1 Lattice Quantum Chromodynamics

We now shift our focus to quantum chromodynamics (QCD). The field of QCD is the study of the strong interaction, one of the four fundamental forces, that describes the interactions between quarks and gluons, the mediator of the strong force. And together quarks and gluons make up hadrons (e.g. protons). Lattice QCD is a nonperturbative approach for modeling and calculating QCD measurements[1]. On the lattice, the quarks live on the sites while the gluon fields exists on the links.

In some ways it is analogous to the previous section in that instead of space coordinates $x(t)$, we deal with fields dependent on $x = (\mathbf{x}, t)$ where x is now a spacetime point. The lattice is now four dimensional and thus size of the lattice N will have a much greater effect on the computational time due to both the extra dimensions and the more involved calculations done per site on the lattice. So although the lattice size will be much smaller than previous models, computational time is much longer.

1.1 Derivation and Discretization

The value $A_\mu(x)$ represents the gauge field at the sites on the lattice but we cannot formulate a discretized version of QCD that is both gauge invariant and in terms of $A_\mu(x)$'s. Instead the variables on the links of the lattice representing the gluon fields, or the link variables, will be used. The link variable from some point x to its neighboring point in the $\hat{\mu}$ direction is the line integral

$$U_\mu(x) = \mathcal{P} \exp\left(-i \int_x^{x+a\hat{\mu}} g A_\mu \, \mathrm{d}y\right) \tag{1}$$

where $\mathcal{P}$ path-orders the integral. These link variables are $SU(3)$ matrices and so gauge invariance is possible due to their gauge transformation:

$$U_\mu(x) \to V(x) U_\mu(x) V^\dagger(x + a\hat{\mu}). \tag{2}$$

for $V(x) \in SU(3)$.

Visually, two examples of the link variable can be seen in figure 1. For the links in the negative direction, the inverse is used. But since the links are $SU(3)$ matrices, the conjugate transpose is equivalent. It is important to note that the $U_\mu(x)$'s paths move away from x while the $U_\mu^\dagger(x)$'s paths move towards x. Thus any path on the path can be created by a product of the link variables.

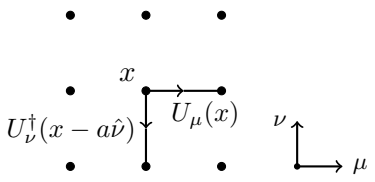

Figure 1: Two link variables on the $\mu\nu$ plane. Reverse directions are represented by the inverse of the link variable.

[1]The importance of lattice QCD being nonperturbative is described in the introduction but is due to the large coupling constant of the strong force.

The simplest path that can be created is the plaquette as shown in figure 2. Mathematically, this can be written as

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

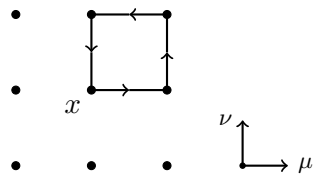

Figure 2: The plaquette which is the simplest guage invariant object that can be created from the link variables.

With this, the Wilson gauge action is

$$S = \beta \sum_x \sum_{\mu>\nu} [1 - P_{\mu\nu}(x)] \tag{4}$$

where $\beta = 6/g^2$ is the lattice coupling. The Wilson action is base on Wilson loops where the plaquette is an example of the simplest Wilson loop. It is the normalized ordered product of links in a path. It's used in equation 4 and will be used in the improved action derived shortly. For small a, if we expand the plaquette at some x_0 as a polynomial in a, due to A_μ being slowly varying because a is small, we find

$$P_{\mu\nu} = 1 - \frac{a^4}{6} \operatorname{Tr}(g F_\mu \nu(x_0))^2 + \mathcal{O}(a^6) \tag{5}$$

where $F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu + ig[A_\mu, A_\nu]$ is the field strength tensor. Taking the continuum limit $a \to 0$, what remains is the continuum action up to second-order shown in equation 6.

$$S = \int \mathrm{d}^4 x \, \frac{1}{2} \sum_{\mu,\nu} \operatorname{Tr} F_{\mu\nu}^2(x) \tag{6}$$

These higher-order terms complicate the action as such:

$$S = \int \mathrm{d}^4 x \sum_{\mu,\nu} \left[\frac{1}{2} \operatorname{Tr} F_{\mu\nu}^2(x) + \frac{a^2}{24} \operatorname{Tr} F_{\mu\nu} (D_\mu^2 + D_\nu^2) F_{\mu\nu} + \ldots \right] \tag{7}$$

The terms D_μ and D_ν are gauge covariant derivatives for their respective directions.

