INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

Carleton University
Ottawa, Canada K1S 5J7

1942-1992

Thesis contains black and white and/or coloured graphs/tables/photographs which when microfilmed may lose their significance. The hardcopy of the thesis is available upon request from Carleton University Library.

A Parallel Implementation of a Cellular Automata based Earthquake Model

By
Zhengping Wang

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Information & Systems Science

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario

May 1997

© Copyright
1997, Zhengping Wang

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-22145-8

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis

A Parallel Implementation of a Cellular Automata based Earthquake
Model

submitted by

Zhengping Wang

Dr. Evangelos Kranakis
(Director, School of Computer Science)

Dr. Jörg-Rüdiger Sack
(Thesis Supervisor)

Carleton University
May 1997

Abstract

In recent years cellular automata models have frequently been applied to simulate earthquake mechanisms. One of the main problems in computer simulations of earthquakes is the computation complexity. This has impeded further exploration of cellular automata applications for earthquake research. The parallel NEighborhood MOdeling (NEMO) system developed by the PARADIGM research group at Carleton University provides a new technique to address the performance issues for such spatial data applications.

In this study, a cellular automata earthquake model was implemented using NEMO. AS an important by-product, the practicality and workability of the NEMO system were tested. We demonstrate that a dramatic improvement in time has been obtained for our application.

A key component of this model implementation is a parallel 2-dimensional Fast Fourier Transform. A procedure to build a “bit reserved order” is proposed which establishes the basis for the iterative Fast Fourier Transform (FFT) algorithm.

Modeling earthquake mechanisms by cellular automata is discussed in detail in this thesis. The ways to control the mechanical properties of rock sample under testing are discussed and tested.

Finally, further work on both NEMO system and earthquake models is suggested.

Acknowledgments

I would like to thank my thesis supervisor Dr. Jörg-Rüdiger Sack for his guidance over the whole procedure of this study. His help and comments have greatly improved this thesis.

I would also like to thank the other members of the Carleton PARADIGM group, especially David Roytenberg. He introduced me to the NEMO system and Cell Driver in details and helped me to solve many implementation problems encountered. It was Prof. Anil Maheshwari and Doron Nussbaum whose suggestion to use FFT greatly improved the model developed in this study. Mark Lanthier devoted his time for free at any time and provided many technical help.

I acknowledge the financial support received from ALMERC Inc. and the National Science and Engineering Research Council (NSERC). Without their support, I would not have had the opportunity to do this work.

Special thanks go to my wife, Zhiqing Xu, for her encouragement and support.

Contents

Abstract	iii
Acknowledgments	iv
1 Introduction and Background	1
1.1 Concepts of Earthquakes	2
1.2 Fractals	3
1.3 Fractal Dynamics of Earthquakes	5
1.4 Significance of the Gutenberg-Richter Law	7
2 Cellular Automata for Earthquake Modeling	9
2.1 Fracture Mechanism	9
2.2 Cellular Automata Model	10
2.2.1 Initialization - Distribution of Strength	11
2.2.2 Propagation - Crack Growth	14
2.2.3 Evaluation	15
3 Cell Driver in NEMO	17
3.1 System Structure	17
3.2 Cell Driver	18
4 Implementation of the Model	20
4.1 Sequential Approach	20
4.1.1 1-Dimensional FFT	21

4.1.2	2-Dimensional FFT	28
4.1.3	Complexity Analysis	30
4.2	Parallel Approach	30
4.2.1	High-level Strategies	30
4.2.2	Trade-offs and Design Decisions	31
5	Results and Discussion	39
5.1	Performance: Sequential vs. Parallel	39
5.2	Fractal Dimension	43
5.3	User-defined Function for Propagation	45
6	Conclusions and Further Work	53
6.1	Conclusions	53
6.2	Suggestions	54
	Bibliography	56

List of Tables

5.1	Experimental results of program performance on a single processor and on multi-processors with different raster sizes using a traditional Fourier transform	42
5.2	Experimental results of program performance on a single processor and on multi-processors with different raster sizes using a Fast Fourier transform	43
5.3	List of ρ and the corresponding minimum number of failed neighbors n_{min}	51

List of Figures

1.1	Triadic Koch island	4
1.2	b -value evolution before and after a major earthquake	8
2.1	“Old” raster with a_{nm} and “new” raster with A_{st}	13
2.2	Different definitions of neighborhood	14
3.1	An overview of NEMO system	18
4.1	Complex plane showing eight 8th root of unity	22
4.2	Butterfly operation used in combining the DFT results of subproblem to parent problem	24
4.3	A tree showing the procedure of DFT for a input vector of 8 elements in the recursive FFT	25
4.4	Procedure for generating a bit-reversed order for a vector of N coefficients	27
4.5	Procedure of “row-column method” for 2-D FFT	29
4.6	A raster divided by 16 processors	31
4.7	Illustration of neighborhoods of cells in a parallel system	34
4.8	Illustration of main components of implementation	35
4.9	Steps of raster initialization	37
5.1	Images of rock sample with cracks after initialization	40
5.2	Images of rock sample after propagating 10 times	40
5.3	Images of rock sample after propagating 30 times	41
5.4	Images of rock sample after propagating 50 times	41

5.5	Graphic illustration of performance on a single processor and on multi-processors for different raster sizes by a traditional Fourier transform	44
5.6	Graphic illustration of performance on a single processor and on multi-processors for different raster sizes by a Fast Fourier transform	44
5.7	Patterns of initial strength distribution along cross sections of raster images	46
5.8	Initial raster images with fractal dimension $D = 2.1$	47
5.9	Initial raster images with fractal dimension $D = 2.6$	47
5.10	Initial raster image. $D = 2.4$. $\rho = 1$	48
5.11	Raster image after iterating 50 times. $\rho = 1$	48
5.12	Raster image after iterating 90 times. $\rho = 1$	49
5.13	Initial raster image. $D = 2.4$. $\rho = 5$	49
5.14	Raster image after iterating 50 times. $\rho = 5$	50
5.15	Raster image after iterating 90 times. $\rho = 5$	50
5.16	The relationship between ρ and crack growth speed	52

Chapter 1

Introduction and Background

Major earthquakes are among the most damaging natural disasters known to human society. Although an individual earthquake is very difficult to predict, systematic observations on a world wide scale result in a number of empirical laws. One of these is the Gutenberg-Richter law which provides seismologists with insight into assessing the probability of major earthquakes. Laboratory experiments on various rock samples contribute more information to the understanding of earthquake mechanism. As such laboratory experiments and field observations are costly and time consuming, computer simulations of earthquake processes provide a more flexible and economic alternative for researchers, because resetting experimental conditions and changing samples can be replaced by simply modifying parameters in a computer program. Unfortunately, the computations involved in such simulation are significantly complex and time consuming, even for a simple simulation model to execute on a single processor computer.

Parallel processing provides a solution to the problems involving complex computations. By connecting many processors into a single system, a complex computational task can sometimes be partitioned into many small, simple pieces and then assigned to each processor evenly. The individual solutions are obtained and combined into a solution to the original task. This technique may dramatically reduce the computation time and greatly benefit certain Geographic Information System operations as these often deal with a large volumes of spatial data. Based on this

motivation, the PARADIGM (PARAllel Digital ImaGe Modeling) group at Carleton University designed and implemented a parallel raster-based Geographic Information System. The system called NEMO is primarily intended as a NEighborhood MOdeling system. It contains three application drivers: Neighborhood Driver, Cell Driver, and Propagation Driver to provide users with fast engines to process time consuming neighborhood models in different categories.

In this study we used the Cell Driver (CD) to implement a cellular automata based earthquake model at the application level. The model may provide a tool to study the mechanisms of earthquakes by analyzing the ways stress accumulates in rock samples and the behavior of rock samples under a given stress.

In this chapter fractals and earthquake mechanisms are discussed. The second chapter describes the cellular automata model used for earthquake modeling. Chapter 3 gives an overview of NEMO and the Cell Driver in particular. Chapter 4 discusses the algorithm and details relating to the implementation of the model. Chapter 5 shows experimental results of this simulation model. Finally, the conclusions summarize and suggest further work for earthquake modeling as well as for the Cell Driver of the NEMO system.

In order to understand the details of cellular automata models for earthquakes, we discuss some issues related to earthquake mechanisms first.

