this function $G_j(m)$. (The letter G signifies that these are the "good" interpolation operators.) A good approximation of all modes on a layer A^j can only be possible when $G_j(m) \geq m$, at least approximately. We can also sum this quantity over all layers of the unigrid; then we would expect $G(m) = \sum_j G_j(m) \geq m$. Otherwise there would exists low eigenmodes that can not be approximated well because the interpolation operators would be too oscillatory.

Of course $G_N(1) = 1$ as the last-point interpolation operator $\mathcal{A}^{[0,N]}$ is the lowest eigenmode of the operator. (The lowest eigenmode gets the number 1 here.)

Let us now look at the function G_j for the Laplace and Dirac operator. Again we use lattice of size 16^2 and 18^2 at $\beta = 1$. Figure 28 and 29 show this function for the two coarsest layers and the summed function in both cases. The data are averaged over ten configurations in each case. The

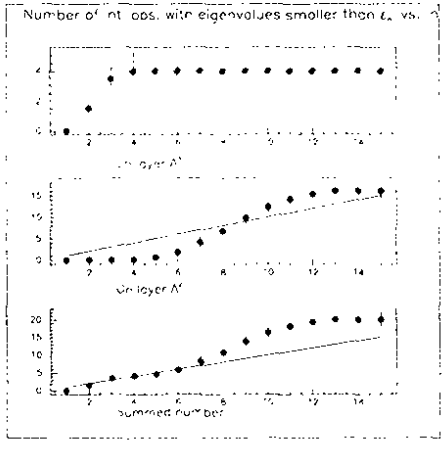


Figure 28: The function $G_j(m)$ and its sum $G(m)$ over all layers for bosons. Good convergence of the algorithm can only be expected for $G(m) \geq m$, otherwise there are not enough smooth interpolation operators with low Rayleigh quotients to approximate the low-lying modes of the operator. See the text for further explanation of the function $G_j(m)$ and of this criterion. All data at $\beta = 1$ on 16^2-lattices.

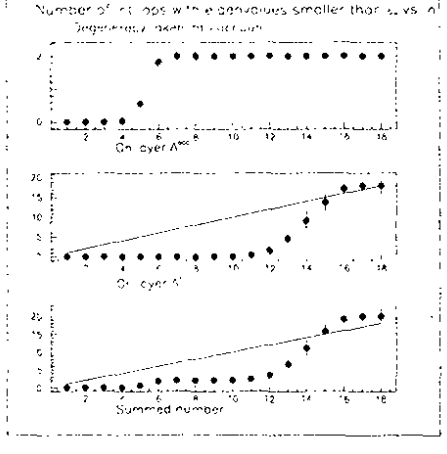


Figure 29: The function $G_j(m)$ and its sum $G(m)$ for fermions. See the previous picture and the text for further explanation. All data at $\beta = 1$ on 18^2-lattices.

figures also show the diagonal that has to be crossed by $G(m)$. The difference is dramatic. For the Laplace-operator, it seems to be possible to approximate the second-lowest and higher modes quite well with the interpolation operators, but for the Dirac operator only the fifth mode and the higher modes can be approximated even with the additional unigrid layer, but even for this there are not enough interpolation operators. The diagonal is crossed immediately in the bosonic case, but for fermions only at $m = 15$.

Let us take a closer look at figure 29. The upper part shows $G_{\text{add}}(m)$, the function on the additional layer, the lower part on Λ^1 which would usually be the coarsest one. Only the modes with mode numbers higher than eleven can be approximated well by the interpolation operators on this layer, whereas for the additional layer this number reduces to five. This justifies the introduction of the additional layer, but it also shows that this cannot be enough to eliminate the bad convergence completely. Even with the additional layer there are two "holes" in the spectrum, one for the four lowest modes, the other for the modes between 7 and 12. The latter arises because there are not enough interpolation operators on the additional layer and the interpolation operators on the next layer have eigenvalues that are too high. This also confirms the conclusions drawn from the error-monitoring: The Dirac operator is problematic because of its many low-lying eigenmodes, *which cannot be represented by a superposition of the interpolation operators.*

That the Laplace operator is so well-behaved with respect to this analysis can be easily understood considering the localization of the modes. It is quite clear that localized modes can be well approximated by interpolation operators with a large support. This raises the hope that in four dimensions, where the low-lying eigenmodes of the Dirac-operator are also localized, ISU will converge much better.

9.4 The Dirac Operator in Different Boundary Conditions

The analysis of the previous section suggests that the lowest modes of the Dirac-operator have in some sense a very global structure which is lost when we restrict the operator to a part of the grid using Dirichlet boundary conditions. To be sure that the problem lies here and is not caused by removing points from the grid (remember that the supports of the largest interpolation operators on layer Λ^{add} have one row and one column of the grid removed), we calculated the eigenvalue spectrum of the full Dirac operator but with different boundary conditions. It is not necessary to do the same analysis for the bosonic case because in the previous section we already saw that imposing Dirichlet boundary conditions does not raise the eigenvalue much. This is obvious, taking the localization into account: adding a boundary in a region where the eigenmode is extremely small will not affect it much. Something similar happens when we change the boundary conditions for the scalar Laplace operator: The lowest eigenmode changes its wavelength from infinity (as it is constant) to twice the grid length therefore its eigenvalue is not raised higher than the second-highest eigenvalue with periodic boundary conditions.

Now let us go back to the Dirac operator: We compared Dirichlet, periodic and anti-periodic boundary conditions, where the anti-periodicity was only in one direction. This latter choice of boundary conditions is not really different from the periodic one because we can get anti-periodic boundary conditions from periodic ones by multiplying the appropriate link matrices by -1 so averages over all configurations should give identical results.

We simulated at $\beta = 1.0$ on lattices of size 18^2. Figure 30 shows the result: Indeed the Dirichlet boundary conditions raise the lowest eigenvalue so that it is higher than the lowest *four* eigenvalues of the case with periodic or anti-periodic boundary conditions whereas the latter two are nearly identical as they should be. Also shown is the eigenvalue spectrum of the Dirac operator with anti-periodic boundary conditions and with one point removed from the grid, i.e. all connections to one arbitrarily chosen point were cut. This does not affect the eigenvalue spectrum much.

Why is the Dirac operator so different from the Laplacian? Its lowest modes seem to have some global structure which forbids us to cut them at some places, as we could do in the other cases. For

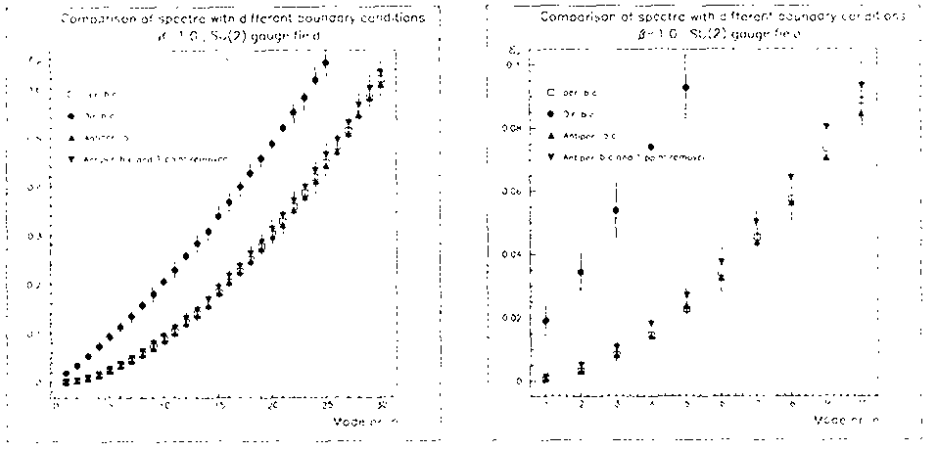


Figure 30: Low part of the spectrum of the Dirac operator with different boundary conditions. The data are averaged over twenty configurations each. All data at $\beta = 1$ on 18^2-lattices.

the scalar Laplace operator, the cuts are not doing much harm because all its modes (except for the lowest) have nodes somewhere. Imposing Dirichlet boundary conditions now lets the system arrange the nodes of its modes on the boundaries. This is possible because of translational invariance. For the bosonic Laplacian, localization does the job: Usually there will always be a localized mode with a localization center not too close to the boundary.

The Dirac operator neither has nodes in its eigenmodes that can be matched with prescribed boundaries, nor are its lowest modes localized. Hence imposing Dirichlet boundary conditions raises the eigenvalues considerably.

What does this tell us for the design of multigrid algorithms? Surely we have to use some boundary conditions for the interpolation operators since they must not extend over the whole grid. But choosing the boundaries simply by using cubic supports will not be right. It seems to be necessary to calculate the best shapes for the supports such that the eigenvalues on the supports will be as small as possible.

One might think of choosing dynamical blocks (as it is done in the Algebraic Multigrid) which also seems justified because the coupling strengths of the Dirac operator are strongly varying at small β. (We define the coupling strength simply as the norm of the connecting SU(2)-vector.) This will be discussed in more detail in section 10.1, where we will identify the optimal shape of the supports.

Even this might not cure the problem. In this case, it would be necessary to deal with the lowest modes in a special way, perhaps by introducing more interpolation operators on the last layer, see section 10.2. Therefore it is an interesting question how the number of low-lying modes behaves when the grid size is increased. We studied the spectrum of the Dirac operator with periodic and with Dirichlet boundary conditions on lattice sizes 12^2–20^2. (20 configurations for each grid size have been calculated.) We define a function $B_D(n)$ which is the number of eigenmodes of the Dirac operator with periodic boundary conditions that are lower than the n-th eigenvalue of the Dirac with Dirichlet boundary conditions. This is analogous to the function $G_j(m)$ introduced in the last section, but here the meaning is reversed because now large values of $B_D(n)$ correspond to a large number of eigenmodes that are not approximated. Figure 31 shows this function at $\beta = 1$. For all grid sizes, $B_D(1) \approx 4$, so the number of modes that can not be approximated by any interpolation operator does not increase

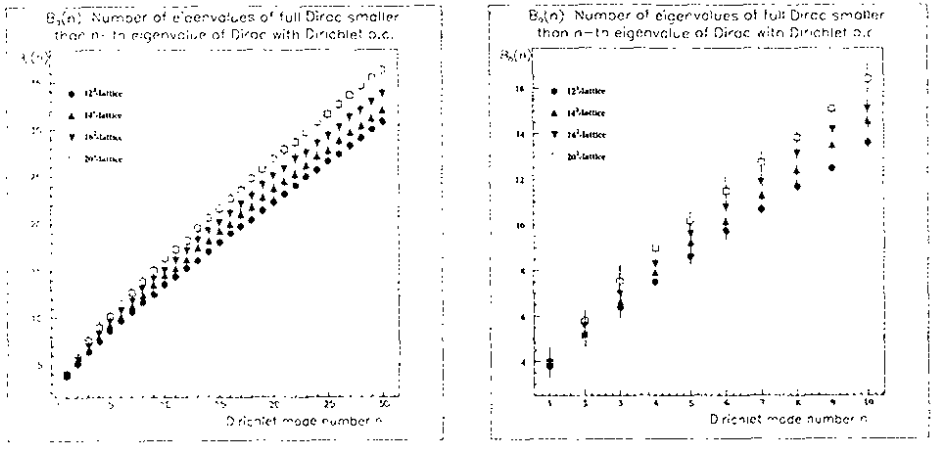


Figure 31: The function $B_D(n)$ for different grid sizes at $\beta = 1$. $B_D(n)$ is the number of eigenmodes of the Dirac operator with periodic boundary conditions that are lower than the $n-th$ eigenvalue of the Dirac with Dirichlet boundary conditions.

much with the grid size.[12] For the higher modes the function grows with the grid size, but of course the number of layers also grows with it, so this might not be a problem. If this is true also for large grids, we can deal with these modes in many ways, e.g. by introducing four interpolation operators on the last layer because the computational cost of this is proportional to the grid size and does therefore not give rise to critical slowing down.

Another possibility would be to use better boundary conditions on the block, but at the moment we do not have any suggestions how this could be done. To do this will be especially difficult because other boundary conditions would require the calculation of the effective operator to be done again and again for each block point. Although this would not give rise to critical slowing down, it would nevertheless introduce a huge constant factor to the computer time.

