Magnetic Boson Condensation in S_3 Lattice Gauge Theory

Candidate Number: 1024186

University of Oxford

A thesis submitted for the degree of

Mathematical and Theoretical Physics

Trinity 2021

This thesis is dedicated to

all of my physics and mathematics teachers from Belgrade

for sparking my interest and preparing me for years to come at Oxford.

Acknowledgements

I would like to thank my supervisor, Prof. Steven H. Simon, for proposing this project and being extremely responsive and helpful.

Additionally, I would like to thank all my tutors from my college for teaching me the past four years.

Abstract

The aim of this dissertation is to explore the magnetic boson condensations in S_3 lattice gauge theory. We start by motivating and introducing topological quantum field theories and discussing their main properties. We describe the theoretical framework of lattice gauge theories and outline key results relating a quantum double of a gauge group and physical quantities in the theory. We work in the Euclidean formalism that allows us to use Markov Chain Monte Carlo (MCMC) algorithms in order to calculate the expectation of the physical operators. We analyse S_3 lattice gauge theory both algebraically and numerically using a specific MCMC algorithm — Heat Bath. We explore the phase space of coupling constants by numerically calculating the expectations of the free energy, its derivatives, and open string operators which tells us if a magnetic boson is condensed. Finally, by looking at the critical regions of the phase space we determine the orders of the phase transitions.

Contents

List of Figures

Chapter 1

Introduction

Topological quantum theories in 2+1D have been very popular in the recent years, both due to the beautiful mathematical theory behind them and their application to quantum computing. It is almost unbelievable how achievements in pure mathematics turned out to accurately describe processes found in nature. For example, Louis Kauffman and Vaughan Jones [12], interested in describing knots and answering questions such as whether two knots the same or not, developed a theory that sets a basis for building topological quantum computers. On the other hand, Shiing-Shen Chern and Jim Simons studied characteristic forms and geometric invariants of abstract manifolds [16], Edward Witten later used their work to formulate a general framework for 2+1D topological quantum theories known as Chern-Simons theory [22].

In this dissertation, we study a specific type of topological quantum theories — lattice gauge theories [5]. A lattice gauge theory is a type of gauge theory where both space and time are discretized. Discrete space and time allow the values of connection field to be chosen from a finite gauge group G, instead of from a Lie algebra in a continuous case. Consequently, the space of system configurations becomes finite and by a Wick rotation [7] one can make a correspondence between a lattice gauge theory

and a system described by classical statistical mechanics, albeit one could also get non-positive weights. This provides an opportunity for us to apply a wide range of tools such as Monte Carlo methods to perform calculations in lattice gauge theories [21]. Initially, these theories were studied as approximations to gauge theories, an example would be Lattice QCD [10]. However, over time they have shown a large set of interesting properties on their own and can be studied independently of their high energy applications.

Firstly, we give a very brief overview of topological quantum theories by describing fundamentals such as Aharanov-Bohm effect [1], Anyon charge-flux toy model, fusion rules and modular S and T matrices [18]. Secondly, we describe the mathematics behind lattice gauge theories. We introduce the concept of quantum doubles (Hopf algebras) [11] whose irreducible representations describe particles of the theory. In particular, we distinguish electric particles, magnetic particles and dyons (particles carrying both electric charge and magnetic flux). Our treatment is not rigorous, due to space constraints, but rather aimed at emphasizing the connection between these abstract objects and field excitations. Finally, we discuss condensation of magnetic bosons and introduce open and closed string operators that we use to calculate the S matrix and identify the phase transition as well as its order. In the next chapter, we present Monte Carlo methods and accurately described all the subtleties involved when applied to a lattice gauge theory. In the final chapter, we show both analytical and numerical results in a case of a theory with the gauge group being S_3 (symmetric group of order 3).

Being limited with both space and time, we chose to write about each topic proportionally to its involvement in the research we have done. Therefore, we completely omitted introducing some important concepts such as gauge theories and only briefly touch on others such as diagrammatic rules of topological quantum theories. Ultimately, we aimed to present a self-contained amount of theory needed for a reader,

relatively unfamiliar with topological quantum field theories, to be able to understand the work we have done.

Chapter 2

Topological Quantum Field Theory

Topology is one of the most interesting fields of geometry. It aims to identify intrinsic properties (topological invariants) of geometrical objects such as number of holes, orientability etc. These remain unchanged when the object is continuously deformed without cutting. In physics, we can describe the life of a particle by describing its world line in the spacetime. That is, a set of points that a particle traces from the moment it is created until it is annihilated. When a set of particles is considered, we can think of them as a collection of world lines. We topologically distinguish two systems only if their world lines cannot be bijectively mapped onto each other by a continuous deformation called *homotopy*. Therefore, *topological quantum field theories* are theories where two systems cannot be distinguished by any measurements if they are topologically equivalent. This has a rather remarkable consequence that it does not matter how close or far away the particles are during their lifetime. We only care about the braiding of their paths.

Topological quantum field theories (TQFT) are the richest when considered in 2+1D and we exclusively focus on these [18]. For example, in 3+1D, a particle can be either a boson which means that the wave function stays the same if we exchange two such particles, or it can be a fermion which means that the wave function picks

up a negative sign when two particles are exchanged. However, in 2+1D, apart from these two possibilities there are anyons which accumulate an arbitrary phase factor when exchanged. One can also have non-abelian anyons which we discuss below. This is a consequence of the fact that in 3 spatial dimensions particles cannot knot their paths. Two world lines can always be continuously deformed to an equivalent system of trajectories where the paths are not knotted. On the other hand, in only 2 spatial dimensions when there is a crossing between paths the system cannot be deformed into the system without the crossing.

One can start defining a topological quantum field theory in a few ways. For example, we could require that any correlator is unchanged when the metric is perturbed. Another approach is to map particles world lines onto quantum amplitudes and develop rules that let us compute the amplitudes given the particle paths. Both approaches can be developed into rigorous mathematical theories. Here, we will take a rather informal overview of the second approach.

We start by describing the Aharanov-Bohm effect which is one of the first experiments that hinted at the importance of topological properties in quantum systems. Afterwards, we will introduce a toy model of anyons considering them as charge-flux composites. This model captures the basic behaviour of particles in topological theories. Finally, we will introduce concept such as S matrix, T matrix and fusion matrices in general anyon theories, that we numerically measure in S_3 lattice gauge theory in order to identify phase transitions.

2.1 Aharanov-Bohm Effect

Consider a classical Young double slit experiment (figure 2.1). A beam of charged particles is shot towards a slit, behind which there is a screen on which the wave intensity is measured. In addition to the traditional experiment we also introduce a

magnetic field confined within the region between the two slits. We assume it has some non-zero value B inside and it is zero outside that region. Therefore, the particles never interact with the field. In the path integral approach to quantum mechanics [7], we can describe the amplitude on the screen using Feynman's propagator:

$$\langle \vec{x}_f, t_f | \hat{U}(\vec{x}_i, t_i; \vec{x}_f, t_f) | \vec{x}_i, t_i \rangle = \int \mathcal{D}\vec{x} e^{\frac{i}{\hbar} S[\vec{x}(t)]} \tag{2.1}$$

Equation 2.1 tells us that the probability of system, evolving from some initial position x_i at time t_i to the final position x_f at t_f, equals the weighted sum over all possible paths with such boundary conditions. Each path picks up a phase that equals the imaginary unit times its action over $\hbar$.

Particles feel no magnetic field, but the quantity that enters the Lagrangian is vector potential $\vec{A}$ which can be non-zero even when $\vec{B} = 0$. The action is modified such that:

$$S_0 \to S + q \int d\vec{x} \cdot \vec{A} \tag{2.2}$$

where q is the charge of the particle.

The additional phase difference between paths 1 and 2 in picture 2.1 introduced by this factor can be written as:

$$e^{\frac{iq}{\hbar} \int_{slit_1} d\vec{x} \cdot \vec{A} - \frac{iq}{\hbar} \int_{slit_2} d\vec{x} \cdot \vec{A}} = e^{\frac{iq}{\hbar} \oint d\vec{x} \cdot \vec{A}}. \tag{2.3}$$

Recalling that $\vec{B} = \nabla \times \vec{A}$ and using Stokes' theorem the integral equals the enclosed magnetic flux between two paths: $\Phi_{enclosed}$. Therefore, when magnetic field is turned on the interference pattern is shifted up or down.

This experiment has two important consequences: (1) vector potential has a real impact on the physical system — it is not an artificially introduced mathematical object and (2) the amplitude depends on the topology of the path — the number of

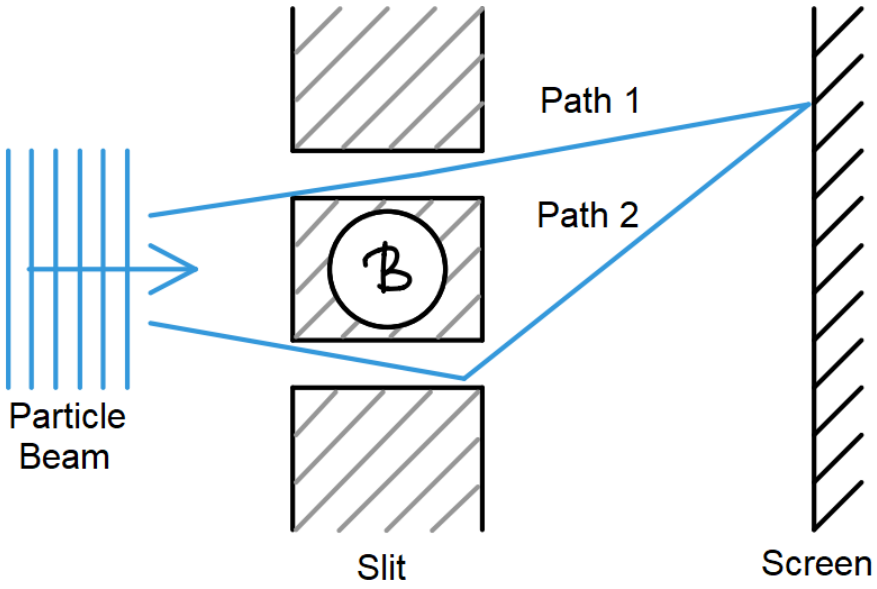

Figure 2.1: Aharanov-Bohm set-up

times the path wraps around the magnetic field.

2.2 Anyon Charge-Flux Composite Toy Model

In order to gain basic intuition about anyon theories, we present a toy model of anyons [18] that views particles as collections of an electric charge q and a magnetic flux Φ — (q, Φ). It can be pictured as in figure 2.2.

From the Aharanov-Bohm effect we learnt that when a charge q moves around the flux Φ it acquires the phase factor $e^{\frac{iq\Phi}{\hbar}}$. Clearly, the same thing happens when a flux Φ is moved around the particle q. Therefore, when an anyon (q_1, Φ_1) moves around another anyon (q_2, Φ_2) (figure 2.2) the system picks up a phase equal to $\frac{i(q_1\Phi_2+q_2\Phi_1)}{\hbar}$. When two anyons are exchange the phase factor is half of when one is moved around the other one.

We can also explain the spin of anyons if we imagine that the charge is slightly displaced compared to the flux (figure 2.2). An internal rotation of 2π is equivalent to the charge being moved around its own flux Φ and therefore the phase factor

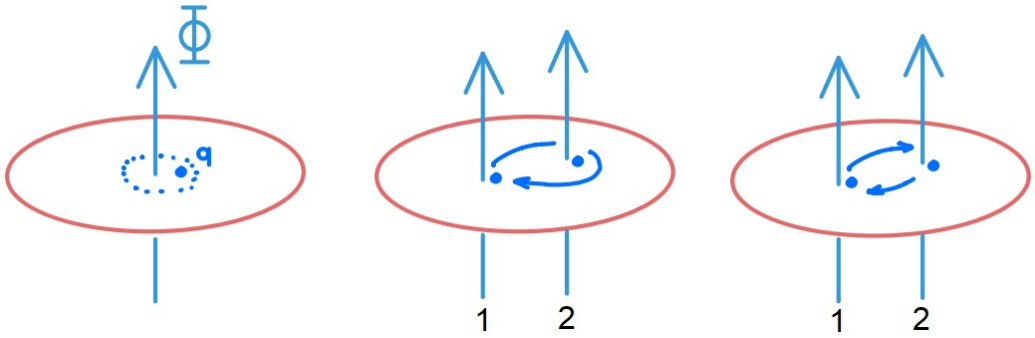

Figure 2.2: Anyon as charge-flux composition, braiding one anyon around another and the exchange of two anyons.

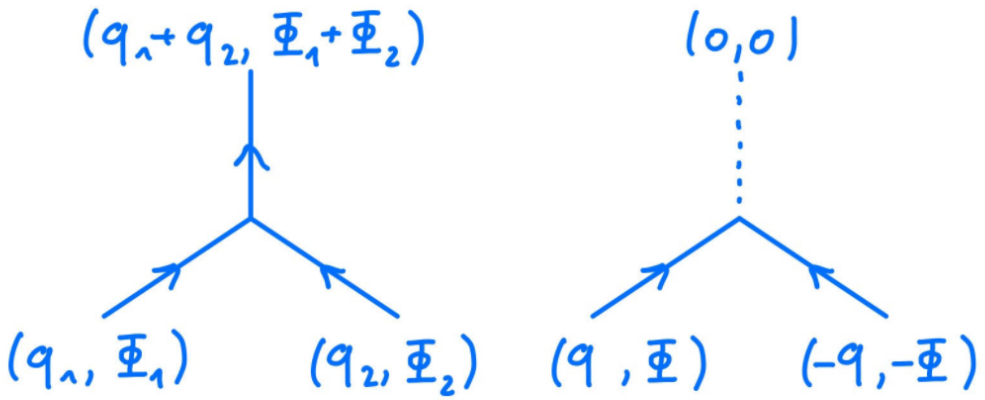

Figure 2.3: Fusion of two arbitrary anyons and fusion of an anyon with its anti-anyon.

acquired is $e^{\frac{iq\Phi}{\hbar}}$. We can immediately recognize that for bosons we have $\frac{q\Phi}{\hbar} = 2k\pi$ and for fermions we have $\frac{iq\Phi}{\hbar} = (2k+1)\pi$.

We can fuse two anyons (q_1, Φ_1) and (q_2, Φ_2) to get the anyon $(q_1 + q_2, \Phi_1 + \Phi_2)$. If we label vacuum as $(0, 0)$, we see that the anti-anyon is given by $(-q, -\Phi)$ (figure 2.3).

2.3 General Anyon Theories

Having introduced the motivation and a toy model of anyons, we are now ready to give an overview of a general anyon theory. Anyon theories are very rigorous mathematical theories which heavily rely on topology, differential geometry, group theory etc. Nonetheless, we focus on physics that sits on top of the mathematical machinery that describes it until the next chapter where we dive deeper into a specific type of a topological quantum field theory — discrete gauge theory. We illustrate main physical phenomena and define key quantities that we will later numerically measure in case of S_3 discrete gauge theory.

2.3.1 Anyons and Fusion

Looking at an anyon theory, we start by introducing particle types. We label particles as a, b, c... It is convenient to use diagrams in order to evaluate quantum amplitude of physical processes. We draw diagrams such that time axis points upwards, one spatial dimension is horizontal and the other one is depicted by over and under crossings of the paths of particles. A particle propagating through time would be drawn as an arrow pointing up in time and its anti-particle with an arrow pointing back in time (figure 2.4).

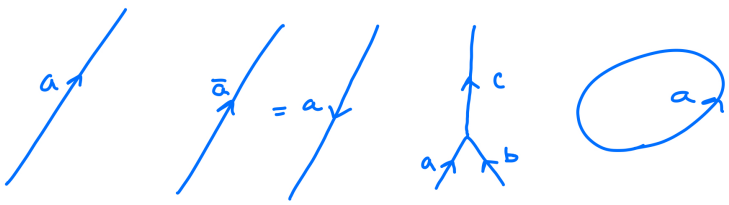

Figure 2.4: Particle propagation, anti-particle, fusion, loop diagram.

