Lattice Geometrodynamics

A Dissertation Presented

by

Mark Corrado Galassi

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Physics

State University of New York

at Stony Brook

December 1992

State University of New York at Stony Brook
The Graduate School

Mark Corrado Galassi

We, the dissertation committee for the above candidate for the Doctor of Philosophy degree, hereby recommend acceptance of the dissertation.

This dissertation is accepted by the Graduate School.

Graduate School

Abstract of the Dissertation

Lattice Geometrodynamics

by

Mark Corrado Galassi

Doctor of Philosophy

in

Physics

State University of New York

at Stony Brook

1992

I present the results of my work aimed at making Regge Calculus a practical tool for numerical relativity. The principal results are a formalism for solving the Regge equations in the general case, a working software package *r3+1* which implements these methods, and a collection of numerical results produced by this software.

The formalism for solving the Regge equations consists of a set of algorithms for constructing and analyzing arbitrary simplicial lattices, calculating deficit angles, and (for the first time) calculating the Jacobian of the Regge equations. I also present a definition of lapse and shift in Regge Calculus.

At this time the software package consists of approximately ten thousand lines of C++ code which offer a user interface, lattice analysis routines, and routines to calculate and solve the Regge equations with arbitrary choice lapse and shift. The Regge equations are solved using Newton's method and conjugate gradient methods.

The numerical runs presented here (1) confirm the validity of the methods used; (2) examine the structure of diffeomorphism freedom in Regge Calculus, giving the first verification that one can choose lapse and shift freely; (3) examine the Bianchi identities and quantify the extent to which time evolution in Regge Calculus preserves constraints, finding that the constraints are proportional to the *third* power of the lattice spacing.

Table of Contents

List of Figures . . . . **vi**

1. Motivation and Plan of the Work **1**

1.1 General Relativistic phenomenology 1

1.2 The need for numerical relativity 2

1.3 Diffulties of numerical relativity 3

1.4 Approaches to numerical relativity 4

1.5 Summary of my approach and results 5

2. Introduction to Regge Calculus **6**

2.1 Simplicial decompositions 6

2.2 The geometry of a simplicial lattice 8

2.3 The Regge action 10

2.4 The (remarkably simple) Regge equations 11

2.5 There is much more to say 12

2.6 Past applications of Regge Calculus 12

3. Simplicial Lattices **14**

3.1 Mock coordinates 15

3.2 3D lattices: T^3 15

3.3 3D lattices: S^3 16

3.4 4D lattices: straightforward 17

3.5 4D lattices: the teepee lattice 18

4. Formulating the Regge Equations: Scanning the Lattice **21**

4.1 Software history and overview 21

4.2 Hashing 22

4.3 Storage and retrieval of *edge* information 24

4.4 Storage and retrieval of *triangle* information 25

4.5 The triangle entourages 25

5. Formulating the Regge Equations: Calculations **26**

5.1 General relativity without coordinates 26

5.2 Minkowski signature 28

6. Solving the Regge Equations **30**
6.1 What to specify and what to solve 30
6.2 Numerical methods 30
6.3 The Jacobian for the Regge equations 31
6.4 Implementation of the Jacobian calculation 32

7. Bianchi Identities, Constraints, Lapse and Shift **34**
7.1 The skeletal Bianchi identities 34
Included paper: Lapse and Shift in Regge Calculus 36

Final Reflections **56**

Bibliography **57**

List of Figures

Chapter 2

2.1 Simplices in low dimensions 6
2.2 Triangulations of the plane 7
2.3 Triangulations of S^2 7
2.4 Triangulatino of S^3 8
2.5 Deficit angles on a 2D surface 9
2.6 Hinges in 2, 3 and 4 dimensions 9
2.7 Angle $\theta_{t,i}$ in triangle t 11

Chapter 3

3.1 A sequence of thin spacetime sandwiches 14
3.2 Representations of T^2 and T^3 15
3.3 A cube broken down into 6 tetrahedra 16
3.4 Triangulated prisms in 3 and 4 dimensions 17
3.5 Template of spacelike edges emanating form a point 19
3.6 The teepee lattice in 1+1 dimensions 19
3.7 The teepee lattice in 2+1 dimensions 20

Chapter 4

4.1 A chained hash table with linked list collision resolution 23

Chapter 5

5.1 A triangle and a tetrahedron carefully marked 26

Chapter 7

7.1 Lapse and shift focusing on the interesting region of spacetime 35
Several figures included in the Lapse and Shift paper 36–55

Reflections

Destiny of the Regge Calculus practitioner 56

Acknowledgements

My friends Martin and Warner have given me the advice and support so sorely needed by an apprentice researcher; but most of all they have cared about me in these years and frequently stuck their necks out for me beyond what professors are supposed to do for their students. I would have left graduate school and physics if I had not met them and had them as my advisors.

Together with other students of my generation in the Institute for Theoretical Physics at Stony Brook, I owe a lot to Peter van Nieuwenhuizen who taught advanced Quantum Field Theory in our first two years here. In his lectures, and in the famous Friday afternoon seminars, Peter taught us a lot of material; but even more important is how he taught us to question and verify what we are told, no matter how obvious it may seem. I feel that this has shaped my generation of ITP students and the ideal we have of scientific honesty, and I take this ideal with me as I leave Stony Brook.

The scientists in the ITP have provided a wonderful environment in a university frequently hostile to students. In particular I have dropped in to discuss problems of physics and life with Jack Smith, Warren Siegel, Fred Goldhaber and Robert Shrock, and always found them willing to spend time with me until the problem was solved. Max Dresden got me started in General Relativity when I first arrived in Stony Brook, and gave me very good advice in my first two years there.

I have also received wonderful hospitality from the Theoretical Astrophysics group (T-6) in Los Alamos. I have spent a lot of time in Los Alamos in the last year and a half, and have benefited greatly from the atmosphere at the lab.

There are times in which I have an almost pathological need for “a little help from my friends”, and in the last five years, when Rosalia was frequently out of town, I was frequently rescued from loneliness by my friends Alison, Eric the housemate, Michelle, Mike (of Stony Brook), Mike (of Los Alamos) and Susan. Looking forward to seeing them would keep me going when my burdens seemed heavy and my loneliness without cure. I do not think they can understand how important they are to me.

My family, and my parents in particular, have always encouraged me to take the academic path in life, and they are very proud and supportive of my decision to be a scientist.

As I write this page (August 15th 1992) my wife Rosalia and I are celebrating 12 years since we first met on a hill in Tuscany. In this time our lives have become so closely merged that my intimacy with her gives meaning to everything I do, and I dedicate my life and work to her.

On the technical side I owe thanks to many people who have helped me with calculations and software.

Warner Miller first proposed that I should “put Regge Calculus on the map” as a method for numerical relativity, and he has worked closely with me on this project for more than a year. All the results I present in this dissertation are truly “joint” results. I also relied heavily on many of the formulæ Warner developed for the Null Strut Calculus. Warner also insisted that we study a topic exhaustively and give it a complete stress test before passing to other topics. This is hard when one wants to push forward and tackle new problems, but I see the wisdom of it now. My paper *Lapse and Shift in Regge Calculus* (included verbatim in this dissertation) sets the scene for future numerical work in Regge Calculus just because I carried out a complete analysis of diffeomorphism freedom and Bianchi identities.

Martin Roček has suggested several interesting problems in Regge Calculus in the past years (concerning characteristic classes and random surfaces). While I was working on this project he stepped through the details of my approach. In particular he worked through the code which calculates the Jacobian matrix for the Regge equations, and spotted the single bug that was causing my calculation to fail.

Leo Brewin visited Warner and me in Los Alamos in the summer of 1992 and showed us his approach to solving the Regge equations in the general case. We sat down and compared numerical methods in great detail, and I was pleasantly surprised to see that Leo had reached almost exactly the same conclusions I had about how the Regge lattice needs to be represented so that the Regge equations can be solved in general circumstances.

My fellow graduate students Fabian Eßler and Eric Laenen frequently listened to me discuss my numerical approach, and spent time at the blackboard with me wading through the very complex derivatives of deficit angles, necessary for calculating the aforementioned Jacobian matrix.

Pablo Laguna, who has worked in finite differencing for a long time, has helped me understand how my results relate to the standard results of numerical relativity.

Michael Fischer (of Stony Brook) and Michael Warren (of Los Alamos) are both top–notch programmers and it has always been a pleasure to consult with them on specific software development issues. In the design stages of my *r3+1* software I also consulted with gurus Michael Galassi and Alan Batie (both of Portland), as I always do when I am starting a major software project. Their big picture of the computer world helped me make some correct decisions from the start.

The tools I used for software development are almost entirely provided by Richard Stallman and the Free Software Foundation. Development was done with GNU emacs, the GNU C++ compiler, the GNU debugger and the GNU Bourne Again Shell (bash). I thank Richard and the FSF for what they have done: their dedication to free software has made many software development projects possible. I am happy and proud to have distributed software of my own under the FSF's General Public License.

I have used other well–known free software in my work: the X window system, the mongo and gnuplot plotting packages, and several other utilities and filters. This dissertation is typeset using Donald Knuth's TEX, and the entire physics community owes him thanks for developing this system.

I have also used some proprietary software. The versions of UNIX running on Sun and Silicon Graphics equipment are proprietary, and I used Mathematica to calculate messy determinants. These are useful products, but the support I got in using them was noticeably inferior to that from the free software community.

I drew most of the diagrams in this dissertation myself, and my lack of skill in drawing is manifest; but I hope these pictures will still help illustrate some of my points. Gennie Fox in Los Alamos helped me scan some of the figures, and gave me advice on improving some of the diagrams.

Chapter 1: Motivation and Plan of the Work

Regge Calculus is thirty one years old: the paper in which Regge invented the method was published in 1961.

Since then there have been several interesting developments in which Regge Calculus has been used to approximate known classical solutions to Einstein's equations, or to calculate path integrals in quantum gravity. Recently there has also been some interest in using Regge Calculus to formulate the Polyakov path integral for the bosonic string. The foundations of Regge Calculus have been explored and are well understood (Wheeler: 1963 and 1974; Miller: 1986a and 1986c). In many of the early papers people would make comments like, "Regge Calculus should prove to be a flexible tool for putting general relativity on a computer." Meanwhile numerical relativists were developing software to approximate general solutions to Einstein's equations via finite differencing, but no parallel development was occuring in Regge Calculus.

Warner Miller (1986b, 1986c, 1988) first attempted solving the Regge equations without making assumptions of symmetry: he developed the Null Strut Calculus, a modification of Regge Calculus which offers certain computational advantages. Still there are no codes today with the same scope as those used in finite differencing. Regge Calculus has *not* proven to be a flexible tool for solving Einstein's equations on a computer.

In this introductory "motivational" chapter I will ramble about the need for numerical relativity and my own motivation for "putting Regge Calculus on the map" as a method for numerical relativity. I will also give an overview of my methods and results.

Section 1.1 General Relativistic phenomenology

The need to go beyond Newtonian gravity was motivated by observations: the orbit of Mercury could not be completely explained by classical methods. General relativity gives the correct answer for the orbit of Mercury, and has given correct answers in all cases in which corrections to Newtonian gravity have been needed. But when do we need these corrections? And can other, easier methods be used? What is the phenomenology of general relativity?

Early general relativistic phenomenology comprised the *three classic tests of general relativity* of which the most important are the deflection of light by the sun and the precession of the perihelia of planetary orbits[†]. These effects have been verified, and interest is shifting away from the solar system to the study of astrophysical phenomena.

Recent observations of energy loss in millisecond pulsars have matched predictions of gravitational wave emissions: these are the first (indirect) results of gravitational wave astrophysics. Gravitational wave astronomy promises to become the link between general relativity and observational astronomy, since we expect that they should also be emitted in great quantities by other astrophysical phenomena, most notably stellar collisions.

[†] The other test proposed by Einstein was the gravitational red shift of spectral lines coming from massive bodies, which tests the principle of equivalence and not the details of general relativity

Gravitational wave astronomy will get a boost in the next decade from the development of the *Laser Interferometer Gravity–wave Observatory* (LIGO). Gravitational radiation from colliding black holes or neutron stars should be observable by LIGO, providing experimental verification of the existence of black holes. Kip Thorne (1990) has set a challenge for theorists to predict the detailed form of the signals that will be observed by LIGO for various possible sources.

Section 1.2 **The need for numerical relativity**

There are few exact solutions of Einstein's equations

$$G_{\mu\nu} = \kappa T_{\mu\nu}$$

(where κ depends on the units and metric signature; in this section $\kappa = 8\pi G$) and they are remarkably specific. The only solutions of phenomenological interest are those of Kerr–Newman and Friedman; most others have little application to real problems in astrophysics and cosmology.

The way in which we construct solutions to Einstein's equations usually involves making a simplifying *ansatz*. That is, we assume that the solution will have a certain form specified by certain parameters, and we obtain differential equations for those parameters and solve them. The classic example is the Schwarzschild metric, where we assume that the metric is static and isotropic. By skillfully manipulating coordinates we can put the metric in the form

$$ds^2 = -B(r)dt^2 + A(r)dr^2 + r^2\big(d\theta^2 + \sin^2\theta d\phi^2\big)$$

where $g_{rr} = A(r)$, $g_{tt} = B(r)$, $g_{\theta\theta} = r^2$ and $g_{\phi\phi} = r^2\sin^2\theta$. It is clear that only two components of the metric are unknown: the assumptions of time independence and isotropy have eliminated the other unknowns.

By plugging this *ansatz* into the Einstein equations we get the solutions $A(r)B(r) =$ constant and $rB(r) = r +$ constant. Enforcing the Newtonian limit for the g_{tt} field:

$$g_{tt} = B \to 1 + 2\phi$$

(where $\phi = -MG/r$) we get

$$B(r) = 1 - \frac{2MG}{r}$$

and

$$A(r) = \frac{1}{B} = \frac{1}{1 - (2MG/r)}$$

yielding

$$ds^2 = -\Big(1 - \frac{2MG}{r}\Big)dt^2 + \frac{dr^2}{1-(2MG/r)} + r^2\big(d\theta^2 + \sin^2\theta d\phi^2\big).$$

We have solved differential equations for $A(r)$ and $B(r)$ and obtained the very particular Schwarzschild metric. The Robertson–Walker metric is obtained by a similar *ansatz*:

the cosmological principle, which states that the universe is homogeneous and isotropic, leads to the imposition of symmetry on each spacelike hypersurface, and restricts the metric to having the form:

$$ds^2 = -dt^2 + R^2(t)\Big\{\frac{dr^2}{1-kr^2} + r^2 d\theta^2 + r^2 sin^2\theta d\phi^2\Big\}\ .$$

This metric can be plugged into the Einstein equations with an energy momentum tensor $T_{\mu\nu}$ made of "dust":

$$T_{\mu\nu} = \rho u_\mu u_\nu,$$

yielding a differential equation for the cosmic scale factor $R(t)$:

$$\dot{R}^2 + k = \frac{8\pi G}{3}\rho R^2.$$

An overview of existing solutions, in particular the two presented above, and of the problems that need to be solved in relativistic phenomenology tells us that:

- there are few analytic solutions, and almost no useful solutions;
- all analytic solutions have a heavy dose of symmetry imposed through a simplifying *ansatz*, although some non–symmetric solutions have been approximated using perturbation theory;
- nobody ever attempts to set up general matter fields and then find general solutions for *all* components of the metric $g_{\mu\nu}$;
- some key problems are not symmetrical: in particular stellar collisions and other strong sources of gravitational waves.

The solution to these problems lies in attacking Einstein's equations numerically and without imposing symmetry. The ideal tool would be a black box which takes input describing the distribution of matter and the initial geometry, and ouputs all components of the metric as the geometry evolves in time. This is the goal of *numerical relativity.*

Section 1.3 **Difficulties of numerical relativity**

Numerical relativity is an important undertaking, but it is also extremely difficult. Some of the problems facing the practitioner are:

- the physical fields *are* the geometry; we cannot make a grid and throw fields at it, as we usually do when solving partial differential equations numerically in other fields of physics;
- setting up coordinate systems to tackle realistic problems is a daunting task;
- computations involving 3+1 dimensions and real physical scales are extremely large, and are only becoming possible with current computer technology.

This last "complaint" is actually common to several fields of numerical physics: certain phase transitions, bound states for quarks and many other solutions to specialized areas of physics are still out of reach because of the sheer size of the problems. But the first two problems are especially felt in numerical relativity, and account for much of the effort put into this field.

Section 1.4 **Approaches to numerical relativity**

Currently the only successful method in numerical relativity is *finite differencing*. It is based on discrete differencing of differential equations in which one would map, for example:

$$\frac{\partial\phi(x,t)}{\partial x} \rightarrow \frac{\phi_{i+1,t} - \phi_{i-1,t}}{2a}$$

and so forth. The list of successful projects carried out with finite differencing is impressive. To name just a few: Smarr *et al.* have simulated collisions of black holes in 1+1 and 2+1 dimensions (Smarr *et al*: 1976; Smarr: 1977; Smarr: 1979); Shapiro and Teukolsky (1991) have investigated naked singularities numerically; Goldwirth and Piran (1992 and references therein), and Laguna *et al.* (1991), have modeled inhomogeneous cosmologies; Cook *et al.* (see Cook 1990) have produced initial data for black hole collisions in 3+1 dimensions; while Choptuik has made significant progress in adaptive grid algorithms. Most of these results are presented the recent proceedings *Frontiers in Numerical Relativity*, edited by Evans *et al.* (1989).

Many people work on finite difference methods, and several groups are preparing to tackle the problem of black hole collisions in this way. Finite difference methods command respect because several problems have already been solved. There are some difficulties with this approach, among them the fact that the Bianchi identities are broken when the Einstein equations are written in finite difference form. This means that if we evolve a surface in time, future surfaces will not satisfy all ten Einstein equations. Another problem lies in the choice of topologies: it is easy to program R^3 and T^3 topologies, but S^3 or more unusual topologies are much more difficult.

Another approach to discretizing relativity is *Regge Calculus* which is the focus of this dissertation. I will introduce it in detail in the next chapter, but I mention some aspects of it here to juxtapose it with finite differencing. Instead of discretizing the Einstein equations directly we discretize the geometry of spacetime by triangulating it with four dimensional simplices. Regge found that metric and curvature information live naturally in this simplicial lattice: the metric is encoded in the lattice edge lengths and the curvature is proportional to the deficit angles. This makes it straightforward to define a simplicial analogue of the Einstein–Hilbert action, and then to obtain a set of algebraic equations to determine the edge lengths. This is a more intrinsically geometric approach, and it has many pleasant formal characteristics: unusual topologies are easily represented, the Bianchi identities have a geometric interpretation (although there are some unresolved issues pertaining to the contracted Bianchi identities; these are discussed in Chapter 7). But, by the same token, it requires a high level of abstraction because simplicial lattices are not as straightforward as cubic grids, and the information does not consist of tensors defined at spacetime points, so a change of perspective is required.

Although very pleasing from a formal point of view, Regge Calculus has not been explored numerically as much as finite differencing, and most of its original applications have been in rather esoteric areas of quantum and classical gravity[‡].

[‡] I list some past applications of Regge Calculus in the next chapter.

Section 1.5 **Summary of my approach and results**

In the spring of 1991, following a suggestion by Warner Miller, I took on the task of making Regge Calculus that "flexible tool" for numerical relativity. I have developed formalisms and software to solve the Regge equations in the general case, and I have applied this software to some cosmological models.

This dissertation contains a complete description of the methods I developed, and a presentation of the results I have obtained with my software. Chapter 2 gives an introduction to Regge Calculus. Chapter 3 introduces the various types of simplicial lattices I use.

Chapters 4, 5 and 6 describe how my software formulates and solves the Regge equations: the way in which my software scans and understands the lattice, determines what should be calculated and what should be specified, calculates deficit angles and Regge equations, and then solves the Regge equations for both initial value data and time evolution. These chapters are very important because the entire strength of my approach lies in the software's ability to formulate and solve the Regge equations on its own. This allows the scientist to experiment with several models with almost no modification of the code.

Chapter 7 discusses formal and practical issues that must be addressed before one can make useful software runs: the Bianchi identities and the nature of constraints, of diffeomorphism freedom and of lapse and shift in Regge Calculus. Chapter 7 introduces the paper *Lapse and Shift in Regge Calculus* by myself in which I analyze all these issues exhaustively. This paper, which is included verbatim, presents the most important results on the viability of Regge Calculus in numerical relativity:

- four of the Regge equations associated with any vertex can be treated as constraints;
- four conditions per vertex can be specified freely, corresponding to a free choice of lapse and shift;
- once lapse and shift values are specified there is no more freedom, and the gauge is completely fixed;
- once the dynamical equations are solved, the constraint equations are nearly satisfied, and they are proportional to the *third* power of the lattice spacing.

Chapter 2: Introduction to Regge Calculus

In 1961 Regge developed a method for discretizing Einstein's theory of general relativity (Regge: 1961). Instead of starting with the field equations, he discretized the spacetime geometry, he then wrote the Einstein–Hilbert action for this skeleton spacetime, and derived the skeleton equations from this action.

There are many steps involved in formulating Regge Calculus, and in reaching the main result (the Regge equations). I will now carry out these steps. Other introductions to Regge Calculus can be found in Regge (1961), Wheeler (1963, 1974) and Williams (1986).

Section 2.1 **Simplicial decomposition**

Any "nice" manifold $\mathcal{M}$ can be decomposed into a collection of *simplices* glued together. This *simplicial complex* will approximate the geometry of $\mathcal{M}$ up to a certain point. I will now define simplices and show some examples of decomposition of everyday manifolds.

