GPUs and Lattice QCD

A thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Science degree in Physics from the College of William and Mary

by

Aaron Dufour

Advisor: William Detmold

Senior Research Coordinator: Henry Krakauer

Date: May 2012

Introduction

The goal of this project is to perform Lattice QCD calculations on GPUs. In particular, we wish to decrease the cost of finding suitable sets of paths, via the Monte Carlo method, for approximating the path integral.

This project began with an application of these techniques applied to a much simpler problem, the one-dimensional quantum harmonic oscillator. This problem acted as a stepping stone to the ultimate goal by using the Monte Carlo technique without the complications of QCD, which include multiple dimensions and more complicated field variables.

Quantum chromodynamics, or QCD, is the theory governing the interactions of quarks via gluons. Lattice QCD is a technique for numerically investigating systems, using the rules of QCD. This technique is motivated by the fact that in QCD, the force-carrying particles can interact, thus greatly increasing the complexity as compared to QED. Due to the greater complexity, numerical techniques are required.

We will show an application of a simple form of Lattice QCD, in which we only consider the gluon action. This simplified version, called Yang-Mills SU(3) theory, is preferable because changing the field only causes local changes in the gluon action, making it computationally cheaper. Since we use only a single GPU, this decrease in computational complexity is necessary.

One-Dimensional Lattice QM

For our one-dimensional quantum mechanics problem, assume we wish to evaluate the evolution of a position from time t_i to time t_f . We can use a sum over all possible paths:

$$\langle x(t_f)|e^{-\hat{H}(t_f-t_i)}|x(t_i)\rangle = \int \mathcal{D}x(t)e^{-S[x]}$$

Here, $S[x]$ is the classical action. In order to calculate the sum over all paths, we represent an approximation of a path from $x(t_i)$ to $x(t_f)$ as a vector of $x(t)$ at regular intervals a, i.e.

$$x(t_i + ja) \text{ for } j = 0, 1, ..., N$$

where $t_f = t_i + aN$. Now that $x(t)$ has been approximated as a vector of discrete points, we can turn the integral over all paths into an integral over all possible values of each point (using the notation $x(t_n) = x_n$, for convenience):

$$\int \mathcal{D}x(t)e^{-S[x]} \to A \int_{-\infty}^{\infty} dx_1 dx_2 ... dx_{N-1} e^{-S[x]}$$

We do not integrate over x_0 or x_N because they are to be held fixed as boundary conditions. For our purposes, we don't need to worry about the normalization factor A.

Now, we need a way of approximating the classical action for a given path. The classical action is:

$$S[x] = \int_{t_i}^{t_f} \left(\frac{m \cdot \dot{x}(t)^2}{2} + V(x(t)) \right)$$

First, we find an approximation for the interval $[t_j, t_{j+1}]$. The obvious approximations are the trapezoidal sum for the integral over $V(x)$ and the finite difference approximation for $\dot{x}$:

$$\frac{m}{2} \left(\frac{x_{j+1} - x_j}{a} \right)^2 + \frac{1}{2}(V(x_j) + V(x_{j+1}))$$

Summing over these, we get our lattice action:

$$S_{lat}[x] = \sum_{j=0}^{N-1} \left(\frac{m}{2} \left(\frac{x_{j+1} - x_j}{a} \right) + \frac{1}{2}(V(x_j) + V(x_{j+1})) \right)$$

Finally, we have an equation for the propagator that can be evaluated to within a constant by numerical methods:

$$\langle x(t_f)|e^{-\hat{H}(t_f - t_i)}|x(t_i)\rangle = A \int dx_1 dx_2 ... dx_{N-1} e^{-S_{lat}[x]}$$

We wish to find the difference between energy levels, so we need to evaluate an excited state. In order to do this, we must interrupt the propagation of the groundstate with construction/destruction operators. In this case, we use the $\tilde{x}$ operator, so we are evaluating the odd-parity states:

$$\langle\langle x(t_2)x(t_1)\rangle\rangle = \frac{\int dx \langle x|e^{-\hat{H}(t_f-t_2)}\tilde{x}e^{-\hat{H}(t_2-t_1)}\tilde{x}e^{-\hat{H}(t_1-t_i)}|x\rangle}{\int dx \langle x|e^{-\hat{H}(t_f-t_i)}|x\rangle}$$

