Algorithms for Lattice QCD

Alan David Simpson

Doctor of Philosophy

University of Edinburgh

1991

To my parents

and Clare

Abstract

Quantum Chromodynamics (QCD) is the present theory of the strong interactions between quarks and gluons. To simulate QCD on a computer we need to discretise the field theory onto a space-time lattice. After outlining the standard Wilson action for lattice QCD, we describe the improved Clover fermion action with reduced discretisation errors. This thesis describes various techniques required to simulate lattice QCD and their implementation on the UKQCD Grand Challenge supercomputer, Maxwell, which is a parallel computer consisting of 64 nodes. The ideas behind Monte Carlo (MC) simulation are introduced through their use to study spin systems in statistical physics. Various MC algorithms are outlined with particular emphasis on Stochastic Cluster MC and attempts to apply this to lattice gauge simulations.

One of the basic quantities to calculate in lattice QCD is the quark propagator. This requires the inversion of very large fermion matrices and takes an enormous amount of supercomputer time. We investigate a simple Red-Black preconditioning of the matrix and compare the performance of an Over-relaxed Minimal Residual inversion algorithm with various Conjugate Gradient algorithms. The quark propagators are calculated using Maxwell and we give details of our implementation of the inversion routines and the performance obtained.

We present preliminary results from an investigation into the hadron mass spectrum. These are based on a sample of 9 gauge configurations on a $24^3 \times 48$ lattice at $\beta = 6.2$. There are descriptions of how lattice masses are calculated and of Wuppertal smearing, which is a technique that may be used to improve the signal. We conclude with a comparison of the spectrum of masses obtained from the Wilson fermion action and the Clover action.

Declaration

This thesis has been wholly composed by me and contains my own work except where indicated below. The study described in Chapter 2 was performed with the initial guidance of David Richards. Chapter 3 describes some work which was the result of collaboration with Stephen Booth, David Henty and David Richards. The results in Chapter 4 were obtained as part of the UKQCD collaboration using analysis programs written by Chris Allton and Jim Simone.

Some of the material in Chapter 4 has already been accepted for publication as

- *First Results from the UKQCD Collaboration*,
  UKQCD Collaboration, C. R. Allton, H. Duong, C. T. Sachrajda, I. M. Barbour, A. Bell, C. T. H. Davies, S. J. Hands, E. Klepfish, R. M. Baxter, S. P. Booth, K. C. Bowler, S. Collins, D. S. Henty, R. D. Kenway, C. McNeile, B. J. Pendleton, D. G. Richards, J. N. Simone, A. D. Simpson, B. E. Wilkes, J. E. J. Bottomley, F. R. Devlin, I. T. Drummond, R. R. Horgan, K. Decker, A. Hulsebos, A. C. Irving, A. McKerrell, C. Michael, M. Prisznyak, P. W. Stephenson, G. Martinelli, M. Teper and J. F. Wheater,
  Nucl. Phys. **B (Proc. Suppl.)**, *to be published*

Other related work to which I have contributed includes

- *Towards the Continuum Limit of $SU(2)$ Lattice Gauge Theory*,
  UKQCD Collaboration, S. P. Booth, K. C. Bowler, D. S. Henty, R. D. Kenway, B. J. Pendleton, D. G. Richards, A. D. Simpson, A. C. Irving, A. McKerrell, C. Michael, P. W. Stephenson, M. Teper and K. Decker,
  Phys. Lett. **B**, *to be published*

Alan D. Simpson

Acknowledgements

The last 18 months have been the most intellectually stimulating of my life. For this, special thanks must go to those people with whom I have worked most closely — David Richards, David Henty and Stephen Booth. Their help and collaboration have been invaluable.

I would like to thank Brian Pendleton, for many helpful conversations, and Richard Kenway, for his supervision, advice and his careful reading of this thesis. I am very grateful for the use of various computer programs written by Chris Allton, Jim Simone, Craig McNeile and Ken Hawick. I acknowledge the generous financial support of the Department of Education for Northern Ireland.

My enjoyment of the 3 years has been enriched by the many interesting people I have met. Not least among these have been Peter Dayan, Nick Dorey, Tom Tollenaere, Hon Yau, Pete Rawlins, Jonathan Pryce, Martin Simmen and Sharri Cohn.

Finally, I must express my debt of thanks to the two people who made this thesis possible — Ken Bowler, who has given me so much encouragement and help, particularly during the long second year, and my wife, Clare, whose support throughout this time has given me the strength to finish.

Contents

Chapter 1

Monte Carlo Simulations of Lattice QCD

The first part of this chapter will give a short introduction to Monte Carlo (MC) simulations in statistical physics followed by an outline of the standard algorithms and their problems. § (1.3) describes various algorithms, called Stochastic Cluster MC methods, which are more efficient in the physically interesting region. There is a review of attempts to generalise these to gauge theories.

The rest of this chapter is about the lattice formulation of Quantum Chromodynamics, which describes how quarks and gluons interact. This will outline the discretisation of the gauge fields on the lattice and the MC algorithms which are used to simulate them. We will describe the Wilson action for lattice fermions as well as improved actions with smaller discretisation errors, such as the Clover action.

1.1 Introduction to MC Simulations

In statistical physics, we want to calculate the expectation value of the various physical observables of interest. The expectation value of an observable, $\mathcal{A}$, is defined as

$$\langle \mathcal{A} \rangle = \frac{1}{Z} \int \mathcal{D}S \ \mathcal{A}(S) e^{-\beta \mathcal{H}(S)}, \tag{1.1}$$

where the *Hamiltonian*, $\mathcal{H}$, is the total energy of the system. The integral is over the set of all possible configurations of the microscopic variables, $\{S\}$, which define the state of the statistical system. The partition function, Z, is defined as

$$Z = \int \mathcal{D}S \ e^{-\beta \mathcal{H}(S)} \tag{1.2}$$

by the requirement that $\langle 1 \rangle = 1$.

Thus a major problem in statistical physics boils down to the evaluation of large (potentially infinite) dimensional integrals. These cannot be performed except for a few trivial cases and so we need an alternative approach, e.g., simulating the system on a computer. Since computers have only a finite amount of memory, we can only simulate systems whose state can be described by a finite number of microscopic variables. One such class of systems is the set of lattice systems where the elements exist at discrete points. Many physical systems have an obvious lattice description, e.g., the ionic lattice of magnetic metals, while others require a suitable lattice discretisation. In particular, § (1.4) will describe how we discretise a continuum field theory onto a space-time lattice. For lattice systems, the Hamiltonian is generally a sum of local energy terms, i.e.,

$$\mathcal{H} = \sum_l E_l \tag{1.3}$$

where the sum is over all the local interactions, l. Simple models of magnetic materials consider the interaction between spins situated on neighbouring lattice sites. A generic partition function is

$$Z = \int \left[\prod_i \mathrm{d}\sigma_i\right] \exp\left(\beta \sum_{\langle i,j \rangle} \sigma_i \cdot \sigma_j\right), \tag{1.4}$$

where $\langle i, j \rangle$ refers to pairs of nearest neighbour sites. If the spin variables, σ_i, can only have the values ± 1, then this is called the Ising spin model, whereas, if they are unit vectors in n-dimensional space, this is the partition function for the $O(n)$ model. For gauge theories the local interactions will be products of group elements around an elementary plaquette of the lattice.

The expectation value of a lattice observable is

$$\langle \mathcal{A} \rangle = \frac{1}{Z} \sum_{\text{all configurations, } \{S\}} \mathcal{A}(S) e^{-\beta \mathcal{H}(S)}. \tag{1.5}$$

Like the integral in Eq. (1.1), this sum cannot usually be explicitly evaluated as the number of terms in the sum increases exponentially with the size of the lattice. As we are unable to evaluate all the terms in the sum, we are forced to estimate the sum over all terms by using a sample of them. The naïve approach is simply to generate a random configuration on the lattice and then to weight this with $\exp(-\beta \mathcal{H}(S))$. For N configurations, the expectation value is then

$$\langle \mathcal{A} \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathcal{A}(S_i) e^{-\beta \mathcal{H}(S_i)}}{e^{-\beta \mathcal{H}(S_i)}}. \tag{1.6}$$

The main problem with this method is that if we simply choose configurations of the microscopic lattice variables randomly, most of the configurations make only a small contribution to the total sum we are interested in.

What we really need is some method in which configurations are generated with a probability distribution $\propto \exp(-\beta \mathcal{H}(S))$. This is known as *importance sampling.* If we are able to do this, the expectation value for any observable can simply be found by averaging over the generated configurations.

$$\langle \mathcal{A} \rangle = \frac{1}{N} \sum_{i=1}^{N} \mathcal{A}(S_i) \tag{1.7}$$

How, though, are we to do this? Rather than generate the required configurations independently, [Metropolis 53] was the first to suggest using a *Markov process* to generate a sequence of configurations whose probability distribution tends to that required. It can be shown that this occurs if the Markov process satisfies *detailed balance* (see below) and is *ergodic*, i.e., there is a non-zero probability in going from configuration S to configuration S' in a finite number of steps.

1.1.1 Detailed Balance

Consider some stochastic process which transforms a configuration S into a new configuration S' which may differ by one or more terms in the Hamiltonian. This process is said to satisfy *detailed balance* if the following condition holds.

$$P(S \to S')e^{-\beta\mathcal{H}(S)} = P(S' \to S)e^{-\beta\mathcal{H}(S')} \tag{1.8}$$

where $P(S \to S')$ is the transition probability between states S and S'. Any ergodic process which satisfies this condition will generate a sequence of configurations S_i such that, for i large, any particular configuration will have probability of appearing in the sequence proportional to its Boltzmann weight, $\exp(-\beta\mathcal{H}(S))$.

1.1.2 Metropolis Algorithm

Probably the simplest algorithm which satisfies the above conditions is known as the *Metropolis algorithm* from [Metropolis 53]. A random update is proposed in the configuration and the energy change $\delta\mathcal{H}$ is evaluated. The update is then accepted with probability

$$P(S \to S') = \begin{cases} 1 & \text{if } \delta\mathcal{H} \leq 0 \\ \exp(-\beta\,\delta\mathcal{H}) & \text{if } \delta\mathcal{H} > 0 \end{cases} \tag{1.9}$$

or rejected. It is easy to see that this algorithm trivially satisfies detailed balance, and is ergodic providing the changes are chosen in a sufficiently random manner.

For lattice spin systems, the obvious update involves proposing a change to one of the spins and evaluating the change in energy for the links to its nearest neighbours. We can then either *sweep* through the lattice in some given order proposing changes to each spin successively or propose the updates to randomly chosen spins. To ensure that the algorithm satisfies detailed balance, the choice of the new spin, σ_i', only has to be reversible in the sense that

$$P(\sigma_i'|\sigma_i) = P(\sigma_i|\sigma_i').$$

We can choose the new spin independently of the old value and thus satisfy this condition trivially. However, this may produce a low acceptance rate and so we often propose a new spin from some symmetric interval around the old spin. If this region is too small, the system moves very slowly in configuration space, whereas, if the region is too large, the acceptance rate is again small. The width of the region is usually chosen so that the acceptance rate is near 50%

It is important to bear in mind that the Metropolis algorithm is equally applicable to any scheme to propose new configurations. For example, we could make the accept/reject step after a number of spins had been changed. Unless the updates had been chosen in a sufficiently intelligent manner, the acceptance rate would be vanishingly small. The *stochastic cluster* methods formulated by [Swendsen 87] and [Wolff 89a] overcome this difficulty by attempting to update physical clusters. Further details of these methods can be found in § (1.3). The *Hybrid Monte Carlo* algorithm from [Duane 87] uses an artificial, energy conserving, dynamics to update all the lattice variables simultaneously. The integration of the equations of motion produces a finite step-size error in the energy which can be corrected for using a Metropolis accept/reject step.

Though there are variant methods with similar properties, e.g., Glauber dynamics for spin systems, Metropolis is regarded as the canonical local updating procedure. It is usually relatively straightforward to apply and implement for a large range of different problems. However, the most important criterion by which to judge any MC simulation method is how much computer time is required to generate an independent configuration. Since we can sample only a finite number of configurations, this determines how unbiased a sample of states we can obtain in a given amount of time. It is obvious that successive configurations produced by the Metropolis procedure will be highly correlated. Even after each sweep of the lattice, there will still be large correlations. To quantify this we need to consider the autocorrelations of the sequence of configurations.

1.1.3 Autocorrelations

We define the autocorrelation of an observable, $\mathcal{A}$, at separation t, by the normalised correlation function

$$C_{\mathcal{A}}(t) = \frac{\langle \mathcal{A}(S_i)\mathcal{A}(S_{i+t})\rangle - \langle \mathcal{A}(S_i)\rangle^2}{\langle \mathcal{A}^2(S_i)\rangle - \langle \mathcal{A}(S_i)\rangle^2}. \tag{1.10}$$

The average is taken over the N configurations $\{S_n, \ldots, S_{n+N}\}$ where the first n configuration are discarded for thermalisation. From $C_{\mathcal{A}}(t)$, we can define the autocorrelation time in a number of ways. The exponential autocorrelation time, $\tau_{\mathcal{A}}^{\text{exp}}$, can be estimated from the fall-off of $C_{\mathcal{A}}(t)$, though the cleanest measurement is from the integrated autocorrelation time

$$\tau_{\mathcal{A}}^{\text{int}} = \frac{1}{2}\sum_{t=-\infty}^{+\infty} C_{\mathcal{A}}(t).$$

In practice we have to sum over an appropriate window in t, which should be self-consistently chosen to be a few τ. Further details can be found in the appendix of [Madras 88].

Since the uncertainty in $\mathcal{A}$,

$$\Delta\mathcal{A} \propto \sqrt{\frac{\tau_{\mathcal{A}}}{N}},$$

$\tau_{\mathcal{A}}$ determines how much computer time is necessary to reduce the statistical errors in an observable to an acceptable size. The autocorrelation time of a particular algorithm is

$$\tau = \sup_{\mathcal{A}} \tau_{\mathcal{A}}^{\text{int}}. \tag{1.11}$$

This is the quantity which determines how long it takes to produce independent configurations.

Critical Slowing Down

For nearly all MC algorithms, the autocorrelation time becomes much longer when the correlation length, ξ, becomes large. In the Metropolis algorithm, all the updates are local and the information essentially performs a random walk across the lattice. Therefore, τ grows like ξ^2, i.e., the *dynamical critical exponent* for Metropolis is $z \approx 2$.

For spin systems, ξ is large near phase transitions which are usually the most interesting regions to study. At a second order phase transition, $\xi \to \infty$ and this problem is known as *critical slowing down* (CSD). In lattice gauge theory, we want to keep the physical correlation length fixed while reducing the effect of the grid by allowing the lattice spacing to tend to zero. Again, this is the region where the correlation length in lattice units is large and we have the problem of CSD. This is thus one of the major problems facing computer simulations of systems on large lattices. The following section will outline various MC algorithms which are more efficient than Metropolis in the region where ξ is large.

1.2 MC Algorithms

Due to problems such as CSD, much effort has been put into developing more sophisticated algorithms both for spin systems and gauge theories. The following sections will describe some of these approaches which have met with a degree of success.

1.2.1 Heatbath

If the computational cost of proposing and deciding on an update to a variable, σ_i, is much less than the cost of calculating the required local potential, V_i, it may make sense to propose a number of updates to each variable before proceeding to the next. If n updates are proposed, this method is called n-*Metropolis*. As n becomes large, this algorithm effectively equilibrates the local variable with respect to its neighbours. A further improvement on this method is possible for some Hamiltonians. If we can choose a new value of σ_i with probability distribution $\exp(-V_i(\sigma_i))$, this is equivalent to ∞-Metropolis. This method is called the *Heatbath* algorithm since the σ_i has been locally equilibrated as if it had been brought into contact with a heatbath.

For a particular example, consider the XY-model in 2 dimensions, where the unit spins can be defined by their angle with some arbitrary direction. We write the Hamiltonian as

$$\mathcal{H} = -\sum_{\langle i,j\rangle} \sigma_i \cdot \sigma_j.$$

The local potential for spin i can be written in the form

$$V_i(\sigma_i) = -R_i \cos(\theta_i - \Theta_i), \tag{1.12}$$

where the sum of the neighbouring spins is written using polar coordinates,

$$(R_i, \Theta_i) = \sum_{\mu=1}^{2} (\sigma_{i+\mu} + \sigma_{i-\mu}). \tag{1.13}$$

Therefore, for a heatbath update we need to select random numbers from the distribution

$$p(\theta) \propto \exp(a\cos(\theta)).$$

This can be achieved using standard techniques for non-uniform random deviates (see e.g., Chapter 7 in [Press 88]).

For other systems, the required probability distribution may be much more complicated which makes this algorithm prohibitively expensive. However, for systems where the local potential is simple, heatbath is much more efficient than Metropolis even though it still suffers from CSD, i.e., $z_{HB} \approx 2$.

1.2.2 Overrelaxation

This was motivated by approaches to the solution of partial differential equations (PDEs) and turns out to be the most successful local MC algorithm. It involves choosing the change in the spin variable to be as large as possible while keeping the energy constant [Adler 81]. For the XY-model described above, the new angle for spin σ_i would be

$$\theta_i' = \theta_i + \omega(\Theta_i - \theta_i) \tag{1.14}$$

where Θ_i is defined by Eq. (1.13) and ω is the relaxation parameter. We normally set $\omega = 2$ so that the energy is unchanged and we avoid the need for a Metropolis accept/reject step. Since the updates are microcanonical (and deterministic) it is necessary to use this in conjunction with standard MC sweeps to satisfy ergodicity, i.e., every N^{th} overrelaxed sweep is followed by a Metropolis or Heatbath sweep.

For appropriate values[1] of N, the dynamical critical exponent, z, can usually be reduced to about 1 (see, e.g., [Gupta 88]). This is the same exponent found by [Adler 88] for the free field limit. The reduction in z is very significant as the algorithms with the smallest dynamical critical exponent will always be optimal for systems with sufficiently large correlation lengths.

At the present time, it appears that the overrelaxed update is the most efficient MC method which updates single spins. Attempts to further reduce CSD have mostly concentrated on methods which perform collective updates on appropriate sets of local variables.

1.2.3 Collective MC Methods

As we have already seen, to evaluate the very large dimensional sums in lattice physics, we implement importance sampling via an MC algorithm which is ergodic and satisfies detailed balance. To simulate the system effectively, we require MC algorithms which decorrelate the configurations as rapidly as possible, i.e., which move the system efficiently through configuration space. Critical slowing down occurs as the system becomes correlated over large length-scales and so a possible approach to solving this problem is to attempt collective updates of the system. The Metropolis method can be applied to proposed updates of arbitrarily large sets of spins. However, since $\langle|\delta\mathcal{H}|\rangle$ increases rapidly as the number of spins to be updated increases, the acceptance rate quickly goes to zero. We need to choose the collective update intelligently. There are a number of different classes of algorithms which have been developed which allow non-local updates.

Multigrid MC. This idea also comes from the solution of PDEs though it is similar in spirit to the Renormalisation Group approach in field theory. As

[1] For many systems, N_{optimal} seems to be in the range $4 \rightarrow 8$.

described in [Goodman 86], it explicitly involves updating blocked spins on varying length scales. It successfully reduces or eliminates CSD for many models but is often complicated to implement.

Fourier Acceleration. This is used along with Molecular Dynamics methods and allows updates on many length scales through a transformation to momentum space (see [Batrouni 85]). It is complicated to program and, for gauge theories, may necessitate gauge fixing.

Stochastic Cluster MC. These methods attempt to update clusters of similar spins. Neighbouring spins of similar value are linked and so the average cluster size will be related to the correlation length. Local information is used to perform a non-local update which makes the algorithms, in general, relatively straightforward to implement. § (1.3) gives a more detailed review of the use of Stochastic Cluster MC for both spin systems and gauge theories.

1.3 Stochastic Cluster MC

This section will attempt to provide some motivation for the use of the Stochastic Cluster MC methods for various lattice spin systems and for the interest in their generalisation to lattice gauge theories. § (1.3.1) will outline the original algorithms for discrete spin systems. This will be followed by a section describing how the ideas can be extended to spin systems with continuous symmetry. § (1.3.3) will indicate how these various methods satisfy detailed balance to show that they are indeed valid MC algorithms. The final section will outline the difficulties of generalising these ideas to interesting gauge systems and describe what progress has been made.

1.3.1 Cluster Updating for Discrete Spins

Since CSD is due to the long-wavelength modes, we need some method of updating configurations on all length scales. Stochastic cluster MC selects clusters of spins which can be flipped with at most a small energy change. The average size of the clusters is determined by the physics of the system.

[Kasteleyn 69] and [Fortuin 72] described an equivalence between configurations of Potts spins and configurations of clusters in the percolation problem. It was not until [Sweeney 83] that an attempt was made to turn these ideas into a working MC algorithm by simulating the percolation clusters. However, this algorithm is tricky to program and is restricted to two dimensions. Therefore, it was not until the breakthrough by Swendsen and Wang in 1987 that a truly successful algorithm resulted (see [Swendsen 87]).

The Swendsen-Wang Algorithm

This method was devised for q-state Potts models with a partition function of the form

$$Z = \sum_{\{\sigma\}} \exp\left[\beta \sum_{\langle i,j\rangle} \left(\delta_{\sigma_i,\sigma_j} - 1\right)\right], \tag{1.15}$$

where the spins may have the values $1, 2, \ldots, q$. Following the ideas of Kasteleyn and Fortuin, the algorithm in [Swendsen 87] consists of stepping through the lattice and laying down bonds (i.e., *freezing the interaction*)[2] with probability $p_f = 1 - \exp(-\beta)$ between neighbouring sites with the same spin. We define a *cluster* of sites as a set of sites connected by frozen bonds. We have now decomposed the whole lattice into clusters of identical spins, since there are no bonds between sites with different spins. If there are N_c clusters, we can now write the

[2]Throughout this section we will regard the two phrases 'laying down bonds' and 'freezing the interaction' as being interchangeable.

partition function in terms of the bond configuration as

$$Z = \sum_{\{\text{bonds}\}} p_f^b (1-p_f)^n q^{N_c} \tag{1.16}$$

where b = number of frozen bonds and n = number of interactions which did not form a bond. Each site in a cluster can now be assigned the same random spin value, and, after deleting the bonds we now have a new configuration of Potts spins. This method is ergodic since there is a non-zero probability that no bonds are formed and that each cluster consists of only one spin. We will later show that detailed balance is satisfied for algorithms of this type (see § (1.3.3)), and so this is a valid MC method.

We are able to update the configuration in a very non-local manner though growing the clusters requires only local information, and so the number of operations required for a sweep through the lattice remains proportional to the size of the system (see [Hoshen 76]). Since the new configuration of spins can be very different from the old one, we would hope that this method would go a long way towards eliminating the problem of CSD. Indeed numerical tests in [Swendsen 87] show that it reduces the dynamical critical exponent, z, from greater than 2 to less than 1 for a variety of systems. This is better than the lower bound proven for single spin-flip dynamics from [Kawasaki 72], and is a major incentive to investigate similar collective updating methods for other systems.

1.3.2 Cluster Updating for Continuous Spins

The Swendsen-Wang (SW) algorithm is restricted to Potts models and is most effective for the small values of q, which have already been well investigated. However, even the simplest continuous spin model — the 2-dimensional $O(2)$ or XY model — has been the subject of much controversy. Therefore, many people were interested in generalising SW to $O(n)$ models. The attempts by

[Niedermayer 88] were interesting if not entirely successful, so we will review them here before going on to Wolff's successful algorithm from [Wolff 89a].

The class of cluster algorithms defined in [Niedermayer 88] are applicable to a wide variety of lattice systems. They can, in principle, be used on any Hamiltonian which is a sum of local interaction terms, E_l (see Eq. (1.3)), and which is invariant under some global symmetry transformation. Consider the partition function

$$Z = \sum_{\{\sigma\}} \exp\left[-\beta \sum_l E_l(\{\sigma\})\right]. \tag{1.17}$$

The first step is to freeze the interactions with some arbitrary local probability functions $p_{f,l}[E_l(\{\sigma\})]$. This configuration of frozen interactions again defines a set of clusters. In general, when the clusters are updated to give a new spin configuration, we will have to make a Metropolis accept/reject step on the change in energy around the cluster boundaries.

To be more specific we must consider a particular choice of the bond probabilities

$$p_{f,l}(E_l) = p_f(E_l) = \begin{cases} 1 - e^{\beta(E_l - E_0)} & \text{if } E_l < E_0 \\ 0 & \text{if } E_l \geq E_0 \end{cases} \tag{1.18}$$

with E_0 as a free parameter. These probabilities are such that p is maximum when E_l is minimum i.e., bonds are made between similar spins, and so the clusters should be similar to the physical clusters.

When $E_0 \leq E_{\mathrm{min}}$ all the clusters consist of single spins and we have recovered the single-spin update. For the choice $E_0 = E_{\mathrm{max}}$, the clusters decouple, i.e., they can be updated completely independently. For Potts models, this is identical to the SW algorithm. Thus, by varying a single parameter, we have control over the average size of the clusters. However, there is no *a priori* method of determining the optimum value of E_0.

In principle, this method could be applied to a wide variety of systems and the value of E_0 changed between steps. It turns out that, for $O(n)$ models, the following algorithm is more useful (for a comparison see [Frick 89]).

The Wolff Algorithm

At the present time, the most successful algorithm for spin systems is that developed by [Wolff 89a]. This introduced two separate ideas

1. a modification of the bond probabilities, and
2. the *single cluster* (1C) update.

For the sake of continuity, we shall investigate these in the above order.

Bond Probabilities. Wolff had been experimenting with an algorithm similar to Niedermayer's but found that either the whole system tended to consist of only one cluster or else the acceptance rate in the Metropolis accept/reject step was very small. He then thought of a new probability function for the bonds which effectively maps a continuous spin system onto a random Ising model. He suggested choosing a random n-vector $\vec{r}$ and considering a cluster spin flip in the hyperplane orthogonal to $\vec{r}$. [Hasenbusch 90] was simultaneously developing an essentially identical algorithm which is conceptually simpler and so we will describe his version here. The $O(n)$ spin model is described by the Hamiltonian

$$\begin{aligned} \mathcal{H} &= -\sum_{\langle i,j\rangle} \vec{\sigma}_i \cdot \vec{\sigma}_j && (1.19) \\ &= -\sum_{\langle i,j\rangle} \sum_{\mu=1}^{n} \sigma_i^\mu \sigma_j^\mu. && (1.20) \end{aligned}$$

where $\vec{\sigma}$ is a vector in $\mathcal{R}^n$ with components σ^μ. The lattice is now decomposed into clusters with the probability of freezing the interaction between i and j being

$$p_f = 1 - \exp\left[-\beta\left(\sigma_i^1\sigma_j^1 + \left|\sigma_i^1\sigma_j^1\right|\right)\right]. \tag{1.21}$$

This is a mapping of the $O(n)$ model to a random Ising model. We can now proceed *à la SW* and change the sign of all the σ^1 in a cluster with probability 0.5 or else use the single cluster method below.

In the Wolff version, we would now carry out a random global rotation of all spins on the lattice (or, equivalently, choose a new vector $\vec{r}$). Hasenbusch suggests repeating the clustering procedure for each of the n components in turn before carrying out a random global rotation. It is not clear whether there is any particular advantage in either method.

Single Cluster Updating. The other improvement suggested by [Wolff 89a] is that rather than decompose the lattice into clusters and then flip each cluster with probability 0.5, we should grow only a single cluster at a time. This is done by choosing a site on the lattice randomly and then growing the cluster associated with that site. We therefore choose clusters weighted by their size and Wolff showed that the average size of clusters grown in this way is directly related to the magnetic susceptibility, χ. This 1C method was shown to be better than SW for Ising spins by [Wolff 89b].

The study in [Frick 89] compared the cluster methods of Wolff and Niedermayer with standard Metropolis for the $O(4)$ model in 4-dimensions. It was found that Wolff's virtually eliminated the problem of CSD while that of Niedermayer had little, if any, advantage over local updating in terms of real time. Presumably, this is because Wolff's method updates more efficiently the large physical clusters which cause CSD.

Cluster Simulations on Parallel Computers

One of the features of cluster algorithms is that a bond between 2 spins on one part of the lattice can mean that widely separated spins become members of the same cluster. This makes this type of algorithm difficult to implement on a distributed memory parallel computer where each processor only has direct access to part of the memory. Most lattice simulations on parallel computers divide the lattice geometrically over processors. This is a very inefficient approach for the 1C algorithm since only a fraction of the lattice is involved in the cluster growth. [Burkitt 89] implemented a parallel SW algorithm for Ising spins on an array of transputers. However, they found that they could only make efficient use of a small number of transputers so that the lattice size per processor remained large. Using additional processors caused the whole simulation to slow down as the amount of communication increased.

We performed simulations on a similar transputer array to investigate the Wolff algorithm for the XY-model in 2 and 3 dimensions. To make use of the advantages of the 1C algorithm, we used an approach called *task farming* where separate simulations are run on each transputer. The results obtained had reasonably small statistical errors and were similar to those in [Gupta 88] which used an overrelaxed algorithm. However, [Gupta 88] presented results on lattices of up to 512^2 while our lattices were limited by memory to about 100^2. On this size of lattice, we saw little effect from CSD but were unable to get sufficiently near to the phase transition to accurately determine the critical exponents.

While implementing cluster algorithms on standard computers is usually quite straightforward, it is often difficult to do this efficiently on parallel computers. The most reasonable approach may involve a combination of task farming and geometrical decomposition.

