Parallel Algorithms for Lattice QCD

Thesis submitted by Duncan Roweth for
the degree of Doctor of Philosophy

Department of Physics
University of Edinburgh
October 1987

Declaration

The work in chapter 3 was part of a collaborative project within the department of physics at Edinburgh University. The hybrid monte carlo algorithm of chapter 4 was developed by B.J.Pendleton, S.Duane, A.D.Kennedy and myself. All other work in this thesis is my own unless otherwise stated.

Material from this thesis appears or is to be published in the following :

"Hadron mass calculations at $\beta = 5.7$ and 6.0" K.C. Bowler, C.B. Chalmers, R.D. Kenway, G.S. Pawley and D. Roweth. Nucl. Phys. B284, (1987), pp 299-333.

"Hadron mass calculations at $\beta = 6.15$ and 6.3" K.C.Bowler, C.B. Chalmers, R.D. Kenway, D. Roweth and D.S. Stephenson. Edinburgh Preprint 87/403.

"A high statistics study of $\langle\bar{\psi}\psi\rangle$" I.A. Barbour, K.C. Bowler, P.E. Gibbs and D. Roweth. Phys. Lett. B158, (1985), pp 61.

"Extending Quark Propagators in Time" C.B. Chalmers, R.D. Kenway and D. Roweth. Phys. Lett. B184, (1986), pp 63-68.

"Algorithms for Calculating Quark Propagators on Large lattices" C.B. Chalmers, R.D. Kenway and D. Roweth. J. Comp Phys. 70, (1987), pp 500-520.

"Accelerating Gauge Field Dynamics" S. Duane, R.D. Kenway, B.J. Pendleton and D. Roweth. Phys. Lett. B176, (1986), pp 143-148.

"A Hybrid Monte Carlo Algorithm" S. Duane, A.D. Kennedy, B.J. Pendleton and D. Roweth. Edinburgh Preprint 87/402.

"Design and Performance Estimation for Transputer Arrays." D. Roweth. J. of Systems and Software. Vol 6 Num 1 (1986).

"Distributed Solution of Sparse Matrix Problems in Lattice Gauge Theory" D. Roweth. In preparation.

"Simulation of Quantum Electodynamics on an array of Transputers" K.C. Bowler, R.D. Kenway and D. Roweth. In preparation.

Acknowledgements

I would like to thank my supervisors Richard Kenway and Ken Bowler for their help and encouragement, and Brian Pendleton, Catherine Chalmers and Simon Duane for many useful discussions throughout our work.

I would like to thank INMOS for sponsoring my Ph.D under the CASE awards scheme. I have had many interesting discussions with Roger Shepherd and David May, and have enjoyed working for them.

Finally I would like to thank David, Tien, Catherine, and Brian for making my stay in Edinburgh an enjoyable one.

Abstract

We develop parallel algorithms for the simulation of lattice gauge theories. These algorithms are used to obtain numerical estimates of the low lying hadron mass spectrum in the quenched approximation to lattice quantum chromodynamics. We use the standard Wilson pure gauge action and the staggered fermion action. Results are obtained for β values in the range $5.7 \leq \beta \leq 6.3$ on lattices of size 16^4 and $16^3 \times 24$. Our analysis of baryon propagators suggests that there are significant finite size effects at low quark mass.

An efficient monte carlo algorithm for the simulation of dynamical fermions theories is presented. A simulation of lattice quantum electrodynamics with dynamical electrons is used to compare this approach with the Langevin and hybrid algorithms. This simulation was performed on an array of Transputers.

We discuss the parallelism inherent in lattice gauge theory simulations and its exploitation on the DAP and on arrays of Transputers. We investigate the implementation of dynamical fermion simulations on large arrays of Transputers and find it feasible.

Contents

Chapter 1

Introduction

Quantum Chromodynamics (QCD) is a theory of the strong nuclear force describing the interaction of quarks and gluons, the constituents of hadronic matter. It is a non-Abelian gauge theory. Unlike Quantum Electrodynamics (QED), the successful Abelian gauge theory of electrons and photons, it requires more than perturbative analysis. Features of QCD such as quark confinement are inherently non-perturbative. QCD can be formulated on a lattice; this provides a mechanism for studying its non-perturbative features and in particular provides a framework for numerical simulation of the mass spectrum.

In common with many other 'natural' problems the computational requirements of lattice QCD simulations are immense; they are beyond the capabilities of exisiting supercomputers. Highly parallel computers probably offer us the only possibility of performing such computations.

1.1 Gauge Theories on a Lattice

1.1.1 Gauge Theories

Gauge theories are pre-eminent in modern theoretical particle physics. They achieved this position following the success of Quantum Electrodynamics, a $U(1)$ Abelian gauge theory. The position was strengthened further by the discovery of the W and Z intermediate vector bosons [Arnison et. al. (UA1 collaboration) 1983] at CERN, as predicted by the $SU(2) \otimes U(1)$ electroweak gauge theory of [Glashow Salam and Weinberg].

Gauge theories can be defined in a variety of ways, perhaps the simplest is to impose local symmetry conditions on a Lagrangian (see [Cheng and Li 1984]). The free field Lagrangian

$$\mathcal{L} = \bar{\psi}(x)\left(i\gamma^{\mu}\partial_{\mu} - m\right)\psi(x) \tag{1.1}$$

has a global $U(1)$ invariance under change of phase,

$$\begin{aligned} \psi(x) &\rightarrow \psi'(x) = e^{-i\alpha}\psi(x) \\ \bar{\psi}(x) &\rightarrow \bar{\psi}'(x) = e^{i\alpha}\bar{\psi}(x). \end{aligned} \tag{1.2}$$

We 'gauge' this symmetry by replacing α by a space-time dependent phase $\alpha(x)$. The derivative term $\bar{\psi}(x)\partial_\mu\psi(x)$ becomes

$$\bar{\psi}'(x)\partial_\mu\psi'(x) = \bar{\psi}(x)e^{i\alpha(x)}\partial_\mu\left(e^{-i\alpha(x)}\psi(x)\right) \tag{1.3}$$

$$= \bar{\psi}(x)\left(\partial_\mu - i\partial_\mu\alpha(x)\right)\psi(x) \tag{1.4}$$

The second term spoils the invariance. We require a derivative of the form

$$D_\mu\psi(x) \rightarrow [D_\mu\psi(x)]' = e^{-i\alpha(x)}D_\mu\psi(x) \tag{1.5}$$

This is not possible for a free theory. We must introduce a new vector field $A_\mu(x)$, the gauge field, and construct a covariant derivative

$$D_\mu\psi(x) = \left(\partial_\mu + ieA_\mu(x)\right)\psi(x) \tag{1.6}$$

This satisfies eq. 1.5 provided that $A_\mu(x)$ transforms as

$$A_\mu(x) \rightarrow A'_\mu(x) = A_\mu(x) + \frac{1}{e}\partial_\mu\alpha(x) \tag{1.7}$$

We must also specify the dynamics of the gauge fields. The simplest gauge invariant term we can use is

$$\mathcal{L}_1 = -\frac{1}{4}F_{\mu\nu}F^{\mu\nu} \tag{1.8}$$

where $F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu$ is the field strength tensor. $F_{\mu\nu}$ is related to D_μ by

$$[D_\mu D_\nu - D_\nu D_\mu]\,\psi = ieF_{\mu\nu}\psi \tag{1.9}$$

The full Lagrangian is then

$$\mathcal{L} = -\frac{1}{4}F_{\mu\nu}F^{\mu\nu} + \bar{\psi}(x)\left(i\gamma^\mu\partial_\mu + ieA_\mu(x)\right)\psi(x) - m\bar{\psi}(x)\psi(x) \tag{1.10}$$

It describes a massless photon A_μ whose coupling to a matter field ψ is determined by the U(1) symmetry. The photon has no self coupling and so the theory is free without the matter fields.

1.1.2 Quantum Chromodynamics

A Little History The discovery of the proton and neutron was followed by that of a myriad of mesons and excited states of the nucleons. Gell-Mann and Zweig suggested that all hadronic matter (baryons and mesons) is made from 3 *fundamental* particles, the up, down and strange quarks [Gell-Mann 1964] [Zweig 1964]. They constructed families of particles according to representations of an $SU(3)$ flavour symmetry.

The $SU(3)$ of flavour is broken - the strange quark has higher mass than the up and down quarks. Further quarks (charm and bottom) were proposed to explain the existance and decay of the J/Ψ and B mesons and subsequently discovered. (We should qualify the use of "discovered"; quarks have never been seen in isolation but there is evidence for pointlike fermionic objects carrying fractional charge within a proton.) A sixth, the top quark, is proposed. We now think of quarks as coming in generations: a pair of quarks with a lepton and a neutrino [Gross and Wilczek 1973]. The basic dynamics of quarks is thought to be largely independent of flavour, provided that the number of generations is not too high.

Han and Nambu proposed a hidden exact $SU(3)$ symmetry, colour, in 1965. Without it the Δ^{++} would contravene the spin-statistics rules for fermions [Han and Nambu 1964]. Experimental support for this idea came from the measurement of

$$R = \frac{\sigma\left(e^+e^- \rightarrow \text{hadrons}\right)}{\sigma\left(e^+e^- \rightarrow \mu^+\mu^-\right)}$$

Without the extra colour degrees of freedom the predicted cross-section for decay to hadrons is a factor of 3 smaller than that measured.

Deep inelastic scattering experiments demonstrated that at high energies (short distances) quarks are free (or weakly bound), and carry about 50% of the proton's momentum. At lower energies (longer distance scales) the quarks are tightly bound, presumably confined. The parton model of the strong nuclear force proposes a world made from valence quarks and dynamical quark anti-quark pairs bound together with gluons. The quarks and gluons carry colour charge, but physical states are all colour singlet.

A theoretical basis for these phenomena was provided by the construction of non-Abelian gauge theories [Yang Mills 1954] and the discovery of asymptotic freedom [Politzer 1973].

Asymptotic Freedom The functional dependence of the coupling constant g in a field theory on a=distance (energy) scales is given by the renormalisation group β function

$$\beta(g) = -\frac{\partial}{\partial(\ln a)}g \tag{1.11}$$

A perturbative expansion of $\beta(g)$ at weak coupling gives

$$\beta(g) = -\beta_0 g^3 - \beta_1 g^5 + O(g^7) \tag{1.12}$$

where $\beta_0 = \frac{-1}{16\pi^2}\left(11 - \frac{2}{3}n_f\right)$ for an $SU(3)$ gauge theory with n_f flavours of fermions. For $n_f \leq 16$ $\beta_0(g) < 0$ as $g \rightarrow 0$ implying that the coupling constant goes to zero at short distances.

The strength of the forces grows with increasing distances scales, to the point where the perturbation expansion breaks down. This is consistent with confinement, but does not imply it.

QCD Non-Abelian gauge theories are the only renormalisable 4-dimensional theories that are asymptotically free. They were developed by Yang and Mills (see [Yang Mills 1954]). Consider the local symmetry transformation

$$\psi(x) \to \psi'(x) = e^{-i\alpha^a(x)\tau^a}\psi(x) \tag{1.13}$$

where the τ^a are generators of an $SU(3)$ symmetry group (the Gell-Mann $\frac{\lambda}{2}$ matrices) and ψ is a 3 component field. As in the Abelian case a consistent theory requires gauge fields $A^a_\mu(x)$, one per generator, which transform according to

$$A^{a\prime}_\mu(x) = A^a_\mu(x) + f^{abc}\alpha^b A^c_\mu(x) - \frac{1}{g}\partial_\mu\alpha^a(x) \tag{1.14}$$

(where the f^{abc} are the structure constants of the gauge group) and the covariant derivative

$$D_\mu\psi(x) = (\partial_\mu - ig\tau \cdot A_\mu(x))\,\psi(x) \tag{1.15}$$

The QCD Lagrangian is

$$\mathcal{L} = -\frac{1}{4}F^a_{\mu\nu}F^{a\mu\nu} + \bar{\psi}i\gamma^\mu D_\mu\psi - m\bar{\psi}\psi \tag{1.16}$$

The field strength tensor $F^a_{\mu\nu}$ is given by

$$F^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu + gf^{abc}A^b_\mu A^c_\nu \tag{1.17}$$

Much of the complex structure of QCD is due to self couplings amongst the gauge fields in the Yang-Mills term $F^a_{\mu\nu}F^{a\mu\nu}$.

The predictions of QCD at high energies have been tested by deep inelastic scattering and the physics of quark-gluon jets. Our direct experience with the consequences of QCD is at much lower energies; this determines the strength of couplings, the hadron masses and the values of matrix elements. These quantities are essentially non-perturbative. Lattice regularisation schemes enable us to study QCD in this limit, making possible predictions that can be tested experimentally.

1.1.3 Lattices

Wilson showed that it is possible to construct a gauge invariant theory on a lattice [Wilson 1974]. The lattice provides a non-perturbative regulator, cutting off all wavelengths less than twice the latttice spacing. It also provides a framework for establishing the connection between the Feynman path integral (FPI) formulation of quantum field theories and a statistical spin system. Consider a ϕ^4 theory in 4 dimensions.

$$\mathcal{L} = \frac{1}{2}\left(\partial_\mu\phi\right)^2 + \frac{1}{2}m^2\phi^2 + \frac{g^2}{4!}\phi^4 \tag{1.18}$$

(see [Ramond 1981]). In the FPI formalism the expectation value of an observable O is given by

$$\langle O\rangle = \frac{1}{Z}\int [d\phi]\, O \exp\left(i\int_t^{t'} d\tau \int d^3x\mathcal{L}(x,\tau)\right) \tag{1.19}$$

where

$$Z = \int [d\phi] \exp\left(i\int_t^{t'} d\tau \int d^3x\mathcal{L}(x,\tau)\right) \tag{1.20}$$

Introducing a lattice allows us to define the functional integral $\int [d\phi]$ properly. Consider a hypercubic lattice of spacing a where sites are labelled by $n = (n_0, n_1, n_2, n_3)$ and the 4 directions in the lattice are labelled by $\hat{\mu}$. If we replace $\phi(x)$ by $\phi(n)$, $\partial_\mu\phi$ by the forward difference $\frac{1}{a}\left(\phi(n+\hat{\mu}) - \phi(n)\right)$ and the ordinary integral by a sum over sites weighted by a^4 eq. 1.19 can be written as a multiple integral

$$\langle O\rangle = \frac{1}{Z}\int \prod_n d\phi(n) e^{-S(\phi)} \tag{1.21}$$

This is analogous to a statistical spin system in the canonical ensemble, with the correspondence $\frac{S}{g^2} \longleftrightarrow \frac{E}{KT}$.

1.1.4 Wilson's Pure Gauge Theory

In Wilson's pure gauge formulation of QCD the gauge fields are path-dependent 'phases' (elements of the gauge group) living on the links of a hypercubic lattice. They transport colour charge from a site to its neighbours, and are defined by writing

$$U_\mu(n) = e^{iagA_\mu(n)} \tag{1.22}$$

where A_μ is an element of the lie algebra of the gauge group. The gauge transformation law is

$$U_\mu(n) \rightarrow U'_\mu(n) = \Omega(n)U_\mu(n)\Omega^{-1}(n+\hat{\mu}) \tag{1.23}$$

the $\Omega \in SU(3)$ are arbitrary group rotations at each lattice site.

We require that the lattice action should be gauge invariant and that it should reduce to the Yang-Mills action in the continuum limit. This suggests that we should construct it from integrals of A_μ around small closed contours. Motivated by this Wilson proposed the action

$$S_G = \frac{\beta}{2N}\sum_\Box ReTr(U_\Box) \; + \; \text{hermitean conjugate} \tag{1.24}$$

where

$$U_\Box = U_\mu(x)\, U_\nu(x+\hat{\mu})\, U_\mu^\dagger(x+\hat{\nu})\, U_\nu^\dagger(x) \tag{1.25}$$

is the product of links around an elementary square, or plaquette, of the lattice (see figure 1.1). This trace of such plaquettes is manifestly gauge invariant.

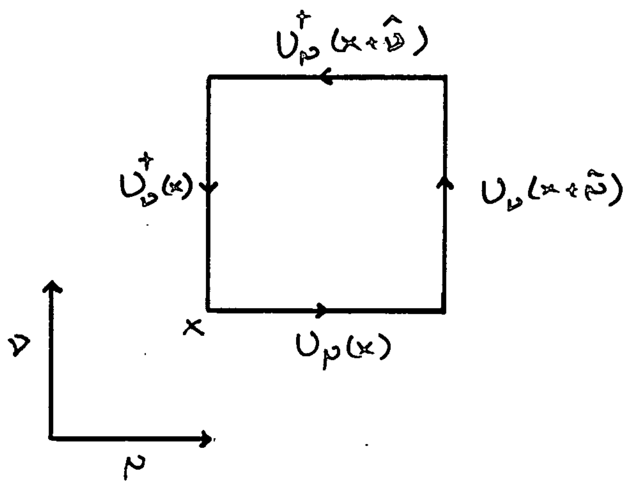


Figure 1.1: The plaquette

Applying our definition of lattice derivatives to eq. 1.22

$$A_\nu(n+\hat{\mu}) = A_\nu(n) + a\partial_\mu A_\nu(n) \tag{1.26}$$

and making use of the Baker-Campbell-Hausdorf relation we have

$$\begin{aligned} U_\Box &= e^{igaA_\mu} e^{iga(A_\mu + a\partial_\mu A_\nu)} e^{-iga(A_\mu - a\partial_\nu A_\mu)} e^{-igaA_\nu} \\ &= e^{iga^2(\partial_\mu A_\nu - \partial_\nu A_\mu) + ig[A_\mu, A_\nu] + \cdots} \\ &= e^{iga^2 F_{\mu\nu}} + O(a^4) \end{aligned} \tag{1.27}$$

for small a. Expanding the exponential and taking the trace,

$$TrU_\Box = Tr\left(1 + iga^2 F_{\mu\nu} - \frac{1}{2} g^2 a^4 F_{\mu\nu} F^{\mu\nu} + \cdots\right) \tag{1.28}$$

Now $Tr(F_{\mu\nu}) = 0$ as the generators of an $SU(N)$ group are traceless, and so

$$Tr\left(F_{\mu\nu}F^{\mu\nu}\right) = Tr\left(F^a_{\mu\nu}\frac{\lambda^a}{2} F^{b\mu\nu}\frac{\lambda^b}{2}\right) = \frac{1}{2} F^a_{\mu\nu} F^{a\mu\nu} \tag{1.29}$$

Finally we replace the sum over sites with an appropriately normalised integral and

$$\frac{\beta}{2N}\sum_\Box Tr(U_\Box) \; + \text{h.c} \; = \; -2\left(\frac{\beta}{2Na^4}\int d^4x \frac{g^2 a^4}{2} F^a_{\mu\nu} F^{a\mu\nu}\right) \tag{1.30}$$

$$= -\frac{1}{4}\int F^a_{\mu\nu} F^{a\mu\nu} \tag{1.31}$$

setting $\beta = \frac{2N}{g^2}$ we recover the Yang-Mills action.

Asymptotic Scaling

By means of numerical simulation we can measure the value of a quantity on the lattice at finite a. But in order to extract a prediction for the physical value we must first take the $a \to 0$ limit, removing the cutoff. Consider some observable O, of dimension d. In lattice units

$$O = a^d f(g) \tag{1.32}$$

where $f(g)$ is a dimensionless function containing the physical information. The physical value of O is independent of a. To make sense of measurements of O at finite a we must first understand the way in which the coupling g depends upon a, and thus the functional dependence of f on a. The work described in section 1.1.2 provide us with this information at weak coupling.

For example, masses measured on the lattice have the form $m = \frac{1}{a} f(g)$, and so diverge as we take the continuum limit. But all masses have the same functional dependence upon g as $g \to 0$ and so ratios of masses will be independent of g in the scaling regime. We must determine how close to $g = 0$ it is necessary to work before we see this scaling.

To date lattice QCD simulations have been performed for $\beta \sim 6$, ($g \sim 1$). Monte carlo renormalisation group (MCRG) analysis [Bowler et. al. 1986] suggests that deviations from scaling are less than 10% for $\beta > 6.1$. Measurements of the deconfinement temperature (in the absense of fermions) [Kennedy et. al. 1986] show scaling setting in at around $\beta = 6.15$.

Quark Confinement

Quark confinement is demonstrable on the lattice at strong coupling [Tomboulis 1983]. Consider an experiment in which a quark anti-quark pair are separated adiabatically to a distance R, held apart for a time T and then brought together again. The amplitude for this process is $< f \mid e^{-HT} \mid i >$ in a euclidean formulation where i and f are the initial and final states. Wilson represented this process by a generalisation of the plaquette, to an R by T loop,

$$W(R,T) = Tr \prod_{\text{loop}} U_\mu(n) \tag{1.33}$$

In the limit $T \to \infty$ the loop $W(R,T) \sim e^{-TV(R)}$ where $V(R)$ is the inter-quark potential. Measurements of ratios of Wilson loops support a potential of the form

$$V(R) \sim \sigma R + \frac{c}{R} \tag{1.34}$$

where σ is the string tension. These measurements are from simulations at intermediate coupling. Confinement in the continuum limit is dependent on there not being a phase transition at weaker coupling.

1.1.5 Fermions on the Lattice

A naive formulation of fermions on a lattice is to take the dirac action in euclidean space

$$S_F = \int d^4x \bar{\psi}(x)(\gamma_\mu D_\mu + m)\psi(x) \tag{1.35}$$

where $D_\mu = \partial_\mu + igA_\mu$ and replace the partial derivative with a central difference

$$S_N = \frac{1}{2a}\sum_{n\mu} \bar{\psi}(n)\gamma_\mu \left(U_\mu(n)\psi(n+\hat{\mu}) - U^\dagger_\mu(n-\hat{\mu})\psi(n-\hat{\mu})\right) + m\sum_n \bar{\psi}(n)\psi(n) \tag{1.36}$$

where the fermion fields $\psi(n)$ are associated with sites of the lattice. The links are necessary to make the action gauge invariant. For free fermions the momentum space propagator corresponding to the naive action eq. 1.36 has the form

$$\tilde{G}_N(p) = \left(\frac{i}{a}\sum_\mu \gamma_\mu \sin p_\mu + m\right)^{-1} \tag{1.37}$$

which has poles for $\sum_\mu \sin p_\mu = 0$ in the $m \to 0$ limit. In addition to the expected pole at zero momentum there are a further 15; the fermions having "doubled" in each dimension.

Wilson Fermions

Wilson proposed a mechanism for giving the unwanted fermions a mass of order $\frac{1}{a}$, decoupling them as $a \to 0$. He did this by adding a second derivative term to the action. For free fermions

$$S_W = S_N - \frac{1}{2a}\sum_{n\mu} r\bar{\psi}(n)\left(\psi(n+\hat{\mu}) + \psi(n-\hat{\mu})\right) \tag{1.38}$$

This gives the propagator the form

$$\tilde{G}_W(p) = \left(\frac{i}{a}\sum_\mu \left(\gamma_\mu \sin p_\mu - \frac{r}{a}\cos p_\mu\right) + m\right)^{-1} \tag{1.39}$$

for which the only pole as $a \to 0$ is at $p_\mu = 0$. There are several disadvantages with this approach; the r term explicitly breaks chiral symmetry and the value of the critical mass (to which we must tune m for zero mass quarks) must be determined numerically.

Staggered Fermions

Susskind proposed a mechanism for thinning out the unwanted fermion degrees of freedom by recognising that the Dirac multiplets decouple in the lattice formulation. This can be seen (in [Kogut and Susskind 1975] and [Susskind 1977]) via the transformation [Kawamoto and Smit 1981]

$$\begin{aligned}\psi(n) &= T(n)\chi(n) \\ \bar{\psi}(n) &= \bar{\chi}(n)T^\dagger(n)\end{aligned} \tag{1.40}$$

where $T(n) = \gamma_0^{n_0}\gamma_1^{n_1}\gamma_2^{n_2}\gamma_3^{n_3}$. This spin-diagonalises the naive lattice action. In terms of the χ fields

$$\begin{aligned}S_N &= \frac{1}{2a}\sum_{n\mu}\bar{\chi}(n)T^\dagger(n)\gamma_\mu\left(U_\mu(n)T(n+\hat{\mu})\chi(n+\hat{\mu}) - U_\mu^\dagger(n-\hat{\mu})T(n-\hat{\mu})\chi(n-\hat{\mu})\right) \\ &+ m\sum_n\bar{\chi}(n)\chi(n)\end{aligned} \tag{1.41}$$

contracting the gamma matrices and the T's we get

$$\begin{aligned}S_N &= \frac{1}{2a}\sum_{n\mu}\bar{\chi}(n)\eta_\mu(n)\left(U_\mu(n)\chi(n+\hat{\mu}) - U_\mu^\dagger(n-\hat{\mu})\chi(n-\hat{\mu})\right) \\ &+ m\sum_n\bar{\chi}(n)\chi(n)\end{aligned} \tag{1.42}$$

where $\eta_\mu(n) = (-1)^{n_1+n_2+\cdots n_{\mu-1}}$, the Kawamoto-Smit phases. Eq. 1.42 is diagonal in spinor space so 3 of the 4 copies may be thrown away, reducing the fermion degeneracy to 4.

We define quark fields $q^{\alpha a}$ and $\bar{q}^{\alpha a}$ on a lattice of spacing $2a$ for which $x_\mu = 2y_{mu} + \eta_\mu$ and $\eta_\mu = 0$ or 1 [Kluberg-Stern et. al. 1983]

$$q^{\alpha a}(x) = \sum_\eta \Gamma_\eta^{\alpha a}U_\eta(x)\chi(2y+\hat{\eta}) \tag{1.43}$$

$$\bar{q}^{\alpha a}(x) = \sum_\eta \bar{\chi}(2y+\hat{\eta})U_\eta^\dagger(x)\Gamma_\eta^{*\alpha a} \tag{1.44}$$

where $\Gamma_\eta^{\alpha a} = \gamma_0^{\eta_0}\gamma_1^{\eta_1}\gamma_2^{\eta_2}\gamma_3^{\eta_3}$, α labeling the Dirac index, and a the flavour index.

A $U(1)\otimes U(1)$ remnant of the $U(4)\otimes U(4)$ chiral symmetry of the Dirac action remains, but there are still 4 fermions rather than 1. Kluberg-Stern et. al. have provided a flavour interpretation of this degeneracy.

From a computational point of view the staggered fermion formulation is simpler than that for Wilson fermions, and there is only a quarter of the data.

Neither Wilson nor staggered fermions supply the desired one-fermion model with chiral symmetry. This is a consequence of the Nielsen-Ninomiya no-go theorem which states that chiral symmetry must be broken if we wish to construct an undoubled fermion theory with a local action [Nielsen and Ninomiya 1981].

1.1.6 Making Measurements in a Fermionic Theory

To calculate a physical observable O in a lattice theory we must measure the expectation value of a corresponding operator; one with the correct continuum limit. The general form for such an expectation value (using the Wilson gauge action and staggered fermions) is

$$\langle O \rangle = \frac{1}{Z} \int [dU]\,[d\chi]\,[d\bar{\chi}]\;\; O(U,\chi,\bar{\chi})\; e^{-S(U,\chi,\bar{\chi})} \tag{1.45}$$

where

$$Z = \int [dU]\,[d\chi]\,[d\bar{\chi}]\, e^{-S(U,\chi,\bar{\chi})} \tag{1.46}$$

The fermion fields χ and $\bar{\chi}$ are Grassmann variables. The quadratic form of the fermionic action allows us to integrate over them [Mathews and Salam 1954/5], bringing a factor of $\det(\not{D}+M)$ down from the exponential, leaving only the pure gauge action. The determinant represents the contribution to the action of closed fermion loops. For example

$$\langle \chi\bar{\chi} \rangle = \frac{1}{Z} \int [dU] \det(\not{D}+M)\;\; (\not{D}+M)^{-1} e^{-S_G(U)} \tag{1.47}$$

and

$$Z = \int [dU]\, e^{-S_{\text{eff}}(U)} \tag{1.48}$$

where the effective action S_{eff} is given by

$$S_{\text{eff}}(U) = S_G(U) - \operatorname{tr}\ln(\not{D}+M) \tag{1.49}$$

the integral over the gauge fields remains, and must be performed numerically (see section 1.2).

To measure quantities such as hadron masses in a lattice simulation we must construct an operator with the appropriate quantum numbers and measure it on a set of configurations. As an example of this process we consider the general meson operator in the next section.

Identification of Meson Operators

We combine the quark $q_a(x)$ with appropriate gamma matrices Γ_λ, $\lambda = 1, 16$, in pairs or triples to form operators corresponding to mesonic and baryonic bound states. See [Chalmers 1987] and [Gilchrist et. al. 1984] for a thorough review.

Mesons The general meson operator is

$$J_{\lambda\delta}(x) = \sum_a \bar{q}_a(x)\,(\Gamma_\lambda \otimes \Gamma_\delta^*)\, q_a(x) \tag{1.50}$$

The indices are a colour, λ spinor, and δ flavour. We use eq. 1.43 to write eq. 1.50 in terms of the one component staggered fermion fields.

$$J_{\lambda\delta}(x) = \sum_{a\eta\eta'} \bar{\chi}(2y+\eta) U_\eta^\dagger(x) \Gamma_\eta^{*a\alpha} \left(\Gamma_\lambda^{\alpha\beta} \otimes \Gamma_\delta^{*ab}\right) \Gamma_{\eta'}^{b\beta} U_{\eta'}(x) \chi(2y+\eta') \tag{1.51}$$

Solving the lattice Dirac equation for χ is computationally very expensive. We cannot conceive of being able to evaluate all 16 propagators from each point of the hypercube at $2y$. Instead we impose the restriction $\eta = \eta'$, and work with the 'local' meson operators; eq. 1.51 becomes

$$J_{\lambda\delta}(x) = \sum_{a\eta} \bar{\chi}(2y+\eta)\mathrm{Tr}\left[\Gamma_\eta^\dagger \Gamma_\lambda \Gamma_\eta \Gamma_\delta^\dagger\right] \chi(2y+\eta) \qquad (1.52)$$

For the trace to be non-zero $\lambda = \delta$ so the general local meson operator is

$$J_\lambda = \sum_{a\eta} \bar{\chi}\chi \mathrm{Tr}\left[\Gamma_\eta^\dagger \Gamma_\lambda \Gamma_\eta \Gamma_\lambda^\dagger\right] \qquad (1.53)$$

The rest-frame meson propagators from origin o to time t are [Gilchrist et. al. 1984].

$$M_\lambda(t) = \sum_{\underline{x}} \langle J_\lambda(x) J_\lambda(o)\rangle \qquad (1.54)$$

We calculate them for $\Gamma_\lambda = 1,\ \gamma_5,\ \gamma_0\gamma_5,\ \gamma_\mu,\ \gamma_0\gamma_\mu, \gamma_5\gamma_\mu$ and $\gamma_\mu \otimes \gamma_\mu$. To match the $M_\lambda(t)$ with physical states we evaluate their J^{PC}. The parity operation is determined by

$$\begin{aligned} \mathcal{P} q(x) \mathcal{P}^{-1} &= P q(x_0, \underline{x}) & (1.55) \\ P &= \gamma_4 \otimes I & \\ & & (1.56) \end{aligned}$$

where the I acts on flavour space and the γ_4 on spinor space. Operating with P on $J_\lambda(x)$ we obtain the continuum parity identifications shown in table 1.1. The meson propagators are expected to have the general form

λ	J^{PC}	meson
1	0^{++}	$\delta(980)$
$\gamma_5, \gamma_4\gamma_5$	0^{-+}	$\pi(140)$
$\gamma_\mu, \gamma_4\gamma_\mu$	1^{--}	$\rho(770)$
$\gamma_\mu \otimes \gamma_\mu$	1^{+-}	$B(1235)$
$\gamma_5\gamma_\mu$	1^{++}	$A_1(1270)$

Table 1.1: Identification of continuum J^{PC} quantum numbers for the low lying meson propagators

$$M_\lambda(t) \sim \sum_i A_i e^{-m_i^+(t-t_o)} + (-1)^{t-t_o} e^{-m_i^-(t-t_o)} \qquad (1.57)$$

where the index i labels the ground state, first excited state, etc and $+$ and $-$ label states of positive and negative parity.

1.2 Simulating Field Theories

In order to calculate the expectation value of some operator $\Omega(U)$, we must evaluate

$$\langle\Omega\rangle = \frac{1}{Z}\int [dU]\Omega(U)e^{-S(U)} \tag{1.58}$$

where the partition function Z is given by

$$Z = \int [dU]e^{-S(U)} \tag{1.59}$$

The Monte Carlo method computes $\langle\Omega\rangle$ by generating U-field configurations at random with probability $P_S(U) = \frac{1}{Z}e^{-S(U)}$, and then measuring

$$\bar{\Omega} \equiv \frac{1}{T}\sum_{t=1}^{T}\Omega\left(U_t\right) \tag{1.60}$$

on a sequence $\{U_t\}$ of such configurations. As $T \to \infty$ we find that

$$\bar{\Omega} = \langle\Omega\rangle + O\left(\frac{1}{\sqrt{T}}\right) \tag{1.61}$$

The most useful technique for generating a sequence of configurations with the desired distribution is to construct a Markov process. A Markov process is a stochastic procedure which generates a new configuration U' from its predecessor U with probability $P_M\left[U \mapsto U'\right]$ which depends only upon U'.

Define two ensembles E_1 and E_2 where ensemble E_2 resulted from the application of our Markov process to E_1. Let their separation be

$$|E_1 - E_2| = \sum_U |P_1(U) - P_2(U)| \tag{1.62}$$

where $P_i\left(U\right)$ is the probability of finding configuration U in ensemble E_i. Now

$$P_2\left(U'\right) = \sum_U P_M\left[U \mapsto U'\right]P_1\left(U\right) \tag{1.63}$$

Let E_S be the equilibrium ensemble, that which the Markov process transforms to itself. Using the detailed balance condition

$$P_S(U)P_M\left[U \mapsto U'\right] = P_S(U')P_M\left[U' \mapsto U\right] \tag{1.64}$$

we find that

$$\begin{aligned} |E_2 - E_S| &= \sum_U \left|\sum_{U'} P_M\left[U' \mapsto U\right](P_1(U') - P_S(U'))\right| \\ &\leq \sum_{UU'} P_M\left[U' \mapsto U\right]|P_1(U') - P_S(U')| = |E_1 - E_S| \end{aligned} \tag{1.65}$$

since $\sum_{U'} P_M[U \mapsto U'] = 1$ and so a Markov process will converge to an equilibrium fixed point P_S provided that it is strong ergodic ($P_M[U' \mapsto U] > 0$) and satisfies detailed balance. In fact all that is required is that U' be obtainable from U in some finite number of steps, weak ergodicity.

The details of the Markov process are not specified. We discuss desirable features of such a process in chapter 4. Two schemes that have been used widely are the Heat-Bath [Creutz 1980] [Pietarinen 1981] [Cabibbo and Marinari 1982] and Metropolis [Metropolis et. al. 1953] algorithms.

Heat-Bath The Heat-Bath algorithm is analogous to bringing each degree of freedom in contact with a thermal source. This causes it to fluctuate over all its possible values. Removing the heat bath leaves the link in a given state with probability $e^{-\beta S}$. Detailed balance is clearly satisfied.

Metropolis In the basic Metropolis algorithm we propose an update to each link and evaluate the change in action ΔS that would result from acceptance of the new link. If $\Delta S < 0$ we accept the change. If not we generate a random number r between 0 and 1 with uniform distribution and accept the change if $r < e^{-\beta\Delta S}$. In a multi-hit Metropolis algorithm we make n attempts to update each link. Heat-Bath and Metropolis are equivalent in the limit $n \to \infty$.

To compute the average value $\bar{\Omega}$ we equilibrate the system, then measure $\Omega(U_t)$ on a sequence of statistically independent configurations.

1.2.1 Fermion Simulations

The Wilson action for a pure gauge theory is local, and so evaluating the change in the action is relatively easy. When we include fermions we must generate configurations with probability distribution $\det(\not{D}+M)\,e^{-S_G}$. The presence of the determinant makes the action highly non-local, and thus unsuitable for numerical simulation - we would have to evaluate the change in the determinant every time we wished to update a link.

A great deal of work has gone into ways of getting around this problem. A few of the suggested solutions are discussed below. Write $\mathcal{M} = (\not{D}+M)$. $\Delta S_{\text{eff}} = S_{\text{eff}}(U+\delta U) - S_{\text{eff}}(U)$ and from eq. 1.49 we obtain

$$e^{-\Delta S_{\text{eff}}} = e^{-\Delta S_G} \det\left(1+\mathcal{M}^{-1}\delta\mathcal{M}\right) \tag{1.66}$$

The Scalapino Sugar Algorithm Given initial knowledge of the Green's function $\mathcal{M}^{-1}$ the [Scalapino and Sugar 1981] algorithm updates the inverse in response to every change in the link variables. By rank annihilation

$$(\mathcal{M}+\delta\mathcal{M})^{-1}_{ij} = \mathcal{M}^{-1}_{ij} - \sum_{kl} \frac{\mathcal{M}^{-1}_{ik}\,\delta\mathcal{M}_{kl}\,\mathcal{M}^{-1}_{lj}}{1+\delta\mathcal{M}_{kl}\,\mathcal{M}^{-1}_{lk}} \tag{1.67}$$

The matrix $\delta\mathcal{M}$ has non-zero entries only for sites neighbouring the link being updated. The indices k and l run over these elements, only. To correct for rounding errors in $\mathcal{M}^{-1}$ it is necessary to reset $\mathcal{M}^{-1}\mathcal{M}$ to unity by applying the procedure

$$\mathcal{M}^{-1} \rightarrow 2\mathcal{M}^{-1} - \mathcal{M}^{-1}\mathcal{M}\mathcal{M}^{-1} \tag{1.68}$$

This algorithm is very slow, and can only be used for small lattices It can be speeded up by breaking the lattice up into blocks and only updating sub-matrices of $\mathcal{M}^{-1}$ [Scalapino and Sugar 1981].

Pseudo-Fermions The pseudo-fermion algorithm [Fucito et. al. 1982] uses a Monte Carlo algorithm to evaluate $\mathcal{M}^{-1}$ retaining only terms of order δU. It is very slow at low mass, and the systematic errors introduced necessitate that the changes δU are small.

Block-Lanczos Algorithm The matrix $\mathcal{M}$ has 24 non-zero elements per row or column. Changing a link induces a change $\delta\mathcal{M}$ in $\mathcal{M}$ which is non-zero only at the intersection of the 6 rows and columns of $\mathcal{M}$ corresponding to the end-points of the link. The Lanczos algorithm is used to calculate the 6 columns of $\mathcal{M}^{-1}$ necessary to update the determinant. Given this data [Barbour et. al. 1985c] a multi-hit metropolis update can be performed without further inversions. This idea is intended to permit the updating of all the links within a hypercube. Hypercubes are selected at random and brought into local equilibrium before moving to the next.

Equation of Motion Algorithms The equation of motion algorithms arose from the application of molecular dynamics to field theory simulation [Calaway and Rahman 1982] [Polonyi and Wyld 1983]. A hamiltonian is constructed from the action of the field theory and an artificial dynamics in an extra dimension τ, the simulation time. Hamilton's equations then determine the evolution of the coordinates (degrees of freedom of the field theory) and their conjugate momenta in the extra time dimension.

Dynamical fermions represented by the Grassmann fields ψ and $\bar{\psi}$ are replaced by bosonic fields χ and χ^* with non-local interactions:

$$\begin{aligned} P_S(U) &= \frac{1}{Z}\int[d\bar{\psi}][d\psi]e^{-S(U)-\bar{\psi}\mathcal{M}\psi} \\ &= \frac{1}{Z'}\det(\mathcal{M})e^{-S(U)} \\ &= \frac{1}{Z''}\int[d\chi^*][d\chi]e^{-S(U)-\chi^\dagger(\mathcal{M}^\dagger\mathcal{M})^{-1}\chi} \end{aligned} \tag{1.69}$$

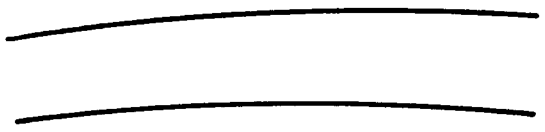
The hermitean matrix $\mathcal{M}^\dagger\mathcal{M}$ is used to ensure convergence of the bosonic Gaussian integrals. The form of the kernel $\mathcal{M}$ for staggered fermions allows us to keep χ fields on even lattice sites only, avoiding the apparent doubling in $\det(\mathcal{M}^\dagger\mathcal{M})$

above. The equations are discretised in τ. Evolving the fermion fields requires calculation of $(\mathcal{M}^\dagger \mathcal{M})^{-1}\chi$ once per timestep. These algorithms are discussed in detail in chapter 4.

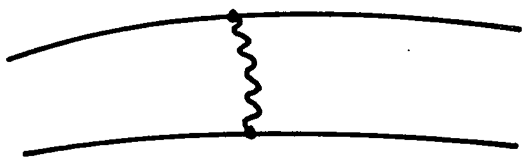
1.2.2 The Quenched Approximation

For a reasonably large lattice (say 16^4) repeated calculation of $\mathcal{M}^{-1}$ or det $(\mathcal{M})$ is out of the question, and the equation of motion schemes would require dedicated use of a supercomputer for several years. What information can we get from a lattice theory without having to commit such resources to a project ?

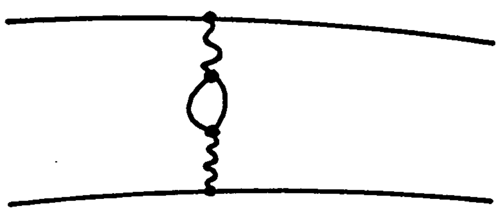
We can describe the fermionic simulation schemes using Feynman diagrams. A propagator in an external field is represented by figure 1.2a. The integrand in



a. The propagator.



b. Adding gluon interactions.



c. Adding dynamical fermion loops.

Figure 1.2: Graphical features of fermion simulations.

the integration over gauge fields can be split into two parts.

$$e^{-S_G(U)} \text{ and } e^{\mathrm{Tr}\ln(\mathcal{M})}$$

The first, the pure gauge term, permits self interaction of the gluons, figure 1.2b. The second allows closed fermion loops in the intermediate states, figure 1.2c. It is this second term that causes most of the problems in the simulation. Dropping it amounts to ignoring fermion loops. This is believed to be reasonable at high quark mass where pair creation is suppressed. Is it reasonable at low masses ?

In string models of mesons the gauge field interaction between the valence quarks is characterised by a string tension T. Including fermions breaks the string, lowering the tension to T'. If these models are appropriate then inclusion of the fermions may be approximately equivalent to adjusting β in the pure gauge theory until $T = T'$.

The empirical Zweig rule [Zweig 1964] states that disconnected quark diagrams are suppressed, it accounts for low cross-sections in $\phi \rightarrow 3\pi$ decays; the ϕ ($\bar{\mathrm{s}}\mathrm{s}$) must go through an intermediary gluonic state to decay into πs ($\bar{\mathrm{u}}\mathrm{d}$, $\mathrm{d}\bar{\mathrm{d}}$ and $\mathrm{d}\bar{\mathrm{u}}$). Dropping the determinant amounts to enforcement of the Zweig rule for all flavours.

Setting the determinant to 1 is called the quenched or valence approximation. It makes fermion simulations possible in reasonable amounts of time (hundreds of hours) on existing supercomputers.

1.3 Parallel Computing

Simulating a lattice gauge theory requires massive computational resources. Current supercomputers (such as the CRAY X-MP, CYBER 205) are powerful enough to perform large scale calculations in the quenched approximation, but they are very expensive to run and few groups can afford sufficient time on them. These machines, vector processors, rely on pipelined floating-point processors and very short cycle times to achieve their speed. They are built from state-of-the-art technology, and have now reached the point where delays in signal propagation across the computer restrict the peak speed.

Parallel computers offer a way out of this 'technology trap'. Their performance is derived from having large numbers of processors co operating on the same task. In general, they are much cheaper to build and run than a vector processor, but more difficult to program. This difficulty is mitigated by the relative simplicity of lattice simulation codes. Much is still to be learnt about how to use these machines effectively, but the rapid growth in their power will make it feasible to perform realistic dynamical fermion simulations in the near future.

1.3.1 Parallel Processor Architectures

There are two basic parallel processor architectures denoted SIMD and MIMD. SIMD (Single Instruction Multiple Data) computers execute the same instruction on every processor at the same time, each applying it to their own data. In an MIMD (Multiple Instruction Multiple Data) computer each processor runs its own program acting on its own data.

In order to cooperate on solving a single problem the processors must be able to communicate with each other. SIMD machines usually have direct serial connections between each processor and its neighbours. The DAP (see appendix A) for example is a 2-d grid of 4096 processors and each has North, South, East and West connections to its neighbours. (There are also row and column broadcast functions.)

There are several different approaches to the connection of processors in MIMD machines (i) buses and (ii) point-to-point communications networks.

Bus Based Machines In a bus based machine all the processors communicate with each other, or with all the memories via a bus - a high bandwidth data path. The bus must be fast enough to service all the processors and memories, and must contain arbitration logic to prevent more than one processor attempting to write to a given memory location at the same time.

Processor Networks A simpler model of parallel processing is that each processor should have its own memory, and should communicate with a small number of other processors via point to point links. The Intel Hypercube, NCube and Meiko Computing Surface are examples of this type of machine. Data in the memory of one processor that is required by another must be passed between them along communications links.

(For a review of the use of supercomputers in QCD computations see [Toussaint 1987].)

