W&M ScholarWorks

Dissertations, Theses, and Masters Projects

Theses, Dissertations, & Master Projects

1991

U(1) string tension

Arthur F. Griesser
College of William & Mary - Arts & Sciences

Follow this and additional works at: https://scholarworks.wm.edu/etd

Part of the Physics Commons

Recommended Citation
Griesser, Arthur F., "U(1) string tension" (1991). *Dissertations, Theses, and Masters Projects.* William & Mary. Paper 1539623802.
https://dx.doi.org/doi:10.21220/s2-4mvf-cq91

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9207754

U(1) string tension

Griesser, Arthur F., Ph.D.

The College of William and Mary, 1991

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

U(1) STRING TENSION

A Dissertation

Presented to

The Faculty of the Department of Physics

The College of William and Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Doctor of Philosophy

by

Arthur Griesser

1990

APPROVAL SHEET

This dissertation is submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

[illegible]

Arthur Griesser

Approved, December 21 1990

[illegible]

Carl E. Carlson

[illegible]

Franz L. Gross

[illegible]

Robert T. Siegel

[illegible]

J. Michael Finn

Carl M. Andersen

Carl M. Andersen

DEDICATION

To My Superlative Wife, My Family, And The Zilinks

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Embarking upon graduate studies I sometimes wondered, idly thumbing through dissertations, whether I would wax as cloyingly maudlin as my predecessors when it came time to pen my acknowledgments. I now find that any reasonably complete compilation of acknowledgments would easily triple this volume. So instead of thanking John Hunter (who suggested building a computer), the excellent physicists at William and Mary who have taught me so much, my committee members (especially Mike Finn), the semiconductor manufacturers who donated chips, my friends (especially Doug Baker), favorite novelists, favorite musicians, and everyone else who has enriched my life, all the way back to my fourth grade teacher, I shall tip my hat only to Carl E. Carlson, physicist extraordinaire and volleyball hero, whose superhuman patience, generosity, and penetrating insight are destined to become legends, if they are not already. Besides, if her contributions to this work were chronicled in full, Jennifer L. Poor would die of embarrassment.

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The long distance force between quarks in the U(1) approximation to quantum chromodynamics is calculated on a home made reduced instruction set computer optimized for that purpose. It is found that previous calculations were in error by as much as 85% due to contamination by the Coulomb interaction. The Coulomb constant, measured for the first time in this work, agrees with analytically obtained values.

U(1) STRING TENSION

INTRODUCTION

Lattice gauge calculations provide alternatives to perturbative methods, based on first principles, and devoid of the assumptions of phenomenological models. Lattices have been used[1] to calculate gluonic and hadronic mass spectra, place an upper limit on the mass of the Higgs particle, and calculate hadronic weak matrix elements, but this work will be concerned with the interquark force, or string tension, the motivation behind the conception[2] of lattice gauge calculations by Kenneth G. Wilson in 1974.

Meson spectroscopy suggests a linear $q-\bar{q}$ potential, resulting in confinement due to a force between quarks which, for large distances, is independent of the interquark separation. Apparently gluon-gluon interactions bind chromoelectric flux into a tube or string. An appealingly intuitive model considers the gluonic cloud to be an incompressible fluid, which wets the quarks. When the quarks are far apart, any further separation requires work, since the formation of additional surface sunders glue-glue bonds. The string tension is thus the surface tension of glue. Asymptotic freedom occurs when the quarks are very close, and separating them slightly does not increase the surface area of the spherical glue drop in which they are embedded.

When sea quarks are omitted, SU(3) lattice gauge calculations do indeed demonstrate confinement which disappears when the coupling grows weak enough. Although one would not naively expect it, a string tension has been rigorously proven to exist[3] for all values of the coupling constant for 2+1 dimensional compact QED, which can be regarded as an Abelian U(1) approximation to SU(3) QCD. There is a puzzle in the measurement of this U(1) string tension: although its simplicity makes it a favorite testing ground for lattice calculations, no one has explained why apparently sensible modifications (SG) (PS) [4,5] yield U(1) string tensions very different from those produced by the standard algorithm (AHO) (IP) (WS). Why not sweep this disparity under the rug? Other lattice calculations rely on the sting tension to set the coupling strength and the spatial scale. Furthermore, previously published string tensions may well be contaminated by the $1/r$ Coulomb force expected from Gauss' law in two spatial dimensions. If you can not accurately calculate a relatively simple quantity such as the U(1) string tension, how can you possibly hope to obtain realistic answers to difficult problems, such as hadron masses? This work reports a more accurate string tension measurement for the Z(256) approximation to U(1), performed on a home made computer roughly comparable, for this calculation, to a Cray 1. The additional accuracy enabled more reliable separation of the string tension from the Coulomb force.

CHAPTER I. REVIEW OF LATTICE METHODOLOGY

The Statistical Mechanical Analogy for Path Integral Calculations

In Feynman's path integral approach to quantum mechanics[6], a system is described by its position in configuration space. The non-relativistic probability of a transition from one configuration to another is given by the square of the amplitude which is obtained by adding the amplitudes of every possible "fundamental" virtual trajectory in configuration space. The (unnormalized) amplitude for a single trajectory is a complex number given by

$$e^{\frac{i}{\hbar}\int_{t_i}^{t_f} dt\, L(q(t),\dot{q}(t))}$$

where L is the classic Lagrangian and q symbolizes all of the coordinates of the path (parameterized by t) in configuration space. The integration over paths is defined by discretizing time and integrating over the coordinates at each time:

$$\langle q_f t_f | q_i t_i \rangle = \int \mathcal{D}[q(t)]\, e^{\frac{i}{\hbar}\int_{t_i}^{t_f} dt\, L(q(t),\dot{q}(t))} \equiv \lim_{N\to\infty} \Omega \left[\prod_{n=1}^{N-1} \int_{-\infty}^{\infty} dq(t_n) \right] e^{\frac{i\Delta t}{\hbar}\sum_{n=0}^{N-1} L(q(t),\dot{q}(t))}$$

where $\Delta t \equiv (t_f - t_i)/N, \quad t_n \equiv t_i + n\Delta t, \quad \dot{q}(t_n) = [q(t_{n+1}) - q(t_n)]/\Delta t,$ and Ω is a constant which drops out after normalization. To simplify calculations a change of variables $t \to -i\tau$ is now made which converts the rapidly oscillating amplitudes to nicely damped exponentials. The behavior for real t is later obtained from that for τ by analytic continuation. This change of variable, resulting in a Euclidian metric, has an interesting side effect: if

$L=\frac{1}{2}\dot{q}^2-V(q)$ then the sign change due to $(1/dt)^2$makes *L(t)=L(-iτ)=-H(τ).* With the substitution $\beta \equiv N\,\Delta\tau/\hbar$, the transition probability becomes

$$\langle q_f t_f | q_i t_i\rangle = \lim_{N\to\infty}\Omega\left[\prod_{n=1}^{N-1}\int_{-\infty}^{\infty}dq(\tau_n)\right]e^{\frac{\beta}{N}\sum_{n=0}^{N-1}L(-i\tau_n)} = \lim_{N\to\infty}\Omega\left[\prod_{n=1}^{N-1}\int_{-\infty}^{\infty}dq(\tau_n)\right]e^{-\frac{\beta}{N}\sum_{n=0}^{N-1}H(\tau_n)}.$$

The quantum statistical (unnormalized) density matrix[7] is given by

$$\rho(x,x')=\sum_i \psi_i^*(x)e^{-\beta'\hat{H}}\psi_i(x') \qquad \beta'=1/k_bT.$$

To calculate the expected value of some quantum mechanical observable, when one quantal system is in thermal equilibrium with many others, one operates on the x' dependence of ρ and then takes the trace:

$$\bar{A}=\tfrac{1}{Z}\int_{-\infty}^{\infty}dx\sum_i \psi_i^*(x)\hat{A}e^{-\beta'\hat{H}}\psi_i^*(x)=\tfrac{1}{Z}\sum_i\left\langle\psi_i(x)\middle|\hat{A}e^{-\beta'\hat{H}}\middle|\psi_i(x)\right\rangle,$$

normalized by Z. The invariance of the trace under similarity transforms allows one to use any complete set for the ψ. A common choice is energy eigenfunctions,

$$\bar{A}=\tfrac{1}{Z}\sum_i\langle A\rangle_i e^{-\beta' E_i}.$$

To show the relationship between path integrals and thermodynamics, use position eigenstates (DW) instead:

$$\rho(q,q')=\int_{-\infty}^{\infty}dx\,\delta(x-q)\,e^{-\beta'\hat{H}}\delta(x-q')=\langle q|e^{-\beta'\hat{H}}|q'\rangle.$$

Defining $q_n\equiv q$ and $q_0\equiv q'$, and employing the closure of position eigenstates, one obtains

$$\langle q|\left(e^{-\frac{\beta}{N}\hat{H}}\right)^N|q'\rangle=\left[\prod_{n=1}^{N-1}\left(\int_{-\infty}^{\infty}dq_n\langle q_{n+1}|e^{-\frac{\beta}{N}\hat{H}}|q_n\rangle\right)\right]\langle q_1|e^{-\frac{\beta}{N}\hat{H}}|q_0\rangle.$$

If β'/N is sufficiently small,

$$\langle q_{n+1}|e^{-\frac{\beta'}{N}\left(\frac{\hat{p}^2}{2m}+V\right)}|q_n\rangle \cong \langle q_{n+1}|e^{-\frac{\beta'}{N}\frac{\hat{p}^2}{2m}}e^{-\frac{\beta'}{N}V}|q_n\rangle = \int_{-\infty}^{\infty} dq\langle q_{n+1}|e^{-\frac{\beta'}{N}\frac{\hat{p}^2}{2m}}|q\rangle\langle q|e^{-\frac{\beta'}{N}V}|q_n\rangle.$$

Expanding the position eigenstates in momentum eigenstates and using

$$\langle q|p\rangle = \frac{1}{\sqrt{2\pi\hbar}}e^{\frac{i}{\hbar}pq}, \ \langle \mathrm{p}|\mathrm{q}\rangle = \frac{1}{\sqrt{2\pi\hbar}}e^{-\frac{i}{\hbar}pq} \quad \text{and} \quad \Delta q_n \equiv q_{n+1} - q_n$$

provides

$$\begin{aligned}\langle q_{n+1}|e^{-\frac{\beta'}{N}\frac{\hat{p}^2}{2m}}|q_n\rangle &= \int_{-\infty}^{\infty} dp\langle q_{n+1}|p\rangle e^{-\frac{\beta'}{N}\frac{p^2}{2m}}\langle p|q_n\rangle \\ &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dp\, e^{\frac{i}{\hbar}pq_{n+1}}e^{-\frac{\beta'}{N}\frac{p^2}{2m}}e^{-\frac{i}{\hbar}pq_n} \\ &= \frac{1}{2\pi\hbar}\int_{-\infty}^{\infty} dp\, e^{-\left(-\frac{i}{\hbar}p\Delta q_n+\frac{\beta'}{N}\frac{p^2}{2m}\right)}\end{aligned}$$

which is readily integrated by completing the square and shifting variables, to yield

$$\langle q_{n+1}|e^{-\frac{\beta'}{N}\frac{\hat{p}^2}{2m}}|q_n\rangle = \sqrt{\frac{mN}{2\pi\hbar^2\beta'}}\, e^{-\frac{mN}{2\beta'\hbar^2}(\Delta q_n)^2}.$$

Together with the definitions

$$\Delta\tau' \equiv \beta'\hbar/N, \quad q(\tau_n') \equiv q_n, \quad \text{and} \quad H'(\tau_n') \equiv \frac{1}{2m}\left(\frac{m\Delta q_n}{\Delta\tau'}\right)^2 + V(q_n)$$

this results in

$$\rho(q,q') = \lim_{N\to\infty}\sqrt[N]{\frac{mN}{2\pi\hbar^2\beta'}}\left[\prod_{n=1}^{N-1}\int_{-\infty}^{\infty} dq(\tau_n')\right]e^{-\frac{\beta'}{N}\sum_{n=0}^{N-1}H'(\tau_n')}$$

which, except for the primes, is identical to the expression for the path integral. This similarity motivates the application of statistical mechanical techniques to quantum mechanics.

Although the density matrix is essential for the calculation of some observables, the measurable of classical thermodynamics can all be obtained from the partition function

$$Z = \mathrm{Tr}(\rho(q,q)) = \int_{-\infty}^{\infty} dq\langle q,-i\beta'\hbar|q,0\rangle.$$

To calculate the thermal average of an expectation value, operate with $\hat{A}$ on the q' dependence of ρ, set $q' = q$ and then integrate over q:

$$\overline{A} = \tfrac{1}{Z} \lim_{N\to\infty} \Omega \prod_{n=0}^{N-1} \int_{-\infty}^{\infty} dq_n\, A(q_0) e^{-\frac{\beta}{N} H'(\tau_n')} = \tfrac{1}{Z} \int_{-\infty}^{\infty} dq \langle q, -i\beta'\hbar | \hat{A} | q, 0 \rangle.$$

Since this system is supposed to be in thermal equilibrium, measurement of A at some $\tau_n \neq \tau_0$ should give the same result and $A(q_0')$ can be replaced by

$$\tfrac{1}{N} \sum_{n=0}^{N-1} A(q_n')$$

which reduces the statistical errors in $\overline{A}$ when the multiple integrations are performed by Monte Carlo techniques.

Importance Sampling Monte Carlo Integration

Trapezoidal integration approximates a function *f(x)* by *N* rectangles, each with a height equal to the function evaluated at the midpoint of the rectangle and a width $2\Delta x = (x_{\max} - x_{\min})/N$. The error associated with a single rectangle is easily found by Taylor expanding *f(x)* about the midpoint of the rectangle and integrating exactly the first nonzero error term. The symmetry due to expansion about the midpoint eliminates the odd terms, leaving an error per rectangle

$$\frac{1}{2!}\frac{\partial^2 f(x)}{\partial x^2}\int_{-\Delta x}^{+\Delta x} dx\, x^2$$

proportional to $1/N^3$. Since these errors are highly correlated they should be added to yield the total error, for a *d* dimensional integral, $1/N^{\frac{2}{d}}$. In contrast (DW), if the integral is expressed in terms of the mean value of *f(x)*, measured by averaging *f(x)* evaluated at evenly distributed random samples of *x*, the error would be proportional to $1/\sqrt{N}$. This Monte Carlo technique, although worthless for less than four dimensions, is the method of choice for higher dimensional integrations. Even the relatively simple calculation considered here has three million dimensions. A further improvement to the Monte Carlo method can be obtained by sampling the integrand where it contributes the most to the integral, instead of randomly. If the density of samples is given by an unnormalized function *P(x)*, the integral can be written as

$$\int_{x_1}^{x_2} dx\, f(x) = \left(\frac{1}{N}\sum^{N}\frac{f(x)}{P(x)}\right)\int_{x_1}^{x_2} dx\, P(x).$$

(A detailed derivation can be found in the appendix.) Usually *f(x)* is the product of two functions, one of which is equated to *P(x)*. Normalization of the resulting integral eliminates the integration over *P(x)*. The sequence of samples distributed with probability *P(x)* is obtained from the equilibrium distribution of a Markov chain generated by the Metropolis algorithm.

Markov Chains

A Markov chain (DW) is a sequence of probability distributions, $P_i(x)$, $P_{i+1}(x)$ ... where P_{i+1} is specified by P_i : some fraction *T(a,b)* of the probability density $P_i(a)$ will be assigned to $P_{i+1}(b)$. $P_{i+1}(b)$ will contain contributions from P_i for every value of a:

$$P_{i+1}(b) = \int da\ P_i(a)\ T(a,b)$$

Since all of P_i's probability is to be redistributed $\int db\ T(a,b) = 1.$

It can now be verified that P_{i+1} is normalized if P_i is normalized:

$$\begin{aligned}\int db\ P_{i+1}(b) &= \int db \int da\ P_i(a)\ T(a,b)\\ &= \int da\ P_i(a)\left[\int db\ T(a,b)\right] = \int da\ P_i(a)\end{aligned}$$

If the probability distribution is to be stable $\left(P_i(x) = P_{i+1}(x) \equiv P_E(x)\right)$ the flux out of $P_i(a)$ must equal the flux into $P_{i+1}(a)$:

$$\int db\ P_E(a)\ T(a,b) = \int db\ P_E(b)\ T(b,a)$$

This can be achieved if the integrands are equal, a condition known as microreversibility. As expected

$$\begin{aligned}P_{E+1}(b) &= \int da\ P_E(a)\ T(a,b) = \int da\ P_E(b)\ T(b,a)\\ &= P_E(b)\int da\ T(b,a) = P_E(b).\end{aligned}$$

To show that the distribution converges toward the equilibrium distribution, define the deviation from the equilibrium distribution as

$$D_i \equiv \int dx\ \left|P_i(x) - P_E(x)\right|.$$

Then

$$\begin{aligned} D_{i+1} &= \int dx \left| \int dy\, P_i(y)\, T(y,x) - P_E(x) \right| \\ &= \int dx \left| \int dy\, \left(P_i(y) - P_E(y)\right) T(y,x) \right| \\ &\le \int dx \int dy\, \left|P_i(y) - P_E(y)\right| T(y,x) = D_i . \end{aligned}$$

The equality can hold only when $P_i = P_E$: if $P_i \neq P_E$ then for some y $P_i(y) - P_E(y)$ must be negative. (If this were not so P_i would not be normalized.) $D_{i+1} < D_i$ for $P_i \neq P_E$ also implies there can be only one equilibrium: the distribution can not simultaneously converge to two.

The Metropolis Algorithm

The Metropolis algorithm[8] is way to generate random values of x with known probability $P_E(x)$. Construct an ensemble of calculators, each of which is to perform the same Monte Carlo integration. Any convenient distribution is used to assign an initial value x_i to each of the calculators. Each calculator then selects a new x according to the following algorithm:

```
Generate a trial $x$, $x_T$, with probability $T_0(x_i,x_T) = T_0(x_T,x_i) \neq 0$.
Calculate $r \equiv P_E(x_T) / P_E(x_i)$.
If $r \geq 1$,  $x_{i+1} = x_T$.
If $r < 1$ then accept $x_T$ with probability $r$:
    Generate a random number $R$ uniformly distributed in (0,1).
    If $R \leq r$ then $x_{i+1} = x_T$.
    If $R > r$ then $x_{i+1} = x_i$.
```

If this is done, the distribution of x in the ensemble will evolve as a Markov chain, because the Metropolis algorithm satisfies microreversibility:
If $r \geq 1$ then $T(x_i,x_T) = T_0(x_i,x_T)$ and $T(x_T,x_i) = \frac{1}{r}T_0(x_T,x_i)$. Conversely, if $r < 1$ then $T(x_i,x_T) = rT_0(x_i,x_T)$ and $T(x_T,x_i) = T_0(x_T,x_i)$. Either way

$$\frac{T(x_i,x_T)}{T(x_T,x_i)} = r = \frac{P_E(x_T)}{P_E(x_i)}.$$

After enough steps have been taken x will be distributed according to $P_E(x)$. The calculators never interacted, so except in the proof, there is no need for more than one.