1.3.3 Detailed Balance for Cluster MC

The proofs of Detailed Balance for most of the methods described above (and for those in § (1.3.4)) have many features in common. In this section, we will show that the SW-limit of Niedermayer's algorithm (see § (1.3.2)) satisfies detailed balance.

Consider a general Hamiltonian which is a sum of local interactions E_l as in Eq. (1.3). The probability of freezing an interaction is

$$p_f(E_l) = 1 - e^{\beta(E_l - E_{\max})}. \tag{1.22}$$

Let the transition probabilities be of the form

$$P(S \to S') = W(S \to C)A(S, C \to S'), \tag{1.23}$$

where $W(S \to C)$ is the probability of forming a set of clusters, C, from the system configuration, S, and $A(S, C \to S')$ is the probability of forming a new system configuration, S', given a particular cluster decomposition. For simplicity we shall look at the case where

$$A(S, C \to S') = A(S', C \to S), \tag{1.24}$$

i.e., A shall have a uniform distribution.[3] [Niedermayer 88] proves a more general case which replaces this requirement with a Metropolis accept/reject step.

The detailed balance condition (see Eq. (1.8)) becomes

$$\frac{W(S \to C)}{W(S' \to C)} = \frac{e^{-\beta\mathcal{H}(S')}}{e^{-\beta\mathcal{H}(S)}}. \tag{1.25}$$

The cluster configuration C is completely defined by which interactions are frozen and which are deleted. Therefore the transition probability can be written as

$$W(S \to C) = \prod_{l \in F} p_f(E_l) \prod_{l \in D} \left[1 - p_f(E_l)\right], \tag{1.26}$$

[3] This is an important case as the equivalent of this condition is satisfied by SW, the Wolff-Hasenbusch algorithms and the simple lattice gauge algorithms outlined in § (1.3.4).

where F is the set of frozen interactions and D is the set of deleted ones. We thus have

$$\frac{W(S \to C)}{W(S' \to C)} = \frac{\prod_{l \in F} p_f(E_l) \prod_{l \in D} [1 - p_f(E_l)]}{\prod_{l \in F} p_f(E'_l) \prod_{l \in D} [1 - p_f(E'_l)]} \tag{1.27}$$

$$= \frac{\prod_{l \in D} [1 - p_f(E_l)]}{\prod_{l \in D} [1 - p_f(E'_l)]}, \tag{1.28}$$

since the frozen interactions are, by definition, unchanged. Substituting the explicit form for the probabilities, Eq. (1.22), we get

$$\frac{W(S \to C)}{W(S' \to C)} = \prod_{l \in D} \frac{e^{\beta(E_l - E_{\max})}}{e^{\beta\left(E'_l - E_{\max}\right)}} \tag{1.29}$$

$$= e^{\beta \sum_{l \in D}\left(E_l - E'_l\right)}. \tag{1.30}$$

Since the only difference in energy between the two configurations S and S' comes from the change in the deleted interactions, we have

$$\frac{W(S \to C)}{W(S' \to C)} = e^{\beta[\mathcal{H}(S) - \mathcal{H}(S')]} \tag{1.31}$$

which is the required result.

Summary of Requirements

A version of this proof can be developed for any cluster method which satisfies the following two requirements.

1. Any of the interactions which are deleted were done so with probability $\propto \exp(\beta E_l)$.
2. The new values for these interactions are chosen uniformly from the set of possibilities compatible with the frozen interactions as in Eq. (1.24).[4].

[4]As noted previously, [Niedermayer 88] showed that this condition can be adapted using a Metropolis step.

The algorithm is ergodic since there is a non-zero probability that no interactions are frozen.

Note, however, that these requirements only guarantee a *correct* cluster MC method, not that it will satisfy any measure of *goodness* relative to a standard MC method. In particular, unless the cluster MC efficiently updates the large-scale structures which cause CSD, the autocorrelation time, τ, may still be large and CSD still a major problem.

1.3.4 Cluster Updating for Gauge Theories

Ever since the introduction of cluster updating by Swendsen and Wang, there has been a great deal of interest in the possibility of using this type of method for the MC simulation of Lattice Gauge Theories (LGTs). These type of models have probably used up more supercomputer time than all other physics simulations put together, and so algorithmic improvements would be especially welcome. The only progress so far has been for various special cases, e.g., abelian gauge theories in 3D, and it is, as yet, far from clear how to generalise the ideas to the non-abelian gauge theories, such as QCD, which are of over-riding interest.

Z_2 Gauge Theory

The most important step towards the use of Cluster MC methods for gauge theories has been that made by [Benav 89]. They have successfully generalised the ideas of SW to Z_2 gauge theory in 3 dimensions which is dual to the 3D Ising spin model. In Z_2 LGT, the Ising variables live on the links of the lattice. We label the gauge field on the link from site x in direction μ as $U_\mu(x)$. The link from $x + \hat{\mu}$ to x is then $U^\dagger_\mu(x)$ which, for abelian gauge theories is simply $U^\star_\mu(x)$. The partition function for Z_2 gauge theory is then

$$Z = \sum_{\{U\}} \exp\left[-\beta \sum_{\square} E^{\square}_{\nu\mu}(x)\right], \tag{1.32}$$

where $-E^{\square}_{\nu\mu}(x) = U^{\square}_{\nu\mu}(x)$ is the product of the U's around the plaquette in the $\nu\mu$-plane from site x, as shown in Fig. 1.1. This is invariant under the gauge transformation

$$U_\mu(x) \to \Omega(x) U_\mu(x) \Omega(x+\hat{\mu}) \tag{1.33}$$

where $\Omega(x)$ is a Z_2 variable on each lattice site.

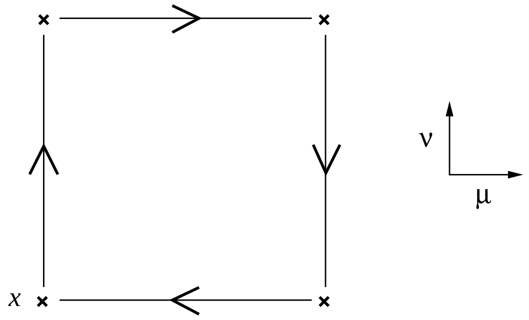


Figure 1.1: Products of gauge fields which make up $U^{\square}_{\nu\mu}(x)$.

The cluster simulations are carried out on the dual lattice, with the plaquettes being dual to links and a unit cell on the original lattice mapping to a site on the dual lattice. The continuum *Bianchi identity* $dF = 0$ is equivalent to the condition that the product of $E^{\square}$s connected to a dual site must be equal to $+1$. The plaquettes are deleted with probability

$$p = e^{\beta(E^{\square}-1)}. \tag{1.34}$$

This satisfies the requirements for detailed balance (see § (1.3.3)), providing the new configuration is chosen uniformly from those consistent with the frozen interactions. This is the major difficulty in cluster updating for gauge theories. [Benav 89] arranges this by considering the clusters of deleted interactions and using the Bianchi identity. If a *spanning tree* is constructed for each deleted cluster, then the requirement that each *loop* is randomly chosen to be ± 1 is equivalent to choosing those deleted links which are not part of the spanning tree to be ± 1. This is an explicit random choice for the interactions which were deleted. Thus, for a discrete gauge theory, the allowable configurations are enumerable and a uniform choice can be made.

So far, the results for this method are encouraging. The dynamical critical exponent, z, is reduced from about 2.5 to 0.73.[5] The major limitations are that this method appears to be restricted to 3 dimensions and is difficult to generalise even to U(1) gauge theory (see § (1.3.4)). More investigation is still needed into the usefulness of this approach before any real conclusions can be made.

$U(1)$ Gauge Theory

There has been some progress (see [Irving 90]) in extending [Benav 89] to Z_n gauge theory, for any n, but still in 3 dimensions. In the limit of n large, this is $U(1)$ lattice gauge theory with

$$E^{\Box}(i,\mu,\nu) = -\mathrm{Re}\left[U_{i,\mu}U_{i+\mu,\nu}U^{\dagger}_{i+\nu,\mu}U^{\dagger}_{i,\nu}\right] \tag{1.35}$$

$$= -\cos\theta_{\Box}, \tag{1.36}$$

where the link variables $U_{i,\mu}$ are of the form $e^{i\theta_{i,\mu}}$, and the plaquette angle

$$\theta_{\Box} = \theta_{i,\mu} + \theta_{i+\mu,\nu} - \theta_{i+\nu,\mu} - \theta_{i,\nu}. \tag{1.37}$$

This method again considers the dual lattice and the deletion probability is still Eq. (1.34). The update is carried out as before, by explicitly considering the clusters of deletions. The uniform choice for the new configuration is effected by adding a random *flux* to loops of deleted links on the dual lattice so that the Bianchi identity is not violated.

[5] An alternative, but more unwieldy approach to Z_2 or Z_3 gauge theory using *surface percolation* in [Brower 90] obtained a similar reduction of z from ~ 2.1 to $z \simeq 0.61$ supporting the idea of universality classes for cluster updating.

A preliminary investigation of this algorithm was performed on small lattices using a few transputers. Apart from the difficult problem of loop identification, there are also problems on finite lattices due to loops which encircle the lattice which requires an additional constraint. For low values of β, the autocorrelation time was found to be small. However, when β was increased, the efficiency of the algorithm rapidly decreased as the decomposition of the large clusters into loops was slow. This is a major problem as the interesting regime for $U(1)$ is at $\beta > 2$. This difficulty may be related to the fact that this algorithm is unable to embed discrete variables into the continuous variables as in [Wolff 89a].

Status of Cluster MC for Non-Abelian LGTs

Non-abelian LGTs are gauge theories where the link variables do not commute, e.g., are members of a non-abelian group such as $SU(N)$. Attempts to construct a cluster MC algorithm for such theories have all the difficulties of the continuous $U(1)$ method plus the difficulty of defining a flux around a loop. The only partial success has been for a $SU(2)$ theory on a $L^3 \times 1$ lattice by flipping the signs of the Polyakov loops (see [Benav 91]). Other recent work, e.g., [Caracciolo 91] has suggested that it may be impossible to truly generalise cluster methods to systems where the elements are members of a group with a non-trivial manifold, such as $SU(3)$.

1.4 The Lattice Formulation of QCD

QCD is the present candidate for the field theory of the strong interaction. It describes how quarks interact via gluons to form hadronic matter, e.g., protons.

The Lagrangian for QCD in the continuum is

$$\mathcal{L} = \frac{1}{4} F_{\nu\mu} F_{\nu\mu} + \bar{\psi}(\not{D} + m)\psi, \tag{1.38}$$

where the first term is the gauge or gluonic term and ψ, $\bar{\psi}$ are the fermionic quark fields. QCD is analogous to Quantum Electrodynamics which describes how charged electrons interact via photons. The main difference is that the gluons, which are the analogues of the photons, also carry colour charge and so interact with each other. This radically changes the behaviour of the field theory. Unlike QED, it is very difficult to solve QCD using perturbation theory in the coupling constant since, at low energies, the effective coupling for QCD is $O(1)$.

Regularising QCD using a space-time lattice leads to a systematically improvable, non-perturbative method of calculating the properties of the field theory using the MC techniques developed for statistical physics. To relate the results correctly to continuum physics, we need to check that the effects of the lattice discretisation are small. For a detailed review of lattice QCD, see [Kenway 89].

1.4.1 Gluons on the Lattice

In lattice QCD, the gluons are represented by $SU(3)$ matrices on the links of a 4D hypercubic lattice. As in § (1.3.4), we label the link from site x in the μ direction as $U_\mu(x)$ and the link from $x + \hat{\mu}$ to x is then $U_\mu^\dagger(x)$. Following [Wilson 74], we take the action for an $SU(N)$ gauge theory to be

$$S_G = -\frac{\beta}{N} \sum_x \sum_{\nu,\mu\neq\nu} \mathrm{Re}\ \mathrm{Tr}\left[U_{\nu\mu}^{\square}(x)\right], \tag{1.39}$$

where $\beta = \frac{2N}{g^2}$ takes the role of inverse temperature.

$$U_{\nu\mu}^{\square}(x) = U_\nu(x) U_\mu(x+\hat{\nu}) U_\nu^\dagger(x+\hat{\mu}) U_\mu^\dagger(x) \tag{1.40}$$

is the product of the gauge matrices around a fundamental plaquette of the lattice as shown previously in Fig. 1.1. For lattice spacing a, this lattice action differs from the continuum action by terms of $O(a^2)$.

1.4.2 Naïve Lattice Fermions

The quarks are fermionic fields on the sites of the 4D lattice. Since continuum fermions are antisymmetric under interchange, the lattice fermions are Grassmann variables — anti-commuting complex numbers. The obvious way to discretise the free fermion action is to replace the derivative with a central difference. This gives the Naïve fermion action,

$$S_{NF} = \sum_x \left\{ \sum_\mu \bar{\psi}(x)\gamma_\mu \frac{\psi(x+\hat{\mu}) - \psi(x-\hat{\mu})}{2} + m\bar{\psi}(x)\psi(x) \right\}. \tag{1.41}$$

Since continuum QCD is a gauge theory, we need the lattice action to be invariant under the lattice gauge transformation

$$\begin{aligned} U_\mu(x) &\rightarrow \Omega(x)U_\mu(x)\Omega^\dagger(x+\hat{\mu}) \\ \psi(x) &\rightarrow \Omega(x)\psi(x) \\ \bar{\psi}(x) &\rightarrow \bar{\psi}(x)\Omega^\dagger(x), \end{aligned} \tag{1.42}$$

where $\Omega(x)$ is a local $SU(3)$ matrix. The only physical (i.e., gauge invariant) objects are either closed loops of gauge fields such as the plaquette or else strings of gauge fields terminated by fermions, of the form $\bar{\psi}U_1U_2\ldots U_n\psi$. To make the naïve fermion action gauge invariant, we introduce gauge fields linking the fermions on different sites, to get

$$S_{NF} = \sum_x \left\{ \sum_\mu \bar{\psi}(x)\gamma_\mu \frac{U_\mu(x)\psi(x+\hat{\mu}) - U_\mu^\dagger(x-\hat{\mu})\psi(x-\hat{\mu})}{2} + m\bar{\psi}(x)\psi(x) \right\}. \tag{1.43}$$

We can write this in the matrix form

$$S_{NF} = \bar{\psi}(x)\mathcal{M}_{x,y}^{NF}\psi(y)$$

with

$$\mathcal{M}^{NF}_{x,y} = \sum_{\mu} \frac{\gamma_\mu}{2}(U_\mu(x)\delta_{x+\hat{\mu},y} - U^\dagger_\mu(x-\hat{\mu})\delta_{x-\hat{\mu},y}) + m\delta_{x,y}.$$

The free fermion propagator is the inverse of this matrix in a unit gauge configuration, i.e., $U_\mu(x) = 1$, for all x and μ. This gives

$$(\mathcal{M}^{NF}_{x,y})^{-1} = \int \frac{\mathrm{d}^4 p}{16\pi^4} \frac{e^{ip(x-y)}}{(m + i\sum_\mu \gamma_\mu \sin p_\mu)}. \tag{1.44}$$

The continuum limit is when the lattice spacing is small relative to the inverse fermion mass, i.e., $m \to 0$ in lattice units. In this limit, the dominant contribution to Eq. (1.44) comes when all the $\sin p_\mu = 0$ which occurs for $p_\mu = 0, \pi$. Therefore, this lattice action represents 2^4 continuum fermions! This problem is known as *fermion doubling*.

1.4.3 Wilson Fermions

Since only the continuum limit of the lattice fermion action is of physical interest, [Wilson 75] noticed that it is possible to avoid the problem of fermion doubling by introducing a term of the form

$$W = -\frac{r}{2}\sum_{x,\mu}\bar{\psi}(x)[U_\mu(x)\psi(x+\hat{\mu}) + U^\dagger_\mu(x-\hat{\mu})\psi(x-\hat{\mu}) - 2\psi(x)], \tag{1.45}$$

which explicitly vanishes in the continuum limit. This gives the fermion doublers masses of the order of the lattice cutoff, a^{-1}. The Wilson fermion action is then

$$\begin{aligned} S_{WF} \;\; = \;\; & -\sum_x \Big\{ \sum_\mu \bar{\psi}(x)\frac{1}{2}\left[U_\mu(x)(r-\gamma_\mu)\psi(x+\hat{\mu}) + U^\dagger_\mu(x-\hat{\mu})(r+\gamma_\mu)\psi(x-\hat{\mu})\right] \\ & + (m+4r)\bar{\psi}(x)\psi(x)\Big\}. \end{aligned} \tag{1.46}$$

Introducing the Wilson hopping parameter κ, where

$$\kappa = \frac{1}{(2m+8r)}, \tag{1.47}$$

gives

$$S_{WF} = \left(\frac{1}{2\kappa}\right) \bar{\psi}(x)\mathcal{M}^{WF}_{x,y}\psi(y), \tag{1.48}$$

with

$$\mathcal{M}^{WF}_{x,y} = \delta_{x,y} - \kappa \not{D}_{x,y}. \tag{1.49}$$

We make the choice $r = 1$ which means that $\not{D}$ becomes,[6]

$$\not{D}_{x,y} = \sum_{\mu} \left(U_\mu(x) P^-_\mu \delta_{x+\hat{\mu},y} + U^\dagger_\mu(x-\hat{\mu}) P^+_\mu \delta_{x-\hat{\mu},y} \right), \tag{1.50}$$

where the *projection operators*,

$$P^\pm_\mu = (1 \pm \gamma_\mu), \tag{1.51}$$

project out two spinor components of the four-spinor ψ.

One of the fundamental quantities studied in lattice QCD is the quark propagator, $\mathcal{M}^{-1}$. For Wilson fermions, we calculate components, ψ, of the quark propagator by solving equations of the form

$$\left[\mathcal{M}^{WF}_{x,y}\right]^{ij}_{\alpha\beta} \psi^j_\beta(y) = \eta^i_\alpha(x) \tag{1.52}$$

where i, j are colour indices, and α, β are spin indices. In this work, the source, η, is spatially localised, i.e., it is a point-like source.

Wilson fermions do not have the problem of fermion doubling which makes them much easier to relate to physical particles. The cost of avoiding fermion doubling is that the fermion action now differs from the continuum action by a term of $rO(a)$.

[6] This definition of $\not{D}$ is different from the continuum one and is the one we use throughout the rest of this thesis.

1.4.4 Improved Fermionic Actions

When discretising the continuum action so that it can be simulated on a lattice, we want the effects of the discrete lattice spacing, a, to be as small as possible. When we avoid fermion doubling by using the Wilson action, the major discretisation error comes from the fermions since the gauge action differs from the continuum only at $O(a^2)$. Therefore, we concentrate our efforts towards improved actions on the fermionic sector.

Two-Link Improved Action

Various people, (including [Hamber 83]), have suggested adding a term to the fermion action which cancels the $O(a)$ term in the Wilson action. Let the fermion action be

$$S_{2L} = S_{WF} + \Delta S_{2L}$$

where the extra term is

$$\begin{aligned} \Delta S_{2L} &= \frac{r}{8}\sum_{x,\mu}\Big\{\bar{\psi}(x)U_\mu(x)U_\mu(x+\hat{\mu})\psi(x+2\hat{\mu}) \\ &\qquad + \bar{\psi}(x+2\hat{\mu})U_\mu^\dagger(x+\hat{\mu})U_\mu^\dagger(x)\psi(x) \\ &\qquad - 2\bar{\psi}(x)\psi(x)\Big\}. \end{aligned} \tag{1.53}$$

This is not a nearest-neighbour action as it involves sites separated by a double link. We call this the *Two-Link Improved* action. Since the gauge action in Eq. (1.39) only differs from the continuum by terms of $O(a^2)$, we now have a lattice action for full QCD which differs from the continuum action (at tree level) by terms of $O(a^2)$.

For us, the major problem with this action is that it is difficult to simulate on a parallel computer. Since it involves sites separated by 2 links, the amount of communication between processors will be larger (see § (3.2) and § (3.4)), which will greatly decrease the efficiency of the simulation. Additionally, the *red-black* preconditioning of § (2.3.1) no longer decouples sites of different parity.

Clover Action

[Sheikholeslami 85] suggested making a change of variables in the path integral for the Two-Link action to return to a nearest neighbour action. Rotating the quark fields gives

$$\begin{aligned} \psi &\rightarrow \psi' = \left(1 - \frac{r}{4}(\gamma \cdot \overrightarrow{D} - m_0)\right)\psi \\ \bar{\psi} &\rightarrow \bar{\psi}' = \bar{\psi}\left(1 + \frac{r}{4}(\gamma \cdot \overleftarrow{D} + m_0)\right). \end{aligned} \tag{1.54}$$

where $\overrightarrow{D}$ and $\overleftarrow{D}$ are the lattice covariant central difference operators acting on the right and left respectively. [Sheikholeslami 85] claims this is an *isospectral* transformation which does not change the mass spectrum. The action now has the form

$$S_C = S_{WF} + \Delta S_C$$

where

$$\Delta S_C = -ig_0\frac{r}{4}\sum_{x,\nu,\mu}\bar{\psi}(x)F_{\nu\mu}\sigma_{\nu\mu}\psi(x) \tag{1.55}$$

and

$$\sigma_{\nu\mu} = \frac{1}{2}\left[\gamma_\nu, \gamma_\mu\right].$$

$F_{\nu\mu}$ is a lattice definition of the field strength tensor which we normally take from

$$g_0F_{\nu\mu}(x) = \frac{1}{4}\sum_j \frac{1}{2i}\left[U_j^\Box - U_j^{\Box\dagger}\right] \tag{1.56}$$

where the sum is over the 4 plaquettes in the $\nu\mu$-plane around site x, as shown in Fig. 1.2. We call this action the *Sheikholeslami–Wohlert* action or *Clover* action, from the representation of $F_{\nu\mu}$.

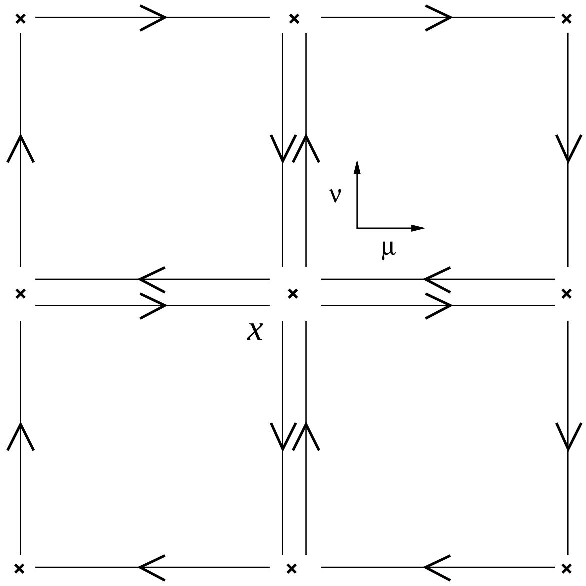


Figure 1.2: Lattice representation of $F_{\nu\mu}$.

Writing the action in the form

$$S_C = \left(\frac{1}{2\kappa}\right) \bar{\psi}(x)\mathcal{M}^C_{x,y}\psi(y),$$

gives the Clover fermion matrix

$$\mathcal{M}^C_{x,y} = A\delta_{x,y} - \kappa \not{D}_{x,y}, \tag{1.57}$$

where the local term, A, is

$$A = 1 - \frac{i\kappa}{2} g_0 F_{\nu\mu}\sigma_{\nu\mu}. \tag{1.58}$$

Since this action only connects nearest-neighbour sites, it is almost as straightforward to simulate as the Wilson action. The extra term, $F_{\nu\mu}\sigma_{\nu\mu}$, is purely local and can, therefore, be simulated efficiently on a parallel computer.

To investigate matrix elements between physical states, we use the on-shell condition, $\gamma\cdot \overrightarrow{D}= -m_0$, in Eq. (1.54) to give

$$\begin{aligned} \psi &\rightarrow \psi' = \left(1 - \frac{r}{2}\gamma\cdot \overrightarrow{D}\right)\psi \\ \bar{\psi} &\rightarrow \bar{\psi}' = \bar{\psi}\left(1 + \frac{r}{2}\gamma\cdot \overleftarrow{D}\right). \end{aligned} \tag{1.59}$$

If we let $G(x,y)$ be the inverse of $\mathcal{M}^C_{x,y}$, then the improved propagator is

$$G^I(x,y) = \left(1 - \frac{r}{2}\gamma\cdot \overrightarrow{D}\right) G(x,y) \left(1 + \frac{r}{2}\gamma\cdot \overleftarrow{D}\right). \tag{1.60}$$

Since G^I contains the rotations, the procedure to evaluate all the correlation functions is now the same as for the Wilson action.

To calculate the improved quark propagator from a source η, we first calculate an intermediate propagator which is the solution of

$$\mathcal{M}^C_{x,y}\phi(y) = \eta(x)\left(1 + \frac{r}{2}\gamma\cdot \overleftarrow{D}\right) \tag{1.61}$$

with η rotated on the right. This can be solved in the same way as Eq. (1.52). We can then obtain the required components of the improved quark propagator as

$$\psi(x) = \left(1 - \frac{r}{2}\gamma\cdot \overrightarrow{D}\right)\phi(x). \tag{1.62}$$

Further information on the Clover action can be found in [Heatlie 91] and references therein.

1.4.5 Configuration Generation

The partition functions for the various lattice QCD actions can all be written in the form

$$Z_{\mathrm{QCD}} = \int \mathcal{D}\bar{\psi}\, \mathcal{D}\psi\, \mathcal{D}U\ e^{-\bar{\psi}\mathcal{M}\psi + \frac{\beta}{3}\sum_{\Box}\mathrm{Re\,Tr}[U^{\Box}]}. \tag{1.63}$$

The fermions are Grassmann variables and so are very complicated to simulate directly on a computer. We perform the integration over the fermion fields analytically to give

$$Z_{\mathrm{QCD}} = \int \mathcal{D}U\ \det\mathcal{M}(U)\, e^{\frac{\beta}{3}\sum_{\Box}\mathrm{Re\,Tr}[U^{\Box}]}. \tag{1.64}$$

The problem with this form of the partition function is that $\det\mathcal{M}$ is highly non-local.

The Quenched Approximation

One way to overcome this problem is simply to set $\det \mathcal{M}(U) = 1$. This is normally called the *Quenched* approximation, and is an expedient with little physical justification. The action for quenched lattice QCD is

$$Z_{\mathrm{Quenched}} = \int \mathcal{D}U \; e^{\frac{\beta}{3}\sum_{\Box} \mathrm{Re}\,\mathrm{Tr}[U^{\Box}]}. \tag{1.65}$$

Generating gauge configurations using this action ignores effects from fermion vacuum polarisation and so has a similar effect to the fermion mass becoming infinite.

The experimental justification that the quenched action might still give reasonably physical results comes from the *Zweig* rule as shown in Fig. 1.3. The decay $\phi \to \bar{K}K$ is preferred relative to $\phi \to 3\pi$, even though the phase space for the former is small. [Zweig 64] suggested that this was due to the fact that strong decays involving $q\bar{q}$ annihilation are suppressed.

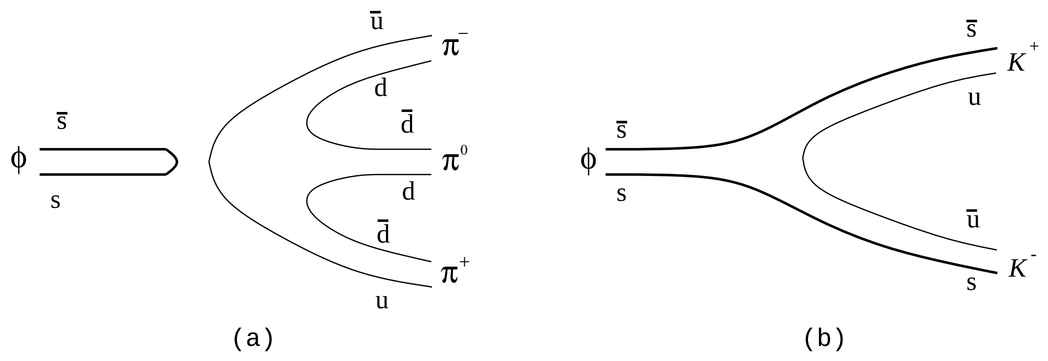


Figure 1.3: Quark diagram for ϕ decays. Decay (a) is disallowed by the Zweig rule since it involves $s\bar{s}$ annihilation.

The quenched approximation is used in all the simulations described in this thesis. We generate gauge configurations independently of the fermion action and so the fermion action only controls how the quark propagates in the gauge fields.