A fusion of particles is a state in which two or more particles are close together so that far away they look like a single particle. In the toy model example, this was a deterministic process and two anyons would always fuse to a specific anyon.

This is known as an abelian theory. In the case of nonabelian theories we write $a \times b = c + d + ...$, which means that by fusing a and b we can get c, d or something else. If particles a and b are together seen as a particle c, we say that we have a system of particles a and b in the channel c or that a and b fuse to c. The fusion is commutative $a \times b = b \times a$, because physically the order in which we write a and b does not have any meaning. Diagrammatically, we represent this by drawing two joining lines for a and b and one line coming out of them for the resulting particle (as in figure 2.4). We also allow a reverse process of splitting of a certain particle into two other particles. We define a special particle that we call *vacuum* and label it as I. Vacuum is a unique particle for which $I \times a = a$ for any a and we represent it by a dotted line. For a particle a, we define its anti-particle $\bar{a}$ which is a unique particle for which we have $a \times \bar{a} = I + ...$ Note that particle and anti-particle could fuse to a non-vacuum particle.

Generally, fusion can be written as:

$$a \times b = \sum_c N_{ab}^c c. \tag{2.4}$$

Here N_{ab}^c are fusion multiplicities, integers that tell us how many times c occurs on the right hand side of the equation. We can immediately conclude the following properties of these coefficients: commutativity $N_{ab}^c = N_{ba}^c$, time reversal $N_{ab}^c = N_{\bar{a}\bar{b}}^{\bar{c}}$, trivial vacuum fusion $N_{aI}^c = \delta_{ac}$ and anti-particle uniqueness $N_{ab}^I = \delta_{a\bar{b}}$. Sometimes, it is useful to define a matrix N_a for each particles a whose entries are labeled by b and c and equal to $[N_a]_b^c = N_{ab}^c$.

We also define a quantity called a *quantum dimension* d_a of a particle a. It counts the internal degrees of freedom of the particle. A Hilbert space of N particles of type a has the dimension scaling as d_a^N in the limit of large N. It can be derived from 2.4 that d_a is equal to the largest eigenvalue of N_a [18]. We also define a total quantum dimension of the theory as $\mathcal{D} = \sqrt{\sum_a d_a^2}$. It can be shown using diagrammatic rules

that the quantum dimensions satisfy fusion rules [18].

$$d_a \cdot d_b = \sum_c N_{ab}^c d_c \tag{2.5}$$

Example: Ising Anyons

One of the simplest non-trivial examples of a nonabelian anyon theory is Ising theory. It has only 3 particles $\{I, \psi, \sigma\}$. The fusion in given by:

$$\begin{aligned} &I \times \sigma = \sigma, \ I \times \psi = \psi \\ &\psi \times \psi = I \\ &\sigma \times \sigma = I + \psi \\ &\psi \times \sigma = \sigma. \end{aligned} \tag{2.6}$$

We see that $\bar{\psi} = \psi$, $\bar{\sigma} = \sigma$, $d_\psi = 1$, $d_\sigma = \sqrt{2}$, $\mathcal{D} = 2$ and we can easily read off the N matrices.

2.3.2 Braiding and Modular Matrices

Apart from the fusion matrices, we are also interested in the braiding properties of particles. We define a *twist* as an internal rotation by 2π. Upon twisting, a single particle state $|a\rangle$ acquires a phase factor θ_a.

Note that for bosons we have $\theta_a = 1$, fermions $\theta_a = -1$ and for anyons in general we require θ_a to lie on the unit circle in the complex plane. When we adiabatically braid a particle a around a particle b in channel c the wave function picks up a phase factor equal to $\frac{\theta_c}{\theta_a \theta_b}$ which is known as the ribbon identity [18].

We define a *modular* T matrix as: $T_{ab} = e^{-2\pi i \frac{\bar{c}}{24}} \theta_a \delta_{ab}$, where $\bar{c}$ is the central charge of the theory. The central charge is defined through the following equation:

$$e^{2\pi i \frac{\bar{c}}{8}} = \frac{1}{\mathcal{D}} \sum_a d_a^2 \theta_a \tag{2.7}$$

By using diagrammatic rules we can show that a simple loop diagram of a evaluates to d_a. Equivalently, we could have defined d_a as the value of the loop diagram and then show that they satisfy the fusion rules 2.5.

Another key quantity is a *modular* S matrix whose entries S_{ab} multiplied by $\mathcal{D}$ equal to the following diagram (figure 2.5):

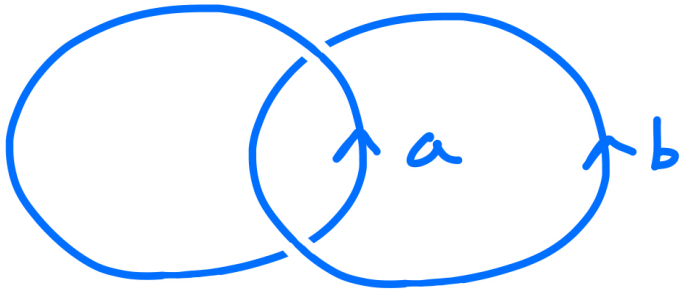

Figure 2.5: S matrix element.

Note that the $S_{Ia} = S_{aI} = \frac{d_a}{\mathcal{D}}$. Therefore, if we set the first index in rows and columns of the S matrix to correspond to I, we can read off the quantum dimensions for all particles from the first row/column of the matrix.

It turns out that knowing the S matrix is enough to reconstruct N matrices. This result is known as Verlinde formula [18]:

$$N_{ab}^{c} = \sum_{x} \frac{S_{ax} S_{bx} S_{cx}^{*}}{S_{Ix}}. \tag{2.8}$$

To conclude, we have defined important quantities such as N, modular T and S matrices. In the next section, we will see how these can be calculated based on the irreducible representations of quantum doubles underlying the discrete lattice gauge theory.

Chapter 3

Lattice Gauge Theory

In this section, we specialise towards a specific set of TQFTs of interest. These are lattice gauge theories. A general set-up consists of a discrete set of positions ($x \in \{k|k \in \mathbb{Z} \wedge 0 \leq k < N_x\}$ and $y \in \{k|k \in \mathbb{Z} \wedge 0 \leq k < N_y\}$) on a two-dimensional torus and a discretized time $t \in \{k|k \in \mathbb{Z} \wedge 0 \leq k < N_t\}$. We also introduce some finite gauge group G. Since gauge fields transform by being acted upon by elements of the gauge group when being moved in space and time, we define links $U_{ij} \in G$ connecting two neighbouring positions (positions where only one of x, y or t differs by one) i and j in the spacetime. We also care about the direction and if $U_{ij} = g$ then $U_{ji} = g^{-1}$. These are the connection fields and serve the same purpose as vector potential in QED [17] or coloured gluon fields in QCD [10].

The whole richness of the theory lies within the gauge group. We will see how the particles arise based on group's conjugacy classes and irreducible representations of its centralizers and how these are unified into the framework of quantum doubles.

3.1 Euclidean Approach

3.1.1 Wick Rotation

In order to use the full machinery of statistical physics methods, we perform Wick rotation on our system [7]. This introduces imaginary periodic time. Therefore, time and space become equal and we can treat the system as 3D space with Euclidean metric. The path integral becomes identified with a partition function:

$$\mathcal{Z} = \sum_{C} e^{-S_E(\{U\})}. \tag{3.1}$$

Here the sum is performed over all configurations $\{U\}$ (all possible combinations of values of U_{ij}) and $S_E(\{U\}) = -iS(\{U\})$. Note that if the action initially has complex terms, S_E might not be real. This can produce non-positive weights e^{-S_E}. Nonetheless, in our problem S_E is always real and the weights are always positive. Since, we will only be dealing with Euclidean action we drop the index E.

This allows us to use the familiar formula for calculating the expectations of the operators:

$$\langle O \rangle = \frac{1}{\mathcal{Z}} \sum_{C} O(\{U\}) e^{-S(U)}. \tag{3.2}$$

Note that this is a rather powerful technique as the system can now considered to be purely classical. Moreover, it opens up the possibility of using known statistical methods for approximating the equation 3.2 [4].

3.1.2 Gauge Transformations

Let us introduce some external scalar field ϕ_i^α as a function of site i. Superscript α refers to the representation of the gauge group. We can perform global and local gauge transformations on this field. A local gauge transformation is:

$$\phi_i^\alpha \to D_\alpha(g_i)\phi_i^\alpha. \tag{3.3}$$

Where $D_\alpha(g_i)$ is the representation matrix of g_i in α. A global gauge transformation is the one where all g_i are the same for all sites i.

Typical actions that are studied in continuous QFTs consist of kinetic and potential energy. Potential energy is often based on terms such as powers of $\phi_i^{\alpha*} \cdot \phi_i^\alpha$. However, kinetic energy might contain non-local terms such as $\phi_i^{\alpha*} \cdot \phi_j^\alpha$. This is precisely what one sees when one attempts to discretize the familiar kinetic term $\frac{1}{2}m|\dot{\phi}|^2$. Upon global gauge transformations these terms are, indeed, invariant:

$$\phi_i^{\alpha*} \cdot \phi_j^\alpha \to \phi_i^{\alpha*} D_\alpha(g^{-1}) D_\alpha(g) \phi_j^\alpha = \phi_i^{\alpha*} \cdot \phi_j^\alpha. \tag{3.4}$$

However, it obviously fails to be gauge invariant when acted by local gauge transformations because the term in the middle is not generally equal to one:
$D_\alpha(g_i^{-1})D_\alpha(g_j) \neq 1$. This is exactly why we need to have the connection field and compute non-local terms by first parallel transforming the field values to the same site. The correct expression that we need is: $\phi_i^{\alpha*} D_\alpha(U_{ij})\phi_j^\alpha$. Now, local gauge transformation can be written as:

$$\phi_i^{\alpha*} D_\alpha(U_{ij})\phi_j^\alpha \to \phi_i^{\alpha*} D_\alpha(g_i^{-1}) D_\alpha(\tilde{U}_{ij}) D_\alpha(g_j)\phi_j^\alpha = \phi_i^{\alpha*} D_\alpha(g_i^{-1}\tilde{U}_{ij} g_j)\phi_j^\alpha. \tag{3.5}$$

From here, we can easily see how the connection field should transform upon local gauge transformations as:

$$U_{ij} \to \tilde{U}_{ij} = g_i U_{ij} g_j^{-1} \tag{3.6}$$

and similarly upon global gauge transformations:

$$U_{ij} \to \tilde{U}_{ij} = g U_{ij} g^{-1}. \tag{3.7}$$

3.1.3 Gauge Field Dynamics

So far, we have defined our space and time, possible gauge field values and deduced the transformation laws for the link values. The last ingredient that we need is the action used in the equations 3.1 and 3.2. It seems that the action can be arbitrarily defined. However, the transformation law 3.5 as the requirement for gauge invariance immediately excludes any products of U_{ij} values that is not a closed loop. So, lets consider some closed loop product U_L:

$$U_L = U_{i_1 i_2} \cdot U_{i_2 i_3} \cdot ... \cdot U_{i_N i_1}. \tag{3.8}$$

Upon local gauge transformations we have:

$$U_L \to g_{i_1} U_{i_1 i_2} g_{i_2}^{-1} \cdot g_{i_2} U_{i_2 i_3} g_{i_3}^{-1} \cdot ... \cdot g_{i_N} U_{i_N i_1} g_{i_2}^{-1} = g_{i_1} U_L g_{i_1}^{-1}. \tag{3.9}$$

There are two important things to note about this expression. Firstly, upon local gauge transformations the value of the loop stays within the same conjugacy class. Therefore, the terms that enter the action can only be class functions of the loop values. Secondly, the value of the loop upon local gauge transformations depends on the site i_1 where where we start the loop product, luckily the conjugacy class does not. Thus, when evaluating the loop value we have to properly define the reference point unless we only care about the conjugacy class.

The action of the theory is a Wilson action [19]. It is the most general action that reproduces Yang-Mills action at the leading order when Yang-Mills theories are discretized:

$$S = -\sum_{p}\sum_{\alpha}\beta_{\alpha} Re(\chi_{\alpha}(U_p)). \tag{3.10}$$

The first sum runs over all plaquettes p (four neighbouring vertices forming a square), χ_α is the character in the irreducible representation α and β_α are the coupling constants. Since $Re(z) = \frac{z+z^*}{2}$ and $\chi_\alpha(U_p)^* = \chi_\alpha(U_p^{-1})$, this essentially means that we take the average value of the plaquette p computed in both clockwise and anti-clockwise direction.

The equation 3.10 is said to be in the representation basis. Since there is the same number of conjugacy classes and irreducible representations, we can make use of the character orthogonality to write the action in the conjugacy class basis [5]:

$$S = -\sum_{p}\sum_{A}\beta_{A}\delta_{A}(U_p). \tag{3.11}$$

Here the second sum runs over all conjugacy classes A and δ_A is a function that evaluates to 1 if $U_p \in A$ and to 0 otherwise.

The coupling constants $\beta_A \geq 0$ are free parameters of the theory. Based on their values different phases may be realised. The system always tends towards configurations where all $U_p \in A$ for which β_A is maximal amongst all coupling constants. For example, a vacuum is a state where all $U_p = e$ (group identity element), this can be realised by setting all $\beta_A = 0$ except for $A = e$.

3.2 Simple Field Excitations

Having defined the action of the theory, we can begin identifying its particles. We will see that here the particles can carry electric charge, magnetic flux or both (dyons). In order to identify these particles, we define the operators that let us compute their quantum dimensions and S matrix elements. Defining the operators, we rely on

Elitzur's theorem [6], which we state without the proof. It says that all gauge non-invariant operators have zero expectation. For electric charges the operator is a Wilson loop and for magnetic fluxes it is a 't Hooft loop. These are simple field excitations which we describe now.

3.2.1 Electric Charges

In gauge theories, the electric charge is defined as an internal degree of freedom. It is usually described by a vector v_α that transforms according to some irreducible (reducible representations are products of irreducible ones) representation α of the gauge group G. For example, in QED this is a scalar and the gauge group is $U(1)$ and in QCD this is a vector of colours and the gauge group is $SU(3)$.

In continuous gauge theories, when a particle moves along its world line, its internal charged gets parallel transported according to the connection field $A^\alpha_\mu(x)$. The *Wilson loop* operator [20] describes how the internal charge changes when particle travels around a loop:

$$W[C] = \mathrm{Tr}\,\mathcal{P}\exp\left(i\oint_C A^\alpha_\mu(x)dx^\mu\right). \tag{3.12}$$

Here $\mathcal{P}$ stands for path ordering and it is analogous to time ordering an exponential in quantum mechanics.

Since this is the evolution operator, it precisely corresponds to the closed path diagram of a particle carrying only the electric charge. In case of lattice gauge theories, we know that the values of the links are from the gauge group whereas the values of A^α_μ are from the Lie algebra in continuous ones. Recalling that the group elements can be obtained by taking the exponential of the proper algebra elements, we can write the Wilson loop operator as a trace in a representation α of the product of links around the path:

$$W[i_1 \to i_2 \to ... \to i_N \to i_1] = \chi_\alpha(U_{i1i2} \cdot U_{i_2i_3} \cdot ... \cdot U_{i_Ni_1}). \tag{3.13}$$

We have shown earlier, that such a product is gauge invariant and therefore it does not vanish by Elitzur's theorem.

We can immediately use this operator in order to find out the quantum dimension of an electric particle. In vacuum, where we evaluate loop diagrams, all links take the value of the trivial group element. Thus, the equation 3.13 reduces to $d = \chi_\alpha(e)$ which equals $dim(\alpha)$.

To conclude, we have seen that the electric particles are completely described by irreducible representations of the gauge group, their loop diagrams are connected to Wilson loop operators and their quantum dimensions are equal to the dimensions of the corresponding representations.

