FINITE ELEMENT MESH OPTIMIZATION USING GENETIC ALGORITHMS

by

Arsen V. Tonoyan

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF AEROSPACE AND MECHANICAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY
WITH A MAJOR IN MECHANICAL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

2004

The University of Arizona ®
Graduate College

As members of the Final Examination Committee, we certify that we have read the dissertation prepared by Arsen V. Tonoyan

entitled Finite Element Mesh Optimization Using Genetic Algorithms

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy

A. Arabyan Ara Arabyan	11/2/04 date
W. Chen Wayne Chen	11/2/04 date
Bruce Simon Bruce Simon	11/2/04 date
	date
	date

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

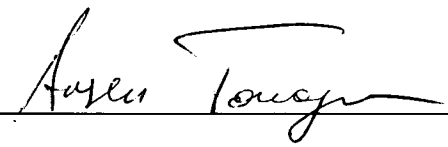
I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

A. Arabyan — 11/29/04
Dissertation Director: — date

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: 

ACKNOWLEDGEMENTS

This manuscript would not be possible without Dr. Ara Arabyan (University of Arizona) bringing to my attention the problem of finite element mesh optimization and foreseeing the applicability and robustness of Genetic Algorithms for this type of optimizations.

I deeply appreciate Dr. Arabyan's guidance and advice during the research and preparation of this dissertation.

I would like to thank my committee members Dr. Bruce Simon, Dr. Weinong Chen, Dr. Achintya Haldar and Dr. Tribikram Kundu for their time and patience in reviewing this manuscript.

I owe a big deal of my notion of Mechanics and Finite Element Methods to my teachers Dr. Vladimir Sargsyan and Dr. Michael Gabrielian (Yerevan State University); Dr. Movses Kaldjian (American University of Armenia) and Dr. Ara Arabyan (University of Arizona and American University of Armenia); Dr. Parviz Nikravesh and Dr. Bruce Simon (University of Arizona).

I deeply appreciate support of my friends and colleagues from the University of Arizona 1994-1999, from the American University of Armenia and its Engineering Research Center 1992-1994, and from the Yerevan State University 1985-1992.

Finally, I would like to thank my dear wife Anna Knyazyan for her love, support and encouragement during all these years.

DEDICATION

I would like to dedicate this work to my mother, Mareta Karapetian, who has raised three children by herself with dignity in this difficult world and to my brother, Aram Tonoyan, who since my early years has planted fascination towards mathematics in me and has taught me how to learn and think both in science and in life.

Table of Contents

LIST OF FIGURES **8**

ABSTRACT **13**

NOTATIONS AND ABBREVIATIONS **14**

CHAPTER 1 ..... **16**

INTRODUCTION **16**

STRUCTURAL OPTIMIZATION 19

SIGNIFICANCE 21

STRUCTURE OF THE THESIS 22

CHAPTER 2 ..... **24**

GENETIC ALGORITHMS **24**

CHAPTER 3 ..... **32**

ERROR ESTIMATION IN FINITE ELEMENT MESHES **32**

OUTLINE OF "Z^2" ERROR ESTIMATOR 33

CHAPTER 4 ..... **38**

FORMULATION OF MESH OPTIMIZATION PROBLEM AND GENERAL APPROACH **38**

CHAPTER 5 ..... **49**

IMPLEMENTATION OF GAS SCHEME AND RESULTS **49**

FINITE ELEMENT MESH OPTIMIZATION IN 1D [1] 49

Example 1. A cantilever beam 50

FINITE ELEMENT MESH OPTIMIZATION IN 2D 58

Example 2. A thin square plate 59

Example 3. A thin plate with an opening 72

CHAPTER 6 ..... **97**

MESH GENERATION IN COMMERCIAL FINITE ELEMENT ANALYSIS SOFTWARE **97**

CHAPTER 7 ..... **111**

CONCLUDING REMARKS **111**

SUGGESTED DIRECTIONS FOR ENHANCEMENTS 114

APPENDIX. PYTHON SCRIPT FOR ERROR CALCULATION IN ABAQUS **123**

REFERENCES **126**

List of Figures

FIGURE 1.1. ADAPTIVE MESH OF SHORT CANTILEVER BEAMS - MESH ENRICHMENT - ADAPTIVE REFINEMENT TO ACHIEVE 5 PERCENT ACCURACY [22] 18

FIGURE 1.2. ADAPTIVE MESH OF SHORT CANTILEVER BEAMS - MESH REGENERATION - ADAPTIVE REFINEMENT TO ACHIEVE 5 PERCENT ACCURACY [22] 18

FIGURE 1.3. THE DESIGN DOMAIN AND RESULTS OF (A) 10×16, (B) 15×24, (C) 20×32 OPTIMIZATIONS [31] 20

FIGURE 2.1. EXAMPLES OF BINARY AND REAL-NUMBER *CHROMOSOMES* 25

FIGURE 2.2. SCHEMATIC REPRESENTATION OF GAS PROCESS 26

FIGURE 2.3. EXAMPLE OF ONE-POINT *CROSSOVER* OPERATION 28

FIGURE 2.4. EXAMPLE OF TWO-POINT *CROSSOVER* OPERATION 28

FIGURE 2.5. EXAMPLE OF *MUTATION* OPERATION 29

FIGURE 3.1. PROBLEM OF LINEAR ELASTICITY OF A STRUCTURE AND ITS FINITE ELEMENT MODEL 34

FIGURE 4.1. STATIC EQUILIBRIUM IN FINITE ELEMENT PROBLEM DISCRETIZATION 39

FIGURE 4.2. FE ANALYSIS FLOWCHART 40

FIGURE 4.3. EVALUATION OF OBJECTIVE FUNCTION AND FITNESS ASSIGNMENT 42

FIGURE 4.4. SCHEMATIC REPRESENTATION OF FE MESH OPTIMIZATION USING GAS 44

FIGURE 4.5. SQUARE PLATE WITH FOUR ELEMENTS 46

FIGURE 4.6. BEAM (TRUSS) WITH THREE ELEMENTS 47

FIGURE 5.1. CANTILEVER BEAM 50

FIGURE 5.2. FINITE ELEMENT MODEL OF THE BEAM 50

FIGURE 5.3. LOAD TYPES 51

FIGURE 5.4. AVERAGE BEHAVIOR FOR THE BEAM WITH A LINEARLY DISTRIBUTED LOAD (TYPE 1): RANDOM SEARCH (1), GAS WITH BINARY CODING (2), AND GAS WITH REAL CODING (3) 53

FIGURE 5.5. AVERAGE BEHAVIOR FOR THE BEAM WITH A STEP LOAD (TYPE 2): RANDOM SEARCH (1), GAS WITH BINARY CODING (2), AND GAS WITH REAL CODING (3) 53

FIGURE 5.6. OPTIMAL MESHES: FOR LINEARLY DISTRIBUTED LOAD (#1) AND STEP LOAD (#2) 55

FIGURE 5.7. CURVES OF EQUAL ERROR (SCALED WRT MINIMUM ERROR) FOR LOAD TYPES 1 AND 2 55

FIGURE 5.8. TICK MARK CONTOUR OF DISPLACEMENT *Y*-COMPONENT FOR LINEARLY DISTRIBUTED LOAD (#1) AND STEP LOAD (#2) 56

FIGURE 5.9. SQUARE PLATE UNDER PLANE STRESS CONDITIONS 60

FIGURE 5.10. IN-PLANE MAXIMUM PRINCIPAL STRESS FOR PLATE PROBLEM: BANDED CONTOUR 61

FIGURE 5.11. IN-PLANE MAXIMUM PRINCIPAL STRESS FOR PLATE PROBLEM: LINE CONTOUR 61

FIGURE 5.12. RANDOMLY GENERATED FINITE ELEMENT MESH OF THE PLATE 62

FIGURE 5.13. OPTIMAL MESHES FOR 9-, 16-, 25- AND 36-ELEMENT MODELS: GENERAL QUADRILATERAL ELEMENTS 64

FIGURE 5.14. MESHING WITH RECTANGULAR ELEMENTS 65

FIGURE 5.15. OPTIMAL MESH FOR 36-ELEMENT MODEL: GENERAL RECTANGULAR ELEMENTS 66

FIGURE 5.16. FUNCTIONAL RELATIONS, BIASING 68

FIGURE 5.17. OPTIMAL MESH FOR 36-ELEMENT MODEL: RECTANGULAR ELEMENTS, BIASING REPRESENTATION 69

FIGURE 5.18. OPTIMAL MESH FOR 36-ELEMENT MODEL: RECTANGULAR ELEMENTS, SINE-COSINE REPRESENTATION 70

FIGURE 5.19. CONVERGENCE OF GENETIC ALGORITHMS FOR: (A) 25-ELEMENT MESH WITH QUADRILATERAL ELEMENTS (TWO SEPARATE RUNS); (B) 36-ELEMENT MESH WITH QUADRILATERAL ELEMENTS; (C) 36-ELEMENT MESH WITH GENERAL RECTANGULAR ELEMENTS; AND (D) 36-ELEMENT MESH WITH RECTANGULAR ELEMENTS, HARMONIC REPRESENTATION 71

FIGURE 5.20. SQUARE PLATE WITH OPENING UNDER PLANE STRESS 73

FIGURE 5.21. IN-PLANE MAXIMUM PRINCIPAL STRESS FOR PLATE WITH OPENING: BANDED CONTOUR 74

FIGURE 5.22. IN-PLANE MAXIMUM PRINCIPAL STRESS FOR PLATE WITH OPENING: LINE CONTOUR 74

FIGURE 5.23. FINITE ELEMENT MODEL OF THE PLATE WITH OPENING 75

FIGURE 5.24. OPTIMAL MESHES FOR N_{ACTIVE} = 50 AND N_{ADAPT} =25 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 0 / 100 78

FIGURE 5.25. OPTIMAL MESHES FOR N_{ACTIVE} = 50 AND N_{ADAPT} =25 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 3 / 97 79

FIGURE 5.26. OPTIMAL MESHES FOR N_{ACTIVE} = 50 AND N_{ADAPT} =25 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 50 / 50 79

FIGURE 5.27. OPTIMAL MESHES FOR N_{ACTIVE} = 50 AND N_{ADAPT} =25 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 80 / 20 80

FIGURE 5.28. OPTIMAL MESHES FOR N_{ACTIVE} = 50 AND N_{ADAPT} =25 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 100 / 0 80

FIGURE 5.29. OPTIMAL MESHES FOR N_{ACTIVE} = 50 AND N_{ADAPT} =50 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 0 / 100 81

FIGURE 5.30. OPTIMAL MESHES FOR N_{ACTIVE} = 50 AND N_{ADAPT} =50 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 3 / 97 81

FIGURE 5.31. OPTIMAL MESHES FOR N_{ACTIVE} = 50 AND N_{ADAPT} =50 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 50 / 50 82

FIGURE 5.32. OPTIMAL MESHES FOR N_{ACTIVE} = 50 AND N_{ADAPT} =50 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 80 / 20 82

FIGURE 5.33. OPTIMAL MESHES FOR N_{ACTIVE} = 50 AND N_{ADAPT} =50 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 100 / 0 83

FIGURE 5.34. OPTIMAL MESHES FOR N_{ACTIVE} = 200 AND N_{ADAPT} =25 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 0 / 100 83

FIGURE 5.35. OPTIMAL MESHES FOR N_{ACTIVE} = 200 AND N_{ADAPT} =25 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 3 / 97 84

FIGURE 5.36. OPTIMAL MESHES FOR N_{ACTIVE} = 200 AND N_{ADAPT} =25 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 50 / 50 84

FIGURE 5.37. OPTIMAL MESHES FOR N_{ACTIVE} = 200 AND N_{ADAPT} =25 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 80 / 20 85

FIGURE 5.38. OPTIMAL MESHES FOR N_{ACTIVE} = 200 AND N_{ADAPT} =25 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 100 / 0 85

FIGURE 5.39. OPTIMAL MESHES FOR N_{ACTIVE} = 200 AND N_{ADAPT} =50 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 0 / 100 86

FIGURE 5.40. OPTIMAL MESHES FOR N_{ACTIVE} = 200 AND N_{ADAPT} =50 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 3 / 97 86

FIGURE 5.41. OPTIMAL MESHES FOR N_{ACTIVE} = 200 AND N_{ADAPT} =50 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 50 / 50 87

FIGURE 5.42. OPTIMAL MESHES FOR N_{ACTIVE} = 200 AND N_{ADAPT} =50 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 80 / 20 87

FIGURE 5.43. OPTIMAL MESHES FOR N_{ACTIVE} = 200 AND N_{ADAPT} =50 CASE WITH INITIAL POPULATION QUADRILATERAL / RECTANGULAR MESH RATIO OF 100 / 0 88

FIGURE 5.44. CONVERGENCE OF GAS FOR N_{ACTIVE} = 50 AND N_{ADAPT} = 25, AND VARIOUS RATIOS OF QUADRILATERAL / RECTANGULAR MESHES IN INITIAL POPULATION 90