1.1 Concepts of Earthquakes

Most earthquakes occur by sliding along pre-existing fractures. As long as the frictional force binding the crustal rocks together along an active fault is sufficiently strong, elastic strain continues to build in the adjacent rock and thus a great amount of energy is stored. Rupture begins at a point on the fault plane where the frictional force is first overcome, which causes shear displacement suddenly along the fault plane and a large quantity of energy is released - an earthquake occurs [38].

Geologists describe the mechanism that generates earthquakes along plate boundaries as *stick-slip* friction. Lithospheric plates do not glide smoothly past one another in the motion. Instead, the strong friction that exists between plates causes them to

stick fast under stress, then to slip suddenly, and again to stick, resulting in a series of disjointed motions. Each slip generates an earthquake and releases a certain quantity of energy.

1.2 Fractals

Many geological phenomena are scale invariant. The fractal distribution is the only statistical distribution that is scale invariant. Linear equations cannot describe scale invariance and fractal statistics; nonlinear equations are required.

Fractal concepts can be applied to geological problems in a variety of ways. One is the frequency-size distribution. Under a variety of circumstances, the frequency-size distributions of fragments, faults, mineral deposits, oil fields, and **earthquakes** have been shown to be fractal [27] [42].

Another class of models that yield fractal statistics involves the cellular automata concept. Bak and Tang (1989) gave a cellular automata model for seismicity.

A fractal set is defined by the following relation [30]:

$$N_i = \frac{C}{r_i^D} \quad (1.1)$$

where N_i is the number of objects with a characteristic linear dimension (size) r_i , D is the *fractal dimension*, and C is a constant of proportionality. Relationship 1.1 is applicable to deterministic fractal sets. An example is given in Figure 1.1. The triadic Koch island is constructed by placing a triangle with length $1/3$ of side in the center of each side. In the three examples illustrated, the length of each side is $r_1=1/3$, the number of such sides is $N_1=3$ for Figure 1.1(a); $r_2=1/9$, $N_2=12$ for Figure 1.1(b); and $r_3=1/27$, $N_3=48$ for Figure 1.1(c). Equation 1.1 is satisfied if $D = \ln 4 / \ln 3 = 1.26186$ for this example. The implication of a fractal dimension D between 1 and 2 is that the perimeter of the region is no longer a continuous differentiable line.

We can apply fractal concepts to statistical distributions using:

$$N = \frac{C}{r^D} \quad (1.2)$$

Figure 1.1: Triadic Koch island

where N represents the number of objects with linear dimension *greater than* r . A fractal distribution expresses the non-linear relationship between small and large objects (events).

To demonstrate the fractal concept further, we look at how it is applied to a fragment study.

If we fragment different materials, for example, quartz, coal and gravel, separately, it is easily noticed that small fragments are always much more abundant than larger ones. A more careful examination reveals that the number N of fragments of size greater than r satisfies fractal Equation 1.2 as long as we select D properly. The physical significance of Equation 1.2, then, is its capability to describe fragmentability of the material. The value of D is determined by and therefore describes the mechanical characteristics of the material in question. As a matter of fact, studies show that the value of D is 1.89 for quartz [16], 2.5 for coal [6] and 2.82 for gravel [16].

In summary, a fractal distribution is a statistical description of the relationship between small objects (events) and large objects through the quantity of fractal dimension D . The higher the value of D is, the smaller the relative proportion of large objects will be.

1.3 Fractal Dynamics of Earthquakes

The frequency-magnitude distribution of earthquakes is characterized by the *Gutenberg-Richter (1954) relation*. It says that in many circumstances the number N of earthquakes of magnitude greater than m occurring in a specified region and a period of time satisfies the empirical relation:

$$\log N = a - bm \quad (1.3)$$

where a and b are constants. This relationship has been found to be applicable both regionally and worldwide [38] [8].

This law is applicable over several orders of magnitude, which indicates that the mechanism of small earthquakes is essentially the same as the mechanism of large earthquakes, since otherwise their relative frequency cannot be expected to obey a simple law.

Aki (1981) and Turcotte (1992) showed that the Gutenberg-Richter relation is equivalent to the definition of a fractal distribution given in Equation 1.2.

The fractal distribution equation 1.2 can be rewritten in the following logarithmic format:

$$\log N = \log C - D \log r \quad (1.4)$$

We can show that the Gutenberg-Richter relation has the same format as equation 1.4.

To relate magnitude m to size r , Kanamori and Anderson (1975) show that the moment of an earthquake M is related to its magnitude by:

$$\log M = cm + d \quad (1.5)$$

where c and d are constants.

The magnitude m can, therefore, expressed as:

$$m = \frac{1}{c}(\log M - d) \quad (1.6)$$

The moment M is related to the area of the fault break A representing size r by:

$$M = \alpha A^{3/2} \quad (1.7)$$

where α is constant.

Replacing M in equation 1.6 by equation 1.7, we have:

$$m = \frac{1}{c} \log(\alpha A^{3/2}) - \frac{d}{c} \quad (1.8)$$

Now, the Gutenberg-Richter relation 1.3 can be rewritten in a "frequent-area" format:

$$\log N = a - b \left(\frac{1}{c} \log(\alpha A^{3/2}) - \frac{d}{c} \right) =$$

$$\left(a - \frac{b}{c} \log \alpha - \frac{bd}{c} \right) - \frac{3b}{2c} \log A$$

Considering the area $A = r^2$ and letting

$$\log C = a - \frac{b}{c} \log \alpha - \frac{bd}{c}$$

we have:

$$\log N = \log C - \frac{3b}{c} \log r \quad (1.9)$$

This function is equivalent to the equation 1.4. In other words, the Gutenberg-Richter relation is a fractal distribution with:

$$D = \frac{3b}{c}$$

Since c is usually around 1.5, $D = 2b$. Therefore, the fractal dimension for regional seismic activity is twice the b value.

1.4 Significance of the Gutenberg-Richter Law

Thousands of seismic stations established around the world are monitoring earthquakes daily. Given a certain period of time, say one year, thousands of earthquakes with various magnitudes can be recorded. It is easily noticed that earthquakes of small magnitude occur extremely frequently, however, most of these can not be detected by human senses directly, and big earthquakes occur rather rarely. The Gutenberg-Richter relation reveals the fact that the number N of earthquake decreases *logarithmically* when the magnitude rises.

It is a very significant discovery that the “chaotic” and “irregular” earthquake phenomena can be expressed by such a simple function. By global investigation, it is found that the b -value in Gutenberg-Richter law is approximately 1.0. It varies from 0.8 to 1.2 for the different regions [3]. But for a given region, this value is a constant during “normal” periods with a very small statistical variation.

More significantly, it was found that the b -value rises strongly for a period of time and reaches a peak. It drops suddenly immediately before a major earthquake occurs [19] (Figure 1.2). This rule verified numerous times in both field observations and lab experiments reveals the power-law distribution of moment. The common measure of this distribution is the b value of the Gutenberg-Richter relation. Nowadays, one of the routines of a seismic monitor station is to record the number of earthquakes with any magnitude and calculates the b -value. If it goes up to an abnormal high level, a warning should be issued, because it might be the eve of a major earthquake.

Figure 1.2: b -value evolution before and after a major earthquake

Chapter 2

Cellular Automata for Earthquake Modeling

2.1 Fracture Mechanism

Using laboratory experiments material science and rock engineering provide fundamental insight into the understanding of fracture mechanisms.

When a rock sample is compressed under *remote external stress*, *local stress* is evenly distributed over the rock only if the rock is homogeneous, no elastic deformation occurs and no boundary effect exists. Such an idealized situation hardly exists in reality. In real rock the *local stress intensity factor*, K , is a function which depends on the remote external stress, the mechanical properties of the rock as well as the geometry of existing cracks (for example, pores within the sample). A crack will be introduced into the rock when the local stress intensity factor exceeds the strength, a certain critical threshold, also called *fracture toughness*, K_c .

Laboratory experiments show that when a crack is introduced into a rock sample, the local stress intensity in the neighborhood of the crack is altered. As the crack grows larger, the rate of crack development declines. This is because the stress intensity factor is locally relieved, in other words, the energy is locally released as a result of crack growth. Costin (1987) gave an empirical expression to describe this **negative feedback** of crack growth:

$$K \propto \frac{d-a}{d}(\pi a)^{1/2}$$

where a is the crack length and d is the size of some notional “domain” of stress relaxation.