3.2.2 Magnetic Fluxes

To think about magnetic fluxes, we use what we know from the anyon toy model. More specifically, recall that when an electric particle is moved around the magnetic flux it picks up a phase factor. Since the smallest loop is a plaquette, we can think of magnetic fluxes living at the centres of the plaquettes. To generalise this idea we allow the magnetic flux value to be from the gauge group and thus moving an electric particle v_α around the flux g we have:

$$v_\alpha \to D_\alpha(g)v_\alpha. \tag{3.14}$$

However, there is still some redundancy in labelling magnetic fluxes by group elements. To see this, consider another situation where we first perform a global gauge transformation h, move the particle around some other flux g' and perform an inverse global gauge transformation h^{-1}. So, we have:

$$v_\alpha \to D_\alpha(h^{-1}g'h)v_\alpha. \tag{3.15}$$

These two cases are equivalent if $g = h^{-1}g'h$. Therefore, the magnetic fluxes when modded out by gauge symmetry are labeled by the conjugacy classes of the gauge group.

In quantum mechanics, any state can be written as a linear superposition of basis states. Similarly, we allow a flux state to be a super position of basis fluxes. Basis fluxes $|g\rangle$ are labeled by group elements $g \in G$. Formally, we say that a flux state lives in the group algebra $\mathbb{C}G = \{\sum_{i=1}^{|G|} c_i \cdot |g_i\rangle \mid c_i \in \mathbb{C}, g_i \in G\}$ of the gauge group G. It is useful to define a *braiding* operator $\mathcal{R}$ [5] that acts on a state of two fluxes $|g_1\rangle \otimes |g_2\rangle$ (which, conventionally, means that g_2 is to the right of g_1) as:

$$\mathcal{R}|g_1\rangle \otimes |g_2\rangle = |g_1 g_2 g_1^{-1}\rangle \otimes |g_1\rangle. \tag{3.16}$$

Physically, $\mathcal{R}$ moves the flux g_2 to the left of g_1 in the clockwise direction. To move the flux g_2 around the flux g_1 back to its starting point we apply the operator twice:

$$\mathcal{R}^2|g_1\rangle \otimes |g_2\rangle = |(g_1 g_2)g_1(g_1 g_2)^{-1}\rangle \otimes |g_1 g_2 g_1^{-1}\rangle. \tag{3.17}$$

To determine the composition of an arbitrary flux state $|h\rangle$ we calculate the following matrix element for every basis state $|g_i\rangle$:

$$\langle h| \otimes \langle g_i|\mathcal{R}^2|g_i\rangle \otimes |h\rangle = \langle g_i|hg_ih^{-1}\rangle\langle h|(g_ih)h(g_ih)^{-1}\rangle. \tag{3.18}$$

If $h = g_j$ the right hand side of the equation simplifies to $\delta_{g_i, g_j g_i g_j^{-1}}$. Here δ function evaluates to one if the fluxes are the same and zero otherwise. Recall that the fluxes are the same if the corresponding elements are in the same conjugacy class.

To evaluate diagrams including magnetic particles we use the *'t Hooft* operators. These operators need to be gauge invariant and in one to one correspondence with the conjugacy classes. To describe a particular diagram we select a number of plaquettes in which we force the existence of the magnetic flux. For example, if we evaluate a loop diagram we select a closed loop of centres of plaquettes (a loop on a dual lattice) and force the flux through those plaquettes. On the other hand, if we want to evaluate a line diagram (which is equivalent to a magnetic flux line from a source to a sink) we pick a number of consecutive plaquettes and force the flux through them. To force a particular flux h through a plaquette p with the value U_p we remove the factor $e^{-S(U_p)}$ from the action and insert the factor $e^{-S(h^{-1}U_p)}$. The last thing that we need to consider the parallel transport. Simply inserting h^{-1} would not put all plaquettes on an equal footing. Therefore, we select a base point i_0 and insert the flux h such that we first transport it to the base point by $U^{-1}_{i_0,p}$, insert the flux by h^{-1} and finally transport it back to the plaquette by $U_{i_0,p}$. Here $U_{i_0,p}$ is the connection between the base point i_0 and the corner of a relevant plaquette p. Putting all of this together, the amplitude of the flux of conjugacy class A living on a set of plaquettes P is described by the 't Hooft operator:

$$H^A(P) = \sum_{h\in A} \prod_{p\in P} e^{S(U_p)-S(U^{-1}_{i_0,p}h^{-1}U_{i_0,p}U_p)}. \tag{3.19}$$

Similarly to the case of electric particles, we can evaluate the loop diagrams in vacuum by hand. All links and plaquettes equal e and all exponents in 3.19 are 0. Therefore, the quantum dimension of a magnetic flux equals the number of elements in the conjugacy class.

3.3 Dyons and Quantum Doubles

In the previous section, we have seen how purely electric and magnetic particles can be described. It was also based on the continuous gauge theories. In this section, we unify the two approaches and define the notion of a *Quantum Double*. In this framework, the particles (electric, magnetic and dyons) are irreducible representations of quantum doubles. Finally, we outline how to calculate modular S and T matrices and the fusion coefficients.

This section might appear difficult to a reader who is not familiar with lattice gauge theories. However, it is not crucial for a reader to understand the derivations in detail but rather to appreciate the importance of the final results: dyons being labeled by a conjugacy class and an irreducible representation of its centralizer, and equations 3.25 and 3.26. More detailed review can be found in [5].

3.3.1 Enumerating Dyons

We already stated that dyons can carry both magnetic flux and electric charge. Therefore, we write a dyon state as: $|g, v_\alpha\rangle$. However, v_α does not necessarily live in the the space acted by the representation of the gauge group. To see this, we take a flux g_i and move it around the dyon:

$$\langle g, v_\alpha| \otimes \langle g_i|\mathcal{R}^2|g_i\rangle \otimes |g, v_\alpha\rangle = \langle v_\alpha|D_\alpha(g_i)|v_\alpha\rangle \delta_{g,g_i g g_i^{-1}}. \tag{3.20}$$

Again the delta function evaluates to one if g and $g_i g g_i^{-1}$ are in the same conjugacy class, so only such g_i elements are allowed in equation 3.20. Therefore, α is a representation of the group of these elements. That group is a *centralizer* N^A of a conjugacy class A:

$$N^A = \{x_i^A \mid g_i^A = x_i^A g_1^A (x_i^A)^{-1},\ g_i^A \in A\}, \tag{3.21}$$

where g_1^A is arbitrary element from A fixed in advance and $x_1^A = e$.

To conclude, a dyon, particle in a lattice gauge theory, is labeled by a pair consisting of a a conjugacy class and the irreducible representation of its centralizer (A, α_{N^A}).

3.3.2 Quantum Doubles

Formally a *quantum double* $D(G)$ of a finite group G is a *Hopf algebra* which is a noncommutative algebra with some additional structure [11]. It would take a lot of time to formally introduce this object and thus, we chose to define it in terms of the physics of our system.

A general particle state is described by some flux h and charge v_α. We define an operator g that performs a global gauge transformation:

$$g|h, v_\alpha\rangle = |ghg^{-1}, D_\alpha(g)v_\alpha\rangle. \tag{3.22}$$

We also define the *flux projection* operator P_g as:

$$P_g|h, v_\alpha\rangle = \delta_{gh}|h, v_\alpha\rangle. \tag{3.23}$$

One can easily derive the following identities:

$$\begin{aligned} P_g P_{g'} &= \delta_{gg'} P_g \\ hP_g &= P_{hgh^{-1}} h. \end{aligned} \tag{3.24}$$

A set of operators $\{P_g h\}_{g,h\in G}$ generates a quantum double $D(G)$.

A quantum double is a central object of the theory because the particles are its irreducible representations and the braiding properties of the theory are determined by the algebra of the quantum double [11].

3.3.3 Ribbon Operator

The operator that performs an internal rotation by 2π turns out to be the central element of the quantum double $c = \sum_g P_g g$ which is also called a *ribbon* operator.

By considering the ribbon operator, we can show that the spins of the dyons are given by [5]:

$$D_\alpha(h_1^A) = e^{2\pi i s_{(A,\alpha)}} \mathbb{I}_\alpha. \tag{3.25}$$

Where $s_{(A,\alpha)}$ is the spin of the dyon. This equation can be used for reconstructing the T matrix. We can immediately note that there is exactly one purely magnetic boson per conjugacy class (except for vacuum) described by the trivial representation.

3.3.4 S Matrix

By considering the braiding properties defined by the algebra of a quantum double it can be shown that the S matrix elements are given by [18]:

$$S_{(A,\alpha),(B,\beta)} = \sum_{[h_i^A, h_j^B]=0} \mathrm{Tr}(\alpha((x_i^A)^{-1} h_j^B x_i^A)) \, \mathrm{Tr}(\beta((x_j^B)^{-1} h_i^A x_j^B))^*. \tag{3.26}$$

This is a crucial result for us. Later in this chapter, we will introduce operators that let us compute the S matrix for a given configuration of links $\{U\}$. In the vacuum, we expect that the expectation of those operators coincide with the values predicted by 3.26. However, in a broken phase they will be different which means that by comparing the values that we compute and the ones predicted by 3.26 we can identify the phase transition.

3.3.5 Fusion and S Matrix

Similarly to the Clebsch-Gordon decomposition of product states in quantum mechanical, a product of two irreducible representations of $D(G)$ can be written as a direct sum of irreducible representations of $D(G)$ [5]. This precisely corresponds to the fusion of anyons described in the previous chapter. In general we have:

$$(A,\alpha)\otimes(B,\beta)=\bigoplus_{(C,\gamma)} N^{(C,\gamma)}_{(A,\alpha),(B,\beta)}(C,\gamma). \tag{3.27}$$

Having found the S matrix elements given by 3.26, we can use Verlinde formula 2.8 to find the fusion coefficients $N^{(C,\gamma)}_{(A,\alpha),(B,\beta)}$.

3.3.6 Dyonic Line and Loop Operators

In order to evaluate diagrams involving dyons, we need to define operators similar to 't Hooft and Wilson. Since, we view a dyon as a combination of the electric charge and magnetic flux the operator should be a hybrid of the two. In the limiting cases where $A=\{e\}$ it should reduce to 't Hooft operator and when α is trivial it should reduce to Wilson operator. These operators were first derived in [2] and later tested in [3].

Physically, we want to put the charge on the vertices and flux through the plaquettes. We put charges on a loop on the direct lattice and force the fluxes through the set of plaquettes pierced by a corresponding loop on a dual lattice. A dual lattice C can be chosen to be a lattice displaced by $(\frac{1}{2},\frac{1}{2},-\frac{1}{2})$. We label the plaquettes of the dual loop by p_j and their values by U_{p_j}. Every plaquette has a corresponding site j on the real lattice. A site j can contain 1 (site (2) in figure 3.1) plaquette or, if on the edge, it could contain either 0 (site (3) in figure 3.1) or 2 (site (2) in figure 3.1) plaquettes. Additionally, we define $U_{j-1,j}$ to be a value of the connection between sites $j-1$ and j.

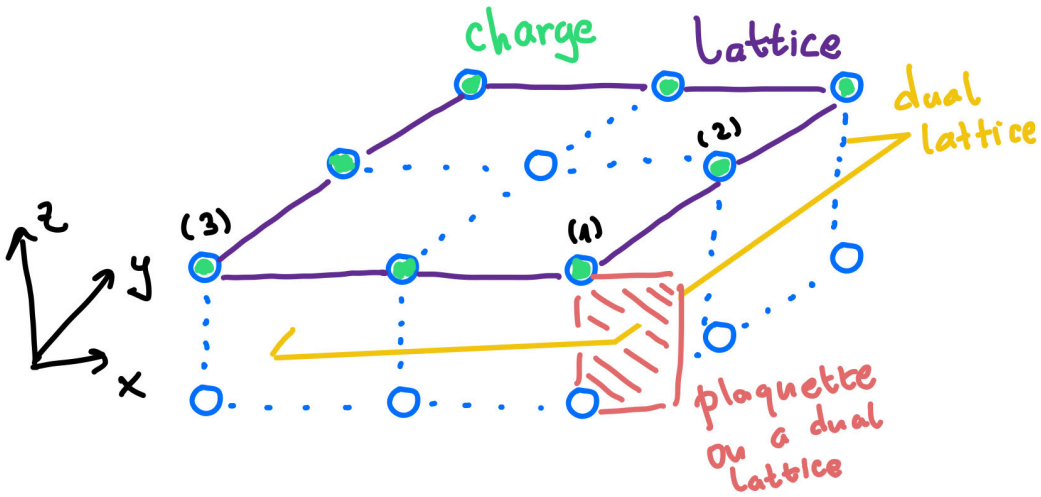

Figure 3.1: An example of a loop on a lattice, the corresponding loop on the dual lattice, charge positions, plaquette on a dual lattice and sites that correspond to 0 (3), 1 (2) and 2 (1) plaquettes.

Having defined the plaquettes on which the flux lives, we can follow exactly the same procedure we did in order to define 't Hooft operator. Recall that we need to introduce the base point i_0. We label the connection term between i_0 and j that enters 't Hooft 3.19 as $k_{p_j} = U_{i_0 j}^{-1} h^{-1} U_{i_0 j}$. To account for the presence of the electric charge we can use the product of D_α for each link the particle moves through. When a loop is formed the product becomes the trace and we recover a Wilson loop operator. However, simply inserting $D_\alpha(U)$ for a link with the value U would not work because α is the representation of the centralizer and U is from G. Therefore, we need to have a map from group elements to the centralizer of the conjugacy class. A good choice for this map, that makes a reference to the base point i_0 is:

$$g \to x^{-1}_{U_{j-1,j} k_{p_j} U^{-1}_{j-1,j}} U_{j-1,j} x_{k_{p_j}}. \tag{3.28}$$

Using this map we define the dyonic operator as:

$$\Delta^{(A,\alpha)}(C) = \sum_{h\in A} \prod_{p_j \in C} D_\alpha(x^{-1}_{U_{j-1,j}k_{p_j}U^{-1}_{j-1,j}} U_{j-1,j} x_{k_{p_j}}) e^{S(U_{p_j})-S(k_{p_j}U_{p_j})}. \tag{3.29}$$

Note that this operator is gauge invariant. When $A = \{e\}$ the exponential bit vanishes and all x elements are equal to e, so the operator indeed reduces to the Wilson loop operator. On the other hand, when α is trivial, the D_α terms are all equal to 1 and we recover the 't Hooft operator.

We define the *open string* operator to be 3.29 evaluated on a straight line and the *closed string* operator to be 3.29 evaluated on a loop. The expectation of these operators correspond to the line and loop diagrams. If we want to evaluate S matrix elements we need to specify two loops and the corresponding dual loops C_1 and C_2 that link each other once. Then the S matrix element is given by: $S_{(A,\alpha),(B,\beta)} = \frac{1}{\mathcal{D}}\langle \Delta^{(A,\alpha)}(C_1)\Delta^{(B,\beta)}(C_2)\rangle$. Any other diagram can be evaluated in a similar way by carefully specifying the set of vertices on a real and dual lattice.

3.4 Condensation

In this section we describe the general theory of condensations in topological quantum field theories. This condensation happens in the sense of Bose-Einstein condensation. We explain the mechanism of condensation and outline the general procedure of finding the resulting quantum field theories after condensation. We refer to a general algorithm [14] that helps finding the resulting TQFTs and some results specific to lattice gauge theories [5].

When a particular particle condensates, it essentially behaves as a vacuum particle. A vacuum particle can be created or annihilated at any time — adding vacuum particle to the spacetime diagram does not change anything. Similarly, a condensed particle can be freely emitted or absorbed by the condensate. For this to be possible,

a particle needs to be a boson. In other words, it must not experience any phase when exchanged with and braided around other particles.

3.4.1 General Condensation Process

The condensation process contains 2 steps. We start with a topological quantum field theory $\mathcal{A}$ and we identify a boson J that we want to condense. We then identify a fusion algebra $\mathcal{T}$ conditioned on the fact that J is condensed. Finally, we exclude confined particles from $\mathcal{T}$ and we get a new topological quantum field theory $\mathcal{U}$ that corresponds to the condensed $\mathcal{A}$ theory.

