Monte Carlo Simulations of Random Surfaces

with Extrinsic Curvature

by

Leping Han

DISSERTATION

Submitted in partial fulfillment of the requirements for the Doctor of Philosophy in

Physics in the Graduate School of Syracuse University

December, 1994

Approved _____

Geoffrey Fox

Date _____

Abstract

Random surfaces are of great interest as a model of gravity, but also provide the basis for an exploration of membranes. Many random surface models have been proposed to describe fluctuating fluid membranes. These models have been extensively studied both analytically and numerically. We analyze numerically the critical properties of two-dimensional discretized random surfaces with extrinsic curvature embedded in a three-dimensional space. We measure a variety of local observables and use a finite size scaling analysis to characterize as much as possible the regime of crossover from crumpled to smooth surfaces. We exclude some possible explanations for the observed cross-over behavior and provide more questions for further study. The computational issues and technical implementations of Dynamically Triangulated Random Surfaces on computers, including parallel computers, are discussed in detail.

To my parents

Vita

NAME OF AUTHOR:

Leping Han

DATE AND PLACE OF BIRTH:

September 17, 1965, in Beijing, China

DEGREES AWARDED:

Master in Computer Engineering, 1993, Syracuse University

B.S. in Physics, 1987, Beijing University

Contents

List of Figures

Acknowledgements

Studying physics at Syracuse University has been a wonderful and memorable experience. I would like to extent my genuine gratitude to everyone involved without much hope to do justice to all of them in this small note.

I thank my advisor Geoffrey Fox, for his encouragement and guidance as I pursued research that combined physics and computational science. He not only provided me with a precious environment that allowed me to completely focus on research, but also with his insights pointed out a bright future direction as a professional career for me. I also thank Paul Coddington for the guidance in computational physics that he provided on each day throughout these years. I am especially grateful to Mark Bowick and Enzo Marinari for giving me this thesis topic and pointing out my physics research directions. Thanks also go to David Edelsohn, who taught me about UNIX and the C language, and Paul Souder, who helped me write my first Monte Carlo simulation program used in a Muonian experiment at Los Alamos National Lab. I also enjoyed fruitful discussions with Allen Su, John Apostolakis and Marco Falcioni.

I thank the Physics Department of Syracuse University for providing the opportunity for my graduate education and Northeast Parallel Architecture Center for superior computer resources.

Finally, I thank Lejing Han and Hurang Hu for their constant help. I am greatly indebted to my wife, Min Lu for her unending support. It is she who makes this endeavor possible.

Leping Han

Syracuse, January 24, 1995

1 Introduction

In the twentieth century, two different formalisms have emerged which can in principle, explain all known physical phenomena: general relativity, which gives us a compelling description of the large scale structure of the universe, and the quantum theory, which has unified the theory of sub-atomic particles. It seems that nature at its most fundamental level, would have two sets of laws, based on two sets of principles, two sets of equations, and operating on two different scales. It seems hard to believe that nature could create a world split in such an artificial fashion. Therefore, to find a combined quantum theory of gravity is a major goal in physics today.

General relativity was historically derived from geometry. First came Einstein's equivalence principle and the geometrical picture of general covariance. From the geometry came a unique action, the Einstein-Hilbert action. Then came the classical theory of curved manifolds. Last comes the yet unfinished quantum theory of gravitation. Thus, the historical sequence can be viewed as follows:

$$Geometry \rightarrow Action \rightarrow Classical\ Theory \rightarrow Quantum\ Theory$$

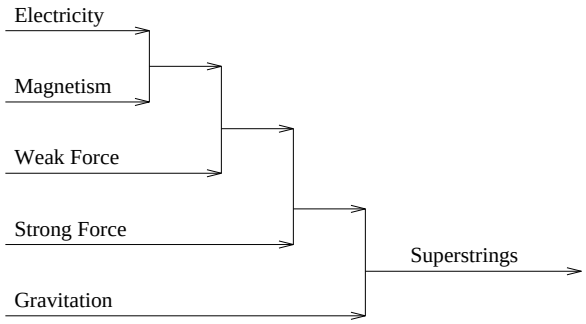

Figure 1: Superstring is a possible candidate that can unite gravity with rest of the particle forces

In late 70s and early 80s, string theory has emerged as the leading candidate for a theory of all known forces, including a finite theory of quantum gravity (Fig. 1). However, string theory has its own difficulties. The most striking feature of quantum string models is perhaps their prediction of a critical space-time dimension outside of which the quantum theory is problematical. For one of the most popular models, the bosonic model, the critical space-time dimension is 26.

The bad feature of the bosonic model is the presence of a tachyon in the spectrum, which violates causality. It is natural to seek super-symmetric models with an equal number of bosonic and fermionic variables. It is indeed a general rule of super-symmetry that super-symmetric multiplets tend to cancel unwanted effects, which is related to the zero point energy. Soon after that, Neveu and Schwarz found another bosonic model realizing such a symmetry [1]. But this model is consistent (apart from the tachyons) still only in specific critical dimensions of space-time, i.e. $d = 10$. It is still far away from our physical space-time 4 dimensional space.

People are more interested in a string world-sheet with the standard action embedded in a physically meaningful dimension. However, even just for dealing with 4 dimensional space, we still do not know how to define the path integral as used in quantum field theory. People then started to think of using lattice formulations. Dynamically Triangulated Random Surfaces are considered to be a good discretized version of the world-sheet. It is expected that if a phase transition is found in the discretized theory, a well-defined path-integral is obtained. Thus, the formulation in the continuum limit is found.

In my thesis, I present the results of a large-scale simulation of a Dynamically Triangulated Random Surface with extrinsic curvature embedded in three-dimensional flat

space. We measure a variety of local observables and use a finite size scaling analysis to characterize as much as possible the regime of crossover from crumpled to smooth surfaces. We also will discuss the computational approach and the challenging issues involved in the study.

All work presented here was done in a collaborative manner by the computational physics group at Syracuse University. I have been particularly devoted to developing the random surface program, optimizing the program, taking data and analyzing data and parallelizing the random surface program on different types of high performance parallel computers.

First, I want to briefly review several important fields and the relations among them. These fields are string theory, quantum field theory, statistical mechanics, quantum gravity and Monte Carlo simulation. They play a major role in modern physics. I have been very excited and challenged by the fact that the study of random surfaces is more or less related to different aspects of all these fields.

1.1 String Theory

In classical theory, a point particle is described by a vector which points from the origin to the location of the particle $x_\mu(\tau)$, where τ parameterizes the location of the point along the trajectory. If the particle is a free relativistic particle that moves in space-time, based on classical theory, its world line is time-like and also has minimal proper length.

The appropriate action is

$$S = -m \int ds \tag{1}$$

where m is the mass and ds is the path length along the world line (Fig. 2). The path

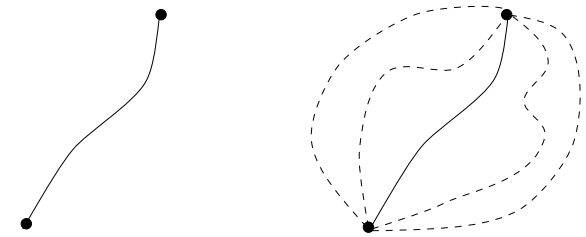

Figure 2: The essential difference between classical mechanics and quantum mechanics. The action principle is more fundamental than the equation of motion at the quantum level

length ds can be written in terms of the coordinates:

$$ds = -\sqrt{\dot{x}_\mu^2} d\tau \tag{2}$$

where the dot refers to differentiation with respect to the parameter τ.

Fig. 2 illustrates the essential difference between classical mechanics and quantum mechanics. Classical mechanics assumes that a particle executes just one path between two points based either on the equations of motion or on the minimization of the action. By contrast, quantum mechanics sums the contributions of probability function (based on action) for all possible paths between two points. Although the classical path is the one most favored, in principle all possible paths contribute to the path integral. Thus, the action principle is more fundamental than the equations of motion at the quantum level.

One of the key properties of the action is its reparametrization invariance which means the action is invariant under an arbitrary reparametrization of the variable τ, ie under a change of coordinates $\tau \to \tilde{\tau}(\tau)$.

A string is a one-dimensional extended object. The world history of the string is a

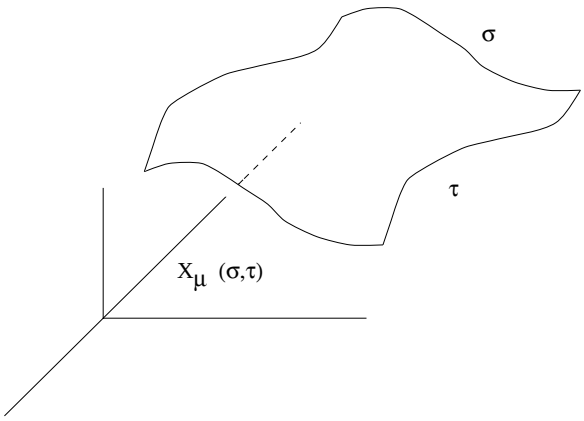

Figure 3: The two-dimensional worldsheet swept out by a string

two-dimensional surfaces in space-time. The natural generalization of the particle case is to postulate that a free string (with free boundaries if it is open) is described by a surface with the following properties:

1. The surface is time-like, i.e., it possesses everywhere time-like and space-like directions (except possibly at the boundaries).

2. It has extreme area, i.e., it is an "extremum surface."

A string is described by a vector which points from the origin to its location in space-time $X_\mu(\sigma, \tau)$. Strings sweep out two-dimensional surfaces as they move in space-time (Fig. 3).

Let $\mathcal{M}$ be a fixed two dimensional manifold. Surfaces with the topology of $\mathcal{M}$ are described by maps [2]

$$X : \mathcal{M} \to \mathbf{R}^d. \tag{3}$$

The parametric equations of the string world surface are

$$X^\mu = X^\mu(x^\alpha) \tag{4}$$

where $\mu = 0, 1, ..., d-1$ and $\alpha = 0, 1$. We assume that $x^\alpha \equiv (\tau, \sigma)$ provides a good

parameterization.

One possible formulation for the action of string as the area of the world sheet was introduced by Nambu, Goto and Hara. The area A of the surface swept out by the string is given by

$$A = \int_M d^2x \sqrt{|\det h|} \tag{5}$$

where

$$h_{ij} = \frac{\partial X^\mu}{\partial x^i} \frac{\partial X^\mu}{\partial x^j} \tag{6}$$

is the induced metric in the surface embedded in d-dimensional space

$$ds^2 = dX^\mu dX^\mu = \frac{\partial X^\mu}{\partial x^i} \frac{\partial X^\mu}{\partial x^j} dx^i dx^j \tag{7}$$

The action of the free string is proportional to the area and hence given by the Nambu-Goto action [3]

$$S\left[X^\mu(x^\alpha)\right] = -\frac{1}{2\pi\alpha'} \int_{\tau_1}^{\tau_2} d\tau \int_0^{\pi(open) or 2\pi(closed)} d\sigma \sqrt{|\det h|} \tag{8}$$

where α' is a constant with dimension of length-squared in units of $\hbar$.

The above action (8) has one drawback: it is not quadratic in the fields. This is a serious problem if one wants to quantize the string by means of path-integral methods, for the usual Lagrangian form of the path integral is only valid for quadratic actions.

Polyakov [4] proposed that in the quantum theory one replace the area action by an equivalent classical action that depends on an additional intrinsic metric g_{ij} (continuum model). The new action is a functional of both the intrinsic metric g_{ij} and the coordinates X with the quadratic form:

$$S[X, g] = \int_M d^2x \frac{1}{2} \sqrt{|\det g|} g^{ij} \frac{\partial X^\mu}{\partial x^i} \frac{\partial X^\mu}{\partial x^j}. \tag{9}$$

Surprisingly, almost all of the main features of the Nambu-Goto string have some forms of analogue in the first quantized point particle theory. Many basic methods of quantization can be carried over directly from the point particle case [3].

Classically, one can eliminates g_{ij} from the equation (9) algebraically and substituting this solution back to the action. The resulting expression is the Nambu-Goto action, namely, the area of the world sheet (definitely a reparametrization-invariant expression). Quantum mechanically the elimination of g involves performing a path integral [3]. In general an extra "Liouville mode" is left over (the Weyl invariance is broken) except in the special case of d = 26. There are many papers in the literature [2, 3, 5, 6] showing the ways to derive the critical dimension, i.e., $d = 26$.

1.2 Quantum Field Theory

Any quantum theory has to be understood on the level of quantum field theory. So does string theory. We can think of string theory as a field theory by direct analogy with ordinary quantum field theory. The basic object of ordinary quantum field theory is a field $\phi(x^\mu)$ where x^μ is a coordinate on space-time. The dynamics of this field can be described by an action $S[\phi]$. Given $S[\phi]$ one may then compute, e.g., the amplitude $A(x^\mu, y^\mu)$ for a particle observed at x^μ to appear at y^μ via functional integration:

$$A(x^\mu, y^\mu) = \int \mathcal{D}\phi^{iS[\phi]}\phi(x^\mu)\phi(y^\mu) \tag{10}$$

In string theory, particles are replaced by strings. While the location of a particle is described by a single coordinate x^μ, the location of a string is described by a function $x^\mu(\sigma)$, where σ is a parameter along the string. One then wishes to compute. e.g., the amplitude $A[x^\mu(\sigma), y^\mu(\sigma)]$ for a string observed at $x^\mu(\sigma)$ to appear at $y^\mu(\sigma)$. This re-

quires a string field $\phi[x^\mu(\sigma)]$ whose domain is the space of all configurations $x^\mu(\sigma)$. Given some action $S[\phi]$, the amplitude may then be computed just as in ordinary quantum field theory:

$$A[x^\mu(\sigma), y^\mu(\sigma)] = \int \mathcal{D}\phi e^{iS[\phi]}\phi[x^\mu(\sigma)]\phi[y^\mu(\sigma)] \tag{11}$$

In summary, string field theory is quantum field theory with space-time replaced by string space; i.e., the space of all string configurations.

But the key question again in string field theory is to find a suitable action $S[\phi]$. To construct the action, what kinds of properties should the action have?

The principle of renormalizability tells us that the action should be constructed out of covariant expressions in X^μ and g_{ij} which are of dimensions two or less and which respect the Euclidean invariance of physical space-time. The most general such action consists of three terms [2]:

$$S[X^\mu, g_{ij}] = A\frac{1}{2}\int_M d^2x\sqrt{g}g^{ij}\partial_i X^\mu \partial_j X^\mu + B\int_M d^2x\sqrt{g} + C\frac{1}{4\pi}\int_M d^2x\sqrt{g}R \tag{12}$$

The first term in the above is just the classical action we had previously discussed. The second term is a cosmological constant. The third term is a topological invariant called the Euler characteristic χ:

$$\chi(M) = \frac{1}{4\pi}\int_M d^2x\sqrt{g}R. \tag{13}$$

The Euler characteristic $\chi(M)$ is related to the topology of the manifold via the relation

$$\chi(M) = 2 - 2h - b, \tag{14}$$

where h is the number of handles on the surface and b the number of boundaries.

1.3 Field Theory And Statistical Mechanics

The connection between field theory and statistical mechanics is best understood using the Feynman path integral approach [7].

A standard computation in quantum mechanics is the probability amplitude for a transition of a point particle from the initial state q_i at time t_i to the final state q_f at the time t_f. It is called a propagator, and can be expressed in the Heisenberg picture as

$$K(q_f t_f; q_i t_i) = \langle q_f t_f | q_i t_i \rangle \tag{15}$$

The Feynman formulation of quantum mechanics states that

$$K(q_f t_f; q_i t_i) = \sum_{all\ possible\ paths} e^{iS/\hbar} \tag{16}$$

where S is the classical action for a path $x(t)$ connecting the initial point to the final point (Fig. 2). The sum over all possible paths is generally written as a "path integral" (or functional integral)

$$K(q_f t_f; q_i t_i) = \int [\mathcal{D}x(t)] e^{iS/\hbar} \tag{17}$$

In the case of quantum field theory, the fundamental objects are not paths but fields $\Phi(x)$, and the propagators are vacuum expectation values of time ordered products of the fields. In the Feynman formulation they are expressed as

$$\langle \Phi(x_1)\Phi(x_2) \rangle \equiv \langle 0|T[\Phi(x_1)\Phi(x_2)]|0\rangle = \int [\mathcal{D}\Phi(x)]\Phi(x_1)\Phi(x_2)e^{iS/\hbar} \tag{18}$$

where the action is the space-time integral of the Lagrangian

$$S(\Phi(x)) = \int d^4x \mathcal{L}(\Phi(x)) \tag{19}$$

The functional integral represents the sum over all possible field configurations $\Phi(x)$. The vacuum expectation values of the above equation (18) can be obtained from a

"vacuum generating functional"

$$Z = \int [\mathcal{D}\Phi(x)] e^{iS/\hbar} \tag{20}$$

in a similar manner to the calculation of correlation functions and other thermodynamic variables from the partition function

$$Z = \sum_s e^{-H(s)/kT} \tag{21}$$

in statistical mechanics. Here H is the Hamiltonian, kT is the Boltzmann constant times the temperature, and the sum runs over all possible states of the system.

We wish to convert Z in equation (20) to a form similar to the partition function in statistical mechanics. We change variables to imaginary time (called the Euclidean time in quantum field theory), i.e. perform a Wick rotation $t \rightarrow -it$, the Euclidean action $S_E = -iS$ and the function Z can be expressed as

$$Z = \int [\mathcal{D}\Phi(x)] e^{-S_E/\hbar} \tag{22}$$

If we identify $S_E/\hbar$ with H/kT, the partition function of statistical mechanics and the generating functional of Euclidean quantum field theory are formally identical. The result provides the link between quantum field theory and statistical mechanics. The $\hbar$ appears here because we are dealing with quantum mechanics system. In classical mechanics the actual path of the particle is determined by finding an extremum for the action S. In quantum mechanics all paths contribute to the probability amplitude with a weight $e^{iS/\hbar}$. And $\hbar$ sets the scale of quantum fluctuations just as the temperature sets the scale for thermal fluctuations. Correlation functions in statistical mechanics are equivalent to Euclidean space propagators in field theory.

The equation (22) is purely formal until a workable definition is given to the functional integral. For example, a free particle's path integral reads

$$\Sigma \rightarrow \int \mathcal{D}x = \lim_{N \rightarrow \infty} \int \prod_{i=1}^{3} \prod_{n=1}^{N} dx_{i,n} \tag{23}$$

where the index n labels N intermediate points that divide the interval between the initial and the final coordinates. We will take the limit when N approaches infinity. We must also be able to regularize all the usual divergences. The lattice approach solves both these problems. On a finite lattice, the functional integral is approximated by a well defined finite dimensional integral, and (23) is obtained by taking the continuum limit, in which the lattice spacing goes to zero while the lattice volume tends to infinity. It can be shown [8] that the correlation length ξ of a lattice field theory is related to the mass gap m by

$$\xi \sim m^{-1} \tag{24}$$

Having properly defined the Euclidean space generating functional and seen that it is mathematically equivalent to a partition function, powerful techniques from statistical mechanics such as low and high temperature expansions, mean field theory, renormalization group analysis and Monte Carlo simulation may be applied to the study of quantum field theory. Such techniques are especially useful when the theory is discretized onto a lattice.

Over the past 60 years, many predictions of the Standard Model of elementary particle interactions, including predictions from QED and QCD, have been confirmed to a high degree of accuracy by accelerator experiments. Exact solutions of the QFT equations do not exist at this time. In most cases the predictions of the standard model have been obtained using perturbation theory because electro-magnetic, weak,

and strong interactions at high energies involve small values of the coupling constant. Small values are especially applicable for scattering experiments involving electrons or protons at very high energies of 100 Gev and more. However, other phenomena such as nuclear structure, proton structure, and nuclear matter occur at high coupling and therefore prohibit the use of perturbation theory. One of the most important challenges facing quantum field theorists today is to solve QFT in the nonperturbative regime. A potential solution to this challenge was found in the late 1970s and early 90s, with the development of a formalism called lattice field theory (LFT). In its most useful formulation, LFT makes much use of numerical techniques, particularly Monte Carlo methods familiar in condensed-matter physics. The application of LFT to the strong interactions has required hundreds and sometimes thousands of hours of supercomputer time. LFT has helped spur the development of supercomputers – especially parallel architectures, as LFT requires only local interactions.

It is hoped that the experience learned from LFT can be applied to the study of string theory or even more directly to random surfaces. Therefore, in the context of random surfaces, we want to use the path integral approach to quantize strings with the partition function is given by

$$Z = \sum_{surfaces} exp(-area). \tag{25}$$

where one is supposed to sum over surfaces with all topologies.

1.4 Quantum Gravity

As we mentioned above, one of the key motivations of studying string theory is to find a theory that contains quantum gravity. Quantum Field Theory and the theory of

General Relativity are, separately, probably the two most successful physical theories of this century. Nobody has yet been able to bring the two together into one complete and consistent quantum theory of gravity.

Thus, the construction of a theory of quantum gravity which encompasses both General Relativity(GR), or some of its extensions, and the principles of Quantum Mechanics is one of the most difficult and exciting problems of theoretical physics. It is well-known that there are at least two basic difficulties. Firstly, one major impediment to such a theory is that, unlike gauge field theories, gravity with the Einstein-Hilbert action

$$S = \frac{1}{16\pi G_N}\int d^4x\sqrt{g}R \tag{26}$$

is not renormalizable, at least not by the usual methods of perturbation theory. Secondly, the Einstein equations (the evolution equations for the metric) are derived from an action principle, but the Einstein-Hilbert action is unbounded from below, and a naive path integral formulation of a quantum theory is expected to be mathematically ill defined. Among a list of attempts to define quantum gravity including canonical quantum gravity, standard field theory using perturbation theory, quantum field theory in curved space time, the path integral approach to Euclidean quantum gravity, quantum cosmology, discrete models of quantum gravity, and topological field theory [9, 10]. There is not a single proposal for a quantum theory of gravity that is self-consistent as a physical theory, even allowing for a great deal of incompleteness. If one examines the theories closely, they are not flawless to begin with.

Although quantum field theory and general relativity seem totally incompatible, the past two decades of intense theoretical research have made it increasingly clear that the secret to this mystery most lies in the power of gauge symmetry. At present, the

most promising hope for a truly unified and finite description of these two fundamental theories is the superstring theory (Fig. 1). Superstrings possess by far the largest set of gauge symmetries ever found in physics. This has lead a number of physicists to adopt the position that General Relativity is only the low energy limit of some other quantum theory, such as superstring theory. To pursue a quantum theory of gravity, one needs a nonperturbative method of calculation: the methods of lattice field theory, which have already been applied to gauge theories, are immediately suggested.

In the lattice formulations of quantum gravity, one tries to construct a discretization procedure which makes the functional integration over metrics (or over the other degrees of freedom which describe classical gravity) meaningful, by truncating this infinite dimensional integral into a sum over a finite number of variables. Then one should look for critical values of the parameters (the coupling constants) of the discretized theory where a scaling behavior occurs, so that the details of the discretization become irrelevant and a continuum theory can be constructed. This lattice approach has been very successful for studying ordinary relativistic quantum field theories, such as non-Abelian gauge theories. When trying to apply these ideas to General Relativity one hopes that a quantum theory can be obtained from a path integral formulation based on the Lagrangian formulation of General Relativity, namely from the Einstein-Hilbert action, which is a functional of the metric tensor g, taken as the dynamical variable.

Therefore, we are lead to consider the functional integral over the Euclidean metric

$$\int D[g] e^{S_E[g]} \tag{27}$$

with the Euclidean EinsteinHilbert action

$$S_E[g] = \frac{1}{16\pi G_N} \int d^D x \sqrt{|g|}(-R + \Lambda) \tag{28}$$

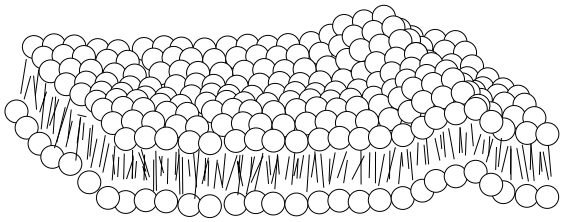

Figure 4: A biological membrane composed of a lipid bilayer: two layers of amphiphilic molecules with polar hydrophilic heads and long hydrophobic hydrocarbon tails.

where D is the dimension of space-time, G_N Newton's constant and Λ the cosmological constant.

1.5 Random Surfaces

More recently the connection between random surfaces, quantum gravity and string theory has been the subject of many investigations. In statistical mechanics and condensed matter physics problems involving interfaces (for instance wetting) or two dimensional defects (for instance charge density waves), as well as two dimensional films or membranes [11], such as monolayer of surfactant molecules at an oil-water interface in a microemulsion, cell membranes and liquid membranes as shown in (Fig. 4), are naturally formulated in terms of fluctuating surfaces.