We can rewrite the numerator and denominator as sums over energy levels by evaluating $\langle x|e^{-\hat{H}(t_f-t_2)} = \sum_n e^{-E_n(t_f-t_2)}\langle E_n|$, etc.:

$$\langle\langle x(t_2)x(t_1)\rangle\rangle = \frac{\sum_{n,m} e^{-E_n(t_f-t_2)}\langle E_n|\tilde{x}e^{-\hat{H}(t_2-t_1)}\tilde{x}|E_m\rangle e^{-E_m(t_1-t_i)}}{\sum_n e^{-E_n(t_f-t_i)}}$$

By rearranging, we arrive at:

$$\langle\langle x(t_2)x(t_1)\rangle\rangle = \frac{\sum_{n,m} e^{-(E_n t_f - E_m t_i)}\langle E_n|\tilde{x}e^{-\hat{H}t-(E_m t_1 - E_n t_2)}\tilde{x}|E_n\rangle}{\sum_n e^{-E_n(t_f-t_i)}}$$

We can take $t_f - t_i \gg t_2 - t_1$ to eliminate the sums:

$$\langle\langle x(t_2)x(t_1)\rangle\rangle \approx \frac{e^{-E_0 T}\langle E_0|\tilde{x}e^{-(\hat{H}-E_0)t}\tilde{x}|E_0\rangle}{e^{-E_0 T}} = \langle E_0|\tilde{x}e^{-(\hat{H}-E_0)t}\tilde{x}|E_0\rangle$$

Quantum Harmonic Oscillator

For the harmonic oscillator, we have the potential $V(x) = \frac{x^2}{2}$, so our lattice action is:

$$S_{lat}[x] = \sum_{j=0}^{N-1} \left(\frac{m}{2} \left(\frac{x_{j+1} - x_j}{a} \right) + \frac{1}{2} \left(\frac{x_j^2}{2} + \frac{x_{j+1}^2}{2} \right) \right)$$

In this case our approximation of $\langle\langle x(t_2)x(t_1)\rangle\rangle$,

$$\langle E_0 | \tilde{x} e^{-(\hat{H}-E_0)t} \tilde{x} | E_0 \rangle = 0$$

because $\tilde{x}$ switches parity, so E_0 cannot propagate. Now, we can assume t is large (but still much smaller than T) to get:

$$G(t) = \langle\langle x(t_2)x(t_1)\rangle\rangle \approx |\langle E_0|\tilde{x}|E_0\rangle|^2 \cdot e^{-(E_1-E_0)t}$$

Then, we can calculate the difference between the lowest two energy levels by:

$$\log \left(\frac{G(t)}{G(t+a)} \right) \approx a(E_1 - E_0)$$

Quantum Harmonic Oscillator Results

In figures 1 and 2, we see results for the quantum harmonic oscillator with $a = \frac{1}{4}, T = 10$. For the quantum harmonic oscillator, we see results about what we expect for approximations of $E_1 - E_0$ when interrupting with the x operator. When the interruption is with the x^3 operator, we see contamination in the expected trend (figure 2).