9.5 *The κ-Criterion*

By now we have identified the problem clearly: The Dirac-operator possesses many low-lying eigenmodes that cannot be approximated well by the interpolation operators used. It would, however, be nice to see exactly which mode makes the largest problems. In this section we will present a tool for this. It will be seen that this tool even allows a reasonable *prediction* for the convergence rate of any ISU-like algorithm with a certain prescription for determining the interpolation operators. However, as the method requires the determination of all eigenmodes, it can only be used on not too large grids.

So let us now explain the method in detail:

The fact that some eigenmodes cannot be represented well by the interpolation operators means that they possess a large contribution from a function that is *orthogonal* to all interpolation operators. (Here we have to look at the interpolation operators as a bunch of functions living on the fundamental lattice and being indexed by the coarse-grid coordinate x and the layer-index j.) In other words, we try to identify that part of the eigenmode belonging to the *nullspace* of the interpolation operator. As

[12] In these considerations, we restricted the analysis to half of the modes (the even modes) due to the total decoupling of even and odd modes for the Dirac operator. All these modes are degenerated.

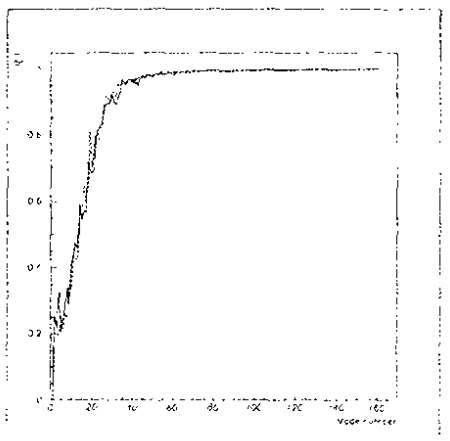


Figure 32: The norm of the orthogonal part of the eigenmodes $\|\zeta^i\|$ versus the mode number for the Dirac operator. The data are on an 18^2-lattice for three β-values (one configuration each):
Solid lines: $\beta = 1$
Dashed lines: $\beta = 2$
Dotted lines: $\beta = 20$

we saw in section 4.1.3 this part has to be eliminated by relaxation.

We therefore define a vector ζ^i for each eigenmode χ^i by calculating the following extremum:

$$\min_{s_{x,j}} \underbrace{\| \chi^i - \sum_{x,j} s_{x,j} \cdot A^{[6,j]} \|}_{= \zeta^i(z)} = \|\zeta^i\| \quad . \tag{68}$$

This defines ζ^i as that part of χ^i that is orthogonal to all interpolation operators.

Calculating ζ^i is not so difficult as it looks like; it is done by first determining from the interpolation operators an orthonormal set of functions using Schmidt's orthonormalization method. Then finding the orthogonal part of χ^i is easy.

We now ask for which modes $\|\zeta^i\|$ is small and for which it is large. It might be tempting to assume that it is small for the higher and large for the lower modes, but this is not true. The highest modes are too oscillatory to be approximated well by the interpolation operators, their $\|\zeta^i\|$ therefore should be large. *In contrast, the smoother, low-frequency modes have a smaller orthogonal part.* That this is true can be seen from figure 32, but it seems to contradict what we have said before, that the lowest modes can not be approximated well.

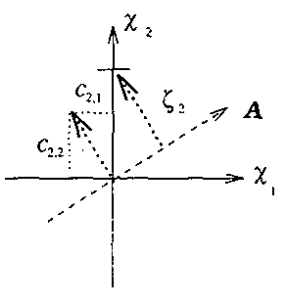
The reason for this apparent contradiction is that we have to take into account whether the orthogonal part will produce difficulties for our algorithm. We have to ask whether it has oscillatory parts that can be eliminated by relaxation or not. The criterion in section 9.3 did this in some sense: *It is not reasonable to assume that the orthogonal part of an eigenmode is a high-frequency part* when the lowest eigenvalue of the interpolation operators is much larger than the eigenvalue of the mode considered. However, the problems from which this criterion suffers (that it fails when we have a large number of high-eigenvalue interpolation operators) will not arise with the new criterion described here.

To find out if the orthogonal part is oscillatory or not, the first idea would be just to relax on this part and measure the reduction factor ϱ of the relaxation. Unfortunately, this is also not a good idea as can be deduced from figure 33. Here we have shown the relaxation rates of the orthogonal part ζ^i versus the mode number i after several relaxation sweeps on the fundamental level. For the low-lying modes, the picture looks as we expected, however, for the high-frequency modes the reduction factor is as bad as for the low frequencies. This is due to the fact that the used Gauß-Seidel relaxation mixes high and low frequencies. This is well-known for the simple scalar Laplace equation, where doing *checkerboard* Gauß-Seidel relaxation *maps the highest to the lowest mode.*

But instead of relaxing we can simply *expand ζ^i again into the eigenmodes*:

$$\zeta^i = \sum_j c_{ij} \chi^j \quad , \tag{69}$$

where the c_{ij} are the expansion coefficients. The following figure pictures this successive decomposition steps in an intuitive way:



Thus we know how strong the contribution from each mode to the orthogonal part is. This coefficients, however, results in n^2 numbers, n for each mode. To get one number for each mode we finally define

$$\kappa_i = \sum_j c_{ij} / \varepsilon_j \quad . \tag{70}$$

that is, we divide each coefficient by the corresponding eigenvalue and sum over all coefficients belonging to one mode. There will be an appreciable contribution only from those terms corresponding to a not too small coefficient belonging to a small eigenvalue. For the very small eigenvalues, even tiny coefficients will contribute which seems quite reasonable. Figure 34 shows κ_i for the same configurations as in the previous figures.

In studying κ_i we see that the very lowest modes do not contribute too strongly which is due to the exact treatment of the zero-mode. (The coefficient c_{i0} has to be zero for all i.) The largest contribution arises from the modes with numbers around 20. We can also see that κ_i decreases strongly with β as expected. (Remember that for $\beta = 20$ the convergence rate is reasonably good.) In fact, it seems that the *area under the curve* might be a measure for the convergence time. So we define a single number κ for each configuration by setting $\kappa = \sum_i \kappa_i$. Figure 35 shows this number versus the convergence time for several configurations on an 18^2-lattice. The correlation of the two quantities is satisfactory. (Correlating κ with β also looks quite good, but of course β is itself correlated with τ.)

So it might even be possible to *predict* a convergence time for an algorithm of which we know the interpolation operators. There is of course a strong caveat to this: in all our analysis we have not taken the relaxation procedure into account. In the end, it will surely make a difference whether we do $V(1.0)$- or $V(5.5)$-sweeps or use higher cycle indices (W-cycles, etc.). Our analysis is similar in spirit to the usual *two-grid-analysis* where one calculates reduction rates with the assumption that on

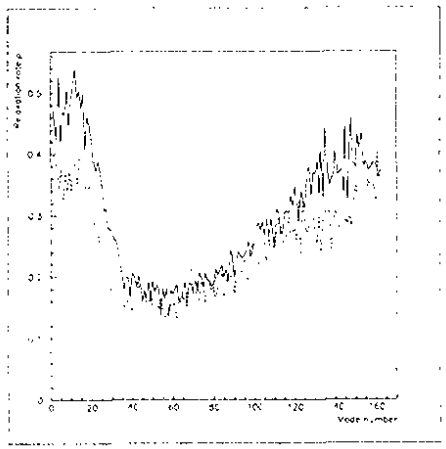


Figure 33: The relaxation rate the orthogonal part of the eigenmodes $\|\zeta^i\|$ versus the mode number. See the text for additional explanation. The configurations are the same as in the previous figure. The data are on an 18^2-lattice for three β-values (one configuration each):
Solid lines: $\beta = 1$
Dashed lines: $\beta = 2$
Dotted lines: $\beta = 20$

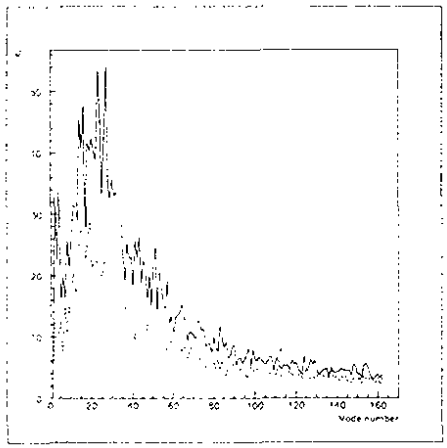


Figure 34: The quantity κ_i versus the mode number. This quantity is defined as follows: Expand the orthogonal part ζ^i of the i-th eigenvector again into the eigenvectors: $\zeta^i = \sum_j c_{ij}\chi^j$. Afterwards, divide the expansion coefficient by the eigenvalue and sum them up, $\kappa_i = \sum_j c_{ij}/\varepsilon_j$. So this should be large, when the orthogonal part has an appreciable contribution from the low modes. The configurations are again the same as in the previous figures. The data are on an 18^2-lattice for three β-values (one configuration each):
Solid lines: $\beta = 1$
Dashed lines: $\beta = 2$
Dotted lines: $\beta = 20$

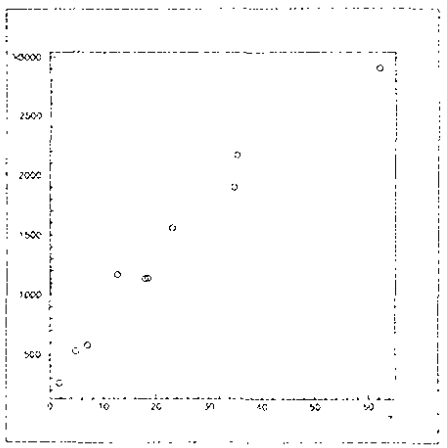


Figure 35: $\kappa = \sum_i \kappa_i$ versus the convergence time τ for ten configurations generated on an 18^3-lattice at various β-values between 1 and 20.

the coarse level the equation will be solved exactly. We implicitly made a similar assumption here: All those parts of the eigenmodes that can be represented on the coarse levels will not cause any problems at all. This is equivalent to solving all the coarse-grid equations exactly and simultaneously. It might be instructive to compare these ideas with the theorem presented in section 5.3. Despite of these idealizations, the method presented here gives valuable insight into the problems of the algorithm.

10 ISU Improved

Wherein we use what we have learned in the last chapters to make ISU much more powerful, see our hero (nearly) win and have to ask whether the price of this is to dear.

10.1 The New Blocking-Scheme

One of the problems we encountered with the algorithm in its momentary form, as explained in section 9.3, was that the eigenvalues of the interpolation operators were too large to allow a good approximation of the low-lying modes. A possible remedy might consist in choosing the block-shapes more cleverly, so that the eigenvalues are lowered.

That this is a sensible idea can also be seen from the following argument: It is known from Algebraic Multigrid that dynamical blocks are a good choice when the connection strengths between different points vary strongly. In the case of the Dirac operator, we define the connection strength between points x and $x+\mu$ as $\|\mathbf{D}(x, x+\mu)\|$. This quantity is shown in figure 36 for an 18^2-lattice at $\beta = 1$. We can learn two things from this picture: There are some variations in the strengths of the couplings Whereas the horizontal and vertical couplings have the same strength (which is clear from the definition of the operator), the diagonal couplings are varying and, at this small value of β, are usually stronger than the others. Secondly, the variations are not extremely strong. If we define a coupling $\mathbf{D}(x, x+\mu)$ as strong when $\|\mathbf{D}(x, x+\mu)\| > \gamma \cdot \max_\nu \|\mathbf{D}(x, x+\nu)\|$ with $\gamma \approx 0.25$, which is the usual definition in the AMG algorithm, then practically all connections will be called strong. In this case, we need not choose the block dynamically and can fix a blocking scheme beforehand that is appropriate for an operator with only strong connections.

Nevertheless we can lower the eigenvalues of the interpolation operators with the help of the following observation: if we choose the boundary such that the connections cut by the boundary are as weak as possible, we will get low eigenvalues. As the diagonal bonds are usually stronger than the others (at small β), we should cut as few of them as possible. This can be achieved by choosing

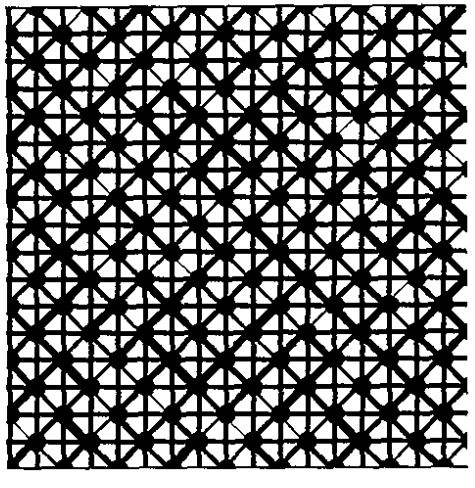


Figure 36: Connection strengths of the squared Dirac operator on an 18^2-lattice at $\beta = 1$.

