Determining the String Tension Using QCD Lattice Simulations

Author: Piter Annema
Supervisor: Prof. dr. Elisabetta Pallante

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science
in the
Van Swinderen Institute
Rijksuniversiteit Groningen

July 19, 2019
Quantum Chromodynamics is the leading theory for describing the strong interaction, but some analytic solutions have proven to be difficult to find. Lattice QCD is a numerical approach of solving the theory. By discretizing the theory and placing it on a lattice a very effective means of simulating quarks and gluons can be found. In this thesis we describe the building blocks and techniques used to develop lattice QCD simulation software from the ground up. In order to reduce simulation times certain aspects of the software were parallelized, making it suitable for deployment on a supercomputer. This software is then used to determine the static quark potential and the string tension which confirms an important property of QCD called color confinement.
Acknowledgements

I would like to thank prof. dr. Pallante for her supervision and guidance as well as the meetings which helped me improve my understanding of the theory and interpret results from the simulations.

Additionally I would like to thank Jelle Bor for introducing me to the subject and guiding me through the project. All the work I present in this thesis has been made in close collaboration with Jelle.
Contents

Abstract i

Acknowledgements ii

1 Introduction 1

2 Theoretical Framework 2
  2.1 Quantum Chromodynamics 2
    2.1.1 The Continuum QCD Action 2
    2.1.2 The SU(3) Group 3
  2.2 Monte Carlo Algorithms 3
    2.2.1 Markov Chain 3
    2.2.2 Metropolis Algorithm 3

3 Lattice Simulations 5
  3.1 Structure of the Lattice 5
  3.2 Navigating the Lattice 5
    3.2.1 The Path Operator 6
    3.2.2 The Loop Operator 6
    3.2.3 The Plaquette Operator 7
  3.3 Updating the Lattice 7
    3.3.1 Metropolis Monte Carlo on a Lattice 8
    3.3.2 Improved Action 9
    3.3.3 Finding Relevant Configurations 10
  3.4 Wilson Loops 11
  3.5 Measuring the String Tension 12
    3.5.1 Finding more values of r 13
  3.6 Parallelizing Lattice Updates 14
    3.6.1 The Slice Masking Pattern 15

4 Concluding Remarks 17

Bibliography 18
Chapter 1

Introduction

Since its inception about four decades ago, lattice QCD has been the leading theory for predicting the behavior of quarks and the strong nuclear force. By discretization of continuum space-time, where the quantum field theory is defined, and placing the fields on this lattice, numerical solutions can be produced. These numerical methods have proved to be very successful in predicting and verifying the behavior of quarks and gluons. However, software for running such lattice QCD simulations is hard to come by, and software that is available in this field is often either very difficult to use or built for very specific environments. Thus, the need to develop such a program ourselves arose. In the past months we have endeavoured to do so and in this thesis we present the results as well as the basic principles of lattice QCD.

First we will briefly discuss the basics of the continuum theory of quantum chromodynamics. We will treat some aspects of the theory that will return later on in the discretized case. One of the most prominent features of QCD is color confinement, which suggests that quarks or gluons cannot be isolated. This phenomenon is traceable in lattice QCD from the linear nature of the static quark potential as a function of spatial separation of quarks and it is in agreement with experimental results. The slope of this linear potential is called the string tension.

The Monte Carlo class of algorithms forms a central aspect in numerical simulations across many fields of physics. A special kind of Monte Carlo algorithm, the Metropolis Monte Carlo algorithm turns out to be very suitable for generating lattices which are suitable for measuring physical quantities on. A short overview of this algorithm will be provided in a general context in preparation to its application on the lattice.

Having established the basics of the QCD continuum theory we will proceed to formulate the discretized theory of QCD on a lattice in chapter 3. There we will discuss the general structure of such a lattice and define what is known as the link variable. This link variable will form the main building block for all observables on the lattice we used. With these observables we will then go back to the aforementioned Metropolis Monte Carlo algorithm and employ it to generate lattice configurations. After generating said lattice configurations we can start measuring physical observables. Among them is the static quark potential, leading us to the string tension and the probe of color confinement of the strong interaction.