Little has been said regarding x_T selection. The trial x can not be chosen completely randomly: When $P(x)$ is strongly peaked, a randomly chosen x will invariably result in a minuscule r, sox_T will never be accepted and no new configurations will be generated. It is better to generate a trial x

close to x_i, which is normally accomplished by the addition of a uniformly distributed perturbation to only one of the many variables defining the state x. Applying the Metropolis algorithm to each of the variables sequentially is referred to as sweeping the lattice. A single sweep will result in a new state highly correlated to the old one, so many sweeps must be performed to generate a new configuration. Another modification of the algorithm, especially advantageous when the acceptance is low and the calculation of r is slow, is to update a single variable several times (each update is called a hit) before moving on the next one. In the infinite hit limit this is known as the heat bath algorithm. It will be shown that multiple hits are not computationally efficient for the calculation reported here.

Glue on a Lattice

The gluonic SU(M) QCD Lagrangian density is

$$\mathcal{L} = -\tfrac{1}{2}\,\mathrm{Tr}(\mathbf{F}_{\mu\nu}\mathbf{F}^{\mu\nu})$$

where the gluonic Faraday tensor and vector potential have been "decolorized" by contraction with the group generators

$$\begin{aligned}\mathbf{F}_{\mu\nu} &\equiv \mathbf{T}^a F^a{}_{\mu\nu} = \mathbf{T}^a\,(\partial_\mu A^a{}_\nu - \partial_\nu A^a{}_\mu + g f^{abc} A^b{}_\mu A^c{}_\nu) \\ &= \partial_\mu \mathbf{A}_\nu - \partial_\nu \mathbf{A}_\mu - ig\left[\mathbf{A}_\mu, \mathbf{A}_\nu\right]\end{aligned}$$

$$\mathbf{A}_\mu \equiv A^a{}_\mu\,\mathbf{T}^a,$$

a process reversible trough the use of the generator's orthonormalization constraint

$$\mathrm{Tr}(\mathbf{T}^a\ \mathbf{T}^b) = \tfrac{1}{2}\delta^{ab}.$$

To place this theory on a lattice, each link between neighboring points $\vec{x}_i$ and $\vec{x}_j$ is associated with a variable

$$\mathbf{U}_{ji} \equiv e^{ig(\vec{x}_j - \vec{x}_i)^\mu\,\mathbf{A}_\mu\left(\frac{\vec{x}_j + \vec{x}_i}{2}\right)}.$$

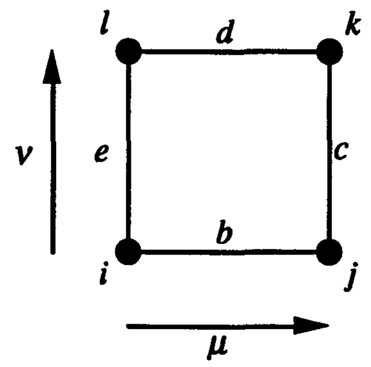


Figure 1. A coordinate system.

The product of the link variables around the smallest possible square on the lattice (known as a plaquette)

$$\mathbf{U}_{\square_{\mu\nu}} \equiv \mathbf{U}_{il}\mathbf{U}_{lk}\mathbf{U}_{kj}\mathbf{U}_{ji} = e^{-iga\mathbf{A}_\nu(e)}e^{-iga\mathbf{A}_\mu(d)}e^{iga\mathbf{A}_\nu(c)}e^{iga\mathbf{A}_\mu(b)}$$

with three applications of the Baker-Hausdorf identity (DW)

$$e^{\mathbf{A}}e^{\mathbf{B}} = e^{\mathbf{A}+\mathbf{B}+\frac{1}{2}[\mathbf{A},\mathbf{B}]} \qquad \text{(provided higher commutators vanish),}$$

becomes (neglecting terms of order a^3)

$$e^{iga\left(-\mathbf{A}_\nu(e)-\mathbf{A}_\mu(d)+\mathbf{A}_\nu(c)+\mathbf{A}_\mu(b)\right)-\frac{1}{2}g^2a^2\left\{[-\mathbf{A}_\nu,-\mathbf{A}_\mu]+[-\mathbf{A}_\nu,\mathbf{A}_\nu]+[-\mathbf{A}_\mu,\mathbf{A}_\nu]+[-\mathbf{A}_\nu,\mathbf{A}_\mu]+[-\mathbf{A}_\mu,\mathbf{A}_\mu]+[\mathbf{A}_\nu,\mathbf{A}_\mu]\right\}}$$
$$= e^{iga^2\mathbf{F}_{\mu\nu}}.$$

The Wilson action is defined as

$$S_\square \equiv \sigma(1-\tfrac{1}{M}\Re\mathrm{Tr}(\mathbf{U}_\square)) = \sigma(1-\tfrac{1}{2M}\mathrm{Tr}(\mathbf{U}_\square+\mathbf{U}_\square{}^\dagger)).$$

Evaluating the argument of the trace to lowest order

$$e^{iga^2\mathbf{F}_{\mu\nu}} + e^{-iga^2\mathbf{F}^\dagger{}_{\mu\nu}} = \sum_{n=0}^{\infty}\frac{(iga^2\mathbf{F}_{\mu\nu})^n}{n!} + \sum_{m=0}^{\infty}\frac{(-iga^2\mathbf{F}_{\mu\nu})^m}{m!}$$
$$= 2\sum_{l=0}^{\infty}\frac{(iga^2\mathbf{F}_{\mu\nu})^{2l}}{(2l)!} \cong 2 - g^2a^4\mathbf{F}_{\mu\nu}\mathbf{F}_{\mu\nu} \quad \text{(no implied sum)}$$

since $\mathbf{F}_{\mu\nu}$ is Hermitian. The action and Lagrangian density are therefore related by

$$\sum_{\mu\ \nu} S_{\square_{\mu\nu}} = -\frac{\sigma g^2a^4}{2M}\mathcal{L}.$$

When the exponential of this action is used as the path integral's integrand, one obtains $\frac{\beta}{N} = \frac{\sigma a g^2}{M}$. If the lattice spacing in the time dimension equals the lattice spacing in spatial directions then $c\,\Delta\tau = a$, $\frac{\beta}{N} = \frac{a}{\hbar c}$, and $\sigma = \frac{M}{g^2\hbar c}$.

To demonstrate the action's gauge invariance, consider application of the Baker-Hausdorf identity to the following transformation of a link variable (to first order in Λ):

$$\begin{aligned} e^{-i\Lambda_j} e^{igaA_\mu} e^{i\Lambda_i} &= e^{-i\Lambda_j + igaA_\mu + \frac{1}{2}[-i\Lambda_j, igaA_\mu] + i\Lambda_i + \frac{1}{2}[igaA_\mu, i\Lambda_i]} \\ &= e^{-i(\Lambda_j - \Lambda_i) + igaA_\mu + \frac{1}{2}ga\{[\Lambda_j, A_\mu] - [A_\mu, \Lambda_i]\}} \\ &= e^{-igaA'_\mu} \end{aligned}$$

where

$$\mathbf{A}'_\mu \equiv \mathbf{A}_\mu - \tfrac{1}{g}\partial_\mu \Lambda - i[\Lambda, \mathbf{A}_\mu] = \mathbf{T}^a\left(A^a{}_\mu - \tfrac{1}{g}\partial_\mu \Lambda^a + f^{abc}\Lambda^b A^c{}_\mu\right)$$

is the transformation of $\mathbf{A}_\mu$ required for local gauge invariance. Any product of link variables corresponding to a closed loop is gauge invariant because the gauge changing term to the right of a link variable will cancel with the gauge term to the left of the next link variable.

To calculate the interquark potential, the energy change is measured when a pair of massive quarks is plunged into the gluonic sea for a time T. To do this, a term

$$\mathcal{L}_{\text{int}} = 2\,\text{Tr}(\mathbf{J}_\mu\ \mathbf{A}_\mu) = J^{a\mu} A^a{}_\mu$$

corresponding to the quark-gluon interaction is added to the Lagrangian density and the propagator is calculated by the statistical mechanical analog explained earlier, using Monte Carlo integration with the Metropolis algorithm's importance sampling. This result, based on the complete interacting q-A field system is equated to that of a system composed of gluons and immobile quarks which evolve separately under the influence of a "fake" q-q potential that contains, in concentrated form, the complicated q-A and A-A interactions:

$$\begin{aligned}\int \mathcal{D}(\mathbf{A})\, e^{\frac{i}{\hbar}\int dt\,(L_{q-A}+L_{A-A})} &= \left\langle q_f \mathbf{A}_f t_f \middle| q_i \mathbf{A}_i t_i \right\rangle \\ &= \left\langle q_f \mathbf{A}_f \middle| e^{-(H_{q-A}+H_{A-A})N\Delta t} \middle| q_i \mathbf{A}_i \right\rangle \\ &= \left\langle q_f \middle| e^{-H_{q-q}T} \middle| q_i \right\rangle \left\langle \mathbf{A}_f \middle| e^{-H_{A-A}N\Delta t} \middle| \mathbf{A}_i \right\rangle \\ &\cong e^{-VT} \left\langle q_i \middle| q_i \right\rangle \left\langle \mathbf{A}_f t_f \middle| \mathbf{A}_i t_i \right\rangle.\end{aligned}$$

Usually periodic boundary conditions are applied and $\mathbf{A}_f = \mathbf{A}_i$ is also integrated. Examining the interaction Lagrangian density in more detail, for a quark transition from color *i* to color *f* the transition current is given by

$$\begin{aligned}J^{a\mu} &= ig\,\overline{q}_f \gamma^\mu\, \mathbf{T}^a q_i \\ &= ig\mathbf{T}^a{}_{fi}\, \overline{\psi}_f\, \gamma^\mu\, \psi_i \\ &\equiv ig\mathbf{T}^a{}_{fi}\, J^\mu_{i\to f}\end{aligned}$$

where ψ_i is the Dirac wave function for the i^{th} color component of q_i, $J^0_{i\to f} = 1/a^3$ for lattice points r' equipped with a quark, and $J^{\mu\neq 0}_{i\to f} = 0$. Inserting this interaction into the path integral's Lagrangian density, averaging over the quark color at the initial lattice point, and summing over the quark colors on lattice points r' yields an additional term

$$\begin{aligned}\prod_r e^{\frac{\beta}{N}a^3 L_{\text{int}}} &= \tfrac{1}{M}\prod_{r'}\left[\sum_{i_{r'}=1}^{M}\right]\left[\prod_{r'} e^{iga^4\, J^\mu_{i_{r'}\to i_{(r'+1)}} A^b_{\mu_{r'}} \mathbf{T}^b{}_{i_{(r'+1)}\, i_{r'}}}\right] \\ &= \tfrac{1}{M}\mathrm{TR}\prod_{r'}\left[e^{igaA^b_{\mu_{r'}}\mathbf{T}^b}\right] = \tfrac{1}{M}\mathrm{TR}\prod_{r'}\mathbf{U}_{r'\mu_{r'}}.\end{aligned}$$

Gauge invariance demands that r' form closed loops. Provided periodic boundary conditions are used, this will be the case if $T = N\,\Delta\tau$, but more commonly links along $\mu \neq 0$ are added to form a closed loop with $T < N\,\Delta t$. The resulting expectation value, $W(R,T) = Z(J)/Z$ with

$$Z(J) \equiv \int \mathcal{D}(\mathrm{A})\, e^{\frac{i}{\hbar}\int dt\,(L_{q-A}+L_{A-A})}$$

$$= \prod_{r\mu}\left(\int d\mathbf{U}_{r\mu}\right) e^{\sum\limits_{r\mu\nu} S_{\Box_{\mu\nu}}}\ \tfrac{1}{M}\mathrm{TR}\prod_{r'}\mathbf{U}_{r'\mu_{r'}}$$

and

$$Z \equiv \int \mathcal{D}(\mathrm{A})\, e^{\frac{i}{\hbar}\int dt\,(L_{A-A})}$$

$$= \prod_{r\mu}\left(\int d\mathbf{U}_{r\mu}\right) e^{\sum\limits_{r\mu\nu} S_{\Box_{\mu\nu}}},$$

is known as a Wilson loop. Wilson loops with $T < N\,\Delta\tau$ are only approximations because they use $J^{\mu\neq0}_{i\to f} = \frac{1}{a^3}$ rather than zero: symmetry considerations imply that the extra loop segments do not cancel for $T < N\,\Delta\tau$.

Thus the interquark potential can be expressed as

$$V(R) = -\underset{T\to N\,\Delta t}{\mathrm{Limit}}\ \tfrac{1}{T}\ln W(R,T).$$

For the U(1) case link variables assume the form

$$\mathbf{U}_{r\mu} = e^{i\theta_{r\mu}}$$

so with the definitions

$$\Delta_\mu \phi_r \equiv \phi_r - \phi_{r-\hat{\mu}}$$

$$\tilde{\Delta}_\mu \phi_r \equiv \phi_{r+\hat{\mu}} - \phi_r$$

$$\theta_{r\,\mu\,\nu} \equiv \tilde{\Delta}_\mu \theta_{r\,\nu} - \tilde{\Delta}_\nu \theta_{r\,\mu}$$

the Wilson action becomes

$$S_{\Box_{r\mu\nu}} = \sigma(1 - \cos\theta_{r\,\mu\,\nu}).$$

CHAPTER II. ANALYTIC RESULTS FOR U(1)

Strong Coupling

In the limit $\sigma \to 0$, glue only lattice gauge calculations can be performed exactly analytically[9]. Examining only the U(1) case, and cancelling some common factors, the Wilson loop can be written as

$$W = \left[\prod_{\text{Links}} \left(\int dU \right) e^{\sigma \sum_{\text{Plaquettes}} \Re U_{\square}} \prod_{\text{Boundary}} U \right] \Bigg/ \left[\prod_{\text{Links}} \left(\int dU \right) e^{\sigma \sum_{\text{Plaquettes}} \Re U_{\square}} \right].$$

Substituting the multinomial expansion

$$\left(\sum_{i=1}^{I} x_i \right)^m = \prod_{i=1}^{I} \left(\sum_{m_i=0}^{m} {}^{*} \right) \frac{m! \prod_{i=1}^{I} x_i^{\,m_i}}{\prod_{i=1}^{I} m_i!}$$

(where the asterisk indicates that the sums are constrained so that $\sum_{i=1}^{I} m_i = m$)

into the Taylor expansion of the exponential and using I for the number of plaquettes, one obtains

$$e^{\sigma \sum \Re U_{\square}} = \sum_{m=0}^{\infty} \tfrac{1}{m!} \left(\sigma \sum_{i=1}^{I} \Re U_{\square_i} \right)^m = \sum_{m=0}^{\infty} \left\{ \left(\tfrac{\sigma}{2}\right)^m \prod_{i=1}^{I} \left(\sum_{m_i=0}^{m} {}^{*} \right) \frac{\prod_{i=1}^{I} \left(U_{\square_i} + U_{\square_i}^{\dagger} \right)^{m_i}}{\prod_{i=1}^{I} m_i!} \right\}.$$

Fortunately, most of these terms vanish when integrated over dU:

$$\int dU\, U^{a}\, U^{*b} = \frac{1}{2\pi}\int_0^{2\pi} d\phi\; e^{i(a-b)\phi} = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{if } a = b \end{cases}.$$

It is clear that the lowest order nonzero term in the numerator must contain $m_i = 1$ for plaquettes with links on the Wilson loop boundary, so that all the Us in the product taken around the loop will be multiplied by a corresponding U^* from $(\Re U_{\Box_i})^{m_i}$. If only the plaquettes along the boundary of the Wilson loop are given $m_i \neq 0$ then for loops >2x2, $(\Re U_{\Box_i})^{m_i}$ will contain unmatched U^*s coming from unmatched links. A nonzero result can be obtained only if there are no boundary plaquettes with $m_i \neq 0$. Many such surfaces can be constructed (for example, the torus containing the Wilson loop could be tiled with plaquettes everywhere except inside the loop) but due to σ^m the largest contribution comes from the smallest number of plaquettes, the inside surface of the loop (providing the loop is less than half the size of the lattice). The result for an NxM loop is therefore $(\frac{\sigma}{2})^{\mathrm{N \cdot M}}$, which yields a string tension of $-\ln(\sigma/2)$.

Mean Field Theory

A mean field theoretical approximation[10] replaces all the link variables save one with their averages

$$\mathrm{U}_{\Box} \cong \langle \mathrm{U}_{il} \rangle \langle \mathrm{U}_{lk} \rangle \langle \mathrm{U}_{kj} \rangle \mathrm{U}_{ji} = \mathrm{U}_{ji} \langle \mathrm{U}_{ji} \rangle^{3} \equiv \mathrm{U}_{ji}\,\theta^{3}.$$

A slightly more cumbersome variational approach is available for SU(N)[11,12]: this mean field approximation is limited to U(1) because it immediately discards the non-Abelian nature of the fields. Calculating the expectation value of a plaquette under the presumption that θ is real, and using p to denote the number of plaquettes that share each link,

$$\langle \mathrm{U}_{\Box} \rangle = \theta^{4} = \frac{\prod\limits_{\text{Links}} \left(\int d\mathrm{U} \right) \Re(\mathrm{U}_{\Box})\, e^{-\sum\limits_{\text{Sites}} \sum\limits_{\text{Plaquettes}} s_{\Box}}}{\prod\limits_{\text{Links}} \left(\int d\mathrm{U} \right) e^{-\sum\limits_{\text{Sites}} \sum\limits_{\text{Plaquettes}} s_{\Box}}}$$

$$= \frac{\int_0^{2\pi} d\phi\, \Re(\theta^{3} e^{i\phi}) e^{-p\sigma\left(1-\Re(\theta^{3} e^{i\phi})\right)}}{\int_0^{2\pi} d\phi\, e^{-p\sigma\left(1-\Re(\theta^{3} e^{i\phi})\right)}}$$

$$= \frac{\theta^{3} \int_0^{2\pi} d\phi\, e^{p\sigma\theta^{3}\cos\phi} \cos\phi}{\int_0^{2\pi} d\phi\, e^{p\sigma\theta^{3}\cos\phi}} = \frac{\theta^{3}\,\mathrm{I}_1(p\sigma\theta^{3})}{\mathrm{I}_0(p\sigma\theta^{3})}.$$

This self consistency equation for θ can be solved numerically. For an RxT Wilson loop this gives $\theta^{2(R+T)}$, which does not result in a string tension at all.

