Radboud University Nijmegen

Faculty of Science

Monte Carlo simulations of dually-weighted maps

Thesis BSc Physics and Astronomy

Author:
Jasper Stokmans

Supervisor:
dr. Timothy Budd

Second reader:
prof. dr. Wim Beenakker

July 13, 2023

Abstract

To this day, there exists no theory of gravity at the Planck scale as attempts to formulate such a theory have encountered considerable difficulties. One approach is Lattice Quantum Gravity, which needs a second or higher-order phase transition to advance its development. Using a novel approach to simulating a single slab of 2+1 dimensional Causal Dynamical Triangulations (CDT), this thesis attempts to determine a more effective method of simulating phase transitions of higher order in CDT. The numerical results indicate that the model used does not possess the phase transition observed in fully-fledged CDT simulations of the same geometry.

Contents

1 Introduction

In modern theoretical physics, one can roughly distinguish two main theories that each predict certain phenomena with unparalleled precision. On the one hand, there is Einstein's theory of General Relativity, which describes gravity for large objects like celestial bodies. On the other hand, there is Quantum Field Theory, which describes the interactions taking place on the subatomic scale. Both theories have been proven to make accurate predictions about their respective domains. However, both theories cannot tell us anything about gravity at the smallest length scales [1]. Therefore, theoretical physicists have the desire to find a theory that can describe gravity even at these extremely small scales.

Theorists have been attempting to find this theory for decades, exploring different avenues of approach. One of these avenues has led to the development of Lattice Quantum Gravity, which attempts to find a quantised theory of gravity by "glueing" geometrically flat objects together in such a way that the resulting geometry corresponds to the classical spacetime geometry at macroscopic length scales but still shows quantum mechanical behaviour at microscopic length scales.

For the theory produced by Lattice Quantum Gravity to be valid for small length scales, one must take the limit of the geometry where the size of its building blocks goes to zero [2]. In taking this limit, the physics described by the model must remain unchanged. This imposes a scale invariance condition on the theory of Lattice Quantum Gravity, meaning that the geometry constructed in this way will have the same overall structure regardless of the scale at which it is observed. To satisfy this scale-invariance condition, one must take the limit of lattice spacings to zero near a so-called UV-fixed point in the parameter space [2].

Finding such a UV-fixed point is therefore key to developing the theory of Lattice Quantum Gravity further. Searches for such a UV-fixed point focus on second or higher-order phase transitions in the model for quantum gravity. It is known that at such phase transitions the correlation length in terms of the lattice spacing of physical observables diverges. This diverging correlation length means that one can take the limit of the lattice spacing to zero without losing possible correlations between two points on the lattice, which means the model is scale-invariant at such a phase transition.

However, determining such a second-order phase transition has proven to be more easily said than done. Most phase transitions either turn out to be of first order or resist attempts to determine their order. The latter phase transitions are of course the more interesting and will be the subject of this thesis. More specifically, this thesis will focus on a phase transition in a specific model for quantum gravity in 2 spatial dimensions and one time-like dimension. The model considered here will be 2+1 dimensional Causal Dynamical Triangulations (CDT), which will be introduced in section 2.2.

In 2+1 dimensional CDT, there exists a phase transition that is most likely of first order [3]. However, the problem with this phase transition is that the current simulation techniques suffer from a phenomenon called critical slowing down when performing simulations near the phase transition, making measurements near the phase transition difficult. A Dually-Weighted model could provide an alternative way to perform these simulations that does not suffer from critical slowing down.

While the phase transition studied in this thesis is most likely of first order, the possibility exists that in variants of 2+1 dimensional CDT there exist phase transitions that are of higher order. These models may also lend themselves to being simulated using Dually-Weighted models, since critical slowing down does not occur exclusively in 2+1 dimensional CDT. The relative simplicity of the model studied here does make it a good starting point.

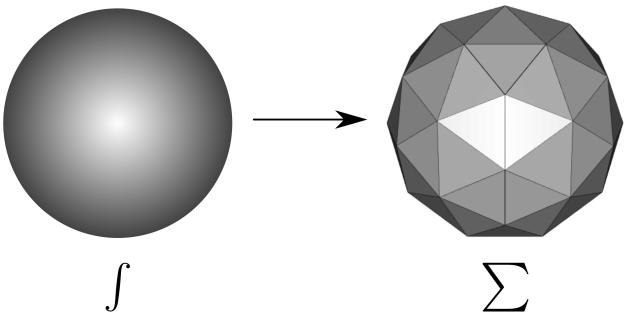

Figure 1: A schematic depiction of what is meant by "discretising a geometry". The sphere on the left can be approximated using flat triangles to produce the object on the right. Figure adapted from: [7]

2 Theory

2.1 Towards a quantum theory of gravity

As mentioned in the introduction, in theoretical physics one can roughly distinguish two major theories: Quantum Field Theory (QFT) and General Relativity (GR). QFT describes interactions taking place on a very small scale using a path integral formalism, where the quantum mechanical behaviour of a field is described by an integral over all possible trajectories from point A to point B [4]. These paths are all weighted using an appropriate action. This path integral gives the transition amplitude between states of a system.

GR describes gravity as the curvature of spacetime, the more spacetime is curved, the stronger the gravity. Each spacetime "shape" has an associated metric, which describes the properties of the spacetime. For example, flat spacetime is described by the Minkowski metric $\eta_{\mu\nu}$.

The mutual shortcoming of these theories is that they cannot describe gravity at the very smallest scale, the so-called Planck scale ($l_p = 1.6 \times 10^{-35}$ m) [1, 5]. Moreover, when attempting to find such a theory of quantum gravity using standard perturbative methods, one encounters divergences. Such divergences also appear in QFT, where they are dealt with via a process called renormalisation [4]. In the case of gravity, however, the divergences are non-renormalisable, which means other approaches are necessary.

There exist different approaches to the problem of finding a quantum theory of gravity [2]. In this thesis, we will focus on so-called Lattice Quantum Gravity. Lattice Quantum Gravity aims to describe gravity by using an approach very similar to the one used in QFT [4, 6]. It begins by considering gravity as described by a metric field and constructing a path integral over these metrics [1],

$$Z = \int \frac{\mathcal{D}g}{\text{Diff}} e^{\frac{i}{\hbar} S_{EH}}. \tag{2.1}$$

Since the path integral in Equation 2.1 is difficult to solve, one now attempts to transform it into a sum over discretised geometries. Discretisation in this sense means that a "shape", like the sphere in Figure 1, is approximated using flat pieces. One such method of discretisation for spacetime geometries is Causal Dynamical Triangulations (CDT) which will be discussed in the next chapter.

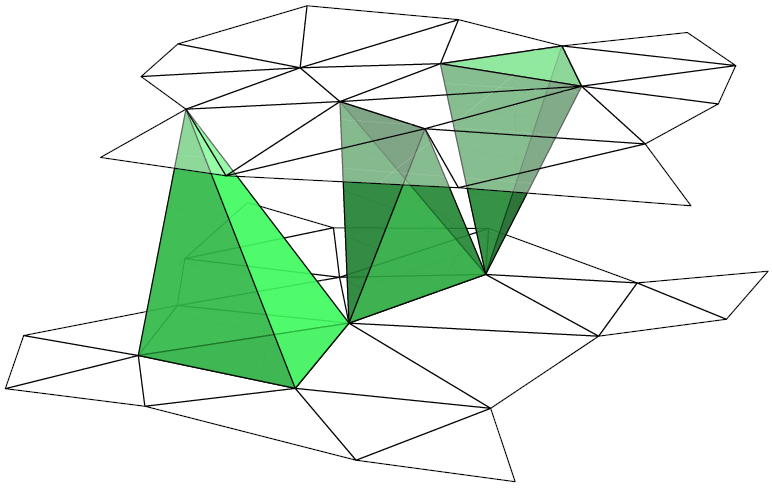

Figure 2: Three different simplices in 2+1 dimensional CDT (colored green). From left to right: a 31-simplex, a 22-simplex and a 13-simplex. Figure from [1]

2.2 2+1 Dimensional CDT

In this section, the basic idea of 2+1 dimensional CDT is explained to a degree that should be sufficient to understand the remainder of this thesis. This is by no means a formal definition of the model, but rather a conceptual introduction. A more formal definition is given by Ambjørn et. al. in [3].

The CDT model is a model that discretises spacetime while maintaining a preferred time direction. The CDT model can be viewed as consisting of layers of discretised two-dimensional geometries indexed by a time parameter $t \in \mathbb{N}$. These geometries can be constructed by attaching discrete building blocks, like triangles, to each other. If triangles or their higher dimensional equivalents, such as tetrahedra, are used, the constructed object is called a triangulation. The individual building blocks are geometrically flat and all have the same, fixed shape[1] but the way they are connected to each other gives the layer a curved geometry.

Now, these layers are in turn connected to each other by using three-dimensional building blocks called simplices. These building blocks come in three different flavours as shown in Figure 2. These simplices are geometrically flat, but the way they are connected again gives rise to curved spacetime.

The resulting model can be interpreted as a discrete version of spacetime where each layer corresponds to a time t [8]. It is a network of nodes referred to as vertices connected by links. These links can be spatial when they lie in a surface of constant time, or time-like when they connect two such surfaces. The continuum limit of the model is obtained by taking the number of simplices to infinity.

Mathematically, CDT turns the continuous path integral of Equation 2.1 into a discrete sum over CDT configurations [3]:

$$Z_{CDT} = \sum_{\mathcal{T}_T(S^1 \times S^2)} \frac{1}{C(T)} e^{-S_E(N_0, N_3, T)}, \tag{2.2}$$

[1] In the figures in this thesis, the building blocks do not have the same shape. This is done to make them easier to draw.

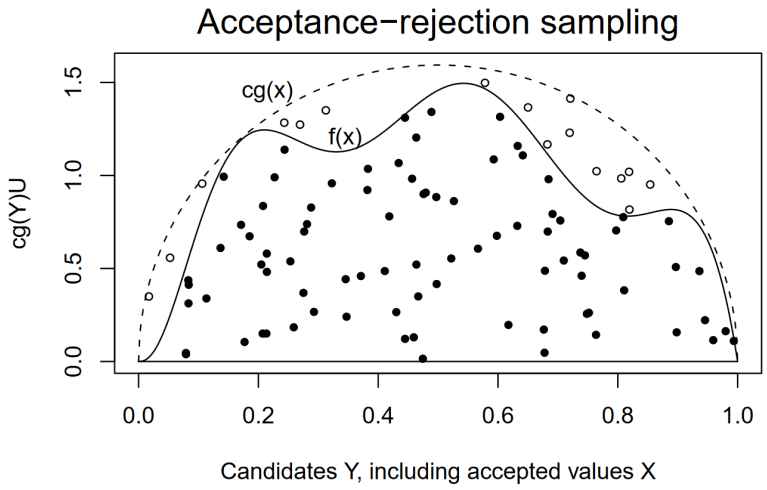

Figure 3: An illustration of acceptance-rejection sampling. The target distribution is given by $f(x)$ and is shown by a solid line. We sample from the distribution $cg(x)$ to find the horizontal coordinate and sample from a uniform distribution to find the vertical coordinate of a sample point. Such a point is accepted if it lies below $f(x)$ (solid dots) and rejected if it is not (circles). Source: [9].

where the sum runs over CDT configurations with geometry $S^1 \times S^2$. Each geometry is weighted with a Boltzmann factor depending on the Euclidean three-dimensional Regge action S_E. The action depends on three parameters: the number of vertices N_0, the number of simplices N_3 and the total number of layers $T+1$. The factor $1/C(T)$ prevents double counting of equivalent geometries. Since the equations involved in CDT models are often difficult to solve analytically, one has to resort to the use of computer simulations to determine the behaviour of such a model. Such simulations usually take the form of Markov-Chain Monte Carlo simulations, which will be discussed in the next section.