FIGURE 5.45. CONVERGENCE OF GAS FOR N_{ACTIVE} = 50 AND N_{ADAPT} = 50, AND VARIOUS RATIOS OF QUADRILATERAL / RECTANGULAR MESHES IN INITIAL POPULATION 91

FIGURE 5.46. CONVERGENCE OF GAS FOR N_{ACTIVE} = 200 AND N_{ADAPT} = 25, AND VARIOUS RATIOS OF QUADRILATERAL / RECTANGULAR MESHES IN INITIAL POPULATION 92

FIGURE 5.47. CONVERGENCE OF GAS FOR N_{ACTIVE} = 200 AND N_{ADAPT} = 50, AND VARIOUS RATIOS OF QUADRILATERAL / RECTANGULAR MESHES IN INITIAL POPULATION 93

FIGURE 5.48. MINIMUM / MAXIMUM ENVELOPE OF GAS CONVERGENCE OVER CHOICE OF INITIAL POPULATION FOR VARIOUS N_{ACTIVE} AND N_{ADAPT} 95

FIGURE 5.49. MEAN VALUE OF GAS CONVERGENCE OVER CHOICE OF INITIAL POPULATION FOR VARIOUS N_{ACTIVE} AND N_{ADAPT} 96

FIGURE 6.1. CONTOUR OF DISPLACEMENT MAGNITUDE FOR 'FINE MESH' PLATE WITH OPENING 99

FIGURE 6.2. CONTOUR OF DISPLACEMENT MAGNITUDE USING MEDIAL AXIS AND ADVANCING FRONT MESHING TECHNIQUES 101

FIGURE 6.3. CONTOUR OF DISPLACEMENT MAGNITUDE USING RECTANGULAR AND TRIANGULAR MESHING TECHNIQUES 102

FIGURE 6.4. CONTOUR OF DISPLACEMENT MAGNITUDE USING RECTANGULAR MESHING WITH VARIOUS BIAS SEEDING RATIOS 104

FIGURE 6.5. CONTOUR OF DISPLACEMENT MAGNITUDE USING QUADRILATERAL MESHING WITH VARIOUS BOUNDARY SEEDING LOCATIONS 105

FIGURE 6.6. CONTOUR OF IN-PLANE MAXIMUM PRINCIPAL STRESS FOR 'FINE MESH' PLATE WITH OPENING .. 107

FIGURE 6.7. CONTOUR OF IN-PLANE MAXIMUM PRINCIPAL STRESS USING MEDIAL AXIS AND ADVANCING FRONT MESHING TECHNIQUES 108

FIGURE 6.8. CONTOUR OF IN-PLANE MAXIMUM PRINCIPAL STRESS USING RECTANGULAR AND TRIANGULAR MESHING TECHNIQUES 108

FIGURE 6.9. CONTOUR OF IN-PLANE MAXIMUM PRINCIPAL STRESS USING RECTANGULAR MESHING WITH VARIOUS BIAS SEEDING RATIOS 109

FIGURE 6.10. CONTOUR OF IN-PLANE MAXIMUM PRINCIPAL STRESS USING QUADRILATERAL MESHING WITH VARIOUS BOUNDARY SEEDING LOCATIONS 109

FIGURE 7.1. FINITE ELEMENT MODELS OF A PLATE: (A) A 16-ELEMENT MESH WITH FIXED NUMBER OF NODES ALONG EACH BOUNDARY; (B) A 16-ELEMENT MESH WITHOUT A RESTRICTION ON NUMBER OF NODES ALONG BOUNDARIES 113

FIGURE 7.2. MESH GENERATION USING POSITIONS OF ELEMENT CENTERS 116

FIGURE 7.3. QUARTER-SYMMETRY MODEL OF PISTON ASSEMBLY 118

ABSTRACT

In finite element analysis, structures are modeled as meshes of elements and nodes appropriate for the geometry, boundaries and loading of each structure. Typically, it is desirable to have a mesh which is finer in parts of the structure where stress gradients are high and coarser where such gradients are low. This is usually done by experienced engineers using intuition and previous experience. Otherwise, a fine mesh throughout the structure can be used which results in high computational costs.

In this work, the possibility of using genetic algorithms for optimizing finite-element meshes is studied. The method is implemented on a number of simple loaded structures. The meshes used are generated using a number of parameters that can be varied randomly. Then the parameters are varied using operators appropriate to genetic algorithms such that the value of an objective function is minimized within a defined precision and iteration limit. The objective function used in this study is an energy-based error norm. The results obtained with this method are compared to those obtained from a commercial finite element package that incorporates its own mesh optimization algorithms.

Notations and Abbreviations

GAs	Genetic Algorithms
FE	Finite Element
FEA	Finite Element Analysis
FEM	Finite Element Method
DOF	Degrees of Freedom
Z^2	Zienkiewicz and Zhu error estimator
OS	Operating System in a computer
$\boldsymbol{P}(t)$	population at time t (iteration)
$J(\boldsymbol{x})$	optimality criterion over vector of design parameters, $\boldsymbol{x}$, for FE optimization
$\mathbf{K}^e$	element stiffness matrix
$\mathbf{K}$	global stiffness matrix
$\mathbf{u}$	vector of generalized displacement fields
$\mathbf{u}_0$	vector of displacements on the boundary Γ_u
$\underline{\mathbf{u}}^e$	nodal displacements for a particular element, where superscript e stands for individual finite element
$\underline{\mathbf{u}}$	vector of displacements all nodal points
$\mathbf{f}$	vector of nodal forces incorporating the body forces and the boundary conditions

$\mathbf{b}$	vector of body forces
$\mathbf{t}_0$	vector of tractions on the boundary Γ_t
$\boldsymbol{\sigma}$	generalized stress in domain Ω
$\mathbf{N}$	shape functions in FE formulation
$\mathbf{T}$	transformation matrix relating nodal displacements to element stresses
$\mathbf{S}$	first order strain differential operator matrix
$\mathbf{D}$	stress-strain matrix
$\mathbf{e}_u$	displacement local error
$\mathbf{e}_\sigma$	stress local error
$\|x\|$	Euclidian norm of vector x
$\mathbf{u}^M$	displacement vector for the "fine mesh" model evaluated at the fine mesh nodes ("exact" solution)
$\mathbf{u}^{N\to M}(\mathbf{x})$	displacements vector interpolated from the "coarse mesh" model at the fine mesh nodes
U	output for displacement magnitude field from ABAQUS
S_{max}	output for in-plane maximum principal stress field from ABAQUS
N_{Active}	number of active members (individual meshes) in a current population
N_{Adapt}	number of newly created members (children), to be considered as replacements in every generation (iteration)

Chapter 1

Introduction

The problem of constructing optimal finite element (FE) meshes has been considered by a number of authors in the past decades [2, 4, 6, 7, 8, 9, 10, 11]. To find better finite element approximations, many different mesh refinement techniques have been considered. The most common techniques are *p*-refinement [11], where the polynomial order of FE approximation is varied to obtain higher accuracies; and *h*-refinement [2, 4], where new mesh points (nodes) are added locally to produce a finer mesh. The latter procedure is also known as mesh subdivision or enrichment. Mesh enrichment has been used in much of the early research on adaptive analysis procedures [23, 24, 25, 26]. With mesh enrichment the local mesh is divided so that the elements with the largest local estimated error are refined. In this process, guided by an error estimator, the error distribution is controlled precisely and the final mesh obtained is, in general, very close to the optimal mesh.

However, the disadvantage of mesh enrichment is that many steps of adaptive refinement are required to avoid elements being refined unnecessarily. Since every adaptive refinement also involves a reanalysis of the problem considered, mesh enrichment is in general not an efficient procedure for practical computations [22]. The adaptive procedure using mesh regeneration, on the other hand, needs fewer reanalysis steps to achieve a prescribed accuracy. A comparison of mesh enrichment with mesh regeneration by Zhu, Hinton, and Zienkiewicz [22] for a short cantilever beam under plane strain conditions using bilinear elements is shown below. As can be seen, starting from the same initial mesh, the desired accuracy of 5% is achieved in six adaptive steps using mesh enrichment (Figure 1.1) versus two adaptive steps using mesh regeneration (Figure 1.2). In addition, in the adaptive analysis procedures the mesh regeneration process has been used successfully in many other applications [3, 27, 28, 29, and 30].

The main theme of this study is so-called *r*-refinement [6, 7], in which an existing finite element mesh is modified by changing the locations of nodes to achieve accuracy improvements. Thus in this process accuracy improvement is achieved without changing the number of nodes, that is degrees of freedom (DOF) in the mesh. Even though by this process only improvements to an existing solution are possible and a specified accuracy cannot be obtained, a combination of *r*-refinement and mesh enrichment may attain desired accuracy at the lowest possible computational expense, resulting in a finite element mesh with the fewest possible elements.

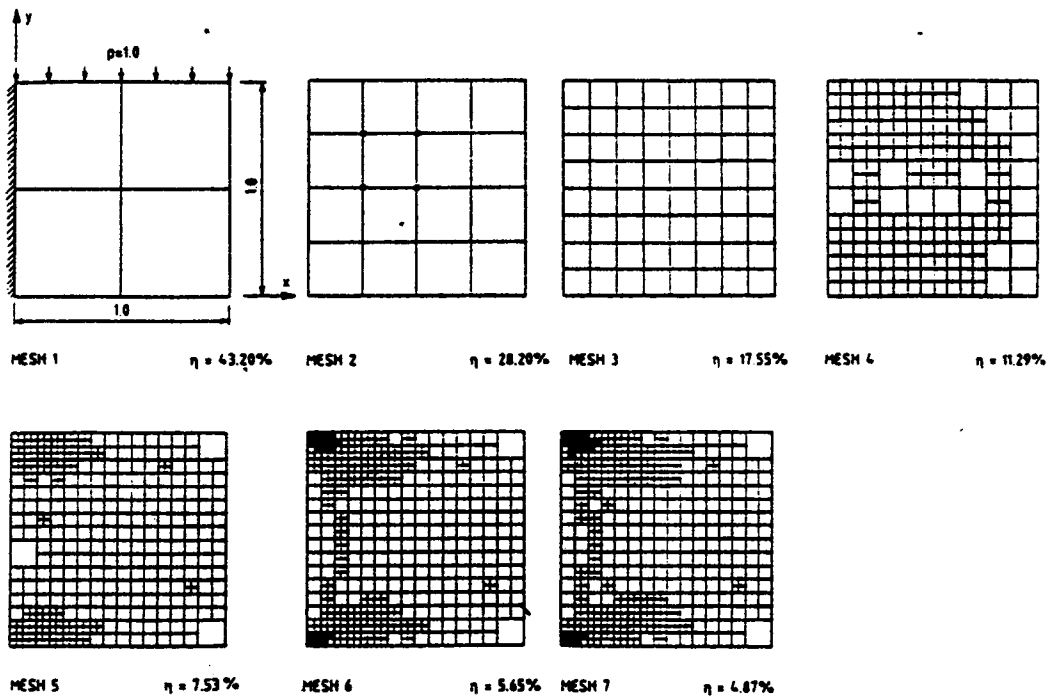


Figure 1.1. Adaptive mesh of short cantilever beams - mesh enrichment - adaptive refinement to achieve 5 percent accuracy [22]

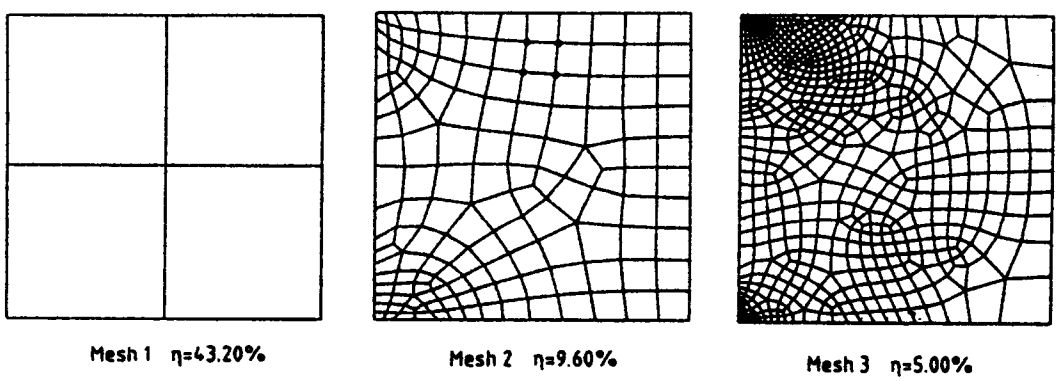


Figure 1.2. Adaptive mesh of short cantilever beams - mesh regeneration - adaptive refinement to achieve 5 percent accuracy [22]

The optimization technique used here to refine a given mesh is different from the methods used by other researchers and is based on the recently developed technique of genetic algorithms (GAs) [12, 13, 15, 16]. Compared to traditional gradient-based optimization methods, GAs are more capable of dealing with complex multidimensional problems and finding global minima of complicated objective functions with numerous local minima. In addition, unlike gradient-based optimization methods GAs do not require a user to make an initial guess that is close to the global minimum being sought. More importantly, the GAs process is explicitly parallel and has the potential to achieve considerable gains in computational speed with the use of parallel computers.