To intuitively understand why we need these steps, consider an abelian TQFT where all particles a have $d_a = 1$. When the condensation occurs, J can be freely created and annihilated. Therefore, we can no longer distinguish particles a and b if $J \times a = b$ or $J \times b = a$. This means that particles a and b become the same particle type t in $\mathcal{T}$. Now, suppose that a and b do not have the same spin. In that case, t does not have well define braiding properties and is not allowed to move throughout the space. Thus, it can only live at the one-dimensional boundary of the system. We say that such a particle is *confined*. This is the reason why $\mathcal{T}$ is not a valid TQFT but just a fusion algebra. By excluding all confined particles we arrive at a valid TQFT $\mathcal{U}$.

In a general, non-abelian, TQFT we say that a particle a *splits* or *restricts* to one or more particles t from $\mathcal{T}$. This means that a particle a becomes a superposition of particles t from $\mathcal{T}$. We write this as: $a \to \sum_{t \in \mathcal{T}} n_a^t t$. Here coefficients n_a^t are integers and tell us how many times a particle t is contained in a. On the other hand, we say that a particle t from $\mathcal{T}$ *lifts* to particles a from $\mathcal{A}$ and write $t \to \sum_{a \in \mathcal{A}} n_a^t a$. This tells us which particles from a restricts to t.

It is useful to think about a superposition of particles a as a vector $\vec{a}$ where each entry tells us how many times a particular particle appears in the superposition. We

define the n matrix as a matrix whose entries are n_a^t and write the restriction of a superposition of particles from $\mathcal{A}$ to a superposition of particles from $\mathcal{T}$ as $\vec{a} = n \cdot \vec{t}$. Similarly, we write the lifting process as $\vec{t} = n^T \cdot \vec{a}$. By removing rows and columns that correspond to the confined particles, we can analogously define the n matrix between $\mathcal{A}$ and $\mathcal{U}$ with $\vec{a} = n \cdot \vec{u}$.

We label the vacuum particle of $\mathcal{T}$ as ϕ. The dimensions of particles t need to be consistent with the restriction process — $d_a = \sum_{t \in \mathcal{T}} n_a^t d_t$. Additionally, we require that $d_\phi = 1$. To invert this equation, we define a *quantum embedding index* $q = \frac{\sum_{a \in \mathcal{A}} n_a^t d_a}{d_t}$. This turns out to be independent of t [14] and by setting $t = \phi$ we get $q = \sum_{a \in \mathcal{A}} d_a^\phi d_a$ — the sum of dimensions of condensed particles. Finally, for d_t we have $d_t = \frac{1}{q} \sum_{a \in \mathcal{A}} n_a^t d_a$.

The fusion algebra $\mathcal{T}$ needs to be consistent with the fusion rules of $\mathcal{A}$. This means that if we perform a fusion of particles a and b from $\mathcal{A}$, the result must be the same as if we first restrict a and b to particles from $\mathcal{T}$, fuse them according the the fusion rules of $\mathcal{T}$ and finally lift the resulting particles back to $\mathcal{A}$. Additionally, all dimensions d_t and d_a must be consistent with the corresponding restrictions and lifts. When searching for the fusion algebra $\mathcal{T}$ we need to make sure that it is does not violate these two conditions and, of course, that $n_J^\phi = 1$ which means that J is condensed.

3.4.2 Finding Lifting Coefficients

The main problem is that there is no known procedure that straightforwardly determines the possible coefficients n_a^u (now $u \in \mathcal{U}$) except in some simple cases. We used a combination of trial and error by hand and an algorithm that finds the possible candidates for n between $\mathcal{A}$ and $\mathcal{U}$. The algorithm is based on constructing M matrix whose entries are $M_{ab} = \sum_{u \in \mathcal{U}} n_a^u n_b^u$. Authors of [14] prove that this matrix commutes with S and T matrices of $\mathcal{A}$. Additionally, they prove that if $\tilde{S}$ and $\tilde{T}$ are

modular matrices of $\mathcal{U}$ we have that: $n\tilde{S} = Sn$ and $n\tilde{T} = Tn$. Finally, the algorithm constructs all valid M matrices, extracts n matrices and checks whether n matrices are valid. However, the resulting n matrices are not guaranteed to be valid solutions and still have to manually checked against the fusion rules. Another drawback of the algorithm is that it runs in exponential complexity in $|G|$ as we have no better way than iterating over all integer M matrices subject to the constraints of consistent quantum dimensions.

Overall, this is a highly non-trivial step and we are lucky to be working with a simple theory that consists of a small number of particles. It would be an interesting research problem to incorporate fusion rules into this algorithm and perhaps derive more constraints that would make the run-time polynomial.

3.4.3 Condensation of Magnetic Fluxes in Lattice Gauge Theories

It turns out that in the case of the lattice gauge theories, there is a very useful algebraic results that defines the resulting $\mathcal{U}$ theory when purely magnetic bosons condensate. It is a rather formal derivation [13] with a powerful conclusion. It says that if we want to condense a magnetic boson whose conjugacy class is A the resulting $\mathcal{U}$ theory is $D(G/K_A)$ where K_A is the smallest subgroup of G that contains all elements of A. This is a very nice result but it is specific to discrete lattice theories and still does not make the algorithm above run in the polynomial time as it only fixes the dimensions of n. This helps excluding potential candidates but does not help with brute-forcing possible M matrices which is the most computationally expensive step.

Chapter 4

Monte Carlo Methods

In this section we describe the theory behind numerical methods that we used. We start by giving a very quick introduction to the theory of Markov Chains [15] and then use its properties in order to formulate the Heat Bath algorithm [4] that is used for computing the expectation values of the operators. Working on this problem, we have stumbled upon a number of obstacles characteristic to it. Therefore, we decided to thoroughly discuss them in order for our work to be more accessible and reproducible.

4.1 Markov Chains

Markov chains are one of the simplest stochastic processes. Their main property, called a Markov property, is that they are memory-less, that is that the future states depend only on the current, but not the past states. We define a stochastic process as an indexed set of random variables $\{X_t\}_{t\geq 0}$. In case $t \in \mathbb{N}$, it is a discrete time Markov chain, and if $t \in \mathbb{R}$ we call it a continuous time Markov chain. We focus on discrete time Markov chains. Formally, Markov property can written as:

$$P(X_{k+1} = x_{k+1}|X_k = x_k, X_{k-1} = x_{k-1}, ..., X_0 = x_0) = P(X_{k+1} = x_{k+1}|X_k = x_k). \tag{4.1}$$

Here X_i are random variables and x_i are concrete values from a countable space on which random variables are defined. Note that this implies the same condition to hold for any X_n, $n \geq k$ not just X_{k+1}. We can understand this equation in a sense that only the latest information about the system matters when we want to predict some future state.

4.1.1 Markov Chain Dynamics

The key insight into the chain dynamics is to look at one step transitions. We define the transition matrix as:

$$P_{ij} = P(X_n = x_i|X_{n-1} = x_j). \tag{4.2}$$

Here, we use slightly different notation than above: $X_k \in \{x_1, x_2, ..., x_N\}$ and we say that the system at time k is in state i if $X_k = x_i$. Since Markov chains are memory-less, the transition matrix is constant and independent of n. To describe the dynamics of a Markov chain, suppose that at initial time $n = 0$ the system is described by some probability distribution $\pi_i^0 = P(X_0 = x_i)$. We call $\pi^0 = \{\pi_i^0\}_{i=0}^N$ a state vector, and think of it as a column vector. The probability distribution over states in the next time-step π_i^1 is given by the following equation:

$$\pi_i^1 = \sum_{j=0}^{N} P_{ij}\pi_j^0. \tag{4.3}$$

In plain words, the equation says that the probability of ending up in some state i is a sum over starting states j of the probabilities that state starts at j multiplied

by the probability of transitioning from j to i. Obviously, this can be written as a matrix equation $\pi^1 = P\pi^0$. Moreover, we also have $\pi^n = P\pi^{n-1}$, for any time-step n. Applying this equation inductively, we find that: $\pi^{n+k} = P^k\pi^n$.

Now, we see that just by knowing the transition matrix and the initial state vector, we can easily compute probabilities of being in any other state at any other time step.

4.1.2 Classification of States

In order to understand what can happen in a Markov chain, we provide a few definitions about the properties of the states. We say that a state i *leads* to the state j if the probability of reaching j after we reached i is non-zero. If two states lead to each-other we say that they *communicate*. It is interesting to note that we can break that chain into communicating classes which are equivalence classes as communication is an equivalence relation. A certain communicating class is *absorbing* if all states within the class lead only to the states within the same class. Similarly, a state is absorbing if it leads to no other states than itself. The chain itself is said to be *irreducible* if it consists of only one communicating class. We say that a state is *recurrent* state if the probability of it being visited infinitely many times is 1 after it has already been visited. On the contrary, the state is said to be *transient* which means that the recurrence probability is zero. Additionally, we call a state *positive recurrent* if the expected number of steps before returning the same state is finite. We define a state to be *aperiodic* if $(P^n)_{ii} \neq 0$ for every n. Finally, we say that the chain is *ergodic* if all states are positive recurrent and aperiodic.

4.1.3 Invariant Distributions

Equipped with the previous definitions, we can introduce the crucial theorems on whose grounds our numerical methods stand. Firstly, we say that a distribution λ is invariant if:

$$\lambda = P\lambda. \tag{4.4}$$

This means that if at any time-step n, $\pi^n = \lambda$, then the system's state vector remains λ forever.

Now we state crucial theorems regarding the invariant distribution [15]:

Theorem 1 *If a Markov chain is irreducible, the following statements are equivalent:*

i) There exist a positive recurrent state.

ii) All states are positive recurrent.

iii) There exist an invariant distribution.

This is a rather powerful theorem because it guarantees the existence of an invariant distribution. Our method aims to reach the invariant distribution starting from any distribution. The following theorem provides the conditions that need to be meet in order for that property to hold [15]:

Theorem 2 *If a Markov chain is irreducible, aperiodic and it has an invariant distribution λ, we have $\lim_{n\to\infty} \pi^n = \lambda$ regardless of π^0.*

Using the previous two theorems we can conclude that if the chain is ergodic, it has an invariant distribution and it converges to it as $n \to \infty$.

4.2 Markov Chain Monte Carlo

On a high level, our problem can be viewed as an attempt to evaluate equations 3.1 and 3.2. This might seem as a trivial problem since we have a closed form equation for $S(\{U\})$. However, doing it by brute force we are, actually, trying to solve an exponentially hard problem. Suppose that our lattice is $N \times N \times N$ then we have $3N^3$ links. Each link can have one of $|G|$ (group order) values. Therefore, the overall

cost of the algorithm is $|G|^{3N^3}$. In particular, for $G = S_3$ and $N = 5$ we need to perform about $6.41 \cdot 10^{291}$ iterations. There is not a computer in the world today that can finish this calculation before the end of the universe. Therefore, we have to use approximate methods.

Based on the theory of Markov chains, we want to find a Markov chain whose invariant distribution is the Boltzman distribution of the system: $P(\{U\}) = \frac{e^{-S(\{U\}}}{\mathcal{Z}}$. If we achieve that, we can let the chain evolve, record the values of the operators and take the average in the end. The simple average works because, ultimately, the frequency with which we visit each state is proportional to the Boltzman probability. This may seem a bit sketchy as it is obvious that we still cannot visit all the states. However, the probability is exponential which means that only states around the maximum of the probability distribution function significantly contribute to the relevant expectation.

Another relevant application of this method is that it can be used to optimize a function defined over variables taking values from some finite sets. Suppose that we have a function $E(\{U\})$ where $\{U\}$ is a collection of variables where each U variable can have a finite number of different values. Then, we can define an analogous physical system with degrees of freedom being U variables and energy being the function we want to minimize. Then by constructing a Markov chain, like the one described above, and recording the values of E as the chain evolves we will eventually pick up the minimal value. Moreover, the minimal value should be the one that appears the most in this sequence of energies.

4.2.1 Constructing the Chain

It turns out that there are many ways of constructing the chain with the desired invariant probability distribution. In fact, different chains might have different properties such as the computational time, parallelisability of the algorithm and relaxation time

(expected time needed for reaching the invariant distribution from a random initial distribution).

Consider the equation 4.4. The i-th component of the RHS can be expanded as $\sum_j P_{ij}\lambda_j$. The same componenet of the LHS is simply equal to λ_i. Using $\sum_j P_{ji} = 1$ we can rewrite it as $\sum_j \lambda_i P_{ji}$. Therefore, the equation 4.4 can be rewritten as:

$$\sum_j (\lambda_i P_{ji} - \lambda_j P_{ij}) = 0. \tag{4.5}$$

One way to solve this equation is to impose, so called detailed balance condition:

$$\frac{P_{ij}}{P_{ji}} = \frac{\lambda_i}{\lambda_j}. \tag{4.6}$$

In our case, this translates to:

$$\frac{P_{ij}}{P_{ji}} = e^{S(\{U\}_j) - S(\{U\}_i)}. \tag{4.7}$$

Where $S(\{U\}_i), S(\{U\}_j)$ are actions for configurations of links $\{U\}_i$ and $\{U\}_j$ respectively.

4.2.2 Heat Bath Algorithm

Now we turn to the Heat Bath algorithm [4] which is a specific Markov Chain Monte Carlo method that we used to compute the expectation of the relevant operators.

The algorithm works as follows. We start with a configuration where each link has some random value. We traverse all links U_k for $k \in \{1, 2, ..., 3N^3\}$. Let S^k be the local action which equals the action term that contains U_k, this term contains the sum of actions of four plaquettes that include the link U_k. For each group element g_i for $i \in \{1, 2, ..., |G|\}$, we compute the value of $S^k(U_k = g_i)$ and the local partition function $\mathcal{Z}_k = \sum_{i=1}^{|G|} e^{-S^k(U_k = g_i)}$. Finally, we generate a new value for U_k from a

probability distribution given by $P(U_k \to g_i) = \frac{e^{-S^k(U_k=g_i)}}{\mathcal{Z}_k}$. Note that while we are generating a new value for U_k all other links have fixed values generated by all the previous iterations. This process of visiting all links is called a *sweep*. One sweep is considered to be one time-step of the Markov chain and after each sweep we record the value of the relevant operator. Finally, the expectation of the operator is the average of the recorded values. By evolving the chain in this way, the transition probability between any two states is positive. Therefore, the chain is ergodic and it reaches the invariant distribution.

The algorithm, described above, works perfectly if we record infinitely many values. However, if the number of time-steps that we use is comparable to the relaxation time we might not get accurate results. We define the *relaxation time* as an expected number of iterations we need to perform, starting from a random state, in order to get close to the minimal action state. During the relaxation time, the system goes through some very unlikely states and if the value of the operators are extreme in those states the measurement will be disturbed significantly. To fix this, prior to any experiment we first estimate the relaxation time and then start evaluating and recording the values of the operators only once we reached that number of iterations. In figure 4.2.2, we see that in a specific case the relaxation time can be estimated to be 100.

Another common algorithm is a Metropolis-Hastings algorithm [4]. This method makes a transition with probability 1 if it is energetically favourable: $S(\{U\}_j) - S(\{U\}_i) \geq 0$ and with probability $e^{S(\{U\}_j)-S(\{U\}_i)}$ otherwise. Even though, this algorithm has smaller relaxation time, the Heat Bath algorithm is better at exploring the space and thus less likely to get trapped around locally optimal states.