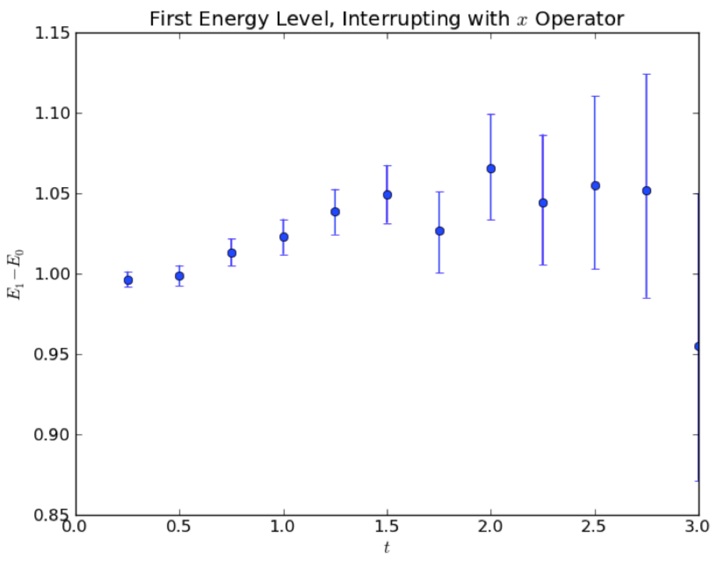

Figure 1: The graph of $\log\left(\frac{G(t)}{G(t+a)}\right) \cdot \frac{1}{a}$, approximating $E_1 - E_0$. The expected value of 1 can be seen. We look at $t \in [0, 3]$ because we assumed $t \ll T$.

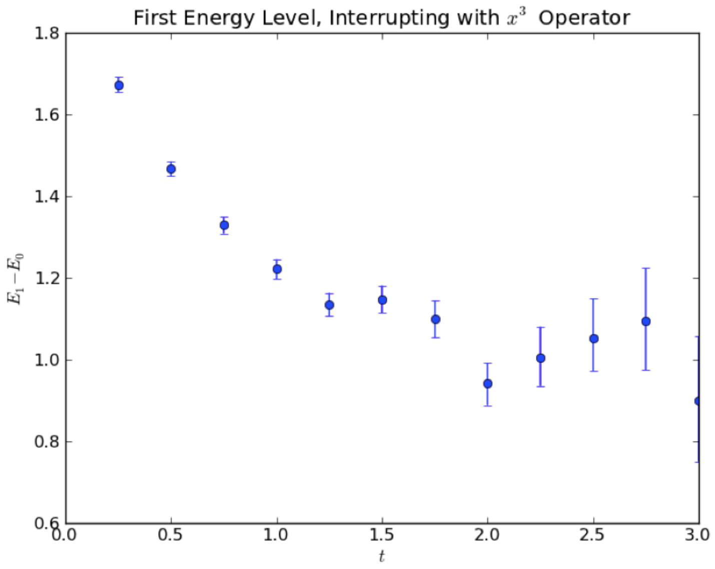

Figure 2: The graph of $\log\left(\frac{G(t)}{G(t+a)}\right) \cdot \frac{1}{a}$. The effect of higher energy levels can be seen contaminating the expected small-t trend of $E_1 - E_0 = 1$.

Discretization of the Gluon Action

The continuum action in terms of the field strength tensor is

$$S = \int d^4x \frac{1}{2} \sum_{\mu,\nu} Tr F_{\mu\nu}^2(x)$$

where the field strength tensor is a traceless 3×3 Hermitian matrix, written in terms of the gauge field as

$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu + ig[A_\mu, A_\nu]$$

An important characteristic of this theory is its invariance under SU(3) gauge transformations. When discretizing the theory, it is important to keep exact gauge invariance.

To this end, we discretize the field by storing values for the links between sites rather than for the sites themselves. The value at a link is the integral between its ends. For the link between x and $x + a\mu$, the link value is:

$$U_\mu(x) \equiv \mathcal{P} exp(-i \int_x^{x+a\mu} gA \cdot dy)$$

where $\mathcal{P}$ path-orders the integral the As along the integration path.

Using U_μ rather than A_μ confers the advantage of allowing for exact gauge invariance on the lattice. Where the gauge transformation of $F_{\mu\nu}$ by the x-dependent SU_3 matrix $\Omega(x)$ is

$$F_{\mu\nu} \to \Omega(x) F_{\mu\nu} \Omega(x)^\dagger$$

the U_μ variables are SU_3 matrices that transform as

$$U_\mu(x) \to \Omega(x) U_\mu(x) \Omega(x + a\mu)^\dagger$$

A link variable $U_\mu(x)$ is shown as a directed line from x to $x + a\mu$, which represents the path of the integral:

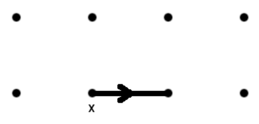

Figure 3: $U_\mu(x)$ on the lattice, represented by points.