A zero dimensional simplex is simply a point P. In one dimension it s a line segment (P_0, P_1). In two dimensions a simplex is a triple of points which can be interpreted as the vertices of a triangle: (P_0, P_1, P_2). In three dimensions it is a 4–tuple of points (P_0, P_1, P_2, P_3) which defines a tetrahedron. A 4–simplex is a 5–tuple of points $(P_0, P_1, P_2, P_3, P_4)$ which can be interpreted as the vertices of a 4–dimensional polyhedron. See Fig. 2.1 for pictures of these low–dimensional simplices.

Simplices have certain properties which prompt their definition and their use in Regge Calculus:

1. A simplex is a *rigid* figure. Simplices are always completely determined by the lengths of the edges, whereas more elaborate figures (like a square in two dimensions) are still floppy when the edges have been specified.
2. An n–simplex is the polyhedron in n dimensions with the least number of edges and vertices, and can be used to build any other.
3. The definition of the boundary of a simplex is quite straightforward. In two dimensions: $\partial(P_0, P_1, P_2) = (P_0, P_1) + (P_1, P_2) - (P_0, P_2)$, and the generalization to higher dimensions is direct. This makes them useful in defining integration of differential forms on manifolds, and in formulating Stokes' theorem (Flanders: 1963).

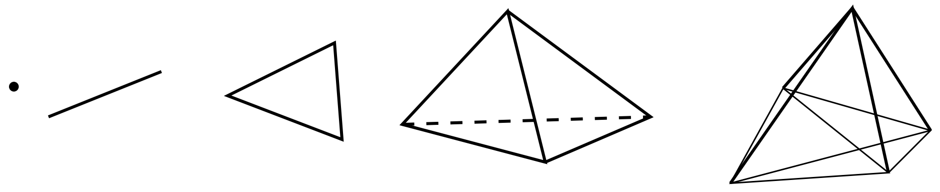

Fig. 2.1: A 0–simplex (point), 1–simplex (segment), 2–simplex (triangle), 3–simplex (tetrahedron) and 4–simplex, respectively.

Now for some examples, in two dimensions first. The plane can be tiled in several ways, two of which are quite straightforward. One very regular tiling is composed of identical equilateral triangles; another consists of tiling the plane with squares and drawing diagonals through the squares. These are the same triangulation, really, but with different initial assignments of the edge lengths.

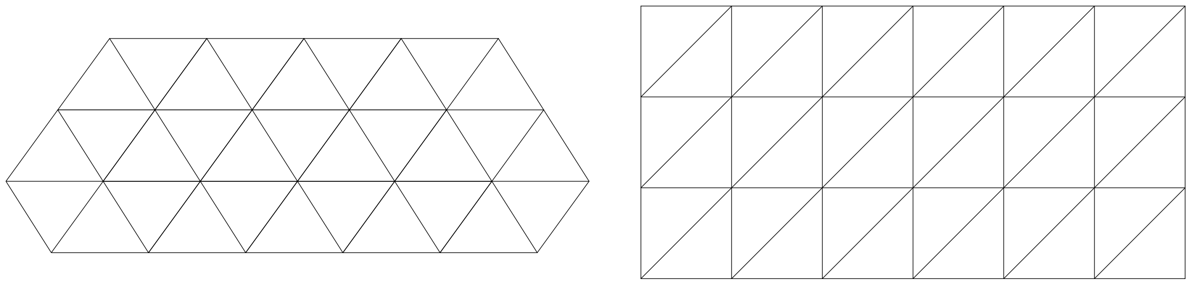

Fig. 2.2: Two decompositions of the plane, into equilateral and rectangular triangles respectively.

The simplest approximation to a sphere S^2 is the surface of a tetrahedron: it breaks the sphere into four triangles. This triangultion of S^2 is interesting because it also applies to higher dimensional spheres: they can all be thought of as the boundary of the single simplex in one higher dimension, since the simplex represents a ball of its dimension: $S^n = \partial B^{n+1}$. S^2 can also be modeled by an octagon, and in fact by the surfaces of several famous solids.

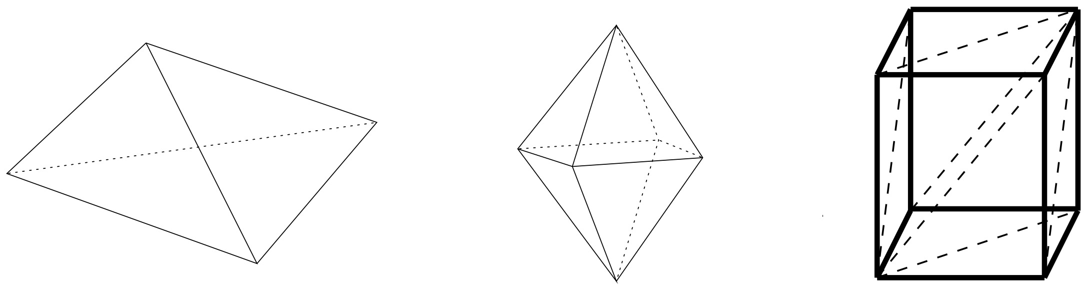

Fig. 2.3: Decomposition of S^2 into the surface of a tetrahedron, an octahedron and a cube respectively.

A two dimensional torus T^2 can be modeled by triangulating a rectangle and identifying opposite sides, so the triangulations shown in Fig. 2.2 for the plane can both be clipped to model T^2.

Three-dimensional flat space R^3 can be tiled by decomposing it into a cubic lattice and then triangulating the cubes properly (this is done in Chapter 3). The same applies to T^3: take a parallelepipedal chunk of R^3 and identify opposite faces to get the torus topology.

We can also decompose S^3 in several ways. The simplest is the 5–tetrahedron model, where the 5 tetrahedra form the boundary of the 4–simplex $(P_0, P_1, P_2, P_3, P_4)$. There are two ways of visualizing it: we can place the points $P_0, \ldots, P_4$ at the origin and unit vectors of R^4, and see the boundary of the 4–simplex as embedded in R^4. Alternatively we can draw a tetrahedron in R^3, then glue four other tetrahedra to each face of the first one, and identify the four points that stick out of this contsruction. The identification imposes S^3 topology on this complex (see Fig 2.4). There are two other decompositions of S^3 with identical regular tetrahedra: they have 16 and 600 tetrahedra (Wheeler: 1963, Collins and Williams: 1973). Other triangulations will have non–regular tetrahedral blocks; see for example our algorithm for triangulating S^3 as the boundary of the 4–cube in section 3.2.

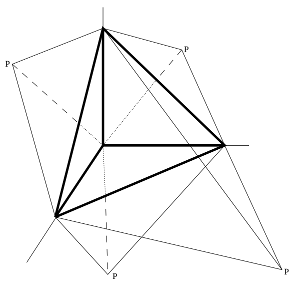

Fig. 2.4: Triangulation of S^3 taking a tetrahedron, gluing four other tetrahedra to its faces and identifying their loose vertices P.

The examples shown above should help us understand how certain "frequently occurring" four dimensional topologies can be triangulated. As it turns out, the four dimensional spacetimes used in numerical relativity will always single out a time direction in such a way that the the topology will be $\mathcal{M}^{(3)} \times R$. In Chapter 3 I will present two ways in which I triangulate $\mathcal{M}^{(3)} \times R$ given a triangulation of $\mathcal{M}^{(3)}$.

Section 2.2 **The geometry of a simplicial lattice**

Having discussed the topological decomposition of a lattice into a simplicial complex, we are now ready to examine how the geometry of spacetime can be represented on this simplicial lattice. The first thing to notice is that in Regge's construction, each simplex[†] is taken to have a flat interior. The inside of each simplex is a section of Minkowski space, and all distances within the simplex can be calculated using flat–space methods. Since the simplices are rigid, the geometry of the entire complex will be encoded in the edge lengths of the simplices.

To formulate general relativity we must also identify the curvature in our simplicial complexes. Regge did this by introducing the notion of a *deficit angle*. Let us first look at this in two dimensions. Consider a collection of triangles hinging on a vertex P. The sum of all internal angles $\sum \theta_i$ at P should be a full angle 2π. Buf if this point were on a curved surface (see Fig. 2.5), the angles would not add up to quite 2π. The difference

[†] From now on, if I don't specify the dimension of a simplex explicitly, I will mean a 4–simplex.

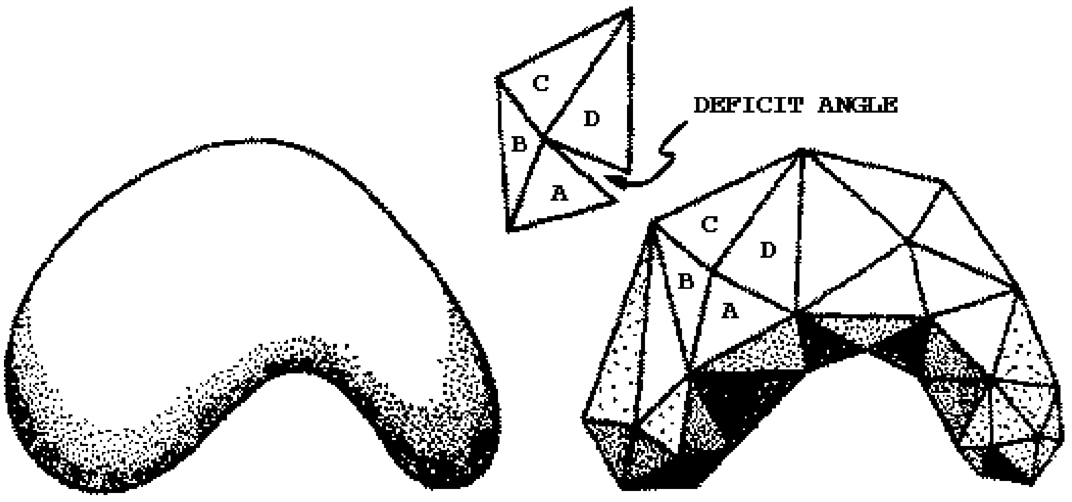

Fig. 2.5: Wheeler's (1963) visualization of a deficit angle in two dimensions.

$\epsilon_P = 2\pi - \sum \theta_i$ is called the deficit angle at point P. The scalar curvature curvature at P is then given by ϵ_P divided by the area dual to P.

In three dimensions we can have several tetrahedra hinging on a common edge e with length l_e. Each tetrahedron will have a dihedral angle on e, and the sum of these dihedral angles would be 2π in flat space. The deficit angle is $\epsilon_e = 2\pi - \sum_i \Delta_{i,e}$, where $\Delta_{i,e}$ is the dihedral angle of tetrahedron i on its edge e. The scalar curvature will then be concentrated on edge e, and will be $l_e \epsilon_e$ divided by the volume dual to e.

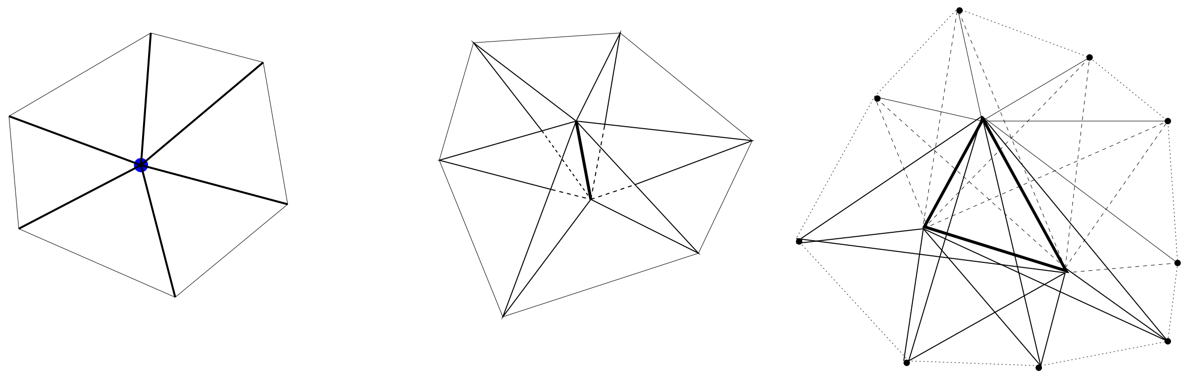

Fig. 2.6: Simplices in two, three and four dimensions hinging on points, edges and triangles respectively.

As one might expect by analogy with the lower dimensions, in four dimensions we contemplate a triangle t with area A_t upon which some simplices σ_i hinge. Each simplex will have a dihedral angle $\Delta_{i,t}$ on t, and the sum of these dihedral angles would be 2π in flat space. The deficit angle associated with triangle t is then

$$\epsilon_t = 2\pi - \sum_i \Delta_{i,t}. \tag{2.1}$$

The scalar curvature is concentrated on the triangles, and is proportional to $A_t \epsilon_t$ divided by the 4–volume dual to t.

Section 2.3 **The Regge action**

We have seen that the scalar curvature is always concentrated on simplices of the dimension of the manifold minus 2 (simplices of *codimension* 2) which we call hinges or bones. The actual expression for the curvature is equal to the hinge volume times the deficit angle divided by the volume dual to the hinge. We now want to write the Einstein–Hilbert action in n dimensions:

$$I_{EH} = \int R\sqrt{|g|}d^n x \tag{2.2}$$

for our Regge skeleton.

Since the volume element $\sqrt{|g|}$ multiplies R in equation (2.2), we do not need to divide by it in Regge's expression for the scalar curvature R, and the action in n dimensions is simply

$$I_R = \sum_h V_h \epsilon_h, \tag{2.3}$$

where h is the hinge of dimension $n-2$, and V_h is the $n-2$–dimensional volume of h. Here is the expression for equation (2.3) in 2, 3 and 4 dimensions:

$$I_R^{(2)} = \sum_P \epsilon_P \qquad (n=2) \tag{2.4a}$$

$$I_R^{(3)} = \sum_e l_e \epsilon_e \qquad (n=3) \tag{2.4b}$$

$$I_R^{(4)} = \sum_t A_t \epsilon_t \qquad (n=4). \tag{2.4c}$$

I will now pass to the derivation of the Regge equations from the Regge action, but there are some comments that can be made on the curvature of simplicial complexes or *piecewise linear spaces*, so I will insert them here.

For two dimensional manifolds $\mathcal{M}$ without boundary the Einstein–Hilbert integrand is equal to the Euler class, hence the action is a topological quantity: the Euler number for that manifold $\chi_{\mathcal{M}}$. One consequence of this is that the action is a constant for a given topology, so its variation δI will always be zero and there will be no field equations in two dimensions.

This translates to Regge Calculus: the sum of deficit angles in two dimensions is always equal to $\chi_{\mathcal{M}}$, and there are no Regge equations in two dimensions.

In higher dimensions the Euler class is not proportional to the scalar curvature, so the action is not a topological invariant. In $n=4$ the Euler characteristic is given by

$$\chi \equiv \frac{1}{128\pi^2}\int R_{\alpha\beta\gamma\delta}R_{\mu\nu\rho\sigma}\epsilon^{\alpha\beta\mu\nu}\epsilon^{\gamma\delta\rho\sigma}\sqrt{|g|}d^4x.$$

Although Regge (1961) has written an expression for the full Riemann tensor $R^{\mu}{}_{\nu\rho\sigma}$, it is not clear how to take powers of it. Cheeger, Müller and Schrader (1984) have formulated the Euler class for general piecewise linear spaces, and given much insight into the nature of curvature for simplicial manifolds. Hamber and Williams (1984, 1986) have explored Regge Calculus models of gravity with higher powers of R in the action.

A deeper understanding of the curvature of piecewise linear spaces is very important in simplicial quantum gravity with higher order actions. It is not yet clear how useful the results of Cheeger *et al.* will be in numerical Regge Calculus.

Section 2.4 **The (remarkably simple) Regge equations**

We now want to obtain the skeleton analogue of Einstein's equations starting from the Regge action. In the variational formulation of general relativity we vary the action I_{EH} with respect to the metric. In Regge Calculus the metric is encoded in the edge lenths, so we want to take the variation of I_R with respect to the edge lenths l_i.

Let us carry out the calculation. Let index i represent the edges, l_i their lengths, and $t_{|i}$ the triangles that hinge on i, then

$$\delta I_R = \delta\Big(\sum_t A_t \epsilon_t\Big) = \sum_t \sum_i \Big(\frac{\partial A_t}{\partial l_i}\delta l_i \epsilon_t + A_t \frac{\partial \epsilon_t}{\partial l_i}\delta l_i\Big).$$

Regrouping the terms in the sums, we can invert the order of summation:

$$\cdots = \sum_t \Big(\sum_i \Big(\frac{\partial A_t}{\partial l_i}\epsilon_t + A_t\frac{\partial \epsilon_t}{\partial l_i}\Big)\delta l_i\Big).$$

The term $\partial A_t/\partial l_i$ is equal to $\cot\theta_{t,i}$ (or 0 if edge i is not part of triangle t), where $\theta_{t,i}$ is the angle opposite edge i in triangle t as shown in Fig. 2.7.

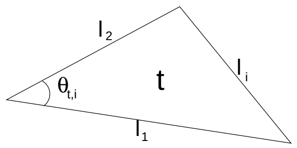

Fig. 2.7: The angle $\theta_{t,i}$ in triangle t.

The other term $\partial\epsilon_t/\partial l_i$ is much more messy, but Regge, in his original paper (1961), proved that in taking the variation of I_R we can treat ϵ_t as constant. This streamlines the expression for δI_R:

$$\delta I_R = \sum_i \Big(\sum_{t_{|i}} \cot\theta_{t,i}\epsilon_t\Big)\delta l_i.$$

Setting this equal to zero we get an equation for each edge i in the lattice:

$$\sum_{t_{|i}} \cot\theta_{t,i}\epsilon_t = 0. \qquad (2.5)$$

These are the Regge equations. They are remarkably simple, thanks to the fact that we can treat ϵ_t as constant when varying the action‡. There is one equation for each edge: an algebraic equation which couples several edges from the triangles $t_{|i}$ surrounding i and the simplices in the *entourage* of $t_{|i}$.

Section 2.5 **There is much more to say . . .**

Prima facie one might think that one can solve the equations for all the edges and thus completely determine the edge lengths. As it turns out there is more to it than that. We must understand if the equations are all independent, if any of the unknowns can be specified, what to do on the boundaries (for example initial and final surfaces) when an edge might not have a complete entourage with which to calculate the Regge equations.

So the derivation of the Regge equations is not the last step in formulating Regge Calculus. We must address several other issues before we are ready to use Regge Calculus as the "flexible tool" everyone used to believe it would become. That has been a major part of my work in the past year and a half: I set out to make Regge Calculus into a viable tool for numerical relativity, and have had to answer the questions mentioned above and several others as well. My results will be presented in the remaining chapters.

Section 2.6 **Past applications of Regge Calculus**

Notwithstanding the absence, until now, of general 3+1 dimensional solutions, Regge Calculus has had several interesting applications in both classical and quantum gravity, together with some very powerful mathematical results. I will mention a few here, and a more comprehensive review together with a bibliography can be found in Williams and Tuckey (1992).

- *The Schwarzschild metric.* Wong (1971) and later Collins and Williams (1972) worked out approximations to the Schwarzschild metric.Williams and Ellis investigated the nature of geodesics in simplicial spacetimes (1981, 1984), and Brewin (1988, 1989, 1992) completed this work obtaining the correct value for the precession of the perihelion of Mercury.
- *Time evolution of model universes.* Collins and Williams (1973, 1974) first solved the Regge equations for the expanding universe using models of S^3 with 5, 16 and 600 tetrahedron models. They made the same *ansatz* that is made in obtaining the Friedman solution (isotropy and homogeneity), and came up with behaviour for the scale factor which (in the 600 cell model) approaches that of the analytic solution. Lewis (1982) worked out a solution for the flat Friedman universe and the Kasner universe using cubic blocks, and showed that the solutions approach the analytic solution as the lattice is refined. This type of simulation is one of the goals of numerical relativity, but these particular models do not hold any promise for going beyond the known analytic solutions.

‡ As it turns out, in Chapter 6 I will return and actually take the derivative of the deficit angle because it is used to calculate the Jacobian.

- Quantum gravity in three and four dimensions. Ponzano and Regge (1968) showed that there is a relationship between the $6j$ symbols used in quantum mechanical angular momentum theory and the path integral for 2+1 dimensional Regge Calculus (see also Hasslacher and Perry: 1981). This has not yet been used to calculate amplitudes in 2+1 dimensional quantum gravity, but it is a surprising and elegant result.
  Roček and Williams (1982, 1984, 1985) studied the propagator and the nature of conformal transformations in four dimensional Euclidean quantum Regge Calculus.
  More recently Hartle (1985, 1986) has investigated simplicial minisuperspaces in an attempt to carry out the path integral for the wave function of the universe.
- Two dimensional quantum gravity and *random surfaces* (Boulatov *et al.*: 1986). A couple of years ago there was a burst of activity in the field of random surfaces: a method of discretizing the Polyakov path integral for the bosonic string. Together with the introduction of scalar fields on the two dimensional curved manifold, this method allowed the numerical computation of path integrals for Polyakov strings. It was also found that path integrals for random surfaces are related to the perturbation expansion in matrix models, but it was only possible to carry out the calculations for strings living in less than one dimension.
- The *Null Strut Calculus*. Developed by Miller (1986c, 1988), this was the first attempt to solve the Regge equations without imposing symmetry: a full 3+1 scheme. Null Strut Calculus is a modification of Regge Calculus in which three dimensional blocks on successive spacelike surfaces are connected by null edges. This drastically simplifies the formulæ for calculating volumes and deficit angles.