Pure Gauge Algorithms

We therefore need to generate configurations of gauge fields using the partition function of Eq. (1.65). We can use the same algorithms as described for the simulation of spin systems in § (1.1) and § (1.2). For a given link, $U_\mu(x)$, the terms in the action which involve it come from the 6 plaquettes of which it is a member, i.e., the action as a function of the link is

$$S(U_\mu(x)) = -\frac{\beta}{3}\left(\mathrm{Re}\,\mathrm{Tr}\,[U_\mu(x)R(x,\mu)]\right) + E_0 \tag{1.66}$$

where E_0 does not depend on $U_\mu(x)$. The *staple*, $R(x,\mu)$, is the sum over the 3-link paths from $x+\hat{\mu}$ to x,

$$R(x,\mu) = \sum_{\nu\neq\mu}\left[U_\nu(x+\hat{\mu})U_\mu^\dagger(x+\hat{\nu})U_\nu^\dagger(x) + U_\nu^\dagger(x-\hat{\nu}+\hat{\mu})U_\mu^\dagger(x-\hat{\nu})U_\nu(x-\hat{\nu})\right]. \tag{1.67}$$

Since the change in the action is purely local, we can adapt the heatbath and over-relaxation algorithms to this problem. For the heatbath, we should replace $U_\mu(x)$ with $U'_\mu(x)$ generated with probability distribution

$$P(U'_\mu(x)) \propto e^{\frac{\beta}{3}\mathrm{Re}\,\mathrm{Tr}\left[U'_\mu(x)R(x,\mu)\right]}.$$

For $SU(3)$ it is complicated to generate a matrix from this distribution. However, [Creutz 80] developed an efficient algorithm for $SU(2)$ gauge theory which was later improved by [Kennedy 85]. We follow the suggestion by [Cabibbo 82] to use this on a set of $SU(2)$ subgroups to update the $SU(3)$ matrices. We take $SU(2)$ subgroups of the form

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & h_{11} & h_{12} \\ 0 & h_{21} & h_{22} \end{pmatrix}, \quad \begin{pmatrix} h_{11} & h_{12} & 0 \\ h_{21} & h_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} h_{11} & 0 & h_{12} \\ 0 & 1 & 0 \\ h_{21} & 0 & h_{22} \end{pmatrix}. \tag{1.68}$$

An update on a link, consists of successively performing $SU(2)$ heatbaths with each of these subgroups.

The update is of the form $U \to HU$ where H has an $SU(2)$ matrix, h, embedded in it as above. We want the probability distribution to be

$$Q(H) \propto e^{\frac{\beta}{3}\mathrm{Re}\,\mathrm{Tr}[HUR]}. \tag{1.69}$$

If x is the 2×2 submatrix of UR corresponding to h, then

$$\mathrm{Tr}\,[HUR] = \mathrm{Tr}\,[hx] + \mathrm{const.} \tag{1.70}$$

Noting that any 2×2 complex matrix can be uniquely written in the form

$$x = \xi_1 u_1 + i\xi_2 u_2$$

with $\xi_i \in \mathcal{R}$ and $u_i \in SU(2)$, we can show that the probability distribution in Eq. (1.69) becomes

$$Q(H) \propto e^{\frac{\xi_1\beta}{3}\mathrm{Re}\,\mathrm{Tr}[hu_1]}. \tag{1.71}$$

Since $hu_1 \in SU(2)$, we generate an $SU(2)$ matrix, $v = hu_1$, distributed as in Eq. (1.71) and let

$$h = vu_1^\dagger.$$

If $U'_\mu(x)$ is the matrix produced after updating with each of the subgroups in Eq. (1.68) then it will have the correct Boltzmann distribution providing $U_\mu(x)$ had. This, along with the fact that the choice of $SU(2)$ subgroups allows ergodicity, is enough to guarantee that this is a correct MC algorithm.

For an over-relaxation algorithm, we want to make a large change in the link while keeping the change in energy small. For a local action of the form Eq. (1.66), [Adler 81] suggests the new link

$$U'_\mu(x) = g_0 U^\dagger_\mu(x) g_0, \tag{1.72}$$

where g_0 is an element of $SU(3)$ which approximately minimises the local action. We might think of taking $R^\dagger$ where R is the staple from Eq. (1.67). However, as this is not a group element, we have to project onto $SU(3)$ before inverting. The action is not exactly conserved and so we need to augment the over-relaxation with a Metropolis accept/reject step.

However, we can perform the over-relaxations on the $SU(2)$ subgroups defined in Eq. (1.68). With the same notation as for the $SU(2)$ heatbaths, we set

$$h = u_1^\dagger u_1^\dagger \tag{1.73}$$

which does conserve the action. Since this algorithm is microcanonical, we follow each n over-relaxation sweeps with a heatbath sweep to ensure ergodicity. Investigations by [Michael 91] suggest that a value of $n \approx 4$ keeps the autocorrelation time small.

Chapter 2

Inverting the Fermion Matrix

The most time consuming part of any QCD simulation involves inverting the appropriate fermion matrix. We need to calculate fermion vectors, ψ, which are the solutions to equations of the form

$$\mathcal{M}\psi = \eta, \tag{2.1}$$

where $\mathcal{M}$ is the fermion matrix. For *quenched* simulations, this is required for calculating the quark propagators by solving Eq. (2.1) with a point-like source, η, (see Eq. (1.52)). In *dynamical* simulations, we need to solve a similar equation for every iteration of the Hybrid Monte Carlo algorithm as described by [Duane 87].

The main reason for the computational difficulty of this inversion is the size of the fermion matrix. For Wilson fermions on a $24^3 \times 48$ lattice,

$$\begin{aligned} \text{Order}(\mathcal{M}) &= (24^3 \times 48) \times (3 \text{ colours}) \times (4 \text{ spins}) \\ &= 7,962,624. \end{aligned}$$

This problem would be completely intractable if it were not for the fact that the matrix is so sparse. For nearest neighbour actions, each row contains just over

100 non-zero elements. However, it is still prohibitive to perform the inversion exactly (which is an [Order$((\mathcal{M})]^3$ operation), and we must use an approximate, iterative method.

§ (2.1) and § (2.2) will outline a number of different algorithms which can be used for matrix inversion, with particular emphasis on Minimal Residual (MR) and Conjugate Gradient (CG) methods. The next section will describe how to implement a simple form of preconditioning, called Red-Black preconditioning, for the Wilson and Clover actions. § (2.4) and § (2.5) will describe investigations into the performance of CG and MR, respectively. The final section compares their performance and explains the optimal choice of algorithm for the physical region we study.

2.1 Iterative Methods for Matrix Inversion

Iterative methods to solve large sets of coupled linear equations work by starting off with a guess at the solution and successively improving it. This involves moving downhill in some very large dimensional error surface. In essence, this is an example of a minimisation problem where the quantity to be minimised is the error in the solution. There are a number of ways of doing this depending on the particular problem involved. A number of common methods are described and compared below. As we use up most of our computer time inverting the fermion matrix, it is important that we make an informed choice of algorithm.

The available algorithms fall into two distinct classes.

1. General Relaxation Methods.

2. Successive Line Minimisation Methods.

Most of the methods which are used for fermion matrix inversions fall into the second of these, where we successively minimise the error in the solution along appropriate directions. The main difference between these is in the choice of directions. Before comparing various line minimisation methods, for completeness we describe two general algorithms and their suitability for this problem.

2.1.1 Simulated Annealing

This method is similar to the Metropolis algorithm, in that random changes are proposed to the solution vector and are accepted or rejected depending on the change in some error function, e.g.,

$$\mathcal{E} = |\mathcal{M}\psi - \eta|, \tag{2.2}$$

or

$$\mathcal{E} = |\mathcal{M}\psi - \eta|^2. \tag{2.3}$$

The acceptance probability is

$$P_{\mathrm{acc}} = \begin{cases} 1 & \text{if } \delta\mathcal{E} \leq 0 \\ \exp(-\beta\,\delta\mathcal{E}) & \text{if } \delta\mathcal{E} > 0 \end{cases} \tag{2.4}$$

After each sweep through the lattice, the temperature is slowly decreased i.e., $\beta \to \beta + \delta\beta$, so that uphill steps become increasingly unlikely. This has the effect of gradually cooling the system so that it is likely to end up in its ground state. In the limit where $\delta\beta \to 0$, this gives the exact solution. This is one of the main advantages of this method, along with the fact that it copes well with very complicated energy surfaces (i.e., not smooth surfaces) since it only ever requires the value of $\mathcal{E}$, not its gradient. However, even for non-pathological surfaces, this method can be very slow indeed. Also, for $\delta\beta$ small but finite, the solution can get trapped in local minima and give the wrong answer. Therefore, in practice, simulated annealing is only useful for difficult problems where an approximate solution is adequate.

2.1.2 Gauss-Seidel

With this algorithm, it is necessary to label the lattice sites in some order, such as that given in § (3.2.1). We can then sweep through the lattice using the relaxation scheme described in [Weingarten 81], i.e.,

$$\psi_{i+1}(x) = \psi_i(x) + \frac{\omega}{\mathcal{M}_{x,x}} \left\{ \eta(x) - \sum_{y \geq x} \mathcal{M}_{x,y}\psi_i(y) - \sum_{y > x} \mathcal{M}_{x,y}\psi_{i+1}(y) \right\}, \qquad (2.5)$$

where x and y are site labels, ω is an over (under) relaxation parameter and i is the iteration label. The major difficulty with this algorithm is that there is no guarantee that a sweep of the lattice will improve the estimate of the solution. As shown for quenched lattice QCD in [Rossi 87], Gauss-Seidel converges quickly to the solution for $(\kappa_c - \kappa) > 0.05$, where κ_c is defined by the occurrence of zero eigenvalues in Eq. (2.1) or, physically, by the vanishing of the quark mass. However, as $\kappa \to \kappa_c$, GS becomes unstable and there are no values of ω which allow it to converge. As we are most interested in investigating the critical region on large lattices, GS is not particularly suitable.

2.2 Successive Line Minimisation Methods

The rest of the methods below attempt to find the minimum of an error function by performing line minimisations along various directions. Since the error functions are usually quadratic forms, this is particularly easy. The main difference between the methods is how the successive minimisation directions are chosen. We want to choose new directions such that they result in a significantly reduced error yet do not require a large amount of work to calculate.[1]

2.2.1 Steepest Descent

Steepest Descent is one of the most obvious ways of attempting to solve any minimisation problem, such as minimising the error in the solution of a set of coupled equations. It involves always moving towards the local minimum in the error surface along the steepest slope, i.e., the minimisation direction is always along the local gradient. As well as being able to calculate the value of the error function at each point in the multi-dimensional space, we must also be able to calculate $\nabla\mathcal{E}$ everywhere. Like most deterministic algorithms, SD often gets trapped in local minima from which it may take many iterations to escape. This problem can be lessened by using a variable step-size and allowing backward steps once the algorithm is trapped. By necessity, this makes the algorithm more complicated, and there is still no guarantee of convergence in a finite number of iterations.

The biggest problem with SD is that, by necessity, each step must be perpendicular to the one before. Hence, if the error surface has the form of a long, narrow valley (see Fig. 2.1), SD is forced to take a large number of small steps near the valley bottom, rather than the more direct route along it.

[1]These methods are somewhat analogous to a hill-walker searching for the deepest valley in some region. The differences in the algorithms are due to the different capabilities of hill-walkers and computers (and, of course, to the different dimensionality of the space they inhabit). Hill-walkers find it easy to see where they are and how the hill slopes but move relatively slowly — they would therefore frequently update the direction they are moving in. When solving similar problems with a computer, function evaluation (i.e., discovering where you are) is expensive while moving is cheap. Therefore, we should use the computer to take larger, discrete steps towards the local minimum using all the information at its disposal.

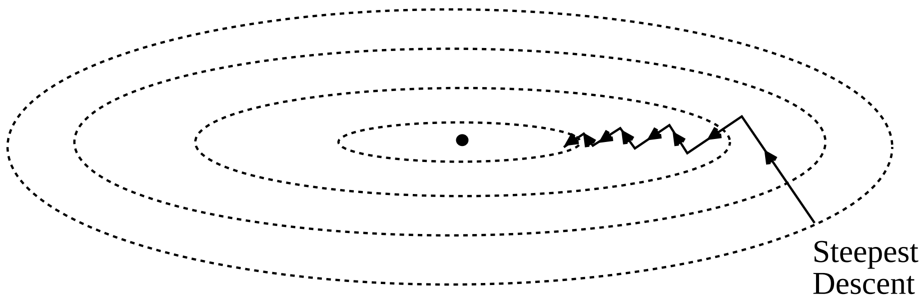


Figure 2.1: Problem with Steepest Descent.

2.2.2 Conjugate Residual

This is a class of methods described by [Eisenstat 83] and [Oyanagi 86] which attempt to avoid the problem of SD shown in Fig. 2.1 by minimising along conjugate directions. The vectors p and q are conjugate with respect to $\mathcal{M}^\dagger\mathcal{M}$ if they satisfy

$$(\mathcal{M}p, \mathcal{M}q) = 0 \tag{2.6}$$

where (a, b) is $\sum_l a_l^\dagger b_l$. The form of CR(k) can be seen in Table 2.1.

$$\begin{aligned} r_0 &= \eta - \mathcal{M}\psi_0 \\ p_0 &= r_0 \end{aligned}$$

repeat until convergence

$$\begin{aligned} \alpha &= \frac{(\mathcal{M}p_i, r_i)}{(\mathcal{M}p_i, \mathcal{M}p_i)} \\ \psi_{i+1} &= \psi_i + \alpha p_i \\ r_{i+1} &= r_i - \alpha\mathcal{M}p_i \\ p_{i+1} &= r_{i+1} - \sum_{j=i-k+1}^{j=i} \frac{(\mathcal{M}p_j, \mathcal{M}r_{i+1})}{(\mathcal{M}p_j, \mathcal{M}p_j)} p_j \end{aligned}$$

Table 2.1: The Conjugate Residual algorithm — CR(k).

At each iteration, the solution vector, ψ, is updated along p_i so as to minimise the norm of the residual, $r_{i+1} = \mathcal{M}\psi_{1+1} - \eta$. The value of α in Table 2.1 minimises

$$(r_{i+1}, r_{i+1}) = |\mathcal{M}(\psi_i + \alpha p_i) - \eta|^2 \tag{2.7}$$

as a function of α. Table 2.1 also shows how the new minimisation direction is explicitly constructed to be conjugate to the previous k directions. If k is sufficiently large that the full set of directions, $\{p\}$, are conjugate, then the method is guaranteed to converge in a finite number of iterations. However, for large systems, the storage and computation required are enormous. Since the memory required is proportional to k, we are usually restricted to small values of k. Indeed, since the lattice size used in many QCD simulations is constrained by the amount of computer memory available, CR(k) is often not a feasible algorithm.

It is important to note that this algorithm and the others in this section update the residual, r, rather than re-calculate it at each iteration. Since we are working with finite precision, we cannot achieve an exact solution and so we continue the algorithm until some carefully chosen convergence criterion has been satisfied. Usually this is when $|r|^2 < R^2$ for some appropriate value of R. Once this condition is satisfied, we restart the algorithms, entailing the direct evaluation of r, to ensure that rounding errors have not prevented the residual being accurately updated. This self-correction ensures that we end up with a solution of the desired accuracy.

2.2.3 Over-relaxed Minimal Residual

Minimal Residual (MR) is the limit of CR(k) when $k = 0$ and so the storage required is small. In this limit, the new minimisation direction reduces to the residual, r, and so MR is essentially a steepest descent method. However, we also allow the possibility of optimising the convergence using an over-relaxation parameter, ω, as shown in Table 2.2.

The basic premise of over-relaxation is that if the solution is moving in a good

$$
\begin{aligned}
r_0 &= \eta - \mathcal{M}\psi_0 \\
&\text{repeat until convergence} \\
\alpha &= \frac{(\mathcal{M}r_i, r_i)}{(\mathcal{M}r_i, \mathcal{M}r_i)} \\
\alpha' &= \omega\alpha \qquad\qquad \text{(over-relaxation)} \\
\psi_{i+1} &= \psi_i + \alpha' r_i \\
r_{i+1} &= r_i - \alpha' \mathcal{M} r_i
\end{aligned}
$$

Table 2.2: The Over-relaxed Minimal Residual algorithm — ORMR.

direction, then moving further in that same direction will still be good. For some problems, this premise will be completely wrong. In certain cases, the slope on the other side of the valley will be very steep and so continuing even a little past the minimum could greatly increase the residue. Nevertheless, it is possible that over-relaxation, by allowing longer steps, can help to lessen the problem of SD-type methods. This improvement is shown schematically in Fig. 2.2.

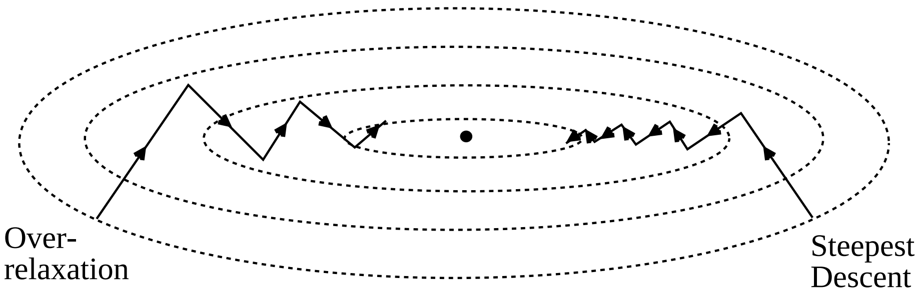


Figure 2.2: Over-relaxation *v.* Steepest Descent.

2.2.4 Conjugate Gradient

The various Conjugate Gradient (CG) methods, first described in [Hestenes 52], are based on a clever prescription for implicitly ensuring that all the minimisation directions remain conjugate. [Rossi 87] contains a useful summary of how this is performed.

When the directions are all conjugate, the main problem of SD is avoided as shown schematically in Fig. 2.3. It also provides the guarantee that, in a space of dimensionality N, CG will converge to the solution in $O(N)$ steps with exact arithmetic. The convergence is monotonic in the error function appropriate to the algorithm.

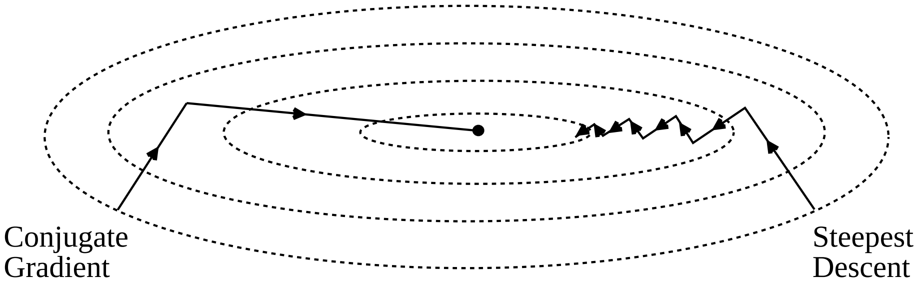


Figure 2.3: Conjugate Gradient *v.* Steepest Descent.

The main problem with the CG methods is that they require the coefficient matrix to be Hermitian, positive definite. The fermion matrices we need to invert do not satisfy these conditions and so we must use versions of CG which effectively invert $\mathcal{M}^\dagger\mathcal{M}$, or $\mathcal{M}\mathcal{M}^\dagger$. Since the convergence properties of inversion algorithms often depend on the relative magnitudes of the eigenvalues of the matrix, i.e., the conditioning of the matrix, $\mathcal{M}^\dagger\mathcal{M}$ and $\mathcal{M}\mathcal{M}^\dagger$ will generally be more difficult to invert than $\mathcal{M}$.

We have looked at two variants on the CG algorithm. The major difference between them is in the definition of the error function. Least Residual (LR) minimises the difference between $\mathcal{M}\psi$ and η i.e.,

$$\mathcal{E} = |\mathcal{M}\psi - \eta|^2, \tag{2.8}$$

while Least Norm (LN) minimises the difference between ψ and ψ_{exact}, the exact solution of Eq. (2.1). For LN,

$$\mathcal{E} = |\psi - \psi_{\text{exact}}|^2. \tag{2.9}$$

Our implementation of the algorithms is based on that given in [Oyanagi 86].

2.2.5 Least Residual CG

With this algorithm, the equation to be solved is

$$\mathcal{M}^\dagger \mathcal{M}\psi = \mathcal{M}^\dagger \eta. \tag{2.10}$$

Clearly, any solution of Eq. (2.10) is also a solution of the equation we are interested in, Eq. (2.1). The steps required for LR are shown in Table 2.3.

$$\begin{aligned}
r_0 &= \eta - \mathcal{M}\psi_0 \\
p_0 &= 0 \\
&\text{repeat until convergence} \\
p_{i+1} &= p_i + \frac{\mathcal{M}^\dagger r_i}{(M^\dagger r_i, M^\dagger r_i)} \\
\alpha &= (\mathcal{M}p_{i+1}, \mathcal{M}p_{i+1})^{-1} \\
\psi_{i+1} &= \psi_i + \alpha p_{i+1} \\
r_{i+1} &= r_i - \alpha \mathcal{M} p_{i+1}
\end{aligned}$$

Table 2.3: The Least Residual algorithm — CGLR.

Each iteration minimises the residual along direction p_{i+1}, i.e., the value of α is determined by the requirement that

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}(r_{i+1}, r_{i+1}) = 0.$$

Note that to ensure that all the direction vectors p_i are orthogonal, each iteration involves a multiplication by $\mathcal{M}$ and $\mathcal{M}^\dagger$.

2.2.6 Least Norm CG

Contrary to the above, LN solves the equation

$$\mathcal{M}\mathcal{M}^\dagger \rho = \eta \tag{2.11}$$

If ρ_{exact} is the solution to Eq. (2.11), then the solution to the required equation, Eq. (2.1), is $\psi_{\text{exact}} = \mathcal{M}^\dagger \rho_{\text{exact}}$. The form of the algorithm is shown in Table 2.4.

$$\begin{aligned} r_0 &= \eta - \mathcal{M}\psi_0 \\ p_0 &= 0 \end{aligned}$$

repeat until convergence

$$\begin{aligned} p_{i+1} &= p_i + \frac{\mathcal{M}^\dagger r_i}{(r_i, r_i)} \\ \alpha &= (p_{i+1}, p_{i+1})^{-1} \\ \psi_{i+1} &= \psi_i + \alpha p_{i+1} \\ r_{i+1} &= r_i - \alpha \mathcal{M} p_{i+1} \end{aligned}$$

Table 2.4: The Least Norm algorithm — CGLN.

The value of α in Table 2.4 is chosen to minimise

$$(\rho_{i+1} - \rho_{\text{exact}}, \mathcal{M}\mathcal{M}^\dagger[\rho_{i+1} - \rho_{\text{exact}}]) = (\psi_{i+1} - \psi_{\text{exact}}, \psi_{i+1} - \psi_{\text{exact}}),$$

i.e., each iteration of this algorithm reduces the difference between the current guess for the solution and the (unknown) exact solution. It is interesting to note that an iteration can be written without any explicit mention of ρ. Since $|\psi - \psi_{\text{exact}}|$ is an unmeasurable quantity, the only way to judge the accuracy of the solution is to look at the residual, as usual. Although LN does converge monotonically to the exact solution in terms of $|\psi - \psi_{\text{exact}}|$, it is not guaranteed to be monotonic in the quantities readily available as convergence criteria. This is one of the problems investigated in § (2.4.2).

As can be seen by comparing Table 2.4 to Table 2.3, the operations required for the LN algorithm are very similar to those required in LR. § (2.4), describes various tests comparing the performance of the two CG algorithms.

2.2.7 Summary

As we have seen, a large class of algorithms can be built from the following building blocks considered from the viewpoint of a parallel implementation.

Matrix $\times$ Vector. This is the only way in which the fermion matrix, $\mathcal{M}$, enters into the algorithms. This means that we do not need to store the matrix but only require the ability to multiply a given vector by it. Usually, multiplication by $\mathcal{M}^\dagger$ is a trivial modification. In our simulations, the matrix $\mathcal{M}$ always connects only nearest neighbour sites e.g., see Eq. (1.46). Therefore, this operation requires communication between nearest neighbour processors. Details of our communication strategy are given in § (3.4).

Update Vector. This is a completely local operation with each processor responsible for updating components of the vector corresponding to its own section of the lattice. These operations come in a number of related forms,

saxpy $y = y + ax$,

sxpay $y = x + ay$ and,

caxpy $y = x + \alpha y$,

where x, y are complex vectors, $a \in \mathcal{R}$ and $\alpha \in \mathcal{C}$.

Scalar product. Calculating the norm of a vector or the complex scalar product of two vectors require local operations followed by a global summation over the processor lattice. To be able to check reproducibility, it is essential that the global additions are performed in a particular order. We do this by summing over each of the 4 directions in turn.

Since we require a very high performance from the code, these routines must be written in assembly language. Details of these assembler routines may be found in [Booth 91]. Once they have been written, it is easy to switch between different matrix inversion algorithms.

All the algorithms are deterministic, so, in addition to the choice of algorithm to use, we have only a limited number of other choices to make.

Initial Guess. The choice of ψ_0 must not affect the final solution (or else we have a serious problem!), but may change the number of iterations taken to converge. Any effect is likely to be small, so we want to choose something which can be generated cheaply. Obvious choices include setting ψ_0 to be zero everywhere, or setting it equal to the source vector η. For a local source, there is very little difference between these alternatives. Since κ is small, the fermion matrix, $\mathcal{M}$, is almost diagonal and so we normally set $\psi_0 = \eta$ as the initial guess. However, we are often calculating the propagators on the same configuration for a number of closely spaced values of κ. Since the matrices to be inverted are very similar, it may prove advantageous to re-use the solution to the previous κ value as the initial guess to the next. Therefore, we need to compare the effects of taking ψ_0 to be the source, η, or the solution corresponding to a nearby κ.

Convergence Criteria. This can be quite a difficult choice depending fundamentally on the use to which the solution is to be put. In practice, the only information we have which can be used to make the decision to stop the inversion algorithm are the residual vector, r, and the direction vector p. Of these, r is the obvious choice, since it is a direct measure of the goodness of the solution. Therefore, we will always stop the inversion when $|r|^2 < R^2$ and we are left with choosing an appropriate value of R. This should be determined by the physics which we are investigating.

For an upper bound on R, we require that the pion propagator (see § (4.1.1)) should no longer change as we decrease R. This is an easy test but does not guarantee that R is small enough to ensure that the other hadron propagators are similarly unaffected. The form of these propagators make them difficult to compute (see e.g., § (4.1.2)), and so this is a test that can only be carried out retrospectively. The only lower bounds on R are empirical ones. We want to ensure that we obtain the maximal amount of physics from a given amount of computer time. Therefore, we do not want to use a tighter convergence criterion than is necessary. Also, if R is set to be very small, the errors due to the finite precision arithmetic become large. This means that, since we update r rather than re-calculating it at each iteration, the true residual may be much larger than the updated one. This gives a hardware limitation on the accuracy we can achieve. For this project, we work in 32 bit precision and set $R \sim 10^{-7} \rightarrow 10^{-8}$ which seems to satisfy all the above requirements.

2.3 Preconditioning

Since so much of the computer time is spent in matrix inversion, we should not only investigate different inversion algorithms but also other ways to make the inversions faster. Preconditioning, in general, attempts to make the matrix easier to invert. However, many methods, such as the ILU decomposition described in [Oyanagi 86], are difficult to implement and very inefficient on present-day parallel computers since the preconditioning cannot be carried out independently on different sites. § (2.3.1) describes a simple preconditioning, applicable to nearest-neighbour actions, which decouples sites of opposite parity.

2.3.1 Red-Black Preconditioning

To calculate the components of the quark propagator for Wilson fermions, we are looking to solve equations of the form (Eq. (1.52))

$$(1 - \kappa \not{D})\psi = \eta, \tag{2.12}$$

where $\not{D}$, defined by Eq. (1.50), connects only nearest-neighbour sites, and η is a point-like source of a particular colour and spin.

We are interested in the region where $\kappa \sim 0.15$ and so the off-diagonal elements of the fermion matrix are small. If we multiply both sides of Eq. (2.12) by $(1 + \kappa \not{D})$, we get

$$(1 - \kappa^2 \not{D}^2)\psi = (1 + \kappa \not{D})\eta, \tag{2.13}$$

where the off-diagonal elements of the coefficient matrix are of $O(\kappa^2)$. Since the coefficient matrix is thus more diagonally dominant, the solution should converge to the required precision in fewer steps. This can be regarded as a first order preconditioning of the fermion matrix (see [Gupta 89]).

Due to the nearest-neighbour form of $\not{D}$, there is a natural *checkerboard* decomposition of the lattice. The parity, p, of a site x is defined to be

$$p(x) = (x_0 + x_1 + x_2 + x_3) \bmod 2. \tag{2.14}$$

With this *even/odd* or *red-black* ordering, the Wilson fermion matrix has the form

$$\mathcal{M}^{WF} = \begin{bmatrix} 1 & -\kappa \not{D}_{eo} \\ -\kappa \not{D}_{oe} & 1 \end{bmatrix} \tag{2.15}$$

and the fermion vectors are written as, e.g.,

$$\psi = \begin{bmatrix} \psi_e \\ \psi_o \end{bmatrix}.$$

Noting that $\not{D}^2$ couples even sites to even sites and odd sites to odd sites, we see that Eq. (2.13) decouples into even and odd parts, giving

$$(1 - \kappa^2 \not{D}_{eo} \not{D}_{oe})\psi_e = \eta_e + \kappa \not{D}_{eo}\eta_o. \tag{2.16}$$

The odd part of the solution is trivially related to the even part. The odd part of Eq. (2.12) is $\psi_o - \kappa \not{D}_{oe}\psi_e = \eta_o$, which gives

$$\psi_o = \eta_o + \kappa \not{D}_{oe}\psi_e. \tag{2.17}$$

To see how the residues from the red-black algorithm compare with those from a standard inversion, we define

$$\text{res(n.m)} = \text{(LHS of Eq. (n.m)) - (RHS of Eq. (n.m))}.$$

This gives

$$\begin{aligned} \text{res}(2.12) &= \psi - \kappa \not{D}\psi - \eta \\ &= \begin{bmatrix} \psi_e - \kappa \not{D}_{eo}\psi_o - \eta_e \\ \psi_o - \kappa \not{D}_{oe}\psi_e - \eta_o \end{bmatrix} \end{aligned}$$

in the red-black notation. Eliminating ψ_o using Eq. (2.17), we get

$$\begin{aligned}
\mathrm{res}(2.12) &= \begin{bmatrix} \psi_e - \kappa \not{D}_{eo}(\eta_o + \kappa \not{D}_{oe}\psi_e) - \eta_e \\ (\eta_o + \kappa \not{D}_{oe}\psi_e) - \kappa \not{D}_{oe}\psi_e - \eta_o \end{bmatrix} \\
&= \begin{bmatrix} \psi_e - \kappa^2 \not{D}_{eo} \not{D}_{oe}\psi_e - (\eta_e + \kappa \not{D}_{eo}\eta_o) \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} \mathrm{res}(2.16) \\ 0 \end{bmatrix}
\end{aligned}$$

by definition. What this means is that, if we take an even solution of Eq. (2.16) with residue r and reconstruct the odd part via Eq. (2.17), we have a solution, ψ, of Eq. (2.12) with the same residue r. Therefore, a solution obtained to any particular residue by the red-black algorithm is as good a solution to the original equation, Eq. (2.12), as a solution to the standard inversion with the same residue.

