Non-Perturbative Methods in QFT: Problem Set HT

James Graham

April 24th, 2017

Problem 1

We consider the compact version of the $U(1)$ gauge theory on a cubic lattice in $D = 4$ Euclidean spacetime dimensions with lattice spacing a. The group elements are

$$U_l = \exp(i\theta_l) \qquad \theta_l \in (-\pi, \pi]$$

on each link l of the lattice. We can index our links by the site from which they start and the direction in which they go:

$$\theta_l = \theta_\mu(n) \qquad n \in \mathbb{Z}^4$$

We use U_l when we go along a link in a positive direction and $U_l^\dagger$ when we go along a link in a negative direction. Now we consider a plaquette, which is the boundary of an elementary square on the lattice. We define $U_p = \prod U_l$ of the links on the plaquette, where we pick up two U_l and two $U_l^\dagger$, where we will have to index our links carefully. The lattice action is

$$\beta S[U] = \beta \sum_p (1 - \mathrm{Re}(U_p))$$

where we sum over all the plaquettes on the lattice, and our partition function is

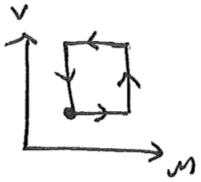
$$Z(\beta) = \int_{-\pi}^{\pi} \prod_l \mathrm{d}\theta_l e^{-\beta S[U]}$$

As we argued in the lecture, as $\beta \to \infty$, the only contributions we get to the partition function are from those U with $S[U] \to 0$ and the fields become smooth on the scale of the lattice spacing (we shall state what this means below).

We redefine our phases by a gauge field that takes as an argument a vector of Euclidean spacetime coordinates:

$$\theta_\mu(n) = aA_\mu(x = an + a\hat{\mu}/2)$$

where $\hat{\mu}$ is the unit vector in the μ direction. This A_μ, like the phase θ_μ, is a scalar defined at a particular point between sites on the lattice, but as we take the lattice spacing to zero these A_μ quantities will coalesce into a Euclidean four-vector defined at a spacetime point $x = an$. We are ready to write the plaquette operator in the μ, ν plane with its "lower left" corner, as it were, at an (see the sketch below).



$$U_p = \exp\Big(\underbrace{iaA_\mu(an + a\hat{\mu}/2) + iaA_\nu(an + a\hat{\mu} + a\hat{\nu}/2) - iaA_\mu(an + a\hat{\nu} + a\hat{\mu}/2) - iaA_\nu(an + a\hat{\nu}/2)}_{(*)}\Big) \quad (1)$$

As $\beta \to \infty$, we have $\mathrm{Re}(U_p) \to 1$ and so the argument of the exponential goes to zero. We can Taylor expand the argument of the exponential:

$$\begin{aligned}(*) &= ia\left(A_\nu(an + a\hat{\mu} + a\hat{\nu}/2) - A_\nu(an + a\hat{\nu}/2)\right) - ia\left(A_\mu(an + a\hat{\nu} + a\hat{\mu}/2) - A_\mu(an + a\hat{\mu}/2)\right) \\ &= ia\left(a\partial_\mu A_\nu(an + a\hat{\nu}/2) + \frac{a^2}{2}\partial_\mu^2 A_\nu(an + a\hat{\nu}/2)\right) \\ &\qquad - ia\left(a\partial_\nu A_\mu(an + a\hat{\mu}/2) + \frac{a^2}{2}\partial_\nu^2 A_\mu(an + a\hat{\mu}/2)\right) + 3^{\mathrm{rd}} \text{ derivative terms}\end{aligned}$$

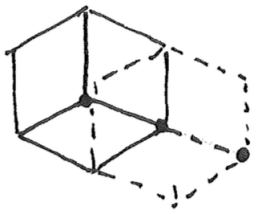
On the scale of the lattice spacing (i.e. a finite) the derivatives of A_μ have to go to zero; this is what we mean when the fields become smooth on the scale of the lattice spacing. On the other hand, if we take the lattice spacing to zero, we are allowed to have nonzero derivatives of A_μ and the terms that go to zero the slowest with lattice spacing are the first derivatives. As $a \to 0$ we get

$$(*) = ia^2\left(\partial_\mu A_\nu(an + a\hat{\nu}/2) - \partial_\nu A_\mu(an + a\hat{\mu}/2)\right) \quad (2)$$

We are ready to find the lattice action in the limit $\beta \to \infty$, $a \to 0$. In the limit $x \to 0$, $e^{ix} = 1 + ix - x^2/2 + \mathcal{O}(x^3)$, so each term in the lattice action becomes to lowest order in a

$$\frac{\beta}{2}a^4\left(\partial_\mu A_\nu(an + a\hat{\nu}/2) - \partial_\nu A_\mu(an + a\hat{\mu}/2)\right)^2 \quad (3)$$

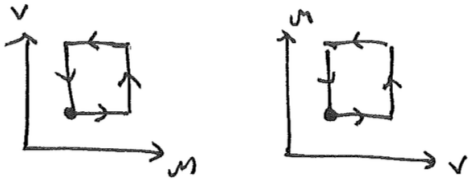
Now we need to come up with a sensible way of summing over all plaquettes in terms of sites and axes. This is achieved by summing over sites and by summing over plaquettes spanned by all pairs of axes extending in the positive direction from each site. For a three-dimensional analog of this situation, see the sketch below.



We can sum over all $\binom{4}{2}$ unique pairs of axes at each site or we can double-count and sum over all ordered airs of axes. This latter situation is easier to handle. So our total lattice action as $\beta \to \infty$, $a \to 0$ is

$$S[U] = \beta\sum_p(1 - \mathrm{Re}(U_p)) = \frac{\beta}{4}\sum_{n,\mu\,\nu} a^4\left(\partial_\mu A_\nu(an + a\hat{\nu}/2) - \partial_\nu A_\mu(an + a\hat{\mu}/2)\right)^2 \quad (4)$$

Note that when we double-count our plaquettes in this fashion, we go around each plaquette once the "proper" way and once the "wrong" way (see sketch below).



When we go backwards around the plaquette we pick up the hermitian conjugate of U_p in (1), which amounts to a minus sign in (2). But this means we have two extra minus signs in (3) so we actually get the same contribution to the action from each traversal of the plaquette, so formula (4) is correct and needs no further modification to account for the orientation of our plaquettes. We need to change now the differential in the partition function's integral:

$$\mathrm{d}\theta_l = a\mathrm{d}A_\mu(x = an + a\hat{\mu}/2) \implies \prod_l \mathrm{d}\theta_l = \prod_{\mu,x} a\mathrm{d}A_\mu(x)$$

We can now write down our partition function:

$$Z(\beta) = \int \prod_{\mu,x} a\mathrm{d}A_\mu(an + a\hat{\mu}/2) \exp\left(-\frac{\beta}{4}\sum_{n,\mu\,\nu} a^4 \left(\partial_\mu A_\nu(an + a\hat{\nu}/2) - \partial_\nu A_\mu(an + a\hat{\mu}/2)\right)^2\right)$$

We can neglect the a from the product in the measure because it amounts to a normalization that drops out when we compute correlation functions. In the limit $a \to 0$ we get $an + a\hat{\mu}/2 \to an = x$ and $\sum_n a^4$ becomes $\int \mathrm{d}^4x$. Define a coupling g by $\beta = 4/g^2$ and $F_{\mu\nu} = \partial_\mu A_\nu(x) - \partial_\nu A_\mu(x)$. Then our partition function is

$$Z(g) = \int \prod_{\mu,x} \mathrm{d}A_\mu(x) \exp\left(-\frac{1}{g^2}\int \mathrm{d}^4x \sum_{\mu\,\nu} F_{\mu\nu}F_{\mu\nu}\right)$$

Finally, we take advantage of the Euclidean metric and Einstein summation to write

$$\boxed{Z(g) = \int \prod_{\mu,x} \mathrm{d}A_\mu(x) \exp\left(-\frac{1}{g^2}\int \mathrm{d}^4x\, F_{\mu\nu}F^{\mu\nu}\right)}$$

Problem 2

When we discussed the $SU(3)$ gauge theory in the strong-coupling limit, we found the lightest glueball mass am_g with the correlator

$$\lim_{n_t\to\infty} \langle \tilde{U}^\dagger_{ij}(\bar{n}, n_t)\tilde{U}_{ij}(\bar{n}, 0)\rangle \to |c|^2\, e^{-am_g n_t}$$

where $\tilde{U}_p \equiv U_p - \langle U_p\rangle$. We also discussed that, because of how integrals of U_p^k for various powers k work with the Haar measure, we find that when we expand the exponential $e^{\beta S[U]}$ in its Taylor series, we get the lowest-order contribution at order β^{4n_t}. This corresponds to tiling the minimal tube that has at its ends $U^\dagger_{ij}(\bar{n}, n_t)$ and $U_{ij}(\bar{n}, 0)$.