Weak Coupling

Analytic calculations for weak coupling in U(1) have been reported by Polyakov[13], Banks et al (BMK) and Müller and Rühl (MR) [14]. (BMK) and (MR) both begin by Fourier expanding the exponentiated Wilson action:

$$Z(J) = \prod_{r,\mu}\left[\int_0^{2\pi} d\theta_{r\mu}\right]\prod_{\substack{r,\mu\\ \nu>\mu}}\left[e^{-\sigma(1-\cos\theta_{r\mu\nu})}\right]e^{i\theta_{r\mu}J_{r\mu}}$$

$$= \prod_{\substack{r,\mu\\ \nu>\mu}}\left[\sum_{l_{r\mu\nu}=-\infty}^{\infty}\right]\prod_{\substack{r,\mu\\ \nu>\mu}}\left[e^{-\sigma}\,\mathrm{I}_{l_{r\mu\nu}}(\sigma)\right]\prod_{r,\mu}\left[\int_0^{2\pi} d\theta_{r\mu}\right]e^{i(\theta_{r\mu}J_{r\mu}+\theta_{r\mu\nu}l_{r\mu\nu})}$$

expressing $\theta_{r\mu\nu}$ in terms of $\theta_{r\mu}$ it can be shown that $\theta_{r\mu\nu}l_{r\mu\nu} = \theta_{r\mu}\,\Delta_\nu l_{r\mu\nu}$. With this substitution the integrals reduce to delta functions

$$Z(J) = \prod_{\substack{r,\mu\\ \nu>\mu}}\left[\sum_{l_{r\mu\nu}=-\infty}^{\infty}\right]\prod_{\substack{r,\mu\\ \nu>\mu}}\left[e^{-\sigma}\,\mathrm{I}_{l_{r\mu\nu}}(\sigma)\right]\prod_{r,\mu}\left[2\pi\,\delta(J_{r\mu}+\Delta_\nu l_{r\mu\nu})\right]$$

Substitution verifies that a solution to the constraints is given by

$$l_{r\mu\nu} = -\varepsilon_{\mu\nu\lambda}(\Delta_\lambda l_r + B_{r\lambda})$$

where

$$B_{r\lambda} \equiv \varepsilon_{\nu\lambda\mu}\,\hat{3}_\mu\,\Delta_3^{-1}\,J_{r\lambda},$$

$$\Delta_3^{-1}\,\phi_r \equiv \sum_{j=-\infty}^{r_3-1}\phi_{(r_1,r_2,j)},$$

$\hat{3}$ is a unit vector in the 3 direction, and l_r is an integer-valued scalar field. Using the constraint equation, renormalizing the Bessel functions by dividing them by $\mathrm{I}_0(\sigma)$, dropping terms that cancel when ratios are taken, and then asymptotically expanding the Bessel functions gives

$$Z(J) = \prod_{r}\left[\sum_{l_r=-\infty}^{\infty}\right]\prod_{r,\mu}\left[\frac{I_{(\Delta_\mu l_r + B_{r\mu})}(\sigma)}{I_0(\sigma)}\right]$$

$$= \prod_{r}\left[\sum_{l_r=-\infty}^{\infty}\right]\prod_{r,\mu}\left[e^{\sum_{k=1}^{\infty} g_{2k}(\Delta_\mu l_r + B_{r\mu})^{2k}}\right].$$

The leading behavior of g_{2k} is σ^{1-2k}. The calculations diverge at this point. (MR) convert the sums over l_r to integrals, and equate

$$\ln\left(Z(J)/Z\right) = \sum_{n=1}^{\infty}\frac{W_n(J)}{n!\,\sigma^n}$$

from which they extract

$$W_2(R,T) = \tfrac{2}{D}W_1(R,T)$$

$$K_\mu(x) \equiv \sin^2 \pi x k_\mu$$

$$W_1(R,T) \equiv \tfrac{1}{2}\int_0^1 d^D k \frac{K_1(L)K_D(T)\left[K_1(1)+K_D(1)\right]}{K_1(1)K_D(1)\sum_\mu K_\mu(1)}.$$

For large R, $W_1(R,R) \cong \xi R + \eta R \ln R$ so (MR) predicts the U(1) coulomb force, but not the string tension.

Table 1. Numerical evaluations of $W_1(R,R)$ by (AHO)

R	$W_1(R,R)$	R	$W_1(R,R)$	R	$W_1(R,R)$
1	.3333	5	3.9709	9	8.779
2	1.0399	6	5.1004	10	10.082
3	1.9187	7	6.283	20	24.50
4	2.9050	8	7.511	50	75.

Unlike (MR), (BMK) obtains a string tension (without a Coulomb force) after a tortuous derivation. Instead of directly converting the sums over l_r to integrals, they truncate the expansion of the Bessel function ratio to order $1/\sigma$. The Poisson sum formula is then applied to integrate l_r. The result is

mathematically the partition function of a gas of magnetic monopoles interacting with two R x T rectangular sheets of monopoles of opposite polarity separated by one lattice spacing. Incomplete shielding by the monopole gas results in an energy density responsible for a string tension of

$$\frac{4\sqrt{2}}{\pi\sqrt{\sigma}} e^{-.253\pi^2\sigma} .$$

CHAPTER III. EXPERIMENTAL RESULTS FOR Z(256)

The Experimental Apparatus

The calculations described here were performed on SPAM (Stochastic Processing Automaton in Memory) which can be thought of as a pipelined microprogrammed RISC optimized for spin glass calculations, built out of memories and small scale logic. Reduced Instruction Set Computers are based on the observation that most of a CPU's instructions are rarely used: the resources they consume would be better applied toward improving the performance of the instructions that occupy the most time. Microprogrammed CPUs derive their internal control signals from a (sometimes very wide) memory. Each instruction, or opcode, consists of numerous microstates obtained sequentially from this memory.

An overview of SPAM is shown in Figure 1. It consists of an address generator, four megabytes of memory, fast hardware random number generators, a group logic unit, an arithmetic logic unit, and a look up table for exponentials. The address generator contains two counters, microstate lookup tables, and one adder for every dimension of the lattice. As well as providing control bits used throughout SPAM, the microstate ROM provides relative offsets to the adders, which convert the offsets to absolute addresses by adding the contents of the node counter, which contains the address of the lattice point associated with the variable being updated. When the microstate counter reaches its preprogrammed maximum value, it resets itself to zero, increments the node counter, and the first microstate is applied to the new lattice point. This scheme is possible because the same rather small set of instructions is applied to each lattice variable. Periodic boundary conditions are automatically implemented when the adder overflows into unused bits, provided the lattice size in each dimension is an integer power of two. The lattice size can be changed by replacing a small simple printed circuit card that combines the addresses for each dimension into a single 22 bit address, which is then provided to 4 Mb of byte wide dynamic memory.

The memory consists of four cards, each containing thirty six 150 nSec 41256 DRAM chips, controlled by an Advanced Micro Devices AM2968PC. The DRAM controller contains an address multiplexor and a refresh counter. Advancements in memory technology now allow 4Mb SIMMs made from memories with internal refresh counters (CAS-before-RAS refreshing) which would considerably simplify the design. Three of the memory cards contain parity error detection hardware: according to the memory data sheet, a cosmic ray induced memory error could be expected once every few months. Except for errors deliberately induced for testing purposes, no errors were ever observed in a year's operation. Debugging the error detection circuitry on the fourth card was therefore not considered worthwhile. When a calculation is in progress, SPAM is guaranteed to access the memory at half the required refresh frequency. Since experiment showed manufacturers specifications for this parameter are about a thousand fold higher than necessary, memory refresh was provided only when SPAM was not using the memory. Although the controlling 68000 sees the lattice memory as word wide, the two bytes are multiplexed to one when SPAM accesses the memory.

Group elements from the lattice memory are fed to a group multiplier, which in the current implementation consists of four parallel look up tables. Parallel lookup tables could be replaced by smaller serial tables, but increasing the length of the pipeline would reduce performance and complicate microstate programming. Limiting the group to 2^8 elements allows a single ROM with sixteen address bits to serve as a multiplier. Two of the tables, generating the group product of the partial product with either the new link variable or its inverse (selected by microstate), are fed to the partial product register which provides the other input to the group multiplier. This register can be set to the group identity element by one of the control microstates. The remaining two tables generate the 16 bit Wilson action of the product, which forms one input to an ALU (made from four 74LS181 and one 74LS182) which takes the sum or difference of groups of plaquettes. The ALU is 16 bits wide, but an 82C54 overflow counter makes it look like 48 bits if only additions are performed, enabling summation of measurements made at each lattice location. Differences in the action are exponentiated by a 16 bit look up table.

Good random numbers are difficult to generate, and SPAM needs two for every trial link. The most popular pseudorandom number generator is the linear congruential algorithm:

$$n_{i+1} = a \cdot n_i + b \text{ modulo } c,$$

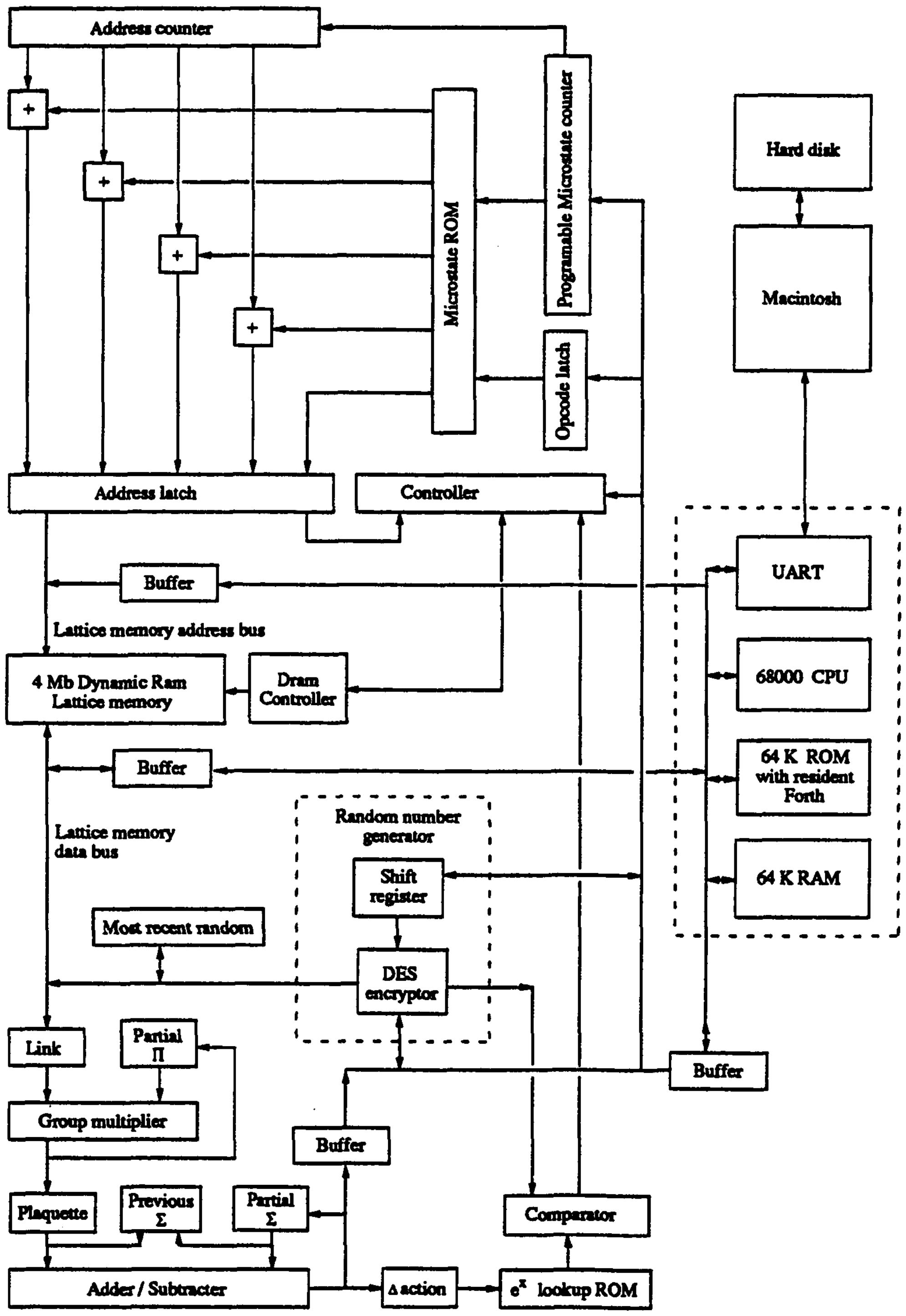
where a, b, and c are carefully chosen constants. Unfortunately the modulo operation is easily implemented in hardware only for unsatisfactory values of c. Shift register sequences[15] are easily generated in hardware, and they make excellent pseudorandom bit streams, but these bits can not be concatenated to produce random bytes. (PFT) suggests encrypting unsatisfactory random numbers using the DES algorithm. To test this notion, all of the random number tests suggested by Knuth[16] were implemented. Shift register sequences for an 88 cell register with feedback taps selected by Stahnke[17] spectacularly failed the poker test and the run test. The same random numbers after DES (PFT) encryption by an AM9518 data encryptor chip failed miserably only the coupon test. Since the resulting random numbers proved perfectly satisfactory, the failure of this test is distressing only in that it suggests the validity of rumors that DES was compromised by the NAS. It would be interesting to see if LUCIFER does a better job. In principle the encryptor could scramble its own output, but the length of the resulting sequence would be uncertain. The 88 bit shift register guarantees that the sequence would not have repeated itself if numbers were extracted from this generator at its maximum rate of two million bytes a second, starting from the creation of the universe, assuring there will be no overlap between the sequences initiated by two randomly selected seeds. Perhaps randomness would improve if the shift register sequence xor DES output was fed into the encryptor. Another way to improve randomness, suggested by Knuth, is to generate two pseudorandom numbers, one of which is used as an address for a small cache. The second random number replaces the one in the selected cache location, which becomes the next random number. This method of scrambling the sequence of pseudorandoms is not difficult to implement in hardware, but simulations showed no great improvement in randomness.

The controller generates some additional control signals from microstate bits, and initiates a new microstate once the dynamic RAM controller indicates the current memory access is finished. When the last lattice point has been processed, the controller stops requesting new microstates, and reactivates memory refresh.

SPAM is controlled by a small microcomputer built around a Motorola 68000 which runs a resident Forth obtained by modifying Pocket Forth[18], a public domain Forth interpreter/compiler based upon FLINT[19] (also public domain). The 68000 under the control of Forth selects opcodes for SPAM (by writing to the opcode latch supplying the six most significant bits to the microprogram memory), which applies them to each lattice point. The 68000 also initializes the random number generator (by setting the contents of the shift register, and several registers inside the encryptor) and can read or write to the lattice memory whenever it is not in use by SPAM. The 68000 communicates at 96000 baud with an Apple Macintosh which provides mass storage and a convenient user interface. It would have been better to attach the 68000 to the Mac's SCSSI port. Normally the Macintosh runs Mach-2 Forth[20] incorporating a terminal emulator and an assembler. The user can then send Forth source code or commands to the 68000 through the terminal emulator, or the two Forths can communicate with each other directly. Coupled programs are easily written. Both Forths are quite fast because they are subroutine threaded, but when every microsecond counts assembly language can be compiled on the Macintosh and downloaded to the 68000.

The speed of this computer is currently limited by the 250 nSec cycle time of the 150 nSec memories used in the lookup tables and the lattice. Now that dense memories as fast as 12 nSec are available this machine could be made much faster. Additionally, several GLU/ALUs could could be controlled by the the same address generator/ microstate controller, as in APE.

Figure 2. SPAM.



Microprogramming Language

Each of SPAM's 64 opcodes are composed of up to 512 microstates, specified by a simple microprogramming language. The microinstruction compiler (along with a decompiler and a hardware simulator) runs on the Macintosh, which also burns the lookup tables and opcode ROMS. In addition to Forth's usual amenities, the micro programming language consists of four compiler directives and eight pipeline control instructions together with two relative path specification instructions (e.g. z, -z) for each dimension. Control microinstructions precede the link with which they are stored. This can be a little confusing because most microinstructions act on information further down the pipeline. The complete microprogram used for this calculation can be found in Opcode appendix. After compilation the microprogram is burned into ROMs.

Table 2. Pipeline control microinstructions.

Instruction	Meaning
Minus	Subtract current plaquette sum from previous plaquette sum
Save	Save a link, if it was accepted
NewRand	Generate a new random number
NewNode	Store link address for later use by Save
DiffInit	Store current plaquette sum for later use by Minus
UseRand	Use a random number instead of a lattice variable
NewΠ	Set the partial group product to the identity element
NewΣ	Set the partial sum over plaquettes to zero

Table 3. Compiler directives.

Directive	Meaning
ParseInit	Initialize the compiler
P	Process the loop provided since the last P
EndOpcode	Finish the current opcode, start the next one
RomSave	Save the contents of each ROM in a separate binary file

Other Home Made Computers

The Array Processor Experiment, APE[21], of the Bologna-CERN-Padova-Rockefeller-Pisa-Rome collaboration is an elegantly simple SIMD machine featuring a circular array of memories coupled to a circular array of 32 bit floating point processors through a switch, all of which is under the control of microcode broadcast by a bit slice sequencer. A 3081 mainframe emulator calculates memory addresses, controls the sequencer, and executes some portions of the calculation. The switch connects each processor to one 16 Mbyte memory bank, so that processor n is connected to memory n+m modulo the number of processors. Each 64 Mflop APE FPU consists of four register files, four floating point multipliers, and four floating point ALUs, configured to multiply and add complex numbers. Microcode is generated by a modified optimizing FORTRAN which relieves the programmer of any need of hardware expertise. The current APE consists of 16 FPUs, but a seven million dollar 4096 FPU APE[22], with a projected speed of 100 Gflops, is currently under construction. For the sake of comparison, a ten million dollar Cray XMP is capable of about 1 Gflop.