Structural Optimization

Optimization of structures using GAs has been considered by many authors in recent years [31, 32, 33, 34]. Though these studies only conceptually are related to the topic considered in this study, that is FE mesh optimization, they still contain valuable information on the usage of GAs in structural optimization problems. The studies discussed below have one common objective: minimize the total weight of the structure considered.

Chapman, Saitou and Jakiela in their paper [31] consider an example of a cantilevered plate subject to a vertical load. The design domain is divided into a grid of pixels. *Chromosomes*, used in GAs, consist of one-dimensional strings of binary digits, where the number of digits, or *genes*, in each chromosome equals the number of pixels in

the design domain. A gene with a value of 1 places material in its corresponding design domain element (pixel), while a gene with a value of 0 places void in the corresponding element. Thus a chromosome is mapped into a design domain which defines material topology in the domain being optimized. The FE model optimization is done for several grids and the results are shown in Figure 1.3.

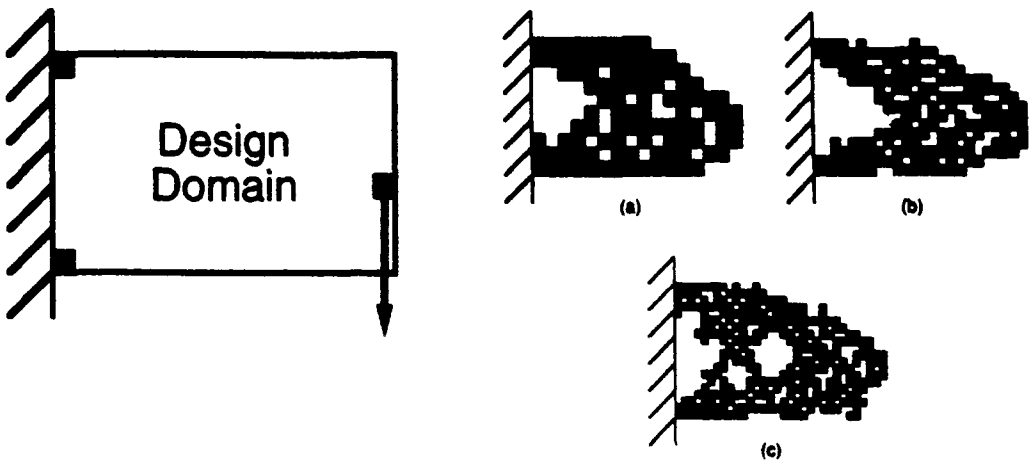


Figure 1.3. The design domain and results of (a) 10×16, (b) 15×24, (c) 20×32 optimizations [31]

With the 20×32 discretization, $18000 \approx 2^{14}$ search locations were examined. As the entire space contains 2^{640} locations, the GAs searched only a fraction of the space before finding a near optimum solution. An exhaustive search would examine all 2^{640} locations, and random search would likely find the optimum location after searching half of the locations. This example shows that the GAs are more efficient than other basic

techniques which are also able to search in discrete, discontinuous, multi-modal search spaces.

Significance

Throughout the history of research and applications of finite element methods, decisions on the applicability of results and assessment of their accuracy has been frequently done by 'experience' of previous computations and rules on permissible element shapes and sizes. Automatic mesh generators, currently used in commercially available finite element packages, require user intervention to assign priorities for finer areas in the finite element discretization. This has made the application of the finite element method an "art" and created the belief that a good mesh can be constructed only by an "experienced" and "fully qualified" engineer.

This situation is changing thanks to recently introduced error estimates that are capable of assessing the discretization error at relatively small cost. Automating the refinement of the approximation is indeed possible in many applications, and quantitative and/or adaptive procedures will almost always produce better results than those guided by "experience", though the latter is always helpful. One of the main benefits of finding an optimal mesh for a finite element analysis problem is that it can be applied to a whole class of similar structural analysis problems with perhaps some minor educated adjustments. Moreover, the use of an automatic mesh generator which is able to produce

an optimal mesh can be a good teaching and training tool for an "inexperienced" engineer and even a structural behavior study tool for a finite element expert.

Structure of the Thesis

As a foundation for this thesis, the GAs methodology is introduced in Chapter 2 using general terminology specific to evolutionary theory such as chromosome, individual, population, regeneration, and survival of the fittest. The procedure for GAs is discussed schematically, and details of some possible genetic operators such as crossover and mutation are examined.

In Chapter 3, the fitness value (optimality criterion) used in this study is defined as the energy norm error of a finite element solution. The Zienkiewicz-Zhu or Z^2 error estimator is described as a basis for the energy error norm calculation.

In Chapter 4 the GAs technique is applied to structural analysis problems and the finite element mesh optimization problem is formulated. The terminology is adapted to suit the structural lexicon, and various types of design variables, also called optimization parameters, are defined.

Next, in Chapter 5 finite element mesh optimization using GAs methodology is applied to several structural analysis problems in both one- and two-dimensional space. The results of these implementations are discussed along with an examination of the convergence progress through iterations. In the case of one-dimensional optimization GAs with real coding are compared to GAs with binary coding and a random search

algorithm with binary coding. As GAs with real coding clearly outperform the other two methods, it is chosen for finite element optimization examples in two dimensions.

The examples considered in Chapter 5 show that one of the biggest difficulties in the application of the finite element mesh optimization involves unique mesh generation from a random set of optimization parameters. In view of this along with the fact that GAs optimization could be invaluable in future development of commercial finite element analysis software, Chapter 6 explores application of automatic mesh generation in a state of the art finite element analysis suite, ABAQUS. Here one of the examples of Chapter 5 is evaluated for the same optimality criterion values using various mesh generation techniques available in ABAQUS/CAE. Results from the two methods are compared and discussed.

Finally, in Chapter 7 a discussion of the results and some concluding remarks are offered along with suggestions for possible directions of future enhancements and applications of GAs methodology for solution-based optimization in automatic finite element mesh generators.

Chapter 2

Genetic Algorithms

GAs were developed and analyzed by John Holland [12] of the University of Michigan in the early 1970's. GAs are general-purpose explicitly parallel search algorithms that use the principle of natural selection and evolution. The basic idea is to maintain a population of species that represent candidate solutions to the current problem. The population evolves over time through recombination, mutation and competition (survival of the fittest) to produce new generations that are more "fit" with respect to an objective function or "fitness".

GAs are iterative procedures that maintain and operate on a population, $\boldsymbol{P}(t) = \{\boldsymbol{x}_1(t), \boldsymbol{x}_2(t),..., \boldsymbol{x}_n(t)\}$, of n data structures that represent candidate solutions, $\boldsymbol{x}_i(t)$ ($i = 1, ..., n$), to an objective function $f(\boldsymbol{x}(t))$.

Each structure, $\boldsymbol{x}_i(t)$ referred to as an *individual* or *chromosome*, consists of smaller units called *genes*. The most common representation of a gene is a bit (i.e. 0 or 1). Genes can also be characterized by any number system including real number

representation. Any chromosome can be represented by any gene set to a desired level of precision. Figure 2.1 shows examples of a binary chromosome of length nine (i.e. consists of 9 binary genes, 0 or 1) and a real-number chromosome of length five (i.e. consists of 5 real number genes, a_i).

binary → 101110010

real-number → $a_1\ a_2\ a_3\ a_4\ a_5$

Figure 2.1. Examples of binary and real-number *chromosomes*

During each iteration step, called a *generation*, the current population is evaluated against the fitness function and then, based on the results, a new population of chromosomes is formed using a number of operations. This procedure is shown schematically in Figure 2.2.

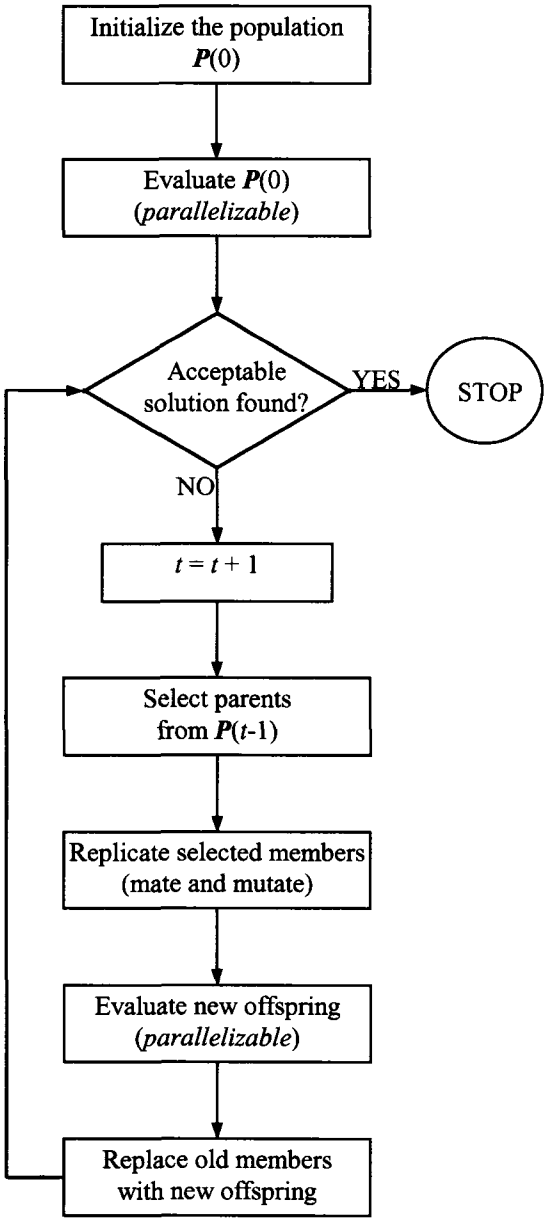


Figure 2.2. Schematic representation of GAs process

The initial population, $\boldsymbol{P}(0)$ should represent a random sample from the search space, and any available knowledge about possible solutions must be used so that the members of the population are "feasible" for the problem and include as much information about the solution as possible. Then each member (individual) of the population is evaluated and assigned a *fitness* measure as its solution to the objective function. When the whole population has been evaluated, a new population of individuals is created in two steps.

First, the individuals in the current population are selected for reproduction as *parents* based on their relative fitness, i.e. high-performing individuals have higher chances to be selected for reproduction, while poorly performing ones might not be chosen at all. Next, the selected individuals are altered to form a new set of individuals for evaluation by means of idealized *genetic operators*. The primary genetic search operator is the *crossover*, which combines the features of two parent structures to form two similar *offspring*. There are many different crossover types; the simplest exchanges corresponding portions of a string representation of the parents, which can be implemented by choosing a point in the string at random, called crossover point, and swapping the strings to the right of the crossover point (see Figure 2.3).

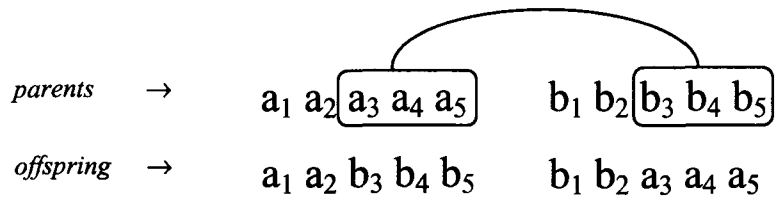


Figure 2.3. Example of one-point *crossover* operation

Another example is two-point crossover operation, in which two points in the string are chosen at random and then the strings in parent structures are swapped between these two points (see Figure 2.4).

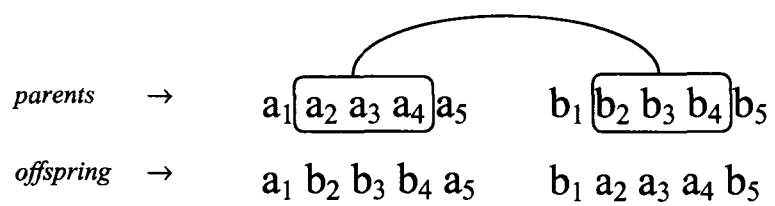


Figure 2.4. Example of two-point *crossover* operation

The crossover usually draws only on information present in the individuals. Some information might not be selected in the random initialization and some might be lost during the selection of previous iterations. In that case, crossover cannot produce new individuals that contain this information, and another genetic operator, *mutation* (see Figure 2.5), provides the way to introduce new information into the population. The mutation operator alters some elements in the string of a selected individual (parent), which ensures that all points in the search space are reachable.

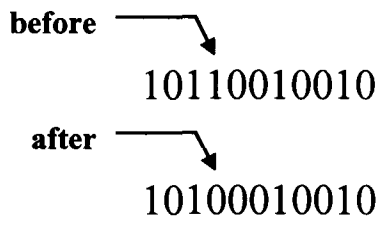


Figure 2.5. Example of *mutation* operation

The resulting offspring are then evaluated and inserted in the population based on their performance (fitness), replacing older members with lower performance to the objective function.