1.3.2 Programming Parallel Computers

There are several simple paradigms for exploiting parallelism in a problem and mapping it onto a parallel processor: (i) Job or Event parallelism, (ii) Pipelining and (iii) Geometric Parallelism.

Job Level Parallelism When a program must be run large numbers of times with different parameters it is often most efficient to run these independent tasks concurrently on different processors. This form of parallelism is trivial, but useful. We will assume that it is exploited whenever feasible. It contributes nothing towards solving the problem of distributing a large program.

Pipelining A vector processor exploits parallelism in the floating point multiply function by assigning the various steps to a pipe of processors; each performs a given task on some data and then passes the result on to the next. This approach can be generalised to **algorithmic parallelism** [Askew et. al. 1986] where some number of processors, connected in a network that reflects the structure of the problem, each perform their own task on data fed to them, and then pass the results on. For algorithmic parallelism to be successful the work load must be balanced uniformly across the processors, and there must be sufficient work to do to allow time for the data to be moved from one stage to the next. Processors like the Transputer (see appendix B) make it easy to construct machines dedicated in this way to a specific problem.

Geometric Parallelism A geometric problem decomposition divides up the data amongst the processors so as to preserve *processor data locality*. For example, in dividing up a lattice we allocate some number of sites to each processor in such a way as to ensure that neighbouring sites are always held on the same or neighbouring processors. The success of this approach is dependent upon the algorithm requiring only local operations. Geometric parallelism is the standard way of using an SIMD machine.

1.3.3 Computational Model

We explore the applicability of these models of parallel computation to QCD simulation in chapters 2 and 5. Our computational model is that of a large number of processors with local memory (only) communicating via point-to-point links. In this model, the onus is on the user (rather than a clever compiler) to break up the problem in a suitable fashion.

Criteria for Success Success in this context is considered to be an algorithm and an architecture that can perform a realistic dynamical fermion simulation. To achieve this it will be necessary to use large numbers of processors (whatever their power). The key ingredients for success are that the total useful processing power of the machine should rise linearly (or nearly linearly) with the number of processors and that the slope of the linear rise should be as close to 1 as possible. We define the ratio Γ to be

$$\Gamma = \frac{\text{compute time}}{\text{communication time}} \tag{1.70}$$

The compute time is the time spent doing useful work that has not arisen from the partitioning of the problem. If $\Gamma \ll 1$ then the partitioning is not a success as the computer becomes swamped with messages. If $\Gamma \gg 1$ then the job is computation bound, and could be split across more processors thus reducing the time taken to complete it.

In a processor like the Transputer, computation and communication can go on in parallel. In this case we divide the I/O into two categories, that which can be overlapped with computation, and that which cannot. We seek to mimimize the latter and keep $\Gamma > 1$. In our analysis, extra work that arises from the partitioning of a problem is included with I/O that cannot be overlapped.

General Purpose Parallel Processors Vector machines force us to use vector structures (for high efficiency), but are otherwise general purpose. The processor network architectures allow the construction of application specific configurations, the inter-connection network being set up as appropriate for each problem before the job starts. This gives us the freedom to investigate a variety of schemes for partitioning a problem amongst the processors available, finally selecting the configuration which is most appropriate.

Contents of Thesis

In chapter 2 we develop the basic algorithms required in lattice gauge theory simulation, for calculating sums of oriented plaquettes and solving systems of linear equations. We concentrate on the latter as it dominates, severely restricting what can be achieved. We discuss the parallelism inherent in the problems and its exploitation on the DAP and on arrays of Transputers.

In chapter 3 we present results of hadron mass calculations carried out on the DAP. Mass estimates are presented for a range of β values on 16^4 and $16^3 \times 24$ lattices.

In chapter 4 we review the status of equation of motion simulations and the possibilities for large scale dynamical fermion simulations. We introduce a hybrid monte carlo algorithm suitable for such simulations. We describe an algorithm for simulating quantum electrodynamics on a hypercubic array of processors, implement it on 16 Transputers and use it to compare the efficiency of the Langevin, hybrid and hybrid monte carlo algorithms.

In chapter 5 we study array processor architectures suitable for QCD simulation. We consider a variety of simple geometric arrays and arrays of algorithmic cells.

There are 2 appendices, discussing details of the parallel machines used.

Chapter 2

Basic Algorithms for Lattice Gauge Theory

In this chapter we discuss the basic algorithms for solving the problems set up in section 1.2, calculating sums of plaquettes and solving the lattice Dirac equation. Both calculations are necessary whatever the fermion simulation scheme. We analyse the computational work required and the level of parallelism.

We use the plaquette calculations to illustrate the different forms of parallelism discussed in section 1.3. We then concentrate on the computationally demanding problem of solving the systems of linear equations. We analyse the structure of the matrix operators and their eigenvalue spectrum. We discuss iterative methods of solution and preconditioning in section 2.2.2.

2.1 The Gauge Sector

The Wilson action eq. 1.24 is defined in terms of a sum over plaquettes. Each link is a member of 6 such plaquettes. To calculate the contribution of a link to the action we must trace the sum of these plaquettes. We do this by calculating the 6 "staples" (see figure 2.1), summing them, multiplying in the link to be updated and taking the trace. The staple calculation requires two 3×3 complex matrix-matrix products. We begin by calculating these in the obvious way

$$(a + ib)(c + id) = (ac - bd) + i(ad + bc) \tag{2.1}$$

where a, b, c, d are 3×3 real matrices. This takes 198 floating point operations (108 multiplies and 90 adds). We can reduce the number of operations required by factorising the product

$$(a + ib)(c + id) = ((a + b)(c - d) + ad - bc) + i(ad + bc) \tag{2.2}$$

which requires 180 operations, and replaces multiplications by additions. (see table 2.1 for details. We can reduce the work further by only using two rows of

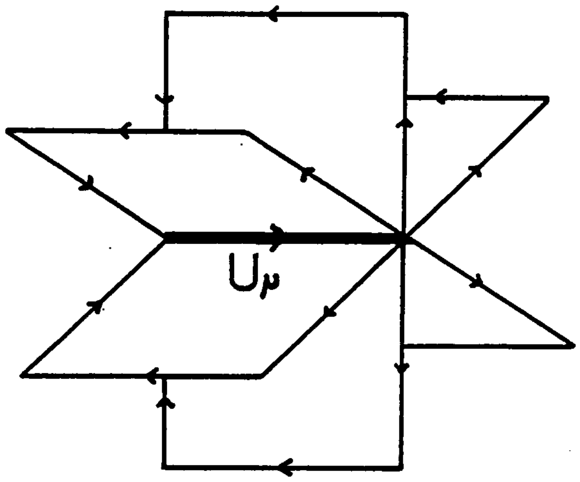


Figure 2.1: Plaquettes including the link $U_\mu(x)$

Computation	*	+
real matrix multiplication	27	18
complex matrix multiplication	108	90
complex matrix multiplication (factorised)	81	99
2 by 3 complex matrix product	54	69
reconstruct third row	24	18
staple	132	156

Table 2.1: Operations required to calculate a staple

the $SU(3)$ matrices, and reconstructing the third.

$$\begin{pmatrix} (a+ib)_{00} & (a+ib)_{01} & (a+ib)_{02} \\ (a+ib)_{10} & (a+ib)_{11} & (a+ib)_{12} \\ & & \end{pmatrix} \bullet \begin{pmatrix} (c+id)_{00} & (c+id)_{01} & (c+id)_{02} \\ (c+id)_{10} & (c+id)_{11} & (c+id)_{12} \\ (c+id)_{20} & (c+id)_{21} & (c+id)_{22} \end{pmatrix} \quad (2.3)$$

Such a 2 by 3 product requires 123 operations (54 multiplies and 69 adds) using a slightly modified version of the factored complex product.

A staple requires 2 such products and the reconstruction of the third row of the result, a total of 288 operations (132 multiplies and 156 additions). We have retained the operation count in terms of additions and multiplies as their relative rate of execution is processor dependent. For example floating-point Transputers perform adds at nearly three times the speed of multiplies (see appendix B). An L^4 lattice has $12L^4$ staples.

The staples for $U_\mu(x)$ involve data only from sites $x, x+\hat{\mu}$ and $x+\hat{\nu}$ for $\hat{\mu}, \hat{\nu} = 0,1,2,3$ and $\mu \neq \nu$. If we divide the lattice sites into two classes of parity "even" and "odd":

for site $x = (x_0, x_1, x_2, x_3)$

$$x \text{ is } \left\{ \begin{matrix} \text{even} \\ \text{odd} \end{matrix} \right\} \text{ if } (-1)^{x_0+x_1+x_2+x_3} = \left\{ \begin{matrix} 1 \\ -1 \end{matrix} \right\} \quad (2.4)$$

then the staple calculations on an even site require data from that site and neighbouring odd sites and vice versa.

The Metropolis algorithm (see section 1.2) requires that we evaluate the change in action resulting from the proposed update of a link. We calculate this by multiplying the sum of staples by the change in the link. Equation of motion schemes simply require the sum of all plaquettes involving a given link (to update the momenta associated with that link).

2.1.1 Parallelism in the Gauge Sector

The Geometric Approach We can (in principle) calculate the 6 staples for a given link direction on all the even (or odd) sites at the same time, $\frac{1}{2}L^4$ fold parallelism. This "parallel update" procedure allows us to update all the links in that direction on all even or odd sites simultaneously, a good example of the geometric parallelism paradigm of section 1.3. This level of parallelism can be exploited on a large processor array such as the DAP (see appendix A), but only for relatively small lattices (up to $8^3 \times 16$).

On a smaller array of more powerful processors, such as Transputers (see appendix B) we exploit geometric parallelism by subdividing the lattice, allocating regions to each processor. We should aim to do this in such a way as to achieve *processor-data locality.* All the data associated with a given lattice site should be stored on a processor and, depending upon its position within the region, data for neighbouring sites should either be on that processor, or on one of its neighbours in the array.

Having partitioned our problem in this way we should aim to transfer all the data required from neighbouring processors while working on the plaquettes that require only local data. For this to be efficient we depend upon *surface-to-volume* effects. There must be sufficient work in the bulk (volume) calculation to overlap the (surface) communication between processors. This idea is illustrated in figure 2.2 for 4×4 regions of a (μ, ν) plane on each processor. All the solid

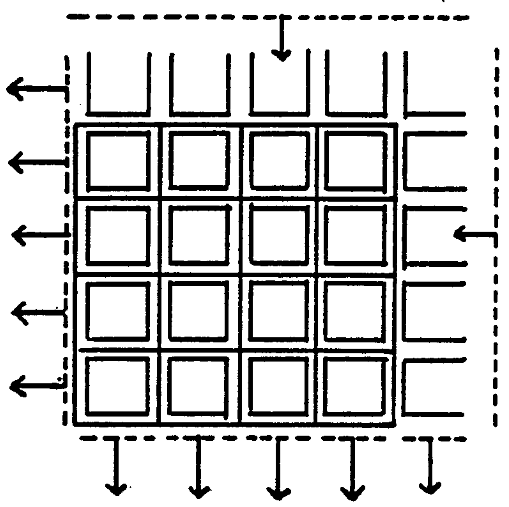


Figure 2.2: Plaquette calculations on an array processor.

plaquettes can be calculated using local data, as can the solid partial plaquettes. The dashed links must be transferred. Only two lattice directions are shown. A practical geometric partitioning must allocate a four dimensional region of the lattice to each processor to preserve the *processor-data locality* property. The drawback with this approach is that for small hypercubic regions there are many more sites in the surface than in the bulk. (In chapter 5 we show this scheme to be efficient and easy to implement.)

Algorithmic Decomposition There are several easily identifiable processes involved in a plaquette calculation: calculating the staples, summing them, multiplying the result by a link, generating random links (in a metropolis or heat-bath simulation), and calculating energies, for example. All of these tasks can be performed in parallel, operating on different data. If we assign each task to a processor and connect them suitably we could build a pipeline specifically for performing plaquette calculations. This approach is an example of algorithmic program decompostion. Transputers allow us the flexibility to do this without going to the lengths and expense of purpose-built hardware. The feasibility of such a scheme, and the problems associated with it are discussed in chapter 5.

2.2 The Fermion Sector

In this section we consider the systems of linear equations that arise in dynamical fermion simulations and quenched calculations of quantities such as hadron masses. We first analyse their structure and sparsity and then their eigenvalue spectra. We then discuss techniques for solving the equations.

2.2.1 The $\not{D}$ Operator

Structure We begin with the staggered fermion action from section 1.1.5

$$S = S_G + \sum_x \bar{\chi} \left(\not{D} + M\right) \chi \tag{2.5}$$

where $M = mI$ and $\not{D}$ is given by

$$\begin{aligned}
\not{D}_{x,y} &= \frac{1}{2}\eta_0(x)\left(U_0(x)\delta_{x_0+1,y_0} - U_0^\dagger(y)\delta_{x_0-1,y_0}\right)\delta_{x_1,y_1}\delta_{x_2,y_2}\delta_{x_3,y_3} \\
&+ \frac{1}{2}\eta_1(x)\left(U_1(x)\delta_{x_1+1,y_1} - U_1^\dagger(y)\delta_{x_1-1,y_1}\right)\delta_{x_0,y_0}\delta_{x_2,y_2}\delta_{x_3,y_3} \\
&+ \frac{1}{2}\eta_2(x)\left(U_2(x)\delta_{x_2+1,y_2} - U_2^\dagger(y)\delta_{x_2-1,y_2}\right)\delta_{x_0,y_0}\delta_{x_1,y_1}\delta_{x_3,y_3} \\
&+ \frac{1}{2}\eta_3(x)\left(U_3(x)\delta_{x_3+1,y_3} - U_3^\dagger(y)\delta_{x_3-1,y_3}\right)\delta_{x_0,y_0}\delta_{x_1,y_1}\delta_{x_2,y_2}
\end{aligned} \tag{2.6}$$

In eq. 2.6 and what follows we have set $a = 1$. As usual the U's are 3×3 complex unitary matrices, elements of the gauge group SU(3). To write this as a matrix we introduce the following (arbitrary) ordering of sites in the lattice

$$x_{id} = x_0 N^3 + x_1 N^2 + x_2 N + x_3 \tag{2.7}$$

$\not{D}$ is a complex matrix of size l^2 where $l = 3L^4$, for a lattice of size L^4, it has 8 entries (U's) per row for $L > 3$. The action of $\not{D}$ on a vector χ (ordered similarly) is given by

$$\left(\not{D}\chi\right)(x) = \frac{1}{2}\sum_\mu \eta_\mu(x)\left\{U_\mu(x)\,\chi(x+\hat{\mu}) - U_\mu^\dagger(x-\hat{\mu})\chi(x-\hat{\mu})\right\} \tag{2.8}$$

We will use this as our working definition of the matrix operator $\not{D}$. From eq. 2.8 we see that $\not{D}^\dagger = -\not{D}$.

The operator $\not{D}$ connects a site on the lattice to each of its neighbours, but not to itself. If we use the definition of odd and even sites given in section 2.1 then the action of $\not{D}$ on a vector with non-zero entries on the even sites only gives a vector with non-zero entries on the odd sites and zeros on the even sites, and vice versa.

The Equations In calculating hadron masses we must measure expectation values such as

$$\langle \chi(o)\bar{\chi}(x)\rangle = (\not{D}+M)^{-1}_{x,o} \tag{2.9}$$

the quark propagator from an origin o to all sites x in the lattice (see section 1.1.6 and chapter 3). This requires the solution of

$$(\not{D}+M)^{AC}_{x,y}\chi^{C}_{y} = \delta_{x,o}\delta^{AB} \tag{2.10}$$

for many configurations $\{U(x)\}$. The indices A, B, C refer to colour. We must solve 3 systems of linear equations of size l^2, one for each value of the initial colour A The "odd-even" structure of the matrix $\not{D}$ allows us to decompose eq. 2.10 as follows

$$\begin{aligned} \not{D}x^{odd} + Mx^{even} &= \delta^{even} \\ \not{D}x^{even} + Mx^{odd} &= \delta^{odd} \end{aligned} \tag{2.11}$$

where the vector x^{even} is zero on all the odd sites, and likewise x^{odd} is zero on all the even sites. If we restrict our source to lie on, say, an even site. This gives

$$\left(-\not{D}^2+M^2\right)\frac{x}{m}^{\text{even}} = \delta^{even} \quad \text{and} \quad x^{odd} = -\not{D}\frac{x}{m}^{\text{even}} \tag{2.12}$$

This decoupling of the two systems halves the number of equations to be solved; it can be extended to allow a general source term. The matrix $(-\not{D}^2+M^2)$ is hermitean, and still very sparse.

In dynamical fermion simulations (see section 1.2.1 and chapter 4) we have to solve

$$\left(-\not{D}^2+M^2\right)\chi = \psi \tag{2.13}$$

for χ, where the bosonised fermion field ψ lives on half the sites of the lattice. Given the structure of $(-\not{D}^2+M^2)$ it is natural to put the ψ fields on even sites as defined in eq. 2.4 .

Eigenvalue Spectrum The matrix $\not{D}$ is anti-hermitean. It thus has pure imaginary eigenvalues. Let them be $i\lambda_k, k = 1, 2, \ldots, l$ for real λ_k. Let λ_{min} be the smallest λ_i and λ_{max} the largest. Detailed information about the eigenvalues of $\not{D}$ and their dependence upon the gauge coupling is very difficult to obtain. It amounts to a complete solution of the problem. However, we can extract gross features of the spectrum.

We first use Gershgorin's circle theorem (see [Golub and Van Loan 1983]) to set an upper bound on the spectrum $\lambda_u > \lambda_{\text{max}}$

Theorem 2.1 (Gershgorin) *If $X^{-1}AX = \text{diag}(d_1, \ldots, d_n) + F$ and F has no diagonal entries then the eigenvalues of A are bounded by*

$$\lambda(A) \subset \bigcup_{i=1}^{n} D_i$$

where

$$D_i = \left\{ z \in C : |z - d_i| \leq \sum_{i=1}^{n} |f_{ij}| \right\}$$

"all eigenvalues lie within the union of circles".

Consider the free case of eq. 2.6 first. The U's are now 3×3 identity matrices, and so each row of $\not{D}_F$ has 8 entries each of magnitude $\frac{1}{2}$. Hence $\lambda_u = 4$. For the interacting theory there are 24 non-zero elements per row (8 link matrices). The constraint that each matrix is a member of the group $SU(3)$ allows us to infer that the contribution to $\sum |f_{ij}|$ from each matrix is at most $\frac{1}{2}\sqrt{3}$, hence $4 \leq \lambda_u \leq 4\sqrt{3}$. The actual value of $\lambda_{\max}$ will depend upon the lattice spacing which cuts off the high momentum modes.

A useful lower bound $\lambda_b \leq \lambda_{\min}$ on the spectrum of $\not{D}$ is more difficult to obtain as it will depend (amongst other things) upon the size of the lattice, which restricts the low momentum modes. We can obtain a bound on the eigenvalues of $(-\not{D}^2 + M^2)$, the matrix we will need to use. Consider the eigenvalues λ' of $\not{D} + M$. $\lambda' = m \pm i\lambda$ and so the eigenvalues λ'' of $(-\not{D}^2 + M^2)$ are $\lambda'' = m^2 + \lambda^2$. λ is real, therefore $\lambda_b'' \geq m^2$. For large m this lower bound will be correct, for small m, λ_b'' will be controlled by the finite lattice size.

A Numerical Spectrum Calculation We have studied the spectrum of $\not{D}$ as part of a calculation of $\langle \bar{\psi}\psi \rangle$ [Barbour et. al. 1985b] on an 8^4 lattice. We used a Lanczos algorithm[1] to tri-diagonalise $\not{D}$ and the method of Sturm sequences (guided using Gershgorin's theorem) to extract the extremal eigenvalues. Figure 2.3 shows the first 100 eigenvalues to converge for a configuration at $\beta = 5.7$. Convergence depends strongly upon eigenvalue density. At low β (strong coupling) the density of low eigenvalues is higher, and the rate of convergence lower than at intermediate couplings. We see that $\lambda_u \simeq 4.2$. The leading eigenvalues $\lambda_{\min}$ were found to vary over at least an order of magnitude from configuration to configuration. Figure 2.4 shows the distribution of λ_1 for 32 configurations at $\beta = 5.7$.

2.2.2 Solving Systems of Linear Equations

There are two approaches to the solution of systems of linear equations (i) exact and (ii) iterative. Exact methods such as LDU decomposition and Gaussian elimination (with/without pivoting) are less suited to banded sparse matrix problems as "fill in" occurs; zero entries in the original matrix acquire non-zero values in intermediary working matrices. Much effort has been devoted to the problem of

[1]It is well known (at least in the numerical analysis literature) that the Conjugate Gradient and Lanczos algorithms are similar. A description of how to use Conjugate Gradient to tri-diagonalise a matrix is given in [Golub and Van Loan 1983].

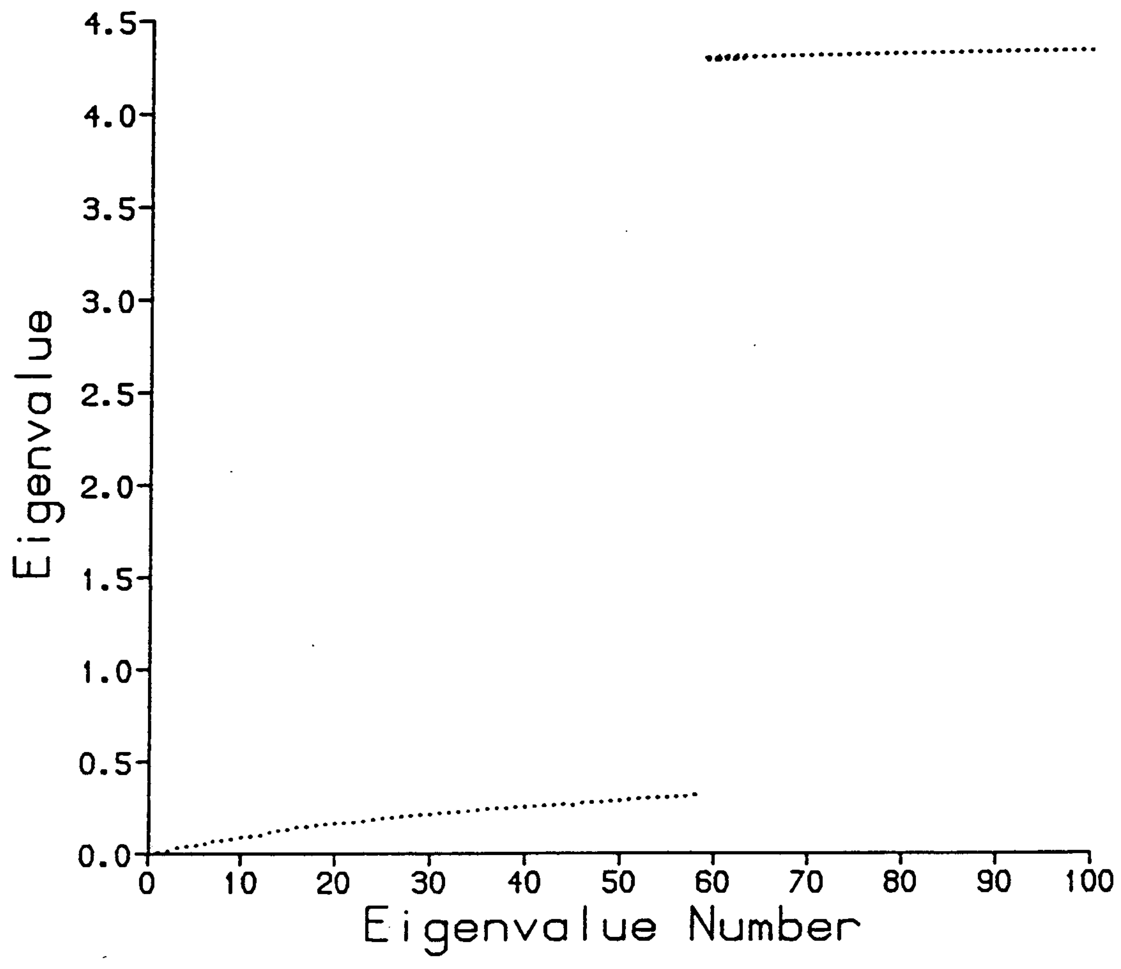


Figure 2.3: The first 100 eigenvalues to converge for an 8^4 lattice at $\beta = 5.7$.

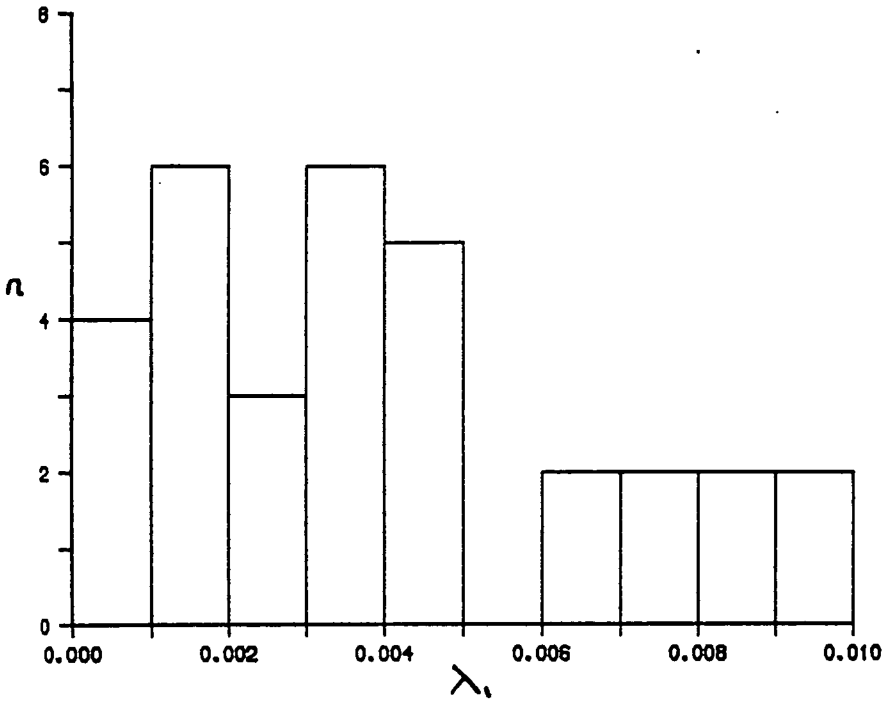


Figure 2.4: Distribution of λ_1 for 32 configuration at $\beta = 5.7$.

minimising fill in for particular matrices, but we cannot afford any. Iterative techniques typically require repeated matrix vector products, which we can perform quickly for sparse matrices such as $\not{D}$. A further advantage of iterative methods is that they can be used to improve an estimate of the solution. In addition we can tune the solver to achieve a particular accuracy - this is not possible with exact methods.

Of the standard iteration schemes, Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR) we only considered using SOR. The others are prohibitively slow unless the matrix is diagonally dominant (this is the case for large m only). To solve $Ax = b$:

Algorithm 2.2 (SOR) *For $i = 1, \ldots, n$ set*

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) + (1-\omega) x_i^{(k)}$$

Repeat for $k = 1, 2, \ldots$. In matrix form the SOR step is given by

$$M_\omega x^{(k+1)} = N_\omega x^{(k)} + \omega b$$

where $M_\omega = D + \omega L$ and $N_\omega = (1-\omega)D - \omega U$, L, D, U being the lower triangular, diagonal, and upper triangular parts of A.

The relaxation parameter ω must be tuned for optimum convergence. We may be able to tune ω by hand under certain circumstances. For this reason SOR is not suitable for problems such as dynamical fermion simulations where the matrix is varying with time. It has been used in quenched quark propagator calculations [Marinari et. al. 1981]. We describe an iterative block SOR algorithm in section 2.2.3. Convergence is slow for low m, when the matrix is not diagonally dominant.

The Conjugate Gradient Algorithm.

The Conjugate Gradient (CG) algorithm [Hestenes and Steifel 1952], was introduced as an exact method for solving systems of linear equations, but is now established as an iterative technique for solving large sparse systems [Reid 1971]. It has the advantage that there are no parameters to tune. See [Golub and Van Loan 1983] and [Concus, Golub and O'Leary 1976] for a thorough treatment of the properties of CG. The algorithm is as follows:

Algorithm 2.3 (Conjugate Gradient) *To solve $Ax = b$ for hermitean A*

```
initial guess $x_0$
$p_0 = r_0 = b - Ax_0$
loop while $r_k^\dagger r_k > \epsilon$ for $k = 0, 1, 2, \ldots$
  $\alpha_k = \frac{r_k^\dagger r_k}{p_k^\dagger A p_k}$
  $x_{k+1} = x_k + \alpha_k p_k$
  $r_{k+1} = r_k - \alpha_k A p_k$
  $\beta_k = \frac{r_{k+1}^\dagger r_{k+1}}{r_k^\dagger r_k}$
  $p_{k+1} = r_{k+1} + \beta_k p_k$
end loop
```

The CG algorithm is known to converge best for matrices with clustered eigenvalues and low condition number $\mathcal{K}(A)$. For hermitean positive definite A

$$\mathcal{K}(A) = \frac{\lambda_{\max}}{\lambda_{\min}} \tag{2.14}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of A. An upper bound on the error $e_k = |Ax_k - b|$ at step k is given by

$$e_k = \left(\frac{\sqrt{\mathcal{K}} - 1}{\sqrt{\mathcal{K}} + 1}\right)^k e_0 \tag{2.15}$$

Using the bounds on the spectrum of $\not{D}$ in section 2.2.1 we can estimate $\mathcal{K}(A)$. Provided m is not too small $\lambda''_{\min} \simeq m^2$ and $(4 + m)^2 \leq \lambda''_{\max} \leq (6.9 + m)^2$ Let $\tilde{\lambda}$

be an estimate for $\lambda''_{\max}$ in this range. The change in the error over k iterations (for k large) is given by

$$\frac{e_k}{e_0} = \left(\frac{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} - 1}{\sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} + 1}\right)^k \sim \left(\frac{1}{1+\frac{m}{\tilde{\lambda}}}\right)^k = \left(1 - \frac{m}{\tilde{\lambda}} + \cdots\right)^k \tag{2.16}$$

and so

$$\ln\left(\frac{e_k}{e_0}\right) \sim \frac{-mk}{\tilde{\lambda}} + \cdots \tag{2.17}$$

If $m \ll \lambda''_{\min}$ then the convergence rate will be higher, but will be controlled by $\lambda''_{\min}$ which is configuration dependent, and will fluctuate accordingly.

Eq. 2.16 gives an upper bound on the rate of convergence of CG. We should also minimise the initial error e_0 by starting with as good a guess x_0 as we have available. In hadron mass calculations it is usual to run at a range of masses, feeding the answer from one run in as the starting point for the next. In dynamical fermion simulations, where we solve the system of equations repeatedly (making $O(dt)$ changes to the coefficients of the matrix and elements of the source each time) we can feed in the previous solution as a starting point. Higher order schemes have been suggested [Gottlieb et al 1987], and are discussed in section 4.6.

Test Implementation of CG As a testbed for the CG algorithm and the preconditioning schemes we wrote a serial CG code to solve eq. 2.10 for a small lattice (usually 4^4). Its rate of convergence as a function of mass is shown in figure 2.5. The leveling off of the rate of convergence as $m \to 0$ is a sign of finite-size effects in the system. The size of the lattice imposes a lower bound on the lowest eigenvalue of $\not{D}$.

Parallelism in the CG Algorithm The CG algorithm requires 2 matrix-vector products (applications of $\not{D}$), 2 scalar products, and 3 vector operations of the form vector:=vector+(scalar*vector) per iteration. The matrix-vector products dominate. The relative importance of matrix-vector and scalar products on an array processor are discussed in chapter 5. There is a very high degree of parallelism in the application of $\not{D}$. To illustrate this we will set the problem up for an array of processors of the same size as the lattice, one processor per site. A more realistic partition will be discussed later.

The operator $\not{D}$ is local, to apply it to a vector for a given site x and direction $\hat{\mu}$ we require only the vector data from $x+\hat{\mu}$ and $x-\hat{\mu}$.

$$(\not{D}\chi)(x) = \frac{1}{2}\sum_{\mu} \eta_\mu(x) \left\{ \underbrace{U_\mu(x)\,\chi(x+\hat{\mu})}_{1} - \underbrace{U^\dagger_\mu(x-\hat{\mu})\chi(x-\hat{\mu})}_{2} \right\} \tag{2.18}$$

In calculating $\not{D}\chi$ we move the (smaller) vector rather than the links. When this is complete we can calculate either term for all even (or odd) sites simultaneously.

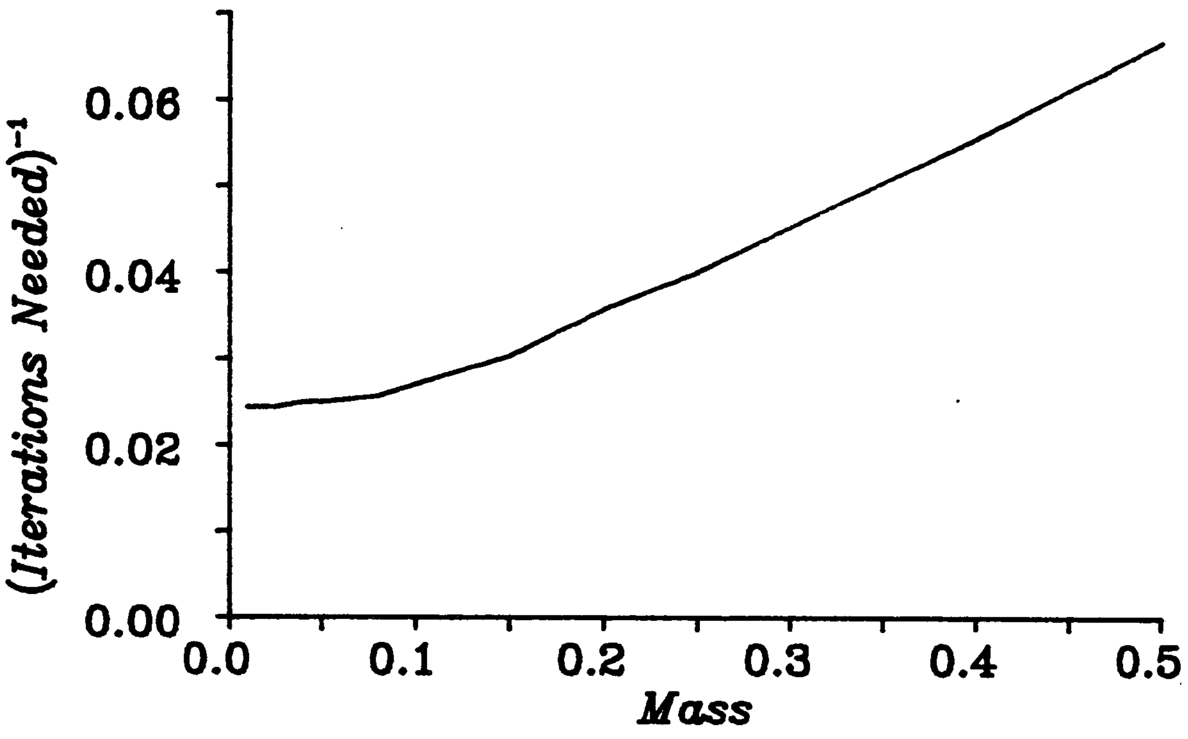


Figure 2.5: Convergence of test CG code as a function of mass for a 4^4 system.

This will use half of the processors. We can improve upon this, the work for term 1 naturally takes place on processors with x parity, and that for term 2 on $1-(x \text{ parity})$. We can calculate both terms in parallel by shifting the vector data for term 1, performing the matrix-vector products, shifting the results for term 2 (onto sites with x parity) and adding. We achieve L^4 fold parallelism. This is illustrated in figure 2.6. The idea of using one processor per site is only feasible

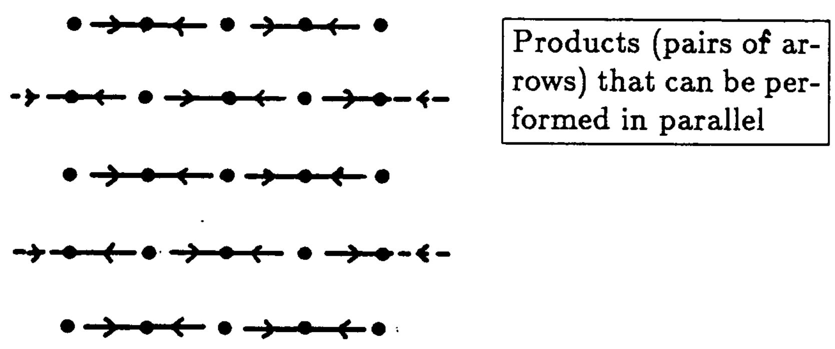

Products (pairs of arrows) that can be performed in parallel

Figure 2.6: Parallelism in application of the $\not{D}$ operator.

on a large array processor such as the DAP (and here only for lattices up to 8^4). But in the $16^3 \times 24$ implementation described next we pack a 16^3 timeslice into one plane of the DAP, and fully exploit the 4096-fold parallelism.

For a smaller array of more powerful processors we would allocate some number of sites to each processor and work on "interior" products while transferring the data for "boundary" products (as outlined earlier). There is a 4-fold algorithmic parallelism in this calculation: performing the finite differences in each of the μ directions. This and the details of exploiting the geometric parallelism on a Transputer array are discussed in chapter 5.

Implementation of CG on the DAP This code was used for calculating the quark propagators needed for our first hadron mass calculations on 16^4 lattices (see chapter 3). It converged smoothly, $ln r_k^\dagger r_k$ being approximately proportional to m in the range 0.01 to 0.50 . The departure of r_k, the iterative residual vector, from $b - Ax_k$ is a sign of the onset of roundoff errors in the CG algorithm; the importance of such errors can be judged by restarting the system using x_k as the new x_0. We see a marked increase in the residual on restarting after 700 CG iterations (see table 2.3.) in 32-bit arithmetic at m=0.01, indicating that roundoff effects have become significant. We restart the solver after 500 iterations at the lowest mass, and run for a further 200 iterations, by which time the desired accuracy is attained. We use the following criteria to decide how long to run the CG solver:-

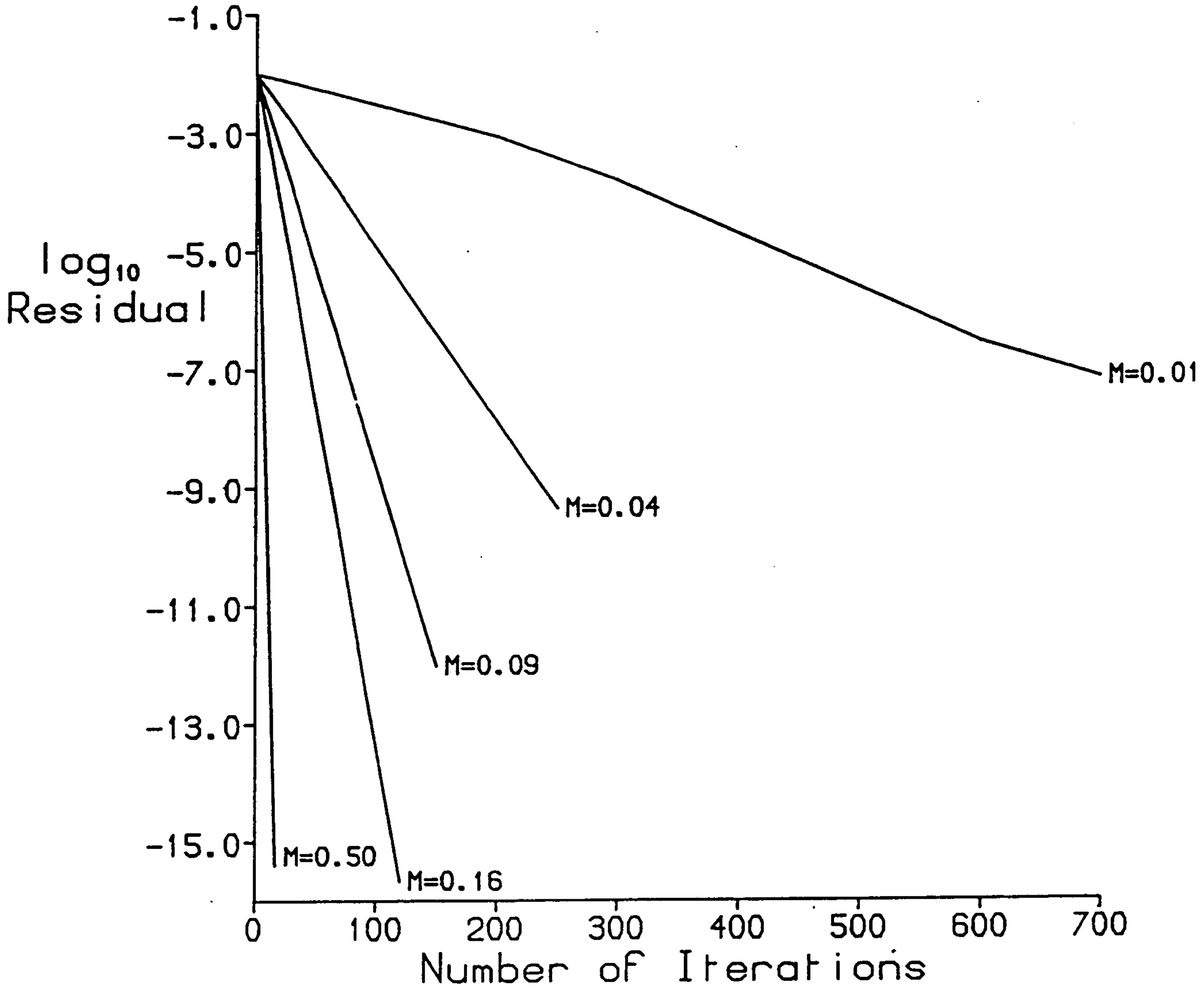


Figure 2.7: Convergence of CG algorithm for 16^4 lattices.

1. That baryon propagators (sums over space and colour of x^3) should be unchanged in the third decimal place on all timeslices under further iteration.

2. That on restarting, the baryon propagators should not change and the norm of the residual vector should not increase significantly.

3. That the baryon propagators obtained should be the same (to the above tolerance) as those obtained from the same configuration after it has undergone a random gauge transformation. (The baryon propagators are independent of choice of gauge, but the quark propagators are not). We performed this test on our first configuration to check the program.

The values of $r_k^\dagger r_k$ sufficient to satisfy these conditions for m in the range 0.01 to 0.5 are given in table 2.2. We apply the above conditions on the last timeslice; closer to the origin the baryon propagators are stable in the 4th and 5th decimal places.

	mass				
β	0.01	0.04	0.09	0.16	0.50
5.7	$500+200$	300	150	120	40
6.0	$500+200$	250	120	100	40
$\log_{10}(r^\dagger r)$	-8	-9	-12	-15	-15

Table 2.2: Number of iterations necessary to meet convergence and stopping conditions for CG algorithm on a 16^4 lattice, and approximate values of the residuals obtained.

iterations	$r^\dagger r$	$r^\dagger r$ on restart
100	0.28E-2	
200	0.32E-3	
300	0.41E-4	
400	0.75E-5	
500	0.77E-6	0.78E-6
600	0.90E-7	
700	0.86E-8	0.15E-7

Table 2.3: Measurements of the residual for CG algorithm on a 16^4 lattice, $m = 0.01$, $\beta = 6.0$.

The CG algorithm performs well, but is expensive in terms of storage. We need to store 3 vectors per colour, each of size $3L^4$ words, which together with the links must be paged through the machine on each iteration as the system is too big to be held in memory. This disk-to-DAP paging of timeslices is done asynchronously, the links for timeslice $(t+1)$ being paged on while those for timeslice t are being used. This is done using the DAP data expansion software, DDX (see appendix A) which enables variables held in COMMON areas to be transferred efficiently between DAP and disk while a program is running.

The CG vectors can only be updated when calculation of the scalars α and β is complete. [Barkai et. al. 1985] have proposed a modified CG algorithm which avoids some of the synchronisation problems at the expense of an extra vector. However, because of the DAP's low I/O rate (approximately 250-300 Kbytes/sec compared to a floating-point performance of around 15 Mflops) the matrix-vector multiply step is I/O bound by itself, and so we use the standard algorithm for a hermitean matrix. The total connect time for a propagator calculation running

Step	cpu	I/O	I/O overhead	Total
Calculate Ap_k	7.2	15	10	17.2
Update vectors	0.6	30	30	30.6
Dot products	0.2	0	0	0.2
Totals	8	45	40	48

Table 2.4: Approximate timing data (in seconds) for 16^4 CG algorithm on the DAP.

on the DAP is a factor of 6 longer than the processor time used. Similar I/O stretch factors are reported in [Barkai et. al. 1985] for a Cyber 205. Details of the timings are given in table 2.4. Almost all the DAP cpu time is used applying the $\not{D}$ operators. Cpu time for updating the vectors is insignificant. The 10 seconds of I/O overhead that occur while the links are being paged in and the $\not{D}$ operators applied is not available. But during the longer (30 second) overhead, while the vectors are paged on and off the machine, we could do useful work.

If all the data required can be held in fast memory then CG is an efficient algorithm. If large quantities of data must be paged between slow and fast memory then there will be huge I/O overheads unless the transfer rate is very high. Under these circumstances it is necessary to use an algorithm with much more computation per word of data transfer. Such an algorithm, Iterative Block SOR is described in section 2.2.3. In dynamical fermion simulations the systems of equations are (regrettably) much smaller, and the data can usually be stored in fast memory.