Momentum-dependent operators are Fourier transforms of position-dependent operators:

$$U_{ij}(\bar{p}, n_t) = \sum_{\bar{n}} \exp(i\bar{p}\cdot\bar{n})U_{ij}(\bar{n}, n_t)$$

and we neglect vacuum contributions for $\bar{p} \neq 0$. In the long-time limit we have the correlator of momentum-dependent operators

$$\lim_{n_t\to\infty} \langle \tilde{U}^\dagger_{ij}(\bar{p}, n_t)\tilde{U}_{ij}(\bar{p}, 0)\rangle \to |c|^2\, e^{-aE_g(\bar{p})n_t}$$

We want to find this dispersion relation $aE_g(\bar{p})$ to lowest order in $\bar{p}$ for this lightest glueball. By the

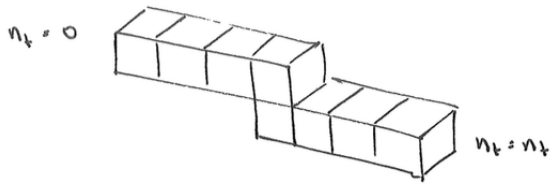
definitions of the operators and of expectation values, we have[1]

$$\begin{aligned}
\langle \tilde{U}^\dagger_{ij}(\bar{p}, n_t)\tilde{U}_{ij}(\bar{p}, 0)\rangle &= \sum_{\bar{n},\bar{k}} \langle e^{i\bar{p}\cdot(\bar{n}-\bar{k})}\tilde{U}^\dagger_{ij}(\bar{k}, n_t)\tilde{U}_{ij}(\bar{n}, 0)\rangle \\
&= \sum_{\bar{n},\bar{k}} e^{i\bar{p}\cdot(\bar{n}-\bar{k})}\langle U^\dagger_{ij}(\bar{k}, n_t)U_{ij}(\bar{n}, 0)\rangle \\
&= \sum_{\bar{n},\bar{k}} e^{i\bar{p}\cdot(\bar{n}-\bar{k})}\frac{1}{Z}\int \prod_\ell \mathrm{d}U_\ell\, U^\dagger_{ij}(\bar{k}, n_t)U_{ij}(\bar{n}, 0)e^{-\beta S[U_\ell]} \\
&= \sum_{\bar{n},\bar{k}} e^{i\bar{p}\cdot(\bar{n}-\bar{k})}\frac{1}{Z}\sum_j \frac{(-\beta)^j}{j!}\int \prod_\ell \mathrm{d}U_\ell\, U^\dagger_{ij}(\bar{k}, n_t)U_{ij}(\bar{n}, 0)\times S[U_\ell]^j
\end{aligned}$$

Clearly, the further apart $\bar{k}$ is from $\bar{n}$, the higher the order to which we get a nonzero contribution from each integral. The first of these terms is of order β^{4n_t} and comes when $\bar{k} = \bar{n}$. This term is of the form $N(b_0\beta)^{4n_t}$, where b_0 is a finite number coming from a single integral and N is the number of sites on our lattice. If $\bar{p} = 0$ then this is all we need, for

$$N(b_0\beta)^{4n_t} \propto e^{-aE_g(0)n_t} \implies aE_g(0) = 4\log\left(\frac{1}{b_0\beta}\right) = am_g$$

as one would expect. If $\bar{p} \neq 0$, then at this order in β we have no momentum dependence so we must press on. What terms correspond to the next lowest order in β? The ones for which $\bar{k}$ is displaced one unit from $\bar{n}$ in any of the spacelike directions on the lattice. Consider the sketch below:



This minimal tube involves $4n_t+4$ plaquettes coming from the exponential of the action, but it is not unique; when we have d spacelike dimensions, then there are $2d \times (n_t - 1)$: $2d$ from $\bar{n} \pm \hat{\mu}$ for each spacelike basis unit vector $\hat{\mu}$ and $n_t - 1$ choices of where to put the 'step' along the n_t axis. We assume the lattice has periodic boundary conditions so the path integral has no dependence on $\hat{n}$. If we also assume that the extent of the lattice is the same in each spatial direction, then by discrete rotational symmetry we should expect the integral not to depend on the direction in which $\bar{k}$ is displaced from $\bar{n}$. Then our next-order term is of the form

$$N\sum_{\hat{\mu}} e^{i\bar{p}\cdot\hat{\mu}}(C\beta)^{4n_t+4} = N(b_1\beta)^{4n_t+4}\sum_{\hat{\mu}}\cos(p_\mu)$$

where C comes from evaluating the path integral and b_1 both b_1 and the factor of two we get from converting the complex exponentials to cosines. We write both terms together and find

$$\langle \tilde{U}^\dagger_{ij}(\bar{p}, n_t)\tilde{U}_{ij}(\bar{p}, 0)\rangle \propto (b_0\beta)^{4n_t}\left[1 + \frac{b_1^{4n_t+4}}{b_0^{4n_t}}\beta^4\sum_{\hat{\mu}}\cos(p_\mu)\right] \propto e^{-aE_g(\bar{p})n_t}$$

Unfortunately, we have no way to tell how b_1 scales compared to b_0 and β, but let's assume that $b_1^{4n_t+4}\beta^4/b_0^{4n_t} \ll 1$. Then in the limit of large n_t we get

$$aE_g(\bar{p})n_t = 4n_t\log\left(\frac{1}{b\beta}\right) - \frac{b_1^{4n_t+4}}{b_0^{4n_t}}\beta^4\sum_{\hat{\mu}}\cos(p_\mu)$$

[1] In the second line we dispense with the $\tilde{U}_p$ notation since in general $\bar{p} \neq 0$.

$$\boxed{aE_g(\bar{p}) = am_g - \frac{b_1^{4n_t+4}}{b_0^{4n_t}} \frac{\beta^4}{n_t} \sum_{\hat{\mu}} \cos(p_\mu)}$$

If a particle's momentum is very small[2], we get

$$aE_g(\bar{p}) = am_g - \frac{b_1^{4n_t+4}}{b_0^{4n_t}} \frac{\beta^4}{n_t}(1 - p^2)$$

This second term is just the β^{4n_t+4}-order term in the dispersion relation for $\bar{p} = 0$, a correction to the mass, giving us a renormalized mass gap m_g^* and letting us rewrite

$$\boxed{E_g(\bar{p}) = m_g^* + C\bar{p}^2} \tag{5}$$

What we have written as m_g^* is obviously not the full mass from evaluating the path integral to all orders; it is only the glueball mass to second lowest order in contributions from the path integral. To get this result we have taken strong coupling ($\beta \ll 1$) and small momentum (large lattice). The next contribution to the path integral is of order β^{4n_t+8} and our complex exponentials are always order 1. Then in (5) we have neglected terms of order $p^4\beta^{4n_t+4}$ and order β^{4n_t+8}. In this respect we are fine. On the other hand, we don't know how large the values of the path integrals are. We know that path integrals often diverge in the continuum limit but in finding correlators we divide by Z, so we probably don't get anything divergent after all. But it's an awkward loose end.

Problem 3

We consider the Metropolis algorithm to update link matrices. We have a gauge field $\{U_l\}$ and update a link matrix U_l by selecting a trial U_l' with probability $\tilde{P}(U_l \to U_l')$, constructed to be ergodic with $\tilde{P}(U_l \to U_l') = \tilde{P}(U_l' \to U_l)$, and accepting the trial link matrix with probability

$$P'(U_l \to U_l') = \min\left[\exp\left(-\beta(S[U_l'] - S[U_l])\right), 1\right]$$

We'd like to find the combined probability to update U_l to U_l', $P(U_l \to U_l')$ in the cases $U_l \neq U_l'$ and $U_l = U_l'$. If the trial link matrix is different from the current link matrix, then the combined probability is the product of the probabilities of first selecting U_l' as the trial link matrix and the probability of then accepting U_l' as the updated link matrix:

$$P(U_l \to U_l' \neq U_l) = \tilde{P}(U_l \to U_l')P'(U_l \to U_l')$$

The situation is different when we don't update the link matrix[3] because there are two mechanisms for this is to occur. We could select the current matrix U_l as our trial link matrix, or we could select and then reject a different link matrix. There is a probability to accept and then reject any other link matrix $U_l'' \neq U_l$, so we have

$$P(U_l \to U_l) = \tilde{P}(U_l \to U_l) + \sum_{U_l''} \tilde{P}(U_l \to U_l'')\left(1 - P'(U_l \to U_l'')\right)$$

Note we are not over counting any cases, since if $U_l'' = U_l$ we have probability zero to reject it and we have no contribution from the second term in the above equation.

We want to have an equilibrium probability density $P^{eq}(U_l) = \exp(-\beta S[U_l])$. This probability distribution is a fixed point of the Metropolis algorithm if we have detailed balance, so we need to show that our algorithm indeed satisfies detailed balance:

$$P^{eq}(U_l)P(U_l \to U_l') = P^{eq}(U_l')(U_l' \to U_l)$$

[2] $p_\mu \ll 1 \forall$ spacelike $\hat{\mu}$. We are only allowed small enough momenta for the Taylor expansion to work when we are on a very large lattice.

[3] Or accept the link matrix, but it is the same as the old one.

There are three cases we have to check. The first is when $U_l = U_l'$, but if this is the case we get detailed balance trivially.

The other two cases involve $U_l \neq U_l'$. Consider first $S[U_l] = S[U_l']$. Then $P'(U_l \to U_l') = 1$ and we have

$$P^{eq}(U_l)P(U_l \to U_l') = P^{eq}(U_l)\tilde{P}(U_l \to U_l') = P^{eq}(U_l')\tilde{P}(U_l' \to U_l) = P^{eq}(U_l')P(U_l' \to U_l)$$

where the second equality follows from the symmetric construction of $\tilde{P}$. If instead we have $S[U_l'] > S[U_l]$, then we have

$$\begin{aligned} P^{eq}(U_l)P(U_l \to U_l') &= \exp(-\beta S[U_l])\tilde{P}(U_l \to U_l')\exp(-\beta(S[U_l'] - S[U_l])) \\ &= \exp(-\beta S[U_l'])\tilde{P}(U_l' \to U_l) \\ &= P^{eq}(U_l)P(U_l' \to U_l) \end{aligned}$$

where the final equality comes from the fact that, for $S[U_l'] > S[U_l]$, we have $P'(U_l' \to U_l) = 1$.

We have shown that this construction of the Metropolis algorithm satisfies detailed balance and so we get the desired equilibrium distribution as a fixed point of the algorithm.

Problem 4

For a $U(1)$ gauge theory on a $D = 2 + 1$ lattice, we want to calculate the following quantities:

- the plaquette average $\langle \mathrm{Re}(U_p) \rangle$;
- the lightest $J^{PC} = 0^{--}$ glueball mass;
- the lightest $J^{PC} = 0^{++}$ glueball mass;
- the confining string tension $a^2\sigma$.