The conjugate matrix $U_\mu^\dagger(x)$ is shown as a directed line from $x + a\mu$ to x:

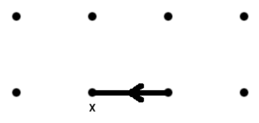

Figure 4: $U_\mu^\dagger(x)$

The paths that we are interested in are closed paths called Wilson loops. They are defined as

$$W(C) \equiv \frac{1}{3} Tr \mathcal{P} exp(-i \oint_C gA \cdot dy)$$

for any closed path C made up of lattice links. A simple loop, the $2a \times a$ rectangle shown in figure 5, would be calculated by

$$W(C) = \frac{1}{3} Tr(U_\mu(x)U_\mu(x + a\mu)U_\nu(x + 2a\mu)U_\mu^\dagger(x + a\nu + a\mu)U_\mu^\dagger(x + a\nu)U_\nu^\dagger(x))$$

As a product of link variables, Wilson loops are invariant under gauge transformations.

Now we need the lattice Lagrangian in terms of link variables. As discussed previously, we are looking for a Lagrangian that is gauge invariant and local,

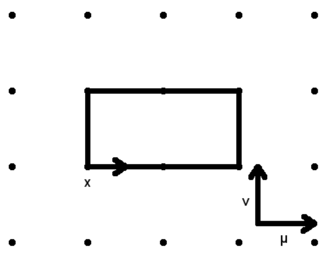

Figure 5: The 2×1 Wilson loop at x.

which points us towards small Wilson loops. We also require that the Lagrangian be symmetric with respect to changes of axes, which is the subset of Lorentz invariance that remains on the lattice. The smallest Wilson loop, called the "plaquette operator", is the Wilson loop around an $a \times a$ square at x:

$$P_{\mu\nu} = \frac{1}{3} Re(Tr(U_\mu(x) U_\nu(x + a\mu) U_\mu^\dagger(x + a\nu) U_\mu^\dagger(x))$$

Through examination of the behavior of $P_{\mu\nu}$, it can be shown that the gluon action can be approximated in terms of $P_{\mu\nu}$ by

$$S = \beta \sum_{x,\mu>\nu} (1 - P_{\mu\nu}(x))$$

where β determines the lattice spacing. This is equivalent to the continuum action given earlier, with correction of order a^2.

Monte Carlo Integration

In order to evaluate QCD quantities, we need a set of lattices such that the probability of a particular path being in the set is e^{-S}, for the action given above. This set of lattices can then be used to evalute the partition function:

$$Z_T = C \int dx_0 \cdots dx_{N_\tau - 1} e^{-S}$$

By interpreting e^{-S} as the probability that a particular configuration is in the set, we can approximate the partition function with an unweighted average. Observables can be calculated by taking integrals over each gluon configuration in the set, and again using an unweighted average.

We get this set of lattice configurations via the Monte Carlo method.

To get lattice $n + 1$ from lattice n, we update the lattice randomly, probabilistically keeping changes based on how they affect the action. To update a single link variable, we multiply it by a random SU_3 matrix. Then, we keep the change if it decreases the action. If it increases the action, we only keep it with probability $e^{-\delta S}$. The goal is for half of these updates to be kept. A single update of the lattice is when the update procedure is applied to each link.

We initially seed the lattice by setting each link equal to I_3. Then, we do the update procedure a number of times to eliminate the effect of our choice for initial values, or "thermalize the lattice". Lepage suggests that $5 \cdot N_{cor}$ times should be enough.

Next, we perform N_{cor} updates, and keep the resulting lattice. By repeating this procedure, we get as many lattices as we need. The value of N_{cor} should be large enough to prevent consecutive saved lattices from being statistically related. For our purpose, we use the value $N_{cor} = 50$.