2.3 Markov-Chain Monte Carlo

This section serves as an introduction to Monte Carlo techniques, specifically Markov-Chain Monte Carlo as these will be used extensively.

Monte Carlo techniques are numerical methods that rely on random numbers to sample from a given distribution [9]. These methods have applications in many numerical simulations, such as simulations of the Ising model or computing integrals [10]. Perhaps the simplest example of a Monte Carlo simulation is to sample from some distribution $f(x)$ by using acceptance-rejection sampling. Acceptance-rejection sampling draws random points from a distribution $cg(x), c \in \mathbb{R}$ that satisfies $\forall_x(f(x) \leq g(x))$. This gives the x-coordinate of a sample, to obtain the y-coordinate one samples from a uniform distribution $cg(x)U$. Here, U is the uniform distribution on the interval $[0, 1]$. This procedure is illustrated in Figure 3.

But what if direct sampling methods like the one above do not perform sufficiently fast or accurately?

In such scenarios, an alternative method can be found in Markov-Chain Monte Carlo (MCMC). These methods do not attempt to sample the distribution directly, but rather they attempt to approach the desired distribution over a large number of steps referred to as Markov-Chain

steps [9]. A Markov-Chain algorithm is concerned with a system that can be in a state X_i. The collection of all the possible states a system can be in is called the state space Γ. An example of a state space is the space $\Gamma = \{1, -1\}^N$, corresponding to an Ising model with N spins. The Hamiltonian of this model is given by [10]

$$H(s) = \sum_{i \sim j} s_i s_j, \tag{2.3}$$

where the sum runs over all neighbouring pairs of spins s_i and s_j.

MCMC is now concerned with sampling states $X_i \in \Gamma$ according to a specific probability distribution, usually determined by the system. In the case of the Ising model on N spins, this probability distribution would be given by

$$p(s) = \frac{1}{Z} e^{-\beta H(s)}, \tag{2.4}$$

where $s \in \Gamma = \{1, -1\}^N$ is a specific state of the system, Z is the normalisation factor (in this case: the partition sum), $\beta = 1/k_B T$ and $H(s)$ is the Hamiltonian of the system. In this case, the goal of MCMC would be to move from a state s to a state s' with a certain probability such that after enough steps, the probability of finding the system in a state s is given by Equation 2.4. If one now denotes the states X_i as row vectors, the probability to move from state X_i to state X_{i+1} can be encoded in a transition matrix P [10]. This transition matrix is constructed such that $p(s) = \pi(s)$, where π is a left-eigenvector of P with eigenvalue one, i.e.: $\pi P = \pi$. For the example of the Ising model, this so-called stationary distribution is given by

$$\pi(s) = \frac{1}{Z} e^{-\beta H(s)}, \tag{2.5}$$

which is simply the probability of finding the system in the state s.

The idea of MCMC is to start from a manually chosen initial state X_0 and apply the transition matrix P until the distribution approximates the stationary distribution with sufficient accuracy. The problem lies in determining a suitable transition matrix, it must produce the stationary distribution when applied often enough to the initial state X_0. We note that the condition $\pi = \pi P$ implies that for all $y \in \Gamma$ [10]

$$\sum_{x \in \Gamma} \pi(x) P(x \to y) = \pi(y) = \pi(y) \sum_{x \in \Gamma} P(y \to x) = \sum_{x \in \Gamma} \pi(y) P(x \to y). \tag{2.6}$$

Since the total probability to move from a state x to any state must be 1. The simplest way to satisfy this condition is called detailed balance and is expressed as follows:

$$\forall_{x,y \in \Gamma} \ (\pi(x) P(x \to y) = \pi(y) P(y \to x)). \tag{2.7}$$

Equation 2.7 states that the flow of probability from any state x into a state y must be the same as the flow of probability from state y into state x.

2.4 Metropolis-Hastings algorithm

One method to produce a suitable transition matrix is the so-called Metropolis-Hastings (MH) algorithm [9]. The MH algorithm uses a manually constructed proposal probability matrix $Q(x \to y)$, which serves to select a possible move from a state x to a state y. It then determines whether to implement this move (accept it) or whether to reject it, which can be captured

in the acceptance probability matrix $A(x \to y)$. The matrix $A(x \to y)$ is chosen such that $P(x \to y) = Q(x \to y)A(x \to y)$ is a suitable transition matrix for MCMC.

It is now possible to derive an expression for the acceptance probability $A(x \to y)$ given a proposal probability $Q(x \to y)$ and a stationary distribution π. From the detailed balance condition of Equation 2.7, it can be derived that the acceptance probability for the transition from a state x to a state y is given by [9]

$$A(x \to y) = \min\left\{1, \frac{\pi(y)P(y \to x)}{\pi(x)P(x \to y)}\right\}. \tag{2.8}$$

One will likely notice that the acceptance probability of Equation 2.8 does not depend on the normalisation of π. Since this normalisation factor is generally unknown, this fact makes the MH algorithm a very powerful tool.

2.5 Measuring observables

If one wishes to perform simulations of lattice quantum gravity, this will be with the intent of doing measurements. Therefore, it is vital that one can confidently do these measurements and compute their errors. In the case of Monte Carlo methods, this means taking into account not only the errors arising from the finite sample size but also the fact that Monte Carlo simulations draw samples from an approximate distribution.

Before one can even think about doing useful measurements, one must be confident that the simulation has approached the desired probability distribution sufficiently closely. This confidence is usually achieved by letting the simulation perform a specified number of moves before perfroming the actual measurements [9]. It can be useful to express the number of these moves in terms of sweeps, where one sweep is usually defined to be the number of moves required to perform on average one move per lattice point. In the example of the Ising model, one sweep would correspond to the total number N of spins present in the model. The sweeps performed to let the simulation approach the desired distribution are called burn-in sweeps and there is no real consensus as to how many burn-in sweeps should be used [9]. One way to estimate the number of burn-in sweeps required is to plot several observables using several different initial configurations and estimating when the measurements of the different simulations agree [10].

Once the simulation has reached a point where one is confident that the distribution being sampled from is approximately the desired distribution, one can perform the actual measurements. These measurements usually involve taking a sum or product. For example: in the Ising model, one might be interested in the absolute total magnetisation of the system

$$|M(s)| = \left|\sum_{i=1}^{N} s_i\right|. \tag{2.9}$$

Here, s is the state of the system and s_i is the value of spin i.

When using MCMC, it is incorrect to simply compute the mean of an observable and its error in the usual way [9]. That is, treating each measurement of the observable as independent from the others. This would neglect the fact that each configuration of the system is generated from the previous one via a (usually small) move, and is therefore very much the same as the previous configuration. Using the Ising model as an example: the observable $|M(s)|$ will not change much after performing one move (i.e. flipping one spin).

To resolve this issue, it is useful to quantify the similarity between two states in the Markov Chain and use it to derive a new error formula that can be applied in the case of MCMC. This

process starts with defining the autocorrelation ρ of an observable $f : \Gamma \to \mathbb{R}$ as [10]

$$\rho(t) = \mathrm{Corr}(f(X_i), f(X_{i+t})) := \frac{\mathbb{E}\left[(f(X_i) - \mathbb{E}[f(X)])(f(X_{i+t}) - \mathbb{E}[f(X)])\right]}{\mathrm{Var}(f(X))}, \tag{2.10}$$

where $\mathbb{E}[f(X)]$ and $\mathrm{Var}(f(X))$ denote the analytical mean and variance of the observable, respectively and the parameter t is often referred to as the simulation time. The autocorrelation is a measure of how correlated, or dependent, the states at step i and step $i+t$ of the Markov-Chain are. However, Equation 2.10 is an exact formula that cannot be applied directly to MCMC simulations. The estimate of $\rho(t)$ obtained from an MCMC simulation is referred to as the sample autocorrelation, $\overline{\rho}(t)$ and can be estimated using the sample autocovariance [10]:

$$\overline{\gamma}(t) = \frac{1}{n-t}\sum_{i=1}^{n-t}(f(X_i) - \overline{f(X)}_n)(f(X_{i+t}) - \overline{f(X)}_n), \tag{2.11}$$

where $\overline{f(X)}_n = \frac{1}{n}\sum_{i=1}^{n} f(X_i)$ is the sample mean. The sample autocorrelation is now given by the sample autocovariance normalised by $\overline{\gamma}(0)$:

$$\overline{\rho}(t) = \frac{\overline{\gamma}(t)}{\overline{\gamma}(0)}, \tag{2.12}$$

since $\overline{\gamma}(0)$ is the variance $\mathrm{Var}(f(X))$ computed over the first n steps of the Markov Chain.

The obtained expression for $\overline{\rho}(t)$ is still not quite suitable to compute the errors in an MCMC simulation since it shows large statistical errors when t is of the same order as n [10]. To circumvent this, one might approximate $\overline{\rho}(t)$ by an exponential decay: $\overline{\rho}(t) \approx e^{-t/\tau_f}$. The autocorrelation time τ_f can be determined either by doing an exponential fit to $\overline{\rho}(t)$ or by taking the first Markov-Chain step where $\overline{\rho}(t) \leq 1/e$. The outcome of a measurement of the observable $f(X)$ can now be expressed as [10]

$$\mathbb{E}[f(X)] = \mathbb{E}[\overline{f(X)}_n] \pm \sqrt{\frac{2\tau_f}{n}\overline{\gamma}(0)}. \tag{2.13}$$

While one is now equipped with a good way of computing the mean of measurements with appropriate error, it might be worthwhile to consider how performing measurements can be optimised. Looking back at Equation 2.9, it can be expected that computing the (absolute) magnetisation will become computationally expensive for large systems. That is, it will take a long time to compute the magnetisation for such a system. A solution would be to keep track of the magnetisation at every Markov-Chain step, which only requires a small update to the magnetisation. However, this approach will result in a total of N updates per sweep since a sweep consists of N Markov-Chain steps. This may not seem like a problem, but each of these measurements only contains a small amount of new information since each configuration is almost identical to the previous one. It is therefore not a very efficient use of computational power to perform a measurement at every Markov-Chain step.

An alternative solution is to only perform measurements when the configuration of the system is appreciably different from the configuration at the last measurement, i.e.: the values of the observable are sufficiently independent. Since the autocorrelation time τ_f is a measure of the simulation time needed between measurements for the observable values to become independent, it makes sense to perform a measurement at least every τ_f sweeps [10]. Waiting more than τ_f sweeps between measurements would be a waste of potentially useful data. However, we have already established that measuring too often is no good either. So one has to estimate a minimum simulation time between measurements.