Two or more cracks can influence each other’s growth rate, if they are located sufficiently close spatially. When cracks approach one another, the rate of crack growth increases. Thus a runaway situation occurs, which leads to highly localized fracture, and finally to the dynamic failure of the rock sample - broken. This can be interpreted as the result of declining strength between the cracks or relative increase of local stress intensity between cracks. Rudnicki and Kanamori (1981) gave the following equation to measure such **positive feedback** quantitatively:

$$K \propto (c \tan(\pi a/2c))^{1/2}$$

where c is the distance from the centers of two adjacent cracks.

For any given point on the rock the state of deformation, then, will depend upon the results of the interplay between positive and negative feedback during fracture development.

2.2 Cellular Automata Model

Cellular automata models have been used to simulate seismic activities through fracture dynamics ([19], [26], [7], [17], and [35]). These models differ in loading the initial distribution (fractal or random), selecting dynamic variables (external stress, local stress intensity, fracture toughness, or combination of the three), defining neighborhood pattern (four neighbors or eight neighbors) and propagation function as well as evaluating and analyzing the model. In this study, the models suggested in [7] and in [17] were primarily used. The essential requirement for the model is to reproduce a power-law distribution measured by the b -value in Gutenberg-Richter law.

The model consists of three phases: 1) initialization, 2) propagation, and 3) evaluation.

Before we discuss the details, we first clarify some terms used in this thesis. A raster (or grid) of $N \times N$ pixels is represented by a two-dimensional array with $N \times N$ elements. Since the size of a *cell* in this study is defined as 1×1 (other sizes are possible), a cell is equivalent to a pixel in the following discussion.

2.2.1 Initialization - Distribution of Strength

Considering that the objective of this model is to simulate the behavior of a real fracture system, a raster is created with an initial fractal distribution of strength analogous to the statistics of strength distribution in the real world. A method to generate a fractal distribution is given in [19]. It consists of six steps:

- **Random distribution** - A grid of $N \times N$ cells is created. For each cell two integers b and c are randomly generated representing attributes.
- **Gaussian distribution** - The values of b and c are converted into the range of $(0.0, 1.0)$ and $(-\pi, \pi)$, respectively. Then, a third attribute a is computed for each cell based on the following function:

$$a = \sqrt{-2 \log b} \sin c \quad (2.1)$$

In this way the values of a are taken from a Gaussian distribution centered about zero.

- **Fourier transform** - In order to generate a fractal distribution, a cell has to become a part of an **integrated** collection - a raster. In other words, a cell should contain not only its own independently generated random value, but also information of other cells belonging to the same raster. A 2-dimensional Fourier transformation is employed to reach that goal.

As illustrated in Figure 2.1, an $N \times N$ “new” raster of complex Fourier coefficients A_{st} is computed using the following function:

$$A_{st} = \left(\frac{1}{N}\right)^2 \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} a_{nm} \exp\left[-\frac{2\pi i}{N}(sn + tm)\right] \quad (2.2)$$

where a_{nm} is the value of a cell in a Gaussian distribution with coordinates n and m in the “old” raster, A_{st} is the new value of a cell with coordinates s and t in the “new” raster. The new value A of each cell has already contained the information of all other cells.

- **Fractalization** - A *radial wave* number K_{st} is assigned to the cell of coordinates s and t in the new raster according to:

$$K_{st} = (s^2 + t^2)^{1/2} \quad (2.3)$$

A fractal dimension D is specified and the corresponding value of β representing fractal dimension of the spectral energy density is calculated by:

$$\beta = 7 - 2D$$

Then, the value A of each cell in the new raster is modified as follows:

$$A'_{st} = \frac{A_{st}}{K_{st}^{\beta/2}} \quad (2.4)$$

where s and t are the coordinates of the cell. At this point each cell of the new raster has already gotten the fractal factor corresponding to the given fractal dimension D .

- **Inverse of Fourier transform** - An inverse 2-dimensional Fourier transform is applied and a raster of $N \times N$ cells in a fractal distribution for the given fractal dimension D is then generated.
- **Strength or stress distribution** - By converting to a scale of real rock's strength and/or stress, a fractal distribution with desired surface fractal dimension has now been generated. In this study, for the reason of simplification, strength, i.e., fracture toughness, is selected as the dynamic variable.

Figure 2.1: "Old" raster with a_{nm} and "new" raster with A_{st}

Figure 2.2: Different definitions of neighborhood

2.2.2 Propagation - Crack Growth

After the initialization stage, the model enters into the propagation phase. During each iteration every cell of the raster adjusts its value of fracture toughness according to its neighbors' states.

Most cellular automata models of seismicity consider only the four nearest neighbors (4-connected model) on a square lattice. In this study, an 8-connected model was applied, allowing the state of a node to be influenced by the states of the eight neighbors as shown in Figure 2.2.

Crack growth proceeds by individual cell's failing. This takes place when the stress intensity exceeds the fracture toughness of a cell. A failed cell becomes part of a crack, and influences the local fracture toughness of its neighbors. Once a cell has failed, it stays in that state for the rest of the simulation.

The local stress intensity is modified by the presence of neighboring failed cells as follows:

- When a cell has a small, but non-zero, number of failed neighboring cells, the stress intensity on that cell is relieved, and is therefore at less than the general level of stress intensity. This simulates the negative feedback when a crack is isolated.
- When a cell has a large number of failed neighbors, the stress intensity on

that cell is greater than the general level of stress intensity. This simulates the positive feedback of crack interaction.

The relative quantity of positive and negative feedback is determined by the number of failed neighbors and is parameterized by the function $f(n)$:

$$f(n) = \begin{cases} (1 + \rho e^{-(n-4)^2/20})e^{-n^2/16} & \text{for } n = 1 \text{ to } 8 \\ 1 & \text{for } n = 0 \end{cases} \quad (2.5)$$

where n is the number of failed neighbors.

For simplicity, instead of adjusting the local stress intensity or the combination of local stress intensity and fracture toughness, the model in this study uses the fracture toughness as the only attribute of the cell and renews the value during propagation by multiplying the previous fracture toughness by $f(n)$:

$$K_{c_{new}} = f(n) * K_{c_{old}} \quad (2.6)$$

The value of $f(n)$ depends on the number of failed neighbors n of the cell in question and the parameter ρ . For each given value of ρ ranging from 0 to 8, there is a corresponding value n_{min} such that $f(n)$ is less than 1 when the number of failed neighbor n_{act} is larger than n_{min} , and $f(n)$ greater than 1 when n_{act} less than n_{min} . Through adjusting the value of n_{min} , parameter ρ determines the relative quantities of negative and positive feedback and therefore the rate of crack growth. The details is discussed in Chapter 5.

2.2.3 Evaluation

There are various ways to evaluate the results of the model. Different authors may have different interests. Henderson and *et al.* (1994) had already verified the feasibility of the cellular automata model by analysis of crack distribution through cluster analysis which show the b -value evolution in the way observed both in the fields and in experiments.

The method was that after each propagation, the number of cracks as well as the size of cracks were counted and the b -value was calculated according to the least biased estimation method suggested by [1]:

$$b = \frac{\log_{10} e}{\overline{m} - m_o} \quad (2.7)$$

where $\overline{m}$ is the average magnitude of an ensemble of earthquakes and m_o is the threshold magnitude, i.e., the magnitude below which earthquakes are not detected.

Chapter 3

Cell Driver in NEMO

With new technology such as the Earth Observation System, the volume of available spatial data increases extremely fast. Geographical Information System these days, therefore, have to deal with an already very large and still increasing volume of spatial data and, at the same time, maintain a good performance. Because of the speed limitations of single processor, many time-consuming applications in GIS are forced to simplify their models by reducing either raster sizes or the numbers of dynamic variables. As a result of such simplification, resolution and accuracy are often sacrificed. To solve these kinds of problems, the PARADIGM (PARAllel Digital ImaGe Modeling) research group at Carleton University designed and developed a parallel system for NEighborhood MOdeling (NEMO). This system is expected to be able to provide users with a fast engine for processing a variety of time-consuming tasks which apply neighborhood modeling for raster spatial data.

3.1 System Structure

The NEMO system has five components: three application drivers, i.e., *Cell Driver*, *Propagation Driver* and *Neighborhood Analysis Driver*, and two client-server drivers, i.e., *Display Manager* and *Raster Database Manager* (see Figure 3.1).

NEMO is capable of processing three types of time consuming raster neighborhood models: Cellular Automata, Propagation, and Neighborhood Analysis. Although the

Figure 3.1: An overview of NEMO system

three application drivers differ functionally, the principles under which they operate are similar. The drivers accept 1) one or more raster images, 2) a user-defined neighborhood function, and 3) a local neighborhood definition as input. And then, the application drivers activate each cell of the raster by calculating the new value for the cell based on the user-defined neighborhood function, output the results to the database and if necessary display the resulting data by Display Manager on the screen [20].