Random SU_3 Matrices

This algorithm relies on being able to generate random SU_3 matrices. It is also useful to be able to tune these matrices to be further from or closer to I_3 so that we can get the desired update acceptance rate (Lepage suggests that 50%

is ideal).

Matrix Generation

In order to generate these matrices, we begin by generating SU_2 matrices. Again, we need to be able to tune the distribution, which we do by defining a variable ϵ. To get our results, we used the value $\epsilon = 0.2$.

First, we generate 3 numbers r_i in the range $[-0.5, 0.5]$ to define the vector r. We transform these into the vector x by:

$$x = \epsilon \cdot \frac{r}{|r|}$$

which normalizes the vector and then takes into account the tuning variable. We also define $x_0 = \sqrt{1-\epsilon^2}$, giving us a unit 4-vector. Finally, we use x as coefficients for the basis matrices to get our matrix:

$$U = \sum_i x_i \cdot u_i$$

where the basis matrices are

$$u_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, u_1 = \begin{pmatrix} 0 & \text{i} \\ \text{i} & 0 \end{pmatrix}$$

$$u_2 = \begin{pmatrix} 0 & \text{-1} \\ 1 & 0 \end{pmatrix}, u_3 = \begin{pmatrix} \text{i} & 0 \\ 0 & \text{-i} \end{pmatrix}$$

To generate an SU_3 matrix, we first generate 3 SU_2 matrices r, s, and t by the method given above. Then, we turn these into SU_3 matrices:

$$R = \begin{pmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix}, S = \begin{pmatrix} s_{11} & 0 & s_{12} \\ 0 & 1 & 0 \\ s_{21} & 0 & s_{22} \end{pmatrix}, T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & t_{11} & t_{12} \\ 0 & t_{21} & t_{22} \end{pmatrix}$$

Finally, our result SU_3 matrix is calculated by

$$X = R \cdot S \cdot T$$

Matrix Testing

To ensure that these matrices have the properties we wanted, we perform a few tests. Most importantly, we check that they are in SU_3. Thus, for a given matrix X, we must check that the following properties hold:

$$X^\dagger X = I_3, det(X) = 1$$

By generating 100000 random matrices, we can see the error distribution. Figure 6 shows the errors for the unitary matrix equation, and figure 7 shows the errors in the determinant. We get a maximum error of about 6 parts in 10^{16}, so the matrices should be close enough to special unitary for our purposes. The strange distribution is likely an effect of the fact that floating-point numbers are not continuous, and that the smallest delta that can be represented is around 10^{16}.

Next, we want to make sure our matrices are covering the full range of SU_3. We do this by decomposing each matrix into coefficients of the basis matrices, and looking at the distribution. We expect that the coefficients to the Gell-Mann matrices have similar distributions. It is also possible to check that ϵ tunes the variance of these coefficients.

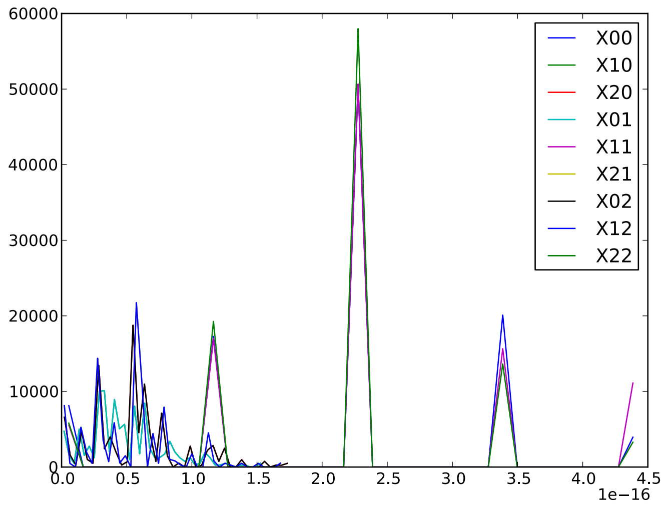

Figure 6: The distributions of the magnitude of each term in $X^\dagger X - I_3$.