A rule of thumb is to not spend more than roughly half the time doing measurements and the rest performing Markov-Chain steps [10]. For the Ising model, Equation 2.9 shows that a full computation of the magnetisation in this model requires $N-1$ additions. Assuming that this requires roughly the same time as $N-1$ Markov-Chain steps (roughly one sweep), a good simulation time between measurements would therefore be at least one sweep [10].

2.6 The problem when simulating the phase transition in 2+1 dimensional CDT

It is now possible to give a more precise statement of the problem encountered when attempting to simulate the phase transition in 2+1 dimensional CDT. First of all, an intuitive understanding of the different phases is required.

As discussed in Chapter 2.2, 2+1 dimensional CDT consists of three different types of simplices: 13-, 31- and 22-simplices, which are shown in the left of Figure 4. Most of these simplices can connect to each other, but 13-simplices cannot connect to 31-simplices. This means that 13- and 31-simplices can form clusters consisting of only the same kind of simplices and that the only way (a cluster of) 13-simplices can connect to (a cluster of) 31-simplices is via 22-simplices.

The phase transition is the transition between a phase with large clusters of simplices of the same kind to a phase where these clusters are relatively small. So on one side of the phase transition, there exists a phase called the de Sitter phase, where the number of 22-simplices is roughly equal to the numbers of 31- and 13-simplices [1]. This means that while there may exist clusters of only 31- or 13-simplices connected to each other, these clusters will generally be small as they will quickly be cut off by a 22-simplex. On the other side of the phase transition, however, the number of 22-simplices has fallen to be very small compared to the number of other simplices. This implies that, in this phase, there exist clusters of either 31- or 13-simplices that can contain numbers of simplices on the order of the total number of simplices of that kind. This phase will be referred to as the crumpled phase. It is the clusters in this crumpled phase that make simulating the phase transition through traditional means difficult.

To understand this difficulty, it is important to know how these simulations are performed in the first place. The simulations of 2+1 dimensional CDT are performed using MCMC methods, making small adjustments to a manually constructed triangulation. These adjustments are the Monte-Carlo moves for the simulation and to understand them, it is necessary to translate the complex problem of simulating CDT to a simpler problem of manipulating planar maps. Figure 4 from [1] shows one slab of 2+1 dimensional CDT, that is, two time slices at times t and $t+1$ connected by simplices.

One can now construct a graph containing the information about the glueing of the triangles in a time slice by letting each triangle correspond to a vertex in this new graph and connecting these vertices when the triangles in the triangulation are adjacent [3]. The graph obtained in this manner is shown in the middle of Figure 4. Since these graphs will be embedded in the underlying geometry of the spatial slices, they are called maps[2]. Since these maps contain no crossings of edges, they are planar maps. The vertices in these planar maps all have three edges emerging from them, so they are of degree three. Maps with the property that all vertices have the same degree n are called n-valent maps, our maps are therefore three-valent.

One can now superimpose these maps for each time slice in a nontrivial way to obtain a bicoloured map that fully specifies the slab [1], such a map is shown in Figure 4. In such a map, the vertices correspond to 31- or 13-simplices depending on their colour and a crossing of a blue and a red edge corresponds to a 22-simplex. It can now be seen that a cluster of

[2]Note that in a map, one can define things that cannot be defined for a graph, like a rotation around a vertex in (counter-) clockwise direction.

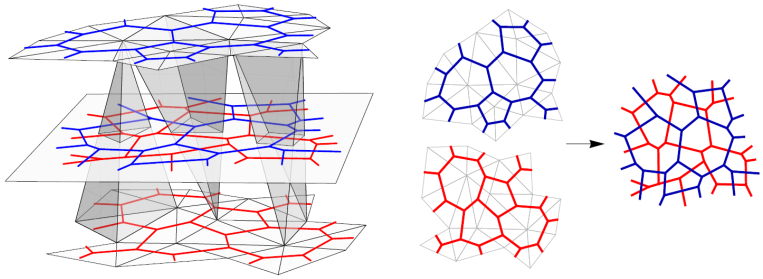

Figure 4: A single slab of 2+1 dimensional CDT with three different simplices. The planar maps corresponding to the time slices have been drawn in blue and red. Note that only three simplices are drawn, in reality, the entire space between the time slices is filled with simplices. Figure from [1]

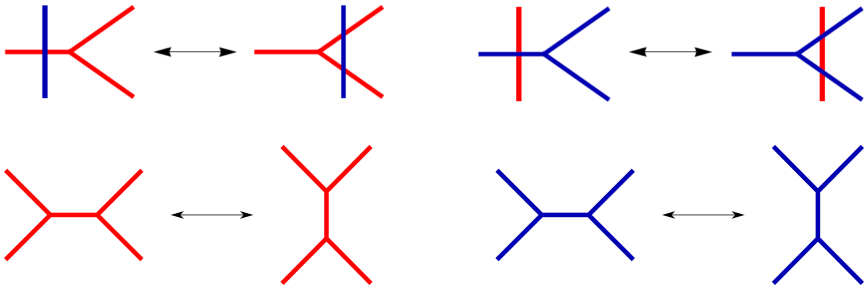

Figure 5: The MC moves relevant for simulating a single slab of 2+1 dimensional CDT. The moves consist of pulling a blue edge over a red vertex, pulling a red edge over a blue vertex and their inverse moves. In addition, since the spatial slices are not fixed, there are additional moves allowed. These moves are the "rotation" of an edge in either the red or blue map and their inverse. Figure adapted from: [1]

31-simplices corresponds to a cluster of red vertices in the bicoloured map surrounded by blue edges (22-simplices). Similarly, a cluster of 13-simplices corresponds to a cluster of blue vertices surrounded by red edges. Therefore, the crumpled phase will manifest itself in the bicoloured map as one or more macroscopic clusters of vertices of the same colour.

We can now understand the difficulty of simulating the phase transition when considering the Markov-Chain moves for such a bicoloured map. The Markov-Chain moves are given by moving a blue vertex over a red edge or moving a red vertex over a blue edge, "rotating" an edge and their inverse moves as shown in Figure 5. Since these moves are only capable of moving one vertex from one cluster to another cluster, breaking up large clusters of vertices will require many moves. Moreover, the moves cannot move the vertices very far, only across a single edge. This means that it requires many moves to move a single vertex over a large number of edges, which is required to distribute the vertices evenly over the clusters. Both these properties make going from the crumpled phase back to the de Sitter phase a process that requires many Markov-Chain steps and is therefore very time-consuming.

This phenomenon where a simulation slows down when approaching a certain point is referred to as critical slowing down. It would be preferred if the simulation was capable of moving an

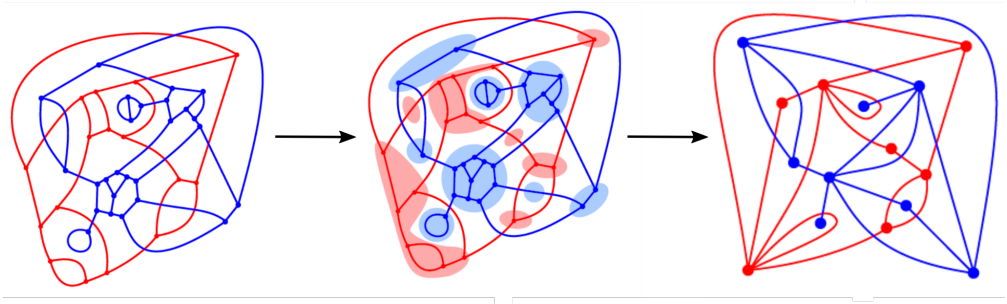

Figure 6: A representation of the process of contracting the clusters of vertices in the original bicoloured map. On the left is the original map. In the middle, the clusters have been marked with the appropriate colours. On the right, the marked clusters have been contracted into a single vertex. Source: T.G. Budd, unpublished.

arbitrary number of vertices of the same colour over an arbitrary number of edges as such an algorithm would not suffer from critical slowing down as much.

2.7 How dually-weighted maps help

Before we delve into the possible solution offered by dually-weighted map models, we will answer the question of what a dually-weighted map is. The dual map of a given map $\mathcal{M}$ is the map where each vertex corresponds to a face in $\mathcal{M}$ and the vertices are connected if the faces they correspond to are adjacent. A dually-weighted map is a planar map where one has assigned weights not only to the vertices of the map itself but also to the vertices of the dual map, hence dually-weighted [11]. Assigning weights to the vertices of the dual map is equivalent to assigning weights to the faces of the original map, $\mathcal{M}$. The degree of the vertices in the dual map is the number of faces adjacent to the face in $\mathcal{M}$ corresponding to this vertex. For example, if $\mathcal{M}$ has a face that is adjacent to four other faces, the degree of the corresponding vertex in the dual map will be four.

To see how such objects might help us to overcome the critical slowing down of simulations, one must first understand how they can be constructed from a CDT configuration. The goal is to produce a map where assigning weights to all vertices produces a dually-weighted map. The resulting map will have as an additional benefit that all faces are squares (they have four bounding edges). Such maps are called quadrangulations, in analogy to the triangulations encountered in a single time slice of CDT.

To produce such a map, consider the slab of CDT of Figure 4. As has been already noted, this slab can be represented by a bicoloured map. In this map, we had clusters of vertices surrounded by edges of the other colour, cutting them off from the other vertices. We can now contract the vertices within a cluster to form a single vertex of that colour such that we have precisely one blue vertex per red face and one red vertex per blue face. This contracting procedure is represented in Figure 6 for a random three-valent bicoloured map.

Since it will be important to be able to distinguish between vertices in the original bicoloured map and the vertices obtained through contracting these vertices, we will refer to the uncontracted vertices as cluster vertices and to the contracted vertices as quadrangulation vertices when necessary.

Now, we simply connect the quadrangulation vertices of a different colour to each other, provided we can do so without crossing a coloured edge. The resulting map is shown in Figure 7.

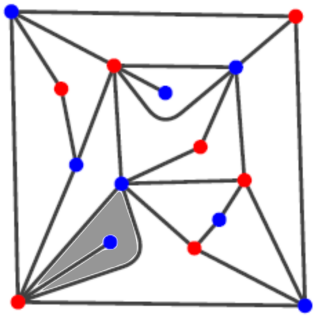

Figure 7: A quadrangulation resulting from connecting the quadrangulation vertices in the right map of Figure 6 without crossing any coloured edges. All faces have exactly four neighbours, hence it is a quadrangulation. Source: T.G. Budd, unpublished.

A map constructed in such a way has faces that are bounded by exactly four edges. This might seem counterintuïtive at first, certainly, the shaded region in Figure 7 is bounded by only two edges, right? It is not, as we consider the edge connecting the blue vertex of degree one as a bounding edge as well. This gives three bounding edges, the fourth edge comes from the fact that we count the number of bounding edges by traversing them in a closed loop (imagine walking along the boundary of a face). Using this definition, the edge connecting the blue quadrangulation vertex of degree one to the red quadrangulation vertex in the lower left will be counted twice, giving a total of four bounding edges.

For these quadrangulations to be equivalent to a single slab of 2+1 dimensional CDT, the partition functions for these two models must be equivalent. This can be achieved by assigning weights to each quadrangulation vertex. Since the blue quadrangulation vertices are the vertices of the dual map of the red map in the right of Figure 6, assigning weights to both colours of quadrangulation vertices corresponds to making the right map in Figure 6 dually-weighted. We have now constructed a dually-weighted map from a single slab of CDT in 2+1 dimensions.