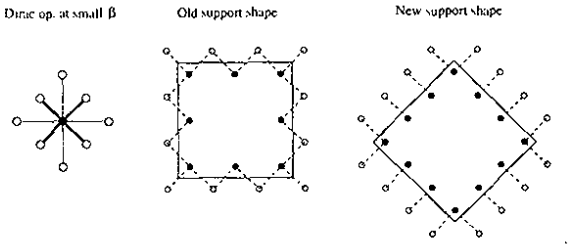


Figure 37: New supports

the boundaries themselves to be diagonal; this will decrease the number of diagonal bonds cut and increase the number of cuthorizontal and vertical bonds, as shown in figure 37. The new supports will be called diamond-shaped supports.

Perhaps it is possible to lower the eigenvalues even further by choosing the blocks still more cleverly than this? To check that indeed the diamond-shaped supports are the best supports possible, we performed the following analysis: generate a gauge field configuration and choose a point of the grid arbitrarily as a starting point of the support to be determined. Then let the support grow by taking all points into it that are strongly connected to one point of the cluster in the sense defined above. Measure the lowest eigenvalue of the Dirac operator restricted to the support as a function of the number of points in the support and perform this analysis for different values of the strength parameter γ.

We did this analysis for $\gamma = 0.5, 0.6, 0.7$ and compared the values with the usual square supports. This is shown in figure 38. We can see that the differences are not dramatic, but there is a clear tendency for the $\gamma = 0.5$-supports to have the lowest eigenvalues. It is also clear that these supports at $\gamma = 0.5$ are very regular and looking at their shapes indeed shows that these supports have the diamond-shapes explained above. (The algorithm will return such diamond-shapes when all connections are considered to be strong.)

This analysis shows that an improvement of ISU should not choose the blocks dynamically, but use the diamond-shaped supports whenever the disorder is high. At larger values of β where the diagonal couplings become weaker the old scheme can still be used.

How do we patch the diamond-shaped supports together to get a multigrid? As we use a unigrid, not a true multigrid, there is in fact no necessity to have a block-lattice with the same symmetry as the fundamental lattice, i.e. we do not need to choose an odd block-factor. Concerning the pseudoflavors, let us concentrate on one of the two decoupled sub-grids. As we are interested in the cases where β is small, the two pseudoflavors on each sub-grid are strongly coupled. Hence it is not even necessary to have interpolation operators with block-centers of both of the pseudoflavors.

Another consideration is the complexity of the effective operator. We have seen that for the old scheme with block-factor 3 the effective operator has more couplings than the fundamental operator, thereby increasing the work to be done on the coarser grids and during the calculation of the effective operators. So a good blocking prescription should give as few couplings as possible to the effective operator. The best we can expect here is to have in the effective operator the same number of couplings as in the fundamental operator.

Taking all these things into account we arrive at a new blocking prescription which we call $\sqrt{2}$-*scheme*. The choice of the block-centers and of the supports is shown in figure 39. Here the sidelength of the fundamental lattice is a power of two as in usual multigrid methods and the supports have sidelength $2^{l+1} - 1$ (3,7,15,31,...), counting the number of points along a diagonal boundary. The

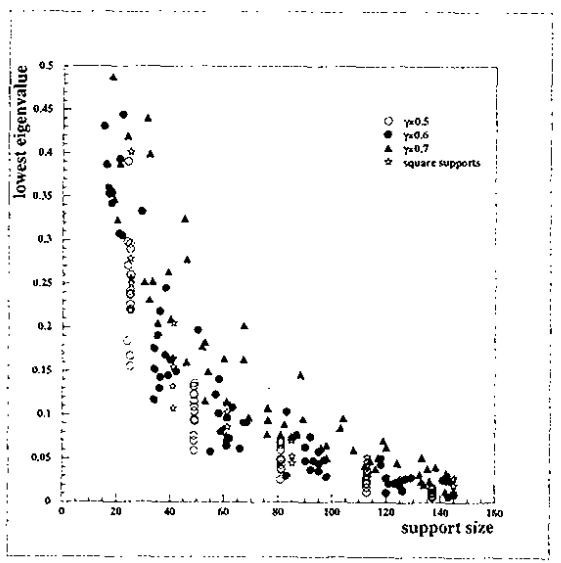


Figure 38: Eigenvalues for different support shapes. The supports corresponding to the three different values of γ were generated by a dynamical method similar to the AMG algorithm as described in the text. The square supports are the standard support shapes as shown in figure 8.

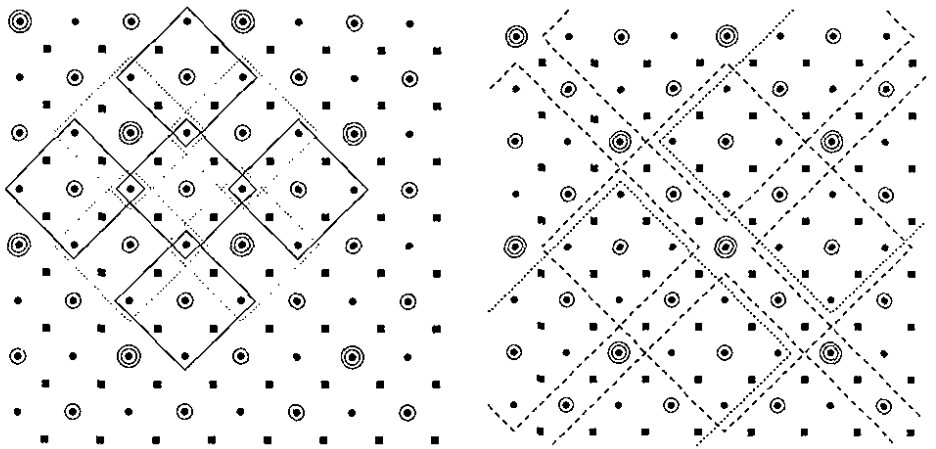


Figure 39: Choice of the block centers and the supports for the $\sqrt{2}$-scheme

sidelength of the support (in lattice units) therefore is $\sqrt{2} \cdot (2^{l+1} - 1)$.

With this choice we have several advantages:

- *We have a smaller block-factor and so more interpolation operators to approximate the low-lying* modes.
- The shape of the supports is the optimal diamond-shape.
- The effective operator is not more complicated than the fundamental operator.

As can be seen from the picture, we only have *two* block-lattices on the 16^2-lattice, one with $2 \cdot 32$ and one with $2 \cdot 8$ points. (We take a factor of 2 out because of the two sub-lattices). Symmetry would also allow an additional layer with $2 \cdot 2$ points[13], but on this layer each support would cover all of the admissible sub-lattice, except for the block-center of the other support. *Therefore we can raise the* number of last-point interpolation operators instead and use three of them (on each sub-lattice), not *one.*

Now it is time to check whether the new scheme works better than the old one. As a first test we used the κ-criterion introduced in section 9.5. Figure 40 shows κ_i for a typical configuration generated at $\beta = 1$ on a 16^2-lattice. Three different cases are pictured, corresponding to one, three, or five interpolation operators on the last point. We can see that using three interpolation operators on the last point instead of one is advantageous, whereas five of them do not decrease the κ_i much further. *Comparing this figure to figure 34 we can see the improvement clearly,* even taking into account that the lattice used here is about 20% smaller.

Our new scheme seems promising. The analysis done here shows one of the advantages of the convergence criteria introduced in the last section: We can at least estimate whether the convergence rate of a new method will be better or worse than that of a known one without having to code the full algorithm. It is only necessary to determine the interpolation operators for the new scheme with any feasible method, actually we calculated them as eigenfunctions on the supports using a standard routine [94]. The sum over all κ_i for a typical configuration at $\beta = 1$ is $\kappa \approx 671$ for three interpolation

[13] This would be similar to the additional layer Λ^{add} introduced in section 5.2

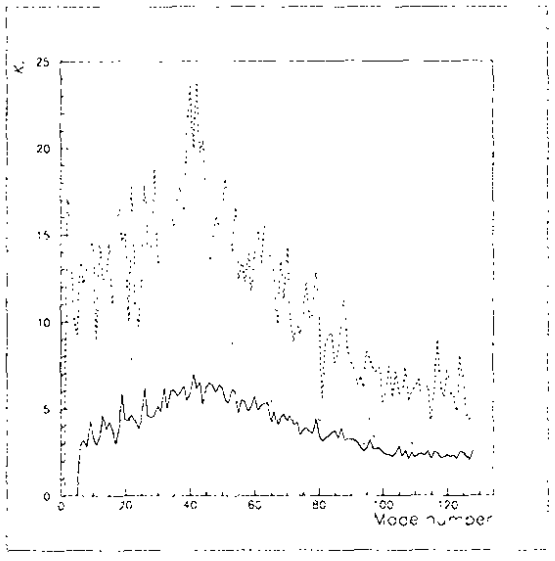


Figure 40: The κ-parameter for the $\sqrt{2}$-scheme at $\beta = 1$ on an 16^2-lattice. Compare this with figure 34.
Dashed line: One interpolation operator on the last point.
Dotted line: Three interpolation operators on the last point.
Solid line: Five interpolation operators on the last point.

operators on the last point, to be compared with $\kappa \approx 2500$ for the old scheme at $\beta = 1$. With the help of figure 35 we would estimate a convergence time τ of about 9 for this κ-value, but this of course strongly depends on the exact cycle-parameters used. Using the new scheme with a $V(2,2)$-cycle in fact gave a convergence time of $\tau = 6.7$ (at a mass value of $\delta m^2 = 2 \cdot 10^{-4}$ as used in the data of figure 35) which is reasonable.

However, there is one disadvantage of the analysis: Checking the same criterion on larger lattices costs large amounts of computer time and storage space. (Storing all eigenvectors on a 32^2-lattice requires about 32 Mbyte.) Hence one would need a better algorithm to determine the eigenvectors. Programming this seemed to be unnecessary, as the full algorithm has to be programmed anyway. So our analysis cannot *replace* the programming of the new algorithm, but it can discriminate promising from less promising ones.

10.2 Multi-Interpolation Operators

In section 9.3 we detected two "holes" in the eigenvalue spectrum, that is, two places where the eigenvalues of the interpolation operators were too large or where not enough of them were present. Although the new scheme seems to have filled these holes on small lattices as we saw in the last section, we cannot be sure that it will be as efficient on larger ones.

Therefore we introduce another improvement which, however, is a quite costly one: We allow more than one interpolation operator per block point. In this section we will develop this idea and present an algorithm for doing this dynamically; new operators will only be introduced when the need arises.

In principle, the new idea is very simple: Instead of having a set of interpolation operators $\mathcal{A}^{[0\,j]}_{zx}$, enumerated by indices j and x, we now have operators $\mathcal{A}^{[0\,j]^\alpha}_{zx}$, where α is the additional index. We call these operators *multi-interpolation operators*. It is of course obvious that the number of multi-interpolation operators must not be too large, otherwise the method will have critical slowing down again. How do we determine these operators?

The interpolation operator $\mathcal{A}^{[0\,j]^{\alpha=0}}_{zx}$ is determined in the usual way as approximate solution of an eigenvalue equation. This is done now with a *fixed* number of inverse iterations, so that there cannot be critical slowing down here. This means that perhaps the approximation is poor, but this does not matter much, for now the other operators enter the stage. For their determination we use the alternative method of calculating interpolation operators described in section 4.4, relying on the principle of indirect elimination.

We calculate an approximate solution to the equation

$$\mathrm{D}\mathcal{A}^{[0\,j]^\alpha}_{zx} = 0 \tag{71}$$

The true solution is obviously zero, but using our algorithm we will not arrive at this solution immediately. After some iterations, the approximate solution has the shape of a *bad-converging mode* (if there is a bad-converging mode at all), and this is exactly what we are interested in: how do these modes look like? During the iteration we measure the convergence time. If it is bad, then our mode will be a helpful interpolation operator, if it is good, there are no bad-converging modes on this length-scale and at this block point, so we may proceed to the next block point. The heart of the algorithm used for this method can be found in appendix B.