Finally we will discuss the parallelization of certain processes in the simulation software. By parallelizing these processes we can utilize the full potential of CPU’s, which in turn gives us the possibility to run simulations on supercomputers. This allows us to generate larger lattices with less noise. But applying such techniques comes with the necessary challenges described in section 3.6.
Chapter 2

Theoretical Framework

In this chapter we will discuss some basic aspects of Quantum Chromodynamics and Yang-Mills theory. An integral part of the theory is the SU(3) symmetry group. The elements of this group form the building blocks of the lattice and their properties are leading in finding the behavior of quarks and gluons. Since the main focus of this thesis is the implementation of QCD on a discrete lattice, the theory in the continuum will be mentioned but not deeply explored. Some fundamental equations of the theory will be stated as-is without further derivation, but their discretized counterparts will be discussed in greater detail in later chapters.

Finally we will discuss the class of stochastic methods called Monte Carlo methods used to generate different lattice configurations.

2.1 Quantum Chromodynamics

Quantum Chromodynamics (QCD) is a quantum field theory which describes the strong interactions between quarks and gluons. The strong interaction is the strongest of the four fundamental interactions. On the subnuclear scale the strong interaction is carried by the gluon which governs the binding force between quarks within a nucleon. QCD is a non-abelian gauge theory based on the SU(3) symmetry group.

2.1.1 The Continuum QCD Action

We start by defining the QCD action which consists of two parts: the fermionic QCD action describing the quarks, and the pure gauge (gluon) action. These parts of the action are defined by the quark field:

\[ \psi(x), \bar{\psi}(x) \]  

And the gauge field for the gluons:

\[ A_\mu(x) \]

We start by defining the fermionic QCD action in Euclidean space-time.

\[ S_F[\psi, \bar{\psi}, A] = \sum_{f=1}^{N_f} \int d^4x \bar{\psi}^{(f)}(x) \left( \gamma_\mu D_\mu + m^{(f)} \right) \psi^{(f)}(x) \]

Here we sum over the flavors \((f)\) of the quarks and \(D_\mu = \partial_\mu + igA_\mu(x)\) is the covariant derivative. In Euclidean space-time we have for the Dirac matrices anti-commutation relation:

\[ \{\gamma_{\mu}, \gamma_{\nu}\} = 2\delta_{\mu\nu} \mathbb{1} \]
We define the gluonic part of the action as

\[ S_G = \int d^4x \frac{1}{4} F_{\mu\nu}^a F_{\mu\nu}^a \]  

(2.5)

with the field strength tensor

\[ F_{\mu\nu}^a = \partial_\mu A_\nu^a - \partial_\nu A_\mu^a + ig f^{abc} A_\mu^b A_\nu^c \]  

(2.6)

Here, \( f^{abc} \) represent the structure constants of the SU(3) group.

### 2.1.2 The SU(3) Group

As mentioned above, quantum chromodynamics is based on the non-abelian symmetry group SU(3). We represent the group elements with \( 3 \times 3 \) hermitian matrices denoted by \( \Omega(x) \) with matrix multiplication as the group operation. Often written as \( \Omega_{\text{color}}(x) \) in QCD these transformations are rotations of color indices. Note the use of the label \( x \); we are dealing with a local symmetry. The most important properties of SU(N) can be found in the name of the group:

- **Special:** the elements of the group have \( \det \Omega(x) = 1 \).

- **Unitary:** the inverse of an element is its hermitian conjugate: \( \Omega(x)\Omega^\dagger(x) = 1 \).

### 2.2 Monte Carlo Algorithms

Monte Carlo methods are algorithms which employ random variables drawn from specific distributions to numerically solve problems. Problems especially suited for these methods are those which are analytically difficult or impossible to solve.

#### 2.2.1 Markov Chain

A Markov Chain is a stochastic process of which the transition to the next state exclusively depends on the current state, a property called *memorylessness*. If we take a sequence of states \( (x_1, x_2, \ldots, x_{n-1}, x_n) \) then the probability of transitioning from a state \( x_i \) to the next state \( x_j \) is determined by some probability function which *only* depends on the current state \( x_i \). If this probability function is chosen correctly, the system will reach some equilibrium. This allows us to use systems of which this equilibrium state cannot be easily determined. The QCD lattice is an example of such a system, and it is perfectly suitable to be modeled as a Markov chain. Each subsequent state of the lattice is only determined by the current state where the probability of moving to the next one is governed by a physical quantity, the *Wilson gauge action*. The algorithm used to determine these state changes is called the *Metropolis algorithm* which makes sure that the lattice Wilson action reaches the promised equilibrium.