Apart from the plaquette average, which is just an average, these quantities (generically aE) are given by the exponent in the limit

$$\lim_{n_t \to \infty} \langle \Phi(t)^\dagger \Phi(0) \rangle \propto e^{-aEn_t}$$

This quantity is time-translation invariant, so we can put zero wherever we like. Also, on a periodic lattice such as the ones we use in this problem, there is nothing special about the time coordinate $n_t = 0$. So instead of calculating $\langle \Phi(t)^\dagger \Phi(0) \rangle$, we calculate

$$\left\langle \frac{1}{L_t} \sum_{T=0}^{L_t - 1} \Phi(t+T)^\dagger \Phi(T) \right\rangle$$

where L_t is the length of the lattice in the time direction.

I calculated the plaquette average and the correlators for $\beta = 2.0, 2.2, 2.3$ on periodic lattices lattices with dimensions $18 \times 18 \times 24$, $22 \times 22 \times 36$, $28 \times 28 \times 40$. Discussion of my results will follow, but first I describe my code[4], which was written in C++. I used the Metropolis algorithm to simulate the lattice theory. In this problem, we treat the $U(1)$ gauge theory with its representation as complex exponentials. Our link matrices, as in problem 1, are of the form $e^{i\theta}$ but in the code we store only the phases, and instead of multiplying link matrices when computing plaquettes, we add (and subtract, for Hermitian conjugates) the phases of those exponentials. I use the words 'link', 'link matrix' and 'phase' interchangeably in what follows.

[4]To see my code and a description virtually identical to this one, but with additional discussion of issues with my code and ways to make it more coherent and faster, see `http://github.com/jngraham/latticeU1_3`

Overview

My code is a cobbled-together concoction of functional and object-oriented approaches. My code is built from six key files:

- `globals.h`, which contains all the parameters we use to build our lattice, create the set V, update the lattice and so on;
- `simulate.cpp`, in which `simulate()` prepares the lattice and our set V and calls our other functions to generate field configurations, calculate our data points and write the data to file;
- `update.cpp`, in which `update()` takes as arguments the lattice and V, both by reference. This is the function that iterates over the lattice and updates each link in the lattice using the Metropolis algorithm;
- `operators.cpp`, which contains several functions that are called from `simulate.cpp` that take as arguments t and T and return values of $\Phi^\dagger(t+T)\Phi(T)$ for the various operators, while the expectations themselves are calculated inside `simulate()`;
- `write.cpp`, in which `write()` takes as arguments pointers to all our data arrays and writes those data to CSV files, which I analyze using MATLAB;

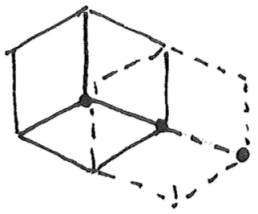
and most importantly:

- `Site.cpp`, which defines the `Site` class and its members. Each `Site` object has a member `double` for the phase of each link going away from the site in a **positive** direction, and a member `Site*` pointer to each neighboring `Site`. There are six such pointers, since our lattice is 2+1-dimensional and there is a neighbor in each direction on each axis. The class has only the default constructor, in which the links phases are set to zero and the pointers are set to null. The null pointers are overwritten immediately when we set up the lattice, and the zero phases are modified when we relax the lattice to equilibrium.

There is also `main.cpp`, which is left over from a previous version of the code in which β was not a global variable and instead we looped over an array of values of β, calling `simulate()` with each value as an argument.

All the arrays in my code are explicitly 1-dimensional, but except for samples' plaquette averages, all the arrays are implicitly multidimensional. Our `lattice` of pointers to `Site` objects has (global) dimensions `Lx`, `Ly` and `Lt`, so a `Site` with coordinates (x, y, t) on the 'actual' lattice is accessed at index `x + Lx*y + Lx*Ly*t`. Similarly, our data arrays (except for the plaquette averages) have (global) dimensions `N_samples` and `Lt`, since we must store $\langle\Phi^\dagger(t+T)\Phi(T)\rangle$ for each time interval for each sample, and so the data point for time interval t in sample i is at index `Lt*i + t`.

I don't things in a wholly (or even mostly) object-oriented way but I keep track of the plaquettes by assigning to each site the plaquettes that have their early, lower left-hand corner on that site. So, looking at the sketch below, one site 'has' the plaquettes outlined with solid lines and the neighbor site 'has' the plaquettes in dashed lines:



Algorithm

The algorithm starts in `simulate()`. The first thing that `simulate()` does is initialize and set to zero arrays of doubles to hold our data, and doubles to hold $\Phi(0)$ for all our operators, which we will need to refer to many times throughout the simulation for each configuration. Then it initializes and sets to zero an array of doubles `V`, which will become our set V. Finally, we create an array `lattice` of pointers to `Site` objects and allocate memory for each one, and loop through all x, y and t coordinates to give each `Site` pointers to its neighbors. I use modular arithmetic to enforce the periodic boundary conditions; if a `Site` has coordinate `x`, then the next and previous `Site` objects on the x axis are `(x+1)%Lx` and `(x+Lx-1)%Lx`, respectively.

The next step is to sample our set V from a normal distribution. We have a global parameter `N_V` for the number of elements of the set; I chose 400. We also have global parameters for the mean and standard deviation of the normal distribution from which we sample elements of V. I chose 0 (naturally) and 0.75 for these quantities, buying me an acceptance rate of around 50% , as suggested in the problem sheet. It is important that $\tilde{P}(U_\ell \to U'_\ell) = \tilde{P}(U'_\ell \to U_\ell)$ to ensure ergodicity per problem 3. This is achieved by also including the additive inverse of every sample in V, so we only sample `N_V/2` times from the normal distribution when we populate V. Once we have our set V we are free to relax our lattice to equilibrium.

Now the actual calculation begins. Our `simulate()` function loops first over the number of samples (which is a global variable) and, in each of those loops, over the number of configurations we take per sample. It is within this second loop that all the action[5] happens. Here we call `update()` to generate a new field configuration and calculate our relevant data. To calculate the correlators, we have a loop over $T \in [0, L_t - 1]$, where we calculate $\Phi(T)$ since we will need it L_t times and to calculate it every time is a waste. Within this loop, we loop over $t \in [0, L_t - 1]$ and call our various operators with t and T, as well as the lattice, as arguments. Using these operators we calculate $\Phi(t+T)^\dagger \Phi(T)$ for each pair of t and T, and add each value (scaled by `N_configs_per_sample` and `Lt`) to the growing sum in the appropriate entry of the appropriate array of data.

Once the loops are done, the bulk of the calculation is over. We delete the pointers to our `Site` objects from `lattice` and call `write()` to write our data arrays to CSV files. With the exception of our plaquette average array, which only has one data point for each sample, these files are constructed to have a different time, ranging from 0 to `Lt-1`, in each column, and the corresponding value of the appropriate observable from the appropriate sample in each row.

Lattice Update Details

The function `update()` takes β by value as an argument and `lattice` and `V` as arguments by reference, so that we can update all the links on all the sites using the Metropolis algorithm. I have a `for` loop over all the indices i on the lattice[6] but I access data using the `Site**`, the pointer to `lattice`, given as an argument, and increment that pointer each time through the loop. This is fairly half-baked, using a combination of features of arrays and linked lists at the same time, but it lets us dereference the pointers in `lattice` more easily and clearly. Within each loop we attempt to update the phases on the x, y and t links. I attempt to update each phase only once, rather than several times as suggested in the lecture. To illustrate, I describe the procedure to update the x link using the Metropolis algorithm; there are only minor differences (namely which pointers we dereference) when we update the y and t links.

First we store the phase of the current x link using `old_link = (*ptr)->xlink` and obtain a trial new link by adding to `old_link` a random phase chosen uniformly from V.

Since our lattice lives in 2+1 dimensions, each link belongs to four plaquettes, so to update each link we must calculate four 'staples', which we use to calculate the action. Each `Site` object has pointers to its neighbors, which themselves have pointers to their neighbors, so we dereference as many pointers as necessary to calculate the staples. For example, each link that points in the x direction belongs to two plaquettes that lie in the xy plane. The (correctly oriented) staples associated with these plaquettes are calculated by

[5] Pardon the pun.

[6] We needn't worry about the coordinates any more since each `Site` has pointers to its neighbors.

```
staple1 = (*ptr)->xnext->ylink - (*ptr)->ynext->xlink - (*ptr)->ylink;
staple2 = - (*ptr)->xnext->yprev->ylink - (*ptr)->yprev->xlink + (*ptr)->yprev->ylink;
```

where `ptr`, to emphasize, is a pointer to `lattice[i]` and is incremented with every loop. Once we have the staples, we have the plaquettes and so can calculate the action of the field configuration. Strictly, our path integral has integrand

$$e^{-\beta S[U]}, \qquad S[U] = \sum_p \left(1 - \mathrm{Re}\,\mathrm{Tr}\, U_p\right) \tag{6}$$

but the Metropolis algorithm only needs the quantity

$$C = \frac{\exp(-\beta S[U'])}{\exp(-\beta S[U])}$$

where U is the current field configuration and U' is the trial field configuration. When we take this ratio, the 1 terms in the action all cancel out, the minus signs cancel out, and the real parts of the traces of plaquettes cancel out when they are unchanged between U and U'. So when we update the phase θ_ℓ on link ℓ only have to calculate

$$C = \exp\left(\beta \sum_j \cos(\theta'_\ell + \xi_j) - \beta \sum_j \cos(\theta_\ell + \xi_j)\right)$$

where ξ_j the phase of staple j around link ℓ. These quantities, the sums of cosines, are what we calculate. We use the properties of exponentiation to only take one exponential for each link we want to update, since exponentiation is much more expensive than subtraction.