Each iteration of the red-black inversion algorithm requires twice as many calls to $\not{D}$ as the standard inversion, i.e., without red-black preconditioning. As will be seen in § (3.5.2), $\not{D}$ is the most time-consuming part of an iteration. However, for the red-black algorithm, $\not{D}$ only acts upon a single parity of ψ, so the amount of work required for an iteration is essentially the same as for the naïve version. The main advantage is that the red-black decomposition is a kind of preconditioning of the fermion matrix. We must therefore compare the number of iterations taken to reach a given precision for the solution for Eq. (2.12) and for Eq. (2.16) and check that these solutions do agree. The main criterion used to decide whether the red-black method should be used is how quickly it converges. For details of comparisons, see § (2.4.1) and § (2.5.2).

2.3.2 Red-Black Preconditioning for Clover action

Since the improved Clover action of § (1.4.4) only connects sites of opposite parity, it is also relatively straightforward to perform a red-black preconditioning on it. To calculate the improved quark propagator, we solve equations of the form

$$\mathcal{M}^C \psi = \eta^R \tag{2.18}$$

where η^R is the rotated source in Eq. (1.61). The Clover fermion matrix is of the form

$$\begin{aligned} \mathcal{M}^C &= A - \kappa \not{D} \\ &= (1 - \kappa \not{D} A^{-1}) A \end{aligned} \tag{2.19}$$

where A is the purely local term of Eq. (1.58). If we left-multiply Eq. (2.18) by $(1 + \kappa \not{D} A^{-1})$, we end up with

$$(A - \kappa^2 \not{D} A^{-1} \not{D}) \psi = (1 + \kappa \not{D} A^{-1}) \eta^R. \tag{2.20}$$

Like Eq. (2.13), this only connects sites of the same parity and so we can solve

$$(A_{ee} - \kappa^2 \not{D}_{eo} A_{oo}^{-1} \not{D}_{oe}) \psi_e = \eta_e^R + \kappa \not{D}_{eo} A_{oo}^{-1} \eta_o^R \tag{2.21}$$

and reconstruct the opposite parity solution from

$$\psi_o = A_{oo}^{-1} (\eta_o^R + \kappa \not{D}_{oe} \psi_e). \tag{2.22}$$

At the start of the inversion, we need to calculate A on even sites and its inverse on odd sites. We calculate A^{-1} using an $L^\dagger D L$ decomposition which introduces small rounding errors. To ensure that this does not cause us any problems, we always re-start a red-black Clover fermion matrix inversion with a standard inversion. This checks that we have a solution to Eq. (2.1) with the required residue. In practice, we find that the rounding errors in calculating A^{-1} have no significant effect on the residue.

2.4 Performance of CG Algorithms

In this section, we compare the properties of two different formulations of the CG algorithm. § (2.4.1) investigates red-black preconditioning for both versions, and § (2.4.2) compares their convergence properties.

2.4.1 Red-Black Preconditioning for CG

We have compared the effect of using the red-black preconditioning for both CG algorithms. These tests have been performed on a $16^3 \times 48$ gauge configuration, which has been equilibrated by 2500 sweeps of the lattice. Details of the equilibration algorithm may be found in § (1.4.5). As shown in § (2.3.1), we should use the same convergence criterion for the red-black and standard inversions. For a range of κ values, which includes the physically interesting region, we determine the number of iterations required to achieve a residue such that $|r|^2 < 1.0 \times 10^{-12}$ for the standard and red-black versions of both algorithms.

The results are given in Table 2.5, and Table 2.6. During these tests we checked that all the methods give the same pion propagator. Comparing the number of iterations taken to converge is a sensible test since the time per iteration is almost independent of whether red-black is used. In fact, the red-black versions are a little faster.

The tables clearly show the advantage of the red-black method in the number of iterations required. The performances of the two CG algorithms are very similar. In both cases, the red-black preconditioning gives an improvement by approximately a factor of 3 at all values of κ. A similar improvement was found for a CG algorithm in [Hockney 90]. The gain is sufficiently large that it seems essential to

Kappa	Iterations Required		Gain
	CGLN	Red-Black	
0.1300	83	29	2.86
0.1400	154	52	2.96
0.1450	250	85	2.94
0.1500	625	210	2.98
0.1510	876	296	2.96
0.1520	1426	473	3.01
0.1523	1854	620	2.99
0.1526	2098	713	2.94
0.1529	2329	793	2.94

Table 2.5: Comparison of number of iterations required to reach a similar accuracy for the red-black and standard Least Norm CG algorithms.

Kappa	Iterations Required		Gain
	CGLR	Red-Black	
0.1300	77	28	2.75
0.1400	139	49	2.84
0.1450	220	78	2.82
0.1500	515	184	2.80
0.1510	720	259	2.96
0.1520	1085	386	2.78
0.1523	1437	516	2.78
0.1526	1928	673	2.86
0.1529	2203	762	2.89

Table 2.6: Comparison of number of iterations required to reach a similar accuracy for the red-black and standard Least Residual CG algorithms.

use preconditioning. Indeed, one of the advantages of the Clover action is that it is an improved action for which this preconditioning is possible. Throughout this chapter, most of the tests will use the red-black preconditioning.

The graphs in Fig. 2.4 show how the residue varies with iteration number for both versions of Least Norm CG at $\kappa = 0.1520$. The slope for the red-black algorithm is 3 times that for the Wilson inversion, as expected from above. However the similarity in forms between the graphs is striking and, perhaps, unexpected. This similarity shows how much the rate of convergence is dominated by the physics of the particular gauge configuration.

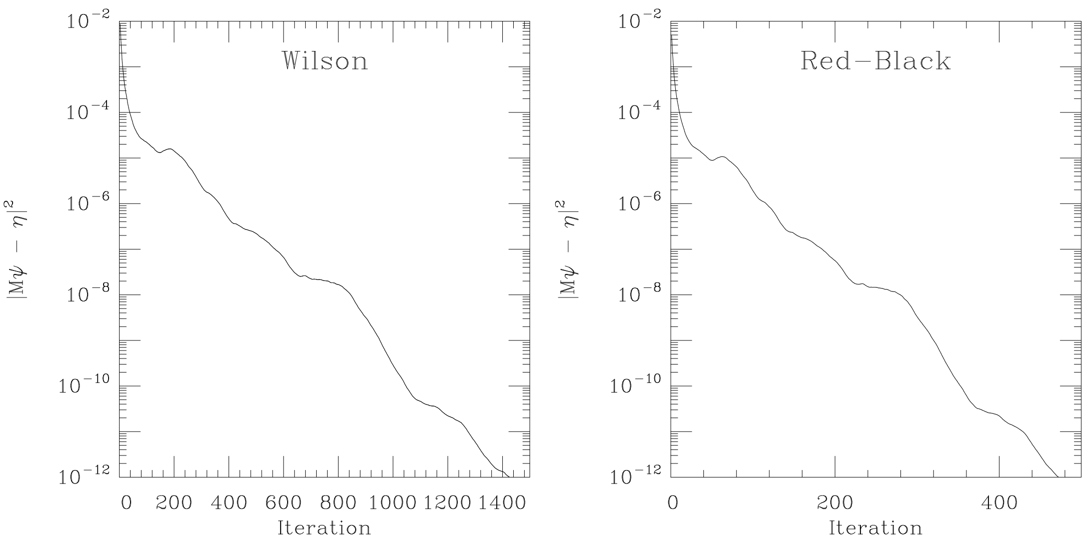


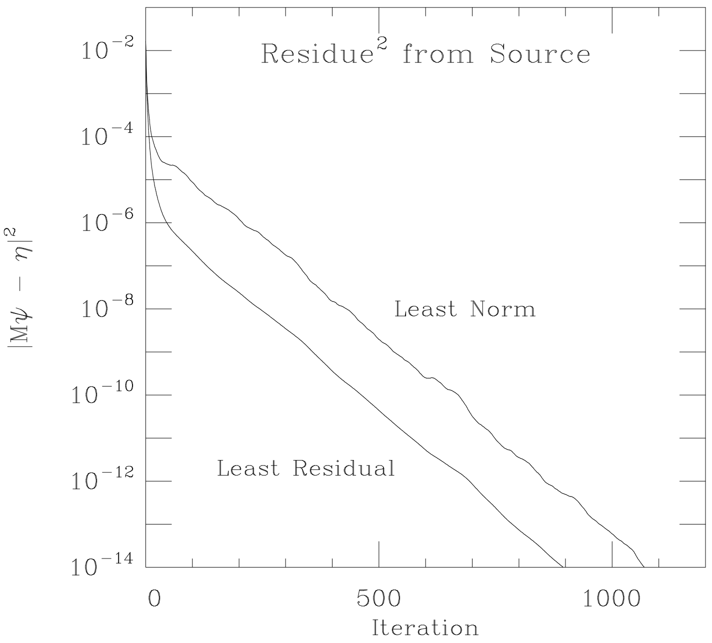
Figure 2.4: Comparison of the standard Wilson inversion (left) to the red-black method for Least Norm CG at $\kappa = 0.1520$. Note the factor of 3 in the scales.

2.4.2 Comparison between LN and LR

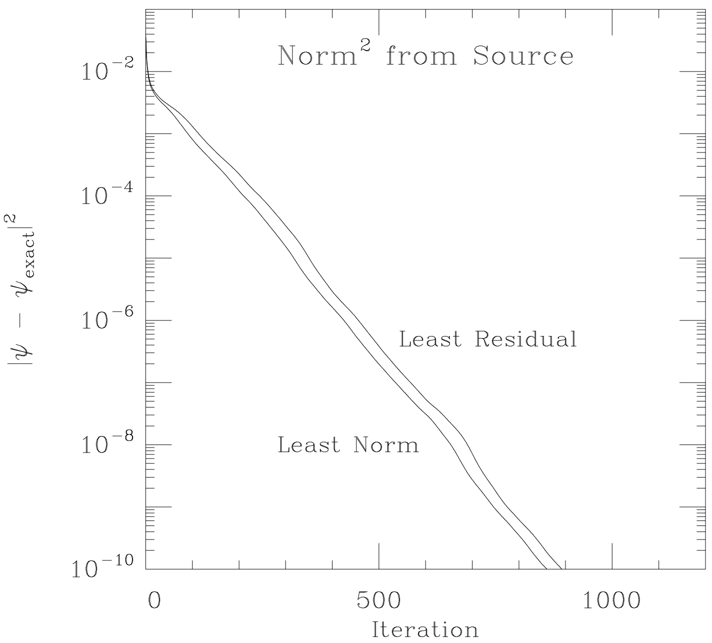
The efficiency of any particular matrix inversion algorithm is essentially determined by how much computer time it takes to get *sufficiently near* to the appropriate solution. The first question to ask is how we determine the nearness of a given approximate solution to the exact solution. The two most obvious measures are those given in Eq. (2.8) and Eq. (2.9). In practice, only the first of these can be evaluated readily. Most inversion algorithms require the evaluation of this residue, and so we get it for free. Correspondingly, we cannot normally calculate the error in ψ, Eq. (2.9), since it involves the unknown exact solution ψ_{exact}. However, it is clear that $|\psi - \psi_{\text{exact}}|$ is at least as good a criterion as r for measuring the nearness of a given approximate solution. To decide between the above two versions of the CG algorithm, it is important to consider both of these measures of distance. For a test program is is relatively easy to obtain a good estimate for $\psi - \psi_{\text{exact}}$. At the start of the test we can invert the fermion matrix with a much stricter convergence criterion than we intend using for the main test and then use this solution as a good approximation to ψ_{exact}. For the *normal* inversions, we used a convergence criterion of $|r|^2 < 10^{-14}$, whereas for the *exact* inversion we used $O(10^{-20})$, which corresponded to roughly twice the number of CG iterations. As described in § (2.2.7), the only freedom in these algorithms is the initial guess of ψ_0. We compare these two versions of CG starting from both the source and from a solution at a nearby κ.

Using the results of § (2.4.1), these tests were performed using red-black preconditioning. We began by calculating our estimate of ψ_{exact} at $\kappa = 0.1524$ on a $24^3 \times 48$ configuration which had been equilibrated for 5200 sweeps. We also calculated the solution at $\kappa = 0.1520$ so that it was available to use as an initial guess. These values were chosen to be as typical as possible. We then made two inversions with each of the two CG algorithms — the first starting from the source, and the second from the $\kappa = 0.1520$ solution. During these inversions, we measured $|\psi - \psi_{\text{exact}}|^2$ as well as the usual residual. The results can be seen in Fig. 2.5 and Fig. 2.6.

Consider first Fig. 2.5 which shows (a) the residue, and (b) the norm, against iteration number for each of the algorithms, starting with $\psi_0 = \eta$. Normally we only have the information from the top graph, i.e., the residues. From this it would

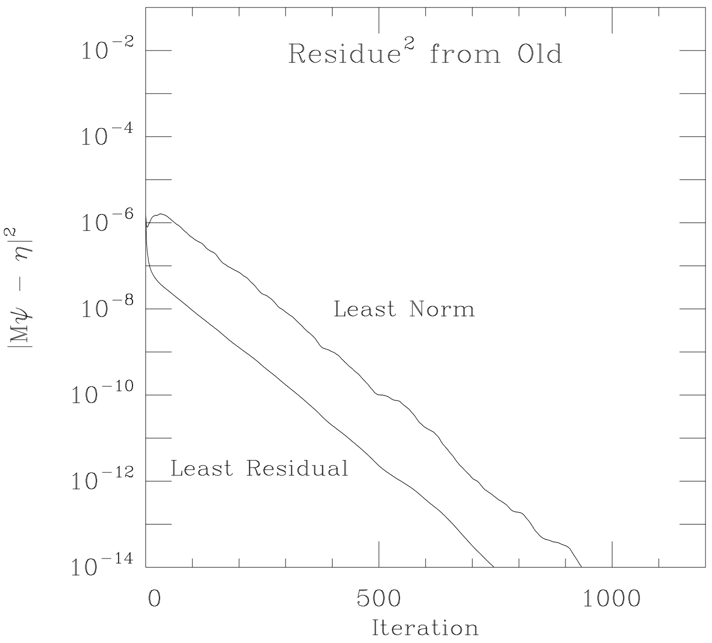


(a) Residue^2 *v.* iteration

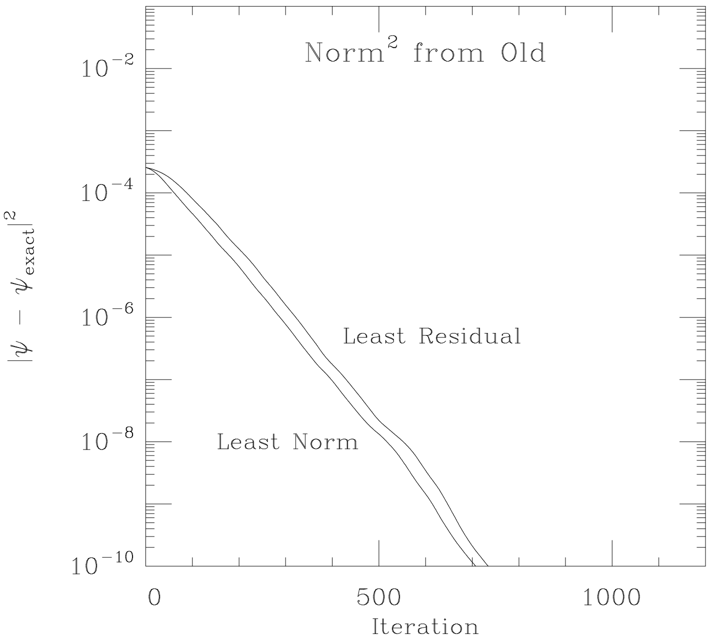


(b) Norm^2 *v.* iteration

Figure 2.5: Convergence properties of the CG algorithms, starting from the source as an initial guess.



(a) Residue^2 $v.$ iteration



(b) Norm^2 $v.$ iteration

Figure 2.6: Convergence properties of the CG algorithms, starting from the solution for a nearby κ value.

appear that the LR algorithm is clearly superior, in that, after a given number of iterations, it has a residue almost two orders of magnitude smaller than LN. This corresponds to a saving of $10\% \rightarrow 20\%$ in the number of iterations required. Moreover, LR is guaranteed to be monotonic in the residue which, we have already concluded, will constitute our convergence criterion. Nevertheless, the information in Fig. 2.5(b) must also be considered, since, at least for propagator calculation, it is the difference between our solution and the exact solution which is most likely to affect the physics. Fig. 2.5(b)[2] shows that after a set number of iterations, it is in fact, the Least Norm algorithm which is nearest to the exact solution. Despite the fact that LR has a significantly smaller residue, after any given number of iterations, LN always has a better solution. On this basis, we always use the LN algorithm since, although it is not guaranteed to be monotonic in the residual, for a given amount of computer time, it gives the most accurate solution.

Initial Guess

Looking at Fig. 2.6(b), which shows the norms starting from the solution at $\kappa = 0.1520$, it is clear that the form is similar to Fig. 2.5(b). The only significant difference is that using an old solution saves about 150 iterations. This is what we expect when $\kappa_{\rm old} - \kappa_{\rm new}$ is small, since, as $(\kappa_{\rm old} - \kappa_{\rm new}) \rightarrow 0$, the number of iterations required will also tend to zero. It is interesting to note that the slopes of the graphs are very similar, indicating that the rate of convergence is essentially independent of the initial guess. The improvement is solely due to the fact that $\psi(\kappa = 0.1520)$ is a good approximation to the solution $\psi(\kappa = 0.1526)$.

However Fig. 2.6(a) is quite different from Fig. 2.5(a). In particular, the LN

[2]It is important to note that the slopes of Fig. 2.5(b) and of Fig. 2.6(b) do not change significantly at small values of $|\psi - \psi_{\rm exact}|^2$. This indicates that our approximation to $\psi_{\rm exact}$ is sufficiently accurate for the purposes of this test.

algorithm is no longer monotonic in the residue — over the first 50 iterations, the residue increases by a factor of 2. For other pairs of κ-values, this effect can be even more dramatic. However, after the initial iterations, the slope of this line is very similar to that for LR and those in Fig. 2.5(a). Note that even while the residue from LN is increasing, $(\psi - \psi_{\text{exact}})$ is still monotonically decreasing.

Initially, we presumed that the improvement achieved by starting from a solution at a nearby κ was due to the overall shapes of the solutions being similar, i.e., the long distance physics being approximately correct. This leads to the idea of letting the initial guess be a different spin/colour component at the same κ, since, sufficiently far from the origin, these should be statistically very similar. Unfortunately, this turns out to be completely unsuccessful, in that it actually takes more iterations than starting from the source. This failure and the fact that corresponding slopes in Fig. 2.5 and Fig. 2.6 are the same seem to indicate that the improvement from using a solution at a nearby κ is actually due to having the correct shape near the origin, i.e., the large components of the propagator are nearly correct. This is borne out by the large improvement in $(\psi - \psi_{\text{exact}})$ from using an old solution.

The 12 components of a $24^3 \times 48$ propagator take up around 800 Mbytes of disc space which means that the logistics of data-handling make it difficult for us to calculate more than a single κ value in a run. We are, therefore, unable to start from an old solution and instead use the source as the initial guess. For the smaller $16^3 \times 48$ propagators, we can start from a solution at a nearby κ and see a gain in computer time of $O(10\%)$.

2.4.3 Summary

It is clear from § (2.4.1) that red-black preconditioning allows such large gains that its use is essential for both variants of the CG algorithm that we have investigated. The graphs in Fig. 2.5 and Fig. 2.6 along with the data in § (2.4.1) show that the basic behaviour of CGLR and CGLN is very similar. For example, the slopes in Fig. 2.5(a) are virtually indistinguishable.

Given this similarity, the criterion used for deciding the relative merits of the two algorithms was the rate of convergence in respect of the norm, $|\psi - \psi_{\mathrm{exact}}|$. This is the quantity most likely to estimate the errors in the physics we investigate. Fig. 2.5(b) and Fig. 2.6(b) indicate that, for a given amount of computer time, the LN algorithm always has a smaller value of the norm. We therefore regard CGLN as the standard CG algorithm with CGLR retained for cross-checking. In § (2.6), we will use CGLN to compare the performance of Minimal Residual against a conjugate gradient algorithm.

2.5 Performance of MR Algorithm

In § (2.5.1), we investigate how the performance of the Minimal Residual algorithm depends on the over-relaxation parameter, ω, and determine its optimal value. This is followed by an investigation into red-black preconditioning for MR.

2.5.1 Over-relaxation Parameter for MR

When ω is very small, the residue hardly changes at each iteration and when it is very large, the estimate of the solution moves wildly through solution space.

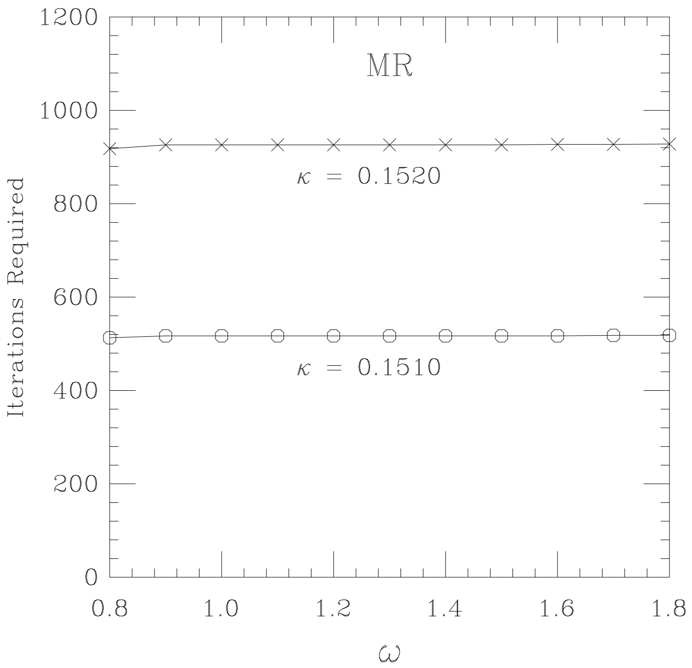
From the discussion in § (2.2.3), we hope to optimise the convergence for some value of ω slightly larger than 1.

To find this optimal value for the relaxation parameter, it is necessary to investigate how the number of iterations required to solve typical problems varies with ω. As in § (2.4.1), these tests have been carried out on a $16^3 \times 48$ gauge configuration equilibrated by 2500 sweeps of the lattice. We use a convergence criterion of $|r|^2 < 1.0 \times 10^{-12}$ and investigate the number of iterations taken to converge for realistic values of κ for both the standard MR and the red-black preconditioned MR. The results are shown in Fig. 2.7.

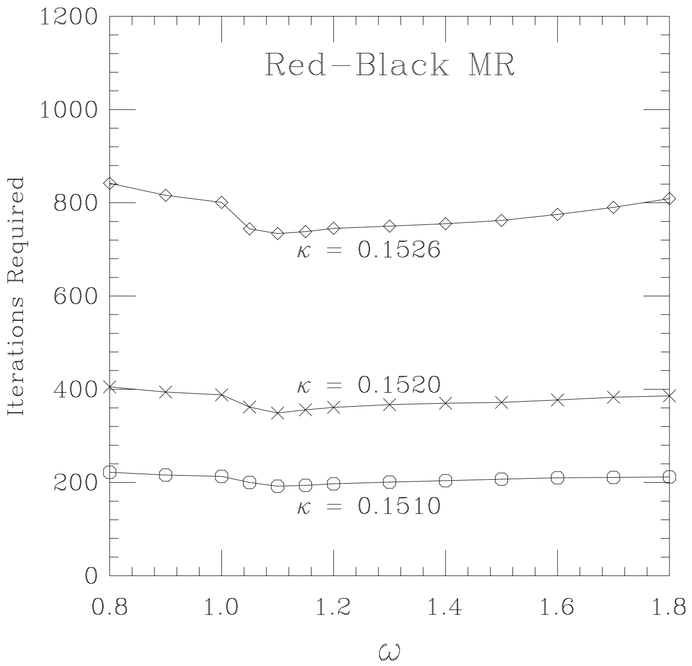
Fig. 2.7(a) shows that the performance of the standard MR algorithm seems to be essentially independent of the value of ω over the range $0.8 \to 1.8$. However, for the red-black algorithm in Fig. 2.7(b), there is an optimal value of $\omega = 1.1$ where the inversion takes 10% fewer iterations than for $\omega = 1.0$. It is important to note that this optimal value of ω is essentially κ-independent and that the performance of the algorithm is not critically dependent on ω in the range $1.05 \to 1.20$. For the Clover action, there is little gain from over-relaxation for either the red-black or standard inversions

These results show much smaller gains than the 30% achieved in [Gupta 89] for values of ω between 1.15 and 1.50. Their results are for dynamical Wilson fermions at $\beta = 5.2 \to 5.6$. This difference is explained by the data in [Hockney 90] which shows that the gain from over-relaxation is smaller at large β.

Throughout the rest of this thesis we will always use Over-relaxed Minimal Residual(ORMR) to mean MR with $\omega \simeq 1.1$ as this seems a reasonable choice for all the inversions we perform.



(a) Standard MR.



(b) Red-Black MR.

Figure 2.7: Number of iterations which MR takes to converge for a range of values of the over-relaxation parameter, ω.

2.5.2 Red-Black for ORMR

This section repeats the evaluation of red-black preconditioning from § (2.4.1) but this time for ORMR. We compare the number of iterations to converge to a residue of $|r|^2 < 1.0 \times 10^{-12}$ for standard and red-black ORMR using a range of κ-values on a $16^3 \times 48$ configuration. We use $\omega = 1.1$, but, as indicated in § (2.5.1), the results are almost independent of ω.

Kappa	Iterations Required		Gain
	MR	Red-Black	
0.1300	54	21	2.57
0.1400	95	37	2.57
0.1450	150	59	2.54
0.1500	364	135	2.70
0.1510	517	192	2.69
0.1520	926	349	2.65
0.1523	1227	472	2.60
0.1526	1835	734	2.50
0.1529	4493	1607	2.80

Table 2.7: Comparison of number of iterations required to reach a similar accuracy for the red-black and standard MR algorithms.

Table 2.7 shows that for the ORMR routine, the improvement from red-black is similar to that for the CG algorithms. There is a gain of almost a factor of 3 at all values of κ. These results are similar to the improvements for the red-black ORMR from [Gupta 89]. As is the case for the CG algorithms, there is a large gain from the red-black preconditioning with no overhead and so we always use the red-black version of ORMR.

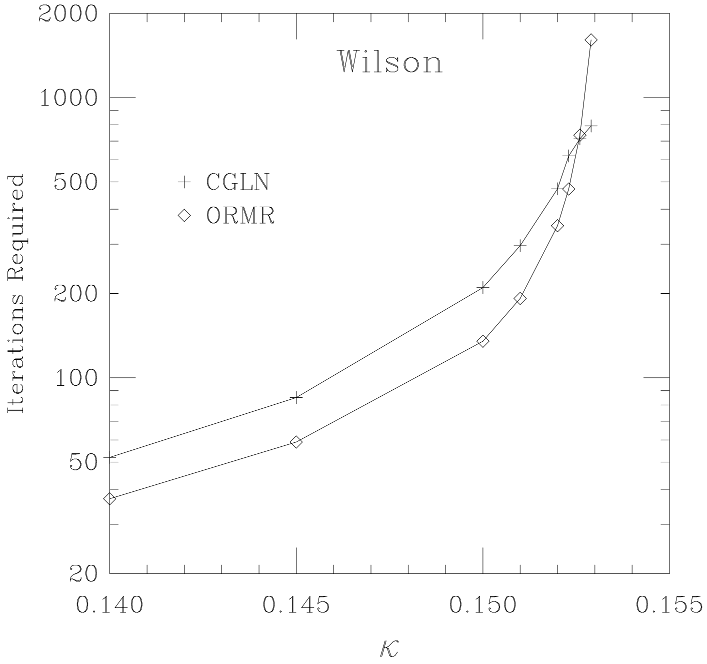
2.6 Choice of Inversion Algorithm

The main advantage which the ORMR algorithm has over CG algorithms is that it does not require a hermitian, positive definite matrix and so can be directly applied to $\mathcal{M}$, rather than, e.g., $\mathcal{M}^\dagger\mathcal{M}$. This means that the problem is less badly conditioned and that each iteration takes only half the computer time of a CG iteration. We can also vary ω to optimise the convergence but, as seen in § (2.5.1), the performance is almost insensitive to ω at the value of β in which we are interested.