In this way we go through all layers of the multigrid, always doing the test iteration to see whether there are bad-converging modes. This is done also on the last layer, where we calculate directly the bad-converging modes of the full operator. Note that this calculation will not have critical slowing down because the bad-converging part of the error has not to be reduced efficiently. (This is the same as for the calculation of the lowest eigenmode which is used as interpolation operator in the usual ISU: this calculation is not affected by the bad convergence of the mode itself.)

Nevertheless, it may happen that it takes a long time to reach the shape of the worst mode, but this will only happen if there are other bad modes as well, whose shapes are also interesting, and after calculating all of them we have to arrive at the shape of the worst mode quickly.

Of course for this method to work it must not happen that on one layer all interpolation operators are converging good and on the next layer there is a huge number of bad-converging modes in each support. If this happens, the number of multi-interpolation operators needed would increase too fast. We hope that the bad-converging modes will themselves have different length-scales, so that each layer only has to address a small part of them.

10.3 Performance of the Improved ISU

So let us put the method to the test. We used the implementation described in appendix B. The appendix also contains a table of the parameters used. Unfortunately, the algorithm contains an abundance of free parameters that can be tuned: The number of sweeps done for the eigenvalue equation, for the equation with righthandside zero, the value of the convergence time above which we consider the iteration as too slow, cycle indices, and most important the maximum number of interpolation operators allowed on each layer.

We decided to study only the last parameter in detail, as this seems to be the most important and varying the others did not change convergence much. After some preliminary studies it seemed that allowing j interpolation operators on layer j would yield acceptable convergence times of $\tau \approx 5$, even for very small β. On increasing β, the convergence times improve, as expected.

Actually, as we used a dynamical approach, i.e. took more than one interpolation operators only when the convergence was not satisfactory, the algorithm chose a smaller number of interpolation operators on the second block-lattice where the support sizes are 7^2. The allowed 2 interpolation

operators per point were not always considered to be needed, the average number used was about 1.4 at $\beta = 1$ and 1.05 at $\beta = 2$, independent of the lattice size as it should be for equilibrated gauge fields.

Only on the last point layer the number needed seemed to be higher than allowed, but this can be understood because we took one block-layer, namely the one containing only $2 \cdot 2$ points, out of the game and wanted to raise the number of interpolation operators on the last layer instead. Therefore this number should be allowed to be larger: The layer omitted would contain four points and would have the layer number N, the overall number of interpolation operators on this and the one-point layer would be $4N + 2(N + 1)$ which should be halved, as we always consider only one of the sub-lattices. (*It seems more reasonable to allow $2(N + 1)$ interpolation operators on the last layer, because the* two sub-grids do not see each other. If we look only on one sub-lattice, this would give $(N + 1)$ as it should.) So we could go as far as allowing $3N + 1$ last-point interpolation operators; this would be 16 for a 64^2-lattice.

The actual number needed seems to be smaller than this. Table 4 shows the values of the convergence times for several lattice sizes, β-values, and choices of the number of interpolation operators allowed on the last point. If we want to achieve $\tau < 5$, this seems to be possible with j interpolation operators on layer j. We will take a closer look at this table later in this section, but we can already see that we have greatly improved the convergence: Remember that with the old scheme convergence *times of $\tau \approx 250$ have been observed at $\beta = 1$ on a lattice of size 54^2; with the new scheme, this value* can be easily reduced by a factor of 50.

What are the computational costs for the algorithm in this case? The calculation of the effective operators and the orthogonalization of the interpolation operators at each point have a cost of $\mathcal{O}(n \ln^2 n)$ on each layer, yielding an overall cost of $\mathcal{O}(n \ln^3 n)$. The cost of interpolation on each point can be most easily estimated by asking how often each point is hit by an interpolation operator. This is on each layer approximately $4j$ (4 because each point lies in four blocks except for points not in the corners and j because of the j interpolation operators on each layer). This work then has to be done on each layer, so the final application of the method to the inhomogeneous equation has costs of $\mathcal{O}(n \ln^2 n)$. *For the calculation of the interpolation operators themselves an additional $\ln n$ appears.* so again we have $\mathcal{O}(n \ln^3 n)$.

Finally, let us look at the relaxation costs. They are proportional to j^2 on each layer, but the number of layer points decreases faster than this, namely like 2^{-dj} with the new blocking scheme. The *overall work can be estimated as $\sum_{j=1}^{\infty} j^2 2^{-dj} = 20/27$* [14], *so the overall work for the relaxation on the* block lattices is negligible compared to the cost of interpolation, as usual in unigrid methods.

Actually, we did not simply relax the blocked equation in the usual way. Instead we tried to solve the system of equations living at each point simultaneously by relaxation, using a fixed number of relaxation sweeps before proceeding to the next point. As the resulting system is rather small, we used 50 iterations at each point which seems to be sufficient; only at the first few cycles through the unigrid it sometimes happened that the equation was not solved to a good accuracy, but this did not affect the overall convergence [15]. One could also solve the system exactly at each point; this of course would give a larger relaxation cost, but as the number of layer points decreases as a power of two, *this would not give rise to another logarithmic factor in the overall cost, it would only increase the* constant. This should perhaps be done on the last layer, where the number of interpolation operators is quite large.

Consider again table 4. Several remarks are in order here: First of all, it is important to ensure *that the measured τ-values really belong to the asymptotic region. This has been done by lowering* the critical parameter to such a small value that τ does not change anymore. This value is $2 \cdot 10^{-4}$ on the smallest and $2 \cdot 10^{-5}$ on the largest lattice size used.

[14] This sum was calculated using Maple [93].

[15] For small lattices with very large numbers of interpolation operators this was not sufficient in a few cases, sometimes yielding very large convergence times. Whenever this occured, we increased the number of iterations, until no effect could be seen anymore.

Lattice size	β	δm^2	# of $\mathcal{A}^{[0\,N]}$	τ	# of runs
8^2	1	$2 \cdot 10^{-6}$	1	6.1 ± 1.0	22
	1	$2 \cdot 10^{-6}$	2	2.37 ± 0.20	50
	1	$2 \cdot 10^{-6}$	3	1.36 ± 0.07	50
	1	$2 \cdot 10^{-6}$	4	0.93 ± 0.06	50
	1	$2 \cdot 10^{-6}$	7	0.66 ± 0.06	50
16^2	1	$2 \cdot 10^{-4}$	3	5.4 ± 0.2	100
	1	$2 \cdot 10^{-5}$	3	5.4 ± 0.2	100
	1	$2 \cdot 10^{-5}$	4	3.05 ± 0.07	50
	1	$2 \cdot 10^{-4}$	5	2.38 ± 0.03	100
	1	$2 \cdot 10^{-5}$	5	2.34 ± 0.04	50
	1	$2 \cdot 10^{-5}$	7	1.3 ± 0.1	50
	2	$2 \cdot 10^{-5}$	4	2.39 ± 0.05	50
	4	$2 \cdot 10^{-5}$	4	1.53 ± 0.03	50
	5	$2 \cdot 10^{-5}$	4	1.36 ± 0.02	50
	8	$2 \cdot 10^{-5}$	4	1.32 ± 0.02	50
	12	$2 \cdot 10^{-5}$	3	1.33 ± 0.02	50
32^2	1	$2 \cdot 10^{-4}$	4	4.7 ± 0.2	15
	1	$2 \cdot 10^{-5}$	4	5.0 ± 0.2	15
	1	$2 \cdot 10^{-4}$	5	3.37 ± 0.07	15
	1	$2 \cdot 10^{-5}$	5	3.40 ± 0.07	15
	1	$2 \cdot 10^{-6}$	6	3.04 ± 0.09	15
	1	$2 \cdot 10^{-5}$	8	2.38 ± 0.05	15
	1	$2 \cdot 10^{-5}$	10	2.04 ± 0.03	10
	1	$2 \cdot 10^{-5}$	13	1.68 ± 0.03	10
	2	$2 \cdot 10^{-4}$	4	3.1 ± 0.1	15
	2	$2 \cdot 10^{-5}$	4	3.1 ± 0.1	15
	2	$2 \cdot 10^{-4}$	5	2.47 ± 0.06	15
	2	$2 \cdot 10^{-5}$	5	2.48 ± 0.06	15
64^2	1	$2 \cdot 10^{-4}$	5	4.33 ± 0.1	15
	1	$2 \cdot 10^{-5}$	5	6.9 ± 0.4	15
	1	$2 \cdot 10^{-4}$	6	3.95 ± 0.09	10
	1	$2 \cdot 10^{-5}$	6	5.0 ± 0.2	10
	1	$2 \cdot 10^{-4}$	7	3.69 ± 0.08	15
	1	$2 \cdot 10^{-5}$	7	4.11 ± 0.12	15
	1	$2 \cdot 10^{-6}$	7	4.14 ± 0.13	15
	1	$2 \cdot 10^{-6}$	8	3.71 ± 0.12	15
	1	$2 \cdot 10^{-6}$	10	3.28 ± 0.35	4
	1	$2 \cdot 10^{-6}$	11	3.12 ± 0.40	4
	1	$2 \cdot 10^{-6}$	13	2.81 ± 0.16	10
	2	$2 \cdot 10^{-4}$	4	3.37 ± 0.08	15
	2	$2 \cdot 10^{-5}$	4	4.5 ± 0.2	15
	2	$2 \cdot 10^{-5}$	6	3.01 ± 0.10	15
	2	$2 \cdot 10^{-4}$	7	2.45 ± 0.06	15
	2	$2 \cdot 10^{-5}$	7	2.64 ± 0.07	15
	2	$2 \cdot 10^{-6}$	7	2.62 ± 0.07	15
	4	$2 \cdot 10^{-6}$	7	1.43 ± 0.01	8

Table 4: Performance of the improved ISU algorithm using the $\sqrt{2}$-scheme and Multi-interpolation *operators. Lattice sizes range from 8^2 to 64^2.*

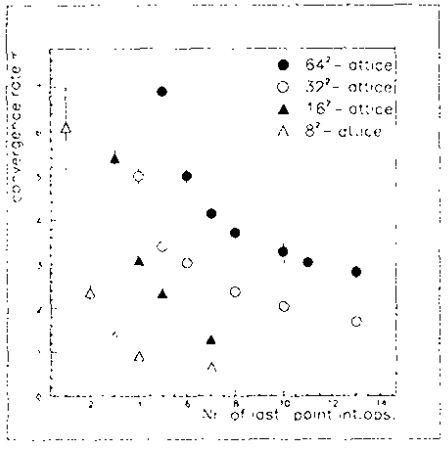


Figure 41: Convergence time τ of the improved ISU for different numbers of interpolation operators on the last point. On the other layers j interpolation operators on layer Λ^j were allowed. The data are at $\beta = 1$ at a value of δm^2 that was in the asymptotic region. The data are taken from table 4.

So let us now try to see how strong critical slowing down is for the new scheme: Figure 41 shows the convergence times τ of the algorithm with j interpolation operators on layer Λ^j at $\beta = 1$ as a function of the number of interpolation operators on the last layer. The critical parameter is always chosen at the asymptotic value so that the convergence time will not be raised when δm^2 is lowered further. The convergence time drops when the number of operators is increased, but the larger the number of operators becomes, the smaller is the improvement. It seems to be difficult to decrease the convergence time below a certain value, which is different for different lattices sizes. This value will not be a truly asymptotic value, for in the limit of n interpolation operators on the last point the convergence time will be zero as the solution becomes exact.

It is not easy to estimate a critical exponent as this depends on how we choose the growth of the number of last-point operators with the lattice size. For the fit we chose the number of interpolation operators on the last layer to grow by three when the lattice extension is doubled because this is the correct behavior as described above and it also gives the best fits. We used a χ^2-method, summarizing the results in table 5. (Data points not available, like for 12 interpolation operators on the last point of the 64^2-lattice, were estimated by linear interpolation.) From these values one can estimate a critical exponent of the improved scheme to be $z \approx 0.45 \pm 0.1$ which is a strong improvement compared to the old value of 1.6. To achieve this, the number of interpolation operators on the last point has to be large enough, otherwise larger lattices perform worse than expected. Large enough, however, is still smaller than the allowed maximal value of $3N + 1$. The possibility that on even larger lattices the performance will be worse and yield a larger critical exponent than this can not be ruled out completely by the data, but it seems not to be very likely.