We uniformly select a z from $[0, 1)$ and compare it to C. If $z < C$, then we accept the new link. If $z \geq C$, then we keep the old link. This takes care of both cases of the transition probability at the same time. If the new action is smaller than the old action, then per (6) the new sum of cosines is larger. Then $C > 1$ and we always accept the new link. If, on the other hand, the new action is larger, then $0 < C < 1$ and we accept the new phase with probability C.

Operator Details

The average plaquette is calculated by iterating over all the `Site` objects in the lattice, and finding $\cos U_p$ for each of the plaquettes that 'belongs' to the site. There are three such plaquettes for every site. This average plaquette is passed by value as an argument to the function that calculates the operator from which we get $m_{0^{++}}$.

I calculate the correlators for $m_{0^{++}}$ and $m_{0^{--}}$ in very similar ways. For a given n_t, I find the indices of all the sites with that time coordinate and then find the phase of the plaquette in the xy slice through the lattice that 'belongs' to each site. The $m_{0^{++}}$ mass comes from the real part of the plaquette so for each configuration we have the operator

$$\Phi(n_t) = \sum_{n_x, n_y} \left[\mathrm{Re}(U_p(n_x, n_y, n_t)) - \langle \mathrm{Re}(U_p(n_x, n_y, n_t))\rangle\right] = \sum_{n_x, n_y} \left[\cos(U_p(n_x, n_y, n_t)) - \langle \cos(U_p(n_x, n_y, n_t))\rangle\right]$$

where the final quantity is what we actually calculate, which makes clear why we pass $\langle \cos(U_p)\rangle$ as an argument in the code[7]. Because we have only discrete rotational symmetry on the lattice, in the above calculation I take for $\langle \cos U_p\rangle$ the average over only the plaquettes that lie in the xy plane. When we extract from the lattice the data we use to calculate $m_{0^{--}}$, we have a much simpler operator

$$\Phi(n_t) = \sum_{n_x, n_y} \mathrm{Im}(U_p(n_x, n_y, n_t)) = \sum_{n_x, n_y} \sin(U_p(n_x, n_y, n_t))$$

[7]Properly we have $\mathrm{Re}(U_p)$ or $\cos(\theta_p)$ but as I said before I treat phases and matrix elements as interchangeable.

The flux tube operator is

$$\Phi(n_t) = \sum_{n_y} \prod_{n_x} U_p(n_x, n_y, n_t) = \sum_{n_y} \exp\Big(\sum_{n_x} i\theta_p\Big)$$

where we save as doubles the real and imaginary parts of $\Phi(n_t)$ and use trigonometric identities to calculate $\Phi^\dagger(n_t)\Phi(0)$ for every n_t for each configuration. We implement this operator by starting from a site with a particular $n_x = 0, n_y, n_t$ adding the phase of its x-pointing link to a total Θ and going to the site's x neighbor; then it's just a matter of adding phases and going to the end of the lattice in the x direction, finding the real and imaginary parts of Θ and summing those quantities over all n_y coordinates. Once we have saved the real and imaginary parts of the Wilson loop correlator to file, it is simple to calculate the norm of the correlator.

Minor Details

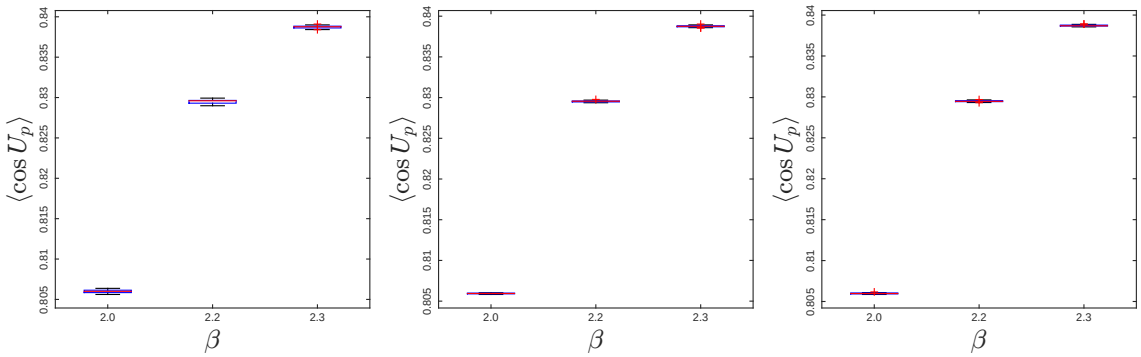
Possibly the most important subtlety in this procedure is how we sample V and so generate trial configurations. I have already discussed that for every $v \in V$, we also have $-v \in V$ to get us ergodicity and buy the Boltzmann factors as a steady state distribution. But just as important are the characteristics of the distribution from which we sample V. I sample V from a normal distribution with $\mu = 0$ and $\sigma = 0.75$ and I give the algorithm only one chance per configuration to update each link. This buys me an acceptance rate of around 50% (anecdotally between 45% and 55% each time I updated the configuration). If I had let the algorithm attempt to update each link several times, I would have had to sample V from a much wider distribution (anecdotally, with 5 attempts per link, I was able to get an acceptance rate of $\sim 90\%$ for $\sigma = 0.75$). On the other hand, had I used this multiple-update approach, I might have had for each sample a better ensemble of field configurations. However, the sheer number (10,000) of field configurations per sample may make this point moot. I am not quite sure how to satisfy oneself that an implementation of the Metropolis algorithm has produced a 'good' ensemble of microstates.

Because the elements of $U(1)$ are periodic mod 2π, what matters is $v \in V \bmod 2\pi$. The wider we make the distribution from which we sample V, the more it effectively approaches a uniform distribution. This is not a problem *per se*, but if we permit qualitatively large changes in our configuration relatively often by having such a wide distribution, it is possible we may end up with high-energy configurations that don't relax to equilibrium. This is why I chose $\sigma = 0.75$.

Again anecdotally, one of my classmates had a simulation that was logically perfect, but he found $\langle \cos U_p \rangle \approx 0.79$ on a small lattice for $\beta = 2.2$, when we should have found (and my simulation found) $\langle \cos U_p \rangle \approx 0.829$. After some discussion about what could possibly be wrong, it emerged that he allowed his code to attempt to update each link several times, and sampled V from a wide normal distribution. The fact that his average plaquette was too small means the average action of his configurations was too large, and so the large increments each time he updated a link phase meant his lattice could not relax to equilibrium.

Another subtle point is the choice of random number generator. When I developed my code, I used the C++ `default_random_engine`, which gives the same sequence of pseudorandom numbers every time. This was not ideal because I initialized a `default_random_engine` in `simulate()`, where I used it to populate V, and in `update()`, where I initialized[8] a new RNG every time I called update, giving the same numbers every time. The problem sheet says that even though the RNG on our personal machines is not great, I thought what I had went too far. So I changed to the MT19937 random generator, seeded by `clock()`. Because I didn't parallelize my simulation, this seed was different every time I simulated the lattice. I got new results every time and I avoided anything sketchy that might have happened due to my lack of understanding of C++.

[8] I think – I am not entirely clear on how C++ works.



(a) $18 \times 18 \times 24$ *(b)* $22 \times 22 \times 36$ *(c)* $28 \times 28 \times 40$

Figure 1: Average plaquette for different lattice sizes and coupling constants.

Results

In this section I discuss the results I found for the observables we were meant to calculate and occasionally digress to discussion of my MATLAB code that prepared figures and calculated fits and the choices I made when writing that program. I report my results in two-sided 99% confidence intervals. I consulted the National Institute of Standards and Technology's table[9] for critical t values. In this experiment we have 24 degrees of freedom so, for a two-sided confidence interval, we use $t_{0.995}(24) = 2.797$. The MAT and XLSX files containing my data can be found in my GitHub repository[10]. If you would like to mess with my data and my MATLAB programs, feel free to download the repository and see what happens.

To my surprise, the correlator for the lightest mass $m_{0^{++}}$ decayed to negative quantities. MATLAB has no functionality to fit data to a curve of the form $C + Ae^{Bt}$, and naively fitting exponentials to data with positive and negative values gave some decaying fits, some growing fits, some negative and some positive curves. This was no good. Once I had picked an interval on which my data looked to be decaying, but were not overtaken by noise, I shifted the data by the minimum of the correlator on that interval so that I fit to data that were positive and decayed to zero; then I shifted the curves back so that I effected something like a $C + Ae^{Bt}$ fit. This way I picked out the exponential decay and nothing else.

Average Plaquette $\langle \cos U_p \rangle$

As hinted, the plaquette average appears to a fairly ultraviolet quantity of this lattice gauge theory. On a small $10 \times 10 \times 10$ lattice I found an average around 0.829 for $\beta = 2.2$. Figure 1 displays box plots of of the average plaquette data, where the lengths of the whiskers indicate the bounds of 99% confidence intervals. The averages are tabulated in Table 1. In Figure 1 the boxes are rather squished, but I thought it was best to display the data in context with several values of β. The only real difference we get from calculating this quantity on a larger lattice with the same number of field configurations is that we have more plaquettes with which to calculate the average, buying us marginally smaller 99% confidence intervals.