The 256 node Columbia group's parallel processor[23,24] is a MIMD two dimensional torus of vector processor augmented 80286 computers. Each 64 Mflop vector processor is composed of two 32 bit floating point processors, four register files, a 256 Kbyte cache, and a writable control store which contains microcode provided by the 80286. The 80286 operates on an independent bus with 128 Kbytes of instruction memory and a Multibus interface to which a hard disk and commercial memory cards may be coupled. In addition to writing opcodes to the control store, the 80286 can access the vector processor's data cache. Software development requires the use of a microcode assembler.

The GF11[25], constructed at IBM's T.J. Watson Research Center, is an overly complicated SIMD arrangement of 566 20 Mflop processors and ten 450 Mbyte hard disks connected by a three level 24 channel switch. Each processor contains an ALU, a floating point unit capable of multiplication and addition, and 2 Mbytes of memory arranged in three levels of increasing speed. The processors are controlled by 180 bits of microcode, which can be modified somewhat by each processor's condition code. Although each processor may communicate with any of the others, and any of the hard disks, only 5% of them can do so at one time. Because these interconnections are specified by 8640 bits, they are chosen dynamically by the controller from a preselected set of 1024 configurations. Extensive effort went into the GF11's software. A precompiler translates a C like language with hardware specific extensions into pure C which, after compilation and execution, produces a sequence of operations to be executed by the GF11, which is then converted to microstates by an optimizer.

Caltech and Fermilab have constructed hypercubical arrays of MIMD single board computers with fast floating point hardware. A hypercube is the multidimensional analog of a cube: an n dimensional hypercube consists of two n -1 dimensional hypercubes connected at their corresponding vertices. An n dimensional hypercube consists of 2^n processors, each of which is connected to n other processors. By judiciously ignoring connections the hypercube can become a lower dimensional rectangular mesh. Each of the 32 nodes in Caltech's machine[26] consists of a 68020-68881 with 4 Mbytes of memory, a

128 Kbyte data cache, and a 16 Mflop XL chip set. Each of the 32 nodes in Fermilab's machine[27] is comprised of eight processors coupled by eight reconfigurable data paths. Each processor is equipped with a 20 Mflop XL chip set, 2 Mbytes of code memory, and 8 Mbytes of data memory.

Choice of Parameters

Before beginning a lattice calculation, one must choose a few parameters that affect the efficiency of the calculation. The same observable can sometimes be measured in many different but equivalent locations or orientations on the lattice. Provided these measurements are not too highly correlated, and that the time required for a measurement is smaller than the time required to generate a new configuration, it is advantageous to average together multiple measurements made on a single configuration. For various loop sizes, relative orientations, and relative displacements, the linear correlation coefficient between two loops was measured on a 16x16x16 lattice, using 100 configurations separated by 50 sweeps. Each such measurement was repeated 60 times in order to measure the error on the correlation coefficient. The results, shown in tables 4-12, indicate measurements of the same loop at different locations and orientations on the lattice, for a single configuration, are essentially uncorrelated.

Two additional parameters, the number of sweeps *T* separating configurations, and the number of hits *H* per sweep, must be traded off against each other. By increasing *H* one hopes to reduce the correlations between configurations so that they can be separated by fewer sweeps. The object is minimize the time required to obtain a new configuration from an old one. To do this, the correlation *R* between sweeps was measured using a single loop per sweep, for $H = 1$ to 9, $T = 1$ to 10, and loop size $L = 1$ to 8. Selected data from these measurements can be found in tables 13-15. Since one would naively expect an exponential decay, for each *L* and *H* the log of *R* was fit to a linear dependence on *T*:

$$\mathrm{Ln}(R) = I(L,H) + S(L,H) \cdot T.$$

The logs of the slope $S(L,H)$ and intercept $I(L,H)$ were then in turn fit to logarithmic dependence on H:

$$\mathrm{Ln}(-I(L,H)) = I_I(L) + S_I(L) \cdot \mathrm{Ln}(H)$$
$$\mathrm{Ln}(-S(L,H)) = I_S(L) + S_S(L) \cdot \mathrm{Ln}(H).$$

Since the time required to generate a new configuration is

$$\mathcal{T} = (Hm + p)\tau T$$

(where m is the number of microstates per hit, p is the number of microstates required to fill the pipeline, and τ is the time required to execute a microstate),

$$\mathcal{T}/\tau = (Hm + p)\ \frac{\mathrm{Ln}(R) + e^{I_I(L)+S_I(L)\cdot \mathrm{Ln}(H)}}{-e^{I_S(L)+S_S(L)\cdot \mathrm{Ln}(H)}}$$

can be used to print, for a given R, a table showing the relative time required to generate a new configuration as a function of H and L. An example for R=.1 is included as table 16. A more qualitative analysis is to generate from $S(L,H)$ a list of half-lives, such as table 17. The conclusions are that more than one hit per configuration would be a waste of time, and that eleven sweeps between configurations were sufficient to achieve an R of .1.

An even more important parameter is the thermalization time. Before measurements are made on the lattice, many configurations must be generated to make certain the Metropolis algorithm has converged. To demonstrate convergence the calculation can be run twice, once starting from a completely disordered lattice, and once starting from a completely ordered lattice. When the two calculations yield the same result the thermalization is complete. Alternatively the time constant for the thermalization can be estimated by fitting measurements to an exponential. Figures 3 and 4 show square Wilson loops averaged over the lattice as a function of the number of 25 sweep configurations. This data is for a 64x64x256 lattice at $\sigma = 2.0$.

Table 4. Linear correlations between 1x1 xy and xy plaquettes as a function of distance and direction.

$\lvert\Delta\vec{r}\rvert$	$\hat{x}$	$\hat{x}+\hat{y}$	$\hat{x}+\hat{y}+\hat{z}$
0	$1.000 \pm .000$	$1.000 \pm .000$	$1.000 \pm .000$
1	$-.035 \pm .032$	$-.047 \pm .022$	$.024 \pm .008$
2	$.016 \pm .030$	$.004 \pm .031$	$.021 \pm .027$
3	$-.051 \pm .028$	$-.016 \pm .035$	$-.032 \pm .023$
4	$.012 \pm .032$	$-.007 \pm .034$	$.002 \pm .029$
5	$.033 \pm .028$	$.024 \pm .028$	$.107 \pm .030$
6	$.033 \pm .026$	$-.021 \pm .038$	$.000 \pm .034$
7	$-.002 \pm .030$	$-.072 \pm .029$	$-.036 \pm .031$

Table 5. Linear correlations between 1x1 xy and xz plaquettes as a function of distance and direction.

$\lvert\Delta\vec{r}\rvert$	$\hat{x}$	$\hat{x}+\hat{y}$	$\hat{x}+\hat{y}+\hat{z}$
0	$-.072 \pm .028$	$-.072 \pm .028$	$-.072 \pm .028$
1	$-.039 \pm .037$	$.011 \pm .032$	$.032 \pm .037$
2	$-.005 \pm .014$	$-.013 \pm .027$	$.008 \pm .031$
3	$.005 \pm .032$	$-.013 \pm .024$	$.017 \pm .047$
4	$.039 \pm .036$	$.038 \pm .024$	$.003 \pm .025$
5	$-.029 \pm .016$	$.012 \pm .031$	$.016 \pm .028$
6	$-.058 \pm .025$	$-.025 \pm .021$	$-.028 \pm .031$
7	$.006 \pm .046$	$.022 \pm .019$	$-.020 \pm .024$

Table 6. Linear correlations between 1x1 yz and yz plaquettes as a function of distance and direction.

$\lvert\Delta\vec{r}\rvert$	$\hat{x}$	$\hat{x}+\hat{y}$	$\hat{x}+\hat{y}+\hat{z}$
0	$1.000 \pm .000$	$1.000 \pm .000$	$1.000 \pm .000$
1	$.111 \pm .035$	$.012 \pm .040$	$.010 \pm .034$
2	$.049 \pm .015$	$-.034 \pm .019$	$-.045 \pm .022$
3	$.004 \pm .036$	$.028 \pm .027$	$.010 \pm .035$
4	$.045 \pm .017$	$.009 \pm .023$	$.014 \pm .043$
5	$-.004 \pm .035$	$-.039 \pm .023$	$.032 \pm .026$
6	$-.030 \pm .022$	$-.053 \pm .023$	$-.012 \pm .049$
7	$.044 \pm .023$	$.034 \pm .020$	$-.069 \pm .029$

Table 7. Linear correlations between 4x4 xy and xy plaquettes as a function of distance and direction.

$\|\Delta\vec{r}\|$	$\hat{x}$	$\hat{x}+\hat{y}$	$\hat{x}+\hat{y}+\hat{z}$
0	1.000 ± .000	1.000 ± .000	1.000 ± .000
1	.165 ± .031	.077 ± .025	.001 ± .020
2	.062 ± .033	-.020 ± .030	.009 ± .018
3	.065 ± .022	-.018 ± .034	.011 ± .026
4	.039 ± .023	-.029 ± .031	-.022 ± .038
5	-.045 ± .041	.002 ± .025	.062 ± .029
6	-.013 ± .030	.082 ± .026	.022 ± .028
7	.051 ± .028	.030 ± .031	.021 ± .027

Table 8. Linear correlations between 4x4 xy and xz plaquettes as a function of distance and direction.

$\|\Delta\vec{r}\|$	$\hat{x}$	$\hat{x}+\hat{y}$	$\hat{x}+\hat{y}+\hat{z}$
0	-.024 ± .033	-.024 ± .033	-.024 ± .033
1	-.064 ± .023	-.008 ± .033	.003 ± .037
2	-.022 ± .022	.027 ± .033	-.040 ± .026
3	.018 ± .027	.004 ± .021	-.057 ± .035
4	.017 ± .035	.017 ± .039	.028 ± .029
5	-.022 ± .025	.027 ± .030	.083 ± .030
6	.025 ± .027	.039 ± .034	.004 ± .038
7	-.026 ± .033	-.034 ± .022	-.039 ± .024

Table 9. Linear correlations between 4x4 yz and yz plaquettes as a function of distance and direction.

$\|\Delta\vec{r}\|$	$\hat{x}$	$\hat{x}+\hat{y}$	$\hat{x}+\hat{y}+\hat{z}$
0	1.000 ± .000	1.000 ± .000	1.000 ± .000
1	.083 ± .026	.031 ± .028	-.007 ± .026
2	.075 ± .029	.025 ± .036	.054 ± .041
3	.091 ± .026	.015 ± .026	-.057 ± .027
4	.041 ± .022	.002 ± .045	-.019 ± .026
5	.005 ± .027	.038 ± .021	.004 ± .037
6	.002 ± .036	-.012 ± .032	-.034 ± .023
7	.026 ± .028	.015 ± .034	.016 ± .030

Table 10. Linear correlations between 8x8 xy and xy plaquettes as a function of distance and direction.

$\lvert\Delta\vec{r}\rvert$	$\hat{x}$	$\hat{x}+\hat{y}$	$\hat{x}+\hat{y}+\hat{z}$
0	1.000 ± .000	1.000 ± .000	1.000 ± .000
1	.044 ± .025	-.027 ± .017	.038 ± .034
2	-.015 ± .020	.046 ± .027	-.003 ± .028
3	.028 ± .025	.031 ± .025	.004 ± .019
4	.007 ± .033	-.005 ± .024	.012 ± .021
5	.013 ± .032	.030 ± .025	-.006 ± .033
6	.007 ± .035	.007 ± .031	-.007 ± .032
7	-.024 ± .039	-.006 ± .024	-.032 ± .036

Table 11. Linear correlations between 8x8 xy and xz plaquettes as a function of distance and direction.

$\lvert\Delta\vec{r}\rvert$	$\hat{x}$	$\hat{x}+\hat{y}$	$\hat{x}+\hat{y}+\hat{z}$
0	-.020 ± .016	-.020 ± .016	-.020 ± .016
1	.007 ± .028	-.008 ± .018	.020 ± .037
2	-.009 ± .022	-.001 ± .025	.015 ± .038
3	-.067 ± .051	.015 ± .033	-.013 ± .040
4	.025 ± .024	-.050 ± .030	-.026 ± .034
5	-.053 ± .023	-.012 ± .036	-.030 ± .022
6	-.046 ± .032	-.007 ± .032	-.044 ± .026
7	-.051 ± .025	-.031 ± .033	.004 ± .038

Table 12. Linear correlations between 8x8 yz and yz plaquettes as a function of distance and direction.

$\lvert\Delta\vec{r}\rvert$	$\hat{x}$	$\hat{x}+\hat{y}$	$\hat{x}+\hat{y}+\hat{z}$
0	1.000 ± .000	1.000 ± .000	1.000 ± .000
1	.011 ± .029	.008 ± .034	-.025 ± .030
2	-.014 ± .025	-.024 ± .024	.011 ± .040
3	-.003 ± .029	.007 ± .027	-.018 ± .040
4	.004 ± .031	.043 ± .021	-.016 ± .033
5	-.009 ± .025	-.003 ± .024	-.047 ± .027
6	.072 ± .029	-.044 ± .036	-.000 ± .032
7	.035 ± .033	.014 ± .022	-.027 ± .031

Table 13. Linear correlations between Wilson loops as a function of sweep separation and loop size, for one hit.

Δt	1x1	2x2	3x3	4x4
1	.734 ± .008	.737 ± .008	.737 ± .010	.713 ± .013
2	.541 ± .012	.565 ± .012	.586 ± .013	.565 ± .014
3	.390 ± .015	.447 ± .014	.479 ± .015	.467 ± .015
4	.292 ± .016	.351 ± .016	.392 ± .017	.386 ± .016
5	.219 ± .016	.278 ± .017	.316 ± .018	.323 ± .017
6	.161 ± .016	.212 ± .019	.262 ± .018	.277 ± .017
7	.115 ± .017	.165 ± .020	.217 ± .019	.238 ± .018
8	.074 ± .016	.133 ± .020	.182 ± .020	.198 ± .018
9	.041 ± .017	.103 ± .019	.156 ± .020	.165 ± .020
10	.025 ± .017	.083 ± .019	.135 ± .020	.147 ± .021

Δt	5x5	6x6	7x7	8x8
1	.662 ± .010	.557 ± .011	.426 ± .014	.338 ± .013
2	.504 ± .014	.381 ± .015	.259 ± .015	.170 ± .012
3	.399 ± .016	.282 ± .016	.174 ± .013	.114 ± .012
4	.334 ± .017	.226 ± .016	.126 ± .015	.066 ± .012
5	.276 ± .018	.197 ± .016	.087 ± .015	.051 ± .012
6	.237 ± .019	.165 ± .016	.061 ± .015	.042 ± .012
7	.214 ± .019	.132 ± .017	.049 ± .014	.033 ± .012
8	.186 ± .019	.109 ± .017	.046 ± .013	.025 ± .011
9	.160 ± .019	.093 ± .016	.041 ± .013	.007 ± .011
10	.147 ± .018	.081 ± .016	.027 ± .012	-.006 ± .010

Table 14. Linear correlations between Wilson loops as a function of sweep separation and loop size, for two hits.

Δt	1x1	2x2	3x3	4x4
1	.600 ± .009	.632 ± .010	.659 ± .011	.648 ± .011
2	.380 ± .013	.445 ± .013	.496 ± .015	.497 ± .014
3	.246 ± .015	.330 ± .014	.399 ± .016	.409 ± .015
4	.175 ± .016	.256 ± .016	.328 ± .017	.340 ± .017
5	.133 ± .016	.202 ± .016	.270 ± .017	.285 ± .017
6	.099 ± .016	.157 ± .018	.221 ± .017	.244 ± .018
7	.066 ± .015	.123 ± .018	.188 ± .017	.214 ± .019
8	.048 ± .015	.109 ± .017	.164 ± .018	.191 ± .019
9	.038 ± .014	.097 ± .017	.138 ± .020	.169 ± .019
10	.036 ± .014	.087 ± .016	.113 ± .020	.143 ± .020

Δt	5x5	6x6	7x7	8x8
1	.558 ± .013	.437 ± .011	.307 ± .011	.219 ± .011
2	.411 ± .015	.304 ± .013	.176 ± .012	.117 ± .011
3	.328 ± .016	.227 ± .014	.138 ± .013	.095 ± .011
4	.266 ± .017	.175 ± .015	.115 ± .013	.062 ± .011
5	.221 ± .017	.135 ± .015	.096 ± .012	.052 ± .010
6	.188 ± .017	.123 ± .016	.072 ± .012	.032 ± .012
7	.168 ± .017	.100 ± .015	.055 ± .012	.026 ± .010
8	.157 ± .016	.088 ± .015	.055 ± .011	.009 ± .012
9	.135 ± .017	.073 ± .015	.053 ± .012	-.004 ± .012
10	.129 ± .018	.071 ± .014	.049 ± .013	-.006 ± .010

Table 15. Linear correlations between Wilson loops as a function of sweep separation and loop size, for four hits.

Δt	1x1	2x2	3x3	4x4
1	.448 ± .010	.520 ± .011	.574 ± .011	.560 ± .013
2	.214 ± .014	.328 ± .015	.401 ± .014	.417 ± .014
3	.130 ± .014	.228 ± .015	.301 ± .016	.321 ± .015
4	.080 ± .013	.170 ± .015	.233 ± .017	.259 ± .017
5	.043 ± .013	.129 ± .015	.186 ± .017	.209 ± .018
6	.042 ± .013	.117 ± .015	.167 ± .017	.188 ± .019
7	.031 ± .014	.097 ± .016	.151 ± .017	.159 ± .017
8	.031 ± .013	.090 ± .016	.118 ± .017	.135 ± .017
9	.015 ± .014	.072 ± .016	.101 ± .018	.116 ± .017
10	.015 ± .015	.063 ± .016	.089 ± .017	.105 ± .016

Δt	5x5	6x6	7x7	8x8
1	.499 ± .010	.355 ± .011	.213 ± .013	.115 ± .011
2	.350 ± .013	.255 ± .014	.135 ± .012	.067 ± .011
3	.283 ± .015	.211 ± .014	.125 ± .011	.033 ± .010
4	.225 ± .016	.150 ± .014	.097 ± .011	.035 ± .012
5	.193 ± .016	.116 ± .014	.063 ± .012	.035 ± .010
6	.168 ± .016	.109 ± .014	.040 ± .011	.012 ± .010
7	.130 ± .016	.086 ± .014	.027 ± .011	.026 ± .009
8	.113 ± .016	.063 ± .014	.033 ± .011	.011 ± .010
9	.093 ± .016	.061 ± .013	.022 ± .010	.019 ± .011
10	.083 ± .016	.045 ± .015	.012 ± .013	-.002 ± .011

Table 16. Number of microstates required to achieve R =.1 as a function of number of hits and Wilson loop size.

hits	1x1	2x2	3x3	4x4	5x5	6x6	7x7	8x8
1	259	333	415	454	415	297	200	135
2	305	437	532	580	534	396	245	149
3	348	531	643	702	648	486	283	150
4	383	617	747	819	756	570	313	139
5	411	693	843	930	860	647	337	117
6	432	760	933	1037	960	718	355	84
7	445	818	1018	1139	1055	785	368	41
8	451	868	1096	1237	1147	846	376	-11

Table 17. Number of sweeps required to halve Wilson loop correlations across configurations as a function of loop size and number of hits.