In string theory, the worldsheet must fluctuate as one is required to integrate over all admissible metrics to enforce diffeomorphism (reparametrization) invariance. In this way new intrinsic degrees of freedom (the conformal modes of the metric) enter the theory. From the statistical mechanics viewpoint one is thus dealing with an exciting class of models described by certain order fields living on a fluctuating substrate. Averaging over metrics corresponds to being in the universality class of translationally and orienta-

tionally disordered fluctuating surfaces or membranes. These are often called liquid-like membranes, as opposed to crystalline or hexatic membranes that are translationally or orientationally ordered respectively [12]. The remarkable fact is that these statistical mechanical models defined on a random mesh are, in a sense, easier to solve than the conventional models defined on a rigid regular lattice. This is because diffeomorphism invariance reduces the number of effective degrees of freedom. It is even possible to admit fluctuations which change the topology of the surface (growth or collapse of handles).

This is certainly of great interest as a model of gravity but also provides the basis for an exploration of membranes. Recently, particularly simple models corresponding to certain types of conformal matter coupled to $2d$-gravity have been exactly solved including the sum over all possible topologies [13, 14, 15]. The idea for the theory of surfaces embedded in $D \leq 1$, advocated by various authors [16, 17] that random lattices may provide a sensible discretization of two dimensional Euclidean quantum gravity was confirmed quantitatively. Exact results for the continuum theory, obtained by conformal fields theory methods, are in agreement with exact solutions for the discretized models.

Hoping to understand more in other than dimension $D \leq 1$, one wants to perform the functional integral by defining the integral on a lattice. We know that if there is a phase transition at a certain point, we can say that this point is the fixed point, and we can make a continuum limit around this point, since the details of the discretization is not important.

In the search for a discretized version of a bosonic string theory, the model of triangulated random surfaces with Gaussian action was proposed in 1985 [16]. It was hoped that this formulation could be used to understand the properties of a bosonic string in sub-critical dimensions $d < 26$. Investigations carried out over the past few

years seem, however, to indicate that for any $d > 1$, it degenerates into a theory of branched polymers [18, 19, 22, 20], Lattice approach also show the surfaces are crumple [23, 24, 25, 26].

To overcome this problem it was suggested to add to the action a new term, proportional to the extrinsic curvature [23, 24, 25, 26], which would smoothen the surface and possibly avoid the collapse into a branched polymer. The importance of the extrinsic curvature for the description of surfaces was pointed out many years ago in condensed matter and biophysics [27, 47]. On a triangulated surface various forms of this term have been investigated [28].

In spite of many new ideas and exciting development in random surfaces, we have to admit that this approach has not established a very clear relationship to the continuum theory. Here we are concerned with an attempt to regularize the formal Euclidean path integral for quantum gravity by a discretization of the integration over the space of metrics. Such a regularization may lead to a well defined theory. However it may be completely unrelated to the "correct" quantum theory of gravity in 4 dimensional real space. The relevant question to ask is therefore whether the such a regularized path integral leads to reasonable physical statements. A much more modest goal is to understand the statistical model resulting from such a discretization, and this is what we want to study, since random surface models have a range of applications in physics and many other fields.

1.6 Monte Carlo Simulation

In the previous section we discussed briefly the Monte Carlo method which is widely used to perform computer simulations of path integrals or statistical systems. We now

present a detailed review of the Monte Carlo algorithm in the this section.

In statistical mechanics, generally we use two approaches, analytic methods and Monte Carlo methods. Analytic methods are almost always approximations. There are very few exact solutions. We know they are good for understanding basic physics, but often break down in regions of interest such as phase transitions and we have little knowledge or control of errors, which are not easily systematically improvable.

Monte Carlo calculations usually involve sums or integrals over very large number of dimensions or configurations. We can compute quantities for any parameter values (including where analytic methods don't work). Errors can be estimated, and are systematically improvable by using more sample configurations and larger systems (i.e. more computer power!).

A typical example of the use of Monte Carlo method is for performing integrals. Instead of choosing x_i at regular intervals in the discretized integral (29), just choose them at random. The error is purely statistical, $\propto 1/\sqrt{N}$ *independent of the dimension of the integral.* Other numerical integration methods have error $\propto 1/N^{1/d}$ [29].

$$I = \int_a^b f(x)dx \approx \frac{(b-a)}{N} \sum_{i=1}^{N} f(x_i) \tag{29}$$

When using Monte Carlo simulation in physical system, we want to use computer to generate possible configurations of the system and measure averages of physically measurable quantities. The measurement average of a physical observable f is

$$\langle f \rangle = \sum_C p(C) f(C) \tag{30}$$

where $f(C)$ is the value of observable f for the configuration C, $p(C)$ is the probability distribution for the configuration as a function of system parameters, and the sum is over all configurations.

A fundamental result in statistical mechanics is that p is the Boltzmann distribution, which is is defined by

$$p(C) = \frac{1}{Z} e^{-S_E} \tag{31}$$

where S_E is the Hamiltonian of the configuration and Z is called partition function and defined by

$$Z = \sum_C e^{-S_E} \tag{32}$$

The partition function is a fundamental quantity in statistical mechanics. All quantities of interest can be extracted from Z, so an analytic formula for Z implies an exact solution of the model. The sum will actually be an integral if the space of possible configurations is continuous rather than discrete.

To calculate sums, we resort to Monte Carlo techniques. As with the Monte Carlo integration mentioned earlier, we could just generate configurations at random, and approximate the measurement of the observable by Monte Carlo averages.

The problem here is that because of the rapidly varying exponential function in the Boltzmann distribution, most randomly chosen configurations will make a negligible contribution to the sum, since S_E will be relatively large. One can estimate that if we want to calculate the exact partition function Z numerically for Ising model to solve for a 32×32 lattice, we would run this calculation for the age of the universe even using the fastest computer.

In order to get sensible, accurate results when simulating statistical systems with a rapidly varying Boltzmann distribution, it is vital to use the idea of importance sampling in Monte Carlo integration.

Clearly the ideal situation would be to sample configurations with a probability given

by their Boltzmann weight $p(C)$. Then the Monte Carlo average for f would just be:

$$\overline{f} = \frac{1}{N}\sum_{i=1}^{N} f(C_i)$$

The next question is how to construct the sampling probability distribution $p(C_i)$.

Let us set up a so-called *Markov chain* of configurations C_t by the introduction of a fictitious dynamics. The "time" t is computer time (marking the number of iterations of the procedure), NOT real time – our statistical system is considered to be in equilibrium, and thus time invariant.

Let $P(A,t)$ be the probability of being in configuration A at time t.

Let $W(A \to B)$ be the probability per unit time, or *transition probability*, of going from A to B. Then:

$$P(A,t+1) = P(A,t) + \sum_B [W(B \to A)P(B,t) - W(A \to B)P(A,t)] \qquad (33)$$

At large t, once the arbitrary initial configuration is "forgotten," we want $P(A,t) \to p(A)$.

For an equilibrium (time independent) probability distribution clearly a sufficient (but not necessary) condition is the so-called detailed balance condition

$$W(A \to B)P(A,t) = W(B \to A)P(B,t) \qquad (34)$$

This method can be used for any probability distribution, but if we choose the Boltzmann distribution

$$\frac{W(A \to B)}{W(B \to A)} = \frac{p(B)}{p(A)} = \frac{e^{-S_E(B)}}{e^{-S_E(A)}} = e^{-\Delta S_E} \qquad (35)$$

where

$$\Delta S_E = S_E(B) - S_E(A) \qquad (36)$$

Note that Z does not appear in this expression. We can calculate the S_E based on the system Hamiltonian.

This dynamic method of generating an arbitrary probability distribution was invented by Metropolis *et al* [30]. There are many possible choices of the W's which will satisfy detailed balance. They chose a very simple one:

$$\begin{aligned} W(A \to B) &= e^{-\Delta E/kT} && \text{if } \Delta E > 0 \\ &= 1 && \text{if } \Delta E \leq 0 \end{aligned}$$

So, if $E(B) > E(A)$

$$\frac{W(A \to B)}{W(B \to A)} = \frac{e^{-[E(B)-E(A)]/kT}}{1} = e^{-\Delta E/kT}$$

and if $E(B) \leq E(A)$

$$\frac{W(A \to B)}{W(B \to A)} = \frac{1}{e^{-[E(A)-E(B)]/kT}} = e^{-\Delta E/kT}$$

So we have a valid Monte Carlo algorithm if:

- We have a means of generating a new configuration B from a previous configuration A such that the transition probability $W(A \to B)$ satisfies detailed balance
- The generation procedure is *ergodic*, i.e. every configuration can be reached from every other configuration in a finite number of iterations

The Metropolis algorithm satisfies the first criterion for all statistical systems. The second criterion is model dependent, and not always true (e.g. for a system at $T = 0$).

Note that the Metropolis algorithm does not specify *how* the changes to the configuration should be made — it just says that any *proposed* change to the system should be *accepted* with a certain probability that depends on the change in energy.

How the changes are made depends on the variables and the model being studied. The only constraints on the update procedure are the two given above, ie, it should be

ergodic and it should not be biased in such a way as to violate detailed balance. Another issue is efficiency — the procedure should sample the configuration space as effectively as possible. There is often some freedom in tuning the algorithm to improve efficiency and performance.

The Monte Carlo evaluation of the path-integral (22) in Dynamically triangulated random surfaces (DTRS) is to construct a Markov chain $\{T_i\}$ of triangulations with equilibrium configuration $\exp(-S_E)$ so that

$$\langle f \rangle = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} f(T_i). \tag{37}$$

Of interest is the phase diagram obtained for different coupling constant.

When studying any macroscopic system with a very large number of degrees of freedom, one invariably makes an approximation and simulate a smaller and/or discretized model system. This introduces systematic errors called *finite size effects*. To overcome this, usually one does a number of simulations at different system sizes and then extrapolates to an infinite system.

For a finite system, one would get rounded peaks rather than divergences. The peaks narrow and increase in height as the lattice size is increased, and the location of the peak shifts slightly [31].

Many problems require an empirical extrapolation to an infinite system. In some instances, such as at second order phase transitions, the form of the asymptotic (large lattice size L) behavior is known. I do not want to discuss these relationships in detail. However, these relations are extremely useful, and even allow us to extract exponents and the critical coupling value at infinite lattice size. However this method works only if we are at "large enough" L. If ξ is very large (but not infinite), may get incorrect

"pseudo-critical" exponents.

This is often the case at a "weakly" first order transition (very small latent heat). So it is often very difficult to distinguish between a first and second order phase transition, or even to say for sure if there is a phase transition at all — is $\xi = \infty$, or just $\xi \gg L$? In this case we may need extremely large system sizes to get correct results.

In general, one doesn't know the form of the finite size scaling. Usually One would try to fit data to a *scaling function*, or go to large enough systems so that results approach a constant (ie, are independent of lattice size) or one can do a simple linear extrapolation.

Although Monte Carlo simulation (using the Metropolis algorithm, for example) appears very straightforward, there are in fact many problems and subtleties that can trap the unwary and produce unreliable results [31, 32]. The simulations and the interpretation of the results must be done with great care.

2 Random Surfaces

String theory has been conjectured to describe the underlying fundamental physics of a wide variety of physical phenomena and models. In its simplest form, the bosonic string, it is a theory of free fluctuating surfaces. The functional integral for the Euclideanized bosonic string is just the partition function for an ensemble of random fluctuating fluid surfaces. Such surfaces are also ubiquitous in nature, being found for example in macro-emulsions and the lipid bilayers that form an important part of cell membranes [57]. These systems are fluid because their component 'molecules' are loosely bound. Their constituents are arranged so that the net surface tension (nearly) vanishes; thus these membranes are subject to large thermal fluctuations. In one important respect, however, these chemical/biological membranes differ fundamentally from the surfaces we discuss and simulate; they are self-avoiding. The world-sheets of the bosonic string, in contrast, generically self-intersect.

The theory of $2d$ fluid random surfaces embedded in R^3, with an extrinsic curvature term (bending rigidity) in the action, has received considerable analytical and numerical attention in the last decade [57, 42, 59]. This is the action that has been primarily investigated throughout the study of random surfaces presented here.

2.1 2D Gravity and Random Surfaces

2.1.1 The Motivation for Random Surfaces – Continuum Model

Let us start by considering only a fluctuating 2d-surface with no matter (order fields) at all. Since there are no embedding string coordinates it is also a model of strings in zero dimensions. The Einstein-Hilbert action with a cosmological constant term for $2d$

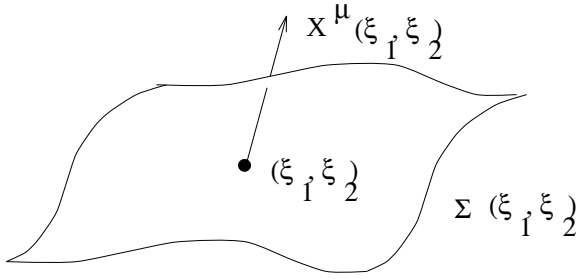

Figure 5: String worldsheet with intrinsic coordinates ξ_1 and ξ_2.

gravity is [37]

$$S[g] = \frac{-1}{16\pi G}\int_\Sigma d^2\xi\sqrt{g}R + \mu\int_\Sigma d^2\xi\sqrt{g} \tag{38}$$

where $g_{\alpha\beta}(\xi_1,\xi_2)$ is the $2d$ metric of the Riemann surface Σ with coordinates ξ_1 and ξ_2 (Fig. 5). These strings sweep out two-dimensional Riemann surfaces as they evolve in Euclidean time. In the first quantized description of string theory one may view the string coordinates describing the embedding of the worldsheet in the target space-time as a collection of scalar fields living on the worldsheet (Fig. 6).

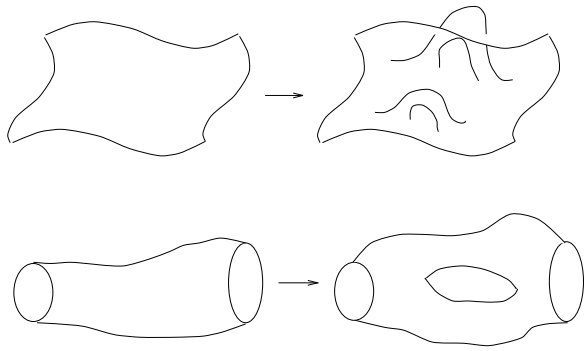

Figure 6: Fluctuations in the topology of worldsheet

The partition function Z then depends on two variables, Newton's constant G and the cosmological constant μ

$$Z[G,\mu] = \int [\mathcal{D}g]\ e^{-S[g]} \tag{39}$$

where the path integral is over all admissible metrics of Riemann surfaces Σ. In two dimensions the action (38) is simple since the first term is a topological invariant by the Gauss-Bonnet theorem

$$S = \frac{-\chi(\Sigma)}{4G} + \mu A(\Sigma) \tag{40}$$

where χ is the Euler characteristic of Σ and A is the area. χ is related to the number of handles, or genus h, by $\chi = 2 - 2h$, where for simplicity we are assuming Σ to be closed (without boundaries). The partition function thus reduces to

$$Z[G, \mu] = \sum_h \int dA e^{\frac{\chi}{4G}} e^{-\mu A} \Omega_h(A) \tag{41}$$

where $\Omega_h(A)$ is the density of states of Riemann surfaces Σ of fixed area A and genus h,

$$\Omega_h(A) \equiv \int_{(h;A)} \mathcal{D}g_{\alpha\beta} \tag{42}$$

$\Omega_h(A)$ is very difficult to calculate as h increases and the sum over genus in (41) diverges [60].

The above expressions are all, in fact, ill-defined. To give them meaning we must regularize the path integrals.

2.1.2 The Discretization of Random Surfaces – Dynamically Triangulated Random Surfaces

We are concerned primarily with the Polyakov form of the string action [4], in which an additional *intrinsic metric* g_{ij} is introduced to describe the surface geometry since this form of action would lead to the discretization of the surfaces more naturally.

One approach is to discretize by replacing Σ by a lattice. A particularly concrete and appealing discretization is to consider all triangulations (or more generally cellular decompositions) of Σ. The surface is thus replaced by a discrete set of n points (vertices)

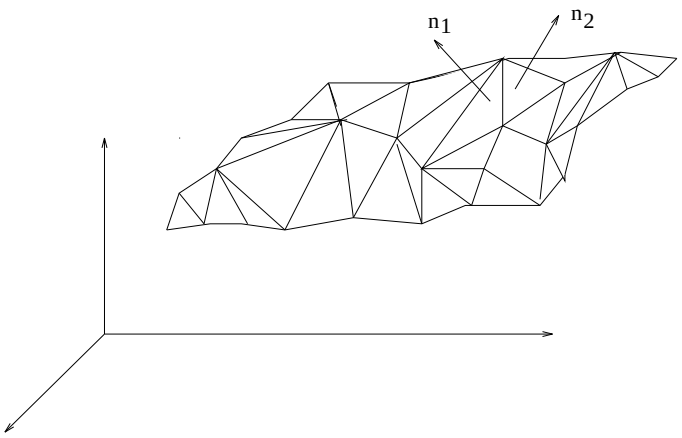

Figure 7: Surface discretized into triangulations each characterized by a unit normal

labelled by an index i. The connectivity of the lattice is described by the adjacency matrix

$$C_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are connected by a link} \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

This defines a metric on the lattice by fixing all links to have length one in the intrinsic metric. Thus all triangles (cells) in the triangulation are equilateral and of fixed area.

In this case, the coordination number at each vertex determines the intrinsic curvature of the surface. The coordinates i label the vertices of the triangulation. Then the discrete analogue of the intrinsic metric is the adjacency matrix C_{ij} whose elements equal 1 if i and j label neighboring nodes of the triangulation, and vanish otherwise. Two-dimensional diffeomorphism invariance reduces to the permutation symmetry of the adjacency matrix at this discrete level. One of the keys, in fact, to the power of this construction is the preservation of this symmetry. Each vertex of the triangulation is embedded in R^3 via the mapping X_i^μ (Fig. 7).

Given the embedding X, we can also associate a unit normal vector $(n^\mu)_{\hat{k}}$ with each triangle on the surface (Roman indices with hats label the triangles). Note that all of

the surface curvature of the triangulations is concentrated along the links and vertices. The surface is still flat in the direction tangent (but not transverse) to each link, so that the mean curvature has support on the links, while the Gaussian curvature is non-zero only at the vertices.

The Euler characteristic follows from Euler's relation $\chi = V - E + F$, for V vertices, E edges (links) and F faces (triangles). Local curvature (intrinsic curvature) is defined by means of the deficit angle

$$R_i = \pi \frac{(6 - q_i)}{q_i}, \tag{44}$$

where q_i denotes the connectivity of the lattice at vertex i.

$$q_i = \sum_j C_{ij} \tag{45}$$

The Gaussian curvature K on the other hand is expressed in terms of the deficit angle in the embedding space (see Chapter 3).

To simulate the integral over metrics the adjacency matrix must be allowed to fluctuate so that the coordination number of a node becomes a dynamical degree of freedom. The local environment of a node is constantly changing. This considerably complicates the study of such models from a computational point of view but also makes them more interesting. These models are called Dynamically Triangulated Random Surfaces DTRS [16]. The basic move to update C_{ij} is a flip on a fundamental parallelogram of two triangles sharing a common edge (Fig. 8).

The discrete version of the partition function (41) replaces integrals over metrics by sums over admissible triangulations and may be written in the form

$$Z[G, \mu] = \sum_{h=0}^{\infty} e^{\frac{2-2h}{4G}} \sum_{n=0}^{\infty} e^{-\mu n} Z_{h,n} \tag{46}$$

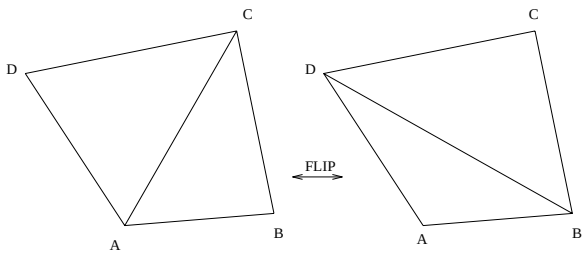

Figure 8: Edge flipping

where $Z_{h,n}$ is the number of distinct triangulations with n vertices and genus h. $Z_{h,n}$ is a discrete version of $\Omega_h(A)$, since A is proportional to the n for fixed area elementary triangles.

2.1.3 The Problem with the Model

It has been shown [13] that by including particular kinds of matter living on the surface, the string susceptibility can be expressed as

$$\gamma_h = 2 - \frac{(1-h)}{12}\left\{25 - c + \sqrt{(1-c)(25-c)}\right\} \tag{47}$$

where c is called central charge, which measures the response of the free energy to the local curvature of freedom of the model. If $c = 1$, the mean surface area

$$\langle A \rangle = -\frac{\partial \log Z}{\partial \mu} \tag{48}$$

diverges as $\frac{1}{\mu - \mu_c}$ [14], where μ_c is the critical cosmological constant.

Diverging surface area is an indication of criticality. Near μ_c one may thus construct a continuum limit with associated critical exponents that are universal in the sense that they do not depend on the fine details of the lattice.

Suppose now that we wish to describe more realistic string models corresponding to surfaces embedded in a target space of dimensionality d greater than one. The surface is

given by $x^\mu(\xi_1, \xi_2)$ ($\mu = 1, ..., d$). These models have $c > 1$ [33]. An immediate problem is then apparent from equation (47). According to the continuum results the string susceptibility is imaginary for $1 < c < 25$. This suggests that the model has an inherent instability.

Analytical and computational investigations [23, 24, 25, 26, 33] established that the continuum limit of these models is dominated by surfaces which degenerate into a branched tree of tubes of diameter of order the lattice spacing. These are called branched polymer configurations and are more one-dimensional than two-dimensional. The origin of these spikes is clear in the Nambu-Goto formulation since an infinitesimally thin long tube has vanishing area and is therefore not suppressed by the area action. The large entropy for such configurations eventually dominates the statistical mechanics of these surfaces. The Polyakov action has been shown to be in the same universality class.

2.2 Extrinsic Curvature

2.2.1 The Construction of Extrinsic Curvature

Since the tubes of the branched polymers have a high extrinsic curvature it is possible to suppress these (and other irregular) configurations by adding a bending rigidity term to the action [18, 21, 34].

To write down the action, we introduce an explicit parametrization of a generic surface $\mathcal{M}$ in R^3 with coordinates (σ_1, σ_2) and the embedding $X^\mu(\sigma_i)$. μ runs from 1 to 3, since we only study the case of a $3d$ embedding space (see Fig. 9). The induced metric (the pullback of the Euclidean R^3 metric via the embedding) is given by

$$h_{ij} = \partial_{\sigma_i} X^\mu \partial_{\sigma_j} X_\mu \ . \tag{49}$$

Figure 9: Surface coordinates (σ_1, σ_2) and embedding coordinates $X^\mu(\sigma_i)$

We will use Greek letters for the embedding space indices; they can be raised and lowered at will since our background space is flat. Associated with each point in $\mathcal{M}$ are tangent vectors t_i^μ and a normal vector n^μ (see Fig. 10). The extrinsic curvature matrix K_{ij} can be defined by

$$\partial_i n^\mu = -K_{ij} t^{\mu\, j} \ . \tag{50}$$

The eigenvalues of this matrix are the inverses of the radii of curvature of $\mathcal{M}$. In three dimensions the extrinsic matrix K_{ij} writes

$$K_{ij} = \begin{pmatrix} 1/r_1 & 0 \\ 0 & 1/r_2 \end{pmatrix} \tag{51}$$

Figure 10: The tangent directions and normal direction at point P.

where r_i are the principal radii of curvature associated with the two principal directions at every point on the surface (see Fig. 11).

One usually describes the geometry of these surfaces in terms of the mean curvature [35, 36]

$$H = \frac{1}{2} h^{ij} K_{ij} \ , \tag{52}$$

and the Gaussian curvature

$$K = \epsilon^{ik} \epsilon^{jl} K_{ij} K_{kl} \ . \tag{53}$$

where ϵ is the totally antisymmetric matrix, i.e., $\epsilon_{11} = \epsilon_{22} = 0$ and $\epsilon_{21} = -\epsilon_{12} = 1$.