To cancel out the higher-order term, we can introduce the rectangle operator. Similar to the plaquette in equation 3, it is the normalized real trace of the path in figure 3. The rectangle

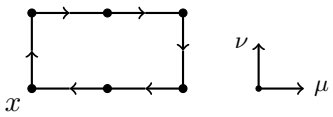

Figure 3: The rectangle operator at point x.

operator has as an expansion

$$R_{\mu\nu} = 1 - \frac{4a^4}{6}\,\mathrm{Tr}(gF_{\mu\nu})^2 - \frac{4a^6}{72}\,\mathrm{Tr}(gF_{\mu\nu}(4D_\mu^2 + D_\nu^2)gF_{\mu\nu}) + \mathcal{O}(a^8). \tag{8}$$

With the different coefficients for the a^4 and a^6 terms, an improved action can be calculated which is accurate up to third-order:

$$S_{imp} = -\beta \sum_x \sum_{\mu>\nu} \left[\frac{5}{3} P_{\mu\nu} - \frac{1}{12}(R_{\mu\nu} + R_{\nu\mu}) \right]. \tag{9}$$

One last modification to the above action is necessary for an accurate quantum action, that is an action that can accurately describe this system. This is the tadpole improvement, a renormalization μ_0 that accounts for the extra terms in the action. Then equation 9 becomes

$$S_{imp} = -\beta \sum_x \sum_{\mu>\nu} \left[\frac{5}{3\mu_0^4} P_{\mu\nu} - \frac{1}{12\mu_0^6}(R_{\mu\nu} + R_{\nu\mu}) \right] \tag{10}$$

where the μ_0's cancel the tadpole contributions. The μ_0's depend only on the lattice spacing and can be found numerically; for example, for a lattice spacing of $a = 0.4$ fm, then $\mu_0 \approx 3/4$. In this case, the contribution by the rectangle operators is increased by about a factor of 2 compared to the action without the tadpole improvement. With the action in equation 10, Monte Carlo steps can be used to run this model.

1.2 Monte Carlo Application

Again, the steps of Monte Carlo in this context is analogous to before. There are added complications that must be sorted out and, in doing so, increase the computing time appreciably. As stated before, the lattice is four-dimensional to account for three spacial dimensions and one temporal dimension. Furthermore, the calculations for the model exist at the links of which each lattice site has four.

We begin by looking at an NxNxNxN lattice with coordinates (x, y, z, t) at the site x and the link in the $\hat{x}$ direction[2]. There are six plaquettes and twelve rectangle operators that affect the link $U_x(x)$ but for now we will focus only on the plaquettes. Figure 4 shows these plaquettes. The plaquettes are equivalent for each plane that $U_x(x)$ is found on and the yz, yt and zt planes are not included (even though they are part of the sum in equation 10) because they do not contain plaquettes of $U_x(x)$ and therefore do not contribute to the action for this link. So for any link, three of the planes will not contribute. It is important to note that each plane has a contribution of $U_\mu(x)$ and of $U_\mu^\dagger(x)$ for some μ-direction.

The picture for $U_t(x)$ is slightly different. Because $n_t > n_i$ for all $i \in \{x, y, z\}$, the plaquette at site x contains the inverse link of $U_t(x)$. Thus the plaquette at site $x - a\hat{\mu}$ must be used as the plaquette containing $U_t(x)$ as shown in figure 5. For links in the t-direction, this is always true while for links in the x-direction, this is never true. The application of this can be seen in the code in figure 7 which will be discribed shortly.

The rectangle operators are analogous but there is one important difference. Since these operators are $a \times 2a$ and not $a \times a$ as are the plaquettes, there are more contributions to the operator. This can be seen in figure 6. As with the plaquette, the paths in this figure are for the x-direction

[2]In the code, the variable $n \in \{0, 1, 2, 3\}$ is used to determine the specific link. This follows more easily with the sum in the action that is over $\mu > \nu$, thus $n_x = 0 \Leftrightarrow \hat{x}$, $n_y = 1 \Leftrightarrow \hat{y}$, etc.