The power of GAs is in their ability to utilize information about the usefulness of a very large number of structural patterns. This helps them to concentrate on the regions that contain structures with consistently high performance. Nonetheless, the information, widely distributed over the search space, helps to avoid stagnation at local optima.

In order for GAs to reach an optimal solution the population needs to evolve through many generations (iterations). In this process the objective function is evaluated for every individual in the initial population and all offspring in each generation. This requires solutions to problems for great number of individuals resulting in large computational costs. However, instead of sequential computations the problems in each generation can be solved in parallel considerably reducing the total computational time.

In cases of optimization problems where the objective function, $f(\boldsymbol{x}(t))$, can implicitly be related to a chromosome, $\boldsymbol{x}_i(t)$, the number of evaluations can be reduced by the use of hybrid methods, i.e. GAs followed by conventional optimization techniques. First, GAs optimization is implemented for several iterations until enough members in the population start to accumulate around points of extrema including the global minimum / maximum point. Once these conditions are satisfied other optimization techniques (e.g. gradient or Newton methods) are invoked to insure faster convergence to the local extrema and the global minimum / maximum is obtained.

To summarize, the main advantages of GAs over conventional optimization techniques are:

(a) no gradients (which are computationally very expensive) are necessary;

(b) multiple extrema are handled easily even for problems with complex objective functions properties of which are not known and there is no information on their behavior around the global minimum / maximum;

(c) Genetic Algorithms are implicitly and explicitly parallel, which leads to tremendous reduction in computation time on parallel processors [12, 17];

(d) GAs are never stuck at local optima, always improve the initial solution, and, with little overhead, perform at least as well as any known optimizers for the same problem [12].

The implementation of GAs involves choice of various parameters and factors which have the ability to appreciably affect GAs convergence and final computational costs. One of the most obvious parameters to influence the progress of GAs optimization

is the population size maintained through all generations. It is the number of active individuals in a population denoted as N_{Active}. Another very important parameter is the number of new individuals, N_{Adapt}, created in every generation and inserted into the population for competition. Both parameters can significantly increase the number of calculations required in each generation and possibly change the convergence rate of an optimization problem.

Chapter 3

Error Estimation in Finite Element Meshes

Error estimation is an important issue in finite element discretization. Users of finite element methods need to know the accuracy of their results before accepting them as correct in practice. One of the most common ways of predicting the accuracy has been based on the "experience" of a user and rules on permissible element formulation, shapes or sizes. Another technique involves computing stresses at a node from adjacent elements. However, these methods can only identify the regions of questionable accuracy and do not produce quantitative measure for error. Alternatively, a study of mesh refinement is one of the most obvious procedures of assessing the accuracy. Unfortunately, this approach is usually too expensive to use in frequent applications. As a result decisions on applicability of the solution of the finite element approximation are often made on a more qualitative basis.

Nonetheless, recently introduced error estimates are capable of estimating the discretization error at relatively small cost. This allows adaptive processes to be used to refine approximation sufficiently to achieve desired accuracy.

Outline of "Z^2" Error Estimator

The error estimator proposed by Zienkiewicz and Zhu [3], called Z^2 because of the initials of the authors, is computed by using a recovered higher order finite element solution and is applicable to elements of any order. A general analysis and the mathematical proof of the convergence of the Z^2 error estimator are provided in Reference [5]. The method is described briefly below.

The problem of linear elasticity (see Figure 3.1) is characterized by the following equation:

$$\mathbf{S}^T\boldsymbol{\sigma} + \mathbf{b} = 0 \tag{3.1}$$

with

$$\boldsymbol{\sigma} = \mathbf{DSu} \qquad \text{in domain } \Omega \tag{3.2}$$

and appropriate boundary conditions

$$\mathbf{n}\boldsymbol{\sigma} = \mathbf{t}_0 \quad \text{on } \Gamma_t \quad \text{and} \quad \mathbf{u} = \mathbf{u}_0 \quad \text{on } \Gamma_u \tag{3.3}$$

where $\mathbf{u}$ is the vector of generalized displacement fields, which is the primary variable; $\boldsymbol{\sigma}$ is the generalized stress; matrix $\mathbf{S}$ is a first order strain differential operator; $\mathbf{D}$ is the stress-strain matrix; $\mathbf{b}$ is the vector of body forces; $\mathbf{t}_0$ is the vector of tractions on the boundary Γ_t; and $\mathbf{u}_0$ is the vector of displacements on the boundary Γ_u.

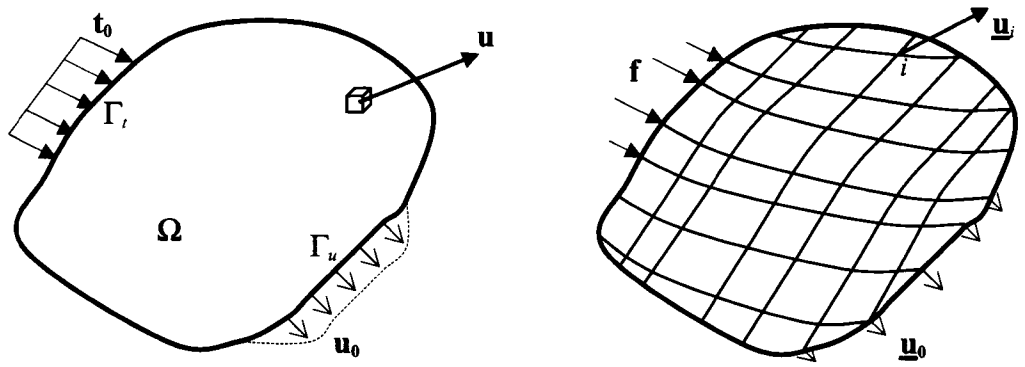


Figure 3.1. Problem of linear elasticity of a structure and its finite element model

The finite element method [19] approximates the displacement field, **u**, at any point within an element as a column vector, $\mathbf{u}_h$

$$\mathbf{u} \approx \mathbf{u}_h = \mathbf{N}\underline{\mathbf{u}}^e \tag{3.4}$$

where **N** are shape or interpolation functions and $\underline{\mathbf{u}}^e$ is the vector of nodal displacements for a particular element, *e*. This representation leads to an algebraic set of equilibrium equations for an element

$$\mathbf{K}^e\underline{\mathbf{u}}^e = \mathbf{f}^e \tag{3.5}$$

where $\mathbf{K}^e = \int_\Omega \mathbf{B}^T\mathbf{D}\mathbf{B}\, d\Omega^e$ ($\mathbf{B} = \mathbf{S}\mathbf{N}$) is the element stiffness matrix. After all element equations are assembled to structure size, equilibrium equations for the entire structure are obtained

$$\mathbf{K}\underline{\mathbf{u}} = \mathbf{f} \tag{3.6}$$

Here, $\underline{\mathbf{u}}$ is the global vector of nodal displacements or degrees of freedom (DOF), $\mathbf{K}=\Sigma_e\mathbf{K}^e$ is the global stiffness matrix, and $\mathbf{f} = \Sigma_e\mathbf{f}^e$ is the vector of global nodal external

forces incorporating the body forces and the boundary conditions. Note that the summation is done according to standard finite element assembly process over all elements in the structure. The global stiffness matrix, **K**, and the vector of nodal forces, **f**, are functions of node positions in the structure, i.e. the finite element mesh.

The approximate displacement fields, $\mathbf{u}_h$ and the approximate stresses, $\boldsymbol{\sigma}_h = \mathbf{DB}\underline{\mathbf{u}}$ contain local errors given by

$$\mathbf{e}_u = \mathbf{u} - \mathbf{u}_h \tag{3.7}$$

$$\mathbf{e}_\sigma = \boldsymbol{\sigma} - \boldsymbol{\sigma}_h \tag{3.8}$$

where $\mathbf{u}$ and $\boldsymbol{\sigma}$ denote the exact solutions.

Error is defined in the energy norm such that

$$\|\mathbf{u}\|^2 = \int_\Omega \boldsymbol{\sigma}^T \mathbf{D}^{-1} \boldsymbol{\sigma} \, d\Omega \tag{3.9}$$

and

$$\|\mathbf{e}\|^2 = \int_\Omega (\boldsymbol{\sigma} - \boldsymbol{\sigma}_h)^T \mathbf{D}^{-1} (\boldsymbol{\sigma} - \boldsymbol{\sigma}_h) \, d\Omega \tag{3.10}$$

The error estimator proposed by Zienkiewicz and Zhu uses the observation that the stresses $\boldsymbol{\sigma}$ can be represented more accurately by a smooth field $\boldsymbol{\sigma}^*$ interpolated by continuous shape functions $\underline{\mathbf{N}}$, i.e.

$$\boldsymbol{\sigma}^* = \underline{\mathbf{N}} \, \underline{\boldsymbol{\sigma}}^* \tag{3.11}$$

Here, the parameters $\underline{\boldsymbol{\sigma}}^*$ are obtained by a projection of $\boldsymbol{\sigma}_h$ requiring that

$$\int_\Omega \mathbf{P}(\boldsymbol{\sigma}^* - \boldsymbol{\sigma}_h) \, d\Omega = 0 \tag{3.12}$$

The interpolation functions $\underline{\mathbf{N}}$ are best chosen to be of the same order as the shape functions $\mathbf{N}$ and frequently, $\underline{\mathbf{N}} = \mathbf{N}$ is taken. The projection operator $\mathbf{P}$ can be used in

various forms. As observed by Brauchli and Oden [20] and others [3, 21], the conjugate form of the projection operator

$$\mathbf{P} = \underline{\mathbf{N}}^{\mathrm{T}} \tag{3.13}$$

is effective, but simple averaging with $P_j = \delta_j$ (where δ_j is the Dirac delta function) or the more complex projection

$$\mathbf{P} = (\mathbf{D}^{-1}\underline{\mathbf{N}})^{\mathrm{T}} \tag{3.14}$$

can be efficiently used.

Now replacing σ by σ^* in equation (3.10), and error estimator ε is obtained as

$$\varepsilon^2 = \int_\Omega (\sigma^* - \sigma_h)^{\mathrm{T}}\mathbf{D}^{-1}(\sigma^* - \sigma_h)\, d\Omega \approx \|\mathbf{e}\|^2 \tag{3.15}$$

The error estimator gives quite accurate estimates of discretization error not only globally but also for individual elements. Clearly the error defined above is the result of a particular discretization of the continuum and depends on the mesh used for discretization. Consequently some meshes will produce smaller error than others. The goal of mesh optimization is to minimize the global error. Thus in finite element mesh optimization, the objective function can be chosen to be the global percentage accuracy η_s in the energy norm. The percentage error is defined as

$$\text{(actual)} \quad \eta_s = \frac{\|\mathbf{e}\|}{\|\mathbf{u}\|} \approx \frac{\varepsilon}{\|\mathbf{u}\|} = \eta^0 \quad \text{(predicted)} \tag{3.16}$$

Noting that

$$\|\mathbf{e}\|_\Omega^2 = \sum_{e=1}^{M} \|\mathbf{e}\|_{\Omega e}^2 \tag{3.17}$$

and

$$\varepsilon^2 = \sum_{e=1}^{M} \varepsilon_e^2 \tag{3.18}$$

it is possible to impose a uniform error distribution for all elements of the mesh, and thus obtaining for each element

$$\|\mathbf{e}\|_e \approx \varepsilon_e = \frac{\eta_s \|\mathbf{u}_h\|}{M^{1/2}}, \tag{3.19}$$

where M is the total number of existing elements.

Chapter 4

Formulation of Mesh Optimization Problem and General Approach

The finite element mesh optimization of a structure with N elements is comprised of finding the vector of design parameters, $\boldsymbol{x}$, which maps into the nodal positions in the structure (i.e. finite element mesh), such that the global error in the finite element solution (objective function), $J(\boldsymbol{x})$, is minimized.

A vector of parameters, $\boldsymbol{x}$, initially randomly chosen or obtained by means of genetic operators during generational iterations, is mapped into the nodal positions and a FE mesh is generated according to element connectivity. Then appropriate loading, boundary and initial conditions are applied at nodal degrees of freedom and a FE problem is formulated. In case of static stress-displacement analysis the FE formulation (Equation 3.6) follows from the statement of static equilibrium which states that the internal forces, I, exerted on the nodes resulting from the element stresses and the external forces, P, acting at every node must balance (Figure 4.1).