One can show that the Gaussian curvature can be computed solely from the metric h_{ij}, while the mean curvature depends explicitly on the embedding X^μ. In three

Figure 11: The two principal directions 1 and 2 at a saddle point

dimensions the mean curvature is

$$H = \frac{1}{2}\mathrm{Tr}(K_{ij}) = \frac{1}{r_1} + \frac{1}{r_2} \tag{54}$$

where r_i are the principal radii of curvature of the surface, and the Gauss curvature is

$$K = det(K_{ij}) = \frac{1}{r_1 r_2} \tag{55}$$

The extrinsic curvature action is defined as

$$S_{EC} = \lambda \int d^2\xi \sqrt{h} (\mathrm{Tr}K)^2 \tag{56}$$

Its discrete form may be written as

$$S_{EC} = \lambda \sum_{<ij>} (1 - \hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j) \tag{57}$$

where i and j represent triangles that share a common edge and $\hat{\mathbf{n}}_i$ is the unit normal to triangle i (Fig. 12).

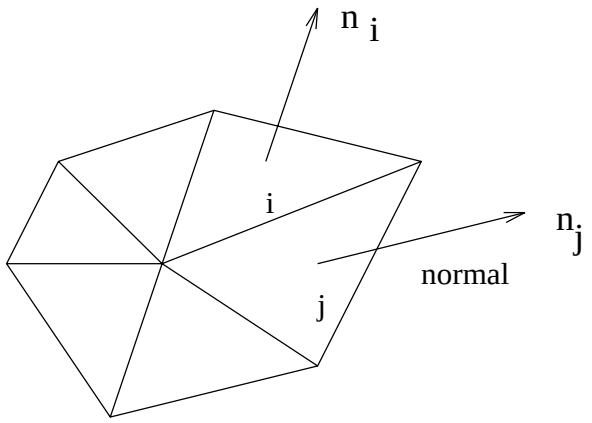

Figure 12: The normals of two adjacent triangles

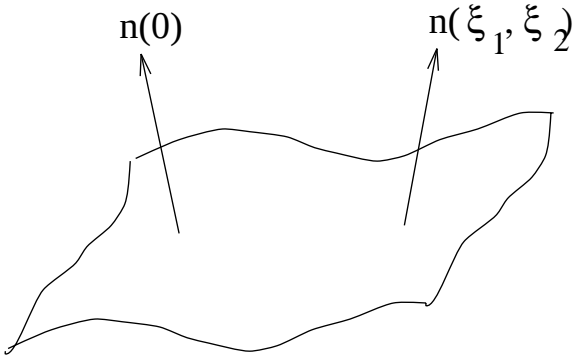

Figure 13: Two normals at two different points on a surface

2.2.2 Perspective – with Extrinsic Curvature

From Fig. 12 and equation (54) S_{EC} given in (57) clearly suppresses local fluctuations in the mean curvature of the surface. But the key question is whether there is long-range order in the normals to the surface (see Fig. 13). The bending rigidity is, in fact, a running coupling — it depends on the scale at which it is measured. A perturbative calculation in the inverse coupling λ^{-1} reveals that strings with bending rigidity are asymptotically free in the same sense as Quantum Chromodynamics. Fluctuations screen the theory and soften the effective bending rigidity as the length scale increases.

The momentum p dependence of λ is found to be [18]

$$\lambda^{-1}(p) = \frac{\lambda_0^{-1}}{1 - \frac{d}{2}\frac{\lambda_0^{-1}}{2\pi}\log\frac{\Lambda}{p}} \tag{58}$$

where Λ is the cutoff or inverse lattice spacing and d is the dimensionality of the target space. At large length scales λ tends to zero and there is no suppression of fluctuations in the alignment of normals to the surface. The two-point function decays exponentially

$$< \hat{\mathbf{n}}(\xi_1, \xi_2) \cdot \hat{\mathbf{n}}(0) >= e^{-\frac{|\xi|}{\xi_p}} \tag{59}$$

with persistence length ξ_p. Thus the surface is always disordered or *crumpled* at length scales r exceeding ξ_p. This conclusion is of considerable interest in the study of liquid membranes as well. A typical example of a liquid membrane found in nature (which can also be manufactured in the laboratory) is a lipid bilayer (Fig. 4). It consists of two layers of amphiphilic molecules with polar hydrophilic heads and long hydrophobic hydrocarbon tails. These bilayers arrange themselves in thin extended sheets. Within the bilayer individual molecules are quite free to diffuse, so that the in-plane elastic constants turn out to be very low. Another candidate liquid membrane is a monolayer of surfactant molecules at an oil-water interface in a micro-emulsion. In fact any flexible interface between three-dimensional phases is a candidate system for a liquid membrane model. We see here a beautiful interplay between string theory, quantum gravity and the statistical mechanics of fluctuating liquid membranes [37].

2.3 The Model of Random Surfaces used in the simulations

We studied the theory defined by the action

$$S = S_{Gauss} + \lambda S_{EC} = \sum_{i,j,\mu} C_{ij}(X_i^\mu - X_j^\mu)^2 + \lambda \sum_{<ij>} (1 - n_{\hat{k}}^\mu \cdot n_{\hat{l}}^\mu) \ . \tag{60}$$

where S_{Gauss} is the discrete form of Polyakov Action (9), S_{EC} is the discrete form of extrinsic curvature action and C_{ij} is the adjacency matrix (43). Thus, for $\lambda > 0$, we have introduced a ferromagnetic interaction in the surface normals. The model defined by this action has been studied in [38, 39, 40, 41, 42, 43] and references therein.

From (50) and the definition of the induced metric, it follows that equation (60) is a discretization of the continuum action [42, 44]

$$S = \int d^2\sigma \sqrt{|\det g|}(g^{ij}\partial_i X^\mu \partial_j X^\mu + \frac{\lambda}{2} g^{ij} h^{kl} K_{ik} K_{jl}) \tag{61}$$

Note that the second term in this action is manifestly positive and reparametrization invariant, and that λ is a dimensionless coupling. So, naively, it is not clear whether it is relevant or not. If it were relevant, one would then anticipate that (since it obeys all of the appropriate symmetries) it should be effectively generated in any string action, and that it should engender ordering of the normals. It should then lead to another renormalization group (RG) fixed point at a finite value of λ, which would characterize a phase transition between the crumpled phase (observed when $\lambda = 0$) and a renormalization 'smooth(er)' phase. The extrinsic curvature term is also higher-derivative, indicating that the field theory described by this action is non-unitary. This fact alone does not imply that the associated string-scattering amplitudes do not satisfy unitarity. Polchinski and Yang [45] do, however, contend that in this case the string theory will not be unitary. Even if this were so, this model could still be an appropriate description of the statistical mechanics of fluctuating surfaces, although not one corresponding to a physical fundamental string theory. Braaten and Zachos [46] have also showed that the generic static classical solutions of a similar higher-derivative theory of rigid strings are unstable. This would also imply that these actions could only be the basis for ef-

fective, but not fundamental, quantum theories of strings. We proceed first to review previous work which has addressed the question of whether or not these theories exhibit a crumpling transition.

2.3.1 Previous Analytical Work

A group analysis [18, 20, 22, 47] indicates, however, that there should be no phase transition at finite coupling when such extrinsic curvature dependent operators are added to the action. The computations of refs. [20, 22, 47] use the action

$$S = \int d^2\sigma \ (\mu_0 \sqrt{\det h} + \frac{1}{\alpha}\sqrt{\det h}(h^{ij}K_{ij})^2) \ , \tag{62}$$

in the regime in which the string tension μ_0 is small (unlike the usual particle physics limit of string theory, which is characterized by large μ_0). In (62) α is the inverse of coupling constant. After integrating out fluctuations of the embedding X^μ between momentum scales Λ and $\tilde{\Lambda}$, it is found that the renormalization of the extrinsic curvature coupling is given to one-loop order by

$$\beta(\alpha) \equiv \Lambda \frac{d\alpha}{d\Lambda} = -\frac{3}{4\pi}\alpha^2 \ , \tag{63}$$

so that α is driven to infinity in the infra-red. This theory thus exhibits asymptotic freedom. Surfaces are smooth (the normals are correlated) below a persistence length[19].

$$\xi_p \sim \exp(\frac{4\pi}{3\alpha_{bare}}) \ , \tag{64}$$

and are disordered above this scale. Some intuition into this result can be gained by observing that this theory is similar to the $O(3)$ sigma model, which is asymptotically free [48]. The normals to M are the analogues of $O(3)$ vectors, though in this case they are constrained to be normal to a surface governed by the action (61).

Without the extrinsic curvature term, (62) is the Nambu-Goto action, while (61), which we use in our simulations, is based on the action quantized by Polyakov. Classically (when the equations of motion for the Polyakov action are solved and substituted back into the action) the two actions are equivalent. It has also been demonstrated [49] that the two quantizations are equivalent in the critical dimension $D = 26$. In lower dimensions (note that the Nambu-Goto action clearly does not make sense for $D < 2$), it is not so clear that quantizations 'based' on the two actions are indeed the same. The work of Polchinski and Strominger [50] suggests that there are alternative quantizations. Distler [51] has also questioned the equivalence of these quantizations in $D = 3$. Indeed, even if the two quantizations are equivalent, it does not automatically follow that the two theories are still the same once an extrinsic curvature dependent term has been added.

In fact, Polyakov in [18] uses a hybrid form of the action (62) and still obtains the same result for the beta function. He introduces an intrinsic metric g_{ij}, chooses the conformal gauge $g_{ij} = \rho\delta_{ij}$ and considers

$$S = \frac{1}{2\alpha}\int d^2\sigma(\mu_o\rho + \rho^{-1}(\partial^2 X^\mu)(\partial^2 X^\mu) + \lambda^{ij}(\partial_i X^\mu \partial_j X^\mu - \rho\delta_{ij})) \ . \qquad (65)$$

Classically, the Lagrange multiplier λ^{ij} constrains the intrinsic metric to equal the induced metric (this equality is not enforced by the classical equations of motion for the original Polyakov action). This constraint should be relaxed quantum mechanically if, as Polyakov [48] argues, the condensate of this Lagrange multiplier assumes a value of the order of the momentum cutoff. If this dynamical assumption is correct, then one can essentially derive the equivalence of this Nambu-Goto like and the original Polyakov quantizations. In the large D (embedding dimension) limit, saddle point calculations

[52] show that λ indeed does acquire a large expectation value, and that for small values of the string tension μ_o, the coupling α is asymptotically free, as the RG calculations suggest.

There are, however, a couple of caveats and suggestions in the analytic literature that do allow for the existence of a crumpling transition for fluid surfaces. Polyakov remarks that if, in the infrared region, fluctuations of the internal geometry (ρ) are suppressed relative to fluctuations of the extrinsic metric, then the beta function is proportional to α and hence the continuum limit of the theory exhibits non-trivial scaling behavior; this presumably cannot be the case in the large D limit. Another RG calculation, performed by Yang [53] using the Polchinski-Strominger action [50] with an extrinsic curvature dependent term, indicates that the two-loop correction (which is proportional to α^3) might be large enough to yield a zero of the beta function, and thus a non-trivial infrared fixed point. The Polchinski-Strominger action is based on the assumption that the Liouville mode ρ effectively decouples (its mass is much greater than the momentum scale set by the string tension); it is not clear why this assumption should hold for the model that we simulate. Finally, note that these computations are perturbative (in $1/D$ or α). It is possible that non-perturbative effects could drive a crumpling transition.

2.3.2 Previous Numerical Evidence

Monte Carlo simulations of the action (61) on dynamically triangulated random surfaces (DTRS) were first performed by Catterall [38], and shortly thereafter by Baillie, Johnston, and Williams [39, 40] and Catterall, Kogut and Renken [43]. They simulated triangulations with the topology of the sphere, and measured the specific heat

$$C(\lambda) \equiv \frac{\lambda^2}{N}(< S_E^2 > - < S_E >^2) \, , \tag{66}$$

on surfaces with up to $N = 144$ nodes (and $N = 288$ nodes in [41]). They found a peak in the specific heat; the peak size appeared to grow with N. A similar model that can be vectorized rather straightforwardly was also considered; the set of planar ϕ^3 graphs was simulated [43, 54]. Each vertex of these ϕ^3 graphs was embedded in R^3 and the action (61) was used; graphs of up to 1000 nodes were simulated (these would be dual to 500 node triangulations). It was found that the specific heat peak grew with N, albeit slowly, as

$$C_{max} = AN^{\omega} + B \ , \tag{67}$$

with $\omega = 0.185(50)$. Further work by Ambjørn *et al* [41, 42], using dynamical triangulations with the topology of the torus and lattices with up to $N = 576$ nodes, indicated that the rate of increase of the peak height severely diminishes with increasing N. The data strongly suggests that in fact the specific heat peak height does not diverge as $N \to \infty$. These authors also measured the bare string tension and mass gap, by embedding the torus in a background toroidal space spanned by a loop, and measuring the dependence of the free energy on the loop size. They found that these measurements (when taken for λ values near the peak position) are consistent with the appropriate scaling relations (with vanishing bare string tension and mass gap, i.e. correlation length tends to be infinite because of the relation (24)) that should characterize a phase transition to smooth surfaces. This measurement, although it constitutes the best evidence there is so far for a real phase transition at $\lambda = \lambda_c$, is still quite an indirect way of measuring correlation functions. As we will discuss, these scaling relations could contradict other observed phenomena such as the absence of diverging correlation times and increasing finite size effects at the putative critical point.

Thus it appears that numerical evidence could allow for the existence of a crum-

pling transition (most probably of higher order), while analytical calculations generally indicate that no such transition should occur.

In [55] the peak was measured in a DTRS simulation that incorporated self-avoidance and the extrinsic curvature term S_E, with a solid-wall potential substituted for the Gaussian term in the action. The results for the specific heat turned out to be very similar to those found in the simulations we have just discussed, for example, in [42]. The specific heat peak is, in this context, considered to be a lattice artifact, because the peak height levels off with large N (of order 500). These simulations included a crude block-spin measurement that suggests that the renormalization group flow of λ is consistent with the analytical result of asymptotic freedom.

Simulations using other discretizations for the extrinsic curvature dependent term have yielded somewhat different results [38, 39]. The specific heat peak, measured in simulations employing what is referred to as the 'area discretization', is rather feeble, and levels off for small values of N (by $N = 72$) The authors interpret this as being indicative of perhaps a third order transition. Actions based on these various discretizations have been simulated for fixed, triangular meshes. These systems model tethered or crystalline membranes, in which the constituent molecules are tightly bound together. In the tethered case, the specific heat peak obtained from simulations of the edge action (61) grows vigorously as a function of N for very large (128×128) lattices [91]. This is strong evidence for the existence of a second order transition which, in this case, is in accord with the analytic results – these calculations are reviewed by Nelson [12] and David [57, 58] and involve mean field and large D computations which suggest that the β function is linear at leading order, with a zero for finite α, i.e. a ultraviolet fixed point. When the alternate area discretization is used in the tethered case, the specific heat

peak again stops growing. Recent work has demonstrated that this other discretization is pathological in the tethered case; the class of 'corrugated' surfaces, which are singular in one direction and smooth in the other, then dominates the path integral [91].

Thus, given the muddle of somewhat contradictory evidence, it is unclear whether or not a crumpling transition exists for fluid surfaces. We have pursued this question by taking high statistics measurements of the specific heat peak, and by measuring many other observables describing the geometry of these surfaces, since observables with different quantum numbers can give quite different information. For example, in the Ising model the magnetization behaves quite unlike the internal energy (which is invariant under the standard Z_2 transformation).

To analyze and interpret this data, we have applied insights gained from work on better understood systems, primarily spin models and lattice gauge theories. Issues of the equivalence of the Nambu-Goto and Polyakov quantizations have also motivated us to compare the intrinsic and induced geometry of the surfaces that we simulate.

2.4 The Goal of the Simulations – Is there a Phase Transition?

In the last few years random surfaces have been extensively explored via numerical simulations on a wide range of computers, including parallel machines [38, 39, 40, 41]. There are some novel but not fully understood results. The full action which is simulated is given by a quadratic interaction term plus the extrinsic curvature term in equation (60). For $\lambda < \lambda_c \simeq 1.5$ one sees the expected crumpled surface (see Fig. 43). The radius of gyration of these surfaces grows only logarithmically with their area corresponding to infinite Hausdorf dimension d_H defined by

$$R_G^2 \simeq A^{\frac{2}{d_H}} \tag{68}$$

where R_G is the radius of gyration. For $\lambda > \lambda_c$ the surfaces become extended and considerably smoother with d_H approaching two, which would be the value one would get for a flat surface (see Fig. 47). The nature of the cross-over at λ_c is still uncertain. It may be that the system is undergoing a true thermodynamic phase transition. If it is of second order then the continuum limit constructed at the critical coupling would be an interesting string theory corresponding to a real extended $2d$ surface rather than a branched polymer with its largely one-dimensional character. In this case it must be that the coupling $\frac{1}{\lambda}$ ceases to vary with scale (there is a fixed point of the beta function $q\frac{d\lambda}{dq}$) at the critical coupling λ_c. At this point there is said to be a *crumpling transition*. This is the most exciting possibility from the string point of view because it would mean that we have successfully regularized and defined the quantum theory of the string with more than one embedding dimension without any instability arising. The challenge would then be to understand the exact nature of the continuum string theory at the crumpling transition and the origin of the fixed point.

It may also be that the observed cross-over is not a true phase transition and that the persistence length is simply reaching the finite size of the surface that is simulated on the computer. In this case it could still be that the surface is always crumpled on sufficiently large distance scales. This is a real possibility for a liquid membrane but would still leave us without a viable lattice regularization of a string in $d > 1$ dimensions. Our study was focused on deciding which of the above possibilities was in fact correct by performing large-scale simulations in three embedding dimensions [59].

Finally it is of great interest to extend the technique of dynamically triangulated surfaces to manifolds of higher dimension, in particular to three and four dimensional manifolds. One can then simulate say four dimensional Einstein-Hilbert quantum gravity

and seek critical points which provide a non-perturbative definition of a perturbatively non-renormalizable quantum field theory. This would be a very exciting development. Preliminary work indeed seems to indicate that there are indeed phase transitions in $4d$ gravity [61]

Evidences from our recent and ongoing numerical simulations of dynamically triangulated random surfaces indicate that there is a non-trivial crossover from a crumpled to an extended surface as the bending rigidity is increased. The results will be present in following sections.

3 Computer Simulations and Measurements

In Section 3.1 we define the quantities we have decided to measure, and explain why they are physically interesting. Next, in Section 3.2, we present the details of our numerical simulations, including a complete high-statistics analysis of the behavior of a set of relevant observables. Since computing correlation functions on dynamically triangulated surfaces is a difficult task, we have focused on elucidating the phase diagram by analyzing local observables in great detail.

3.1 The Observables

To minimize finite size effects, we have considered triangulations with the topology of the torus. The action (61) was used, with the BRST invariant measure utilized also by Baillie, Johnston, and Williams [39], so that

$$Z = \sum_{G \in T(1)} \int \prod_{\mu,i} dX_i^\mu \prod_i q_i^{\frac{d}{2}} \exp(-S_{Gauss} - \lambda S_E) \ , \tag{69}$$

where $d = 3$, q_i is the connectivity of the ith vertex, and $T(1)$ refers to the set of genus 1. The authors of [41, 42] do not include this connectivity dependent term in their measure. The long-distance physics of the simulations is presumably insensitive to the presence of this term. Because we have chosen a different measure, though, our quantitative results cannot be precisely compared with theirs.

We measured a variety of quantities that characterize the extrinsic and intrinsic geometry of these surfaces. These observables include:

1. The edge curvature S_E and the associated specific heat $C(\lambda)$, which is a sensitive

indicator of the presence of a phase transition.

$$C(\lambda) = \frac{\lambda^2}{N}(< S_E^2 > - < S_E >^2) \ . \tag{70}$$

This exhibits a peak at a coupling λ_c which depends on the exact discrete form of the action chosen [38, 39, 40, 41, 42, 43, 54, 59]

2. The squared radius of gyration R_G;

$$R_G \equiv \frac{1}{N}\sum_{i,\mu}(X_i^\mu - X_{\rm com}^\mu)^2 \ , \tag{71}$$

where the com subscript refers to the center of mass of the surface. By measuring the N dependence of the gyration radius, we can extract a value for the extrinsic Hausdorff dimension, which is given by [42]

$$R_G \sim N^\nu \sim N^{\frac{2}{d_{\rm extr}}} \ . \tag{72}$$

3. The magnitude of the extrinsic Gaussian curvature. We measure a discretization of $\int \mid K \mid \sqrt{\mid h \mid}$, with

$$\mid \mathcal{K} \mid = \frac{1}{N}\sum_i \mid 2\pi - \sum_{\hat{j}} \phi_i^{\hat{j}} \mid \ . \tag{73}$$

Here h is the induced matrix (49 and $\phi_i^{\hat{j}}$ denotes the angle subtended by the $\hat{j}$th triangle at the ith vertex. This quantity, therefore, measures the magnitude of the deficit angle in the embedding space averaged over all vertices. We also record the mean square fluctuation of $\mid \mathcal{K} \mid$, denoted by $F[\mid \mathcal{K} \mid]$.

4. The corresponding intrinsic quantity, $\mid \mathcal{R} \mid$, given by

$$\mid \mathcal{R} \mid = \frac{\pi}{3N}\sum_i \mid 6 - q_i \mid \ , \tag{74}$$

and its fluctuations. When the intrinsic and extrinsic metrics are equal, the intrinsic and extrinsic deficit angles are identical, and $K = R/2$.

5. To study the correlation between intrinsic and extrinsic geometry, we also measure the quantity which we refer to as $\mathcal{K} * \mathcal{R}$:

$$\mathcal{K} * \mathcal{R} \equiv \frac{\int K R}{\sqrt{\int K^2 \int R^2}} = \frac{\sum_i (2\pi - \sum_{\hat{j}} \phi_i^{\hat{j}})(6 - q_i)}{\sqrt{\sum_i (2\pi - \sum_{\hat{j}} \phi_i^{\hat{j}})^2 \sum_i (6 - q_i)^2}} \ . \tag{75}$$

This quantity is 1 when the metrics are equal, 0 if they are uncorrelated, and negative when these curvatures are anti-correlated.

6. We measure, finally, the average maximum coordination number of the surface vertices, $\max_i q_i$.

3.2 The Numerical Simulation

In this chapter, I will mainly describe the physical model implemented in the computer program and leave the explanation of the computer program to chapter 5 where other computational issues will be discussed.

In our simulations we have used the standard Metropolis algorithm to update the embedding fields X_i^μ. To sweep through the space of triangulations we performed flips (see reference [25]) on randomly chosen links. Flips were automatically rejected if they yielded a degenerate triangulation; i.e. one in which a particular vertex has fewer than three neighbors, or in which a vertex is labeled as its own neighbor, or where more than one link connects two vertices. (It has been proven in ref. [25, 26] that the entire space of graphs of a given topology can be spanned by only performing these flips.) After a set of $3N$ flips was performed, $3N$ randomly selected embedding coordinates were updated via random shifts from a flat distribution (Fig. 14),

$$X^\mu \to X^\mu + \delta X^\mu \ . \tag{76}$$

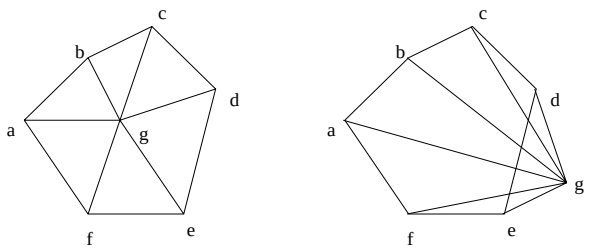

Figure 14: Update node g

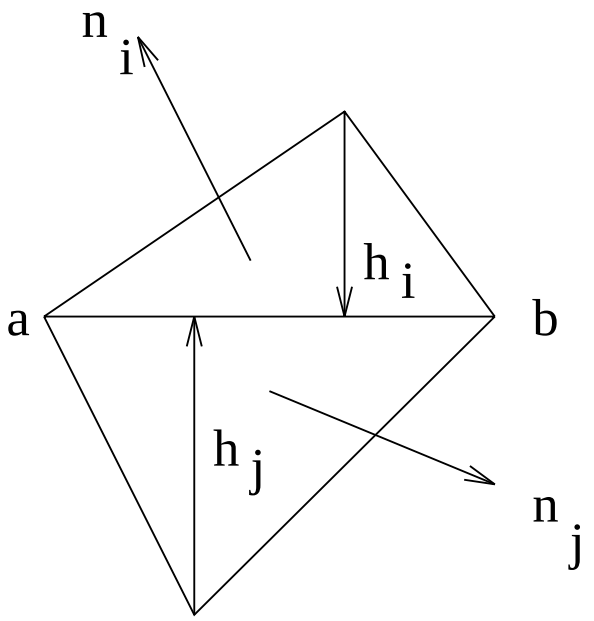

Figure 15: Relation between normals and perpendiculars

The mean magnitude of these shifts

$$< \delta X^{\mu} \delta X^{\mu} > \tag{77}$$

was chosen so that the acceptance rate for updates of the X^{μ} was roughly 50 percent.