3.2 Cell Driver

Raster processing, such as neighborhood modeling, may be a time consuming task. It is particularly true for cellular automata modeling as operations are computed for many iterations. The Cell Driver (CD) in NEMO is intended for general cellular automata GIS applications. After getting the initial data, it processes a new generation by applying a user-defined neighborhood function F to the whole raster at time t . It can iterate this processing for a desired number of generations. After an interval of generation(s) defined by user, the resulting raster data can be displayed on the screen. The driver supports either Von Newman (4-neighborhood) or Moore (8-neighborhood) models by selecting the defaults, or it uses a user-defined local

neighborhood model.

For the convenience of expression, let $g_{i,j}$ be a cell in an $n \times m$ grid corresponding to the location of a point in a geographic terrain. Each cell can store one or more attributes. e.g., stress intensity and/or rock strength. Cell $g_{i,j}$ may be time variant, denoted by $g_{i,j}(t)$ for time t . The neighborhood $N(g_{i,j})$ of a cell $g_{i,j}$ is a set of cells in the raster associated with $g_{i,j}$. Typically, $N(g_{i,j})$ includes all of the cells which are within a specified distance from $g_{i,j}$, e.g., 1 in Von Neumann model and $\sqrt{2}$ in Moore model.

NEMO provides system functions to configure nodes in mesh and CD provides communication functions for exchanging neighborhood information between nodes for the Von Neumann neighborhood model and the Moore neighborhood model. The cellular automata earthquake model described in previous chapter will be executed using the Cell Driver.

Chapter 4

Implementation of the Model

As described in Chapter 2, the cellular automata model of earthquakes consists of three steps: 1) initialization (creating raster data as input), 2) crack growth (propagating based on a user-defined function), and 3) evaluation (displaying and analyzing the resulting data). Among the three steps, the first one, i.e., initialization is the critical part in this case, because the performance of the whole model depends mainly on the complexity of the algorithm and strategies employed during this stage. The propagation part deals with the details of the rock property during mechanical failure. The evaluation part strongly depends on the interests of researchers and analysis methods. In this study we, therefore, focus on the initialization stage.

4.1 Sequential Approach

In order to create a crack system in fractal distribution which minics a pre-existing fault system on a $N \times N$ grid, a 2-dimensional Fourier transformation is used:

$$A_{st} = \left(\frac{1}{N}\right)^2 \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} a_{nm} \exp\left[-\frac{2\pi i}{N}(sn + tm)\right]$$

where a_{nm} is a randomly generated value in a Gaussian distribution stored in the cell with coordinates n and m in the original raster, A_{st} is the new value of a cell with coordinates s and t in the “new” raster as shown in Figure 2.1.

To compute a Fourier transformation in a traditional way following a straightforward calculation, we need to visit each cell in the old raster exactly once in order to get a new value for one cell in the new raster. In other words, for each new value A we have to perform $N \times N$ multiplication. Therefore, for $N \times N$ cells in the new raster the total number of multiplication is in the order of N^4 .

4.1.1 1-Dimensional FFT

To reduce the number of multiplication, in this study a Fast Fourier Transformation (FFT) was used. For ease of illustration, let us look at the principle of a 1-dimensional FFT first [11]. A 1-dimensional Fourier transformation is expressed as follows:

$$A_s = \sum_{n=0}^{N-1} a_n \exp\left[-\frac{2\pi i}{N} sn\right]$$

For each cell we must perform N multiplication to get its value of A in the traditional method. For N cells the total number of multiplication is N^2 .

But notice the symmetrical characteristics of $e^{-\frac{2\pi i}{N} sn}$ with increasing n from 0 to $N-1$. A **complex N th root of unity** is a complex number ω such that

$$\omega^N = 1$$

There are exactly N complex N th roots of unity; these are $e^{\frac{2\pi i k}{N}}$ for $k = 0, 1, \dots, N-1$. Considering the definition of the exponential of a complex number:

$$e^{iu} = \cos(u) + i \sin(u)$$

the symmetrical characteristics of $e^{\frac{2\pi i k}{N}}$ can be illustrated using the complex plane shown in Figure 4.1 if we assume $N = 8$. Here $\omega_N = e^{\frac{2\pi i}{N}}$ is called the **principal N th root of unity**. All other complex N th roots of unity are powers of ω_N ; they are:

$$\omega_N^0, \omega_N^1, \dots, \omega_N^{N-1}$$

By the **Halving Lemma** which states that if $n > 0$ is even, then the squares of the n complex n th roots of unity are the $n/2$ complex $(n/2)$ th roots of unity [11], it is

Figure 4.1: Complex plane showing eight 8th root of unity

not hard to prove and verify that ω_N^k and $\omega_N^{k+\frac{N}{2}}$ have the same square. This property provides an essential base to a divide-and-conquer approach for FFT, because in this case one problem with N distinct values can be reduced to two subproblems, each of which has $N/2$ complex $(N/2)$ th roots of unity. The subproblems with $N/2$ distinct values have the exact same nature as the original problem and can be recursively reduced to two further smaller problems.

Pseudocode

The pseudocode of a recursive FFT is illustrated below [11].

```

RECURSIVE_FFT( $a$ )
1   $N \leftarrow \text{length}[a]$    $\triangleright N$  is a power of 2
2  if  $N = 1$ 
3    then return  $a$ 
4   $\omega_N \leftarrow e^{\frac{2\pi i}{N}}$ 
5   $\omega \leftarrow 1$ 
6   $a^{[0]} \leftarrow (a_0, a_2, \dots, a_{N-2})$ 
7   $a^{[1]} \leftarrow (a_1, a_3, \dots, a_{N-1})$ 
8   $y^{[0]} \leftarrow \text{RECURSIVE\_FFT}(a^{[0]})$ 
9   $y^{[1]} \leftarrow \text{RECURSIVE\_FFT}(a^{[1]})$ 
10 for  $k \leftarrow 0$  to  $N/2 - 1$ 
11   do  $y_k \leftarrow y_k^{[0]} + \omega y_k^{[1]}$ 
12      $y_{k+(N/2)} \leftarrow y_k^{[0]} - \omega y_k^{[1]}$ 
13      $\omega \leftarrow \omega \omega_N$ 
14 return  $y$ 

```

Lines 6 and 7 divide the problem into two subproblems each of which has only half the numbers of coefficients of the current problem. Lines 8 and 9 recursively call the FFT function to solve the subproblems. When the basis is reached, lines 2 and 3 return the value of Discrete Fourier Transform (DFT) of the basis. Lines 11 and 12

Figure 4.2: Butterfly operation used in combining the DFT results of subproblem to parent problem

recover the parent problem by combining the solutions of its two subproblems. Lines 4, 5 and 13 update the principal N th root of unity promptly and properly.

Notice that the value of $\omega_N^k y_k^{[1]}$ is computed twice in the recursiveFFT pseudocode. Therefore, a **butterfly operation** as shown in Figure 4.2 is introduced by adding a temporary variable t . Then, the lines from 10 to 13 can be replaced by the following pseudocode [11].

```

BUTTERFLY( )
  for  $k \leftarrow N/2 - 1$ 
    do  $t \leftarrow \omega y_k^{[1]}$ 
       $y_k \leftarrow y_k^{[0]} + t$ 
       $y_{k+(N/2)} \leftarrow y_k^{[0]} - t$ 
       $\omega \leftarrow \omega \omega_N$ 

```

We list the recursive calls in the RECURSIVEFFT() and arrange the coefficients for DFT in a tree structure such as in the example of 8-element coefficients as shown in Figure 4.3. It is easily verified that RECURSIVEFFT() computes: 1) DFT for the eight 1-element vectors on the leaves in pairs and returns four 2-element vectors to their parents; 2) DFT for the four 2-element vectors in pair to generate two 4-element vectors; and 3) DFT for the two 4-element vectors to get the final one 8-element FFT vector.