The weights assigned to the quadrangulation vertices are now given by [12][3]:

$$C_{l,N} = C_{l,l+2k} = \frac{4^k}{(1+k)!}\frac{(2l+3k-2)!!}{(2l+k)!!}l\binom{2l}{l} \tag{2.14}$$

where $N = l+2k$ is the number of cluster vertices contracted into that particular quadrangulation vertex, l is the degree of the quadrangulation vertex and k is a parameter. Each vertex in the quadrangulation will therefore have two independent parameters: its degree l and the number of cluster vertices N it represents.

These weights have a combinatorial interpretation as they count the number of clusters that can be made with N cluster vertices and l edges connecting the cluster to other clusters, with the constraint that each cluster vertex must be connected to exactly three other cluster vertices.

This combinatorial interpretation can be extended to the number of clusters that can be made with either N total 31- or 13-simplices and that are connected to a total of l 22-simplices. To understand this, first consider the triangulations of the lower spatial slice in Figure 4. The blue cluster vertices now each correspond to a triangle in this spatial slice and are connected if and only if the triangles are adjacent to each other. The weights of Equation 2.14 now count the number of triangulations that can be made using N triangles and with a total "circumference"

[3]With respect to the paper by Krikun, we have set $r = 0$ and $m = l$ since we are considering triangulations with a single boundary of length l.

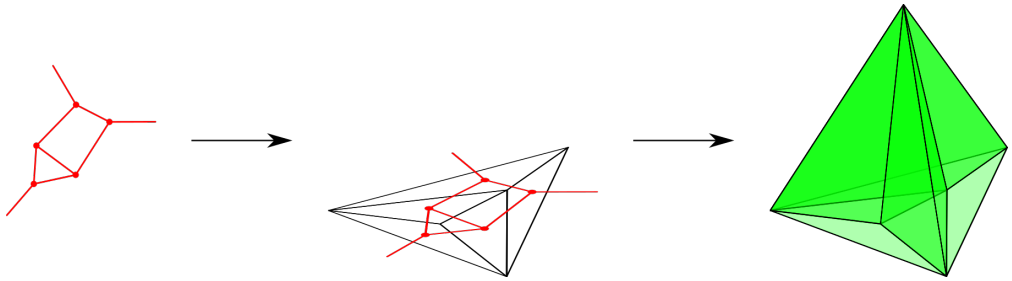

Figure 8: Interpreting a cluster of vertices first as representing a triangulation in a single time slice, and then as 31-simplices in a slab of CDT. Each vertex corresponds to a triangle in the triangulation, which are adjacent if the vertices are connected. Each triangle is the lower face of a 31-simplex and these simplices are adjacent if the triangles are adjacent and there are no 22-simplices involved.

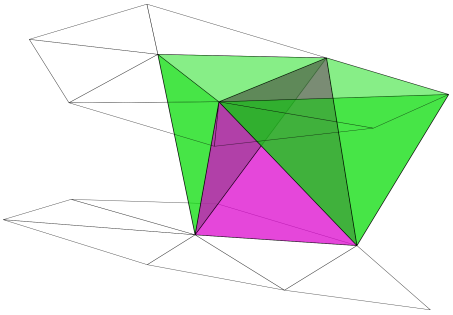

Figure 9: Two 13-simplices that have adjacent faces in the upper spatial triangulation but are not adjacent since they do not share a face. They are separated by a 22-simplex.

of l. This is visualised in the left part of Figure 8. Now the step to clusters of simplices can be made by noting that each triangle in the spatial triangulation (the middle of Figure 8) is a face of a 13-simplex. Since we consider clusters of 13-simplices without any 22-simplices, the 13-simplices are connected if and only if the triangles in the spatial triangulation are connected. If there would have been 22-simplices involved, two triangles in the spatial triangulation could be connected without their corresponding 13-simplices being adjacent. An example of this situation is given in Figure 9.

Using the weights of Equation 2.14, it is possible to write down the partition function for a single slab of CDT as a sum over quadrangulations Q. For a single slab of CDT, the partition sum is given by Equation 2.2 with $T = 1$:

$$Z_{CDT}^{(\text{Single slab})} = \sum_{\mathcal{T}_1([0,1]\times S^2)} \frac{1}{C(T)} e^{-S_E(N_0,N_3,1)}, \tag{2.15}$$

where the Euclidean action depends only on the number of vertices N_0 and the number of simplices N_3 in the triangulation.

Note that the boundary conditions in the time dimension are free as opposed to periodic as in Equation 2.2, resulting in the topology $([0,1] \times S^2)$ of the slab. The the topology of the quadrangulations representing this slab is that of the two-sphere, S^2.

We now restrict our attention to triangulations with fixed N_0 and N_3. It can be shown that

fixing N_0 and N_3 is equivalent to fixing the total number of quadrangulation and cluster vertices in a quadrangulation. This can be done using Euler's formula for planar maps to express N_0 and N_3 in terms of $\prod_{v\in Q} N_v$ and $\sum_{v\in Q} 1$. We can use the fact that the number of edges $\#E$ in a quadrangulation is related to the number of faces $\#F$ by $\#E = 4 \cdot \#F/2 = 2 \cdot \#F$. Using Euler's formula, the number of vertices $\#V$ in a quadrangulation is now given by

$$\#V = \#E - \#F + 2 = \#F + 2. \tag{2.16}$$

Therefore, fixing the number of vertices in a quadrangulation also fixes the number of faces. Which is equivalent to fixing the number of 22-simplices in the corresponding CDT triangulation. As the total number of cluster vertices in a quadrangulation is fixed, the total number of 13- and 31-simplices in the corresponding CDT triangulation is also fixed. Therefore, N_3 is fixed.

The number of vertices in a spatial slice of 2+1 dimensional CDT (which is a triangulation in two dimensions) is again given by Euler's formula. This time, the number of edges can be related to the number of faces as $\#E_{Slice} = \frac{3}{2}\#F_{Slice}$ so that the number of vertices is given by

$$\#V_{Slice} = \#E_{Slice} - \#F_{Slice} + 2 = \frac{\#F_{Slice}}{2} + 2. \tag{2.17}$$

Since a single slab of 2+1 dimensional CDT contains two such slices, the total number of vertices in such a slab is

$$N_0 = \#V_{Red} + \#V_{Blue} = \frac{\#F_{Red} + \#F_{Blue}}{2} + 4, \tag{2.18}$$

where the subscripts "Red" and "Blue" serve to distinguish the two time slices. Recall that the number of faces in both spatial slices corresponds to the number of 13- and 31-simplices in the CDT triangulation and that the sum of these simplices is the total number of cluster vertices in the quadrangulation. Therefore, fixing this total number of cluster vertices corresponds to fixing N_0.

The sum over all CDT triangulations, T, with fixed N_0 and N_3 can now be written as a sum over all quadrangulations. To do this, the vertices v of Q are labelled with N_v, which is the number of cluster vertices the vertex v represents. Recall that each triangulation can be represented by a quadrangulation. However, this representation is not unique as each vertex of the quadrangulation represents $C_{l,N}$ triangulations meaning that each quadrangulation Q counts $\prod_{v\in Q} C_{l_v,N_v}$ CDT triangulations. Using this fact, the partition sum of Equation 2.15 can be rewritten as

$$Z_{CDT}^{(\text{Single slab})} = \sum_{Q\in\mathcal{Q}} \frac{1}{C(Q)} e^{-S_E(N_0,N_3,1)} \prod_{v\in Q} C_{l_v,N_v}, \tag{2.19}$$

where $\mathcal{Q}$ is the set of all quadrangulations with fixed numbers of cluster and quadrangulation vertices and vertex labels N_v.

One can now study a single slab of 2+1 dimensional CDT by simulating the corresponding quadrangulation. By moving the parameter N_v, one can move many cluster vertices over large distances, as desired. The details of such a simulation will be discussed in Chapter 3.

It is worth pointing out that it is not a priori clear that the dually-weighted model constructed in this chapter has the same phase transition as the original single slab of CDT. This is because in the process of translating the single slab of CDT into a dually-weighted map, assumptions have been made on the slab of CDT. These assumptions act as restrictions on the model and could cause the two models to be inequivalent after all.

3 Methods

In this chapter, the details of the MCMC simulations of dually-weighted maps are discussed as well as some of the results obtained through these simulations.

The simulations of the dually-weighted maps described in the previous section will be done using a Markov-Chain Monte Carlo approach. This approach requires that one can sample quadrangulations with fixed numbers of cluster and quadrangulation vertices from the correct distribution which is governed by the weights of Equation 2.14. This distribution is given by [11]:

$$\pi(Q) = \frac{1}{Z}\frac{1}{C(Q)}\prod_{v\in Q} C_{l_v,N_v}, \tag{3.1}$$

which is the probability of finding the statistical system of quadrangulations in the state with vertex degrees l_v and where the vertex v represents N_v cluster vertices. In simulations, the factor $1/C(Q)$ is often implicitly taken into account by working with labelled or marked quadrangulations [1].

In the dually-weighted model under consideration, we will require two Markov-Chain moves: one move to change the structure of the map and one to move the cluster vertices from one cluster to another. These moves will be referred to as edge flips and mass moves, respectively. It will be convenient to consider both of these moves separately. That is, first perform the flip move while keeping the distribution of cluster vertices fixed and then perform the mass move while keeping the map structure fixed. This way, only one parameter of the vertices involved in the move will change as opposed to both of them.

The next challenge is to find suitable methods to sample from the desired stationary distribution of quadrangulations. For the edge flips, the Metropolis-Hastings algorithm was found to be suitable and we will discuss these moves first.

3.1 Implementing the flip moves

For the Metropolis-Hastings algorithm, one needs both a suitable proposal probability and an acceptance probability determined by the stationary distribution that needs to be sampled from. For the proposal probability, we chose to uniformly select an edge in the quadrangulation and flip it according to one of the two ways in Figure 10. This results in a proposal probability given by

$$S(Q \to Q') = \frac{1}{2}\frac{1}{\#E}, \tag{3.2}$$

where Q and Q' are two different quadrangulations with $\#E$ edges. The factor $1/2$ is due to the fact that we want both types of flip moves to be proposed with equal probability. Note that $S(Q \to Q') = S(Q' \to Q)$, so detailed balance is satisfied.