#### 2.2.2 Metropolis Algorithm

The Metropolis-Hastings algorithm is a Markov Chain Monte Carlo (MCMC) algorithm for determining the probability of state changes in a Markov chain. The main feature of this algorithm is that it can draw samples from an unknown probability distribution \( P(x) \) which will prove to be very useful in the context of lattice QCD. Suppose that there is a probability function \( f(x) \) which is known to be proportional
to $P(x)$, this function can then be used to indirectly sample from $P(x)$ using the following algorithm:

- We start with a state $x$, using some probability density $g(x'|x)$ we propose a next candidate state $x'$.$^1$

- Next we determine what is called the acceptance ratio $\alpha$:

$$\alpha = \frac{f(x')}{f(x)} = \frac{P(x')}{P(x)} \quad (2.7)$$

We know that the latter equality must hold because $f \propto P$.

- Now we pick a random number $\eta$ from a uniform distribution between 0 and 1. If $\eta \leq \alpha$ we accept the new state $x'$, otherwise we reject it and keep the original state $x$.

If we repeat the aforementioned steps sufficiently many times, the distribution of states $(x_1, x_2, \ldots, x_{n-1}, x_n)$ will approach the desired distribution $P$.

---

$^1$For the Metropolis algorithm, this probability density must be symmetric, i.e. $g(x'|x) = g(x|x')$. 
Chapter 3

Lattice Simulations

In this chapter we will show how exactly the lattice is structured and how it can be traversed in a meaningful way. From this we produce several operators and explore their behavior under gauge transformations.

3.1 Structure of the Lattice

The lattices used in QCD are four-dimensional space-time hypercubes of evenly spaced points. Three of these dimensions represent the spatial dimensions while the fourth represents the temporal dimension. For the sake of simplicity we always assume that the size of the lattice is the same for each spatial dimension but may differ in the temporal direction. Thus we denote the size of a lattice as $R^3 \times T$. The points on the lattice, or sites, are represented by $x$ or $y$. This is where the massive, static quarks reside. They are connected by the gluon field, which lives on the lines in between these sites.

3.2 Navigating the Lattice

A step from a site $x$ along the $\hat{\mu}$ direction is represented by a link variable $U_{\mu}(x)$. This link variable is a $3 \times 3$ special unitary matrix of SU(3) which transforms under a gauge transformations as:

$$U_{\mu}(x) \rightarrow U'_{\mu}(x) = \Omega(x)U_{\mu}(x)\Omega^\dagger(x+\hat{\mu})$$

(3.1)

The exact opposite step (from $x + \hat{\mu}$ to $x$) can also be taken and is represented by the hermitian conjugate of the link variable. Taking a step from $x$ in the $-\mu$ direction we find that this step is represented by the hermitian conjugate of the link variable belonging to the site we land on.

$$U_{-\mu}(x) \equiv U_{\mu}^\dagger(x - \hat{\mu})$$

(3.2)

Similarly to (3.1), the latter transforms as

$$U_{-\mu}^\dagger(x - \hat{\mu}) \rightarrow U_{-\mu}^\dagger(x - \hat{\mu}) = \Omega(x)U_{-\mu}(x - \hat{\mu})\Omega^\dagger(x - \hat{\mu})$$

(3.3)

under gauge transformations. These results form the basis for finding gauge-invariant operators on the lattice, some of which will be defined in the following sections.
3.2.1 The Path Operator

By stringing link variables together as a path \( P \) between two points \( x, y \) on the lattice we find the path operator \( P \):

\[
P = \prod_{(x,\mu) \in P} U_\mu(x) \quad (3.4)
\]

In the continuum we have a similar object called the gauge transporter, which connects two points \( x \) and \( y \) by a path-ordered (\( P \)) integral along the curve \( C \):

\[
G(x, y) = P e^{i \int_C A \cdot ds} \quad (3.5)
\]

By using (3.1) and (3.3) we can find the gauge transformation of \( P \). If we take two subsequent link variables somewhere along the path, say \( U_\mu(x) \) and \( U_\nu(x + \hat{\mu}) \) with \( \mu, \nu = 1, 2, 3, 4 \) we find that they transform as:

\[
U_\mu(x)U_\nu(x + \hat{\mu}) \rightarrow \Omega(x)U_\mu(x)\Omega\dagger(x + \hat{\mu})U_\nu(x + \hat{\mu})\Omega\dagger(x + \hat{\mu} + \hat{\nu}) \quad (3.6)
\]