Figure 4.3: A tree showing the procedure of DFT for a input vector of 8 elements in the recursive FFT

By understanding this rule, it is now possible to write an *iterative* version of FFT, assuming we know the proper order of the coefficients stored in the leaves of the tree shown in Figure 4.3. The pseudocode of ITERATIVE_FFT is given as follows.

```

ITERATIVE_FFT( $a$ )
1  BIT_REVERSE_COPY( $a, A$ )
2   $N \leftarrow \text{length}[a]$   $\triangleright N$  is a power of 2
3  for  $s \leftarrow 1$  to  $\lg N$ 
4      do  $m \leftarrow 2^s$ 
5           $\omega_m \leftarrow e^{\frac{2\pi i}{m}}$ 
6          for  $k \leftarrow 0$  to  $N - 1$  by  $m$ 
7              do  $\omega \leftarrow 1$ 
8                  for  $j \leftarrow 0$  to  $m/2 - 1$ 
9                      do  $t \leftarrow \omega A[k + j + m/2]$ 
10                          $u \leftarrow A[k + j]$ 
11                          $A[k + j] \leftarrow u + t$ 
12                          $A[k + j + m/2] \leftarrow u - t$ 
13                          $\omega \leftarrow \omega \omega_m$ 
14  return  $A$ 

```

The key issue is to find the proper order of the coefficients. In other words, what does the function `BIT_REVERSE_COPY( )` called in the pseudocode of `ITERATIVE_FFT( )` do?

Bit-Reverse Order

By carefully examining the bit expressions of the 8 elements in their initial order, i.e., 000, 001, 010, 011, 100, 101, 110, 111 (binary format of the sequence: 0, 1, 2, 3, 4, 5, 6, 7) and the order listed at leaf level, i.e., 000, 100, 010, 110, 001, 101, 011, 111 (binary format of the order: 0, 4, 2, 6, 1, 5, 3, 7), it can be found that the second order is just a bit-reverse binary order of the first. In fact, this rule is true in general [11].

Considering that the numbers in the order of leaves will be used many times, our approach is to build a mapping table for the numbers and their bit-reversed expressions instead of computing such numbers repeatedly every time when they are needed. By tracing the rules for building the tree (such as that in Figure 4.3), we established a simple method to build a mapping table (represented by an array) to match the original order and its bit-reversed order by $O(N \lg N)$ additions.

The method is described in Figure 4.4 as the following:

First, for any given number N which is power of 2, i.e., $N = 2^k$ ($k = 0, 1, \dots$), we create a 1-dimensional array `map[ ]` with N element spaces. Now we are determining the bit-reversed order of the sequence: 0, 1, 2, 3, ..., $N - 1$ and store the order in the spaces. The first element in the bit-reversed order is still 0 and therefore we assign 0 to `map[0]`. Then, we just need to carry out the following steps:

- 1) Add the value of `map[0]` by $N/2$, i.e., $0 + N/2 = N/2$, to `map[1]`;
- 2) Add $N/2^2$ to each value of the first two elements, i.e., `map[0]` and `map[1]` and assign the new values to the next two elements, i.e., `map[2]` and `map[3]`, respectively;
- 3) Add the values in the first 4 elements by $N/2^3$ and assign to the next 4 elements, respectively.

In general, we store 0 in the first element and then by step number k (for $k =$

Let $N=16$, then the original order: 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15

Reorder:

In general, for an integer $N=2^k$ ($k = 0, 1, \dots$)

Figure 4.4: Procedure for generating a bit-reversed order for a vector of N coefficients

$1, \dots, \lg N$) we add $N/2^k$ to each value of the first 2^{k-1} element(s) and assign to the next 2^{k-1} elements, respectively.

After step $\lg N$, all elements have been filled and the array *map*[] contains the desired bit-reversed order of vector with N coefficients for FFT.

4.1.2 2-Dimensional FFT

The 1-dimensional FFT can be extended to a 2-dimensional FFT by the “row-column” method discussed in [33], [18] and [28].

We can first compute a 1-dimensional FFT for each row of the raster and store the results as intermediate data. Then we compute another 1-dimensional FFT for each column. In other words, a 2-dimensional FFT can be implemented as two 1-dimensional FFTs, one for the rows and another for the columns. The reason behind this is the “separability” property of the 2-dimensional Fourier transform. As a matter of fact, the 2-dimensional Fourier transform:

$$A_{st} = \left(\frac{1}{N}\right)^2 \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} a_{nm} \exp\left[-\frac{2\pi i}{N}(sn + tm)\right]$$

can be rewritten as:

$$A_{st} = \left(\frac{1}{N}\right)^2 \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} a_{nm} \exp\left[-\frac{2\pi i}{N}sn\right] \exp\left[-\frac{2\pi i}{N}tm\right] =$$

$$\left(\frac{1}{N}\right)^2 \sum_{n=0}^{N-1} \exp\left[-\frac{2\pi i}{N}sn\right] \left(\sum_{m=0}^{N-1} a_{nm} \exp\left[-\frac{2\pi i}{N}tm\right]\right)$$

For a cell with coordinates s and t , each inside sum $\sum_{m=0}^{N-1}$ is actually a 1-dimensional Fourier transform for a row, because only the index m keeps changing. We can compute a total of N such DFT for N rows. The outside sum $\sum_{n=0}^{N-1}$ uses the results of the row DFTs as N coefficients and therefore it corresponds to N 1-dimensional column DFTs in this case. Figure 4.5 shows the procedure graphically.

Figure 4.5: Procedure of "row-column method" for 2-D FFT

4.1.3 Complexity Analysis

In the traditional method, it takes $O(N^2)$ time to compute a 1-dimensional and $O(N^4)$ for a 2-dimensional Fourier transform.

For a 1-dimensional FFT we only take $O(\lg N)$ time to compute the value for one cell. Since there are a total of N cells, the complexity is $O(N \lg N)$ for a 1-dimensional FFT.

In the case of 2-dimensional FFT, by the “row-column” method 1-dimensional FFTs are conducted for each row in the first stage in $O(N * (N \lg N))$ time and then for each column in another $O(N * (N \lg N))$ time. Therefore, the complexity of the whole algorithm is $O(N^2 \lg N)$.

4.2 Parallel Approach

NEMO is a parallel system. To explore the advantage of this system, the simulation of earthquake models in this study was implemented in a parallel approach. The key issue here is to carry out 2-dimensional FFTs in parallel.

4.2.1 High-level Strategies

Parallelism involves (frequently) to partition a task into several subtasks, assigning them to different processors, executing them and combining the results.

Parallelism may introduce new problems, e.g., load balancing, bottleneck for communication, choice of partitioning and implementation issues. A good parallel algorithm should partition the task as evenly as possible and keep every processor (node) equally busy. Bottlenecks in processes should be avoided, otherwise a situation may arise in which only few nodes are working while others are idle and waiting.

Unlike the traditional sequential model in which all information is stored in a single node, a parallel technique often makes the participating nodes share the source and/or intermediate data. One node may need data owned or generated by other nodes in order to do its own job correctly. Thus, communication is required between nodes of the system. To minimize delay, message size and total number of messages

Figure 4.6: A raster divided by 16 processors

should be as small as possible. In addition, the path between nodes which need to exchange data to each other should be as short as possible.

4.2.2 Trade-offs and Design Decisions

NEMO provides system functions to setup topography of nodes into a mesh. Figure 4.6 gives an example in which 16 nodes are configured in a square mesh. The number in this figure represents the node identities. A global raster with 32×32 cells is therefore divided into 16 local rasters and each node in this case just needs to take care of one local raster with only 8×8 cells. In terms of the amount of computation, this partition ensures good load balancing, if the amount of work per cell is equal.

Initiation

For a 2-dimensional Fourier transformation, a node must have knowledge of the entire global raster, even though it just needs to compute values for its own local raster. In other words, a node must know the cell values of all other local rasters. The simplest way to meet this requirement is to let each node broadcast the cell values of its local raster to all other nodes. Once the information exchange is performed, each node can begin to compute 2-dimensional Fourier transform for its local raster. If the number of participating nodes is n , then the number of messages sent by each node is $n - 1$ and the total number of messages therefore is $(n - 1)n$. The size of each message is as big as a local raster.

The delay of communication sometimes can be significant when the path between two nodes is complicated, as e.g., the path between node 0 and node 15 in Figure 4.6. This becomes even more significant when the total number of nodes is big, e.g., 64 nodes.