Preconditioning

The standard iterative inversion algorithm in many fields is the "preconditioned" conjugate gradient algorithm. All the information available about A^{-1} is used to construct N^{-1}, an approximation to A^{-1}. We then solve

$$N^{-1}Ax = N^{-1}b \tag{2.19}$$

instead of $Ax = b$. If N^{-1} is a good approximation to A^{-1} then $\mathcal{K}(N^{-1}A) \ll \mathcal{K}(A)$ and the scheme will converge more rapidly. Iterations of the preconditioned algorithm are more expensive, but significant savings in the number of iterations can be made. The choice of preconditioner N is vital to the success of the technique.

We have considered the following preconditioning schemes: (i) diagonal and block diagonal scaling, (ii) free fermion preconditioning, using the free fermion version of $(\not{D}+M)$ as an approximation to the full $(\not{D}+M)$, (iii) incomplete LDU or Cholesky preconditioning.

Diagonal Scaling Diagonal scaling, dividing through by the diagonal of the matrix A, is generally regarded as vital when the elements of A vary greatly in magnitude. The non-zero elements of $(\not{D}+M)$ and $(-\not{D}^2+M^2)$ are all of $O(1)$. Dividing through by the diagonal (a constant) does not improve our rates of convergence.

Block diagonal scaling requires that we left-multiply $Ax = b$ by W^{-1} where W is some block-diagonal approximation to A. The matrices $(\not{D}+M)$ and $(-\not{D}^2+M^2)$ have block structure, their entries are 3 by 3 complex unitary matrices, but the block diagonal is a multiple of the identity (block diagonal scaling is the same as diagonal scaling for these blocks).

If we single out a direction in the lattice, the time direction say, and work in temporal gauge (all time-like links rotated to the identity) then for Dirichlet boundary conditions in time we can write

$$(\not{D}+M) = \begin{pmatrix} \not{D}_1+M & T & & & & \\ -T & \not{D}_2+M & T & & & \\ & -T & \not{D}_3+M & T & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & -T & \not{D}_{N-1}+M & T \\ & & & & -T & \not{D}_N+M \end{pmatrix} \qquad (2.20)$$

where $\not{D}_t+M$ is the three (spatial) dimensional equivalent of $\not{D}+M$ for timeslice t and $T = \frac{1}{2}(-1)^{x_0+x_1+x_2+x_3}I$. A similar partitioning of $(-\not{D}^2+M^2)$ is discussed in section 2.2.3. If we use this (or any similar blocking scheme) we reveal a block-diagonal structure, but the matrices $(\not{D}_t+M)^{-1}$ are fully dense, just like $(-\not{D}^2+M^2)^{-1}$. This makes calculation and storage of the the diagonal inverses prohibitively expensive. Such block diagonal scaling schemes are not practical.

Given the full density of all inverses of interest we are forced to consider a modified form of the preconditioned CG [Concus, Golub and O'Leary 1976].

Algorithm 2.4 (Modified Preconditioned CG)

initial guess x_0

```
$p_0 = r_0 = b - Ax_0$
solve $Nd_0 = r_0$
loop while $r_k^\dagger d_k > \epsilon$ for $k = 0, 1, 2, \ldots$
    $\alpha_k = \frac{r_k^\dagger d_k}{p_k^\dagger A p_k}$
    $x_{k+1} = x_k + \alpha_k p_k$
    $r_{k+1} = r_k - \alpha_k A p_k$
    solve $Nd_{k+1} = r_{k+1}$
    $\beta_k = \frac{r_{k+1}^\dagger d_{k+1}}{r_k^\dagger d_k}$
    $p_{k+1} = d_{k+1} + \beta_k p_k$
end loop
```

where the solution of $Nd_k = r_k$ is simple compared with solution of $Ax = b$. Block-diagonal scaling can be used in this form, but it is prohibitively expensive; it requires N solutions of the 3 dimensional problem per iteration. We now describe a variety of preconditioning schemes of this form, that do at least seem possibilities.

Free Fermion Preconditioning We know that at short distances (within a proton for example) quarks behave as free particles since QCD is an asymptotically free theory. Consequently, free propagation becomes a better approximation as the lattice spacing decreases, which will be the case for simulations using larger lattices. Further, we know that the iterative solution of the system of equations

$$\left(-\not{D}_F^2 + M_F^2\right) x = b \tag{2.21}$$

eq. 2.12 for free fermions of mass m_F, converges very rapidly (the residual squared $r_k^\dagger r_k$ is reduced from ~ 1 to $\sim 10^{-16}$ by 15 iterations of a CG algorithm). So we would expect this preconditioner to work better the larger and computationally more demanding the task. There is a problem, however, in that within a gauge theory there is no clear distinction between high and low momentum modes unless we fix a gauge. The quark propagator is gauge dependent and will only resemble a free propagator in a gauge where the link variables are chosen to be as close to the unit matrix as possible. We tried using axial gauge in which all the timelike links and as many of the spacelike links as possible are rotated to the identity (this is possible with Dirichlet boundary condirions on the fermions). We did not succeeded in reducing the total time necessary to solve the system of equations.

The group at Cornell University [Batrouni et. al. 1985] report some success using Landau gauge, and storing an approximation to the diagonal of $\not{D} + M$ in momentum space as a preconditioner, translating between momentum space and configuration space using fast fourier transforms (FFT's). They use Wilson fermions. A colleague [Sheard] tried this for staggered fermions, but without success. The number of CG iterations required was generally higher, the iterations took longer, and the iterative gauge transformation procedure necessary to fix Landau gauge is expensive.

LDU Decomposition Systems such as ours can (in principle) be solved exactly by LDU decomposition. $A = LDU$ for lower triangular L, upper triangular U and diagonal D. Cholesky factorisation is a special case of this for hermitean A. This is totally impractical because L and U are fully dense. However, we can use an approximate (or incomplete) LDU decomposition as a preconditioner (This was first done by [Oyanagi 1986] for Wilson Fermions). Here we write

$$A = LDU + E = N + E \tag{2.22}$$

E is an error matrix whose effect we should seek to minimize. In an incomplete decomposition one only keeps the elements of L and U that are non-zero elements of A. Calculation of the factors L and U for ($\not{D}$+M) is made easy by this sparsity condition and the following property of a hypercube lattice (of extent greater than 3). *If sites i and j are neighbours, and sites i and k are neighbours then j is not a neighbour of k.*

This method has been used successfully for Wilson fermions [Oyanagi 1986], but has not proved useful with staggered fermions.

If a good preconditioner can be found for staggered fermions it will greatly reduce the time required to do dynamical fermion simulations. This work is being pursued by several groups, largely without success.

2.2.3 Block Iterative Methods

Consider the partitioning scheme eq. 2.20 for $\not{D}$+M in temporal gauge. We can define a class of Block-Iterative algorithms by regarding each $3N^3 \times 3N^3$ complex block as a component of an $N \times N$ linear system of equations, applying an established iterative algorithm (e.g Jacobi, Gauss Seidel or Successive Over-Relaxation) to the $N \times N$ block equation and using an inner loop to solve the reduced rank systems of equations. Consider the standard SOR scheme (algorithm 2.2) and use it to solve the N by N system of block equations eq. 2.23

$$\begin{pmatrix} \not{D}_1+M & T & & & & \\ -T & \not{D}_2+M & T & & & \\ & -T & \not{D}_3+M & T & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & -T & \not{D}_{N-1}+M & T \\ & & & & -T & \not{D}_N+M \end{pmatrix} \bullet \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ \cdots \\ x_{N-1} \\ x_N \end{pmatrix} = \begin{pmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \cdots \\ \cdots \\ \delta_{N-1} \\ \delta_N \end{pmatrix} \tag{2.23}$$

where $\not{D}_t$ is anti-hermitean like $\not{D}$. We the perform the SOR iteration

$$(\not{D}_t+M)\, x_t^{(k+1)} = \omega\left(\delta_t + Tx_{t-1}^{(k+1)} - Tx_{t+1}^{(k-1)}\right) + (1-\omega)\,(\not{D}_t+M)\, x_t^k \tag{2.24}$$

using an inner CG solver to obtain x_t on each iteration k of the outer SOR scheme. The parameter ω is selected for optimum convergence. The scheme can be extended to solve eq. 2.12 by squaring the partitioned $\not{D}$ operator and adding in the diagonal mass term

$$\not{D}^2 = \begin{pmatrix} \frac{1}{4}I-\not{D}_1^2+M^2 & (\not{D}_2-\not{D}_1)T & -\frac{1}{4}I & & & \\ (\not{D}_2-\not{D}_1)T & \frac{1}{2}I-\not{D}_2^2+M^2 & & & & \\ -\frac{1}{4}I & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & -\frac{1}{4}I \\ & & & \ddots & \frac{1}{2}I-\not{D}_{N-1}^2+M^2 & (\not{D}_N-\not{D}_{N-1})T \\ & & & -\frac{1}{4}I & (\not{D}_N-\not{D}_{N-1})T & \frac{1}{4}I-\not{D}_N^2+M^2 \end{pmatrix} \qquad (2.25)$$

(this result uses the fact that T and $\not{D}$ anti-commute). We then have to solve eq. 2.26 repeatedly.

$$\begin{aligned} \left(cI - \not{D}_t^2 + M^2\right) y_t^{(k+1)} &= \beta_t \\ &= \omega\left(\delta_t + \frac{1}{4}y_{t-2}^{(k+1)} + (\not{D}_{t-1} - \not{D}_t)Ty_{t-1}^{(k+1)}\right. \qquad (2.26) \\ &\quad \left. +(\not{D}_{t-1} - \not{D}_t)Ty_{t+1}^{(k)} + \frac{1}{4}y_{t+2}^{(k)}\right) \\ &\quad +(1-\omega)\left(cI - \not{D}_t^2 + M^2\right) y_t^{(k)} \end{aligned}$$

where $y_t = x_t^{even}$. The method can be used for any choice of temporal fermion boundary conditions eg: For periodic bcs $c = \frac{1}{2}$ and y_1 is identified with y_{N+1}. For Dirichlet bcs $c = \frac{1}{4}$ for y_1 and y_N and $c = \frac{1}{2}$ for y_t with t between 1 and N. Terms on the RHS of eq. 2.26 with $t < 1$ or $t > N$ are dropped.

$\not{D}_t^2$ is hermitean, c and m are real, and so a standard CG algorithm can be used to solve the inner systems of equations. The matrix operator $cI - \not{D}_t^2 + M^2$ is diagonally dominant (its diagonal elements being $\frac{3}{2}$+c+m^2); further, this diagonal dominance remains as $m \to 0$, and so convergence of the inner CG algorithm is very rapid for all m as shown in figure 2.8. We call this scheme IBSOR (Iterative Block SOR).

It would appear that IBSOR requires a huge amount of work, solving N systems of size $3N^3$ per sweep, but this is not the case. β_t (the RHS of eq. 2.26) is accumulated from terms on 5 timeslices and if we assume that the algorithm converges then the error on 3 of these timeslices y_t, y_{t+1}, y_{t+2} is greater than that on the other two y_{t-1} and y_{t-2}. So we should aim to converge the $(k+1)$'th iterate on timeslice t to a level where the residual on this timeslice is some factor lower than that on $(t-1)$. We should not run the inversion so long that we do work which will be wasted on the next sweep, when y_{t+1} and y_{t+2} have also been upgraded. This fits in with our aim of producing a balanced or cpu-dominated program which will be achieved if we can do the necessary iterations in the time taken to page out y_{t-2} and page in y_{t+3} and the next set of links. This proves to be the case.

Tuning the IBSOR algorithm The SOR algorithm does not converge for all values of its parameter ω. The range of acceptable values is determined by the

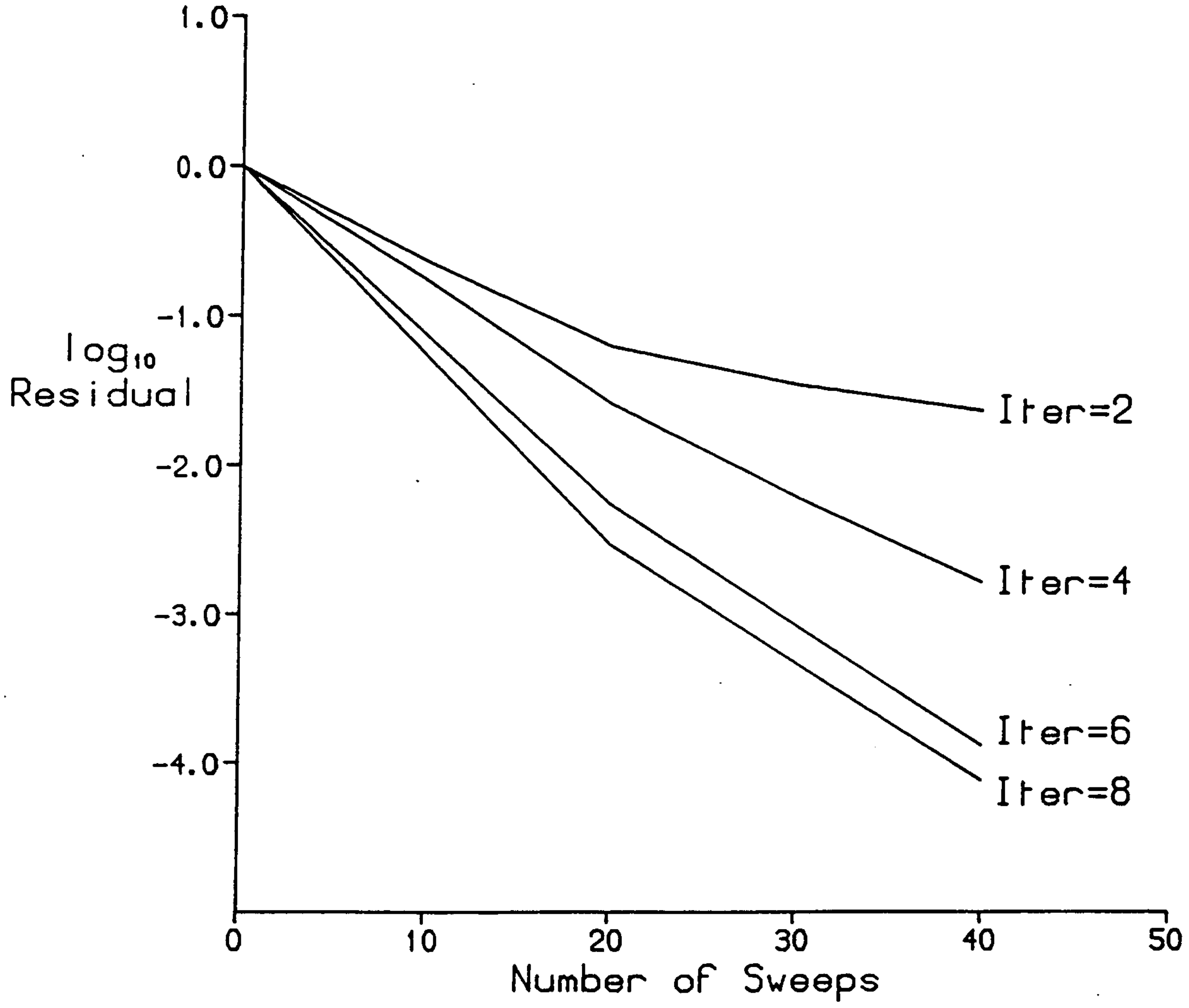


Figure 2.8: Convergence of the inner CG loop as a function of mass.

eigenvalue spectrum of the matrix. We must determine whether there exists a set of values for which IBSOR converges, and if so what the optimum values of ω are. We must also determine whether the tuning of ω depends significantly upon the gauge configuration at fixed β - if it does then the algorithm will be useless as we must calculate propagators on large numbers of configurations.

Calculation of optimum values for ω would require a detailed knowledge of the eigenvalue spectra of the $M_\omega^{-1}N_\omega$ (see algorithm 2.2) which we don't have. Instead we began by setting $\omega = 1$ (the Gauss-Seidel limit of IBSOR) and $m = 0.50$ (a high value) and running the code to obtain a benchmark with which to compare IBSOR. To tune ω for a given m we ran the code for a fixed number of iterations (from a $y_0 = 0$ start), measuring the residual on each sweep for 4 values of ω and noting the rate of fall of the norm of the residual. We then repeated this process using a second configuration. The optimum values of ω were found to differ from one configuration to the next (as one would expect given the known variation in the lowest eigenvalues) but not by an unacceptable degree (see figure 2.9). We continued refining our values for ω until the difference between the upper and lower bounds on the optimum value was approximately equal to the variation between the two test configurations. We have tuned the parameter ω for mass values between 0.01 and 0.50 (see table 2.5.) We find that quark mass is the

mass	0.01	0.04	0.09	0.16	0.50
ω	1.955	1.88	1.70	1.55	1.25

Table 2.5: Optimal values of the parameter ω for a $16^3 \times 24$ lattice at $\beta = 6.0$.

only significant dependent variable. ω is independent of N (for $8 \leq N \leq 32$) and does not need re-tuning from one configuration to the next. When the coupling constant g^2 is changed (in the small range we have explored) only slight adjustments of ω are necessary. The range of acceptable ω values narrows with decreasing mass.

Convergence Rates and Roundoff Errors Figure 2.9 illustrates the variation of the rate of convergence with mass and the slight dependence on configuration (at the lightest mass). The attainable values of $r_k^\dagger r_k$, the residual squared, are limited by the precision to which we compute the propagator. Limiting values of the residuals are given in table 2.6. But because the calculation of β_t is limited to a range of 5 timeslices and there are no global scalars to be accumulated, the timesliced $r_k^\dagger r_k$ falls off exponentially away from the source, with the propagator. This is demonstrated in table 2.6 and figure 2.10. The source term is seen by the algorithm on every sweep, so there is no roundoff-error-induced cumulative drift away from the correct solution. We found that eight iterations of the inner CG solver are more than sufficient at all our mass values (see figure 2.8). This is because of the diagonal dominance of the blocks.

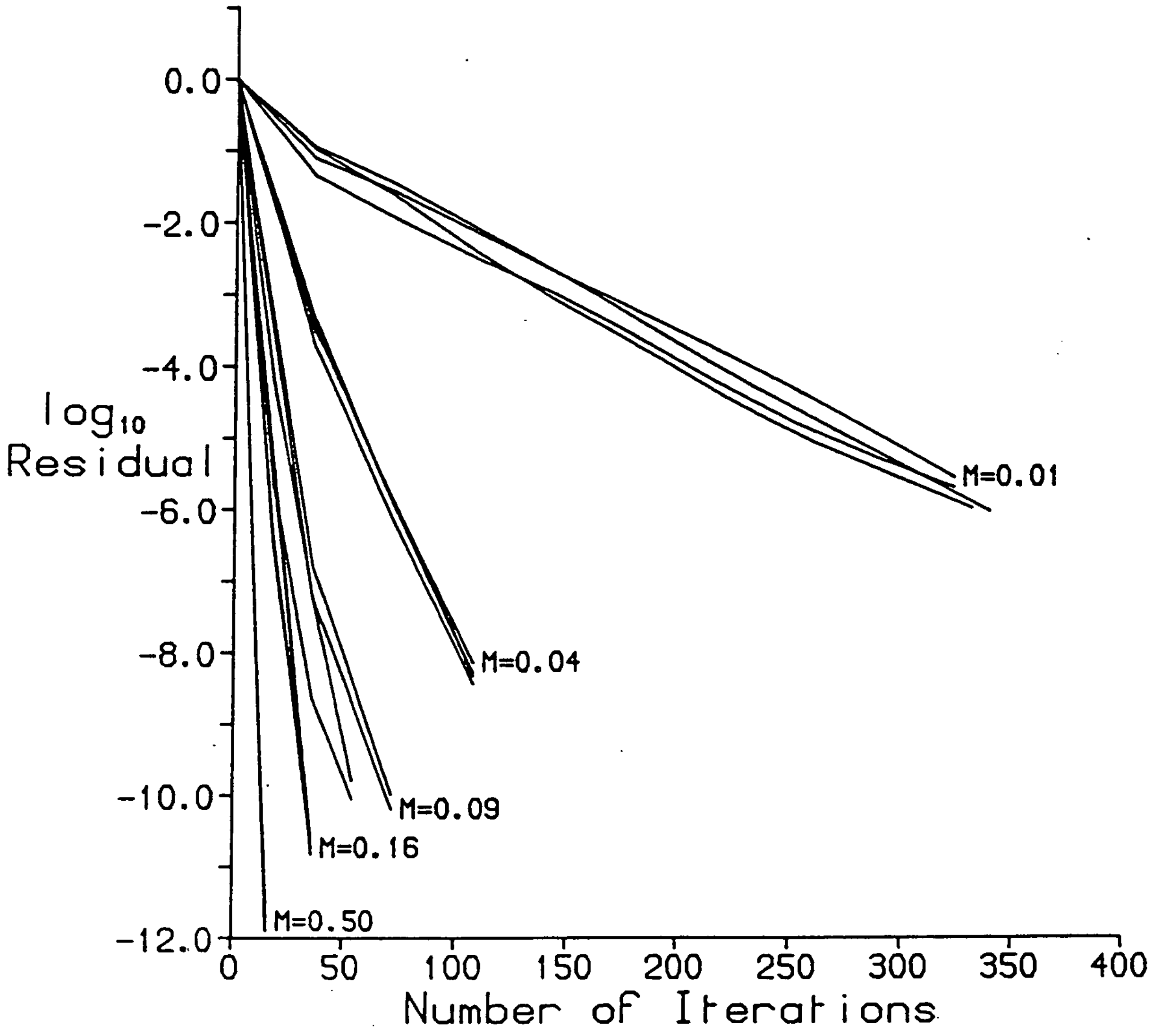


Figure 2.9: Convergence of the IBSOR algorithm.

Fermion type		Squared Residuals			
		limiting	timesliced		
			t_o	t_{o+10}	t_{o+15}
Free					
16^4	32 bit	0.14E-13	0.13E-13	0.92E-18	
	64 bit	0.20E-30	0.18E-30	0.15E-35	
Interacting (32 bit)					
$16^3 \times 24$	$m = 0.01$	0.1E-8	0.68E-10	0.35E-10	0.19E-10
	$m = 0.09$	0.3E-10	0.54E-12	0.22E-13	0.10E-14
	$m = 0.50$	0.1E-12	0.38E-15	0.19E-18	0.86E-22

Table 2.6: Convergence data for IBSOR algorithm.

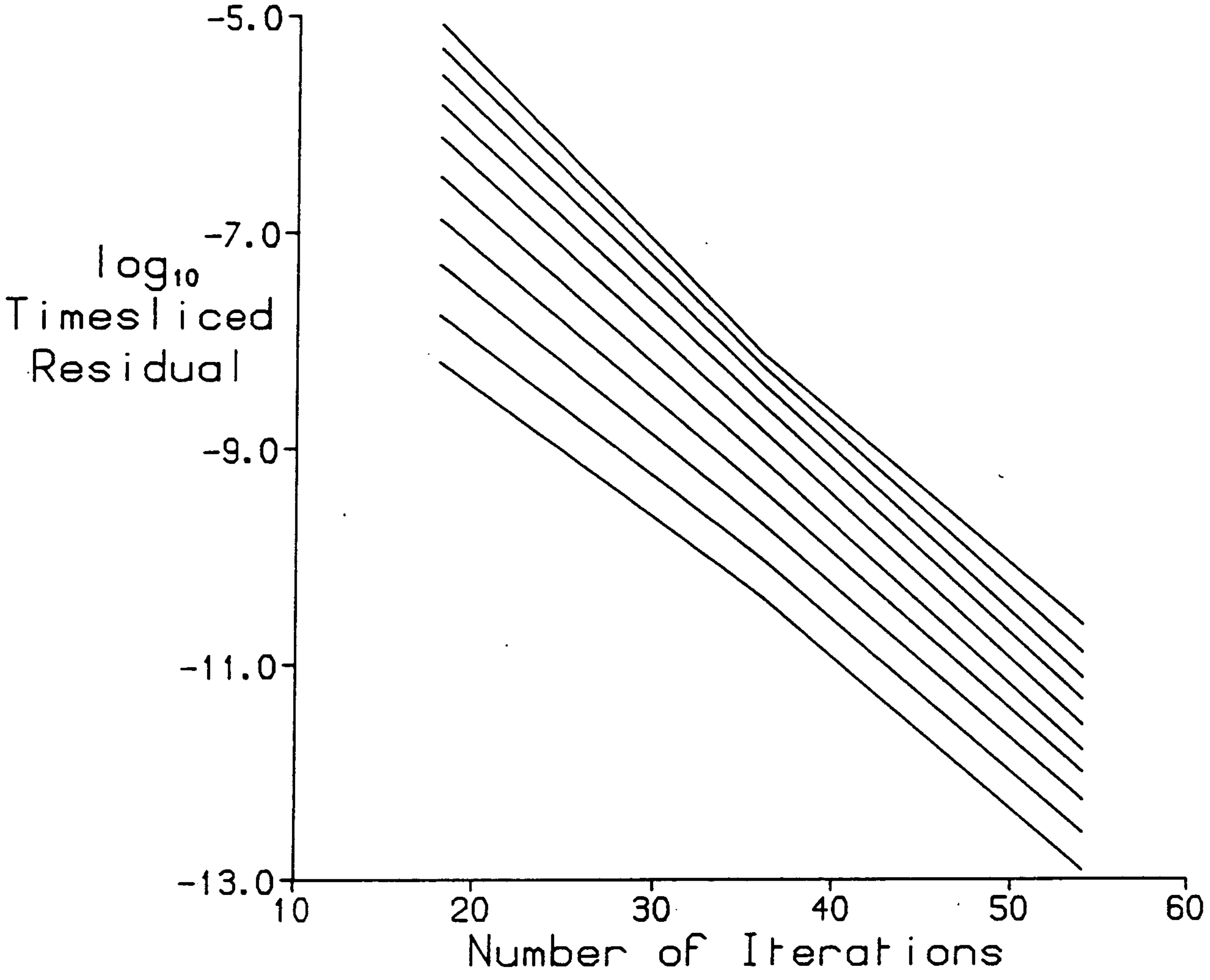


Figure 2.10: Decay of the residual $r_k^\dagger r_k$ with t.

We find that 32-bit arithmetic is sufficient to converge all our propagators (for $0.01 \leq m \leq 0.50$) on a $16^3 \times 24$ lattice. The convergence rate is independent of N (for $8 \leq m \leq 32$) and the behaviour of the timesliced residual shown in figure 2.10 is a feature of all mass values.

Testing and Performance We tested our algorithm by measuring the number of sweeps necessary to obtain agreement between hadron propagators calculated using the CG algorithm and those obtained using IBSOR. In both cases we required convergence of the propagators on the last timeslice to 3 significant figures (this gives 4 or 5 significant figures near the source) and the results agreed to this accuracy.

When we use 6 or 8 iterations of the inner CG solver per sweep we find that our stretch factor is 1.02 (compared with a stretch factor of 6 for our full CG algorithm). The time taken to produce a set of 16^4 propagators drops by a factor of around 3. For $16^3 \times 24$ or larger lattices IBSOR is between 5 and 7 times faster than CG.

Implementation of the IBSOR algorithm on other Computer Systems Our work to date has been on the DAP. Its performance peaks at 15 Mflops and the asynchronous I/O rate is around 250-300 Kbytes/sec. Our conclusions, however, are valid for most computers on which the Conjugate Gradient algorithm has high I/O overheads for large systems of equations. The IBSOR algorithm in its present form has one third of the vector I/O per sweep, and requires significantly fewer sweeps. Consequently, performance is determined by cpu speed, not data transfer rate.

Further, the removal of synchronising scalars means that IBSOR can run at very close to 100 per cent efficiency on multiprocessor systems. If processor 1 is inverting the $(k+1)$'th block equation for y_t then processor 2 can work in parallel on the $(k+2)$'th equation for y_{t-3} without any danger of write conflicts arising. Extending this idea we can fully utilise an p processor system when our lattice has temporal extent $3p$ or larger.

For a larger array of less powerful processors e.g. a few hundred Transputers, other partitioning schemes may be more appropriate. In this case we expect an Iterative Block SOR scheme based on 2 rather than 3 dimensional partitioning would perform well. Hypercube architectures might make use of a 4-dimensional partitioning scheme.

2.2.4 Conclusions

The numerical simulation of QCD is dominated by the solution of the lattice Dirac equation, a large sparse system of linear equations. This problem is inherently parallel, and the parallelism is relatively easy to exploit. If there is sufficient memory then the CG algorithm is suitable for solving this problem and full parallelism of the $\not{D}$ can be exploited. Calculation of the scalar products does not require a significant amount of cpu time because of the high complexity of the matrix-vector product, but imposes some restrictions on the types of parallel processor that can be used (see chapter 5). The dominance of this part of the simulation makes work on preconditioners important, as their effectiveness will have a direct bearing on the time required to perform a realistic simulation.

The number of data accesses per floating point operation is very high in the CG algorithm. If the main store is slow then high I/O overheads will be incurred. Under these circumstances an algorithm with much more computation per word of data transfer, such as IBSOR, may be more suitable.

We consider the optimal architecture for solving the lattice Dirac equation using the CG algorithm in chapter 5.

Chapter 3

Hadron Mass Calculations

3.1 Review of Early Work

Early work in lattice gauge theory yielded surprisingly good results. The first quenched hadron mass calculations [Hamber and Parisi 1981] demonstrated that the pion was the lightest hadron and that chiral symmetry was spontaneously broken. Using the physical pion mass to set the scale they calculated a variety of meson and baryon masses (see table 3.1), and found the pion decay constant to be $f_\pi = 95 \pm 10$MeV. It was hoped that increased lattice sizes would reduce the errors, and that the lattice theory would quickly make predictions that could be tested by experiment.

state	mass (MeV)
ρ	800 ± 100
δ	100 ± 100
A_1	1200 ± 100
nucleon	950 ± 100
Δ	1300 ± 100

Table 3.1: Early hadron mass results [Hamber and Parisi 1981].

Hamber and Parisi used $6^3 \times 10$ and $6^3 \times 12$ lattices at $\beta = 6.0$. Later work showed this to be an inappropriate value of β; it is beyond the deconfining transition for this lattice size [Bowler and Pendleton 1984].

Further work on 8^4 and 10^4 lattices (see [Bowler et. al. 1983] for example) used lower values of β, typically 5.7. These studies began to reveal significant finite-size effects; the physical lattice size is about 1fm, less than the electromagnetic radius of the proton. The proton-to-rho mass ratios calculated were too high by about 60%, and the results obtained using Wilson and staggered fermions did not agree. In addition the temporal extent of the lattices was shown to be too small to extract the ground state data.

The disagreement between Wilson and staggered fermions suggests that $\beta =$

5.7 is not in the scaling regime and so we must work at higher β. Increasing β cuts the physical lattice spacing making further increases in lattice size necessary.

3.2 Aims of our work

The aim of our work has been to distinguish between errors caused by finite-size effects and those caused by non-scaling values of the coupling constant in the quenched approximation, minimise both, and obtain estimates of hadron masses. We have used 16^4 and $16^3 \times 24$ lattices, and have calculated large numbers of propagators to minimise our statistical errors.

β values We have results for $5.7 \leq \beta \leq 6.3$. Studies of finite temperature QCD [Kennedy et. al. 1986] suggest the onset scaling in T_c (the deconfining temperature) at $\beta = 6.15$. Our first runs, at $\beta = 5.7$ and $\beta = 6.0$ used the CG algorithm to calculate quark propagators on 16^4 lattices. Latter data for $\beta = 6.0$, 6.15 and 6.3 were obtained using $16^3 \times 24$ lattices and the IBSOR algorithm.

Fermion Boundary Conditions For periodic temporal boundary conditions the quark propagators decay from the source to the centre of the lattice, and then rise again, halving the temporal extent of the lattice available for measurements. However, this does have the advantage that we can increase the statistics by including the reflection in time. For Dirichlet boundary conditions, $G(t=0) = G(t = N+1) = 0$, the propagators decay across the entire lattice so that in principle more timeslices are available for measurements, but it is necessary to discard data near the boundaries.

We use Dirichlet boundary conditions in time. The spatial boundary conditions can be periodic or anti-periodic. We began by using anti-periodic, but have used periodic as well for the purposes of comparision.

Measurements We concentrate on measurements of the pion, rho and nucleon masses (m_π, m_ρ and m_N), measuring each for a range of quark masses m_q. In addition we construct the mass ratios $\frac{m_\pi}{m_\rho}$ and $\frac{m_N}{m_\rho}$

Similar work has been published by [Barkai et. al. 1985] on $16^3 \times 32$ lattices at $\beta = 6.0$, [Itoh et. al. 1986] and more recently [Gupta et. al. 1987]. Studies of hadron masses using dynamical fermions on small lattices have begun, see for example [Kogut 1986] and [Fukugita et. al. 1986].

3.3 Aspects of Hadron Mass Calculations

In this section we discuss the details of meson and baryon operators used, expected forms for the timesliced propagators and data fitting.

3.3.1 Meson Operators

In section 1.1.6 we described how lattice operators are matched to continuum meson states on the basis of their J^{PC} quantum numbers, and that several operators can correspond to one physical state. (The M_{γ_5} and $M_{\gamma_4\gamma_5}$ operators both have J^{PC} of O^{-+} and are identified with pions.) Kluberg-Stern et. al. show that at finite lattice spacing only the M_{γ_5} state is a Goldstone boson (because of an explicit breaking of flavour symmetry on the lattice) [Kluberg-Stern et. al. 1983]. Other 'pion' states will only be Goldstone-like in the continuum limit.

PCAC tells us that $m\left\langle\bar{\psi}\psi\right\rangle = f_\pi m_\pi^2$ for a Goldstone pion, and so we would expect the mass of the M_{γ_5} state to go to zero as $m_q \rightarrow 0$. The $M_{\gamma_4\gamma_5}$ pion will only have vanishing mass at zero m_q in the continuum limit. We hope to be able to qualitatively judge how far we are from the continuum limit by comparing the two pion states. (There are also two signals for the rho meson which we can compare to the same end.) Having solved eq. 2.10 for $\chi_A(x,t)$ let

$$X_{AB}(\underline{x},t) = |\chi_A(\underline{x},t)\bar{\chi}_B(o)|^2 \tag{3.1}$$

We measure the combinations

$$\begin{aligned}
M_{PS}(t) &= \sum_{\underline{x}}\sum_{AB} X_{AB}(\underline{x},t) \\
M_{SC}(t) &= \sum_{\underline{x}}\sum_{AB}(-1)^{x_1+x_2+x_3+t} X_{AB}(\underline{x},t) \\
M_{VT}(t) &= \sum_{\underline{x}}\sum_{AB}\{(-1)^{x_1}+(-1)^{x_2}+(-1)^{x_3}\} X_{AB}(\underline{x},t) \\
M_{PV}(t) &= \sum_{\underline{x}}\sum_{AB}\left\{(-1)^{x_1+x_2}+(-1)^{x_1+x_3}+(-1)^{x_2+x_3}\right\} X_{AB}(\underline{x},t)
\end{aligned} \tag{3.2}$$

on each configuration and then average over the configurations. It has been shown in [Chalmers 1987] and [Gilchrist et. al. 1984] that at asymptotically large times the expected form of the averaged meson propagators is

$$\begin{aligned}
\langle M_{PS}(t)\rangle &\sim A_\pi e^{-m_\pi(t-t_o)} \\
\langle M_{SC}(t)\rangle &\sim A_{\pi'} e^{-m_\pi(t-t_o)} + (-1)^{(t-t_o)} A_\delta e^{-m_\delta(t-t_o)} \\
\langle M_{VT}(t)\rangle &\sim A_\rho e^{-m_\rho(t-t_o)} + (-1)^{(t-t_o)} A_B e^{-m_B(t-t_o)} \\
\langle M_{PV}(t)\rangle &\sim A_{\rho'} e^{-m_\rho(t-t_o)} + (-1)^{(t-t_o)} A_{A_1} e^{-m_{A_1}(t-t_o)}
\end{aligned} \tag{3.3}$$

but at finite times excited states will also be present. We discuss our attempts to eliminate the excited states in the next section.

3.3.2 Baryons

Our local baryon operators are defined by

$$B(2y+\eta) = \frac{1}{6}\epsilon_{ABC}\chi_A(2y+\eta)\chi_B(2y+\eta)\chi_C(2y+\eta) \tag{3.4}$$

following [Kluberg-Stern et. al. 1983]. A timesliced baryon propagator

$$B(t) = \sum_{\underline{y}} \left\langle B(t, 2\underline{y})\bar{B}(o)\right\rangle \tag{3.5}$$

denoted "EVEN" is defined on the even spatial sub-lattices. It should have the asymptotic form

$$\sum_i A_i^+ e^{-m_i^+(t-t_o)} + (-1)^{(t-t_o)} A_i^- e^{-m_i^-(t-t_o)} \tag{3.6}$$

[Morel and Rodrigue 1984] where the +'s refer to nucleon states, and the −'s have opposite parity. It should satisfy the identities

$$\mathcal{I}m\left\{\sum_{\underline{y}} \left\langle B(t, 2\underline{y})\bar{B}(o)\right\rangle\right\} = 0 \tag{3.7}$$

and

$$\sum_{\underline{y}} \left\langle B(t, 2\underline{y})\bar{B}(o)\right\rangle = \sum_{\underline{y}\underline{\eta}} \left\langle B(t, 2\underline{y}+\underline{\eta})\bar{B}(o)\right\rangle \tag{3.8}$$

The "EVEN = ALL" identity eq. 3.8 implies that there is only one ground state baryon accesible using local lattice operators.

3.3.3 Fitting the Data

We must be able to fit to functions of the form

$$A_1 e^{-m_1 t} + A_2 e^{-m_2 t} \tag{3.9}$$

and

$$A_1 e^{-m_1 t} + (-1)^t \tilde{A}_1 e^{-\tilde{m}_1 t} \tag{3.10}$$

Let $f(t,m) = e^{-mt}$ and $g(t,m) = (-1)^t e^{-mt}$. Having averaged over configurations we have an estimate for y_t, the timesliced propagator and σ_t^2, the variance in its measurements. We fit to the expected form eq. 3.10 by minimising the χ^2 statistic

$$\chi^2 = \sum_{t=t_1}^{t=t_2} \frac{\left(y_t - (A_1 f + \tilde{A}_1 g))\right)^2}{\sigma_t^2} \tag{3.11}$$

(and to eq. 3.9 by replacing $\tilde{A}_1$ by A_2 and g by f in eq. 3.11), t_1 is the first timeslice included in the fit, and t_2 the last. Data points with high (relative) errors, (large σ^2) are suppressed by this fit.

If we define

$$\langle F \rangle = \sum_{t=t_1}^{t=t_2} \frac{F_t}{\sigma_t^2} \tag{3.12}$$

then eq. 3.11 becomes

$$\chi^2 = \left\langle \left(y - (A_1 f + \tilde{A}_1 g\right)^2 \right\rangle \tag{3.13}$$

Minimising this with respect to A_1 gives

$$\frac{\partial \chi^2}{\partial A_1} = -2 \left\langle yf - A_1 f^2 - \tilde{A}_1 fg \right\rangle = 0 \tag{3.14}$$

Therefore

$$A_1 \left\langle f^2 \right\rangle + \tilde{A}_1 \langle fg \rangle = \langle yf \rangle \tag{3.15}$$

Similarly $\frac{\partial \chi^2}{\partial \tilde{A}_1} = 0$ implies

$$\tilde{A}_1 \left\langle g^2 \right\rangle + A_1 \langle fg \rangle = \langle yg \rangle \tag{3.16}$$

Solving these coupled equations for the amplitudes gives

$$\begin{aligned} A_1 &= \frac{1}{\langle f^2 \rangle \langle g^2 \rangle - \langle fg \rangle^2} \left\{ \left\langle g^2 \right\rangle \langle yf \rangle - \langle fg \rangle \langle yg \rangle \right\} && (3.17) \\ \tilde{A}_1 &= \frac{1}{\langle f^2 \rangle \langle g^2 \rangle - \langle fg \rangle^2} \left\{ \left\langle f^2 \right\rangle \langle yg \rangle - \langle fg \rangle \langle yf \rangle \right\} \end{aligned}$$

$$(3.18)$$

Hence A_1 and $\tilde{A}_1$ are dependent variables and we need only vary the mass parameters.

We fit our data in two stages. First we select appropriate ranges of m_1 and $\tilde{m}_1$ and evaluate $\chi^2(m_1, \tilde{m}_1)$ for all points in a 20 by 20 grid. A contour map is drawn on the terminal (see figure 3.1), enabling the user to interactively select a new range of mass parameters. We use this stage to locate the approximate positions of minima in the mass space which are then used by stage 2, a steepest descent minimisation of χ^2. Interactive control of stage 1 of the fit allows us to ensure that we find the best fit; automated fitting or steepest descent alone were found to be inappropriate [1]. (Fitting to a series of exponentials is known to be an ill-conditioned problem, and should be embarked upon with care.)

[1]The fitting program cfit (written by A.Kenway, C.Chalmers and myself) allows us to vary the number of timeslices used, and to divide the data set into blocks.

```
         1.000     1.100     1.200
           |         |         |
           XURPMJGDB97655579BDFI -- 1.200
           XURPMJGDB86544568ACEH
           XURPMJGDB864434579BEG
           XUSPMJGDA864333468BDG
           XVSPMJGDA854223468ACF
           YVSPMJGDA8532223579CE
           YVSPMJGDA8532113479BE
           YVSPMJGDA7531112468BD
           YVSPMJGDA7531002368AD
           YVSPMJGDA8531001357AD
    M2     YVSPMJGDA8531001357AC -- 1.300
           YVSPMJGDB85310012479C
           YVTQNKHEB85310002479C
           ZWTQNKHEB85310*02469B
           ZWTQNKHEB86310002469B
           ZWTQNKHEB86310002468B
           ZWTQNKHEB96420002468B
           ZWTQNKHFC96420002468B
           ZWTROLIFC96421012468B
           ZXUROLIFC97421012468B
           ZXUROLIFCA7531112468B -- 1.400

                     M1
```

Figure 3.1: Contour map of χ^2 produced by the program cfit.

Removing Excited States The expected forms of the timesliced propagators M_{PS} etc. are sums of exponentials in the direct (f) and oscillating (g) channels. We wish to extract the ground states in each channel.

For a large lattice we can do this by only considering timeslices distant from the source as the higher mass excited states decay more rapidly than the ground states. We can then remove contamination from excited states by dropping timeslices close to the source until the fits stabilise. There are 2 problems with this approach (i) the signal to noise ratio in the data drops as we move further from the source making the fit more difficult and (ii) for low quark masses the lattice may not be long enough to remove the effects of an excited state with small amplitude.

An alternative is to perform fits to 3 or 4 mass parameters by using multiple 2 parameter fits (this is described in [Chalmers 1987]).

Errors We estimate errors by repeating the fitting procedure with N_B blocks each of N_C consecutive configurations. This allows us to calculate the standard error in our masses and amplitudes. All errors quoted in section 3.4 are $\pm\sigma$. The accuracy of this procedure increases with sample size. Some of our quoted errors are of poor quality, this is indicated as appropriate.

3.4 Results

Our results are presented in order of increasing β. We draw our conclusions in section 3.5 and then discuss the future.

3.4.1 Hadron Masses at $\beta = 5.7$

We computed quark propagators at 5 mass values in the range $0.01 \leq m \leq 0.50$ for 8 configurations at $\beta = 5.7$. The configurations were generated using a 2-$SU(2)$ subgroup pseudo heatbath program [Bowler et. al. 1986] on a 16^4 lattice. The configurations used were separated by 896 sweeps. The CG algorithm (alg. 2.3) was used to solve the lattice Dirac equation; calculating the 40 propagators required approximately 400 hours of DAP time.

Meson Data

Pions There was no evidence of a signal in the oscillating channel; our fits are to a ground state and an excitation in the direct channel. Our mass estimates are given in table 3.2. Figure 3.2 shows our data and that of [Bowler et. al. 1983] for an $8^3 \times 16$ lattice. Contributions from the excited states are small at high mass, and difficult to estimate at low mass. We quote data for 1-exponential fits. The low mass data extrapolate linearly to zero pion mass at $m_q = 0$. Finite size

m_q	PS	SC	VT	PV
0.50	1.66(1)	2.26(16)	1.93(1)	2.05(3)
0.16	1.02(1)	1.59(17)	1.56(1)	1.46(25)
0.09	0.79(1)	1.42(60)	1.47(3)	1.31(42)
0.04	0.55(1)	1.10(50)	1.24(7)	1.34(-)
0.01	0.27(1)	-	1.06(-)	1.48(-)

Table 3.2: Meson masses at $\beta = 5.7$. Fits are to 2 exponentials, except for PS pion where just 1 was used.

effects in the pion are small at this β value. Our estimates of the pion mass from M_{SC} are not in agreement with those for π_{PS} (see table 3.2). There are large errors in the data from the SC propagator, and there may well be contamination from excited states (see later).

