



Cellular Automata as an Approximate Method in Structural Analysis

by

M. P. Hindley

A dissertation submitted in partial fulfillment
of the requirements for the degree of

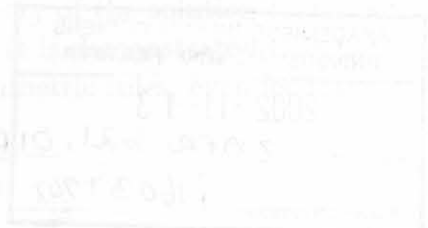
Master of Engineering

in the Department of Mechanical and Aeronautical Engineering,
University of Pretoria

November 2001

Supervisor:

Prof. Albert A. Groenwold





Abstract

Title: Cellular Automata as an Approximate Method in Structural Analysis

Author: Michael Philip Hindley

Supervisor: Prof. A.A. Groenwold

Department: Department of Mechanical Engineering

Degree: Master of Engineering

Keywords: Cellular automata, structural analysis, finite difference method, finite element method, boundary element method

This thesis deals with the mathematical idealization denoted cellular automata (CA) and the applicability of this method to structural mechanics. When using CA, all aspects such as space and time are discrete. This discrete nature of CA allows for ease of interaction with digital computers, while physical phenomena which are essentially discrete in nature can be simulated in a realistic way. The application of such a novel numerical method opens up new possibilities in structural analysis.

In this study, the fundamentals of CA are studied to determine how the parameters of the method are to be evaluated and applied to the established field of structural analysis. Attention is given to the underlying mathematics of structural mechanics, as well as approximate methods currently used in structural analysis, e.g. the finite element method (FEM) and the boundary element method (BEM).

For structural simulations performed with the CA implemented in this study, machine learning based on a genetic algorithm (GA) is used to determine optimum rules for the CA, using finite element, boundary element and analytical approximations as the basis for machine learning.

Rather unconventionally, symmetric problems in structural analysis are analyzed using asymmetric rules in the machine learning process, where the symmetry of the solution found is used as a quantitative indication of the quality of the solution. It is demonstrated that the quality of the asymmetric rules is superior to the quality of symmetric rules, even for those



problems that are symmetric in nature.

Finally, exploiting the inherent parallelism of CA, it is shown that distributed computing can greatly improve the efficiency of the CA simulation, even though the speed-up factor is not necessarily proportional to the number of sub lattices used.

The distributed computing device itself is constructed by combining 18 obsolete Pentium computers in a single cluster. In terms of CPU performance the constructed distributed computer is not state-of-the-art, but it is constructed with no hardware costs whatsoever. In addition, the software used in assembling the cluster is in the public domain, and is also available free of charge. Such a parallel configuration is also known as the poor man's computer. However, faster and more modern machines can simply be added to the existing cluster as and when they become available.

While CA are recent additions to the 'tools' used in structural analysis, increased use of CA as distributed computing becomes more widely available is envisaged, even though the CA rules are at this stage not transferable between different problems or even between meshes of varying refinement for a given problem.



Opsomming

- Titel:** Sellulêre Outomata as 'n Benaderingsmetode in Struktuuranalise
- Outeur:** Michael Philip Hindley
- Leier:** Prof. A.A. Groenwold
- Departement:** Departement van Meganiese Ingenieurswese
- Graad:** Meester van Ingenieurswese
- Sleutelwoorde:** Sellulêre outomata, struktuuranalise, eindige verskil metode, eindige element metode, rand element metode

Hierdie verhandeling is gemoed met die numeriese idealiseringsmetode genoem sellulêre outomata (SO), en die moontlike toepassings van hierdie metode in struktuurmeganika. SO is 'n metode waarin alle fisiese parameters soos tyd en ruimte met diskrete waardes benader word. Die diskrete aard van SO laat eenvoudige interaksies met rekenaars toe, terwyl fisiese verskynsels wat hoofsaaklik diskreet van aard is, op 'n realistiese wyse benader kan word. Die toepassing van so 'n nuwe metode in struktuuranalise laat uiteraard nuwe moontlikhede toe.