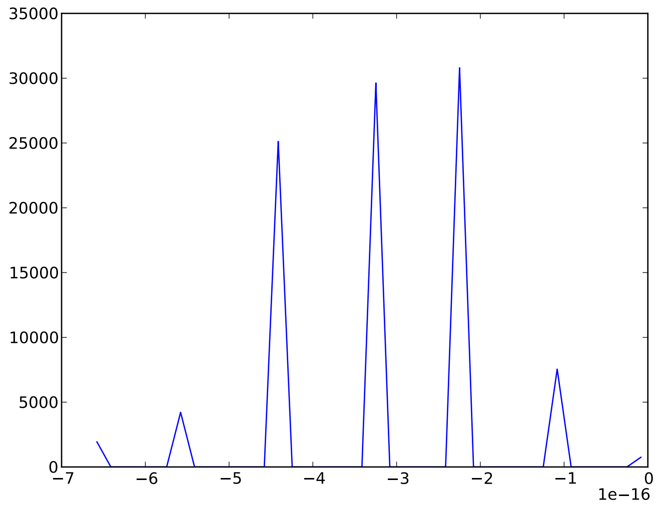

Figure 7: The distribution of the magnitudes of $det(X) - 1$.

The basis matrices for SU_3 are the following:

$$\lambda_0 = I_3, \lambda_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \lambda_2 = \begin{pmatrix} 0 & -i & 0 \\ i & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\lambda_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \lambda_4 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \lambda_5 = \begin{pmatrix} 0 & 0 & -i \\ 0 & 0 & 0 \\ i & 0 & 0 \end{pmatrix}$$

$$\lambda_6 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \lambda_7 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -i \\ 0 & i & 0 \end{pmatrix}, \lambda_8 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

where λ_i for $i = 1..8$ are the Gell-Mann matrices. To decompose a matrix into a summation $X = \sum_i x_i \lambda_i$, we use $x_i = Tr(\lambda_i \cdot X)$. The results for this decomposition on a set of 100000 matrices is shown. In figures 8 and 9, we have the imaginary and real components, respectively, of the x_is for $\epsilon = 0.5$. As expected, the coefficients of the Gell-Mann matrices are very similar. In figures 10 and 11, we have the imaginary and real components, respectively, of the x_is for $\epsilon = 0.3$. Here, we can see that decreasing ϵ decreased the variances. We also see that all of the matrices have a large x_0, corresponding to the matrices being close to I_3, which we need to ensure that the changes to the link variables are small.

Static Quark Potential

We examine the potential between a static quark and a static antiquark because it is a fairly simple, but meaningful, calculation. The potential should be linear with r, the distance between the particles, at long distances and should go like $-\frac{1}{r^2}$ at short distances.

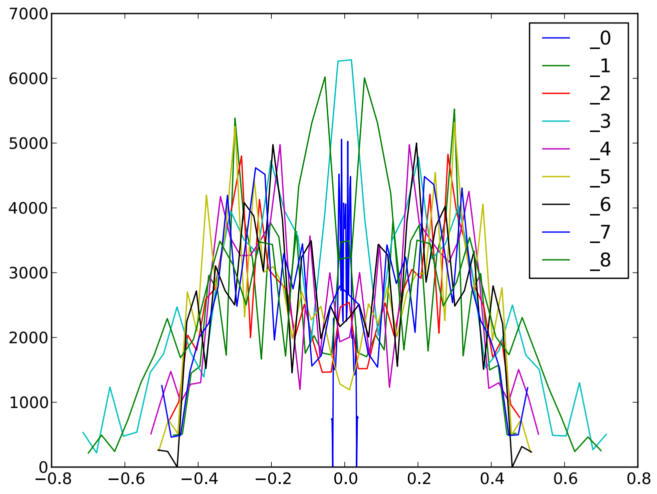

Figure 8: The distribution of the coefficients imaginary components, with $\epsilon = 0.5$.