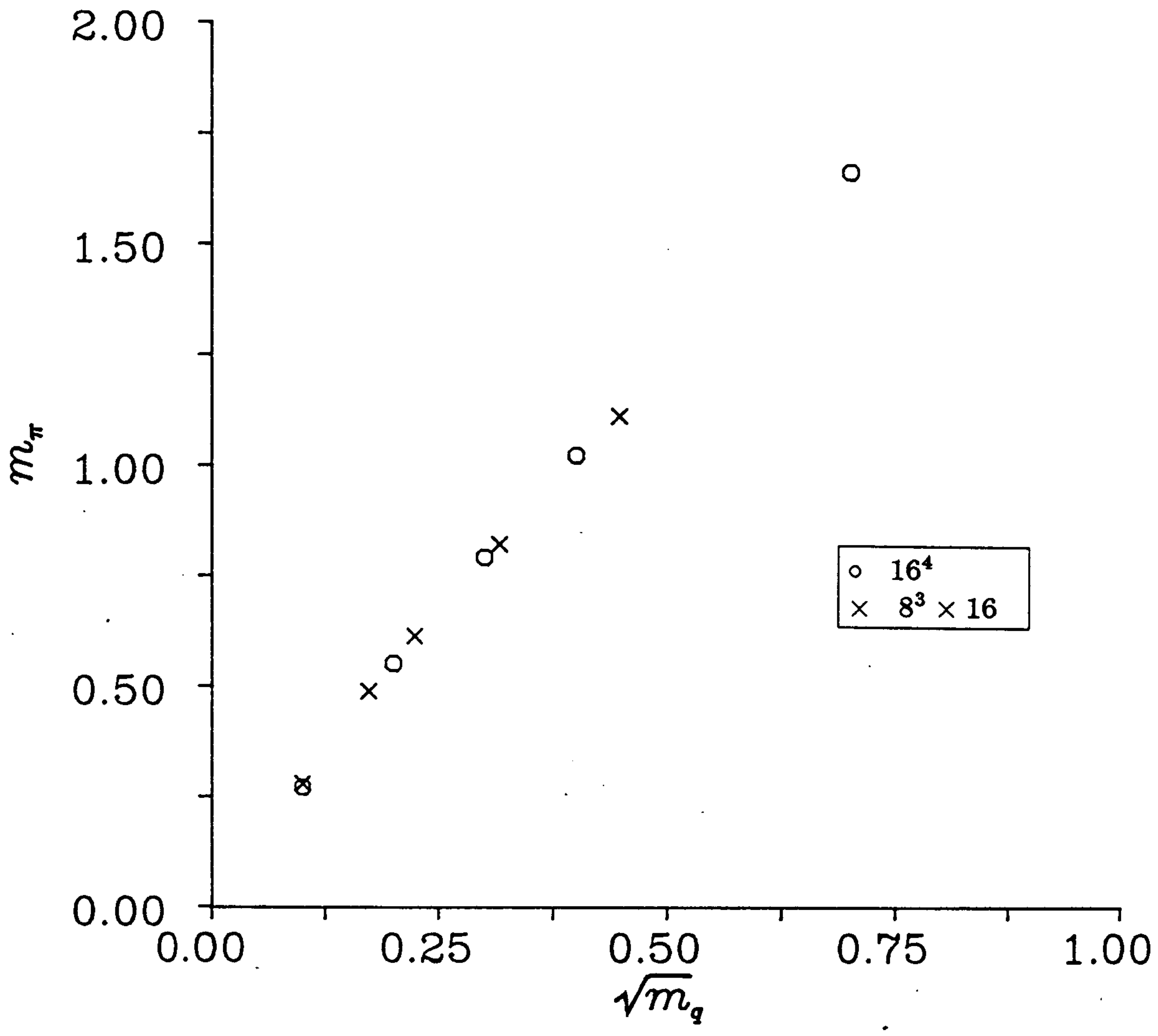


Figure 3.2: Pion Masses at $\beta = 5.7$.

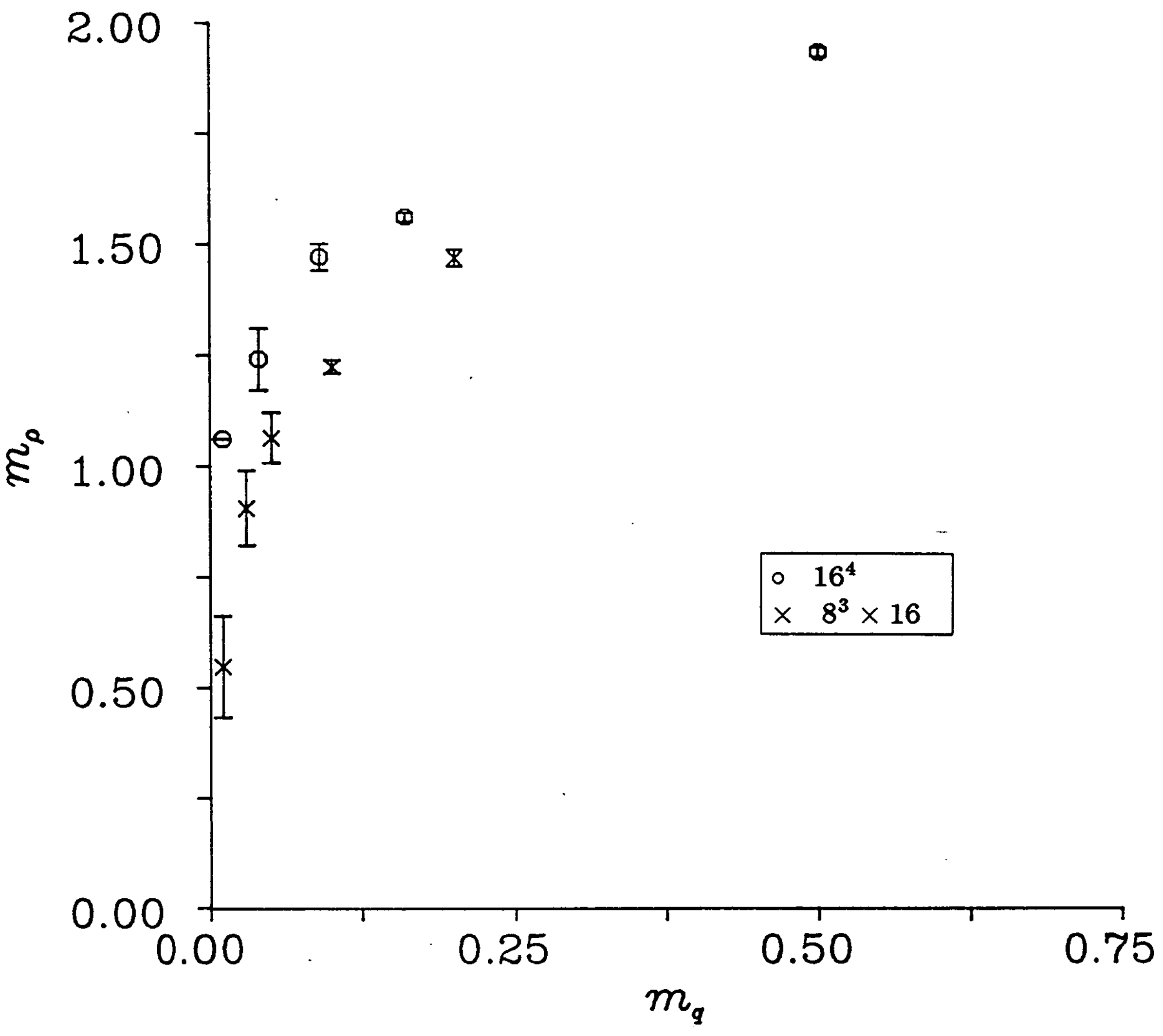


Figure 3.3: Rho masses from the VT and PV propagators at $\beta = 5.7$.

Rho Mesons We fit the sum of two exponentials to the M_{VT} and M_{PV} propagators to extract the rho mass, one for the direct, and one for the oscillating channel. Our estimates of m_ρ are significantly higher than those obtained from the $8^3 \times 16$ lattice. There is some agreement in the estimates from VT and PV timesliced propagators at large quark mass. The rho data is given in table 3.2 and figure 3.3.

Baryons

Our fits to the $(\frac{1}{2})^+$ nucleon data are shown in table 3.3 and figure 3.4. The

m_q	EVEN	ALL
0.50	3.04(1)	3.04(4)
0.16	2.55(16)	2.70(7)
0.09	2.54(-)	2.63(10)
0.04	2.45(-)	2.37(27)
0.01	2.35(9)	-

Table 3.3: $(\frac{1}{2})^+$ nucleon mass estimates at $\beta = 5.7$ from 2-exponential fits to the EVEN and ALL nucleons for timeslices 6-13.

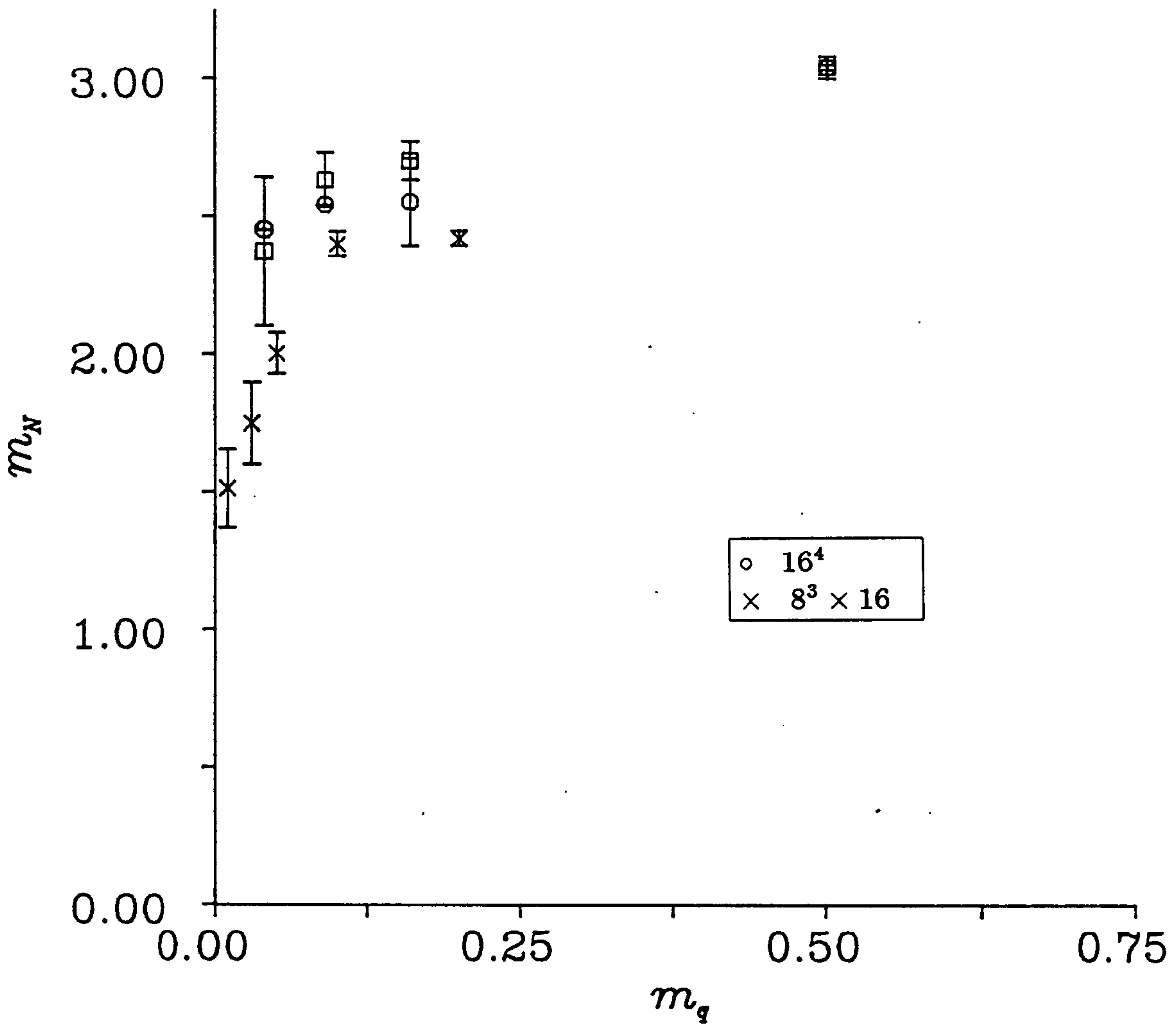


Figure 3.4: Hadron mass estimates for $\beta = 5.7$ on $8^3 \times 16$ and 16^4 lattices.

data quality is poor, and will not support fits to more than 2 parameters. The mass estimates from the EVEN and ALL agree within errors, but the errors are large and the quality of error estimation is low because of the small sample size.

m_q	$\frac{m_\pi}{m_\rho}$	$\frac{m_N}{m_\rho}$
0.50	0.86(1)	1.58(1)
0.16	0.65(1)	1.64(10)
0.09	0.54(1)	1.73(22)
0.04	0.44(2)	1.97(11)
0.01	0.26(-)	2.24(-)

Table 3.4: Pion to rho and nucleon to rho mass ratios at $\beta = 5.7$.

We plot the mass ratios $\frac{m_N}{m_\rho}$ against $\frac{m_\pi}{m_\rho}$ in figure 3.5 (the data is in table 3.4). The starred points are the experimental ratio and the heavy-quark limit. If we are in the scaling regime then the points (for different m_q) should lie on a universal curve, independent of β. Our data is similar to that of [Bowler et. al. 1984a] on the smaller lattice; spatial finite-size effects in the nucleon and rho seem to cancel out. We see no crossover to the light quark (experimental) limit. Further, the mass ratios are not consistent with those at $\beta = 6.0$ (see section 3.4.2) indicating that we are not in the scaling regime. No further work was done at $\beta = 5.7$.

3.4.2 Hadron Masses at $\beta = 6.0$

We used the CG program to calculate a further set of 16^4 propagators, 5 masses for 8 configurations separated by 896 sweeps at $\beta = 6.0$. We used the same masses, boundary conditions and convergence criteria as at $\beta = 5.7$.

Pions We saw no evidence for an oscillating state in M_{PS}, and so we fitted to the ground state pion and an excitation. A linear fit to the data (see figure 3.6) supports a small intercept suggesting that there are finite-size effects at low m_q see [Billoire et. al. 1985b] The pion data from M_{SC} is in rough agreement with that from M_{PS} at all but the lightest quark mass. The masses from the SC propagator are generally higher and agree better with the 1-exponential fits to M_{PS}, suggesting the presence of a radial exciation in the direct channel. We were not able to drop sufficient timeslices from the fit to the SC timeslice propagator to remove the excitation.

Barkai et. al. quote results for 5 configurations at $\beta = 6.0$ on a $16^3 \times 32$ lattice with periodic boundary conditions. Our pion results on the 16^4 lattice are some 20% higher than theirs [Barkai et. al. 1985].

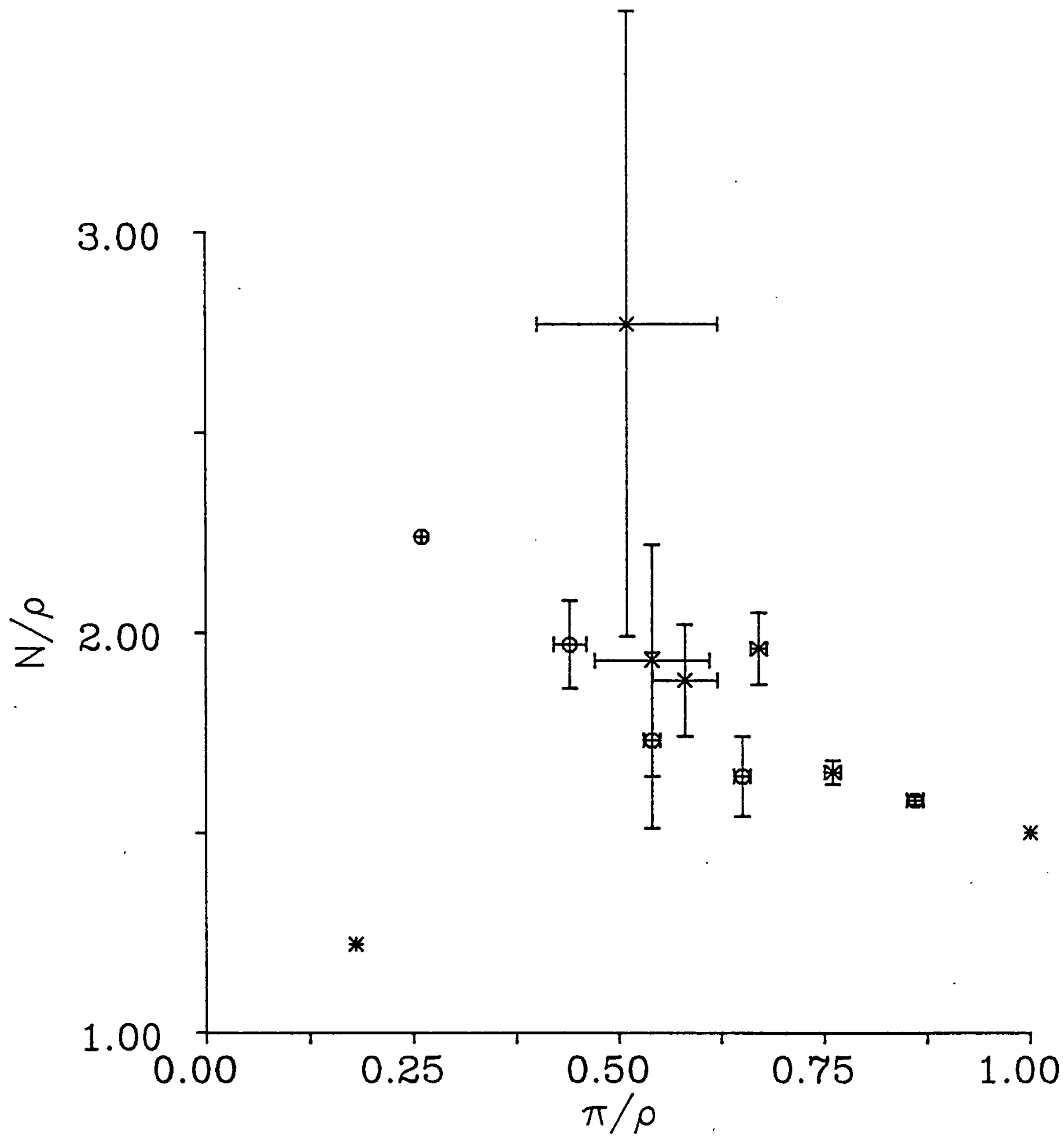


Figure 3.5: Mass ratios plot for $\beta = 5.7$, $8^3 \times 16$ (crosses) and 16^4 (circles) lattices.

m_q	PS	SC
0.50	1.66(1)	1.88(1)
0.16	0.99(1)	1.07(5)
0.09	0.75(1)	0.85(7)
0.04	0.52(2)	0.58(7)
0.01	0.30(2)	0.51(9)

Table 3.5: Pion mass estimates from the PS and scalar SC meson propagators. Fits are to 2 exponentials on timeslices 6-13 and 7-13 respectively.

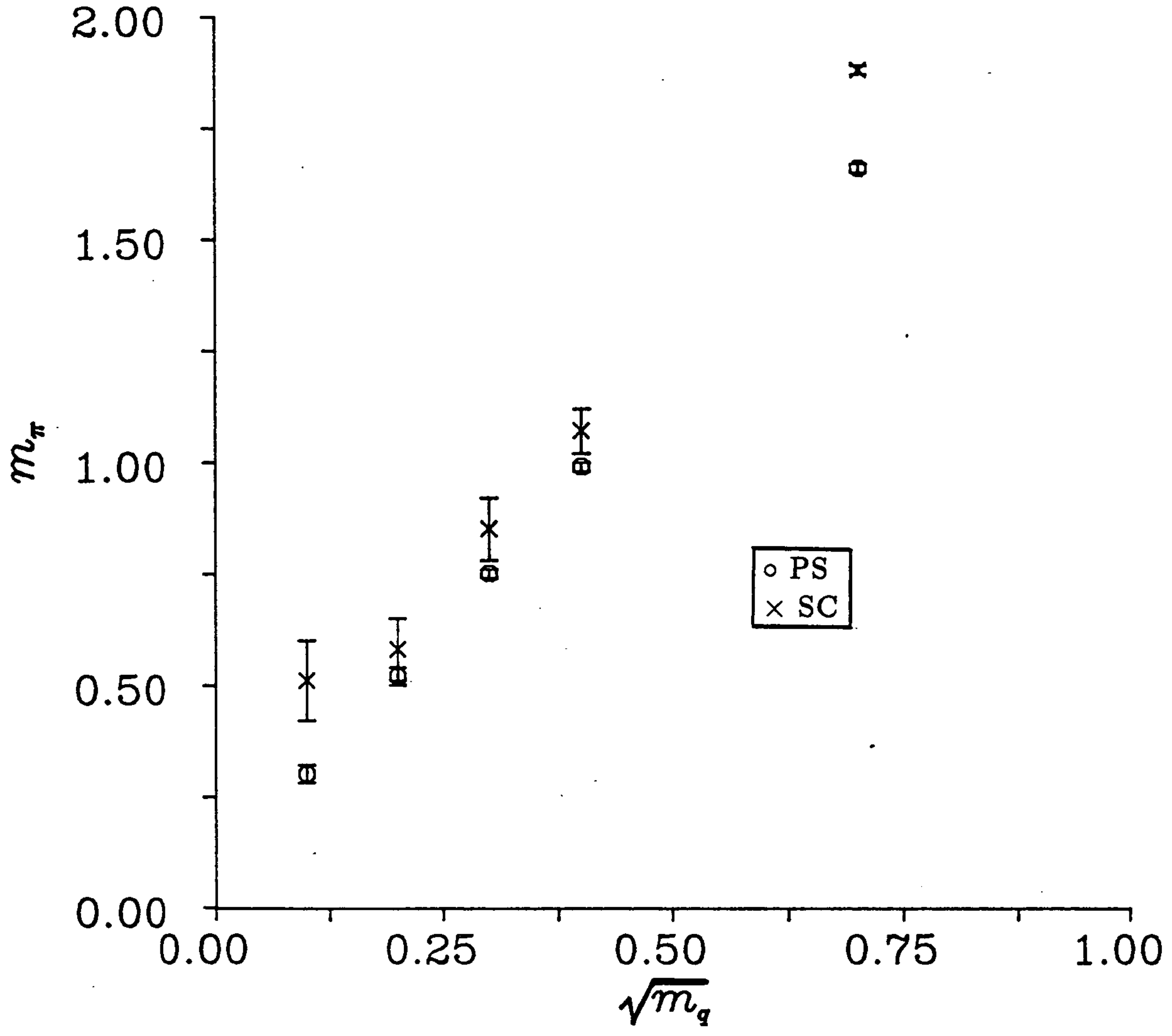


Figure 3.6: Pion mass estimates at $\beta = 6.0$ from a 16^4 lattice.

Rho Mesons and Nucleons. We fit M_{VT} and the baryon propagators with an exponential in the direct channel and another in the oscillating channel. There is no evidence for stabilisation in the fits as we drop timeslices near the source. The masses fall as we remove timeslices, and then the fits break down due to lack of data. This suggests that there are significant radial excitations in the propagator data. There are similar signals for the rho in both channels, but the lack of asymptotic behaviour prevents us from attaching any importance to this.

Development work on the IBSOR algorithm (section 2.2.3) was completed during this run. In view of our need for longer lattices and the higher efficiency of IBSOR we restarted our $\beta = 6.0$ run on a $16^3 \times 24$ lattice.

3.4.3 Hadron Masses at $\beta = 6.0$ on a $16^3 \times 24$ Lattice

The $16^3 \times 24$ configurations were constructed by periodically extending 16^4 configurations in time, prior to gauge fixing. Hadron propagation over more than 16 timeslices is not observable in our calculations and so we do not expect this extension to introduce systematic errors. The results that follow are for 5 masses on 32 configurations with successive configurations separated by 224 pseudo-heatbath sweeps. The errors are estimated by dividing the data into 4 consecutive bins of 8 configurations and repeating the fits for each of the 4 sets.

The Mesons

Pions Our pion mass estimates are obtained by fitting the sum of 2 exponentials to M_{PS}. The data is presented in table 3.6 and figure 3.7. A linear fit to the data yields,

$$m_\pi = (0.006 \pm 0.01) + (2.40 \pm 0.04)\sqrt{m_q}$$

Spatial finite-size effects appear to be small over this range of quark masses ($0.01 \leq m \leq 0.50$). Comparison of these results with those from the 16^4 lattice reveals a finite-time effect, reflecting our failure to extract excited states properly on the smaller lattice. This is judged to be the cause of the discrepancy between our earlier results and those of Barkai et. al. On the larger lattice we are in complete agreement with them, although our errors are larger. We performed a 3-exponential fit to the SC propagator; ground and excited states in the direct channel and a ground state (for the O^{++} state associated with the delta) in the oscillating channel. Inclusion of the excitation in the direct channel did not change the estimates of m_π at high m_q, but gave lower masses and better fits at low m_q. The two pion mass estimates agree to within 10% for $m_q \leq 0.16$. They both extrapolate linearly in $\sqrt{m_q}$ to $m_\pi = 0$ within errors.

The Rho There was no evidence for stabilisation of the rho in fits to M_{VT} on the smaller lattice. On the $16^3 \times 24$ lattice the fits stabilize and a clear ground

	PS		SC		
m_q	m_1	m_2	m_1	m_2	$\tilde{m}_1$
0.50	1.648(11)	2.09(39)	1.85(6)	3.31(63)	1.94(6)
0.16	0.966(5)	1.47(80)	1.05(10)	2.80(48)	1.23(2)
0.09	0.725(6)	1.29(4)	0.78(2)	2.61(27)	1.00(1)
0.04	0.488(8)	1.25(17)	0.53(2)	4.23(65)	0.77(1)
0.01	0.245(13)	1.35(24)	0.28(5)	4.74(68)	0.55(4)

Table 3.6: Pion mass estimates at $\beta = 6.0$ for $16^3 \times 24$ lattice. Timeslices 9-19 were used for PS, and 7-18 for SC.

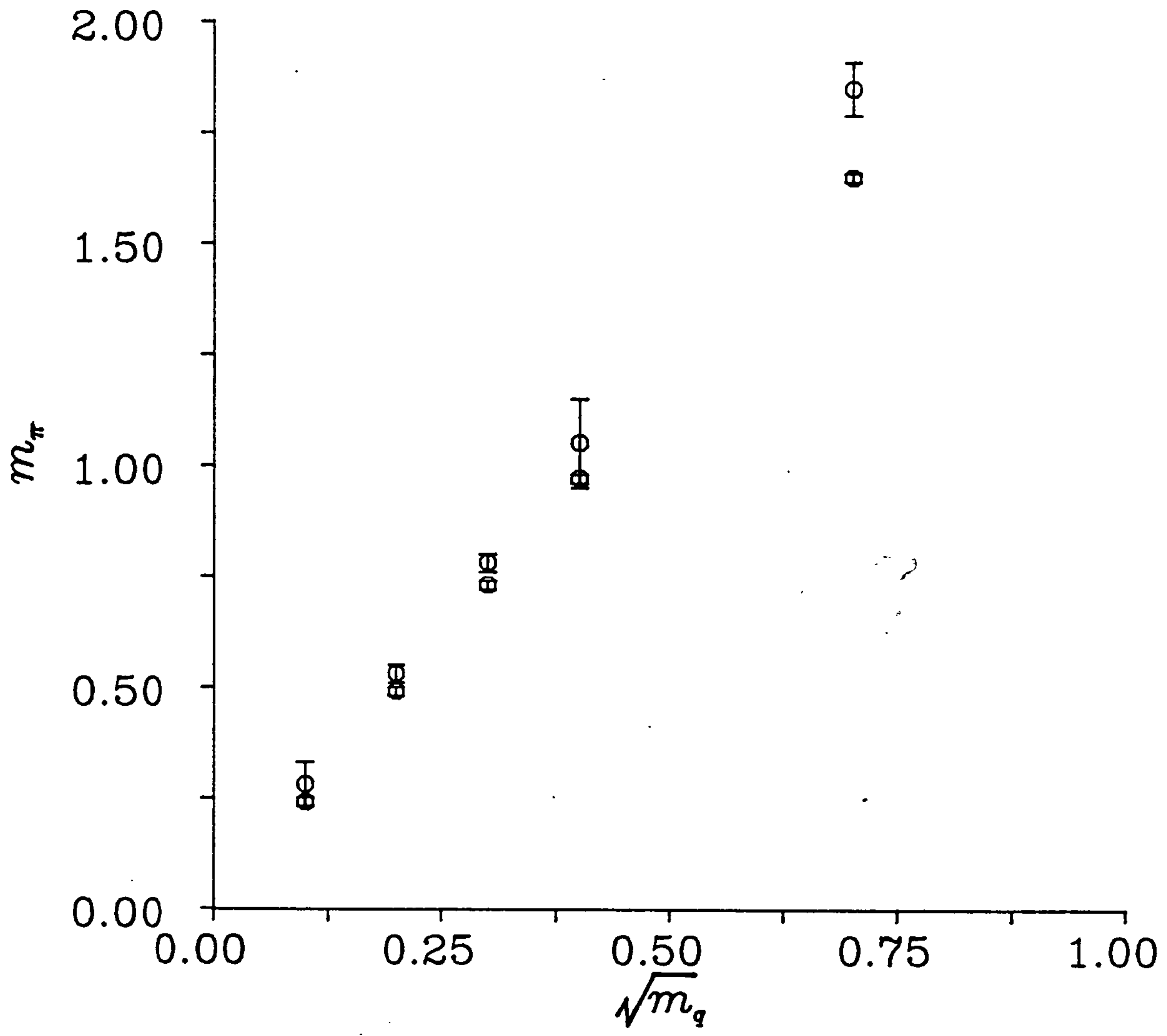


Figure 3.7: Pion mass estimates from PS and SC propagators at $\beta = 6.0$ for $16^3 \times 24$ lattice.

state is exposed in the 2-exponential fits to timeslices 11-19. Inclusion of a third exponential, an excitation in the rho channel, causes only a small drop in the estimates at all but the lowest mass (although the data at $m = 0.01$ is not of sufficient quality for this 3-parameter fit to be taken seriously). Rho mass

m_q	VT	PV
0.50	1.82(1)	1.90(2)
0.16	1.13(2)	1.15(2)
0.09	0.89(1)	0.90(3)
0.04	0.71(2)	0.69(3)
0.01	0.62(7)	0.66(18)

Table 3.7: Rho mass estimates from 2-parameter fits to the VT and PV propagators at $\beta = 6.0$.

estimates from the PV propagator are given in table 3.7, they were obtained using a 2-parameter fit to timeslices 10-19. The 2 estimates of m_ρ, with different flavour content, agree to within 7% at all masses studied, from which we conclude that there is evidence for flavour-symmetry restoration in the meson sector at $\beta = 6.0$.

3.4.4 Baryons

Our results for 2-exponential fits to the baryon propagators are given in tables 3.8, 3.9 and 3.10 and are plotted in figures 3.9 and 3.10. It is difficult to extract reliable mass estimates from them.

The 2-exponential fits (table 3.8) stablize as we drop timeslices near the source, but in doing this errors grow because the signal-to-noise ratios are low for timeslices 16-20. The resulting mass estimates are in serious disagreement with those of Barkai et. al. The only significant difference between their study and ours is the choice of spatial boundary conditions.

We measure 2 baryon operators, EVEN and ALL, as described in section 3.3.2. They should give the same results on an infinite lattice [Morel and Rodrigue 1984]. We see clear discrepancies between them, increasing in magnitude as $m \to 0$ (see figure 3.9).

To study our baryon data further we peformed a series of 4-parameter fits to the timeslice propagators (see tables 3.9 and 3.10 for the masses), including an excitation in both channels. We see a large contribution to the baryon propagator coming from both excited states. The effect is most dramatic in the EVEN propagator where the masses of the excitation are low. Significant excitations have also been found using periodic boundary conditions [Itoh et. al. 1986]. The ground state masses above are plotted in figure 3.10 and are used for the mass ratio plot figure 3.11. Our mass ratios for $\beta = 6.0$ are given in table 3.11. The

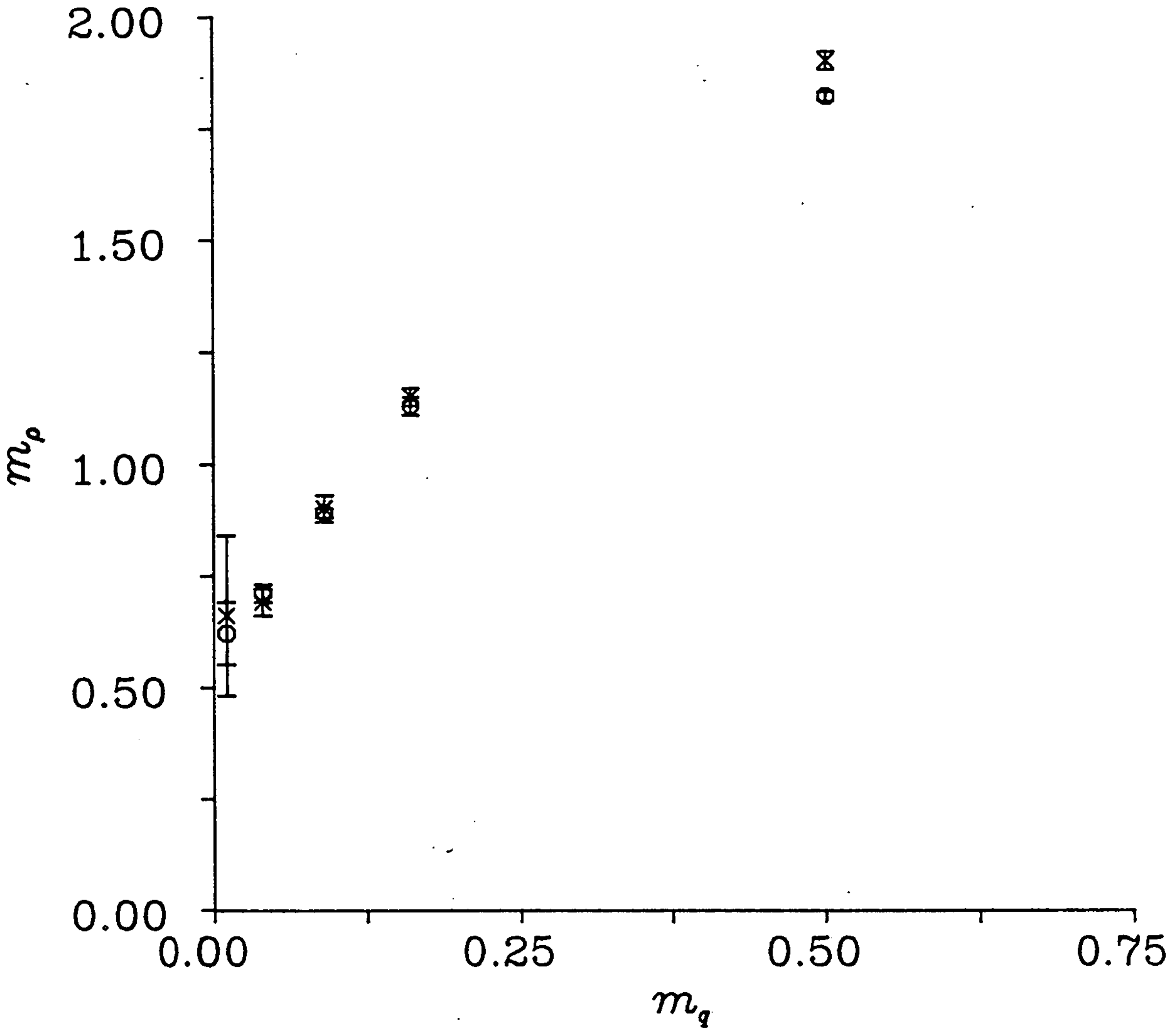


Figure 3.8: Rho mass as a function of m_q at $\beta = 6.0$.

	EVEN		ALL	
m_q	m_1	$\tilde{m}_1$	m_1	$\tilde{m}_1$
0.50	2.92(1)	3.22(10)	2.91(1)	3.31(63)
0.16	2.17(4)	2.52(30)	2.17(14)	2.80(48)
0.09	1.73(9)	1.89(16)	1.80(21)	2.61(27)
0.04	1.31(15)	1.40(11)	1.06(15)	-
0.01	1.42(34)	1.40(37)	1.33(26)	-

Table 3.8: Baryon masses from the EVEN and ALL propagators at $\beta = 6.0$ using 2-parameter fits.

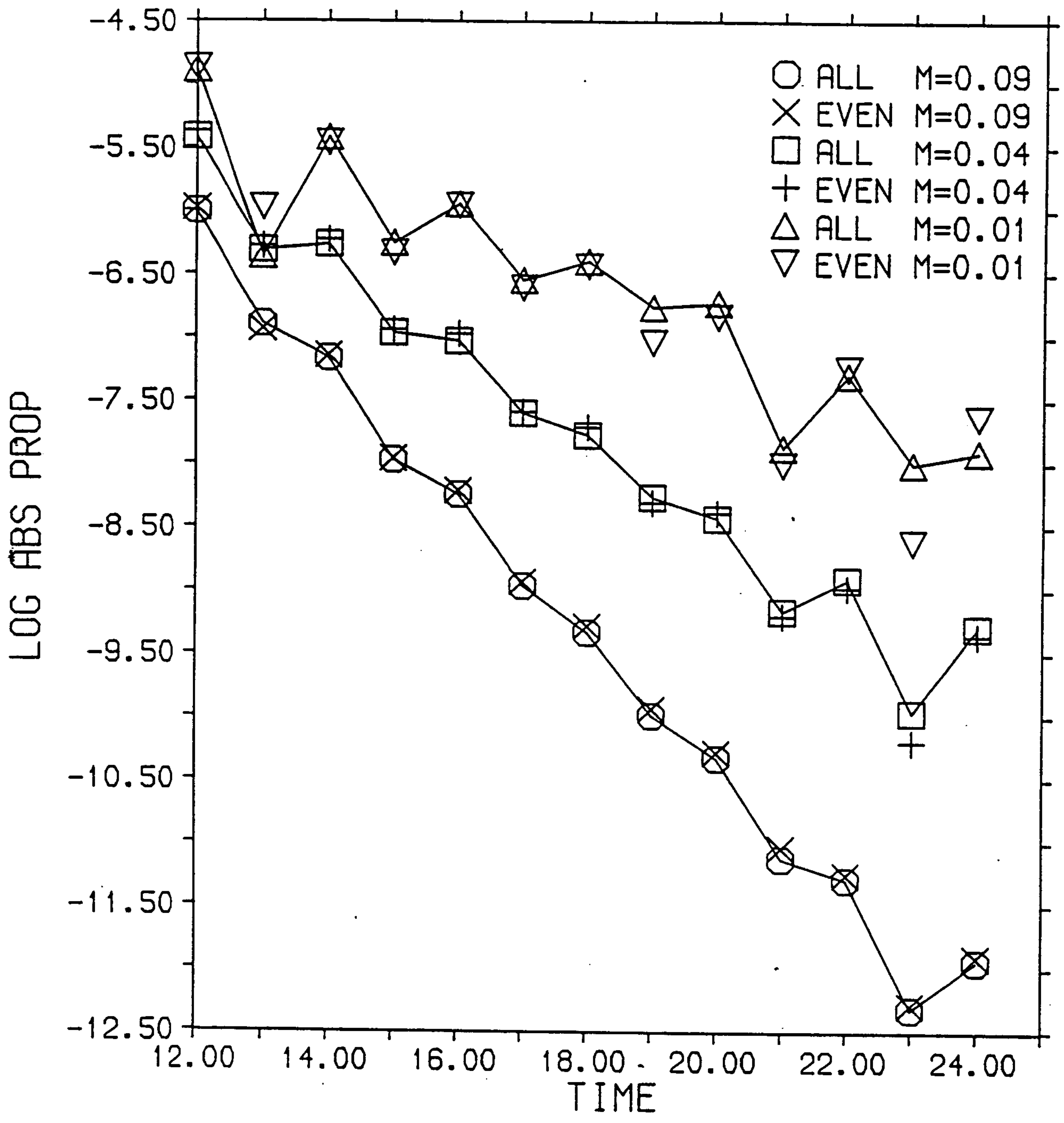


Figure 3.9: EVEN and ALL baryon propagators for $\beta = 6.0$.

m_q	m_1	m_2	$\tilde{m}_1$	$\tilde{m}_2$
0.50	2.88(15)	3.02(4)	3.18(23)	3.55(22)
0.16	1.75(16)	2.37(5)	1.19(28)	2.40(7)
0.09	1.93(11)	2.22(8)	0.89(14)	2.24(8)
0.04	1.02(9)	2.14(10)	1.05(21)	2.14(11)
0.01	0.94(14)	1.80(11)	0.59(49)	1.80(16)

Table 3.9: Baryon masses from the EVEN propagators at $\beta = 6.0$ using 4-parameter fits.

m_q	m_1	m_2	$\tilde{m}_1$	$\tilde{m}_2$
0.50	2.72(13)	3.01(5)	3.15(6)	4.84(29)
0.16	1.80(22)	2.52(7)	2.26(6)	2.99(12)
0.09	1.52(26)	2.41(6)	1.90(11)	2.72(11)
0.04	1.07(19)	2.32(3)	1.94(22)	2.67(12)
0.01	0.90(24)	2.16(8)	1.75(-)	1.75(31)

Table 3.10: Baryon masses from the ALL propagators at $\beta = 6.0$ using 4-parameter fits.

m_q	$\frac{m_\pi}{m_\rho}$	$\frac{m_N}{m_\rho}$
0.50	0.91(4)	1.60(11)
0.16	0.88(3)	1.59(16)
0.09	0.84(5)	1.38(13)
0.04	0.75(11)	1.57(11)
0.01	0.43(9)	1.62(63)

Table 3.11: Mass ratios at $\beta = 6.0$

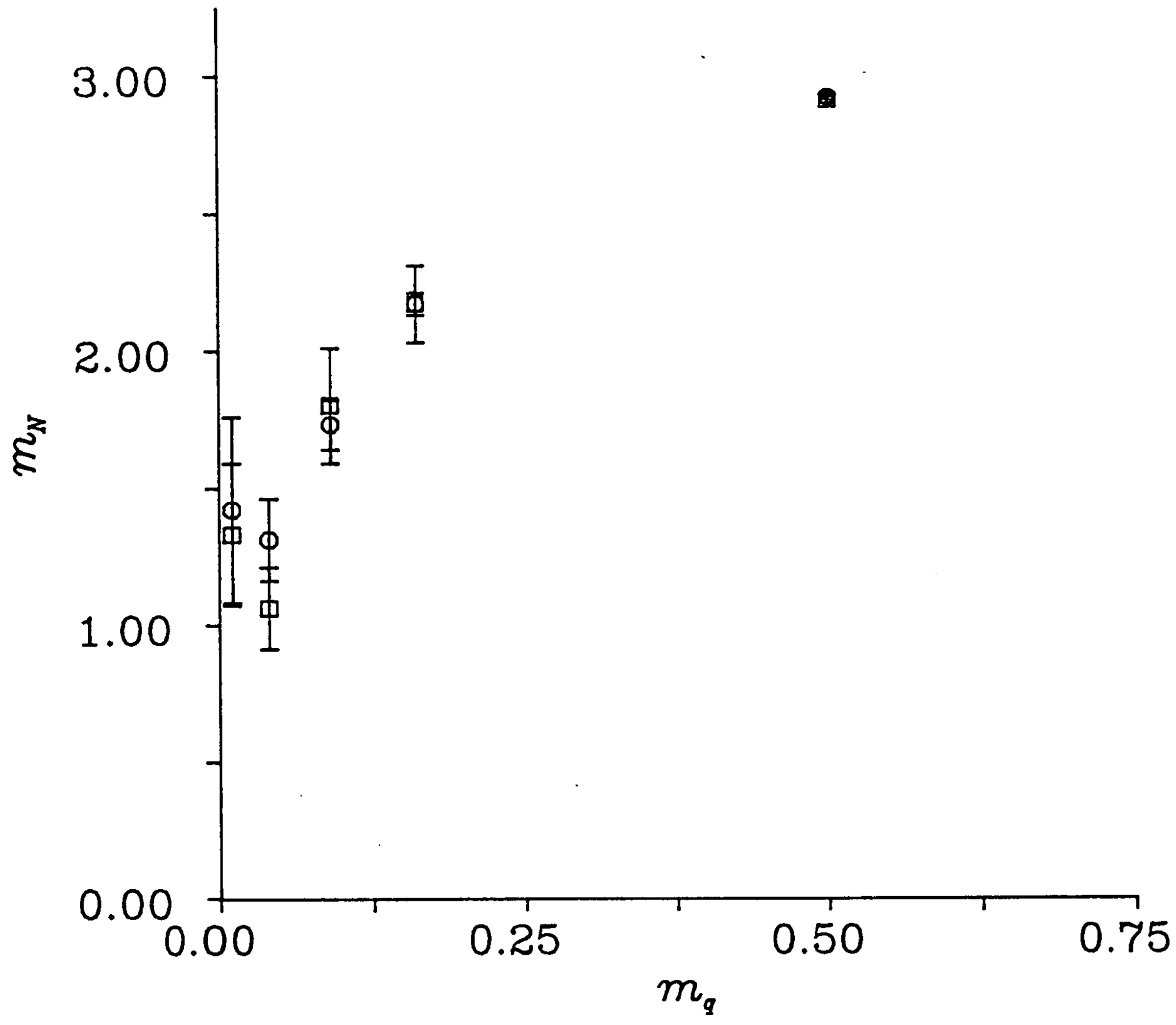


Figure 3.10: Hadron masses at $\beta = 6.0$.