In hierdie studie word die fundamentele werking van die SO bestudeer om te bepaal hoe die parameters van die metode benader kan word, om dan ook toegepas te kan word op die gevestigde gebied van struktuuranalise. Aandag word verder ook gegee aan die onderliggende wiskundige begrippe van struktuurmeganika, asook benaderingsmetodes wat reeds gevestig is in struktuuranalise, soos byvoorbeeld die eindige element metode (EEM) en die rand element metode (REM).

In struktuursimulasies met behulp van die SO wat ontwikkel is, word masjien-onderrig met behulp van 'n genetiese algoritme (GA) gebruik om die optimale reëls vir die SO te bepaal. Die basis vir die leerproses is eindige element modelle, rand element modelle, asook analitiese benaderings.

Simmetriese probleme in struktuuranalise word op 'n onkonvensionele manier geanaliseer met behulp van onsimmetriese reëls in die masjienonderrig proses, terwyl die simmetrie van die voorspelde oplossings gebruik word as 'n kwalitatiewe aanduiding van die geskiktheid



van die reëls. Daar word aangetoon dat die kwaliteit van onsimmetriese reëls beter is as die kwaliteit van simmetriese reëls, selfs vir die probleme wat inherent simmetries is.

Laastens, deur gebruik te maak van die inherente parallelisme van die SO, word aangetoon dat verspreide berekenings die effektiwiteit van SO-berekenings kan verhoog, alhoewel die versnellingsfaktor nie noodwendig eweredig aan die aantal onderverdelings in die struktuur is nie.

Die verspreide rekenaar is saamgestel deur 18 verouderde Pentium-rekenaars in 'n enkele berekeningsgroep saam te stel. In terme van SVE-prestasie is die verspreide rekenaar nie op die voorgrond van tegnologie nie, maar is saamgestel sonder enige hardeware koste. Die sagteware wat gebruik is om die berekeningsgroep saam te stel is in die publieke domein, en is ook kosteloos beskikbaar. So 'n parallelle rekenaar staan ook bekend as die "arm man se parallellerekenaar". Vinniger en beter rekenaars kan egter eenvoudig tot die berekeningsgroep toegevoeg word wanneer hulle beskikbaar is.

Terwyl SO 'n onlangse stuk 'gereedskap' is wat in struktuuranalise gebruik word, word verwag dat SO toenemend gebruik sal word soos verspreide berekenings meer algemeen beskikbaar raak, nieteenstaande die feit dat die reëls op hierdie stadium nie oordraagbaar is tussen verskillende probleme nie, en ook nie vir verskillende grade van maasverfyning vir 'n bepaalde probleem nie.



Acknowledgments

I would like to express my sincere gratitude towards the following persons:

- Prof. Albert A. Groenwold, my supervisor, for his guidance and support throughout this study.
- To both my parents for their support, and for granting me the opportunity to study. I would also like to thank them for their financial backing.
- To my fellow students for turning a hard working environment into a pleasurable social experience.
- To every person involved with development of GNU (GPL) software. All the work performed in this thesis, including programming, editing and writing of this thesis, was accomplished using free software granted under the GNU public license.

I would like to dedicate this thesis to Laura Vatta.



Contents

Abstract	ii
Opsomming	iv
Acknowledgments	vi
List of Figures	xiv
List of Tables	xv
List of Abbreviations	xvi
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	3
1.3 Approach	4
2 Introduction to cellular automata	5
2.1 Informal definition	5
2.1.1 Simple example	6
2.2 Formal definition	6
2.3 Neighborhood definition and size in 2-D	7
2.4 Boundary conditions	8
2.5 Initial conditions	9
2.6 Geometry	9
2.6.1 One dimension	10
2.6.2 Two dimensions	10
2.6.3 Three dimensions	14
2.7 Cell state and precision	14