Using this proposal probability, the acceptance probability becomes

$$A(Q \to Q') = \min\left\{1, \frac{S(Q' \to Q)\pi(Q')}{S(Q \to Q')\pi(Q)}\right\} = \min\left\{1, \frac{\pi(Q')}{\pi(Q)}\right\}. \tag{3.3}$$

In Equation 3.3, the stationary distribution for quadrangulations is given by Equation 3.1. Since only four vertices are affected in either of the two flip moves, the quotient in Equation 3.3 simplifies significantly and becomes

$$A(Q \to Q') = \min\left\{1, \frac{C_{l-1,N}C_{k-1,M}C_{p+1,F}C_{q+1,G}}{C_{l,N}C_{k,M}C_{p,F}C_{q,G}}\right\}, \tag{3.4}$$

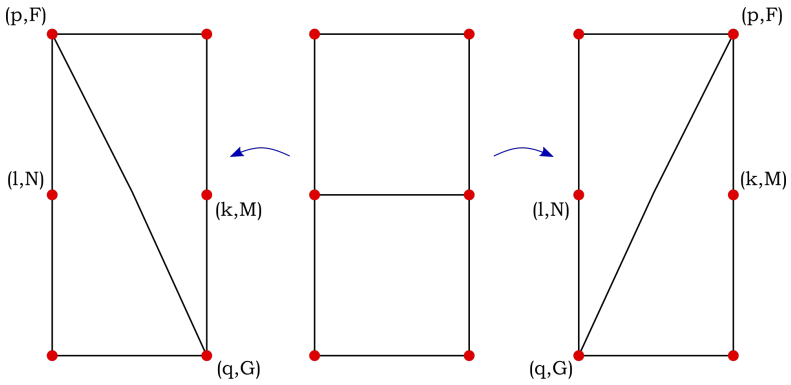

Figure 10: The two possible flip moves in a quadrangulation. Note that they are not their own inverse, making it necessary to have two flip moves. This is in contrast to the situation in triangulations, where we only require one flip move.

where the indices l, k, p, q, N, M, F, G correspond to the vertex parameters as shown in Figure 10. It is now a natural step to attempt to simplify this expression further by plugging in the explicit forms of the weights $C_{l,N}$ given by Equation 2.14 and fixing N, M, F and G. However, this brings with it a problem, which is that $N = l + 2k$. Because of this relation between N and k, changing the value of l by one and keeping N fixed will result in a half-integer value for k, which is not allowed. This problem can be easily circumvented by introducing a new parameter $f = k + 1$ which counts the number of faces interior to a cluster as shown in Figure 11. We call this new parameter the mass of the vertex. It can be shown that fixing the total mass in a quadrangulation Q is equivalent to fixing the total number of cluster vertices:

$$\begin{aligned}
\sum_{v\in Q} N_v &= \sum_{v\in Q}(l_v + 2f_v - 2) \\
&= \sum_{v\in Q} l_v - \sum_{v\in Q} 2 + 2\sum_{v\in Q} f_v \\
&= 4n - 2(n+2) + 2\sum_{v\in Q} f_v \\
&= 2n - 4 + 2\sum_{v\in Q} f_v, \qquad (3.5)
\end{aligned}$$

where n is the number of faces in the quadrangulation and the sum runs over all vertices, of which there are $n + 2$. The derivation of Equation 3.5 also uses that the sum of the degrees of all vertices in a map is equal to twice the number of edges and that in a quadrangulation, each square adds two edges.

The problem of half-integer values can now be solved by fixing f instead of N. This can be seen by rewriting the relation $N = l + 2k$ to find:

$$N = l + 2k \Rightarrow k = \frac{N - l}{2}. \qquad (3.6)$$

One can now use the definition $f = k + 1$ to find:

$$f = k + 1 = \frac{N - l}{2} + 1 \in \mathbb{Z}. \qquad (3.7)$$

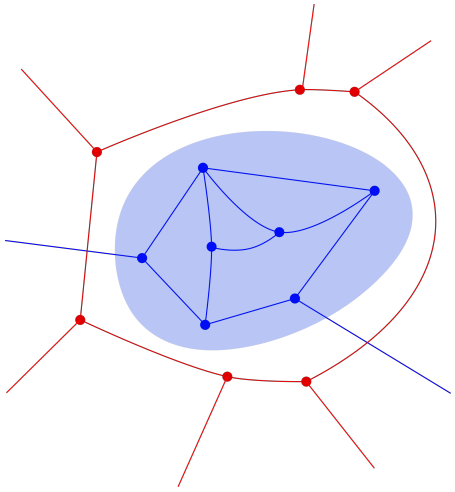

Figure 11: A visual representation of the meaning of the mass f of a quadrangulation vertex. The cluster that will be contracted into a quadrangulation vertex is shaded in blue. Its mass f is the number of faces in the shaded region, so 4 in this case.

For ease of notation we define: $C_{l,f} := C_{l,l+2f-2}$.

Due to the high similarity of terms in the nominator and denominator of Equation 3.4, it is convenient to consider a fraction of the form $C_{l\pm1,f}/C_{l,f}$. These fractions can be rewritten as

$$\begin{aligned} \frac{C_{l-1,f}}{C_{l,f}} &= \frac{2l+f-1}{2l+3f-5}\frac{l+1}{2(2l+1)} \\ \frac{C_{l+1,f}}{C_{l,f}} &= \frac{2l+3f-3}{2l+f+1}\frac{2(2l+1)}{l}. \end{aligned} \tag{3.8}$$

These fractions no longer contain any factorials, making them easy to compute numerically even for large values of l and k.

3.2 Implementing the mass move

To give the quadrangulation the correct distribution of cluster vertices, we follow the same procedure as for the flip moves. We start by determining a suitable algorithm to sample from the stationary distribution 3.1. We opt to again use the Metropolis-Hastings algorithm to perform the mass moves.

The algorithm will select two different vertices at random and attempt to move a random amount of mass from one vertex to the other. The vertices will be chosen uniformly amongst all vertices and are demanded to be distinct. The probability of choosing a given pair of such vertices is $(N(N-1))^{-1}$. Let these vertices have degrees l, k, and masses f, g and assume the mass is moved from the vertex with degree l to the other vertex. Then the amount of mass Δ that is being moved will be drawn uniformly from the set $\{n|1 \leq n \leq f\}$. Note that $\Delta = 0$ amounts to doing nothing so we exclude it as an option. The probability of selecting a particular value of Δ is now $(\#\{n|1 \leq n \leq f\})^{-1} = f^{-1}$, so the proposal probability for such a mass move will be

$$S(Q \to Q') = \frac{1}{N(N-1)}\frac{1}{f}, \tag{3.9}$$

where N is the total number of vertices in the simulation. The first factor takes into account the uniform selection of two unique vertices, while the second factor takes into account the uniform selection of Δ. Note that this proposal probability does not satisfy detailed balance unless $f = g + \Delta$, this means it will contribute a prefactor to the acceptance probability depending on f.

The acceptance probability will again take the form of Equation 3.3. But since only two vertices participate in the move, there will only be two factors in the nominator and denominator that do not cancel out. However, these factors will depend on five parameters: f, g, l, k and Δ. The acceptance probability will be

$$A(Q \to Q') = \min\left\{1, \frac{f}{g+\Delta}\frac{C_{l,f-\Delta}C_{k,g+\Delta}}{C_{l,f}C_{k,g}}\right\}, \tag{3.10}$$

where the fraction of weights can be written out as

$$\frac{C_{l,f-\Delta}C_{k,g+\Delta}}{C_{l,f}C_{k,g}} = \frac{f!g!(2l+f-1)!!(2k+g-1)!!(2l+3(f-\Delta)-5)!!(2k+3(g+\Delta)-5)!!}{(f-\Delta)!(g+\Delta)!(2l+3f-5)!!(2k+3g-5)!!(2l+(f-\Delta)-1)!!(2k+(g+\Delta)-1)!!}. \tag{3.11}$$

The expression of Equation 3.11 presents a problem, the double factorials do not cancel out as they did in the case of the flip move. The reason for this is that in the case of the flip move, the only parameters that changed during the move were the degrees of the participating vertices. These degrees are multiplied by a factor of two everywhere in Equation 3.11, so the terms containing the degrees are always even. Furthermore, the degree of the participating vertices could only change by one, making all but one term in each pair of double factorials cancel out. Both of these things are not true for the mass move, where there is a range of mass that can be moved and the factors multiplying the mass are odd.

These factorials are not only computationally expensive to calculate but they also grow very fast. The size of the double factorial terms results in numerical accuracy problems that can have profound consequences on the result. Therefore, we need an alternative method of determining the acceptance probability.

3.2.1 First Attempt: Approximation for Large Mass

As a first attempt, one could realise that the problems with the double factorials only arise when the numbers we wish to take the (double) factorial of become large. A natural thing to do would therefore be to take the asymptotic limit of $C_{l,f}$ as the parameters become large. As a first attempt, we take the limit of large mass, f, since this will be the largest parameter. Since the double factorial behaves differently depending on whether f is even or odd, there are two different limits to consider[4]:

$$\lim_{\substack{f\to\infty \\ f \bmod 2=0}} C_{l,f}; \tag{3.12}$$

$$\lim_{\substack{f\to\infty \\ f \bmod 2=1}} C_{l,f}, \tag{3.13}$$

[4]For those unfamiliar with the notation: $f \bmod 2$ gives the remainder of f when dividing by 2.

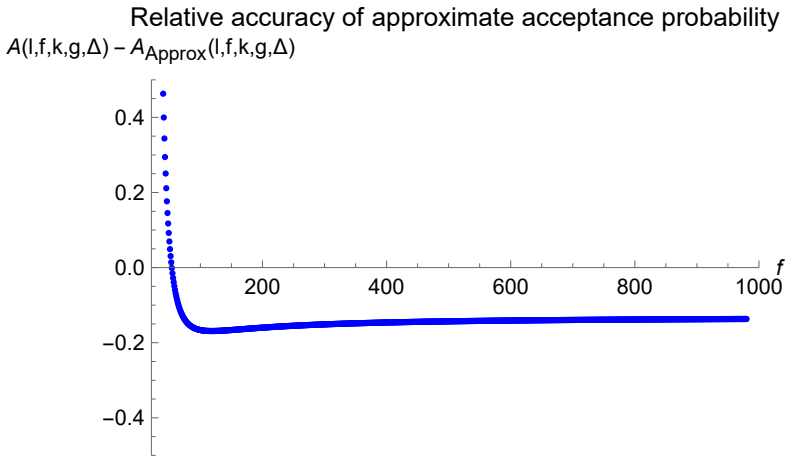

Figure 12: The difference between the exact value of Equation 3.11 and the value obtained using the approximate weights $C_{l,f}$ as in Equation 3.14. The values of the parameters used for this plot are: $l = 6, k = 9, g = 100, \Delta = 20$.

where $f \in \mathbb{N}$. It turns out that both cases show the same asymptotic behaviour if expanded up to second order in f around $f = \infty$:

$$\lim_{\substack{f\to\infty \\ f \bmod 2=0}} C_{l,f} = \lim_{\substack{f\to\infty \\ f \bmod 2=1}} C_{l,f} = \frac{2^{2f-\frac{9}{2}}\, 3^{\frac{3f}{2}+l-4}\, l\,(36f - 24l(2+l) - 47)}{f^{\frac{7}{2}}\sqrt{\pi}} \binom{2l}{l}. \tag{3.14}$$

The asymptotic forms of $C_{l,f}$ can now be substituted into Equation 3.11 to obtain an approximate expression for Equation 3.10:

$$A(Q \to Q') \approx \min\left\{1, \frac{f^{\frac{7}{2}} g^{\frac{7}{2}} (47 + 36f - 24l(2+l) - 36\Delta)(47 + 36g - 24k(2+k) + 36\Delta)}{(f-\Delta)^{\frac{7}{2}}(g+\Delta)^{\frac{7}{2}}(47 + 36f - 24l(2+l))(47 + 36g - 24k(2+k))}\right\}. \tag{3.15}$$

Unfortunately, the approximation of Equation 3.15 does not fare too well, as can be seen from Figure 12. In this figure, the approximated acceptance probability deviates from the exact acceptance probability by about 0.17, even for large values of f. While this might be due to the fact that g is too small, the approximation also diverges from the exact value for small values of f, which is to be expected since the approximation consists of taking the large f limit. However, to implement this approximation effectively, one needs to know for what range of values of the parameters it is valid. It turns out that determining the location of the deviation as a function of the other parameters is not easy.