If we look at the data points taken at $\beta = 2$, we see that here the critical exponent seems to be smaller. Compare the value of $\tau = 2.39$ on the 16^2-lattice with the τ-value of 2.62 on the 64^2-lattice. Here the growth of the number of last-point operators is too small (we should have taken ten interpolation operators at the larger lattice size), but still the critical exponent estimated from these two points is $z \approx 0.07$, comparing the 32^2-value with the 64^2-value, we get $z \approx 0.08$. Furthermore,

# of $\mathcal{A}^{[0,N]}$ on 8^2	on grid size 16^2	32^2	64^2	crit. exp. z	χ^2
4	7	10	13	0.55 ± 0.04	1.84
---	--	10	13	0.46	--
3	6	9	12	0.34 ± 0.03	1.5
---	6	9	12	0.32 ± 0.04	1.15

Table 5: Estimation of the critical exponent z of the improved ISU-method at $\beta = 1$, evaluated for different prescriptions of choosing the number of interpolation operators on the last point. At higher β the exponent quickly approaches zero.

judging from the two values at $\beta = 4$ on 16^2- and 64^2-lattices, there is *no critical slowing down* at this value of β, despite the fact that we took fewer interpolation operators on the last layer for the larger lattice size than usual.

So it seems that critical slowing down is appreciable only at very high values of the disorder. Remember that this was different with the old version. There we found the same critical exponent of 1.6 at $\beta = 1$ and at $\beta = 9$.

We could also adapt our scheme of taking j interpolation operators on layer Λ^j. We only did one preliminary study of this idea, allowing $j + 1$ interpolation operators instead. With this prescription, we produced data at 32^2 and 64^2 at $\beta = 1$ with $\delta m^2 = 2 \cdot 10^{-6}$ using 7 interpolation operators on the last point for the smaller and 8 for the larger lattice. The convergence times were $\tau = 1.93 \pm 0.04$ and $\tau = 2.37 \pm 0.07$ (ten configurations each), giving a critical exponent of only $z \approx 0.22$. So we see that it might be possible to reduce critical slowing down further when the parameters are optimized even at $\beta = 1$.

Table 4 also shows that increasing β leads to a drastic decrease of the convergence time. Even at $\beta = 2$ this effect can be seen clearly. There is again no critical slowing down for physically scaled gauge fields. This can be seen, e.g., comparing the data for the 32^2-lattice at $\beta = 1$ and 64^2-lattice at $\beta = 4$. On the larger lattice the convergence time already has the excellent value of 1.4. Therefore the convergence time not only shows no critical slowing down when the parameters are scaled physically (this we already achieved with the first implementation), it will also have a small absolute value.

On small lattices we also checked the behavior for much larger values of β – the convergence time quickly approaches the value of 1.3. However, it has to be kept in mind here that at large β our block scheme is probably sensitive to the righthandside of the equation: As we always use points of the same pseudoflavor as block centers, when β is large not much interpolation can be done on the other pseudoflavor points. (In the limit of $\beta = \infty$ they would decouple completely.) Our data were produced with a righthandside equal to a delta-function on a lattice point of the same pseudoflavor than the block points. If the righthandside were more complicated, involving other points as well, one could either return to the old blocking scheme or introduce a copy of the block-lattice on the other points to treat them correctly.

These results show that we have (nearly) achieved the task we set ourselves: *We have found an algorithm with no critical slowing down except at extremely high values of the disorder.*

10.4 Comparison with Conjugate Gradient

Unfortunately, this nice success might only be of theoretical nature: It might still be the case that the lattice sizes on which we will see the pay-off of our algorithm are unrealistically large. One should compare the algorithm *on CPU-time grounds* with another algorithm, namely Conjugate Gradient, which is the most-used Dirac solver today. An introduction to the Conjugate Gradient can be found in [56, 57].

However, this comparison proves difficult for several reasons:

- The two algorithms are affected by different parameters: ISU treats the zero-mode correctly, therefore it does not slow down when the lowest eigenvalue of the problem approaches zero. On the other hand, its work grows faster than the volume, whereas Conjugate Gradient only depends on the condition number. The latter is only true asymptotically, the time needed in the Conjugate Gradient algorithm until the error decays exponentially depends on the lattice size. A careful analysis of the behavior of Conjugate Gradient in gauge field backgrounds can be found in [69].
- Most of the work of ISU is done in the calculation of the interpolation operators, not in the solution of the equation itself. When we start the algorithms Conjugate Gradient begins at once to reduce the error, whereas ISU at first does nothing to it. Only after the overhead calculation is complete does ISU start to reduce the error. So the comparison also depends on *how much* we want to reduce the error.
- The last point is of a technical nature: The ISU algorithm is quite complicated. The FORTRAN implementation of it used in this work has not been designed for speed but for ease of understanding, debugging and adapting to different problems. No optimization of the algorithm has been performed whereas Conjugate Gradient is nowadays a streamline-shaped method, usually containing no superfluous calculation. In addition to this, our FORTRAN version uses a large amount of its CPU-time to calculate array indices; this could be avoided in another programming language like C or C++ where more general data structures than arrays are allowed. In addition, as was stated in the last section, the improved ISU algorithm has a vast number of parameters that could be optimized.

A thorough comparison of the two algorithms is thus beyond the scope of this work. Nevertheless, a short and cursory look at the two methods shall be done here to give at least some order-of-magnitude estimates.

We chose always to work at the small mass value of $\delta m^2 = 2 \cdot 10^{-6}$. This should be small enough to show the problems of the Conjugate Gradient algorithm, but is not too far in the asymptotic region of ISU. We decided to reduce the residual by ten orders of magnitude which seems to be a realistic value. The results are not too sensitive against a change of this value because for the ISU algorithm the overhead is the most time-consuming step whereas Conjugate Gradient needs most of its time to achieve the final convergence rate. This is another interesting difference between Conjugate Gradient and ISU: The ISU algorithm approaches its asymptotic convergence time from below, Conjugate Gradient may even increase the residual for some time.

We performed tests on lattice sizes 16^2-64^2. We used a CPU-time profiling tool [95] to measure the time spent in each of the subroutines of the program. It was found that ISU actually spends a large fraction of its time simply calculating array addresses; as this could be remedied by storing these addresses we took the most time-consuming address-routine out of the comparison (this still left many address-calculations that are embedded within other routines). We measured the factor by which Conjugate Gradient was faster than ISU, this factor was equal to 18, 12, and 6 for increasing lattice sizes at $\beta = 1$. Increasing β increased these numbers slightly, Conjugate Gradient performed worst when the disorder was high.

These numbers should not be taken too seriously, but they give us an estimate of the CPU-times needed by the two methods. Actually we believe that they are quite promising: There is a vast potential of improving the ISU program used so far. Of course Conjugate Gradient can be accelerated as well by using preconditioning and other methods. A careful comparison will be done in the future using a four-dimensional version of ISU.

In addition, the analysis showed that the most time-consuming routines of ISU are those that should be parallelizable without too much difficulties, namely the relaxation on the fundamental grid and the blocking procedures.

10.5 ISU with Dynamical Block-Centers

As we have seen, dynamical block-centers are not necessary for the solution of the Dirac equation. Other problems, like the random resistor network described in section 9.2, will involve this complication. In this section we will briefly sketch a possible scheme for calculating such block centers.

We do not intend to invent a completely new algorithm for this. Instead we want to use the method of the Algebraic Multigrid for choosing the block-centers once the connection strengths are given. For the determination of the first block lattice we can use this algorithm without any changes.

On the next layer, this is not true. As we are dealing with a unigrid, the connection strengths of the blocked operator will not contain the correct information we are looking for—they were determined using interpolation operators that can only deal with this length scale, not with all length scales. Therefore we need to find a prescription for calculating connection strengths on the block lattice which does not suffer from this problem.

The idea for this is simple, if costly: During the calculation of the interpolation operators on this layer we have measured convergence times of the algorithm on the blocks. To determine whether two blocks should be connected, we can measure the convergence time of the algorithm on the union of the two blocks using a righthandside of zero. If the convergence time on this block union is small, then the connection is weak, if the convergence time is large, the connection is strong because in this case adding an interpolation operator on the larger block would improve the convergence on the larger scale.

In this way we proceed for all points. As we only have to connect it with its neighbours, the work for each point will be proportional to the number of neighbours. Therefore the overall work will be $\mathcal{O}(n \ln^2 n (\#\text{ of neighbours}))$. The determination of the block lattices from the connection strengths itself has only costs of $\mathcal{O}(n)$ since it is taken from the Algebraic Multigrid.

Two points have to be noted here: Firstly, due to the nested way of calculating them, we always preserve the inclusion property of the supports. This means that each larger block is patched together from smaller ones on the next-smaller length scale. This is advantageous because it will ease the final iteration of the interpolation operators. If we would allow the boundaries of the large block to deviate from this prescription (for example by taking additional points in or out), then it might happen that not all of the block is covered by interpolation operators on each length scale and we might get additional critical slowing down only for the interpolation operators.

Secondly, although the method is expensive, it has the advantage of being fully in the spirit of our approach: We use the convergence of the algorithm on different length-scales to determine good interpolation operators.

So we have seen that it is possible to generalize ISU to work with dynamical block-centers. One possible future project is to implement this idea for the random resistor network to prove that with this improvement ISU will show no critical slowing down.

11 Other Multigrid Approaches to the Dirac Equation

Wherein we study other multigrid methods for the Dirac equation see that ISU is fundamentally different from them.

11.1 The Two Different Kinds of Disorder

In a Lattice Gauge Theory there are two different kinds of disorder which have to be clearly discriminated to understand other approaches to the problem of solving the Dirac equation by means of a multigrid method.

The first kind is the one we were speaking about throughout this thesis: At finite β, the field strength is non-zero and the parallel transporter around a plaquette is non-trivial, yielding a finite discrepancy vector. This is the true, physical disorder in our problem.

Due to the gauge field nature there is another kind of disorder, actually only an apparent one: Even at $\beta = \infty$ we can perform a gauge field transformation such that the gauge field looks non-trivial on a first glance.

Every multigrid algorithm must be able at least to deal with the second kind of disorder, so that its convergence rate is the same for two physically equivalent gauge fields. Two different approaches are possible here: One can either write an algorithm which is explicitly gauge covariant which is the case for ISU, or one can start with fixing the gauge to eliminate this freedom. The second approach has the disadvantage that fixing the gauge might itself yield critical slowing down.

Let us now look at the other approaches to the problem. The overview given here is based in parts on [66].

11.2 The Weizmann Approach

The group of the Weizmann Institute uses a multigrid approach that stresses the ordered features of the problem. The idea is that gauge fixing can get rid of apparent disorder and that, physically, one should be interested in large grids with large values of β.

This approach actually is part of a full multigrid project for Quantum Field Theory studied at the institute which is described in [72]. This includes multigrid methods for Monte Carlo simulations, ideas to eliminate the volume factor from some such calculations, multigrid gauge fixing algorithms, and a multigrid Dirac solver.

Here we are only interested in the approach for solving the Dirac equation. As this involves gauge fixing, one has to say, however, that at the moment the multigrid gauge fixing algorithm proposed by Brandt only has been tested in two dimensions for the gauge group U(1). It is not clear that it can be easily generalized to other cases.

The basic idea of the multigrid Dirac solver is that at large enough β the problem is more similar to a standard problem than one might think. The proposed method switches between the usual lattice discretization of the gauge fields (used for relaxation) and the so-called A-discretization which is similar to a weak coupling expansion, see [5], and uses the gauge field algebra instead of the group. The A's are used for blocking.

With this method, one can then used weighted interpolation as in the AMG context. However, there might be problems with bad-converging modes. It is proposed to deal with these modes by *recombining iterants*, a method which is similar in effect, if not in spirit, to our last-point updatings with several interpolation operators as used in section 5.4.

In our opinion, the method looks not too promising as it stands because the number of low-lying, bad-converging modes will probably grow too fast to allow eliminating them by this simple method.

Up to now, the most complete study of the ideas can be found in [73]. However, as this work does not present results for a full multigrid algorithm, but only for a two-grid analysis on a 16^2-lattice, the

status of the method seems unclear so far. It seems as if it could eliminate critical slowing down in the continuum limit, but the number of iterants that have to be recombined has not been studied as a function of the lattice size.

11.3 The Parallel Transported Multigrid (PTMG)

The Parallel Transported Multigrid algorithm [74] is formulated in a fully gauge covariant way, thereby dealing with the apparent gauge field disorder. The effective operators are calculated by parallel transporting the gauge field links that are to be eliminated to the appropriate place and then averaging over them. This is done in one direction at a time to avoid ambiguities in the prescription. The new effective gauge field links will usually not lie in the gauge group, so they are projected back to the group to make the method stable. This forfeits the Galerkin choice.