2.8	Cell rule	15
2.9	Cellular automata classification	16
2.10	CA internal representation possibilities	18
2.11	Partial differential equations, continuous systems and cellular automata . . .	19
3	Cellular automata in structural analysis	22
3.1	Introduction	22
3.1.1	CA parameters in structural analysis	23
3.1.2	CA implementation	25
3.2	Problem formulation	25
3.2.1	Problem descriptions	25
3.2.2	Neighborhood descriptions	26
3.2.3	Analysis criteria	27
3.3	Determination of the optimum cell rule	30
3.3.1	Motivation	30
3.3.2	Cell rule comparison definition	31
3.4	Problem analysis and results	32
3.4.1	Extended neighborhood	32
3.4.2	Moore neighborhood	37
3.5	Cellular automata computational conclusions	52
4	Parallel processing implementation	53
4.1	Introduction	53
4.1.1	Parallel computing concepts	53
4.1.2	Technology	54
4.1.3	Classification of parallel machines	54
4.1.4	Interconnecting structures and graphs	55
4.1.5	Parallel computing models and complexity measures	55
4.2	Parallel virtual machine, linux xpvm	56
4.2.1	Linux	56
4.2.2	Parallel virtual machine	57
4.2.3	Xpvm	59
4.3	Parallelization of cellular automata	62
4.3.1	Parallelization method	62
4.3.2	Sequential sub-lattice update method	64
4.3.3	Parallel lattice computation	66

4.4	Parallelization of a genetic algorithm	70
4.4.1	Introduction	70
4.4.2	Implementation method	70
4.5	Computational conclusions	72
5	Concluding Remarks	73
5.1	Summary of contributions	73
5.2	Conclusions	74
5.3	Directions for future work	74
A	Established structural approximation methods	80
A.1	Governing equations	80
A.1.1	One dimension	82
A.1.2	Two dimensions	82
A.1.3	Mechanics of the boundary value problem	82
A.2	Finite difference method	86
A.2.1	Introduction	86
A.2.2	Methodology	86
A.2.3	Methods of successive approximations	89
A.2.4	Practicality for structural analysis	91
A.3	Finite element method	93
A.3.1	Introduction	93
A.3.2	Methodology	94
A.3.3	Implementation	101
A.4	Boundary element method	102
A.4.1	Introduction	102
A.4.2	Basic principles	103
A.4.3	Fundamental solution	105
A.4.4	Boundary integral formulation	106
A.4.5	Boundary element formulation	108
A.4.6	Implementation	109
A.5	Finite element method and boundary element method comparison	115
B	Numerical optimization	117
B.1	Optimization formulation	117
B.1.1	Mathematical definition	117

List of Figures

2.1	Various CA neighborhood definitions	8
2.2	Shear mapping of the hexagonal lattice to the square lattice	11
2.3	Mapping of the nearest neighbor in the shear mapping	11
2.4	Shift mapping of the hexagonal lattice to the square lattice	11
2.5	Mapping of the nearest neighbors in the shift mapping	12
2.6	Mapping of the triangular lattice to the square lattice	13
2.7	Mapping of the nearest neighbors in the triangular mapping	13
2.8	Mapping of two triangular cells into one square cell	13
3.1	One-dimensional bar problem	24
3.2	One-dimensional computational cost comparison	24
3.3	Formulation of CA problem 1 investigated	26
3.4	Formulation of CA problem 2 investigated	26
3.5	Formulation of CA problem 3 investigated	26
3.6	Extended neighborhood description	27
3.7	Moore neighborhood description	27
3.8	Extended neighborhood RMS error comparison	34
3.9	Extended neighborhood RMS error closer comparison	34
3.10	Extended neighborhood computational cost comparison	35
3.11	Problem 1 FEM 8×8 stress plot	36
3.12	Problem 1 CA optimized 32 cell states stress plot	36
3.13	Problem 1 FDM 4 neighbors stress plot	36
3.14	Problem 1 CA optimized 1024 cell states stress plot	36
3.15	Problem 1 CA optimized 32768 cell states stress plot	36
3.16	Problem 1 Extended FDM stress plot	36
3.17	Cell state accuracy round off occurrence	38
3.18	Problem 1 FEM 16×16 stress plot	39
3.19	Problem 1 FDM 4 neighbors stress plot	39