Chapter 3: Simplicial Lattices

Tullio Regge (1961) chose simplicial blocks to decompose spacetime manifolds, rather than cubic blocks. One advantage of a simplicial block is that it is "rigid": this means that the block is completely determined by the lengths of the edges.

Regge Calculus has also been formulated for complexes containing some non–simplicial blocks. Collins and Williams (1974) and Lewis (1982) use trapezoidal faces in their cosmological models, re–deriving the Regge equations from the action to account for the different types of hinges. Miller (1986c, 1988) developed the *Null–Strut Calculus*, a version of Regge Calculus in which spacelike hypersurfaces are connected by null (zero–length) edges; this greatly simplifies the equations. Null–Strut Calculus also uses non–simplicial blocks. In all of these schemes some extra conditions have to be imposed to "rigidify" the spacetime blocks.

In my formulation I have chosen to use only simplicial blocks. This has made it possible to write software which figures out the Regge equations and their derivatives on its own, with no hard–coded functions and very few assumptions (see Chapters 4, 5 and 6). This has allowed me to radically change my choice of lattice, of constraints, and of lapse and shift and *never* have to change the mechanisms by which the equations are calculated and solved.

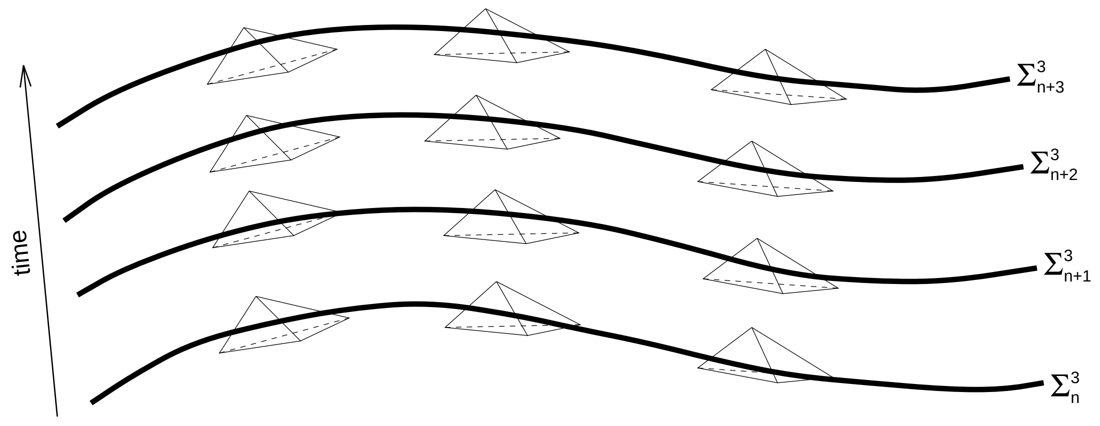

Fig. 3.1: A sequence of thin spacetime sandwiches.

I build the four dimensional lattice from a sequence of "thin" spacetime *sandwiches*, each sharing a slice of bread with the previous and next layers. This is done so I can formulate an *initial value* problem on the first sandwich, and then evolve it in time in small increments. The boundaries of each sandwich are three dimensional hypersurfaces, approximated by tetrahedral complexes. The middle region of each sandwich is a layer of four dimensional spacetime, approximated by a complex of 4–simplices.

In this chapter I give my prescription for triangulating the three dimensional surfaces and then the four dimensional sandwiches.

Section 3.1 Mock coordinates

In formalism presented here I do not introduce coordinate systems inside the simplices, but I do use "mock" coordinates to identify points in the lattice as I draw up the lists of tetrahedra and simplices. This is because I typically generate lattices starting with cubic lattices and triangulating the cubes. Mock coordinates are simply integer grid coordinates (i, j, k, t), where $0 <= i < n_x$, $0 <= j < n_y$, $0 <= k < n_z$ and $t = 0, 1 \text{or} 2$.

Mock coordinates should not be taken too seriously: they are merely an algorithmic aid in setting up the initial spacelike surfaces, and the (i, j, k) coordinates will rarely give a good indication of the space coordinates of a point in the lattice. For example, when *r3+1* builds the five tetrahedron model of S^3, it chooses $n_x = 5$ and $n_y = n_z = 0$, which means that the five coordinates points in S^3 would all be on the x axis, which does not make much sense. That is why I call them *mock coordinates*.

Section 3.2 3D lattices: T^3

The simplest three dimensional manifold (without boundary) to triangulate is the three–torus T^3. To get a better grasp on this it is easier to examine the situation for T^2: we take a triangulation of a rectangle and identify all points and edges that are on opposite sides of the rectangle.

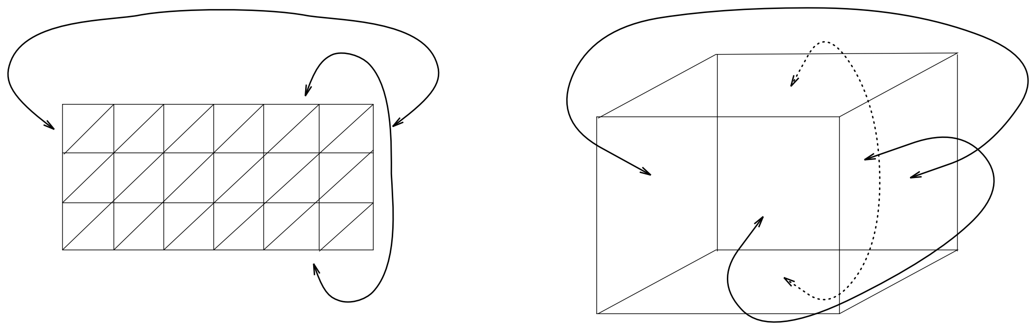

Fig. 3.2: Representation of T^2 and T^3: triangulate a rectangle (or cube) and then identify the opposite sides (or faces).

Triangulating T^3 is completely analogous: take a three dimensional cubic grid with dimensions n_x, n_y, n_z and identify the points (i, j, k) with $i = 0$ and $i = n_x$, $j = 0$ and $j = n_y$, $k = 0$ and $k = n_z$. This wrapping function gives us the T^3 topology.

We now have a collection of $N = n_x \cdot n_y \cdot n_z$ cubes, and we decompose each individual cube into six tetrahedra by adding diagonal braces as in Fig. 3.3. If we call the vertices of the cube A, B, C, D, E, F, G, H, then the six tetrahedra will be $\{A, C, G, H\}$, $\{A, C, D, H\}$, $\{A, B, D, H\}$, $\{A, B, F, H\}$, $\{A, E, F, H\}$ and $\{A, E, G, H\}$ etc...

This scheme guarantees that when two cubes are connected on a face, the triangles that form that face on each cube will match. This is called the Quantity Production Lattice (QPL) (Miller: inchworm, dissertation).

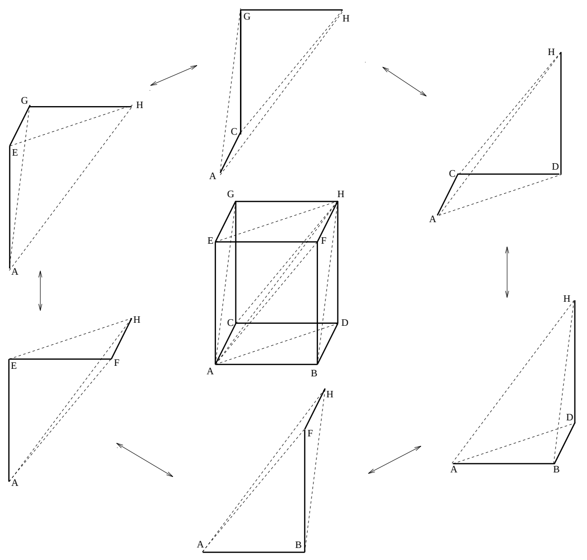

Fig. 3.3: A single cube broken down into six tetrahedra in a manner consistent with its neighbours.

Section 3.3 **3D lattices: S^3**

Making an arbitrarily fine simplicial lattice with T^3 topology is quite straightforward. It is more difficult to do the same with S^3 topology. There are only three regular complexes that model S^3: the 5–cell, 16–cell and 600–cell models (Wheeler: 1963, Collins and Williams: 1974), so using triangulations of S^3 with regular tetrahedra is not possible.

One solution is to notice that the boundary of a 4–cube ∂C^4 has the topology of a 3–sphere. If C^4 is triangulated into 4–simplices, then its boundary will be automatically triangulated into tetrahedra. I use this method for generating S^3 surfaces of arbitrary resolution. The drawback is that, if the edge lengths are inherited from the 4–cube, the resulting surface will be flat in most regions and will have all its curvature concentrated at the corner impurities. This can be solved by assigning edge lengths throughout the sphere in such a way as to smooth the curvature out.

One straightforward algorithm for doing that consists of formulating an average scalar curvature $\overline{R}$ for each hinge (in three dimensions the edges are the hinges on which curvature is concentrated, in two dimensions each point is a hinge). The we shorten all edges connected to the hinge h, except the hinge itself. This has the effect of reducing the deficit angle ϵ_h. We can then compare the value of ϵ_h/V_h (where V_h is the volume dual to hinge h) to the average curvature $\overline{R}$, and continue this process until the deficit angle is close to the required curvature.

Let us look at this in detail in two dimensions, where the S^2 is modeled as the boundary of a cube. Assume that each face of the cube was divided into N squares, hence $2N$ triangles. Then we have $8N$ vertices in the triangulation, of which 8 are saddled with a deficit of $4\pi/8 = \pi/2$ at the beginning. The average curvature we would like to have for each angle is $4\pi/(8N)$.

So we run through each vertex P, calculate the deficit angle ϵ_P, divide it by the area element A_p and compare it to the desired average scalar curvature, and keep increasing or decreasing the edges attached to h until the we get a smooth average scalar curvature.

When I describe my algorithms for solving the Regge equations I will come back to the impurities in the S^3 lattice to discuss the fact that there are fewer Regge equations in the region containing the impurity. This is the other difficulty that has to be addressed before large scale S^3 models become viable.

Section 3.4 **4D lattices: straightforward**

Given a triangulation of a three dimensional surface Σ^3, we now want to break $\Sigma^3 \times R$ into 4–simplices. Since we are working in a thin sandwich formulation, it is sufficient to triangulate $\Sigma^3 \times I$, where I is the interval on the real line, and piece several of these sandwiches together. The most straightforward algorithm for doing so is:

(1) assume we have a triangulation of the 3–surface Σ^3 made up of tetrahedra T^i, whose vertices are labeled $[T^i_0, T^i_1, T^i_2, T^i_3]$; similarly, Σ^3_{N+1} will be triangulated in the same manner, with vertices labeled $T^{i'}$; it is crucial that the points in each tetrahedron be ordered: $T^i_0 < T^i_1 < T^i_2 < T^i_3$: this will be used in the proof of consistency given below;
(2) for each tetrahedron T^i form the four dimensional prism $\overline{T^i T^{i'}}$;
(3) break the prism down into four 4–simplices with vertices:

$$[T^i_0,\ T^i_1,\ T^i_2,\ T^i_3,\ T^{i'}_3] \tag{3.1a}$$

$$[T^i_0,\ T^i_1,\ T^i_2,\ T^{i'}_2,\ T^{i'}_3] \tag{3.1b}$$

$$[T^i_0,\ T^i_1,\ T^{i'}_1,\ T^{i'}_2,\ T^{i'}_3] \tag{3.1c}$$

$$[T^i_0,\ T^{i'}_0,\ T^{i'}_1,\ T^{i'}_2,\ T^{i'}_3] \tag{3.1d}$$

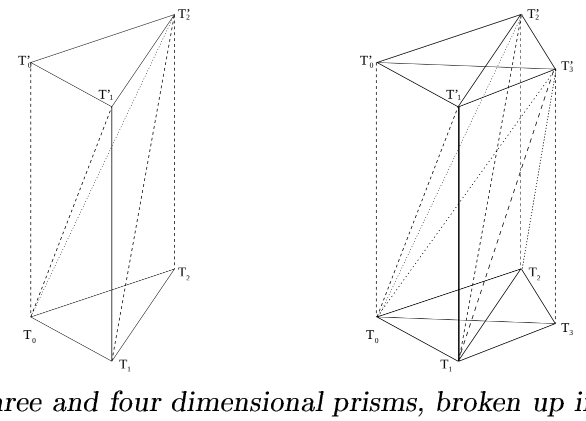

Fig. 3.4: Three and four dimensional prisms, broken up into simplices.

We now have a triangulation of each prism, but in putting together the simplices in two adjacent prisms we must be sure that there are no inconsistencies. Two prisms share a parallelepiped, and in the process of triangulating the prisms, this boundary parallelepiped will be triangulated into tetrahedra. Consistency here means that the two different triangulations of the boundary parallelepiped must match. In other words, the diagonal braces inserted to triangulate the prisms must be the same on the boundaries.

I will prove that the algorithm above is consistent in 2+1 dimensions, but the proof does not make any reference to two dimensions, so it is correct in general. Consider two prisms $\overline{TT'}$ and $\overline{UU'}$, where T, U, T' and U' are triangles on the present and future surfaces respectively. Also assume that the bases T and U share an edge $T_0 = U_0$, $T_1 = U_1$, and the same for T' and U'. This means that the prisms themselves will share a timelike quadrilateral face. When we carry out the triangulation in the algorithm above we will introduce diagonal braces $\overline{T_0T_1'}$ and $\overline{U_0U_1'}$, but never $\overline{U_1U_0'}$, because of the ordering $U_0 < U_1$. These braces will be the same for both prisms. Because of this ordering of points in T and U, we will never have the situation where $T_0 = U_1 and\ T_1 = U_0$, which is the only way in which inconsistent braces could be introduced when triangulating adjacent simplices.

Section 3.5 **4D lattices: the teepee lattice**

I use the 4D lattice presented in the previous section to generate initial data on the first sandwich. This usually gives rise to large systems of coupled equations: for a spacelike lattice with $3 \times 3 \times 3$ vertices, we end up with a system of 216 algebraic equations with 216 unknowns. When this system is solved with Newton's method, each iteration involves solving a system of 216×216 linear equations. In general, with $N = n_x \times n_y \times n_z$ points on a surface, the initial value problem will involve $8N \times 8N$ systems.

If we were to solve the time evolution problem using this kind of lattice, each time step would involve an $11N \times 11N$ system. Each iteration in Newton's method would be extremely slow in converging. To get around this problem, Barret *et al.* (1992) have developed a four dimensional lattice which effectively decouples the Regge equations used for time evolution into an eight–stage evolution procedure in which at each stage we only solve 15 equations for 15 unknowns. In this section I describe the lattice construction which allows breaking evolution down into stages.

Assume that we have three dimensional surfaces Σ^3_N and Σ^3_{N+1} already triangulated with the QPL. The vertices in Σ^3_N are uniform: each vertex has the same number of spacelike edges emanating from it, and these edge lengths are periodic. One can form a template describing all the spacelike edges emanating from a point (Fig. 3.5) and translate that template from point to point to generate all the spacelike edges in the lattice.

In generating what I call the *decoupling* or *teepee* lattice I break the uniformity between points. Referring to Fig. 3.3 we will partition vertices on Σ^3 into 8 types: A, B, C, D, E, F, G, H according to their "mock' coordinates in the cubic lattice of origin. To make this procedure clear, I will outline it in 1+1, 2+1 and 3+1 dimensions.

- In **1+1 dimensions** the spacelike hypersurface is one dimensional. This line is broken up into segments (one dimensional simplices) which will have mock coordinates 0, 1, 2, 3, ... Call the even points 0, 2, 4, $\ldots A$–type vertices, and the odd points 1, 3, 5, $\ldots B$–type vertices. Focusing on an A–type vertex, call the adjacent vertices B_+

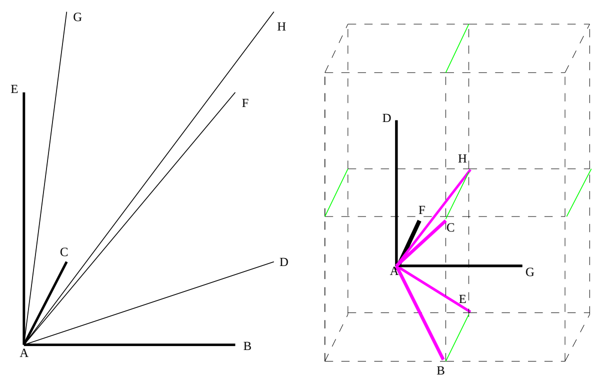

Fig. 3.5: A template of spacelike edges emanating from a point in Σ^3 in both the cubic lattice and the QPL.

and B_-, and call their images on the future surface A', B'_+ and B'_- (see Fig. 3.6). Now we lift the *teepees*: extend edges from A' to A, B_+ and B_-. This is the A–teepee. Then connect B'_+ to B_+ and to A', and B'_- to B_- and A'. This generates B–type teepees which lean on their A–type neighbours.

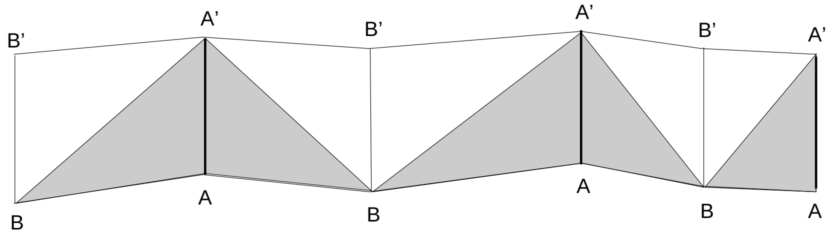

Fig. 3.6: The teepee lattice in 1+1 dimensions (with A–type teepees shaded).

If we now examine what edges are needed to calculate deficit angles at an A–type of vertex, we see that we only need information from the past sandwich and the edges in the A–teepee. This means that we can solve for that portion of lattice before we examine the B–teepees. The process of time evolution would now be split in two stages.

- In **2+1 dimensions** I introduce four types of teepees. The steps are:
  1. Label the edges of a two dimensional QPL: A (mock coordinates $(2i, 2j)$), $B(2i+1, 2j)$, $C(2i, 2j+1)$ and $D(2i+1, 2j+1)$.
  2. Erect vertex A into the future point A'.
  3. Connect A' to the neighbouring B, C, D–type vertices. This forms the A–teepee.
  4. Erect vertices of type B into the future B'.
  5. Connect B' to its respective A', C and D vertices. Notice that brace $B'A'$ is an edge in the $(N+1)^{th}$ spacelike hypersurface.
  6. Erect C to C' and connect C' to its respective A', B' and D vertices.
  7. Erect D to D' and connect D' to its respective A', B' and C' vertices. Notice how D' only has one connection to the N^{th} surface.

The edges forming the A–teepee (not including the surface Σ^3_N, which has already been solved) are AA', B_+A', B_-A', C_+A', C_-A', D_+A', D_-A': 7 edges which must be

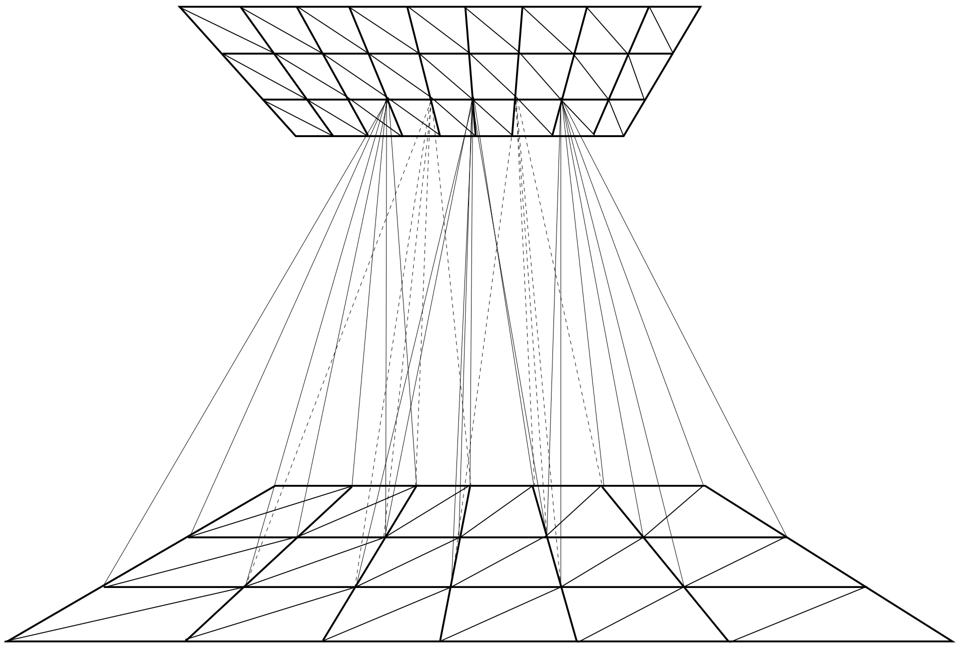

Fig. 3.7: The teepee lattice in 2+1 dimensions; the solid lines are A–type teepees, the dashed lines are B–type.

determined to solve the A–teepee. We then need 7 equations to solve for these, without relying on any edge except the past sandwich and the A–teepee. If we examine the edges AB_+, AB_-, AC_+, AC_-, AD_+, AD_-, AA', we see that calculation of the deficit angles on these edges does not take us outside of the A–teepee, thus we have 7 Regge equations with which to solve for the 7 unknowns in teepee A.