Size	1 hit	2 hits	3 hits	4 hits
1	2.2 ± 0.1	1.8 ± 0.1	1.4 ± 0.1	1.2 ± 0.1
2	2.8 ± 0.2	2.5 ± 0.2	2.3 ± 0.2	2.3 ± 0.2
3	3.4 ± 0.3	3.3 ± 0.3	2.9 ± 0.3	2.8 ± 0.3
4	3.7 ± 0.4	3.7 ± 0.4	3.2 ± 0.3	3.3 ± 0.4
5	3.6 ± 0.4	3.6 ± 0.5	3.3 ± 0.5	3.1 ± 0.4
6	2.8 ± 0.3	2.8 ± 0.4	3.0 ± 0.7	2.8 ± 0.5
7	1.9 ± 0.2	2.7 ± 0.5	2.8 ± 0.8	2.5 ± 0.9
8	1.5 ± 0.2	1.8 ± 0.4	1.4 ± 0.4	2.4 ± 1.6

Size	5 hits	6 hits	7 hits	8 hits
1	1.2 ± 0.1	1.3 ± 0.1	1.4 ± 0.1	1.5 ± 0.2
2	2.3 ± 0.2	2.5 ± 0.3	2.7 ± 0.4	2.6 ± 0.3
3	2.8 ± 0.3	3.0 ± 0.4	3.1 ± 0.4	3.0 ± 0.4
4	3.3 ± 0.4	3.4 ± 0.5	3.4 ± 0.5	3.2 ± 0.4
5	3.5 ± 0.6	3.4 ± 0.6	3.5 ± 0.6	3.2 ± 0.5
6	3.2 ± 0.7	3.2 ± 0.7	3.4 ± 0.8	2.9 ± 0.8
7	2.3 ± 0.7	2.4 ± 0.7	2.8 ± 1.2	1.9 ± 0.6
8	2.3 ± 2.0	1.6 ± 1.0	1.4 ± 0.8	1.3 ± 1.1

Figure 3. Wilson loop thermalization

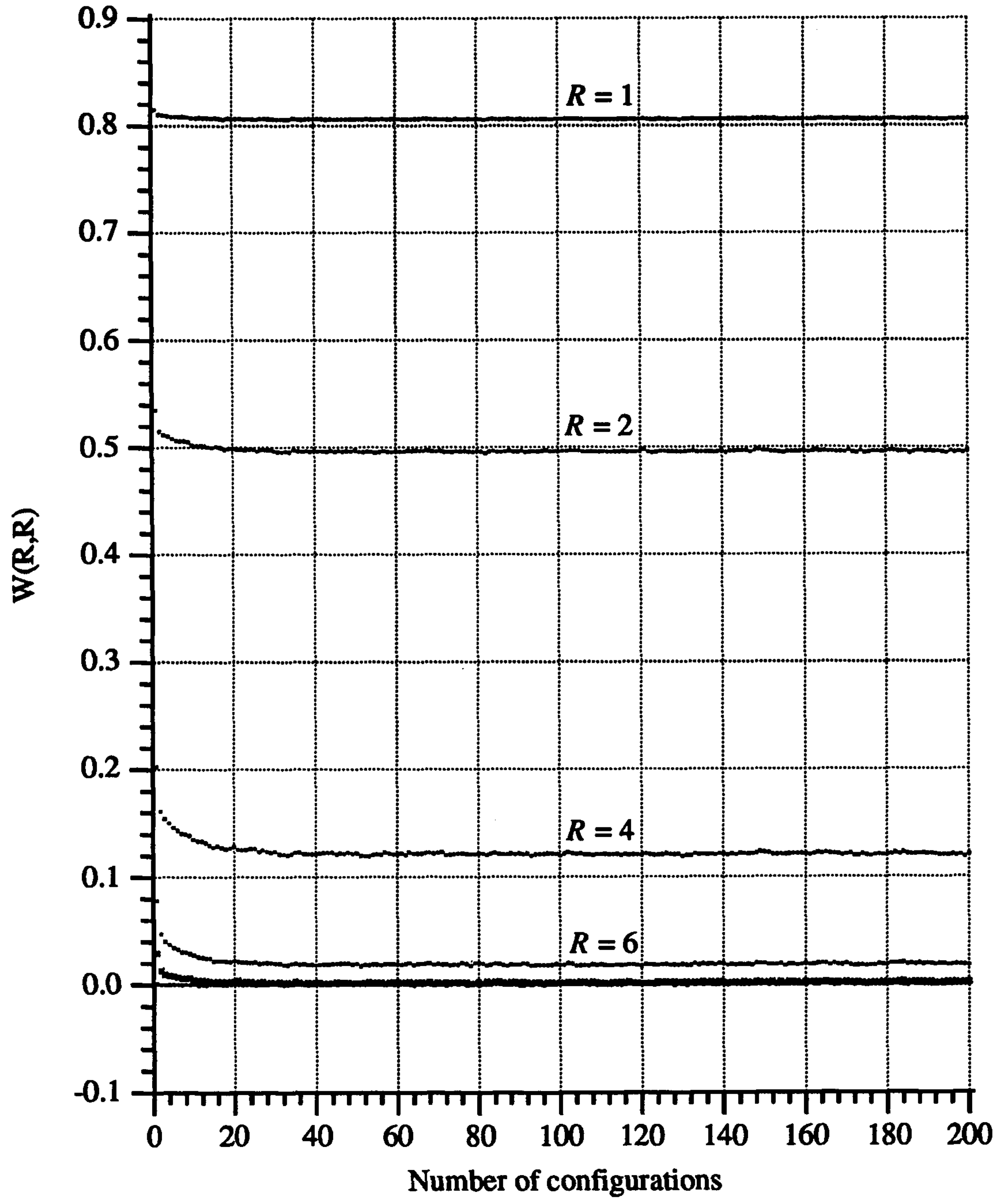
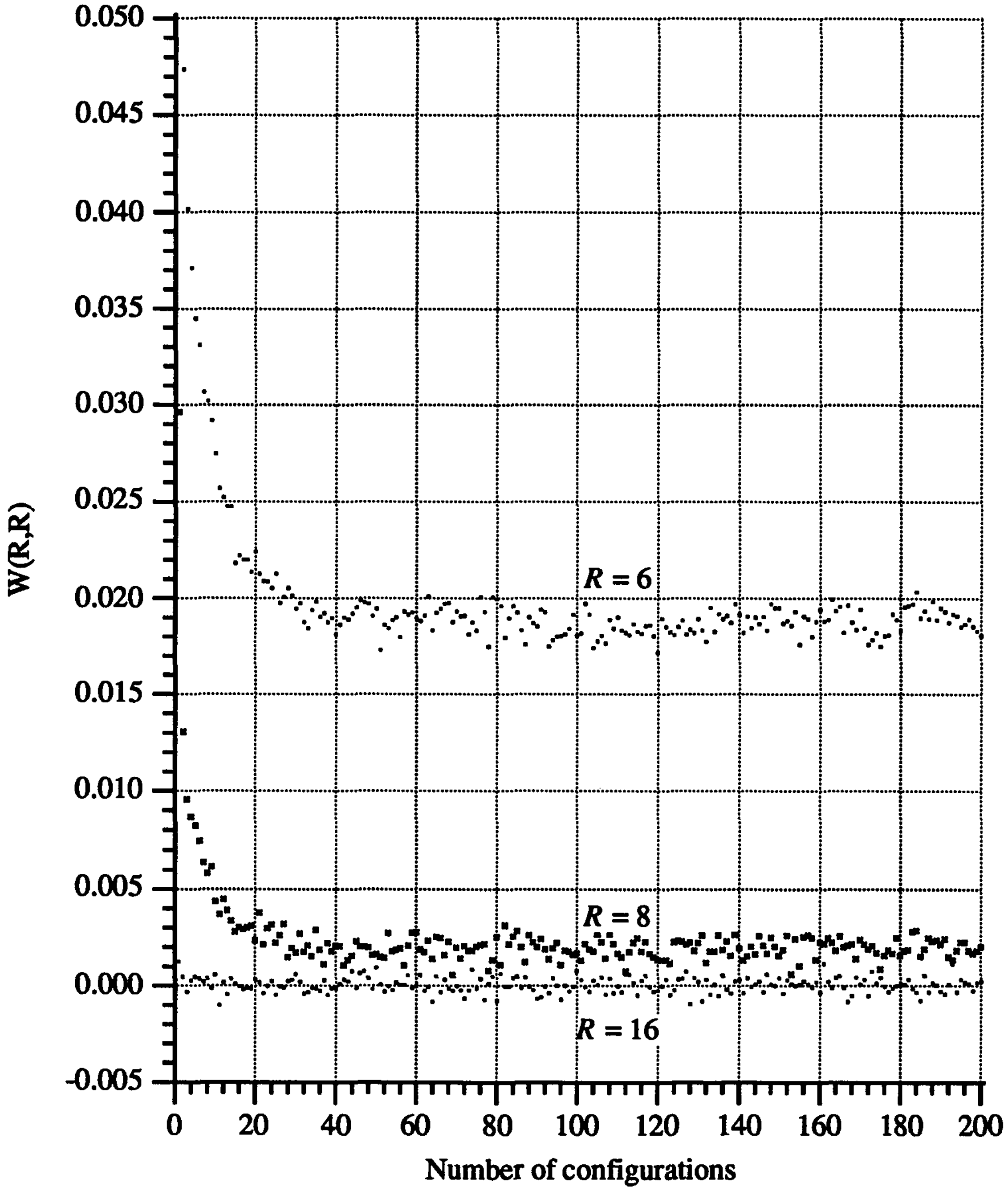


Figure 4. Wilson loop thermalization on expanded scale



Data Analysis

As explained earlier, string tension measurements should really use Wilson loops with $T = N \Delta \tau$, but that increases computation time because large loops have small values that are easily swamped by noise. Plotting $\ln(W(R,T))$ from (IP) versus T for fixed R, one finds that for R>3 the data is superlatively linear, allowing one to write

$$\ln(W(R,T)) = -\mathrm{V}(R)\, a\, T + b$$

Inserting the expected form of V(R),

$$\mathrm{V}(R) = d\, a\, R + e - f\,/\,(a\, R\,) + h \ln R,$$

symmetry in T and R demands three additional terms,

$$-\, e\, a\, R + f R\,/\,T - h\, a\, R \ln T.$$

Specializing to square loops,

$$\ln(W(R,R)) = -d a^2 R^2 - 2e\, a\, R + 2f + b\; - 2\, h\, a\, R \ln R,$$

so the string tension d and the coefficient of the Coulomb force law h can both be obtained from a single fit to data from square Wilson loops. (IP) used two fits, one to determine V(R), the second to obtain d, with the claim the use of additional data from non-square loops increases accuracy. This is not entirely true because the time spent calculating the non-square loops could have been applied to square loops. The use of non-square loops would allow the measurement of f, which is contaminated by b in the case of square loops. Lücher[28] has predicted

$$f = \pi\,(D\text{-}2)\,/\,24$$

due to string vibrations, which has been verified to the 40% level by (IP).

The square Wilson loops shown in table 18 were measured on a 64x64x256 lattice at $\sigma = 2.0, 2.2,$ and 2.4. A thermalization of 5000 sweeps was used, and the configuration separation and number of configurations is shown in table 19. The configuration separation is larger than the value calculated in the section on parameters because the larger lattice was afflicted with the higher correlations shown in table 20. Block averaging, recommended by (AHO), was not effective in removing correlations. These loops agree with measurements made by (IP).

Table 18. Square Wilson loops as a function of size.

Size	$\sigma = 2.0$	$\sigma = 2.2$	$\sigma = 2.4$
1	.805959 ± .000020	.829459 ± .000015	.846787 ± .000010
2	.496166 ± .000064	.551374 ± .000047	.591892 ± .000033
3	.260946 ± .000096	.326443 ± .000076	.376711 ± .000058
4	.121183 ± .000103	.177858 ± .000087	.225017 ± .000076
5	.050303 ± .000082	.090234 ± .000087	.127786 ± .000083
6	.018759 ± .000065	.042870 ± .000070	.069359 ± .000079
7	.006237 ± .000055	.019143 ± .000057	.036186 ± .000068
8	.001940 ± .000044	.008087 ± .000049	.018211 ± .000052
9	.000532 ± .000039	.003184 ± .000045	.008804 ± .000048
10	.000031 ± .000034	.001166 ± .000034	.004127 ± .000039
11	.000049 ± .000035	.000425 ± .000036	.001807 ± .000034
12	.000033 ± .000036	.000152 ± .000036	.000806 ± .000031
13	.000025 ± .000036	.000028 ± .000038	.000364 ± .000031
14	-.000006 ± .000037	.000015 ± .000034	.000128 ± .000028
15	.000014 ± .000036	-.000032 ± .000035	.000027 ± .000030
16	.000039 ± .000041	-.000064 ± .000031	-.000073 ± .000031

Figure 5. Logarithm of square Wilson loops versus separation.

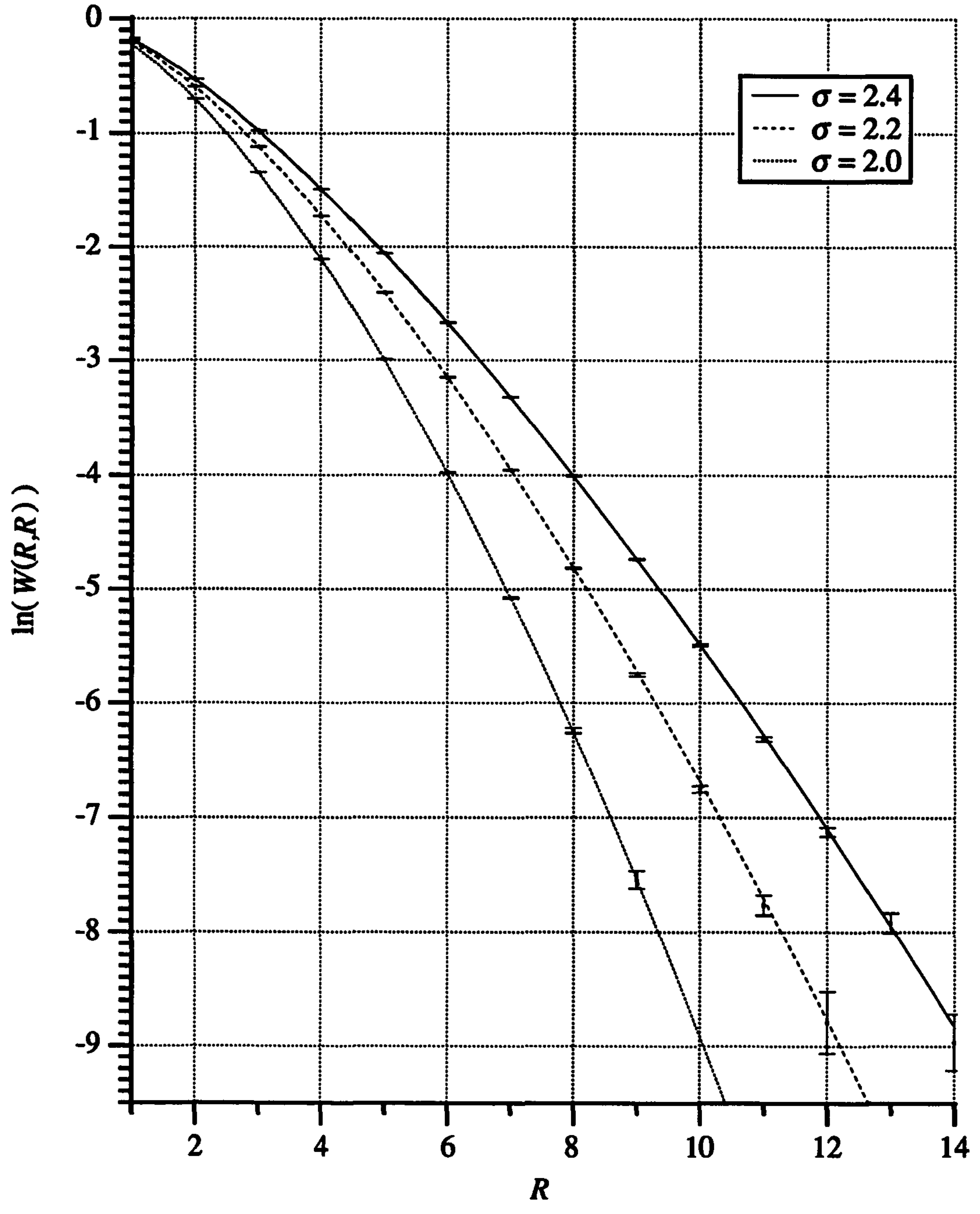


Table 19. Number of sweeps between configurations and number of configurations versus σ.

σ	Sweeps/Configuration	Configurations	Loops
2.0	200	125	3-9
2.2	175	143	4-12
2.4	125	200	5-14

Table 20. Unexpected correlations between configurations separated by 25 sweeps, for a $\sigma = 2.0$ 64x64x256 lattice.

Δt	1x1	2x2	3x3	4x4
1	.200 ± .023	.299 ± .036	.353 ± .041	.396 ± .024
2	.074 ± .026	.158 ± .031	.230 ± .026	.274 ± .033
3	.062 ± .020	.172 ± .025	.198 ± .035	.185 ± .036
4	.055 ± .018	.096 ± .034	.120 ± .044	.155 ± .042
5	.016 ± .023	.070 ± .032	.152 ± .036	.156 ± .033
6	.015 ± .033	.048 ± .032	.106 ± .043	.145 ± .038
7	.032 ± .033	.071 ± .043	.075 ± .051	.120 ± .055
8	-.012 ± .032	.034 ± .035	.094 ± .046	.103 ± .056
9	.056 ± .029	.073 ± .029	.107 ± .032	.114 ± .035
10	-.005 ± .034	.034 ± .026	.082 ± .026	.119 ± .030

Δt	5x5	6x6	7x7	8x8
1	.373 ± .028	.212 ± .031	.138 ± .034	.097 ± .028
2	.261 ± .039	.140 ± .026	.060 ± .035	.007 ± .023
3	.177 ± .044	.110 ± .029	.046 ± .033	.012 ± .035
4	.165 ± .037	.091 ± .034	.058 ± .028	.013 ± .029
5	.152 ± .031	.097 ± .034	.070 ± .032	-.021 ± .018
6	.150 ± .035	.086 ± .032	.066 ± .027	.007 ± .035
7	.119 ± .041	.088 ± .036	.011 ± .020	.039 ± .023
8	.101 ± .047	.031 ± .049	.015 ± .031	.037 ± .026
9	.067 ± .043	.049 ± .038	-.006 ± .026	.029 ± .033
10	.055 ± .040	.058 ± .037	.029 ± .028	-.004 ± .032

For the sake of comparison with published values, this data was fit with h set to zero, without singular value decomposition, with the results shown in table 21. Due to incompleteness of G(T) it is necessary to omit loops smaller than some cutoff. The smallest cutoff consistent with a Q>.1 was chosen. Q (PFT) is the probability of seeing a worse χ^2, presuming the model is correct and errors on data are normally distributed.