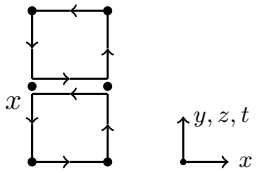

Figure 4: Two plaquettes that effect the link $U_x(x)$ with one being the inverse path or $U_x^\dagger(x)$. Since the lattice is four dimensional, the three planes of x are analogous for calculation of $U_x(x)$.

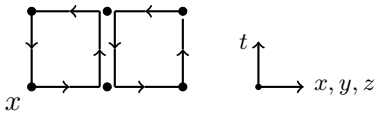

Figure 5: Two plaquettes that effect the link $U_t(x)$. This picture holds for the planes where $\mu > \nu$.

but these paths will not be so for the t-direction. Rather, there is a rotation as before and for the same reason. Also, from equation 10, there are terms $R_{\mu\nu}$ and $R_{\nu\mu}$. Thus there are six rectangle operators, double the number of plaquettes. For $U_x(x)$, there are only two staples for $R_{\nu\mu}$ and four for $R_{\mu\nu}$ and vis versa for $U_t(x)$. The staple being defined as the path operator without the path link in question. To sum up, if $n_\mu > n_\nu$, then $R_{\mu\nu}$ will be a sum of four staples and $R_{\nu\mu}$ will be the sum of two.

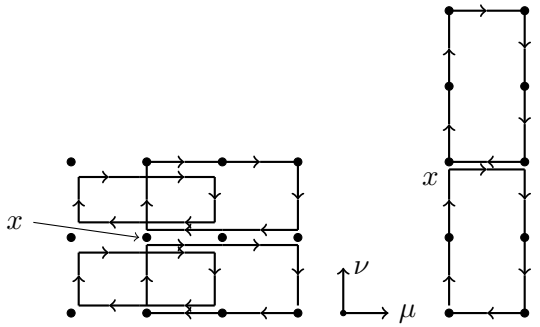

Figure 6: The possible configurations of the rectangle operator. Depending on the direction, the rectangles may be rotated as with the plaquettes. For $U_t(x)$, the rectangles are rotated 90° relative to $U_x(x)$.

In the code in 7, the function `nPlaq` calculates the staple and conjugate staple for the link at lattice site ($x,\ y,\ z,\ t,\ n$) where n specifies the link. The values `staples` and `stapleC` are the sums of the staples and conjugate staples, respectively. They are used to calculate ΔS (along with the rectangle staples for the improved action).

In the previous model, a small random change ϵ was added to a time slice for a path x and the resulting change in the action was calculated. The same is done again except the path links are $SU(3)$ matrices, not numbers, so a 'small' random $SU(3)$ matrix is added to the path link, that is a matrix close to the identity. Thus we take $U \to MU$ if U is a path link and $M \in SU(3)$ is close to the identity. The small change in U is given by the product MU which is also an $SU(3)$ matrix since $SU(3)$ is a group and thus is closed under this operation. This resulting change is used to calculate the change in the action, ΔS.

```
def nPlaq(x, y, z, t, n):
    a = [0] * 4
    a[n] = 1
    staples = 0
    stapleC = 0

    for i in range(3 - n):
        staples += mm(L[m(x+a[0])][m(y+a[1])][m(z+a[2])][t][n+i+1],
                      ct(L[x][m(y+a[-i])][m(z+a[1-i])][m(t+a[2-i])][n]),
                      ct(L[x][y][z][t][n+i+1]))
        stapleC += mm(ct(L[x][m(y-a[-i])][m(z-a[1-i])][m(t-a[2-i])][n+i+1]),
                      L[x][m(y-a[-i])][m(z-a[1-i])][m(t-a[2-i])][n],
                      L[m(x+a[0])][m(y+a[1]-a[-i])][m(z+a[2]-a[1-i])][t-a[2-i]][n+i+1])
    for i in range(n):
        staples += mm(ct(L[m(x-a[i+n])][m(y-a[i+n-1]+a[1])][m(z-a[i+n-2]+a[2])][m(t+a[3])][i]),
                      ct(L[m(x-a[i+n])][m(y-a[i+n-1])][m(z-a[i+n-2])][t][n]),
                      L[m(x-a[i+n])][m(y-a[i+n-1])][m(z-a[i+n-2])][t][i])
        stapleC += mm(L[x][y][z][t][i],
                      L[m(x+a[i+n])][m(y+a[i+n-1])][m(z+a[i+n-2])][t][n],
                      ct(L[x][m(y+a[1])][m(z+a[2])][m(t+a[3])][i]))
    return staples, stapleC
```

Figure 7: Python code for calculating the regular and conjugate transpose staples for $U_\mu(x)$, a link in direction n_μ. The two for loops are to account for the change in the calculation of the staples for each link direction.