We can make a similar analysis of the B–teepee, and we find that using only edges in the past sandwich and in the A–teepee (which has already been solved) we can determine the remaining unknowns. The same goes for C and D–teepees.

- In **3+1 dimensions** (see Fig. 3.3 and Fig. 2 in the paper included in Chapter 7) I identify eight types of edges (A, B, C, D, E, F, G, H) and carry out the same algorithm as in 2+1 dimensions, just with more vertex types.

Chapter 4: Formulating the Regge Equations: Scanning the Lattice

In this and the next two chapters I will describe the collection of methods I use to formulate and solve the Regge equations in the general case. This is a very difficult task, and it requires sophisticated data structures and traversal algorithms, and the development of new formalisms at several stages. In this chapter I describe how my software intelligently scans the lattice, so that it can apply the equations presented in the next two chapters chapter to do all the calculations needed for the Regge equations and the Jacobian matrix.

To organize this chapter I will present the methods in the order in which the software operates. My program *r3+1* first prepares the three dimensional triangulation of Σ^3, then the four dimensional triangulation of $\Sigma^3 \times I$; these triangulations are described in previous chapters. Then *r3+1* scans the list of simplices and extracts from it lists of edges and triangles in the lattice, and useful cross-referencing data. It re–scans the lattice again for a list of what simplices hinge around each triangle. Then it assigns initial lengths to these edges according to a metric which the user can program in. The program can now understand the lattice so well that it can begin calculating dihedral angles, deficit angles, Regge equations and their derivatives.

I will now describe this analysis of the lattice in depth. A warning: this chapter describes in some detail the algorithms and software I have developed, and makes several references to actual C and C++ code. You can skip this chapter and take all this lattice analysis as a black box. Chapters 5 and 6 will still describe the workings of the program, but will focus more on the calculation of volumes, angles, Regge equations, Jacobians and so on: it is back in the realm of physics rather than computer science.

Section 4.1 Software history and overview

The *r3+1* program is completely original. I started writing it on paper in the summer of 1991, based on an experimental 2+1 dimensional version I had developed in May of 1991. The first successful calculation of deficit angles and Regge equations was in October 1991. The first correct calculation of the Jacobian for the Regge equations was in January 1992, and in March I was looking at the first general solutions for the Euclidean initial value and time evolution equations. In April I implemented the *teepee* lattice and adapted the software to C++ so that I could introduce a timelike *tag* to edge and volume arithmetic, allowing the user to choose a Minkowski signature. At the same time I was introducing the definitions of lapse and shift presented in Chapter 7 and (Galassi, 1992a), and I introduced a general mechanism to program in a fit to an analytic metric for generating the initial data.

I developed the software using GNU emacs, the GNU C++ compiler and the GNU debugger (gdb), running alternately on Silicon Graphics equiptment (at Stony Brook) and Sun workstations (in Los Alamos).

The principal *r3+1* program consists of about ten thousand lines of C++ code. It has a visual interface which makes use of the unix cursor management package *curses*, and which allows the user to probe any aspect of the lattice structure, and find out any length, area, deficit angle, connectivity information or Regge equation at any time. Any edge length in the lattice can be modified in real time by the user, a very useful feature if one wants to perturb away from a known solution.

The user can choose the type of lattice, the method of solution of the Regge equations (direct search or Newton's method), the method of calculation of the Jacobian (analytic or numerical), the lattice spacing, the lapse and shift settings, and several other parameters. The user is also prompted with choices to solve the initial value problem or to carry out time evolution steps.

Section 4.2 Hashing

A very powerful method for storing data for quick retrieval is the *hash table*. A very complete treatment of hashing can be found in Knuth (1973, vol. 3) and Tannenbaum *et al.*. Gentle explanations with less detail can be found in van Wyk and Aho *et al.*. To set the scene for describing hash tables, let us look briefly at some more common data structures. For the examples here I will use a set of n data points in which an individual record is described by a C structure which could be called `einfo`:

```
  /* structure with edge information */
int p1, p2;   /* endpoints of that edge; also the search key */
int edge_ind; /* index into the list of edge lengths */
```

A typical search will involve asking for the edge index corresponding to the end points (p, q).

If this data is stored in a straight *list*, then we search for the element with key (p', q') by examining each element of the list in to see if it matches: $p = p'$ and $q = q'$. This procedure, called a *sequential search* can take quite some time if we have a large list: it involves involve $O(n)$ retrievals and comparisons.

One improvement is to sort the list by key (one has to assign an ordering to pairs of points (p, q) to do this), and then implement a dictionary–style search called a *binary search*. In a binary search one takes the midpoint of the sorted list, checks if the search key is greater or less than the midpoint (or equal if we are lucky), and restricts the search to half of the list. This procedure is repeated recursively until we home in on the correct key. Binary search is an $O(\log_2 n)$ algorithm, which makes for extremely fast retrieval, but one must sort the list (which can be time consuming) and store it in an array, which means that it can only be used if we know beforehand how much data we will have, so we can store it all in a big array.

The fastest way of looking up data is by *table lookup*. Suppose there were a unique integer key for an edge information structure e: $k(p, q)$. Then we store the edge structure in an array element `estruct_array[`$k(p, q)$`]`. When we are looking for the record with key k we just retrieve that element. This is the fastest possible algorithm: it takes *constant* time to retrieve data, no matter how large the list, but it requires that k map injectively into the data array.

Hashing is a modification of table lookup, appropriate for cases in which the function $k(p, q)$ is not injective. If we can produce a function $h(p, q)$ which maps (p, q) to an integer in a range `[0 ...hsize-1]`, and which is almost injective (in that it scatters the data quite randomly throughout `[0...hsize-1]` with few repetitions), then we can try another scheme. We define an array `ehashtab[hsize]` of size `hsize`, and store structure e (with points (p, q)) at location `ehashtab[`$h(p, q)$`]`. The function $h(p, q)$ is called a *hash function*. So far it looks just like table lookup, but there is a catch: since $h(p, q)$ is not perfectly injective, we might have a clash where two sets of endpoints (p, q) and (r, s) map to the same value under h: $h(p, q) = h(r, s)$; so we must handle this collision in both the storage and retrival of e. There are several schemes for *collision resolution*; I will now present some of the more common ones, including the one I use in *r3+1* .

One method is to look for a free space in the table, starting with $h(p, q) + 1$, and put e in that free slot. The same would be done in searching: we look for (p, q) in the hash table starting at $h(p, q)$. If `ehashtab[`$h(p, q)$`]` is not the entry we wanted, we try `ehashtab[`$h(p, q)$`] + 1`, and `ehashtab[`$h(p, q)$`] + 2` until we find the correct entry, just as in the sequential search mentioned above. This is called resulution by *open addressing*; it is a primitive scheme that tends to cause *clustering* about a few points. There can be long sequences of "busy" cells, which will force us to frequently do long linear searches. Deletion of a record is also very difficult with open addressing because the table has to be reorganized after any element is deleted.

An improvement on open addressing is *linear probing* in which we try to place "collided" data items at `ehashtab[`$h(p, q)$`] + p`, `ehashtab[`$h(p, q)$`] + 2p` and so on. Clustering is reduced if p is a rather large number relatively prime to `hsize`. This scheme reduces clustering, and performance will be good until the table is nearly full.

Both open addressing and linear probing are static: there is no provision to make the hash table grow if new records are added. To address this problem, and to present a better solution to the problem of clustering, we can introduced collision resolution by *chaining*, where each node in the hash table is the head of a dynamic data structure, typically a linked list. A diagram of this form of storage is shown in Fig. 4.1. The hash table can grow forever because the linked lists are dynamic data structures. Of course, once the linked lists become too long then the search time increases from being approximately constant to being limited by the time it takes to search a linked list. If `hsize` is the size of the hash table, and m is the number of elements in the table, the average number of probes required for a search is $O(\texttt{hsize}/n)$, which is bounded as $n \to \infty$ if we always keep the table large enough that the load factor $\texttt{hsize}/n$ is not too close to 1. Detailed analysis of hash table performance is presented in Knuth (1973, vol. 3) and Tannenbaum *et al.*.

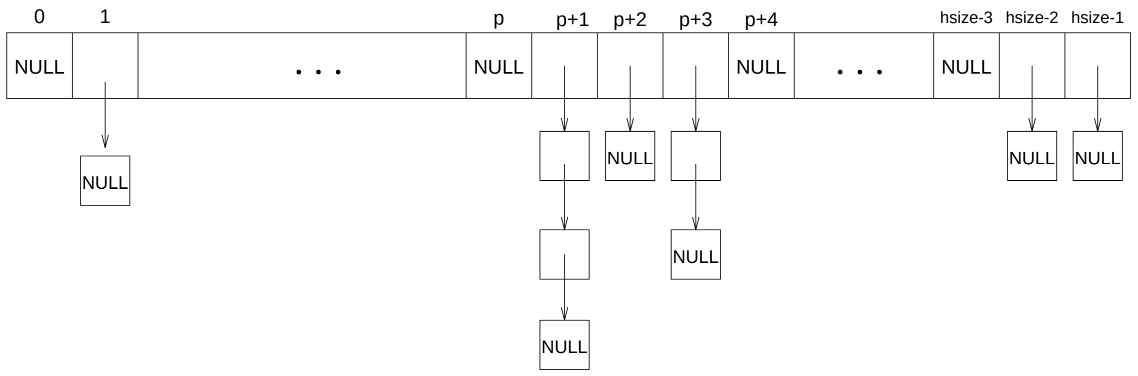

Fig. 4.1: A chained hash table with linked list collision resolution.

Similar schemes use binary trees instead of linked lists to do resolution by chaining; the difference only becomes noticeable when the number of chained records becomes large, which is usually a hint that it is time to increase the size of the hash table; so there are few cases in which a more sophisticated chain is needed.

When we decide to use hashing to store and retrieve data we must make a few choices that will have great influence on the performance of our searches. We choose the hash function h which operates on the key of record e (in our case h is a function of points p and q, which are parts of the record e); we choose a size for the hash table; and we choose a method of collision resolution. A good hash function is one that will scatter the data in such a way that few records collide. *Perfect* hash functions cannot always be found, but there are good heuristics to help choose a nearly optimal function. The size of the hash table is usually chosen to be a prime number (or a number with few divisors), and approximately the same size as the number of records we expect to have.

In *r3+1* I make use of chained collision resolution with linked lists; I choose my hash table size to be the number of records minus 1; and I use the hash function

$$h(p,q) = [p + (q << 4)] \bmod \texttt{hsize}$$

where $q << 4$ means left shifting q by 4, effectively multiplying q by $2^4 = 16$. This set of choices works nicely for the hashing of edge information by endpoint–pairs: in all the simulations I have run I never have more than three records chained in a list, and the hash table does not have too many zero entries.

Section 4.3 Storage and retrieval of *edge* information

We start with a list of simplices stored as 5–tuples $(P_0, P_1, P_2, P_3, P_4)$, where each P_i is the label representing a point. Each simplex will have ten edges $\overline{P_i P_j}$ for $i < j$, which suggests a very straightforward algorithm for generating a list of all edges in the lattice: we run through the entire list of simplices and add the ten edges contributed by each to a hash table of edges. There will be several cases in which more than one simplex contains a given edge, but it is possible, and easy, to verify if an edge identified by its endpoints P_i and P_j is already in the hash table.

The algorithm for forming the edge information hash table is quite straightforward:

(1) prepare a counter called `edge_index` which will record the position of each edge we add to the hash table; set `edge_index` to zero;
(2) loop through all simplices $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ in the manifold;
(3) loop through all pairs of points (σ_i, σ_j) where $i < j$ for simplex σ;
(4) see if (σ_i, σ_j) is already in the hash table; if it is, pass on to the next pair of points (i.e. return to step 3);
(5) otherwise form an edge information entry with points (σ_i, σ_j) and `edge_index` and add it to the hash table; now increment the `edge_index` and continue looping through all point pairs (step 3).

We now have information about evey edge in the lattice, and we want to assign lengths to all the edges. One way would be to add a field

```
double length;      /* length of this edge */
```

to the edge information structure, but it turns out to be convenient to keep all the actual edge lengths stored in an array, so I allocate an array `edgelengths[]` for this purpose, and index it by the `edge_index` value for each edge.

The value of each edgelength is then initialized to some "guess" of the value of the metric. These edge lengths can now be retrieved very quickly (at hash table lookup speed) from their endpoints (p, q) by finding the hash table entry of $h(p, q)$ and extracting the edge index, and using that to find the edgelength. This quick retrieval of edge information from the edge's endpoints was my goal in designing these data structures because *r3+1* does this kind of operation all the time.

The other operation involving edges and their endpoints is: given an edge index, what are its endpoints? This information can be obtained by straight table lookup without need for hashing, so I keep an array of point pairs indexed by the aforementioned edge index.

Section 4.4 Storage and retrieval of *triangle* information

A similar set of operations has to be carried out for the triangles in the lattice: deficit angles are associated with triangles, and they play a crucial part in calculating the Regge equations and their derivatives. Because of this we need a mechanism to retrieve triangle information from the three points that define a triangle. This is entirely analogous to the problem of obtaining edge information from the edge's endpoints, and I solve it in the same way in my software.

The program forms a hash table for triangles with a hash function defined on point *triples*. The function I use is

$$h(p,q,r) = [p + (q << 4) + (r << 8)] \bmod \texttt{trihash\_size}$$

which is completely analogous to the hash function for edges.

The triangle hash table is filled, just as the edge hash table was, by running through all simplices σ, taking triples of points σ_i, σ_j, σ_k $(i < j < k)$, and adding them to the hash table with care to avoid duplication.

Section 4.5 The triangle entourages

In calculating deficit angles the software will frequently need to calculate the sum of all dihedral angles about a given triangular hinge t. To do so it must find all the simplices that share t, and associate them with t (this is called the *entourage* of t). This is the last and most lengthy of the lattice–scanning procedures, but it only has to be done once for a given type and size of lattice, after which the entourages can stored in and read from a file.

Although slow, the algorithm for calculating the entourages is quite simple:

(1) loop through all triangles t in the lattice;
(2) loop through all simplices σ in the lattice;
(3) if σ contains t, add it to the entourage; otherwise pass on to the next simplex.

It is probable that exhanging steps (1) and (2) and adding some smarts will increase the speed of this algorithm, but I have not yet found a need to speed this one up.

Chapter 5: Formulating the Regge Equations: Calculations

The Regge equations are given in equation (2.5). They involve the deficit angles associated with triangles and the internal angles of said triangles, so to actually calculate the Regge equations we need a collection of formulæ for determining angles in our lattice. In this section I will describe how I calculate angles, and put the angles together to calculate the Regge equations. I will first discuss the Euclidean case, where the flat space metric is $g_{\mu\nu} = \delta_{\mu\nu}$ and then introduce a Minkowski signature. The next chapter describes the calculation of the Jacobian.

Section 5.1 General relativity without coordinates

Regge (1961) titled his seminal paper *General Relativity Without Coordinates.* This is because in a Regge lattice all information can be obtained from the edge lengths, and there is no need to introduce coordinate systems within each simplex. Some practitioners of Regge Calculus do indeed introduce coordinates as an aid in calculating deficit angles (Collins and Williams: 1973; Brewin: private communication), but following Wheeler (1963) and Miller (1986c), I prefer to stay away from coordinate systems and opt for the simplicity of deriving every quantity from the lattice edge lengths.

Here are some examples in two dimension of how quantities associated with triangles can be calculated from the triangle's edges. Referring to Fig. 5.1, where the edges are l_1, l_2 and a, the angle between l_1 and l_2 is θ, the half–perimeter of the triangle is $s = (l_1+l_2+a)/2$, and the area is $\mathcal{A}$:

$$\sin\theta = \frac{2l_1l_2}{\mathcal{A}} \tag{5.1}$$

$$\cos\theta = \frac{l_1^2 + l_2^2 - a^2}{2l_1l_2} \tag{5.2}$$

$$\cot\theta = \frac{l_1^2 + l_2^2 - a^2}{4\mathcal{A}} \tag{5.3}$$

$$\mathcal{A} = \sqrt{s(s-l_1)(s-l_2)(s-a)} \tag{5.4}$$

$$= \frac{1}{2!} \ \det{}^{1/2} \begin{pmatrix} a^2 & \frac{1}{2}(a^2 - l_1^2 - l_2^2) \\ \frac{1}{2}(a^2 - l_1^2 - l_2^2) & a^2 \end{pmatrix} \tag{5.5}$$

Equation (5.4) is Heron's formula which expresses the area $\mathcal{A}$ in terms of the edge lengths alone. The determinant form of this is equation (5.5), shown in Wheeler (1963) together with a demonstration of how it generalizes to higher dimensional volumes.

So the two dimensional dihedral angle (an ordinary angle, that is) can be calculated from the formula for sine or cosine (or both can be used to figure out the correct quadrant); these can in turn be calculated from edge information alone, showing that deficit angles can be obtained from the edge lengths. This generalizes to higher dimensions: contemplating

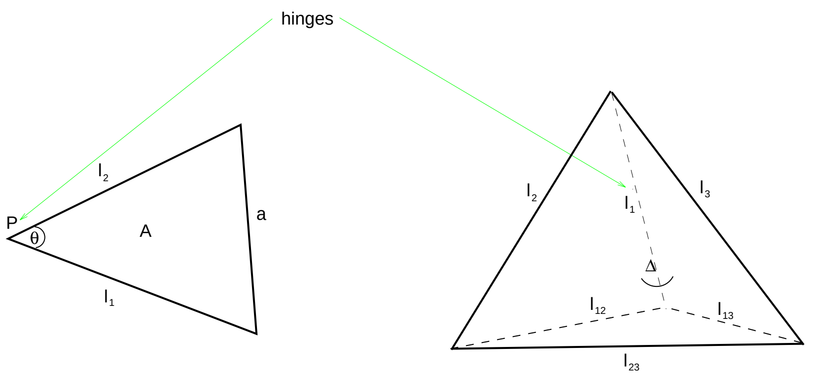

Fig. 5.1: A triangle and a tetrahedron with everything marked to guide us through writing formulæ.

the tetrahedron in Fig. 5.1: Δ_{l_1} the dihedral angle hinging on l_1, $\mathcal{A}_1$ and $\mathcal{A}_2$ are the areas of the triangles that share l_1, V is the volume of the tetrahedron. We have:

$$\sin \Delta_l = \frac{3\mathcal{A}_1\mathcal{A}_2}{2V} \tag{5.6}$$

$$V = \frac{1}{3!} \quad \det{}^{1/2} \begin{pmatrix} l_1^2 & \frac{l_1^2-l_{12}^2+l_2^2}{2} & \frac{l_1^2-l_{13}^2+l_3^2}{2} \\ \frac{l_1^2-l_{12}^2+l_2^2}{2} & l_2^2 & \frac{l_1^2-l_{23}^2+l_3^2}{2} \\ \frac{l_1^2-l_{13}^2+l_3^2}{2} & \frac{l_2^2-l_{23}^2+l_3^2}{2} & l_3^2 \end{pmatrix} \tag{5.7}$$

Let us increase dimension once more: consider a 4–simplex, and one of its triangles t. Let Δ_t be the dihedral angle formed by the two tetrahedra hinging on t, then:

$$\sin \Delta_t = \frac{4V_1^{(3)}V_2^{(3)}}{2V^{(4)}} \tag{5.8}$$

where $V_1^{(3)}$ and $V_2^{(3)}$ are the tetrahedra defining the dihedral angle. So far so good, but the formula for four dimensional volumes $V^{(4)}$ is extremely complicated:

$$V^{(4)} = \frac{1}{4!} \quad \det{}^{1/2} \begin{pmatrix} l_1^2 & \frac{l_1^2-l_{12}^2+l_2^2}{2} & \frac{l_1^2-l_{13}^2+l_3^2}{2} & \frac{l_1^2-l_{14}^2+l_4^2}{2} \\ \frac{l_1^2-l_{12}^2+l_2^2}{2} & l_2^2 & \frac{l_2^2-l_{23}^2+l_3^2}{2} & \frac{l_2^2-l_{24}^2+l_4^2}{2} \\ \frac{l_1^2-l_{13}^2+l_3^2}{2} & \frac{l_2^2-l_{23}^2+l_3^2}{2} & l_3^2 & \frac{l_3^2-l_{34}^2+l_4^2}{2} \\ \frac{l_1^2-l_{14}^2+l_4^2}{2} & \frac{l_2^2-l_{24}^2+l_4^2}{2} & \frac{l_3^2-l_{34}^2+l_4^2}{2} & l_4^2 \end{pmatrix} \tag{5.9}$$

This determinant has many terms and is expensive to calculate, but it is a necessary step in the calculation of deficit angles. There is also a cosine formula in higher dimensions, analogous to that in two dimensions which comes from the cosine theorem. In four dimensions it is:

$$\cos\Delta_t = \frac{1}{36V_1^{(3)}V_2^{(3)}V_2} \times \det\begin{pmatrix} \frac{l_3^2-l_{34}^2+l_4^2}{2} & \frac{-l_1^2+l_{13}^2-l_{34}^2+l_4^2}{2} & \frac{-l_2^2+l_{23}^2-l_{34}^2+l_4^2}{2} \\ \frac{-l_1^2+l_{14}^2+l_3^2-l_{34}^2}{2} & \frac{l_{13}^2+l_{14}^2-l_{34}^2}{2} & \frac{-l_{12}^2+l_{14}^2+l_{23}^2-l_{34}^2}{2} \\ \frac{-l_2^2+l_{24}^2+l_3^2-l_{34}^2}{2} & \frac{-l_{12}^2+l_{13}^2+l_{24}^2-l_{34}^2}{2} & \frac{l_{23}^2+l_{24}^2-l_{34}^2}{2} \end{pmatrix} \tag{5.10}$$

We can use either the cosine or the sine formula to determine the hyperdihedral angles. Knowing beforehand in which quadrant the angle will be is useful because we can choose either one. If we don't know *a priori* then we can use both sine and cosine values to determine the quandrant. This gets more complicated when I introduce a Minkowski signature.