Table 21. Measured U(1) string tensions

Reference	σ **= 2.0**	σ **= 2.2**	σ **= 2.4**
(SG)	.110±.006	.055±.003	
(PS)	.098±.002	.050±.002	
(AHO)	.048±.002	.022±.001	
(IP)	.054±.003	.032±.003	
(WS)	.049±.003	.029±.002	.017±.001
This work	.0549±.0005	.0315±.0004	.0190±.0004

There appears to be no good reason why previous measurements have omitted the Coulomb term. (IP) indicate that their string tension measurements are independent of the lower cutoff on R, but this is not a good test: It would be far better to prove the Coulomb term is unimportant by including it in the fit, and showing that its coefficient is consistent with zero. Accordingly the loops in table 18 were fit with h as a parameter, with and without singular value decomposition. The results in tables 22 and 23 can be compared with tables 24 and 25, a reanalysis, by the same method, of (IP) data.

Table 22. String tension and Coulomb coefficient measured without SVD

σ	Loop Sizes	Q	$d\,a^2$	$2\,h\,a$
2.0	1-9	.062	.0352±.0007	.165±.003
2.0	2-9	.755	.0413±.0022	.122±.015
2.2	2-12	.913	.0205±.0008	.123±.006
2.4	2-14	.503	.0092±.0004	.131±.003

Table 23. String tension and Coulomb coefficient measured with SVD

σ	Loop Sizes	Q	$d\,a^2$	$2\,h\,a$
2.0	2-9	.077	.0358±.0004	.160 ±.001
2.0	3-9	.429	.0374±.0010	.157±.003
2.2	3-12	.291	.0174±.0004	.151±.001
2.4	3-14	.041	.0077±.0002	.146±.001
2.4	4-14	.438	.0085±.0004	.142±.002

Table 24. Recalculation of string tension and Coulomb coefficient from (IP) data, without SVD

σ	Loop Sizes	$d\,a^2$	$2\,h\,a$
2.0	2-10	.046± .008	.08 ± .06
2.2	2-12	.021± .005	.13 ± .03

Table 25. Recalculation of string tension and Coulomb coefficient from (IP) data, with SVD

σ	Loop Sizes	$d\,a^2$	$2\,h\,a$
2.0	2-10	.036±.002	.160±.004
2.2	2-12	.0187±.0009	.152±.003

To compare these experimental values of h with theory, the constants ξ = .3274±.0089 and η = .2926±.0067 were extracted by least square fit from the data in table 1, with 3% error assumed for all loops except for 50x50, which was assigned 10% error. Together with a $1/3!\sigma^3$ uncertainty due to W_3, these give the values of h shown in table 26.

Table 26. U(1) Coulomb constant calculated from (MR)'s weak coupling prediction.

σ	$2ea$	$2ha$
2.0	.1910±.0086	.1707±.0073
2.2	.1714±.0069	.1531±.0058
2.4	.1554±.0058	.1388±.0048

CHAPTER IV. CONCLUSIONS

The U(1) String Tension

It is clear that the Coulomb constant h is nonzero, that experimental SVD measurements of h are in agreement with theory, and that previous string tension measurements have been off by as much as 85% because they have neglected this term. Curve fitting is a subtle art however, and the exact values of the string tension measured here depend upon your choice of Q and whether you believe in singular value decomposition. According to (PFT) a Q as small as .001 may be acceptable if the errors are nonnormally distributed, as they are here. However, since every effort has been made to overestimate errors, it would seem safer to demand a high Q. In the case of table 25, SVD reduced errors by a factor of four, producing results consistent with non-SVD results from slightly better data, in table 22. Furthermore, the Coulomb constants in table 23, calculated by SVD, are in better agreement with theory than the non-SVD results in table 22. It therefore seems reasonable to accept the high Q SVD values shown in table 27.

Table 27. Final values for the U(1) string tension and Coulomb constant.

σ	$d\,a^2$	$2\,h\,a$
2.0	.0374±.0010	.157±.003
2.2	.0174±.0004	.151±.001
2.4	.0085±.0004	.142±.002

Figure 5 contains a plot of the experimental values from table 27 superimposed on analytical results. The modified weak coupling curve is an attempt (WS) to compensate for the use of the Villain action by replacing σ with an effective σ. This is done by equating the ratio of the zeroth and first Fourier coefficients of the Wilson action with the same ratio for the Villain action, which gives

$$\sigma_{\text{Villain}} = \frac{1}{2\ln\left[\frac{I_0(\sigma_{\text{Wilson}})}{I_1(\sigma_{\text{Wilson}})}\right]}.$$

Figures 7-9, based on the data in table 27, show Coulomb and string tension contributions to the U(1) potential. The two terms added to the Wilson loop data are corrections for the use of square loops. For σ = 2.4, for example, the Coulomb contribution is as large as the string tension's contribution for all loops smaller than 28x28.

Although it is heartening that the Coulomb constant measured in this experiment agrees with values obtained by numerical integration of expressions derived analytically by (MR), the correction this term provides has driven the string tension even further from values calculated by the dual method (SG).

Although SPAM was constructed for a single calculation, a few simple modifications would render it useful for other purposes. Indeed, it could readily be converted into a simulator of cellular automata, which can be shown[29] to be capable of performing (with negligible efficiency) any desired computation. The next few sections will detail a few additional reasonably efficient calculations for which SPAM could be modified.

Figure 6. Analytical and numerical string tensions versus sigma.

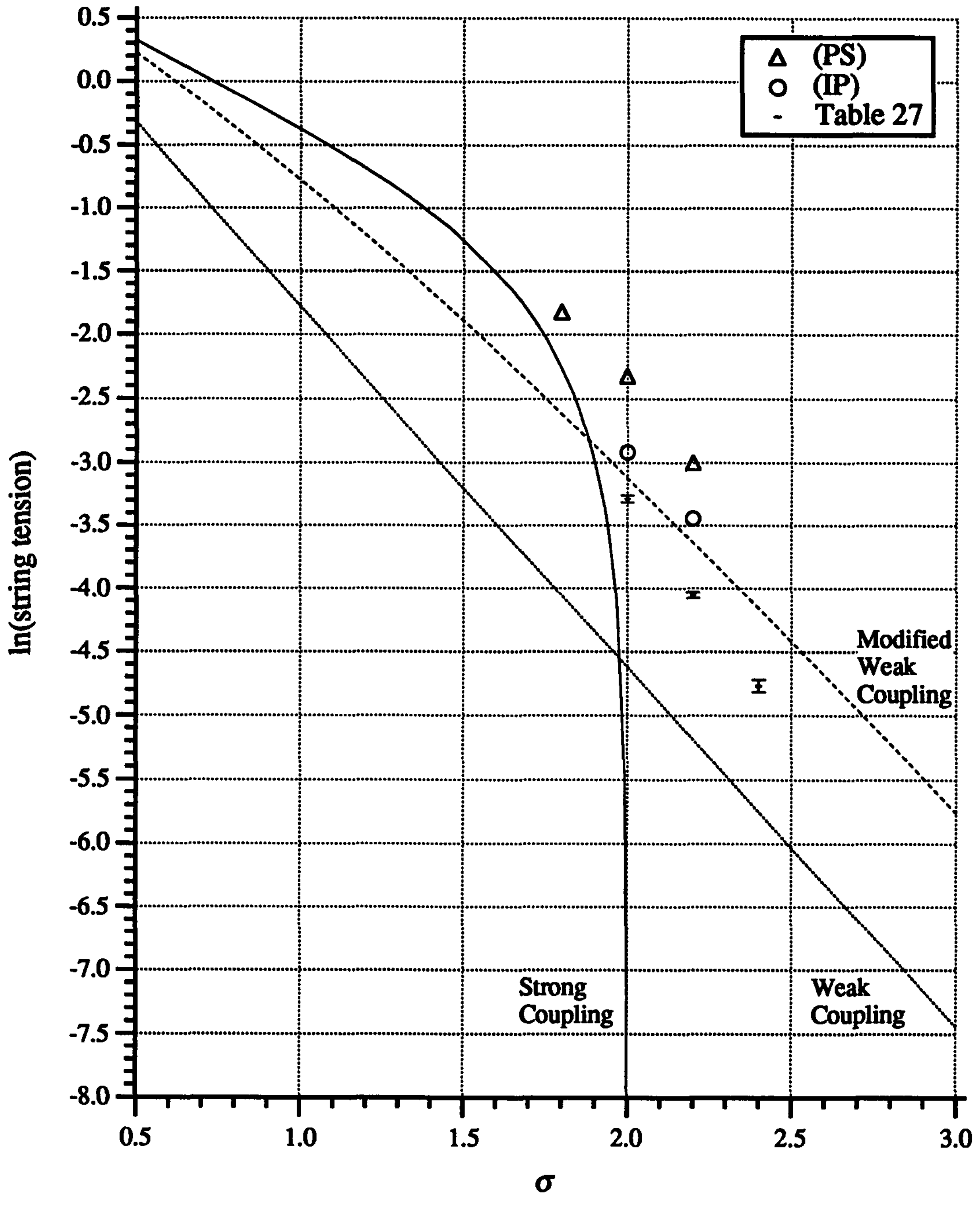


Figure 7. String tension and Coulomb contributions to the U(1) potential at $\sigma = 2.0$

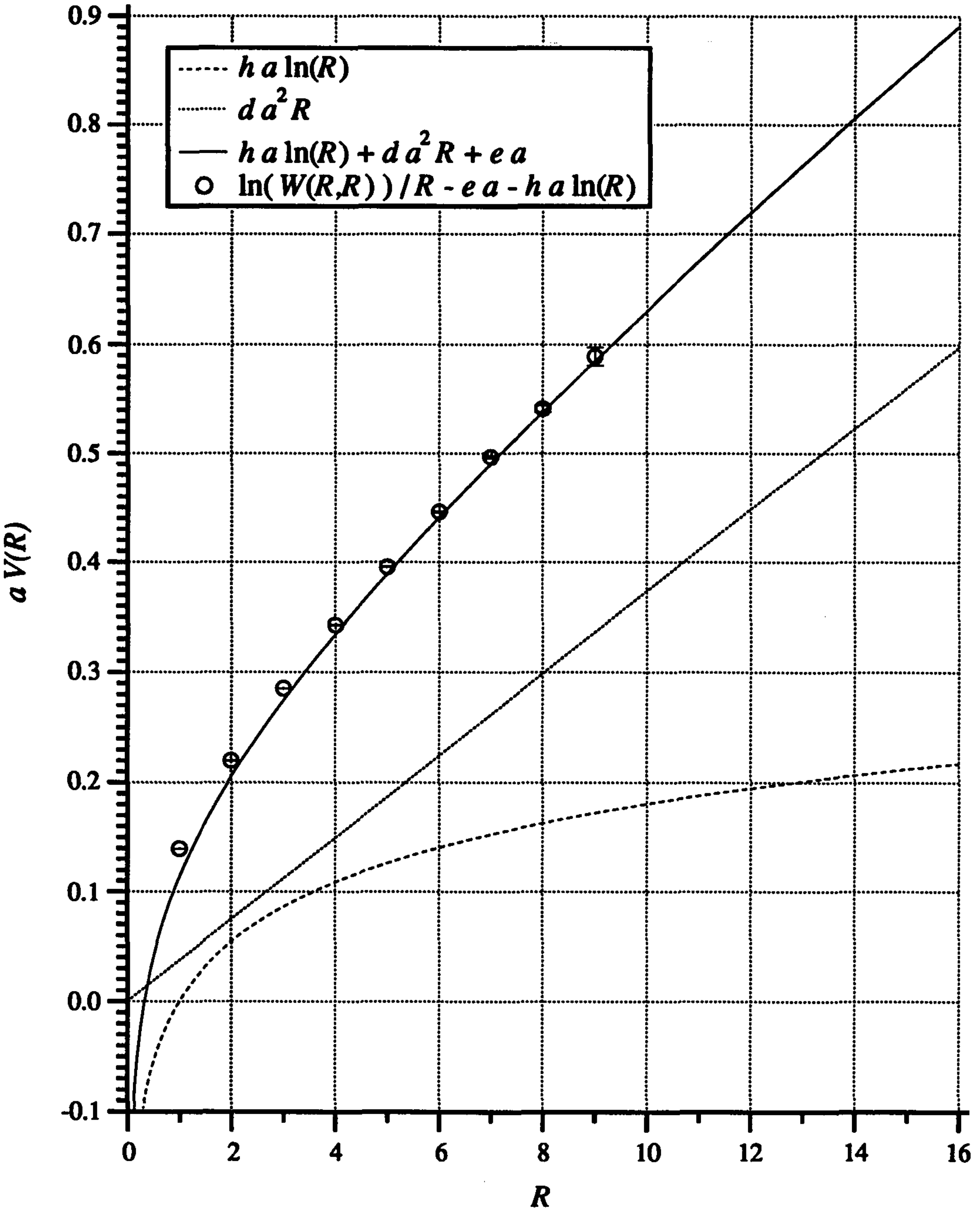


Figure 8. String tension and Coulomb contributions to the U(1) potential at $\sigma = 2.2$

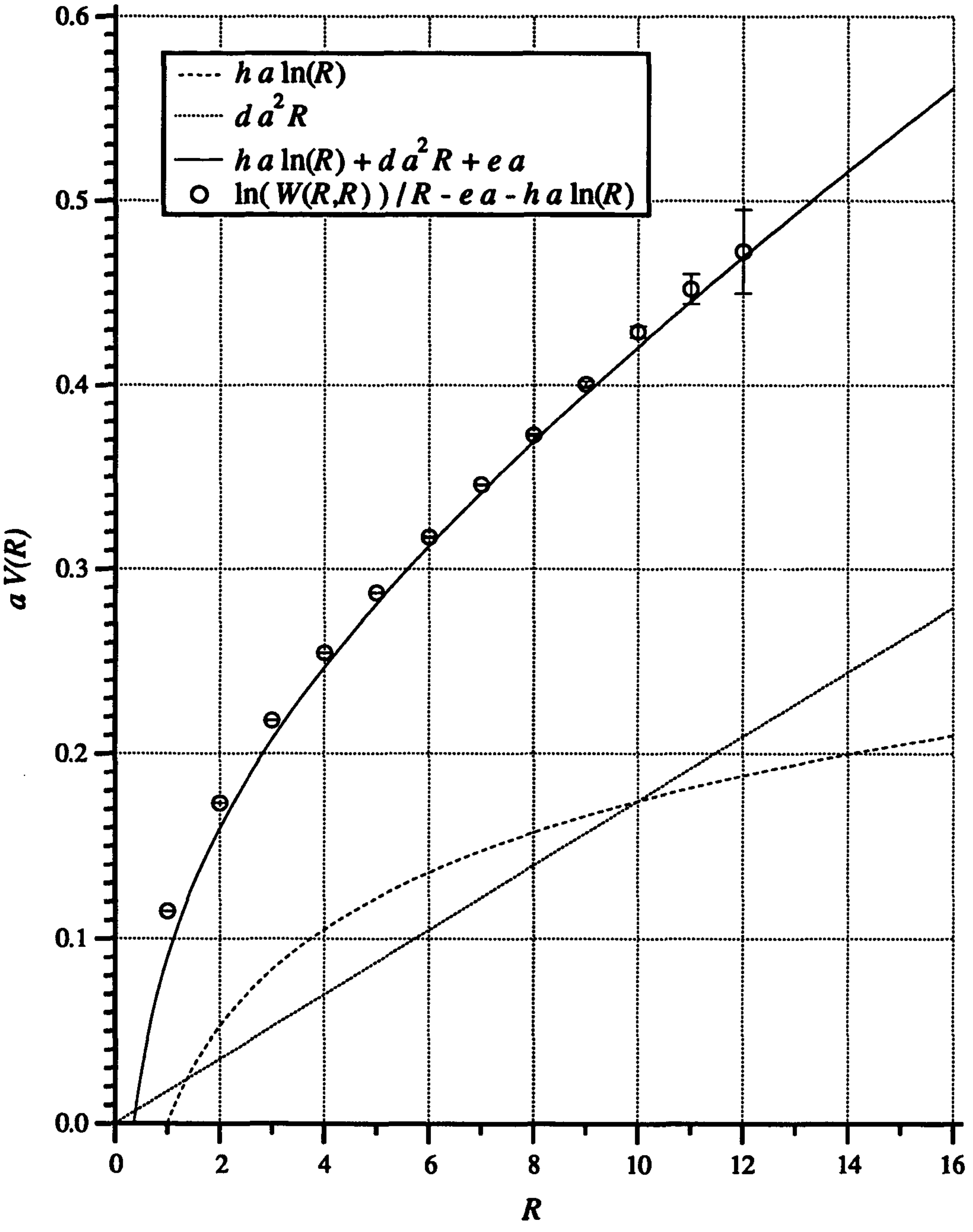
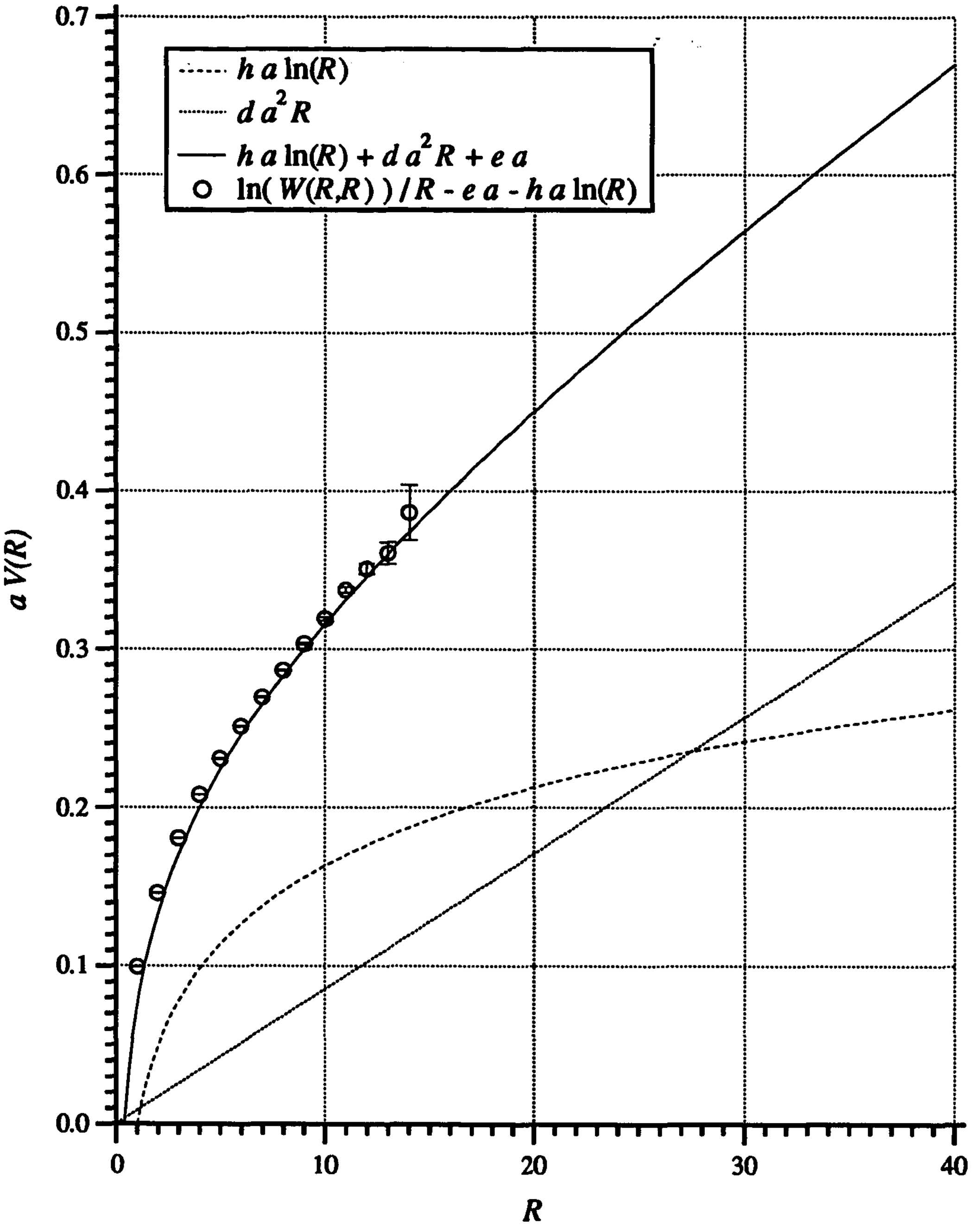