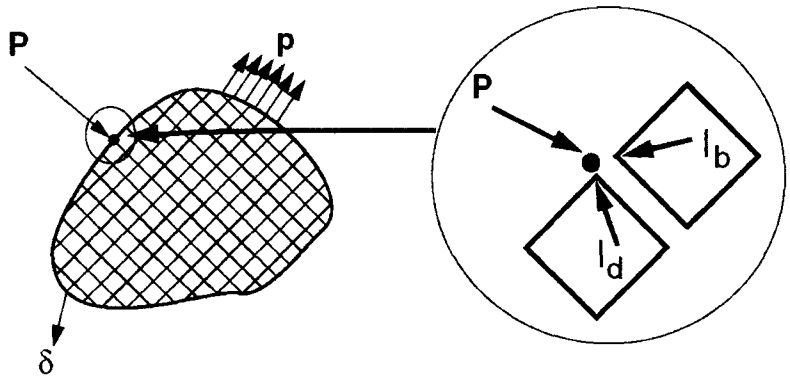


Figure 4.1. Static equilibrium in finite element problem discretization

Then, the nodal displacements, **u**, and element stresses, σ, are found by finite element structural analysis (see flowchart in Figure 4.2) in every iteration of GAs using the following equations:

$$\mathbf{Ku} = \mathbf{f} \tag{4.1}$$

$$\sigma = \mathbf{Tu} \tag{4.2}$$

where **T** is a transformation matrix relating nodal displacements to element stresses. The global stiffness matrix of the structure, **K**, and the external nodal force vector, **f**, are functions of nodal positions in the structure (i.e. the finite element mesh being optimized) which are in turn functions of the vector of decision variables, $\boldsymbol{x}$:

$$\mathbf{K} = \mathbf{K}(\boldsymbol{x}) \quad \text{and} \quad \mathbf{f} = \mathbf{f}(\boldsymbol{x}) \tag{4.3}$$

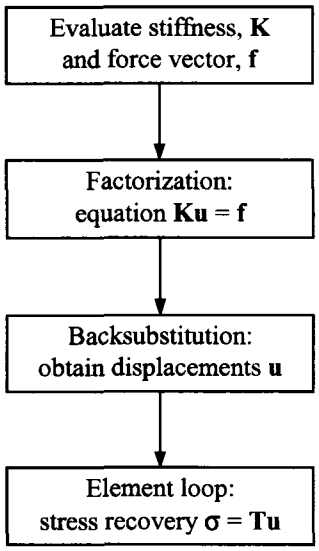


Figure 4.2. FE analysis flowchart

Next, the global error in the finite element solution is defined as the energy norm of either nodal displacements or element stresses. This can be determined using finite element solution with *M* elements (referred to as "fine mesh" solution) and comparing it to the solution of a finite element mesh being optimized with *N* elements ("coarse mesh"), where *M*>>*N*. This "fine mesh" can be obtained by refining the mesh successfully until no discernible change is observed in the solution and thus presumed to be the "exact solution". This actually has a potential to reduce the computational costs since the "fine mesh" solution is obtained only once in the beginning of the optimization whereas an error estimator needs to be applied for every "coarse mesh" problem in the GAs procedure. However, in cases when the "exact solution" is not possible to obtain

with confidence, the Z^2 error estimator (described in Chapter 3 above) needs to be used for objective function evaluations.

To establish a benchmark to test effectiveness of GAs in mesh optimization, a "fine mesh" solution will be used in this study as the "exact solution" for error calculations (i.e. the objective function evaluation) in the nodal displacements. Thus, the objective function, J, can be defined as the relative error

$$J(\boldsymbol{x}) = \left\| \mathbf{u}^{M} - \mathbf{u}^{N \to M}(\boldsymbol{x}) \right\| \Big/ \left\| \mathbf{u}^{M} \right\| \tag{4.4}$$

where $\mathbf{u}^{M}$ is displacement vector from the "fine mesh" model evaluated at the fine mesh nodes ("exact" solution), and $\mathbf{u}^{N \to M}(\boldsymbol{x})$ is the displacement vector from the finite element mesh model being optimized ("coarse mesh" model), interpolated to the nodes in the "fine mesh" model. Note that nothing is known about the behavior of the objective function $J(\boldsymbol{x})$ in the optimization space, the mathematical formulation cannot be explicitly written in terms of optimization variables, and thus the gradients are not possible to calculate. In addition, there may be numerous constraints on the vector of parameters, $\boldsymbol{x}$, due to the structural geometry, loading and boundary conditions. Hence genetic optimization algorithms are probably the best tool to solve this optimization problem.

This process of evaluating the value of objective function, $J(\boldsymbol{x})$, from a vector of design parameters, $\boldsymbol{x}$, is schematically shown below in Figure 4.3.

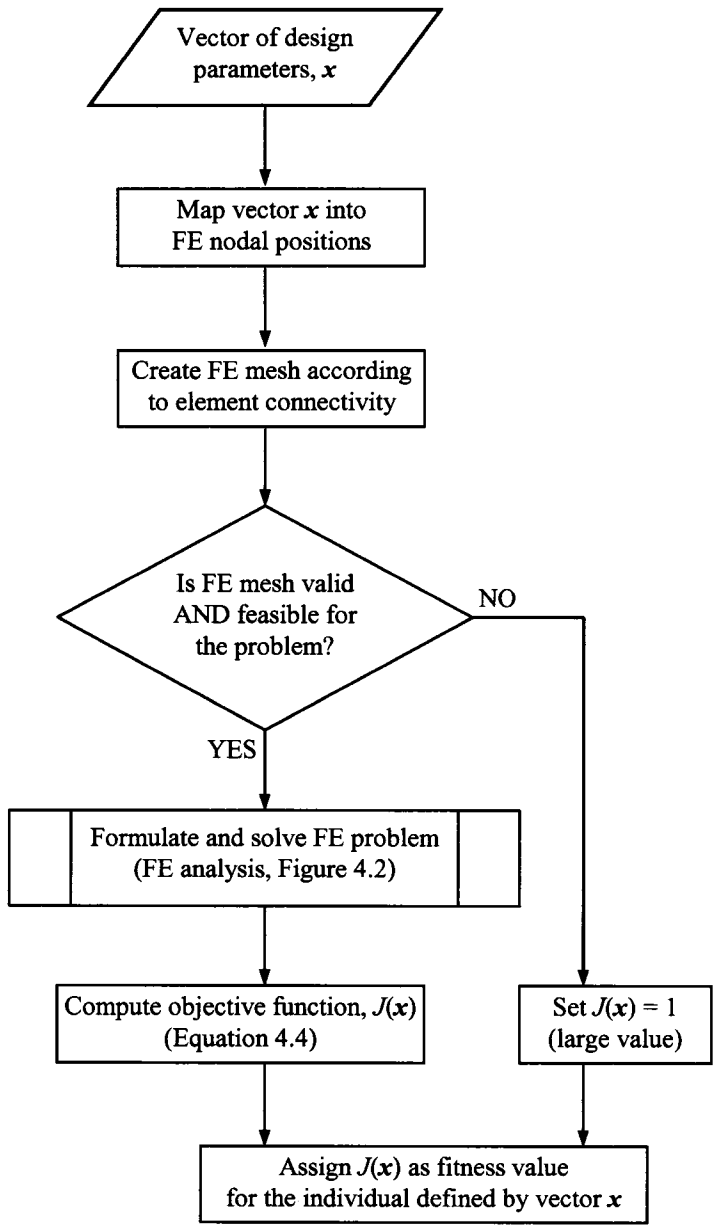


Figure 4.3. Evaluation of objective function and fitness assignment

The procedure by which GAs can be used to solve the mesh optimization problem is the following (schematically shown in Figure 4.4 below). Genetic Algorithms operate on a population of vectors of design variables, referred to as *optimization parameters*, as described in Chapter 2 above. Each set of optimization parameters, referred to as an *individual* or *chromosome* in the population, defines nodal positions in the structure which in its turn defines a finite element mesh. The stiffness matrix, **K**, and force vector, **f**, are calculated for that particular mesh and the nodal displacements and element stresses are then calculated according to above equations. Then, obtained displacements or stresses for each individual are used to compute the error, as discussed above, which is assigned as the fitness of that particular individual. When the whole population has been evaluated, new offspring of the population is formed (by means of crossover and/or mutation) and new individuals are created which define new meshes with their stiffness matrices and force vectors. Finally, optimality criterion is calculated for these new individuals as they are inserted into the population replacing members with higher fitness values, and the process is repeated until the desired accuracy has been achieved or iteration limit has been reached.

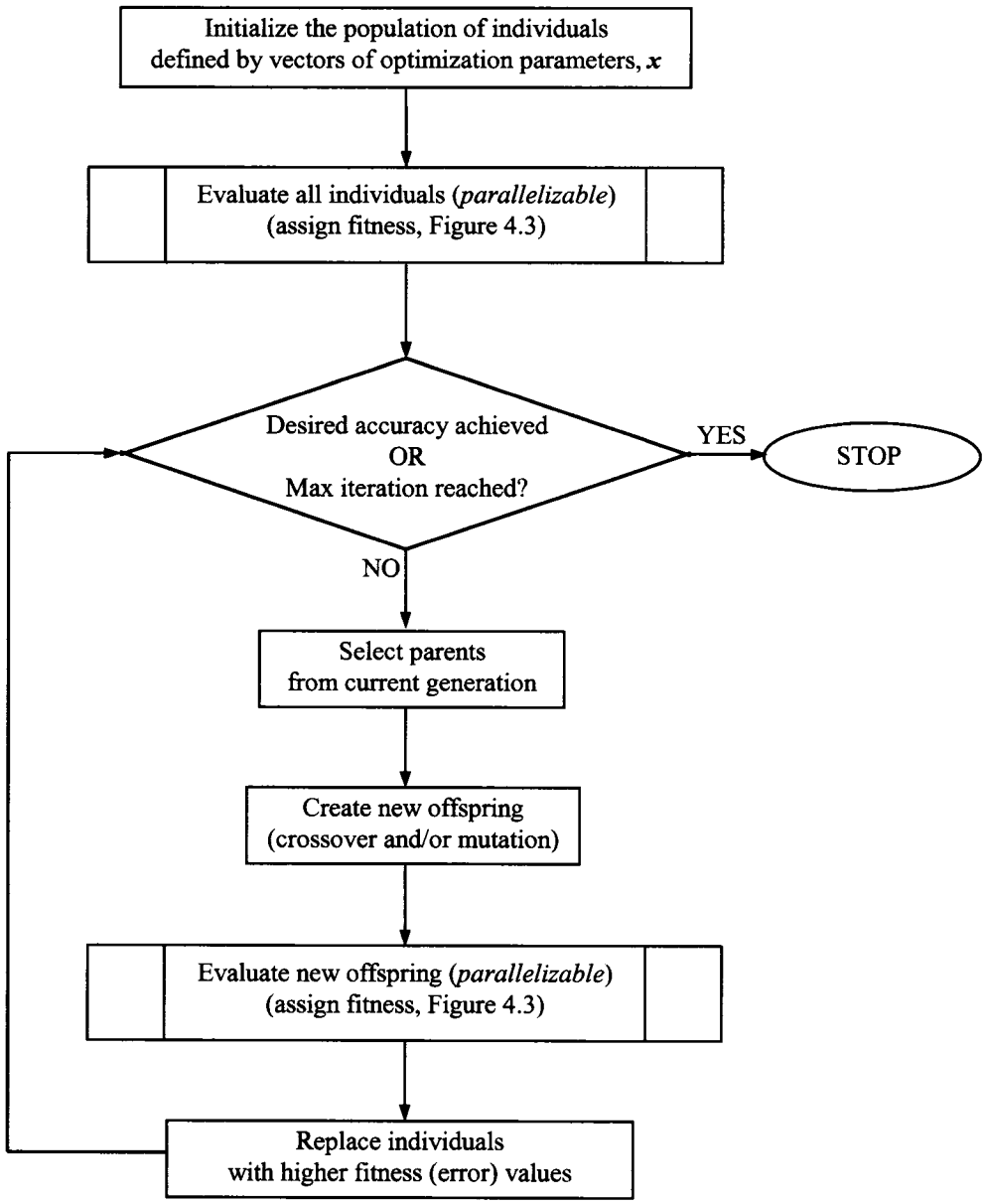


Figure 4.4. Schematic representation of FE mesh optimization using GAs

Each chromosome (set of decision variables) represents a finite element mesh and that correspondence must be unique. In other words, a chromosome has to define a unique mesh and thus must have only one measure of fitness, though the same mesh can be represented by two completely different chromosomes. There are infinitely many ways of defining a mesh from a given set of parameters, but once the method is chosen, the representation has to be unique.

A mesh is defined by the positions of its nodes and their connectivity (elements). Here, since the type of refinement used in this work is *r*-refinement, it is clear that the number and numbering of the nodes do not change regardless of their positions. One possible method of generating a mesh is to have each parameter of the chromosome define an x, y or z, coordinate of a particular node in the mesh. As an example let us consider a square plate with four quadrilateral elements and nine nodes (Figure 4.5). Since it is required to have a node in each corner, the only nodes free to change their positions are nodes 2, 4, 5, 6, and 8. Moreover, nodes 2, 4, 6, and 8 can move only along the boundary they are located on. Thus, the chromosome for the above described representation in this example consists of six parameters, $(x_2, y_4, x_5, y_5, y_6, x_8)$.