3.2.2 Second Attempt: Crossing-out Terms

Given the difficulties with the approximation approach to the acceptance probability, it might be worthwhile to consider another approach that does not rely on approximate expressions, but rather on clever simplification of the expression to be calculated. For this, we note that the expression of Equation 3.11 consists of a fraction of two products. Usually, when faced with such a problem, the first simplification step would be to cross out equal terms in the nominator and denominator. But it was noted earlier that the parity of f and g gives rise to two different

double factorials and that the fraction in Equation 3.11 can therefore not be simplified in a general manner. It is thus necessary to evaluate each case individually. That is, to perform the simplification for each mass move individually. Needless to say, this requires an algorithm so that it can be done by the simulation code itself.

The algorithm used to simplify Equation 3.11 computes the value of each factor in nominator and denominator and stores them in a single associative container (more specifically, a C++ map, which is comparable to a Python dict). The container uses the value of the factor as the key identifier of the entry and assigns to each entry an integer value. This value will be raised by one if the factor appears in the nominator and lowered by one if the factor appears in the denominator. The resulting container will contain all factors and an integer value, which is the exponent of the factor. For example, consider the case where $l = 1, f = 3, k = 1, g = 1, \Delta = 1$, then:

$$\begin{aligned}\frac{C_{l,f-\Delta}C_{k,g+\Delta}}{C_{l,f}C_{k,g}} &= \frac{3!\,1!\,(2+3-1)!!\,(2+1-1)!!\,(2+3(3-1)-5)!!\,(2+3(1+1)-5)!!}{(3-1)!\,(1+1)!\,(2+3\cdot 3-5)!!\,(2+3-5)!!\,(2+(3-1)-1)!!\,(2+(1+1)-1)!!} \\ &= \frac{3!\cdot 1!\cdot 4!!\cdot 2!!\cdot 3!!\cdot 3!!}{2!\cdot 2!\cdot 6!!\cdot 0!!\cdot 3!!\cdot 3!!} \\ &= \frac{3\cdot 2\cdot 4\cdot 2\cdot 2\cdot 3\cdot 3}{2\cdot 2\cdot 6\cdot 4\cdot 2\cdot 3\cdot 3}. \end{aligned} \tag{3.16}$$

The associative container will now contain the key-value pairs: $(6,-1),(4,0),(3,1),(2,0)$ since $6, 4, 3$ and 2 are all the factors that appear when expanding the double factorials[5]. For specific values of the parameters, Equation 3.11 can now be expressed as the product of all keys in the container raised to the power of their value. So the fraction of Equation 3.16 can be expressed as $6^{-1}\cdot 3$. Note that this procedure has reduced the number of multiplications required in this example from 13 down to 1.

One might wonder if this algorithm will run into numerical accuracy problems for large values of the parameters, as the integers being divided may be of vastly different orders of magnitude. This problem can be circumvented by dividing the largest factor in the nominator by the largest factor in the denominator first, multiplying these with the intermediate answer F and iterating this procedure. This way, the ratio of two factors will always be close to one, since the factors are not expected to change much from the largest to the second largest. This has another problem, which is that F might reach a point where it no longer fits in its designated computer memory before all factors have been accounted for. In such an event, the simulation will (depending on the programming language, which in our case is C++) simply assign F the value zero or infinity, depending on whether it was too small or too large to fit in the memory. All ratios of factors following this will not be able to change the value of F.

In an attempt to circumvent this problem, one might try to follow a multiplication of F with a factor greater than one by a multiplication with a factor smaller than one. The question that remains is how to find ratios that are greater and smaller than one efficiently. Luckily, it turns out that the largest and the smallest factors are either both in the nominator or both in the denominator. One can therefore alternate between taking the largest and smallest factor in both the nominator and denominator and divide them. An example of the procedure is given by taking $l = k = g = \Delta = 2, f = 3$:

$$\begin{aligned}\frac{C_{2,3-2}C_{2,2+2}}{C_{2,3}C_{2,2}} &= \frac{11\cdot 9\cdot 7\cdot 6\cdot 5\cdot 5\cdot 4\cdot 3\cdot 3\cdot 3\cdot 2\cdot 2\cdot 2\cdot 2}{8\cdot 7\cdot 6\cdot 5\cdot 5\cdot 4\cdot 4\cdot 4\cdot 3\cdot 3\cdot 3\cdot 2\cdot 2\cdot 2} \\ &= \frac{11\cdot 9\cdot 2}{8\cdot 4\cdot 4}. \end{aligned} \tag{3.17}$$

[5]In the simulation code, a key of 1 is allowed, but since this should not affect the results of the computation, it is omitted here.

Here, the largest factors are marked in red and the smallest factors in blue. Applying the aforementioned procedure, we find:

$$\frac{C_{2,3-2}C_{2,2+2}}{C_{2,3}C_{2,2}} = \frac{11}{8}\cdot\frac{2}{4}\cdot\frac{9}{4}, \tag{3.18}$$

where indeed the factors alternate between being larger than one and smaller than one.

This algorithm of expanding the factorials, crossing out identical factors and computing the product of surviving factors could theoretically be very accurate and efficient. When the computation of the product of ratios is done cleverly, the only limiting factor would be the numerical accuracy. It is also potentially faster than computing the fraction $C_{l,f-\Delta}C_{k,g+\Delta}/C_{l,f}C_{k,g}$ directly since the number of multiplications is generally greatly reduced. However, neither benefit applies if one cannot prevent divergences in the intermediate answer F. The latter proves to be difficult, even with the mentioned precautions. It will therefore again be worthwhile to explore a different method of computing Equation 3.11.

3.2.3 Third Attempt: Gamma Functions

For the third attempt to compute $C_{l,f-\Delta}C_{k,g+\Delta}/C_{l,f}C_{k,g}$, we write the (double) factorials in terms of the gamma-function. One can write factorials in terms of gamma functions as

$$n! = \Gamma(n+1), \tag{3.19}$$

for $n \in \mathbb{N}$. Double factorials can be written in terms of single factorials, but the expression depends on the parity of the integer in the double factorial. This results in the two expressions:

$$\begin{aligned} n!! = (2k)!! &= 2^k k! \\ n!! = (2k+1)!! &= \frac{(2k)!}{2^k k!}, \end{aligned} \tag{3.20}$$

with $k \in \mathbb{Z}$. These expressions can both be written in terms of gamma functions using Equation 3.19.

Note that Equation 3.11 consists of several fractions of similar form:

$$\begin{aligned} &\frac{C_{l,f-\Delta}C_{k,g+\Delta}}{C_{l,f}C_{k,g}} = \\ &\frac{f!g!(2l+f-1)!!(2k+g-1)!!(2l+3(f-\Delta)-5)!!(2k+3(g+\Delta)-5)!!}{(f-\Delta)!(g+\Delta)!(2l+3f-5)!!(2k+3g-5)!!(2l+(f-\Delta)-1)!!(2k+(g+\Delta)-1)!!} = \\ &\frac{f!}{(f-\Delta)!}\cdot\frac{g!}{(g+\Delta)!}\cdot\frac{(2l+f-1)!!}{(2l+3f-5)!!}\cdot\frac{(2k+g-1)!!}{(2k+3g-5)!!}\cdot \\ &\qquad\frac{(2l+3(f-\Delta)-5)!!}{(2l+(f-\Delta)-1)!!}\cdot\frac{(2k+3(g+\Delta)-5)!!}{(2k+(g+\Delta)-1)!!}. \end{aligned} \tag{3.21}$$

Since the ratios of double factorials all have such a similar form, we will only rewrite one in terms of gamma functions. Doing this for $(2l+f-1)!!/(2l+3f-5)!!)$ with f even gives:

$$\frac{(2l+f-1)!!}{(2l+3f-5)!!} = \frac{2^{2-f}\,\Gamma\left(\frac{1+f}{2}+l\right)}{\Gamma\left(\frac{3}{2}(f-1)+l\right)}. \tag{3.22}$$

Doing the same for f odd gives the exact same result, meaning there is no need to distinguish between even and odd vertex mass. Note that this need never existed for the degree of the vertex, since it is always multiplied by an even integer.

Since the gamma functions grow as fast as factorials, it is necessary to take precautions to avoid numbers that are too large for the simulation to handle. To this end, we consider the logarithm of Equation 3.11. This allows Equation 3.11 to be written as a sum of logarithms. Since the full expression is rather cumbersome and mostly contains similar terms, we give only the logarithm of Equation 3.22, which can be written as

$$\log\left(\frac{(2l+f-1)!!}{(2l+3f-5)!!}\right) = \log(2^{2-f}) + \log\left(\Gamma\left(\frac{1+f}{2}+l\right)\right) - \log\left(\Gamma\left(\frac{3}{2}(f-1)+l\right)\right), \quad (3.23)$$

where the first logarithm will drop out in the final expression for the acceptance probability.

The sum of these logarithms will be small enough to store in computer memory and perform operations on. The final result of these operations can then be converted back by taking the exponential. This procedure has as a benefit that one can use the built-in functions of the C++ cmath library, which contains a function for computing the logarithm of the gamma function, $\ln(\Gamma(x))$. This function has accuracy as good as a direct computation while being significantly faster and less prone to coding errors than the other methods discussed previously. It is for these reasons that using log-gamma functions is the preferred method for computing Equation 3.11 and will be used in the remainder of this thesis.

3.3 Improving the mass move

In the previous section, it was discussed how the acceptance probability for the mass move can be computed efficiently. This section will delve into improving this move to make the simulations more efficient. In this context, more efficient means that fewer proposed moves are required to achieve the same results.

To achieve a higher efficiency for the simulations, one might try to tweak the proposal probability to propose more moves that have a larger probability of being accepted. For this, it can be useful to look at the acceptance probability for the mass move given by Equation 3.10 and attempt to determine for what values of Δ it is largest. However, due to the erratic nature of the acceptance probability and its dependence on five variables, this is more easily said than done. Although the acceptance probability is difficult to analyse, one can expect it to be relatively large for small values of Δ since the configuration does not change much when applying such a small mass move.