Figure 9. String tension and Coulomb contributions to the U(1) potential at $\sigma = 2.4$



Brains of SPAM

Neural networks[30] are crude attempts to simulate the brains of animals. Although not entirely correct, a model of the brain satisfactory for many purposes expresses the output of neuron *i* at time *k* as

$$f_i(t_k) = \mathrm{S}\left(\sum_{j=1}^{N} f_j(t_{k-1})\, g_{ij} + I_i\right)$$

where S is a nonlinear function similar to

$$1/\left(1-e^{-\alpha x}\right) \quad \text{or} \quad \tanh(\alpha x),$$

I_i is a sensory input to neuron *i* (most of which are zero) and g_{ij} are the strengths of couplings between neurons. A neural network does not have to be stable. It has been shown that symmetric couplings $g_{ij} = g_{ji}$ with $g_{ii} = 0$ result in stable systems. The f_i can be considered the components of a vector in an *N* ($N \approx 10^{11}$ for humans) dimensional space. If g_{ij} is regarded as the j^{th} component of $\vec{g}_i$ and if α is large enough so that S can be regarded as a step function, then neuron *i* produces an output which indicates whether f and $\vec{g}_i$ are on the same side of the hyperplane perpendicular to $\vec{g}_i$. Pattern recognition can be obtained by selecting $\vec{g}_i$ to define hyperplanes which surround some part of the input pattern space: learning consists of adjustments of the $\vec{g}_i$. In a model proposed by D. Hebb, couplings are increased when they are used:

$$g_{ij}(t_k) = g_{ij}(t_{k-1}) + \gamma f_i\, f_j$$

Hebbian learning does work, but it converges slowly, so in practice the $\vec{g}_i$ are chosen by optimization techniques. Boltzmann training is the application of simulated annealing (PFT) to the selection of optimum weights. Typically an input pattern $\vec{I}$ is presented and the network is updated until it stabilizes. $E \equiv \left|\vec{f} - \vec{T}\right|^2$ where $\vec{T}$ is the desired output plays the role of the Hamiltonian in simulated annealing. E is calculated, a weight is altered, the network is allowed to restabilize, E is recalculated, and the new weight is accepted or rejected according to the Metropolis algorithm. After all the weights have been optimized the process is repeated with a new $\vec{I},\vec{T}$ pair. Unfortunately the change in f_i produced by a single weight is not necessarily localized, implying SPAM's built in Metropolis algorithm could not be used: the number of microstates required to evaluate ΔE exceeds the maximum number of microstates per opcode. This is not a problem because SPAM can save a lot of time just by updating the network. Tap weights would be represented by bytes stored in a square matrix with neural outputs occupying the diagonals, the group multiplier would perform algebraic multiplication, the squashing function S would reside in SPAM's e^x lookup table, and the only essential hardware modification would be a buffer between the S output and the data bus, to allow the output values of the neurons to be saved. The maximum of 512 microstates per opcode, with four microstates devoted to emptying the pipeline, and two microstates per input, would limit the number of neurons to 254. The use of six bits to hold address offsets would reduce this to 64. Fortunately these limitations can be changed with a few day's labor, to yield a memory limit of 1024 neurons.

Molecular Configurations On SPAM

With a few hardware changes SPAM could find energy minima of liquid crystals or clusters of atoms. First one has to express the Hamiltonian in a form suitable for incorporation into a few lookup tables. For a system of interacting spheres one could construct a three dimensional NxMx3 matrix q_{ijk} where i represents time, j specifies the particle, and k designates a component of the position vector. Each coordinate is represented by a single byte, resulting in a 256^3 lattice containing M particles. The group multiplier would be reprogrammed to return, for inputs A and B, $(A-B)^2 \cdot m / 2\Delta t$. In addition to providing the translational kinetic energy, this would be used to calculate the square of the distance between particles which would be fed back to a potential lookup table through an extra latch. Liquid crystal simulations require the additional term

$$\sum_j \sum_k \vec{S}_{ik} \bullet \vec{S}_{ij} \; f\left(\left|\vec{q}_{ik} - \vec{q}_{ij}\right|^2\right).$$

If the orientation of each molecule is stored in a byte the $\vec{S} \bullet \vec{S}$ lookup is easy, and the product could be stored in the partial product latch.

A simpler approximation may prove fruitful in such calculations. As long as the particles do not move very far, each one could be allowed to occupy only a small region near its mean position. The mean positions from a lattice upon which a sublattice for each particle is located. The sublattices can be made large enough to overlap completely with neighboring sublattices. The advantages of this are that:

- Only neighboring particles interact, vastly simplifying the potential calculation.

- Since the particles move over a smaller area, the binary number that describes their position can use fewer bits. If the sublattice contained less than 256 positions, the potential and kinetic energies could each be obtained by a single lookup.
- In the case of liquid crystals, if orientational information were included with the position information, the potential could include orientation dependent steric effects and rotational kinetic energy. Unfortunately, 8 bits are inadequate for this, although a crude $\vec{S} \bullet \vec{S}$ term could be included in two dimensional calculations.

APPENDIX

Summary of Statistics[31]

$\bar{x} \equiv \frac{1}{N}\sum_{n=1}^{N} x_n$	Arithmetic mean
$\Delta x_i \equiv x_i - \bar{x}$	Deviation of x_i
$\sigma_N{}^2(x) \equiv \overline{(\Delta x)^2} = \overline{x^2} - \bar{x}^2$	Mean square deviation of x
$\sigma^2(x) \equiv \lim_{N\to\infty} \sigma_N{}^2(x)$	Variance of x
$\sigma_N(x) \equiv \sqrt{\sigma_N{}^2(x)}$	RMS deviation of x
$\sigma(x) \equiv \sqrt{\sigma^2(x)}$	Standard deviation of x
$\varepsilon(x) \equiv \sigma_N(x)/\sqrt{N-1}$	Adjusted standard error of $\bar{x}$
$E(\Omega)$	Error on Ω
$S_N(x) \equiv \sigma_N(x)\sqrt{N/(N-1)}$	Adjusted RMS deviation

Even the best measurement of some quantity x is bound to be contaminated by some error, which can be reduced by averaging multiple measurements. Although other quantities, such as the geometric mean, or the most probable value of a measurement, or the value for which half of the measurements will be smaller, may sometimes be considered, the arithmetic mean is normally regarded as the best estimate of the true value. The best estimate of the error on the mean is the adjusted standard error. The error on a single measurement (the precision of the measuring apparatus) is the standard deviation, the best estimate of which (for a finite amount of data) is the adjusted RMS deviation. *E(Ω)* will be used to denote the error on **Ω**: the adjusted RMS deviation for a single measurement, the adjusted standard error for the average of many uncorrelated measurements, or a more complicated quantity for averages of correlated measurements.

Given two sets of data *x* and *y*, the linear correlation coefficient (PFT) between them is defined by

$$r(x,y) \equiv \frac{\overline{\Delta x \cdot \Delta y}}{\sigma_N(x)\cdot\sigma_N(y)} = \frac{\overline{xy} - \bar{x}\,\bar{y}}{\sigma_N(x)\cdot\sigma_N(y)}$$

which is limited to -1 $\leq r \leq$ +1. If r = 0 then the data are not linearly related. Multiple data sets will show fluctuations in *r*, so one must select a threshold beneath which the correlation is insignificant, and another threshold above which the data is to be deemed linearly related. One possibility is to demand that the error on the slope must be small enough that the slope is distinguishable from zero if x and y are to be considered linearly related. If $y = ax+b$ it can be shown that

$$a = \overline{\Delta x \cdot \Delta y}/\sigma_N{}^2(x) = r\sigma_N{}^2(y))/\sigma_N{}^2(x)$$

$$\varepsilon(a) = \frac{\sigma_N{}^2(y)}{\sigma_N{}^2(x)}\sqrt{\frac{1-r^2}{N-2}}$$

so

$$\frac{\varepsilon(a)}{a} = \sqrt{\frac{1-r^2}{(N-2)r^2}} \quad \lesseqgtr \quad f$$

results in a threshold of $|r| \quad \gtrless \quad 1/\sqrt{f^2(N-2)+1}$

Error Estimates on Correlated Measurements

Correlated data does not change $\bar{x}$ or $\sigma_N(x)$ as one can see by using each point twice. The problem is in $\varepsilon(x)$ which would be incorrectly reduced by about $1/\sqrt{2}$ due to the worthless doubled terms. To estimate the correct error (AHO), construct N block averages, each containing M consecutive values of x:

$$z_i = \tfrac{1}{M}\sum_{j=(i-1)M+1}^{i\cdot M} x_j = \tfrac{1}{M}\sum_{k=1}^{M} x_{i,k} \qquad x_{i,k} = x_{(i-1)M+k}\,.$$

M is chosen to be greater than the correlation length, so that the Z_i are uncorrelated. Then

$$\sigma_N{}^2(z) = \tfrac{1}{N}\left[\sum_{i=1}^{N}(z_i)^2\right] - \left[\tfrac{1}{N}\sum_{i=1}^{N} z_i\right]^2$$

$$= \tfrac{1}{NM^2}\sum_{i=1}^{N}\left[\sum_{k=1}^{M} x_{i,k}{}^2 + 2\sum_{k=1}^{M-1}\sum_{l=k+1}^{M} x_{i,l}x_{i,k}\right] - \left[\tfrac{1}{NM}\sum_{i=1}^{N}\sum_{k=1}^{M} x_{i,k}\right]^2$$

Now exchange the order of summation and denote averages over i by angle brackets:

$$\langle x_l\rangle \equiv \tfrac{1}{N}\sum_{i=1}^{N} x_{i,l}$$

$$\sigma_N{}^2(z) = \tfrac{1}{M^2}\left[\sum_{k=1}^{M}\langle x_k{}^2\rangle + 2\sum_{k=1}^{M-1}\sum_{l=k+1}^{M}\langle x_l x_k\rangle - \left(\sum_{k=1}^{M}\langle x_k\rangle\right)^2\right].$$

Expanding the last term:

$$\sigma_N{}^2(z) = \tfrac{1}{M^2}\left\{\sum_{k=1}^{M}\left[\langle x_k{}^2\rangle - \langle x_k\rangle^2\right] + \sum_{k=1}^{M-1}\sum_{l=k+1}^{M}\left[\langle x_l x_k\rangle - \langle x_k\rangle\langle x_l\rangle\right]\right\}$$

$$= \tfrac{1}{M^2}\left\{\sum_{k=1}^{M}\sigma_N{}^2(x_k) + 2\sum_{k=1}^{M-1}\sum_{l=k+1}^{M}\left[r(x_k x_l)\sigma_N{}^2(x_k)\sigma_N{}^2(x_l)\right]\right\}$$

All the $\sigma_N{}^2(x_k)$ should be equal. Denoting them by σ and also noticing that $r(x_k x_l)$ should depend only upon l-k,

$$\sigma_N{}^2(z) = \tfrac{1}{M^2}\sigma\left[M + 2\sum_{p=1}^{M-1}(M-p)r(p)\right].$$

presuming $r(1) >> r(\neq 1)$ and $m >> 1$,

$$\sigma_N{}^2(z) = \tfrac{1}{M^2}\sigma[1 + 2r(1)]$$

We do not really want to measure $\sigma_N{}^2(z)$ directly from fluctuations of block averages: we want a correlation correction to apply to a single block, the entire data set of M points. If our entire data set is crammed into $i=1$ then we have only one datum with which to calculate $\sigma = \sigma_N{}^2(x_k)$. Since we have presumed the correlations are small, however $\sigma_N{}^2(x_k)$ calculated by using every H^{th} data point should be a satisfactory substitute. Even $H=1$ should be adequate, resulting in

$$E(\bar{x}) = \sigma(z) \cong \varepsilon(\bar{x})\sqrt{1 + 2r(1)}.$$

Least Square Fits by Singular Value Decomposition

A linear least square fit (PFT) is the minimization of $\chi^2 \equiv \left|\vec{\mathbf{b}} - \mathbf{A}\cdot\vec{\mathbf{a}}\right|^2$

where

$\mathbf{x}_i$	are the values of the independent variable
$\mathbf{y}_i$	are the values of the dependent variable
$\mathbf{E}_i \equiv E(\mathbf{y}_i)$	are the errors on $\mathbf{y}_i$
$\mathbf{f}_i$	are the basis functions
$\mathbf{A}_{ij} = \left[\mathbf{f}_j(\mathbf{x}_i)\right]/\mathbf{E}_i$	is called the design matrix
$\mathbf{b}_i \equiv \mathbf{y}_i/\mathbf{E}_i$	and
$\mathbf{a}_i$	consists of the sought coefficients of $\mathbf{f}_i$.

Normally A is singular so no solution$\vec{\mathbf{a}} = \mathbf{A}^{-1}\,\vec{\mathbf{b}}$ with χ=0 exists. To minimize χ^2, equate to zero derivatives with respect to $\mathbf{a}_k$:

$$\frac{\partial\chi^2}{\partial\mathbf{a}_k} = 0 = 2\left[\mathbf{A}^{\mathrm{T}}(\vec{\mathbf{b}} - \mathbf{A}\vec{\mathbf{a}})\right]_k$$

$$\vec{0} = \mathbf{A}^{\mathrm{T}}\,\vec{\mathbf{b}} - \mathbf{A}^{\mathrm{T}}\mathbf{A}\vec{\mathbf{a}}$$

If the data and model are good, $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ is not singular and

$$\vec{\mathbf{a}} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\,\vec{\mathbf{b}}$$

(known as the normal equations) solves the problem. If, however, your model is insensitive to some combination of basis functions, $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ will be nearly singular and you will wind up with meaningless, often huge, coefficients which cancel mercurially to yield a reasonable χ^2. That is when you need singular value decomposition (PFT), which enables you to excise the singularities and almost-singularities of A^{-1}, thereby removing linear combinations of basis functions to which the fit is insensitive.

Any matrix A having its number of rows greater than or equal to its number of columns can be "diagonalized" as the product of three other matrices

$$A = U\, W\, V^T$$

where $U^T U = 1,\quad V V^T = 1,\quad V^T V = 1,$ and W is diagonal with positive elements. This decomposition is almost unique, and furthermore (denoting the j^{th} column of V by $\vec{V}_j$) the $\vec{V}_j$ with $W_{jj} \neq 0$ span the nullspace of A, and the $\vec{U}_j$ with $W_{jj} = 0$ span the range of A. The inverse of A is then readily constructed:

$$A^{-1} = V\, W^{-1}\, U^T$$

If A is singular one or more of the elements of W^{-1} will be infinite, in which case an approximation $\underline{A}^{-1}$ to A^{-1} is constructed by replacing the infinite elements of W^{-1} by zero: each infinite element of W^{-1} corresponds to a dimension of the nullspace which ought to be ignored.

If $\vec{b}$ is within the range of A, the solution to $\vec{b} = A\vec{a}$ will not be unique since any element $\vec{n}$ in the nullspace of A can be added to $\vec{a}$ without changing $\vec{b}$. $\underline{\vec{a}} \equiv \underline{A}^{-1}\vec{b}$ provides the solution with minimum magnitude:

$$\underline{\vec{a}} = \sum_j \vec{V}_j \left[\underline{W}^{-1}{}_{jj} \left(\vec{U}_j\right)^T \vec{b} \right],$$

must be orthogonal to $\vec{n}$ because each $\vec{V}_j$ nullspace basis vector is multiplied by zero thanks to the definition of $\underline{W}^{-1}$. Since $\underline{\vec{a}}$ and $\vec{n}$ are orthogonal, $|\underline{\vec{a}} + \vec{n}| \geq |\underline{\vec{a}}|$.