These are the principal formulæ we need to calculate the deficit angles, Regge equations and (when we take their derivatives) the Jacobian matrix. They all depend only the edge lengths of the lattice, and there has been no need to introduce any other considerations.

Section 5.2 **Minkowski signature**

The formulæ I presented above are all valid in Euclidean space, but need to be modified in a Minkowski setting. The easiest way to do this is to assign a tag to all timelike edges and define a new arithmetic between geometric quantities. The product of timelike quantities is a spacelike quantity with negative magnitude, the product of timelike with spacelike quantities is timelike, and the product of spacelike quantities is spacelike. This arithmetic is identical to that of real/imaginary numbers: let the timelike tag be a multiplicative factor of $\sqrt{-1}$, carry out the imaginary number arithmetic, and the results will be correct.

This allows us to take all the equations presented above and use them, with the understanding that some of the edge lengths involved are imaginary, and the operations are not real number operations any more. When setting initial values for the edge lengths we set the straight timelike edges to be imaginary. The diagonal time–directed edges will then be real† and the edges on the spacelike surface will be real.

Examining Eqs. 5.1–5.9 we see that they always depend on the *squares* of the edge lengths. The volume formulæ have a square root, but the quantity in the square root will always be a pure real number, either positive or negative. This means that we will always have either purely real or purely imaginary quantities. Wheeler (1963) suggests that we assign real areas and real deficit angles to timelike triangles, and imaginary areas and deficit angles to spacelike triangles. This means that the product $A_t\epsilon_t$ will always be

† This is true if the lapse is smaller than the characteristic spacing, which should almost always be the case; see the discussion of the Courant condition in the Lapse and Shift paper.

real, and the Regge action $\sum_t A_t\epsilon_t$ will be real too, and the Regge equations will be real or imaginary according to whether they are associated with a real or imaginary edge.

Having all these elements we can calculate the deficit angle associated with an arbitrary triangle t:

(1) start with $\epsilon_t = 2\pi$;
(2) loop through the entourage of simplices σ_m surrounding t
(3) calculate the hyperdihedral angle $\Delta_{\sigma_m,t}$ from the sine and cosine formulæ above and subtract it from ϵ_t.

And here is the algorithm that calculates the Regge equation for an edge e_i, assuming that the deficit angles have already been calculated and stored in an array `angles[]`.

(1) initialize the i^{th} Regge equation $E_i = 0$;
(2) loop through all triangles t that hinge on edge e_i; the deficit angle is $\epsilon_t =$ `angles`[`t`];
(3) obtain the cotangent of the angle in t opposite to e_i—call it `cot_theta`;
(4) add `cot_theta`×`angles[t]` to the Regge equation E_i.

After cycling through the entourage of e_i we are left with the Regge equation E_i.

Now we know how to calculate the Regge equations, let us try to solve them!

Chapter 6: Solving the Regge equations

In Chapter 4 I described how *r3+1* makes all lattice information readily available so that it can formulate the Regge equations; in Chapter 5 I showed how deficit angles and Regge equations are calculated. Here I describe the procedures I use to solve the equations. In Section 6.1 I start with a discussion of what data should be specified and what should be solved, and I translate that into the data structures used to specify effective equations and unknowns. In Section 6.2 I outline Newton's method in multidimensions, and show how it is used. Finally in Section 6.3 I describe my dartboard procedure for calculating the Jacobian matrix.

Section 6.1 What to specify and what to solve

One important question to be solved is which edges need to be specified and for which must we solve the equations. In my introductory chapter on Regge Calculus I mentioned that I do not solve all the equations for all the edges, rather I form two mappings that pick out the dynamical equations and unknowns from the entire `edgelist[]`.

To give an example, let us say that we are doing time evolution of an A–type teepee with $N_3 = 4 \times 4 \times 4 = 64$ points on the spacelike grid. Then for the two spacetime sandwiches we are considering we have $22N_3 + 15N_3 = 2368$ edges involved in our calculations. But in the teepee lattice (see the paper in Chapter 7) we only have 11 Regge equations (plus 4 lapse and shift conditions) for 15 unknowns, so we form a map which translates the 11 equations into edge indices, and another map which translates the 15 unknowns into edge indices. I call these maps the `eqn_map[]` and `unk_map[]`.

These maps are specified by the user and they form the only input to *r3+1* other than the initial geometry. I am working on having the software determine these maps on its own by scanning the lattice, but there are still details to be worked out. The principle difficulty is deciding what to do when some points in the lattice have a different number of edges emanating, as in the impurities in the S^3 lattice described earlier. One of my results, presented in the paper included in Chapter 7, shows that dynamical and constraint equations can be swapped, which is the first step in solving this issue; the rest are details.

Section 6.2 Numerical methods

The most common technique for solving coupled systems of algebraic equations is Newton's method (Press *et al.* 1988). If we have n algebraic equations E_i with n unknowns u_j, Newton's method arrives at a solution by finding tangent hyperplanes to the surfaces defined by the equations and following the slope of those surfaces until we reach the solution; it is quite analogous to Newton's method in one dimension.

In one dimension, if we have a function $E(u)$ and we want to find a root for E, the algorithm is:

(1) choose an initial guess to the root, call it $u^{(0)}$;

(2) compute the next approximation to the root

$$u^{(0)} = u^{(0)} - \left(\frac{dE(u^{(0)})}{dt}\right)^{-1} E(u^{(0)});$$

(3) continue iterating until $E(u^{(k)}$ is close enough to zero.

In n dimensions we replace the derivative dE/du with the Jacobian matrix of the Regge equations:

$$J_{ij} \equiv \frac{\partial E_i}{\partial u_j}. \tag{6.1}$$

We then start with an initial guess for the values of the unknowns $u_j^{(0)}$, and find the next approximation $u_j^{(1)}$ by solving the system of equations

$$\sum_j J_{ij}(\vec{u}_0)dx_j = -E_i(\vec{u}_0) \tag{6.2}$$

and computing

$$u_j^{(1)} = u_j^{(0)} + dx_j. \tag{6.3}$$

This very effective method requires the solution of a system of linear equations (equation 6.2), and the calculation of the derivatives of the equtions to be solved (equation 6.1).

The Regge equations E_i are to be accompanied by some extra equations which determine the lapse and shift at each point. The equations that determine lapse and shift are derived and implemented in the paper which is included in Chapter 7 (Galassi, 1992a); these equations can be added to the usual Regge equations to carry out time evolution. The derivatives of these equations with respect to the unknown edge lengths must then be added to the Jacobian.

Section 6.3 Calculating the Jacobian for the Regge equations

The calculation is quite messy, but it has to be done:

$$J_{ij} = \frac{\partial E_i}{\partial u_j} \tag{6.4}$$

$$= \frac{\partial \sum_{t|i} \epsilon_t \cot\theta_{t,i}}{\partial u_j}. \tag{6.5}$$

This can be broken into:

$$\frac{\partial \cot\theta_{t,i}}{\partial u_j} = \frac{\partial \frac{l_1^2+l_2^2-a^2}{4\mathcal{A}_t}}{\partial u_j} = \frac{2\left(l_1\delta_{l_1u_j} + l_2\delta_{l_2u_j} - a\delta_{au_j}\right)\mathcal{A}_t + \left(l_1^2+l_2^2-a^2\right)\frac{\partial\mathcal{A}_t}{\partial u_j}}{4\mathcal{A}_t^2} \tag{6.6}$$

The derivative of the area is a primitive function in *r3+1* and is easily implemented. The part which is difficult to program is the Kronecker δ which determines if the edge u_j is a part of the triangle t. This is another area in which *r3+1* uses its lattice information to figure out which of these terms is nonzero.

The other term, with the derivative of the deficit angle, is the messy one:

$$\frac{\partial \epsilon_t}{\partial u_j} = \frac{\partial\left(2\pi - \sum_{\sigma_{|t}} \Delta_{\sigma,t}\right)}{\partial u_j} = -\sum_{\sigma_{|t}} \frac{\partial \cos^{-1}\left(\frac{M}{36V_1V_2}\right)}{\partial u_j}$$

where M is the determinant of the 3×3 matrix in equation (5.10) and V_1 and V_2 are the volumes of the tetrahedra which determine the hyperdihedral angle. Now we need to calculate the derivative of the hyperdihedral angles $\Delta_{\sigma,t}$:

$$\frac{\partial \cos \Delta_{\sigma,t}}{\partial u_j} = -\sin \Delta_{\sigma,t} \frac{\partial \Delta_{\sigma,t}}{\partial u_j}$$

but, at the same time

$$\frac{\partial \cos \Delta_{\sigma,t}}{\partial u_j} = \frac{\partial \frac{M}{36V_1V_2}}{\partial u_j} = \frac{V_1V_2\frac{\partial M}{\partial u_j} - M\frac{\partial(V_1V_2)}{\partial u_j}}{36V_1^2V_2^2}. \tag{6.7}$$

Putting it all together we have:

$$\frac{\partial \Delta_{\sigma,t}}{\partial u_j} = \frac{-1}{48\mathcal{A}_t V^{(4)}(\sigma)} \left\{ \frac{\partial M}{\partial u_j} - \frac{M}{V_1}\frac{\partial V_1}{\partial u_j} - \frac{M}{V_2}\frac{\partial V_2}{\partial u_j} \right\}. \tag{6.8}$$

Notice that the terms $\partial V/\partial u_j$, and $\partial M/\partial u_j$ will also contain Kronecker δs once they are expressed in terms of the edge lengths.

Section 6.4 Implementation of the Jacobian

With equations (6.6) and (6.8), and the formulæ developed in Chapter 5, we can calculate the Jacobian. The way I do this in *r3+1* is indirect. It is inefficient to choose an individual component J_{ij} and calculate it alone, so my procedure operates on an entire row at once, filling in entries along that row in a sparse order. Here is the algorithm, which is used unmodified for every type of situation my program handles, be it the straightforward lattice, the teepee lattice, the initial value problem, or time evolution:

(1) initialize the Jacobian $J_{ij} = 0$ for all i, j;
(2) loop through all equations, call the index `eq_i` and figure out which triangles hinge on the edge corresponding to `eq_i`, call that list `t_entour[]`;
(3) cycle through the triangles `t_entour[k]`, get the three edge lengths in `t_entour[k]`; they are `eq_i` (the same as the equation edge), `e_1` and `e_2`;
(4) the components of J which will get contributions are $J_{eq_ieq_i}$, $J_{eq_ie_1}$ and J_{ie_2}, add to them the appropriate quantity from the derivative of the cotangent term in equation (6.6).

Now the inner loop in which we go through the simplices in the entourage of the triangles `t_entour[k]`:

(5) loop through all simplices `simp_tour[m]` surrounding triangle `t_entour[k]`, and obtain all the edges involved in that simplex, together with its volume $V^{(4)}$ and the volumes of the two tetrahedra that determine the dihedral angle: V_1 and V_2.

(6) the elements $J_{eq_i u_j}$ will get contributions from all edges u_j which are in the simplex `simp_tour[m]`, and the contributions are calculated from equation (6.8);

(7) the last step is to fill in the Jacobian matrix with the derivatives of the lapse and shift conditions; this is straightforward because the lapse and shift conditions are hardcoded, so I just type in the appropriate formula.

The calculation of the Jacobian matrix performed by *r3+1* is the first ever, not surprisingly now that we see what is involved in calculating and implementing J_{ij}.

Chapter 7: Bianchi Identities, Constraints, Lapse and Shift ...

In continuum general relativity there are ten Einstein equations

$$G_{\mu\nu} = 0$$

with which we solve for the ten independent components of the metric tensor $g_{\mu\nu}$. This looks quite straightforward at first: ten equations for ten unknowns. But closer examination of the structure of the Einstein equations shows that the ten $G_{\mu\nu}$ components are not all independent. We have contracted Bianchi identities

$$D_\mu G^{\mu\nu} = 0 \tag{7.1}$$

which give four differential relationships between the equations, thus reducing the number of independent Einstein equations to six. How do we determine all ten components of the metric with only six equations? As it turns out there are only six unknown components of the metric because when we choose a coordinate system we are imposing four conditions on the metric. So a more accurate summary is that we have six equations plus four coordinate conditions with which to determine the ten metric components. The six equations usually used to solve for the metric are the G_{ij} components which are second order in time. These are called the *dynamical* equations. The other four components $G_{0\mu}$ are spectators; these are called the *constraints.*

Section 7.1 The skeletal Bianchi identities

In Regge Calculus the nature of the Bianchi identities is not as clear. The ordinary Bianchi identities

$$D_\lambda R_{\mu\nu\rho\sigma} + D_\sigma R_{\mu\nu\lambda\rho} + D_\rho R_{\mu\nu\sigma\lambda} = 0 \tag{7.2}$$

or, more compactly:

$$R_{\mu\nu[\rho\sigma;\lambda]} = 0$$

were first discussed in skeleton form by Regge himself (1961), and he showed that they have a topological interpretation and are exactly satisfied on a Regge lattice (see also Roček and Williams 1981).

But the skeletal *contracted* Bianchi identities do not follow from the skeletal *ordinary* Bianchi identities. The contracted identities were first formulated by Miller (1986a) who studied the Regge lattice from the point of view of Cartan's "boundary of a boundary" principle. In the continuum this principle allows a more deeply geometrical derivation of both ordinary and contracted Bianchi identities (see Misner, Thorne and Wheeler, 1974, Chapter 15). Miller applied the boundary of a boundary principle to a Regge lattice and found that, because finite rotations do not commute, we should expect the contracted Bianchi identities to be approximately but not exactly satisfied. This would imply that we have diffeomorphism freedom in Regge Calculus, and we can choose four lapse and shift conditions for time evolution, but we should only expect the evolution to be approximate:

the constraints would be “displeased” spectators and not satisfied exactly. I have shown (see the *Lapse and Shift* paper below) that the constraints are more closely satisfied as the lattice is refined; more precisely, the constraints are proportional to the *third* power of the lattice spacing.

For numerical relativity the question of diffeomorphism freedom is particularly important, in part because it is not an exact symmetry any more (!), but most of all because we want our coordinate grid (or, in Regge Calculus, our simplicial decomposition) to be *adaptive*, in other words we want to focus more coordinate points (or simplices) around the interesting parts of spacetime (for example, the location of the black hole collision). This is achieved by choosing lapse and shift adaptively so that they focus the coordinates toward the region of interest, as shown in Fig. 7.1. But of course, lapse and shift can only be chosen freely if we have diffeomorphism freedom.

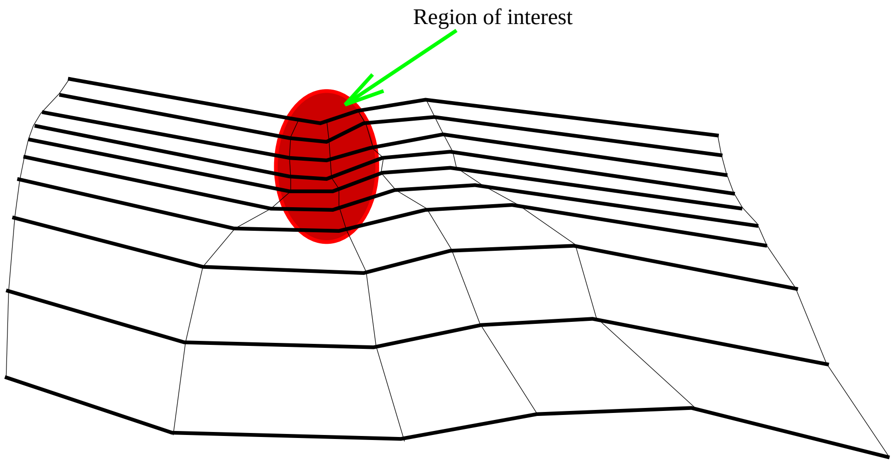

Fig. 7.1: lapse and shift focusing on the interesting region of spacetime.

I set out to study the nature of diffeomorphism freedom in Regge Calculus using the *r3+1* software, and found some very interesting results which are presented in my paper *Lapse and Shift in Regge Calculus* (Galassi, 1992a). Since this paper validates my methods and software, and provides the first application of *r3+1* which has been studied exhaustively, I include it verbatim in this dissertation to conclude this chapter.

In this paper I give a definition of lapse and shift in Regge Calculus, I implement that definition, I examine the behaviour of convergence on choice of lapse and shift, I verify that the choice of constraints is arbitrary, I examine the behaviour of the constraints for flat and curved spacetimes, and I quantify the dependence of constraints on the lattice spacing. But enough said here: the paper is quite complete.

LA–UR–92–2611
ITP–SB–92–44

Lapse and Shift in Regge Calculus

Mark Galassi[†]
Institute for Theoretical Physics, SUNY at Stony Brook
Stony Brook, NY 11794
and
Theoretical Astrophysics Group, Theoretical Division,
Los Alamos National Laboratory, Los Alamos, NM 87545

ABSTRACT
I use my 3+1 dimensional Regge Calculus code to give the first explicit verification that there is an approximate diffeomorphism invariance in Regge Calculus. In particular I evolve a neighbourhood in a spacelike hypersurface numerically, and show that one may choose lapse and shift freely. I use my numerical approach to analyze the structure of this discrete diffeomorphism group. I also study the constraints in Regge Calculus, and find that they are proportional to the *third* power of the lattice spacing.

PACS numbers: 04.20.Cv, 04.20.Me
Submitted to *Physical Review D*, August 7, 1992; accepted Dec. 2, 1992; appeared April 15, 1993.

1. Regge Calculus and Numerical Relativity

Recently there has been a great amount of effort aimed at solving Einstein's equations numerically in support of the proposed Laser Interferometric Gravity Wave Observatory (LIGO) [1,2]. Several groups are working on the three–dimensional black hole collision problem [3,4,5]. This problem is especially interesting for the following reasons.

- One avoids the complication of introducing source terms (matter) into the equations.
- It has a relatively small parameter space: a black hole is completely described by its mass, charge, spin, position and momentum.
- It will provide experimental evidence for the existence of black holes, if such a gravity wave signal is detected by LIGO.

The method used prevalently in solving Einstein's equations numerically is finite differencing [6,7,8]. Another method which discretizes Einstein's equations is Regge Calculus, in which a simplicial lattice is used to approximate a spacetime solution of General Relativity. Here the Regge equations — a simplicial analogue of the Einstein equations — are used to solve for the length of the edges in the lattice, and thus determine the lattice spacetime geometry. (See Regge's original paper [9]; reviews can be found in [10,11,12].)

[†] email: rosalia@max.physics.sunysb.edu

Miller [13] has shown that Regge Calculus meets the few broad criteria used to define the finite element method [14].

In the past year I have developed formalisms and software to solve the Regge equations without imposing symmetry, both to obtain initial value data [15] and to carry out time evolution [16]. Regge Calculus has still to be proven viable in tackling realistic 3+1 dimensional problems. Toward this goal, I study in this paper the diffeomorphism structure of Regge Calculus — an essential analysis if we are to understand the stability and accuracy (e.g. the conservation of energy–momentum) of this approach.

In this paper I use my software to examine the structure of the Regge equations without imposing special conditions: I solve the full set of equations without any simplification. In Sec. 2 I describe the type of lattice I use to do time evolution. In Sec. 3 I give my definition of lapse and shift in Regge Calculus. Sec. 4 briefly describes the time evolution equations (a scheme to be described more fully in [17]). In Sec. 5 I discuss Bianchi identities and constraints in Regge Calculus. Sec. 6 gives the essential elements of the numerical formalism used, and in Secs. 7, 8, 9 and 10 I describe my numerical runs and present data which supports my main conclusions:

- Four of the equations at each vertex can be treated as constraints.
- Four conditions per vertex are freely specifiable, corresponding to a free choice of lapse and shift. There are some limitations on the choice of lapse and shift due to Courant instabilities, physical timescales and numerical considerations. I discuss them briefly.
- When lapse and shift values are specified, the gauge is uniquely fixed.
- Once the dynamical equations are solved, the constraint equations are also satisfied up to third order in the lattice spacing.

2. Lapse and Shift: My Lattice Geometry

In this section I will describe the lattice geometry used in my simulation. I assume here that I have already evolved N time steps to the N^{th} spacelike hypersurface, and I wish now to evolve a neighborhood of a point on this surface. However, before examining the evolution procedure, it will be useful to describe the lattice geometry of the spacelike hypersurfaces.

In Regge Calculus each three dimensional spacelike hypersurface is composed of tetrahedra. The interior of each tetrahedron is a section of flat Euclidean space. The geometry of each tetrahedron is uniquely fixed by the values of its six edgelengths. Consequently the three–geometry of a spacelike hypersurface is determined by the collection of all the edgelengths of its tetrahedral tiles. It is possible, and convenient numerically, to tile a three geometry uniformly with tetrahedra. In other words, I choose a tetrahedral lattice such that each vertex has the same number of edges emanating from it. This is commonly referred to as a "regular lattice" in finite element language[†].