Considering that \( \Omega(x + \hat{\mu})\Omega\dagger(x + \hat{\mu}) = 1 \) since the gauge transformation matrices are elements of the SU(3) group; equation (3.6) reduces to:

\[
U_\mu(x)U_\nu(x + \hat{\mu}) \rightarrow \Omega(x)U_\mu(x)\Omega\dagger(x + \hat{\mu})U_\nu(x + \hat{\mu} + \hat{\nu}) \quad (3.7)
\]

This is true for all subsequent link variables in a path, thus we find that all transformation matrices but the ones at both ends of a path cancel, yielding the gauge transformation for the path operator:

\[
P \rightarrow P' = \Omega(x)P\Omega\dagger(y) \quad (3.8)
\]

where \( P \) is defined by the set of link variables \( P \) connecting the sites \( x \) and \( y \).

![Figure 3.1: Example of a path operator \( P \) defined by path \( P \) connecting the points \( x \) and \( y = x + 2\hat{\mu} \)](image)

3.2.2 The Loop Operator

A special kind of path operator is one where we end up on the same site as where we started. In other words, we define a path operator along a closed loop \( \mathcal{L} \) such that it transforms as:

\[
P_\mathcal{L} \rightarrow P'_\mathcal{L} = \Omega(x)P_\mathcal{L}\Omega\dagger(x) \quad (3.9)
\]

Taking the trace and considering the cyclic property of the trace we find that:

\[
L = \text{Tr} P_\mathcal{L} \rightarrow L' = \text{Tr} \left[ \Omega(x)P_\mathcal{L}\Omega\dagger(x) \right] = \text{Tr} \left[ P_\mathcal{L}\Omega\dagger(x)\Omega(x) \right] = \text{Tr} P_\mathcal{L} = L \quad (3.10)
\]

Thus we have found a gauge-invariant operator which will be used to derive two types of quantities on the lattice, the Wilson gauge action and the Wilson Loop.
Note that the shortest loop is a step in some direction $\hat{\mu}$ followed by a step back. Since all link variables are unitary by definition, we get:

$$U_\mu(x)U_\mu^\dagger(x) = 1$$  \hspace{1cm} (3.11)

### 3.2.3 The Plaquette Operator

The shortest loop that the lattice besides the trivial loop (3.11) is defined by the path along a unit square in some plane $\mu \neq \nu$:

$$P_{\mu\nu}(x) = U_\mu(x)U_\nu(x + \hat{\mu})U_\nu^\dagger(x + \hat{\nu})U_\mu^\dagger(x)$$  \hspace{1cm} (3.12)

If we define the plaquette to run in the counter-clockwise order as depicted in figure 3.2, a clockwise counterpart can be constructed in a similar fashion.

$$P_{\mu\nu}^{cw}(x) = U_\nu(x)U_\mu(x + \hat{\nu})U_\mu^\dagger(x + \hat{\mu})U_\nu^\dagger(x)$$  \hspace{1cm} (3.13)

It is easily shown that $P_{\mu\nu} = P_{\nu\mu}^{cw}$. If we want to evaluate all plaquettes only once we must impose the restriction that $\mu < \nu$. Considering this restriction we define the Wilson gauge action:[1, 2]

$$S_G = \frac{\beta}{4} \sum_x \sum_{\mu < \nu} \text{Re} \text{Tr}[1 - P_{\mu\nu}(x)]$$  \hspace{1cm} (3.14)

Where $\beta = 6/g^2$ determines the coupling strength on the lattice. If we take the limit $a \rightarrow 0$ we find:

$$S_G = -\int d^4x \frac{1}{4g^2} F_{\mu\nu}^{a} F_{\mu\nu}^{a} + O(a^2)$$  \hspace{1cm} (3.15)

Here we see that we get back the continuum gluon action from equation (2.4) with an error of $O(a^2)$.