If $\vec{b}$ is not within the range of A, then $\underline{\vec{a}} \equiv \underline{A}^{-1}\vec{b}$ will minimize $\left|\vec{b} - A\underline{\vec{a}}\right|$, exactly the property needed for minimization:

$$\begin{aligned}
\vec{b} - A\underline{\vec{a}} &= \vec{b} - \left(UWV^T\right)\left(V\underline{W}^{-1}U^T\right)\vec{b} \\
&= U\left(1 - W\underline{W}^{-1}\right)U^T\vec{b} \\
&= \sum_j \vec{U}_j\left[\left(1 - W\underline{W}^{-1}\right)_{jj}\vec{U}_j^T\vec{b}\right]
\end{aligned}$$

is orthogonal to

$$\begin{aligned}
A\delta\vec{a} &= UWV^T\delta\vec{a} \\
&= \sum_j \vec{U}_j\left\{W_{jj}\vec{V}_j^T\delta\vec{a}\right\}
\end{aligned}$$

because the construction of $\underline{W}^{-1}$ guarantees that, for a given j, the number in braces and the number in brackets can not both be nonzero.

Furthermore the $\vec{V}_j$ happen to be the principal axes of ellipsoids of constant $\delta\chi^2$ in coefficient space. The lengths of the axes for $\delta\chi^2 = 1$ are the corresponding elements of W^{-1} since

$$\delta\chi^2 = \left|WV^T\delta\vec{a}\right|^2 = \sum_j W_{jj}\vec{V}_j^T\vec{a} = \text{constant}$$

defines a multidimensional ellipse. By zeroing additional elements of W^{-1} the error ellipse can be squashed to one less dimension, removing the corresponding $\vec{V}_j$ from the fit. In the rare case of normally distributed errors, the probability that the correct set of coefficients falls within the ellipse is related to the value of $\delta\chi^2$ in the table in figure 14.5.4 in (PFT).

The probable errors in a_i and the correlations between them can be obtained from the covariance matrix, which is defined by

$$\begin{aligned} C_{ij} &\equiv \sum_{k=1}^{M}\left(E(y_k)\frac{\partial a_i}{\partial y_k}E(y_k)\frac{\partial a_j}{\partial y_k}\right) \\ &= \sum_{k=1}^{M} \partial[a_i]_k\, \partial[a_j]_k \\ &= M\, r(a_i,a_j)\, \sigma_M(a_i)\, \sigma_M(a_j) \end{aligned}$$

from which it can be seen that

$$E(a_i) = \sqrt{C_{ii}}$$
$$r(a_i,a_j) = C_{ij}\big/\sqrt{C_{ij}C_{jj}}\,.$$

C_{ij} may be evaluated by substituting $\vec{a} = \underline{A}^{-1}\vec{b}$ into the definition. The result, after some manipulation, is

$$C_{ij} = \sum_{p} V_{ip}\, V_{jp}\left(\frac{1}{W_{pp}}\right)^2.$$

To illustrate the utility of singular value decomposition, three fits with different tolerances for the elimination of instability were performed. The first fit (figure 9) shows the result provided by solution of the normal equations:

```
ChiSquared =   0.35
ChiSqrEstimate =   4.00 ±   2.83
ChiPerPoint =   0.21
Q =  9.860981e-1

Unstable?: FALSE     Tolerance:  1.000000e-5
NumberOfDataPoints:   8     NumberOfParameters:   4

F?:  Coefficient:   ProbableError:   %Error:  BasisFn:
F1    0.389429       0.441762         113.4    1
F2   -0.445701       0.301470          67.6    R
F3   -0.058625       0.020770          35.4    R^2
F4    0.040851       0.196899         482.0    RlnR
```

Because we have too many (or too similar) basis functions for data of this quality, the fit can not distinguish between basis functions. The fit is unstable, and meaningless coefficients with huge errors are found, which cancel when χ

is computed. The responsible linear combination of basis functions can be found by examining W, the singular value matrix, which in this case shows that error ellipse vector V3 is the culprit:

```
V?:  AxisLength:   SingularValues:   NormalizedSV:  BadVector?:
V1   1.380353e-4    7.244526e+3       1.000000       FALSE
V2   1.838596e-3    5.438933e+2       0.075076       FALSE
V3   5.701015e-1    1.754073e+0       0.000242       FALSE
V4   1.481340e-2    6.750642e+1       0.009318       FALSE
```

Axis V3 is composed mostly of F1 and F2 with an additional contribution by F4:

```
ErrorEllipseAxes:
V?:     F1        F2        F3        F4
V1   -0.0610   -0.2277   -0.9216   -0.3084
V2    0.4092    0.8059   -0.3209    0.2831
V3   -0.7748    0.5288    0.0360   -0.3447
V4   -0.4781   -0.1382   -0.2153    0.8402
```

The fitting routine replaces with infinity all the elements of W which, after normalization, are smaller than the tolerance. This removes the corresponding linear combination of basis functions from the fit. In the present case, when the tolerance exceeds 0.000242,

-.7748 + .5288*R + .0360*R*R - .3447*R*ln(R)

will be removed from the solution. This reduces the errors a lot, with an acceptable increase in χ:

```
ChiSquared =   1.30
ChiSqrEstimate =   4.00 ±   2.83
ChiPerPoint =   0.40
Q =  8.619693e-1

Unstable?:  TRUE    Tolerance:  1.000000e-3
NumberOfDataPoints:   8     NumberOfParameters:   4

F?:  Coefficient:   ProbableError:   %Error:  BasisFn:
F1   -0.039484        0.007122        18.0     1
F2   -0.152971        0.002527         1.7     R
F3   -0.038705        0.003246         8.4     R^2
F4   -0.149963        0.012457         8.3     RlnR
```

Increasing the tolerance further, linear combination V4 is excluded and an additional improvement in the error is observed, but this increases χ excessively.

```
ChiSquared =  28.78
ChiSqrEstimate =   4.00 ±   2.83
ChiPerPoint =   1.90
Q =  8.679712e-6

Unstable?:  TRUE    Tolerance:  1.000000e-2
NumberOfDataPoints:   8     NumberOfParameters:   4

F?:  Coefficient:   ProbableError:   %Error:  BasisFn:
F1   -0.076610        0.000752         1.0     1
F2   -0.163701        0.001482         0.9     R
F3   -0.055426        0.000604         1.1     R^2
F4   -0.084717        0.000522         0.6     RlnR
```

Figure 10. Example fit to (IP)'s data at σ = 2.0, with R $\geq$ 3.

```
ChiSquared =   0.35
ChiSqrEstimate =   4.00 ±   2.83
ChiPerPoint =   0.21
Q =  9.860981e-1

Unstable?: FALSE    Tolerance:  1.000000e-5
NumberOfDataPoints:   8     NumberOfParameters:   4

F?:   Coefficient:    ProbableError:   %Error:  BasisFn:
F1     0.389429         0.441762        113.4    1
F2    -0.445701         0.301470         67.6    R
F3    -0.058625         0.020770         35.4    R^2
F4     0.040851         0.196899        482.0    RlnR

ErrorEllipseAxes:
V?:     F1        F2        F3        F4
V1   -0.0610   -0.2277   -0.9216   -0.3084
V2    0.4092    0.8059   -0.3209    0.2831
V3   -0.7748    0.5288    0.0360   -0.3447
V4   -0.4781   -0.1382   -0.2153    0.8402

V?:   AxisLength:    SingularValues:    NormalizedSU:   BadVector?:
V1   1.380353e-4     7.244526e+3        1.000000        FALSE
V2   1.838596e-3     5.438933e+2        0.075076        FALSE
V3   5.701015e-1     1.754073e+0        0.000242        FALSE
V4   1.481340e-2     6.750642e+1        0.009318        FALSE

CovarianceMatrix:
 1.951538e-1
-1.331405e-1   9.088429e-2
-9.039102e-3   6.189883e-3   4.313743e-4
 8.670913e-2  -5.926282e-2  -4.071146e-3   3.876914e-2

CorrelationCoefficients:
  1.0000
 -0.9997    1.0000
 -0.9852    0.9886    1.0000
  0.9969   -0.9984   -0.9955    1.0000

Error(slope)/slope:
  0.0000
 -0.0097    0.0000
 -0.0711    0.0622    0.0000
  0.0324   -0.0233   -0.0388    0.0000

Significance:
 0.000000e+0
 5.623828e-11   0.000000e+0
 8.069009e-6   3.693655e-6   0.000000e+0
 7.736708e-8   1.066208e-8   2.252422e-7   0.000000e+0
```

More About Importance Sampling

To rigorously derive importance sampling Monte Carlo integration, start out with a uniform distribution and split the integration interval into N regions over which $P(x)$ is approximately constant:

$$\int_a^b dx\; f(x) = \sum_{n=1}^{N} P(x_n) \int_{a+\frac{(n-1)(b-a)}{N}}^{a+\frac{n(b-a)}{N}} dx\; \frac{f(x)}{P(x)}.$$

Instead of multiplying by the weight $P(x)$, use it to adjust the (still evenly distributed) number of samples in each region:

$$= \frac{b-a}{MN} \sum_{n=1}^{N} \left(P(x_n) \sum_{m=1}^{M} \frac{f(x_{nm})}{P(x_{nm})} \right) = \frac{b-a}{MN} \sum_{n=1}^{N} \sum_{m=1}^{M P(x_n)} \frac{f(x_{nm})}{P(x_{nm})}.$$

Convert this into a single region having x distributed with probability $P(x)$

$$= \frac{b-a}{MN} \sum_{k=1}^{M \sum_{n=1}^{N} P(x_n)} \frac{f(x_k)}{P(x_k)} = \frac{b-a}{MN} \sum_{k=1}^{\frac{MN}{b-a}\int_a^b dx\, P(x)} \frac{f(x_k)}{P(x_k)}.$$

As long as the number of points in the sum remains large, it can be rewritten as

$$= \frac{b-a}{MN} \frac{1}{b-a} \left(\int_a^b dx\; P(x) \right) \sum_{k=1}^{MN} \frac{f(x_k)}{P(x_k)}.$$

Opcode Descriptions

```
\   Define some macros:

: XhitO  NewNode  x   NewΠ         y  -x -y    p
         DiffInit x   NewΠ NewΣ    z  -x -z    p
                  x   NewΠ        -y  -x  y    p
                  x   NewΠ        -z  -x  z    p  ;

: YhitO  NewNode  y   NewΠ         x  -y -x    p
         DiffInit y   NewΠ NewΣ    z  -y -z    p
                  y   NewΠ        -x  -y  x    p
                  y   NewΠ        -z  -y  z    p  ;

: ZhitO  NewNode  z   NewΠ         x  -z -x    p
         DiffInit z   NewΠ NewΣ    y  -z -y    p
                  z   NewΠ        -x  -z  x    p
                  z   NewΠ        -y  -z  y    p  ;


: Xhit1           UseRand x NewΠ Minus     y -x -y    p
         DiffInit UseRand x NewΠ NewΣ      z -x -z    p
                  UseRand x NewΠ          -y -x  y    p
         NewRand  UseRand x NewΠ          -z -x  z    p ;

: Yhit1           UseRand y NewΠ Minus     x  -y -x    p
         DiffInit UseRand y NewΠ NewΣ      z  -y -z    p
                  UseRand y NewΠ          -x  -y  x    p
         NewRand  UseRand y NewΠ          -z  -y  z    p  ;

: Zhit1           UseRand z NewΠ Minus     x -z -x    p
         DiffInit UseRand z NewΠ NewΣ      y -z -y    p
                  UseRand z NewΠ          -x -z  x    p
         NewRand  UseRand z NewΠ          -y -z  y    p  ;


: Xhit          UseRand  x NewΠ Minus  y  -x -y   p
        Save    UseRand  x NewΠ NewΣ   z  -x -z   p
                UseRand  x NewΠ       -y  -x  y   p
        NewRand UseRand  x NewΠ       -z  -x  z   p  ;

: Yhit          UseRand  y NewΠ Minus  x  -y -x   p
        Save    UseRand  y NewΠ NewΣ   z  -y -z   p
                UseRand  y NewΠ       -x  -y  x   p
        NewRand UseRand  y NewΠ       -z  -y  z   p  ;

: Zhit          UseRand  z NewΠ Minus  x  -z -x   p
        Save    UseRand  z NewΠ NewΣ   y  -z -y   p
                UseRand  z NewΠ       -x  -z  x   p
        NewRand UseRand  z NewΠ       -y  -z  y   p  ;
```

```
(   Measure an N*M rectangular loop. Adds three loops aligned
along xy, xz, and yz faces. This does not include NewΣ, so all
the loops will be added together. Since the first node is no
different from the others, there is no way to initialize the
sum: it must be read before a measurement begins, and
subtracted from the result.  )
( N M - )
: AW   { N M }
x NewΠ N 1- times> x M times> y N times> -x M times> -y p
z NewΠ N 1- times> z M times> x N times> -z M times> -x p
y NewΠ N 1- times> y M times> z N times> -y M times> -z p
endOpcode ;


\ generates 32 opcodes each of which measures a different sized
\ square Wilson loop.
: measure 32 0 do i 1+ dup AW loop ;


\ ************** begin opcode definitions ********************

parseinit

\ Three opcodes, one to update each lattice dimension:

xhit0
xhit1
30 times> xhit
endOpcode

yhit0
yhit1
30 times> yhit
endOpcode

zhit0
zhit1
30 times> zhit
endOpcode

\ Measurement opcodes:

measure

\ save the contents of each rom in a separate binary file:

RomSave
```

REFERENCES

(AHO) J. Ambjorn, A. J. G. Hey, and S. Otto, Nucl. Phys. **B210** (1982) 347

(BMK) T. Banks, R. Meyerson, and J. Kogut, Nucl. Phys. **B129** (1977) 493

(DW) J. D. Walecka, Special Topics in Nuclear Physics, Seminar at CEBAF (1989)

(IP) A. Irbäck and C. Peterson, Phys. Rev. **D36** (1987) 3804

(MR) V. F. Müller and W. Rühl, Z. Phys. **C9** (1981) 261

(PFT) W. Press et al, *Numerical Recipes,* (Cambridge University Press, Cambridge, 1986)

(PS) C. Peterson and L. Skold, Nucl. Phys. **B255** (1985) 365

(SG) T. Sterling and J. Greensite, Nucl. Phys. **B220** (1983) 327

(WS) R. J. Wensley and J. D. Stack, Phys. Rev. Lett. **63** (1989) 1764

[1] A. S. Kronfeld and P.B. Mackenzie Editors, *Lattice 88 Proceedings of the 1988 Symposium on Lattice Field Theory,* (North-Holland, Amsterdam, 1989)

[2] Kenneth G. Wilson, Phys. Rev. **D10**, 2445 (1974)

[3] M. Göpfert and G. Mack, Comm. Math. Phys. **82** (1982) 545

4 J. Ambjorn, M. Flensburg, and C. Peterson, Nucl. Phys. **B275** (1986) 345

5 J. Flower, S. W. Otto, and S. Callahan, Phys. Rev. **D34** (1986) 598

6 R. P. Feynman and A. R. Hibbs, *Quantum Mechanics and Path Integrals,* (McGraw-Hill, New York, 1965)

7 D. A. McQuarrie, *Statistical Mechanics,* (Harper & Row, New York, 1976)

8 N. Metropolis et al, J. Chem. Phys. **21** (1953) 1087

9 M. Creutz, *Quarks, gluons and lattices,* (Cambridge Press, Cambridge, 1983)

10 R. Balian, J.M. Drouffe, and C. Itzykson, Phys. Rev. **D10** (1974) 3376

11 R. Balian et al., Phys. Rev. **D11** (1975) 2098

12 R. Balian et al., Phys. Rev. **D11** (1975) 2104

13 A. M. Polyakov, Nucl. Phys. **B120** (1977) 429

14 V. F. Müller and W. Rühl, in Proceedings of the 17th Winter School of Theoritical Physics Held in Karpacz, Poland edited by L. Turko and A. Pekalski

15 S. W. Golomb, *Shift Register Sequences,* (Holden-Day, San Francisco, 1967)

16 D. E. Knuth, *The art of Computer Programming Volume 2, Seminumerical Algorithms,* (Addison-Wesley, Reading, Mass. 1973)

17 W. Stahnke, Math. Comp. **27** (1973) 977

18 C. Heilman, CompuServe address [7Ø566,1474]

19 G. Y. Fletcher, Dr. Dobbs Journal, M&T Publishing Inc. **123** January 1987

20 Palo Alto Shipping Company, P.O. Box 7430, Menlo Park, CA 94026

21 G. Parisi et al, in *Lattice Gauge Theory Using Parallel Processors* edited by X. Li et al, (Gordon and Breach, New York, 1987)

22 The APE Collaboration, in *Lattice 88 Proceedings of the 1988 Symposium on Lattice Field Theory,* edited by A. S. Kronfeld and P.B. Mackenzie, (North-Holland, Amsterdam, 1989)

23 M. Gao, in *Lattice Gauge Theory Using Parallel Processors* edited by X. Li et al, (Gordon and Breach, New York, 1987)

24 F. Butler, in *Lattice 88 Proceedings of the 1988 Symposium on Lattice Field Theory,* edited by A. S. Kronfeld and P.B. Mackenzie, (North-Holland, Amsterdam, 1989)

25 M. Denneau and D. Weingarten, in *Lattice Gauge Theory Using Parallel Processors* edited by X. Li et al, (Gordon and Breach, New York, 1987)

26 N. H. Christ, in *Lattice 88 Proceedings of the 1988 Symposium on Lattice Field Theory,* edited by A. S. Kronfeld and P.B. Mackenzie, (North-Holland, Amsterdam, 1989)

27 M. Fischler et al, in *Lattice 88 Proceedings of the 1988 Symposium on Lattice Field Theory,* edited by A. S. Kronfeld and P.B. Mackenzie, (North-Holland, Amsterdam, 1989)

28 M. Lücher, Nucl. Phys. **B180** (1981) 317

29 T. Toffoli and N. Margolus, *Cellular Automata Machines: A New Environment for Modeling,* (MIT Press, Cambridge, Massachusetts, 1987)

30 P. D. Wasserman, *Neural Computing: Theory and Practice,* (Van Nostrand Reinhold, New York 1989)

[31] N. C. Barford, *Experimental Measurements: Precision, Error and Truth,* (Second edition, John Wiley & Sons, New York, 1985)

VITA

Arthur Francis Griesser

Born in Plainfield, New Jersey on July 18, 1954. B.S. in Chemistry and Physics Magna Cum Laude from Western Carolina University, June 1976, M.S. in Physics from the College of William and Mary, May 1984, Ph.D. in Physics from the College of William and Mary, May 1991.