[†] While it is possible to form a regular lattice with R^3 or T^3 topology, it is not possible to tile a three-sphere regularly with an arbitrarily large number of tetrahedra. I have methods for handling other topologies (where one must deal with impurities), but in this paper I will consider only R^3 and T^3 topologies.

To generate such a regular tetrahedral lattice with an arbitrary number of cells, I start with a cubic lattice and insert diagonal braces within each cube (3 face diagonals and 1 body diagonal), as shown in Fig. 1a. I now have a collection of tetrahedra which fit together to form a flat 3–torus. I can obtain curved spacelike surfaces by varying the edgelengths away from their flat space values. The advantage of this three dimensional lattice is that I can generate arbitrarily high resolution *simplicial* lattices: I start from a high resolution cubic lattice, and add diagonals according to my prescription. This lattice is called the Quantity Production Lattice (QPL).

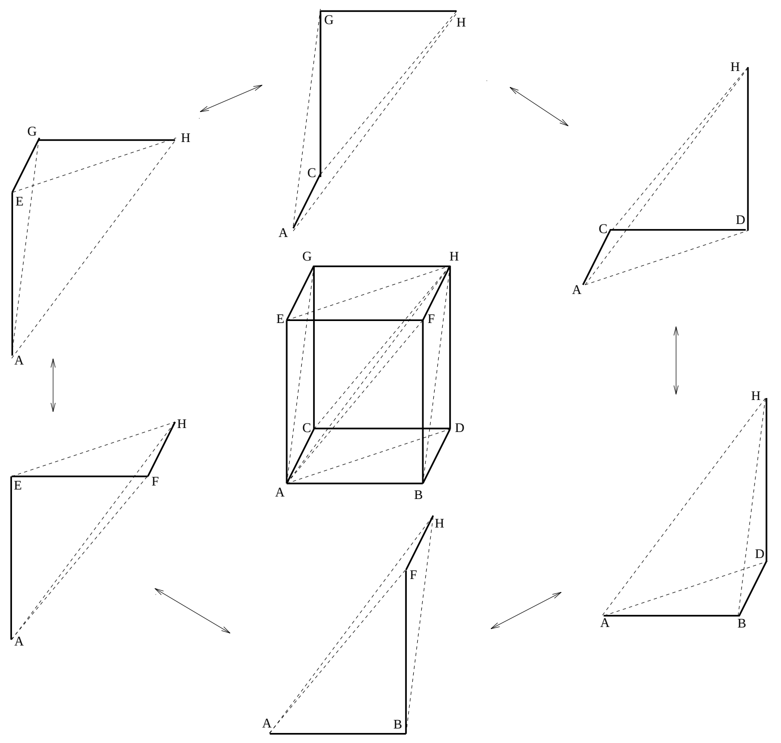

Fig. 1a: The Quantity Production Lattice: a single cube broken out into the tetrahedra that form it.

I then assign lengths to the edges in this lattice. Rather than "inheriting" the lengths from the cube in Fig. 1a, which would result in a complex of rectangular tetrahedra, I use a scheme which yields *isosceles* tetrahedra. Assign length Δl to all the cube face diagonals (AD, AF, AG, etc…), and assign length $\Delta l\sqrt{3}/2$ to the cube edge and body diagonals (this is shown in Fig. 1b; see also [13] and [18]). The isosceles tetrahedra have a non–degenerate reciprocal lattice, which presents some advantages numerically.

The spacetime geometry sandwiched between two successive spacelike hypersurfaces is composed of simplices, each of which has a flat Minkowski interior. This simplicial structure is formed using an algorithm developed recently by Barrett et. al [17]. This

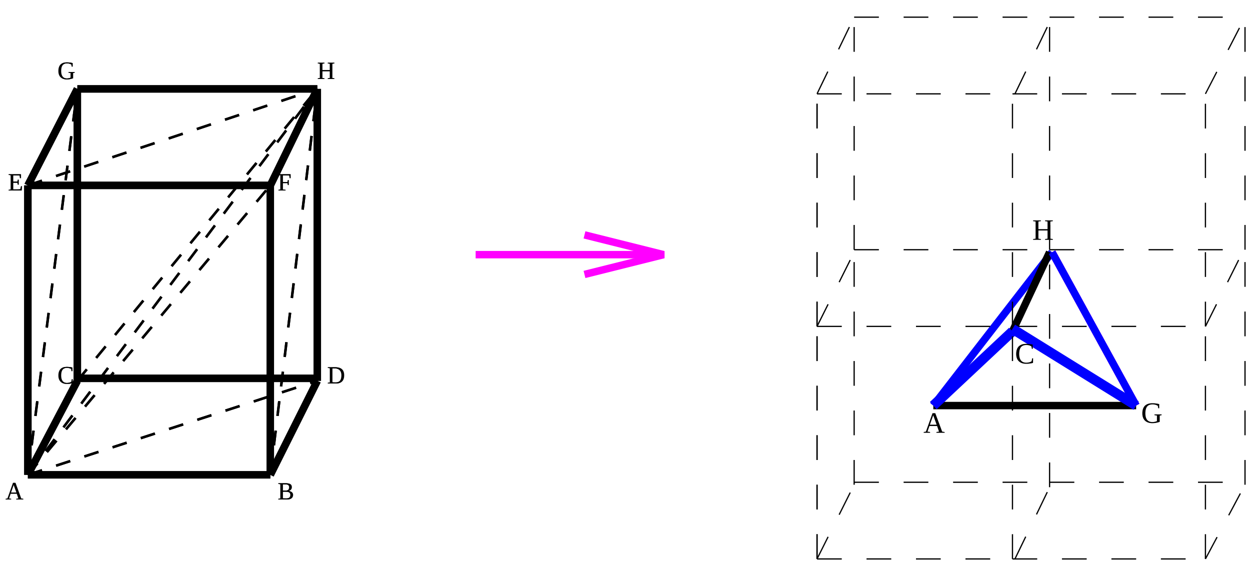

Fig. 1b: The Quantity Production Lattice, with edge lengths set so as to yield isosceles tetrahedra. One tetrahedron (ACGH) is shown here.

decoupling algorithm is a generalization of a method proposed originally by Sorkin [19,20]. Here I outline the procedure.

Step 1. Label the vertices of the QPL lattice of the $N^{\rm th}$ spacelike hypersurface with the letters A, B, C, D, E, F, G and H as shown in Figs. 1a and 1b. Here the (x, y, z) coordinates of vertex A are all even, i.e. $\{A\} = \{(2i, 2j, 2k),\ \text{for } i, j, k \in\ Z\}$. Let $\{B\} = \{(2i + 1, 2j, 2k)\}$, where the x component of B is odd. Then $\{C\} = \{(2i, 2j + 1, 2k)\}$, $\{D\} = \{(2i + 1, 2j + 1, 2k)\}$, $\{E\} = \{(2i, 2j, 2k + 1)\}$, $\{F\} = \{(2i + 1, 2j, 2k + 1)\}$, $\{G\} = \{(2i, 2j + 1, 2k + 1)\}$, and $\{H\} = \{(2i + 1, 2j + 1, 2k + 1)\}$. There are now vertices of eight kinds.

Step 2. Pick an A vertex, say $A = (2i, 2j, 2k)$. Erect a timelike edge from A into the future to a point A'. This A' will be the corresponding vertex on the $(N + 1)^{\rm th}$ spacelike hypersurface.

Step 3. There are fourteen spacelike edges emanating from the A–type vertex on the Nth surface. Point A is connected to two B's, two C's, ..., and two H's. Connect A' to each of these fourteen vertices by diagonal braces. Here, I have connected A' to its respective B, C, D, E, F, G, and H vertices with diagonal braces. This forms a teepee–like structure above each A vertex. I have generated one new 4–simplex for each tetrahedron sharing a vertex A — twenty four in total (see Fig. 2).

Step 4. Repeat Step 2, and 3 for each i, j and k in the lattice. At the end of this pass I have a collection of four–dimensional teepees centered at each A–type point.

Step 5. Erect a timelike edge from B into the future to a point B'.

Step 6. Connect B' to its respective A', C, D, E, F, G, and H vertices with fourteen braces. Notice that brace $A'B'$ is a spacelike edge in the $(N + 1)^{\rm th}$ spacelike hypersurface.

I continue this process through the remaining six vertices. When I get to vertex H' it will be connected solely by fourteen spacelike edges (of the $(N + 1)^{\rm th}$ spacelike hypersurface) to its neighbouring A', B', ..., and G' vertices. I will have generated a new spacelike hypersurface, and the geometry between the two spacelike hypersurfaces will be composed of 4–simplices.

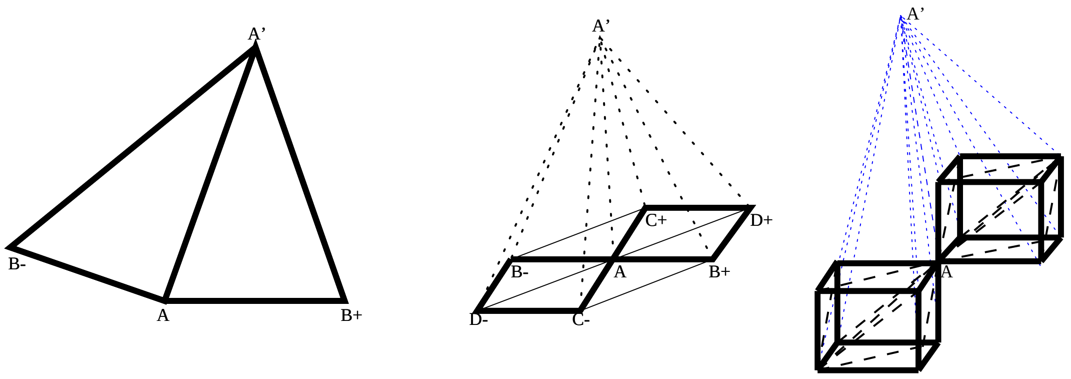

Fig. 2: An A–type vertex teepee in 1+1, 2+1 and 3+1 dimensions, respectively. Some spacelike diagonals are omitted in the 3+1 figure.

This lattice construction (which I call the *teepee* lattice, or the *decoupling* lattice) effectively decouples the Regge equations used for time evolution into an eight–stage evolution procedure in which at each stage one solves many sets of 15 equations for 15 unknowns[†]. In particular, I first solve all the A–type vertices, then all the B–type, ..., and finally all the H–type vertices. The A–type vertices can be evolved using only information contained between the N^{th} and $(N-1)^{th}$ spacelike hypersurfaces. The B–type vertices use in addition the information from A–type teepees. The C–type uses A and B teepee information and so forth...

3. Lapse and Shift: Their Definition

For the purpose of this paper it will suffice to examine the evolution of a single A–type vertex with coordinates $\{i,j,k\}$. In this section I will define the lapse and shift vector for the teepee–like structure above vertex A. The definition of lapse and shift is more transparent in 1+1 dimensions, and is generalizable to 3+1 dimensions. Therefore, I refer the reader to Fig. 3 and define the lapse and shift vectors. The vector $\overrightarrow{AA'}$ connecting vertex A to teepee summit A' can be decomposed into a component orthogonal to vector $\overrightarrow{AB}_+$ and a component along $\overrightarrow{AB}_+$:

$$\overrightarrow{AA'} = \underbrace{\overrightarrow{OA'}}_{\text{Lapse}} + \underbrace{\overrightarrow{AO}}_{\text{Shift}} \tag{1}$$

[†] Before implementing this decoupling scheme I used to do time evolution by solving $\approx (11N \times 11N)$ matrix systems, where N is the number of vertices on a spacelike surface. Using this method I solve N separate 11×11 systems (actually 15×15 when I include the lapse and shift conditions).

These two components define the lapse and shift, respectively.

$$\begin{pmatrix}\text{Lapse from}\\ A \text{ to } A'\end{pmatrix} = \overline{OA'} = \frac{2\mathcal{A}_{AA'B_+}}{\overline{AB}_+}, \tag{2}$$

and

$$\begin{pmatrix}\text{Shift from}\\ A \text{ to } A'\end{pmatrix} = \overrightarrow{AO} = \frac{\overline{AO}\ \overrightarrow{AB}_+}{\overline{AB}_+} = \frac{\overline{AA'}^2 + \overline{AB}_+^2 - \overline{A'B}_+^2}{2\ \overline{AB}_+^2}\ \overrightarrow{AB}_+, \tag{3}$$

where $\mathcal{A}_{AA'B_+}$ is the area of triangle $(AA'B_+)$. This definition corresponds precisely to the continuum definition.

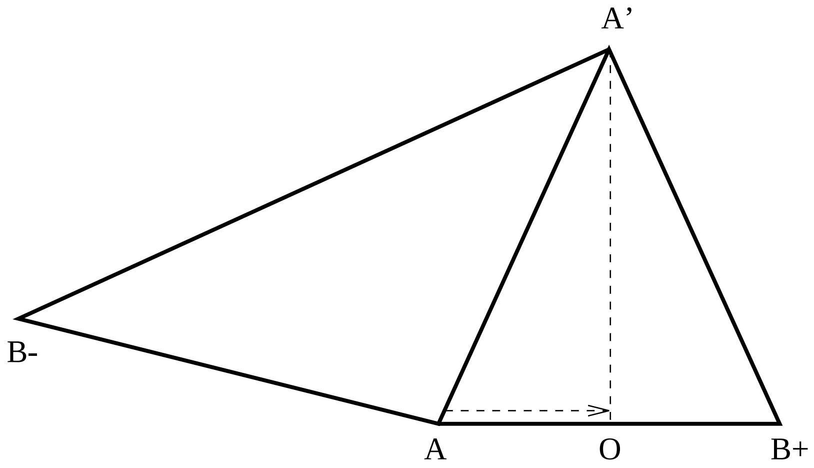

Fig. 3: An A–type vertex (in 1+1 dimensions) with lapse and shift marked.

The equations for lapse and shift given above depend only on the edgelengths of the right–most triangle $(AA'B_+)$, and do not depend on $\overline{AB}_-$ or $\overline{B_-A'}$. I could have derived the lapse and shift using a tangent plane generated at A defined as a weighted average of $\overrightarrow{B_-A}$ and $\overrightarrow{AB}_+$; however, this refined definition will introduce corrections to Eqns. (2–3) only of higher order in the continuum limit (see the Appendix). Therefore, I avoid the added complexity in favor of the more simplistic definition given above, unless a higher–order definition is needed. In a real sense, it is equivalent to the difference in Newton's (rectangle) approximation to an integral versus the trapezoidal approximation.

This definition of lapse and shift requires a well defined normal at each vertex in the 3–geometry. In Regge Calculus each spacelike hypersurface is made of tetrahedra, and a tetrahedron in such a hypersurface defines a normal direction, since it is a section of tangent space (the interior of the tetrahedron is flat by construction).

To define the lapse and shift vector in 3+1 dimensions I first pick an A–type vertex in the N^{th} spacelike hypersurface. Then by construction (the QPL lattice described in Sec. 2) there will be twenty four tetrahedra sharing vertex A — twenty four normals upon which I may define the lapse and shift. Following the argument forwarded in the Appendix I choose, without loss of generality, one of these tetrahedra to define the normal to the spacelike hypersurface at A. I choose tetrahedron $(AC_+D_+H_+)$ as representing the tangent space at A (this is the 3+1 analogue of choosing edge (AB_+) as the tangent space of A in the 1+1 dimensional example illustrated above).

The ten edgelengths of simplex $(A'AC_+D_+H_+)$ uniquely define the lapse and shift at vertex A. In particular, and following the 1+1 dimensional example, the vector $\overrightarrow{AA'}$ connecting vertex A to the teepee summit A' can be decomposed as follows:

$$\begin{aligned}\overrightarrow{AA'} = & \underbrace{\overrightarrow{AA'}\cdot\overrightarrow{AC}}_{\text{Shift Along }\overrightarrow{AC}} \frac{\overrightarrow{AC}}{\overline{AC}^2} + \underbrace{\overrightarrow{AA'}\cdot\overrightarrow{AD}}_{\text{Shift Along }\overrightarrow{AD}} \frac{\overrightarrow{AD}}{\overline{AD}^2} \\ & + \underbrace{\overrightarrow{AA'}\cdot\overrightarrow{AH}}_{\text{Shift Along }\overrightarrow{AH}} \frac{\overrightarrow{AH}}{\overline{AH}^2} + \underbrace{\overrightarrow{AA'}\cdot\overrightarrow{OA'}}_{\text{Lapse}} \frac{\overrightarrow{OA'}}{\overline{OA'}^2}. \end{aligned} \tag{4}$$

The point O is the image of vertex A' when projected onto the the tangent plane defined by tetrahedron $(AC_+D_+H_+)$ along the normal vector. The coeficient α is the lapse from A to A', and the three coeficients β_C, β_D, β_H are the components of the shift vector in the triad base defined by the same tetrahedron.

$$\begin{pmatrix}\text{Lapse from}\\ A \text{ to } A'\end{pmatrix} = \alpha = \begin{pmatrix}\text{Altitude of}\\ \text{Simplex } (A'ACDH)\end{pmatrix} = \frac{\overrightarrow{AA'}\cdot\overrightarrow{OA'}}{\overline{OA'}} = \frac{4\,{}^{(4)}V_{A'ACDH}}{{}^{(3)}V_{ACDH}} \tag{5}$$

$$\begin{pmatrix}\text{Shift Component}\\ \text{along } \overrightarrow{AC}\end{pmatrix} = \frac{\beta_C\overrightarrow{AC}}{\overline{AC}} = \overrightarrow{AA'}\cdot\overrightarrow{AC}\ \frac{\overrightarrow{AC}}{\overline{AC}^2} = \frac{\overline{AA'}^2 + \overline{AC}^2 - \overline{A'C}^2}{2\ \overline{AC}^2}\ \overrightarrow{AC} \tag{6}$$

$$\begin{pmatrix}\text{Shift Component}\\ \text{along } \overrightarrow{AD}\end{pmatrix} = \frac{\beta_D\overrightarrow{AD}}{\overline{AD}} = \overrightarrow{AA'}\cdot\overrightarrow{AD}\ \frac{\overrightarrow{AD}}{\overline{AD}^2} = \frac{\overline{AA'}^2 + \overline{AD}^2 - \overline{A'D}^2}{2\ \overline{AD}^2}\ \overrightarrow{AD} \tag{7}$$

$$\begin{pmatrix}\text{Shift Component}\\ \text{along } \overrightarrow{AH}\end{pmatrix} = \frac{\beta_H\overrightarrow{AH}}{\overline{AH}} = \overrightarrow{AA'}\cdot\overrightarrow{AH}\ \frac{\overrightarrow{AH}}{\overline{AH}^2} = \frac{\overline{AA'}^2 + \overline{AH}^2 - \overline{A'H}^2}{2\ \overline{AH}^2}\ \overrightarrow{AH} \tag{8}$$

where I have dropped the + subscripts to avoid clutter.

This definition extends to the other types of vertices with minor adjustments: for E, F, G and H–type vertices I choose a tetrahedron on the $(N+1)^{th}$ surface (instead of the N^{th}) as a basis for defining lapse and shift.

4. Evolving an A–type Vertex

To evolve the 3-geometry from the N^{th} spacelike hypersurface to the $(N+1)^{\text{th}}$ spacelike hypersurface, I must first evolve forward the A-type vertices, then the B-type vertices, . . ., and finally the H-type vertices. However, for my purposes, to demonstrate the lapse and shift freedom in Regge calculus, I will evolve here a single A-type vertex, say $A(2i, 2j, 2k)$.

Vertex $A(2i, 2j, 2k)$ in the N^{th} spacelike hypersurface is the meeting point of twenty four spacelike tetrahedra. I evolve each of these off the spacelike hypersurface. This forms a teepee–like structure above vertex $A(2i, 2j, 2k)$ as mentioned in Sec. 2. This teepee consists of twenty four timelike 4–simplices. The base of each of these simplices is one of the twenty four spacelike tetrahedra sharing $A(2i, 2j, 2k)$. These twenty four simplices all share the timelike "vertical" edge $\overline{AA'}$ (the edge connecting vertex $A(2i, 2j, 2k)$ on the N^{th} spacelike hypersurface to vertex $A'(2i, 2j, 2k)$ on the $(N+1)^{\text{th}}$ spacelike hypersurface). In addition, there are fourteen diagonal edges reaching from $A'(2i, 2j, 2k)$ to the N^{th} spacelike hypersurface: $\overline{A'B}_+$, $\overline{A'B}_-$, $\overline{A'C}_+$, $\overline{A'C}_-$, $\ldots$, $\overline{A'H}_+$, and $\overline{A'H}_-$; where $B_+ = B(2i, 2j+1, 2k)$, $B_- = B(2i, 2j, 2k-1)$, $C_+ = C(2i, 2j+1, 2k)$, $C_- = C(2i, 2j-1, 2k)$, $\ldots$, $H_+ = H(2i+1, 2j+1, 2k+1)$ and $H_- = H(2i-1, 2j-1, 2k-1)$.