Lattice Size	$\beta = 2.0$	$\beta = 2.2$	$\beta = 2.3$
$18 \times 18 \times 24$	0.8060(1)	0.8295(1)	0.8387(1)
$22 \times 22 \times 36$	0.80593(3)	0.82954(5)	0.83876(6)
$28 \times 28 \times 40$	0.80598(4)	0.82947(6)	0.83871(6)

Table 1: Average plaquette tabulated by lattice size and coupling constant

[9] `http://www.itl.nist.gov/div898/handbook/eda/section3/eda3672.htm`
[10] `http://github.com/jngraham/latticeU1_3`

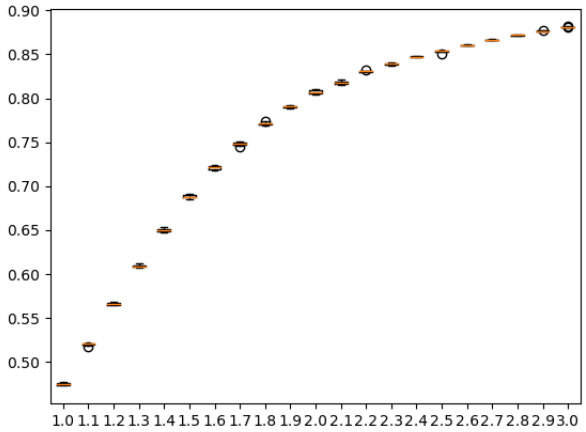


Figure 2: Average plaquette on a $10 \times 10 \times 10$ *lattice for several values of* β*.*

Out of curiosity, I did a simulation for a wide range of β on a small lattice with a small number of configurations per sample. Box plots from this simulation are shown in Figure 2. It appears there may be a second-order phase transition near around $\beta \sim 1.9$, but it is unclear. The average plaquette is of course bounded from above by 1, so the curvature of the plot of $\langle \cos U_p \rangle$ vs. β must change somewhere, but whether there is a discontinuity in $\mathrm{d}\langle \cos U_p \rangle / \mathrm{d}\beta$, we don't know.

Lightest Glueball Mass $m_{0^{--}}$

The quantity $am_{0^{--}}$ is obtained from the exponent of the exponential fit to the correlator of the operator

$$\Phi(\bar{p} = 0, n_t) = \sum_{\bar{n}} \mathrm{Im}(U_p(\bar{n}, n_t))$$

The averages of my data over all samples, including error bars indicating 99% confidence intervals, are plotted in Figure 4 in Appendix A. To capture the exponential decay but none of the noise, I chose to fit exponential curves to my data on the interval $[1, 10]$ for the $18 \times 18 \times 24$ lattice, and on the interval $[1, 14]$ for the larger lattices. My experiment's data here are largely positive but I shifted the data according to the minimum on the appropriate interval[11] before finding my exponential fits[12]. In these data we see an interesting feature: that as n_t approaches the end of the lattice, the correlation function grows. This was initially a surprise to me, but it really should not have been. Because we have constructed the lattice with periodic boundary conditions, the time interval between $n_t = 0$ and, say, $n_t = 39$ on the largest lattice is 1, not 39. This feature, in addition to the worries of noise dominating the data, is a good reason not to fit exponential curves to the entire time interval for which we sampled the correlator. This also inspired me first to consider computing $\frac{1}{2}\langle \Phi^\dagger(t)\Phi(0) + \frac{1}{2}\Phi^\dagger(-t)\Phi(0)$ before I decided to work with

$$\left\langle \frac{1}{L_t} \sum_{T=0}^{L_t - 1} \Phi(t+T)^\dagger \Phi(T) \right\rangle$$

[11] $[1, 10]$ or $[1, 14]$.

[12] For this to work, one has to fit on an interval in which the correlator **does** decay to an equilibrium value; if instead we fit to only part of the decay, not only would the fit possibly be less accurate given the smaller number of points MATLAB has to work with, but the shifted data would seem to decay to a negative value and we cause ourselves the same problem we are trying to solve.

Lattice Size	$\beta = 2.0$	$\beta = 2.2$	$\beta = 2.3$
$18 \times 18 \times 24$	0.454 ± 0.038	0.360 ± 0.037	0.379 ± 0.058
$22 \times 22 \times 36$	0.428 ± 0.048	0.325 ± 0.043	0.277 ± 0.040
$28 \times 28 \times 40$	0.422 ± 0.037	0.291 ± 0.034	0.277 ± 0.032

Table 2: Lowest glueball mass $am_{0^{--}}$ tabulated by lattice size and coupling constant with bounds of 99% confidence intervals.

Lattice Size	$\beta = 2.0$	$\beta = 2.2$	$\beta = 2.3$
$18 \times 18 \times 24$	1.736 ± 0.201	1.858 ± 0.204	2.083 ± 0.611
$22 \times 22 \times 36$	$1.812 \pm .142$	2.036 ± 0.286	2.067 ± 0.352
$28 \times 28 \times 40$	1.782 ± 0.133	2.329 ± 0.760	2.020 ± 0.292

Table 3: Lowest glueball mass $am_{0^{++}}$ tabulated by lattice size and coupling constant, including bounds on 99% confidence intervals.

My results are displayed in Table 2. It is easy to pick out by eye the exponential decays in Figure 4 so I am reasonably confident in my results. What qualitative statements can we make about this glueball? If the average plaquette is explicitly 'fairly ultraviolet', then one might expect this glueball mass and other energies in the problem to be more infrared and so depend on the size of the lattice. It is clear that the glueball gets lighter when we change β from 2 to 2.2, but it is not clear if this is true when we increase β to 2.3. At first glance the glueball gets lighter as the lattice size increases but the confidence intervals overlap except when we change the lattice size from $18 \times 18 \times 24$ to $22 \times 22 \times 36$ for $\beta = 2.3$. We can speculate that $am_{0^{--}} = 0.28$ is the limit of the glueball mass as the lattice becomes infinite for $\beta = 2.3$.

Is there a phase transition? Perhaps. Creutz (1981, p.136) writes about finding a phase transition in the plaquette average by looking at how many configurations it takes for an initial average to relax to an equilibrium average. We could use a similar procedure to look for a transition in $m_{0^{--}}$ as we change β. Near a phase transition one would expect the correlator to have difficulty 'deciding' how quickly to decay with t. We may be able to observe such effects by plotting $\langle \Phi^\dagger(t+T)\Phi(T)\rangle$ as we iterate the Metropolis algorithm from an initial condition, and seeing how many configurations it takes for the correlator to relax.

Lightest Glueball Mass $m_{0^{++}}$

The quantity $am_{0^{++}}$ is obtained from the exponent of the exponential fit to the correlator of the operator

$$\Phi(\bar{p} = 0, n_t) = \sum_{\bar{n}} \left[\mathrm{Re}(U_p(\bar{n}, n_t)) - \langle \mathrm{Re}(U_p(\bar{n}, n_t)) \rangle \right]$$

where the average plaquette is invariant under time and space translations because to compute it involved summing the plaquettes in all orientations in all time and space locations.

To my surprise, the correlator decays to a negative value. Appendix B has figures that show both the average of this correlator at each time over 25 samples with error bars indicating a 99% confidence interval, and scatter plots of all the data along with the 25 exponential fits. As I mentioned at the top of this section, each time I fit an exponential curve to a sample, I shifted the data so the minimum value of the sample on the time interval in which I calculated the data was set to zero. Looking at Figure 6, I felt it best to fit to my data for $n_t \in [1, 5]$. This has the disadvantage of introducing a certain amount uncertainty in the results of the MATLAB fitting toolbox, but I was not sure how to deal with these confidence intervals since my aim was to find $am_{0^{++}}$ for each sample and to average these quantities at the end.

My results are displayed in Table 3. There is nothing clearly going on here; we are hindered in our analysis by the wide confidence intervals. Does the glueball get lighter or heavier as β increases? Unclear. Is this glueball heavier on a larger lattice? Possibly; the average values of $m_{0^{++}}$ increase for $\beta = 2.2$ not not for the other values of coupling. However, the confidence intervals are rather large so we cannot say

anything with conviction. One feature of these data is that the glueball seems to have the same mass on all three lattices for $\beta = 2.3$. Whether this means that all our lattices are effectively infinite for this coupling is unclear. Our plots in Appendix B show that the correlator for this glueball decays very quickly so this constant value may be an artifact of our low time resolution on the lattice, but we note that our exponential fits do give different masses for the other coupling strengths. All we can say with confidence is that the $m_{0^{++}}$ particle is heavier than the $m_{0^{--}}$ particle.

Again, from looking at Figure 6, this glueball appears to be fairly heavy, and the correlator seems to go to noise after $n_t = 4$ or $n_t = 5$, so I chose to fit to my data on $[1, 5]$. I get a much lighter glueball, albeit one still heavier than $m_{0^{--}}$, by fitting exponential curves to my data on, say, $[1, 10]$. This of course raises the question of whether I am picking and choosing my data to get the results I want to see. To an extent this glueball **has** to be heavier from $m_{0^{--}}$ and to an extent I am picking and choosing my data to get the result I want.

Confining String Tension $a^2\sigma$

We obtain a flux tube energy from the exponential fit to the correlator of the operator

$$\Phi(\bar{p} = 0, n_t) = \sum_{n_y} \prod_{n_x=1}^{L_x} U_x(n_x, n_y, n_t)$$

In the long time limit, the correlator gives us

$$\langle \Phi^\dagger(n_t)\Phi(0)\rangle \to e^{-aE_f(l)n_t} \tag{7}$$

where the flux tube energy is given by the Nambu-Goto formula

$$\lim_{l\to\infty} E_f(l) = \sigma l - \frac{\pi}{6l} + \mathcal{O}\left(\frac{1}{l^3}\right)$$

On a lattice, the length of the string is given by an integer multiple of the lattice spacing. Our flux-tube operator winds around the lattice in the x direction, so we end up with $l = aL_x$ for three different values of L_x. In the end, we will only end up with one value of the confining string tension $a^2\sigma$ for each value of β. When we convert to lattice units, we end up with

$$aE_f(L_x) = a^2\sigma L_x - \frac{\pi}{6L_x} \tag{8}$$

and the coefficient in front of the first term is precisely what we want to calculate.