3.20 Problem 1 CA optimized 32 cell states stress plot	39
3.21 Problem 1 CA optimized 256 cell states stress plot	39
3.22 Problem 1 CA optimized 256 cell states stress plot	39
3.23 Problem 1 CA optimized 32768 cell states stress plot	39
3.24 Problem 2 BEM 16 × 16 stress plot	41
3.25 Problem 2 FDM 4 neighbors stress plot	41
3.26 Problem 2 CA optimized 32 cell states stress plot	41
3.27 Problem 2 CA optimized 256 cell states stress plot	41
3.28 Problem 2 CA optimized 2048 cell states stress plot	41
3.29 Problem 2 CA optimized 32768 cell states stress plot	41
3.30 Problem 2 BEM 8 × 8 stress plot	42
3.31 Problem 2 BEM 16 × 16 stress plot	42
3.32 Problem 2 FEM 8 × 8 stress plot	42
3.33 Problem 2 FEM 16 × 16 stress plot	42
3.34 Problem 3 FEM 8 × 8 stress plot	44
3.35 Problem 3 CA optimized 32 cell states stress plot	44
3.36 Problem 3 CA optimized 64 cell states stress plot	44
3.37 Problem 3 CA optimized 512 cell states stress plot	44
3.38 Problem 3 CA optimized 1024 cell states stress plot	44
3.39 Problem 3 CA optimized 32768 cell states stress plot	44
3.40 Problem 3 FEM 16 × 16 stress plot	46
3.41 Problem 3 CA optimized 32 cell states stress plot	46
3.42 Problem 3 CA optimized 128 cell states stress plot	46
3.43 Problem 3 CA optimized 512 cell states stress plot	46
3.44 Problem 3 CA optimized 2048 cell states stress plot	46
3.45 Problem 3 CA optimized 32768 cell states stress plot	46
3.46 Problem 3 FEM 16 × 16 stress plot	48
3.47 Problem 3 CA optimized 64 cell states stress plot	48
3.48 Problem 3 CA optimized 128 cell states stress plot	48
3.49 Problem 3 CA optimized 512 cell states stress plot	48
3.50 Problem 3 CA optimized 8192 cell states stress plot	48
3.51 Problem 3 CA optimized 32768 cell states stress plot	48
3.52 Asymmetric neighborhood problem description	49
3.53 Asymmetric rule orientation for Problem 3 FEM 8 × 8 1024 cell states	50
3.54 Asymmetric rule one problem 3 stress plot, FEM 8 × 8 1024 cell states . . .	51



3.55 Asymmetric rule two problem 3 stress plot, FEM 8×8 1024 cell states . . . 51

3.56 Asymmetric rule three problem 3 stress plot, FEM 8×8 1024 cell states . . 51

3.57 Asymmetric rule four problem 3 stress plot, FEM 8×8 1024 cell states . . . 51

3.58 Asymmetric rule five problem 3 stress plot, FEM 8×8 1024 cell states . . . 51

3.59 Asymmetric rule six problem 3 stress plot, FEM 8×8 1024 cell states 51

4.1 Photo of the 18 node Linux “Souper” Computer, named GOH, constructed
 in this study 60

4.2 XPVM screen shot 61

4.3 Original square lattice 63

4.4 Four sub-lattice split 63

4.5 Nine sub-lattice split 63

4.6 Typical convergence for four sub-lattices 70

4.7 GA calculation with nine population members 71

A.1 The main coordinates, displacements, stresses and tractions in three dimensions 81

A.2 One-dimensional bar problem 83

A.3 Equilibrium of an infinitesimal bar element 83

A.4 Finite difference two-dimensional grid 87

A.5 Finite difference torsion bar example cross section 88

A.6 Typical ritz function used in the analysis of one-dimensional bar problem . . 94