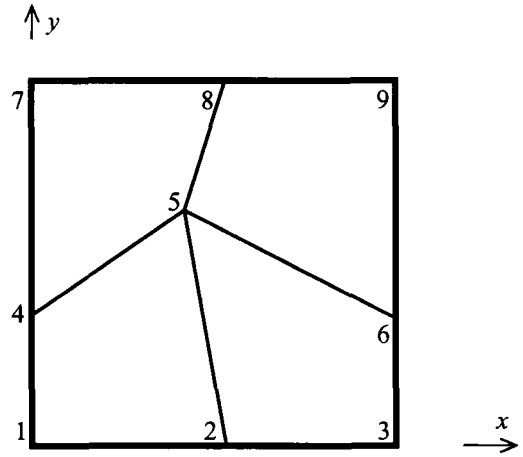


Figure 4.5. Square plate with four elements

This is obviously a physical representation of a mesh by a set of parameters and perhaps the most natural one, but as will be seen in the following chapters it is not the best choice because the number of parameters increases rapidly as the problem size increases.

Other methods of representation of a mesh by a set of parameters include functional correspondence between a chromosome and a mesh. Instead of directly assigning each parameter to a node coordinate, the parameters are used to represent a functional form which defines nodal positions. For simplicity let us consider a one-dimensional structure (beam or truss) with three elements and four nodes (Figure 4.6). If the previously described "natural" chromosome-mesh correspondence is applied, then the chromosome would consist of two parameters, (x_2, x_3), and their number would increase with the number of elements in the FE approximation.

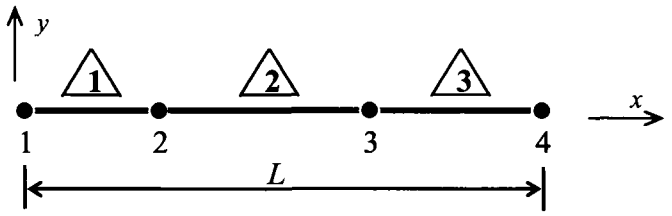


Figure 4.6. Beam (truss) with three elements

To show an example of a functional correspondence between a chromosome and a mesh denote L_i as the length of i^{th} element ($i = 1, 2, 3$). Now the lengths can be defined as follows. First, temporary lengths are computed

$$L_i^* = |A \cos(\omega i) + B \sin(\omega i)|, \quad (i = 1, 2, 3) \tag{4.5}$$

and then L_i^* is scaled

$$L_i = L_i^* / \Sigma_i L_i^* \tag{4.6}$$

so that

$$\Sigma_i L_i = L \tag{4.7}$$

where L is the total length of the beam (truss).

Here, the chromosome consists of three parameters, (A, B, ω), which is more than in "natural" representation case (two parameters, $x_2$and x_3). The number of parameters could have been even bigger if another functional representation was chosen, however it will not increase no matter how many elements are chosen in the finite element approximation.

Additional examples of possible functional correspondence between a chromosome in GAs and a FE mesh are discussed in the next chapter. The discussion includes their generality and limitations along with advantages and disadvantages of using "natural" versus functional representations.

After a chromosome is transformed into node positions, the nodes are connected according to FEM rules to create elements. The elements are then analyzed for feasibility and validity. If an element has crossed sides or element numbering is inconsistent with FEM definitions, then the entire mesh and thus the chromosome that produced it, is discarded from the population. This rule does not need to be applied in some cases, e.g. in case a mesh is constructed by element lengths as described in the above example (Figure 4.6), as there could not be any inconsistency or overlapping of elements. Additional rules can be applied to eliminate a mesh (chromosome) from consideration before computing its fitness (the error in the finite element solution). These rules can include, but are not limited to elements with severe shape distortion or large aspect ratio, tests of zero-energy deformation modes, lack of invariance, absence of rigid-body motion capability, ill-conditioning of the stiffness matrix, and any prior knowledge about the type of elements, their arrangement and the behavior (stiff regions or supports, etc.) of the structure being analyzed.

Chapter 5

Implementation of GAs Scheme and Results

In this chapter several examples in one- and two-dimensional space are considered for finite element mesh optimization using GAs techniques. A "natural" and several functional representations of the FE mesh by a random set of optimization parameters are examined along with their advantages and disadvantages in the implementation of GAs optimization. Results from all these cases are compared and discussed.

Finite Element Mesh Optimization in 1D [1]

To create basis for optimization methodology discussed in above chapters and to establish the applicability of these techniques, a simple one-dimensional problem is considered. The optimization is implemented with GAs with real coding, GAs with binary coding and a random search algorithm. In all cases optimization parameters are defined to represent locations of the FE nodes directly, referred here as "natural" representation.

Example 1. A cantilever beam

Consider the example of a cantilever beam of length 10 m, with a distributed load (see Figure 5.1). The Young's modulus and cross-sectional moment of inertia of the beam are 210 GPa and 4×10^{-4} m^4, respectively.

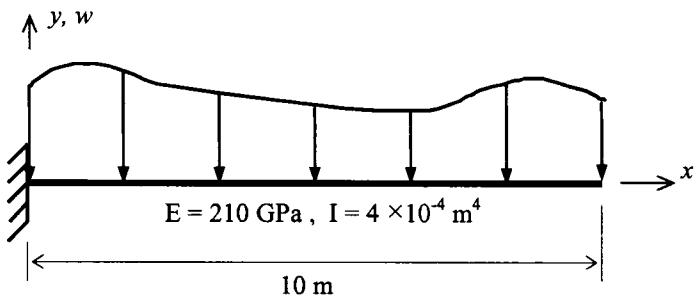


Figure 5.1. Cantilever beam

The beam is modeled with standard four-DOF beam elements. A 129-element model with uniform mesh (fine mesh) is used as a "true" solution, and a 3-element model is used as a "coarse mesh" model (Figure 5.2).

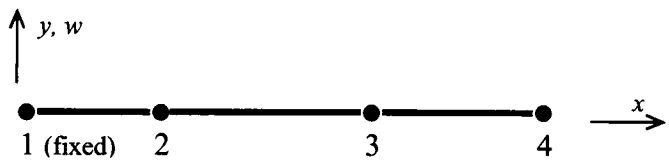


Figure 5.2. Finite element model of the beam

The finite element model to be optimized is considered under a linear load (type 1) and step load (type 2), (see Figure 5.3).

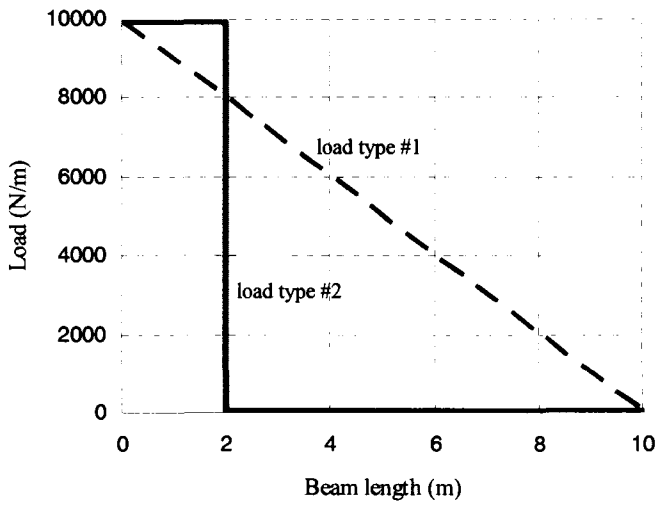


Figure 5.3. Load types

To find the 'best' possible mesh for the beam problem, two optimization parameters are chosen: the positions of internal nodes 2 and 3 along the beam axis. Boundary nodes (i.e. nodes 1 and 4) remain fixed at all times. The objective function J is defined by (4.4), where $\mathbf{u}^M$ is (258×1) displacement vector from the "fine mesh" model evaluated at the fine mesh nodes ("true solution") and $\mathbf{u}^{N\to M}(\boldsymbol{x})$ is a (258×1) displacements vector interpolated from the "coarse mesh" model, again at the fine mesh nodes. Note that the "fine mesh" solution, adopted for the purposes of illustration in this study, needs to be calculated only once. The optimization is performed using

generational GAs with roulette-based selection and elitism [15]. Both "coarse mesh" solution and GAs optimization are implemented as a complete MATLAB code. As the behavior of GAs is essentially statistical, the results are averaged for over 100 runs to ensure statistically valid behavior.

The progress of convergence for both types of loading are shown in Figure 5.4 and Figure 5.5 below for a random search with binary coding, GAs with binary coding, and GAs with real coding. A random search algorithm examines candidate solutions by randomly selecting individuals from the optimization domain without using any feedback from already analyzed individuals. Here, the binary coding is referred to optimization algorithms where the finite element mesh nodes can only be placed in certain discrete locations thus allowing for a binary representation in chromosomes. In contrast, GAs with real coding use real number representation for the genes and the nodes now can be placed anywhere (limited to a computer's OS precision) within the structure. The objective function is evaluated for all individuals in the population at every generation and the error of the fittest individual (referred as current minimum) is plotted though iterations.

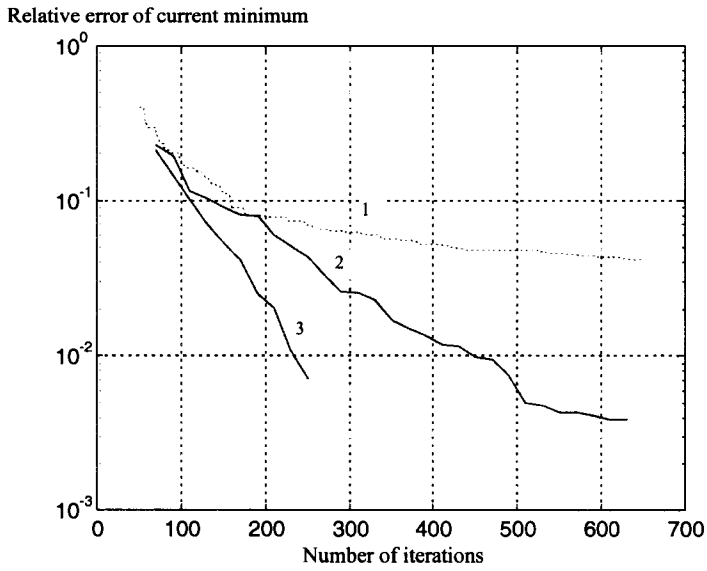


Figure 5.4. Average behavior for the beam with a linearly distributed load (type 1): random search (1), GAs with binary coding (2), and GAs with real coding (3)

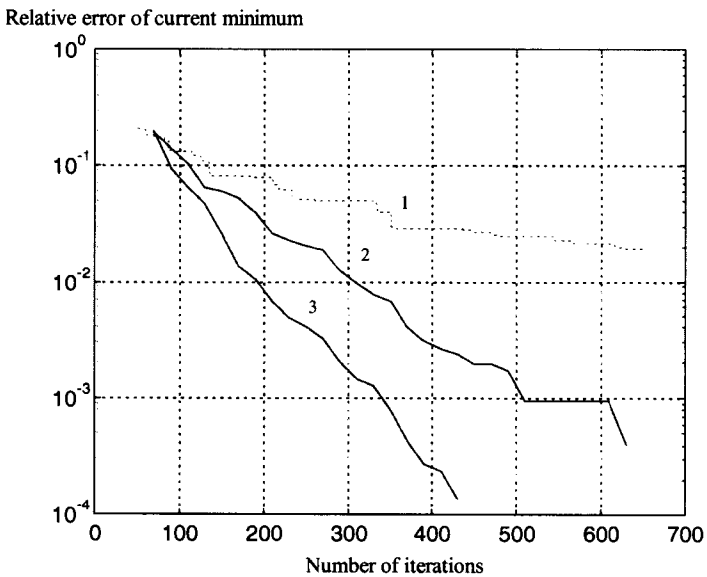


Figure 5.5. Average behavior for the beam with a step load (type 2): random search (1), GAs with binary coding (2), and GAs with real coding (3)

As it can be clearly seen from the plots, GAs with real-number coding outperforms GAs with binary coding, which in turn outperforms random search algorithm. As expected the random search algorithm has very slow convergence and will take many multiple times the number of iterations to reach minimum error values obtained by GAs, making it practically unreachable. On the other hand, GAs with binary coding, clearly showing much faster convergence rate, can reach either a goal error value or possible global minimum in a reasonable number of iterations. However they too fall behind GAs with real coding. For example, the plots show that in order to obtain an optimal mesh with an error value of 1% GAs with binary coding require over 1.5 times more iterations than GAs with real coding. Note that the major computational cost in these optimizations is in the evaluation of the objective function which is the same for all these algorithms. Recognizing that all the optimization algorithms considered here are fully parallelizable, further improvements in computational time can be achieved using conventional GAs suites with parallel realization.

Optimal meshes and curves of equal optimality criterion, $J(\boldsymbol{x})$, for the considered loads are plotted in Figure 5.6 and Figure 5.7, respectively.