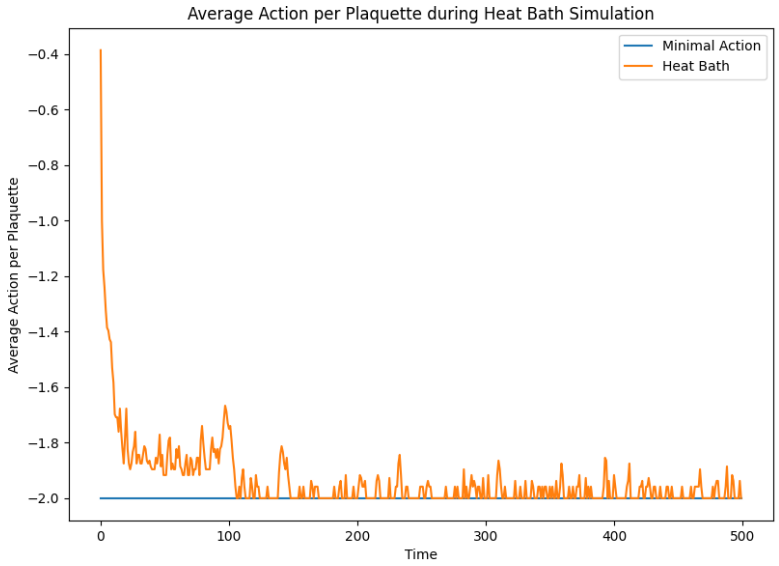

Figure 4.1: $D(S_3)$ Heat Bath Simulation on a 4^3 lattice with $\beta_e = 2$, $\beta_\tau = 0$, $\beta_\sigma = 0$

4.3 Estimating Errors

We need to know, how confident we can be in our results. Since we are essentially computing the average of random values (our measurements) we can use the central limit theorem. Let O be a random variable that represents our operator. Then, the mean of the values we measure, using equation 3.2, should be distributed as:

$$\langle O \rangle \sim \mathcal{N}(\mathbb{E}[O], \frac{Var[O]}{n}). \tag{4.8}$$

To estimate the average we can use the standard maximum likelihood estimator $\mathbb{E}[O] \approx \frac{1}{n}\sum_{i=0}^{n} O_i$. To estimate the variance we could use the unbiased estimator given by: $Var[O] \approx \frac{1}{n-1}\sum_{i=0}^{N}(O_i - \frac{1}{n}\sum_{j=0}^{n} O_j)^2$.

The expectation formula is always true. However, the variance formula is valid only when the samples O_i are independent. Unfortunately, Markov Chain methods rely on the evolution of a chain which samples a new state from the previous one. This makes the samples highly correlated and formula practically useless. For example,

when we measure open string operators and apply 4.8, in some cases, the 1 sigma interval goes way beyond the domain of possible values.

An unbiased estimator for the variance of the asymptotic distribution of $\langle O \rangle$ is given by [8]:

$$Var[\langle O \rangle] = Var[O_1] + 2 \cdot \sum_{i=2}^{\infty} Cov[O_1, O_i]. \tag{4.9}$$

There are multiple ways of estimating this variance. For simplicity, we decided to use non-overlapping batch means (BM) [9]. Practically, we assume that there is some relaxation time τ such that $Cov[O_i, O_{i+\tau+k}] = 0$ for $k \geq 0$. We first estimate the value of τ and then divide measurements in batches of length τ. We compute means in each batch $\tilde{O}_k = \frac{1}{\tau} \sum_{i=1}^{\tau} O_{(k-1)b+i}$ for a number of batches b and then apply 4.8 on $\{\tilde{O}_k\}_{k=1}^{b}$ to find the variance of these means $Var[O]_\tau$. Since, we estimate both mean and variance we use Student's t distribution to give confidence intervals. Specifically, α confidence interval is given by:

$$\frac{\sum_{i=1}^{n} O_i}{n} \pm t_{b-1,1\pm\alpha/2} \sqrt{\frac{Var[O]_\tau}{b}}. \tag{4.10}$$

We estimate the value of τ to be around 500 sweeps. Additionally, we use the equation 4.10 to determine the stopping time of the simulation. For example, by fixing the threshold ratio of the width of the 95% confidence interval and the mean value, we stop the simulation once the measured ratio is below the threshold.

Chapter 5

$D(S_3)$ Quantum Group

We want to use all the theory we have developed so far in order to study the condensation of magnetic bosons in non-abelian lattice gauge theories. The smallest non-abelian group is S_3, the permutation group of order 3. In this chapter, we use the theory from Chapter 3 in order to make analytical predictions of relevant quantities and then use methods from Chapter 4 to numerically probe the system and describe the phase transitions.

5.1 S_3 Group

A general permutation group S_N is a group of $N!$ elements that represent the permutations of N distinct objects. For example, S_2 can be represented as:

$$S_2 = \left\{ \left(\begin{smallmatrix} 1 & 2 \\ 1 & 2 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 2 \\ 2 & 1 \end{smallmatrix}\right) \right\}. \tag{5.1}$$

In general a member σ of S_N is a bijective function $\sigma : \{1, 2, ..., N\} \to \{1, 2, ..., N\}$. In the notation used in 5.1, we write this element as:

$$\sigma = \left(\begin{smallmatrix} 1 & 2 & \cdots & N \\ \sigma(1) & \sigma(2) & \cdots & \sigma(N) \end{smallmatrix}\right), \tag{5.2}$$

The identity element is the one for which $\sigma(i) = i$ and since the function σ is bijective the inverse always exists and we label it by σ^{-1}. We define the sign of the permutation as:

$$sgn(\sigma) = \Pi_{i<j} \frac{\sigma(j) - \sigma(i)}{j - i} \tag{5.3}$$

Turning our attention to S_3 we have:

$$S_3 = \left\{ \left(\begin{smallmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{smallmatrix}\right) \right\}. \tag{5.4}$$

Since we will be working with these elements a lot, we give them special labels:

$$S_3 = \{e, \sigma, \sigma^2, \tau_1, \tau_2, \tau_3\}. \tag{5.5}$$

One can think of τ_i as permuting the two elements other than i, σ as cyclically shifting all elements to the left and σ^2 as cyclically shifting all elements to the right or equivalently applying σ twice.

In this new notation, we can write the multiplication table as:

	e	σ	σ^2	τ_1	τ_2	τ_3
e	e	σ	σ^2	τ_1	τ_2	τ_3
σ	σ	σ^2	e	τ_2	τ_3	τ_1
σ^2	σ^2	e	σ	τ_3	τ_1	τ_2
τ_1	τ_1	τ_3	τ_2	e	σ^2	σ
τ_2	τ_2	τ_1	τ_3	σ	e	σ^2
τ_3	τ_3	τ_2	τ_1	σ^2	σ	e

(5.6)

Using multiplication table 5.6 we find that there are 3 conjugacy classes. These are:

$$
\begin{aligned}
C^e &= \{e\}, \\
C^\sigma &= \{\sigma, \sigma^2\}, \\
C^\tau &= \{\tau_1, \tau_2, \tau_3\}.
\end{aligned}
\tag{5.7}
$$

The corresponding centralizers are:

$$
\begin{aligned}
X_e &= S_3, \\
X_\sigma &= \{e, \sigma, \sigma^2\}, \\
X_\tau &= \{e, \tau_1\}.
\end{aligned}
\tag{5.8}
$$

It is useful to note that $X_\sigma \cong \mathbb{Z}_3$ and $X_\tau \cong \mathbb{Z}_2$. Thus, their irreducible representations are:

$$
\begin{aligned}
0_{\mathbb{Z}_2} &: e \to 1, \tau_1 \to 1, \\
1_{\mathbb{Z}_2} &: e \to 1, \tau_1 \to -1.
\end{aligned}
\tag{5.9}
$$

for X_τ and:

$$
\begin{aligned}
0_{\mathbb{Z}_3} &: e \to 1, \sigma^2 \to 1, \sigma \to 1, \\
1_{\mathbb{Z}_3} &: e \to 1, \sigma^2 \to e^{i\frac{2\pi}{3}}, \sigma \to e^{i\frac{4\pi}{3}}, \\
2_{\mathbb{Z}_3} &: e \to 1, \sigma^2 \to e^{i\frac{4\pi}{3}}, \sigma \to e^{i\frac{2\pi}{3}}.
\end{aligned}
\tag{5.10}
$$

for X_σ. The numbers $0, 1$ and $0, 1, 2$, in the examples above, refer to the charge q of the representation $\mathbb{Z}_N$, so that $q_{\mathbb{Z}_\mathbb{N}}(k) = e^{i\frac{2\pi q}{N}}$.

Group S_3 has two 1D irreducible representations and one 2D irreducible representation. Apart from the trivial one $1_{S_3}(\sigma) = 1$, one can easily show that $s_{S_3}(\sigma) = sgn(\sigma)$ is also an irreducible representation. The 2D representation is given by the following map:

$$
\begin{aligned}
2_{S_3} : e \to \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \sigma \to \begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}, \sigma^2 \to \begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}, \\
\tau_1 \to \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix}, \tau_2 \to \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}, \tau_3 \to \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix}.
\end{aligned} \tag{5.11}
$$

Here we chose to label these representation with $1, s$ and 2.

5.2 Particles of $D(S_3)$

Finally, we can begin to identify particles in $D(S_3)$. Recall that the particles are described by a conjugacy class A and an irreducible representation of its centralizer α — (A, α). In $D(S_3)$, there is vacuum, two electric particles, two magnetic particles and three dyons. We label them as follows:

$$
\begin{aligned}
&I = (C^e, 1_{S_3}) \\
&e_s = (C^e, s_{S_3}), e_2 = (C^e, 2_{S_3}) \\
&\Sigma_0 = (C^\sigma, 0_{\mathbb{Z}_3}), \mathrm{T}_0 = (C^\tau, 0_{\mathbb{Z}_2}) \\
&\Sigma_1 = (C^\sigma, 1_{\mathbb{Z}_3}), \Sigma_2 = (C^\sigma, 2_{\mathbb{Z}_3}), \mathrm{T}_1 = (C^\tau, 1_{\mathbb{Z}_2}).
\end{aligned} \tag{5.12}
$$

5.2.1 Spins, S Matrix and Fusion Rules

Using the equation 3.25, we compute the spins of the particles and present them in the table 5.1.

I	e_s	e_2	Σ_0	Σ_1	Σ_2	T_0	T_1
1	1	1	1	$e^{i\frac{2\pi}{3}}$	$e^{-i\frac{2\pi}{3}}$	1	-1

Table 5.1: Spins of particles in $\mathrm{D(S_3)}$

We see that there are 5 bosons (including I), 1 fermion and 2 anyons. Out of

these 5 bosons, 2 are purely magnetic excitations: Σ_0 and T_0. Therefore, we will be looking at their condensates.

Based on the equation 3.26 from Chapter 3, we compute the S matrix. The results multiplied by the total quantum dimension of the theory are shown in the table 5.13.

	I	e_s	e_2	Σ_0	Σ_1	Σ_2	T_0	T_1
I	1	1	2	2	2	2	3	3
e_s	1	1	2	2	2	2	-3	-3
e_2	2	2	4	-2	-2	-2	0	0
Σ_0	2	2	-2	4	-2	-2	0	0
Σ_1	2	2	-2	-2	4	-2	0	0
Σ_2	2	2	-2	-2	-2	4	0	0
T_0	3	-3	0	0	0	0	3	-3
T_1	3	-3	0	0	0	0	-3	3

(5.13)

We can immediately conclude some interesting properties. First of all, we can read off the quantum dimensions by looking at the first row. This allows us to compute the total quantum dimension of the theory: $\mathcal{D}_{S_3} = 6$. Next, we can see that particle e_s has the same effect as vacuum on e_2 and Σ particles and that e_s picks up a minus sign when braided around T particles. Therefore, we call this particle a quasi-vacuum. Finally, we see that the S matrix element between any two different Σ particles gives -2 and between different T particles it is -3, whereas Σ and T particles do not braid with each other (twrapping Σ around T gives a perfect destructive interference).

In the end, we use the relationship between fusion matrices and the S matrix 2.8 to obtain the fusion rules of the theory. The fusions are determined by the following rules:

$$
\begin{aligned}
&I \times a = a; \\
&e_s \times e_s = I; \\
&e_s \times \mathrm{T}_i = \mathrm{T}_j,\ i \neq j; \\
&e_s \times a = a,\ a \neq I, e_s, \mathrm{T}; \\
&\mathrm{T}_i \times \mathrm{T}_i = I + e_2 + \Sigma_0 + \Sigma_1 + \Sigma_2; \\
&\mathrm{T}_0 \times \mathrm{T}_1 = e_s + e_2 + \Sigma_0 + \Sigma_1 + \Sigma_2; \\
&\mathrm{T}_i \times a = \mathrm{T}_0 + \mathrm{T}_1,\ a \neq \mathrm{T}_i; \\
&\Sigma_i \times \Sigma_i = I + e_s + \Sigma_i; \\
&\Sigma_i \times \Sigma_j = e_2 + \Sigma_k; \\
&\Sigma_i \times e_2 = \Sigma_j + \Sigma_k; \\
&e_2 \times e_2 = I + e_s + e_2.
\end{aligned}
\tag{5.14}
$$

It is useful to note that the quasi-vacuum e_s has the same fusion effect as the vacuum I on all particles except T_i. Additionally, there is a clear symmetry between fusion rules of different Σ_i particles and between fusion rules of different T_i particles. Finally, we want to point out that every particle in the theory is its own anti-particle.

5.3 Condensation

In this section, we attempt to map out the phase space of the theory and explain when condensations of Σ_0 and T_0 occur. We identified 4 possible phases: trivial phase, phase in which Σ_0 is the only condensed particle, both Σ_0 and T_0 are condensed, and both T_0 and e_2 are condensed.

To look for phase transitions we plot the average free energy per plaquette against the coupling constants and its first derivatives with respect to the coupling constants. These derivatives are flux concentrations or percentages of plaquettes having a particular flux and we label them as $\langle \delta_X \rangle$ for a particular conjugacy class X. To confirm the theoretical predictions we evaluate open string operators and show that they acquire

non-zero values when the corresponding particles are condensed.

Finally, to determine the order of a phase transition we look for coexistence of phases at the critical point in the space of coupling constants. More precisely, we run algorithm multiple times and look at the histogram of measured values of $\langle \delta_X \rangle$. If it has two peaks, it is the first order, otherwise it is the second order transition.

In topological quantum field theories, values of the diagrams do not depend on the scale of the paths. Authors who first applied this method to a lattice gauge theory $D(\bar{D}_2)$, reported that the expectation values do not depend of the size of these in numerical simulations either [3]. Therefore, we performed calculations on a relatively small sized lattices $N \leq 6$.

5.3.1 Σ_0 condensation

Using the results from the Condensation section of chapter 3, we predict that the corresponding $\mathcal{U}$ theory, when Σ_0 is condensed, is $D(\mathbb{Z}_2)$. We give a full set of lifts (n_a^t coefficients):

$$\begin{aligned} & I \to \phi, \\ & \Sigma_0 \to \phi + e_s, \\ & e_s \to e_s, \\ & e_2, \Sigma_1, \Sigma_2 \to X, \\ & \mathrm{T}_0 \to \mathrm{T} + t_0, \\ & \mathrm{T}_1 \to \mathrm{T} + t_1. \end{aligned} \tag{5.15}$$

The embedding index is $q = 3$ and the consistent dimensions of new particles are $d_X = 2$, $d_T = 2$, $d_{t_i} = 1$. Based on the twist factors we see that particles X and T are confined, while particles ϕ, e_s, t_0 and t_1 are unconfined particles and correspond to the particles of $D(\mathbb{Z}_2)$ theory.

5.3.2 T_0 condensation

When T_0 is condensed we predict that the resulting $\mathcal{U}$ theory is a trivial TQFT (vacuum only). However, we find two possible ways how this can be realised:

$$\begin{aligned} &I \to \phi, \\ &\Sigma_0 \to \phi + e_s, \\ &\mathrm{T}_0 \to \phi + X \\ &e_s \to e_s, \\ &e_2, \Sigma_1, \Sigma_2 \to X, \\ &\mathrm{T}_1 \to X + e_s \end{aligned} \tag{5.16}$$

and

$$\begin{aligned} &I \to \phi, \\ &e_2 \to \phi + e_s, \\ &\mathrm{T}_0 \to \phi + X \\ &e_s \to e_s, \\ &\Sigma_0, \Sigma_1, \Sigma_2 \to X, \\ &\mathrm{T}_1 \to X + e_s. \end{aligned} \tag{5.17}$$

In both cases $q = 6$ and consistent dimensions are $d_{e_s} = 1$ and $d_X = 2$. Obviously, e_s and X are confined.