We choose to limit the mass Δ that can be moved to be at most $\lfloor f/2 \rfloor + 1$. By restricting the range of possible values of Δ that can be proposed, the proposal probability of Equation 3.9 will need to be altered. The set of values Δ can take will now be $\{n | 1 \leq n \leq \lfloor f/2 \rfloor + 1\}$ so that the probability of selecting a particular value of Δ will be $(\lfloor f/2 \rfloor + 1)^{-1}$. The proposal probability for this improved move will be:

$$S(Q \to Q') = \frac{1}{N(N-1)} \frac{1}{\lfloor f/2 \rfloor + 1}. \quad (3.24)$$

Another move worth attempting is a move that selects two vertices at random and exchanges their masses. The proposal probability for such a move will be

$$S(Q \to Q') = \frac{1}{N(N-1)}, \quad (3.25)$$

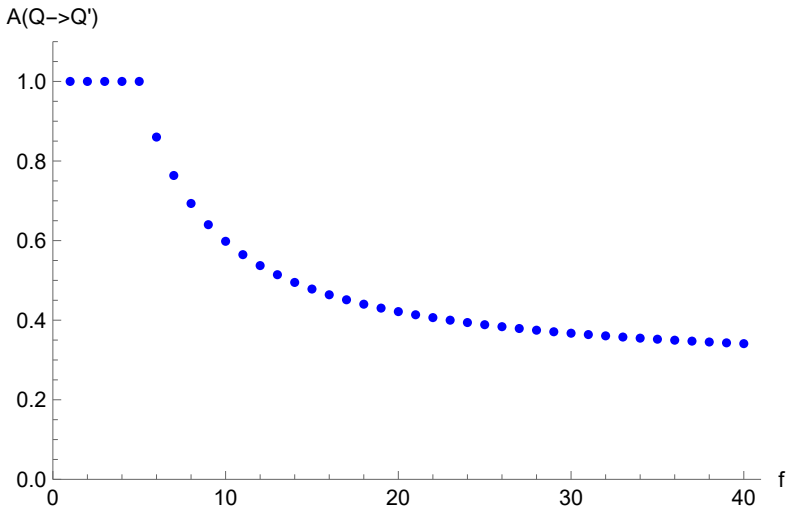

Figure 13: The acceptance probability of a mass move that exchanges the masses of two random vertices as a function of one of the vertex masses. The plot was made using $l = 4, k = 2, g = 5$ and with f ranging from 1 to 40. The acceptance probability is decent for all values of f. In particular, for $f \geq g$ it is equal to one.

and the resulting MH acceptance probability will be

$$A(Q \to Q') = \min\left\{1, \frac{C_{l,g} \cdot C_{k,f}}{C_{l,f} \cdot C_{k,g}}\right\}, \tag{3.26}$$

where l, k and f, g are the degrees and masses of the participating vertices, respectively. The acceptance probability of Equation 3.26 will generally be quite significant, as can be seen in Figure 13.

Theoretically, the moves that exchange vertex masses are not necessary for the simulation to be able to sample the full space of quadrangulations. In practice, however, it is useful to include these moves to increase the efficiency of the simulation. That is, the simulation will first attempt to move a small mass $1 \leq \Delta \leq \lfloor f/2 \rfloor + 1$ from the vertex with mass f to the other vertex. If this fails, it will attempt to exchange both masses.

4 Results

We will now discuss the results from simulating the model of dually-weighted maps. The code used for these simulations can be found at [13]. The main goal of these simulations is to determine the existence of a phase transition that corresponds to the phase transition in a single slab of 2+1 dimensional CDT. To find such a phase transition, it is necessary to know what it is expected to look like.

Recall that the phase transition in a single slab of CDT is from a phase where there are macroscopic clumps of 13- and 31-simplices to a phase where these clumps are much smaller. Since the size of a clump is related to the mass and degree of a quadrangulation vertex by $N = l + 2f - 2$, a large clump will manifest itself in the quadrangulation as a vertex with a large degree or a large mass. However, the mass and degree of a quadrangulation vertex are not independent. This can perhaps be better understood using the three-valent maps in the leftmost

part of Figure 6. For increasingly larger clusters of vertices in such a map, it is increasingly unlikely the vertices of such a cluster can all link to three other vertices inside the cluster. This forces some of them to link with vertices from other clusters, increasing the number of outgoing edges. Since this number of outgoing edges is exactly the degree of the vertex in the quadrangulation, the degree of this vertex will increase with the mass of the vertex. It is therefore expected that a large cluster of simplices manifests itself as a quadrangulation vertex with both a large mass and a large degree. The expected phase transition in the Dually-Weighted model is therefore from a phase where the degrees and masses of the vertices are rather uniform to a phase where there exist a few vertices with large degree and large mass.

To probe the existence of a phase transition, we measure four observables:

- the maximum vertex degree ($\max\{l_v | v \in Q\}$);
- the maximum vertex mass ($\max\{f_v | v \in Q\}$);
- the sum of all squared vertex degrees $\left(\sum_{v\in Q} l_v^2\right)$;
- the sum of all squared vertex masses $\left(\sum_{v\in Q} f_v^2\right)$,

where Q is the quadrangulation on which the measurements are performed. The reason for including the last two observables is that, in contrast to the first two, they contain information on the entire quadrangulation instead of just one vertex. By the reasoning performed earlier, the observables should all show a jump at the phase transition.

The results from the simulations are shown in Figure 14. These simulations were performed with 1002 vertices and a total mass ranging from 10 to 2010. Each point is the average of 40000 measurements of that observable at that total mass.

What is immediately clear is that there is no jump in any of the observables, which indicates there is no phase transition. The maximum mass remains relatively low, meaning that no macroscopic amount of mass has gathered at a single vertex. What is more suspicious is the fact that the maximum vertex degree is very large for small total masses N. From the discussion so far, there is no apparent reason for the vertices to prefer having a large degree when there is little mass in the simulation. A possible explanation for this will be given later.

To perform a check on the simulation code, we measure the distribution the mass move samples from to confirm that it is indeed the distribution it should be. This can be done by selecting two vertices and repeatedly applying the mass move on them while keeping the map itself fixed. That is, we repeatedly attempt to move mass between the same two vertices but do not perform any flip moves.

The mass of one of these vertices should follow the distribution given by

$$\pi(l,k,f,N) = \frac{1}{Z}\frac{1}{C(Q)}\prod_{v\in Q} C_{l_v,f_v} \propto C_{l,N}C_{k,N-f}. \tag{4.1}$$

Here, l and k are the degrees of the participating vertices, f is the mass of one of these vertices and N is the total mass of these two vertices. Note that l, k and N are kept fixed during the test so f is the only free parameter determining the quadrangulation. This is the reason the mass f of a single vertex should follow the distribution of Equation 4.1, even though this is a distribution of quadrangulations. By keeping track of the mass of one of the vertices during the test, a histogram of the vertex mass can be made.

Performing such a test for a quadrangulation with 102 vertices, a total mass of 10000 and vertices with degrees $l = 1$ and $k = 3$ and total mass $N = 16$ gives rise to the histogram

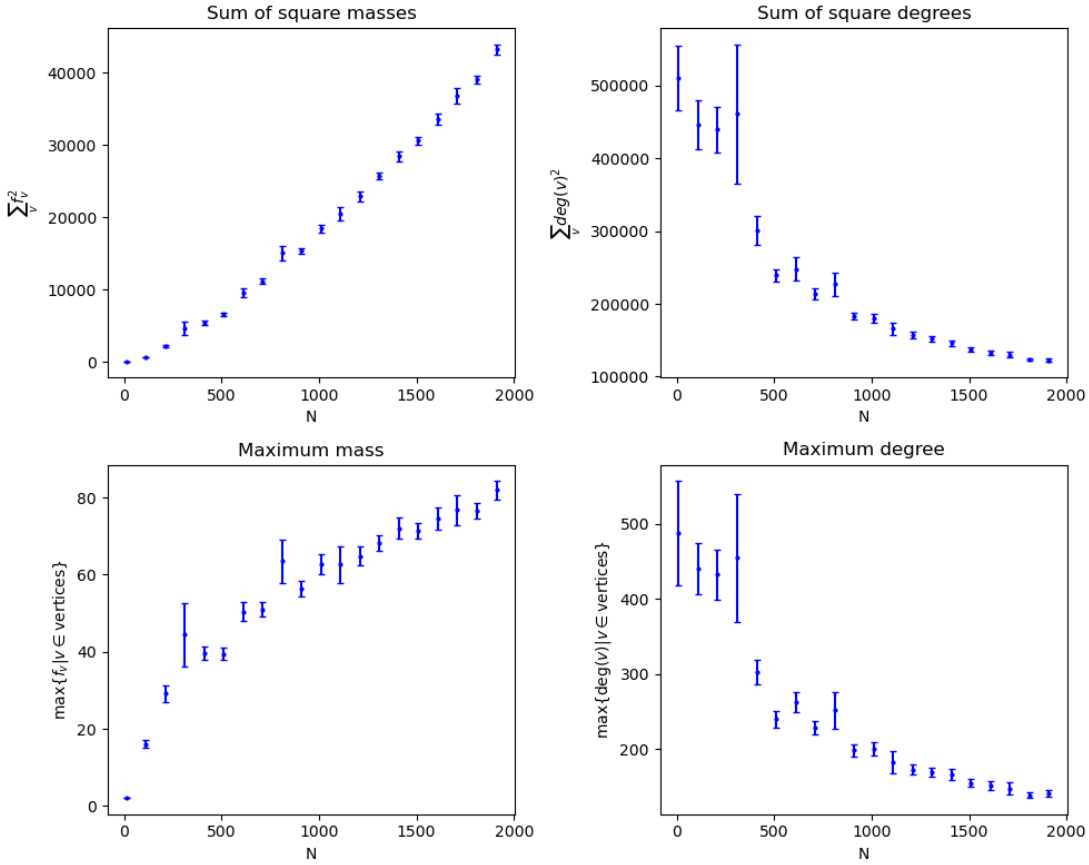

Figure 14: Results from a simulation of quadrangulations with 1002 vertices and a total mass N. The maximum mass remains rather low for all values of N. However, the maximum degree shows odd behaviour for low N, where it assumes very large values.

shown in Figure 15. The exact distribution of Equation 4.1 is shown in Figure 16. While the histogram in Figure 15 shows the correct preference for large masses, the overall shape is far from correct. The scale on the y-axis in this histogram is also much broader than that for the exact distribution due to the existence of a single histogram entry of mass zero. Neglecting this etnry gives the histogram in Figure 17, which has a shape that is more similar to, but is still not in good agreement with the exact distribution. This might indicate a flaw in the code that could have impacted the results of the simulations.

This discrepancy between the measured and exact mass distributions does, however, not explain the oddly large vertex degrees present in quadrangulations with small total mass. This behaviour can be explained using an approximation to the weights of Equation 2.14 for large degrees and zero mass:

$$C_{l,N} = C_{l,l-2} \approx 2^{2l-7} \cdot \frac{8l+15}{l^{5/2}\sqrt{\pi}}. \tag{4.2}$$

These weights decrease with increasing degree l of the vertex, suggesting that such a vertex is unlikely to exist. However, it is the probability of finding a particular quadrangulation that

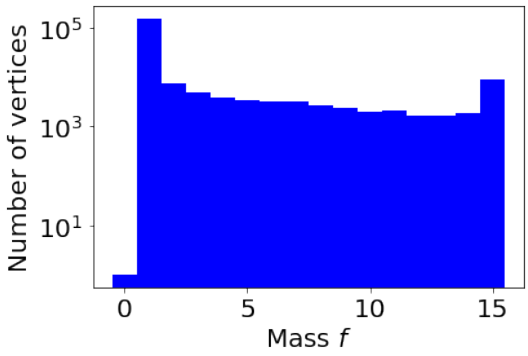

Figure 15: A histogram of the mass of a single vertex in a simulation of quadrangulations with 102 vertices and a total mass of 10k. The mass of the vertex has been recorded 200k times to make the histogram. The vertices involved had degrees of $l = 1$ and $k = 3$, respectively and a total mass of $N = 16$. The preference for large masses is clearly visible, but the shape is hard to judge given the scale on the y-axis.