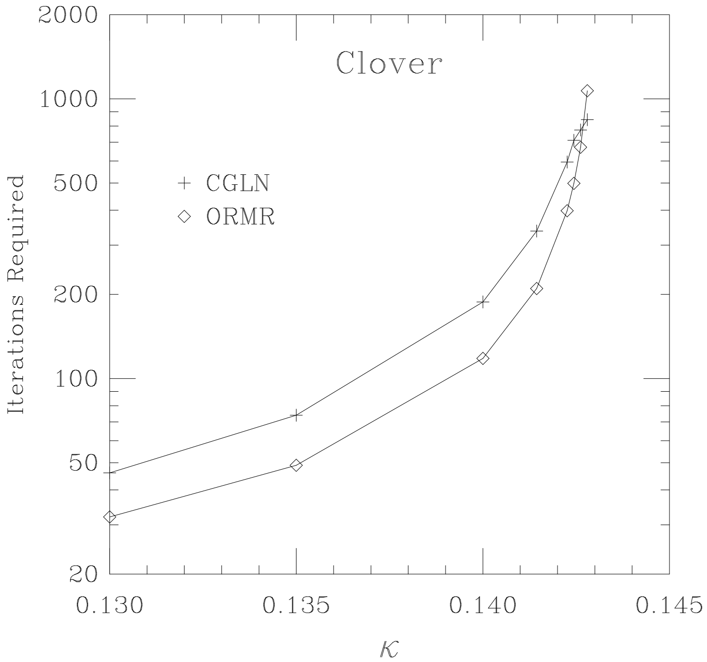
On the other hand, the CG algorithms are guaranteed to converge for all values of κ while MR fails as κ approaches some critical value. We can formulate a version of CG, called CGLN, which monotonically reduces the error in the solution rather than the error in the equation.

To decide which algorithm is most suitable to investigate the physics in which we are interested, we have compared the number of iterations required for the red-black versions of ORMR and CGLN. This test has been carried out on a $16^3 \times 48$ lattice for both the Wilson and Clover actions and the results are shown in Fig. 2.8.

The 5 largest κ values for the Wilson action and 4 largest for the Clover action correspond to those used in the investigation of the hadron spectrum in Chapter 4. The results for the two different actions are very similar. At small values of κ, ORMR takes significantly fewer iterations to converge. This is in addition to the fact that each iteration takes only 50% of the time of the corresponding CG iteration. As κ approaches the critical region, it is clear that ORMR suffers more from critical slowing down than CGLN. Nevertheless, at all values of κ other than the largest, the number of iterations taken by ORMR is equal to or smaller than that for CGLN and so ORMR is at least a factor of 2 faster. At the largest κ



(a) Wilson Action.



(b) Clover Action.

Figure 2.8: Comparison of CGLN and MR for the Wilson and Clover actions.

values, the situation is less clear. For the Clover action, ORMR seems to still have a slight advantage while for Wilson the performance is about equal.

In general, the fact that ORMR does not require the inversion of $\mathcal{M}\mathcal{M}^\dagger$ seems to outweigh the ability of CGLN to minimise along conjugate directions at least for most of the κ values used. One of the advantages of concentrating on inversion algorithms involving successive line minimisations is that it is easy to switch between different algorithms. We always invert using ORMR, but if there is difficulty converging, e.g., at the largest κ values, we can automatically switch to CGLN which always converges.

Chapter 3

QCD on Maxwell

The first section in this chapter will describe Maxwell, the computer we use to perform QCD simulations. This is a SERC funded parallel supercomputer, sited in Edinburgh, which is used for *Grand Challenge* projects in Physics and Chemistry. The primary user of Maxwell is the United Kingdom Quantum Chromodynamics collaboration (UKQCD), which is a collaboration of many of the UK's lattice gauge theorists. Much of the work on which this thesis is based was performed as a member of UKQCD.

§ (3.2) will describe how data can be transferred between different processors on Maxwell and outlines the types of communication routines required in QCD simulations. The following section will give details on how to implement the MC algorithms efficiently and the verification tests that have been performed on them. § (3.4) will describe the implementation of $\not{D}$, which is the major routine of the inversion algorithms. The chapter will finish with a description of the tests carried out on the matrix inversion code and with details of the performance obtained.

3.1 Maxwell

Maxwell is a concurrent vector supercomputer manufactured by a British company called Meiko. It consists of 64 vector processors connected together with the aid of transputers. Each node consists of an i860 vector processor and 2 transputers which all have access to 16 Mbytes of memory. Each of the transputers

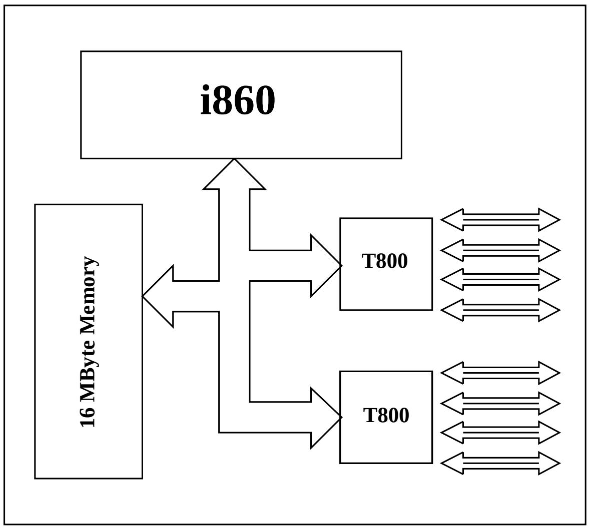


Figure 3.1: A symbolic representation of a single node on Maxwell.

has 4 *bi-directional* links which can be connected to other nodes. Therefore one can view a node as simply a vector processor with 16Mbytes of memory and 8 connections to the outside world. These links can be connected in a variety of different ways to give an appropriate processor topology for the particular problem. Since lattice QCD is defined on a 4-D space-time lattice, the natural topology is a 4-D hypercuboid with each of the processors responsible for its own hypercubic section of the lattice.

Fig. 3.2 shows how 32 processors can be connected as a $[2, 1, 4, 4]$ grid. Note that since the links are bi-directional, we only require extra physical connections to implement the periodic boundary conditions on those directions with more than two processors and therefore that each processor in the grid has exactly five

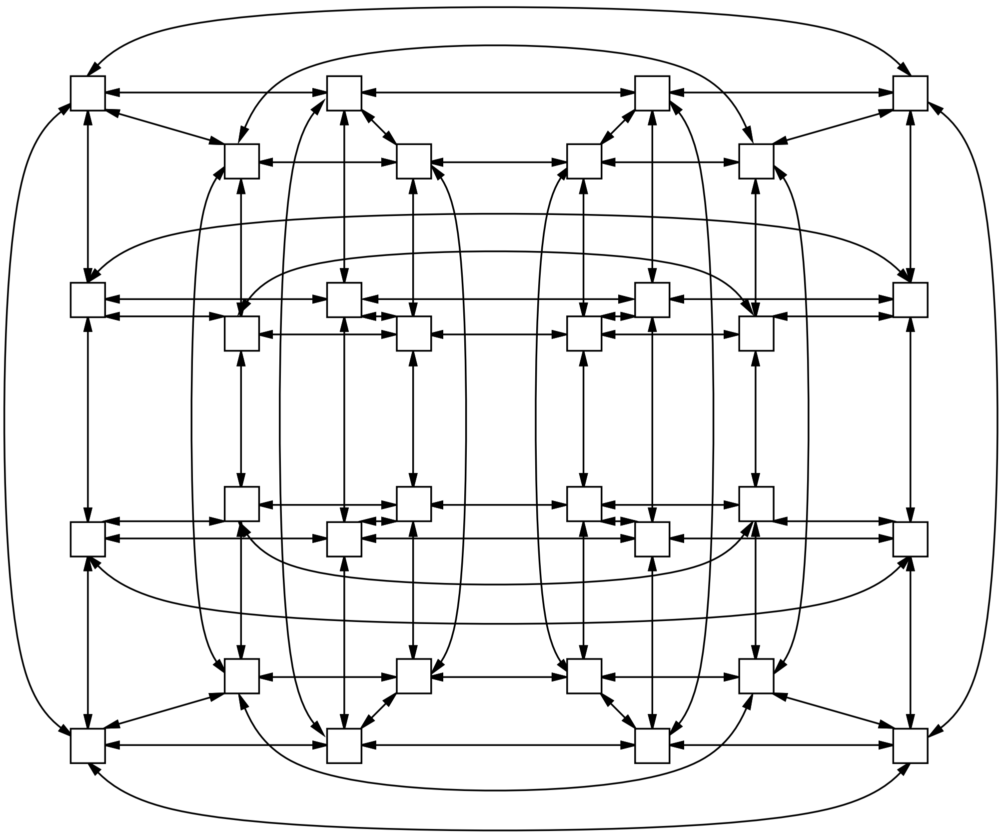
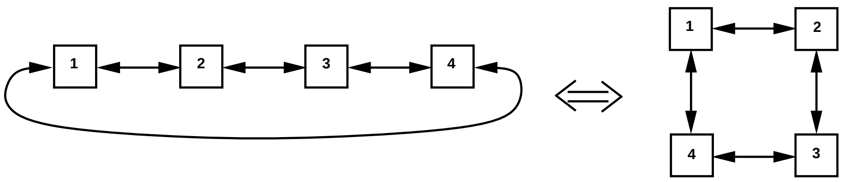


Figure 3.2: Processor topology for QCD.

connections. To obtain a $[2, 2, 4, 4]$ grid, one can overlay another copy of Fig. 3.2 and link corresponding processors.

Since 4 processors linked periodically are equivalent (bar re-labelling) to two directions of 2 processors,



the grid in Fig. 3.2 can also be regarded as a $[2, 2, 2, 4]$ grid or, indeed, as equivalent to a 2^5 binary hypercube. For algorithms, such as Fast Fourier Transforms (FFTs), which require communications between all processors in the grid, it may be more convenient to consider the processor topology as a binary hypercube.

3.2 Communication Between Processors

The major difference between implementing QCD on Maxwell and on a more conventional supercomputer, such as a Cray, is that each processor on Maxwell only has information about a fraction of the lattice. Since we often perform operations where the operands are on different sites, we will need to transfer information between processors. It is therefore important that the topology of the lattice is consistent across processors. The next section describes how we ensure this using *neighbour tables* and how we shift information between processors. § (3.2.2) gives a brief description of our communication harness, CSTools, and our major communication routines. The final part of this section describes how we implement different Boundary Conditions on a parallel computer.

3.2.1 Neighbour Tables

Each processor labels the sites on its local lattice by 2 numbers — the *parity* of the site from Eq. (2.14) and a position index (given in some arbitrary but unique way.[1]) The fermion vectors are stored in an array over the pairs of site labels, with the 3 colours and 4 spinor components of each site stored contiguously. The whole *topology* of the lattice is contained in a number of *neighbour table* arrays, which are lists of site indices. The neighbour of site (par, index) in the *positive* μ direction is given by

$$(\text{par, index}) + \hat{\mu} = (\text{!par, up[mu][par][index]}), \tag{3.1}$$

where !par is used to mean the opposite parity, since neighbouring sites must always be of different parity. On a parallel computer, such as Maxwell, some of these neighbouring sites will be on an adjoining processor. We deal with this by

[1] In practice, we simply label the sites of a given parity by sequentially looping over the x, y, z and t directions.

giving a unique index to each of these *off-processor* sites pointing to a separate part of the array from the local sites. We shall call this separate region the *tail*, as in practice it will be tagged onto the local part (the *body*) of the array. As a more concrete example, consider how we would set

$$\rho(x+\hat{\mu}) = \sigma(x),$$

where ρ and σ are simply vectors of real numbers. This involves shifting an array of numbers, σ, in the positive μ direction. Conceptually, we would do this by looping over the sites $x = (\text{par}, \text{index})$ and calculating the appropriate neighbour via Eq. (3.1). We can then copy the element of σ into the appropriate section of ρ. If the neighbouring site is not local to the processor, we store the corresponding value of σ in the appropriate part of ρ's tail. After we have looped over all the sites, the tail of ρ will contain those elements of σ, which correspond to the *upper boundary* of the local lattice in the μ direction. We then send these elements to the processor in the positive μ direction while simultaneously receiving a corresponding set from the processor in the negative μ direction which is to be stored in the local body of ρ. In practice, we would extract the boundary elements at the beginning so that we could be communicating them at the same time as re-arranging the elements on the local lattice. All of the boundary communications which we require will be of a similar form to this example, (see e.g., § (3.4) and Eq. (3.7)).

3.2.2 Communications via CSTools

CSTools is a collection of software packages which handle the communication between processors in a way which is mostly hidden from the user. Versions of CSTools are available for various computer hardwares so that programs which use CSTools are somewhat independent of any particular hardware. However, for a program to run efficiently, it is still necessary for it to be tailored to the target machine. Each processor may be given a unique label so that any other processor in the network may send messages to it. CSTools will transmit the message through any intermediate processors in some more-or-less optimal fashion. Since lattice QCD is a grid-based problem which requires only nearest neighbour communications, we do not require all the utility of CSTools. Its main strength for our purposes is that it deals with inter-node and intra-node transfers in a way which the user can ignore. Technical details of CSTools may be found in [Meiko 89].

For our problem we require only a small number of simple communication routines. Indeed, apart from the initialisation, the communications are all variations on two basic routines.

Boundary Transfer To calculate Wilson loops or staples (see § (3.3.1)) we need, e.g., to multiply $SU(3)$ matrices at sites x and $(x + \hat{\mu})$. As described in § (3.2.1), this requires getting boundary data from the neighbouring processor in the $+\mu$ direction and, correspondingly, sending data to the processor in the $-\mu$ direction. In other words, *shifting* data needs boundary communication. To implement $\not{D}$, (see § (3.4)) we shift data in all 4 directions simultaneously.

Global Summation All the CG and MR routines for the matrix inversions have vector scalar products as a component. After the local calculations for the product have been performed, each processor needs to know the sum of the results over the whole lattice. We could make each processor send its local result to a nominated master processor which could calculate and then broadcast the global answer. This is inefficient as it leads to bottlenecks in the communication structure. If each processor performs the global sum, it is important that they all sum the numbers in the same pre-determined order to ensure that each gets exactly the same result. Otherwise, when the code was re-run, the rounding errors would differ slightly and it would be impossible to check reproducibility.[2] The algorithm we use to calculate the global sum is to successively loop over the 4 directions summing over the corresponding plane of processors. This distributes the communications evenly over the network and we can also easily ensure that the order of the summations is the same everywhere.

Further details of the specific implementation of these routines may be found in [Booth 91].

3.2.3 Implementing Boundary Conditions

Since all lattice simulations are necessarily on a finite grid, we are forced to decide what to do at the edge of the lattice. The following are the most common boundary conditions (BCs) to impose.

Periodic BCs If there are L_μ sites in the μ-direction, labelled $0, 1, \ldots, L_\mu - 1$, then, for periodic BCs in the μ-direction, we set

$$\rho(x_0, \ldots, L_\mu, \ldots) = \rho(x_0, \ldots, 0, \ldots).$$

These are the most common BCs since they are easy to implement.

Anti-periodic BCs These require

$$\rho(x_0, \ldots, L_\mu, \ldots) = -\rho(x_0, \ldots, 0, \ldots)$$

These are often compared with periodic BCs to give an indication of how large a difference changing the BCs make. This, in turn, indicates how large *finite-size effects* are.

[2]For a Metropolis type accept/reject step in, e.g., an HMC simulation, even slightly different results on different processors could lead to different decisions and hence disaster!

Dirichlet BCs For these we set the elements outside the lattice to be zero, i.e.,

$$\rho(x_0, \ldots, L_\mu, \ldots) = 0 = \rho(x_0, \ldots, -1, \ldots).$$

There are various other BCs which are possible, such as *skew-periodic*, but the above are usually preferred if they can be implemented efficiently.

To investigate finite-temperature QCD, the BCs in time must be periodic for the gauge fields and anti-periodic for the anti-commuting fermions. For the lattices we use, the time direction is much longer than the spatial directions, and so the BCs in t are relatively unimportant. We always use periodic BCs for the gauge fields in all directions, as $-U \notin SU(3)$, However, even at zero temperature, we retain anti-periodic BCs in t for all fermionic fields ψ.

$$\psi(\vec{x}, t = L_0) = -\psi(\vec{x}, 0) \tag{3.2}$$

We are only ever interested in quantities, such as $\bar{\psi}(x)U_\mu(x)\psi(x + \hat{\mu})$, which are invariant under the gauge transformation in Eq. (1.43). It is therefore more convenient to implement Eq. (3.2) by flipping the sign of the gauge links which cross the time boundary. For the fermionic sections of the code, we set

$$U_0(\vec{x}, t = L_0) \rightarrow -U_0(\vec{x}, t = L_0), \tag{3.3}$$

where U_0 are the links in the x_0 or t direction. For the sections of the code concerned with the gauge fields (such as making the $F_{\nu\mu}\sigma_{\nu\mu}$ for the Clover action), we ensure the links have their original values. By varying the fermionic BCs in the spatial and temporal directions, we have numerically checked that the effect of different BCs on, e.g., the pion mass is small. This indicates that the errors in our calculations due to finite-size effects are probably not too important.

When using non-periodic BCs on a parallel computer, it is vital to identify the boundaries correctly. CSTools allows us to do this by letting each processor know its coordinates in the processor grid and the size of the grid. A particular processor only flips its links on the time boundary if it is at the end of the grid in the t-direction. We have checked that this is executed correctly by comparing propagators calculated on different sizes of processor grid (see § (3.5.1)).

3.3 Configuration Generation on Maxwell

Both the algorithms we use for generating gauge configurations break down into two sections (see § (1.4.5)). The following steps are required to update a particular link, $U_\mu(x)$.

1. Calculate the local potential using the staple defined in Eq. (1.67),

$$\begin{aligned} R(x,\mu) &= \sum_{\nu\neq\mu}\Big[U_\nu(x+\hat{\mu})U_\mu^\dagger(x+\hat{\nu})U_\nu^\dagger(x) \\ &\qquad\qquad + U_\nu^\dagger(x-\hat{\nu}+\hat{\mu})U_\mu^\dagger(x-\hat{\nu})U_\nu(x-\hat{\nu})\Big]. \end{aligned}$$

   The part of the staple in the $\mu\nu$-plane is shown in Fig. 3.3.

2. Use the staple to update the link via the three $SU(2)$ subgroups. This is then a local operation.

The main difference between the Over-relaxed (OR) algorithm and the Cabibbo-Marinari (CM) heatbath is in the choice of the $SU(2)$ matrices used for the update. The timings for the update of a $24^3 \times 48$ lattice distributed over 32 processors are shown in Table 3.1.

We define a *compound sweep* of the lattice to be 5 OR steps plus 1 CM heatbath on each link. At the end of each sweep we reunitarise the links to prevent *drifting* due to rounding errors. A compound sweep of the lattice takes 126 seconds of which

Algorithm	Routine	Time Taken (secs)
Over-relaxation	Staple	11.9
	$SU(2)$ updates	5.8
	Total	17.7
Heatbath	Staple	11.9
	$SU(2)$ updates	21.2
	Total	33.1
Reunitarisation		4.4

Table 3.1: Performance of Pure Gauge algorithms.

60% is spent calculating staples. The $SU(2)$ updates are local and relatively straightforward to optimise. However, the staple routine requires communication between processors and we must therefore implement it intelligently.

3.3.1 Calculating Staples and Loops

We see from the previous section that the major part of the time used to generate configurations is spent calculating the products of gauge matrices shown in Fig. 3.3.

To update the *dotted* link, $U_\mu(x)$, we need to sum all the upper and lower staples for all the $\nu \neq \mu$ directions. In Fig. 3.3, the circle at one end of each link indicates on which site the link is stored. For the sake of clarity, we shall assume site x has *even* parity.

To calculate the upper staple in the ν-direction, we first need to shift the appropriate odd U_μ links downwards in the ν-direction and shift the odd U_ν links downwards in the μ-direction. These two boundary transfers can be overlapped

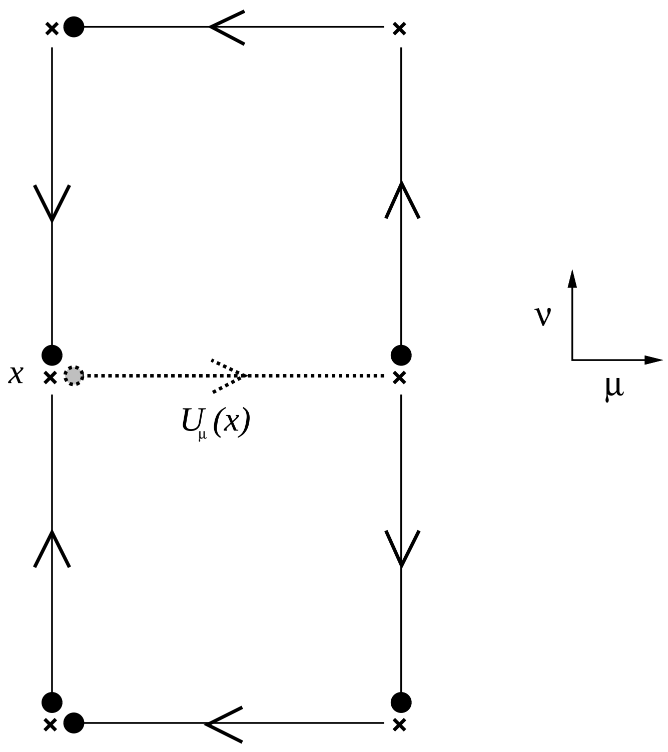


Figure 3.3: Staple in $\mu\nu$-plane required to update $U_\mu(x)$.

to effectively double the inter-node communication bandwidth. After this the necessary three links are all on site x, and the product is now local.

Due to the way that links are stored on sites, calculating the lower staple is slightly more complicated and we cannot overlap the communications as above. So, to keep the amount of communication to a minimum, initially we calculate the staple on site $(x - \hat{\nu})$. We need to shift $U_\nu(x - \hat{\nu} + \hat{\mu})$ to there. Site $(x - \hat{\nu} + \hat{\mu})$ has even parity, so the even U_ν links are communicated downwards in the μ-direction for the lower staple to be calculated at $x - \hat{\mu}$. The result is now communicated upwards in the μ-direction to site x.

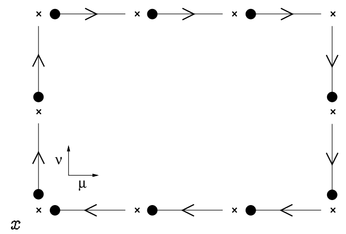
Wilson Loops

These are the products of links around closed rectangles of the lattice. We define the $n \times m$ Wilson loop from site x in the $\nu\mu$-plane to be

$$
\begin{aligned}
W^{\nu\mu}_{nm}(x) \;=\; \mathrm{Tr}\Big[& U_\nu(x)\ldots U_\nu(x+(n-1)\hat{\nu}) \\
& \times U_\mu(x+n\hat{\nu})\ldots U_\mu(x+n\hat{\nu}+(m-1)\hat{\mu}) \\
& \times U^\dagger_\nu(x+(n-1)\hat{\nu}+m\hat{\mu})\ldots U^\dagger_\nu(x+m\hat{\mu}) \\
& \times U^\dagger_\mu(x+(m-1)\hat{\mu})\ldots U^\dagger_\mu(x)\Big].
\end{aligned}
$$

For example,

$$W^{\nu\mu}_{23}(x) = \mathrm{Tr}$$



The 1×1 loop is simply the plaquette which enters into the gauge action.

To calculate a Wilson loop, we calculate the upper and lower L-segments independently. If we start with the links pointing to site $(x + n\hat{\nu} + m\hat{\mu})$ and work backwards towards x, these can be calculated in a similar manner to the staple using shifts and $SU(3)$ multiplications.[3]

3.3.2 Tests of MC Algorithms

It is very difficult to be absolutely sure that any MC algorithm has been correctly implemented, since there is a non-zero probability of any configuration being produced. However, it is important to be as confident as possible in the code. Many

[3] There are more efficient methods of calculating Wilson loops, but we only want to measure a number of small loops to, e.g., check equilibration. For our purposes, then, this method will be sufficient.

of the routines can be verified *ex situ* on test data. To check that we are generating reasonable configurations, we can measure a number of Wilson loops. We performed long runs on small lattices, (such as $4^3 \times 8$), and compared the results to those tabulated in [Hasenfratz 84]. We were able to get similar statistics and values for a number of loops without difficulty. This test was performed for the heatbath algorithm and repeated for the combination of 5 OR plus 1 CM heatbath which we actually use.

We have also compared loops measured on our $16^3 \times 48$ configurations with results tabulated in [Hasenfratz 84]. These data came from [Barkai 84] and were calculated on a 16^4 lattice.[4] They made 100 measurements separated by 10 Metropolis sweeps and the naïve error is quoted. For our data we made 200 measurements separated by 10 sweeps, starting from configuration 2000. The data was divided into $10 \rightarrow 20$ *bins*, and the error was estimated using the *jackknife* method from [Yang 86]. The results are shown in Table 3.2. The major difficulty with this test is the quality of the benchmark data. Since these use the Metropolis algorithm and the quoted error does not take into account correlations between successive measurements, it is undoubtedly a lower bound on the real error. Comparing the error estimates on our data using the naïve and jackknife methods suggests that the errors on the [Barkai 84] data are underestimated by at least a factor of $2 \rightarrow 3$. The other problem is that our data has loops including the t-direction. Since we are only comparing small loops, the finite-size effects should make only a small difference. Despite these difficulties, the agreement between the two sets of data is reasonably good, in that the only real discrepancy is in the plaquette. This apparent discrepancy is almost certainly due to the underestimation of the errors.

Further tests of the configuration generation require a full analysis. [Michael 91] has looked at the static quark potential on configuration 4400 from the $24^3 \times 48$

[4] Actually a $16^3 \times 32$ lattice, but they only measured loops orthogonal to the long axis.

Loop	$16^3 \times 48$ Data	[Barkai 84]
1×1	0.61361 (3)	0.61371 (1)
1×2	0.41008 (4)	0.41018 (6)
1×3	0.27940 (4)	0.27951 (7)
1×4	0.19127 (4)	0.19138 (7)
2×2	0.21703 (4)	0.21718 (8)
2×3	0.12333 (5)	0.12348 (7)
2×4	0.07156 (3)	0.07168 (6)
3×3	0.06257 (5)	0.06275 (7)
3×4	0.03312 (5)	0.03321 (6)
4×4	0.01637 (3)	0.01631 (5)

Table 3.2: Comparison of Wilson Loops at $\beta = 6.2$.

lattice. The results are in good agreement with [Perantonis 90] on slightly smaller lattices, though more statistics are clearly needed! While calculating the quark propagators, we also check that the pion mass (see Chapter 4), agrees with known results on similar lattices from [ELC 91].

Test of Over-relaxed Algorithm

The OR algorithm is explicitly constructed so that the change in the action is zero. This corresponds to the average plaquette, W_{11} remaining constant. The equilibration properties of the OR algorithm are excellent, so we would expect the larger loops to change rapidly. Starting from an equilibrated configuration we performed 100 sweeps of the OR algorithm, measuring the Wilson loops after each sweep. The results for this test can be seen in Fig. 3.4.

We can see that the plaquette remains constant (to 1 part in 10^8) while W_{23} fluc-

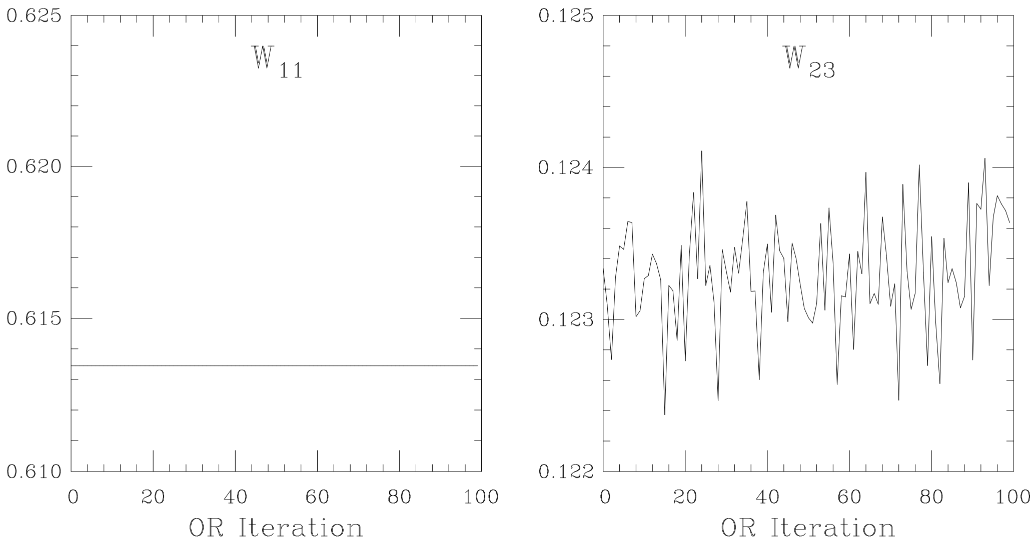


Figure 3.4: Changes in the W_{11} and W_{23} Wilson loops for the Over-relaxation algorithm. The scales on the y-axes differ by a factor approximately equal to the ratio of the average values of the loops. This ensures that relative fluctuations of the same importance would have similar size.

tuates wildly with an autocorrelation time of at most a few iterations. The other loops have a similar behaviour. This is an excellent test that the OR algorithm is correctly implemented and, since the OR and CM algorithms have many routines in common, is also an important test of much of the code.