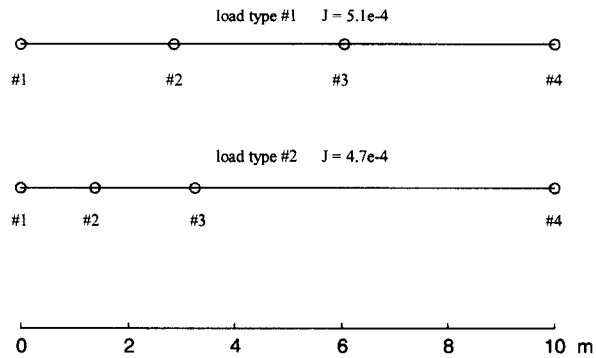


Figure 5.6. Optimal meshes: for linearly distributed load (#1) and step load (#2)

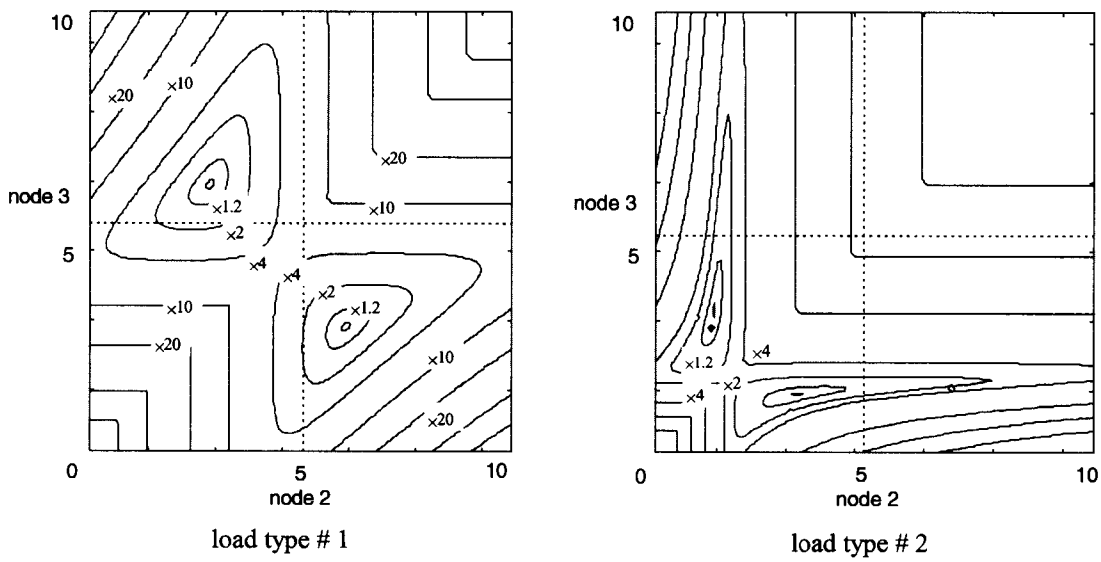


Figure 5.7. Curves of equal error (scaled wrt minimum error) for load types 1 and 2

For comparison the problems with both loading types are solved using ABAQUS, a commercial FE package [35-40]. The beam is modeled with 10 two-node cubic beam elements in a plane (B23). After applying appropriate boundary conditions and loads, both problems are solved for static deformation. Tick mark contour plots of displacement in y-direction, U_2 , are shown in Figure 5.8.

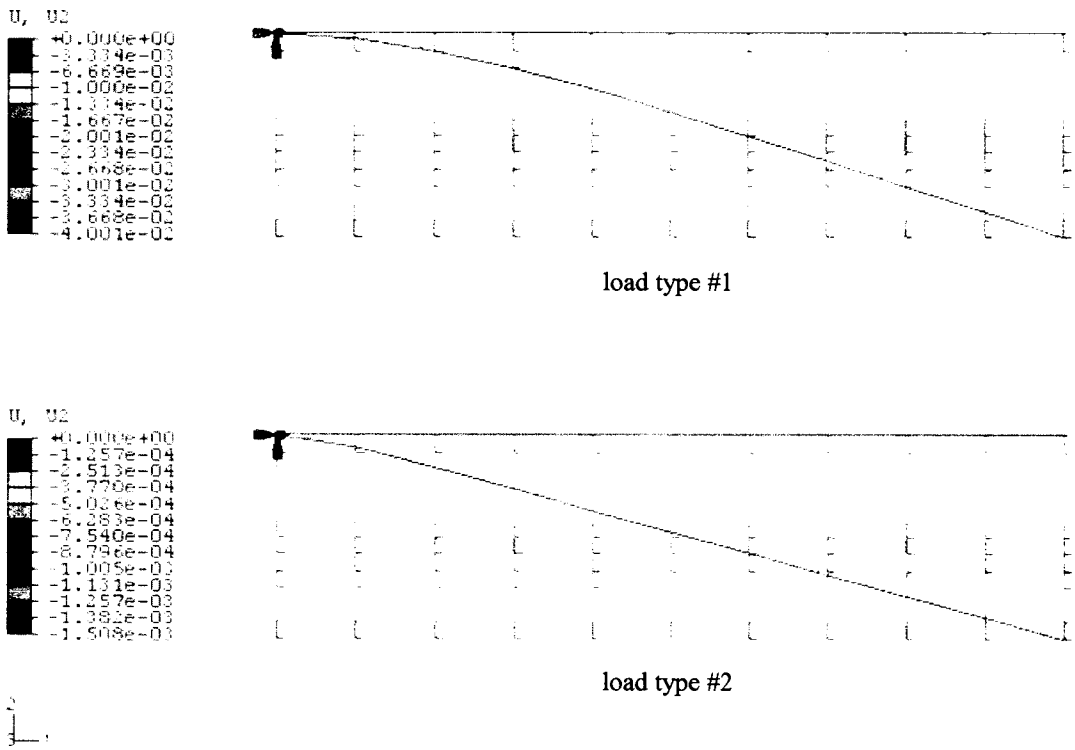


Figure 5.8. Tick mark contour of displacement y-component for linearly distributed load (#1) and step load (#2)

As the above plots show, the areas closer to the beam free end undergoes almost no deformation. These are the regions where the deformed beam is almost straight and it

is more pronounced in the model with step loading (load type #2). Hence it can be seen from Figure 5.6 that the nodes of the optimal meshes obtained by the GAs fall closer to the high displacement gradient locations (areas with larger curvature from ABAQUS analyses), which in this case are the fixed boundary condition and the point of abrupt load change (step load).

Finite Element Mesh Optimization in 2D

Mesh optimization in two dimensions follows exactly the same procedure as in the one-dimensional case, though the implementation involves more complicated schemes and procedures. The first difficulty is the initialization, i.e. obtaining an initial population at random. The main challenge is the creation of a valid mesh from a random set of parameters, preserving the boundary and making sure that no element has edge intercrossing (invalid shape) and node numbering is consistent with FE formulation. The larger the size of a mechanical structure, the more elements are required to achieve reasonable accuracy and the harder it is to come up with a feasible mesh from parameters chosen at random. This, however, can be resolved by partitioning (subdividing) the structure into smaller, simpler regions and considering mesh generation in each region.

Second, as the size of the problem increases, that is as the number of 2-D elements grows, the number of parameters used in optimization grows even more rapidly than in the one-dimensional case. Specifically the size of x grows quadratically by the number of elements as opposed linearly in the 1-D case. Thus, the dimension of the optimization space grows rapidly resulting in significantly larger computational costs. To avoid this problem, other optimization parameters, different than the ones which directly describe nodal positions, were considered. As explained in Chapter 4 the goal is to create some functional correspondence between these parameters and real nodal positions so that the mesh is described with much fewer parameters. Though this has an advantage of

significantly reducing the dimension of the optimization space, it has also a shortcoming that now not all node positions can be chosen with complete randomness. This eliminates some possible meshes which can be candidates for being the "best" mesh. But even in that case it is worth looking at those new functional representations of parameters because it is evident (as it also can be seen from one-dimensional case) that there are many meshes with errors which are very or even negligibly close to the error of the "best" mesh.

Example 2. A thin square plate

As an example in two dimensions, consider a square thin plate of size 10×10 m, with a concentrated load at one of the corners (see Figure 5.9). One edge of the plate is fixed and a concentrated load of magnitude 10 kN is applied at a corner of the plate at a 45° angle, as shown below. The thickness of the plate is 4×10^{-4} m. The Young's modulus and Poisson ratio are 210 GPa and 0.3, respectively.

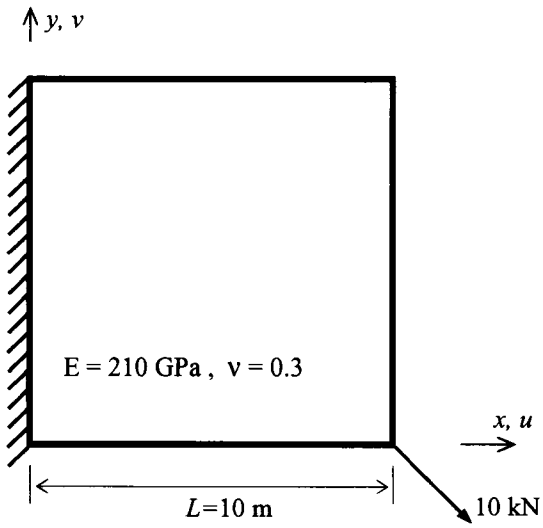


Figure 5.9. Square plate under plane stress conditions

First, the problem is solved using ABAQUS. The plate is modeled with 2500 quadratic plane stress elements with reduced integration (CPS8R). After applying appropriate boundary conditions and loads, the problem is solved for static deformation and stresses are obtained. Banded and line contour plots of in-plane maximum principal stress, S_{max} , are shown in Figure 5.10 and Figure 5.11, respectively.

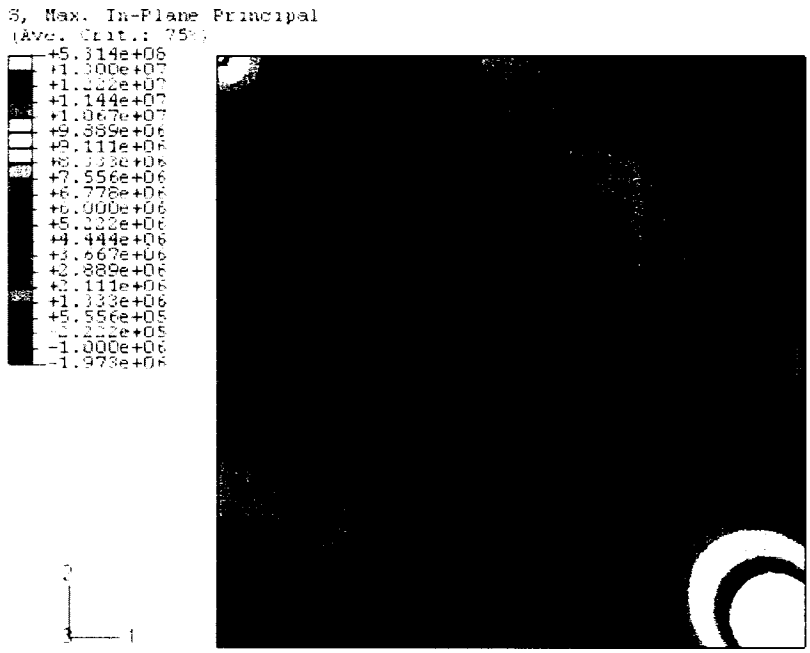


Figure 5.10. In-plane maximum principal stress for plate problem: banded contour

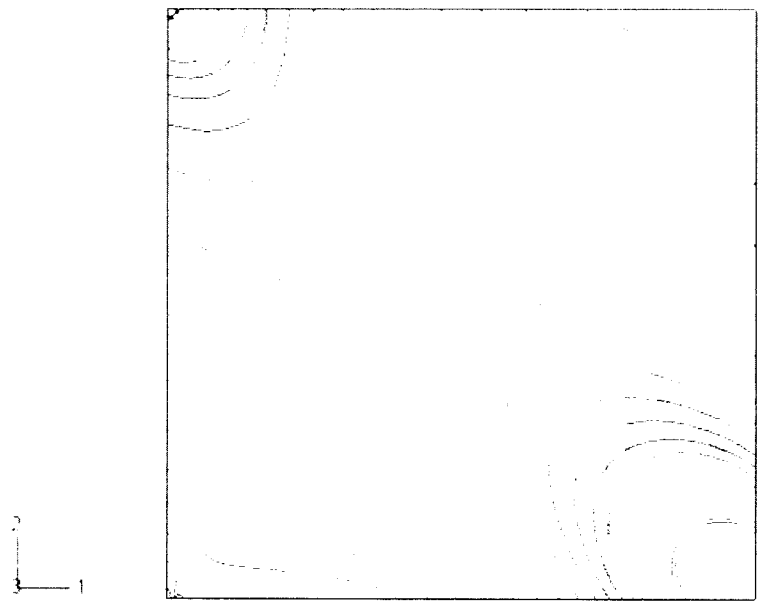


Figure 5.11. In-plane maximum principal stress for plate problem: line contour

Then, the same approach as in the one-dimensional case is applied to this two-dimensional plate mesh optimization problem. The plate is modeled with $(n\text{-}1) \times (n\text{-}1)$ quadrilateral plane stress elements (see Figure 5.12), where n is number of nodes along one side of the plate which is for simplicity assumed to be the same along all sides. The number of optimization parameters in this case is $(2n^2\text{-}4n)$ and the rate of FE problem size increase is quadratic in n.