5.3.3 Simulation Results

To look for a phase transition we first calculate the average free energy per plaquette $\langle F \rangle$ and average flux concentrations for each conjugacy class X: $\langle \delta_X \rangle$. Here $\langle \delta_X \rangle = \frac{1}{3N^3} \sum_{plaquettes\ p} \delta_X(U_p)$ which is a percentage of plaquettes having the flux X. It is hard to visualize the whole phase space and therefore, we provide results in $\beta_e - \beta_\tau$ and $\beta_e - \beta_\sigma$ planes.

Firstly, we show free energy surface plots in figures 5.1 and 5.2.

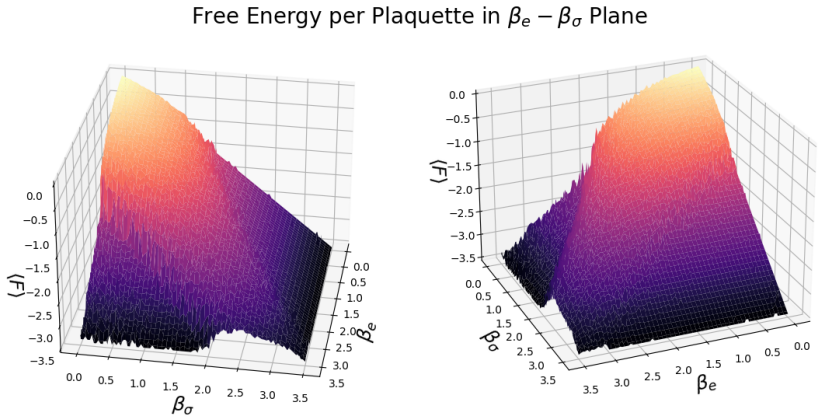

Figure 5.1: Free energy over a region where $\beta_\tau = 0$. The simulations were done on a 4^3 lattice, with a total of 100^2 sampling points, relaxation time of 300 sweeps and 1000 sweeps per point.

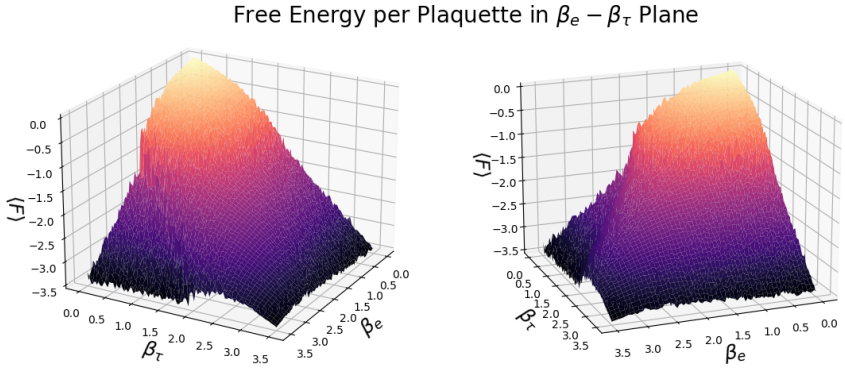

Figure 5.2: Free energy over a region where $\beta_\sigma = 0$. The simulations were done on a 4^3 lattice, with a total of 100^2 sampling points, relaxation time of 300 sweeps and 1000 sweeps per point.

Two plots look very similar. It is useful to note that the surface must by bounded by $\max(\beta_e, \beta_\sigma, \beta_\tau)$. Moreover, if only one coupling constant β is non-zero and large, we should have $\langle F \rangle \to -\beta$. We can indeed verify this by looking at the edges of these plots. This is a good sign that the simulation produces reasonable results.

Next, we see that the surfaces are mostly smooth. There are a few regions where

there seem to be sharp drops. These regions correspond to phase transitions. It is not uncommon for physical quantities to become ill defined at the critical points of phase transition and therefore, it should not be surprising that the Monte Carlo simulation is less accurate there.

To examine the space diagram more closely, we plot the flux concentrations in these planes. We first present and discuss the results in $\beta_e - \beta_\tau$ plane.

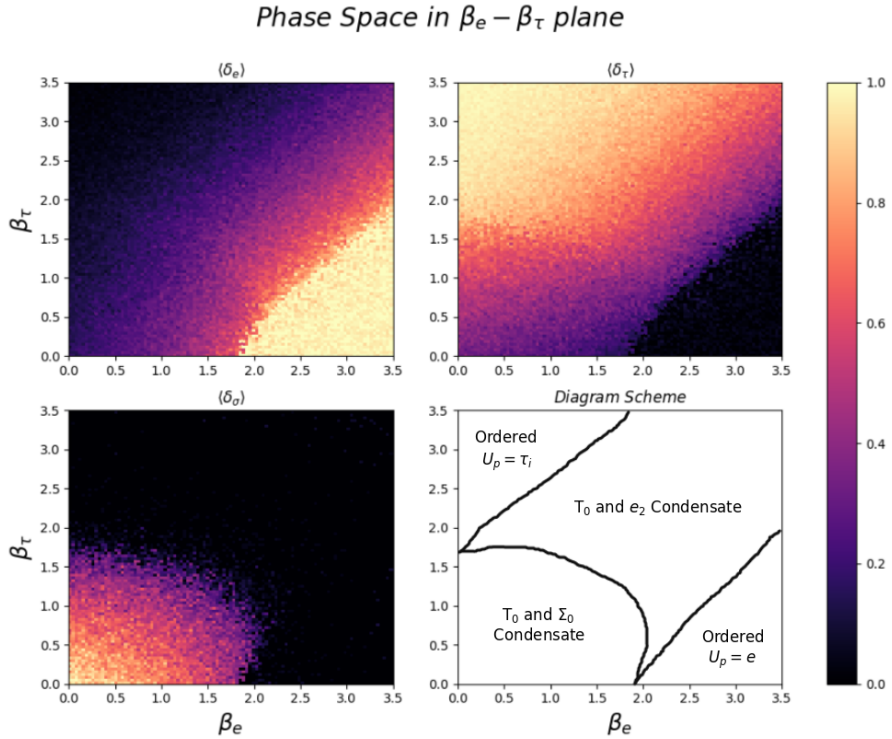

Figure 5.3: Average flux concentrations over a region where $\beta_\sigma = 0$ and diagram scheme. The simulations were done on a 4^3 lattice, with a total of 100^2 sampling points, relaxation time of 300 sweeps and 1000 sweeps per point.

As we can see in figure 5.3, we can roughly distinguish 4 regions in this phase space plane. The easiest regions to explain are ordered e (vacuum) and τ phases. These happen for $\beta_e >> \beta_\tau$ and $\beta_\tau >> \beta_e$. In these phases, (almost) all plaquettes have values equal to e and τ_i respectively. The plot of $\langle\delta_\sigma\rangle$ indicates that the region in between consists of two phases. This is somewhat harder to see just by looking at the other two plots. By considering the algebraic results from the analysis of T_0 we

can conclude that the region closer to the origin corresponds to the phase where T_0 and Σ_0 are condensed. In the region further away from the origin we see that there is no σ flux in plaquettes. Therefore, Σ_0 must be confined and we conclude that the region where there is no σ flux in plaquettes corresponds to a broken phase where T_0 and e_2 are condensed.

Next, we look at the phase space in $\beta_e - \beta_\sigma$ plane.

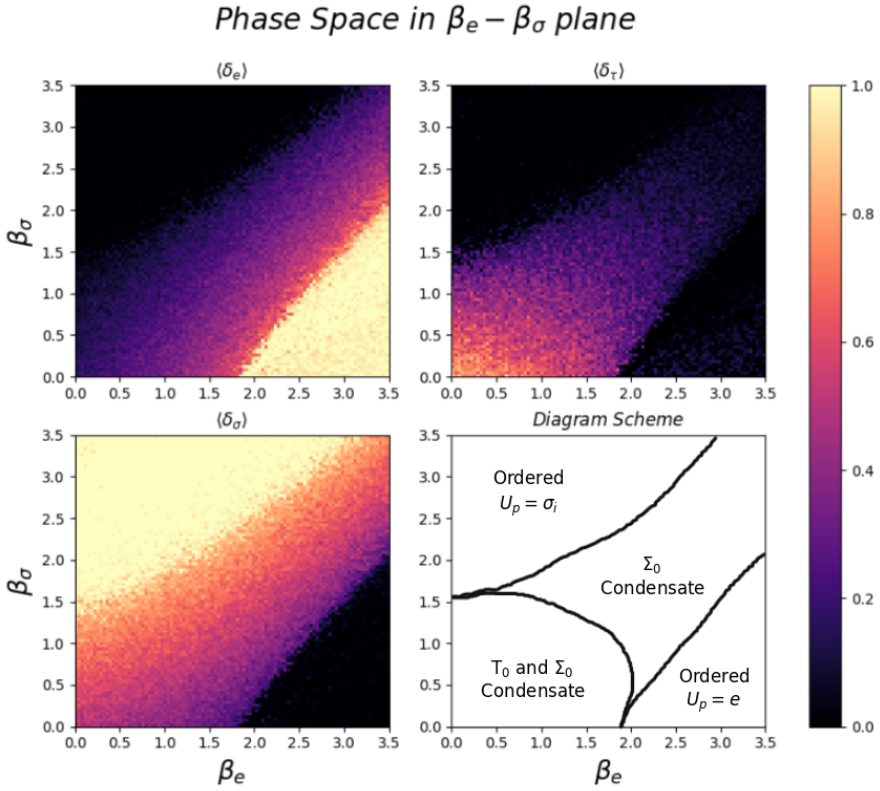

Figure 5.4: Average flux concentrations over a region where $\beta_\tau = 0$ and diagram scheme. The simulations were done on a 4^3 lattice, with a total of 100^2 sampling points, relaxation time of 300 sweeps and 1000 sweeps per point.

By looking at this diagram (figure 5.4) alone, our first guess could be that there are only 3 phases. Ordered e, σ phases and one broken phase that corresponds to the Σ_0 condensate. However, the diagram should look the same as the one in $\beta_e - \beta_\tau$ plane in the limiting case when all $\beta \to 0$. Thus, we conclude that the region near the origin again corresponds to the phase where T_0 and Σ_0 are condensed.

We would like to point out that this behaviour, that all fluxes are condensed near the origin of the phase space, was also observed in the numerical analysis of $D(\bar{D}_2)$ lattice gauge theory [3]. Intuitively, this makes sense as when all coupling constants are zero, link and plaquettes values are all random. This allows all fluxes to be present and hence, all magnetic particles are condensed.

Combining the plots and theory we proposed which parts of the phase spaces corresponds to which phase. However, we have to confirm this by explicitly evaluating open string operators. In a region where a particular flux is condensed, we expect its value to be equal to 1 and 0 when it is not.

In figures 5.5 and 5.6, we see that our educated guesses are indeed correct. We see that as T_0 condensates out of vacuum its open string operator acquires the value of 1, whereas the value of Σ_0 open string is constantly 0 along that trajectory. On the other hand, along the line $\beta_e = \beta_\sigma$, at $\beta_\tau = 0$, Σ_0 open string is constant and equal to 1, while T_0 open string goes from 1 to 0. This corresponds to a phase transition from a trivial TQFT where T_0 and Σ_0 are condensed to $D(\mathbb{Z}_2)$ where Σ_0 is condensed.

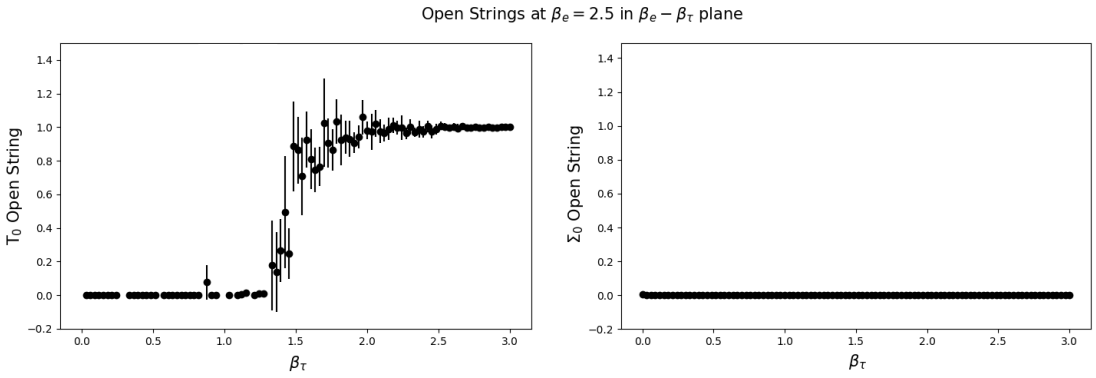

Figure 5.5: T_0 and Σ_0 open string operators as a function of β_τ at $\beta_e = 2.5$ and $\beta_\sigma = 0$. 100 data points obtained on a 5^3 lattice with 5000 iterations each, divided into 10 batches. The displayed errors correspond to 95% confidence intervals. Extreme points with enormous errors were discarded.

The errors displayed correspond to the 95% confidence intervals described in Chapter 4. As we pointed out earlier, the values at the phase transition might be unstable. This is indeed what happens with the operators that change the value along the phase

transition. It might seem that the errors are very large, however the values for small and large values of coupling constant are good enough indicators of the phases and the actual values in the critical regions are not as important.

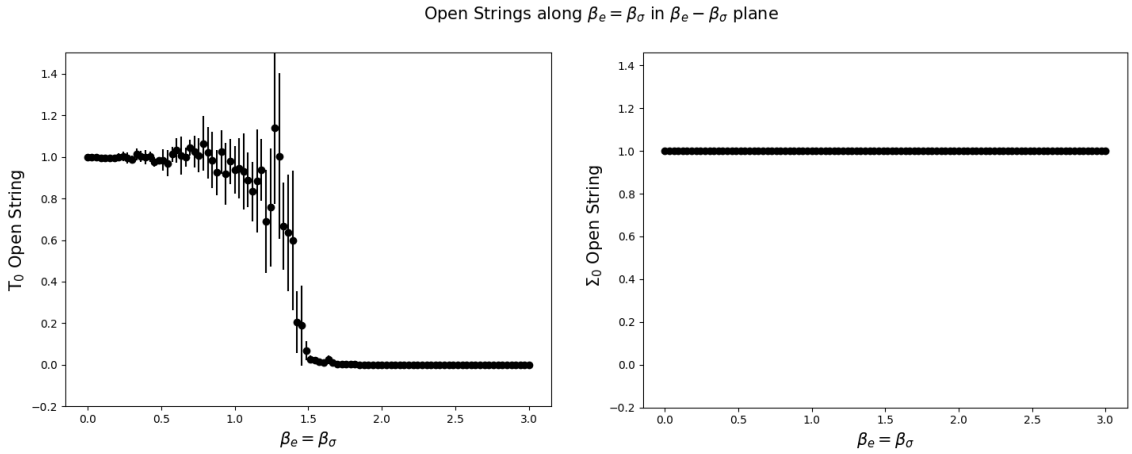

Figure 5.6: T_0 and Σ_0 open string operators as a function of $\beta_e = \beta_\sigma$ at $\beta_\tau = 0$. 100 data points obtained on a 5^3 lattice with 5000 iterations each, divided into 10 batches. The displayed errors correspond to 95% confidence intervals. Extreme points with enormous errors were discarded.

Finally, we establish the orders of phase transitions from vacuum to the broken phases. To do this, we look if there is a coexistence of phases at the critical region of the phase transition.

In figure 5.7, we see the histograms for the concentration of the trivial flux along the phase transition. In the phase where all fluxes are condensed the flux is usually around 0.6. On the other hand, in vacuum the flux is near 1.0 as expected. At the critical value of $\beta_e = 1.825$ and $\beta_\tau = \beta_\sigma = 0$ we see two peaks around 0.6 and 1.0. This indicates the coexistence of phases which is a clear sign of a first-order transition.