It can be very easily shown that the dot product between two normals of two adjacent triangles $\hat{n}_i \cdot \hat{n}_j$ can be converted to the dot product of two perpendiculars as shown in Fig. 15. The relation is

$$\hat{n}_i \cdot \hat{n}_j = -\vec{h}_i \cdot \vec{h}_j / |h_i||h_j| \tag{78}$$

N=36			1.375	1.400	1.425	1.475		λ
			3	3	3	3		$\times 10^6$ sweeps
N = 72			1.375	1.400	1.425	1.475		λ
			3	3	3	3		$\times 10^6$ sweeps
N=144	.8	1.25	1.35	1.40	1.45	1.50	2.0	λ
	3	3	3	3	3	3	3	$\times 10^6$ sweeps
N=288	.8		1.375	1.40	1.425	1.475	2.0	λ
	14.4		21.0	15.0	16.2	13.5	14.4	$\times 10^6$ sweeps
N= 576	.8	1.325	1.375	1.40	1.425	1.475	2.0	λ
	12.0	27.0	27.0	27.0	27.0	27.0	9.6	$\times 10^6$ sweeps
N= 1152					1.425	1.430	1.435	λ
					54	21	18	$\times 10^6$ sweeps
N= 2304			1.40	1.42	1.425	1.430		λ
			5	5	17	5		$\times 10^6$ sweeps

Table 1: A record of the number of sweeps performed at each different λ value for different lattice sizes.

Most of the Monte Carlo simulations were performed on HP-9000 (720 and 750 series) workstations; we also collected some data by simulating lattices on each of the 32 nodes of a CM-5. Our code was in Fortran, with the ranmar, described in reference [62].

In Table 1 we summarize our runs. All these runs were done in two major stages. At the stage one, with the speed of our initial Fortran program, the feasible lattice size we could run was up to 576 nodes. After the runs, we found out that the results were not conclusive, i.e., the result was not at all clear whether a true continuous thermodynamic

phase transition separated the two regimes. There could be a several alternative interpretations. Perhaps the simplest possibility, advocated in [55], is that the persistence length ξ describing the exponential decay of the normal-normal two-point function in the crumpled (disordered) regime simply reaches the finite size of the system at λ_c. In this case the observed smooth regime would be a finite-size artifact with the true continuum theory really being crumpled for all couplings λ, in accordance with perturbative analytical results [18, 20, 22, 47]. Since ξ grows exponentially with λ, according to the one-loop beta-function, this interpretation would imply that λ_c diverges logarithmically with system size N. To resolve this issue and to gain further insight into the model it was clearly desirable to extend the numerical simulations to larger lattice sizes and to clarify the influence of finite-size effects.

Therefore, we put substantial effort into optimizing our program. The new program was much faster (see Chapter 5). Within a reasonable period of time, we ran the simulations for lattice size up to 2304 nodes. Note that in table 1, we have performed quite long runs on the larger lattice sizes. We will discuss in Section (3.4) why we believe runs of this length are just sufficient to yield accurate values of the observables for the largest lattice size ($N = 2304$).

We also plotted 3 dimensional view of random surfaces with different number of nodes at different λ value (see Chapter 5). We found the surfaces look like smoother as λ increases.

In all of our figures the different points will be printed with their associated statistical error (sometimes too small to be visible). The statistical error is computed by means of a standard binning procedure. We will explicitly discuss the cases in which our estimator for the statistical error is not asymptotic.

The lines in these figures are from a histogram reconstruction (see for example [63, 64]). We patch different histograms [65, 66, 67] by weighting them with the associated statistical reconstructions in determination (which we estimate by a jack-knife procedure); this procedure seems to be very effective and reliable. All of the reconstruction curve sets (3: dotted, dashed and continuous for 3 surface sizes on each figure) consist of 3 curves (which sometimes appear as a single one). The middle curve is the histogram reconstruction, and the upper and the lower ones bound the data within the errors obtained by the procedure we have just described.

In our latest series of simulations on lattices, for $N = 144$ we have patched the four histograms originating from $\lambda = 1.35, 1.40, 1.45, 1.50$. For $N = 288$ we have used $\lambda = 1.375, 1.40, 1.425$ and 1.475. For $N = 576$, we chose $\lambda = 1.375, 1.40$ and 1.425. With 1152 nodes we ran 54 million sweeps at $\lambda = 1.425$, 21 million sweeps at $\lambda = 1.430$ and 18 million sweeps at $\lambda = 1.435$. Since the autocorrelation time τ is of order $400,000$ sweeps on the 1152 lattice these runs have at least 45τ measurements. On the data from these three points we use multi-histogram reconstruction [63, 67]. This works well in that three different reconstructions give coherent results. On lattices of 2304 nodes we have poorer statistics. We ran 17 million sweeps at $\lambda = 1.425$ plus approximately 5 million sweeps at $\lambda = 1.40, 1.42$ and 1.43 as a consistency check. On the 2304 lattice histogramming does not work well. This is to be expected since the statistics are not good enough for such a large lattice. Still we have checked that our measurements at $\lambda = 1.425$ give consistent results, that the error estimate is reliable and that we are, with good accuracy, at the peak of the specific heat. In all these simulations for 1152 and 2304 required the equivalent of approximately one year of CPU time on an HP 9000 (720 series) workstation. (for 1152 nodes and 2304 nodes, we also measured all

observables we described above, but we only plotted those of most physical interest.

We also thought of the possibilities that could account for the observed behavior of $C(\lambda)$ without invoking a phase transition. One was based on the analogy between the present model and the $O(3)$ sigma-model in $2d$ [68]. This model is also asymptotically free and consequently disordered at all non-zero temperatures.

The simulations were done on square lattices of volume N = 16, 25, 64, 100, 900, 2,500, 4,900 and 10,000 using the Wolff algorithm [69]. For each point of the N = 25,100 and 10,000 lattices we used 100,000 measurements. We took a measurement every time the Wolff clusters updated a volume exceeding 30 times the volume of the lattice. For the N = 16, 64, 900 and 4,900 lattices the integrated autocorrelation times were between 1 and 2 Wolff updates of the entire lattice. For each point of the N = 16, 64, 900, 2,500 and 4,900 lattices we used 20,000 measurements. We took a measurement every time the Wolff clusters updated a volume exceeding 3 times the volume of the lattice.

We have only drawn the reconstructed, patched curves (with their reliable errors) in the regions where we trust them. For example, close to the pseudo-critical region we can trust a peak pattern only when we can reconstruct the peak by starting from both sides of the transition (without multi-histogram patching). So we have always used single histogram to check these criteria, before constructing the final, multi-histogram data.

3.3 The Phase Diagram

We have measured, as stated previously, a large number of local observables. We will see that a mixed picture emerges from these measurements. For example the observables related to the dynamical triangulations exhibit a characteristic pattern, to be discussed

in detail below.

Figure 16: The edge curvature S_E as a function of λ. As in all other pictures, filled circles and a dotted line correspond to $N = 144$, crosses and a dashed line indicate $N = 288$, and empty squares and a solid line represent $N = 576$.

We start by showing, in Fig. 16, the edge curvature S_E as a function of λ. The crossover region is around $\lambda \simeq 1.4$. For small values of λ, the surface is crumpled (see the latter part of this section). In this region, finite size effects are already negligible for our lattice sizes, and our 3 data points are on top of each other. We can see weak finite size effects by comparing the continuous lines in the transient region. The $N = 144$ dotted line is far from the ones of the two larger lattices, which lie, on the contrary, on top of each other. Finite size effects are larger in the large λ phase. One would expect, close to a phase transition with a diverging correlation length, an increase in finite size effects which we do not observe here. The lattice should feel the presence of the zero

mass excitation, and the finite size corrections should be larger than everywhere else (in the case of periodic boundary conditions they would obey a power-law, rather than decaying exponentially with size). This is surely not firm evidence against the presence of a phase transition, but it does show that the putative critical behavior is atypical.

The errors in the 'flat phase' ($\lambda = 2.0$) are not under control. Our estimators do not plateau under repeated iterations of the binning procedure. In this regime, correlation times are large, as we will discuss in next section. This *caveat* holds for this figure and for all the quantities we have measured.

Figure 17: The edge curvature specific heat, for lattice size 144, 288 and 576, $C(\lambda)$

In Fig. 17 we show the related specific heat $C(\lambda)$, in the same λ region. In Fig. 18 we enlarge the pseudo-critical λ region, in order to show the reconstructed peak of the specific heat. As already noted our reconstruction procedure is quite reliable here.

The specific heat peak for $N = 576$, 1152 and 2304 from the new runs is shown in

Figure 18: As in Fig. 16, but with the multi-histogram reconstruction in the transient region.

Fig. 19.

Table 2 gives the results for the maximum of the specific heat and the associated coupling λ_c as a function of N. We have reanalyzed the data presented in reference [59], using a different method of weighting relative errors when combining histograms. Thus, some of the errors quoted here are smaller than the respective uncertainties in reference [59].

We see that the specific heat peak grows vigorously with N for small lattices, so it appears that there is a crumpling transition very similar to that of the crystalline surface. However the specific heat height growth levels off for larger N. From the data for the three largest lattices, we can extract a specific heat exponent $\omega = .06 \pm .05$, with ω defined as in equation (67), and the constant B set to zero. If we estimate an

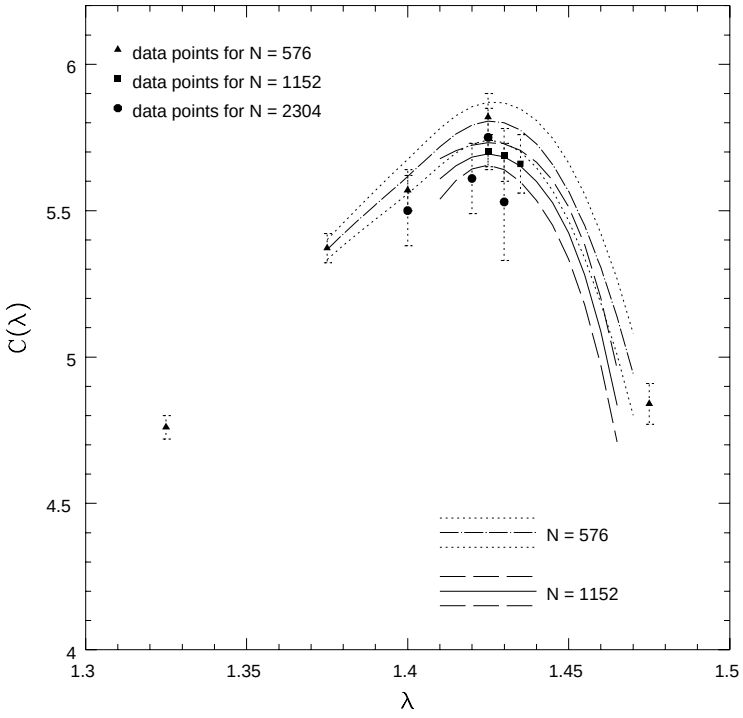

Figure 19: The edge extrinsic-curvature specific heat $C(\lambda)$ as a function of λ. Multi-histogram reconstructions with errors are shown for $N = 576$ (long and short dashed lines) and $N = 1,152$ (solid lines). Four individual data points are also shown for $N = 2,304$ (solid circles). One sees that the specific heat peak has saturated, i.e., it is not growing with the system size N above 576.

effective exponent from the $N = 144$ and 288 lattices, we get $.05 \pm .06$, and from the two largest lattice sizes we get $.07 \pm .06$; this demonstrates that we do not see, within our statistical precision, any sign of a non-pure-power, non-asymptotic behavior. Note that if the constraint that B vanishes is relaxed, our data is not accurate enough to yield a meaningful fit to equation (67). A very small (asymptotically finite) correlation length is sufficient to produce such a small effect on our quite small lattice sizes. These results appear to be consistent with those of the Copenhagen group [42], and they are not so

N	$C^{(max)}$	λ_c
36	$3.484 \pm .008$	$1.425 \pm .035$
72	$4.571 \pm .015$	$1.410 \pm .015$
144	$5.37 \pm .08$	$1.395 \pm .017$
288	$5.55 \pm .05$	$1.410 \pm .015$
576	$5.81 \pm .06$	$1.425 \pm .010$
1152	$5.69 \pm .04$	$1.425 \pm .010$
2304	$5.75 \pm .10$	$1.425 \pm .010$

Table 2: The maximum of the specific heat and its position, with errors, for different lattice sizes.

far from the ones of the Urbana group [43, 54].

The critical value of λ shifts very slowly to higher values for increasing N for the largest lattices, although the increase is not statistically significant.

In addition the shape of the specific heat (for example the width) is basically unchanged as we go to larger lattices. From Figs. 17, 18, 19 we do not infer evidence of criticality.

In Fig. 20 we show the radius of gyration of the surface, R_G, as defined in (71). Here obviously the volume scaling is non-trivial: larger surfaces have larger radius. The histogram reconstruction already ceases to work for quite low values of λ for the larger lattice. This effect could be related to the interesting finite size scaling behavior of this quantity, which we illustrate in better detail in Fig. 21. Here we plot

$$\nu(N) \equiv \frac{\log \frac{R(N)}{R(\frac{N}{2})}}{\log(2)} \ . \tag{79}$$

This is an effective inverse Hausdorff dimension, which is a function of λ. In the large λ limit $\nu \to 1$ and $d_{extr} \to 2$, as expected for flat surfaces. In the low λ limit d_{extr} becomes very large. In the pseudo-critical region ν is a linear function of λ. Curiously enough, the latter curve yields a Hausdorff dimension of 4, a value characteristic of branched polymers, near the location of the specific heat peak. This value is not particularly reliable though because of finite-size effects and also because it changes rapidly in this region. In ref. [42] a value compatible with ours ($D_H(\lambda_c) > 3.4$) is quoted for the critical theory. We stress however (and also here we are in complete agreement with [42]) that the dimension in the pseudo-critical region depends heavily and quite unusually on N.

Figure 20: The gyration radius R_G defined in equation (71) plotted as in Fig. 16

In both the high and low λ regions finite size effects are quite small (compatible with zero to one standard deviation). In the pseudo-critical region, on the contrary, finite size effects are large. This effect cannot be explained by the shift in λ which one gets

Figure 21: The effective inverse Hausdorff dimension ν as a function of λ, as defined in (72). The filled dots and the dashed curve are from a fit to the $N = 288$ and $N = 144$ data, while the empty dots and solid curve represent the fit to $N = 576$ and $N = 288$.

from the shift of the peak of the specific heat, which is far too small. This behavior is very different from that we discussed for S_E and it seems to indicate the possibility of some sort of critical behavior close to $\lambda = 1.4$.

With our new runs on larger lattice, we note the behavior of the gyration radius at λ_c. For large λ (> 2), the scaling of $R(N) \sim N^{\frac{2}{d_H}}$ with N gives a Hausdorff dimension close to 2 (as we expect for flat surfaces). In the crumpled region the Hausdorff dimension rapidly increases with diminishing λ. We had pointed out that finite size effects were relevant in the sector close to λ_c and that we could not estimate a reliable number from the lattice sizes analyzed. Here the largest lattice we simulated ($N = 2304$) does not give useful data, since the error in R is too large, but on the 1152 and 576 node lattices we get

a fairly precise estimate of R, which allows us to estimate for the Hausdorff dimension at the pseudo-critical point λ_c the value $d_H = 4.35 \pm .3$. This is an intriguing result, since 4 is the extrinsic Hausdorff dimension of a class of branched polymers, as constructed, for instance, in references [70]. Such configurations are expected to dominate the string functional integral for large embedding dimension D.

In Fig. 22 we plot the expectation values of the magnitude of the extrinsic Gaussian curvature $\mid \mathcal{K} \mid$. If the induced metric is equal to the intrinsic metric, then $\mid \mathcal{K} \mid = \frac{\mid \mathcal{R} \mid}{2}$.

Figure 22: The extrinsic Gaussian curvature $\mid \mathcal{K} \mid$ defined in (73), plotted as in Fig. 16

This plot is not substantially different from that of S_E. We note that finite size effects are somewhat larger in this case than for the edge action, but they follow the same pattern (exhibiting a big increase in the flat phase).

The plot of the fluctuations of the extrinsic Gaussian curvature, $F[\mathcal{K}]$, which we present in Fig. 23, shows something very new. A very sharp crossover, with perhaps a

peak developing for large N, dominates the pseudo-critical behavior. Fluctuations do not seem to depend on λ in the crumpled phase, while they drop dramatically, in a very small λ interval, in the flat region. Here again, finite size effects are sizeable in the pseudo-critical region. The position of the crossover does not depend sensitively on N, while the detailed shape at λ_c seems to change slightly with N.

Figure 23: The fluctuations of $\mid \mathcal{K} \mid$.

We measured again in our new runs on larger lattices the fluctuations of the extrinsic Gaussian curvature $|\mathcal{K}|$, and computed the fluctuations of the mean defect coordination number $|q-6|$. We find that on larger lattices the fluctuations of these observables at λ_c also do not grow with N; thus their behavior does not provide unequivocal evidence of the presence of a phase transition.

Table 3 gives the mean-square fluctuations of both observables.

It is difficult to give a precise interpretation of a plot like this, however, the crossover

N	$F[K]$	$F[q-6]$
576	$5.71 \pm .08$	$8.39 \pm .04$
1152	$5.59 \pm .05$	$8.32 \pm .03$
2304	$5.70 \pm .10$	$8.37 \pm .06$

Table 3: The mean square fluctuations of the extrinsic Gaussian curvature $\mathcal{K}$ and the defect coordination number $q-6$, with errors, for different lattice sizes.

is very clear here.

In Fig. 24 we give the intrinsic curvature $\mathcal{R}$ and in Fig. 25 its fluctuations. Both plots are very similar to the related, extrinsic curvature, $\mathcal{K}$ plots. $\mid \mathcal{R} \mid$ drops off rapidly, just as $\mid \mathcal{K} \mid$ does. Through the peak region, though, $\mid \mathcal{K} \mid$ decreases by about a factor of 5 while $\mid \mathcal{R} \mid$ diminishes to only about .6 of its value on the left-hand side of the peak. Since the action explicitly suppresses mean curvature, and the mean and extrinsic Gaussian curvature are closely related (for instance, $H^2 > \frac{K}{2}$), we would expect that for large λ extrinsic fluctuations would be suppressed much more than fluctuations of intrinsic geometry.

In Fig. 26 we plot the intrinsic extrinsic curvature correlation. The plot of $\mathcal{K} * \mathcal{R}$ indicates that intrinsic and extrinsic geometry are strongly correlated for small λ, but as one passes through the peak region they become decorrelated. This is not particularly surprising, given that the action directly suppresses only extrinsic fluctuations. Note that RG calculations based on the Nambu-Goto action plus an extrinsic curvature term (with no dependence on an intrinsic metric) perturb about a background that is both intrinsically and extrinsically flat. Given the observed decorrelation between intrinisic

Figure 24: The intrinsic curvature $|\mathcal{R}|$ defined in (74) , plotted as in Fig. 16

and extrinsic geometry, we would not anticipate that this background appears in the low- temperature limit of the model which we simulate.

In Fig. 27 we plot the expectation value of the maximum coordination number, which has non-trivial scaling behavior. In Fig. 28 we give its scaling exponent, defined analogously to the exponent we have exhibited for the gyration radius. In the pseudo-critical region q_{max} scales (for our 3 lattice sizes) as a power, with an exponent close to 0.1; we do not know if this scaling is meaningful.

We will discuss here correlation times for different quantities . As we already pointed out correlation times become very large in the large λ region. In agreement with ref. [42] (see their Fig. 1) we do not see any increase of the correlation times close to the pseudo-critical point.

We will not present precise estimates of correlation times (exponential or integrated)

Figure 25: The fluctuations of $|\mathcal{R}|$.

[71] – they are too large to get precise estimates. We will limit ourselves to a discussion of a few figures, which give quite a clear idea of what is happening. The comparison with Fig. 1 of ref. [42] cannot be very direct, since our action is different, and because their dynamics may be more effective than ours. Still, the comparison is quite puzzling, since we estimate and exhibit correlation times which are much (orders of magnitude) larger than the ones of [42]. Applying customary methods to estimate τ_{int} can lead to an underestimate of correlation times if more than one time scale is present (that does surely happen with our data if we integrate it on a window of reasonable size).

3.4 Autocorrelation Times

In Fig. 29 we plot S_E for $N = 144$, $\lambda = 1.4$, and in Fig. 30 the gyration radius for these values (with a different time scale). Clearly, the correlation time is at least of order

Figure 26: The intrinsic extrinsic curvature correlation, as defined in (75), plotted as in Fig. 16

40,000 sweeps in the first case and 100,000 sweeps in the second one. In Figs. 31, 32, we plot the same quantities for $\lambda = 1.5$. Here correlation times are larger, of order 50,000 steps for S_E and larger than 150,000 steps for R_G. In Figs. 33, 34 we draw the same plot on the largest lattice we study ($N = 576$) for $\lambda = 1.4$. Here we can see dramatic correlations, with times of at least 100,000 steps for S_E and of at least 1,000,000 steps for R_G.

In Fig. 35 we plot, for the same time history and on the same scale, both S_E and R_G. This figure shows a clear anticorrelation: larger surfaces are flatter and have smaller curvature (this is apparent in the region close to the 2000th step).

Figure 27: The average maximum coordination number of the surface vertices, $\max_i q_i$, plotted as in Fig. 16

Figure 28: The scaling exponent of $\max_i q_i$, plotted as in Fig. 16

Figure 29: S_E as a function of Monte Carlo time ($80,000$ steps) for $N = 144$, $\lambda = 1.4$.

Figure 30: R as a function of Monte Carlo time ($300,000$ steps) for $N = 144$, $\lambda = 1.4$.

Figure 31: S_E as a function of Monte Carlo time ($80,000$ steps) for $N = 144$, $\lambda = 1.5$.

Figure 32: R as a function of Monte Carlo time ($300,000$ steps) for $N = 144$, $\lambda = 1.5$.

Figure 33: S_E as a function of Monte Carlo time ($300,000$ steps) for $N = 576$, $\lambda = 1.4$.

Figure 34: R as a function of Monte Carlo time ($3,000,000$ steps) for $N = 576$, $\lambda = 1.4$.

Figure 35: S_E and R from the same Monte Carlo run, $N = 576$, $\lambda = 1.325$, $20,000$ steps.

4 Data Analysis And Results

4.1 Data Interpretations

Let us review the crux of our observations again. This model of crumpled surfaces appears to exhibit sharp crossover behavior in the region around $\lambda = 1.4$. The sharp growth in the gyration radius and the suppression of curvature fluctuations indicate that the normals acquire long-range correlations, up to the size of the systems we examine. Presumably the zero string tension measurement of [42] also shows that the disordered regime differs from the regime in which the surfaces are ordered (up to scale of the lattices that are simulated) by only a small shift in λ. This evidence might indicate the presence of a phase transition at this point. If so, it is very likely to be of order higher than 2 (or, rather implausibly, it could be second order with an extremely low negative specific heat exponent; our lattices are much too small for us to confidently extrapolate the value of the specific heat exponent as $N \to \infty$).

If the transition were higher order, the peak should exhibit a cusp, but we would need far more accurate data to detect this. The existence of this phase transition would then suggest the existence of a new continuum string theory, though many other issues would have to be resolved) to determine if such a theory is physically desirable.

There are other possible interpretations of our data. We need to consider the influence of finite-size effects, since the surfaces which we simulate are quite small, even smaller than one might naively assume because they are not intrinsically smooth. For instance, random surfaces characteristic of $D = 0$ gravity have a Hausdorff dimension of roughly $d_{intr} = 2.8$ [72, 73]; it has been predicted that surfaces embedded in 1 dimension have Hausdorff dimension $2 + \sqrt{2}$ [72]. Thus, for instance, if the surfaces in

our simulations had an intrinsic dimension of 3, they would have a linear size of fewer than 9 lattice spacings. Of course, our lattices are too small, by one or two orders of magnitude, to really exhibit a convincing fractal structure.

Perhaps the simplest alternative explanation for the presence of this peak is suggested by the arguments of Kroll and Gompper [55]. They argue that the peak occurs when the persistence length of the system approaches the size of the lattice ($\xi_p \sim N^{\frac{1}{d}}$). For couplings above this point, our simulations would simply be measuring finite size effects. For larger λ, fluctuations on a larger scale become more important, but when this scale is greater than the lattice size, these fluctuations are suppressed. Thus one might surmise that the specific heat will drop for large λ. (It clearly goes to zero for small λ because of the presence of the prefactor λ^2; the lattice implements a ultraviolet cutoff that freezes out very short-range fluctuations). As shown in section 4.2, we excluded this possibility with our latest measurements.