### 3.3 Updating the Lattice

Now we have defined all the building blocks of the lattice, we can start using the aforementioned Monte-Carlo methods to generate lattice configurations suitable for measurement. As described in section 2.2.2 we want to bring the lattice to some state of equilibrium driven by the lattice gauge action. In the context of link variables we can modify our Metropolis algorithm to work on the lattice. But to do so we first need to define how we can apply small updates to the lattice and measure the effects on the Wilson gauge action in equation (3.14).
3.3.1 Metropolis Monte Carlo on a Lattice

We start by picking a link variable $U_\mu(x)$ at some site $x$ in the $\mu$ direction. Next we want to propose a candidate link $U'_\mu(x)$ which differs slightly to replace the original link. To find this candidate we define a matrix $M$ which is numerically close to identity such that:

$$U'_\mu(x) = MU_\mu(x)$$  \hspace{1cm} (3.16)

This matrix must ensure that the proposed candidate remains an SU(3) matrix. Following from the group axiom of closure, the matrix $M$ itself then must be a member of this group. We generate a random $3 \times 3$ hermitian matrix $H$ with both real and complex components between 1 and $-1$. Next we define:

$$M^* = \sum_{k=0}^{n} \frac{(i + \epsilon)^k}{k!} H^k \approx e^{i\epsilon H}$$  \hspace{1cm} (3.17)

Which is a Taylor expansion up to order $n$ of the exponential, where $\epsilon$ is determines the strength of the change we want to apply. This forms a unitary matrix, but it is not yet special ($\det M^* \neq 1$) which is easily resolved:

$$M = \frac{M^*}{\det(M^*)^{\frac{1}{2}}}$$  \hspace{1cm} (3.18)

Thus we have found a random SU(3) matrix which represents a small change on a link matrix. Once we have replaced the link matrix $U_\mu(x)$ with our candidate we need to measure the change in action $\Delta S$ in order to decide whether to keep the candidate link variable. Let’s define $S'_G$ as the Wilson gauge action (3.14) after the link variable has been replaced such that:

$$\Delta S = S'_G - S_G$$  \hspace{1cm} (3.19)

It is not difficult to see that the sums over all the plaquettes cancel everywhere but in the direct neighborhood of $U_\mu(x)$. As shown in figure 3.3, the only plaquettes that contribute are those that include the altered link variable.

Now let’s define the path operator $\Gamma_{\mu\nu}(x)$ as the three links which complete the plaquette with $U_\mu(x)$ in the $\mu\nu$-plane:

$$\Gamma_{\mu\nu}(x) = U_\nu(x + \hat{\mu})U_\mu^\dagger(x + \hat{\nu})U_\nu^\dagger(x)$$  \hspace{1cm} (3.20)

As mentioned, we now get that $P_{\mu\nu} = U_\mu(x)\Gamma_{\mu\nu}(x)$. Since only $U_\mu(x)$ changes in this expression when proposing an update candidate it is computationally favorable.
Chapter 3. Lattice Simulations

to only recalculate the link variable after each proposal. The conjugate transpose of
the proposed link variable has a similar corresponding path $\Gamma^\mu_{\nu}(x)$ which completes
the plaquette $P_{\mu\nu}(x - \hat{\nu})$. By taking the differences between these plaquettes with
the updated plaquettes we can easily find the impact of the proposed change from
equations (3.14) and (3.19):

$$\Delta S(x, \mu) = \beta \sum_{\mu \neq \nu} \text{Re} \text{Tr} \left[ \left( U^\mu_\mu(x) - U'^\mu_\mu(x) \right) \Gamma^\mu_{\nu}(x) + \left( U'^\mu_\mu(x) - U^\mu_\mu(x) \right) \Gamma^*_{\mu\nu}(x) \right]$$

(3.21)

By using equation (3.16) we find that this expression can be written as:

$$\Delta S(x, \mu) = \beta \sum_{\mu \neq \nu} \text{Re} \text{Tr} \left[ (1 - M)U^\mu_\mu(x) \Gamma^\mu_{\nu}(x) + (1 - M^\dagger)U'^\mu_\mu(x) \Gamma^*_{\mu\nu}(x) \right]$$

(3.22)

Now we need some probability function to determine whether we accept this new
state. In lattice QCD we use the gauge action to determine whether we will change
state. Let’s propose a probability function based on the lattice state:

$$f = e^{-S_G} \propto P$$

(3.23)

Then we find the acceptance ratio from equation (2.7) to be:

$$\alpha = \frac{f'}{f} = \frac{e^{-S'_G}}{e^{-S_G}} = e^{-(S'_G - S_G)} = e^{-\Delta S}$$

(3.24)

Now that we have established how we propose new states and how we accept them,
we can reformulate the Metropolis algorithm in the context of lattice QCD:

- We take a link variable $U^\mu_\mu(x)$ on the lattice at site $x$ in the $\hat{\mu}$ direction and
propose the candidate $U'^\mu_\mu(x) = MU^\mu_\mu(x)$.