Considering the procedure of the 2-dimensional FFT as realized by two 1-dimensional FFTs, we schedule the information exchange between the nodes in two steps, one for the rows and one for the columns. For example, node 5 in Figure 4.6 needs to know only the values of the cells in node 4, 6, and 7 to perform the 1-dimensional FFT for rows of its local raster; and later only the values of the cells in node 1, 9 and 13 for the column FFT. Thus, before nodes compute their row FFTs, they just need to exchange their local rasters with the nodes at the same *horizontal* level in the mesh. After all nodes have performed their row FFTs, they exchange local rasters with the nodes at the same *vertical* level in the mesh and perform column FFTs. In such an approach, each node will send 3 messages for row FFTs and another three for column FFTs. The total number of messages is $(3 + 3) \times n$ in this example. In general, in the case of square mesh with a total of n nodes, each node needs to send $\sqrt{n} - 1$ messages to the nodes on the same row and another $\sqrt{n} - 1$ messages to the nodes on the same column. The total number of messages for n nodes, therefore, is $2(\sqrt{n} - 1)n$.

Crack Growth

After the initialization of a global raster with stress/strength in a fractal distribution, the simulation enters the “crack growth” stage. Each cell counts the number of its neighbors which are in state “BROKEN” and then calculates a coefficient to adjust its own value according to Equations 2.5 and 2.6. When its new adjusted value exceeds the critical value, the cell changes its state to “BROKEN” and remains “BROKEN” until the end of simulation.

For a cell which is not located at the border of the local raster, e.g., “*a*” in Figure 4.7, all information about its eight neighborhoods is available since they are stored in the same node. But for those cells located on the border, e.g., “*b*” in Figure 4.7, some of its neighbors belong to other nodes. Therefore, a node must exchange information on border cells with the relevant neighboring nodes. The Cell Driver in NEMO provides default functions to perform such communication for neighborhood information exchange.

Illustrating the Components of the Implementation

The implementation of the simulation consists of two parts: one instruction set for host and one for the internal nodes.

The main components are illustrated in Figure 4.8.

Host Functions

Host functions are used to coordinate the entire procedure.

When the program is initiated, the host accepts some user-specified parameters (such as global raster size, total number of iterations, number of generations between two consecutive displays, and so on) and calculates mesh size and neighborhood size as well as sets default parameters for the raster size, neighborhood mode, and cell size.

Then, the host activates the Cell Driver.

After initialization, the host will coordinate the schedule for displaying resulting images on the screen and update the database, if necessary.

Figure 4.7: Illustration of neighborhoods of cells in a parallel system

Figure 4.8: Illustration of main components of implementation

Node Functions

This part of the implementation is the major part of this simulation.

As Figure 4.8 shows, the primary node functions are: 1) initialization of dynamic variable(s) in a fractal distribution and 2) crack growth propagation. These functions are conducted in parallel. Each node performs work only for its local raster. Whenever shared data are needed, nodes will communicate with each other.

Initialization of Local Raster - The details of the initialization function for individual node shown in Figure 4.8 are given in Figure 4.9. The latter figure illustrates the procedure every node follows during initialization stage. Each node generates random values for every cells of its local raster and converts the values to a Gaussian distribution using Equation 2.1 as described in Section 2.2.1.

On the basis of Gaussian distribution, a Fourier filter is introduced by first exchanging local rasters with nodes on the same horizontal level followed by a row FFT and then exchanging local rasters containing row FFT coefficients with nodes on the same vertical level followed by a column FFT.

After applying the Fourier transformation, every cell contains information of the global raster. Using Equations 2.3 and 2.4, each node modifies its cell values to a required fractal distribution.

An inverse Fourier transformation is analogous to a Fourier transform, except for replacing $-e^{\frac{2\pi i}{N}}$ by $e^{\frac{2\pi i}{N}}$ in Equation 2.2.

At this point, the raster with cell values in a fractal distribution is completed. The next step is to convert cell values to stress, strength, or the combination of stress and strength of real rocks by scaling. In this study all cell values are scaled by multiplying coefficient SCALE1 to the real part of the complex numbers and then adding SCALE2 to the products. Coefficients SCALE1 and SCALE2 are adjustable parameters in this case. The purpose is to allow a wider range of the fractal values and all be positive to represent the strength distribution. A threshold as critical value is selected by computing the average of the whole raster. After comparing the scaled value representing strength with the critical value (threshold), the state of a cell is either set to "SOLID", "ALMOST" (solid), or "BROKEN". At this moment,

Figure 4.9: Steps of raster initialization

the raster can be displayed on the screen with color of blue (solid area), red (close to be broken), and white (initial crack representing broken cells).

Fracture Growth - When the initialization of the global raster is completed, each node begins to count how many of its neighborhood cells are in the state “BROKEN”. Using the number of broken neighbors as input the cell will calculate the adjusting coefficient according to Equation 2.6 and multiply the resulting coefficient to its current cell value which is strength. Finally, the cell will re-evaluate its state by comparing its new strength value to the critical value.

The procedure described above is simpler for a cell which is not located on the four borders of its local raster because their all eight neighbors are stored in the memory of the same node. In case of a border cell, it is necessary for the node to get the information from its neighbor node(s). The Cell Driver provides default functions to setup neighborhood definition, neighborhood structure, and to activate neighborhood information exchange (sending and receiving).

At the application level, however, we need to write a function to recognize the neighborhood location, i.e., from which node(s) to get the neighbor information.

Chapter 5

Results and Discussion

By running our program for the cellular automata based earthquake model using the Cell Driver of NEMO, a series of images were generated as shown in Figures 5.1, 5.2, 5.3 and 5.4.

Figure 5.1 is the image representing a rock example with a limited number of initial cracks (white). The blue area represents the solid part of the rock with no internal pores or cracks. The red area indicates where the local stress is already very close to the strength. With continuing iterations, the initial cracks grow. Figure 5.2, 5.3, and 5.4 give the images after 10, 30, and 50 iterations, respectively. In Figure 5.4 a big crack becomes visible, across the entire image representing the failure of the rock - a major earthquake occurs.

A set of experiments was conducted to test our implementation. We concentrate on: 1) comparing the performance of the program when executed on a single processor versus a multiprocessor machine (parallel); 2) determining the influence of the initial fractal dimension D on the model; and 3) determining the influence of parameter ρ in the user-given function for iteration on the model.

5.1 Performance: Sequential vs. Parallel

As discussed in Chapter 4, the most time consuming part in this model is the algorithm used in the initialization phase. In most cases, it takes over 90% of the total

Figure 5.1: Images of rock sample with cracks after initialization

Figure 5.2: Images of rock sample after propagating 10 times

Figure 5.3: Images of rock sample after propagating 30 times

Figure 5.4: Images of rock sample after propagating 50 times

	Raster Size: 64 x 64		Raster Size: 128 x 128		Raster Size: 256 x 256	
	Single Proc.	16 proc.	Single Proc.	16 proc.	Single Proc.	16 proc.
First	55 sec.	11 sec.	13 min. 10 sec.	1 min. 17 sec.	213 min.04sec.	15 min. 00 sec.
Second	55 sec.	11 sec.	13 min. 25 sec.	1 min. 17 sec.	213 min.50sec.	14 min. 20 sec.
Third	55 sec.	11 sec.	13 min. 31 sec.	1 min. 14 sec.	213 min.51sec.	14 min. 30 sec.
Average	55 sec.	11 sec.	13 min. 22 sec.	1 min. 16 sec.	213 min.35sec.	14 min. 37 sec.

Table 5.1: Experimental results of program performance on a single processor and on multi-processors with different raster sizes using a traditional Fourier transform

executing time of the program. Computing the user-defined function for iteration is comparatively inexpensive: each iteration takes a few seconds, even for a 256×256 raster on a single processor.

The raster size determines the time complexity. The raster sizes selected for testing were 64×64 , 128×128 , 256×256 , 512×512 and 1024×1024 . For each selected raster size, tests were conducted using a single processor and 16 processors arranged in a square mesh, respectively. All tests were performed on the same parallel machine - Alex1.

Table 5.1 lists the performance results to initialize the raster with fractal distribution strength for different tests by a traditional Fourier algorithm.

For a raster with 64×64 cells (pixels), it took 55 seconds to initialize the raster on a single processor, while 11 seconds were taken on 16 processors. The three tests for each condition (either on a single processor or on 16 processors) show the same results. When raster size is increased by a factor of 2, i.e., 128×128 , the performance by a single processor was 13 minutes 22 seconds on average with some small variations in the seconds, while only 1 minute and 16 seconds were taken by 16 processors in average for each of the three tests. When the raster size increases to 256×256 , it took 3 hours 33 minutes and 04 seconds to initialize the raster by a single processor and only 14 minutes 37 seconds by 16 processors.