In a slight oversimplification, the method can be thought of as a usual multigrid method using weighted interpolation, but defined in a gauge covariant way through the parallel transporting. Actually, for trivial gauge it reduces to the standard multigrid method, sharing its efficiency.

Out of its definition, one should expect that the method performs well in the continuum limit. This is indeed the case. At small β the method is not efficient because it does not take the true disorder of the gauge field into account. Nevertheless, it has been reported to be competitive in parameter regions currently used in simulations [76], but as far as we know this claim has not been confirmed.

11.4 Ground State Projection Multigrid (GSPMG)

The principle of Ground State Projection Multigrid algorithms is very similar to the ISU algorithm: One chooses the interpolation operators to be the lowest eigenmodes of the restricted problem operator. This, however, is done in a multigrid fashion, not as a unigrid. This has the advantage that calculating the interpolation operators is never problematic since the equations to be solved are always restricted to small domains; however one might expect problems with such an approach as explained in chapter 4.

Three different versions have been studied which might be called the Boston, the Amsterdam, and the Hamburg approach.

11.4.1 The Boston Approach

This algorithm uses square, non-overlapping 2^2 blocks, on which the interpolation operators are calculated, using Neumann boundary conditions and the Galerkin choice. In the continuum limit this means working with constant interpolation operators. It is known that with this choice of operators, W-cycles are necessary to eliminate critical slowing down, but this is acceptable for a true multigrid. Actually, this method is not truly projecting onto the ground state. Instead, in a certain gauge, the "Lorentz Multigrid Gauge", the operators are constants for all gauge fields.

This algorithm, however, shows critical slowing down even as a function of the smallest eigenvalue at fixed β and lattice size. This is probably due to the fact that it will not deal correctly with the lowest eigenmode, except at $\beta = \infty$. Although the algorithm should work well for no disorder, critical slowing down is not eliminated in the continuum limit as long as β stays finite.

11.4.2 The Amsterdam Approach

This GSPMG also uses the Galerkin choice and is similar to the previously described method [78]. The Amsterdam group actually projects onto the ground state of the operator restricted to the block. The method used was gauge covariant for U(1) and used gauge fixing for SU(2). Even for physically scaled gauge fields, there was critical slowing down with a reported critical exponent of $z \approx 0.45$ in the U(1) case, and there was also critical slowing down for decreasing the lowest eigenvalue, as for the previous group. At very large β, the method might be competitive to Conjugate Gradient nevertheless.

11.4.3 The Hamburg Approach

This is another gauge-covariant GSPMG method [69, 70, 71]. The first goal of this group was to proof that, in principle, multigrid methods can deal with arbitrarily large disorder. To this end, a blocking operator was fixed and from it an idealized interpolation operator was determined, being based on renormalization group ideas [3, 63]. This operator could not be used in practical calculations because it is not restricted to the block, but extends over the whole lattice. In a two-grid experiment it was then shown that there is no critical slowing down in this case. This proved that multigrid methods work *in principle* in arbitrarily disordered gauge fields.

Further studies were done with manageable interpolation operators, which do not cover the whole lattice, either using the Galerkin choice or truncated idealized interpolation operators. For the Laplace equation, the method seems to eliminate critical slowing down, when an updating on the last point is added to deal with the lowest mode, see section 5.3, but for the Dirac equation it unfortunately did not.

11.5 Comparing the Other Approaches to ISU

The approaches described above fall into two categories, depending on whether they deal with the true gauge field disorder or only with the apparent disorder created by the freedom of choosing a gauge. The Weizmann and PTMG approach fall into the second category, they (should) work well for very small true disorder.

The GSPMG methods, because of projecting to a restricted eigenvalue problems, stress the underlying disorder of the problem more directly. But being true multigrid methods they differ from the ISU algorithm in a crucial way: The interpolation operators on the first block layer already have to deal with all bad-converging modes of the problem. That the Amsterdam and Boston approach are not able to do this correctly can be seen from the fact that they show critical slowing down at fixed β and grid length, when the smallest eigenvalue is decreased. The idealized GSPMG method of the Hamburg group deals with this problem through the interpolation operators that extend over the whole lattice; restricting them, however, gives rise to problems again. Another difference between the methods is that ISU uses *overlapping* blocks. That this might be a necessary ingredient was found by Kalkreuter [69].

In our opinion, it would be worthwhile to study these approaches using the analyzing tools developed in this thesis. This would enable us to understand the different convergence behavior of the methods and might increase our understanding of the problems further.

12 Summary and Outlook

Wherein we look at what we have achieved so far and consider what other tasks we might assign to ISU.

We have investigated how a multigrid method might overcome the problem of critical slowing down in the computation of propagators in gauge fields. To this end we created the ISU algorithm, based on the following principles:

- A gauge-covariant definition of smoothness which involves the problem operator.
- The principle of indirect elimination that it is easier to calculate a bad-converging mode than to eliminate it directly.
- The observation that a unigrid might be superior to a multigrid because of its greater generality in the choice of interpolation operators.
- The idea of using interpolation operators that are either bad-converging modes or eigenvectors to small eigenvalues of the problem operator restricted to overlapping blocks.
- A nested iteration method that uses already calculated interpolation operators on shorter length-scales to accelerate the calculation of interpolation operators on larger ones.

It was discussed in detail why a recent objection based on the fact that the problem operator does not possess eigenvectors in a strict mathematical sense does not invalidate our method.

The ISU algorithm is based on only a very few assumptions about the problem given. This was demonstrated by connecting it to other ideas, namely wavelet analysis, neural nets and Universal Dynamics.

The importance of correctly treating the lowest eigenmode (defined in the appropriate sense) was analyzed in some detail. Using this understanding we demonstrated the power of the principle of indirect elimination by showing how it can easily deal with almost-zero modes that arise in the presence of instantons in the gauge field, provided that the algorithm used converges well at instanton charge zero. We chose the ISU algorithm to show this, but the method generalizes to any other algorithm as well.

ISU is able to *eliminate critical slowing down completely* in the case of the covariant Laplace equation in arbitrarily disordered gauge fields. We showed how the excellent convergence is explained by the fact that the lowest modes of the Laplace operator are localized when the gauge field is disordered. An explanation for the localization, connecting it to Anderson-localization, was presented and tested. We propose to study localization further using the Transfer-Matrix approach which is the best available method for study Anderson localization. At the moment, however, it is not clear whether this method generalizes to the case of gauge theories.

Unfortunately, the success for the bosonic problem could not be repeated for the Dirac equation. Here we eliminated critical slowing down in the continuum limit, but not when the disorder was held fixed, getting a critical exponent of $z \approx 1.6$ in this case. The difference could be understood by the fact that the lowest modes of the Dirac operator in two dimensions are not localized, a fact we were not able to explain.

To analyze the difference between the two cases further we developed several tools: The *error-monitoring* expands the error into the eigenmodes of the operator to see which of them are responsible for the bad convergence. This method demonstrated that the problems were not due to the smoothing algorithm, but to the slow reduction of the low eigenmodes. It was observed that the Dirac operator possesses a large number of very small eigenvalues, but this alone can not be regarded as an explanation.

We then took a look at the *Rayleigh quotients* of the interpolation operators. As the interpolation operators have to be smooth to approximate the low eigenmodes, they must not be too oscillatory and therefore have to have small Rayleigh quotients. A criterion was given that should be fulfilled at least approximately. Here a dramatic difference could be seen between the Laplace and the Dirac operator, the first one excellently fulfilling the criterion, the latter failing in this.

The lowest eigenvalues of the Dirac operator with periodic boundary conditions are much smaller than those with Dirichlet boundary conditions. As the interpolation operators are restricted to a part of the lattice with Dirichlet boundary conditions, their Rayleigh quotients are thus raised appreciably.

Finally, we introduced the *κ-criterion*: It measures that part of the eigenmodes of the operators that is orthogonal to all interpolation operators (i.e. lies in their nullspace) and weights this part according to how oscillatory it is to see whether it can be eliminated by relaxation. This criterion was shown to be strongly correlated with the convergence time of the method.

The result of these studies can be summarized as follows: *The Dirac operator has many low-lying modes that can not be approximated well by the used interpolation operators.*

We then tried to improve our algorithm. The first idea is to adapt the boundaries of the supports in such a way that the Rayleigh quotients of the interpolation operators are lowered because we found them to be too high in the analysis. A simple prescription, using so-called diamond-shaped supports, was given and evidence was shown that a dynamical choice of the supports is not necessary for the Dirac operator.

The other improvement takes care of the fact that there are too many low-lying eigenmodes of the operator that are not approximable by the interpolation operators. To deal with this we allowed multi-interpolation operators, i.e. more than one interpolation operator may live at each block grid point. The decision how many operators should be used is made dynamically by measuring convergence times. It was found that with a simple prescription, using j interpolation operators per point on layer $\mathcal{N}$, the critical exponent at fixed β could be lowered from the old value of 1.6 to $z \approx 0.45 \pm 0.1$. On increasing β, *critical slowing down completely vanishes*, an effect already to be seen at $\beta = 4$. Furthermore, the absolute convergence times were small, so that the improved method clearly was superior to the old one.

Comparing this method to the Conjugate Gradient algorithm, which is not easy, we found that ISU was about six times slower on large grids, a result that raises hopes due to the many optimization possibilities within the program used at the moment.

Finally, we compared the ISU approach to other ideas proposed for multigrid Dirac solvers.

To summarize, we have found an algorithm that looks promising for the solution of the Dirac equation in a disordered gauge field background. Physically, the case of most interest would be to use the method in a four-dimensional Hybrid Monte Carlo simulation.

What can we say about the usefulness of the algorithm for this case?

First of all, as lattices in four dimensions usually have a smaller linear extension, the number of layers in n will not be very large, so that a factor of $\ln^3 n$ is probably not as forbidding as it looks. Our method should suffer less from its unigrid nature in four dimensions than in two.

More important, the lowest eigenmodes of the Dirac operator in four dimensions are strongly localized. The lowest eigenmodes will not have a global structure that can not be approximated by interpolation operators, so the case of four dimensions might prove easier than that of two.

Finally, if we embed ISU in a standard Hybrid Monte Carlo the changes in the gauge fields will not be very large between two Dirac solution steps. Therefore the interpolation operators will not have to be calculated completely anew, the old operators still contain useful information. This effect could considerably reduce the overhead of our method which was the most costly part. Of course, if we had an ideal Hybrid Monte Carlo algorithm without critical slowing down in the gauge field updates this would no longer be true, but at the moment no such algorithm exists.

So one future project will be to program a four-dimensional version of ISU that can be used in a Hybrid Monte Carlo calculation. We will try to optimize the program and study the possibility of

parallelizing it. A careful comparison with Conjugate Gradient will be done to settle the question whether ISU is competitive or even superior.

In addition to this the method is general enough to be used for other disordered problems. It would be *especially efficient when we have to solve the same equation with several righthandsides because* in this case the interpolation operators will only have to be calculated once.

Our search for a multigrid Dirac solver in two dimensions is at an end. We have found an algorithm with small critical exponent even at extremely large disorder. We have also learned much about *propagators: We discovered localization for the Laplace operator, but not in the Dirac case, and we* saw that the lowest modes of the fermionic propagator in two dimensions have a global structure not allowing to approximate them by a few localized objects. The success of the improved ISU algorithm *was only possible through this understanding.*

We will continue our enterprise in four dimensions, hopeful that ISU will be able to repeat its success there.

ACKNOWLEDGMENTS

This quest for a multigrid method for propagators has been aided by many people. I wish to express my thanks to all of them.

None of it would have been possible without the support and encouragement of Gerhard Mack. His inexhaustible wealth of ideas led out of many blind alleys.

I am indebted to Steffen Meyer, who performed some calculations on his workstations for me and made me aware of the random resistor problem. He also showed great interest in the method and helped with many ideas.

I feel also grateful to Claudio Rebbi, Alan Sokal, Rajan Gupta, Achi Brandt and Sorin Solomon for their hospitality through my visits in the USA and Israel and many interesting conversations.

Thomas Kalkreuter shared much of his multigrid experience and some of his programs with me. Hermann Dilger provided me with a copy of his Hybrid Monte Carlo program for two-dimensional U(1) gauge fields and helped me understanding it. He also teached me much about instanton problems.