Before we get into calculations, we make some qualitative statements about the flux tube energy by staring at the absolute value of the Wilson loop operator in Figure 8. The operator decays with time and rate of decay seems to increase with L_x. These observations are consistent with a positive flux tube energy that is to leading order a positive power law of l. It also appears that the flux tube energy decreases as coupling decreases (as β increases). Fitting to this correlation function was the most most involved part of the data analysis because the flux tube energy has two dependencies here, both of which need to be captured by our fits. I chose to fit on $n_t \in [1, 6]$ for $\beta = 2.0$ and, for the other couplings, I fit on $[1, 10]$ for the small lattice and $[1, 8]$ for the larger lattices. Figure 9 in Appendix C shows all the fits to all my samples. The exponents from these fits, along with the standard error for 99% confidence intervals, are indicated in Table 4.

Lattice Size	$\beta = 2.0$	$\beta = 2.2$	$\beta = 2.3$
$18 \times 18 \times 24$	1.292 ± 0.070	0.755 ± 0.031	0.612 ± 0.044
$22 \times 22 \times 36$	1.757 ± 0.142	1.042 ± 0.048	0.848 ± 0.040
$28 \times 28 \times 40$	2.584 ± 0.801	1.594 ± 0.140	1.292 ± 0.088

Table 4: Flux tube energy for the Wilson loop correlator tabulated by lattice size and coupling constant, including bounds on 99% confidence intervals.

To fit lines of the form $y = mx$ to these data[13], I constructed the following equation in Mathematica for each value of β, filling in the energies from Table 4:

$$\chi^2(a^2\sigma) = \sum_{L_x \in \{18,22,28\}} \frac{(aE + \pi/6Lx - a^2\sigma L_x)^2}{a^2\sigma L_x} \tag{9}$$

I consulted the table of critical χ^2 values provided by Penn State[14]. The critical χ^2 value is 6.635 for a 99% confidence level, so we can solve equation (9) three ways: find the $a^2\sigma$ that minimizes the χ^2 statistic and solve for the upper and lower bounds on $a^2\sigma$ that result in $\chi^2 = \chi^2_{0.01}$. This procedure gets us a sense of the uncertainty we have for the confining string tensions. These values, including the minimum χ^2, are tabulated in Table 5.

β	Optimum $a^2\sigma$	$a^2\sigma$ Lower Bound	$a^2\sigma$ Upper Bound	Minimum χ^2
2.0	0.0842	0.0302	0.235	0.0532
2.2	0.0513	0.0142	0.185	0.0461
2.3	0.0417	0.0102	0.170	0.0358

Table 5: String tension for each β and the χ^2 statistic for fitting each data series to $E + \pi/6L_x = a^2\sigma L_x$ and minimizing χ^2. The critical value is $\chi^2_{0.01} = 6.635$ for confidence level 0.99 with one degree of freedom, obtained by setting $a^2\sigma$ to the upper and lower bounds.

As we suspected by looking at the figures in Appendix C, the string tension decreases as coupling $g \propto \beta^{-1/2}$ decreases. With MATLAB, I took the averages from Table 4 and added to each datum $\pi/6L_x$ per the Nambu-Goto formula. I then made scatter plots of these modified data and plotted as solid lines the fits I found by minimizing χ^2 per equation (9).

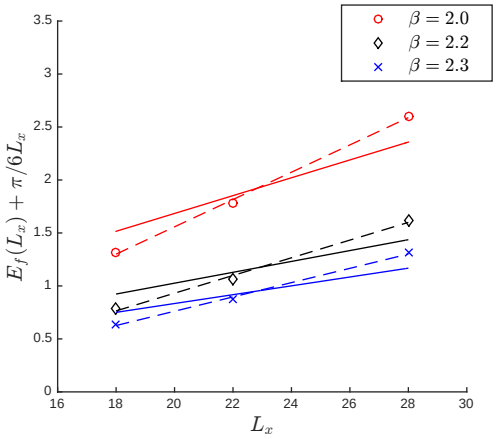


Figure 3: Scatter plot of data from which we calculate the string tension $a^2\sigma$. Each point is an average flux tube energy, to which we have added $\pi/6L_x$ and also plotted both linear fits.

Though one can make better linear fits to the data, indicated by dashed lines in Figure 3, these new fits have nontrivially nonzero y-intercepts and so contradict the Nambu-Goto formula.

Conclusions

The quantity I was able to measure with the most precision was the average plaquette. This is because in these lattices, there are from 10^4 to 10^5 plaquettes. We add them all up and divide by the total; nothing

[13] That is, lines with y-intercept zero.

[14] `http://sites.stat.psu.edu/~mga/401/tables/Chi-square-table.pdf`

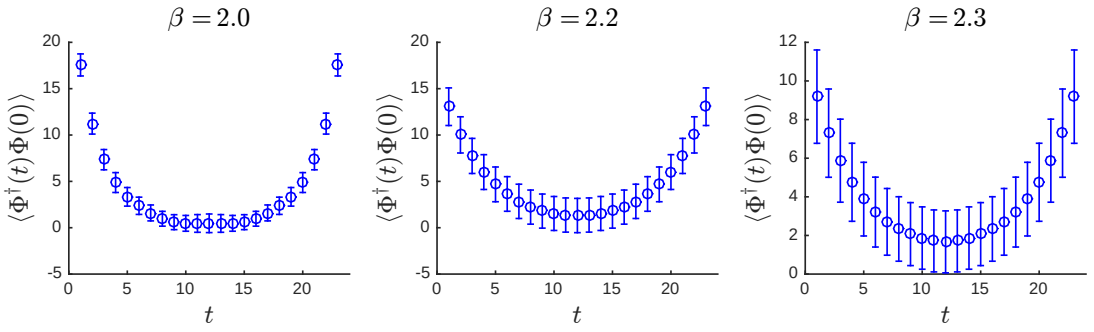
subtle goes on.

Calculating all the other quantities involved fitting exponential curves to a subset of the data, and were less precise. For each configuration, the correlator for each interval t is an average over tens of quantities instead of tens of thousands like we had for the plaquette average. Fitting curves and obtaining exponents was fairly straightforward with the correlator for $m_{0^{--}}$ and for several instances of the Wilson loop correlator, since the energies involved were relatively small and we are able to see the exponential decay before it is washed out by noise. This was much harder and less reliable for the higher-energy quantities, since we had to fit curves to our data on much smaller time intervals, and noise takes over very quickly anyway. This proved particularly hard for the Wilson loop correlator on the largest of the three lattices.

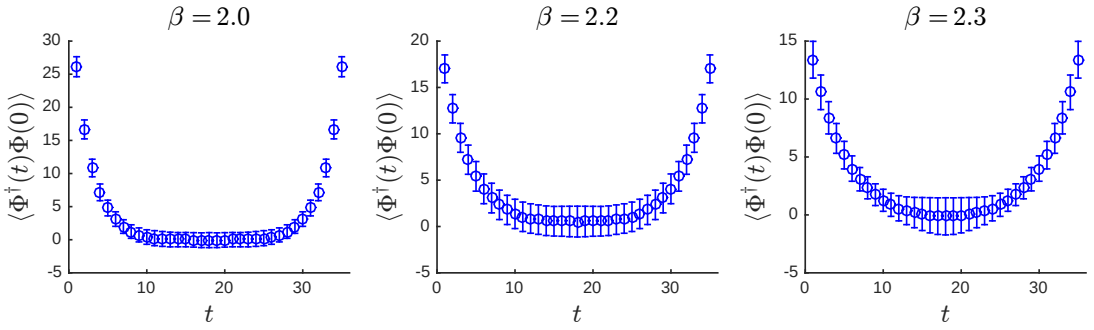
As I have hinted in places, we suffered from having only a coarse resolution in time. If we were able to have slices in the lattice of non-integer time coordinate, we could perhaps more easily see the correlators for $m_{0^{++}}$ and the higher-energy Wilson loops decaying. But how would this work? We would have two sites per unit time, so instead of a $18 \times 18 \times 24$ lattice going from $n_t = 0$ to $n_t = 23$, the time coordinate would only go to 11. But then we would have to reconsider how we update each link. We could certainly still have plaquettes of length 1 in every dimension; this would involve going along two links in the t direction when calculating our plaquette operators. But we'd have to modify the Metropolis algorithm in some fashion to reflect the fact that going from one site to its neighbor in the t direction only really picks up half of a link matrix. We would of course have links in the x and y directions from these half-integer time sites, so we'd have to be careful about how many of what phase we pick up when we go around a plaquette. As far as I can tell in Creutz's discussion of the Metropolis algorithm and quantities that can be extracted from an ensemble of lattice field configurations, there is no material on non-integer coordinates. This was beyond anyone's concern in the 1980s; computers were less powerful than they are today and the largest lattice in the literature at the time had 16^4 sites, only twice as many as we deal with in this problem (Creutz 1983, p.128).

Mike's email on 12th April suggests there are other and better ways to calculate $m_{0^{++}}$ and $a^2\sigma$ so I suppose I shall just have to wait and see.