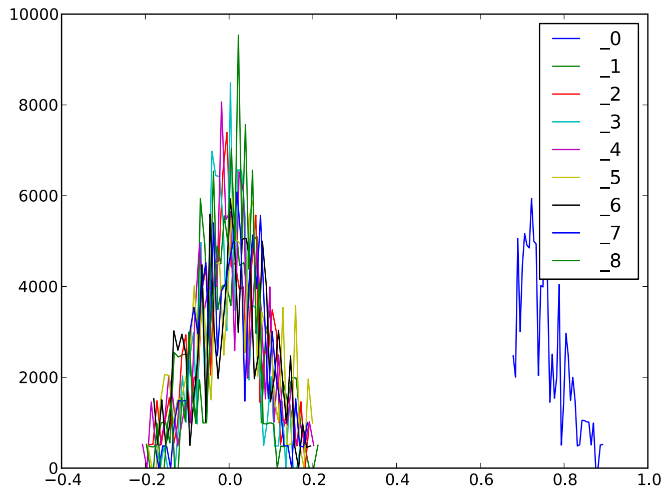

Figure 9: The distribution of the coefficients real components, with $\epsilon = 0.5$.

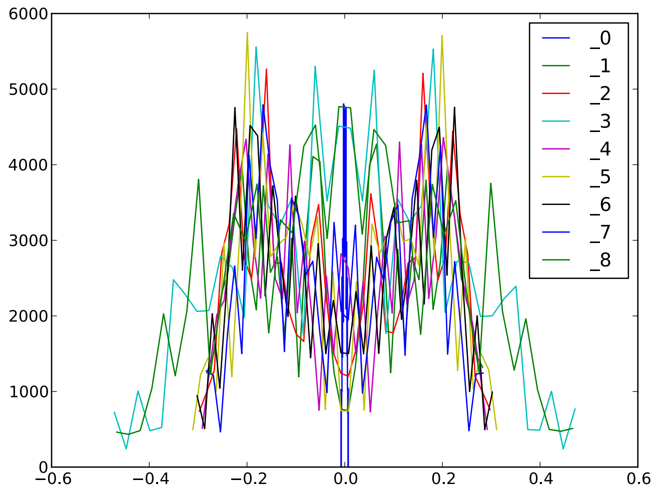

Figure 10: The distribution of the coefficients imaginary components, with $\epsilon = 0.3$.

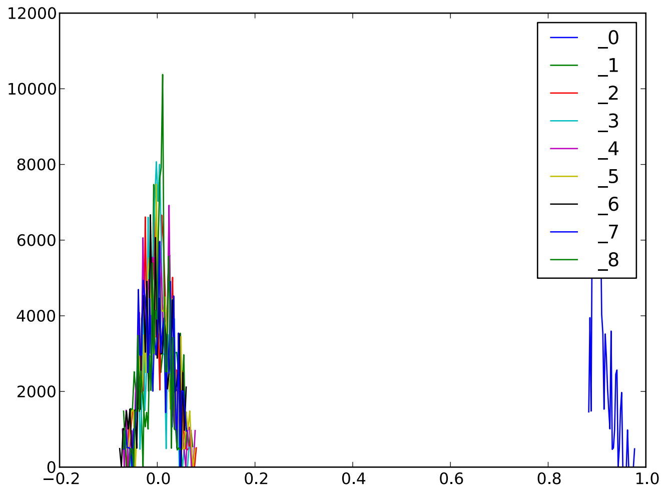

Figure 11: The distribution of the coefficients real components, with $\epsilon = 0.3$.