Further thanks are due to all the other people who helped with suggestions and discussions, among them Rabi Ben-Av, Gyan Bhanot, Martin Grabenstein, Markus Grabowski, Max Griessl, Hans Joos, Ute Kerres, Paul Lauwers, Ruben Levy, Annette Meyer, Bernhard Mikeska, Hubert Simma, Thomas Wittlich, Ulli Wolff and York Xylander.

I am also obliged to Nils Kranz, who carefully read the manuscript and straightened out my style (but is of course not responsible for any blunders left), and to the members of the DESY Computer Center, who were always ready to help with any technical problems.

Financial support by the Deutsche Forschungsgemeinschaft is gratefully acknowledged.

REFERENCES

[1] F. Wilczek (Princeton, Inst. Advanced Study), *Status of QCD*, IASSNS-HEP-93-69, 1993, <Bulletin Board: hep-ph@xxx.lanl.gov - 9311302>

[2] K. G. Wilson, Phys. Rev., **D 10**, 1974, 2445

[3] K. Gawedzki, A. Kupiainen, Commun. Math. Phys. **77**, 1980, 31
K. Gawedzki, A. Kupiainen, Commun. Math. Phys. **99**, 1985, 197
K. Gawedzki, A. Kupiainen, Les Houches Sum. Sch. 1984,185
see also [63]

[4] T. Balaban, *Comm. Math. Phys.* **95**, *1984*, 17

Lattice Gauge Theory

General Reviews and Overviews

[5] M. Creutz, *Quarks, Gluons, and Lattices*, Cambridge University Press, Cambridge 1983

[6] A. S. Kronfeld, Fermilab-Conf-92/040-T, TASI Summer School *Perspectives in the Standard Model*, Boulder TASI 91, 421

[7] A. S. Kronfeld, P. B. Mackenzie, Ann. Rev. Nucl. Part. Sci. **43**, 1993, 793

[8] M. Creutz, (Brookhaven), BNL-49465, 1993
< Bulletin Board: hep-lat@ftp.scri.fsu.edu - 9309016>

[9] G. Boyd, M. Sc. thesis, University of Capetown, 1989

[10] T. DeGrand, (Colorado U.), COLO-HEP-322, 1993

Numerical Results in Lattice Gauge Theory—Overviews

[11] T. DeGrand (Colorado U.), COLO-HEP-275, 1992 , in Schladming 1992, Proceedings, *Computational methods in field theory*, 159

[12] M. Creutz, *Quantum fields on the computer*, Singapore, World Scientific, 1992

[13] T. DeGrand, *Techniques and Results for Lattice QCD Spectroscopy*, in [12]

[14] F. Karsch, E. Laermann, Rept. Prog. Phys. **56**, 1993 1347

[15] A. D. Sokal, *Bosonic Algorithms*, in [12]

[16] M. Creutz, *Algorithms for simulating fermions*, in [12]

[17] R. Gavai, in [12]

[18] C. Bernard, A. Soni, in [12]

[19] B. Petersson, Nucl. Phys. **B 30**, (Proc. Supp.), 1993, 66

[20] F. Csikor , Z. Fodor, J. Hein, K. Jansen, A. Jaster, I. Montvay, DESY-94-222, 1994,
<Bulletin Board: hep-lat@ftp.scri.fsu.edu - 9411052>

Numerical Results in Lattice Gauge Theory—Special Numerical Work

[21] M. Creutz, L. Jacobs, C. Rebbi, Phys. Rev. Lett. **42**, *1979*, *1390*
M. Creutz, Phys. Rev. Lett. **43**, 1979, 553
K. G. Wilson, in *Recent Developements in Gauge Theories*, ed. 't Hooft *et al.*, Plenum Ress, New York, *1980*

[22] F. Butler, H. Chen, J. Sexton, A. Vaccarino, D. Weingarten, Phys. Rev. Lett. **70**, 1993, 2849

[23] UKQCD collaboration, Liverpool preprint LTH 303, 1993
< Bulletin Board: hep-lat@ftp.scri.fsu.edu - 9304012>

[24] M. Lüscher, R. Narayanan, R. Sommer, P. Weiß, U. Wolff, Nucl. Phys. **B 30**, (Proc. Supp.), 1993

Numerical Results in Lattice Gauge Theory—Special Algorithms

[25] S. Duane *et al.*, Phys. Lett. **B 195**, 1987, 216
R. Gupta, G. W. Kilcup, S. R. Sharpe, Phys. Rev. **D 38**, 1988, 1278

[26] M. Creutz *et al.*, Phys. Rep. **95**, 1983, 201

[27] R. H. Swendsen, J.-S. Wang, Phys. Rev. Lett. **58**, 1987, 86
U. Wolff, Phys. Rev. Lett. **62**, 1989, 361
U. Wolff, Nucl. Phys. **B 17**, (Proc. Supp.), 1990, 93

[28] R. Ben-Av, G. Bhanot, Phys. Lett. **B 305**, 1993, 131

[29] The program used for generating the quenched and unquenched U(1) gauge field configurations was written by G. Schierholz and H. Dilger.

Other Topics in Lattice Gauge Theory

[30] J. Kogut, L. Susskind, Phys. Rev. **D 11**, 1975, 395
T. Banks, J. Kogut, L. Susskind, Phys. Rev. **D 13**, 1976, 1043
L. Susskind, Phys. Rev. **D 16**, 1977, 3031
H. Sharatchandra, H. Thun, P. Weiß, Nucl. Phys. **B 192**, 1981, 205

[31] T. Banks, A. Casher, *Nucl. Phys.* **B 167**, *1980*, *103*
T. Banks, A. Casher, Nucl. Phys. **B 169**, 1980, 215

[32] M. Atiyah, I. Singer, Ann. Math. **87**, 1968, 484

[33] H. Dilger, DESY-94-145, 1994,
<Bulletin Board: hep-lat@ftp.scri.fsu.edu - 9408017>
H. Dilger, DESY 93-181

[34] K. G. Wilson, in *New Phenomena in Subnuclear Physics*, ed. A. Zichichi, Plenum Press, New York, 1977
H. B. Nielsen, M. Ninomiya, Nucl. Phys. **B 185**, 1981, 20, *Errata* Nucl. Phys. **B 195**, 1982, 541
H. B. Nielsen, M. Ninomiya, Nucl. Phys. **B 193**, 1981, 173
H. B. Nielsen, M. Ninomiya, Phys. Lett. **B 105**, 1981, 219
L. H. Karsten, Phys. Lett. **B 104**, 1981, 315

[35] M. Lüscher, P. Weiß, Nucl. Phys. **B 290**, 1987, 25
M. Lüscher, P. Weiß, Nucl. Phys. **B 295**, 1988, 65
see also H. Neuberger, Nucl. Phys. **B 17**, (Proc. Supp.), 1990, 17

[36] C. Rebbi, talk given at the *Workshop on Multiscale Phenomena*, Eilat, February 1995

[37] C. Rebbi, private communication

[38] T. Kalkreuter,Phys. Rev. **D 51**, 1995, 1305

[39] G. Mack, Nucl. Phys. **B 235** [FS11], 1984, 107

[40] I. Montvay, DESY-Preprint 92-001
T. Kalkreuter, private communication

Anderson Localization

[41] J.T. Edwards, D.J. Thouless, J. Phys. C **5**, 1972, 807

[42] P. W. Anderson, Phys. Rev. **109**, 1958, 1492

[43] D. J. Thouless, Phys. Rep. **13**, 1974, 93

[44] Y. Nagoka, H. Fukuyama (Ed.), *Anderson Localization*, Springer Series in Solid State Physics, 1982

[45] J. Fröhlich, T. Spencer, Phys. Rep. **103**, 1984, 9
J. Fröhlich, T. Spencer, Comm. Math. Phys. **88**, 1983, 151

[46] D. J. Thouless, in R. Balian (Ed.), *Ill-condensed matter*, Les Houches 1978, North Holland 1979

[47] J. L. Pichard, G. Sarma, J. Phys **C14**, 1981, L 127 and L 617
A. MacKinnon, B. Kramer, Phys. Rev. Lett. **47**, 1981, 1546
A. MacKinnon, B. Kramer, Zetischr. Phys. **B 53**, 1983, 1

[48] E. Abrahams, P. W. Anderson, D. C. Licciardello, T. V. Ramakrishnan, Phys. Rev. Lett. **42**, 1979, 673

[49] K. Ishii, Progr. Theor. Phys. Suppl. **53**, 1976, 77

Random Resistor Network

[50] R. G. Edwards, J. Goodman, A. Sokal, Phys. Rev. Lett **61**, 1988, 1333

[51] This algorithm was written by J. Goodman and given to me by A. Sokal.

[52] A. Coniglio, J. Phys **A 15**, 1982, 3829

[53] G. G. Batrouni, A. Hansen, J. Stat. Phys. **52**, 1988, 747
G. G. Batrouni, A. Hansen, M. Nelkin, Phys. Rev. Lett **57**, 1986, 1336

Numerical Mathematics

General References

[54] R. Varga, *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1962

[55] D. Young, *Iterative Solutions of Large Linear Systems*, Academic Press, New York 1971

[56] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, *Numerical Recipes*, Cambridge UP, 1989

[57] J. R. Shewchuk, Carnegie Mellon University CMU-CS-94-125, 1994

Multigrid Method in General

[58] A. Brandt, *Multigrid Techniques: 1984 guide with applications to Fluid Dynamics*, GMD-Studie Nr. 85

[59] W. Hackbusch, *Multigrid Methods and Applications*, Springer Series in Computational Mathematics 4, 1985

[60] W. L. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia 1987

[61] J. Ruge, K. Stüben, in S. McCormick, *Multigrid Methods*, Frontiers in Applied Mathematics Vol. 5, SIAM, Philadelphia 1987

Multigrid Method in Lattice Gauge Theory

[62] G. Parisi, Cargèse Lectures, in G. t'Hooft et. al., *Progress in gauge field theory*, Plenum Press, New York 1984

[63] G. Mack, Cargèse Lectures, in G. t'Hooft et. al., *Nonperturbative Quantum Field Theory*, Plenum Press, New York 1988

[64] J. Goodman, A. Sokal, , Phys. Rev. **D40**, 1989, 2035

[65] R. C. Brower, K. J. M. Moriarty, E. Myers, C. Rebbi, *The multigrid method for fermion calculations in quantum chromodynamics*, in : S. McCormick, *Multigrid Methods*, Marcel Dekker, New York, 1988

[66] T. Kalkreuter, proceedings of *Modelling 94*, to appear in J. Comp. Apl. Math.
<Bulletin Board: hep-lat@ftp.scri.fsu.edu - 9409008>

[67] A. Sokal, Phys. Lett. **B317**, 1993, 399

[68] T. Kalkreuter, G. Mack, M. Speh, Int. J. Mod. Phys. **C3**, 1992, 121

[69] T. Kalkreuter, Phys. Rev. **D 48**, 1993, 1926
T. Kalkreuter, Ph.D. thesis, preprint DESY-92-158 – a shortened version has appeared in Int. J. Mod. Phys. **C 5**, 1994, 629

[70] T. Kalkreuter, Nucl. Phys. **B 34** (Proc. Supp.) 1994, 768

[71] T. Kalkreuter, preprint DESY-94-150

[72] A. Brandt, Nucl Phys. **B 26**, (Proc.Suppl.), 1992 , 137

[73] M. Rosantzev, M. Sc. thesis, 1993

[74] R. Ben-Av, M. Harmatz, P. G. Lauwers, S. Solomon, Nucl. Phys. **B 405**, 1993, 623
R. Ben-Av, M. Harmatz, P. G. Lauwers, S. Solomon, Nucl. Phys. **B 374**, 1992, 249

[75] M. Harmatz, P. Lauwers, S. Solomon, T. Wittlich, Nucl. Phys. **B 30** (Proc. Supp.), 1993 , 192

[76] V. Vyas, Wuppertal preprint WUB 91-10, 1991
V. Vyas, Wuppertal preprint WUB 92-30, 1992

[77] R. C. Brower, R. G. Edwards, C. Rebbi, E. Vicari, Nucl. Phys. **B 366**, 1991, 689

[78] A. Hulsebos, J. Smit, J. C. Vink, Nucl. Phys. **B 368**, 1992, 379

[79] R. G. Edwards, Ph.D. thesis, New York University, 1989

[80] M. Bäker, T. Kalkreuter, G. Mack, M. Speh, Int. J. Mod. Phys. C 4, 1993, 239

[81] M. Bäker, Nucl. Phys. B 42, (Proc. Supp.), 1995, 846
<Bulletin Board: hep-lat@ftp.scri.fsu.edu - 9411054>

[82] M. Bäker, Int. J. Mod. Phys. C6, 1995, 85

[83] M. Bäker (Hamburg U.), diploma thesis (in German), March 1993

[84] M. Bäker, G. Mack, M. Speh, Nucl. Phys. B 30, (Proc. Supp.), 1993, 269
<Bulletin Board: hep-lat@ftp.scri.fsu.edu - 9211031>

[85] M. Bäker, Nucl. Phys. B 42, (Proc. Supp.), 1995, 849

Wavelets, Neural Nets, etc.