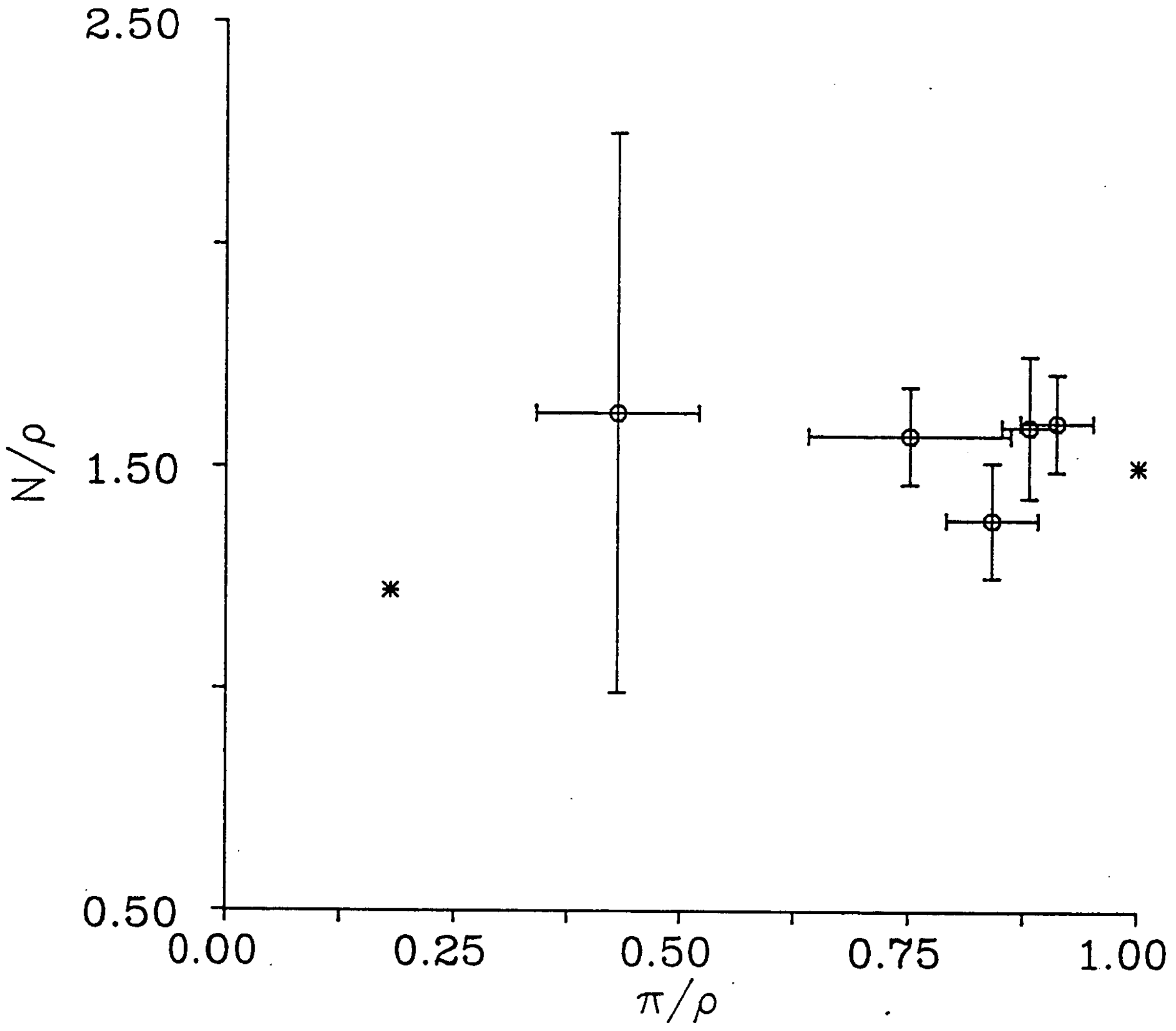


Figure 3.11: Mass ratios plot for $\beta = 6.0$.

mass ratios for low m_q are significantly lower than those at $\beta = 5.7$, and show signs of crossing over to the light-quark limit. The errors are too large to permit firm conclusions being drawn from this.

The discrepancies between our data and that of Barkai et. al. coupled with the apparent failure of the EVEN=ALL identity suggests that significant finite size effects are present in our baryon propagators. To study this problem further we considered both periodic and anti-periodic spatial boundary conditions in our next run.

3.4.5 Hadron Masses at $\beta = 6.15$

Our data at $\beta = 6.15$ are drawn from full $16^3 \times 24$ configurations generated using a 3-$SU(2)$ subgroup pseudo-heatbath program, and separated by 176 sweeps. We calculated propagators on a set of 32 configurations using periodic boundary conditions, and then re-ran the first 24 using anti-periodic spatial bcs. The aims of this run were (i) to test systematic errors in the baryon propagators, and (ii) to compare the meson data with that obtained at $\beta = 6.0$.

The Meson Sector

Pions Our mass estimates from 2-parameter fits to the PS propagator are given in table 3.12 for periodic spatial boundary conditions. The choice of boundary conditions was found to have no significant effect upon either the propagators or the masses extracted at all but the lowest quark mass, where the errors were (relatively) large. Preliminary work at low masses ($0.001 \leq m \leq 0.003$) suggests that finite-size effects are large. This would explain the discrepancies at $m = 0.01$. 1- and 2-exponential fits were made to the PS propagator; excitations are present in the first 7 timeslices from the source so a 2-exponential fit was necessary to expose the ground state. We plot the masses obtained from SC and

m_q	PS	SC
0.50	1.633(5)	1.78(5)
0.16	0.916(6)	0.96(1)
0.09	0.662(2)	0.69(1)
0.04	0.422(4)	0.43(2)
0.01	0.208(3)	0.24(3)

Table 3.12: Pion masses at β=6.15 for a $16^3 \times 24$ lattice.

PS propagators in figure 3.12. Note that we have kept the masses in lattice units constant, so that since the lattice spacing is smaller at $\beta = 6.15$ than at 6.0, the physical masses are higher. Both curves extrapolate to zero within errors but that for the masses from the SC propagator has higher curvature and may be contaminated by excitations. The masses agree to within 5% for $m \leq 0.16$.

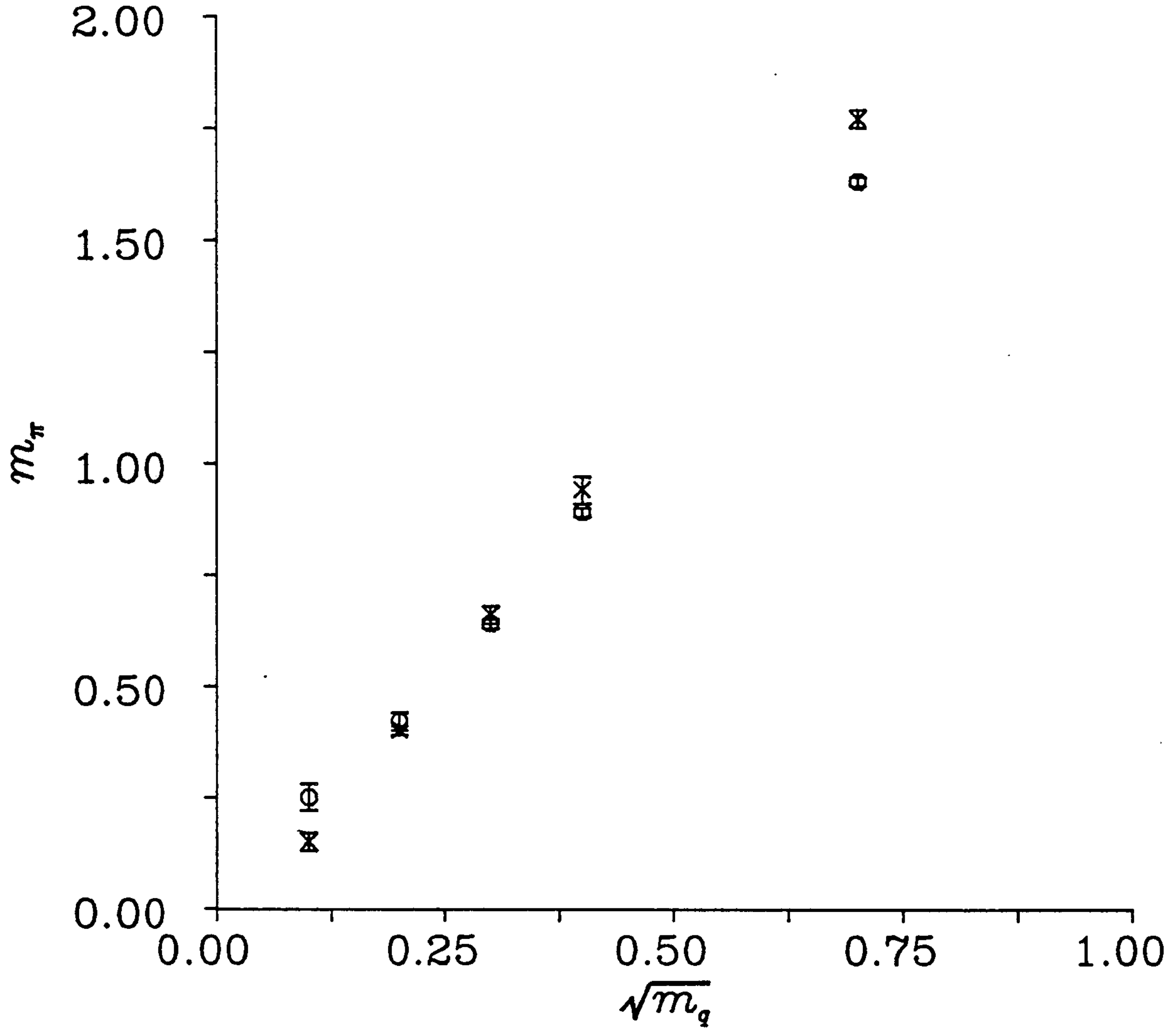


Figure 3.12: Pion masses from PS and SC propagators at $\beta = 6.15$ from propagators calculated on a $16^3 \times 24$ lattice.

The Rho Meson The $\beta = 6.15$ rho data is very clean and there is no significant dependance upon the spatial boundary conditions used. The 2-exponential fits in table 3.13 stabilise as timeslices are removed; results quoted use timeslices 11-19 and 12-19. The agreement between the 2 sets of rho mass estimates is very good,

m_q	VT	PV
0.50	1.77(1)	1.80(1)
0.16	1.02(2)	1.00(2)
0.09	0.76(1)	0.74(1)
0.04	0.54(2)	0.53(1)
0.01	0.38(5)	0.34(12)

Table 3.13: Rho masses at β=6.15 for a $16^3 \times 24$ lattice.

except at the high mass. This is a feature of many of our estimates of meson masses. We attribute it to the much reduced correlation length at high masses; and hence the greater sensitivity to flavour symmetry breaking lattice artifacts.

Baryon Data

We performed 2-exponential fits to the baryon timeslice propagators. This exposed the ground states successfully and gave similar answers to the 4-parameter fits. The statistical errors are much smaller for periodic boundary conditions than for anti-periodic.

Comparison of Boundary Conditions Table 3.14 gives our estimates of the $(\frac{1}{2})^+$ masses taken from fits to the baryon propagators with periodic and anti-periodic boundary conditions. We see clear evidence for the EVEN=ALL

	periodic		anti-periodic	
m_q	EVEN	ALL	EVEN	ALL
0.50	2.81(1)	2.80(1)	2.90(1)	2.91(1)
0.16	1.64(2)	1.65(2)	2.18(8)	1.95(16)
0.09	1.23(2)	1.25(1)	1.44(13)	1.60(12)
0.04	0.86(2)	0.86(2)	0.92(8)	1.27(15)
0.01	0.58(3)	0.51(7)	0.45(22)	1.62(29)

Table 3.14: Baryon masses at β=6.15 for a $16^3 \times 24$ lattice. The fits are for timeslices 11-19.

identity in the baryon propagators using periodic spatial boundary conditions. This is not the case when use anti-periodic spatial boundary conditions.

Carpenter and Baillie have shown that the free Wilson fermion propagator on an infinite lattice is bounded above by the periodic propagator and bounded below by the anti-periodic propagator on a finite spatial lattice (see figure 3.13). A similar result is shown in figure 3.14 for EVEN baryon propagators at $\beta =$

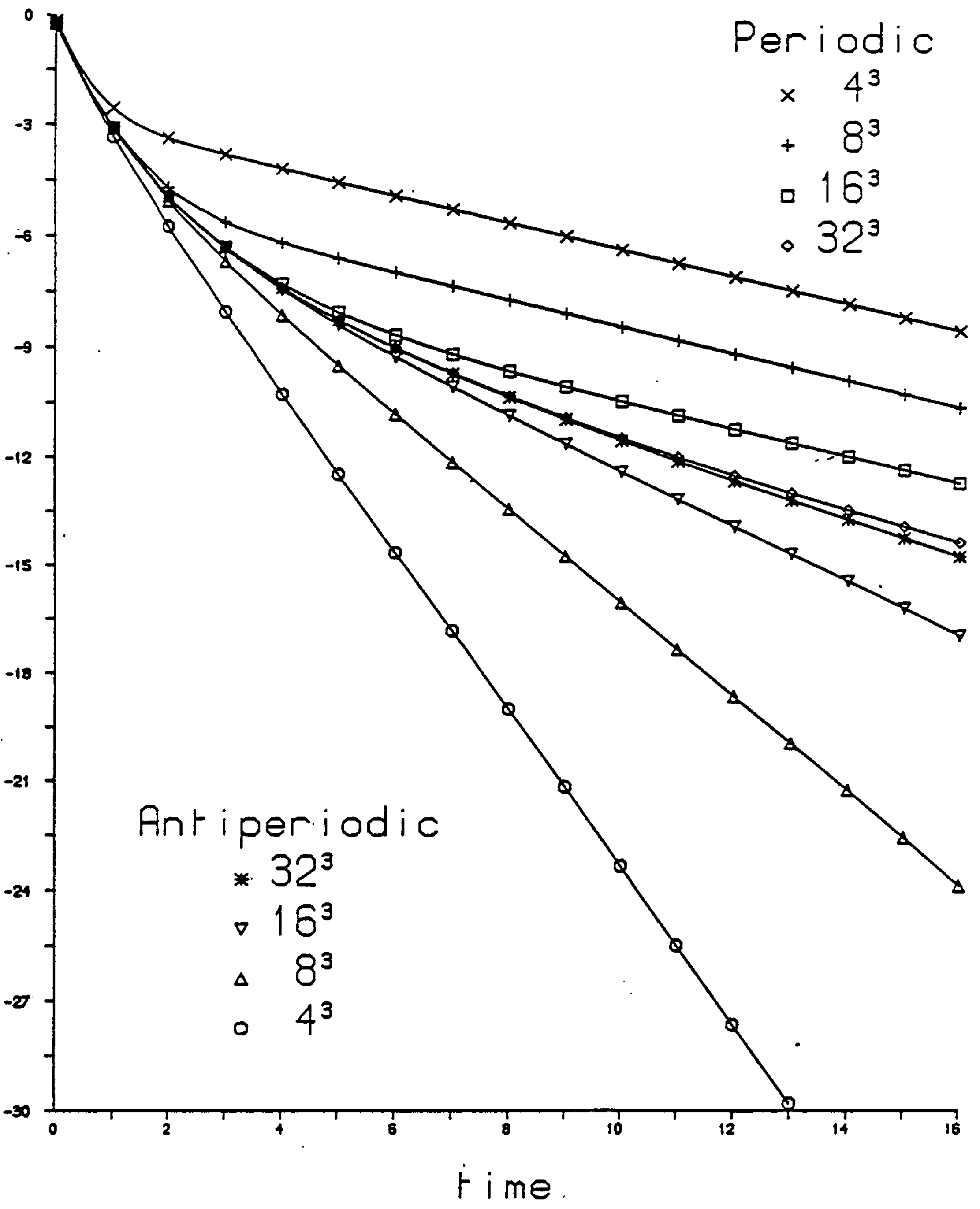


Figure 3.13: Free Wilson fermion propagator as a function of spatial lattice size and boundary conditions.

6.15. Particularly at small times the discrepancies between the two propagators increase with decreasing mass. The magnitude of the periodic propagator is always higher, and the masses obtained lower than those using anti-periodic boundary conditions. We are forced to conclude that there are significant finite-

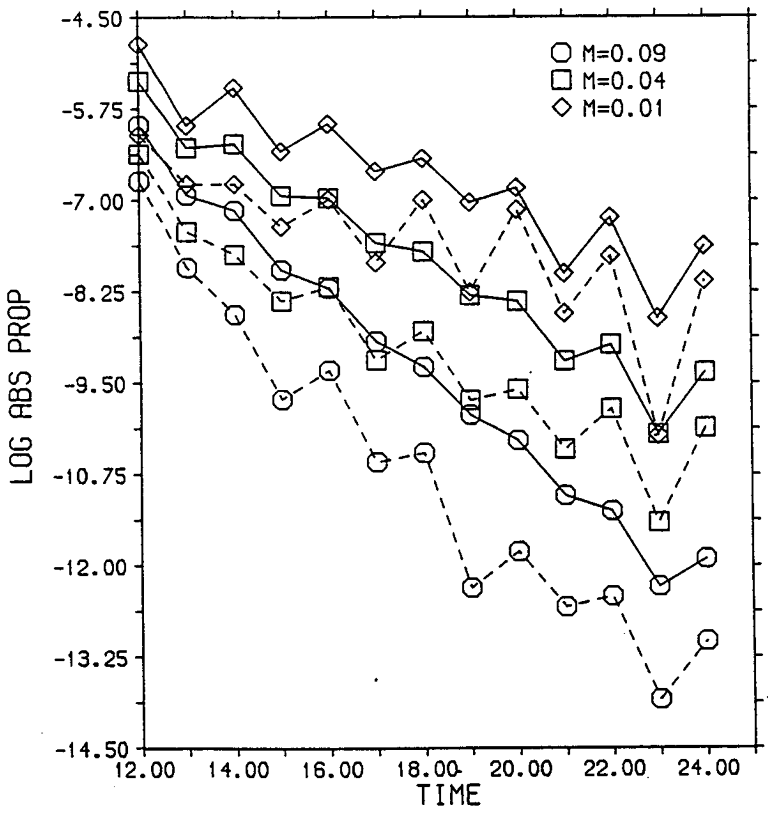


Figure 3.14: Staggered fermion nucleon propagator as a function of spatial boundary conditions and mass at $\beta = 6.15$ (periodic is solid line).

size effects in our $\beta = 6.15$ baryon data.

Mass Ratios We have computed the proton-to-rho and pion-to-rho mass ratios using the baryon data from the periodic propagators. The mass ratios are given in table 3.15 and are plotted in figure 3.15 In spite of the finite-size effects there is clear evidence for a turn-over to the light-quark regime. If we extrapolate the mass ratios to the physical value of $\frac{m_\pi}{m_\rho}$ using a linear fit to the last 3 points we obtain $\frac{m_N}{m_\rho} = 1.50 \pm 0.09$. The lowest value obtained for $\frac{m_\pi}{m_\rho}$, of 0.53 is rather too high to allow us to attach great weight to this prediction.

m_q	$\frac{m_\pi}{m_\rho}$	$\frac{m_N}{m_\rho}$
0.50	0.93(1)	1.59(1)
0.16	0.90(1)	1.61(4)
0.09	0.87(1)	1.61(3)
0.04	0.78(1)	1.59(2)
0.01	0.55(5)	1.51(10)

Table 3.15: Mass ratios at $\beta = 6.15$ for $16^3 \times 24$ lattice. Fermion boundary conditions are periodic in the spatial directions.

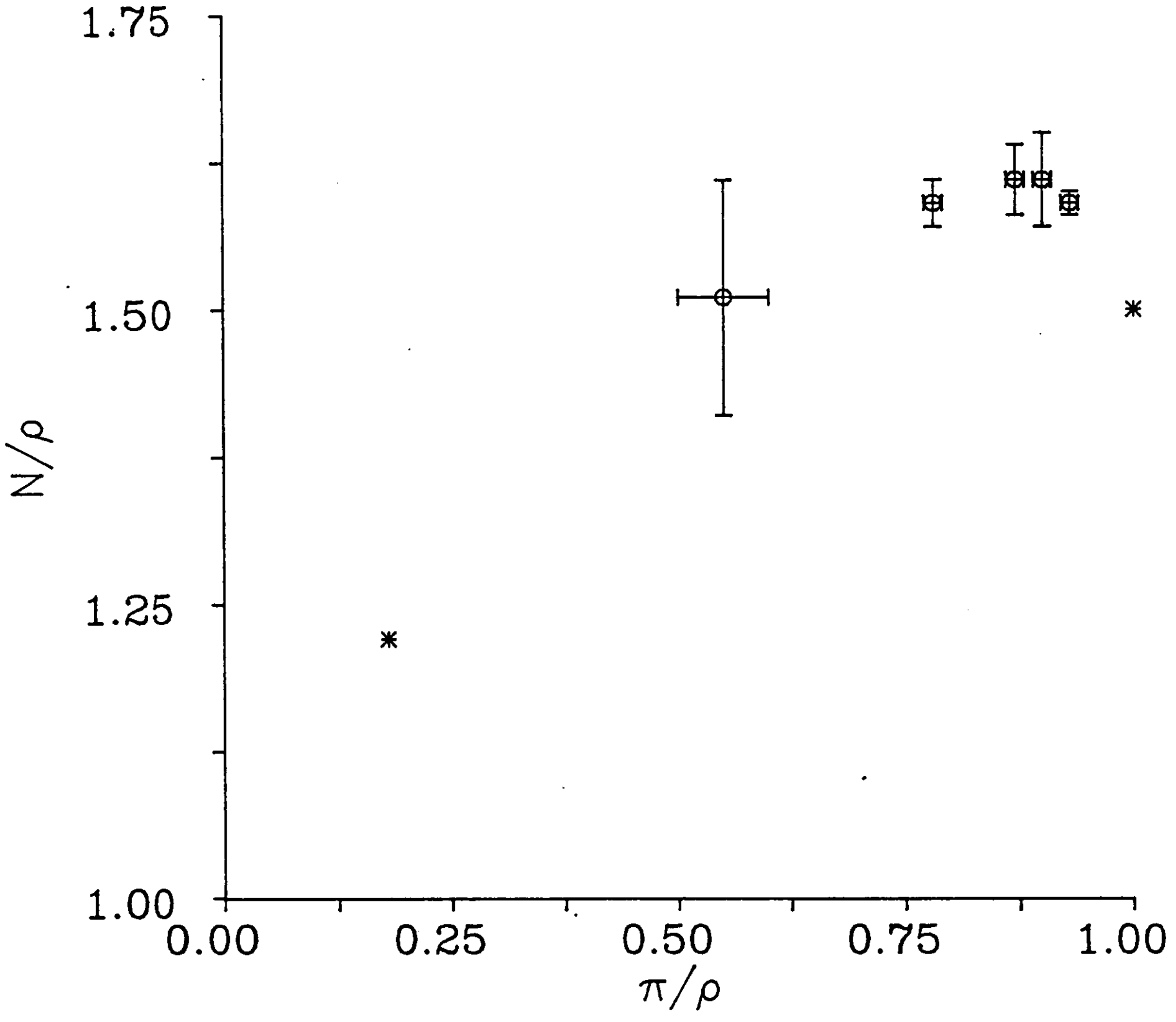


Figure 3.15: Mass ratios at $\beta = 6.15$ for $16^3 \times 24$ lattice. The fermion boundary conditions are periodic in the spatial directions.

3.4.6 Hadron Masses at $\beta = 6.3$

Our $\beta = 6.3$ results are based on analysis of 32 16^4 configurations extended periodically in time to $16^3 \times 24$ with masses in the range $0.01 \leq m \leq 0.50$ and 4 configurations with $m = 0.0025$ and $m = 0.0$. The mass fits are given in table 3.16. A 16^4 lattice at $\beta = 6.0$ is approximately the same physical size as an 11^4 lattice at $\beta = 6.3$. Given this we did not expect sensible baryon data given the degree of finite size effects in the $\beta = 6.15$ results, but aim to study the mesons.

The Mesons

2-exponential fits are necessary to expose the ground state of the pseudoscalar propagator. A linear fit to the data for $0.01 \leq m \leq 0.16$ extrapolates to $m_\pi = 0$ within relatively large errors but the curvature of $m_\pi(\sqrt{m_q})$ is significantly greater than at $\beta = 6.15$. A 3-parameter fit to the masses gives

$$m_\pi = (0.11 \pm 0.04) + (1.15 \pm 0.13)\sqrt{m_q} + (1.98 \pm 0.13)m_q$$

(see figure 3.16). The low statistics results at $m = 0.0025$ and $m = 0.0$ are consistent with this curve. Convergence of these low mass runs was rapid, implying that finite size effects give $\not{D}^2$ a large leading eigenvalue ($\lambda_1 \gg m_q$ for small m_q).

m_q	PS	SC	VT	PV
0.50	1.63(1)	1.77(2)	1.76(2)	1.79(2)
0.16	0.89(1)	0.94(3)	0.98(1)	0.97(1)
0.09	0.64(1)	0.66(2)	0.72(1)	0.72(1)
0.04	0.42(2)	0.40(1)	0.50(3)	0.49(3)
0.01	0.25(3)	0.15(2)	0.33(6)	0.45(5)

Table 3.16: Meson mass estimates at $\beta = 6.3$.

The results of 2-exponential fits to the rho from VT and PV propagators are given in table 3.16; they are in excellent agreement. We observe that the pion and rho masses are nearly degenerate; this has been seen before on smaller lattices at lower β [Bowler et. al. 1984a]. In the absence of finite-size effects the Goldstone pion is the lightest particle. At $m_q = 0.01$ the lightest particle is the pion from the SC propagator. This operator seems to be less affected by the finite size effects that push up the mass of the PS pion. At intermediate masses flavour symmetry is better than at lower β.

The Baryons

The mass fits to the EVEN and ALL propagators are similar within errors, but the errors are very large. The errors in our mass ratios are too high to draw

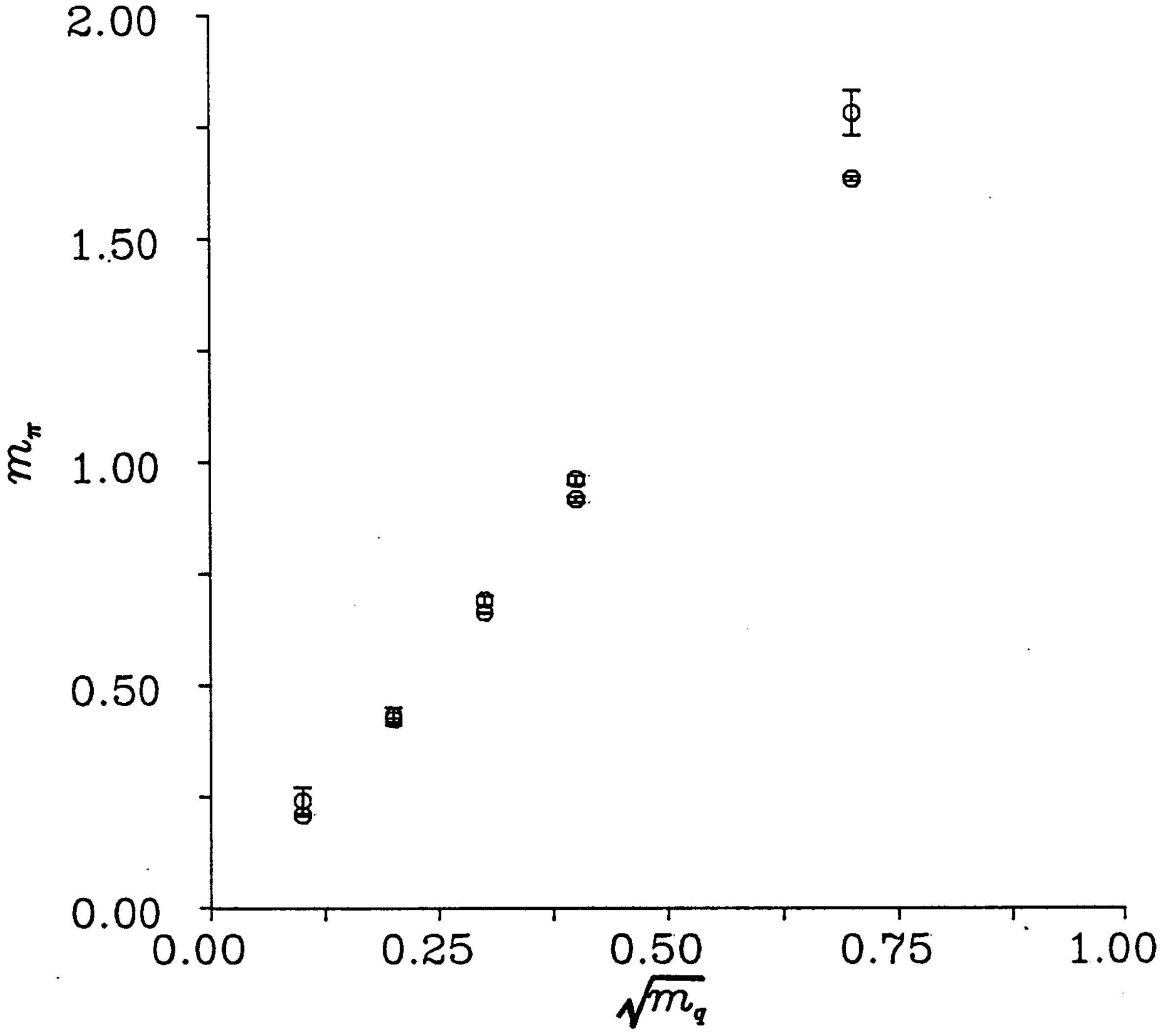


Figure 3.16: Meson mass estimates at $\beta = 6.3$.

conclusions. The finite m_π as $m_q \to 0$ keeps $\frac{m_\pi}{m_\rho}$ high, pushing all the data away from the physical light quark regime.

3.5 Conclusions

Data Fitting Fitting to sums of exponentials is not a well conditioned problem. It is preferable to fit to ground states (only) removing excitations by dropping timeslices close to the source. We found it necessary to use lattice of temporal extent greater than 16 for this procedure to stabilise; 24 timeslices seems appropriate for Dirichlet boundary conditions in time. Propagator decay over more than 16 timeslices is difficult to see; it would require greatly improved statistics. The signals are clearer for periodic boundary conditions making fiting easier.

Restoration of Flavour Symmetry Our estimates of m_π and m_ρ are each taken from two propagators with different flavour components. At $\beta = 5.7$ the masses obtained from the two channels differ by several standard deviations. At $\beta = 6.0$ there are signs of flavour symmetry restoration, especially in the rho. At $\beta = 6.15$ and $\beta = 6.3$ we see good agreement, in both mesons at intermediate mass. The pion is a better test as only in the continuum limit will the non-Goldstone pions be driven to zero mass at zero quark mass.

Finite Lattice Size Effects A consistent picture of finite size effects emerges from our data. At low β the lattice spacing is large and so the lattice is of sufficient size to contain the hadrons. The EVEN=ALL identity holds for the baryons, and the pion masses extrapolate linearly to zero. As we increase β the lattice spacing decreases, and with it the size of the lattice. The pion masses no longer extrapolate to zero and $\not{D}^2$ acquires a large leading eigenvalue (relative to the mass). Failure of the EVEN=ALL identity and dependence on spatial boundary conditions are identified as effects of calculating baryon propagators in too small a box. As expected both effects get worse as we lower the quark mass.

Mass Ratios At low β our mass ratios are in line with earlier work. There is no crossover to the physical regime; $\frac{m_N}{m_\rho}$ increases as $m_q \to 0$. As we increase β the mass ratios for low m_q fall towards the physical value. In figure 3.17 we show our mass ratios for $\beta = 6.0$ and $\beta = 6.15$ and those of [Barkai et. al. 1985] for $\beta = 6.0$. If the hadron masses are scaling then the data should lie on a single universal curve. Our results are consistent with there being one curve, but the errors are too large to make definitive statements.

Future Prospects Our data suggests that there is a window between $\beta = 6.0$ and $\beta = 6.15$ in which we can work, free from the effects of strong coupling and free from finite-size effects in the mesons (at least). We observe flavour symmetry

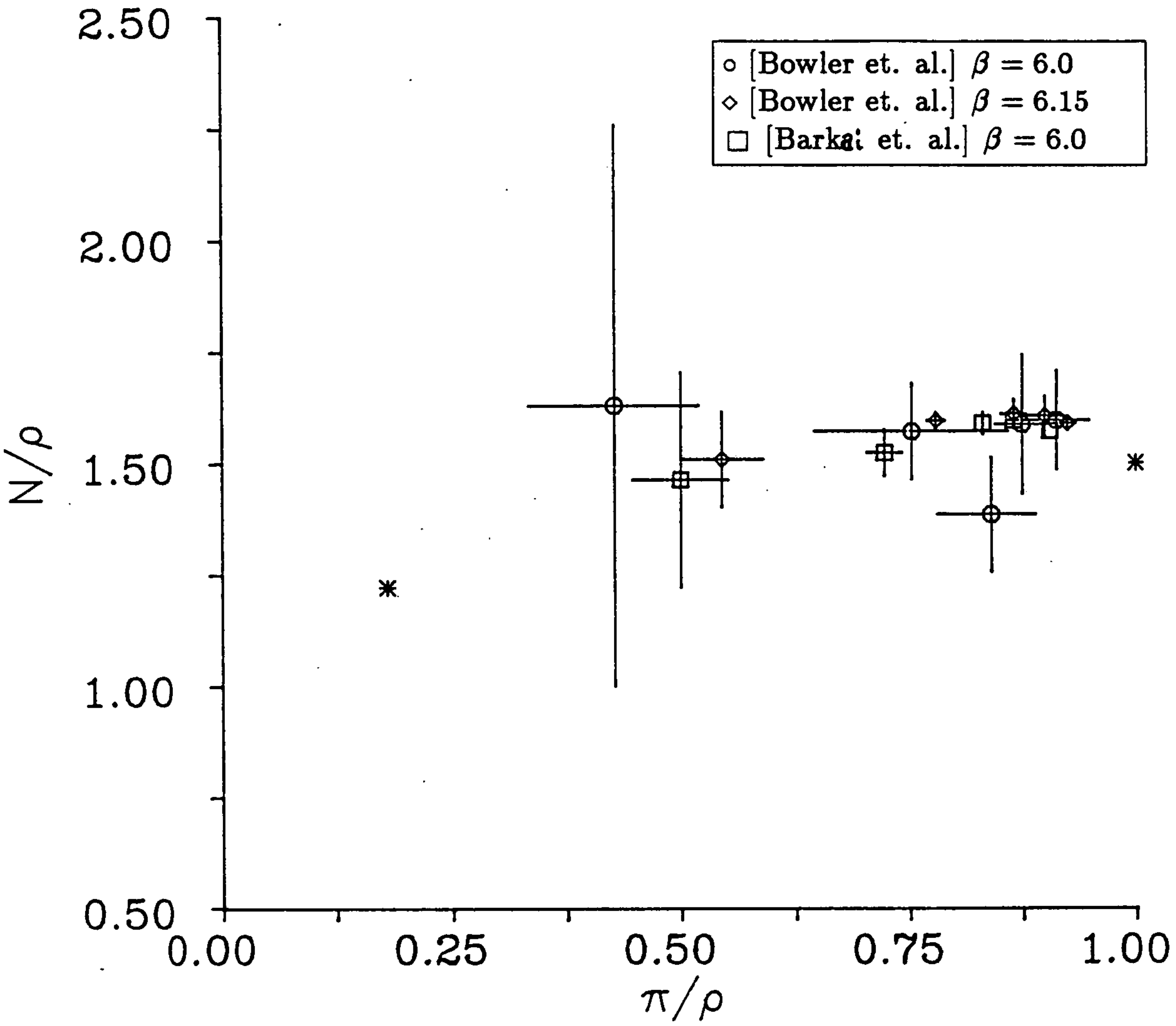


Figure 3.17: Summary of mass ratio data from large lattice simulations.

restoration and reasonable mass ratios in this range of β. In order to improve on our results it will be necessary to use lower masses (to bring $\frac{m_\pi}{m_\rho}$ closer to the physical value) and larger lattices (to cut baryon finite size-effects). Higher statistics are also desirable to improve the quality of error estimation, and to cut the errors. If we are to extract definitive results we must also show independence of choice of boundary conditions; we have only demonstrated this for the mesons.

We find that the rate of convergence of the CG algorithm is a good guide to the magnitude of finite-size effects in the pion data. If convergence rates remain linear in m_q as m_q falls then the leading eigenvalue of $\not{D}^2$ is small, and the pion is free from finite size effects. This can easily be tested while propagators are being generated; it is not necessary to wait until there is a large sample of data to average over.

To obtain the maximum useful information we should work at the lowest mass for which this condition is met. For this to be practical it is essential to find efficient preconditioners that cut the computational cost of solving the systems of equations.

Since we started our study (some 2 years ago) other groups (notably [Konig et. al. 1984], [Itoh et. al. 1986] and [Gupta et. al. 1987]) have begun similar work on larger lattices. Advances in computer performance over this period of time make our suggestions feasible.

Chapter 4

Equation of Motion Algorithms

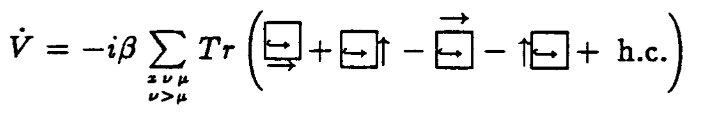
In this chapter we discuss the application of "Equation of Motion" schemes for the evolution of statistical mechanical systems to lattice gauge theory. These algorithms enable us to update all the degrees of freedom (fermionic as well as gauge) at the same time. This makes the study of fermionic systems possible; QCD simulations on small lattices are underway, and are reviewed in section 4.3.

In section 4.1 we consider the molecular dynamics (MD), Langevin, and hybrid algorithms, setting them up for the pure theory. Fermions are added in section 4.2. In section 4.4 we discuss a hybrid monte carlo algorithm. In order to measure their usefulness as a function of computer time required we compare equilibration times and stepsize errors.

Our numerical results are for compact QED with staggered fermions. This theory allows us to set up and test the systematics of the algorithms on lattices of moderate size without the high computational requirements of the full fermionic $SU(3)$ theory (we do of course still have to solve a large system of linear equations once per timestep). This model has been looked at analytically in the strong coupling limit by [Jolicoeur et. al. 1984] and numerically using pseudo-fermion techniques and a $Z(12)$ approximation to the gauge group by [Azcoiti et. al. 1986]. The implementation of our simulation on an array of Transputers is discussed in section 4.5.2 and our results are presented in section 4.6.

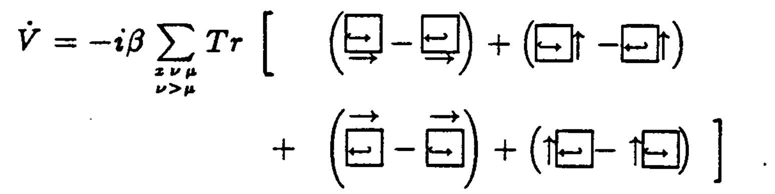
4.1 A Review of Current Techniques

We begin by describing the use of a molecular dynamics algorithm for lattice gauge theory simulations [Callaway and Rahman 1982] and [Polonyi and Wyld 1983]. One constructs a hamiltonian for the evolution of a gauge theory in an extra time dimension, simulation time. The algorithm is similar to that used widely in chemical physics with the added constraint that the link variables must remain members of the gauge group. The MD algorithm is set up for a where the lattice site x is at the bottom left corner of the box. The arrow emerges from the site at which the momentum term is inserted and points in the direction of the momentum. Making use of the cyclic properties of the trace



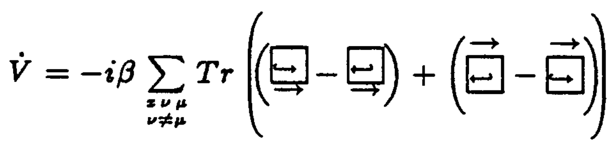
(4.7)

Putting in the hermitean conjugate terms gives



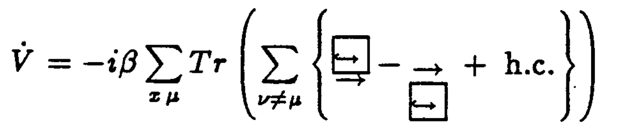
(4.8)

Removing the restriction $\nu > \mu$ leaves us with just two pairs of diagrams (the others are obtained by reflecting in the $\nu = \mu$ axis).



(4.9)

Now rearrange the summation of the second term by shifting from $x + \hat{\nu}$ to x

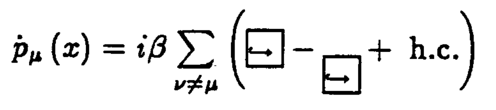


(4.10)

the lowered box represents a plaquette at $x - \hat{\nu}$.

$$\dot{T} = \sum_{x\,\mu} Tr\left(p_\mu(x)\,\dot{p}_\mu(x)\right) \tag{4.11}$$

and so eq. 4.3 implies



(4.12)

The dynamics associated with eq. 4.1 are now fully specified.

In order to integrate the equations numerically we discretise them by introducing a finite timestep dt. We evolve the fields using the "leapfrog" algorithm (see figure 4.1). The discretisation errors are estimated below

$$\begin{aligned} p(t + \tfrac{dt}{2}) - p(t - \tfrac{dt}{2}) &= \int_{t-\frac{dt}{2}}^{t+\frac{dt}{2}} \dot{p}(t')dt' & (4.13) \\ &= \dot{p}dt + O(dt^3) & (4.14) \end{aligned}$$

$$\begin{aligned} U(t + dt) &= \exp\left(i\int_{t}^{t+dt} p(t')dt'\right) U(t) & (4.15) \\ &= \exp\left(ip(t + \tfrac{dt}{2})dt\right) U(t) + O(dt^3) & (4.16) \end{aligned}$$

pure $SU(N)$ theory. (We use a simple model, a single scalar q, to illustrate the Langevin and hybrid algorithms.)

4.1.1 A Molecular Dynamics Algorithm for a Pure $SU(N)$ Lattice Gauge Theory

Consider the hamiltonian

$$H = S_g + \frac{1}{2}\sum_{x\,\mu} Tr\left(p_\mu^2(x)\right) = V + T \tag{4.1}$$

S_g is the usual pure gauge action

$$S_g = -\beta \sum_{\substack{x\,\nu\,\mu \\ \nu>\mu}} Tr\left(U_\mu(x)\,U_\nu(x+\hat{\mu})\,U_\mu^\dagger(x+\hat{\nu})\,U_\nu^\dagger(x) + \text{h.c.}\right) \tag{4.2}$$

where the $U_\mu(x)$ are elements of the gauge group $SU(N)$. $\beta = \frac{2N}{g^2}$ and all plaquettes are oriented as shown in figure 1.1.

Our hamiltonian eq. 4.1 is not the same as the pure gauge hamiltonian obtained from the lagrangian associated with eq. 4.2, rather it governs the evolution of our system in an extra time dimension τ, the computer time for which we run the simulation. As usual the hamiltonian H is a constant of the motion

$$\dot{H} = \frac{\partial H}{\partial p_\mu}\dot{p}_\mu + \frac{\partial H}{\partial U_\mu}\dot{U}_\mu = 0 \tag{4.3}$$

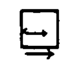
We use this to deduce an equation of motion for the gauge fields. In addition we require that the U's remain elements of the gauge group. This is ensured by requiring that their equation of motion has the form

$$\dot{U}_\mu(x) = ip_\mu(x)\,U_\mu(x) \tag{4.4}$$

[Duane et al 1986] with $p_\mu(x)$ hermitean and traceless. The p's play the role of conjugate momenta in this scheme. From eq. 4.3

$$\begin{aligned}\dot{V} = -i\beta \sum_{\substack{x\,\nu\,\mu \\ \nu>\mu}} \; Tr\; \Big[& \; p_\mu(x)\,U_\mu(x)\,U_\nu(x+\hat{\mu})\,U_\mu^\dagger(x+\hat{\nu})\,U_\nu^\dagger(x) \\ + & \; U_\mu(x)\,p_\nu(x+\hat{\mu})\,U_\nu(x+\hat{\mu})\,U_\mu^\dagger(x+\hat{\nu})\,U_\nu^\dagger(x) \\ - & \; U_\mu(x)\,U_\nu(x+\hat{\mu})\,U_\mu^\dagger(x+\hat{\nu})\,p_\mu(x+\hat{\nu})\,U_\nu^\dagger(x) \\ - & \; U_\mu(x)\,U_\nu(x+\hat{\mu})\,U_\mu^\dagger(x+\hat{\nu})\,U_\nu^\dagger(x)\,p_\nu(x) \\ + & \; \text{h.c.} \Big]\end{aligned} \tag{4.5}$$

We can express this diagramatically by defining



$$\equiv p_\mu(x)\,U_\mu(x)\,U_\nu(x+\hat{\mu})\,U_\mu^\dagger(x+\hat{\nu})\,U_\nu^\dagger(x) \tag{4.6}$$

$$\begin{tikzcd}
\text{stage} & t-\frac{dt}{2} & t & t+\frac{dt}{2} & t+dt \\
 & & U_\mu(x) \arrow[d] & & \\
1. & & \dot{p}_\mu(x) \arrow[d] & & \\
2. & p_\mu(x) \arrow[rr] & {} & p_\mu(x) \arrow[d] & \\
3. & & U_\mu(x) \arrow[rr] & & U_\mu(x)
\end{tikzcd}$$

Figure 4.1: Evolution using "leapfrog" algorithm.