The one-loop renormalization group calculation (64) predicts that the persistence length grows as $\xi_p \sim \exp(C\lambda)$; C is inversely proportional to the leading coefficient of the beta function. We would expect that the peak position should shift to the right with increasing N in this scenario as

$$\lambda_{peak}(N') - \lambda_{peak}(N) = \frac{\ln(\frac{N'}{N})}{d_{intr}C}. \tag{80}$$

Quite a large value of C is needed to explain the rapid crossover; roughly values of $C \sim 10, d_{intr} \sim 3$ are more or less consistent with the magnitude of the peak shift and crossover width. The RG calculations using different forms of the action yield $C = \frac{4\pi}{3}$ (see equation 64), but this may not apply to the action we simulate.

This reasoning also indicates that the peak should widen as the lattice size increases;

we do not observe this at all. It seems plausible though that these arguments, based only on the leading term of the high lambda expansion, are too naive.

An alternative scenario, which builds on the ideas in the above paragraph, is suggested by the tantalizing similarities between the results of our fluid surface simulations and what has been observed for the $d = 4$ $SU(2)$ Lattice Gauge Theory [63] and for the $d = 2$ $O(3)$ model. Let us discuss the case of the $O(3)$ model.

The $O(3)$ model, which is asymptotically free, exhibits a specific heat peak near $\beta = 1.4$. This peak was first measured via Monte Carlo simulations by Colot [74]. It can also be obtained by differentiating the energy data measured by Shenker and Tobochnik [75, 76]. The origin of this peak is understood [76, 77]; it is due to the fluctuations of the sigma particle, a low-mass bound state of the massless $O(3)$ pions. The sigma induces short-range order, and contributes to the specific heat as a degree of freedom only at high temperatures (when the correlation length in the system becomes smaller than its inverse mass). The peak thus occurs at the beginning of the crossover regime, when the correlation length is several lattice spacings.

According to the low temperature expansion, the correlation length grows as $\xi \sim \exp(2\pi\beta)/\beta$. Thus one would expect a fairly rapid crossover in the $O(3)$ model; the correlation length should increase by roughly a factor of 9 when β is shifted by about .35. In fact, the presence of the sigma significantly modifies this low-temperature expansion result [77] in this intermediate regime, but does not qualitatively destroy the rapidity of the crossover. Indeed, despite heroic efforts, it has been impossible to extend computationally beyond this regime and precisely verify the asymptotic low-temperature relation for the correlation length [78, 79]. Such a crossover is indeed observed, though the histogramming is not so apparent that it is as dramatic as the crossover behavior observed

for fluid surfaces. To quantitatively compare the width of the crossover regimes for these two models it would be necessary to measure a correlation length (perhaps extracted from the normal-normal correlation function) in these random surface simulations.

The numerical simulations show a distinct peak in the specific heat which grows for small lattices and then saturates, just as we find in the model of a rigid string treated here. This may be seen in figure 36 where we have plotted the specific heat $C(\beta)$ for the two dimensional $O(3)$ model. It is very clear that the peak levels off quickly for $N \geq 100$ and that "β_c" is not increasing with the size of the lattice. Measurements of the asymptotic value of $C(\beta)$ have been reported in the past [74, 80]. The authors of [81, 82, 83] explain the peak as the excitation of an extra degree of freedom, the so-called σ-particle [84]. The would-be transition occurs when the mass of the σ-particle becomes comparable to the inverse correlation length of the $O(3)$ model. It may be that there is a similar interpretation of the observed peak of $C(\lambda)$. Also, the peak position shifts to the right as L grows, and then appears to stabilize for large L.

This is more or less what we observe in our simulations of fluid surfaces, on lattices of small size. We point out these similarities largely to emphasize that there does exist an asymptotically free theory (with low mass excitations) which exhibits crossover behavior qualitatively similar to that observed in our simulations.

The analogy is perhaps deeper, though, since the fluid surface action (with extrinsic curvature) in certain guises looks like a sigma model action. So, perhaps it would not be so surprising from this point of view to find a sigma particle in these theories perhaps associated with $(\hat{n}^2 - 1)$, in which $\hat{n}$ denotes the unit normal to our surfaces.

Another additional possibility is that fluctuations of the intrinsic geometry (the Liouville mode) are responsible for short-range order and contribute to the specific heat

peak.

4.2 A True Phase Transition?

Clearly the maximum of the specific heat curve C_{max} is effectively constant for surfaces with 576 or more nodes. The (pseudo)-critical coupling λ_c is also constant for $N = 576$ and above. With the present data we can definitely exclude the presence of a divergence in the specific heat. The growth of the specific heat peak observed on small lattices [38, 39, 40] does not reflect true asymptotic behavior. These results also invalidate the interpretation raised in the introduction [55, 59]. A one-loop renormalization group calculation shows that the persistence length ξ grows with bending rigidity λ as $\exp(\frac{3}{4\pi}\lambda)$. Equating ξ with the spatial extent of the lattice $N^{1/d_{in}}$, where d_{in} is the intrinsic Hausdorff dimension of the lattice, one would see that they become comparable at a coupling $\lambda_c \sim \frac{3}{4\pi d_{in}}\ln(N)$. In the continuum limit $N \to \infty$, λ_c diverges. Since, for reasonable values of d_{in}, we do not see the increase in λ_c with N predicted by the above relationship we can state with some confidence that the origin of the observed specific heat peak is not explained by the persistence length becoming comparable to the extent of the lattice.

It is still possible that there is a true continuous phase transition, the crumpling transition, occurring at λ_c. Assuming a continuous transition, a standard finite-size-scaling argument only tells us that $\omega = \frac{\alpha}{\nu d} < 0$, where α is the exponent governing the divergence of the specific heat, $C(\lambda) \sim |\lambda - \lambda_c|^{-\alpha}$, ν is the analogous exponent for the correlation length and d is the intrinsic Hausdorff dimension of the surface. In other words there may be a cusp singularity at λ_c as, for example, in the case of the superfluid (λ) transition in He^4 [85, 86], for which $\alpha = -0.0127 \pm 0.0026$. Since we do

not have a measurement of νd, which may even be rather large, we have no reliable idea of the exponent α itself. Generally speaking one finds that second order transitions on fixed lattices become higher order on dynamical lattices, as for example in the case of the $2d$-Ising model [87, 88]. Since there seems to be a 2nd order crumpling transition for non-self-avoiding tethered (fixed-triangulation) surfaces [89, 90, 91, 92], it would be consistent for the transition to be higher than 2nd order when the model is coupled to gravity.

4.3 Conclusions

We have thus explored the phase diagram of fluid random surfaces with extrinsic curvature. With lattice size up to 576 nodes, we were not able to determine if our model undergoes a phase (crumpling) transition at finite coupling. We have observed dramatic crossover behavior for particular observables in our Monte Carlo simulations, but on the other hand, the correlation times and certain finite-size effects do not behave as one would expect in the presence of a phase transition. With larger lattice up to 2304 nodes, we found that the extrinsic-curvature specific heat peak ceases to grow on lattices with more than 576 nodes and that the location of the peak λ_c also stabilizes. The evidence for a true crumpling transition is still weak. If we assume it exists we can say that the finite-size scaling exponent $\frac{\alpha}{\nu d}$ is very close to zero or negative. On the other hand our data does rule out the observed peak as being a finite-size artifact of the persistence length becoming comparable to the extent of the lattice.

The behavior of other lattice models also indicates that it is possible that we are observing the effects of finite-mass excitations on small lattices, rather than a phase transition. We hope that future work will clarify this murky state of affairs, to determine

if there indeed exists a crumpling transition for fluid surfaces.

All told our work gives only weak evidence for a continuum crumpling transition. The strongest evidence in favor of such a transition, at present, is the scaling behavior of the string tension and mass gap reported in [42]. This highlights the need for more extensive measurements on these important observables.

4.4 Future Work

There remains much to be done to clarify whether or not a crumpling transition occurs for a finite value of the extrinsic curvature coupling λ. It would be interesting (and probably a fair amount of work) to apply Wilson renormalization group techniques to the actual action (61) which we simulate, to determine the leading coefficient of the beta function. Additionally, perhaps a calculation of $1/D$ corrections to the large D computations already performed could unearth evidence of a sigma-type excitation in these theories (the effects of the sigma appear as $1/N$ corrections in the $O(N)$ model).

We also used the data to examine the behavior of complex zeroes (in complex λ space) of the partition function of our simulations [93]. It has been shown (in the case of $SU(2)$ lattice gauge theory) that such zeroes, when they are near but do not approach the real axis in the infinite volume limit, occur in theories which exhibit specific heat peaks with no associated phase transition [63]. Low temperature expansions also indicate that the $O(3)$ model susceptibility has a complex singularity near the real axis [94]– presumably this corresponds to a zero of the partition function and is a manifestation of the sigma.

Of course, simulations on large lattices, with better statistics, should also help us evaluate whether a crumpling transition exists. We could use parallel computers. However, as I will explain in chapter 5, the real parallelization of random surfaces program

is extremely hard. The alternative could be to find new algorithms, in order to evade the long auto-correlation times that have characterized our simulations so far.

Even if no such transition exists for finite λ, one could still attempt to study a continuum theory in the strong coupling limit, as is done for QCD, for instance. To do so, we would like to examine global quantities, such as masses extracted from normal-normal correlation functions, rather than just the local quantities (e.g., energy) that we have measured. Measuring these correlations requires a definition of distance on these triangulated lattices; the most successful definition of the metric is based on the propagation of massive particles (via inversion of the Laplacian) on these lattices [95].

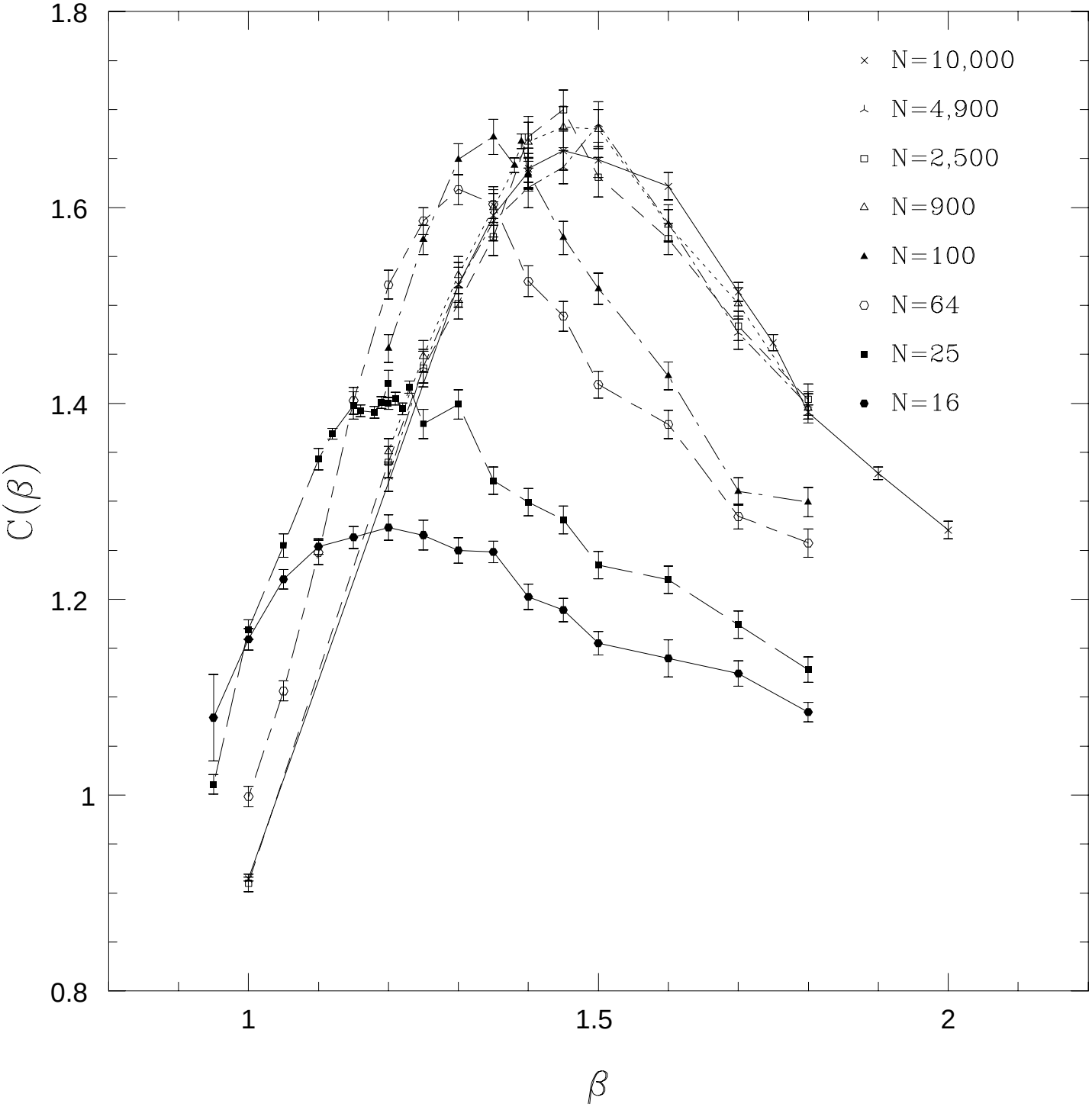

Figure 36: The specific heat $C(\beta)$ of the two-dimensional $O(3)$ non-linear sigma model as a function of β for lattice volumes $N = 16, 25, 64, 100, 900, 2,500, 4,900$ and $10,000$. The peak saturates quickly for $N \geq 100$ and "β_c" does not increase with the volume.

5 Computer Implementations and Issues in Computational Science

As we have seen in the previous chapters, Monte Carlo simulations of random surfaces require enormous computer power. The use of high performance computers is central to our goal of understanding random geometry in many guises. With the most recent advances in the field of supercomputing and massively parallel computing, computation can thus lead to a vastly improved understanding of the fundamental interactions of nature from gauge theories of the strong and electro-weak interaction to the most elusive interaction in the quantum domain so far – gravity.

However, with dramatic changes in technology ahead, we have to ask ourself the following question: how do we approach the problem of high-performance architecture design and high-performance engineering and scientific computing? For example, the new technology makes feasible massive parallelism. How much additional effort should be invested in increasing the performance on a single processor before we seek higher levels of performance on multiple processors? There are no simple answer to these questions. The first is always essential to the latter. We need a combination of solutions, and what we choose almost certainly will be application dependent, since at this stage we have not yet constructed a general machine that would be equally effective for all high performance applications.

In the past, we have seen many different techniques used in hardware to improve performance for the individual processor, such things as instruction buffers, cache memories, pipelined execution and RISC computer architecture [96, 97], have appeared in many commercial machine implementations. And people created parallel computers to

overcome the current limits of technology in order to achieve even faster speed. However, can an application easily and effectively utilize these hardware capabilities? This is always a complicated story. In terms of software issues, it has to address to both the level of compiler design and the level of user's applications and algorithms. As computer clock speed keeps increasing, the issue of how well applications can utilize each computer's cycle becomes even more important, since many numerical applications are so demanding of computational cycles and call for sophisticated floating-point processors in the architecture [98]. Parallel computers bring even bigger challenge to software design and call for innovations of completely new algorithms.

We also note that with the increasing power of high performance computers, the power of graphics system has been increasing and made it possible for the scientific and engineering communities to gain new insight into their disciplines. Both the size of scientific problems and the quantity of data generated present a great challenge to us: namely how to understand the results of their computations. Solving these problems interactively is essential and requires significant computation and graphics power.

5.1 The Challenge of Random Surface Simulations to Computation

The computer science challenges for computationally intensive application with complex data structure, such as random surface simulations, are centered on how to improve individual processor performance and how to decompose the problem and map it onto parallel processors.

The original code we used was written by Baillie, Johnston and Williams [99], and was based on Distributed Irregular Mesh Environment(DIME), a general software package for handling dynamically triangulated meshes [100]. Dynamic unstructured meshes

of this type are also used for finite element simulations and computational fluid dynamics. This code was written in C, with the mesh represented as a linked list. The generality of the code meant that it used a lot of unnecessarily complicated data structures, which increased the memory requirements and decreased the efficiency of the program.

In order to improve the performance of the code, we rewrote the program from scratch without using DIME. The new program was written in Fortran, in order to make it easier to parallelize by using Fortran 90, Fortran D, or CM Fortran. The new code was simpler and more specialized. Consequently it ran approximately 8 times faster than the previous code on the IBM RS/6000.

However this new code still ran very slowly on certain machines, especially the Intel Touchstone Delta, which uses Intel *i*860 processors. For this code we realized only about 1 MFlop on the *i*860, which is theoretically an 80 MFlop processor for 32-bit operations.

We believed that we could substantially improve the performance of the program on the *i*860 and the IBM RS/6000 by careful optimization of the code, and in particular by making better use of the data cache. A floating point operation can be done in a single cycle on modern RISC processors, however access to data not in the cache may take many cycles. To get near optimal performance from these processors we therefore need to be very careful to structure the code in such a way as to make best use of the data cache, by keeping data as local as possible. This is a non-trivial problem for applications with dynamic data structures such as the simulation of dynamically triangulated random surfaces. This is very similar to problems of data locality on parallel processors, where there is an even greater penalty in accessing data which is non-local. For programs such as parallel random surface simulation, dynamic domain decomposition is required for

both load balancing and efficient access to data. A similar approach can also be applied to optimize sequential programs on processors with a hierarchical memory structure.

Unstructured numerical applications [100, 101] have been one of the most computationally demanding areas and have the most complicated but the richest data structures. Naturally people like to use them to evaluate the overall performance of a computer system in dealing irregular problems for both hardware and software aspects [101, 102].

The Monte Carlo simulation of dynamically triangulated random surfaces happens to be one of the most suitable applications to be investigated for state-of-the-art computers. It has very irregular with complicated data structures (see Fig. 37) and demands very intensive floating point calculations. The computation is very time consuming. For example, running on HP 9000 24 hours a day, the computation takes about 3 months to have one statistically meaningful physical data point for a mesh with 1152 nodes, even though our code is the fastest sequential code for dynamical random surface simulation.

We had done the largest simulations and most accurate measurements of physically interesting observables so far on dynamically triangulated random surfaces with 2304 nodes, which gave us more meaningful physical results. However, correlation length for most of observables is a major problem. The simulation takes hundreds of thousands of sweeps of updating the mesh to yield one statistically independent configuration. The existence order and critical exponents of this *crumpling transition* are still uncertain. That is the driving force for us to speed up the code by optimizing it, looking new algorithm and using parallel computer. The study of computational behavior of the random surface simulation on high performance computer also provides a basis for ongoing High Performance Fortran compiler support [102].

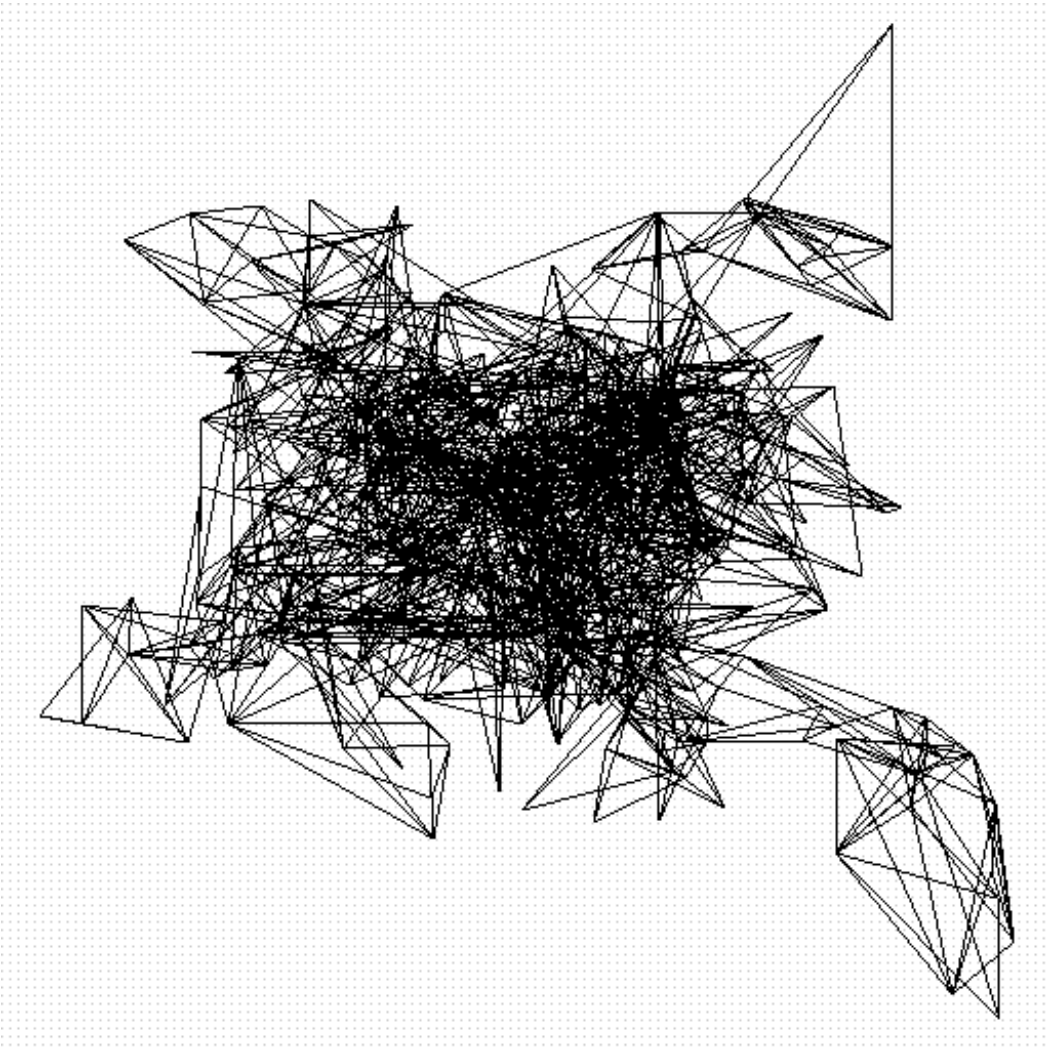

Figure 37: Wireframes for 144 nodes at $\lambda = 0.8$ (crumple phase)

5.2 The Random Surfaces Fortran Program

The program used for the Monte Carlo simulation of string theories was written in Fortran. The following outlines the crucial part of main program.

```
program main-outline


CALL mesh-set-up
CALL global-data-initialization


DO i = 1, n_measurements
  DO j = 1, n_sweeps
    CALL update-nodes
    CALL update-links
  ENDDO
  CALL measurements
ENDDO


END
```

The program first constructs a initial mesh. Fig. 38 shows a initial mesh with torus topology. We can see that one sweep consists of two parts : *update-nodes* and *update-links* . The subroutine *update-nodes* is used to update each node's position of mesh in the three dimensional space Fig. 8. The subroutine *update-links* is then used to change the connectivity of the mesh, i.e., to try to flip each link between nodes in the mesh Fig. 14. The Metropolis algorithm is used to update both the nodes and links of the

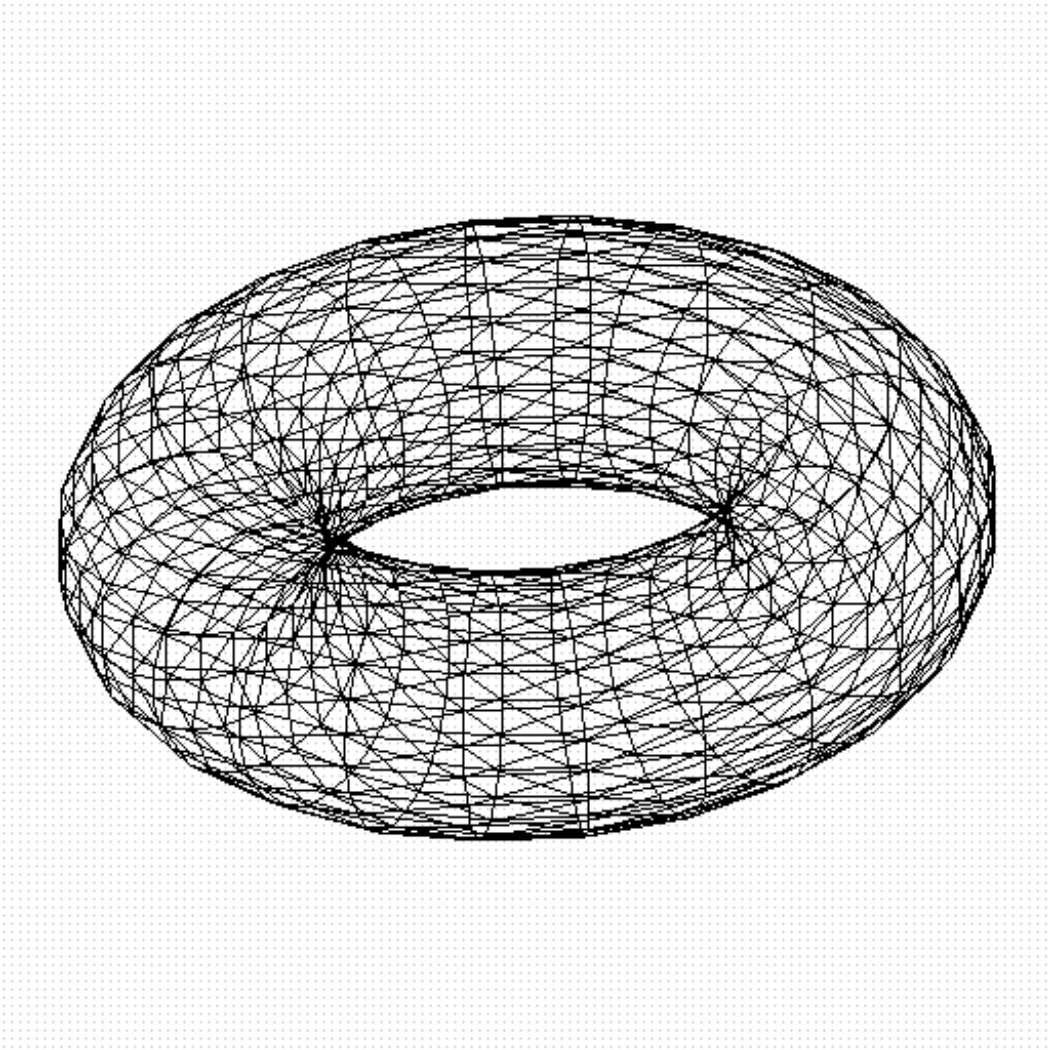

Figure 38: Wireframes of a torus with 144 nodes

surface. As the simulation proceeds, the geometrical data and the physical observable associated with nodes and edges have to be recalculated.