Let $\mathcal{S}_{\mu\nu}(C)$ represent the staple(s)[3] for $U_\mu(x)$ in the $\mu\nu$ plane for path C and $\mathcal{S}^*_{\mu\nu}(C)$ be the staple(s) for $U^\dagger_\mu(x)$. Then the change in the unimproved action from equation 4 for site x is

$$\Delta S(x) = \frac{\beta}{3}\,\mathrm{Re}\,\mathrm{Tr}\left[(I-M)U_\mu(x)\sum_{\nu\neq\mu}\mathcal{S}_{\mu\nu}(P_{\mu\nu}) + U^\dagger_\mu(x)(I-M^\dagger)\sum_{\nu\neq\mu}\mathcal{S}^*_{\mu\nu}(P_{\mu\nu})\right] \tag{11}$$

and the improved action is

$$\begin{aligned}\Delta S_{imp}(x) = \frac{\beta}{3}\,\mathrm{Re}\,\mathrm{Tr}\Bigg[&(I-M)U_\mu(x)\left(c_1\sum_{\nu\neq\mu}\mathcal{S}_{\mu\nu}(P_{\mu\nu}) - c_2\sum_{\nu\neq\mu}[\mathcal{S}_{\mu\nu}(R_{\mu\nu}) + \mathcal{S}_{\mu\nu}(R_{\nu\mu})]\right)\\ &+U^\dagger_\mu(x)(I-M^\dagger)\left(c_1\sum_{\nu\neq\mu}\mathcal{S}^*_{\mu\nu}(P_{\mu\nu}) - c^2\sum_{\nu\neq\mu}[\mathcal{S}^*_{\mu\nu}(R_{\mu\nu}) + \mathcal{S}^*_{\mu\nu}(R_{\nu\mu})\right)\Bigg]\end{aligned} \tag{12}$$

where M is a random $SU(3)$ matrix close to the identity, $c_1 = 5/3\mu_0^4$ and $c_2 = 1/12\mu_0^6$. With this in mind, one sweep of the lattice can be completed with the following steps:

1. Choose a site x and a link n.
2. Choose a random $SU(3)$ matrix M such that $U_\mu(x) \to MU_\mu(x)$.

[3]Here I say "staple(s)" because there can be more than one staple coming from the $\mu\nu$ plane like with the rectangle operator (where there can be two). It is implied these are being summed first in the equation which is more useful as a template for visualizing ΔS than a rigorous formula.

3. Calculate the change in action ΔS with equation 12.
4. If $\Delta S < 0$, then keep the change $MU_\mu(x)$.
5. Otherwise assign it a probability $\exp(-\Delta S)$ of changing.
6. Repeat steps 1-5 for every site on the path.

These steps are very similar to those from before, the main changes coming in the equation for the action which is much more costly in this model; the lattice is higher dimensional and the Monte Carlo steps require more computations and are computed on matrices. In figure 8, equation 12 is implemented under `dS` where the matrices staples and stapleC are calculated as shown in figure 7 and `rects` and `rectC` are calculated in a similar but longer function `nRect`. The $SU(3)$ matrices used as the small Monte Carlo variable are generated beforehand and stored in `SU3list` along with their inverses.

```
def dS(staples, stapleC, rects, rectC, x, y, z, t, n):
    M = SU3list[np.random.randint(0, 2 * numMatrices)]
    U = L[x][y][z][t][n]
    deltaS = (beta / 3) * np.real(np.trace(np.dot(U - mm(M, U), c1 * staples - c2 * rects) + \
                                    np.dot(ct(U) - ct(mm(M, U)), c1 * stapleC - c2 * rectC)))
    return deltaS, M

def sweep():
    for p in product(range(N), range(N), range(N), range(N), range(4)):
        x, y, z, t, n = p
        staples, stapleC = nPlaq(x, y, z, t, n)
        rects, rectC = nRect(x,y,z,t,n)
        for _ in range(N_ma):
            deltaS, M = dS(staples, stapleC, rects, rectC, x, y, z, t, n)
            if deltaS < 0 or np.random.random() < np.exp(-deltaS):
                L[x][y][z][t][n] = np.dot(M, L[x][y][z][t][n])
```

Figure 8: The code that evolves the lattice. The array `L` is the lattice with $N^4 * 4$ sites given N as the side length of the lattice and an $SU(3)$ matrix at each site with the total length of `L` being $36 * N^4$. The function `sweep` completes a sweep of the lattice and `dS` calculates the change in the action. At each site $(x,\ y,\ z,\ t,\ n)$, the Metropolis algorithm is done `N_ma` times so as to let the link variable reach an equilibrium with its neighbors before moving on to the next link.