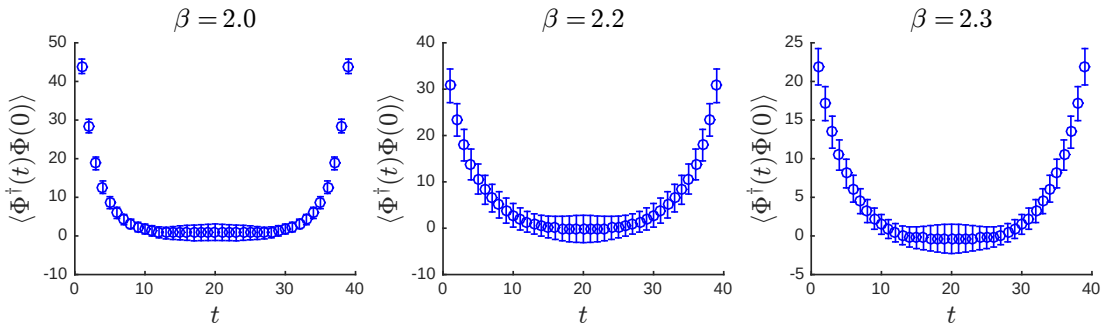
A $m_{0^{--}}$ Figures



(a) $18 \times 18 \times 24$ *lattice*

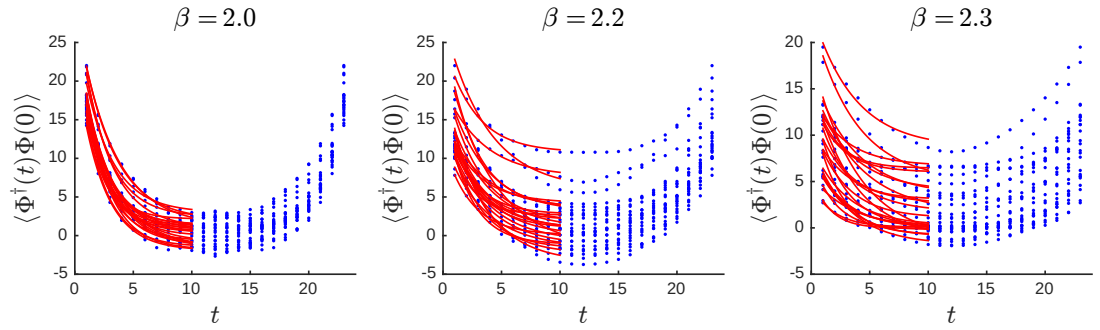


(b) $22 \times 22 \times 36$ *lattice*

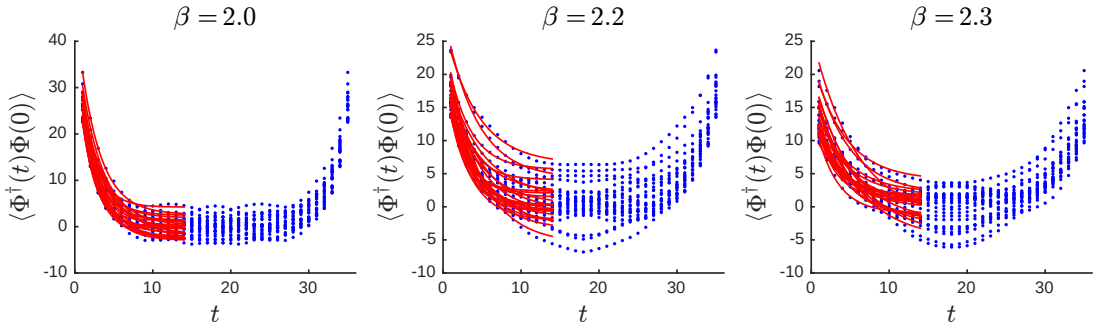


(c) $28 \times 28 \times 40$ *lattice*

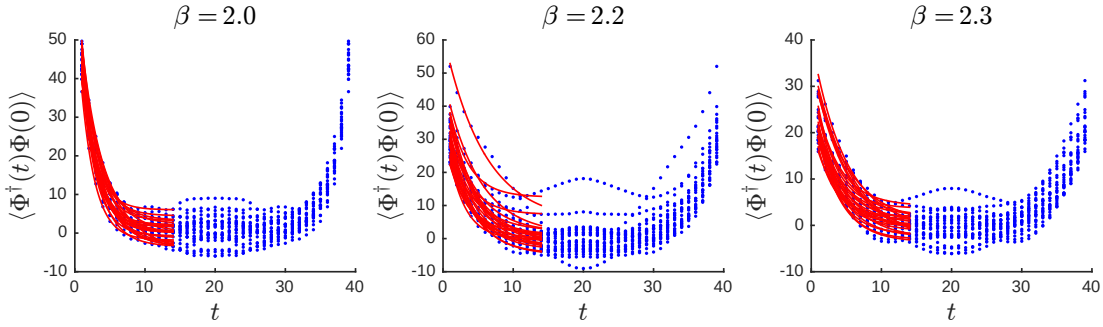
Figure 4: Average of correlator for $m_{0^{--}}$ *over samples for various lattice sizes; error bars indicate 99% confidence intervals.*



(a) $18 \times 18 \times 24$ *lattice*



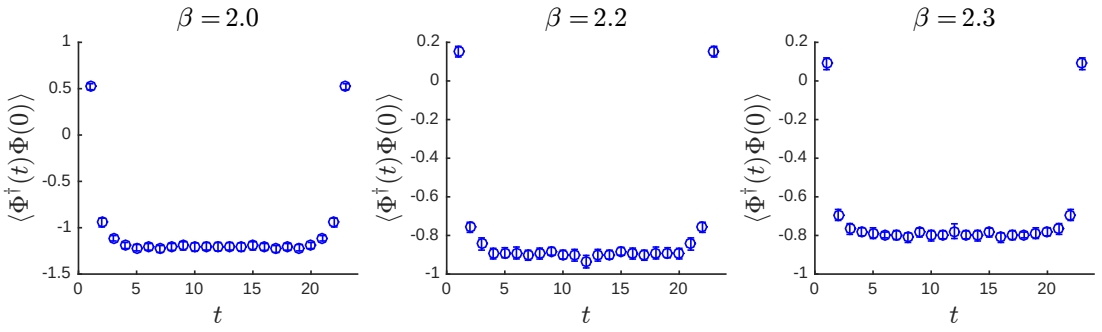
(b) $22 \times 22 \times 36$ *lattice*



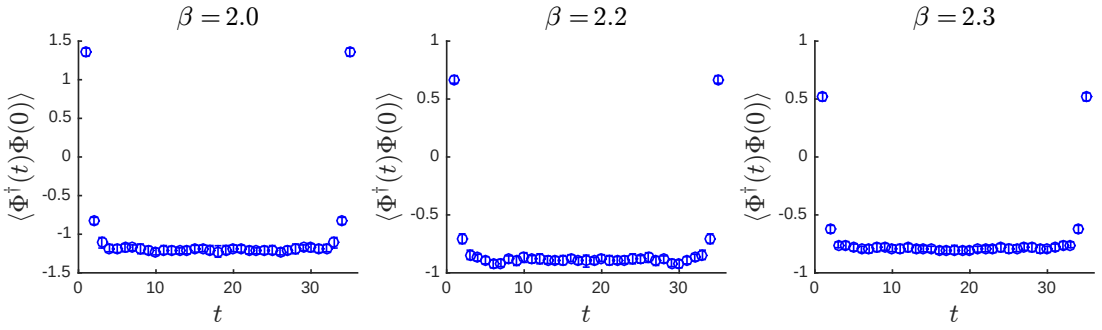
(c) $28 \times 28 \times 40$ *lattice*

Figure 5: Correlator data points including exponential fits, from which we find the data given in Table 2