A.7 Linear combination of the ritz functions 94

A.8 Simple bar FEM mesh 95

A.9 Simple membrane FEM element 97

A.10 FEM element patch test 102

A.11 The fundamental solution in two dimensions BEM 107

A.12 Constant element solution BEM 110

A.13 Corner point 110

A.14 Simple boundary element problem 112

A.15 Applying boundary conditions 112

A.16 Applying complex load 113

A.17 Constant deformation of a simple geometry 113

A.18 Constant deformation of a refined mesh 114

C.1 Boundary cell calculation 124

C.2 CA simulation main 125

C.3 Stress contour plot 126

C.4 Genetic algorithm 127

C.5 CA parallel implementation number one 128

C.6 CA parallel implementation number two 129

C.7 CA parallel implementation number three 130

C.8 GA parallel implementation 131

2.1 A simple one-dimensional, two state, cellular automata 14

2.2 Number of possible combinations 14

2.3 Possible particles 15

2.4 Neighborhood size differs use for different mesh sizes 15

2.5 Internal Computer Representation for a computerized CA mapping to a finite element mesh 15

3.1 Displacement state comparison of BEM and CA in a one dimensional FEM 17

3.2 Optimized rules for 5 x 5 mesh on Pyram at frequency of 1000 Hz 17

3.3 Optimized rules for 16 x 16 mesh on Moore graphlet and FEM problem 17

3.4 Optimized rules for 16 x 16 mesh on Moore graphlet and FEM problem 17

3.5 Optimized rules for 8 x 8 mesh on Moore graphlet and FEM problem 17

3.6 Optimized rules for 16 x 16 mesh on Moore graphlet and FEM Problem 17

3.7 Optimized rules for 16 x 16 speed of numerical state and internal displacement FEM Problem 2 17

3.8 Optimized rules 5 x 8 FEM problem 14 quadrants 17

4.1 Sequential block update method with GPU and state update of 1000 nodes 17

4.2 Local convergence speed large system for a unit BEM 17

4.3 Parallel convergence speed performance of CA vs FEM 17

5.1 Established finite difference computational rules 17

5.2 Displacements calculated by FEM single element 17

5.3 Stresses Calculated by FEM single element 17

5.4 Internal stresses (BEM) 17

5.5 Internal stresses refined mesh (BEM) 17

5.6 Internal displacements refined mesh (BEM) 17

6.1 17

6.2 17

6.3 17

List of Tables

2.1	A simple one-dimensional, two state, cellular automata	6
2.2	Number of possible combinations	14
2.3	Possible precision	15
2.4	Neighborhood size difference for difference mesh types	18
2.5	Internal Computer Representation for a complicated CA mapped to a square computational lattice	19
3.1	Displacement error comparison of BEM and CA in comparison to FEM . . .	28
3.2	Optimized rules for 8×8 mesh on Extended neighborhood FEM problem 1 .	33
3.3	Optimized rules for 16×16 mesh on Moore neighborhood FEM problem 1 .	37
3.4	Optimized rules for 16×16 mesh on Moore neighborhood BEM problem 2 .	40
3.5	Optimized rules for 8×8 mesh on Moore neighborhood FEM problem 3 . .	43
3.6	Optimized rules for 16×16 mesh on Moore neighborhood FEM Problem 3 .	45
3.7	Optimized rules for 16×16 mesh for more cell states on a Moore neighborhood FEM Problem 3	47
3.8	Optimized rules 8×8 FEM problem 3 (asymmetric)	49
4.1	Sequential block update method with 32768 cell states and finite difference rule(N, Number of sub-lattices)	65
4.2	Local convergence speed improvement for 4 sub-lattices	66
4.3	Parallel computing speed performance of CA with PVM	69
A.1	Established finite difference computational molecules	92
A.2	Displacements calculated by FEM (single membrane element)	101
A.3	Stresses Calculated by FEM (single membrane element)	102
A.4	Internal stresses (BEM)	114
A.5	Internal stresses refined mesh (BEM)	115
A.6	Internal displacements refined mesh (BEM)	115



List of Abbreviations

1-D	One dimensional
2-D	Two dimensional
3-D	Three dimensional
BEM	Boundary element method
CA	Cellular automata
CPU	Central processing unit
CSE	Computational science and engineering
FDM	Finite difference method
FEM	Finite element method
GA	Genetic algorithm
IPC	Inter-process communications
MIMD	Multiple instruction multiple data
MPP	Massive parallel processors
NFS	Networked file system
NOW	Network of workstations
OS	Operating system
PDE	Partial differential equation
PVM	Parallel virtual machine
RISC	Reduced instruction set computing
RMS	Root mean square
SIMD	Single instruction multiple data
VLSI	Very large scale integration