The node update part loops over all nodes. In the loop it randomly picks an attempted position for the current node and calculates the energy change of the system between the old position and the attempted position Fig. 14. The Metropolis algorithm determines if the attempted position should really be used to update the node's position or not. The energy calculation is based on the length of the edges and the so called edge action, i.e. the dot-product of two normal direction vectors of two triangles along the common edge. This dot-product is actually calculated in terms of the two altitudes in the triangles along the common edge. The edge action calculation is very expensive since the calculation not only involves the data associated with all nearest neighbors but also the next nearest neighbors Fig. 39. However the node update is still less irregular in the sense that the connectivity is not changed. Therefore the data retrieving process from cache or main memory for the calculation of node update is still relatively smooth. It probably does not experience very frequent interruptions or jumps since hopefully most of the required data are laid out in memory consecutively, so useful data may stay in the cache for a while before moved out. With fixed connectivity in *update-nodes*, the number of usage of branching condition statements and pointer operations cause much less memory jumps than in subroutine *update-links* where the connectivity is changing as each link flips. Therefore *update-nodes* produces less cache miss and page fault than *update-links* . However, even in the case of update node, the mesh is still irregular. Each node may have different number of neighbors, and therefore a different number of edges. These complexities may cause load balance problems for later parallel computation.

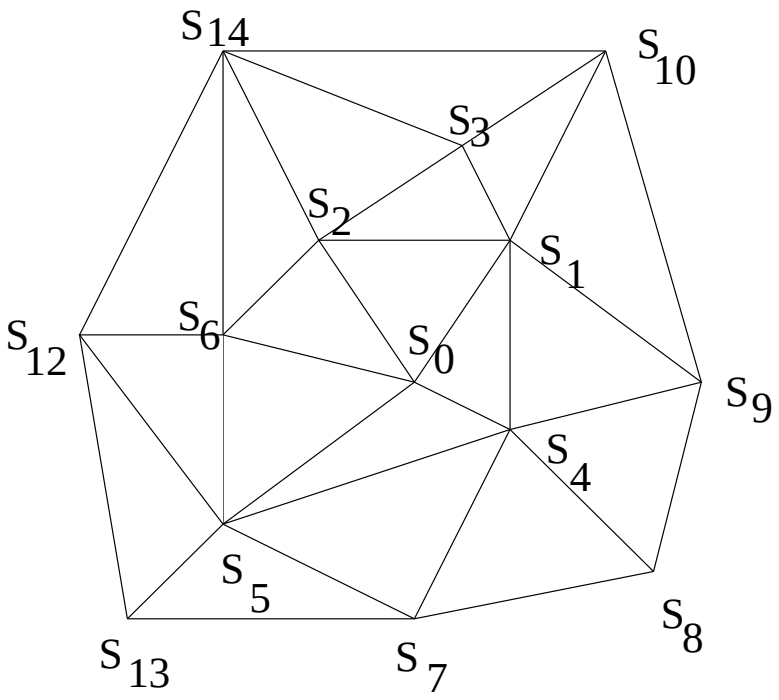

Figure 39: The nearest neighbors of node S_0: $S_1,\ S_2,\ S_6,\ S_5,\ S_4$ and the next nearest neighbors: $S_9,\ S_3,\ S_{14},\ S_{12},\ S_7$. To compute the edge action along edge S_1S_2, triangles $S_1S_3S_2$ and $S_0S_1S_2$ involved.

The subroutine *update-links* is used to randomly pick a link and attempt to flip it to another diagonal as in Fig. 8. The calculation of the energy change between the attempted configuration and old configuration is still similar to the one in *update-nodes* which uses the Metropolis algorithm. However at this time, not only the calculation is as expensive as in the *update-nodes* case, but also the data structure layout in the memory associated with the nodes and edges may become very irregular. One may have to search information throughout a big area of memory to calculate a observable since the pointers may point to different places as we add links to one node and remove links from others. The data associated with the changing edges are scattered all over the memory and logically closely related data information may move further apart in memory as the

links flip. Data locations are less predictable. Therefore, as the simulation proceeds, the data associated with the changing edge may be moved in and out from the cache more frequently, and the data required to calculate observables may be distributed over different pages in virtual memory space. Therefore the behavior of *update-links* part may appear to be more "violent" in terms of memory access and becomes more interesting.

All of these characteristics are very good features to be used as a probe to investigate the performance of application software, compiler and memory hierarchy. By monitoring the performance of different aspects in the random surface program we may gain some insights into many computational issues.

5.3 Optimization of the Random Surface Code for RISC

(1) Optimization and Memory Hierarchy

Our Fortran code for the random surface simulations ran very slowly on the Intel *i*860, and we were convinced that we could improve the speed of the code substantially for this and other modern RISC processors by some careful optimization.

We believed that the poor performance of the program on the *i*860 processor was probably due mainly to cache misses, since the off-cache memory access for this processor takes many cycles, whereas a floating point operation can be done in a single cycle. This is the case for most RISC processors, and can be a problem in utilizing these processors efficiently. We therefore attempted to understand why the program ran so slowly, and to try to optimize the program for RISC processors such as the *i*860 and the RS/6000 by investigating possible problems, in particular memory access.

Note that this is also an essential step in optimizing parallel programs, since we want to distribute data over processors in such a way that the memory accessed by any

processor is as local as possible, to minimize communication costs between processors. On a sequential machine we have a similar kind of memory hierarchy, with on-cache memory corresponding to on-processor memory for a parallel machine, and off-cache memory corresponding to memory on a different processor, which requires extra time to access.

Based on the idea of memory having a hierarchy structure, and the particular characteristics of certain RISC chips, we can outline a fairly systematic method for optimizing the program. These methods are quite general, and can be applied to any numerically intensive program run on a fast RISC processor. We have benchmarked the results of these optimization techniques on the Intel *i*860, IBM RS/6000, HP 7000, DECstation 5000, and Sun SPARC 1+.

(2) Optimizing Compilers and Basic Optimization Techniques

We first investigate what the compiler provides. Optimization flags with compilers provide local and global optimizations, and may perform some pipelining. A comparison of the speed of the program before and after compiler optimization for all the different machines is shown in Table 4.

For the *i*860, we found that the speed is increased by almost a factor of 3 just by using the NOIEEE compiler flag. This causes float and double divides, which are otherwise extremely slow since they are done in software, to be done using an inline divide algorithm. This gives substantial speed-up of functions such as division, square root, exponential and arccos which are used in the strings program. In a similar vein, the RS/6000 Fortran compiler has a flag (qrndsngl) which needs to be turned on in order to ensure strict adherence to the IEEE arithmetic standard (i.e. IEEE standard

arithmetic is *not* the default for this compiler). The strings code runs about 10% slower with this flag set.

The random number generator subroutine was used many times in the inner loop of the program, and each call to this subroutine has a function call stack overhead. The IBM and Intel compilers provide an inlining compiler flag, which allow calls to specified functions, or functions of less than a certain size, to be placed into the code (inlined) rather than using a function call. By using the inline operation on the random number generator we get about a 4% speed-up for the *i*860, but no noticeable speed-up for the RS/6000. The same results are obtained by manual inlining. There is a trade-off here, in that bad inline effects may actually decrease the performance of the code, by increasing the code size and thus increasing overhead due to the small size of the instruction cache. Compilers for other machines may do automatic inlining at a certain level of compiler optimization, and one should be careful to check that this does in fact increase and not decrease the performance of the program.

We know that current compiler technology has not yet matured to the point of automatically taking advantage of many of the features of modern RISC chips. Thus the user needs to carefully reconstruct the program and fully expose the availability of pipelining and data locality to the compiler, in order to take maximum advantage of the data cache.

Firstly, we must try to keep the most frequently used variables in the cache, and reuse them as much as possible while they are still there. This may require re-ordering parts of the code. We also tried to reduce unnecessarily large array sizes so that all, or at least most, of the working set data remains in the cache. Thus after the transient period for loading this data onto the cache, most of the memory access is to fast cache memory.

Even better is to keep data which is re-used in registers. This can be done explicitly by the programmer in C by using the *register* variable type. In Fortran we need to allocate such data to temporary locally declared variables, and hope the compiler takes advantage of this.

Secondly, in trying to optimize code, one usually thinks of an assignment as being fast and a multiplication as being slow, however this is not the case with modern RISC processors. On the *i*860, for example, a load operation takes two clock cycles and a store operation three cycles while add or multiply take just one. We should thus eliminate redundant assignments and minimize memory access, use constants rather than variables, and try to avoid data conversion.

Thirdly, we try to minimize the number of expensive calculations, such as division, square root, exponential and arccos, which all occur in the strings code. If a calculation is expensive and repeatedly used in several places, it may be possible to calculate it once and assign it to a global variable, so further calculations are reduced to using this "look-up table". We should also try to utilize the concurrent add and multiply operations. An addition combined with a multiplication will be performed in about the same time as a multiplication on most RISC processors.

Fourthly, branches (IF statements) and loops (DO or FOR statements) can have a large overhead. We should try to avoid IF statement since they slow down any pipelining. Optimizing loops is somewhat problematic. Using the DO loop is a very good way to utilize the instruction cache, since it allows a lot of code to reside in the generally small instruction cache space. In most processors the CPU can take care of the loop while the floating point unit deals with the operations in the loop. However there is still a substantial overhead in using a DO loop, so we should merge DO loops as much

as possible, and in cases where the loop contains very few instructions it should be "unrolled". This means that if the number of iterations of the loop is very small (for example, a loop over 3 dimensions), the loop should be eliminated and the instructions written out explicitly for each iteration (each dimension). If the number of iterations of the loop is large, say 100 iterations of a single line of code, then it should be unrolled into a loop over say 5 iterations of 20 lines of code. The optimal number of lines of code in the loop after such a split should be such that the loop just fits in the instruction cache. unrolling of loops should be done by the compiler, but this is not always the case.

Also, since data is usually loaded into the cache a few words at a time, we should try to keep the loop stride smaller than the length of this cache line to avoid possible "thrashing" of the cache (continual reading and writing of data to and from cache), and also increase the chance of pipelining multiple read and write operations since the working data will be in same cache line or nearby.

Another crucial point to note for multiple loops over elements of arrays of more than one dimension is that the loops must be ordered in such a way that the inner loop corresponds to the array index which changes fastest. In Fortran this is the first index, while in C it is the last index, so optimal loop ordering will depend on the language used.

(3) Program-Specific User Optimization

After implementing the generic optimization techniques given above, we turned to more problem-specific methods to speed up the code. Here we are still interested in minimizing memory accesses and maximizing cache usage, but now we are aiming to do this by studying the specifics of our particular problem, and finding ways to change the

algorithm so as to better match the data structures and data access to the hierarchical cache memory of the processor. By appropriately ordering the operations at each mesh point, and in particular the traversal of the mesh data structure, so as to maximize data locality, we realized quite substantial gains in performance.

Here we show a typical example. Initially, each node in the mesh has six nearest neighbor nodes, so it has six connecting links. Each link has a pointer *node_of_link* to point to its node and a pointer *link_next_link* to point to its neighbor links. Initially everything is in order (Fig. 40). As the mesh is changed during the simulation by flipping the links, the associativity between nodes and links and neighboring links changes. Links pointing to the same node may now not be located in contiguous sections of the memory, and the link pointer may not point to the nearest neighbor link among the links around a node (Fig. 41). That is the case in the original version of the code, for which the links around a node are not ordered after flipping. However, the calculation of physical quantities associated with each link requires the information at the nearest neighbor links around their boundary nodes. Since the links are not ordered, to figure out the left side link and right side link of a given link, the original code first looks for the left side and right side nodes S_1 and S_2 (Fig. 40, and then makes a comparison to see which link connects the node pair (S_0,S_1) and (S_0,S_2).

```
      ...
      s1 = node_left_to_link(this_link)
      s2 = node_right_to_link(this_link)
111   temp_link = start_link_of_node(s0)
112    if (node_of_link(temp_link) .eq. s2) then
          s0s1 = temp_link
```

Figure 40: Mesh connectivity information distribution at the beginning

Figure 41: Mesh connectivity information changes after flips

```
            goto 113
         else
            temp_link = link_next_link(temp_link)
            goto 112
         end if
  113    continue
         ...
```

Here we have changed the algorithm to improve data locality. At the time we flip the link, we re-order the links immediately, each link getting a pointer to its nearest neighbor link (Fig. 42). So we get its nearest neighbor links (S_0S_1, S_0S_2, etc.) basically for free.

```
         ...
         s0s1 = link_next_link(this_link)
         s0s2 = link_last_link(this_link)
         ...
```

By implementing this and making other similar changes, we achieved the biggest improvement in speed. It eliminates a lot of IF statements, pointer chain searching operations, and several unnecessary large arrays. It also provides much useful information for other calculations at no extra cost.

In general, it is much harder to exploit data locality in a dynamic mesh than a fixed mesh. Because the neighbors of nodes and links and their corresponding ordering are changing as the simulation continues, the neighboring physical memory locations may not reflect the neighbor of a link or a node, and vice versa. The optimization may

Figure 42: Mesh connectivity information re-order after flips

require a run-time preprocessing strategy to predict more precisely the data locality.

(4) Performance Comparisons

We have compared the performance of this code and the effect of these optimization techniques on a number of current RISC-based processors and workstations, in particular the Intel *i*860, IBM RS/6000, HP 7000, DECstation 5000, and Sun SPARC 1+. Table 4 shows the performance of the program on all these different machines for various system sizes. We show timings for the code before any optimization, with compiler optimization, and with optimization by the user involving rewriting parts of the code as outlined above.

It is interesting to note that the slower processors (the DEC and Sun machines) gain relatively little from these optimizations or compiler optimization, whereas the more powerful processors (IBM and HP) show substantial improvement. The biggest improvement is found for the *i*860, which is most likely due to the complexity of its architecture and the consequent difficulty in creating good compilers for this processor.

(5) Conclusions

From the study of irregular dynamical mesh program optimization, we have learned that we need to be careful to construct the code and the data structures so that they make most efficient use of the memory hierarchy characteristics and pipelining of modern RISC processors. Current compilers can provide good optimization, but do not recognize and exploit all the information on data locality and pipelining opportunities. Substantial gains in performance can be achieved if the user is prepared to restructure the code to help provide the compiler with this information, especially programs with adaptive, irregular data structures such as the random surfaces application.

Our results show that the dynamical part of the program needs to have more aggres-

	Unoptimized code	Optimized code
IBM RS/6000-550	3.8	3.7
	6.9	8.2
HP 7000	4.1	3.5
	11.7	12.1
Intel i860	1.5	3.0
	2.8	4.7
DEC 5000	2.5	2.6
	2.5	2.6
Sun SPARC 1+	1.0	1.0
	1.3	1.5

Table 4: Comparison in performance for random surface strings code on a number of different workstations. These are normalized so that the slowest is 1. It is difficult to obtain an exact flop count for this code, due to the use of intrinsic functions, however the numbers given here are also very close (probably within 10%) to the MFlop rate for this code. The two values presented are for Fortran codes with (bottom) and without (top) compiler optimization.

sive code re-arrangement or even algorithm modification in order to achieve a relatively big performance improvement. Simply using the basic general optimization techniques does not have real effect on the dynamical part although it does speed up the static part. That means the compiler needs to be more intelligent so that it can look deeper into the program and extract more information about data association if the compiler wants to do this level optimization. That is one issue that High Performance Fortran will eventually address. Dynamical run-time techniques have to be used to explore more precisely the behavior of data locality. we are trying to extract more general strategies from ours and other similar type of irregular applications to support the High Performance Fortran development. This study is a pre-investigation for the on-going development of the High Performance Fortran compiler by Syracuse University, Rice University and University of Maryland.

The approaches we used here can be applied to improve the performance on almost all major types of RISC processors. These data locality techniques can also be used by data parallel compilers such as those for High Performance Fortran to allow these languages to outperform conventional Fortran even on sequential processors. Thus the High Performance Fortran not only achieves the parallelism but also exploits the capacity of each individual processor, especially for very irregular problems.

5.4 Parallelization of Random Surafces Program – on Distributed Network, SIMD & MIMD

5.4.1 Independent Parallelsm

Monte Carlo simulation approximates a large or infinite sum or integral over all possible states of the system by a finite sum over representative states of the system (usually

called *configurations*). This is a statistical process, with a corresponding statistical error which is proportional to the square root of the number of independent states sampled. Monte Carlo simulations therefore have a large amount of trivial parallelism, in that different processors can be generating different configurations completely independently, by using different starting configurations and different random numbers. These independent results can then be combined to reduce the statistical error in the simulation. As long as the system size is small enough so that each processor can hold all the data for a simulation, this procedure will give perfect speedup over a MIMD parallel computer or an array of workstations.

We have used this method of trivial parallelization running independent random surface simulations on networks of workstations and on MIMD parallel computers such as the Intel Touchstone Delta and the nCUBE/2. These runs used different parameter values or different meshes with different random numbers for each simulation. This kind of "job level" parallelism is common to many kinds of simulations in science and engineering. Usually simulation involves studying changes in the system as parameters are varied. Sometimes the choice of new parameters will depend on the results of the simulation with the current parameters, so this is a sequential process. However in many cases one would like to know results for a large number of parameter values which can all be run independently.

This approach is ideally suited to very coarse-grained parallel machines, and especially for distributed computing over networks of workstations. There are at least three methods of invoking job level parallelism on distributed networks, which we discuss below. We have done our physics production runs with the strings program using the first two methods, neither of which we found to be satisfactory. This has led us to think

of programming paradigms and systems software which would be ideal for this type of physics application, and this is what we propose as the third method.

(1) The Brute Force Approach

The simplest but most tedious method is just to log on to any available machines in the network and submit a different job to each of these machines. This method is too time consuming to use on any more than a handful of workstations. It would be possible to set up a shell program to remotely run these jobs on different processors, however deciding which machines to use generally requires checking that the machine is not already in use. Physicists and other scientists who might wish to use networked workstations in this way usually do not have the systems programming knowledge to implement such a method.

(2) Portable Parallel Software

Job level parallelism can be used on distributed computing networks and MIMD parallel computers using parallel software. It is easy to set up parallel code to do this, since only data input and output need to be managed, since all computation is sequential and requires no communication. The problem with this method is that although the programs are run independently, they generally require synchronization every time data is input or output. If results are output frequently, this communication may cause unacceptable performance degradation. Synchronization overhead can be reduced by altering the code slightly to buffer data on each processor and thus output larger blocks of data less frequently. This was not required for the strings code, which performed at virtually 100% efficiency when distributed across the RS/6000 network, even for the smallest system sizes used.

Synchronization during data output can also lead to performance problems if one or more of the processors in the network is slower than the others, perhaps because it is also running other jobs. This can be avoided by writing the data from each processor into separate files which can be accessed asynchronously. This can be done in Express, for example. This approach allows all the processors to run completely independently.

We still have the problem of "load balancing", in the sense that some machines in the network may be loaded with other jobs, and thus take much longer to finish their work. It would be helpful if there were some kind of dynamic load balancing functionality, whereby if a machine became loaded with another job, then the process being run under Express would be migrated to another unloaded machine in the network, if one was available.

(3) A Transparent Network

It would be preferable to be able to run multiple jobs over the network without having to deal with parallel software (such as Express) at all. Ideally, one would like to be able to just set up a standard Unix runfile which submitted multiple jobs, with different input and output files, to the *network*, rather than to a particular workstation. Some clever systems software would then distribute these jobs to whichever machines on the network had the smallest load and were not being used interactively. Again, some load balancing software would be required, so that if an interactive session was started on a workstation, the job would be migrated to another workstation. This would all be transparent to the user, who would just submit jobs to the network and not care where they were run.

This works very well for relatively small systems, however for large systems we

strike a snag related to the problem of critical slowing down, which we have seen is a major problem for Monte Carlo simulations of spin models near phase transitions. Random surface simulations also suffer badly from extremely long correlation length. A consequence of this is that the *equilibration time* grows rapidly with the size of the system. This is the time required for the Monte Carlo algorithm to move the system from the initial state, which can be any configuration, to an uncorrelated state representative of the system being simulated, i.e. with a particular temperature or system parameter (in the case of string theories, this is the extrinsic curvature parameter λ). For trivial parallelism, each processor has to run for at least the equilibration time before it can start producing usable data, so this is effectively wasted time. For large systems, this time may be so great that in practice it is not feasible to equilibrate the system on a single processor, whereas a system being simulated in parallel over many processors may be equilibrated and produce useful data in a reasonable time period.

5.4.2 Parallel Random Surfaces

Have seen the problem with independent parallelsm in the previous section, we also started to work on parallelizing the random surface code on SIMD machines such as the CM-2 and MIMD machines such as the CM-5, the Intel Delta, and the new IBM RISC-based parallel machine. Since this application involves a dynamic irregular mesh, we would expect that SIMD architectures would fare poorly on this problem. However they may do reasonably well on problems with fixed triangulated surfaces which do not change during the simulation.

Efficient parallelization of a dynamically triangulated random surface is a very difficult problem even on MIMD machines, since dynamic load balancing will be required.

This is because the mesh changes throughout the simulation, so that the neighbors of a point, which will originally reside on the same processor or a neighboring processor using standard domain decomposition, will change so that they may end up on distant processors, thus increasing the communication cost. Also the irregularity of the mesh means that it is not simple to determine which points can be updated in parallel, as it is for the regular fixed lattices we use for the simulation of spin models.

In DTRS simulations, both the vertex and edge update operations are generally done using the Metropolis Monte Carlo algorithm [31, 30]. The update of the variables on each vertex depends only on the values at neighboring vertices. In general this will also depend on next-nearest neighbor vertices, but for simplicity we will consider only the nearest neighbor problem. The general case is a simple extension of this problem. The Metropolis algorithm requires that dependent (neighboring in this case) values cannot be updated simultaneously (i.e. in parallel).

If we describe the mesh in terms of an undirected graph with vertices and edges, then the Monte Carlo update of the mesh is achieved using two basic operations: the update (or "flip") of an edge of the graph, which changes the structure of the graph (see Fig. 8); and the update of the variables associated with a vertex of the graph, which, for example, might be the position of the vertex in the embedding space, in which case the update is just a "move" of the vertex (see Figure 14). These two updates are sufficient to generate all possible triangulated meshes.