The coordinate fields are defined at times $0, dt, 2dt, \ldots$ and the momenta at times $\frac{dt}{2}, \frac{3dt}{2}, \ldots$. The errors in the links and momenta are of $O(dt^3)$ using "leapfrog", better than those for "Verlet" or low order Runge Kutta algorithms. Leapfrog has the further advantage that the link and momenta updates both involve the addition of quantities of order dt; there are no additions of quantites of order dt^2 - that would cause additional round-off errors. The system is evolved in three stages:

1. calculate $\dot{p}(t)$ from $U(t)$.
2. $p(t+\frac{dt}{2}) = p(t-\frac{dt}{2}) + dt\dot{p}(t)$,
3. $U(t+dt) = \exp\left(ip(t+\frac{dt}{2})\right) U(t)$.

The timestep errors can be tested by calculating the total energy. To obtain the kinetic energy we must either "half step" the momenta backwards to obtain $p_\mu(x,t)$ or average $p_\mu(x,t-\frac{dt}{2})$ and $p_\mu(x,t+\frac{dt}{2})$. The potential energy can be calculated using the plaquette data required for updating the momenta.

4.1.2 The Langevin Algorithm

In this section and the next we use a model with one scalar degree of freedom q; the generalisation to a gauge theory is straightforward. Let the probability that q takes a certain value be given by the Boltzmann weight $e^{-S(q)}$. We want to measure expectation values of functions of q in the canonical ensemble,

$$\langle F(q) \rangle = \frac{1}{Z} \int dq F(q) e^{-S(q)} \tag{4.17}$$

We can evaluate such integrals by constructing a stochastic dynamics for q [Parisi and Wu 1981] and measuring ensemble averages.The simplest such dynamics is the Langevin equation

$$\frac{\partial q}{\partial \tau} = -\frac{\partial S}{\partial q} + \eta(\tau) \tag{4.18}$$

where $\eta(\tau)$ is white noise, i.e.

$$\langle \eta(\tau) \rangle = 0 \text{ and } \langle \eta(\tau)\eta(\tau') \rangle_\eta = 2\delta(\tau - \tau') \tag{4.19}$$

We first set up a Fokker-Planck equation to show that the probability distribution for $q(\tau)$ becomes the Boltzmann distribution as $\tau \to \infty$. Write eq. 4.18 as a difference equation for q

$$\begin{aligned} q_{n+1} &= q_n - d\tau \frac{\partial S}{\partial q} + \sqrt{d\tau}\eta \\ &= q_n - \Delta q \end{aligned} \tag{4.20}$$

where $q_n = q(\tau_n)$ and $q_{n+1} = q(\tau_n + d\tau)$. In numerical studies this differencing would be replaced by a more accurate procedure (second order Runge-Kutta, say). Consider the time evolution of the probability distribution $P(q_n, \tau_n)$,

$$P(q_{n+1}, \tau_{n+1}) = \left\langle \int dq_n P(q_n, \tau_n) \delta(q_{n+1} - q_n + \Delta q) \right\rangle_\eta \tag{4.21}$$

Now $\Delta q \to 0$ as $d\tau \to 0$ so expand in Δq

$$\begin{aligned} P(q_{n+1}, \tau_{n+1}) &= \Bigg\langle \int dq_n P(q_n, \tau_n) \Big(\delta(q_{n+1} - q_n) \\ &\quad -\Delta q \frac{\partial}{\partial q} \delta(q_{n+1} - q_n) + \frac{1}{2} (\Delta q)^2 \frac{\partial^2}{\partial q^2} \delta(q_{n+1} - q_n) + \cdots \Big) \Bigg\rangle_\eta \\ &= P(q_{n+1}, \tau_n) - \left\langle \frac{\partial}{\partial q} P \Delta q \right\rangle_\eta + \frac{1}{2} \left\langle \frac{\partial^2}{\partial q^2} P (\Delta q)^2 \right\rangle_\eta + \cdots \end{aligned} \tag{4.22}$$

We can also Taylor expand $P(q_{n+1}, \tau_{n+1})$ as

$$P(q_{n+1}, \tau_{n+1}) = P(q_{n+1}, \tau_n) + \left. \frac{\partial P}{\partial \tau} \right|_{\tau_n} d\tau + \cdots \tag{4.23}$$

Equating powers of $d\tau$

$$\frac{\partial P}{\partial \tau} = \left\langle \frac{\partial}{\partial q} P \frac{\partial S}{\partial q} \right\rangle_\eta + \frac{1}{2} \left\langle \frac{\partial^2}{\partial q^2} P \eta^2 \right\rangle_\eta + O(d\tau^2) \tag{4.24}$$

using the properties of the noise η (see eq. 4.19) we have

$$\frac{\partial P}{\partial \tau} = \frac{\partial}{\partial q} \left(P \frac{\partial S}{\partial q} \right) + \frac{\partial^2}{\partial q^2} P \tag{4.25}$$

to $O(d\tau)$. The Boltzmann distribution $P = e^{-S(q)}$ is a stationary point of this Fokker-Planck equation.

To obtain the rate of approach to this stationary point we construct an "imaginary-time Schrödinger equation" $-\frac{\partial}{\partial\tau}\Psi = H\Psi$. To do this define a "wave-function"

$$P(q,\tau) = e^{-\frac{1}{2}S(q)}\Psi(q,\tau) \qquad (4.26)$$

and substitute it into eq. 4.25

$$-\frac{\partial}{\partial\tau}\Psi(q,\tau) = \left(-\frac{\partial^2}{\partial q^2} + \frac{1}{4}\left(\frac{\partial S}{\partial q}\right)^2 - \frac{1}{2}\frac{\partial^2 S}{\partial q^2}\right)\Psi(q,\tau) \qquad (4.27)$$

$$= \left(\frac{\partial}{\partial q} + \frac{1}{2}\frac{\partial S}{\partial q}\right)^\dagger \left(\frac{\partial}{\partial q} + \frac{1}{2}\frac{\partial S}{\partial q}\right)\Psi(q,\tau) \qquad (4.28)$$

If we write the general solution to eq. 4.27 as an expansion in the eigenstates ϕ_i of the energy operator H,

$$\left(-\frac{\partial^2}{\partial q^2} + \frac{1}{4}\left(\frac{\partial S}{\partial q}\right)^2 - \frac{1}{2}\frac{\partial^2 S}{\partial q^2}\right)\phi_i = \lambda_i\phi_i \qquad (4.29)$$

then the general solution to our Fokker-Planck equation can be written

$$P(q,\tau) = \text{const}\ \phi_0(q)\left(\phi_0(q) + \sum_{i=1}^{\infty} c_i e^{-\lambda_i\tau}\phi_i(q)\right) \qquad (4.30)$$

where $\phi_0 = e^{-\frac{1}{2}S(q)}$. The operator H is positive semi-definite from eq. 4.28 and so the λ_i are real and positive. The rate of convergence is exponential, and is controlled by the first non-zero eigenvalue of the energy operator. $\lambda_0 = 0$ is a unique ground state of eq. 4.29 and so the stationary point $P = e^{-S(q)}$ is the equilibrium distribution. Time averages calculated using Langevin dynamics will agree with the ensemble averages we require.

4.1.3 The Hybrid Algorithm

The MD algorithm of section 4.1.1 is similar to that used extensively in chemical physics. It is well known that such algorithms sometimes fail to represent the canonical ensemble correctly; the exchange of energy between modes is over too short a timescale for thermalisation to occur.

If the evolution is not ergodic then the measured time averages will be wrong (different from the ensemble averages). We can detect this "ergodicity breaking" in simple systems, but detection becomes much more difficult as the number of degrees of freedom grows. Very long time tails, and incorrect distribution of energies will signal the problem.

The Langevin algorithm avoids these problems by the addition of noise, which guarantees ergodicity. However, phase space is sampled at a rate proportional to the square root of the number of timesteps.

Figure 4.2 characterises the dynamics of the MD and Langevin algorithms. For MD the system rapidly traverses smooth paths through phase space whereas for Langevin dynamics it walks randomly.

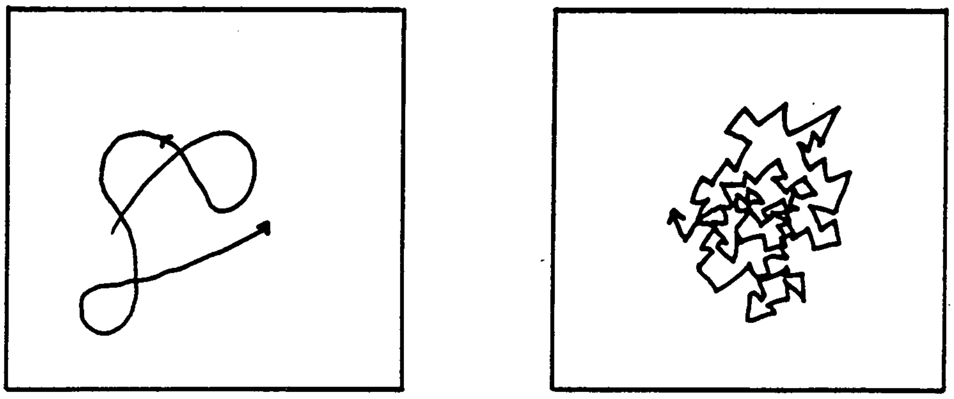


Figure 4.2: Motion through phase space for MD and Langevin algorithms.

The hybrid algorithm [Duane 1985] combines the best properties of both MD and Langevin: the rapid evolution through phase space and the guarantee of ergodicity. To see the relationship between the two, we write them as discrete difference equations.

Langevin:

$$q_{n+1} = q_n + \Delta\xi - \frac{1}{2}\Delta^2\frac{\partial S}{\partial q} \tag{4.31}$$

where the noise $\langle \xi_n \xi_{n'} \rangle = \delta_{nn'}$ and the timestep is $\tau_{n+1} - \tau_n = \frac{1}{2}\Delta^2$.

Microcanonical:

$$\begin{aligned} q_{n+1} &= 2q_n - q_{n-1} - \Delta^2\frac{\partial S}{\partial q} \\ &= q_n + \frac{1}{2}\left(q_{n+1} - q_{n-1}\right) - \frac{1}{2}\Delta^2\frac{\partial S}{\partial q} \end{aligned} \tag{4.32}$$

where $\tau_{n+1} - \tau_n = \Delta$.

The following analogies can be drawn.

Langevin		Microcanonical
noise	$\Longleftrightarrow$	velocity
$\tau_{n+1} - \tau_n = \frac{1}{2}\Delta^2$	$\Longleftrightarrow$	$\tau_{n+1} - \tau_n = \Delta$

(4.33)

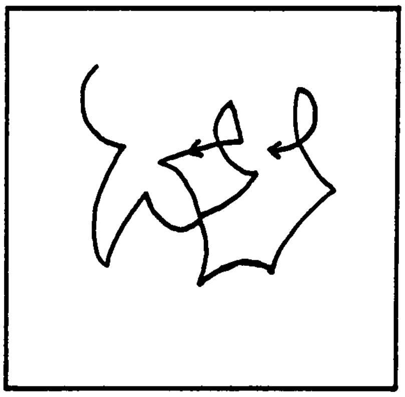


Figure 4.3: Motion through phase space for hybrid algorithm.

In the hybrid scheme we make either a Langevin step (probability $p\Delta$) or an MD step (probability $1 - p\Delta$)

$$q_{n+1} = q_n + \Delta v_n - \frac{1}{2}\Delta^2 \frac{\partial S}{\partial q} \tag{4.34}$$

with

$$v_n = \left\{ \begin{array}{c} \frac{1}{2\Delta}(q_{n+1} - q_n) \\ \xi_n \end{array} \right\} \text{probability} \left\{ \begin{array}{c} 1 - p\Delta \\ p\Delta \end{array} \right\} \tag{4.35}$$

The parameter p, "hits per unit time" should be optimised. Duane has shown that for simple systems the best choice for p is twice the frequency of the slowest mode of the system (a plot of kinetic energy as a function of time will give this information in an MD simulation). We schematically represent the hybrid dynamics in figure 4.3.

4.2 Including Dynamical Fermions.

The inclusion of dynamical fermions in traditional monte carlo schemes is difficult, and very expensive in terms of computer time. At best it is necessary to calculate columns of the inverse of the fermion matrix every time we wish to update some small number of links. Equation of motion schemes enable us to update all the fermion fields by solving one system of linear equations. Consider the hamiltonian (in computer time)

$$\begin{aligned} H &= V+T && (4.36)\\ V &=\beta \sum_{\substack{x\nu\mu\\ \nu>\mu}} Tr\left(\boxed{\vdash}+ \text{h.c.}\right) + \sum_{x\,y} \psi^{\dagger}(x)\left(-\not{D}^2+M^2\right)^{-1}\psi(y) && (4.37)\\ T &= \frac{1}{2}\sum_{x\mu} p_{\mu}^2(x) + \omega^{-2}\sum_{x}\pi^{\dagger}(x)\pi(x) && (4.38)\end{aligned}$$

where ψ is a 'bosonised' fermion field living on the even sites of the lattice (see section 1.2.1) and π its conjugate momenta. Now

$$\begin{aligned} \dot{V} = \dot{V}_G &+ \sum_{x\,y}\dot{\psi}^{\dagger}(x)\left(-\not{D}^2+M^2\right)^{-1}\psi(y) && (4.39)\\ &+ \sum_{x\,y}\psi^{\dagger}(x)\left(-\not{D}^2+M^2\right)^{-1}\dot{\psi}(y)\\ &- \sum_{x\,y}\psi^{\dagger}(x)\left(-\not{D}^2+M^2\right)^{-1}\left[\frac{d}{dt}\left(-\not{D}^2+M^2\right)\right]\left(-\not{D}^2+M^2\right)^{-1}\psi(y)\\ \dot{T} = \dot{T}_G &+ \omega^{-2}\sum\left(\dot{\pi}^{\dagger}\pi + \pi^{\dagger}\dot{\pi}\right) && (4.40)\end{aligned}$$

where $\dot{V}_G$ and $\dot{T}_G$ are the contributions to $\dot{V}$ and $\dot{T}$ from the pure gauge term. The first two fermionic terms in $\dot{V}$ govern the dynamics of the fermion fields, the third, $\dot{V}_3$, contributes to $\dot{p}$. The equations of motion for the fermi sector are

$$\begin{aligned} \dot{\psi} &= \frac{\partial H}{\partial \pi^{\dagger}} = \omega^{-2}\pi\\ &\Rightarrow \psi \longrightarrow \psi + \frac{dt}{\omega^2}\pi && (4.41)\\ -\dot{\pi} &= \frac{\partial H}{\partial \psi^{\dagger}} = -\left(-\not{D}^2+M^2\right)^{-1}\psi\\ &\Rightarrow \text{solve } \left(-\not{D}^2+M^2\right)\phi = \psi\\ &\qquad \pi \longrightarrow \pi - dt\phi && (4.42)\end{aligned}$$

The fermionic contribution to the gauge update is

$$\begin{aligned} \dot{V}_3 &= -\sum_{x\,y}\psi^{\dagger}(x)\left(-\not{D}^2+M^2\right)^{-1}\left[\frac{d}{dt}\left(-\not{D}^2+M^2\right)\right]\left(-\not{D}^2+M^2\right)^{-1}\psi(y)\\ &= \sum_{x\,y}\phi^{\dagger}(x)\left[\dot{\not{D}}^{\dagger}\not{D} + \not{D}^{\dagger}\dot{\not{D}}\right]\phi(y) && (4.43)\end{aligned}$$

since $\phi = (-\not{D}^2+M^2)^{-1}\psi$ (note $\phi = 0$ on odd sites), and so

$$\dot{V}_3 = \sum_{x}\left(\dot{\not{D}}\phi\right)^{\dagger}\not{D}\phi + (\not{D}\phi)^{\dagger}\dot{\not{D}}\phi = 2\mathcal{R}e\sum_{x}(\not{D}\phi)^{\dagger}\dot{\not{D}}\phi \qquad (4.44)$$

Now

$$(\not{D}\phi)(x) = \frac{1}{2}\sum_{\mu} \eta_\mu(x)\left\{U_\mu(x)\,\phi(x+\hat{\mu}) - U^\dagger_\mu(x-\hat{\mu})\phi(x-\hat{\mu})\right\} \tag{4.45}$$

Therefore

$$\left(\dot{\not{D}}\phi\right)(x) = \frac{1}{2}\sum_{\mu} \eta_\mu(x)\, i\left\{p_\mu(x)\,U_\mu(x)\,\phi(x+\hat{\mu}) + p_\mu(x-\hat{\mu})U^\dagger_\mu(x-\hat{\mu})\phi(x-\hat{\mu})\right\} \tag{4.46}$$

and so eq. 4.44 becomes

$$\begin{aligned} \dot{V}_3 &= \mathcal{R}e\sum_{x\,\mu} \eta_\mu(x)\, ip_\mu(x)\left\{(\not{D}\phi)^\dagger(x)U_\mu(x)\,\phi(x+\hat{\mu})\right\} \\ &+ \mathcal{R}e\sum_{x\,\mu} \eta_\mu(x)\, ip_\mu(x-\hat{\mu})\left\{(\not{D}\phi)^\dagger(x)U^\dagger_\mu(x-\hat{\mu})\phi(x-\hat{\mu})\right\} \end{aligned} \tag{4.47}$$

Now shift from $x-\hat{\mu}$ to x in the second term

$$\begin{aligned} \dot{V}_3 &= \mathcal{R}e\sum_{x\,\mu} \eta_\mu(x)\, ip_\mu(x)\left\{(\not{D}\phi)^\dagger(x)U_\mu(x)\,\phi(x+\hat{\mu})\right\} \\ &+ \mathcal{R}e\sum_{x\,\mu} \eta_\mu(x)\, ip_\mu(x)\left\{(\not{D}\phi)^\dagger(x+\hat{\mu})U^\dagger_\mu(x)\,\phi(x)\right\} \end{aligned} \tag{4.48}$$

Define

$$\hat{\phi} = \begin{cases} \phi & \text{on even sites} \\ \not{D}\phi & \text{on odd sites} \end{cases} \tag{4.49}$$

$$\begin{aligned} \dot{V}_3 &= \mathcal{R}e\sum_{x\,\mu} \eta_\mu(x)\, ip_\mu(x)\left\{\hat{\phi}^\dagger(x)U_\mu(x)\,\hat{\phi}(x+\hat{\mu})\right\} \\ &+ \mathcal{R}e\sum_{x\,\mu} \eta_\mu(x)\, ip_\mu(x)\left\{\hat{\phi}^\dagger(x+\hat{\mu})U^\dagger_\mu(x)\,\hat{\phi}(x)\right\} \\ &= -\mathcal{R}e\sum_{x\,\mu}(-1)^{x_0+x_1+x_2+x_3}\eta_\mu(x)\, ip_\mu(x)\left\{\hat{\phi}^\dagger(x)U_\mu(x)\,\hat{\phi}(x+\hat{\mu})\right\} \\ &= -\sum_{x\,\mu} p_\mu(x)\,\mathcal{I}m\left\{\varsigma_\mu(x)\,\hat{\phi}^\dagger(x)U_\mu(x)\,\hat{\phi}(x+\hat{\mu})\right\} \end{aligned} \tag{4.50}$$

where $\varsigma_\mu(x) = (-1)^{x_0+x_1+x_2+x_3}\eta_\mu(x)$ and so

$$\dot{p}_\mu(x) = \text{gauge term} + \mathcal{I}m\sum_{\mu}\left\{\varsigma_\mu(x)\,\hat{\phi}^\dagger(x)U_\mu(x)\,\hat{\phi}(x+\hat{\mu})\right\} \tag{4.51}$$

To calculate the fermion contribution to $\dot{p}_\mu$ we must solve the system of equations $(-\not{D}^2+M^2)\,\phi = \psi$, form $\hat{\phi}$ by applying $\not{D}$ to ϕ, and calculate the second term in eq. 4.51 -which is the forwards part of a full $\not{D}$ applied to $\hat{\phi}$ (up to a phase).

The fermion momenta term is only necessary in an MD simulation. We follow the ideas of [Gottlieb et al 1987] in the stochastic schemes, refreshing the fermions with an exact heat bath on each hit, and not evolving them between hits.

$$\psi = \frac{1}{\sqrt{2}}\left(\not{D}\eta^{\text{odd}} + M\eta^{\text{even}}\right) \tag{4.52}$$

where the η's are gaussian distributed random fields.

4.3 Summary of Results of Other Groups

Several groups have begun dynamical fermion simulations on moderate size lattices. [Kovacs, Sinclair and Kogut 1986] use the hybrid algorithm on lattices up to $10^3 \times 6$ with staggered fermions, [Fukugita et. al. 1986] use the Langevin algorithm with Wilson fermions on a $9^3 \times 18$ lattice.

The Illinois/Argonne group have been studying QCD at finite temperature, initially in the presence of 4 light dynamical quarks and subsequently with 2 light and 1 heavy quark. They find evidence for a finite-temperature phase transition at low quark mass. The transition is sharp, implying an abrupt change in energy densities. This is potentially of relevance to heavy-ion collider experiments in which it may be possible to observe the transition between hadronic matter and a quark-gluon plasma.

The Japanese group have been calculating hadron masses. They find significant changes to the masses relative to the quenched approximation - the greater part of which can be absorbed as changes in the coupling. They obtain values for the delta-to-nucleon mass ratio consistent with experiment, but large finite size effects and statistical errors prevent them from drawing firm conclusions. Discretisation errors in the Langevin algorithm make it necessary for them to run at a range of timesteps, and extrapolate to $d\tau = 0$.

4.4 The Hybrid Monte Carlo Algorithm

We now describe the hybrid monte carlo algorithm which builds the ideas of equation of motion schemes into a monte carlo simulation.

We start with the definition of a Markov process in section 1.2. It is convenient to construct this process in two parts. First we choose a new configuration ϕ' with probability $P_C\left[\phi \mapsto \phi'\right]$ by some as yet unspecified procedure, and then we accept ϕ' with some probability $P_A\left[\phi \mapsto \phi'\right]$ or reject it and keep the old configuration ϕ instead. One choice of P_A which enables detailed balance to be satisfied for any P_C is a simple generalization of the Metropolis algorithm

$$P_A\left[\phi \mapsto \phi'\right] = \min\left(1, \frac{P_S(\phi')P_C\left[\phi' \mapsto \phi\right]}{P_S(\phi)P_C\left[\phi \mapsto \phi'\right]}\right) \tag{4.53}$$

where $P_S(\phi)$ is the probability of finding configuration ϕ in the equilibrium distribution. We require a method for choosing candidate configurations which can be computed efficiently and whose reverse probability $P_C\left[\phi' \mapsto \phi\right]$ is easy to obtain. Since we are proposing to update all fields ϕ simultaneously we insist that the acceptance rate P_A is large and that it doesn't depend too strongly on the size of our system. Finally, we want to minimise the correlation between successive configurations.

An elegant method for doing this follows the idea of the hybrid molecular dynamics / Langevin algorithm [Duane et al 1987]. Our procedure for generating a new configuration ϕ' is to select the momenta π at random from a gaussian distribution of mean 0 and variance 1

$$P_G[\pi] \propto e^{-\pi^2/2} d\pi \tag{4.54}$$

and then to let the system evolve deterministically through (ϕ, π)-phase space according to Hamilton's equations for a fixed time τ_0. This evolution defines a mapping on phase space by

$$(\phi, \pi) \mapsto (\phi', \pi') = (\Phi(\phi, \pi), \Pi(\phi, \pi)). \tag{4.55}$$

The value of τ_0 does not enter the proof that our algorithm works, so we have not bothered to make explicit the fact that the two functions Φ and Π depend on this extra parameter. So the probability P_H for choosing this candidate phase space "configuration" is

$$P_H[(\phi, \pi) \mapsto (\phi', \pi')] = \delta(\phi' - \Phi(\phi, \pi)) d\phi' \delta(\pi' - \Pi(\phi, \pi)) d\pi'. \tag{4.56}$$

We accept this candidate with probability $P_A[(\phi, \pi) \mapsto (\phi', \pi')] = \min(1, e^{-\delta H})$, where $\delta H \equiv H(\phi', \pi') - H(\phi, \pi)$. So the transition probability for the coordinate is

$$P_M[\phi \mapsto \phi'] = P_G[\pi] P_H[(\phi, \pi) \mapsto (\phi', \pi')] P_A[(\phi, \pi) \mapsto (\phi', \pi')] \tag{4.57}$$

and is evidently a function also of the initial and final momenta π and π'. In the case where our integration of the equations of motion conserves energy, that is $\delta H = 0$, we have $P_A = 1$, so we accept every step: this limit is just the usual hybrid algorithm with τ_0 being the interval between momentum refreshments, and leads to the desired equilibrium distribution $P_S \propto e^{-H} d\phi d\pi$. In the algorithm proposed here we integrate the equations of motion approximately, so that $H(\phi, \pi)$ is not exactly conserved ($\delta H \neq 0$), but in such a way that the Markov process we have defined satisfies detailed balance with the same equilibrium probability distribution.

$$P_S(\phi) P_M[\phi \mapsto \phi'] = P_S(\phi') P_M[\phi' \mapsto \phi] \tag{4.58}$$

To see what is required, we write out explicitly what this means. In order for the reverse transition to be generated by the Molecular Dynamics, it is clear that the signs of all the momenta must be reversed, and further that the approximate integration have the appropriate symmetry. So the condition to be satisfied reads

$$\begin{aligned} & e^{-H(\phi,\pi)} d\phi d\pi \delta(\phi' - \Phi(\phi, \pi)) d\phi' \delta(\pi' - \Pi(\phi, \pi)) d\pi' \min(1, e^{-\delta H}) \\ = \; & e^{-H(\phi', -\pi')} d\phi' d\pi' \delta(\phi - \Phi(\phi', -\pi')) d\phi \delta(\pi + \Pi(\phi', -\pi')) d\pi \min(1, e^{\delta H}) \end{aligned} \tag{4.59}$$

The acceptance probability provides just the right factor to convert $e^{-H(\phi,\pi)}$ into $e^{-H(\phi',-\pi')}$, since $H(\phi, \pi)$ is invariant under $\pi \to -\pi$. The remaining terms amount to the statement

$$\delta(y - Y(x)) = \delta(x - X(y)), \tag{4.60}$$

where we used the temporary shorthand x for (ϕ, π) and Y for the function defined by the MD mapping. X is the inverse function to Y, and the identity only holds if the transformation from x to y has Jacobian 1. The leapfrog integration scheme has both the properties crucial to the success of the idea: it has τ-reversal symmetry and it generates an area preserving map. The only extra feature needed is the half-step at the beginning and end of the interval τ_0 which is already included in the standard hybrid algorithm. In section 4.5 we determine

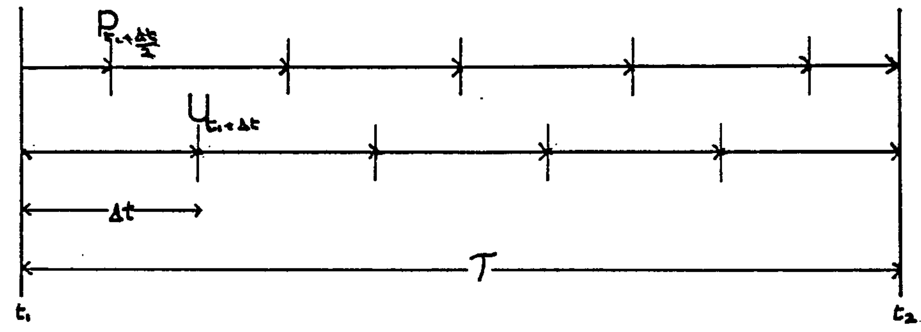


Figure 4.4: A leapfrog update scheme for the hybrid monte carlo algorithm.

the acceptance rates for a full simulation of QED with fermions. Before doing this, however, we indicate reasons why the method has a chance of working, and suggest a small but vital refinement.

The acceptance rate is determined by step-size errors in the total energy H, an extensive quantity. We would thus expect to have to have to adjust dt with L^2 (the coefficients must be determined numerically). The accumulated error over an MD run of length τ, $\Delta H = H(\tau) - H(0)$ is known to oscillate; we should tune τ to minimise ΔH.

The hamiltonian serves two purposes. First it defines the equilibrium distribution and thus determines the acceptance test and the heat-bath used to refresh the fermion fields. Second, it provides a means of evolving the fermion fields from one timestep to the next. The only reason why we should use it to evolve the fields is that in exact arithmetic for $dt = 0$ it will yield $\Delta H = 0$, 100% acceptance. In principle we could use any hamiltonian to evolve the fields. In finite precision arithmetic at a large value of dt, the integration scheme introduces errors. Perhaps we could compensate for these by using a different hamiltonian to evolve the fields. It has been suggested [Zinn-Justin 1986] that the dt errors can be accounted for by a change in β. If this is the case we should be able greatly reduce our discretisation errors by evolving the fields at a new value of the coupling β', returning to the original value β only to do the accept/reject step. We present results on the effectiveness of this technique in section 4.6.

4.5 A Simulation of QED

We want to be able to compare the efficiency of the equation of motion algorithms described above. We did not have computer resources to do this for the $SU(3)$ theory. Instead we have chosen to use the computationally simpler QED theory. We use compact $U(1)$, and staggered fermions. The equations of motion, a special case of those discussed earlier, are outlined in section 4.5.1. We coded the simulation in occam for an array of Transputers. Details of the implementation are given in section 4.5.2

4.5.1 Equations of Motion for QED

The Wilson action for pure compact $U(1)$ lattice gauge theory in 4 dimensions is

$$S_g = \beta \sum_{\substack{x\,\nu\,\mu \\ \nu>\mu}} \left(1 - \frac{1}{2}\left(U_\mu(x)\, U_\nu(x+\hat{\mu})\, U_\mu^\dagger(x+\hat{\nu})\, U_\nu^\dagger(x) + \text{h.c.}\right)\right) \tag{4.61}$$

where the links $U_\mu(x) = e^{i\theta_\mu(x)}$ and $0 \le \theta < 2\pi$. In terms of the thetas

$$\begin{aligned} S_g &= \beta \sum_{\substack{x\,\nu\,\mu \\ \nu>\mu}} \left(1 - \frac{1}{2}\left(\exp\left(i\boxed{\mapsto}\right) + \exp\left(-i\boxed{\mapsto}\right)\right)\right) \\ &= \beta \sum_{\substack{x\,\nu\,\mu \\ \nu>\mu}} \left(1 - \cos\left(\boxed{\mapsto}\right)\right) \end{aligned} \tag{4.62}$$

where $\boxed{\mapsto}$ is now $\theta_\mu(x) + \theta_\nu(x+\hat{\mu}) - \theta_\mu(x+\hat{\nu}) - \theta_\nu(x)$.

The simulation-time hamiltonian has three parts, the pure gauge action, a gauge kinetic term and a fermion term.

$$H = \beta \sum_{\substack{x\,\nu\,\mu \\ \nu>\mu}} \left(1 - \cos\left(\boxed{\mapsto}\right)\right) + \frac{1}{2}\sum_{x\,\mu} p_\mu^2(x) + \sum_{x\,y} \psi^\dagger(x)\left(-\not{D}^2 + M^2\right)^{-1} \psi(y) \tag{4.63}$$

On langevin steps (hits) the fermions fields are refreshed with an exact heat bath,

$$\psi = \frac{1}{\sqrt{2}}\left(\not{D}\eta^{\text{odd}} + M\eta^{\text{even}}\right) \tag{4.64}$$

where the η's are gaussian distributed random fields. They are not updated between hits [Gottlieb et al 1987], for the Langevin, hybrid and hybrid monte carlo algorithms. A fermion momenta term is added for MD simulations (see section 4.2. We update the links using a leapfrog algorithm. By analogy with the $SU(N)$ case above the equations of motion are

$$\begin{aligned}
\dot{\theta}_\mu(x) &= p_\mu(x) \\
\dot{p}_\mu(x) &= f_\mu(x) + \frac{i\beta}{2}\sum_{\nu\neq\mu}\left(\exp\left(i\boxed{\hookrightarrow}\right) - \exp\left(i_{\boxed{\hookrightarrow}}\right) - \text{h.c.}\right) \qquad (4.65)\\
&= f_\mu(x) - \beta\sum_{\nu\neq\mu}\left(\sin\boxed{\hookrightarrow} - \sin_{\boxed{\hookrightarrow}}\right)
\end{aligned}$$

$f_\mu(x)$, the fermion contribution to $\dot{p}_\mu(x)$, is given by

$$f_\mu(x) = Im\sum_\mu\left\{\varsigma_\mu(x)\,\hat{\phi}^\dagger(x)U_\mu(x)\,\hat{\phi}(x+\hat{\mu})\right\} \qquad (4.66)$$

where

$$\hat{\phi} = \left\{\begin{array}{c}\phi\\ \not{D}\phi\end{array}\right\}\text{on}\left\{\begin{array}{c}\text{even}\\ \text{odd}\end{array}\right\}\text{sites}$$

and $\phi = \left(-\not{D}^2+M^2\right)^{-1}\psi$ as described in section 4.2.

Each timestep requires the solution of a large sparse set of linear equations $\left(-\not{D}^2+M^2\right)\phi = \psi$, computation of the $f_\mu(x)$, and calculation of the plaquettes. The identity $\boxed{\hookrightarrow} = -\boxed{\hookleftarrow}$ for the $U(1)$ theory allows us to calculate the plaquettes in the planes for which $\nu > \mu$ (only).

4.5.2 A QED Simulation Program for 16 Transputers

The code fullqed implements the hybrid monte carlo algorithm in occam 2 on a binary hybercube of 16 Transputers. It was developed to run on a Meiko Computing Surface. With modifications to the file I/O handler it will run on a FPS-T20 if an occam 2 compiler is provided. The code for each processor is fully vectorised. **Fullqed** can also perform hybrid and langevin simulations. (A number of other programs exist: **uone**[1] is a (non-vectorised) pure MD code as is **uone_int** which uses integer arithmetic; qed_md is an MD code with dynamical fermions.) A binary hypercube (see figure 4.5) was the natural configuration to use first. It allows partitioning of a 4-d lattice onto a 4-d processor array giving each processor a 4-d sub-lattice to work on. The code for each processor is similar to that for a single processor working on the sub-lattice alone. The restriction to a binary hypercube is forced upon us; a general 4-d hypercube requires 8 inter-processor links but the Transputer only has 4. This configuration is by no means ideal, but its simplicity greatly eased the task of writing such a code in occam. See chapter 5 for analysis of the suitability of various architectures for lattice gauge theory simulations.

[1]This program was written by R.D. Kenway, K.C. Bowler and myself

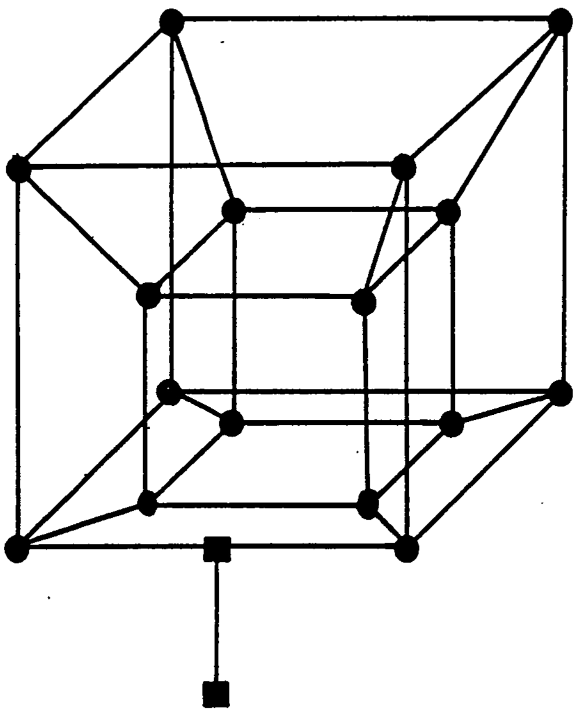


Figure 4.5: A binary hypercube configuration with a message passer inserted on one of the links.

Structure of the Program

The code **fullqed** is really three programs (i) the simulation code itself, which runs on each node of the hypercube, (ii) a message passer process, and (iii) the file I/O handler. We discuss (ii) and (iii) briefly first.

```
PROC node( .... )
  ... declarations
  ... procedure definitions
  SEQ
    ... initialisation of look-up tables
    ... load parameter file and configuration
    SEQ splash=0 FOR max.splashes
      SEQ
        ... randomise momenta
        SEQ md=0 FOR max.md+1
          SEQ
            ... decisions
            ... predictor
            ... fermion contribution to pdot
            ... gauge contribution to pdot
            ... leapfrog update
            ... energy measurements for md sweep
        ... energy measurements for splash
        ... accept/reject
        ... output data
    ... write configuration
:
```

Figure 4.6: The structure of the **fullqed** program for a node.

In building a hypercube configuration all 4 links from all 16 processors are used. In order to get messages in and out it is necessary to provide memory mapped communication between one of the processors and the world beyond or insert an additional processor into one of the sides of the hypercube. We don't have a processor suitable for the former, and so a 17'th processor is added. We run a message passing process on this device. (This slows down some of the communication as data must be forwarded by the message passer.)

The file I/O handler controls (a) the reading and writing of unformatted configuration files, (b) reading in the parameter file and (c) the writing of results files compatible with our standard statisical error analysis program **nnstats**[2]. This process runs on a Transputer connected to the message passer.

At the toplevel the simulation code appears to be a serial program, but it runs on all 16 processors. The inter-processor I/O is at a very low level. This I/O synchronises the program, so that broadly speaking each processor is doing the same thing at the same time.

Figure 4.6 illustrates the top level structure of the program as it appears in the program development enviroment OPS. Lines beginning with ... are "folds"; when "opened" they reveal their contents.

In the hybrid monte carlo algorithm the inner loop (over MD steps) must be performed 11 times to evolve the system through 10 timesteps. The momenta are "half-stepped" on the first and last MD steps. The links are not updated on the last step (see figure 4.4). This extra work is avoided when the program is set up to run the hybrid algorithm.

The Partitioning Scheme used is as follows

- L^4 lattice $x_\mu = 0, 1, \ldots, L-1 \;\; \mu = 0, 1, 2, 3$
- 2^4 processor array $p_\mu = 0, 1 \;\; \mu = 0, 1, 2, 3$ processor id $p = \sum_\mu p_\mu 2^\mu$
- Processor p is responsible for a n^4 sub-lattice

$$x_\mu = \begin{cases} 0, \ldots, n-1 & p_\mu = 0 \\ n, \ldots, L-1 & p_\mu = 1 \end{cases} \quad \mu = 0, 1, 2, 3$$

where $n = \frac{L}{2}$. The θ_μ, p_μ variables associated with site x, $\dot{p}_\mu$, χ, and ψ are all held on the processor responsible for site x. A key feature of this partitioning scheme is that it maintains the locality properties of the action. All data for sites in the neighbourhood of x is either on the processor responsible for x or on one of its immediate neighbours.

The Fermions

The fermionic contribution to the update comprises two parts; solution of the system of equations, and calculation of f_μ.

Solving the Equations We calculate $\phi = \left(-\not{D}^2 + M^2\right)^{-1} \psi$ using a Conjugate Gradient algorithm (see section 2.3). We first predict ϕ_{n+1} in terms of $\phi_n, \phi_{n-1}, \cdots$ by fitting a polynomial to the existing data and extrapolating to ϕ_{n+1}. The predictor uses a guess of 0 on the first sweep (after the hit), the old ϕ on the

[2]written by S. Duane and B.J. Pendleton

next, and so on up to some maximum (currently 3rd) order. The calculation is easy ($2n^4$ operations per processor for the first order, $3n^4$ for the second and $5n^4$ for the third). No I/O is required, but we must keep ϕ for some number of earlier timesteps. The effectiveness of the predictor is discussed in section 4.6. The prediction is used as the starting point for the CG solver which then iterates until the desired precision in $r^\dagger r$ is obtained.

```
PROC dslash( .... )
  ... declarations
  SEQ
    ... zero Da
    SEQ mu=0 FOR 4
      ... abbreviations for this direction
      SEQ
        ... gather boundary data for output
        PAR
          ... $e^{i\theta_\mu(x)}a(x+\hat{\mu})$ for interior sites
          ... send $a(x_\mu = 0)$ in $\mu$ direction
          ... get $a(x_\mu = n)$ from $\mu$ direction
        ... $e^{i\theta_\mu(x)}a(x+\hat{\mu})$ for boundary sites
        ... gather boundary data for output
        PAR
          ... $e^{i\theta_\mu(x-\hat{\mu})}a(x-\hat{\mu})$ for interior sites
          ... send $a(x_\mu = n-1)$ in $\mu$ direction
          ... get $a(x_\mu = 0)$ from $\mu$ direction
        ... $e^{i\theta_\mu(x-\hat{\mu})}a(x-\hat{\mu})$ for boundary sites

:
```

Figure 4.7: Calculating $\not{D}a$

Two phases of the CG solver require inter-processor communication; the matrix vector products (applications of $\not{D}$), and the scalar products. The rest of the code simply calculates the new CG vectors p, r and x (see section 2.3), on each processor. The structure of the $\not{D}$ procedure is illustrated in figure 4.7. All sines and cosines of links are precomputed with added phases and boundary condition signs . The 'gather' processes involve collecting all the data (of a given parity) that lies in a boundary. We do this using lookup tables that list these sites. The data to be transferred is collected into an array in on-chip memory, from where it is transferred en'mass to the neighbouring processor. This I/O is prioritised above the concurrent communication process (see appendix B) - the code to do this has been omitted from figure 4.7 for clarity.

Scalar products are calculated locally, and then this data is accumulated. Each processor first adds its partial sum to that of its neighbour in the 0 direction. This is then repeated for the other 3 directions, the most recent partial sum being transferred at each stage.

The equation solver procedure is also responsible for the fermion heat bath as this requires $\not{D}$ acting on gaussian noise, which in turn requires the sine and cosine of all the links.

Fermionic Term in $\dot{p}_\mu$ This cross-term is calculated when the inversion is complete. It requires the application of $\not{D}$ to ϕ to form $\hat{\phi}$ on odd sites and the computation of

$$f_\mu(x) = Im \sum_\mu \left\{ \varsigma_\mu(x)\, \hat{\phi}^\dagger(x) U_\mu(x)\, \hat{\phi}(x+\hat{\mu}) \right\}$$

We do this using the forward part of a $\not{D}$ applied to ϕ, with an extra loop over even and odd sites.

The Gauge Fields

Calculation of the gauge field contribution to $\dot{p}_\mu$ follows the scheme outlined in section 2.1. We select a plane to work with, and start on the plaquettes. While doing this we transfer the boundary links (see figures 4.8 and 2.2). We then complete the plaquettes and take the sine; this step dominates the calculation. If we are measuring the gauge potential energy we also evaluate $\sum \cos(\Box)$. Each plaquette is used to update 4 momenta, $\dot{p}_\mu(x)$, $\dot{p}_\nu(x+\hat{\mu})$, $\dot{p}_\mu(x+\hat{\nu})$ and $\dot{p}_\nu(x)$. The first two $\dot{p}_\mu(x)$ and $\dot{p}_\nu(x)$ are calculated while the boundary data for $\dot{p}_\nu(x+\hat{\mu})$ and $\dot{p}_\mu(x+\hat{\nu})$ is transferred, the momenta are then completed.

It is possible to overlap more work with the inter-processor communication by separating off all the calculations that require only local data and performing them in parallel with the above for the sites in the boundaries (only). We made these changes to the first version of the program, but the improvement in total cpu time required was negligible (This is because the calculation is dominated by taking the sine of the plaquettes). We did not make these changes to **fullqed** as they complicate the coding.

Odds and Ends and Random Numbers In addition to the calculation of the p_μ we must (i) perform the leapfrog update, (ii) calculate the energies and (iii) generate random numbers. The gauge field update is straightforward, it requires no I/O. We calculate the plaquette energy (if it is required) while updating the $\dot{p}$. The gauge kinetic energy, fermion energy and $\bar{\psi}\psi$ each require a scalar product (performed as described above).

Two kinds of random numbers are required, uniform and gaussian distributed. We used a linear congruential pseudo-random number generator to obtain the

```
PROC gauge( .... )
  ... declarations
  SEQ mu=0 FOR 4
    SEQ nu=mu+1 FOR 3-mu
      SEQ
        ... gather links to be transferred
        PAR
          ... start the plaquettes $\Box = \theta_\mu - \theta_\nu$
          ... send $\theta_\mu(x_\nu = 0)$ in the $\nu$ direction
          ... send $\theta_\nu(x_\mu = 0)$ in the $\mu$ direction
          ... get $\theta_\mu(x_\nu = n)$ from the $\nu$ direction
          ... get $\theta_\nu(x_\mu = n)$ from the $\mu$ directions
        ... complete the plaquettes and take sine
        ... gather data to be transferred
        PAR
          ... calculate $\dot{p}_\mu$ and $\dot{p}_\nu$
          ... send $\Box_\mu(x_\nu = n)$ in the $\nu$ direction
          ... send $\Box_\nu(x_\mu = n)$ in the $\mu$ direction
          ... get $\Box_\mu(x_\nu = 0)$ from the $\nu$ direction
          ... get $\Box_\nu(x_\mu = 0)$ from the $\mu$ direction
        ... calculate $\dot{p}_\nu\,(x + \hat{\mu})$ and $\dot{p}_\mu\,(x + \hat{\nu})$

:
```

Figure 4.8: Calculating the Gauge Field Contribution to $\dot{p}_\mu$

uniform distribution. In order to avoid generating the same sequence of numbers on each processor (although this was useful for debugging) we modified the multiplier and the initial seed so that each processor generates its own sequence. The 16 numbers produced, one per processor, are the same 16 as would be generated by a single processor calling the unmodified random number generator 16 times. The accept/reject decisions are made by one processor and then distributed.