Reproducibility Tests

The MC routines are very sensitive to communication or other hardware errors since they might skew the distribution of the configurations in some unknown fashion which would be very difficult to detect. This is in contrast to the matrix inversion routines where we can readily check the answer by substitution, and then restart if necessary. Since configuration generation takes only around 10% of the total computer time for a QCD simulation, we always use *reproduced* configurations as a precaution, i.e., we check the configurations by running the same job on 2 sets of 32 processors. As well as providing confirmation of the configurations, this is also a good test of both the hardware and software.

3.4 Implementation of $\not{D}$

As has been seen in § (2.2.4), one of the main building blocks of any inversion algorithm is the action of $\not{D}$ on a fermion vector. It is important that this is performed in as efficient a way as possible, as it is invariably the most time-consuming part of the inversion (see § (3.5.2)). Since $\not{D}$ connects neighbouring sites it will require communication between neighbouring processors on Maxwell. This section describes how we implement $\not{D}$ on Maxwell and details our communication strategy.

In Eq. (1.50), we defined $\not{D}\psi$ to be

$$\not{D}\psi(x) = \sum_{\mu} \left(U_\mu(x) P^+_\mu \psi(x+\hat{\mu}) + U^\dagger_\mu(x-\hat{\mu}) P^-_\mu \psi(x-\hat{\mu}) \right).$$

The projection operators, $P^\pm_\mu = (1 \pm \gamma_\mu)$, project out two-spinor components of the four-spinor ψ. The main reason we chose the Wilson r-parameter in Eq. (1.45) to be 1 was to allow us to work with 2-spinors rather than 4-spinors. If we define the two-spinors, χ by

$$\chi^\pm_\mu(x) = P^\pm_\mu \psi(x), \tag{3.4}$$

we can write

$$\not{D}\psi(x) = \sum_{\mu} \left(U_\mu(x) \chi^+_\mu(x+\hat{\mu}) + U^\dagger_\mu(x-\hat{\mu}) \chi^-_\mu(x-\hat{\mu}) \right). \tag{3.5}$$

With the same notation, $\not{D}^\dagger$ has the form

$$\not{D}^\dagger\psi(x) = \sum_{\mu} \left(U_\mu(x) \chi^-_\mu(x+\hat{\mu}) + U^\dagger_\mu(x-\hat{\mu}) \chi^+_\mu(x-\hat{\mu}) \right). \tag{3.6}$$

This is very similar to $\not{D}$, and we can see that the procedure needed to calculate it will be the same.

Letting $\not{D}$ or $\not{D}^\dagger$ act on a four-spinor, ψ, can be divided into three stages.

1. Decompose 4-spinor to 2-spinors
2. $SU(3)$ matrix $\times$ 2-spinor
3. Reconstruct 4-spinor from 2-spinors

3.4.1 Decomposition and Reconstruction

These are the routines which execute the γ-algebra. They are local to each processor and so their only performance limitation is *Memory Bandwidth* (see § (3.5.2)). There are two advantages to having routines for the spinor algebra which are separate from the colour algebra.

- It allows the computationally expensive $SU(3)$ multiplication routines to act explicitly on 2-spinors rather than 4-spinors half of whose elements would be zero. Providing the spinor routines take up less time than the $SU(3)$ routines, this can be a major saving in time since it removes redundant arithmetical operations. The performance figures in § (3.5.2) show that these routines take about 40% of the time taken by the $SU(3)$ multiplications. Thus, using 2-spinors contributes an overall saving of 30% in the time for a $\not{D}$.

- In addition to the improved performance, the separation of the spin and colour operations makes the code easier to verify, since they can be tested independently. We have C routines which mimic each of the assembler routines, so we can *mix-and-match* and compare the results obtained. The structure of the code is also improved with separate routines which makes it more readily understandable.

The main disadvantage of explicitly using 2-spinors is the need for additional temporary storage. However, this can often be overlapped with workspace used elsewhere in the program. In any case, it is a price worth paying for the advantages outlined above.

3.4.2 SU(3) Multiplications

§ (3.5.2) shows that the inversion program spends around 2/3 of its time in this routine. To calculate $\not{D}\psi(x)$, we require the following products.

$$\begin{aligned} \pi^+_\mu(x) &= U_\mu(x)\chi^+_\mu(x+\hat{\mu}) \\ \pi^-_\mu(x) &= U^\dagger_\mu(x-\hat{\mu})\chi^-_\mu(x-\hat{\mu}) \end{aligned} \tag{3.7}$$

The operands for π^+ are on different sites and therefore, potentially on different processors, whereas the multiplication in π^- is local to $(x-\hat{\mu})$ though the result is needed at site x. Therefore, both of these require communication between neighbouring processors (see § (3.2.1)).

Overlapping Communication and Calculation

An efficient strategy for the implementation of $\not{D}$ is shown in Table 3.3. The notation used is a *pseudo-code* based on OCCAM, a language designed for parallel programming on transputers (see [Bowler 89]). Statements or blocks of statements within a `SEQ`...`END` construction are executed *sequentially* as in a standard computing language. The main difference between OCCAM and, for example, C, is the facility to program a number of actions to be performed simultaneously — blocks of statements within a `PAR`...`END` construction are executed in *parallel*. This maps directly on to the ability of most parallel computers to e.g., communicate between nodes at the same time as performing local calculations. As communication can

```
SEQ
   PAR
      calculate π⁻ on site (x − μ̂)
      PAR μ = 0 to 3
         move χ⁺_μ downwards in μ-direction
      END
   END
   PAR
      calculate π⁺ on site x
      PAR μ = 0 to 3
         move π⁻_μ upwards in μ-direction
      END
   END
END
```

Table 3.3: Pseudo-code outlining our strategy for implementing $\not{D}$ efficiently.

be a major overhead, we must use the time while data is being exchanged to continue useful work.

In our case, we can see how the $SU(3) \times$ 2-spinor section of $\not{D}$ divides into 2 blocks of code which are executed sequentially. Within these, the various actions are performed simultaneously. In the first half, we calculate the local products required for π^- at the same time as communicating the appropriate boundary elements of χ^+_μ downwards in each of the 4 directions (see § (3.2.1)). In the second half, we communicate the previously calculated products π^-, at the same time as calculating π^+ which requires the χ^+ boundaries communicated in the first half. From this, it should be clear that the division of the work into halves is determined by the data interdependencies i.e., in the first half we communicate the data needed for the later calculations, while in the second half we communicate the previously calculated results.

This should minimise the amount of time each processor spends waiting for data to arrive from its neighbours. In practice, the situation can be a little more complicated. For example, if the i860 is running optimised assembler routines at the same time as the transputers are starting a boundary transfer, the transputers are unable to get sufficient access to the memory to progress with the communication (see § (3.5.2)). In effect, the amount of overlap tends to zero as the efficiency of the assembler improves. At the same time, the stress on the hardware increases which seems to cause it to fail more frequently. For $\not{D}$, we do not normally attempt to overlap the communications and calculations, though we still communicate in all directions simultaneously. This increases the iteration time by about 10%, but seems to reduce significantly the frequency of hardware problems.

3.5 Code Testing and Performance

The previous sections have described the algorithms we use and their implementation on Maxwell. § (3.5.1) describes some of the tests we have performed to ensure the accuracy of our implementation of the inversion algorithms, e.g., a comparison with the analytic free field pion propagator. § (3.5.2) gives details of the overall performance of Maxwell as well as particular details of the various routines which constitute an iteration of CG or MR.

3.5.1 Verification of Matrix Inversion Routines

We have carried out individual tests on all the various communication, assembler and *C* routines involved in the propagator inversion codes. Throughout this project, we have had the policy that all the i860 assembler should be exactly mirrored by *C* routines written for clarity rather than performance. Any bugs in the assembler are likely to be completely disjoint from corresponding bugs in the *C*. This makes errors much easier to track down and also enables us to readily eliminate the assembler as the cause of any problems which appear.

Once all the individual routines have been verified, we have then carried out extensive tests on the various high level propagator routines. Whichever inversion algorithm is used for a given propagator component and whether or not we use the red-black decomposition (§ (2.3.1)), we should always see the same physics. In particular, the pion propagator should not vary. As we will see from Eq. (4.9), the timesliced pion propagator is given by

$$\Pi(t) = \sum_{\vec{x}} |G(\vec{x}, t; 0)|^2 , \tag{3.8}$$

and is therefore easy to measure.

Free Field Pion Propagator

In free field, i.e., for a unit gauge configuration, it is possible to calculate the pion propagator analytically by summing the appropriate Fourier components numerically. For Wilson fermions on an $L^3 \times L_t$ lattice, the fermion propagator can be written (see [Karsten 81]) as

$$G(\vec{x}, t) = \frac{1}{L^3 \times L_t} \sum_{\vec{k}} \sum_{k_0} e^{i\,k\cdot x} G(\vec{k}, k_0), \tag{3.9}$$

with

$$
\begin{aligned}
G(\vec{k}, k_0) &= \frac{-i\gamma_0 \sin k_0 - iK + [(1-\cos k_0) + M]}{\sin^2 k_0 + K^2 + [(1-\cos k_0)+M]^2} \qquad (3.10)\\
K &= \sum_{i=1}^{3} \gamma_i \sin k_i \\
M &= \sum_{i=1}^{3} (1-\cos k_i) + m.
\end{aligned}
$$

The momentum sum is over

$$k_\mu = \frac{2\pi(n_\mu + \delta_\mu)}{L_\mu}, \qquad n_\mu = 0, 1, \ldots, L_\mu - 1,$$

where $\delta_\mu = (\frac{1}{2})\,0$ for (anti)periodic boundary conditions in the μ-direction.

The fermion propagator can be evaluated numerically as it stands but it is possible to save computer time by performing the k_0 sum analytically. Details of this can be found in [Carpenter 85]. The pion propagator can then be calculated using Eq. (3.8).

The free field pion propagator is thus an important benchmark test for the propagator code, since we can calculate it exactly using an entirely independent technique. In particular, since all physical quantities are gauge invariant, we compare the analytic propagator with that calculated on a random gauge transformation of unity. This is a much better test than simply using the unit gauge since the unit matrix is obviously special. This has become one of our standard tests of the code which has been repeated whenever any upgrades have been made.

Having ensured that the single processor propagator code satisfied the above test, our initial tests of the parallel version used a configuration which was a gauge transform of unity on various small configurations of processors with various lattice sizes, boundary conditions and quark masses. A list of some of our tests is given in Table 3.4. By varying the processor configuration, we were able to check that the topology as defined by the neighbour tables (see § (3.2.1)) was consistent

Global Lattice Size	Range of Kappa	Processor Configuration	Boundary Condition in Time	Red-Black
$4^3 \times 8$	0.114	(1,1,1,1)	Per	No
	$\rightarrow$ 0.123	(1,1,1,1)	Anti	No
		(1,1,1,1)	Per	Yes
		(1,1,1,1)	Anti	Yes
		(1,1,1,2)	Per	Yes
		(2,1,1,1)	Anti	Yes
$6^3 \times 12$	0.114	(1,1,1,1)	Per	No
	$\rightarrow$ 0.123	(1,1,1,1)	Anti	No
		(1,1,1,1)	Per	Yes
		(1,1,1,1)	Anti	Yes
		(1,1,1,2)	Per	Yes
		(1,1,1,3)	Per	No
		(1,1,1,3)	Per	Yes
		(1,1,1,3)	Anti	No
		(1,1,1,3)	Anti	Yes
		(1,3,1,1)	Per	Yes

Table 3.4: Pion Propagator Tests.

with the various communication routines. This also checks that the communications routines perform as intended. One must be careful when implementing non-periodic boundary conditions on a parallel machine that the appropriate global boundaries are properly identified (see § (3.2.3)). The agreement with the analytic results for anti-periodic boundary conditions is an important confirmation that we are implementing these correctly. We also use these tests as further verification of the red-black trick from § (2.3.1).

Similar checks have been performed on larger lattices and arrays of processors (up to $12^3 \times 24$ on a [2, 2, 2, 4] processor grid) as a more realistic test of the code. All the various matrix inversion routines used have satisfied these tests.

For the Clover action in § (1.4.4), the extra term should vanish for a unit gauge configuration, which can be readily checked. A full analytic calculation with the propagator rotations is quite difficult, so we have had to consider alternative tests. Having ensured that the pion propagators for the Clover action were physically reasonable, we have compared our results with those obtained by [ELC 91] on slightly smaller lattices. Our results for the pion masses at corresponding values of κ are the same as theirs within statistical error. More rigorous tests have also been performed by directly comparing pion propagators with results from our collaborators in Southampton and Rome. Their code uses a very different inversion algorithm, Gauss-Seidel (see § (2.1.2)), is written in `FORTRAN` rather than C, and runs on a completely different computer architecture. It was written entirely independently of the Edinburgh code and has been extensively verified. Pion propagators for both the Wilson and Clover actions were computed on a test configuration — a random $4^3 \times 8$ configuration generated on the Rutherford Cray. The successful comparison of these independent calculations provides us with further confidence in both codes.

3.5.2 Performance of Matrix Inversion Routines

Due to the large amount of computer time required in a typical QCD calculation and the expense of computer hardware, it is essential that efficient use is made of the available resources. Therefore, the issue of performance is second only to that of correctness of the code.

To give some idea of how Maxwell compares to other supercomputers, we have measured its performance for a variety of different routines. This is the only way of estimating the performance attainable for lattice physics problems. The figures below are in Gigaflops, where

$$1 \text{ Gigaflops} = 10^9 \text{ floating point operations per second.}$$

These are only intended to be approximate figures for a reasonable lattice size on a large processor grid.

Peak performance:	5.1
Staple (§ (3.3.1))	0.6
$SU(3) \times$ 2-spinor:	2.4
$\ldots +$ boundary transfer:	1.4
CG – single node ($\times 64$):	1.3
Full Conjugate Gradient:	1.1

The *peak performance* is the maximum performance as determined by the 80 MHz clock speed of the i860. The highest performance we have seen for user code is about 3 Gflops, for code which does not require much data to be transferred between the i860 and the memory. As can be seen from the above table, communication of boundary data takes approximately 40% of the time for the routine

Routine	Times Called	Total Time (ms)	Mflops Achieved	Mflops Expected
scalar product	2	7	27.0	46
global summation	2	6	–	–
$y_i = y_i + (a * x_i)$	$3+2$	51	9.7	46
4-spinor $\rightarrow$ 2-spinor	4	85	9.3	17
$SU(3) \times$ 2-spinor	4	424	20.4	36
2-spinor $\rightarrow$ 4-spinor	4	88	13.4	< 17
CG Iteration		661	17.1	≈ 30

Table 3.5: Performance of Conjugate Gradient.

which is the major component of $\not{D}$. The table indicates that the maximum performance for a multi-processor physics application is around $1.0 \rightarrow 1.5$ Gflops.

To be more specific, we have also measured the timings for each of the routines which constitute an iteration of a CG or MR algorithm. Table 3.5 gives details of the corresponding performances in Megaflops per processor. These were measured on a 16^4 lattice distributed over a 2^4 hypercube of processors. It is difficult to make accurate measurements of the timings of the routines, since the the act of measurement disturbs the performance. On conventional computers, this is only a minor worry, but on complicated architectures it may be a major effect. The consistency of these measurements was checked by timing various subgroups of routines and ensuring that the total time was the same as the sum of the individual times. While most of these times were measured *in situ*, it was often helpful to cross-check timings with test programs.

The first thing to notice is that 90% of the time for a CG iteration is spent in the three routines which together constitute a $\not{D}$, and they determine the overall performance of the code. The main optimisation effort has therefore been concentrated on these routines, and the program has been structured so that, for example, the memory is allocated in the way most suitable for them. The right-hand column of maximal performance figures are taken from the original proposal document for the UKQCD project [Booth 90]. They were determined by considering the processor speed and memory bandwidth of the i860.

The overall efficiency of the code does not vary significantly when the lattice size per processor is changed. For example, we often have a $24^3 \times 48$ lattice distributed over 32 processors, which gives a local lattice size of $12^3 \times 6$. This takes around 1.7 seconds for a CG iteration which is essentially entirely accounted for by the effect of a larger local volume. An MR iteration has half the number of calls to $\not{D}$ and, for all lattice sizes, takes just over 50% of the time of a CG iteration.

For the Clover action, there are two additional routines — multiplication by

$$A = 1 - \frac{i\kappa}{2} g_0 F_{\nu\mu} \sigma_{\nu\mu}$$

and its inverse (see, e.g., § (2.3.2)). These are both local operations and therefore efficient. In fact, the multiplication by A achieves a performance of over 3 Gflops on 64 processors and is the fastest routine. However, the corresponding routine to multiply a fermion vector by A^{-1} is not yet written in assembler and is thus significantly slower. At present, an iteration for the Clover action takes around 75% longer than for Wilson though this overhead should be reduced by at least a factor of 2 when the inverse multiplication is optimised.

Generally, the main limitations to Maxwell's performance are

Memory Bandwidth — the maximum rate at which data can be transferred between the i860's registers and the main memory on the board, and

Communication Speed — the maximum rate at which data can be transferred between processor nodes.

The first of these is the reason why, e.g., the *saxpy* operation, $y_i = y_i + (a * x_i)$, and the $2 \leftrightarrow 4$-spinor routines have turned out to be somewhat slower than expected. This problem may be alleviated by careful assembler coding. The *communication speed* is probably the major limitation of all parallel computers today. Maxwell's links are fast (20 Megabits/sec) and the communications are straightforward to use, yet the main calculation routine still spends 40% of its time communicating boundary data. There are two ways in which this overhead could be reduced.

- Faster hardware links could be used, which is technically difficult, or
- Communication could be fully overlapped with calculation. The difficulty with this approach is that our hand-coded assembly routines use up so much of the memory bandwidth that the transputers cannot get a chance to start up the communications.

On the whole, the performance figures are quite encouraging. Except for one of the routines required for the Clover action, coding improvements are likely to give, at most, a few per cent improvement. The only major possible improvement would come from better inter-processor communications.

What do the Performance Figures Mean?

For the range of κ values which we use, i.e., $\kappa = 0.1510 \rightarrow 0.1530$, on the $24^3 \times 48$ lattices the matrix inversion takes between 250 MR iterations and 2500 CG iterations. This means that the time taken to calculate a single component of the Wilson quark propagator is between 5 and 70 minutes. Additionally, it takes around 8 to 10 minutes to write a component onto the disc. This makes it inefficient to use much smaller values of κ since it would then take longer to write the propagator than to calculate it. Overall, the total time taken to generate all 12 components of the Wilson quark propagator varies from 3 to 16 hours. Since that is about the maximum amount of data a single disc can hold, these times indicate how often we need to tape up the data onto Exabytes. They are therefore vital in determining a workable *modus operandi*, and indeed determining the feasibility of the project. Our timings show that operator intervention is required only a few times per day which is completely reasonable.

Chapter 4

Hadron Spectroscopy

§ (4.1) will describe the operators used to investigate the various mesons and baryons in which we are interested. It will also indicate how the corresponding correlation functions can be constructed by tying together appropriate quark propagators. The next section shows how these correlators can be used to determine the lattice masses of the particles using effective masses and fits to the timesliced propagators. This will be followed by a description of Wuppertal smearing which can be used to improve the signal.

The main section, § (4.4), presents preliminary results for the hadron mass spectrum for both the Clover and Wilson fermion actions. These results have been determined on a sample of 9 $24^3 \times 48$ configurations of gauge fields at $\beta = 6.2$. The Chapter will finish with a summary of the results so far.

4.1 Hadron Correlators

On each configuration, we use the matrix inversion algorithms of Chapter 2 to solve Eq. (2.12) for sources of each colour and spin. We can then construct the quark propagator, G, which is the correlation function

$$G^{ij}_{\alpha\beta}(\vec{x}, t; 0) = \left\langle 0 \left| \psi^i_\alpha(\vec{x}, t) \bar{\psi}^j_\beta(0) \right| 0 \right\rangle , \tag{4.1}$$

where roman letters (i, j) are colour indices, and greek letters (α, β) are spin indices. Normally we evaluate this for a number of different bare quark masses or, equivalently, values of the hopping parameter, κ. The following sections will show how these quark propagators can then be used to investigate the properties of lattice hadrons.

4.1.1 Meson Operators

Meson operators are all of the form

$$\mathcal{O}(x) = \bar{\psi}(x) \Gamma \psi(x) \tag{4.2}$$

where Γ can be any one of the 16 linearly independent γ-matrix combinations, $1, \gamma_5, \gamma_\mu, \gamma_5\gamma_\mu, \sigma_{\mu\nu}$. A generic meson propagator is shown in Fig. 4.1.

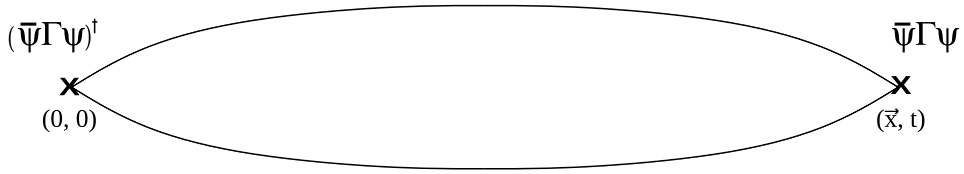


Figure 4.1: Meson propagator.

The properties of the γ-matrices under charge conjugation and the parity operation can be used to build operators with the required quantum numbers. To study any particular meson, we choose a linear combination of the meson operators with the same quantum numbers. For example, one possible pion operator is

$$\pi(x) = \bar{\psi}(x)\gamma_5\psi(x). \tag{4.3}$$

Therefore, to measure the lattice mass of the pion, we look at the correlation function,

$$C(\vec{x},t) = \left\langle 0 \left| \pi(\vec{x},t)\, \pi^\dagger(0) \right| 0 \right\rangle . \tag{4.4}$$

In Minkowski space

$$\begin{aligned} \pi^\dagger &= \left((\psi^\dagger\gamma_0)\gamma_5\psi)\right)^\dagger \\ &= \psi^\dagger\gamma_5\gamma_0\psi \\ &= -\bar{\psi}\gamma_5\psi, \end{aligned}$$

from the hermiticity and commutation properties of the γ matrices. This gives

$$C(x) = -\left\langle 0 \left| \bar{\psi}^i_\alpha(x)\,(\gamma_5)_{\alpha\beta}\,\psi^i_\beta(x)\,\bar{\psi}^j_\gamma(0)\,(\gamma_5)_{\gamma\delta}\,\psi^j_\delta(0) \right| 0 \right\rangle . \tag{4.5}$$

Wick contraction gives

$$\begin{aligned} C(x) &= -(\gamma_5)_{\alpha\beta}\,(\gamma_5)_{\gamma\delta}\,\left\langle 0 \left| \psi^i_\beta(x)\,\bar{\psi}^j_\gamma(0) \right| 0 \right\rangle \left\langle 0 \left| \bar{\psi}^i_\alpha(x)\,\psi^j_\delta(0) \right| 0 \right\rangle , \qquad (4.6) \\ &= (\gamma_5)_{\alpha\beta}\,(\gamma_5)_{\gamma\delta}\,G^{ij}_{\beta\gamma}(x;0)\,G^{ji}_{\delta\alpha}(0;x), \qquad (4.7) \end{aligned}$$

using Eq. (4.1). Using the lattice Dirac equation gives the result

$$G^{ji}_{\delta\alpha}(0;x) = (\gamma_5)_{\delta\rho}\,G^{\dagger\,ji}_{\rho\sigma}(0;x)\,(\gamma_5)_{\sigma\alpha}. \tag{4.8}$$

Therefore

$$\begin{aligned} C(x) &= -(\gamma_5)_{\alpha\beta}\,G^{ij}_{\beta\gamma}(x;0)\,(\gamma_5)_{\gamma\delta}\,(\gamma_5)_{\delta\rho}\,G^{\dagger\,ji}_{\rho\sigma}(0;x)\,(\gamma_5)_{\sigma\alpha} \\ &= \mathrm{Tr}\left[\gamma_5 G(x;0)\gamma_5\gamma_5 G^\dagger(0;x)\gamma_5\right] \\ &= \mathrm{Tr}\left[G(x;0)G^\dagger(0;x)\right] \\ &= \left|G^{ij}_{\alpha\beta}(x;0)\right|^2 \qquad (4.9) \end{aligned}$$

using ${\gamma_5}^2 = 1$. The pion correlator is the simplest to measure since it is simply the sum over colour and spin of the absolute value of the quark propagator squared. We calculate the quark propagators, G, using an analytic continuation to Euclidean space.

Throughout this Chapter, the only mesons which will be studied are the pion and the rho. Since the ρ is a vector meson, the corresponding operator must have a free Lorentz index. The operators we use are listed in Table 4.1.

Operator	J^{PC}	Particle
$\bar{\psi}\gamma_5\psi$	0^{-+}	π
$\sum_{i=1}^{3}\bar{\psi}\gamma_i\psi$	1^{--}	ρ

Table 4.1: The meson operators and their particle assignments.

4.1.2 Baryon Operators

Since baryons contain 3 valence quarks, a generic operator for a baryon has the form

$$\mathcal{O}_B(x) \sim \epsilon_{ijk}(\psi^i(x)\Gamma\psi^j(x))\psi^k(x) \tag{4.10}$$

with the appropriate symmetrisations. For the isospin $\frac{1}{2}$ particles, we need Γ to be antisymmetric while for the $I = \frac{3}{2}$ particles it should be symmetric.

Operator	$I(J)$	Particle
$(uC\gamma_5 d)u$	$1/2(1/2)$	Nucleon (N)
$(uC\gamma_i u)u$	$3/2(3/2)$	$\Delta(3/2)$

Table 4.2: The baryon operators and their particle assignments.

Table 4.2 shows the operators we use to investigate the nucleon and Δ. The matrix C is the charge conjugation operator while u and d represent different flavours of quark field. To construct a $J = \frac{3}{2}$ operator requires a free spin index. The operator associated with the Δ in Table 4.2 also has contributions of the form $p_i N$ so we have to project out the Δ state by taking the appropriate combinations of components.

The parity operator is γ_0 and the parity eigenstates are $(1 \pm \gamma_0)$. In the representation we use, these project out the upper two spinor components and the lower two spinor components, respectively. The upper two spinor components represent the baryon propagating forward in time, while the lower two spinor components can be used to investigate the backwards propagating baryon.

4.2 Correlation Functions and Masses

Consider the correlation function of an operator, $\mathcal{O}$, with the same quantum numbers as one of the light hadrons, i.e.,

$$C(\vec{x}, t) = \left\langle 0 \left| \mathcal{O}(\vec{x}, t)\, \mathcal{O}^{\dagger}(0) \right| 0 \right\rangle . \tag{4.11}$$

If we insert a complete set of states labelled by spatial momentum, $\vec{p}$, and particle, n, we get

$$C(\vec{x}, t) = \sum_n \int \frac{\mathrm{d}^3\vec{p}}{16\pi^3 E(\vec{p})} \left\langle 0 \left| \mathcal{O}(\vec{x}, t) \right| n, \vec{p} \right\rangle \left\langle n, \vec{p} \left| \mathcal{O}^{\dagger}(0) \right| 0 \right\rangle . \tag{4.12}$$

With the Schrödinger representation of $\mathcal{O}$, we use the operator P to transform to momentum space to get

$$\begin{aligned} C(\vec{x}, t) &\propto \sum_n \int \mathrm{d}^3\vec{p} \left\langle 0 \left| P\mathcal{O}(0)P^{-1} \right| n, \vec{p} \right\rangle \left\langle n, \vec{p} \left| \mathcal{O}^{\dagger}(0) \right| 0 \right\rangle & (4.13) \\ &\propto \sum_n \int \mathrm{d}^3\vec{p} \left\langle 0 \left| \mathcal{O}(0) \right| n, \vec{p} \right\rangle \left\langle n, \vec{p} \left| \mathcal{O}^{\dagger}(0) \right| 0 \right\rangle e^{-ip\cdot x}. & (4.14) \end{aligned}$$

Summation over the spatial volume of each timeslice projects out the states with zero spatial momentum, giving

$$C(t) = \sum_{\vec{x}} C(\vec{x}, t) = \sum_n a_n e^{i m_n t}, \tag{4.15}$$

where the coefficient, a_n, is related to the overlap of $\mathcal{O}$ with the particle n. Working with a periodic, Euclidean lattice, this becomes

$$C(t) = \sum_n \left(F_n e^{-m_n t} + B_n e^{-m_n (L_t - t)} \right) \tag{4.16}$$

where the second term is from the backward propagating particle. As $t \to \infty$ and $(L_t - t) \to \infty$, we are left with only the contribution from the lightest state which has non-zero overlap with the operator $\mathcal{O}$, i.e.,

$$C(t) = F e^{-m_0 t} + B e^{-m_0 (L_t - t)}. \tag{4.17}$$

4.2.1 Extracting Masses from Propagators

For each κ value, we start with 800 Mbytes for each configuration, obtained from the required 12 fermion matrix inversions. We reduce this by tying propagators together and summing over timeslices, to a set of 48 numbers for each operator. We now want to reduce this information further by using each timesliced propagator to extract a single number — the mass of the lightest particle in that channel.