This is the well-known *graph coloring* problem. An *undirected graph* G is a set of vertices V and a set of edges E. The edges are of the form (i, j) where $i, j \epsilon V$. A *coloring* of a graph G is a mapping $c : V \rightarrow \{1, 2, \ldots, s\}$ such that $c(i) \neq c(j)$ for all edges $(i, j) \epsilon E$. $c(i)$ is referred to as the *color* of vertex i. Vertices i and j are said to be

neighbors if $(i,j) \epsilon E$. The number of vertices is denoted by V.

Note that for the DTRS simulations, we are updating both the vertices and the edges, so for a parallel implementation we need a vertex coloring and an edge coloring. We will concentrate solely on the problem of coloring the vertices of a graph.

To discretize the continuum world as a triangulated mesh is used in many simulations in computational science. This is particularly common for partial differential equation (PDE) solvers [100, 103]. These meshes are irregular and often adaptive, that is, they change during the course of the simulation [104]. For example, in computational fluid dynamics, the simulation of airflow over an airplane would use a finer mesh in turbulent regions and a coarser mesh in regions of laminar flow. The positions of these regions are not known initially, but must be identified during the course of the simulation.

Here we introduce some parallel graph coloring algorithms based on well-known sequential heuristic algorithms, and compare them with some existing parallel algorithms. These algorithms are implemented on both SIMD and MIMD parallel architectures and tested for speed, efficiency, and quality (the average number of colors required) for coloring random triangulated meshes and graphs from sparse matrix problems.

Since this problem is so challenging to parallelize, we expect that the implementation of a parallel algorithm for this dynamic irregular mesh problem will also provide input for new parallel languages and compilers, such as Fortran D and High Performance Fortran.

We would like to speed up the graph coloring part of these algorithms by doing the coloring in parallel. However our goal is to reduce the run-time for the whole computation. There is a trade-off here between the time spent in coloring the graph and in updating the edges and vertices of the graph. For an adaptive grid PDE solver, many

updates will generally occur between adaptive refinements of the graph which require a new graph coloring. In this case the percentage of the time spent in coloring the graph is very small, so it is worth spending more time to get a better coloring, which should improve the parallelism and reduce the update time. However for a DTRS simulation, every iteration involves an edge update, which changes the structure of the underlying graph, so the graph must be re-colored after every iteration. The graph coloring could therefore provide a substantial overhead unless it is much faster than the update time. In this case we are mainly interested in speed, and may be willing to make do with a good coloring, rather than a better coloring which takes much longer. We are therefore interested in studying and comparing a variety of parallel graph coloring algorithms.

For most of the applications that requires graph coloring, finding a good graph coloring with a small number of colors is only part of the problem. We must also be concerned with load balancing, since each coloring is followed by an update step in which the variables associated with vertices of the same color are updated in parallel. Thus we need also to ensure that the number of vertices of each color on every processor is approximately the same, in order to obtain good load balance in the update step. Our goal is therefore not just to obtain a good coloring, but to obtain a *balanced* coloring, that is, to minimize the number of colors required taking into account the load balance constraint, that the number of vertices of each color be approximately the same on every processor. It may be advantageous to make do with a larger number of colors if this makes the load more evenly balanced.

We concentrated on the first of these two goals, *optimal* graph coloring and suggest how the algorithms presented may be modified to achieve *balanced* graph coloring.

I did not have enough time to continue this investigation. Several other people

in NPAC went on and implemented the parallization of random surfaces, using the action with only Guassian term. They found The LDF algorithm appears to perform exceedingly well in both architectures. The processing time required remains lower compared to many of the other algorithms even as the size of the problem grows larger. In particular the SDL and MIS algorithms require much more communication in each pass, which would account for their poor performance at large problem sizes and with more parallel processors. Even noting that the communication required by the LDF algorithm is equivalent to that required by the J–P algorithm, it consistently performs better. The LDF algorithm consistently achieves between 5 and 6 colors on planar graphs, which is excellent considering the NP–Hard nature of the optimal 4-coloring. The LDF algorithm is fairly easy to implement and to understand, which makes it relatively easy to incorporate into many parallel architecture codes that may be written at some point in the future.

For some algorithms, it may be important that there is no bias towards updating certain vertices before others. This is known as the requirement for *detailed balance* [31, 105]. For example, the LDF and SDL algorithms will both tend to color large-degree vertices first, assigning them as a small number "color". If the vertices are updated in color order, then there will be a bias towards updating these vertices first. This problem can be avoided by picking the color sets to be updated in random order. With the Jones-Plassmann and MIS algorithms, this problem does not arise because there is no bias towards a particular set of vertices.

For applications such as DTRS, which require regular re-coloring of the graph, the LDF algorithm is probably the best to use, since it takes the same time as J-P, but requires fewer colors. For applications such as PDEs, the SDL algorithm, which takes

longer but uses fewer colors, may well be preferable since the coloring is performed once only.

A further refinement of the coloring algorithm is a balanced coloring, requiring that the roughly equal numbers of vertices have each color. Having a small number of vertices of any one color means that there is not much parallelism to be exploited in the update step for the parallel application (such as a PDE solver or random surface simulation) that is using the results of the graph coloring. A balanced distribution of colors makes it easier to load balance the work of updating and effectively exploit a parallel machine. A simple modification to the sequential versions of the algorithms described here will achieve a balance of colors. Instead of picking the smallest available color, one picks the least used of all available colors [106]. This idea can be adapted readily to a MIMD implementation; each processor chooses from the least used color in its local patch of vertices. In a SIMD implementation this is not possible because each vertex knows only the colors of its neighbors. An alternative strategy in this case would be to pick a color at random from the set of legal colors for a vertex. This could result in a slight increase in the average number of colors required, however this cost should be outweighed by the improved load balance. No experiments with color-balancing were performed.

And they also disappointedly found no speedup with the number of processors used on the iPSC/860. One reason for this may be that the algorithms as coded sent the set of weights for the whole graph to every processor and the set of colors for the whole graph to every processor.

Much pioneer work has been done in parallelization of random surfaces. However, we still have lots of more work to do. We have not had a parallel random surface parallel program. If we add in the extrinsic curvature term in the action in the parallel program,

things would become even more complex.

5.5 Visualization of Monte Carlo Simulation in 3 Dimensions

Graphics and visualization are increasingly becoming indispensable tools for analyzing and understanding large data sets and complex physical problems. Many of our physics applications involve complex and irregular data structures, and visualization tools are extremely useful for analyzing and understanding the physical system and the complex algorithms used to simulate the system. Graphical tools can also be a useful aid in debugging such algorithms. The IBM RS/6000 workstation provides a powerful graphics capability which we have used extensively in our studies of physics applications and parallel algorithms.

The machine with 24-bit color also is licensed for the Advanced Visualization System (AVS) from Stardent and Mathematica from Wolfram Research.

Graphics and visualization are extremely useful in the study of Monte Carlo simulation of spin models, on many different levels. They can aid the understanding of the physical system, the algorithms used to study the physical system, and the implementation and debugging of these algorithms.

(1) Visualization Using AVS

Two dimensional dynamically triangulated random surfaces consisting of hundreds of points embedded in three dimensional space-time are very difficult to picture without some sophisticated visualization tools. For our research, we are particularly interested in seeing how the structure of the surface changes from smooth to crumpled as we change the curvature parameter, and the variation in shape of the surfaces which are typical of particular parameter values. We also wish to see how rapidly the surface changes

during the simulation, and how certain structures, such as large spikes, can be created and take a long time to disappear, which contributes to the problem of critical slowing down. New Monte Carlo algorithms which can reduce critical slowing down in random surface simulations is an area we have been studying, and plan to investigate more in the future. As we have seen with spin models, visualization is extremely useful for the study of such algorithms.

Each 2 dimensional triangular cells cell is defined by vertices and connecting edges. We have used the Application Visualization System (AVS), since is particularly powerful at handling 3D images, especially for unstructured data. It also has a very simple and powerful user interface, which allows a complex visualization program to be constructed merely by connecting separate modules using an interactive graphical environment. AVS also allows the user to interactively adjust lighting, shading, coloring, and orientation of the image. Another beneficial feature is that it is available for many different systems, including Silicon Graphics, Sun, DEC and IBM workstations.

We have written an interface module to read in a file containing the random surface data, and convert it into a format which is understandable by AVS. AVS then does the rendering and display of the surface. Rendering, rotating, and zooming this complex 3-D image under AVS running on Sun or DEC platforms can be very slow, however with the IBM RS/6000 these operations are done very quickly, especially when special graphics hardware is utilized.

We used different color maps to reflect different properties of the surface, such as curvature, convexity, etc. We also used Guraud shading, which interpolates the color between nodes of the surface.

Figures 43, 44, 45, 46 and 47 show several configurations of a torus with 2304 nodes

at different λ, ranging from 0.3 to 2.0. We can see the surface changes from a crumpled one to a smooth one as λ increases.

Here the colors represent the curvature of the surface, with blue corresponding to smooth sections of the surface, and red representing sharply peaked elements. Concave sections of the surface are shown by the addition of green to the color of a vertex, giving yellow, green, or orange hues.

(2) Animation

We also developed a module which enables us to visualize the system by animating the simulation, rather than just displaying individual meshes. This is done by running the random surface program on the IBM RS/6000 workstation, and using a Unix pipe to pass the data to an AVS module which we have written to take the data and convert it into AVS format for rendering and real-time display. We have also included a pause option to freeze the image at any time. In order to display the results of the simulation in animated form, we need the power of the IBM RS/6000 to both run the simulation fast enough, and to do the rendering of the complex surfaces extremely rapidly. The IBM RS/6000 is not only much faster than the DEC or Sun workstations which also support AVS, but has a much larger range of colors, and can use hardware rendering to make the animation much faster and smoother.

We made a video of this animation, which is invaluable in helping to explain random surface simulations when making presentations of our work at conferences and seminars. This animation capability allows us to better study the dynamics of the system, in particular the change in the surface when passing through the crumpling transition. It is also a great help in trying to understand the causes of critical slowing down, and

should help us in our attempts to find new and better Monte Carlo algorithms.

(3) Using AVS to Display Parallel Simulations on a Network

The most recent version of AVS provides a parallel module execution function. We are therefore able to visualize different simulations running on different machines in the network in parallel. Here the amount of data being received, and the amount of rendering required for multiple images, is so large that we need a very powerful machine such as the IBM RS/6000 for this to be useful. An example of the RS/6000 monitor when running AVS in this fashion is shown in Fig. 48.

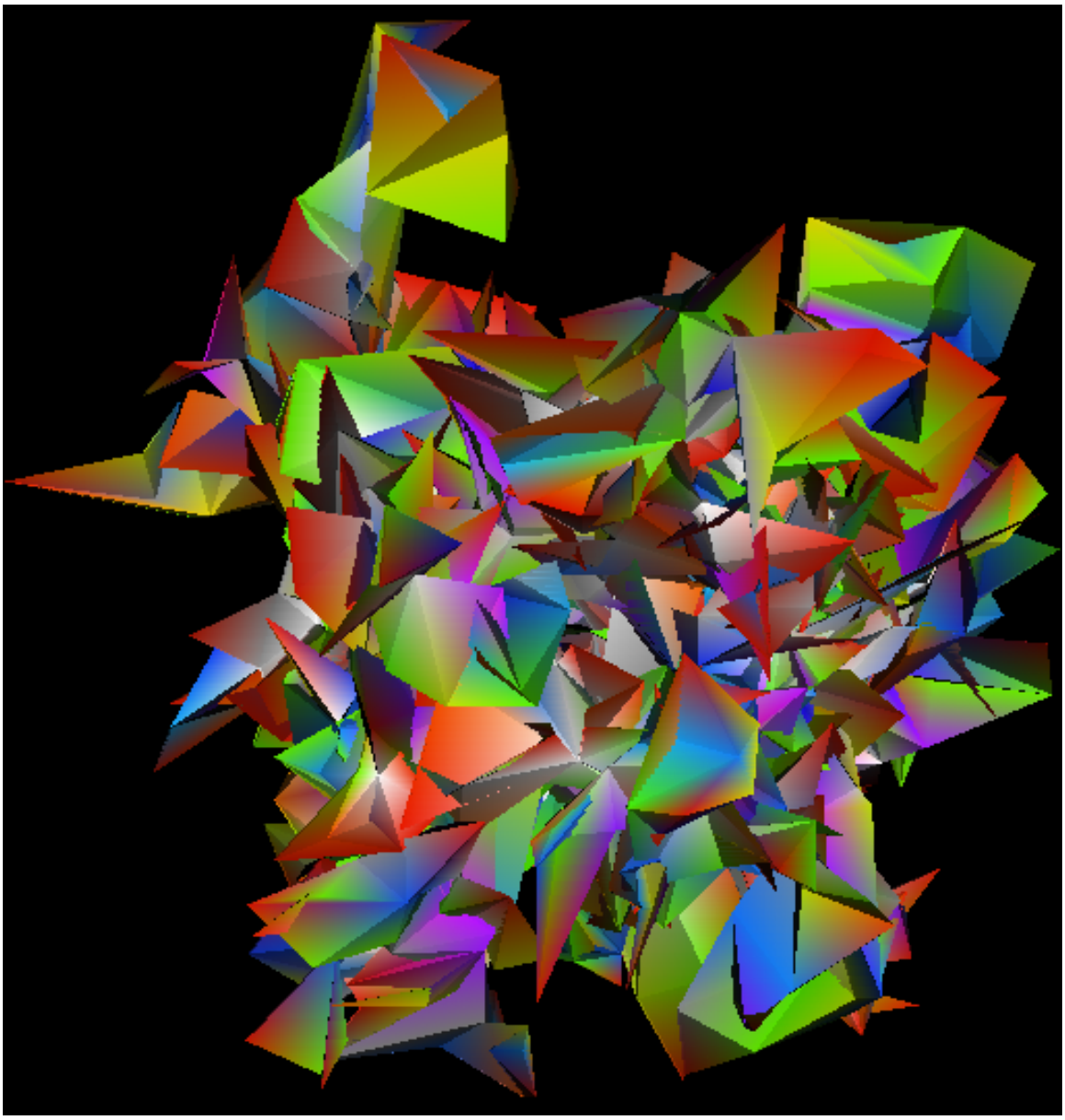

Figure 43: A configuration with 2304 nodes at $\lambda = 0.8$.

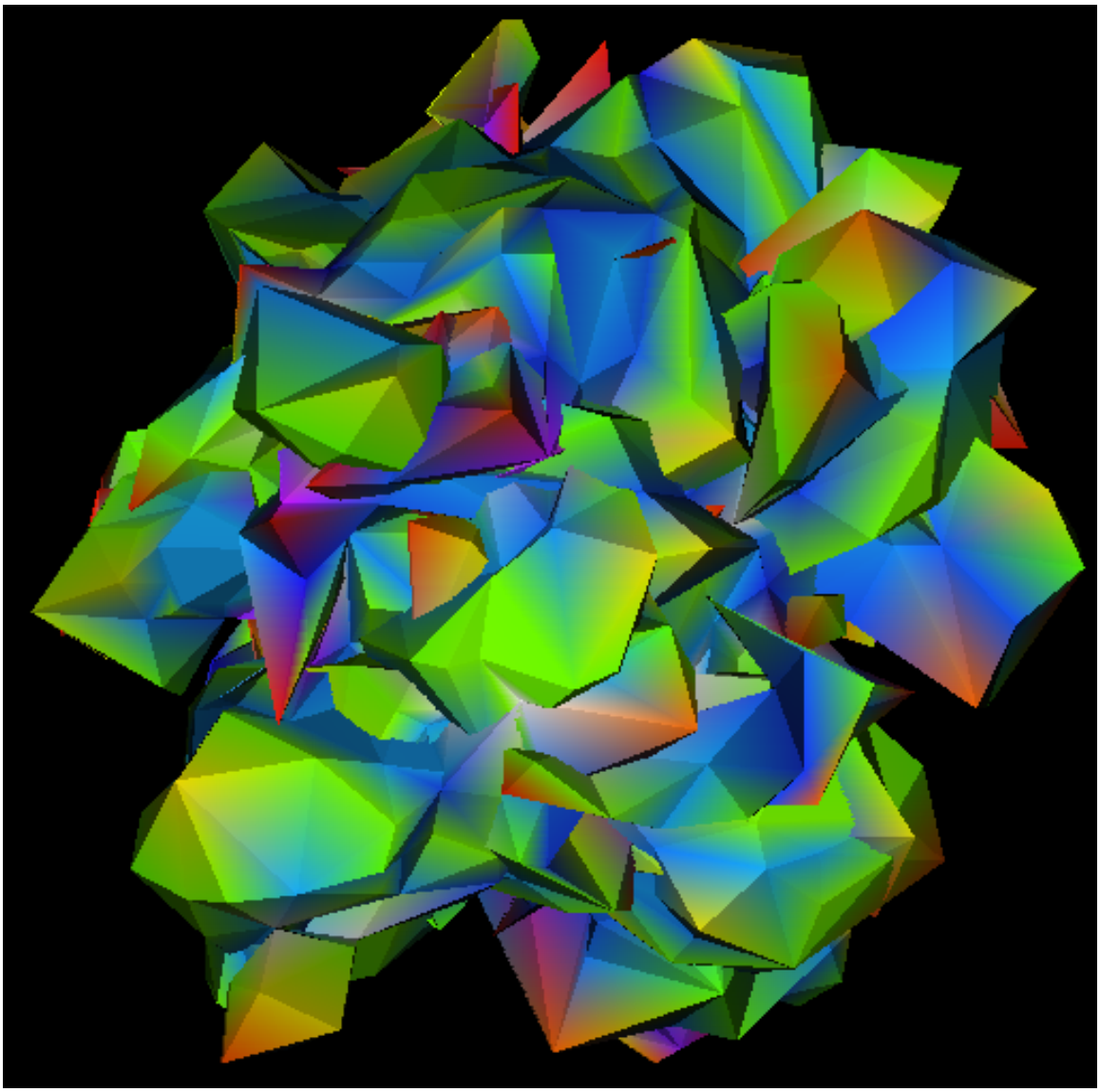

Figure 44: A configuration with 2304 nodes at $\lambda = 1.4$.

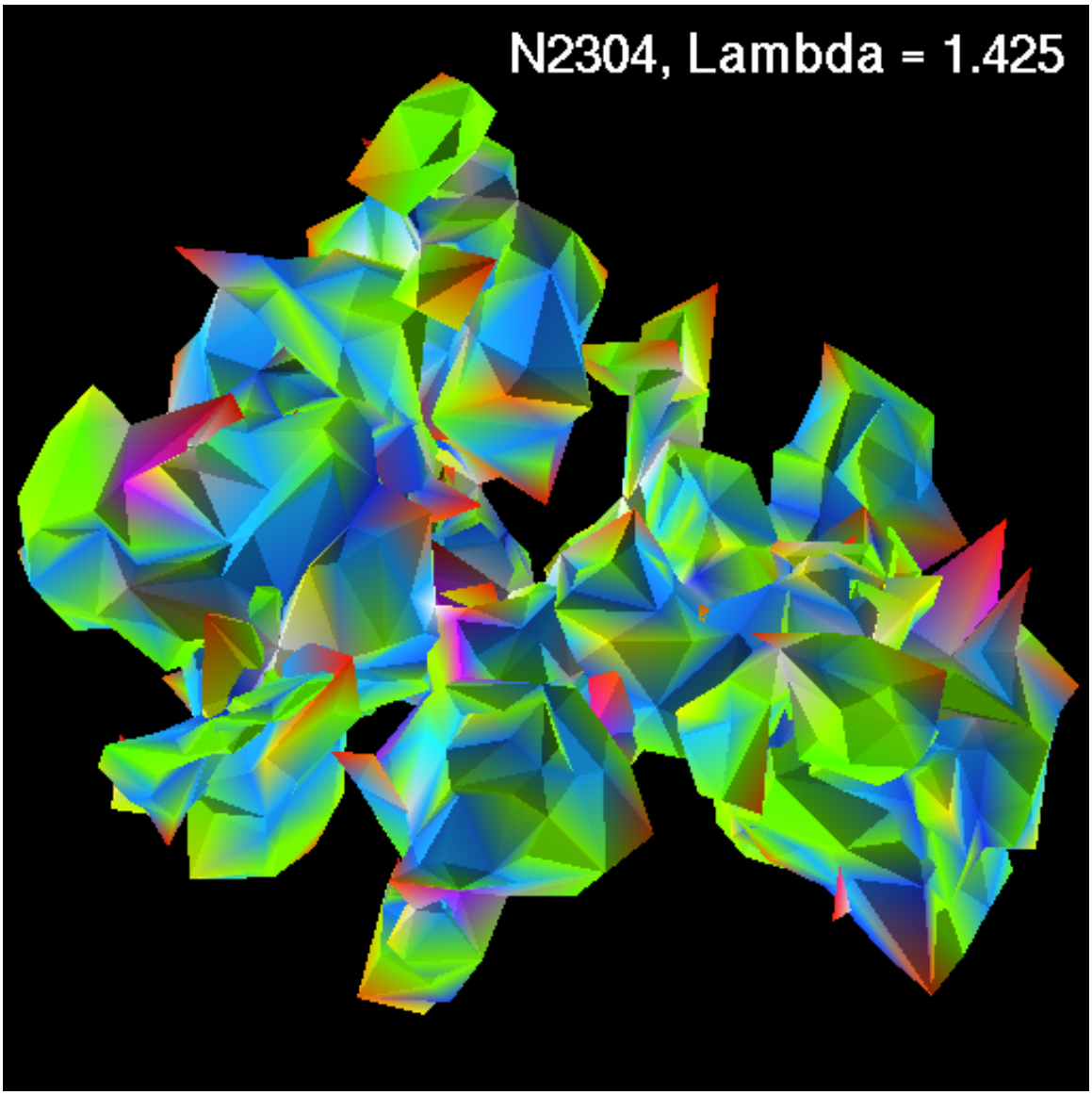

Figure 45: A configuration with 2304 nodes at $\lambda = 1.425$.

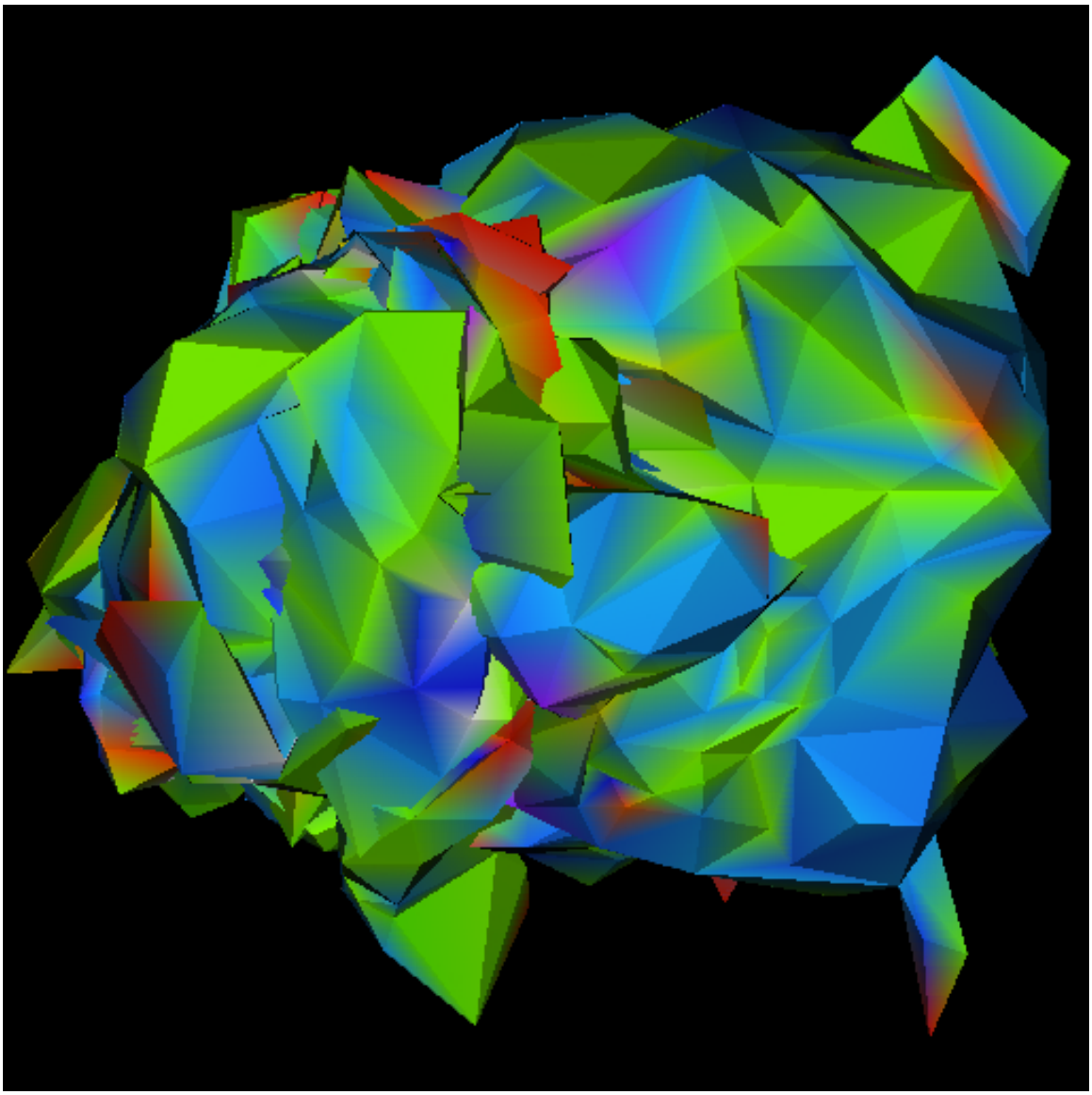

Figure 46: A configuration with 2304 nodes at $\lambda = 1.5$.