I assume I know all edgelengths up to and including the N^{th} spacelike hypersurface. There are then 15 unknowns. One unknown is the "vertical" edge, and the other fourteen are the "diagonal" edges. I solve for these edgelengths using the Regge equations, which are the skeleton version of the Einstein field equations. There is one Regge equation for each internal edge in the lattice. Each equation is obtained by varying the simplicial analogue of the Hilbert action (the Regge action) with respect to infinitesimal variations in the length of the associated edge: $L \longrightarrow L + \delta L$.

$$\delta(I) = \delta\left(\sum_{h_{|L}} A_h \epsilon_h\right) = 0 \qquad \Longrightarrow \qquad \sum_{h_{|L}} \cot(\theta_{h,L})\epsilon_h = 0. \tag{9}$$

Here, the sum is over all triangle hinges h sharing edge L, A_h is the area of triangle hinge h, $\theta_{h,L}$ is the internal angle of triangle h opposite edge L, and ϵ_h is the deficit angle associated to hinge h (it is equal to the curvature concentrated at triangle h times the area dual to hinge h, or equivalently it is the change in angle a vector would undergo when parallel transported around triangle h). I refer to (Eq. 9) as the Regge equation associated to edge L. There are then fifteen Regge equations that depend on my fifteen unknowns. These equations are associated to the following egdes: $\overline{AB}_+$, $\overline{AB}_-$, $\overline{AC}_+$, $\overline{AC}_-$, $\ldots$, $\overline{AH}_+$, $\overline{AH}_-$, and $\overline{AA'}$.

There are fifteen equations for fifteen unknowns. This is the skeleton analogue of the continuum where we have ten equations for ten unknowns. Not all of the equations are independent. Four of the ten equations in the continuum are constraint equations — there are four contracted Bianchi identities per point in the continuum. Similarily, Miller has shown that there are four simplicial contracted Bianchi identities per vertex in Regge calculus ([21], see Sec. 5 also). Therefore, there are effectively only (15-4=11) eleven evolution equations at vertex A. Just as in the continuum one is free to fix four of the ten metric components by imposing one lapse and three shift conditions, so too in Regge Calculus one is free to impose four lapse and shift conditions on the fifteen unknowns; effectively reducing the unknowns to eleven. This is fortunate: I now have eleven Regge equations, and four lapse and shift conditions at vertex A to solve for the fifteen unknowns. The other four Regge equations are expected to be automaticly satisfied up to the resolution of the lattice (this is demonstrated explicitly in Sec. 9).

Which of the 15 equations associated to vertex A are the constraint equations? Can I pick freely any 11 of the 15 equations to evolve A? What freedom do I have in fixing the lapse and shift? May I choose any four conditions to impose on the 15 unknown edges? Is there an optimal choice of equations and lapse and shift conditions? These are the questions I address in the remainder of this section and in the next sections.

We know that in the continuum there are 6 components of the Einstein tensor which are second order in time (G_{ij}). The other four ($G_{0\mu}$) are usually treated as constraints, but one can use some or all of the $G_{0\mu}$ in the place of some G_{ij} to evolve the metric, thus obtaining constrained or partially constrained evolution.

There is no analogous "natural" splitting of the Regge equations into dynamical and constraint equations. In Sec. 8 I make several choices of constraints and of lapse and shift conditions, I analyze the convergence rates for these choices, and the degree to which the other four "constraint" equations are satisfied. I evolve an A–type vertex for various values of lapse and shift, in flat spacetime and in the Kasner universe.

5. Bianchi Identities and Constraint Equations

In continuum General Relativity the contracted Bianchi identities

$$\nabla \cdot G = 0$$

tell us that there are four relations between the ten components of the Einstein tensor. This allows us to treat four of the Einstein equations (typically $G_{0\mu}$) as constraints. Regge showed that there is a skeleton analogue of the ordinary Bianchi identity [9,23] which is of topological origin, but stated no corresponding contracted Bianchi identity.

Miller [21] developed the theoretical structure of the Regge equations using the Élie Cartan moment–of–rotation construction†. He showed that the $\partial \circ \partial \equiv 0$ principle, when applied in its (2–3–4)–dimensional form to the simplicial lattice, gives rise to a contracted Bianchi identity at each vertex. Therefore not all the Regge equations are independent. He concluded that four of the Regge equations per vertex are constraint equations.

Because finite rotations do not commute, he predicted that the contracted Bianchi identities in Regge Calculus should only be approximate, and gave an estimate that the approximation should be (using the dimensionless form of the Regge Equations (Eqn. 9)) third order in the lattice spacing. This is verified in Sec. 9.

Kheyfets, Miller and Wheeler [26] then tested this hypothesis in a Null-Strut Calculus model of the Kasner cosmology. This model had four equations for three unknowns, and they found that if three of the equations were solved, the fourth (constraint) equation was automatically solved. Here I use my new software to verify the degree to which the constraints are satisfied, and I do so with arbitrary choice of lapse and shift, and with several different lattice spacings.

† Similar work was done independently by Barrett [22]; some aspects of diffeomorphism freedom in Regge Calculus were also studied by Roček and Williams [23], Hartle [24], and Piran and Strominger [25].

The relationship between the ordinary and contracted Bianchi identities and the constraint equations in Regge Calculus has not yet been exhaustively studied, and could do with further investigation.

6. Numerical Formalism: A Glimpse

Over the last year I have developed a rather general formalism and software package to formulate and solve the Regge equations in 3+1 dimensions. One important design specification was that the software should set the equations up on its own once the lattice and lapse and shift are specified. (This has not been done before: all numerical solutions in Regge Calculus so far have been specific, have made several assumptions, and most of all have not solved for all edgelengths in the lattice.[‡]) This has allowed me to quickly experiment with various ideas I have had for different lattices, different gauge choices, and different initial values for edgelengths. My software also prepares the Regge equations for different topologies, such as $S^3 \times R$, although I still have to handle the impurities introduced in the S^3 topology. To formulate the equations on its own, my software takes a list of simplices as a primitive description of the lattice. The simplist is analyzed and from it I form tables containing information on point–to–point connectivity, which simplices hinge about given triangles, and other information about the lattice structure.

In Regge Calculus each edge can yield an algebraic equation (Eq. 9), but some edges don't have proper entourages, and others are treated as constraint equations. So one other input my software uses is a mapping of edge indices into dynamical equations. Similarly not all edges are unknowns: I specify four edges per point as lapse and shift, and other edges are determined in the previous thin sandwich. So the last input I give is a mapping of edge indices to unknowns. Once the three inputs are given, the user can start the initial value problem and the time evolution. For the purposes of this paper I will describe what happens in a time evolution step, and in particular in the time evolution of the A–type vertices.

As mentioned in Sec. 4, I have 15 equations from which I choose 4 constraints. Several choices are possible, and turn out to be roughly equivalent (see Sec. 8). I are left with 11 dynamical equations, and I add to these four equations that specify lapse and shift (equations 5-8). Given these fifteen algebraic equations, the software calculates the Jacobian matrix of their partial derivatives:

$$J_{ij} = \frac{\partial E_i}{\partial u_j}$$

where E_i $(i \in [0,14])$ are Regge equations and lapse/shift conditions, and u_j $(j \in [0,14])$ are the unknown edges. Once the Jacobian is calculated, I can do an iteration of Newton's method by solving $\sum_j J_{ij} du_j = -E_i$, where du_j are the corrections to u_j. I iterate until the 15 equations have converged satisfactorily. The convergence criterion is $\sum |E_i|/n < \epsilon$ $(n = 15)$; the graphs and tables in this paper are produced using $\epsilon = 10^{-10}$.

[‡] In a recent visit to Los Alamos, Leo Brewin has shown me a numerical scheme he is developing which is remarkably similar to mine, and does indeed solve for all the edges without imposing symmetry.

7. Numerical Runs: Organization

Here I present the results of my runs. I have made a lot of numerical experiments so the presentation of results would be confusing without an organizing principle.

In presenting my results I want to show:

- that several different choice can be made of which four equations to discard (i.e. to consider as constraints); and
- how convergence behaves depending on what choice is made

Once I have determined which four equations serve well as constraints, I want to show:

- that the four constraint equations are satisfied: i.e. that they converge together with the dynamical equations, at least up to a certain point which depends on the curvature and lattice spacing;
- that one can choose lapse and shift arbitrarily, and all the edges are then completely determined by the dynamical equations; and
- how convergence behaves as a function of the choice of lapse and shift.

Following this guideline, I present my results. First I show that I can treat various sets of four equations as constraints (Sec. 8), then that the constraint equations are satisfied once the dynamical equations have converged (Sec. 9), and finally that I can choose arbitrary lapse and shift (Sec. 10).

8. Numerical Runs: Choosing Constraints

I have made three quite different choices of constraints (and hence of dynamical equations), in my attempt to show that several choices of constraints are possible. Some choices might seem “natural”, in the sense that the constraints are chosen to come from edges with a common characteristic. For example, in the second choice the constraints are chosen to be the three straight spacelike edges in the isosceles QPL (AD, AF and AG) together with the straight timelike edge AA'. The edge AA' is treated as a constraint in both the first and second choices, but in the third choice the (only) timelike edge in the A–type teepee is treated as a dynamical equation, instead of a constraint. I find that it makes almost no difference which edges are chosen as constraints: convergence is negligibly faster for the “natural” choice, and the constraints converge only slightly more.

The runs presented here were made with approximately 10% random perturbations of the fifteen edges from flat space and the Kasner model. The edges, varied initially, converge back to the correct solution quickly. The fact that different random perturbations converge to the same result confirms my thesis that the fixing of lapse and shift fixes the gauge completely.

I conclude that I can “swap out” any four equations at will. In the remainder of this paper I will use the somewhat “natural” choice of constraints: AD, AF, AG, AA'.

9. Numerical Runs: Convergence of Constraints

All runs in this Sec. were made with a constant lapse and zero shift. I set the lattice spacing parameter $\Delta l = 0.1$ and the lapse $\Delta t = 0.01$. I write the Regge equations and constraints (equation 9) in a dimensionless form, so when I plot $\sum E_i^2$ and $\sum C_i^2$, these are always pure numbers.

Constraint choice	Model	# of iter.	$\sum E_i^2$	$\sum C_i^2$
AD,AF,AG,AA'	flat space	5	1.15×10^{-24}	1.77×10^{-19}
	Kasner	6	1.43×10^{-19}	2.33×10^{-8}
AB,AC,AE,AA'	flat space	4	1.15×10^{-24}	4.09×10^{-24}
	Kasner	7	1.33×10^{-19}	8.07×10^{-7}
AB,AC,AE,AH	flat space	4	1.19×10^{-25}	1.39×10^{-24}
	Kasner	7	1.62×10^{-21}	8.07×10^{-7}

Table 1: Convergence of dynamical equations and constraints for various choices of constraints. One sees that the choice of constraints is arbitrary, since one obtains good convergence rates and accuracy of constraints for all three choices.

- Flat space

This numerical run re-establishes the correctness of my solution of the Regge equations for this decoupling lattice, and shows that fixing lapse and shift uniquely determines a solution with no other freely specifiable edges.

In this first run I set constant lapse and zero shift for a single A–type vertex, then I randomly perturb the fifteen edges by about 10% from the flat-space values. The result is a rapid convergence of the fifteen unknowns back to flat space values in just four iterations of Newton's method (see Fig. 4). The constraints also converge to zero, showing that in Regge Calculus the constraints are solved exactly in flat space once the dynamical equations converge.

- Kasner universe

In the second run I model an axisymmetric Kasner universe. I make an analytic fit to the Kasner metric

$$ds^2 = -dt^2 + t^{2p_x}dx^2 + t^{2p_y}dy^2 + t^{2p_z}dz^2$$

with $p_x = 2/3$, $p_y = -1/3$ and $p_z = 2/3$. Then I randomly perturb the fifteen unknown edges by about 10%, and evolve the A–type teepee for a time step. The result is a rapid convergence of the dynamical equations. The constraint equations converge only up to a certain point, showing that constraints are satisfied only approximately for curved spacetimes.

- Refining the lattice spacing

In this run (a series of runs, actually) I examine how the accuracy in the constraints improves when the lattice is refined. I carried out analogous simulations to the previous run (Kasner model) for various lattice spacings. The results are presented in Table 2 and Fig. 5, which show that refinement of the time interval does not improve convergence of the constraints in the Kasner model, whereas refinement of the spacelike lattice parameter Δl improves it dramatically.

The data in Fig. 5 is closely fit by a function

$$\sqrt{\sum C_i^2(\Delta l)} = \sqrt{\sum C_i^2(\Delta l_0 = 0.1)} \times \left(\frac{\Delta l}{\Delta l_0}\right)^3$$

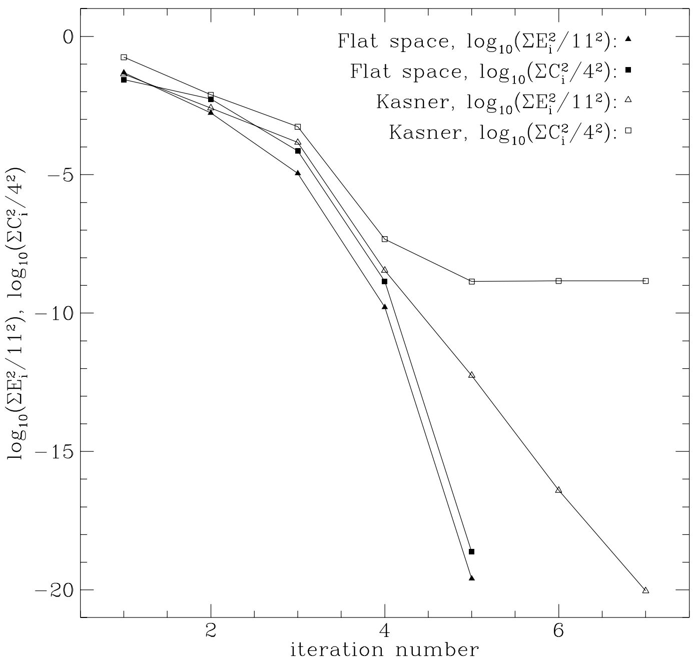

Fig. 4: A plot of the sum of the Regge equations $\sum E_i^2/11^2$ and the constraints $\sum C_i^2/4^2$ versus iteration number in Newton's method. The constraints converge up to machine accuracy in flat space, but only approximately for the Kasner model.

$(\Delta l, \Delta t)$	$\sum C_i^2$	$\sqrt{\sum C_i^2}$ratio
(0.1, 0.07)	3.30×10^{-10}	1.04
(0.1, 0.05)	3.40×10^{-10}	1.03
(0.1, 0.04)	3.45×10^{-10}	1.02
(0.1, 0.03)	3.49×10^{-10}	1.01
(0.1, 0.02)	3.54×10^{-10}	1.01
(0.1, 0.01)	3.59×10^{-10}	1.00
(0.1, 0.005)	3.61×10^{-10}	1.00
(0.3, 0.01)	2.57×10^{-7}	26.7
(0.2, 0.01)	2.28×10^{-8}	7.97
(0.1, 0.01)	3.59×10^{-10}	1.00
(0.08, 0.01)	9.48×10^{-11}	1/1.95
(0.05, 0.01)	5.62×10^{-12}	1/7.99
(0.02, 0.01)	2.30×10^{-14}	1/125

Table 2: Constraint dependence on lattice spacing for Kasner model. The accuracy of the constraints depends on the space parameter, but not on the time step.

which is also plotted in the figure. Wc conclude that the constraints are satisfied up to third order in the lattice spacing.

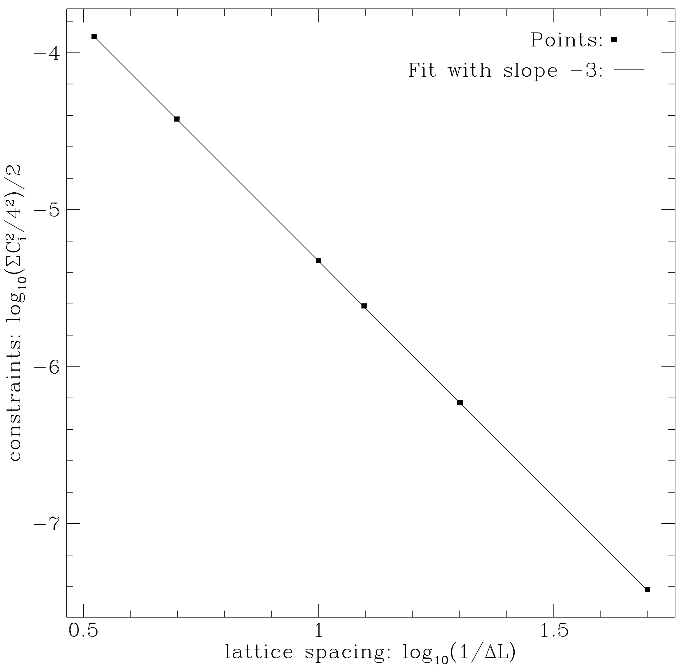

Fig. 5: Deviation in the constraints as a function of lattice spacing. Notice how the slope of the log *plot is -3, confirming that the constraints are satisfied to third order in the lattice spacing.*

10. Numerical Runs: Choosing Lapse and Shift

One of the principal results presented in this paper is that there is a diffeomorphism freedom in Regge Calculus; hence one can choose lapse and shift freely, and the solution to the Regge equations will be unique for that choice. However, in a numerical scheme the "freedom" to specify lapse and shift is limited by the Courant condition, by the physical timescales of the problem, and by the nature of the numerical methods used.

In an *explicit* time evolution scheme, the fields at a point p on the $(N+1)^{\text{th}}$ spacelike surface Σ_{N+1} are calculated from the values in a region R (the differencing domain) of the previous surface Σ_N (and possibly previous surfaces as well). If the past light cone of point p intersects a region of Σ_N contained in R, then we have a *Courant stable* scheme. Otherwise we have a *Courant unstable* scheme, and the computed solution can diverge exponentially from the correct solution. See Ref. [27] and the references therein for a discussion of the Courant instability.

The Courant condition limits my possible choices of lapse and shift: if the lapse is too large, the past light cone of a point p on Σ_{N+1} will be too large; and if the shift is too great then the past light cone of p will be shifted over and might go outside of the region R. Nevertheless, the application of the Courant condition to Regge Calculus, and to my lattice in particular, is not as clear as it is in finite differencing. First of all, the information in Regge Calculus does not consist of fields defined on the vertices of the lattice; rather it resides in the edgelengths, some of which are on Σ_N, some on Σ_{N+1}, and some in between.

Then Regge Calculus is an implicit scheme, but my decoupling lattice has a small domain of dependency because of the partial decoupling of the Regge equations. I will carry out a more complete analysis of Courant instabilities for this type of lattice in another paper, but it is important to note that lapse and shift are limited in principle.

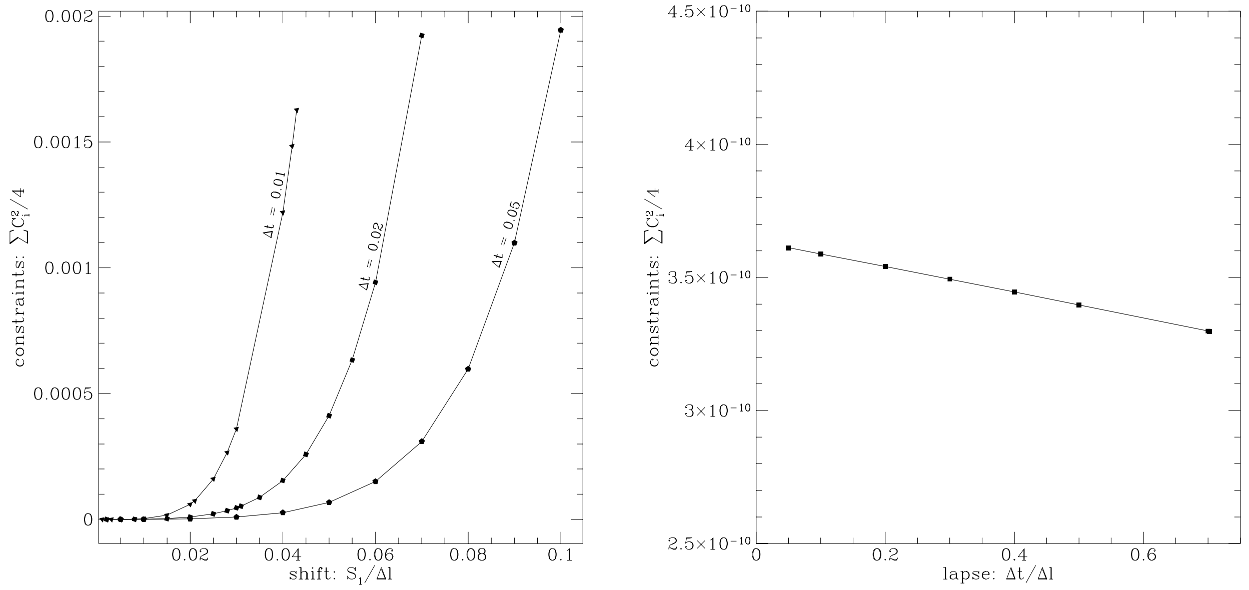

Fig. 6a and 6b: Behaviour of the solution for various shifts and lapses; Δl is 0.1. Increasing shift causes a drifting of the constraints, which happens to different degrees according to the lapse (6a). The dependence on lapse is instead very slight when shift is zero (6b).

My choice of lapse is also limited by physical considerations: if a problem has a natural time scale, then I want my lapse to be smaller than that scale or I will miss some of the action in the time evolution.

My numerical scheme introduces another limitation. I solve the Regge equations using Newton's method in fifteen dimensions. This method converges quite rapidly, but depends on an initial "guess". These numerical problems, unlike the Courant condition, can be solved when they arise in individual applications.

Here I test my software for various values of lapse and shift, to show that for each choice I get unique solutions to the Regge equations.