A gaussian distribution is required for refreshing the fermions and the gauge momenta on langevin steps. It is calculated using a polar Box-Muller algorithm running on every processor. It calls the modified linear generator.

Testing

In an MD simulation total energy is conserved for sufficiently small *dt*. This proved to be useful in testing fullqed. At all stages of its development we were able to set up a long MD run at low *dt* and check total energy.

We tested the gauge sector of **fullqed** by comparing values of the plaquette with existing data; our $U(1)$ results agree with those of [Lang 1986]. The gauge momenta and fermion fields are generated from gaussian distributions. We checked that the gauge kinetic and fermion energies are $\frac{1}{2}$ per degree of freedom. A correct value of the fermion energy points towards correctness of the $\not{D}$ procedure. We check this (and the CG solver) by comparing $\langle\bar{\psi}\psi\rangle$ and the propagator for free fermions with data obtained using other programs. The hardest part of the calculation to test is the fermion contribution to $\dot{p}$, where we rely on conservation of energy.

Performance

We measured the performance of **fullqed** running on a hypercubic array, with no I/O to the outside world. The relative importance of the two sectors and the measured rate of compute to I/O are shown in table 4.5.2. For a fermion mass of 0.25 a hybrid monte carlo sweep comprising 10 md steps takes about180 seconds. Performance deteriorates very slightly when a single message passer is inserted

	Relative importance	Compute to I/O ratio
Fermions	95 %	28
Gauge Fields	5 %	7

Table 4.1: Performance data for **fullqed** at $m = 0.25$. The fermion data is for the $\not{D}$ routine.

on one of the edges as the transfers in this direction have to be forwarded by the message passer. This delay only effects data accumulation procedures, as the overlapped computation-to-communication ratio $\Gamma \gg 1$ for the updates. Ideally **fullqed** should run on a mass store board and 15 other processors. Replacement of the T414B-15s by T800-20 floating point Transputers should cut total run time by about 6 or 7.

Uone_int runs 8 times faster than the 32-bit real code **uone**. It uses lookup tables for the sine and cosine functions. In practise this code simulates the gauge group $Z(32768)$, the link variables being represented by integers. Further (large) increases in speed could be made if a small subgroup of $U(1)$ was used from the outset. We did not try to code the fermion update in fixed-point arithmetic.

4.6 Results

In this section we present results on the behaviour of the hybrid monte carlo algorithm; and a comparison of the effectiveness of the equation of motion schemes. All results have been obtained using the code **fullqed** running on 16 Transputers.

4.6.1 Performance of the Hybrid Monte Carlo Algorithm

We have studied three aspects of the performance of the hybrid monte carlo algorithm, the acceptance rates, $\beta\beta'$ tuning and the effectiveness of the predictor. We ran the simulation on the disordered side of the phase transition where the density of low eigenvalues (of $\not{D}^2$) is higher - and hence the inversion problem harder. We used $\beta = 0.8$ for our fermionic simulations (at both high and low mass), away from the transition. Results obtained near the transition are similar but with large errors.

Acceptance Rates Figure 4.9 illustrates the dependence of the acceptance rate upon stepsize and lattice size. Data is for the pure gauge theory on 4^4, 8^4, and 12^4 lattices, and for the dynamical theory on 4^4 and 8^4. For an equilibrated configuration fluctuations in the acceptance rates are very small. Figure 4.10 is constructed by reading values of dt corresponding to 50%, 60%, 70% and 80% acceptance from figure 4.9. The data on acceptance rates (a/r) is consistent with

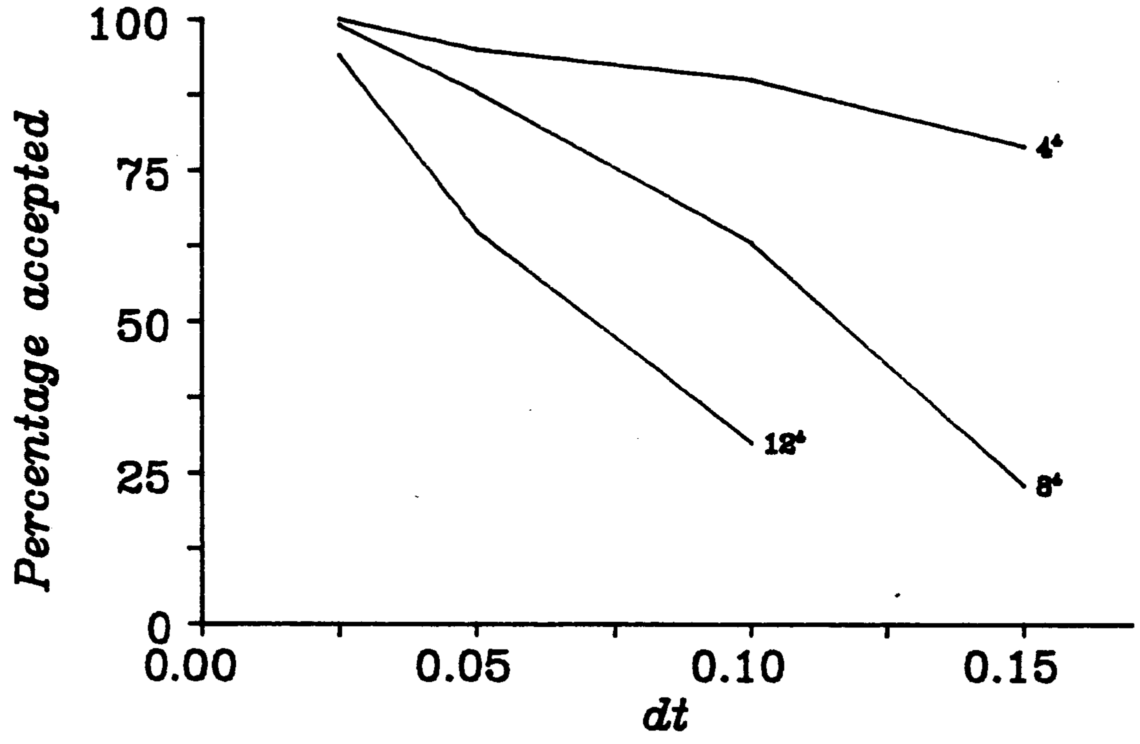


Figure 4.9: Acceptance rates as a function of dt and lattice size.

$a/r = e^{-L^2 dt^2}$. We were not able to obtain accurate data for the dynamical theory on 12^4 lattices due to lack of computer time.

It is tempting to extrapolate the data in figure 4.10. Doing this gives 50%

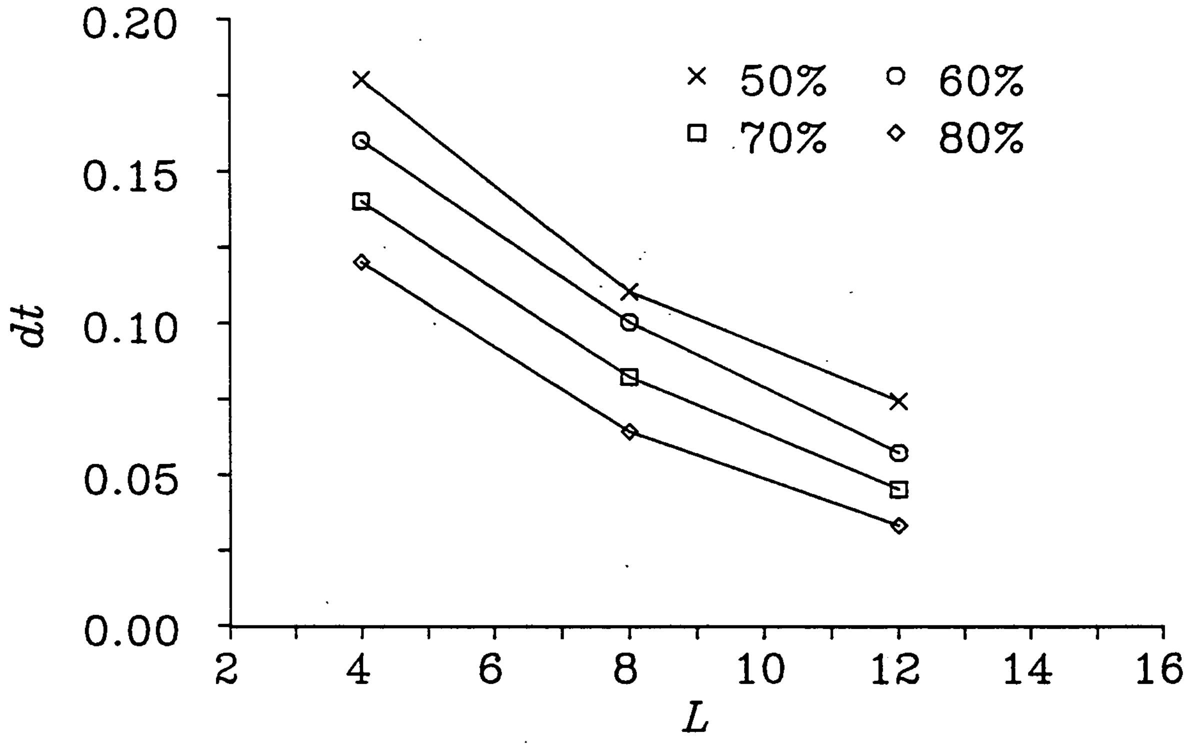


Figure 4.10: Values of dt corresponding to acceptance rates of 50, 60, 70 and 80 per cent. For the pure theory.

acceptance rates at a timestep of 0.05 for 10^4 lattices and 0.03 for 16^4 (both with dynamical fermions). These timesteps are significantly larger than those reported as suitable for hybrid simulations [Kogut 1983].

$\beta\beta'$ Tuning We began by working with an 8^4 lattice at $dt = 0.15$, for which the acceptance rate $\sim$ 23% without tuning. Figure 4.11 shows the acceptance rate as a function of $\beta' - \beta$. The distribution is approximately gaussian with peak

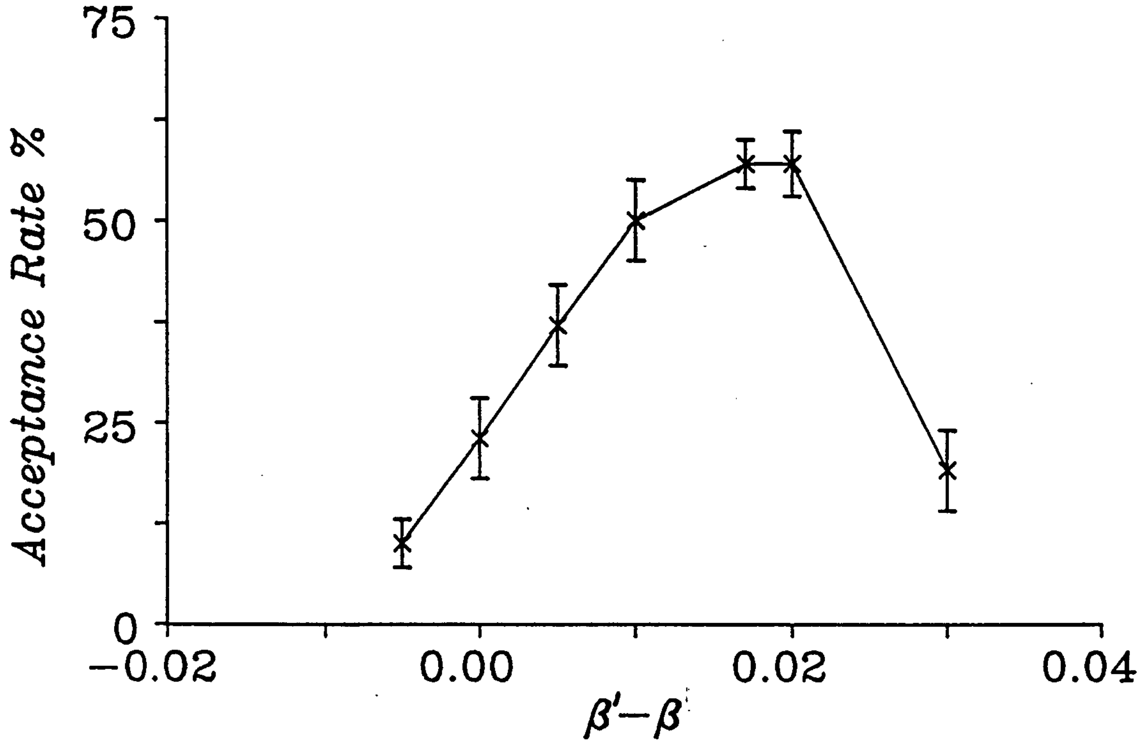


Figure 4.11: Acceptance rate as a function of $\beta' - \beta$

shifted towards $\beta' > \beta$. For $\beta' = \beta + 0.02$ we obtain an accepance rate of 57 %. Figure 4.12 shows the acceptance rate as a function of dt with optimal tuning and that without for an 8^4 lattice.

Predictors **Fullqed** spends the great bulk of its time solving the matrix problem, 93% for 30 CG iterations (this is the minimum number used, for high masses, many more are needed for low masses). The effectiveness of the predictor is thus of vital importance. We have tested predictors of order 1 to 4. Figure 4.13 shows the number of CG iterations required to evolve the system through 1 time unit. The solver is run until $r^\dagger r < 0.0005$ at masses of 0.25 (left) and 0.05 (right). It is clear that for high m and small dt the predictor is very effective, but at large dt and small m there is no improvement. The m behaviour is not suprising; all the predictor does (when it works) is provide a better initial guess for χ, it

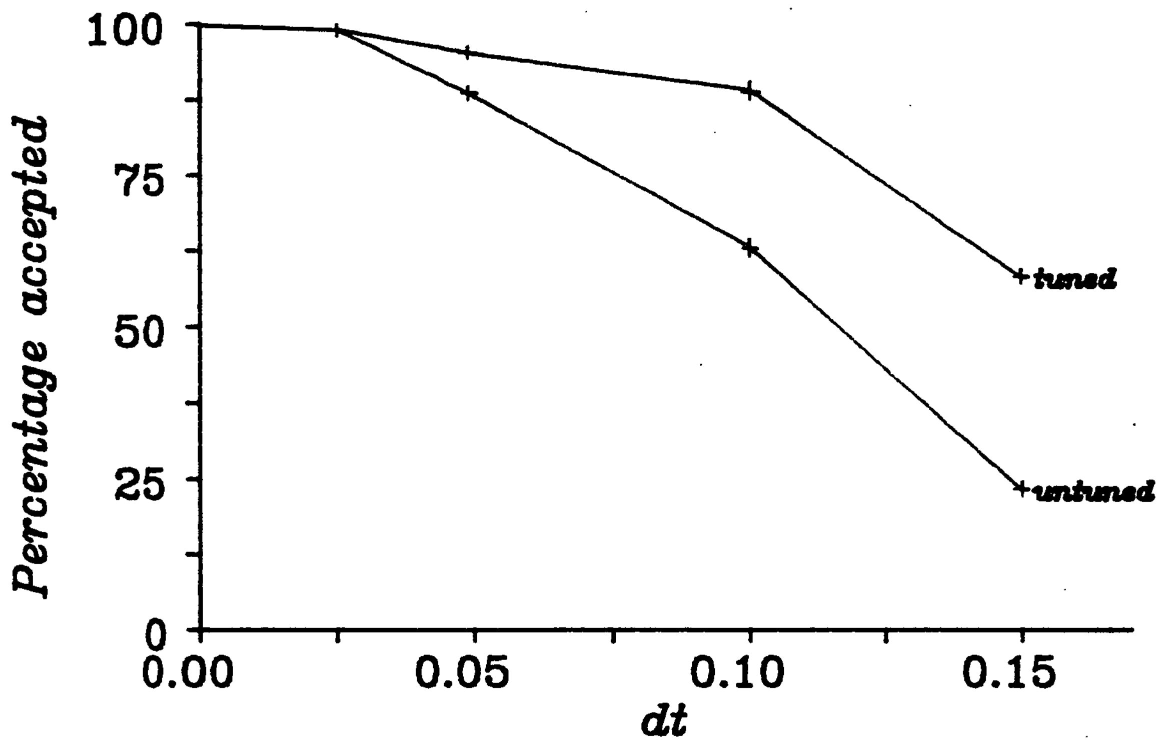


Figure 4.12: Tuning the hybrid monte carlo algorithm. Acceptance rates as a function of dt for $\beta = \beta'$ and optimal $\beta' - \beta$.

does not alter the rate of convergence - which is proportional to m. Further, as m increases the matrix becomes increasingly diagonal dominant and order dt changes to the links are of less significance. We see little or no improvement at large values of dt whatever the mass. If we combine figures 4.12 and 4.13 we obtain the computer time required to evolve the system through one time unit as a function of dt. The data is for an 8^4 system. We find the optimum timestep to be 0.075 for an 8^4 system at low mass.

4.6.2 Comparison of Equation of Motion Algorithms

We compare equilibration times and stepsize errors for the four equation of motion schemes discussed; MD, langevin, hybrid and hybrid monte carlo.

Equilibration

Figure 4.15 shows the equilibration of the average plaquette over 50 time units at $\beta = 0.97$ for the pure theory. Our results for the dynamical theory are similar,

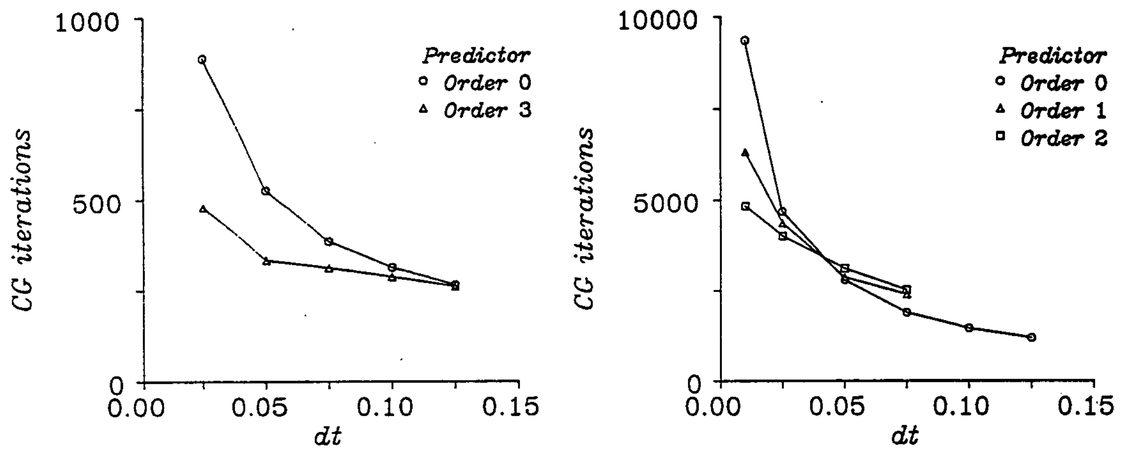
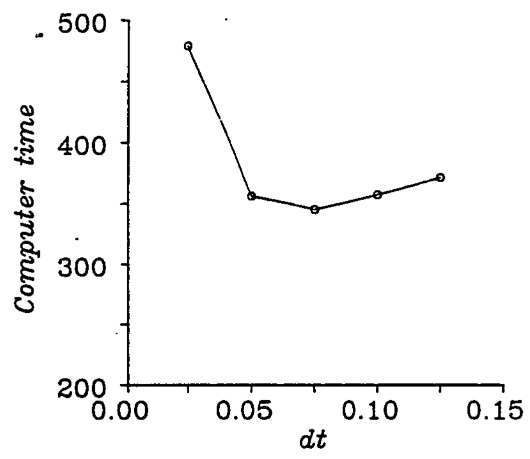


Figure 4.13: Number of CG iterations required to evolve the system through 1 time unit, as a function of *dt* and predictor order. Fermion masses are 0.25 (left) and 0.05 (right). 8^4 lattice.



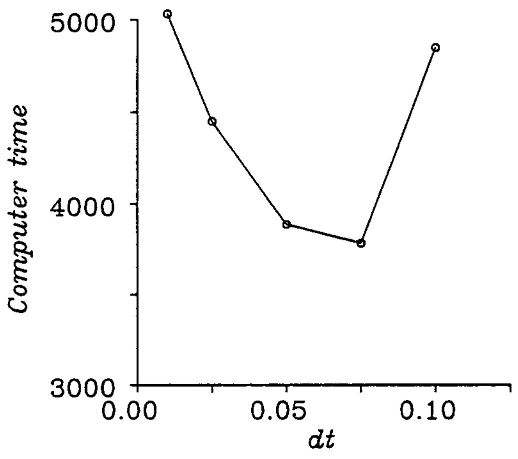


Figure 4.14: Computer time required to evolve the system as a function of *dt*, masses are 0.025 (left) and 0.05 (right).

but of poorer quality because of the computer time required to perform such calculations.

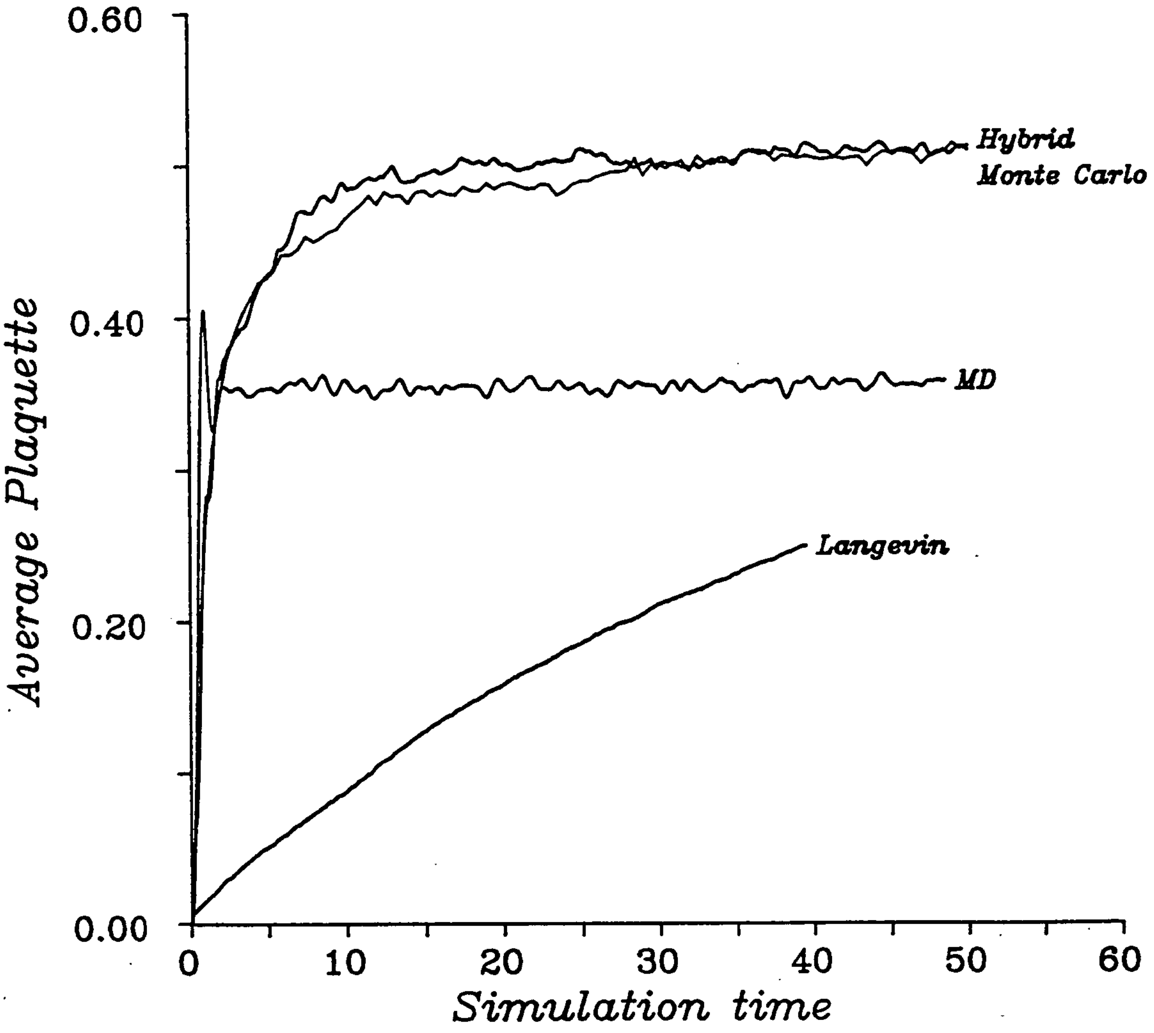


Figure 4.15: Equilibration times for equation of motion schemes.

The average plaquette equilibrates rapidly in the MD simulation, but to the wrong value. The gauge momenta equilibrate to an appropriate energy ($\frac{1}{2}$ per degree of freedom). The system must conserve energy and so the potential energy is constrained to be the initial energy minus the kinetic energy. This fixes the average plaquette at a value determined by the initial conditions. We must determine β once the simulation is complete.

Adding noise to the system ensures that we equilibrate at the correct value of the plaquette. The optimum hit frequency for hybrid is between once and twice per unit time. Equilibration rates drop rapidly as more noise is added. The Langevin limit, noise every timestep is the worst case. The hybrid monte carlo simulation converges at the same rate as the hybrid scheme. The timesteps are 10 times the size, acceptance rates are about 60% for this system, and so the evolution is 6 times as rapid.

4.6.3 Timestep Errors

We measured values of the average plaquette (for a pure 8^4 system) and $\langle\bar{\psi}\psi\rangle$ for a fermionic system (at $m = 0.25$ and $\beta = 0.8$) as a function of dt for hybrid and monte carlo algorithms. The results are shown in figures 4.16 and 4.17. We

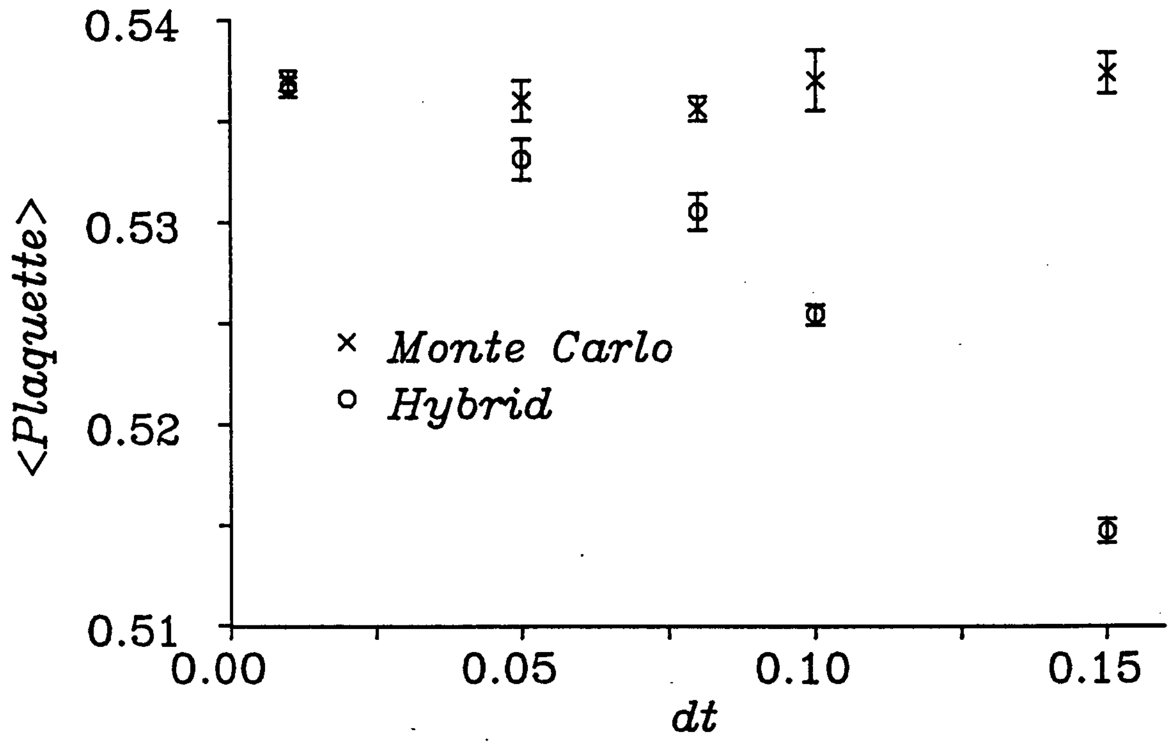


Figure 4.16: Timestep errors in the average plaquette for a pure gauge system at $\beta = 0.97$ on an 8^4 lattice.

see evidence for order dt^2 errors using hybrid. There are no dt errors using the monte carlo algorithm, as we would expect. Results for the plaquette in the fermionic system are similar to those shown in figure 4.16, but are of poorer quality. Timestep errors force us to use small values of dt in hybrid simulations. The hybrid monte carlo algorthim allows us to select the value of dt at which the system evolves most rapidly.

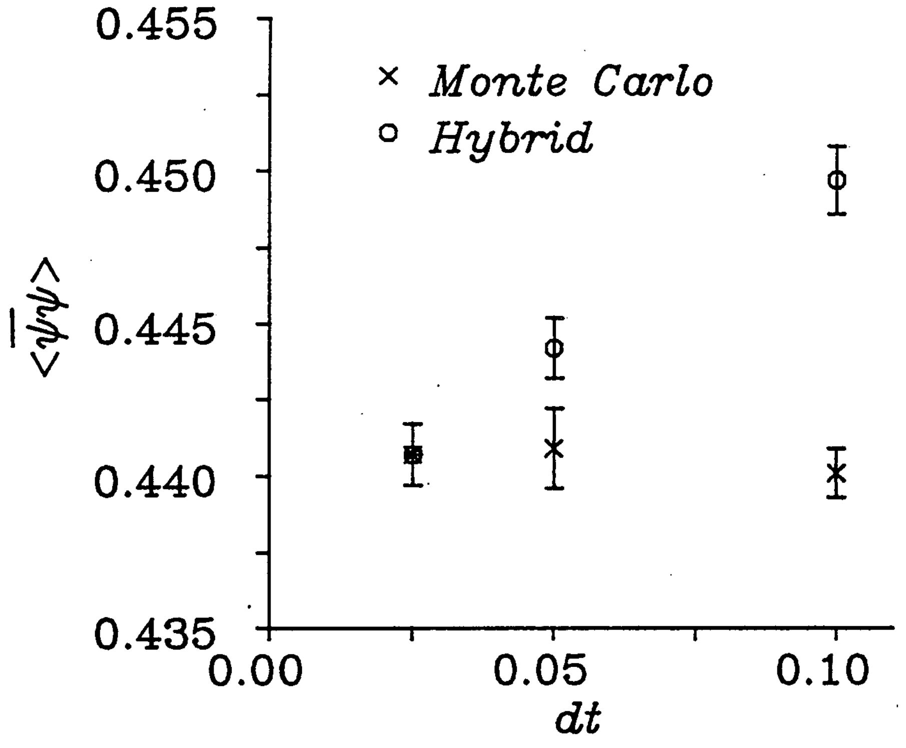


Figure 4.17: Timestep errors in the $\langle\bar{\psi}\psi\rangle$ for a high mass fermionic system at $\beta = 0.8$ on an 8^4 lattice.

4.6.4 Results for QED

We would like to be able to investigate phase structure of the theory, but a 10 hour run of the dynamical system for an 8^4 lattice at a high mass completes only 200 time units, falling to about 30 time units at low masses (using the hybrid monte carlo algorithm). We are forced to conclude that performing realistic simulations on the current configuration of Computing Surface is not practical. We would require at least a factor 50 improvement in speed. Achieving this efficiently on a large array of floating point Transputers is discussed in chapter 5.

4.7 Conclusions

The Molecular Dynamics Algorithm explores phase space rapidly but in an unsatisfactory manner. The conservation of total energy makes it difficult to select a value of β. To run a simulation at that β, an extra tuning stage is necessary.

In addition there is the question of ergodicity. We have seen very long time tails at low masses in **fullqed** and in a QCD code running on a $4^3 \times 8$ lattice. Other authors [Polonyi and Wyld 1983] report similar problems. We know ergodicity is going to be a problem; it seems best to avoid the problem by adding noise.

The Hybrid Algorithm retains the best feature of MD, rapid exploration of phase space, but the addition of noise removes any problems with ergodicity. It is interesting to note that the optimum value for the frequency of adding noise is 1 hit per unit time - close to the MD limit of the hybrid algorithm. As we increase the frequency of application the equilibration rate drops.

The Langevin Algorithm is very slow (this is not suprising as it is a random walk process). In common with the hybrid algorithm it is necessary to run at a range of timesteps $d\tau$ and then extrapolate to $d\tau = 0$. This further increases the already high computational requirements.

The Hybrid Monte Carlo Algorithm explores phase space rapidly, and on moderate size lattices the large timestep used makes the simulation faster than hybrid by a factor of about 6. Increasing the lattice size necessitates reductions in the timestep, but not to an unacceptable degree. The total energy of the system is extensive; so fluctuations in $H \sim L^2$ but we can compensate for this by tuning the value of β' used to evolve the fields.

Timestep errors are of order dt^2 for hybrid algorithms (see figure 4.16 and [Duane 1985]). There are no timestep errors in the hybrid monte carlo algorithm. The Cornell group, using the Langevin algorithm at large dt, say that the errors introduced can be explained in terms of shifts in the coupling and cite [Zinn-Justin 1986]. We can make use of this relation to tune β'.

The Computational Requirements of equation of motion algorithms are high; we must still solve a large sparse system of equations every timestep. We can make some use of predictors, but they are at their least effective for the range of parameters we are most interested in. What is really required is (i) an effective preconditioner and (ii) a computer capable of solving the system of equations for low mass fermions on a 16^4 lattice in less than a minute.

The basic algorithms involved in dynamical fermion simulation are local, and inherently parallel (see chapter 2). In chapter 5 we discuss the use of large arrays of processors for such simulations.

Chapter 5

Designing a Parallel Processor for Lattice QCD

The aim of this chapter is to determine whether we can build a parallel processor of sufficient power to tackle simulation problems in lattice QCD. The designs are intended for the equation of motion algorithms of chapter 4. We concentrate on the question of whether it is possible to partition the problems of chapters 2 and 4 in such a way that we achieve linear (or near linear) speedup as processors are added. If this is possible then we will be able to use large numbers of medium-high power processors to perform our simulations. We consider a variety of geometric array architectures (rings, grids and hypercubes) as well as algorithmic cells (clusters of processors arranged to reflect the structure of a problem) and simple arrays of such cells.

We partition our problems for each machine, and analyse the following aspects of the implementation :

1. Synchronising steps, communication that cannot be overlapped with work and any extra work that results from using more than one processor.

2. Computation that cannot be overlapped with communication.

3. Work that can be overlapped with communication if necessary.

4. Communication that can be overlapped with work.

In splitting a task amongst several processors we will incur costs due to communication between them. Processors such as the Transputer allow us to reduce these costs by overlapping computation and communication. The parameter Γ of chapter 1 reflects this overlap; it is the ratio of term 3 to term 4 (above). Provided synchronous communication and extra work (term 1) are negligible then Γ large signals a successful implementation if the processors have sufficient power to solve the problem.

We describe a calculation as being *local* when it requires data from a processor and/or its immediate neighbours, *internal* when the calculation requires no communication and *global* when data is (potentially) required from many (or all) processors.

We discuss partitioning of the gauge sector in section 5.1, and of the lattice Dirac equation in section 5.2. Other aspects of the calculation are either internal or can be performed sufficiently infrequently as to be negligible.

In chapter 2 we showed that the gauge update and the dominant parts of an iterative solution of the lattice Dirac equation (the matrix-vector multiply steps) were local; only the scalar products require global information. This suggests that there will be natural mappings of the lattice onto a geometric array that preserve *processor-data locality*.

For both problems we begin by discussing a simple partitioning of the lattice onto a torus. We use this scheme as a benchmark with which to compare mappings onto other architectures.

We have argued that communication-to-computation ratios are surface-to-volume ratios. For a given number of processors arranged in a d-dimensional array the surface becomes more important as d increases. However, we are restricted to an array of L^d processors for an L^d lattice[1] so to increase the number of processors used we should increase d. Success is dependent on us being able to find a configuration with sufficient processing power for which the computation-to-overlapped-communications ratios are acceptable and synchronising work is negligible.

In discussing the compute-to-I/O ratios we initially use algorithm rather than processor-dependent performance estimates. We then convert these ratios to numbers using data on the performance of floating-point Transputers.

5.1 The Gauge Sector

The gauge sector calculations play a small part in a dynamical fermion simulation, less than 7% in fullqed. However, to achieve a large speed-up on the full program we must work on both sectors. If we concentrate solely on the fermions then the relative importance of the gauge sector will increase and the total speed-up will fall. (We would also like to be able to switch the fermions off, and run pure gauge calculations on a large lattice. For this to be possible we require a large speed-up in the gauge sector.) Our analysis is for equation of motion schemes in which all gauge links are updated simultaneously. Different problems are faced in a pure-gauge heat-bath program where only half the links in any one direction can be updated at the same time. We will point out where analysis of the latter problem departs from that of ours.

[1]If we used an array of linear size greater than L we would have to through-route some of the communications - this would not be efficient on current Transputers.

In chapter 2 we showed that the gauge sector of the simulation requires (i) that we calculate and sum the staples, (ii) that we multiply in the link to be updated and take the trace and (iii) that we perform the exponential update. (i) is local, (ii) and (iii) are internal, so we concentrate on the staples. We express compute time in terms of numbers of staples calculated and I/O in terms of complex 3 by 3 matrices transferred. Detailed compute to I/O ratios, Γ, are given in section 5.1.4.

5.1.1 Staples on a Torus

Divide an L^4 lattice into hyper-rectangular regions of size $n_0 n_1 L^2$ and allocate them to an array of T_0 by T_1 processors (where $T_i = \frac{L}{n_i}$) in the form of a torus (see figure 5.1). The directions 0 and 1 are referred to as external, 2 and 3 internal.

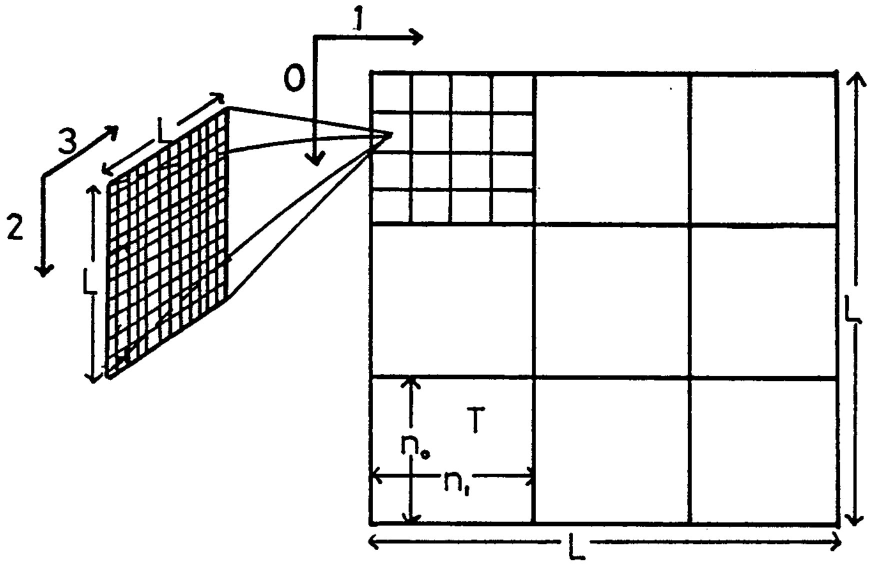


Figure 5.1: Partitioning an L^4 lattice onto a grid of $T \times T$ processors.

A Simple Scheme

For simplicity we first consider the case of a $1 \times 1 \times L^2$ sub-lattice allocated to each processor. This is the worst case as far as communication-to-work ratios are concerned. Single out a direction $\hat{\mu}$, and consider updating links in this direction. We must calculate 6 staples, two for each free $\hat{\nu}$ direction. Two cases must be considered, (i) the selected link lies in the plane held internally ($\hat{\mu} = 2$, $\hat{\mu} = 3$ see figure 5.2) and (ii) it does not ($\hat{\mu} = 0$, $\hat{\mu} = 1$ see figure 5.3).

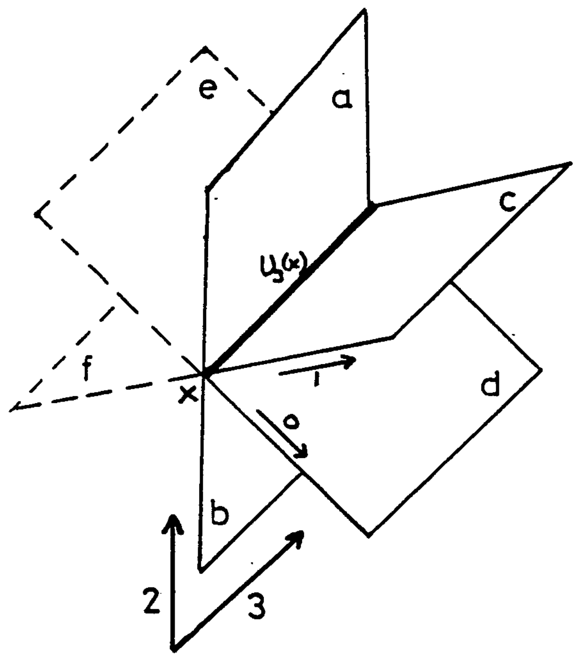


Figure 5.2: Staples required to update an internal direction.

Internal Directions Of the 6 staples, data for 2 is held internally (a and b in figure 5.2), and 2 require a $\hat{\mu}$ directed link to be passed from a neighbouring processor (in the positive $\hat{\nu}$ directions c and d). Data for the last 2 staples (e and f) are held on a processor in the negative $\hat{\nu}$ direction and can be transferred as complete staples. We should first calculate staples a,e,f (e,f to be passed forward), while transferring the data required for c and d (links $U_\mu(x+\hat{0})$ and $U_\mu(x+\hat{1})$), then calculate b,c,d while transferring the completed staples e and f. We then add in e and f. This is repeated for the second internal direction. There are L^2 links in each internal direction. The compute-to-communications ratio is 3 staples calculated to 4 matrices transferred (2 in, 2 out) during each phase. Up to 4 communications links can be used in parallel throughout.

The External Directions seem to be more difficult as none of the staples use solely internal data. However, if staples associated with the internal directions are completed first the links required to complete staples g,h,i,j in figure 5.3 ($U_2(x+\hat{\mu})$, $U_2(x+\hat{\mu}-\hat{\nu})$, $U_3(x+\hat{\mu})$ and $U_3(x+\hat{\mu}-\hat{\nu})$) have already been transferred.

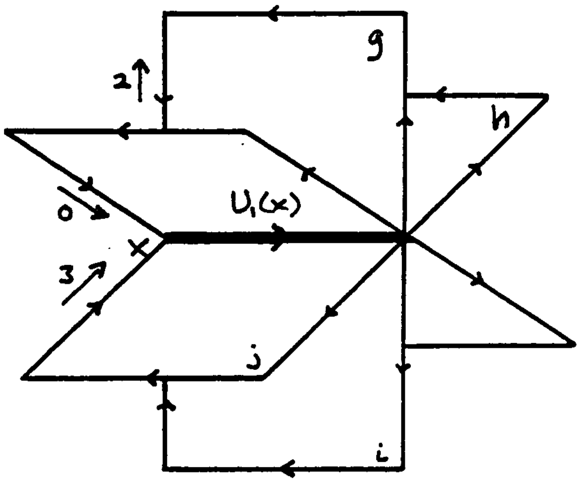


Figure 5.3: Staples required to update an external direction.

All that remains is to calculate the staples with two external directions. We do this as shown in figure 5.4. 2 links must be transferred, $U_0(x+\hat{1})$ and $U_1(x+\hat{0})$, 4 staples calculated, and then 2 staples moved. The I/O for this can be performed in parallel with calculating staples g,h,i,j. In total we must calculate 12 staples

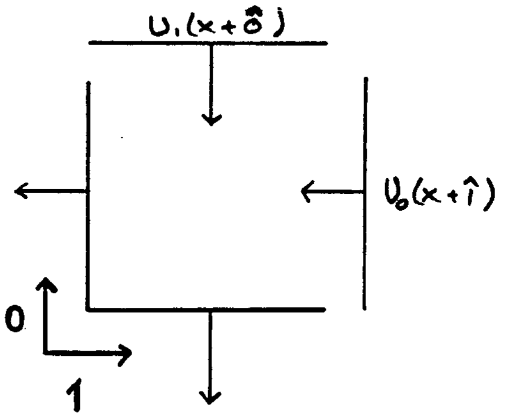


Figure 5.4: Calculating staples in the processor plane.

and perform 8 transfers. We overlap the transfer of links with staples of types g and h, then do the staples in the processor plane, then overlap moving these with calculating staples c and d. The compute-to-I/O ratio is 1 staple to 1 matrix transfer. We can make use of up to 4 communications links per processor.

In a pure gauge heat-bath algorithm we will only have the data for staples a and b. Further, we will have imposed an "internal directions first" ordering on the update.

General Case

Our simple scheme for a $1 \times 1 \times L^2$ region on each processor had minimal computation to overlap with communication. The generalisation to a $n_0 n_1 L^2$ sub-lattice is straightforward.