From Lepage, the propagator for a nonrelativistic quark with mass $M \to \infty$ is

$$G(x,t) = \left[\mathcal{P}\exp\left(-i\int_0^t gA_0(x,t)dt\right)\right]^\dagger \delta^3(x)$$

On the lattice, the propagator becomes

$$G(x,t) = U_t^\dagger(x,t-a)U_t^\dagger(x,t-2a)\cdots U_t^\dagger(x,0)$$

The propagator for an antiquark is $G^\dagger$. Thus, the static potential $V(r)$ can be found using the $r\times t$ Wilson loops, $W(r,t)$. Such a Wilson loop corresponds to taking a quark and an antiquark and pulling them apart a distance r, allowing them to propagate for time t, and then annihalating them. For large t, we approximate

$$W(r,t) \approx Ce^{-V(r)t}$$

To calculate $V(r)$, we compute $W(r,t)$ for successive values of t, getting

$$\frac{W(r,t)}{W(r,t+a)} \approx aV(r)$$

which is done using W values averaged over the configurations in our set.

Error Prediction

To predict the error, we use a resampling method called "bootstrapping". For each lattice configuration that we saved, we calculate $W(r,t)$ over the range of r and t that we are interested in. Then, for each set of final $W(r,t)$ values, we take an unweighted average over a random sampling (with duplication allowed) of our initial set. With enough resamplings, we can calculate the mean and standard deviation.

Results

The change from a CPU version of the Monte Carlo procedure to a GPU version gave an increase in speed of around 150 times. This allowed us to increase from 3-dimensions to 4-dimensions, as well as double the number of sites in each dimension, with almost no loss in computation time. It is worth noting that the GPU version used is very rough, and has room for improvement. Further, because the GPU version is already parallel, it would be possible to use multiple GPUs without a significant change in the algorithm.

A check of the average plaquette values for a lattice generated by this GPU program agrees with the value given by Lepage, indicating that we are generating valid configurations. However, the GPU version of the Wilson loop function gives bad results, indicating that it is faulty. Since the CPU version of the Wilson loop function is too slow on the large lattices generated by the GPU, we examine a much smaller lattice - 4 sites in each of the 3 spatial direction and 12 sites in the time direction. Figure 12 shows the plot of $W(r,t)$ against t with a fixed r. We see an exponential decay as t increases, as expected from the equation for $W(r,t)$ above.

Conclusion

We've shown how to use the Monte Carlo method to perform some basic calculations in QCD as well as on the Quantum Harmonic Oscillator system. Although no physical QCD quantities were calculated, our calculations of Wilson loops shows that W decays exponentially with t.

A method for examining the output of a random $SU(3)$ matrix generator is

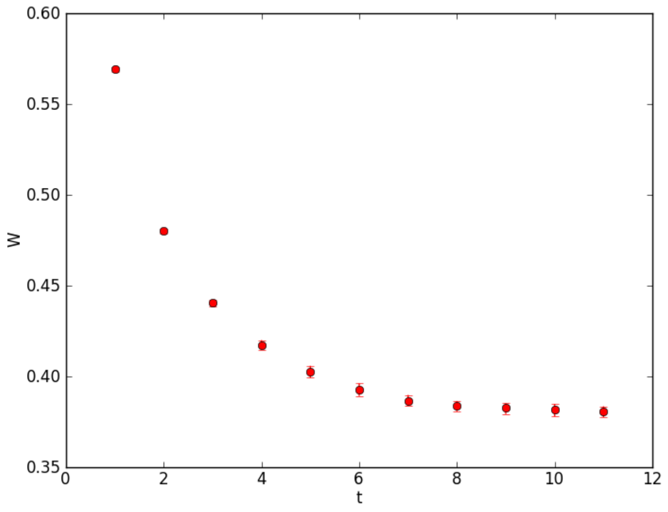

Figure 12: The graph of $W(r,t), r = 3a$ on a small lattice. The t axis is labeled in units of a. Error bars indicate one standard deviation, as determined by the bootstrapping method.

shown, as well as results from this method, which agreed qualitatively with our expectations.

Bibliography

Gattringer C., Lang C.B., *Quantum Chromodynamics on the Lattice: An Introductory Presentation*, Lect. Notes Phys. 788 (Springer, Berling Heidelberg 2010).

G.P. Lepage, *Lattice QCD for Novices*, arXiv:hep-lat/0506036.