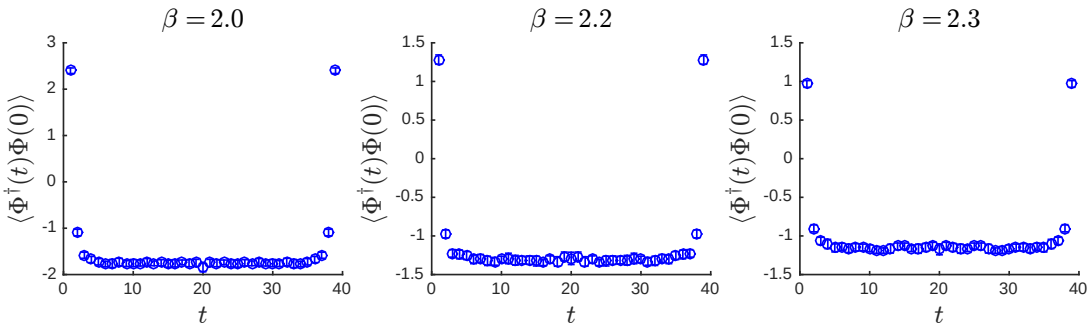
B $m_{0^{++}}$ Figures



(a) $18 \times 18 \times 24$ *lattice*

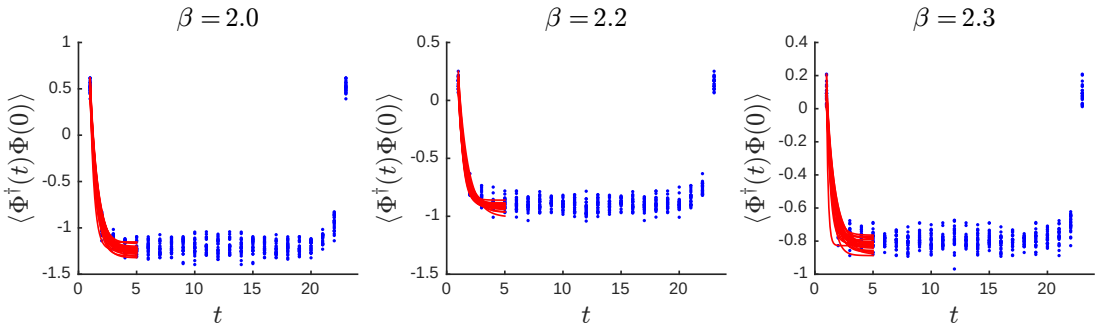


(b) $22 \times 22 \times 36$ *lattice*

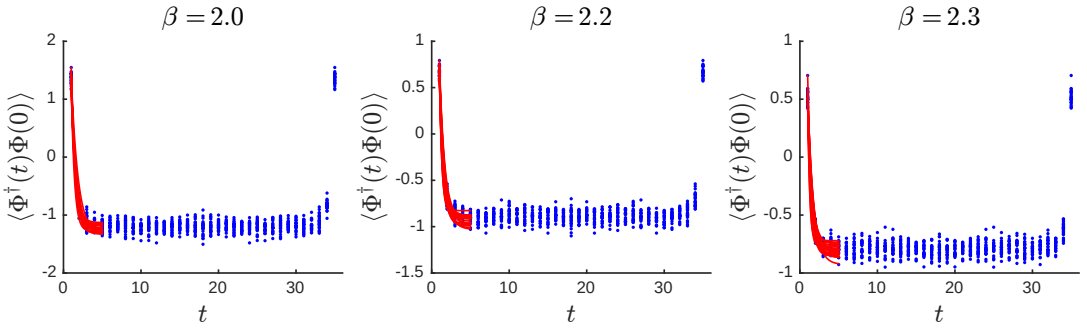


(c) $28 \times 28 \times 40$ *lattice*

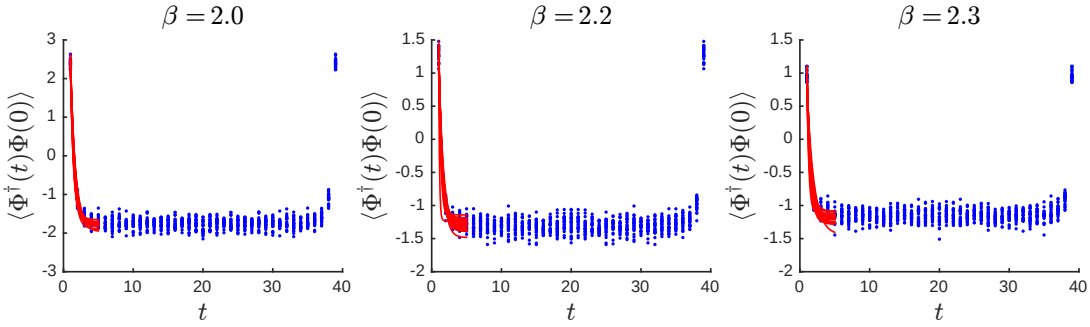
Figure 6: Average of correlator for $m_{0^{++}}$ *over samples for various lattice sizes; error bars indicate 99% confidence intervals for the correlator.*



(a) $18 \times 18 \times 24$ *lattice*



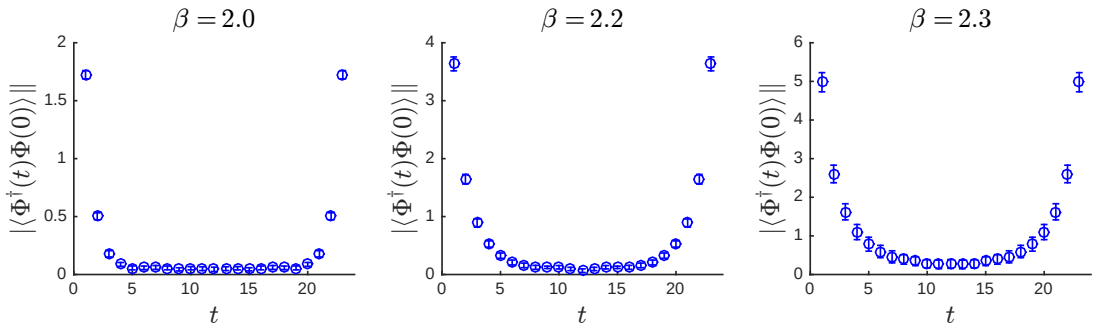
(b) $22 \times 22 \times 36$ *lattice*



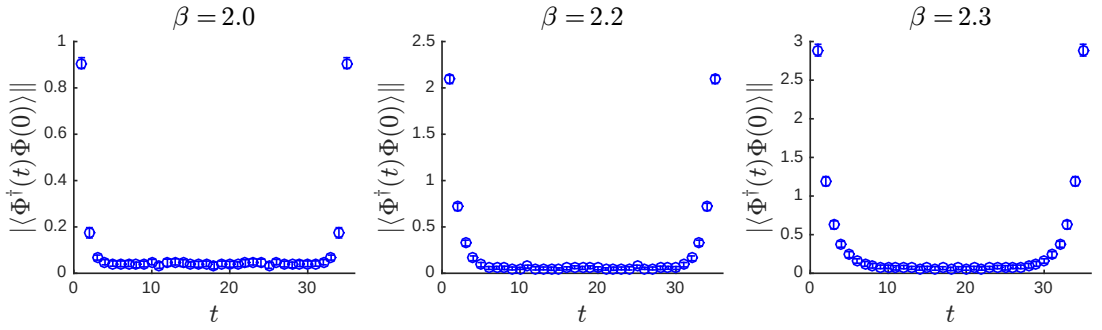
(c) $28 \times 28 \times 40$ *lattice*

Figure 7: Correlator data points including exponential fits, from which we find the data given in Table 3.

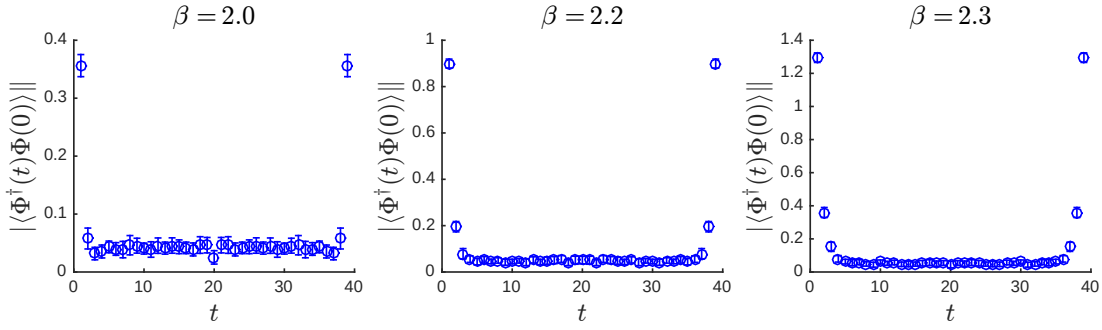
C $a^2\sigma$ Figures, Norm of Wilson Loop Correlator



(a) $18 \times 18 \times 24$ *lattice*

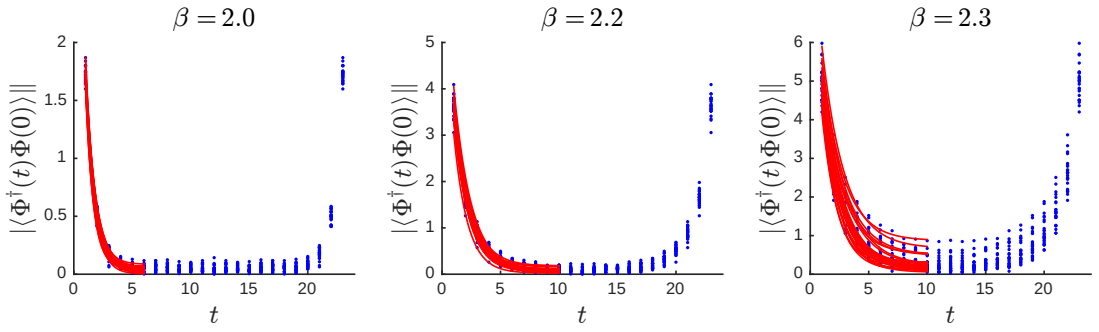


(b) $22 \times 22 \times 36$ *lattice*

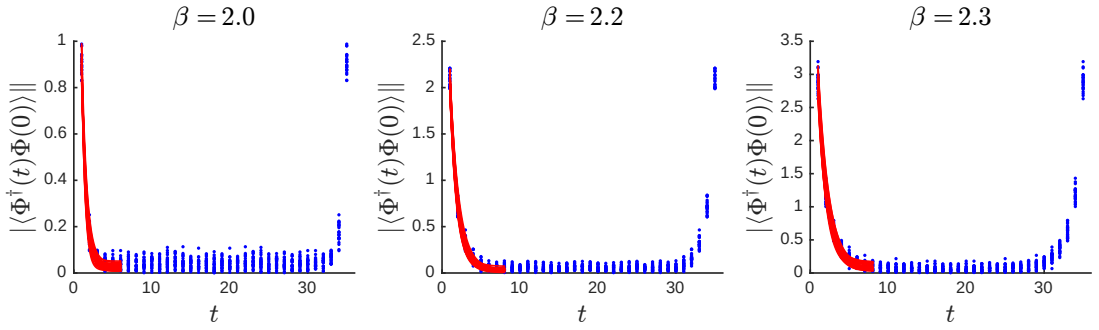


(c) $28 \times 28 \times 40$ *lattice*

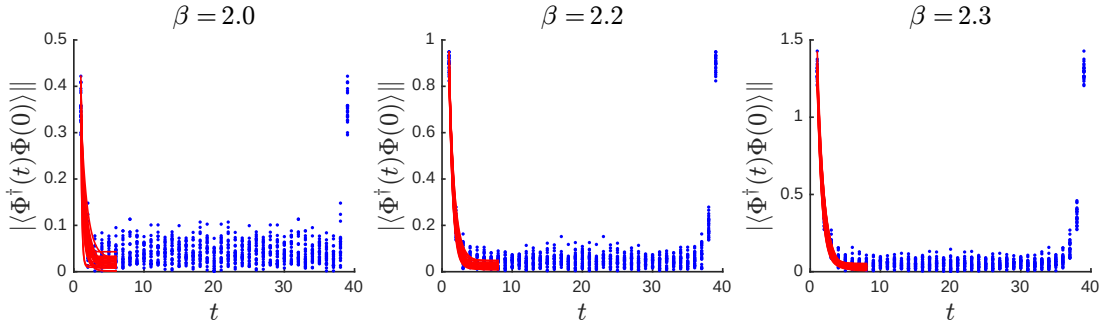
Figure 8: Average of absolute value of Wilson Loop correlator over samples for various lattice sizes; error bars indicate 99% confidence intervals for the correlator.



(a) $18 \times 18 \times 24$ *lattice*



(b) $22 \times 22 \times 36$ *lattice*



(c) $28 \times 28 \times 40$ *lattice*

Figure 9: Correlator data points including exponential fits, from which we find the data given in Table 4.