- Changing the shift

Here is a set of runs to show that a range of shifts can be chosen for a given choice of lapse. These runs are all based upon a flat spacetime lattice with a lattice size parameter $\Delta l = 0.1$ and lapses of 0.01, 0.02 and 0.05.

With each of these lapses, I vary the shift in the '1' direction S_1, and see how the constraints behave. The results are shown in Fig. 6a, and show that convergence of the constraint equations gets worse as the shift to lapse ratio $S_1/\Delta t$ is increased. This confirms that I can choose the shift and the Regge equations will still converge, but it also shows that the computed solution will drift farther from the analytic solution as the shift increases.

- Changing the lapse

Here I run a Kasner simulation with zero shift and different values of lapse, to examine how the choice of lapse influences convergence rates. Fig. 6b shows the behaviour of the constraints with various settings for the lapse Δt: the lapse does *not* influence the degree of convergence appreciably. The values of lapse in Fig. 6b go up to $\approx 0.7 \cdot \Delta l$, which puts the teepee near the light cone, and hence near the limit of Courant instabilities. My software does not currently allow one to pass the light cone.

11. Future Applications

The basic 3+1 dimensional software to solve the Regge equations on a $T^3 \times R$ lattice was completed in June 1992. I am now beginning to apply my tools to interesting and observationally relevant astrophysical and cosmological phenomena. Two cases are most interesting.

- *Inhomogeneous cosmologies*, where fully 3+1 dimensional codes are absolutely necessary. The major issues to be addressed for inhomogeneous cosmologies are:
  - Finding equation maps and unknown maps in the $S^3 \times R$ lattice. I are working on an intelligent algorithm to form the equation maps and unknown maps for arbitrary lattices, with no assumptions of regularity. This is necessary because some manifolds, when modeled with the QPL will have impurities: some vertices with a different number of edges emanating from them. They have to be handled separately when forming the equation map and unknown map. The result (presented above) that one can choose any of the edges at a point as dynamical equations is the first step on the way to solving this problem.
  - Introducing matter into the Regge equations. This has been done in a specific case [5], but I need to develop a formalism to incorporate the right hand side of Einstein's equations ($T_{\mu\nu}$) into the Regge equations.
  - Developing a $Tr(K) =$ constant slicing of the manifold.
  - Developing fully four dimensional visualization of the results.
- *Two black hole collisions*, where, once again, fully general 3+1 dimensional codes are required. The difficulties involved here are:
  - Finding a good way of expressing the boundary conditions for two black hole collisions (this alone is a very difficult problem which has made finite difference approaches to this problem difficult).
  - Developing a useful slicing of the manifold which makes the lattice finer near the black holes and more coarse in the background.

I will first tackle the problem of inhomogeneous cosmologies: most of the difficulties here are close to being solved (though not necessarily implemented).

Acknowledgements

I would like to thank Arkady Kheyfets, Leo Brewin, Martin Roček and Pablo Laguna who have worked through many of the details in this paper with me.

I would also like to thank the Free Software Foundation for developing the GNU C++ compiler and many other free software utilities which were used to develop the $3+1$ Regge Calculus software. I gratefully acknowledge the AFOSR which supported this work.

Most of all I would like to thank Warner Miller who developed the eight–stage evolution procedure with me and wrote down the expressions for lapse and shift.

Appendix

Here I show that the definition of lapse and shift using a weighted average of normal vectors at a vertex introduces a second order correction to the first order expressions (2–3) derived in Sec. 3. I conclude that I may avoid the added complexity of the weighted–average definition in favor of the single–tile definition of lapse and shift given in Sec. 2. This result extend, without loss of generalization, to 3+1 dimensions.

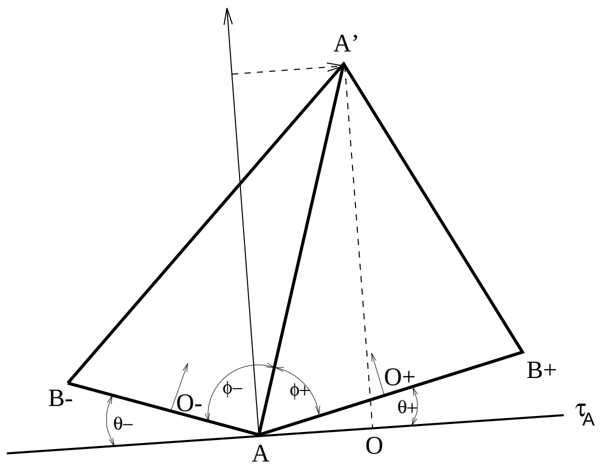

Fig. A1: An A–type vertex in 1+1 dimensions; O_- and O_+ are midpoints of AB_- and AB_+ respectively; τ_A is parallel to O_-O_+.

I define the tangent space ($\mathcal{T}_A$) at vertex A (c.f. Fig. A1) as a weighted average of the two tangent spaces defined by segments $\overline{AB}_+$ and $\overline{AB}_-$. In particular, $\mathcal{T}_A$ is the line at A defined as a difference vector.

$$\overrightarrow{O_-O}_+ = \overrightarrow{AO}_+ - \overrightarrow{AO}_- = \frac{1}{2}\overrightarrow{AB}_+ - \frac{1}{2}\overrightarrow{AB}_-. \qquad (A1)$$

Therefore, the two angles in Fig. A1 are determined by (A1). In particular,

$$\frac{\overline{AB}_+}{2}\sin\theta_+ = \frac{\overline{AB}_-}{2}\sin\theta_-, \qquad (A2)$$

and

$$\pi = \theta_+ + \theta_- + \phi_+ + \phi_-. \qquad (A3)$$

The lapse and shift from A to A' can now be defined using this weighted average tangent space.

$$\alpha = \begin{pmatrix}\text{Lapse from}\\ A \text{ to } A'\end{pmatrix} = \overline{OA'} = \overline{AA'}\sin(\phi_+ + \theta_+). \qquad (A4)$$

$$\beta \;=\; \begin{pmatrix}\text{Shift component}\\ \text{from A to A}'\end{pmatrix} \;=\; \overline{AO} \;=\; \overline{AA'}\cos\left(\phi_+ + \theta_+\right). \tag{A5}$$

I are now in the position to compare (A4–A5) to (2–3), respectively.

The extrinsic curvature at A is a function of, among other quantities, the angles θ_+ and θ_- [29,30]:

$$K_A \;=\; \frac{-4\sinh(\pi - \phi_+ - \phi_-)}{\overline{AB}_- + \overline{AB}_+} \;=\; \frac{-4\sinh(\theta_+ + \theta_-)}{\overline{AB}_- + \overline{AB}_+}. \tag{A6}$$

Since the extrinsic curvature is bounded, and the edge lengths are small ($\overline{AB}_-$ and $\overline{AB}_+ \sim O(\Delta l)$ where Δl is the characteristic lattice spacing), then the angles θ_+ and θ_- must be small:

$$\theta_\pm \;\sim\; \frac{1}{2}K_A\Delta l \;=\; O(\Delta l). \tag{A7}$$

These two conditions, and consequence, are necessary conditions for an accurate triangulation. The requirement that the angles be small is directly related to the degree of conservation of energy–momentum in Regge Calculus. Using this result, I may Taylor expand (A4–A5) in powers of θ_+. In particular,

$$\alpha \;=\; \underbrace{\overline{AA'}\sin\phi_+}_{O(\Delta l)} + \underbrace{\overline{AA'}\theta_+\cos\phi_+}_{O(\Delta l^2)} + O(\Delta l^3), \tag{A8}$$

and

$$\beta \;=\; \underbrace{\overline{AA'}\cos\phi_+}_{O(\Delta l)} - \underbrace{\overline{AA'}\theta_+\sin\phi_+}_{O(\Delta l^2)} + O(\Delta l^3). \tag{A8}$$

To first order in Δl, the definition of lapse and shift derived here agree with the definitions in Sec. 3. The corrections to (2 and 3) of Sec. 3 are of second order. Therefore, I may use the simpler definition as long as the spacetime lattice geometry is sufficiently fine. This result is more important in 3+1 dimensions as the expressions for the weighted average lapse and shift are substantially more involved than the single–tetrahedron definition forwarded in Sec. 3.

Bibliography

[1] K. S. Thorne (1987), "Gravitational radiation", in *300 Years of Gravitation*, eds. S. Hawking and W. Israel (Cambridge University Press) pp. 330–458.

[2] K. S. Thorne (1990), "Sources of Gravitational Waves and Prospects for their Detection", Caltech preprint GRP–234.

[3] M. R. Dubal (1992), "Construction of three–dimensional black–hole initial data via multiquadrics", Phys. Rev. D. **45** 1178–1187.

[4] G. B. Cook (1990), "Initial Data for the Two–Body Problem of General Relativity", Ph. D. dissertation, University of North Carolina at Chapel Hill.

[5] T. Nakamura, Y. Kojima and R. Ohara (1984), "A method of determining apparent horizons in three–dimensional numerical relativity", Phys. Lett. **106A** 235–238.

[6] P. Anninos, J. Centrella and R. Matzner (1989), "Nonlinear solutions for initial data in the vacuum Einstein equations in plane symmetry", Phys. Rev. D **39** 2155-2171.

[7] P. Anninos, J. Centrella and R. Matzner (1989), "Numerical methods for solving the planar vacuum Einstein equations", Phys. Rev. D **43** 1808-1824.

[8] P. Anninos, J. Centrella and R. Matzner (1989), "Nonlinear wave solutions to the planar vacuum Einstein equations", Phys. Rev. D **43** 1825-1838.

[9] T. Regge (1961), "General relativity without coordinates", Nuovo Cimento **19**, 558–571.

[10] C. Misner, K. S. Thorne and J. A. Wheeler (1974) in *Gravitation*, chapter 42, (Freeman).

[11] J. A. Wheeler (1963, "Regge Calculus and Schwarzschild geometry", in *Relativity, Groups and Topology* ed. B. DeWitt and C. DeWitt (Gordon and Breach) pp 463–501.

[12] R. M. Williams and P. A. Tuckey (1992), "Regge calculus: a brief review and bibliography", Class. Quantum Grav. **9**, 1409–1422.

[13] W. A. Miller (1986), "Geometric computation: null–strut geometrodynamics and the inchworm algorithm", in *Dynamical Spacetimes and Numerical Relativity*, ed. J. Centrella (Cambridge University Press) pp. 256–303.

[14] E. B. Becker, G. F. Carey and J. T. Oden (1981), *Finite Elements: An Introduction, Vol. I* (Prentice–Hall).

[15] M. Galassi (1993), "Initial Value Data in 3+1 Dimensional Regge Calculus", in preparation.

[16] M. Galassi (1993), "Application of 3+1 Dimensional Regge Calculus to the Kasner Universe", to be submitted to Phys. Rev. D.

[17] J. Barrett, M. Galassi, W. A. Miller, R. D. Sorkin, P. A. Tuckey and R. M. Williams (1992), "A Partially Decoupled Time Evolution Scheme in Regge Calculus", to be submitted to Phys. Rev. D.

[18] W. A. Miller (1986), "Foundations of Null–Strut Geometrodynamics", Ph. D. dissertation, University of Texas at Austin.

[19] R. D. Sorkin (1974), "Development of simplectic methods of the metrical and electromagnetic fields", Ph. D. dissertation, California Institute of Technology.

[20] R. D. Sorkin (1975), "The time–evolution problem in Regge Calculus", *Phys. Rev. D*, **12** 385-396.

[21] W. A. Miller (1986), "The Geometrodynamic Content of the Regge Equations as Illuminated by the Boundary of a Boundary Principle", Foundations of Physics **16** 143–169.

[22] J. W. Barrett (1986), "The Einstein Tensor in Regge's Discrete Gravity Theory", *Class. Quantum Grav.*, **3** 203–206.

[23] M. Roček and R. M. Williams (1982), "Introduction to Quantum Regge Calculus", in *Quantum Structure of Space and Time*, eds. M. J. Duff and C. J. Isham (Cambridge University Press) pp 105-116.

[24] J. B. Hartle (1985), "Simplicial Minisuperspace I. General Discussion", *J. Math. Phys.*, **26** 804–814.

[25] T. Piran and A. Strominger (1986), "Solutions of the Regge Equations", *Class. Quantum Grav.*, **3** 97–102.

[26] A. Kheyfets, W. A. Miller and J. A. Wheeler (1988), "Null-Strut Calculus: The First Test", Physical Review Letters, **61** 2042-2045.

[27] W. Press, B. Flannery, S. Teukolsky and W. Vettering (1988) *Numerical Recipes in C*, Sec. 17.1 (Cambridge University Press).

[28] P. A. Collins and R. M. Williams (1973), "Dynamics of the Friedmann Universe Using Regge Calculus", Phys. Rev. D **7** 965–971.

[29] J. B. Hartle and R. Sorkin (1981), "Boundary Terms in the Action for the Regge Calculus", *Gen. Rel. and Grav.*, **13** 541–549.

[30] A. Kheyfets, N. J. LaFave and W. A. Miller (1989), "A few insights into the nature of classical and quantum gravity via null–strut calculus", *Class. Quantum Grav.*, **6** 659–682.

Final Reflections.

At the end of the *Lapse and Shift* paper I state my future plans for applying the *r3+1* software. I am optimistic regarding the prospects, since a lot of the work to be done is detail work and there are no barriers that seem insurmountable. I feel that Regge Calculus is now a flexible tool for solving Einstein's theory on a computer.

The destiny of the Regge Calculus practioner according to Schulz.

Bibliography

A. V. Aho, J. E. Hopcroft and Ullman (1983), *Data Structures and Algorithms* (Addison Wesley).

P. Anninos, J. Centrella and R. Matzner (1989a), "Nonlinear solutions for initial data in the vacuum Einstein equations in plane symmetry", Phys. Rev. D **39** 2155–2171.

P. Anninos, J. Centrella and R. Matzner (1989b), "Numerical methods for solving the planar vacuum Einstein equations", Phys. Rev. D **43** 1808–1824.

P. Anninos, J. Centrella and R. Matzner (1989c), "Nonlinear wave solutions to the planar vacuum Einstein equations", Phys. Rev. D **43** 1825–1838.

J. W. Barrett (1986), "The Einstein Tensor in Regge's Discrete Gravity Theory", *Classical and Quantum Gravity.*

J. Barrett, M. Galassi, W. A. Miller, R. D. Sorkin, P. A. Tuckey and R. M. Williams (1992), "A Partially Decoupled Time Evolution Scheme in Regge Calculus", to be submitted to Phys. Rev. D.

Becker, Carey and Oden (1981), *Finite Elements: An Introduction, Vol. I* (Prentice–Hall).

D. V. Boulatov, V. A. Kazakov, I. K. Kostov and A. A. Migdal (1986), "Analytical and Numerical Study of a Model of Dynamically Triangulated Random Surfaces", *Nuclear Physics B*, **275** 641–686.

L. Brewin (1992), "Particle paths in a Schwarzschild spacetime via the Regge Calculus", to appear in *Phys. Rev. D.*

J. Cheeger, W. Müller and R. Schrader (1984), "On the Curvature of Piecewise Flat Spaces", *Commun. Math. Phys.* **92** 405–454.

P. A. Collins and R. M. Williams (1972), "Application of Regge calculus to the axially–symmetric initial–value problem in general relativity", *Phys. Rev. D* **5** 1908–1912.

P. A. Collins and R. M. Williams (1973), "Dynamics of the Friedmann Universe Using Regge Calculus", *Phys. Rev. D* **7** 965–971.

G. B. Cook (1990), "Initial Data for the Two–Body Problem of General Relativity", Ph. D. dissertation, University of North Carolina at Chapel Hill.

M. R. Dubal (1992), "Construction of three–dimensional black–hole initial data via multiquadrics", *Phys. Rev. D.* **45** 1178–1187.

C. R. Evans, L. S. Finn and D. W. Hobill editors (1989), *Frontiers in Numerical Relativity* (Cambridge University Press).

Flanders (1963), *Differential Forms* (Academic Press).

M. Galassi (1993a), "Lapse and Shift in Regge Calculus", Los Alamos and Stony Brook preprint: LA–UR–92–2611, ITP–SB–92–44, to appear in *Phys. Rev. D*, April 15 1993.

M. Galassi (1993b), "Initial Value Data in 3+1 Dimensional Regge Calculus", in preparation.

M. Galassi (1993c), "Application of 3+1 Dimensional Regge Calculus to the Kasner Universe", to be submitted to Phys. Rev. D.

D. S. Goldwirth and T. Piran (1992), "Initial Conditions for Inflation", *Physics Reports* **214** 223–292.

H. W. Hamber and R. M. Williams (1984), "Higher derivative quantum gravity on a simplicial lattice", *Nuclear Physics B* **248** 392–414.

H. W. Hamber and R. M. Williams (1986), "Simplicial quantum gravity with higher derivative terms: formalism and numerical results in four dimensions", *Nuclear Physics B* **269** 712–743.

J. B. Hartle (1985), "Simplicial Minisuperspace. I. General Discussion", *J. Math. Phys.* **26** 804–814.

J. B. Hartle (1986), "Simplicial Minisuperspace. II. Some Classical Solutions on Simple Triangulations", *J. Math. Phys.* **27** 287–295.

A. Kheyfets, W. A. Miller and J. A. Wheeler (1988), "Null–Strut Calculus: The First Test", *Physical Review Letters* **61** 2042–2045.

D. Knuth (1973), *The Art of Computer Programming* Vol. 3: Sorting and Searching, (Addison Wesley).

P. Laguna, H. Kurki–Suonio, R. A. Matzner (1991), "Inhomogeneous Inflation: the Initial Value Problem", *Phys. Rev. D* **44** 3077–3086.

S. M. Lewis (1982), "Two cosmological solutions of Regge calculus", *Phys. Rev. D* **25** 306–312.

W. A. Miller (1986a), "The Geometrodynamic Content of the Regge Equations as Illuminated by the Boundary of a Boundary Principle", Foundations of Physics **16** 143–169.

W. A. Miller (1986b), "Geometric computation: null–strut geometrodynamics and the inchworm algorithm", in *Dynamical Spacetimes and Numerical Relativity*, ed. J. Centrella (Cambridge University Press) pp. 256–303.

W. A. Miller (1986c), "Foundations of Null–Strut Geometrodynamics", Ph. D. dissertation, University of Texas at Austin.

Misner, Thorne and Wheeler (1974), in *Gravitation* (Freeman) chapter 42.

T. Nakamura, Y. Kojima and R. Ohara (1984), "A method of determining apparent horizons in three–dimensional numerical relativity", Phys. Lett. **106A** 235–238.

G. Ponzano and T. Regge (1968), "Semiclassical limit of Racah coefficients", in *Spectroscopic and Group Theoretical Methods in Physics*, eds. F. Block, S. G. Cohen, A. De–Shalit, S. Sambursky and I. Talmi (North–Holland) pp 1–58.

Press, Flannery, Teukolsky and Vettering (1988), *Numerical Recipes in C*, Sec. 17.1 (Cambridge University Press).

T. Regge (1961), "General relativity without coordinates", Nuovo Cimento **19**, 558–571.

M. Roček and R. M. Williams (1981), "Quantum Regge Calculus", *Phys. Lett.* **104B** pp 31–37.

M. Roček and R. M. Williams (1982), "Introduction to Quantum Regge Calculus", in *Quantum Structure of Space and Time*, eds. M. J. Duff and C. J. Isham (Cambridge University Press) pp 105–116.

M. Roček and R. M. Williams (1984), "The Quantization of Regge Calculus", *Z. Phys. C* **21** 31–37.

L. Smarr, A. Cadez, B. DeWitt, K. Eppley, (1976), “Collision of Two Black Holes: Theoretical Framework”, *Phys. Rev. D.* **14** 2443–2452.

L. Smarr (1977), “Gravitational Radiation from Distant Encounters and from Headon Collisions of Black Holes: The Zero Frequency Limit”, *Phys. Rev. D* **15** 2069–2077.

L. Smarr (1979), “Numerical Construction of Space–time”, *Proc. R. Soc. (London) A* **368** 15–16.

R. D. Sorkin (1974), “Development of simplectic methods of the metrical and electromagnetic fields”, Ph. D. dissertation, California Institute of Technology.

R. D. Sorkin (1975), “The time–evolution problem in Regge Calculus”, *Phys. Rev. D* **12** 385–396.

A. M. Tannenbaum, Y. Langsam, M. J Augenstein (1990), *Data Structures in C* (Prentice Hall).

K. S. Thorne (1987), “Gravitational radiation”, in *300 Years of Gravitation*, eds. S. Hawking and W. Israel (Cambridge University Press) 330–458.

K. S. Thorne (1990), “Sources of Gravitational Waves and Prospects for their Detection”, Caltech preprint GRP–234.

C. van Wyk (1988), *Data Structures and C Programs* (Addison Wesley).

J. A. Wheeler (1963), “Regge Calculus and Schwarzschild geometry”, in *Relativity, Groups and Topology* ed. B. DeWitt and C. DeWitt (Gordon and Breach) pp 463–501.

R. M. Williams (1986), “Building blocks for space and time”, *New Scientist* **110** 1512, 48–51.

R. M. Williams and P. A. Tuckey (1992), “Regge calculus: a brief review and bibliography”, *Class. Quantum Grav.* **9**, 1409–1422.

C–Y Wong (1971), “Application of Regge calculus to the Schwarzschild and Reissner–Nordstrøm geometries at the moment of time symmetry”, *J. Math. Physs* **12** 70–78.