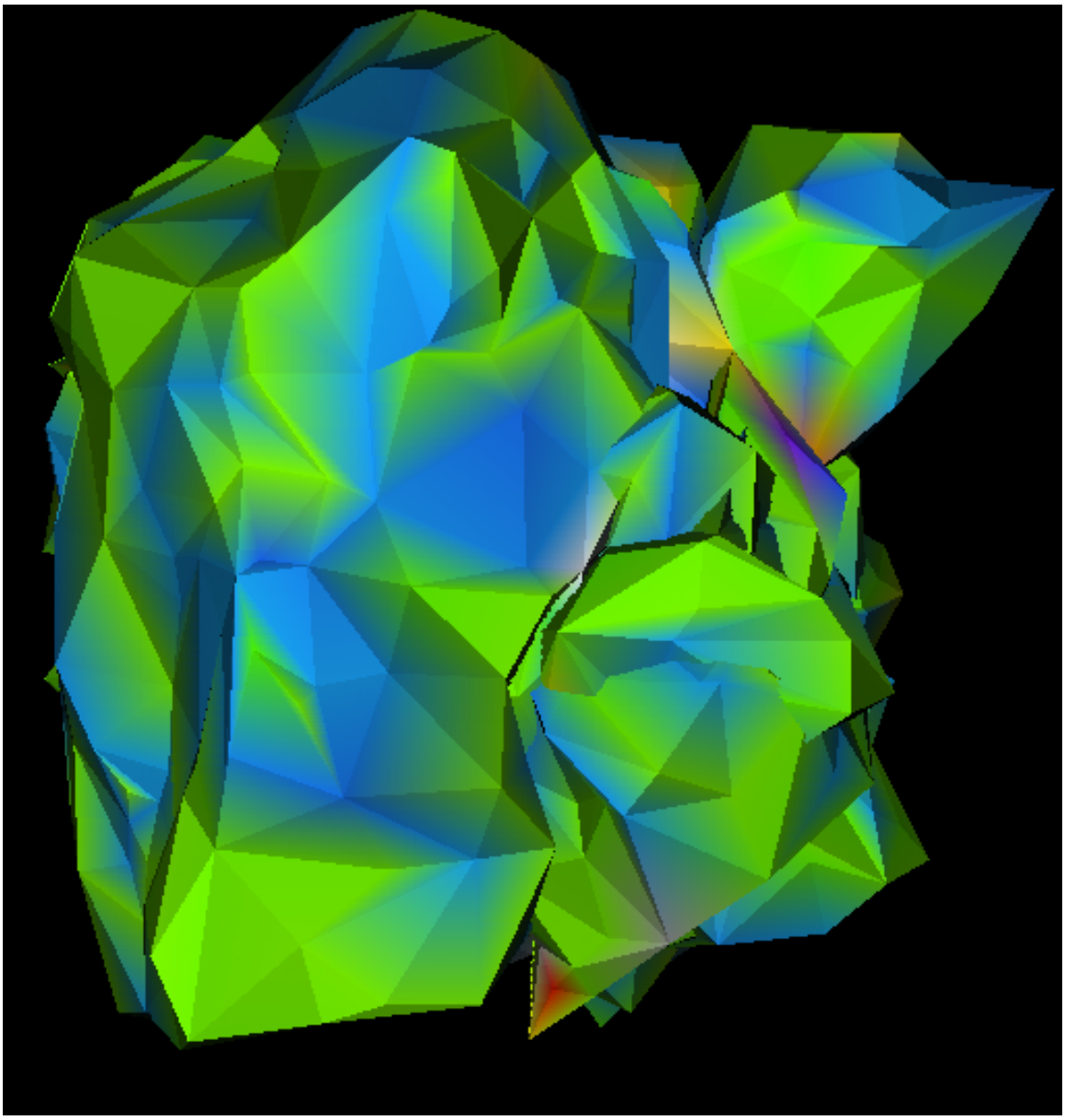

Figure 47: A configuration with 2304 nodes at $\lambda = 2.0$.

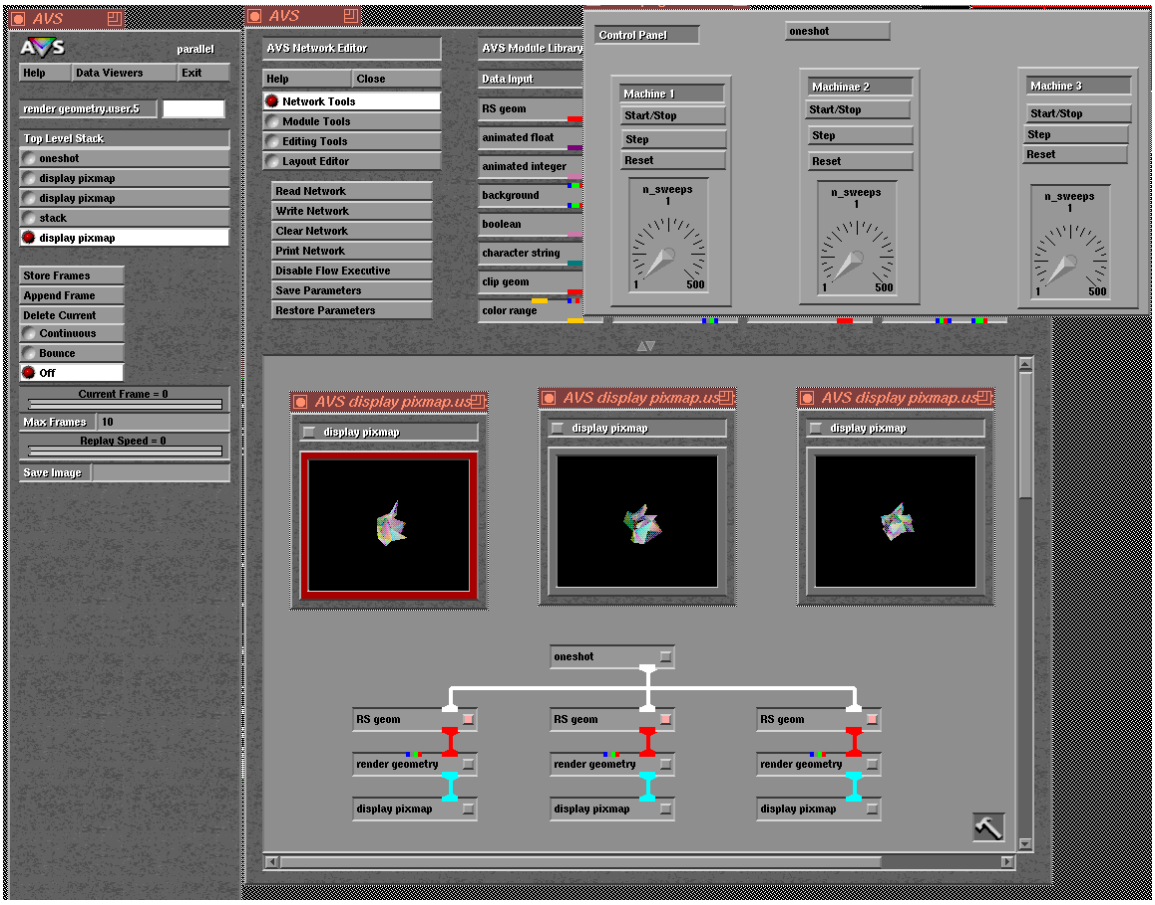

Figure 48: Using AVS to display parallel simulations over a distributed network

6 Conclusion

The main result described in this dissertation is the numerical study of the critical properties of non-self-avoiding fluid like random surfaces with extrinsic curvature by using dynamical triangulations. The topology of the surface is toroidal and the surface is embedded in 3-dimensional space. We have thus explored the phase diagram of fluid random surfaces with extrinsic curvature, but unfortunately we have been unable to determine if our model undergoes a phase (crumpling) transition at finite coupling. We have observed dramatic crossover behavior for particular observables in our Monte Carlo simulations, but on the other hand, the correlation times and certain finite-size effects do not behave as one would expect in the presence of a phase transition.

We found that the extrinsic-curvature specific heat peak ceases to grow on lattices with more than 576 nodes and that the location of the peak λ_c also stabilizes. The evidence for a true crumpling transition is still weak. If we assume it exists we can say that the finite-size scaling exponent $\frac{\alpha}{\nu d}$ is very close to zero or negative.

On the other hand our data does rule out the observed peak as being a finite-size artifact of the persistence length becoming comparable to the extent of the lattice.

The behavior of other lattice models also indicates that it is possible that we are observing the effects of finite-mass excitations on small lattices, rather than a phase transition. We hope that future work will clarify this murky state of affairs, to determine if there indeed exists a crumpling transition for fluid surfaces.

While studying random surfaces, we encountered many challenging computational issues. The use of high performance computers is central to our goal of understanding random surfaces. The irregular or fluctuating meshes makes the computational side very

rich. Careful construction of program for irregular data structure application can lead very big performance increase on some of RISC processors. It is always the essential step before implementation on parallel computer. On parallel computer, programming random surface simulations becomes much harder. We proposed some solutions to deal with domain decomposition for mapping the problem on different types of parallel computers. However, there are still many unresolved issues for having a practical parallel simulation program of dynamical triangulated random surfaces with extrinsic curvature. We also gained insights from 3-dimensional visualization of the random surfaces in which the structure and changes of the surfaces are very difficult to picture.

References

[1] A. Neveu and J. H. Schwarz, Nucl. Phys. **B31** (1971) 86; Phys. Rev. **D4** (1971) 1109.

[2] M. Green and D. Gross, *Workshop on Unified String Theories* (World Scientific, Singapore, 1986)

[3] Lars Brink and Marc Henneaux, Principles of String Theory, (Plenum Press 1988).

[4] A. M. Polyakov, Phys. Lett. **103B** (1981) 207.

[5] Michio Kaku Introduction to Superstrings (Springer-Verlag, New York 1988).

[6] M. B. Green, J. H. Schwarz and E. Witten, *Superstring Theory* Vols. 1 and 2, (Cambridge University Press, Cambridge, 1986)

[7] R. P. Feynman, Rev. Mod. Phys. **20** (1948) 267.

[8] J. Kogut, Rev. Mod. Phys. **51** (1979) 659

[9] Bernd Bruegman, Ph.D. Thesis (1993), Syracuse University.

[10] A. Ashtekar, *Lectures on non-perturbative canonical gravity* (World Scientific, Singapore, 1991)

[11] R. Lipowski, Nature, 349 (1991) 475.

[12] D. Nelson, in *Statistical Mechanics of Membranes and Surfaces*, Jerusalem Winter School, Vol. **5**, edited by D. Nelson, T. Piran and S. Weinberg (World Scientific, Singapore 1989) 167.

[13] V. G. Knizhnik, A. M. Polyakov and A. B. Zamolodchikov, Mod. Phys. Lett. **A3** (1988) 819.

[14] F. David, Mod. Phys. Lett. **A3** (1988) 1651; J. Distler and H. Kawai, Nucl. Phys. **B321** (1989) 509.

[15] E. Brézin and V.A. Kazakov, Phys. Lett. **236B** (1990) 144; M.R. Douglas and S.H. Shenker, Nucl. Phys. **B335** (1990) 635; D. J. Gross and A. A. Migdal, Phys. Rev. Lett. **64** (1990) 127. D. Gross and A. Migdal, Nucl. Phys. **B340** (1990) 333.

[16] V. A. Kazakov, Phys. Lett. **150B** (1985) 282; F. David, Nucl. Phys. **B257** (1985) 45; J. Ambjørn, B. Durhuus and J. Frölich, Nucl. Phys. **B257** (1985) 433.

[17] V. A. Kazakov and A. A. Migdal, Nucl. Phys. **B311** (1988/89) 171.

[18] A. M. Polyakov, Nucl. Phys. **B268** (1986) 406.

[19] L. Peliti and S. Leibler, Phys. Rev. Lett. **54** (1985) 1690.

[20] D. Forster, Phys. Lett. **114A** (1986) 115.

[21] H. Kleinert, Phys. Lett. **174B** (1986) 335.

[22] H. Kleinert, Phys. Lett. **114A** (1986) 263.

[23] A. Billoire and F. David, Phys. Lett. **168B** (1986) 87; Nucl. Phys. **B275** (1986) 617.

[24] J. Ambjørn, B. Durhuus and J. Fröhlich, Nucl. Phys. **B275** (1986) 161; J. Ambjørn, B. Durhuus, J. Fröhlich and P. Orland, Nucl. Phys. **B270** (1986) 457.

[25] D. Boulatov, V. Kazakov, I. Kostov and A.A. Migdal, Phys. Lett. **157B** (1985) 295.

[26] D. Boulatov, V. Kazakov, I. Kostov and A.A. Migdal, Phys. Lett. **174B** (1986) 87; Nucl. Phys. **B275** (1986) 641.

[27] P. B. Canham, J. Theor. Biol. **26** (1970) 61.

[28] F. David, Nucl. Phys. B (Proc. Suppl.) **17** (1990) 51

[29] W. Press, B. Flannery, S. Teukolsky and W. Vetterling, *Numerical recipes in C*, (Cambridge University Press, Cambridge 1988)

[30] N. Metropolis *et al.*, J. Chem. Phys. **21**, 1087 (1953).

[31] For a review of spin models and Monte Carlo algorithms, see Ed. K. Binder, *Monte Carlo Methods in Statistical Physics*, (Springer-Verlag, Berlin, 1986).

[32] P. Coddington, Lecture Notes (unpublished), (Sept. 1994)

[33] D. Kutasov and N. Seiberg, Nucl. Phys. **B358** (1991) 600.

[34] W. Helfrich, J. Phys. **46** (1985) 1263.

[35] Y. Choquet-Bruhat, C. DeWitt-Morette and M. Dillard-Bleick, *Analysis, Manifolds, and Physics* (Elsevier, Amsterdam 1987).

[36] M. P. Do Carmo, *Differential Geometry of Curves and Surfaces* (Prentice-Hall, Englewood, N.J., USA 1976).

[37] M.J. Bowick and E. Marinari, Gen. Rel. Grav. **24** (1992) 1209.

[38] S. Catterall, Phys. Lett. **220B** (1989) 207.

[39] C. Baillie, D. Johnston and R. Williams, Nucl. Phys. **B335** (1990) 469.

[40] C. Baillie, S. Catterall, D. Johnston and R. Williams, Nucl. Phys. **B348** (1991) 543.

[41] J. Ambjørn, J. Jurkiewicz, S. Varsted, A. Irbäck and B. Petersson, Phys. Lett. **275B** (1992) 295.

[42] J. Ambjørn, A. Irbäck, J. Jurkiewicz and B. Petersson, Nucl. Phys. **B393** (1993) 571.

[43] S. Catterall, J. Kogut and R. Renken, Nucl. Phys. **B** (Proc. Suppl.) **99A** (1991) 1.

[44] S. Catterall, Nucl. Phys. B (Proc. Suppl.) **B** (1991) 716.

[45] J. Polchinski and Z. Yang, Phys. Rev. **D46** 1992, 3667

[46] E. Braaten and C. Zachos, Phys. Rev. **D35** (1987) 1512.

[47] W. Helfrich, J. Naturforsch. **28C** (1973) 693.

[48] A. M. Polyakov, *Gauge Fields and Strings* (Harwood Academic Publishers, Chur, Switzerland 1987).

[49] T. Morris, Nucl. Phys. **B341** (1990) 443; J. Govaerts, Int. J. Mod. Phys. **A4** (1989) 173.

[50] J. Polchinski and A. Strominger, Phys. Rev. Lett. **67** (1991) 1681.

[51] J. Distler, Nucl. Phys. **B388** (1992) 648

[52] F. David, Europhys. Lett. **2** (1986) 577; F. David and E. Guitter, Europhys. Lett. **3** (1987) 1169; Nucl. Phys. **B295** (1988) 332; F. Alonso and D. Espriu, Nucl. Phys. **B283** (1987) 393.

[53] Z. Yang, Phys. Lett. **B279** (1992) 47.

[54] S. Catterall, D. Eisenstein, J. Kogut and R. Renken, Nucl. Phys. **B366** (1991) 647.

[55] D. Kroll and G. Gompper, Science **255** (1992) 968; Phys. Rev. **A46** (1992) 3118. G. Gompper and D. Kroll, Europhys. Lett. **19** (1992) 581;

[56] J. F. Wheater, Nucl. Phys. **B** (Proc. Suppl.) 34 (1994) 15.

[57] F. David, *Introduction to Statistical Mechanics of Random Surfaces and Membranes*, in *Two Dimensional Quantum Gravity and Random Surfaces* Jerusalem Winter School, Vol. **8** edited by D. Gross, T. Piran and S. Weinberg (World Scientific, Singapore 1992).

[58] F. David, in *Statistical Mechanics of Membranes and Surfaces*, Jerusalem Winter School, Vol. **5**, edited by D. Nelson, T. Piran and S. Weinberg (World Scientific, Singapore 1989) 158.

[59] M. Bowick, P. Coddington, L. Han, G. Harris and E. Marinari, Nucl. Phys. **B394** (1993) 791 and references therein.

[60] D. Gross and V. Periwal, Phys. Rev. Lett. **60** (1988) 2105.

[61] M. Agishtein and A. Migdal, Mod. Phys. Lett. **A7** (1992) 1039 M. Agishtein and A. Migdal, Nucl. Phys. **B385** (1992) 395 J. Ambjørn and J. Jurkiewicz, Phys. Lett. **B278** (1992)42.

[62] F. James, Comp. Phys. Comm. **60** (1990) 329.

[63] M. Falcioni, E. Marinari, M. L. Paciello, G. Parisi and B. Taglienti, Phys. Lett. **102B** (1981) 270; Nucl. Phys. **B190** (1981) 782; Phys. Lett. **108B** (1982) 331; E. Marinari, Nucl. Phys. **B235** (1984) 123.

[64] A. M. Ferrenberg and R. H. Swendsen, Phys. Rev. Lett. **61** (1988) 2635 and *Erratum*, ibid. **63** (1989) 1658.

[65] A. M. Ferrenberg and R. H. Swendsen, Phys. Rev. Lett. **63** (1988) 1196.

[66] N. A. Alves, B. A. Berg and S. Sanielevici, Nucl Phys. **B376** (1992) 218.

[67] R. H. Swendsen, Physica **A** (1993) 53.

[68] See, for example, J. Zinn-Justin, *Quantum Field Theory and Critical Phenomena* (Oxford University Press, New York, 1989).

[69] U. Wolff, Nucl. Phys. **B322** (1989) 759; Nucl. Phys. **B334** (1990) 581.

[70] J.M. Drouffe, G. Parisi and N. Sourlas, Nucl. Phys. **B161** (1980) 397; J. Ambjørn, B. Durhuus and T. Jonnson, Phys. Lett. **B244** (1990) 403.

[71] A.D. Sokal, in *Computer Simulation Studies in Condensed Matter Physics: Recent Developments*, eds. D.P. Landau *et al.* (Springer-Verlag, Berlin-Heidelberg, 1988); A.D. Sokal, in Proc. of the International Conference on Lattice Field Theory, Tallahassee, October 1990, Nucl. Phys. B (Proc. Suppl.) **20**, 55 (1991).

[72] H. Kawai and M. Ninomiya, Nucl. Phys. **B336** (1990) 115.

[73] A. Migdal, talk given at Syracuse University, April 1992.

[74] J. L. Colot, J. Phys. **A16** (1983) 4423.

[75] S. Shenker and J. Tobochnik, Phys. Rev. **B22** (1980) 4462.

[76] R. Brout, W. Deans and A. Silovy, Phys. Rev. **B27** (1983) 5813.

[77] R. Brout and W. Deans, Nucl. Phys. **B215** (1983) 407; J. Orloff and R. Brout, Nucl. Phys. **B270** (1986) 273.

[78] J. Apostolakis, C. Baillie and G. Fox, Phys. Rev. **D43** (1991) 2687.

[79] P. Hasenfratz and F. Niedermayer, Phys. Lett. **B245** (1989) 522.

[80] G. Martinelli, G. Parisi and R. Petronzio, Phys. Lett. **B100** (1981) 485.

[81] R. Brout and W. Deans, Nucl. Phys. **B215**[FS 7] (1983) 407.

[82] R. Brout, W. Deans and A. Silovy, Phys. Rev. **B27** (1983) 5813.

[83] J. Orloff and R. Brout, Nucl. Phys. **B270**[FS 16] (1986) 273.

[84] W.A. Bardeen, B.W. Lee and R.E. Shrock, Phys. Rev. **D14** (1976) 985.

[85] J.A. Lipa and T. C. P. Chui, Phys. Rev. Lett. **51** (1983) 2291.

[86] See N. Goldenfeld, *Lectures on Phase Transitions and the Renormalization Group,* FIP Vol. 85 (Addison-Wesley, Reading Massachusetts, 1992).

[87] V. A. Kazakov, Phys. Lett. **A119** (1986) 140.

[88] D. V. Boulatov and V.A. Kazakov, Phys. Lett. **B186** (1987) 379.

[89] Y. Kantor and D. Nelson, Phys. Rev. Lett. **58** (1987) 2774; Phys. Rev. **A36** (1987) 4020.

[90] J. Ambjørn, B. Durhuus and T. Jonnson, Nucl. Phys. **B316** (1989) 526.

[91] R. Harnish and J. Wheater, Nucl. Phys. **B350** (1991) 861.

[92] J. F. Wheater and P. W. Stephenson, Phys. Lett. **B302** (1993) 447.

[93] K. Anagnostopoulos, M. Bowick, P. Coddington, M. Falcioni, L. Han, G. Harris and E. Marinari, Phys. Lett. **B317** (1993) 102.

[94] P. Butera, M. Comi and G. Marchesini, Nucl. Phys. **B300** (1989) 1.

[95] M. Agishtein, R. Benav, A. Migdal and S. Solomon, Mod. Phys. Lett. **A6** (1991) 1115.

[96] H. S. Stone, *High-Performance Computer Architecture* (Addison-Wesley Publishing 1993)

[97] W. Stallings, Proceedings of The IEEE, **Vol 76**, No.1, (1988) 38

[98] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, D. W. Walker, *Solving Problems on Concurrent Processors*, Vol. 1, (Prentice-Hall, 1988)

[99] C.F. Baillie, D.A. Johnston and R.D. Williams, Comput. Phys. Commun. **58**, 105 (1990).

[100] R.D. Williams, Proc. of the 3rd Hypercube Conference, Pasadena, 1988, ed. G. C. Fox, (ACM Press, New York, 1988).

[101] R. Das, D.J. Marvriplis, J. Saltz, S. Gupta and R. Ponnusamy, *The Design and Implementation of a Parallel Unstructured Euler Sover Using Software Primitives*,ICASE Report No. 92-12

[102] Z. Bozkus, A. Choudhary, G. Fox, T. Haupt, S. Ranka, M. Wu, *Compiling Fortran 90D/HPF for Distributed Memory MIMD Computers* Syracuse University Internal Report, March 8, 1993, SCCS-444.

[103] N. Chrisochoides and E. Houstis and J. Rice, J. of Parallel and Distributed Computing, **21** (1994) 71

[104] Gordon Erlebacher and Peter R. Eiseman, AIAA Journal, Vol. **25**, Num. **10** (1987) 1356

[105] H. Gould and J. Tobochnik, *An Introduction to Computer Simulation Methods, Vol. 2*, (Addison-Wesley, Reading, Mass., 1988)

[106] G. Huang and W. Ongsakul, *An Efficient Task Allocation Algorithm and Its Use to Parallelize Irregular Gauss-Seidel Type Algorithms*, (8th International Parallel Processing Symposium, Apr. 1994, Cancun, Mexico)