Effective Masses

The quickest method to get an estimate of this mass is by using the values of $C(t)$ on neighbouring timeslices to define an *effective mass* on each timeslice. If L_t is large, we can ignore the backward propagating state, i.e., set $B_n = 0$ in Eq. (4.16). Then

$$m^{\text{eff}}(t) = \log \left(\frac{C(t)}{C(t+1)} \right). \tag{4.18}$$

Near the origin, this will not give a good estimate of the mass of the lightest particle as the signal will be contaminated by higher mass states. However, for larger values of t, the lightest state will dominate and so we hope that m^{eff} will be constant over a range of timeslices.

For many operators the amplitude of the forward and backward propagating states will be the same, i.e., $F_n = B_n$ and, for t large, Eq. (4.17) becomes

$$C(t) = A \cosh m \left(t - \frac{L_t}{2}\right). \tag{4.19}$$

We can use this to define a more precise m^{eff}, using 2 or even 3 timeslices (see [Bowler 83]), but, with low statistics, there is little to gain over Eq. (4.18).

Fig. 4.2 shows a typical effective mass plot for the rho meson. Errors in the effective mass are estimated from the jackknife method over the configurations. This clearly shows that, at timeslices near to the origin, m^{eff} does not give an estimate of the ρ mass due to contributions from other particles in this channel with higher mass. However, we can see that from timeslice 10, there is a plateau in the effective mass and thus a signal for the ρ. After timeslice 18, the signal has disappeared into noise and so we are constrained to estimate the mass in this region. Averaging the effective mass over timeslices $[10, 18]$ gives the same value as averaging over $[11, 17]$ well within errors (see Table 4.3).

Hyperbolic Fitting

In general, we can get a more reliable estimate of the mass by fitting the propagator, $C(t)$, to the appropriate form. It might seem possible to include the contribution from a higher mass state in this fit, e.g., we could fit the ρ propagator to

$$C(t) = A_0 \cosh m_\rho \left(t - \frac{L_t}{2}\right) + A_1 \cosh m_1 \left(t - \frac{L_t}{2}\right).$$

However, fits to a sum of 2 hyperbolic or exponential forms are notoriously unstable. It is better to fit over a region in t where the only significant contribution is from the lightest particle, i.e., where the slope of $\log C(t)$ v. t is constant. The best way of choosing this region is from the plateau in the corresponding effective mass plot. Inconsistency between the effective mass data and the fitted mass is

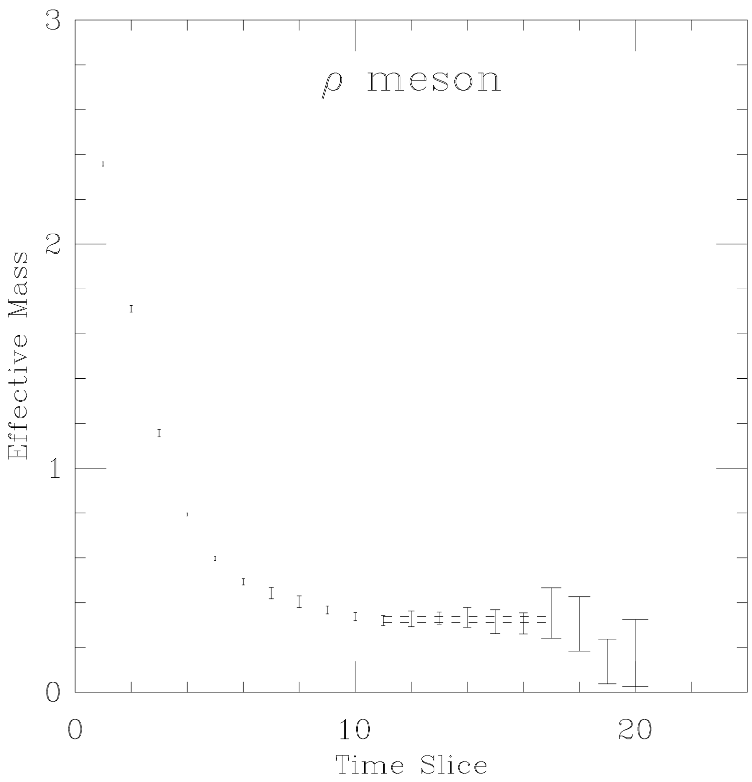


Figure 4.2: The effective mass plot of the rho meson for the Clover action at $\kappa = 0.14262$. This is from 9 configurations and has been averaged over the forward and backward propagating particles. The dotted lines show the error in the averaged effective mass over the region $[11, 17]$.

an indication of serious problems with the data set.

The standard way to fit data to a functional form, such as Eq. (4.19), is to minimise some *goodness-of-fit* measure as a function of the parameters $\vec{a} = (A, m)$. To fit over a region $[t_{\mathrm{min}}, t_{\mathrm{max}}]$, the normal χ^2 is

$$\chi^2(\vec{a}) = \sum_{t=t_{\mathrm{min}}}^{t_{\mathrm{max}}} \frac{(C(t) - C(t;\vec{a}))^2}{\sigma_t^2} \tag{4.20}$$

where $C(t)$ is the propagator data at t with error σ_t and $C(t;\vec{a})$ is the theoretical value obtained from the parameters, $\vec{a}$. However, this ignores correlations between successive timeslices which are typically large and leads to an estimate of χ^2 which is unrealistically small. To obtain a more reliable estimate of the errors in the best fit parameters, we need to take the correlations into account using

$$\chi^2_{\text{corr}}(\vec{a}) = \sum_{t_i}\sum_{t_j} \text{Cov}(t_i,t_j)^{-1}(C(t_i) - C(t_i;\vec{a}))(C(t_j) - C(t_j;\vec{a})). \tag{4.21}$$

The data covariance matrix is

$$\text{Cov}(t_i,t_j) = \frac{1}{N(N-1)}\sum_{k=1}^{N}(C_k(t_i) - \langle C(t_i)\rangle)(C_k(t_j) - \langle C(t_j)\rangle)$$

where the sum is over the N configurations and $C_k(t_i)$ is the value of the correlator on timeslice t_i from configuration k. If the correlations are negligible, Eq. (4.21) reduces to Eq. (4.20).

Region		Rho Mass	
t_{min}	t_{max}	m_ρ^{eff}	m_ρ^{fit}
10	16	0.329(11)	0.329(18)
10	17	0.329(11)	0.329(17)
10	18	0.329(11)	0.331(17)
11	16	0.324(14)	0.316(22)
11	17	0.324(14)	0.317(22)
11	18	0.324(13)	0.316(21)
12	16	0.326(17)	0.321(25)
12	17	0.327(17)	0.322(26)
12	18	0.326(17)	0.320(25)

Table 4.3: Comparison of m_ρ obtained from the effective mass plots and the fitting routine for a number of different regions. This data is for the Clover action at $\kappa = 0.14262$.

Table 4.3 contains various estimates of m_ρ obtained both from the effective mass data and from correlated fits to Eq. (4.19). The estimates using m^{eff} are obtained from a weighted average over the region and the errors quoted ignore correlations between timeslices. The agreement between m^{eff} and m^{fit} is very good for each region. This indicates that neither method has serious systematic problems for this data. There is no systematic decrease of m as t_{min} increases and essentially no dependence on t_{max}. This clearly shows that there is a region in t in which the signal from the lightest state dominates the propagator.

4.3 Smearing

Recently, there has been an increasing amount of evidence that large improvements in the efficiency of lattice gauge measurements can be made by using non-local or *smeared* operators (see e.g., [Michael 89, Güsken 90]). Using larger operators appears to increase the overlap with the lightest states. For propagator calculations, we can use a source, η, which is finite over some non-zero volume. Alternatively, we can smear the solution, ψ, which we call smearing the sink, or, indeed, smear at both the source and the sink. Since we determine the particle masses from the decay of the timesliced correlators, as in Eq. (4.19), we only smear over the spatial volume. Various forms of smearing have been proposed, such as replacing the point source with a 3D cube over some or all of the spatial volume. This is normally performed after gauge-fixing and is then a non gauge-invariant operation. Throughout the rest of this Chapter, we shall only be concerned with *Wuppertal* smearing which is physically motivated and gauge-invariant.

4.3.1 Wuppertal Smearing

To improve the signal for the lightest state, [Güsken 90] investigated smearing techniques using wavefunctions which are smooth and vanish at ∞. Wuppertal smearing uses the 3D scalar propagator as the smearing wavefunction to produce an extended source without the necessity of gauge-fixing. This is rotationally invariant and has a smoothly decaying exponential form. We thus need to solve the 3D scalar Klein-Gordon equation on each timeslice of the lattice. To produce a smeared source, this only needs to be performed on the single timeslice where the source is non-zero. The η in Eq. (2.12) is replaced by η_{sm} defined by

$$(1-\kappa_{sc}\nabla^2_{x,y})(\eta_{sm})^i_\alpha(y) = \delta^i_\alpha(x) \tag{4.22}$$

where

$$\nabla^2_{x,y} = \sum_{\mu=1}^{3}\left(U_\mu(x)\delta_{x+\hat{\mu},y} + U^\dagger_\mu(x-\hat{\mu})\delta_{x-\hat{\mu},y}\right).$$

Eq. (4.22) can be solved using the red-black over-relaxed MR algorithm described in Chapter 2. The amount of smearing is controlled by a single free parameter, κ_{sc}, which is related to the mass of the scalar particle. The value of this parameter is chosen so that a smeared point source has a r.m.s. radius of a few lattice sites. We normally set $\kappa_{sc} = 0.180$.

Similarly, we can smear at the sink by solving

$$(1-\kappa_{sc}\nabla^2)\psi_{sm} = \psi,$$

where ψ is the solution to Eq. (2.12). Propagators with a local source and sink are labelled with LL, those with a local source and smeared sink are LS and the other combinations are labelled in the obvious fashion. In practice, we normally smear at the sink as this has the advantage that it can be undone with a single multiplication by $(1-\kappa_{sc}\nabla^2)$ and then, if necessary, re-smeared with a different κ'_{sc}. Since we can easily recover the LL propagator from the LS propagator, this allows us to write only the LS propagator from Maxwell. The rate at which data can be written from Maxwell is a bottleneck and so this is one of the major advantages of Wuppertal smearing. We have checked that the convergence criterion for the smearing is sufficiently tight by comparing the hadron correlators obtained using LL quark propagators with those from LS propagators which have been unsmeared as described above. Typically, the hadron correlators, even at large distances from the origin, change by no more than 1 part in 10^4 or 10^5.

4.3.2 Smeared and Unsmeared Propagators

To see what has been gained from smearing we should look at effective mass plots obtained from LL and LS propagators. Fig. 4.3 shows plots for the ρ meson at a light quark mass.

There are a number of striking differences between the 2 graphs. Firstly, we can see that m^{eff}_{LS} is much smaller than m^{eff}_{LL} near the origin. The plateau for the LS graph begins around timeslice 9, while for LL this does not occur before timeslice 10 or 11 though both plateau at a mass of about 0.3. It is also noticeable that the LL data appears more correlated then the LS data and is less noisy at large t. This is a reasonably typical case for our smearing where we gain 1–2 timeslices near the origin due to a better overlap with the lightest state and then sometimes lose a timeslice further out due to increased statistical noise.

Comparing Table 4.5 and Table 4.6, the fit to the LL propagator on timeslices $[11, 17]$ gives $m_\rho = 0.30(3)$ while the LS fit from $[10, 17]$ gives $m_\rho = 0.29(3)$. Considering that these results are from light quark mass and how different the graphs appear, the agreement between the two fits is encouraging. It is interesting to note that the off-diagonal elements in the covariance matrix, Cov, are significantly larger for the LL data, which is what we expected from the correlations in the m^{eff} plots. The similarity between the various estimates for the mass allow us to conclude that we are indeed seeing the lightest excitation in both the LL and LS data. Further discussion of the smearing can be found in § (4.5.1).

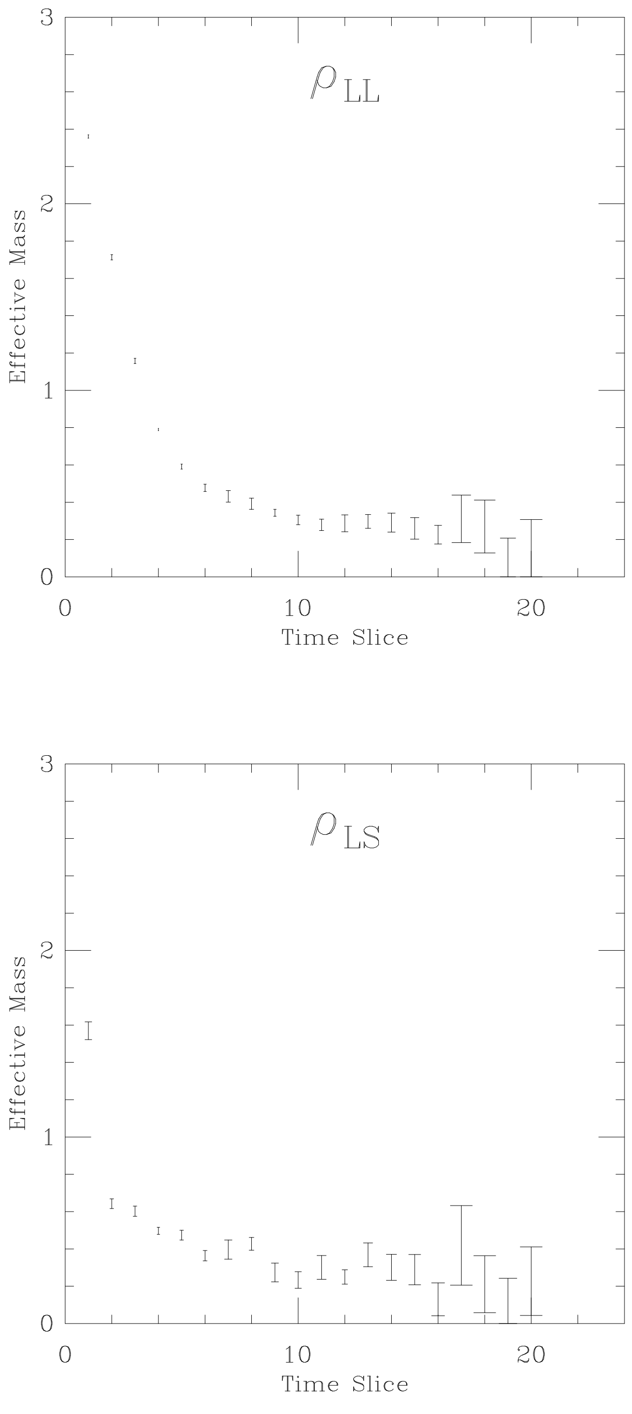


Figure 4.3: Comparison of the ρ meson effective mass plots for LL propagators (top) and LS propagators. These are from 9 configurations using the Clover action at $\kappa = 0.14280$.

4.4 Results

All the results in this section are from $24^3 \times 48$ lattices at $\beta = 6.2$. The lattice size was determined by the amount of memory required for a propagator inversion and by the requirement for a long time direction so that the asymptotic masses can be more clearly seen. At this value of β, we should be approaching the scaling region yet not be too strongly finite size affected on 24^3 lattices at the values of quark mass typically used. The gauge configurations were generated using compound sweeps consisting of 5 OR sweeps followed by a heatbath sweep to ensure ergodicity. Further details can be found in § (1.4.5). We allow 2800 sweeps for equilibration and perform this analysis on 9 configurations separated by 400 sweeps.

For the Wilson action, we have calculated propagators at $\kappa = 0.1510$, 0.1520, 0.1523, 0.1526 and 0.1529. For the Clover action we chose $\kappa = 0.14226$, 0.14244, 0.14262 and 0.14280 by roughly matching the lattice pion masses, determined from a single configuration, with the lighter four pion masses computed in the Wilson case. For the Clover action, in addition to results based on the standard LL quark propagators, we also present results for propagators smeared at the sink, the LS propagators, as described in § (4.3).

Some of the Wilson propagators were calculated using the Conjugate Gradient algorithm from § (2.2.6) before we later switched to an Overrelaxed Minimal Residual algorithm of the form described in § (2.2.3). We found that optimising the overrelaxation parameter gave an algorithm which was a factor of 2 more efficient than CG.

The hadron correlators were calculated on a number of different computers including Maxwell, a DAP[1] and various Sun workstations. As a bug check, we ensured

[1]The DAP is a parallel computer manufactured by AMT consisting of a 64^2 array of bit serial

that the correlators were independent of the machine used to measure them.

4.4.1 Hadron Masses

The particles we have investigated are the pion (π), rho meson (ρ), nucleon (N) and delta (Δ) using the operators listed in Table 4.1 and Table 4.2. The masses were calculated by fitting the timesliced correlators to the appropriate functional form. For the mesons, this is a cosh since the forwards and backwards propagating particles are identical. However, for the baryons we fit to a decaying exponential as we take only the upper or only the lower spinor components which projects out different parity states propagating forwards and backwards states in time (see § (4.1.2)). To increase our statistics we treat the particle travelling in the negative t direction as independent in the ensemble. The correlations in the data when analysed this way should be taken into account by the correlated χ^2 fits. For the baryons, we take the backwards propagating state of the opposite pair of spinor components for the reason given above.

The choice of region over which to fit the propagator is the trickiest part of this exercise! As we fit to only the lightest particle, as in Eq. (4.17), we must choose a region where only this state contributes significantly. This can be roughly determined using the effective mass plot, but it is vital to check that the fit is stable when the fit region is perturbed. In particular, to minimise systematic effects from heavier states, it is important that fitting to $[t_{\rm min}+1, t_{\rm max}]$ agrees within statistical error with the fit from $[t_{\rm min}, t_{\rm max}]$. Nevertheless, this choice is still a source of systematic error.

Initially, our fits were performed by minimising the naïve χ^2 from Eq. (4.20). However, the results presented in Table 4.4, Table 4.5 and Table 4.6, are from

processors.

correlated fits using Eq. (4.21). The correlated χ^2 gives a useful measure of the goodness of the fit and better estimates of the errors in the fit parameters, though the two independent fitting programs generally give the same masses within statistical error. In the tables, as well as presenting the masses from the correlated fits, we also list the region over which each fit was performed, $[t_{\rm min}, t_{\rm max}]$, and the corresponding χ^2/n where the number of degrees of freedom,

$$n = \text{ number of timeslices fitted } - \text{ number of fit parameters.}$$

Discussion of Mass Fits

The results for the Wilson action are in Table 4.4. For each particle, a suitable region to fit over was found by investigating a number of regions for one of the larger κ values and then checking that this was also appropriate for the other values. This was a practical attempt to reduce the arbitrariness of the region given the low statistics. It was also possible to find fit regions for the different particles that largely overlapped, which should reduce errors in mass ratios and differences. The major quantitative measure of the goodness of the fits is the value of χ^2/n. If χ^2 is very different from n, it is indication that there is a problem with the fit. This may be due to the data not being Gaussian distributed, or that the jackknife method is not a good estimate of the true errors in the propagator. For the Wilson meson propagators we get values of χ^2/n between 0.7 and 1.6 with small errors on the masses. These are signs that the signals are good. Even for the baryon fits, $(\chi^2/n) < 3$, though the errors are increasing. It was quite difficult to find a region with a stable fit for the nucleon so the systematic errors are probably not negligible. Nevertheless, there do appear to be reasonable signals for the baryon operators used.

The fits for the Clover LL propagators were taken over the same region as for the corresponding Wilson LL fits. The stability of the fit against small changes in the

κ	m_π	$[t_{\mathrm{min}}, t_{\mathrm{max}}]$	χ^2/n
0.1510	0.294(3)	[12,18]	5.6/5
0.1520	0.218(3)	[12,18]	3.4/5
0.1523	0.193(4)	[12,18]	3.9/5
0.1526	0.166(4)	[12,18]	4.7/5
0.1529	0.132(4)	[12,18]	7.8/5
κ	m_ρ	$[t_{\mathrm{min}}, t_{\mathrm{max}}]$	χ^2/n
0.1510	0.386(6)	[11,17]	7.0/5
0.1520	0.352(9)	[11,17]	4.3/5
0.1523	0.341(10)	[11,17]	3.5/5
0.1526	0.323(12)	[11,17]	5.3/5
0.1529	0.314(19)	[11,17]	7.2/5
κ	m_N	$[t_{\mathrm{min}}, t_{\mathrm{max}}]$	χ^2/n
0.1510	0.604(9)	[12,18]	11/5
0.1520	0.516(12)	[12,18]	12/5
0.1523	0.480(15)	[12,18]	11/5
0.1526	0.432(23)	[12,18]	5.8/5
0.1529	0.375(35)	[12,18]	2.0/5
κ	m_Δ	$[t_{\mathrm{min}}, t_{\mathrm{max}}]$	χ^2/n
0.1510	0.640(10)	[12,17]	5.5/4
0.1520	0.571(12)	[12,17]	4.4/4
0.1523	0.554(15)	[12,17]	5.1/4
0.1526	0.537(19)	[12,17]	7.4/4
0.1529	0.524(35)	[12,17]	11/4

Table 4.4: Hadron masses in lattice units for the Wilson action from correlated fits to the LL propagators in the region $[t_{\mathrm{min}}, t_{\mathrm{max}}]$. The righthand column, χ^2/n, gives a measure of the *goodness-of-fit* for the correlated fit. A good fit should have $\chi^2 \approx n$.

κ	m_π	$[t_{\rm min}, t_{\rm max}]$	χ^2/n
0.14226	0.218(4)	[12,18]	5.1/5
0.14244	0.197(4)	[12,18]	4.8/5
0.14262	0.174(4)	[12,18]	3.8/5
0.14280	0.147(5)	[12,18]	3.7/5
κ	m_ρ	$[t_{\rm min}, t_{\rm max}]$	χ^2/n
0.14226	0.345(11)	[11,17]	0.7/5
0.14244	0.330(14)	[11,17]	0.9/5
0.14262	0.317(22)	[11,17]	1.1/5
0.14280	0.302(34)	[11,17]	3.3/5
κ	m_N	$[t_{\rm min}, t_{\rm max}]$	χ^2/n
0.14226	0.504(13)	[12,18]	5.6/5
0.14244	0.465(19)	[12,18]	2.7/5
0.14262	0.414(30)	[12,18]	0.7/5
0.14280	0.346(40)	[12,18]	1.1/5
κ	m_Δ	$[t_{\rm min}, t_{\rm max}]$	χ^2/n
0.14226	0.591(18)	[12,17]	3.7/4
0.14244	0.572(22)	[12,17]	5.1/4
0.14262	0.544(30)	[12,17]	7.5/4
0.14280	0.503(46)	[12,17]	13/4

Table 4.5: Hadron masses in lattice units for the Clover action from correlated fits to the LL propagators.

κ	m_π	$[t_{\min}, t_{\max}]$	χ^2/n
0.14226	0.218(6)	[10,16]	2.5/5
0.14244	0.197(6)	[10,16]	2.2/5
0.14262	0.174(7)	[10,16]	1.6/5
0.14280	0.147(7)	[10,16]	1.0/5
κ	m_ρ	$[t_{\min}, t_{\max}]$	χ^2/n
0.14226	0.345(10)	[10,17]	7.7/6
0.14244	0.325(14)	[10,17]	9.7/6
0.14262	0.311(23)	[10,17]	12/6
0.14280	0.289(33)	[10,17]	12/6
κ	m_N	$[t_{\min}, t_{\max}]$	χ^2/n
0.14226	0.476(21)	[9,15]	8.3/5
0.14244	0.438(26)	[9,15]	12/5
0.14262	0.421(36)	[9,15]	12/5
0.14280	0.366(28)	[9,15]	8.2/5
κ	m_Δ	$[t_{\min}, t_{\max}]$	χ^2/n
0.14226	0.582(23)	[10,15]	1.9/4
0.14244	0.553(25)	[10,15]	4.6/4
0.14262	0.520(29)	[10,15]	6.8/4
0.14280	0.492(42)	[10,15]	6.4/4

Table 4.6: Hadron masses in lattice units for the Clover action from correlated fits to the LS propagators.

region of fit was again carefully investigated. The only fits where the values of χ^2/n were significantly larger than 1 are for the Δ at high κ. Indeed, some of the ρ and nucleon fits have a suspiciously low χ^2, which may suggest that the errors in the propagator are overestimated. As for the Wilson case, it was difficult to find a suitable region over which to fit the nucleon.

Comparing Table 4.5 and Table 4.6, we can see that the stable fit region for the LS propagators is typically 2 timeslices nearer the origin. The fits for the π give the same mass estimates as the LL fits, though with slightly larger errors. For the ρ meson, the values of χ^2 are somewhat large but not worse than 2 per degree of freedom. The masses are almost identical to the LL case, though the discrepancies increase at higher κ. For the baryons, the comparisons are a little worse. The values of χ^2/n for the LS nucleon fits are larger but probably more believable. The difference in nucleon mass between the LL and LS fits is about 1 σ from zero. For the Δ, the deviation is less significant within the large errors.

Overall, we have usually managed to find fit regions for the different particles, actions and κ values which are similar. This should reduces errors in the comparison of different particles and systematic errors due to the subjective choice of region. There is reasonable agreement within errors between the various masses for the Clover LL and LS propagators. With 9 configurations, the major difficulty appears to be in getting a good signal for the nucleon. In the following sections, we compare the above values for particle masses with various theoretical and experimental expectations.

4.4.2 Determining κ_c from the Pion Mass

We assume that the pion is the Goldstone boson of spontaneously broken chiral symmetry and acquires a mass only as a result of small quark mass terms in the action for QCD (see [Weingarten 83]). This implies that

$$m_\pi^2 \propto m_q$$

for small values of the bare quark mass, m_q. From the definition of κ (Eq. (1.47)), we have $m_q \sim 1/(2\kappa)$ and so, defining κ_c to be the critical value where $m_q = 0$, gives

$$m_q = \left(\frac{1}{2\kappa} - \frac{1}{2\kappa_c}\right).$$

We can thus use the values of the pion mass calculated above to fit a straight line to

$$m_\pi^2 = a + b\left(\frac{1}{2\kappa}\right) \tag{4.23}$$

and determine κ_c from the intercept on the $\frac{1}{2\kappa}$ axis.

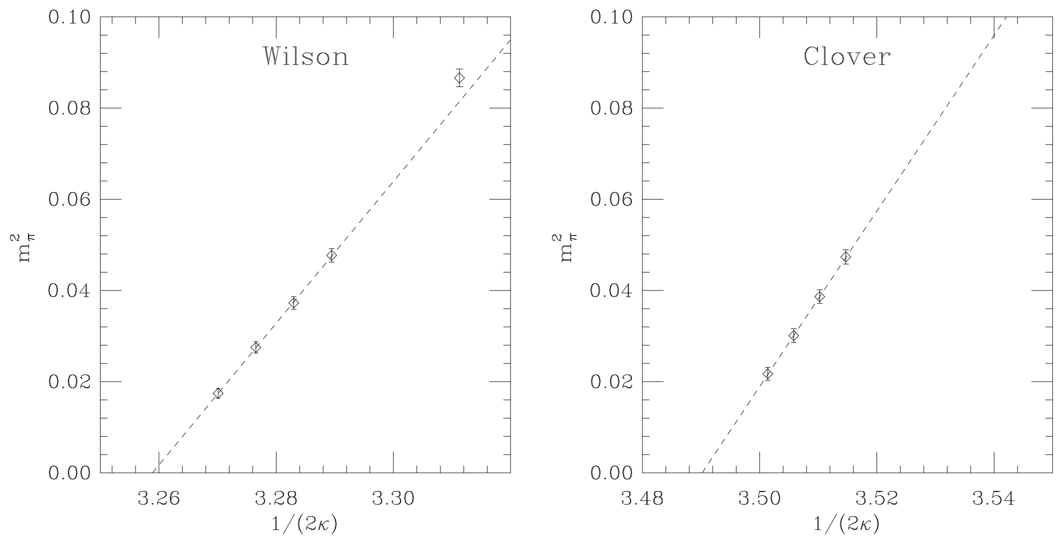


Figure 4.4: Determination of κ_c from m_π^2 for the Wilson (left) and Clover actions.

A straight line fitted to the 4 lightest points for the Wilson data in Fig. 4.4 goes right through these points while even the heaviest pion lies within 2-σ of the line. Indeed, it is possible to include this point in the fit without changing the result within errors or increasing χ^2 significantly. For comparison with the Clover data, we ignore the heaviest point and get

$$\kappa_c^W = 0.15343(5)$$

where the error quoted is purely statistical. It is probably slightly underestimated as it ignores any correlations between the different κ values. The masses for the Clover action from the LL and LS propagators are essentially identical and so we only plot the LL results. The data is also linear to a high degree of accuracy and the intercept gives

$$\kappa_c^C = 0.14326(6),$$

which agrees with the result from the LS data that

$$\kappa_c^C = 0.14326(9).$$

Estimating κ_c using m_π obtained from uncorrelated fits to the propagator gives the same central values while the jackknife error estimates are less than a factor of 2 larger. The above values for both actions are also in close agreement with [ELC 91].

The two graphs in Fig. 4.4 are plotted on similar scales and we can see that the gradient of the Clover graph is about 25% larger than the Wilson graph. Changing κ^W by a given amount produces a smaller change in physical quantities, such as m_π, than changing κ^C by the same amount. This indicates that for the two actions, the bare lattice quark mass requires different renormalisations to be related to a 'physical' quark mass.

4.4.3 Hadron Masses at Different Quark Masses

The graphs in Fig. 4.5 and Fig. 4.6 show the masses of the various hadrons for each κ against the corresponding m_π^2. § (4.4.2) shows that this is proportional to the bare m_q but, as the constant of proportionality is different for the two actions, m_π^2 is more useful for purposes of comparison.