1.3 Measurements

With the derivation of the path integral in mind, we precede with the gluonic path integral similarly with $N = 8$ as a lattice size[4]. We can measure the effect of the Wilson loop operator on the vacuum state and, with the choice of Wilson loop, the measurement will change. First, the lattice is thermalized, as defined earlier, by completing 100 sweeps and then measurements are taken every $N_{cor} = 50$ sweeps. The trace of the Wilson loops for each path link are summed and averaged. In figure 9, the convergence of the Wilson loops are shown. After 100 sweeps, the lattice has reach equilibrium and 50 sweeps is sufficiently long enough to prevent correlation in the measurements.

[4]Even though this lattice size is much smaller than the previous, this simulation requires hours, compared to minutes, to complete 25 measurements with 50 sweeps between measurements. All code was run in Python 3.x on

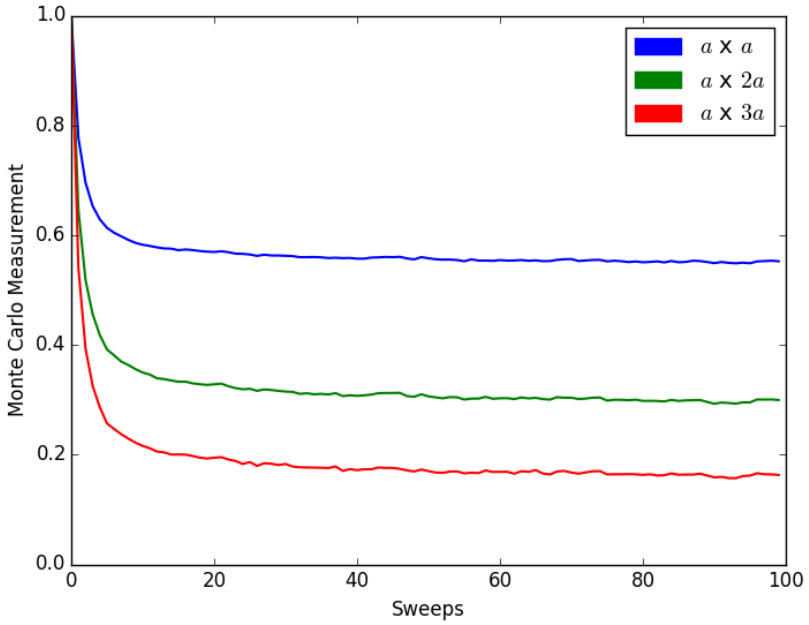

Figure 9: A visualization of the convergence rates of the Monte Carlo measurements. The first 100 sweeps, which are plotted, are used for reaching the equilibrium of the lattice and so measurements are not completed during these sweeps.

The Monte Carlo averages of $a \times a$, $a \times 2a$ and $a \times 3a$ Wilson loops are calculated as shown in the code in figure 10. In general, measurement of an arbitrary Wilson loop W requires only a change in the product of the link variables.

The averages are also calculated for both the unimproved and improved actions:

$$S(x) = \frac{\beta}{\mu_0^4} \sum_{\mu>\nu} P_{\mu\nu}(x) \qquad S_{imp}(x) = -\beta \sum_{\mu>\nu} \left[\frac{5}{3\mu_0^4} P_{\mu\nu} - \frac{1}{12\mu_0^6} (R_{\mu\nu} + R_{\nu\mu}) \right]. \tag{13}$$

This allows us to visual the difference in the second- and third-order accuracy of the discretized action, i.e. the contribution of the rectangle operators. For the unimproved and improved actions, the averages, denoted by $\mathcal{W}$ and $\mathcal{W}_{imp}$, respectively, are

$$\begin{aligned} \mathcal{W}(a \times a) &= 0.497 \pm 0.003 & \mathcal{W}_{imp}(a \times a) &= 0.546 \pm 0.003, \\ \mathcal{W}(a \times 2a) &= 0.260 \pm 0.004 & \mathcal{W}_{imp}(a \times 2a) &= 0.290 \pm 0.004, \\ \mathcal{W}(a \times 3a) &= 0.137 \pm 0.004 & \mathcal{W}_{imp}(a \times 3a) &= 0.156 \pm 0.003. \end{aligned} \tag{14}$$

These are the unweighted means for the respective Wilson loops as shown in figure 10. These results suggest that the larger the area of the Wilson loop, the smaller $\mathcal{W}$ will be.