We conclude that with an increase of raster size from (64, 64), (128, 128) to (256, 256) the gain due to parallelism increases. Speed-ups increases from 5, 10 and 15

	Raster Size: 256 x 256		Raster Size: 512 x 512		Raster Size: 1024 x 1024	
	Single Proc.	16 proc.	Single Proc.	16 proc.	Single Proc.	16 proc.
Executing Time	102 sec.	29 sec.	8 min 2 sec.	56 sec.	21 min.04sec.	2 min 13 sec.

Table 5.2: Experimental results of program performance on a single processor and on multi-processors with different raster sizes using a Fast Fourier transform

times, respectively. The latter speed-up is close to the theoretical optimal value of 16.

Using a FFT algorithm, the performance is improved dramatically. Table 5.2 shows the test results for initializing rasters of varying sizes by FFT algorithm implemented in this study. It takes 102 seconds to initialize a raster of 256×256 cells by FFT, while the executing time is 213 minutes and 35 seconds using a traditional Fourier transform. However, with increasing raster size, the FFT is still very costly when executed on a single raster. When the raster size increases to 512×512 , it took 8 minutes and 2 seconds on a single processor. If the raster size of 1024×1024 is selected, it took 21 minutes and 4 seconds to initialize the raster. Such cost is significant considering the simplicity (only one initial parameter is used, i.e., strength) of this model. Using 16 processors, it took only 56 seconds and 2 minutes 13 seconds to initialize a raster of 512×512 and 1024×1024 , respectively.

The difference between sequential single processor and parallel multi-processors can be well illustrated in Figure 5.5 and Figure 5.6 which use the results in Table 5.1 and Table 5.2, respectively. These results establish our hypothesis that a limited number of parallel processors can play a significant role in earthquake modeling.

5.2 Fractal Dimension

The initial fractal dimension D is one of the parameters which can be selected by end user. It determines the size and density of initial cracks by controlling the strength distribution pattern.

Figure 5.5: Graphic illustration of performance on a single processor and on multi-processors for different raster sizes by a traditional Fourier transform

Figure 5.6: Graphic illustration of performance on a single processor and on multi-processors for different raster sizes by a Fast Fourier transform

Figure 5.7 shows two different patterns of initial strength distributions of rocks. When the strength is less than the critical value (local stress), the cell is “BROKEN” and becomes part of a crack - white cells in the images. If the strength is greater than the local stress but less than the “second critical value”, the cell is closer to be broken and it is set to red in the images. Otherwise, if the cell is solid - blue in that images. Figure 5.7a shows the first pattern in which the initial strength changes frequently and rapidly from cell to cell. The resulting image has a high density of initial cracks with small size. Such pattern can be realized by a big initial fractal dimension D . In contrast to Figure 5.7a, Figure 5.7b shows a gentle wave of strength distribution which causes fewer, but bigger cracks in the image.

In general, the larger the initial fractal dimension D is, the denser the initial cracks in the image are, the smaller the size of each crack, and *vice versa*. Figure 5.8 and 5.9 give examples of the initial raster images with different fractal dimension D .

Figure 5.8 is an initial raster image with fewer and larger cracks generated by setting the fractal dimension to 2.1, while Figure 5.9 shows very high density, small cracks with fractal dimension at 2.6.

In reality, the value ranges from 1.5 to 3.0 for fracture studies. By selection of different values of fractal dimension, earthquake researchers can simulate rock failures in different kinds of rocks with various patterns of strength and/or stress distribution.

5.3 User-defined Function for Propagation

User-defined functions for iteration in cellular automata model control the development of the model. In this study, Equation 2.5, a user-given function, controls the crack development during iterations.

Figure 5.10 to 5.15 give two sets of images which have the same initial raster (Figure 5.10 and Figure 5.13). Figure 5.11 and Figure 5.14 are the images after iterating 50 times while Figure 5.12 and Figure 5.15 show the result after 90 iterations. By comparing the corresponding images in the two sets, it was found that the crack system in Set 1 develops faster than that in Set 2. The reason for such a difference is that the parameter ρ in Equation 2.5 was set to 1 for Set 1 while it was 5 for Set 2.

(a) Pattern One of Strength Distribution

(b) Pattern Two of Strength Distribution

Figure 5.7: Patterns of initial strength distribution along cross sections of raster images

Figure 5.8: Initial raster images with fractal dimension $D = 2.1$

Figure 5.9: Initial raster images with fractal dimension $D = 2.6$

Figure 5.10: Initial raster image. $D = 2.4$, $\rho = 1$

Figure 5.11: Raster image after iterating 50 times. $\rho = 1$

Figure 5.12: Raster image after iterating 90 times. $\rho = 1$

Figure 5.13: Initial raster image. $D = 2.4$, $\rho = 5$

Figure 5.14: Raster image after iterating 50 times. $\rho = 5$

Figure 5.15: Raster image after iterating 90 times. $\rho = 5$

ρ	n_{min}	$f(n)$
1	4	<0
3	5	<0
5	6	<0

Table 5.3: List of ρ and the corresponding minimum number of failed neighbors n_{min}

The value of ρ controls the crack development during iterations. Table 5.3 lists different values of ρ corresponding minimum number of failed neighbors n_{min} required to make $f(n)$ less than 1. If ρ equals 1, $f(n)$ is less than 1 when the cell in question has 4 or more failed neighbors; ρ equals 3, then $f(n)$ will be less than 1 only when the number n of failed neighbors is at least 5; ρ equals 5, then $f(n)$ will be less than 1 only when the cell has at least 6 failed neighbors.

When $f(n)$ is less than 1, the new strength K_{new} for the current iteration will be less than the previous strength K_{old} since $K_{new} = f(n) \times K_{old}$. That means that the current iteration reduces the strength of the cell. This corresponds to “positive feedback” as described in Chapter 2 which occurs only when the number of failed neighbors is greater than a certain value n_{min} and this value is controlled by ρ .

With larger ρ , a cell needs a bigger number of failed neighbors to make the $f(n)$ less than 1 and to generate positive feedback. Therefore large ρ results in a slower development of crack growth. Small ρ , on the other hand, reduces the number of failed neighbors n_{min} which is required for crack growth and consequently causes faster growth of cracks. This rule is well illustrated in Figure 5.16. Cracks will grow only when the neighbors of the cell in question give positive feedback ($f(n)$ less than 1, but positive). This condition will be met only when the actual number of failed neighbor n_{act} is not less than the minimum number of failed neighbor n_{min} . While a large value of ρ corresponds a large n_{min} , for fewer cells in the raster holds $n_{act} \geq n_{min}$; then fewer cells will become broken and cracks grow slower.

Figure 5.16: The relationship between ρ and crack growth speed

By adjusting the value of ρ , the earthquake researcher can control and simulate the behavior of the rock sample under stress.

Chapter 6

Conclusions and Further Work

6.1 Conclusions

The cellular automata earthquake model implemented in this study runs successfully on the Cell Driver of the NEMO system. As the first outside test of the driver, the earthquake model provides an example of a GIS application of the parallel NEMO system. Our implementation proves that NEMO is a system that is well designed in its functionality and dramatically reduces the computation time, especially for spatial data applications on raster images.

The major part of the model which is most time-consuming, was implemented and performs well. As a consequence, it is possible for earthquake researchers to design more complex and more realistic cellular automata models which involve more parameters to simulate the fracture mechanism and therefore earthquake development. This study proves that the mechanical properties such as initial cracks and pore distribution of a rock sample can be controlled by the fractal dimension and that the behavior of the rock during crack growth under stress can be adjusted by a user-defined function.

6.2 Suggestions

From the application point of view, the author believes that some points on both the NEMO system and earthquake modeling are worthy of further improvement.

- **Transparency of Parallelism** - One of the goals of NEMO is to hide the parallel inheritance of the system from user. The Cell Driver reaches this goal in most of the scenarios, such as neighborhood message passing. However, in the case of input which involves not just reading already existing data from a database but by a generating procedure such as that in the initial stage of earthquake modeling, the author had to deal with the parallel issues and design functions for message passing between the different nodes. This requires users of NEMO to be quite familiar with the inside structure of the CD, and some knowledge of the Trollius operating system. If the CD can provide a default message passing function for general use besides neighborhood information exchange, it will greatly reduce the technical requirement on the application developer, and provide transparent parallelism at a greater level.
- **3-D** - The current earthquake model simulates the stress accumulation and/or rock behavior under stress in 2D. If a 3-dimensional simulation is implemented, the model will be much easier to understand for non-seismologists and have greater potential for utility in practice.