Consider an internal direction $\hat{\mu}$; the surfaces to be exchanged have either $n_0 L^2$ or $n_1 L^2$ $\hat{\mu}$-directed links. If we follow the algorithm above then we can (for example) overlap $2(n_0+n_1)$ transfers with $3n_0n_1$ staple calculations (there are now n_0n_1 staples of each type) during both phases for each direction. The compute-to-I/O ratio is $\frac{3n_0n_1}{2(n_0+n_1)}$

We again consider the external directions together. In total we must calculate $12n_0n_1$ staples and transfer $2(n_0+n_1)$ links then $2(n_0+n_1)$ completed staples. The compute-to-I/O ratio is $2n_0n_1$ staples to $(n_0 + n_1)$ matrix transfers. Projected values of Γ for a variety of sub-lattices sizes are given in section 5.1.5.

5.1.2 Other Geometric Structures

The ratio, Γ, of computation to overlapped communication is a volume-to-surface effect; as we raise the dimension of the array the surfaces become more important. If we restrict ourselves to hyper-rectangular partitioning schemes then for fixed lattice size the total number of processors that can be used increases, and the number of sites allocated to each decreases, as we increase the array dimension. On a ring of processors there will be more internal work to do while transferring the surfaces but the total number of processors is small. On a cubic or hypercubic array the number of processors can be very high, but there is little internal work to do while transferring the data.

The high cost of a staple calculation and the local structure of the plaquette action ensure that most simple geometric array structures will be suitable for a plaquette calculation. To illustrate this point consider the extreme case of a hypercubic array in which each processor is responsible for 1 site. It must calculate 24 staples. To do this 12 link matrices must be transferred from neighbours in the forward directions, and 12 must be sent to neighbours in the backwards directions. On completion 12 staples must be sent and received. In total 2 matrices must be transferred for each staple calculated.

If we give each processor a hyper-rectangular sub-lattice then using current Transputers restricts us to ring or grid array architectures. Processors such as that used in the Ncube machine and FPS-T20 have more communication channels and so avoid this restriction, but the bandwidth of their links is lower. In order to build higher dimensional structures from transputers we must combine several processors to build a "cell". We can, for example build a 4-d array from cells of 4 Transputers (see figure 5.8). The work done previously by a single processor must be divided equally amoungst those in the cell, and the extra I/O within the cell must be allowed for; both changes tend to reduce Γ.

5.1.3 An Algorithmic Approach

We have concentrated on the geometric parallelism in the staple calculation. We now look at the problem from the view-point of updating individual link matrices. In chapter 2 we identified the steps that can in principle be performed in parallel, calculating the six staples, summing them and completing the plaquettes, etc. The Southampton group [Askew et. al. 1986] refer to this as an algorithmic approach to parallelism. To exploit it we must design an appropriate machine configuration.

A Simple Working Model Figure 5.5 is a first step. We use it to begin an analysis of the computation and communication involved in this approach. Each

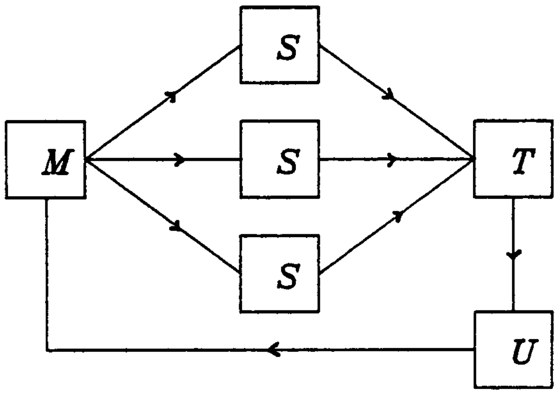


Figure 5.5: Exploiting algorithmic parallelism

of the 3 (S) processors calculates 2 staples. They are summed by (T) which then multiplies in the link to be updated. Processor (U) performs the update and processor (M) manages the memory. For high efficiency all of the processors must be busy (computing) all of the time. To facilitate this input and output data must be buffered and a supply of tasks maintained on all processors.

This simple model has I/O bottlenecks. The characteristic times τ_s and τ_m for calculating staples and transferring matrices are the same as those above, but the (S) processor has to move 4 matrices for each staple, and the (M) processor must output 9 matrices in this time.

Refining the Model We can solve such problems by modifying the design. Adding a second (M) processor doubles the $\{M\} \rightarrow \{S\}$ bandwidth, and increases the number of hard links used for input on the (S) processors, but at the cost of 2 processors that don't contribute to the calculation, rather than 1.

Changing the update algorithm, to heat bath or multi-hit metropolis will necessitate changing the usage of (T) and (U) and may require the addition of further processors for generating trial link variables. These modifications are fairly easy to incorporate as the inter-processor I/O required is low compared with the computation required.

The algorithmic machines outlined are very flexible. There are no restrictions on lattice shape or size, but their compute power is low. In order to perform gauge sector calculations on large lattices we must build arrays of such cells. Doing this further reduces the efficiency as more processors must be added to control and perform the inter-cell communication. Cells such as that in figure 5.6 could be connected in a ring, but the overall efficiency is limited to around 50%.

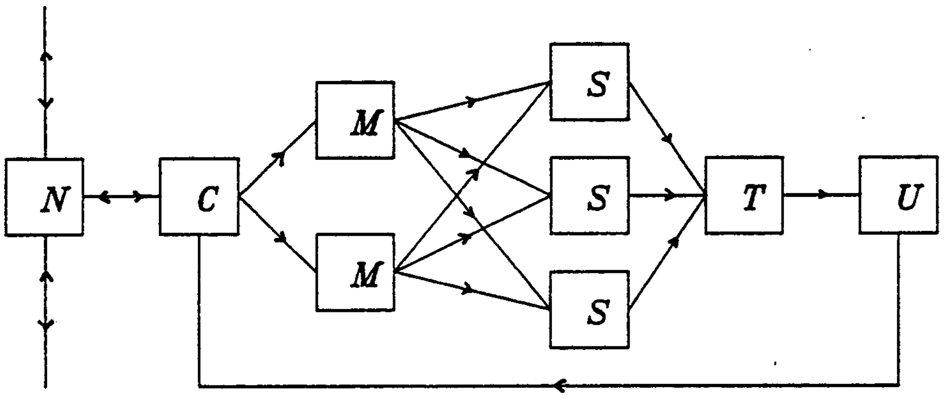


Figure 5.6: Algorithmic cell for gauge sector calculations.

5.1.4 Performance on T800s

Our compute to I/O ratios have been given in staples calculated per link matrix transferred. We now evaluate these ratios for IMS T800-20 processors (floating point Transputers). The timing data is based upon measured performance [Askew 1986]. A timing of 1.35 secs is quoted for 10000 full 3 by 3 complex matrix-matrix muliplies (108 multiplies, 90 additions). A staple calculation requires 286 floating-point operations, and so we estimate that it can be performed in $\sim$ 235 micro-secs on a T800. Transferring a 3×3 complex matrix requires 72 bytes of I/O; this should take $\sim$ 43 micro-secs from internal ram, rather more to external ram (see later). Hence the ratio of the time taken to calculate a staple to that required to transfer a 3 by 3 matrix $\frac{\tau_s}{\tau_t}$ is ~ 5.4

A Transputer can communicate data on all four of its links while performing computation. To exploit this capability it is vital to (a) minimise interference between the two and (b) ensure that communication is prioritised so that data transfer occurs at the first possible moment.

There is not sufficient on-chip memory to perform the entire calculation without recourse to external memory, but there is enough for 50 $SU(3)$ matrices. We should manage the memory as follows (i) the sub-lattice is held off-chip, (ii) links to be transferred out go directly from off-chip memory, (iii) links transferred in stay on-chip, (iv) data for the staples is brought on-chip, and the result left there for accumulation or transfer, (v) the link to be updated is brought to internal memory, (vi) the update is completed, and (vii) the result is moved off again. There should be sufficient space left on-chip for oft-used pieces of code.

The total link bandwidth for a T-800 is about 16 Mbytes/sec, (in and out on all 4 links), as compared with a memory bandwidth of $\sim$ 80 Mbytes/sec from on-chip Ram. We only make use of $\frac{1}{2}$ the link bandwidth for the torus as the transfers are uni-directional. Data transfers require processor cycles to set them up, and use up memory cycles, delaying the processor. Both effects will be small for large Γ.

To obtain the overlapped compute-to-I/O ratios Γ we should multiply the ratios of staples calculated to matrices transferred by $\frac{\tau_s}{\tau_t}$ and by the number of links used . In table 5.1 we give estimates of Γ for a variety of array and lattice

Array Size	lattice size		
	16^4	24^4	32^4
8^2	32.4	48.6	64.8
12^2	-	32.4	-
16^2	16.2	-	32.4
32^2	-	-	16.2

Table 5.1: Projected compute-to-overlapped I/O ratios for staple calculations on a torus.

sizes based on a staple-to-transfers ratio of $\frac{3n_0n_1}{2(n_0+n_1)}$ (this is the worst case, the internal directions), 20 Mbit/sec links and all 4 links being used together.

5.1.5 Summary of Gauge Sector

A torus is the best configuration of Transputers for staple calculations. There is no synchronous I/O, and the compute-to-concurrent I/O ratios are good. The draw-backs of the scheme are (i) it is inflexible and (ii) coding to maximise work overlapped with I/O may be complex. However, efficiency is high, and large arrays can be used for large calculations. The compute to I/O ratios given above are high implying that we could be using more processors.

Exploiting algorithmic parallelism is possible in the gauge sector calculations, but for high efficiency the machine configuration must be tuned to the algorithm being used. To make this possible a high degree of flexibilty is needed within a cell of say 10 processors, some of which must have large amounts of memory. It should be possible to eliminate I/O bottlenecks within such a cell by appropriate load balancing, but overall efficiency is restricted to around 50% by the number of processors that must be used for storing and forwarding data.

5.2 The Fermion Sector

As we have seen numerical solution of the lattice Dirac equation dominates dynamical fermion simulations. We aim to distribute the solution amongst large numbers of cooperating processors. We intend to use the Conjugate Gradient (CG) algorithm and we concentrate on the partitioning of its components: matrix-vector (sections 5.2.1 to 5.2.4) and scalar products (section 5.2.5). The basic matrix-vector product is to apply $\not{D}$ to a vector V. If V is on even sites then

$$(\not{D}V)(x) = \sum_{\mu=1}^{4} \eta_\mu(x) \left(U_\mu(x)\, V(x+\hat{\mu}) - U^\dagger_\mu(x-\hat{\mu})V(x-\hat{\mu})\right) \tag{5.1}$$

$\not{D}V$ is on odd sites. For the purposes of this discussion we assume that the link matrices are distributed according to a hyper-rectangular partitioning scheme that preserves *processor-data locality*. It is not necessary to move link data at any time during a solution of the lattice Dirac equation. Our basic unit of computation will be the matrix-vector product, and that of I/O the vector move; both for complex 3 triples. The time taken to perform these tasks is discussed in section 5.2.6.

5.2.1 Starting with a Torus

As before we consider an n_0 by n_1 by L^2 sub-lattice on each processor. For $\hat{\mu} = 2$ or 3, the internal directions, evaluating

$$U_\mu(x)\, V(x+\hat{\mu}) - U^\dagger_\mu(x-\hat{\mu})V(x-\hat{\mu})$$

on sites of a specified parity requires $n_0 n_1 L^2$ basic products and no I/O as the $V(x+\hat{2})$ and $V(x+\hat{3})$ data is internal. I/O is only necessary for the external directions. Consider the $\hat{\mu} = 0$ terms

$$\underbrace{U_0(x)V(x+\hat{0})}_{1} - \underbrace{U^\dagger_0(x-\hat{0})V(x-\hat{0})}_{2}$$

In order to complete term 1 we must transfer the $V(x+\hat{0})$ data for $x_0 = n_0 - 1$ ($x_0 = 0$ on the next processor) in the negative $\hat{0}$ direction. This requires $\frac{1}{2}n_1L^2$ basic transfers. Data for term 2 is in the correct place to perform the $\frac{1}{2}n_0n_1L^2$ products, but the result vectors must be shifted forwards when $x_0 = 0$, again $\frac{1}{2}n_1L^2$ basic transfers. The transfers are illustrated in figure 5.7. We can overlap

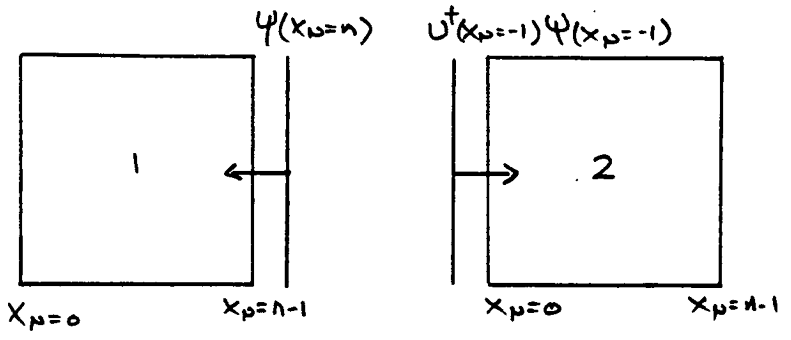


Figure 5.7: Transfers necessary for a partial $\not{D}$ in an external direction.

the work on the second term with the I/O for the first and vice versa. Further we can overlap the I/O for external directions with work for the internal ones. Calculating term 2 for one internal and one external direction requires $n_0n_1L^2$ basic products. We can use this to overlap the n_1L^2 vector transfers. The compute-to-I/O ratio is thus n_0 to 1. It is n_1 to 1 for the other direction. We use 2 links for each direction and the transfers are uni-directional. If we work on term 2 for all 4 directions while transferring the data for term 1 then we can make use of 4 links, giving us a peak compute to I/O ratio of $2n_0$ or $2n_1$ products (whichever is the smaller) to 1 vector transfer

In the limit of n_0 or n_1 equal to 1 the ratio is 2 products to 1 transfer. This is because there is no 'volume', all sites lie in the 'surface'.

5.2.2 Ring

Performance analysis for a ring is similar to that for the torus, except that there is now only one external direction n_0. Computation of the term 2 products for all 4 directions can be overlapped with I/O for the term 1 products in the external direction and the term 1 products can overlap shifting the boundary term 2 results in the external direction. We can only make use of 2 links, and so Compute to I/O ratios are the same those for the torus.

This scheme resticts us to at most L processors for an L^4 lattice and in this limit overlapped work to communications ratios Γ are at their worst. State-of-the-art dynamical fermion simulations will be being done on 16^4 lattices in the near future. 16 floating-point Transputers would not be nearly powerful enough for such calculations. In order to use a ring based machine for dynamical fermion simulations we would require processors of much higher computational power. Increasing the cpu speed reduces Γ so correspondingly higher link bandwidths would be required too.

5.2.3 Hypercube

The code fullqed uses 16 processors in a hypercube configuration. Is a general T^4 hypercube a good machine for solving the lattice Dirac equation in QCD ? We allocate an n^4 sub-lattice to each processor, where $n = \frac{L}{T}$ and arrange the calculation as above. Consider one direction $\hat{\mu}$, perform all the term 2 products in parallel with the I/O for term 1. The overlapped compute-to-I/O ratio is simply the ratio of sites in the bulk to sites in the $\hat{\mu}$ directed surfaces: $\frac{1}{2}n^4$ products to n^3 transfers. If we use 2 links in parallel then there are n products per transfer.

Building a general n^4 hypercube requires 8 links per processor, 2 for each direction. 2 links are used at a time in the scheme above. We can use all 8 links at a time if we overlap the term 2 products for all 4 directions $\hat{\mu}$ with all of the term 1 I/O; a compute to I/O ratio of $4n$ to 1. If we can drive large numbers of links efficiently in parallel this makes general hypercubes an excellent configuration. The Transputer only has 4 links per processor prohibiting the construction of such a hypercube.

5.2.4 Building Bigger Machines from Transputers

To build a general hypercubic array from Transputers we must combine several processors to build a "cell" with 8 inter-cell links (see figure 5.8. This cell can exploit the algorithmic parallelism present in applying $\not{D}$ to a vector. If we make each processor responsible for one direction $\hat{\mu}$ then they can calculate the partial $\not{D}$ for that direction. The compute-to-I/O ratios are n products to 1 transfer as 2 of the processor's links are inter-cell links, and can be used together. Having completed this work we must then add up the 4 terms. This requires 2 transfer steps each of $\frac{1}{2}n^4$ complex triples and $\frac{3}{2}n^4$ vector adds. The I/O falls into category I, as it cannot be overlapped with any significant quantities of work. The efficiency of this scheme is

$$\frac{2\tau_{\text{product}}}{2\tau_{\text{product}} + \tau_{\text{accumulate}}} = \frac{\max(n\tau_p, \tau_v)}{\max(n\tau_p, \tau_v) + n\tau_v + \frac{3}{2}n\tau_a}$$

where τ_a is the time taken to add 2 complex 3 triples, τ_p is the time taken to perform a basic product and τ_v is the time taken to transfer a complex triple. If we assume that $n\tau_p > \tau_v$ then the efficiency is

$$\frac{\tau_p}{\tau_p + \tau_v + \frac{3}{2}\tau_a} \qquad (5.2)$$

Timings for τ_p,τ_v and τ_a are given in section 5.2.6

5.2.5 Scalar Products

The Conjugate Gradient algorithm requires 2 scalar products per iteration; they collect global information. It is essential that our processor array, designed for the distributed computation of sparse matrix-vector products, be capable of accumulating scalar products rapidly. When the vectors are distributed calculation of their scalar product requires a local scalar product and the global accumulation of the local products followed by distribution of their sum.

Calculation of the local scalar products requires the accumulation of one product for each vector element on each processor. The sum of these products can be calculated in two ways: (1) communicate all the local products to a central point, add them and distribute the result. (2) accumulate the result on all processors concurrently.

On a ring of processors there is no real difference between the two methods. (1) requires $\frac{T}{2}$ single word transfers, T adds and then a further $\frac{T}{2}$ transfers. (2) requires T adds and transfers on each processor. On a higher (d) dimensional processor array it takes much longer to collect all the data at one point, at least

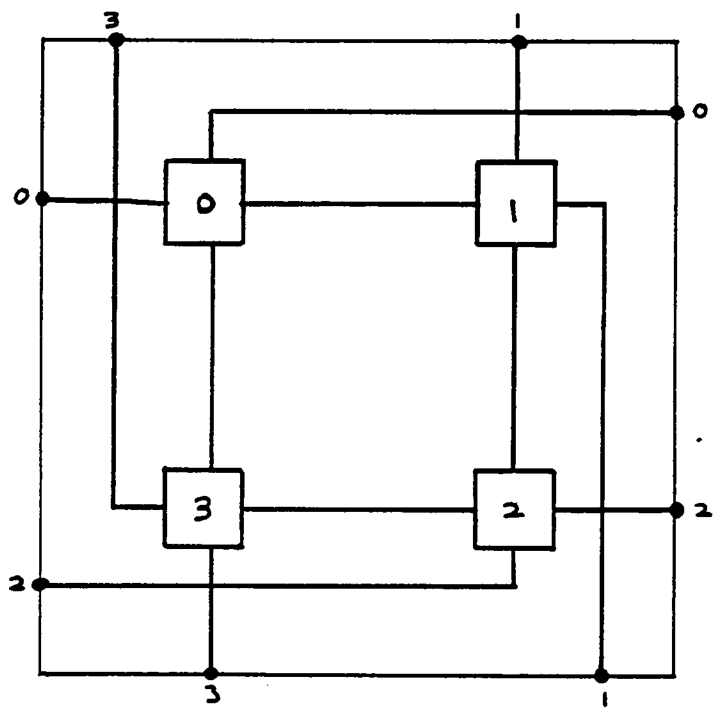


Figure 5.8: Cell of 4 processors.

$\frac{T^d}{l}$ transfers, where l is the number of possible concurrent transfers.

Using method (2) we can obtain the result in dT steps. Each processor is a member of d unique rings, one in each direction. There are $\frac{L^d}{T}$ such rings in each direction. We select a direction and accumulate all partial products on the rings in this direction; this takes T steps as before. We then change to the next direction, and sum the previously obtained partial products in this direction - on all processors. For d dimensions d such ring accumulations are necessary. On completion of dT single word transfers and adds the global scalar product is complete on all T^d processors.

The time taken to accumulate local scalar products is insignificant on all processor arrays considered here.

5.2.6 Timings on T800s

The product of a complex 3×3 matrix and a complex 3 vector requires 36 multiplications and 30 adds. 6 further adds are required while accumulating the results vectors. We should bring the data for the product into on-chip memory as all elements are used several times. τ_p should be ~ 60 micro-secs plus the time taken to load the data. The vector add time $\tau_a \sim 9$ micro-secs.

We must reserve 120 bytes of memory for the product (plus some more for its code), the remainder can be used for I/O buffers. A vector transfer into on-chip memory should take $\sim$ 15 micro-secs. Using this data we estimate the basic ratio

$$\frac{\tau_{\text{product}}}{\tau_{\text{transfer}}} \sim 4$$

for 20 Mbit/sec links. Provided the partial $\not{D}$ calculations are compute bound then the limiting efficiency (eq. 5.2) of the hypercube of cells configuration is

$$\frac{\tau_p}{\tau_p + \tau_v + \frac{3}{2}\tau_a} \sim 0.7$$

5.2.7 Summary of Fermion Sector

We use the data on ratios of basic products to vector moves and the performance data of section 5.2.6 to derive values of Γ for the designs studied. The data in table 5.2 is for a 16^4 lattice, a variety of array sizes are given where possible. The values of Γ given are for T800's with 20Mbit/sec links. An estimate of the efficiency is given, based upon the values of Γ and the amount of synchronous I/O and extra work required.

The small ring and binary hypercube configurations are efficient, but don't have the necessary processing power. The higher efficiency of the hypercube reflects a better partitioning of the data, into 8^4 rather than 1 by 16^3 blocks.

Of the medium power machines the 64 processor torus is best, the $2 \times 2 \times 16^2$ blocks imply a compute to I/O ratio of around 16 for the fast links. Partitioning the data for a binary hypercube of 4 processor cells gives higher values of Γ, but the scheme is less efficient as we must take account of the synchronous I/O required within the cell. A 64 processor machine does not have sufficient power for the 16^4 simulation.

The large machines are inflexible, but deliver the high processing power, and could be used efficiently. Compute overheads on the 256 processor torus are good for T800's. If a huge machine is required then *processor-data locality* necessitates a general hypercubic configuration, and if we want to use Transputers then we are forced to build 4 processor cells. There is some performance degradation due to accumulation of vectors within a cell, but efficiency is otherwise high.

5.3 Conclusions

Simulations of QCD with dynamical fermions require cpu intensive local calculations. We have shown that the criteria for success set out at the beginning of this chapter can be met using floating-point processors.

The geometric processor arrays used are rather inflexible as a result of the fine grain of parallelism used. Smaller arrays would be more flexible but less powerful. Exploiting algorithmic parallelism is more troublesome; efficiency is restricted by high levels of communication within a cell and processors needed for storing data. It is necessary to build arrays of algorithmic cells to obtain the required power and this can reduce efficiency further if processors are required for joining cells.

We advocate the use of a 16^2 grid of processors, with edges connected to form a torus, for dynamical fermion simulations on a 16^4 lattice. Further processors should be used to build replica machines, exploiting the immense job level

	number of processors	products vectors	Γ	efficiency	power
ring	16	2	8	acceptable	low
torus	8^2	4	16	high	medium
	16^2	2	8	acceptable	high
hypercube	2^4	8	32	very high	low
hypercube of	2^4 by 4	8	32	70% efficient	medium
cells	4^4 by 4	4	16	70% efficient	high

Table 5.2: Performance estimates for $\not{D}V$ on ring, torus, hypercube, and hypercube of cells for floating point Transputers, for a lattice of size 16^4.

parallelism.

Appendix A

The Distributed Array Processor

The ICL Distributed Array Processor (DAP) has 4096 bit-serial processing elements (PEs) each with 4096 bits of local memory. All PEs obey the same instruction simultaneously, applying it to local data, thus the machine operates in Single Instruction Multiple Data (SIMD) fashion. The PEs are arranged in a 64×64 grid. The local store can be regarded as a third dimension, giving the DAP the cubic structure shown in figure A.1. Each PE is connected to its 4

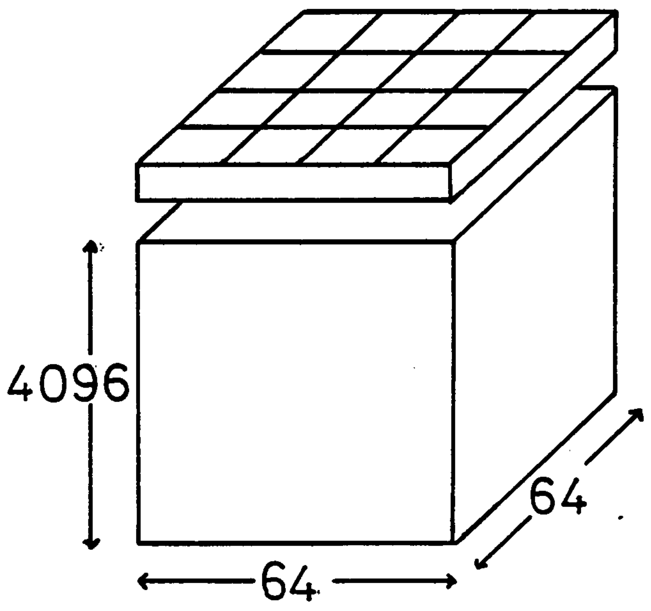


Figure A.1: The DAP.

nearest neighbours. Data can be shifted in any one of the 4 directions (boundary conditions are planar or cyclic). All PEs shift data in the same direction at the same time.The PEs have an activity control bit, if this is set FALSE then an instruction is not applied to the PEs data.

The DAP is programmed in DAP FORTRAN an extension of FORTRAN IV that includes vector and array constructs. The expression

$$X(,) = Y(,) + Z(,)$$

for real arrays X,Y,Z (of default size 64 by 64) causes all PEs to add their values of Y and Z and store the result in X. Instructions are also provided to sum the contents of vectors and arrays and shift data between PEs. Integer and floating point arithmetic is done in software, being built up from basic bit level instructions. An assembler (DAPL) is also available.

Functionally the DAP is a store module of its host 2976 mainframe, it is "active memory". The DAP is called from within a FORTRAN program running on this machine. Data is transferred between host and DAP by COMMON blocks common to both programs. Normally, this data transfer only takes place at the start and end of a job. Staff at the Edinburgh Regional Computer Centre (ERCC) have extended this facility to allow asynchronous paging while a program is running.

The DAPs operate at a rate of approximately 15 $\sim$ 20 Mflops, the data transfer software (DDX) has a peak speed of $\sim$ 250Kbytes/sec when the mainframe is (otherwise) lightly loaded.

The Edinburgh DAPs were decommisioned in July 1987 (when the host mainframe was taken out of service). Neither the DAP nor the broadly similar Goodyear Massively Parallel Processor (MPP) were a commercial success, but a new breed of SIMD machines, the ICL/AMT mini-DAP, the GEC Grid, and TM Connection Machine all use basically the same design.

A.1 Data Packing Schemes

In the 16^3 by N quenched hadron mass calculation programs data is partitioned in the time direction. The 16^3 spatial arrays are then packed into the 64^2 plane of the DAP by dividing it into 4 by 4 squares. Data is moved in the 1 and 2 directions by a shift of length 4.

$$Z(,) = SHNC(Y(,),4)$$

sets $Z(x) = Y(x+\hat{1})$ (with cyclic boundary conditions). 4 masked shifts of length 1 are required to move data in the $\hat{3}$ direction.

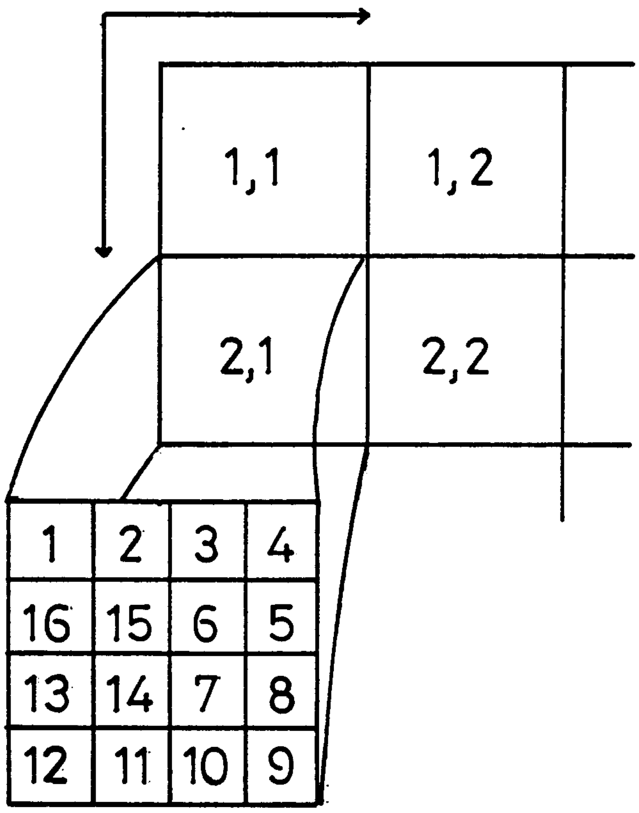


Figure A.2: Packing 16^3 into 64^2.

Appendix B

The Transputer

The Inmos Transputer is a micro-processor with its own local memory and links for communicatioon with other Transputers. It is intended as a component for building concurrent systems. The Transputer gives direct hardware support for concurrent communication and computation. Processor cycles are required to set up a transfer, but after that the interference is minimal.

The first Transputers (T414's) have a 32-bit integer processor (floating point arithmetic is done in software), 2K bytes of on-chip memory, and 4 links rated at 10 Mbit/sec - this was later upgraded to 20 Mbit/sec. The T800 floating point Transputer has an on-chip 64-bit floating point unit capable of around 1.2 Million multiplies per second, 4K bytes of on-chip memory, and faster links. It was developed as part of ESPRIT project P1085. Details of the T800's performance are given in [Askew 1986].

The Meiko Computing Surface is a reconfigurable Transputer array. It comprises compute, I/O and graphics boards - all based on Transputers. The Transputers links are connected together via electronic routing chips permiting the construction of application specific processor networks. A 42 processor Computing Surface was installed at Edinburgh University in April 1986. The Edinburgh Concurrent Supercomputer Project aims to build a machine with 1000 floating point Transputers and over 4 Gbytes of memory and use it for scientific simulations.

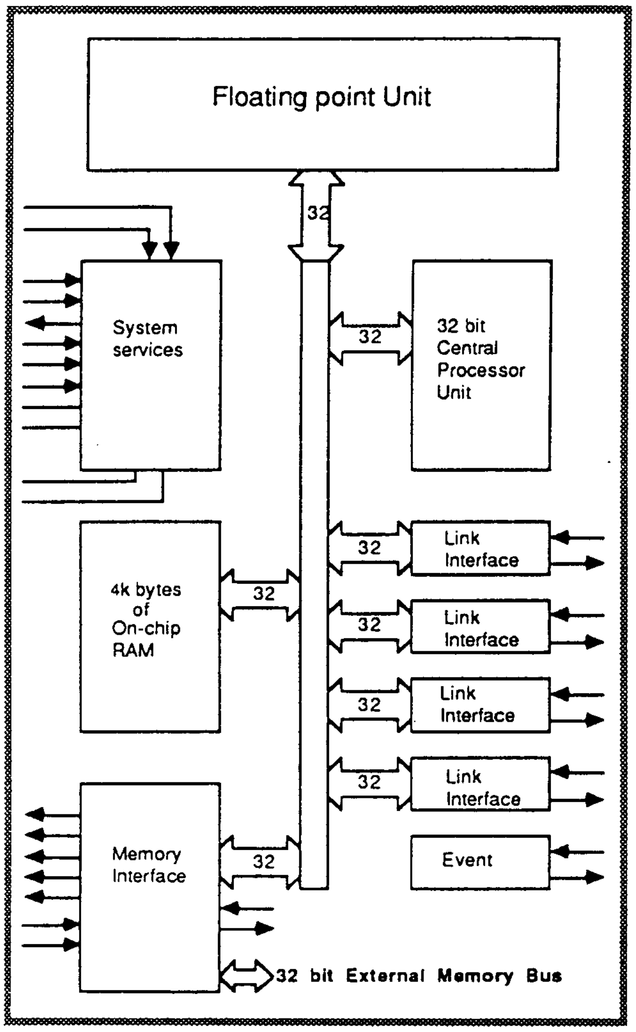


Figure B.1: Schematic diagram of the floating point Transputer.

Bibliography

[Arnison et al (UA1 collaboration) 1983] G.Arnison et al (UA1 collaboration), Phys. Lett. 122B, (1983), pp 103 and Phys. Lett. 122B, (1983), pp 273 and Phys. Lett. 126B, (1983), pp 398.

[Askew 1986] C.R.Askew. "T-800 Performance", Talk to the Occam User Group and subsequent newsletter.

[Askew et. al. 1986] C.R. Askew, et. al. "Monte Carlo Simulation on Transputer Arrays", Southampton University preprint SHEP 85/86-2.

[Azcoiti et. al. 1986] A. Azcoiti, A.Cruz, E.Dagotto, A.Moreo and A.Lugo. University of Illinois preprint P/86/4/58.

[Bagnaia et. al. (UA2 collaboration) 1983] P. Bagnaia et. al. (UA2 collaboration), Phys. Lett. 122B, (1983), pp 476.

[Barbour et. al. 1985a] I.M. Barbour, N.E. Behilil, P. Gibbs, G. Schierholz and M. Teper, "The Recursion Method and its Applications", Ed. D.G. Pettifor and D.L. Weaire, (1985), pp 149-164, (Springer-Verlag Berlin).

[Barbour et. al. 1985b] I.M. Barbour, P. Gibbs, K.C. Bowler and D. Roweth, Phys. Lett. 158B, (1985), pp 61.

[Barbour et. al. 1985c] I.M. Barbour, N.E. Behilil, P. Gibbs, M. Rafiq, K.J.M Moriarty, and G. Schierholz, "Updating Fermions with the Lanczos Method", DESY preprint 85-141.

[Barkai et. al. 1985] D. Barkai, K.J. Moriarty and C. Rebbi, Phys. Lett. 156B, (1985), pp 385 and Comput. Phys. Commun. 36, (1985), pp 1.

[Batrouni et. al. 1985] G.G. Batrouni, G.R. Katz, A.S. Kronfeld, G.P. Lepage, B.Svetitsky and K.G. Wilson, Phys. Rev. D32, (1985), pp 2735.

[Bernard et. al. 1983] C. Bernard, T. Draper and K. Olynyk, Phys. Rev. D27, (1983), pp 227.

[Billoire et. al. 1984a] A. Billoire, R. Lacaze, E. Marinari and A. Morel, Phys. Lett. 136B, (1984), pp 418.

[Billoire et. al. 1984b] A. Billoire, E. Marinari, A. Morel and J.P. Rodrigues, Phys. Lett. 148B, (1984), pp 169.

[Billoire et. al. 1985a] A. Billoire, R. Lacaze, E. Marinari and A. Morel, "The Goldstone Pion in SU(2) Lattice QCD at Low Quark Mass : Scaling and Finite Size Effects", Saclay Preprint PhT 85-180, (1985).

[Billoire et. al. 1985b] A. Billoire, E. Marinari and R. Petronzio, Nucl. Phys. B251, (1985), pp 141.

[Binder 1985] K. Binder, Phase Transitions and Critical Phenomena, C. Domb and M.S. Green Vol. 5, (1985), (Academic Press, New York).

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

[Bowler et. al. 1984a] K.C. Bowler, D.L. Chalmers, A. Kenway, R.D. Kenway, G.S. Pawley and D.J. Wallace, Nucl. Phys. B240, (1984), pp 213.

[Bowler et. al. 1984b] K.C. Bowler, R.D. Kenway, G.S. Pawley and D.J. Wallace, Phys. Lett. 145B, (1984), pp 88.

[Bowler and Pendleton 1984] K.C. Bowler and B.J. Pendleton, Nucl. Phys. B230 [FS10], (1984), pp 109.

[Bowler et. al. 1986] K.C Bowler, A. Hasenfratz, P. Hasenfratz, U. Heller, F. Karsch, R.D. Kenway, G.S. Pawley and D.J. Wallace, "The SU(3) β-function at Large β", Phys. Lett. 179B, (1986), pp 375-378.

[Bowler et. al. 1987a] K.C. Bowler, C.B. Chalmers, R.D. Kenway, G.S. Pawley and D. Roweth, "Hadron Mass Calculations Using Susskind Fermions at $\beta = 5.7$ and 6.0", Nucl. Phys. B284, (1987), pp 299-333.

[Bowler et. al. 1987b] K.C. Bowler, C.B. Chalmers, R.D. Kenway, G.S. Pawley, D. Roweth and D.B. Stephenson, "Hadron Mass Calculations Using Susskind Fermions at $\beta = 6.15$ and 6.3", Edinburgh Preprint 87/403.

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

[Calaway and Rahman 1982] D.J.E. Calaway and A. Rahman, Phys. Rev. Lett. 49, (1982), pp 613.

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

[Chalmers et. al. 1986] C.B. Chalmers, R.D. Kenway and D. Roweth, "Extending Quark Propagators in Time", Phys. Lett. B184, (1986), pp 63-68.

[Chalmers et. al. 1987] C.B. Chalmers, R.D. Kenway and D. Roweth, "Algorithms for Calculating the Quark Propagator on Large Lattices", J. Comp. Phys, (1987), and Edinburgh preprint 86/361.

[Chalmers 1987] C.B. Chalmers. Ph.D thesis, University of Edinburgh 1986.

[Cheng and Li 1984] T-P. Cheng and L-F. Li, "Gauge Theory of Elementary Particle Physics", 1984, (Clarendon Press, Oxford).

[Concus, Golub and O'Leary 1976] P. Concus, G.H. Golub and D.P. O'Leary, in "Sparse Matrix Computation", Ed. J.R. Bunch and D.J. Rose, 1976, (Academic Press, New York).

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

[Duane 1985] S. Duane, Nucl. Phys. B257 [FS14], (1985), pp 1680.

[Duane and Kogut 1985] S. Duane and J.B. Kogut, Phys. Rev. Lett 55, (1985), pp 2774 and Nucl.Phys. B275 [FS17], (1986), pp 385.

[Duane et al 1986] S. Duane, R.D. Kenway, B.J. Pendleton and D. Roweth, Phys. Lett. B176, (1986), pp 143-148.

[Duane et al 1987] S. Duane, A.D. Kennedy, B.J. Pendleton and D. Roweth, Edinburgh Preprint 87/402 and Florida Preprint FSU-SCRI-87-27.

[Feynman 1984] R.P. Feynman, Rev. Mod. Phys. 20, (1948), pp 367.

[Fucito et. al. 1982] F. Fucito, G. Martinelli, C. Omero, G. Parisi, R. Petronzio and F. Rapuano, Nucl. Phys. B210, (1982), pp 407.

[Fukugita et. al. 1986] M. Fukugita, Y. Oyanagi and A. Ukawa, Phys. Rev. Lett. 57, (1986), pp 953.

[Gell-Mann 1964] M. Gell-Mann, Phys. Lett. 8, (1964), pp 214.

[Geurin and Kenway 1980] F. Guerin and R.D. Kenway, Nucl. Phys. B176, (1980), pp 168.

[Gilchrist et. al. 1984] J.P. Gilchrist, H. Schneider, G. Schierholz and M. Teper, Phys. Lett. 136B, (1984), pp 87 and Nucl. Phys. B248, (1984), pp 214.

[Glashow 1961] S.L. Glashow, Nucl. Phys. B22, (1961), pp 579.

[Golub and Van Loan 1983] G.H. Golub and C.F Van Loan, "Matrix Copmutations", (1983), (North Oxford Academic, Oxford).

[Gottlieb et al 1987] S. Gottlieb, W. Liu, D. Toussant, R.L. Renken, and R.L. Sugar "Hybrid Molecular-dynamics algorithms for numerical simulation of QCD" Phys. Rev. D35(8), (1987), pp 2531-2542.

[Gross and Wilczek 1973] D. Gross and F. Wilczek, Phys. Rev. Lett. 30, pp 1343 and Phys. Rev. D8, (1973), pp 3633.

[Gupta et. al. 1987] R. Gupta, G. Guralnik, G. Kilcup, A. Patel, S.R. Sharpe and T. Warnock, Los Alamos preprint CLNS-87/79.

[Hamber and Parisi 1981] H. Hamber and G. Parisi, Phys. Rev. Lett. 47, (1981), pp 1792.

[Hamber et. al. 1982] H. Hamber, E. Marinari, G. Parisi and C. Rebbi, Phys. Lett. 108B, (1982), pp 314.

[Han and Nambu 1965] M. Han and Y. Nambu, Phys. Rev. B189, (1965), pp 1006.

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

[Itoh et. al. 1986] S. Itoh, Y. Iwasaki and T. Yoshie, "Hadron Masses in Quenched QCD," University of Tokyo preprint UTHEP-155, (1986).

[Jolicoeur et. al. 1984] T. Jolicoeur, H. Kluberg-Stern, A. Morel, M.Lev and N. Petersson, Nucl. Phys. B235 [FS11], (1984), pp 455.

[Kawamoto and Smit 1981] M. Kawamoto and J. Smit, Nuclear Physics B192, (1981), pp 100.

[Kennedy et. al. 1986] A.D. Kennedy, "Deconfining Transition in Lattice QCD", Florida State preprint FSU-SCRI 86/46,

[Kluberg-Stern et. al. 1983] H. Kluberg-Stern, A. Morel, O. Napoly and B. Petersson, Nucl. Phys. B220 [FS8], (1983), pp 423.

[Kogut 1983] J.B. Kogut, Rev. Mod. Phys. 55, (1983), pp 775.

[Kogut and Susskind 1975] J. Kogut and L. Susskind, Phys. Rev. D11, (1975), pp 395.

[Konig et. al. 1984] A. Konig, K-H. Mutter and K. Schilling, Phys. Lett. 147B, (1984), pp 145 and Phys. Lett. 157B, (1985), with J. Smit, pp 421.

[Kovacs, Sinclair and Kogut 1986] E.V.E Kovacs, D.K. Sinclair and J. Kogut "Return of the finite temperature phase transition in the chiral limit of lattice QCD", ANL-HEP preprint 86-137.

[Lang 1986] C.B. Lang. "Renormalisation flow in lattice QED", Phys. Rev. Lett. 57, (1986), pp 1986.

[Marinari et. al. 1981] E. Marinari, G. Parisi and C. Rebbi, Phys. Rev. Lett. 47, (1981), pp 1795.

[Mathews and Salam 1954/5] P.T. Matthews and A. Salam, Nuovo Cim. 12, (1954), pp 563 and Nuovo Cim. (1955), pp 120.

[Metropolis et. al. 1953] N.A. Metropolis, M.N. Rosenbluth, A.H. Rosenbluth, E. Teller and J. Teller, J. Chem. Phys 21, (1953), pp 1087

[Morel and Rodrigue 1984] A. Morel and J.P. Rodrigues, Nucl. Phys. B247, (1984), pp 44.

[Nielson and Ninomiya 1981] H.B. Nielsen and M. Ninomiya, Nucl. Phys. B185, (1981), pp 20 and Nucl. Phys. B193, (1981), pp 173.

[Oyanagi 1986] Y. Oyangi, "An Incomplete LDU decomposition of lattice fermions and its application to conjugate residual methods", University of Tsukuba preprint ISE-TR-86-57.

[Parisi and Wu 1981] G. Parisi and Y.-S. Wu, Sci. Sin. 14, (1981), pp 483.

[Pietarinen 1981] E. Pietarinen, Nucl. Phys. B190 [FS3], (1981), pp 349.

[Politzer 1973] H.D. Politzer, Phys. Rev. Lett. 30, (1973), pp 1346.

[Polonyi and Wyld 1983] J. Polonyi and H.W. Wyld, "Microcanonical Simulation of Fermionic Systems, Phys. Rev. Lett. 51, (1983), pp 3357.

[Ramond 1981] P. Ramond, "Field Theory A Modern Primer", 1981, (Benjamin, Reading Massachusetts).

[Reid 1971] J.K.Reid, In Proceedings of Conference on Large Sparse Sets of Linear Equations, 1971, (Academic Press, New York).

[Salam 1968] A. Salam, In "Elementary Particle Theory", Ed. N. Svartholm, 1968, (Almquist Forlag AB, Stockholm)

[Scalapino and Sugar 1981] D.J. Scalapino and R.L. Sugar, Phys. Rev. Lett. 46, (1981), pp 519.

[Sheard] Unpublished.

[Susskind 1977] L. Susskind, Phys. Rev. D16, (1977), pp 3031.

[Tomboulis 1983] E. Tomboulis, Phys. Rev. Lett. 50, (1983), pp 885.

[Toussaint 1987] D. Toussaint, "Supercomputations in QCD", UCSD Preprint UCSD-PTH 87/03.

[Weinberg 1964] S. Weinberg, Phys. Rev. Lett. 19, (1964), pp 1264.

[Weingarten 1982] D. Weingarten, Phys. Lett. 109B, (1982), pp 57. and Nucl. Phys. B215, (1982), pp 1.

[Wilson 1974] K.G. Wilson, Phys. Rev. D10, (1974), pp 2445, and in "New Phenomena in Subnuclear Physics (Erice 1975)", Ed. A. Zichichi, 1977, (Plenum, New York).

[Yang Mills 1954] C.N. Yang and R. Mills, Phys. Rev. 96, (1954), pp 191.

[Zinn-Justin 1986] J. Zinn-Justin. "Renormalisation and stochastic quantisation", Nucl. Phys. B275, (1986), pp 135.

[Zweig 1964] G. Zweig, CERN preprints TH401, TH412, 1964.