- We determine $\Delta S$ using equation (3.22). From this we determine the acceptance
ratio:

$$\alpha = e^{-\Delta S}$$

(3.25)

- Finally we determine a random value $\eta$ between 0 and 1.
If $\eta < \alpha \leq 1$ we accept the proposed update. Otherwise we reject it and return
the lattice to the original state.

For each link variable on the lattice, this set of steps is repeated multiple times (typ-
ically about 10 times) to increase the likelihood of finding a suitable candidate.

3.3.2 Improved Action

As shown in equation (3.14) the Wilson gauge action gives an approximation of the
continuum gluon action in equation (2.4) with an error of $O(a^2)$. We can improve our
action by including longer paths around the link variables. In doing so we can rewrite
the gauge action in equation (3.14) as follows:

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

(3.26)

Where $R_{\mu\nu}$ is a $1a \times 2a$ rectangle operator similar to $P_{\mu\nu}$. In this case we must
distinguish between the rectangle operators $R_{\mu\nu}$ and $R_{\nu\mu}$ since switching the plane
indices results in a $90^\circ$ rotation of the rectangle. Furthermore, by moving the $R_{\mu\nu}(x)$ operator one step over in the $-\hat{\mu}$ direction we get another set of rectangles. Thus we end up with three ways of placing rectangles such that they include the link variable. Just as with the plaquettes in the first term, we include the inverse of the link variable, thus doubling the number of rectangle operators used in the action measurement.

It is important to note that although this improvement significantly reduces lattice artifacts as a result of the MMC procedure and non-zero lattice spacing, the number of link multiplications needed to perform the measurement has also increased considerably. As a result, utilizing this improvement technique severely impacts the computational cost of updating the lattice.

![Figure 3.4: Example of two $a \times 2a$ rectangular paths $R_{\mu\nu}(x)$ and $R_{\mu\nu}(x - \hat{\nu})$ around the link variable $U_\mu(x)$](image)

### 3.3.3 Finding Relevant Configurations

After updating all link variables at every site, the lattice is said to have completed a sweep. By monitoring the average plaquette value defined as

$$P_{\text{average}} = \frac{1}{6 \cdot n_{\text{sites}}} \sum_{x} \sum_{\mu<\nu} \text{Re} \text{Tr} \ P_{\mu\nu}$$  \hspace{1cm} (3.27)

for each sweep, we can determine when the lattice link distribution has reached equilibrium.\(^1\) In figure 3.5 we can see such average plaquette values for two lattices of different sizes. From this figure we can infer several properties of the behavior of the lattice under MMC updates. The first observation is that the plaquette average asymptotically approaches a certain value. This value is independent of lattice size but larger lattices experience less noise from the MMC procedure. The first set of configurations (not thermalized) are not suitable for measurement and need to be discarded. Thus before we begin using the lattice it must go through a process called thermalization in which the lattice is brought to equilibrium by sweeping $n_{\text{therm}}$ times. Typical values for $n_{\text{therm}}$ range from several hundreds to thousands of sweeps.

Another thing to note in figure 3.5, and what is especially apparent for the $4^3 \times 4$ lattice, is that configurations tend to be correlated. In other words, subsequent configurations often have plaquette averages that lie close to each other. This negatively impacts the precision of measurements on the lattice. By skipping a sufficient number of configurations, often denoted as $n_{\text{cor}}$, between each measurement we ensure that

\(^1\)Note that we do not only average over the number of sites in the lattice, but also over the number of plaquettes we can form at each lattice point. This number is equal to the number of unique index pairs $\mu, \nu = 1, 2, 3, 4$ we can form such that $\mu < \nu$. This, is the binomial coefficient $4 \choose 2$, or $\binom{4}{2} = 6$. 