[86] C. K. Chui (ed.), *An introduction to wavelets*, San Diego, USA: Acad. Pr., 1992
C. K. Chui (ed.), *Wavelets: A tutorial in theory and applications*, Boston, USA: Acad. Pr., 1992

[87] M. Farge, Ann. Rev. Fluid Mech. 24, 1992, 395

[88] D. Nauck, F. Klawonn, R. Kruse, *Neuronale Netze und Fuzzy-Systeme* (in German), Vieweg, Braunschweig, 1994

[89] J. A. Freeman, D. M. Skapura, *Neural Networks*, Addison Wesley, 1991

[90] M. Bäker, G. Mack, M. Speh, to appear in: *Handbook of Neural Computation*, IOP Publishing Ltd. and Oxford University Press, 1995

[91] S. Solomon, to appear in *Annual Reviews of Computational Physics*,
<Bulletin Board: hep-lat@ftp.scri.fsu.edu - 9411073>

[92] G. Mack, *Gauge theory of Things Alive and Universal Dynamics*, preprint DESY-94-184,
<Bulletin board: hep-lat@ftp.scri.fsu.edu 9411059>

Computational Aids

[93] Maple V Release 3, Copyright (c) 1981-1994 by Waterloo Maple Software and the University of Waterloo

[94] Numerical Algorithms Group Library Release Mark 16, Numerical Algorithms Group Ltd., Oxford, United Kingdom

[95] Pixie, release 5.2, Silicon Graphics, see *IRIS FORTRAN 77 reference manual*

Appendices

A Proof of the Theorem of section 8.1

In this appendix, the theorem of section 8.1 is proven. The notations used are the same as in the theorem. We start with the obvious

$$2\theta\xi^k_z = \sum_{z'} 2\theta\xi^k_{z'}\delta_{zz'} \quad . \tag{72}$$

For $z \neq z'$, $D_{zz'} + (-)^{z+z'}D_{zz'} = 0$ because of the second condition. Therefore

$$2\theta\delta_{zz'} = D_{zz'} + (-)^{z+z'}D_{zz'}$$

by means of the first condition. Inserting this into the above equation yields

$$2\theta\xi^k_z = \sum_{z'}(D_{zz'} + (-)^{z+z'}D_{zz'})\xi^k_{z'}$$

Define $\varepsilon^{n-k} = 2\theta - \varepsilon^k$:

$$(\varepsilon^{n-k} + \varepsilon^k)\xi^k_z = \sum_{z'} D_{zz'}\xi^k_{z'} + \sum_{z'}(-)^{z+z'}D_{zz'}\xi^k_{z'} \quad .$$

Using the fact that ξ^k is an eigenvertor we get

$$\begin{aligned}
\varepsilon^{n-k}\xi^k_z &= \sum_{z'}(-)^{z+z'}D_{zz'}\xi^k_{z'} \\
\varepsilon^{n-k}(-)^z\xi^k_z &= \sum_{z'}(-)^{z'}D_{zz'}\xi^k_{z'} \\
\varepsilon^{n-k}((-)^z\xi^k_z) &= \sum_{z'} D_{zz'}((-)^{z'}\xi^k_{z'}) \quad ,
\end{aligned}$$

which is the eigenvalue equation for ξ^{n-k} defined above.

B Implementation of the Multi-Interpolation Operator Version of ISU

The implementation of the multi-interpolation operator idea is described here for the interpolation operators on an intermediate layer. The routine for the last-point interpolation operator is very similar.

Do the following steps for all block points and all layers of the Multigrid:

1. Calculate the interpolation operator $\mathcal{A}^{o=0}$ with the usual ISU-method and with at most `itermax` iterations. But different from usual ISU, use all multi-interpolation-kernels on finer layers for the calculation.
2. Set $\alpha = 1$.
3. Initialize $\mathcal{A}^{[0,j]} = \mathcal{A}^{o-1}$.
4. Begin the *prerelaxation*. In this step, Multigrid cycles for the equation (71) $\mathrm{D}\mathcal{A}^{[0,j]o}_{y,x} = 0$ are done. After each cycle, the new approximate solution is orthonormalized to all interpolation operators at this block-grid point *except* $\mathcal{A}^{o-1}$. The reason for this is that the interpolation operator may need some time to get an appreciable contribution which is orthogonal to $\mathcal{A}^{o-1}$, so if we orthonormalize immediately we may only see rounding errors. There are `nr_of_presweeps` presweeps done.
5. Begin the *measurement sweeps*:
6. Orthonormalize to all other interpolation operators at this point, now including $\mathcal{A}^{o-1}$.
7. Do one measurement sweep. In each sweep calculate the reduction factor ϱ which is exactly the norm of the solution after the sweep (before the sweep the norm was one).
8. Look if the reduction factor ϱ is constant because if it is not, we are not in the asymptotic region. Here "constant" means that the change is smaller than the value of the parameter `const_fraction`. If ϱ is not constant, do another measurement sweep, if it is constant, proceed with the next step. (The variable `nr_of_measurements` ensures that we will not waste to much time by waiting for the reduction factor to get constant. After `nr_of_measurements` iterations, print a warning and also proceed with the next step.)
9. Calculate $\tau = -1/\ln \varrho$. If τ is smaller than the parameter `ideal_tau`, then convergence is all right, so proceed with the next block point, step 1.
10. Orthonormalize the interpolation operator to the other interpolation operators and increase the number of interpolation operators living at this point (variable `index_size(j,x)`).
11. If the number of interpolation operators living at this point is smaller than `max_index_size` go to step 3. Otherwise print a warning that you need another interpolation operator to make the convergence time small enough and proceed with the next block point, step 1.

During the Multigrid sweeps, all multi-interpolation operators on the finer layers are used. This means that the effective operators on the finer layers are more complicated, they get two more indices α and γ: $\mathrm{D}^{o}_{\mathrm{eff}\,\alpha\gamma}(y,\mu)$. They are calculated in the usual way.

Parameter	value
`itermax`	20
`nr_of_presweeps`	10
`const_fraction`	0.02
`nr_of_measurement_sweeps`	20
`ideal_tau`	1.5
cycle index ν_1	2
cycle index ν_2	2

Table 6: The parameters used for the improved version of ISU

C Symbols Used

The restricted number of letters in the Latin and Greek alphabets, together with our reluctance to use Hebrew or other unusual letters, unfortunately forces us to use the same symbol with different meanings. To reduce the confusion, we give a short explanation of the uses of each symbol in this thesis, with the exception of obvious ones like $\pi = 3.1415\ldots$

a	Fundamental lattice constant
a	Dilatation scale of the wavelet transform
a_j	Lattice constant on layer j
b	Translation parameter of the wavelet transform
b_i	Link of a local category
c_{ij}	Expansion coefficient in $\zeta = \sum c_{ij} \chi^j$
d	Dimension
$\mathbf{e}$	Error vector
	A single bad-converging mode
$f(t)$	A signal function and
$\tilde{f}(\omega)$	its Fourier transform
$\mathbf{f}$	Righthandside of the fundamental equation
	Part of the mode $\mathbf{e}$ not approximated on a coarse grid
f, g	Arrows in a category
i, j	General indices
i	Dilatation control parameter for the dyadic wavelet transform
j	Translation control parameter for the dyadic wavelet transform
j, k	Indices for multigrid layers
k	Boltzmann's constant
l	Inverse length dimension of the operator $\mathbf{D}$
m	Eigenvalue index
m_π	Pion mass
m_q	Quark mass
n	Number of degrees of freedom in the system, $n = L^d$
$\mathbf{r}$	Residual vector
t	Time
t_{decorr}	Decorrelation time of trajectories
x, y	Points in a block lattice
$[x]$	The block on the fine lattice with center x
z	Critical exponent
z, z'	Points in the fundamental layer
$\mathcal{A}^{[0,j]}$	Interpolation operator
B	Barrier height
$B_D(m)$	Measure for number of lowest modes
C	Path in a local category
$C^{[j,0]}$	Block operator, $C^{[j,0]} = \mathcal{A}^{[0,j]*}$
$\mathbf{D}, \mathbf{D}_0$	Problem operator
$\mathbf{D}'$	Effective operator
$\mathbf{D}_j$	Effective operator on layer Λ^j
$\mathbf{D}\|_{[x]}$	Problem operator restricted to a block x
$\mathbf{D}_{AL}$	The Anderson-Laplace operator
$\not{\mathbf{D}}$	The Dirac operator
$\mathcal{D}$	Measure in the path integral, as in $\mathcal{D}U$

Continued on next page.

$\mathbf{E}$	*The diagonal part of the iteration matrix* M
$\mathcal{F}, \mathcal{F}'$	Operator classes
$\mathcal{F}^{\text{stab}}$	An operator class stable under coarsening
G	Gauge group
$G_j(m)$	Measure for low eigenvalues of the interpolation operators
H	Hamilton operator
$\mathcal{H}^j$	Space of functions on Λ^j
$I_{z,\mu}$	Current between points z and $z + \mu$
K_t	A category at time t
L	Lattice size
L_b	Block factor
$\mathbf{L}$	The lower diagonal part of the iteration matrix M
$\mathbf{M}$	The iteration matrix
N	Number of multigrid layers
$\mathbf{N}$	Matrix of iteration acting on the righthandside
$O(U, \bar{\phi}, \phi)$	*An observable*
$P_{z,\mu}$	Canonical conjugate momentum of $U_{z,\mu}$
Q	Topological charge
R	Resistance
$\mathcal{R}^{[0,j]}_x$	Blocked residual on layer Λ^j
$\mathcal{R}$	Rayleigh quotient
$\mathbf{S}$	Iteration matrix eliminating all error modes except one
S_W	The Wilson action
S_m	The matter field action
T	Temperature
$\mathbf{U}$	The upper diagonal part of the iteration matrix M
$U_{z,\mu}$	Gauge field on link (z, μ)
V	The vector space corresponding to the gauge group
	Potential in the Schrödinger equation
	Voltage in the random resistor
W	*Summed field strength*
$(W_\psi f)(b, a)$	The wavelet transform
X, Y, Z	Objects of a category
α	Multi-index
	Participation ratio
	Expansion coefficient
β	The gauge coupling
	Inverse temperature $\beta = (kT)^{-1}$
γ	Cycle index
	Coupling strength
	Multi-index
γ^μ	Dirac matrices
δm^2	Critical parameter
δt	Molecular dynamics time step size
ε	Eigenvalue, especially ε_0 as lowest eigenvalue
ζ	*Part of an eigenmode orthogonal to all interpolation operators.*
η	Scale parameter $\eta = \sqrt{\beta}/L$
θ	Value of constant diagonal of a matrix in theorem of section 8.1

ι	The identity arrow
κ	Condition number (ratio of highest and lowest eigenvalue of a matrix)
	Measure of the importance of parts of the eigenvectors orthogonal to all interpolation operators.
μ, ν	Directional indices, $\mu, \nu = 1, \ldots,$ dimension.
	Color indices, $\mu, \nu = 1, \ldots, 4$ for SU(2).
ξ	Solution to the fundamental equation.
$\tilde{\xi}$	Approximate solution to the fundamental equation.
ϱ	Residual (error) reduction rate
τ	Inverse asymptotic convergence rate or convergence time, $\tau = -1/\ln \varrho$
φ	The "wavelets father" or scaling function
$\varphi_{ij}(t)$	A special scaling function
ϕ	Pseudo-fermionic variable
χ^i	i-th Eigenmode of an operator
$\psi(t)$	The "mother wavelet"
$\psi_j^{(i)}(t)$	A dyadic wavelet function
ψ	A fermionic field
	A function on the lattice
ω	The frequency in the function $\tilde{f}(\omega)$
Δ	The Laplace operator
Δ_{scalar}	The scalar Laplace operator
ΔH	Energy change
Λ^0	The fundamental lattice
Λ^j	A block lattice
Λ^N	The last point lattice, consisting of one point