With the system modeled based on the assumptions of the system, the measurements are relatively trivial in both computational time and in the complexity of the code. Further measurements can be made on this lattice or more realistic properties can be assumed for the lattice, although the latter may increase the computing time significantly due to its added complexity. Nevertheless, as with the previous models, the potential for added measurements exists.

an Intel i7 processor on Windows 10.

```
def measure(loopLength):
    LAvg = 0
    for p in product(range(N), range(N), range(N), range(N)):
        x, y, z, t = p
        link = measureLink(x, y, z, t, loopLength)
        LAvg += np.real(np.trace(
        mm(L[x][y][z][t][0], L[m(x+1)][y][z][t][1], link[0], ct(L[x][y][z][t][1])) +
        mm(L[x][y][z][t][0], L[m(x+1)][y][z][t][2], link[1], ct(L[x][y][z][t][2])) +
        mm(L[x][y][z][t][0], L[m(x+1)][y][z][t][3], link[2], ct(L[x][y][z][t][3])) +
        mm(L[x][y][z][t][1], L[x][m(y+1)][z][t][2], link[3], ct(L[x][y][z][t][2])) +
        mm(L[x][y][z][t][1], L[x][m(y+1)][z][t][3], link[4], ct(L[x][y][z][t][3])) +
        mm(L[x][y][z][t][2], L[x][y][m(z+1)][t][3], link[5], ct(L[x][y][z][t][3]))))
    return LAvg / N**4 / 18
```

Figure 10: Python code that measures an $a \times ca$ Wilson loop for $c \in \{1, 2, 3\}$. The array `link` determines specific links in the loop based on `loopLength`. The returned value is the normalized average.

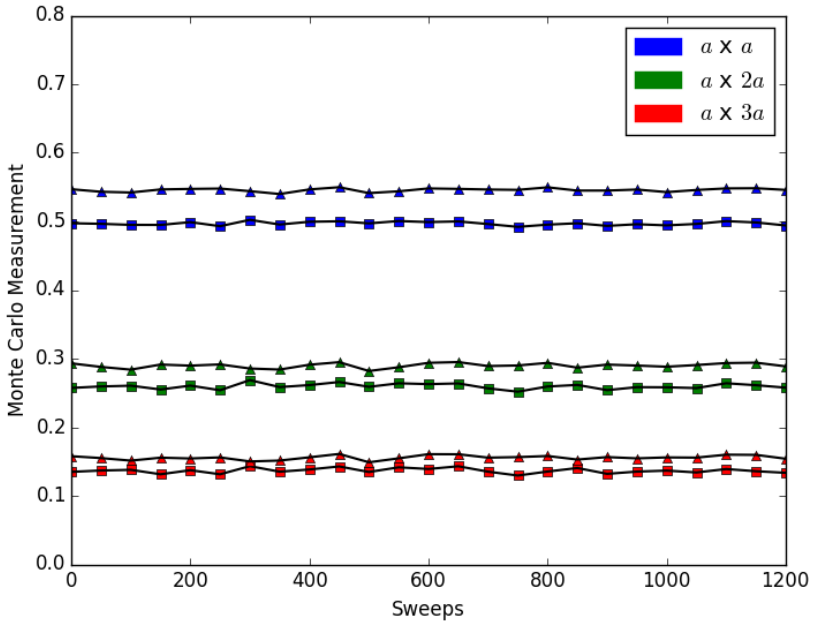

Figure 11: Monte Carlo measurements for $a \times a$, $a \times 2a$ and $a \times 3a$ Wilson loops. The triangles represent the improved action from equation 10 while the squares represent the unimproved action from equation 4. Between each measurement, $N_{cor} = 50$ sweeps were omitted along with 100 sweeps after the lattice was initialized for it to reach equilibrium.