3.4 Wilson Loops

As shown before, all physical observables on the lattice must be gauge-invariant. Besides the plaquette operator, any closed loop on the lattice satisfies this property and can represent some physical quantity. A special type of a closed loop is the Wilson loop, first introduced by Kenneth G. Wilson in 1974. [2] Wilson loops are built from four components. Two of these components $S(x, y, 0)$ and $S(x, y, t)$ are paths which in the spatial dimensions, i.e. they connect two points $x$ and $y$ which lie on the same time slice at separate spatial coordinates. These are so called Wilson lines:

$$S(x, y, t) = \prod_{(k, \mu) \in \mathcal{C}_{x, y}} U_{\mu}(k, t)$$  \hspace{1cm} (3.28)

This line and the analogous one, $S(x, y, 0)$ at $t = 0$, are separated in time but take identical paths in the spatial dimensions. In order to produce a loop we need to connect these lines through the temporal dimension. To do this we take two paths $T(y, t)$ and $T(x, t)$ the so-called temporal transporters, which as the name suggests,
only traverse in the temporal dimension.\footnote{In equation (3.29) we take $\mu = 4$ as the index for the temporal dimension.}

\[ T(x, t) = \prod_{\tau=0}^{t-1} U_4(x, \tau) \quad (3.29) \]

By stringing these paths together such that a closed loop is formed,\footnote{Note that when the conjugate transpose ($\dagger$) of these paths is taken, due to the hermicity of these paths, the order of the underlying link variables is reversed accordingly.} and taking the trace we obtain the Wilson loop:

\[ W_L = \text{Tr} \left[ S(x, y, 0)T(y, t)S^\dagger(x, y, t)T^\dagger(x, t) \right] \quad (3.30) \]

\[ W_L(r, t) = \text{const} \ e^{-V(r)t} \quad (3.31) \]

This brings us to the static-quark potential $V(r)$.\footnote{In equation (3.31) we take $\mu = 4$ as the index for the temporal dimension.}

\[ W_L(r, t)/W_L(r, t + a) = e^{aV(r)} \quad (3.32) \]

This result allows us to measure the static-quark potential using Wilson loops. However, the problem is that our lattice is of limited size in the temporal dimension, so stretching Wilson loops to large enough values for $t$ is computationally expensive. Furthermore, at large $t$ the fluctuations from the Monte Carlo algorithm become dominant and the errors in this ratio become unwieldy. If instead we take the natural logarithm of the ratio in equation (3.32) we get $aV(r)$ for the large $t$ limit.

Thus we should find a linear relation between $W_L(r, t)$ and the temporal separation $t$. By fitting these points against the corresponding values of $t$ the static-quark potential is determined from the slope of the fit. If we exclude Wilson loops with small
temporal separation, our measurements become more representative of the large $t$ limit.

This static-quark potential is expected to have coulombic behavior for small spatial separations, i.e. $V \propto \frac{1}{r}$ for small $r$. For larger spatial separation we expect the static-quark potential to increase linearly with $r$. The slope of this linear relation is called the string tension $\sigma$:

$$V \propto \sigma r$$ (3.33)

\[\begin{align*}
\text{Figure 3.7: Linear fit on } \ln W_L(r, t) \text{ for } r = 1, \sqrt{2}, 2 \text{ and } t \geq 1 \text{ on a } 6^3 \times 12 \text{ lattice.}
\end{align*}\]

### 3.5.1 Finding more values of $r$

\[\begin{align*}
\text{Figure 3.8: Examples of spatial paths which span a non-integer euclidean distance } r \text{ on a lattice with } R = 3.
\end{align*}\]

Because of the discrete nature of the lattice, the number of different values of $r$ we can use to measure the string tension is severely limited. There is however a way of finding values for non-integer spatial separations. This is because it is the euclidean separation
that actually counts. This, and the fact that we have three spatial dimensions to our disposal allows us to significantly increase the number of measurements.

By combining these measurements of the static-quark potential for all available spatial separations, the string tension can be found with a linear fit on the relevant points. The method for determining the static-quark potential appears to be effective on small lattices and produces similar results to Lepage [5].

\[
\begin{align*}
\text{Figure 3.9: Static-quark potential and string tension measured on a } & 10^3 \times 18 \text{ lattice.}
\end{align*}
\]