Next, we look at the phase transition from vacuum to the phase where only Σ_0 is condensed (figure 5.8). In this case, the peaks are closer together but there is still a coexistence of phases at the value of $\beta_\sigma = 1.43$. Therefore, this is also a first order phase transition.

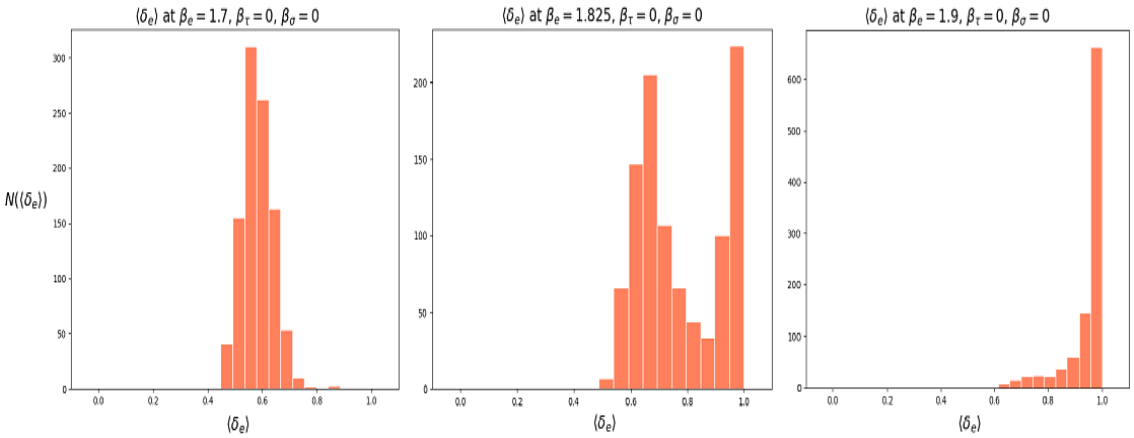

Figure 5.7: A histogram of $\langle \delta_e \rangle$ values at the phase transition along β_e axis at $\beta_\tau = \beta_\sigma = 0$. 1000 measurements obtained on a 4^3 lattice with 300 iterations relaxation time and 1000 iterations used for calculating MC average.

The last phase transition, from vacuum to the phase where T_0 and e_2 are condensed (figure 5.9) depicts a first order phase transition as the peak continuously moves the value from 1.0 to about 0.7.

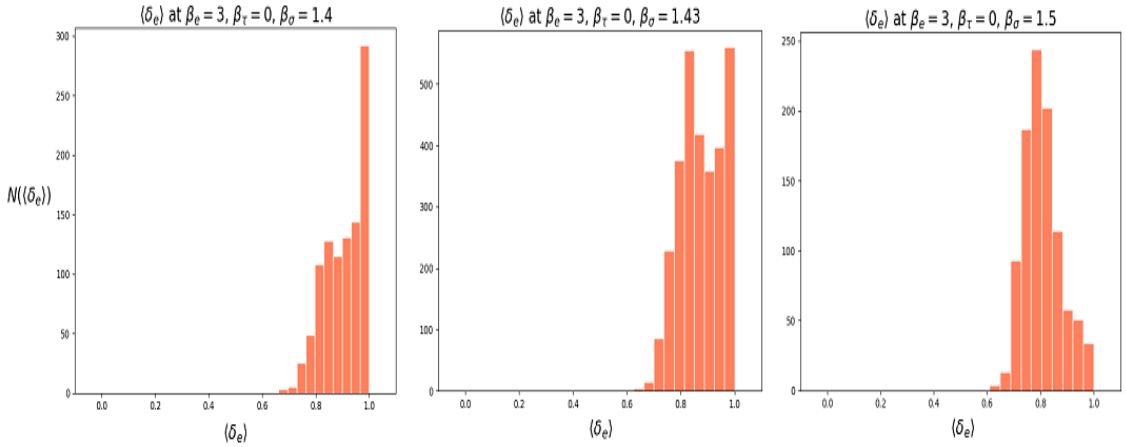

Figure 5.8: A histogram of $\langle \delta_e \rangle$ values at the phase transition along β_σ axis at $\beta_e = 3$ and $\beta_\tau = 0$. 1000 measurements obtained on a 4^3 lattice with 300 iterations relaxation time and 1000 iterations used for calculating MC average.

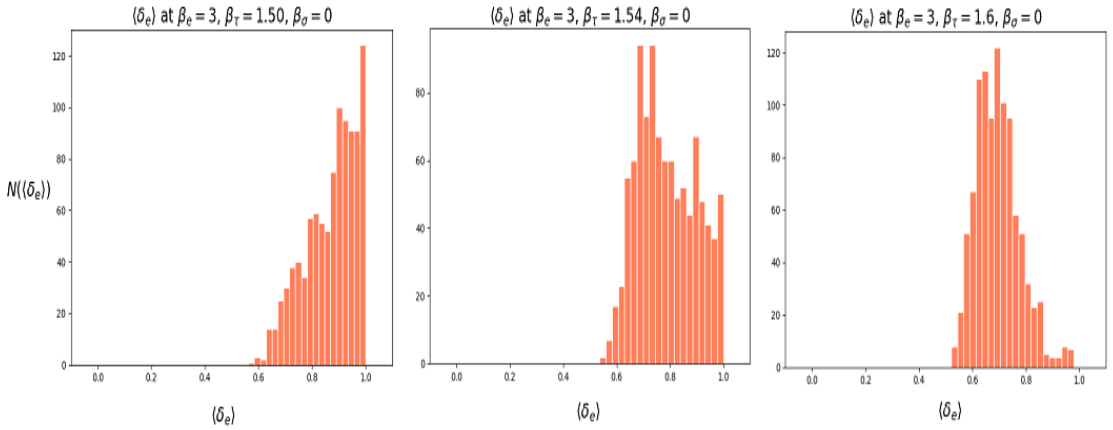

Figure 5.9: A histogram of $\langle \delta_e \rangle$ values at the phase transition along β_τ axis at $\beta_e = 3$ and $\beta_\sigma = 0$. 1000 measurements obtained on a 4^3 lattice with 300 iterations relaxation time and 1000 iterations used for calculating MC average.

Chapter 6

Conclusion

In this dissertation, we presented our research on the phase transitions in S_3 lattice gauge theory. We started by outlining motivation for topological quantum field theories and giving a brief overview of general anyon theories. Next, we introduced necessary concepts of lattice gauge theories mainly based on a physical approach. We also gave a self-contained chapter on the Monte Carlo method that we used starting with the basics of Markov Chains and ending with the discussion of errors of our estimates. Finally, we specialised towards a lattice gauge theory where the underlying gauge group is S_3. We identified magnetic bosons which condense and the resulting theories algebraically. We searched for these condensates numerically, determined the regions in the space of coupling constants in which they occur by looking at the derivatives of free energy and by evaluating open string operators. Finally, we determined the orders of phase transitions by looking for the coexistence of phases at the values of coupling constants for which the phase transitions occur.

Additionally, we would like to emphasize that the code we built for our simulation is very general and immediately applicable to other gauge groups. To verify it, we reproduce some of the results from [3]. In order to obtain the results we presented, one only needs to input the group multiplication table. To compute all S matrix

elements one also needs to input the irreducible representations of the centralizers of conjugacy classes. Unfortunately, the run-time of the code is $O(|G| \cdot N^3)$ per Monte Carlo sweep, assuming that the computation of measurements takes the constant time. For some quantities such as average flux concentrations or free energy there is additional factor of N^3 per iteration. Luckily, we do not need to use large lattices. Even though the scaling with the group order seems to be good, working with larger groups becomes a tedious job. This is because there are more coupling constants and particles and thus, the amount of data analysis needed grows significantly. We believe that this poses a greater challenge in exploring other groups than the run-time itself.

To conclude, we suggest a few possible directions in which this research could be continued. In Chapter 3, we pointed out that one could try to devise a polynomial algorithm for determining the resulting condensed theories when a boson condenses in an anyon theory. Monte Carlo algorithm itself could be parallelised as updating links that are do are not contained in the same plaquette are independent processes. Finally, there are hints that the order of phase transition could be linked to the fusion algebra, studying other groups using this method could possibly open the door of a theory that would explain this.

Appendix A

Code

A significant amount of work was needed in order to built the code that probed the phase transitions. Initially, we constructed our system and built the algorithm in Python. Once we had a working bug-proof version of the code, we rewrote it in C++ in order to obtain good quality data in reasonable time.

Python implementation mainly uses NumPy package. We also used Matplotlib to visualize the lattice and diagrams we were interested in and to plot the results. Finally, we used SciPy to compute confidence intervals 4.10 of the data generated by $C++$ program. $C++$ implementation uses only built-in STL library that comes with Visual Studio 2019. Besides basic data structures such as vectors, we used $\langle$future$\rangle$ library to parallelise the computation on multiple logical units. All simulations were run on a home laptop with a 10-th generation i7 processor with 6 cores and 12 logical units.

We want to emphasize that we invested a large amount of time testing the functions. We compared results against a numerous example done by hand. We verified the correctness of geometrical functions by visualizing the lattice using Matplotlib (see figure A). We also run our code against quaternion group $\bar{D}_2$ and confirmed the previously reported results [3].

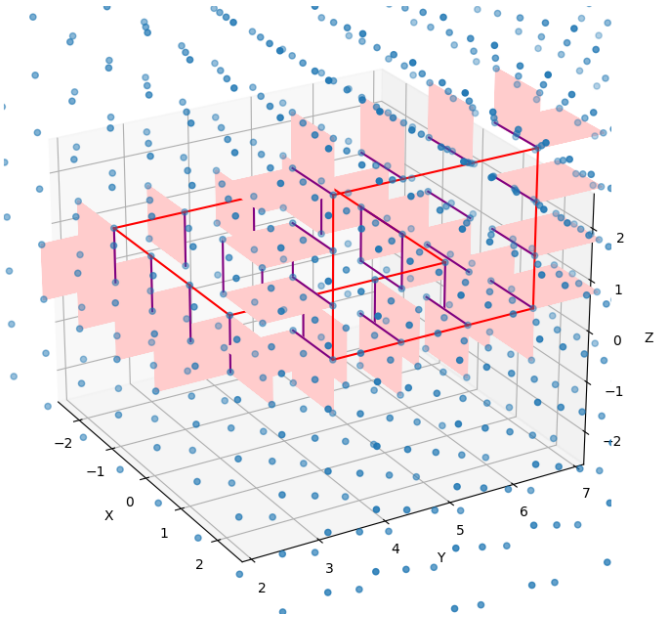

Figure A.1: Visualization of an S matrix element. Lattice is represented by blue dots. Links of the loops are red. Links inside the loops are purple. Corresponding plaquettes are light red.

In the first section we describe how we approached the geometry of the system. In the second section we show header files of the $C++$ programs and discuss code logic and organization. We conclude this appendix with function declarations of the non-trivial functions that we implemented.

A.1 Geometry

As far as geometry is concerned, we have to make sure that we can efficiently store and retrieve: link values, all plaquettes containing a given link and all links contained in a plaquette. It is useful to think about the centres of the links and plaquettes. Link centres are given by $\{(i+\frac{1}{2}, j, k), (i, j+\frac{1}{2}, k), (i, j, k+\frac{1}{2})\}$ which are links along x, y and z axis and plaquette centres are given by $\{(i, j+\frac{1}{2}, k+\frac{1}{2}), (i+\frac{1}{2}, j, k+\frac{1}{2}), (i+\frac{1}{2}, j+\frac{1}{2}, k)\}$ which are plaquettes in YZ, ZX and XY plane. Here i, j, k are all between 0 and

N.

An efficient way of storing this is to include the third variable $d \in \{0, 1, 2\}$ indicating which index picks up $+\frac{1}{2}$ in case of links and which one does not in case of plaquettes. So a link with a centre at $(i + \frac{1}{2}, j, k)$ is stored in a 4D array at the location labelled by i, j, k and 0. Similarly, a plaquette $(i + \frac{1}{2}, j + \frac{1}{2}, k)$ is stored at another 4D array at the location i, j, k and 2. Graphically, we can think of this as assigning 3 links and 3 plaquettes to each site (i, j, k) (see figure A.2). This way of storing links and plaquettes ensures that we access them in constant time.

To find the plaquettes containing a certain link, we note that we can get their centres by keeping $+\frac{1}{2}$ in the coordinate of the link centre and by adding $\pm\frac{1}{2}$ to the other 2 coordinates. This generates a total of 4 plaquettes. For example, a link at $(i + \frac{1}{2}, j, k)$ is contained within plaquettes at $(i + \frac{1}{2}, j \pm \frac{1}{2}, k)$ and $(i + \frac{1}{2}, j, k \pm \frac{1}{2})$.

To find the links within a plaquette, we get their centres by adding $\pm\frac{1}{2}$ to the coordinates that already have a factor of $\frac{1}{2}$ in them. This generates a total of 4 links. For example, for a plaquette in XY plane $(x + \frac{1}{2}, j + \frac{1}{2}, k)$ consists of the links at $(x + \frac{1}{2}, j, k), (x + \frac{1}{2}, j + 1, k), (x, j + \frac{1}{2}, k)$ and $(x + 1, j + \frac{1}{2}, k)$.

Note that when computing the values of plaquettes, we need to keep track whether we use U or U^{-1} for a specific link. If we traverse the link so that a coordinate is increasing we use U, otherwise we use U^{-1}. If we label links of a plaquette as U_1, U_2, U_3 and U_4, the plaquette value is given by $U_1 U_2 U_3^{-1} U_4^{-1}$ or a cyclic permutation of indices depending on how we labeled the links.

Finally, to impose periodic boundary conditions after any addition of $\frac{1}{2}$ we mod out the result by the lattice size.

A.1.1 $C++$ Headers and Code Logic

The code is structured in 5 classes: Group, Vec, Link, Plqt and Lattice.

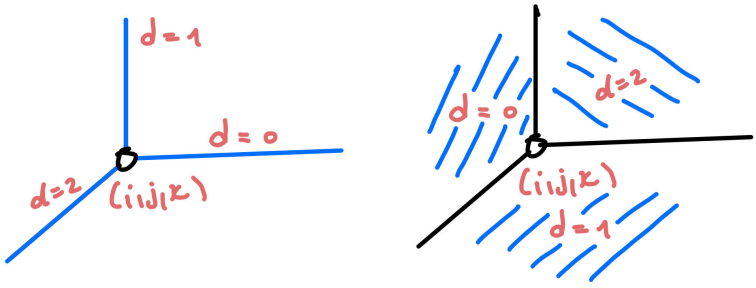

Figure A.2: Assignment of links and plaquettes to a vertex using a variable d.

A.1.2 Group.h

This class provides basic functionality such as compute product of two group elements, inverse, conjugacy classes etc. It also provides a function that samples a random group element given a set of probabilities which is an equivalent of NumPy's choice function.

```
1   #pragma once
2   #include <vector>
3
4   class Group
5   {
6   private:
7       int N;
8       std::vector<std::vector<int>> dot_table;
9       std::vector<int> G;
10      std::vector<int> inverses;
11      std::vector<std::vector<int>> conj_classes;
12      std::vector<int> conj_class_of;
13      std::vector<int> x_elements;
14  public:
15      Group();
16      Group(const std::vector<std::vector<int>>& dot_table);
17      void find_inverses();
18      void find_conj_classes();
19      std::vector<int> get_conj_class(int conj_cls) { return this->conj_classes[conj_cls]; }
20      int size() { return this->N; };
21      int dot(int g, int h);
22      int inv(int g);
23      int x(int g);
24      int conj_class(int g) { return conj_class_of[g]; };
25      int n_conj_classes() { return conj_classes.size(); };
26      int get_random_val();
27      int get_random_val(const std::vector<float>& probs);
28  };
29
```

Figure A.3: Group.h header file.