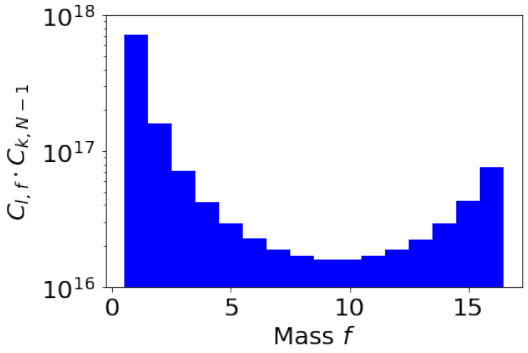

Figure 16: A plot of Equation 4.1 for $l = 1, k = 3$ and $N = 16$. The maximum of this histogram at large vertex degrees agrees with the histogram obtained from numerical simulations.

must be considered. The probability of finding a particular quadrangulation Q is governed by the Boltzmann weight of Q:

$$\begin{aligned} W_{Boltzmann}(Q) &\propto \prod_{v \in Q} C_{l_v, N_v} \\ &\approx \prod_{v \in Q} 2^{2l_v - 7} \cdot \frac{8l_v + 15}{l_v^{5/2}\sqrt{\pi}} \\ &= 2^{\left(\sum_{v \in Q}(2l_v - 7)\right)} \cdot \prod_{v \in Q} \frac{8l_v + 15}{l_v^{5/2}\sqrt{\pi}}, \end{aligned} \tag{4.3}$$

where the sum and product run over all vertices in the quadrangulation Q. The larger the Boltzmann weight of a quadrangulation, the more likely it is to occur. Since the number of faces in a quadrangulation is fixed during a simulation, the number of edges is as well. Also, the sum of all vertex degrees equals twice the number of edges in a map so the sum in the exponent is a constant factor and does not contribute to the behaviour of the simulation.

Using that every weight assigned to a vertex with degree l carries with it a factor of 2^{2l-7}, we find that the behaviour of the Boltzmann weights is approximately dependent on the weight of each vertex divided by this factor:

$$\frac{C_{l,l-2}}{2^{2l-7}}. \tag{4.4}$$

This means that the Boltzmann weight of a quadrangulation will be proportional to the product over all these reduced weights:

$$W_{Boltzmann}(Q) \propto \prod_{v \in Q} \frac{C_{l_v, N_v}}{2^{2l_v - 7}}. \tag{4.5}$$

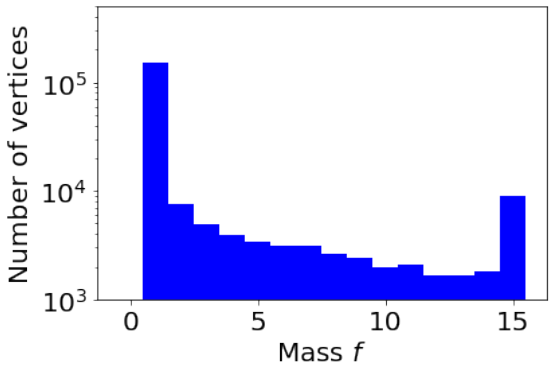

Figure 17: The histogram of Figure 15 with an adjusted range of the y-axis to better show the shape of the histogram. The similarities with Figure 4.1 are now more apparent, but the two histograms are still quite different.

Some values of the reduced weights of Equation 4.4 for fixed $f = 0$ are shown in Figure 18. The values of the reduced weights decrease with increasing degree, but what is interesting is that the decay is rapid at first and then slows down as the degrees of the vertices become larger. Therefore, it will be beneficial for maximising the Boltzmann weights if the vertices in a quadrangulation have a minimal degree. But since the sum of all vertex degrees is fixed, vertices with small degrees must be offset by vertices with large degrees. This means that quadrangulations with a few vertices with very large degrees will have higher Boltzmann weights than quadrangulations where all vertices have approximately the same degree. For example: consider two quadrangulations, Q_1 and Q_2, each with 12 vertices and 20 edges. Assume that Q_1 has 8 vertices of degree four and four vertices of degree 2. Also, assume that Q_2 has a single vertex of degree 18 and all other vertices have degree 2. The ratio of the Boltzmann weights for these two maps would be

$$\frac{W_{Boltzmann}(Q_1)}{W_{Boltzmann}(Q_2)} = \frac{\prod_{v\in Q_1} \frac{C_{l_v,N_v}}{2^{2l_v-7}}}{\prod_{w\in Q_2} \frac{C_{l_w,N_w}}{2^{2l_w-7}}} = \frac{1^8 \cdot 8^4}{(17678835/268435456)^1 \cdot 8^{11}} = 5.79224 \cdot 10^{-5}, \quad (4.6)$$

so Q_2 is strongly preferred over Q_1. This means that the quadrangulations with a few vertices of very large degrees are more likely to occur during simulations than quadrangulations where all vertex degrees are approximately equal.

5 Conclusions

To conclude, we have investigated an alternative method for simulating a single slab of 2+1 dimensional CDT. This method is based on the manipulation of dually-weighted planar maps using Markov-Chain Monte Carlo methods. The goal of this thesis was to find out if this alternative approach would be better suited for simulating the phase transition in 2+1 dimensional CDT than traditional simulation methods, as previous simulations suffered from critical slowing down.

Unfortunately, the results show no sign of a phase transition in the dually-weighted model. A quick check on the generated distribution of vertex masses shows that this part of the simulation code does not quite produce the desired results. So although the results suggest the absence of a phase transition, one cannot draw a definitive conclusion from these results.

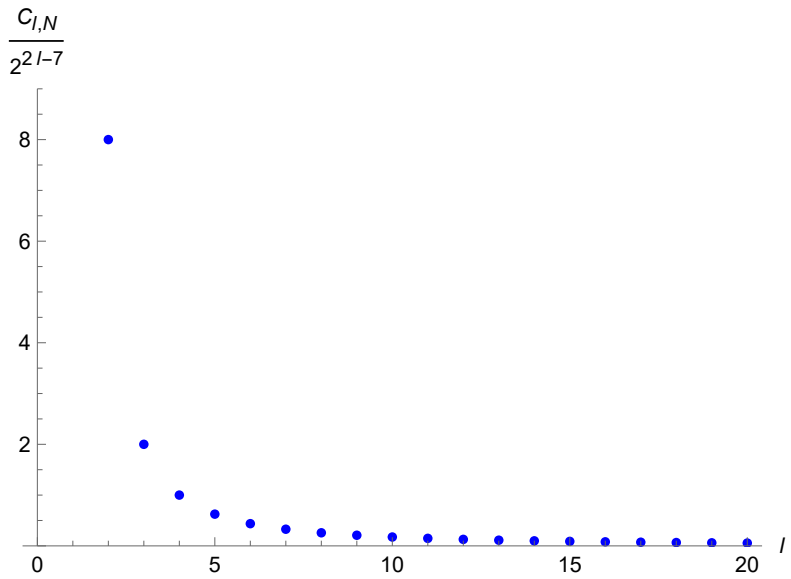

Figure 18: Values of the reduced weights of the vertices $C_{l,N}/2^{2l-7}$ as a function of the vertex degree, l. The weights decrease with increasing vertex degree, which gives rise to a preference for a couple of large vertex degrees and many small degrees.

The reason for the possible absence of a phase transition will most likely have to do with the constraints imposed on the dually-weighted model. One of these constraints is that all maps inside a cluster must be connected, while this is not required by CDT. Since it is currently unknown which constraints possibly cause the phase transition to vanish, further investigation will be necessary.

6 Acknowledgements

I would like to thank Dr. Timothy Budd for being my supervisor during my internship. He has masterfully guided me through this part of my bachelor's by providing excellent explanations on a subject that used to be completely foreign to me and always taking time to answer my questions.

References

[1] T. G. Budd. "Non-perturbative quantum gravity: a conformal perspective". PhD thesis. Universiteit Utrecht, Mar. 20, 2012.

[2] Jan Ambjørn. *Lattice Quantum Gravity: EDT and CDT*. 2022. DOI: 10.48550/ARXIV.2209.06555.

[3] J. Ambjørn, J. Jurkiewicz, and R. Loll. "Non-perturbative 3d Lorentzian Quantum Gravity". In: *Phys. Rev. D* 64 (July 2001). DOI: 10.1103/PhysRevD.64.044011.

[4] Michael E. Peskin and Daniel V. Schroeder. *An introduction to quantum field theory*. Addison-Wesley Pub. Co., 1995, p. 842. ISBN: 978-0-201-50397-5.

[5] F. Wilczek. "On absolute units, I: Choices". English. In: *Physics Today* 58.10 (Oct. 2005). WOS:000232422000002, pp. 12–13. ISSN: 0031-9228. DOI: 10.1063/1.2138392. URL: https://www-webofscience-com.ru.idm.oclc.org/api/gateway?GWVersion=2&SrcAuth=DOISource&SrcApp=WOS&KeyAID=10.1063%2F1.2138392&DestApp=DOI&SrcAppSID=EUW1ED0CB8CKqJtYaP96MzuX545pL&SrcJTitle=PHYSICS+TODAY&DestDOIRegistrantName=American+Institute+of+Physics (visited on 07/05/2023).

[6] Jan Ambjørn. *Quantization of Geometry*. 1994. DOI: 10.48550/ARXIV.HEP-TH/9411179.

[7] Raymond K. W. Wong et al. "Fiber direction estimation, smoothing and tracking in diffusion MRI". In: *The annals of applied statistics* 10.3 (2016), p. 1137.

[8] R. Loll. "Quantum gravity from causal dynamical triangulations: a review". In: *Classical and Quantum Gravity* 37.1 (Dec. 2019), p. 013002. DOI: https://doi.org/10.1088/1361-6382/ab57c7.

[9] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013. URL: https://artowen.su.domains/mc/.

[10] T.G.Budd. *Monte Carlo Techniques*. Nov. 16, 2022. URL: https://hef.ru.nl/~tbudd/mct/intro.html (visited on 06/13/2023).

[11] Vladimir A. Kazakov, Matthias Staudacher, and Thomas Wynter. "Character expansion methods for matrix models of dually weighted graphs". en. In: *Communications in Mathematical Physics* 177.2 (Apr. 1996), pp. 451–468. ISSN: 1432-0916. DOI: 10.1007/BF02101902. URL: https://doi.org/10.1007/BF02101902 (visited on 06/06/2023).

[12] Maxim Krikun. "Explicit enumeration of triangulations with multiple boundaries". English. In: *Electronic Journal of Combinatorics* 14.1 (Aug. 2007). WOS:000249217900001, R61. ISSN: 1077-8926. DOI: https://doi.org/10.48550/arXiv.0706.0681. URL: https://www-webofscience-com.ru.idm.oclc.org/wos/woscc/full-record/WOS:000249217900001 (visited on 07/05/2023).

[13] Jasper Stokmans. *Quadrangulation source code*. Comp. software. July 9, 2023. URL: https://gitlab.science.ru.nl/jstokmans/quadrangulation-flip/-/tree/main (visited on 07/09/2023).