### 3.6 Parallelizing Lattice Updates

Even though lattice QCD was introduced many decades ago, it’s effectiveness only became apparent recently. One of the main reasons for this is the extreme computational load that the technique produces. With the dawn of supercomputers along with increasingly fast personal computers, developments in lattice QCD have skyrocketed over the last 20 years. The availability of sufficiently performant personal computers has allowed physicists in this field to rapidly develop and prototype new software and computational models. Moreover, supercomputers have opened the doors to increasingly large, precise and low noise lattice measurements. One of the main advantages of modern computers is their ability to concurrently execute several different tasks. This is made possible by the availability of multiple processing cores on a CPU or in turn multiple separate CPU’s. Tasks that involve repeated execution of an algorithm with different inputs are especially suitable for parallelization. If one succeeds in efficiently distributing a task over \( n_c \) cores, the theoretical minimum processing time is reduced by a factor \( n_c \). Of course, this reduction cannot be reached as the distribution of tasks is itself a computationally expensive process. This involves reserving memory
for the separate processes, sending input and receiving output. The latter of which are often independent of \( n_c \). Because of this overhead, which in the Python language is exceptionally large, it is important to match the number of tasks created to the number of cores available.[6]

One of the tasks suitable for parallelization is the execution of the Metropolis Monte-Carlo algorithm for each lattice site during the update. Unfortunately, the implementation of parallel programming techniques for this problem is far from trivial. The naive approach would be to simply divide the lattice into several pieces and updating these pieces (or chunks) at the same time. Since updating a lattice link variable requires the measurement of different path operators in the neighborhood of this link, any link used in these measurements cannot be updated concurrently. If we were to do this, the measured change in action, and thus the decisions made in the MMC algorithm would not be representative of the link neighborhood after the update. Thus we would find artifacts of this parallelization in the regions near and on the borders of these chunks. The size of the loop operators used in the improved action determine the stretch of these regions.

Thus we must define methods of cutting the lattice up into pieces, commonly known as masking. Bonnet et al. [7] suggest several masking patterns for both planar and non-planar improved action models. The implementation of several different patterns allows us to find good balance between overhead and processor utilization. The proposed masks proposed by Bonnet et al. [7] quickly produce numbers of tasks in the order of tens to hundreds. Since the development of our software was primarily done on computers with 4 to 24 processor cores, the need for a much cruder mask arose in order to minimize overhead.

3.6.1 The Slice Masking Pattern

One such masking procedure based on spatial or temporal slices resembles that of the linear masking scheme proposed by Bonnet et al. [7].

![Figure 3.10: Example of a $6^3 \times 6$ lattice divided into slices. The grey planes can be updated concurrently when using improved action measurements up to $a \times 2a$ rectangles.](image)

Here we divide the plane along a dimension and have planes which are separated by $3a$ steps by updated simultaneously. The results of which are then written to the original lattice. After this the next set of slices is updated. This process is repeated until the whole lattice is updated. It is easy to see that the effective number of slices one can update along a dimension at the same time is one third of the size of the lattice in that dimension. Thus, for example, a $12^3 \times 24$ lattice sliced along the
temporal dimension \((T = 24)\) can utilize \(24/3 = 8\) cores. It is important to note that, given the periodic boundary conditions, a lattice dimension size which is not divisible by 3 will not make optimal use of this masking pattern. This is because a slice near the boundary will still cause overflow of measurements on the slice just past the boundary. Measurements between parallelized and non-parallelized lattice simulations were compared to ensure that this technique did not affect results. As such we have found an effective masking scheme for parallelization of the lattice update, significantly reducing simulation times.
Chapter 4

Concluding Remarks

In this thesis we have discussed the fundamental aspects of lattice quantum chromodynamics. We have briefly discussed the continuum theory of the quantum field theory for the strong interaction known as QCD. By discretizing the theory and placing it on a lattice we have successfully developed lattice QCD simulation software. Parallelization techniques have been used to make the software suitable for deployment on supercomputers like *peregrine*. [8] The measurement of the static-quark potential and the string tension suggest that the software is in agreement with that of other research groups. [4]

Improvements on lattice generation could be made using several techniques including so-called *blocking and smearing* as well as the more modern *multi-level algorithm*. Implementation of these methods is still in development. Other methods for measuring the string tension could be explored and compared to the method used. [9]

The software that we have developed for running lattice QCD simulations was completely written in the popular programming language known as *Python*. The main reason for choosing this language was our shared familiarity with the language. Furthermore, Python offers a wide range of community driven tools which makes the language very suitable for fast development and prototyping. Python is one of the more concise languages available, but lacks in performance compared to other languages. [10] When it comes to parallel programming, a major weakness of the Python programming language is the so-called *global interpreter lock* or GIL which prevents parallel tasks from concurrently accessing the same memory. [11] For improved results I would suggest writing the software in a language like C/C++. This strongly-typed language offers full control over memory management and the lack of a GIL makes it much more suitable for parallel programming.
Bibliography