A.1.3 Vec.h

This class enables us to easily move around the lattice. It is essentially a $3D$ vector living on a lattice. So, any additions or multiplications automatically mod out the

coordinates by the size of the lattice to account for the periodic boundary conditions.

```cpp
#pragma once
#include <iostream>

class Vec
{
private:
    // Vector has x, y and z.
    // Vector lives in a periodic lattice of sizes Nx, Ny, Nz
    float x;
    float y;
    float z;
    int Nx;
    int Ny;
    int Nz;
public:
    // Constructor
    Vec();
    Vec(float x, float y, float z, int N);
    Vec(float x, float y, float z, int Nx, int Ny, int Nz);

    // Getters
    int get_x() { return this->x; }
    int get_y() { return this->y; }
    int get_z() { return this->z; }

    // Setters
    void set_x(float x) { this->x = x; }
    void set_y(float y) { this->y = y; }
    void set_z(float z) { this->z = z; }


    // Mod out
    void mod_out();

    // Operators
    Vec operator+(const Vec& v);

    Vec operator-(const Vec& v);

    Vec operator*(const float& a);

    bool operator==(const Vec& v);

    bool operator!=(const Vec& v);

    bool operator<(const Vec& v);

    bool operator>(const Vec& v);

    bool operator<=(const Vec& v);

    bool operator>=(const Vec& v);

    friend std::ostream& operator<< (std::ostream& o, const Vec & fred);

    // Hash Function
    float Hash();
};
```

Figure A.4: Vec.h header file.

A.1.4 Link.h and Plqt.h

Link and Plqt classes are very similar. The only difference is that Link has a function get_plqts() for getting plaquettes containing the link and Plqt has a function get_links(). Additionally, link has it's value stored, whereas Plqt does not because it depends on the starting vertex and direction of traversal. We want to emphasize that we generate and store in the memory return arguments for get_links() and get_plqts(). Computing these every time the function are called is very expensive.

```cpp
#pragma once
#include "Vec.h"
#include <vector>

class Link
{
private:
    Vec r;
    int i, j, k, d;
    int g;
    std::vector<std::vector<int>> plqts;
public:
    Link(int g, int i, int j, int k, int d, int Nx, int Ny, int Nz);
    Link(int g, int i, int j, int k, int d, int N) : Link(g, i, j, k, d, N, N, N) {};
    void set_value(int g) { this->g = g; };
    int get_value() { return this->g; };
    Vec get_coordinates();
    std::vector<std::vector<int>> get_plqts();
};
```

```cpp
#pragma once
#include <vector>
#include "Vec.h"

class Plqt
{
private:
    Vec r;
    int i, j, k, d;
    std::vector<std::vector<int>> links;
public:
    Plqt(int i, int j, int k, int d, int Nx, int Ny, int Nz);
    Plqt(int i, int j, int k, int d, int N) : Plqt(i, j, k, d, N, N, N) {};
    Vec get_coordinates();
    std::vector<std::vector<int>> get_links();
};
```

Figure A.5: Link.h (above) and Plqt.h (below) header files.

A.1.5 Lattice.h

Lattice is the largest and the most important class. It stores all links and plaquettes and provides all functions related to Monte Carlo simulation.

```cpp
#pragma once
#include <vector>
#include "Group.h"
#include "Vec.h"
#include "Link.h"
#include "Plqt.h"

class Lattice
{
private:
    int Nx, Ny, Nz;

    std::vector<float> couplings;
    Group G;
    // links are indixed as (i, j, k, d) with centres are (i + 1/2*(d==0), j + 1/2*(d==0), k + 1/2*(d==2))
    std::vector < std::vector< std::vector< std::vector<Link>>>> links;
    // links are indixed as (i, j, k, d) with centres are (i + 1/2*(d!=0), j + 1/2*(d!=0), k + 1/2*(d!=2))
    std::vector < std::vector< std::vector< std::vector<Plqt>>>> plqts;

public:
    // Constructors
    Lattice(Group G, int Nx, int Ny, int Nz);
    Lattice(Group G, int N) : Lattice(G, N, N, N) {};

    // Setting parameters and Values
    void initiate_plqts();
    void initiate_links();
    void randomize_links();
    void set_links(const std::vector<std::vector<int>>& links, int g);
    void set_couplings(const std::vector<float>& couplings) { this->couplings = couplings; };

    // Action and Value Computation
    float element_action(int element);
    int find_plqt_value(const std::vector<int>& plqt);
    float compute_action(const std::vector<std::vector<int>>& plqts);
    float compute_total_action();
    float compute_avg_action();
    std::vector<float> probe_phase_space_point();

    // Monte Carlo
    void MCHB_sweep();

    // String Operators
    float full_open_string(Vec start, int dir, int len, int conj_class);
    float open_string(Vec start, int dir, int len, int h);
};
```

Figure A.6: Lattice.h header file.

The most interesting functions are: MCHB_sweep() which performs one sweep of Heat Bath algorithm, probe_phase_space_point() which returns a vector of values for $\langle \delta_X \rangle$ and $\langle F \rangle$, full_open_string() that calculates the value of a magnetic open string for a given conjugacy class at the current configuration and open_string() that calculates the value of a magnetic open string for a given group element at the current configuration.

A.2 Important Functions

Here we give full declarations of non-trivial functions that we implemented.

```
91  int Group::get_random_val(const std::vector<float>& probs)
92  {
93      // Essentially numpy.choice with some floating point arithmetic error fixings
94      double r = static_cast<double>(rand()) / RAND_MAX;
95
96      int g = -1;
97
98      while (r >= 0)
99      {
100         g++;
101         if (g == this->N)
102             return g - 1;
103         r -= probs[g];
104     }
105     if (g == this->N)
106         g--;
107
108     return g;
109 }
```

Figure A.7: Sample group element according to a given probability distribution.

```
133 std::vector<float> Lattice::probe_phase_space_point()
134 {
135     // For a given configuration it computes the average concentration of all fluxes and the free energy
136     float F = 0;
137     std::vector<float> delta_C = std::vector<float>(this->G.n_conj_classes(), 0);
138     for (int i = 0; i < this->Nx; i++)
139         for (int j = 0; j < this->Ny; j++)
140             for (int k = 0; k < this->Nz; k++)
141                 for (int d = 0; d < 3; d++)
142                 {
143                     int g = this->find_plqt_value({ i, j, k, d });
144                     delta_C[this->G.conj_class(g)]++;
145                     F += this->element_action(g);
146                 }
147
148     F = F / this->Nx / this->Ny / this->Nz / 3;
149     for (int i = 0; i < delta_C.size(); i++)
150         delta_C[i] = delta_C[i] / this->Nx / this->Ny / this->Nz / 3;
151
152     delta_C.push_back(F);
153
154     return delta_C;
155 }
```

Figure A.8: Probe phase space point function.

```cpp
void Lattice::MCHB_sweep()
{
    // Performs one sweep of MCHB
    std::vector<float> actions = std::vector<float>(this->G.size(), 0);
    for (int i = 0; i < this->Nx; i++)
        for (int j = 0; j < this->Ny; j++)
            for (int k = 0; k < this->Nz; k++)
                for (int d = 0; d < 3; d++)
                {
                    Link* link = &this->links[i][j][k][d];

                    for (int g = 0; g < this->G.size(); g++)
                    {
                        link->set_value(g);
                        actions[g] = this->compute_action(link->get_plqts());
                    }

                    // actions = np.exp(-actions); actions = actions / np.sum(actions)
                    // Cheers C++ !
                    float sum = 0;
                    for (int i = 0; i < actions.size();i++)
                    {
                        actions[i] = std::exp(-actions[i]);
                        sum += actions[i];
                    }
                    for (int i = 0; i < actions.size(); i++)
                        actions[i] /= sum;

                    int g = this->G.get_random_val(actions);
                    link->set_value(g);
                }
}
```

Figure A.9: Monte Carlo Heat Bath Sweep function.

```
199  float Lattice::open_string(Vec start, int dir, int len, int h)
200  {
201      int dir2, dir3;
202      dir2 = (dir + 1) % 3; dir3 = (dir + 2) % 3;
203      Vec unit1, unit2, unit3;
204      if (dir == 0)
205      {
206          unit1 = Vec(1, 0, 0, this->Nx, this->Ny, this->Nz);
207          unit2 = Vec(0, 1, 0, this->Nx, this->Ny, this->Nz);
208          unit3 = Vec(0, 0, 1, this->Nx, this->Ny, this->Nz);
209      }
210      else if (dir == 1)
211      {
212          unit1 = Vec(0, 1, 0, this->Nx, this->Ny, this->Nz);
213          unit2 = Vec(0, 0, 1, this->Nx, this->Ny, this->Nz);
214          unit3 = Vec(1, 0, 0, this->Nx, this->Ny, this->Nz);
215      }
216      else
217      {
218          unit1 = Vec(0, 0, 1, this->Nx, this->Ny, this->Nz);
219          unit2 = Vec(1, 0, 0, this->Nx, this->Ny, this->Nz);
220          unit3 = Vec(0, 1, 0, this->Nx, this->Ny, this->Nz);
221      }
222
223      int connection = 0;
224      int full_connection;
225      int i, j, k;
226      int link;
227      int g1, g2, g3, g4;
228      int U_p;
229      float result = 0;
230      for (int cnt = 0; cnt < len + 1; cnt++)
231      {
232          // Compute the plaquette value
233          // go in +dir3 (unit3), -dir2, -dir3, +dir2
234          i = start.get_x(); j = start.get_y(); k = start.get_z();
235          g1 = this->links[i][j][k][dir3].get_value();
```

Figure A.10: Open String function part 1.

```
236
237             start = start + unit3;
238             start = start - unit2;
239             i = start.get_x(); j = start.get_y(); k = start.get_z();
240
241             g2 = this->links[i][j][k][dir2].get_value();
242
243             start = start - unit3;
244             i = start.get_x(); j = start.get_y(); k = start.get_z();
245
246             g3 = this->links[i][j][k][dir3].get_value();
247             g4 = this->links[i][j][k][dir2].get_value();
248
249             // Compute the operator value
250             U_p = this->G.dot(g1, this->G.dot(this->G.inv(g2), this->G.dot(this->G.inv(g3), g4)));
251             full_connection = this->G.dot(this->G.inv(connection), this->G.dot(this->G.inv(h), connection));
252             // full_connection = this->G.inv(h);
253             result += this->element_action(U_p) - this->element_action(this->G.dot(full_connection, U_p));
254
255             // Complete the full square back
256             start = start + unit2;
257             i = start.get_x(); j = start.get_y(); k = start.get_z();
258
259             // Update connection element U_{i_0, i_p}
260             link = this->links[i][j][k][dir].get_value();
261             connection = this->G.dot(connection, link);
262
263             // Move one step
264             start = start + unit1;
265             i = start.get_x(); j = start.get_y(); k = start.get_z();
266         }
267
268         return std::exp(result);
269
270     }
```

Figure A.11: Open String function part 2.

```cpp
    vector<vector<int>> S3_dot = { {0, 1, 2, 3, 4, 5},
                                   {1, 0, 4, 5, 2, 3},
                                   {2, 5, 0, 4, 3, 1},
                                   {3, 4, 5, 0, 1, 2},
                                   {4, 3, 1, 2, 5, 0},
                                   {5, 2, 3, 1, 0, 4} };
    Group S3 = Group(S3_dot);
    int N = 5;
    int iters = 1000;
    int relax_iters = 300;
    Lattice l = Lattice(S3, N);
    l.set_couplings({ 3.0, 0, 0 });
    l.randomize_links();

    clock_t start = clock();
    int print_perc = 5;
    int print_period = print_perc / 100 * iters;
    // Relaxation Period
    for (int i = 0; i < relax_iters; i++)
        l.MCHB_sweep();

    cout << "Relaxation done in: " << static_cast<float>(clock() - start) / CLOCKS_PER_SEC << "s\n";

    // Monte Carlo Iterations
    for (int i = 0; i < iters; i++)
    {
        // Calculate Operator Values
        //
        //          HERE
        //

        l.MCHB_sweep();

        // Print Progress
        if ((i + 1) % print_period == 0)
        {
            cout << "Finished " << (i + 1) / print_period << "% in: ";
            cout << static_cast<float>(clock() - start) / CLOCKS_PER_SEC << "s\n";
        }
    }
}
```

Figure A.12: Template for a full Monte Carlo calculation.

```python
import numpy as np
from scipy.stats import t

def get_confidence_interval(arr, alpha=0.05):
    b = len(arr)

    lo, hi = t.interval(1 - alpha, b - 1)
    mean = np.mean(arr)
    var = np.var(arr, ddof=1)

    return mean, (mean + lo * np.sqrt(var / b), mean + hi * np.sqrt(var / b))
```

Figure A.13: Confidence interval function in Python.

Bibliography

[1] Y. Aharonov and D. Bohm. Significance of electromagnetic potentials in the quantum theory. *Phys. Rev.*, 115:485–491, Aug 1959.

[2] F. A. Bais and J. C. Romers. Anyonic order parameters for discrete gauge theories on the lattice. *Annals Phys.*, 324:1168–1175, 2009.

[3] F. A. Bais and J. C. Romers. The modular S-matrix as order parameter for topological phase transitions. *New J. Phys.*, 14:035024, 2012.

[4] Kurt Binder and Dieter W. Heermann. *Monte Carlo Simulation in Statistical Physics*. Springer Berlin Heidelberg, 2010.

[5] Mark de Wild Propitius and F. Alexander Bais. Discrete gauge theories. In *CRM-CAP Summer School on Particles and Fields '94*, 11 1995.

[6] S. Elitzur. Impossibility of spontaneously breaking local symmetries. *Physical Review D*, 12(12):3978–3982, December 1975.

[7] R. P. Feynman, A. R. Hibbs, and George H. Weiss. Quantum mechanics and path integrals. *Physics Today*, 19(6):89–89, June 1966.

[8] James M. Flegal, Murali Haran, and Galin L. Jones. Markov chain monte carlo: Can we trust the third significant figure? *Statistical Science*, 23(2), 2008.

[9] James M. Flegal and Galin L. Jones. Batch means and spectral variance estimators in markov chain monte carlo. *The Annals of Statistics*, 38(2), April 2010.

[10] P. Hasenfratz. Lattice QCD. In *Recent Developments in High-Energy Physics*, pages 283–356. Springer Vienna, 1983.

[11] Christian Kassel. Drinfeld's quantum double. In *Graduate Texts in Mathematics*, pages 199–238. Springer New York, 1995.

[12] Louis H. Kauffman. *Knots and physics*. World Scientific, 2013.

[13] Rodolfo Martini and Eduardus M. de Jager, editors. *Geometric Techniques in Gauge Theories*. Springer Berlin Heidelberg, 1982.

[14] Titus Neupert, Huan He, Curt von Keyserlingk, Germán Sierra, and B. Andrei Bernevig. Boson condensation in topologically ordered quantum liquids. *Physical Review B*, 93(11), March 2016.

[15] J. R. Norris. *Markov chains*. Cambridge University Press, Cambridge, UK New York, 1998.

[16] Emil Mihaylov. Prodanov. *Aspects of Chern-Simons theory*. Trinity College, 2000.

[17] Jonathan Sapirstein. Quantum electrodynamics. In *Springer Handbook of Atomic, Molecular, and Optical Physics*, pages 413–428. Springer New York, 2006.

[18] Steven H. Simon. *Topological Quantum: Lecture Notes and Proto-Book*. 2021.

[19] David Tong. *Gauge Theory*. 2018.

[20] Kenneth G. Wilson. Confinement of quarks. *Phys. Rev. D*, 10:2445–2459, Oct 1974.

[21] Kenneth G. Wilson. Monte-carlo calculations for the lattice gauge theory. In *Recent Developments in Gauge Theories*, pages 363–402. Springer US, 1980.

[22] Edward Witten. Quantum Field Theory and the Jones Polynomial. *Commun. Math. Phys.*, 121:351–399, 1989.