The key issue here seems to be the generation of a 3-D fractal distribution of stress or strength. This actually should not be too hard to be implemented on the basis of the current 2-D model, because the “separability” characteristics of FFT. A problem arising here is to display data which requires application programmer to be more skillful in the use of 3-D graphic techniques or requires extension to NEMO.

- **Realistic Data** - Due to lack of real data about rock strength from rock engineering and actual stress measurements, the model designed in this study is more theoretical. It can be easily modified by scaling the initial stage according

to the real data. Only with the input of real data, this model will show its potential for prediction of earthquakes.

- **Propagation Model** - Earthquake researchers explored various user-specified functions for controlling crack growth. Some of the functions only consider the states of the cells on the border of the existing cracks. In such cases, a propagation model instead of cellular automata model is more appropriate. Since the number of cells in question during iteration is reduced, especially when the density of cracks is low, the speedup by using the Propagation Driver in the NEMO system to simulate such earthquake models will be very significant.

Bibliography

- [1] Aki.K.. *Maximum likelihood estimate of b in the Formula $\log n = a - bm$* , Bull. Earthquake Res. Inst. Tokyo Univ., 43, 1965, pp. 237-239
- [2] Aronoff.S.. *Geographic Information Systems: A Management Perspective*. WDL Publications. Ottawa, Canada, 1989
- [3] Bak.P. and Chen.K., *Fractal Dynamics of Earthquakes*, in *Fractals in the Earth Sciences* edited by Barton, C.C. and La Point. P.R., Plenum Press, 1995. pp. 227-235
- [4] Bak.P. and Tang,C., *Geophys. Res.*, Vol. 94, No. 15, 1989
- [5] Barton.C.C.. *Fractal analysis of scaling and spatial clustering of fractures*. in *Fractals in the Earth Sciences* edited by Barton,C.C. and LaPoint,P.R., Plenum Press. 1995, pp. 141-177
- [6] Bennett,J.G., *Journal of Inst. Fuel*, 10, 1936
- [7] Ben-Zion,Y. and Rice,J.R., *Earthquake Failure Sequences Along a Cellular Fault Zone in a Three-Dimensional Elastic Solid Containing Asperity and Nonasperity Regions*. Journal of Geophysical Research, Vol. 98, No. B8, 1993, pp. 14109-14131
- [8] Bolt.B.A., *The size of an earthquake*, in *Earthquakes*, W.H. Freeman and Company. New York , 1988
- [9] Campbell.H. and Masser,I., *GIS and Organizations, How effective are GIS in practice*. Taylor & Francis, 1995

- [10] Chase.C.G.. *Geomorph*, Vol.5, No.39, 1992
- [11] Cormen,T.H., Leiserson,C.E., and Rivest,R.L., *Chapter 32 - Polynomials and the FFT in Introduction to Algorithm*, McGraw-Hill Publishing Company, 1990, pp. 776-780
- [12] Costin.L.S., *Deformation and failure in Fracture Mechanics of Rock* edited by Atkinson.B.K.. Academic Press. London. 1987. pp. 167-215
- [13] Franklin.J.A. *Rock engineering*, McGraw-Hill Publishing Company, 1989
- [14] Garzon.M., *Models of Massive Parallelism, Analysis of Cellular Automata and Neural Networks*, Springer, 1995
- [15] Greenwood.G.N., *Multipurpose Natural Resource GIS Databases - Altruism or Synergy? A Perspective from Arid Australia*. in *GIS Applications in Natural Resources 2* edited by Heit.M.. Parker.D., and Shortreid.A., GIS World. Inc.. 1996. pp. 250-291
- [16] Hartmann.W.K.. *Icarus*, 10, 1969, pp. 201
- [17] Henderson.J.R., Main,I.G., MaClean,C., and Norman,M.G., *A fracture-mechanical cellular automata model of seismicity*, PAGEOPH, Vol. 142, No. 3/4, 1994. pp. 545-565
- [18] Harris.D.B.. McClellan,J.H., Chan.D.S.K.. and Schuessler,H.W., *Vector radix fast fourier transform*, IEEE Int. Conf. Acoust.. Speech, Signal Processing, 1977, pp. 548-551
- [19] Huang,J. and Turcotte,D.L., *Fractal Distributions of Stress and Strength and Variations of b-value*, Earth and Planetary Science Letter, Vol. 91, 1988, pp. 223-230
- [20] Hutchinson,D., Küttner,L., Lanthier,M., Maheshwari,A., Nussbaum,D., Roytenberg.D.. and Sack,J.-R., *parallel neighborhood modeling*, research summary, Proc. 8th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'96). 1996. pp. 204-207

- [21] Huxhold, W.E. and Levinsohn, A.G., *Managing Geographic Information System Projects*, Oxford University Press, 1995
- [22] Johnson, A.I., Petterson, C. B., and Fulton, J.L., *Geographic Information Systems (GIS) and mapping - Practices and Standards*, ASTM special technical publication, 1992
- [23] Kanamori, H. and Anderson, D.L., *Theoretical basis of some empirical relations in seismology*, Bull. Seismol. Soc. Am., 1975, No. 65, pp. 1073-1096
- [24] Korvin, G., *Fractal Models in the Earth Sciences*, Elsevier, 1992
- [25] Lam, N.S. and Cola, L.D., *Fractals in Geography*, PTR Prentice Hall, 1993
- [26] Lomnitz-Adler, J., *Automata Models of Seismic Fracture: Constraints Imposed by the Magnitude-Frequency Relation*, Journal of Geophysical Research, Vol. 98, No. B10, 10/1993, pp. 17745-17756
- [27] Mandelbrot, B.B., *The Fractal Geometry of Nature*, Freeman, 1982
- [28] Naccarato, D.F. and Chien, Y.T., *A direct two-dimensional FFT with applications in image processing*, in Proc. 1979 IEEE Comput. Soc. Conf. Pattern Recognition and Image Processing, 1977, pp. 233-238
- [29] National Oceanic and Atmospheric Administration (NOAA), *Earthquake epicenters for magnitudes > 4.5, 1963-1977*, Earthquake Services and Publications, Key to Geophysical Records Documentation No. 15 (Revised), 1983, pp. 7
- [30] Pfeifer, P. and Obert, M., *The fractal approach to heterogeneous chemistry*, edited by Avnir, D., Wiley Chichester, England, 1989, pp. 11-43
- [31] Power, W.L. and Tullis, T.E., *Reviews of the Fractal Character of Natural Fault Surfaces with Implications for Friction and the Evolution of Fault Zones*, in *Fractals in the Earth Sciences* edited by Barton, C.C. and La Point, P.R., Plenum Press, 1995, pp. 89-103

- [32] Richter, C.F., *Elementary seismology*, W. H. Freeman and Company, San Francisco, 1958
- [33] Rivard, G.E., *Direct fast Fourier transform of bivariate functions*, IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-25, June 1977, pp. 250-252
- [34] Ripple, W.J., *Geographic Information Systems for Resource Management: A Compendium*, Falls Church, American Society for Photogrammetry and Remote Sensing, 1989
- [35] Romero-Rochin, V. and Lomnitz-Adler, J., *Avalanches in A Cellular Automata*, Physical Review E, Vol. 51, No. 5, 5/1995, pp. 3968-3977
- [36] Rudnicki, J.W. and Kanamori, H., *Effect of fault interaction on moment, stress drop and strain energy return*, Journal of Geophys. Res. vol. 86, 1981, pp. 1785-1793
- [37] Star, J. and Estes, J., *Geographic Information Systems, an introduction*, Prentice Hall International Ltd., 1990
- [38] Strahler, A.N., *Physical Geology*, Cambridge, Mass., Harper, 1981
- [39] US Geological Survey, *Magnitude and intensity: measure of earthquake size and severity*, Earthquakes and Volcanoes, Vol. 14, No. 6, pp. 209-219
- [40] Tomlinson, R.T., Calkins, H.W., and Marble, D.F., *Computer Handling of Geographical Data*, The Unesco Press, 1976
- [41] Turcotte, D.L., *Fractals and chaos in geology and geophysics*, Cambridge University Press, 1992
- [42] Turcotte, D.L. and Huang, J., *Fractal Distributions in Geology, Scale, Invariance, and Deterministic Chaos*, in *Fractals in the Earth Sciences* edited by Barton, C.C. and LaPoint, P.R., Plenum Press, 1995, pp. 1-38