Within the errors, the graphs are quite similar. For example, in all 3 cases, the

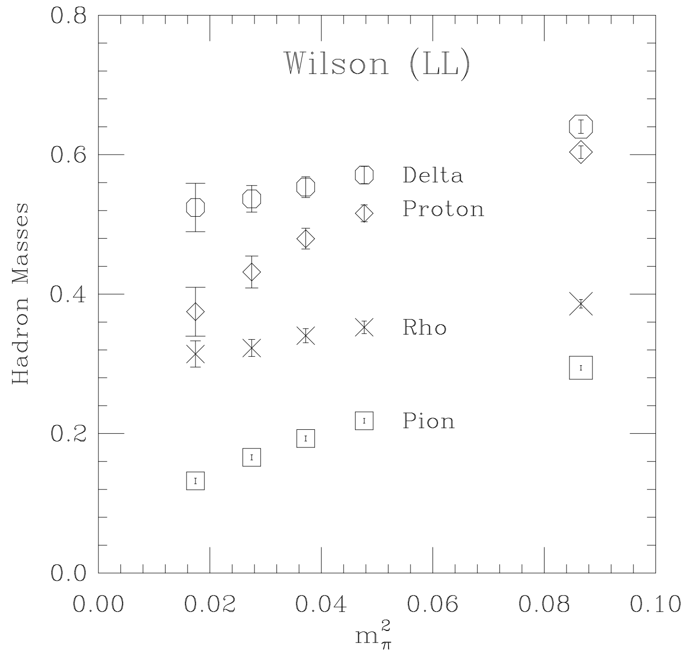


Figure 4.5: Hadron masses against m_π^2 for the Wilson action. The error in m_π^2 is less than the width of the symbols.

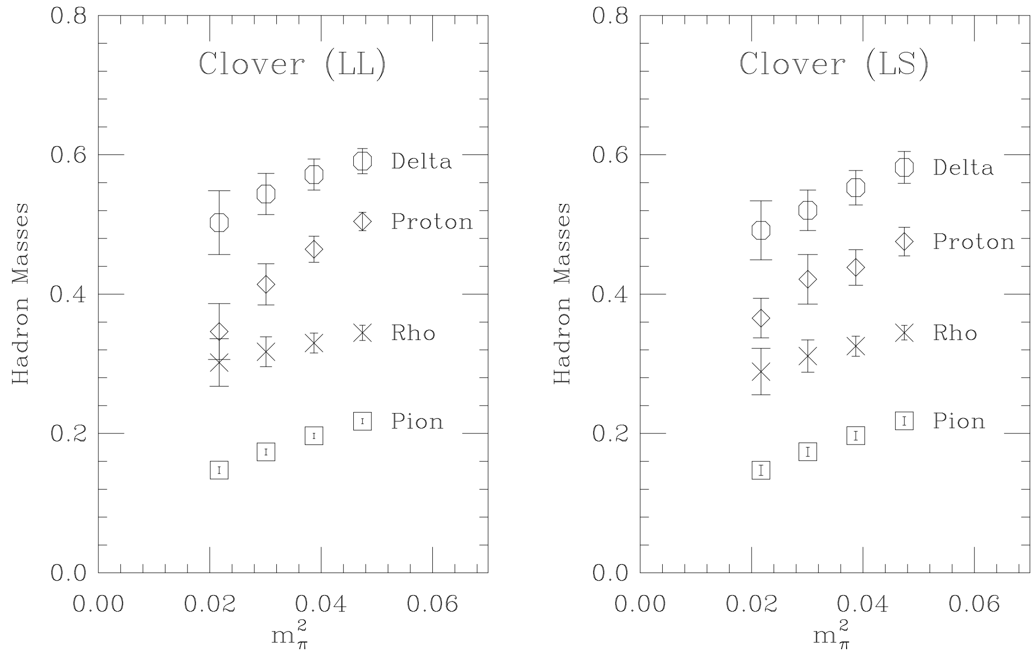


Figure 4.6: Hadron masses against m_π^2 for the Clover action using LL propagators (left) and LS propagators.

ρ meson has reasonably small errors which increase near the critical point. Additionally, excluding the heaviest nucleon point for the Wilson action, the trends in the masses of the ρ, nucleon and Δ are all consistent with linear. For the Wilson action, the N-Δ splitting is small for heavy quark masses and, as expected, increases as m_q decreases. However, the slope of the nucleon mass seems to be too large at small m_q, in that the extrapolation to $m_q = 0$ appears to give $m_N < m_\rho$. The picture is very similar for the Clover (LL) action though the errors are even larger. This problem is probably related to the difficulty of getting a good fit for the nucleon. This hypothesis is supported by the fact that the major difference between the two Clover graphs in Fig. 4.6 is in the nucleon signal.

4.4.4 Determining the Lattice Spacing

We can use the linearity of m_ρ against m_π^2 to extrapolate to the critical point. This can be related to the physical value to give an estimate of the inverse lattice spacing and, hence, to set the scale of the lattice simulations.

Action	Lattice ρ mass	Inverse lattice spacing, a^{-1} (GeV)
Wilson (LL)	0.291(2)	2.64(2)
Clover (LL)	0.267(4)	2.89(4)
Clover (LS)	0.244(6)	3.16(8)

Table 4.7: Lattice value of m_ρ extrapolated to $m_q = 0$ for the different actions. The righthand column gives the corresponding inverse lattice spacing using the physical value, $m_\rho = 770$ MeV.

Table 4.7 shows the results obtained using the 4 lightest ρ masses for each action. The errors quoted are purely statistical and ignore correlations between the data.

The discrepancy of the results for the Clover action using LL and LS propagators is a better indication of the true errors. An estimate for the inverse lattice spacing has also been made on these configurations from the string tension using blocking (see [UKQCD 91]). This gives $a^{-1} = 2.70(3)$ GeV which is in agreement with previous work on 20^4 lattices at the same β value by [Michael 89, Perantonis 90]. The results in Table 4.7 are quite similar, indicating that, for these quantities, we are reasonably near the scaling region, though it appears that the Clover action may give a somewhat larger a^{-1}. However, the disagreement between the LL and LS values makes it difficult to judge the significance of this. Wherever needed, we will regard the inverse lattice spacing from the string tension to be the canonical estimate as it is independent of the fermionic action

4.4.5 Meson Mass Splitting

There is an interesting phenomenological relationship between the masses of vector and pseudoscalar mesons with the same valence quark content. It appears that the quantity $m_V^2 - m_{PS}^2$ is almost independent of the quark mass. For example, for the π-ρ system

$$m_\rho^2 - m_\pi^2 = 0.57 \text{ GeV}^2,$$

while for the K-$K^\star$ system the corresponding quantity is 0.55 GeV^2. This is true even for D_s-$D_s^\star$ where the masses of the mesons are about 2 GeV.

We can use this result as a test of the lattice simulations. In Fig. 4.7 and Fig. 4.8 we have plotted the meson mass splittings for different κ values against m_π^2 which is proportional to the quark mass. From Fig. 4.8 we can see that the Clover LL values of $m_\rho^2 - m_\pi^2$ are independent of κ. Well within errors, the Clover LS data is also linear and in close agreement with the LL results. The 4 equivalent Wilson points in Fig. 4.7 have smaller error bars and are also consistent with a constant. The results from the European Lattice Collaboration, [ELC 91], using 20^4 lattices

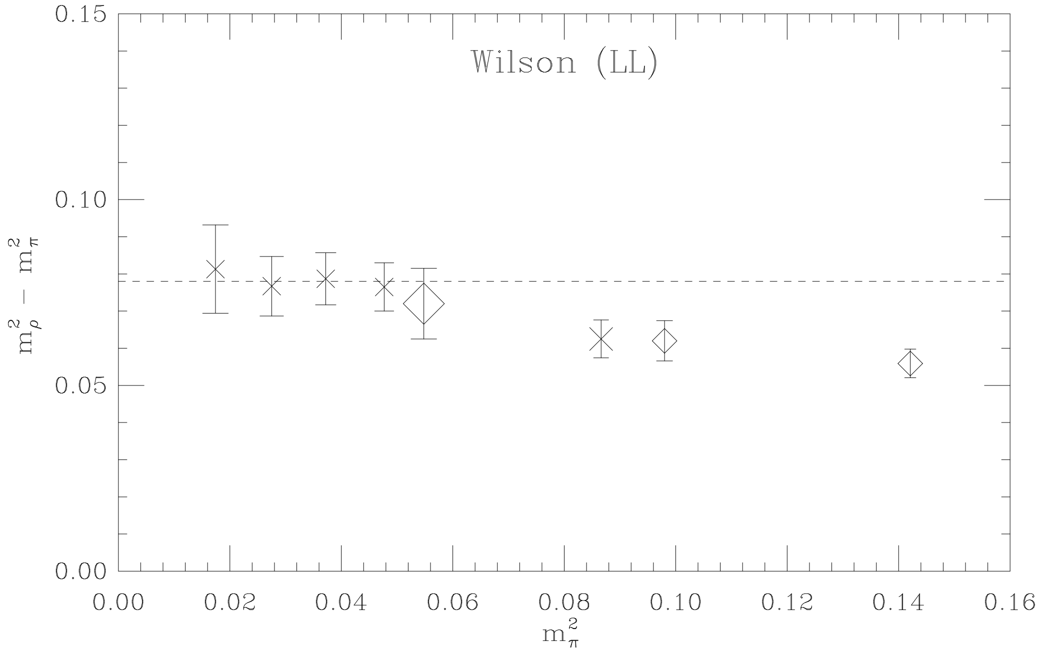


Figure 4.7: Mass splitting of the π and ρ mesons for the Wilson action at different values of m_π^2. The points marked by $\times$ are from this study, while those marked by $\diamond$ are from [ELC 91]. The dotted line represents the weighted average of the 4 points with smallest m_π^2. The error in m_π^2 is less than the width of the symbols.

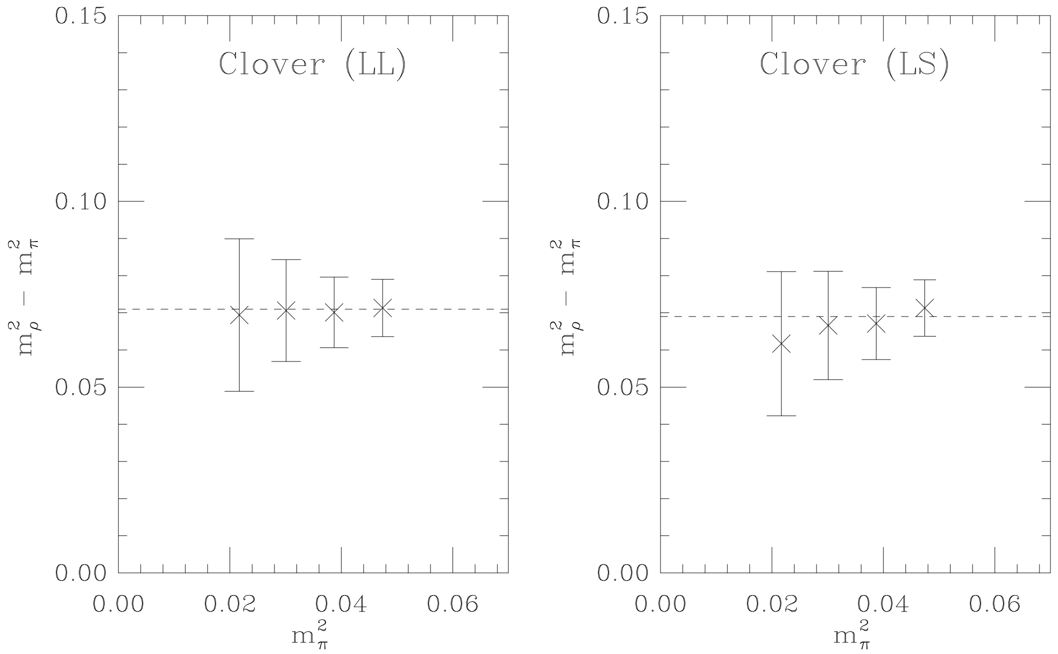


Figure 4.8: Mass splitting of the π and ρ mesons for the Clover action using LL (left) and LS propagators. The dotted lines represent the weighted averages of the data.

at $\beta = 6.2$, are in agreement with our points at larger m_π and show a clear upward slope as m_π decreases. The lighter points are still consistent with this trend and we would need additional statistics to decide whether the slope does flatten at light m_π. A similar slope was found at $\beta = 6.0$ by [APE 90]. To test if this trend is an $O(a)$ effect, it would be very interesting to see data from heavier pion mass points for the Clover action.

Action	$m_\rho^2 - m_\pi^2$	
	Lattice	Physical
Wilson (LL)	0.078(4)	0.57(3)
Clover (LL)	0.071(5)	0.52(4)
Clover (LS)	0.069(5)	0.50(4)

Table 4.8: Lattice and physical values of the π-ρ mass splittings for the different actions. The physical values were calculated using the lattice spacing determined from the string tension, i.e. $a^{-1} = 2.70(3)$ GeV.

For each of the actions we use the weighted average of the 4 points with lightest m_π as the measure of the meson splitting. This is plotted as a dotted line on each of the graphs. The results can also be seen in Table 4.8 along with the physical values deduced using the lattice spacing calculated from the string tension. Using the lattice spacing deduced from m_ρ increases the physical values estimated from the Clover action to around 0.6 GeV^2.

The conclusion is that not only do we see a constant value of $m_\rho^2 - m_\pi^2$ over a range of κ values but that this value is very close to the known experimental number. This is a good indication that our calculations are at light enough quark mass to approach the continuum limit.

4.4.6 Edinburgh Plots

To relate lattice masses to the physical world, it is necessary either to chose some quantity, e.g., the string tension, to set the scale or else to study ratios of masses. The *Edinburgh Plot* is a plot of m_N/m_ρ against m_π/m_ρ. It was invented by [Bowler 85] as a pictorial summary of hadron mass data.

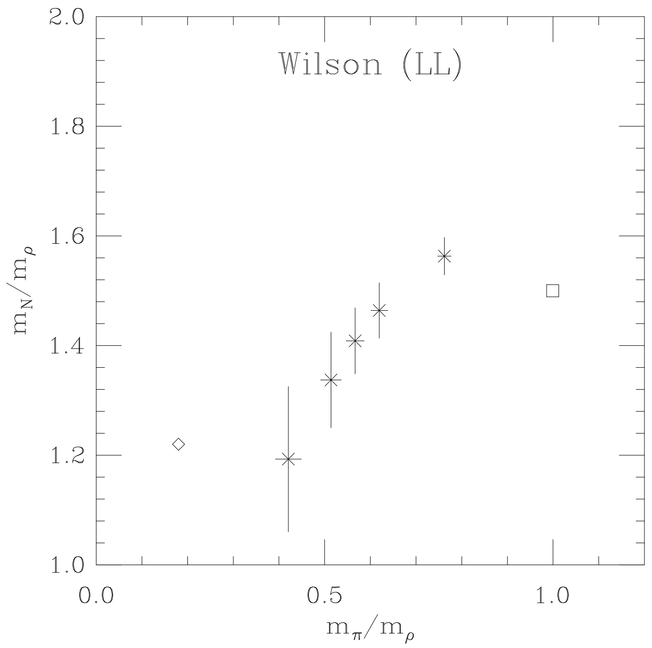


Figure 4.9: Edinburgh plot for the Wilson action. The □ represents the heavy quark limit and the ◇ represents the physical value.

Fig. 4.9, Fig. 4.10 and Fig. 4.11 are the Edinburgh plots for our data. The □ represents the point for infinitely heavy quarks and the ◇ is the experimental point. Fig. 4.9 and Fig. 4.10 are similar in form and in both cases the ratio m_N/m_ρ seems to be falling too quickly. Since the meson data seems to be fine (see e.g., § (4.4.5)), this is probably related to the difficulty of fitting the nucleon. For the smeared propagators in Fig. 4.11, the problem does not appear to be as serious. Nevertheless, these plots are too noisy to draw meaningful conclusions and have really been included only for completeness. More statistics are clearly needed to reduce the error bars.

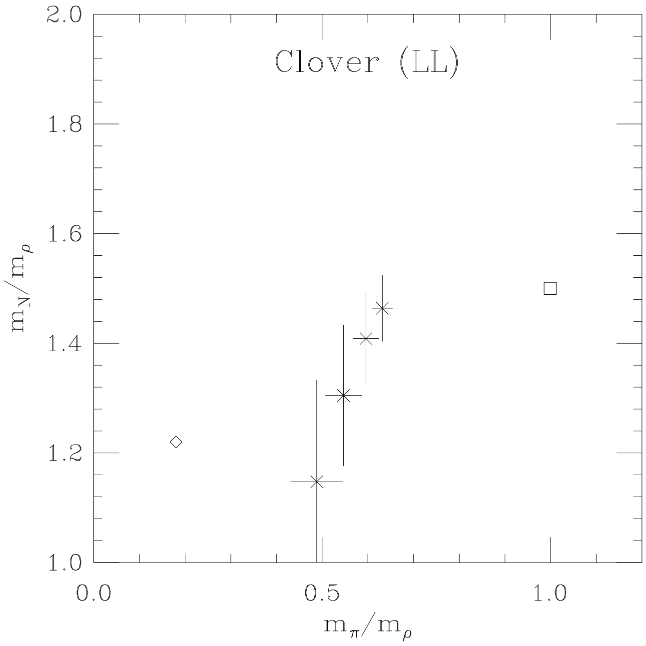


Figure 4.10: Edinburgh plot for the Clover action using LL propagators.

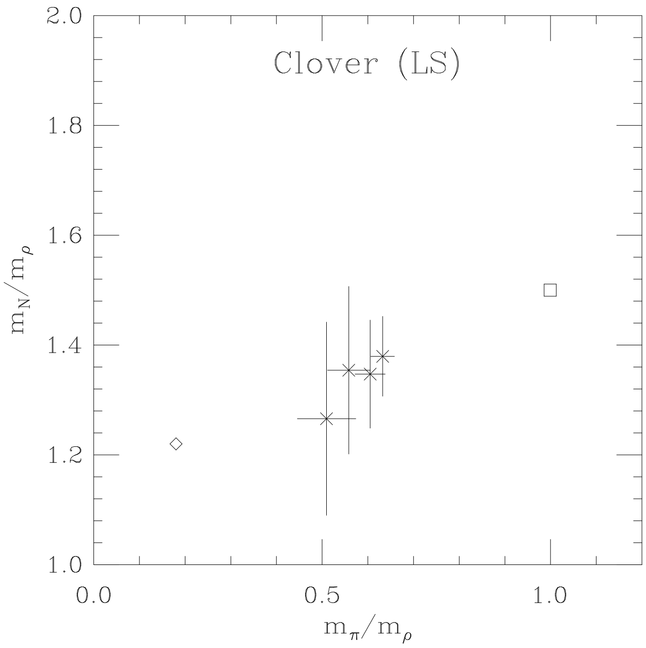


Figure 4.11: Edinburgh plot for the Clover action using LS propagators.

4.5 Summary of Results

The results which have been described in this section are preliminary as they are based on a sample of only 9 configurations. Nevertheless, there are a few key results such as the determination of κ_c for the Clover action and the measurements of meson splitting. The next two sections will summarise the conclusions on the smearing used and on any improvement from the Clover action.

4.5.1 Conclusions on Smearing

Comparing the fit regions for the Clover (LL) and Clover (LS) propagators, (Table 4.5 and Table 4.6), shows that the smearing increases the overlap with the lightest state and allows it to be exposed about 2 timeslices nearer the origin. However, this advantage is off-set by the fact that the signal is noisier further from the origin. Overall, the errors on the fits are smaller for the LL correlators. The only clear gain from the smearing is for the nucleon, where the LS has a small, noisy plateau while it is very difficult to find any stable fit region for the LL data. Since other results suggest that smearing can make large improvements in the data, it may be that the choice of κ_{sc} was poor. There is on-going work investigating the optimisation of the smearing radius.

4.5.2 Conclusions on Clover Action

The main result, so far, from our study of this new action is the determination of κ_c in § (4.4.2). With the statistics presently available, the results for the hadron masses from the Wilson (LL) and Clover (LL) propagators are almost indistinguishable, e.g., there is a strong similarity between the Edinburgh plots in

Fig. 4.9 and Fig. 4.10. It may be the case that the region of comparison between the two actions is sufficiently near the critical point that $O(ma)$ effects are relatively unimportant for the Wilson action since the masses are small. The results for the Wilson action in § (4.4.5) support this view. At low κ the behaviour of $m_\rho^2 - m_\pi^2$ is clearly κ dependent while at larger κ this is hidden in the statistical noise. It would be very interesting to see how the actions compare further from criticality. To make a meaningful comparison, we would also need more statistics.

Bibliography

[Adler 81] S. L. Adler, *Phys. Rev.* **D23** (1981) *2901*

[Adler 88] S. L. Adler, *Phys. Rev.* **D37** (1988) *458*

[APE 90] *The Ape Collaboration*, S. Cabasino, F. Marzano, J. Pech, F. Rapuano, R. Sarno, G. M. Todesco, W. Tross, N. Cabibbo, M. Guagnelli, E. Marinari, P. Paolucci, G. Parisi, G. Salina, M. P. Lombardo, R. Tripiccione and E. Remiddi, $\beta = 6.0$ *quenched Wilson fermions*, Preprint *ROM2F-90-62*

[Barkai 84] D. Barkai, K. J. M. Moriarty and C. Rebbi, *Phys. Rev.* **D30** (1984) *1293*

[Batrouni 85] G. G. Batrouni, G. R. Katz, A. S. Kronfield, G. P. Lepage, B. Svetitsky and K. G. Wilson, *Phys. Rev.* **D32** (1985) *2736*

[Benav 89] R. Benav, D. Kandel, E. Katznelson, P. G. Lauwers and S. Solomon, *J. Stat. Phys.* **58** (1990) *125*

[Benav 91] R. Benav, H. G. Evertz, M. Marcu and S. Solomon, *Critical acceleration of finite temperature SU(2) simulations*, Preprint *DESY-91-021*

[Booth 90] S. P. Booth, K. C. Bowler, R. D. Kenway, B. J. Pendleton and D. G. Richards, *Performance Estimates for QCD Calculations on the Meiko MK086*, Edinburgh Internal Report

[Booth 91] S. P. Booth, *Applications of Parallel Computers to Particle Physics* Ph.D. Thesis, University of Edinburgh, *in preparation*

[Bowler 83] K. C. Bowler, G. S. Pawley, D. J. Wallace, E. Marinari and F. Rapuano, *Nucl. Phys.* **B220** (1983) *137*

[Bowler 85] K. C. Bowler, D. L. Chalmers, A. Kenway, R. D. Kenway, G. S. Pawley and D. J. Wallace, *Phys. Lett.* **162B** (1985) *354*

[Bowler 89] K. C. Bowler, R. D. Kenway, G. S. Pawley, D. Roweth, and G. V. Wilson, *An Introduction to* OCCAM*2 Programming: 2^{nd} Edition*, published by *Chartwell-Bratt*

[Brower 90] R. C. Brower and Suzhou Huang, *Phys. Rev.* **D41** (1990) *708*

[Burkitt 89] A. N. Burkitt and D. W. Heermann, *Comp. Phys. Comm.* **54** (1989) *201*

[Cabibbo 82] N. Cabibbo and E. Marinari, *Phys. Lett.* **119B** (1982) *387*

[Caracciolo 91] S. Caracciolo, R. G. Edwards, A. Pelisetto and A. D. Sokal, *Nucl. Phys.* **B (Proc. Suppl.) 20** (1991) *72*

[Carpenter 85] D. B. Carpenter and C. F. Baillie, *Nucl. Phys.* **B260** (1985) *103*

[Creutz 80] M. Creutz, *Phys. Rev.* **D21** (1980) *2308*

[Creutz 87] M. Creutz, *Phys. Rev.* **D36** (1987) *515*

[Duane 87] S. Duane, A. D. Kennedy, B. J. Pendleton and D. Roweth, *Phys. Lett.* **B195** (1987) *216*

[Eisenstat 83] S. L. Eisenstat, H. C. Elman and M. H. Schults, *SIAM J. Numer. Anal.* **20** (1983) *345*

[ELC 91] C. Allton and C. T. Sachrajda, *Private Communication*

[Fortuin 72] C. M. Fortuin and P. W. Kasteleyn, *Physica* **57** (1972) *536*

[Frick 89] Ch. Frick, K. Jansen and P. Seuferling, *Cluster algorithms in the $O(4)$ ϕ^4 theory in 4 dimensions* Preprint *HLRZ 89-38*

[Goodman 86] J. Goodman and A. D. Sokal, *Phys. Rev. Lett.* **56** (1986) *1015*

[Gupta 88] R. Gupta, J. DeLapp, G. G. Batrouni, G. C. Fox, C. F. Baillie, and J. Apostolakis, *Phys. Rev. Lett.* **61** (1988) *1996*

[Gupta 89] R. Gupta, A. Patel, C. F. Baillie, G. Guralnik, G. W. Kilcup and S. R. Sharpe, *Phys. Rev.* **D40** (1989) *2072*

[Güsken 90] S. Güsken, U. Löw, K.-H. Mütter, R. Sommer, A. Patel and K. Schilling, *Nucl. Phys.* **B (Proc. Suppl.) 17** (1990) *301*

[Hamber 83] H. W. Hamber and C. M. Wu, *Phys. Lett.* **B133** (1983) *351*

[Hasenbusch 90] M. Hasenbusch, *Nucl. Phys.* **B333** (1990) *581*

[Hasenfratz 84] A. Hasenfratz and P. Hasenfratz, *Wilson Loop Expectation Values in $SU(3)$*

[Heatlie 91] G. Heatlie, C. T. Sachrajda, G. Martinelli, C. Pittori and G. C. Rossi *Nucl. Phys.* **B352** (1991) *266*

[Hestenes 52] M. R. Hestenes and E. Stiefel, *J. Res. Nat. Bur. Standards* **49** (1952) *409*

[Hockney 90] G. M. Hockney, *Nucl. Phys.* **B (Proc. Suppl.) 17** (1990) *301*

[Hoshen 76] J. Hoshen and R. Kopelman, *Phys. Rev.* **B14** (1976) *3438*

[Irving 90] A. C. Irving, *Gauge Field Clusters : Progress Report*

[Karsten 81] L. H. Karsten and J. Smit, *Nucl. Phys.* **B183** (1981) *103*

[Kasteleyn 69] P. W. Kasteleyn and C. M. Fortuin, *J. Phys. Soc. Jpn. Suppl.* **26S** (1969) 11

[Kawasaki 72] K. Kawasaki, in *Phase Transitions and Critical Phenomena Vol. 2* edited by C. Domb and M. S. Green (1972)

[Kennedy 85] A. D. Kennedy and B. J. Pendleton, *Phys. Lett.* **B156** (1985) *393*

[Kenway 89] R. D. Kenway, *Rep. Prog. Phys.* **52** (1989) 1475

[Madras 88] N. Madras and A. D. Sokal, *J. Stat. Phys.* **50** (1988) *109*

[Meiko 89] Meiko Scientific, *CS-Tools for SunOS*, *Ref. No. 83–009A00–02.01*

[Metropolis 53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, *J. Chem. Phys.* **21** (1953) *1087*

[Michael 89] C. Michael and M. Teper, *Nucl. Phys.* **B314** (1989) *347*

[Michael 91] C. Michael, *Private Communication*

[Niedermayer 88] F. Niedermayer, *Phys. Rev. Lett.* **61** (1988) *2026*

[Oyanagi 86] Y. Oyanagi, *Comp. Phys. Comm.* **42** (1986) *333*

[Perantonis 90] S. J. Perantonis and C. Michael, *Nucl. Phys.* **B347** (1990) *854*

[Press 88] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes in C — the Art of Scientific Computing*, published by *Cambridge University Press*

[Rossi 87] P. Rossi, C. T. H. Davies and G. P. Lepage, *Nucl. Phys.* **B297** (1988) *287*

[Sheikholeslami 85] B. Sheikholeslami and R. Wohlert, *Nucl. Phys.* **B259** (1985) *572*

[Sweeney 83] M. Sweeny, *Phys. Rev.* **B27** (1983) *4445*

[Swendsen 87] R. H. Swendsen and J-S. Wang, *Phys. Rev. Lett.* **58** (1987) *86*

[UKQCD 91] *UKQCD Collaboration*, S. P. Booth, K. C. Bowler, D. S. Henty, R. D. Kenway, B. J. Pendleton, D. G. Richards, A. D. Simpson, A. C. Irving, A. McKerrell, C. Michael, P. W. Stephenson, M. Teper and K. Decker, *Phys. Lett.* **B**, *to be published*

[Weingarten 81] D. Weingarten and D. N. Petcher, *Phys. Rev. Lett.* **99B** (1981) *333*

[Weingarten 83] D. Weingarten, *Nucl. Phys.* **B215 [FS7]** (1983) *1*

[Wilson 74] K. G. Wilson, *Phys. Rev.* **D10** (1974) *2445*

[Wilson 75] K. G. Wilson, *Adv. Math.* **16** (1975) *176*

[Wolff 89a] U. Wolff, *Phys. Rev. Lett.* **62** (1989) *361*

[Wolff 89b] U. Wolff, *Phys. Lett.* **B228** (1989) *379*

[Yang 86] M. C. K. Yang and D. H. Robinson, *Understanding and Learning Statistics by Computer*, published by *World Scientific Publishing*

[Zweig 64] G. Zweig, CERN report no. *8419/TH 412*