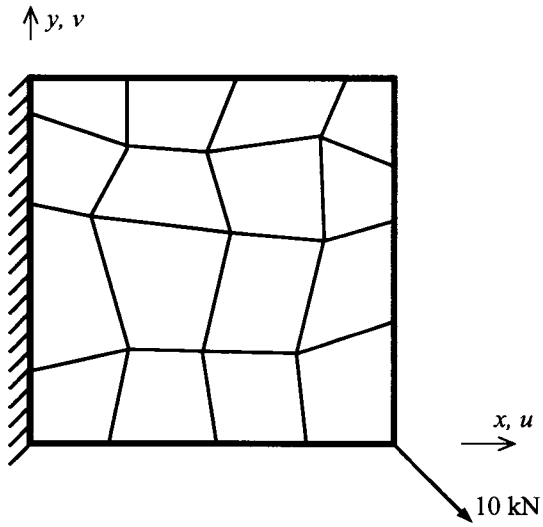


Figure 5.12. Randomly generated finite element mesh of the plate

Several mesh models ("coarse mesh") were considered for optimization: a 9-element model (4 nodes per side) with 16 optimization parameters; a 16-element model (5 nodes per side) with 30 optimization parameters; a 25-element model (6 nodes per side) with 48 optimization parameters; and a 36-element model (7 nodes per side) with 70

optimization parameters. The optimization parameters in these examples are x and y coordinates for the nodes located inside the domain, and either x or y coordinates for the nodes located on the boundary of the domain. In all cases, a 196-element model with a uniform mesh ("fine mesh") is used as the "exact" solution and is calculated only once. Again, the objective function J is defined by (4.4), where $\mathbf{u}^M$ is (420×1) displacement vector from the "fine mesh" model evaluated at the fine mesh nodes and $\mathbf{u}^{N \to M}(\boldsymbol{x})$ is a (420×1) displacements vector interpolated from each "coarse mesh" model, at the fine mesh nodes. All mesh models were successfully tuned with the same GAs used in the one-dimensional case. The resulting optimal meshes for all the models are shown in Figure 5.13.

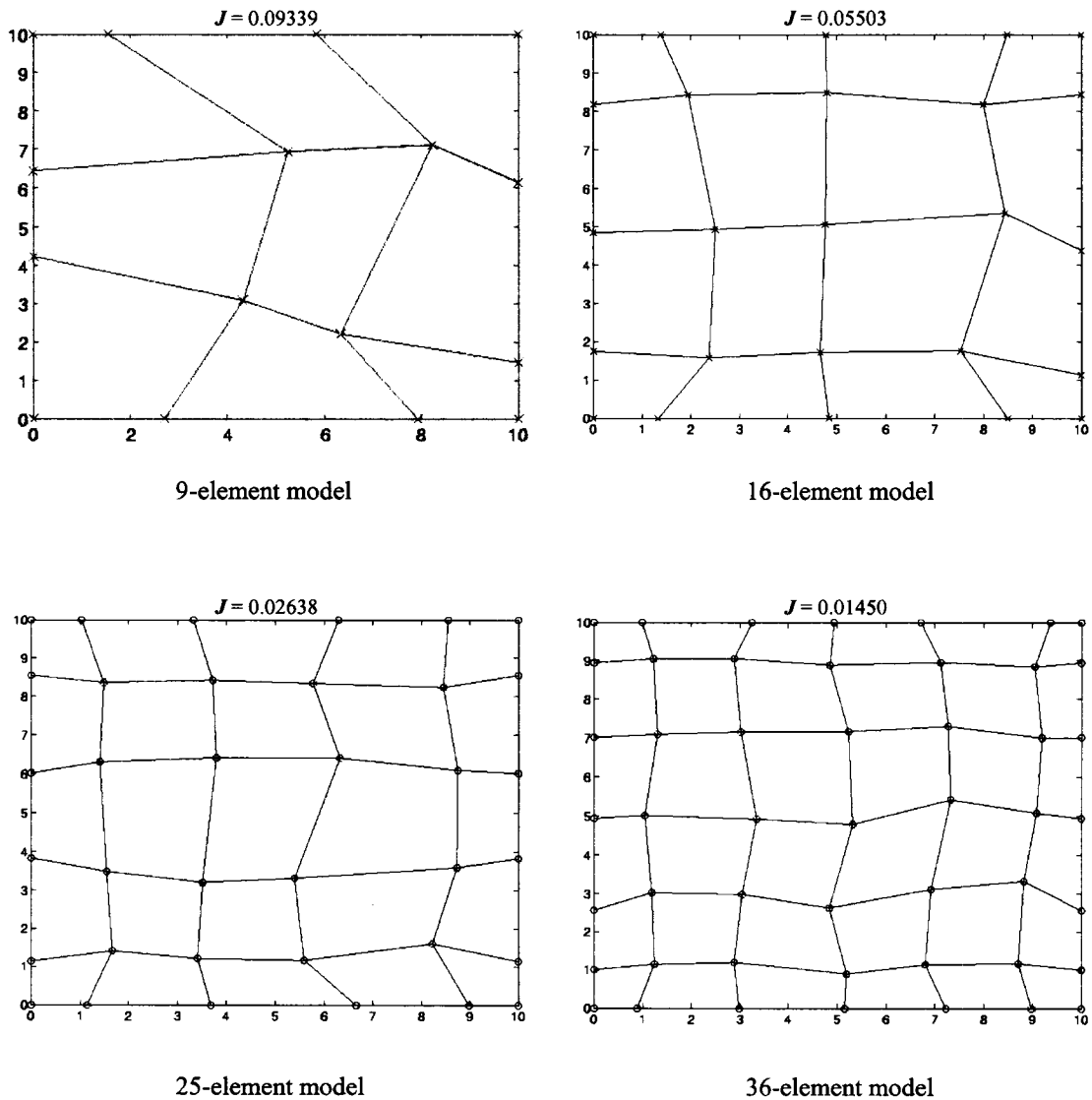


Figure 5.13. Optimal meshes for 9-, 16-, 25- and 36-element models: general quadrilateral elements

As in the one-dimensional case, it can be seen that the nodes and thus element boundaries of the optimal meshes obtained by the Genetic Algorithms fall closer to high gradient regions (see Figure 5.10), which in this case are the corners of the plate

geometry. These regions in the plate are located where the contour lines have higher density (see Figure 5.11).

To avoid the rapid increase of number of parameters, resulting from the use of the Cartesian coordinates of the nodes as decision variables, several functional relations between optimization parameters and nodal positions were implemented. In addition to functional correspondence, some geometrical simplifications were also made. First, all elements were assumed to have rectangular shapes and formed by vertical and horizontal lines (see Figure 5.14). In this case the optimization parameters define the positions of these lines along x and y directions, thus defining the finite element mesh itself.

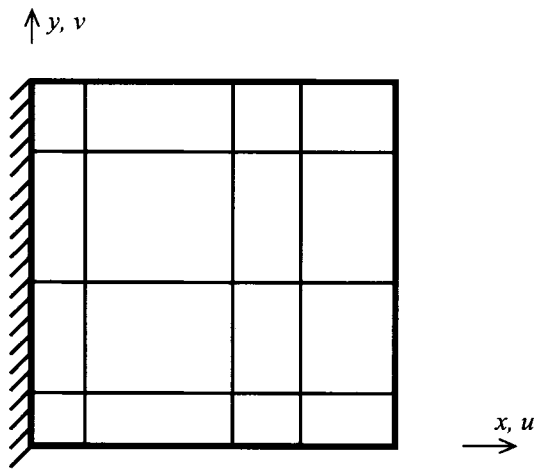


Figure 5.14. Meshing with rectangular elements

The number of optimization parameters is now ($2n$-4). Still, the number of parameters increases with the refinement of the mesh, that is, with increasing number of

elements in the finite element mesh approximation. However, the rate of increase is now linear in n, and the tradeoff is that some meshes are impossible to obtain by this type of meshing.

A 36-element model (7 nodes per side) is considered for this case. The number of optimization parameters here is now 10, much less than in previous (most general) case, 70. The optimal mesh obtained from the Genetic Algorithms for this model is shown in Figure 5.15. As it was expected, the value of the optimality criterion (the approximation error) for this case is larger than that of the previous 36-element general model, but it is comparable to the 25-element model (again general).

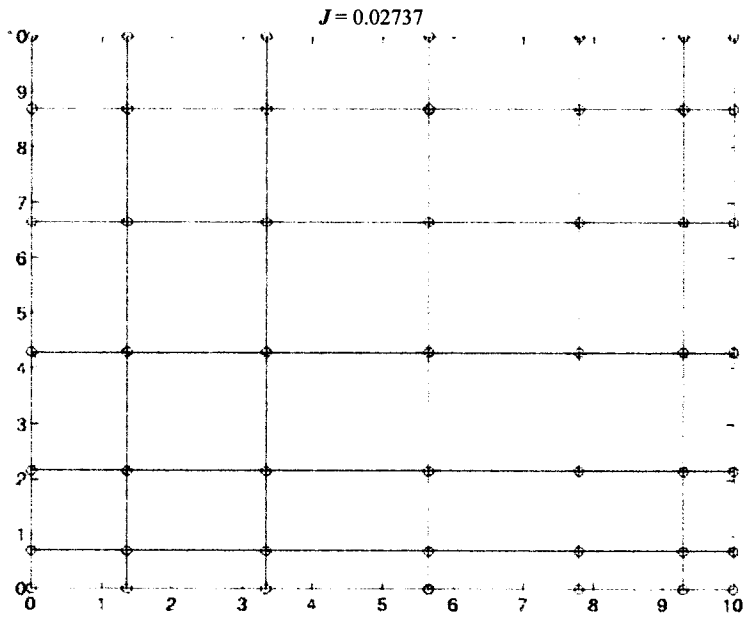


Figure 5.15. Optimal mesh for 36-element model: general rectangular elements

It should be noted that, as expected, the optimization solution gives a higher concentration of optimization parameters (positions of horizontal and vertical lines) around singularity points: the application point of the concentrated force and the sharp corners.

Next, this modification can be simplified further, reducing the total number of optimization parameters. Along with above geometrical simplifications (rectangular elements), different functional relations for positions of vertical and/or horizontal lines was assumed. The main advantage of using additional functional relations is that not only the number of parameters is reduced but also that this number does not increase with mesh refinement. The latter advantage is the most important feature of the functional description of a mesh.

One of the examples of these functional relations is biasing the distribution of line positions along each axis (Figure 5.16). The functional relations in this case for each direction can be written as follows:

$$L_i^{\alpha} = \beta_{\alpha} L^{\alpha}_{i-1}, \quad (\alpha = x, y) \tag{5.1}$$

where L_i^{α} is length of i^{th} element in α-direction ($\alpha = x, y$) counting from left (bottom) of the plate, and $\beta_{\alpha} > 0$ is the biasing coefficient which is different for each direction. Note that the sum of all L_i^{α} in either direction must be equal to L, that is $\sum_i L_i^{\alpha} = L$.

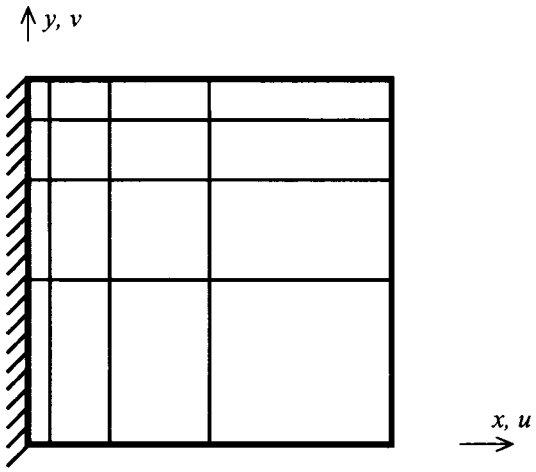


Figure 5.16. Functional relations, biasing

Thus, the biasing leaves only two optimization parameters, β_α ($\alpha = x, y$), and that number does not change with increasing the number of elements used in the mesh. The price of this obvious advantage is that the mesh variations are very limited which makes the "best" mesh unreachable. The explanation of this is really trivial: biasing function variation is very limited and possible shapes of the curves of length L_i versus its index i for this function are only of exponential form. The previous example (36-element model) was used as an illustration for biasing and the optimal mesh obtained for this case is depicted in Figure 5.17. The value of the optimality criterion (the error) indicates the model with biasing is not a very good choice and the method requires further modifications. Thus, more general functional representations are needed to create meshes which are able to reach desired accuracy for finite element solution.

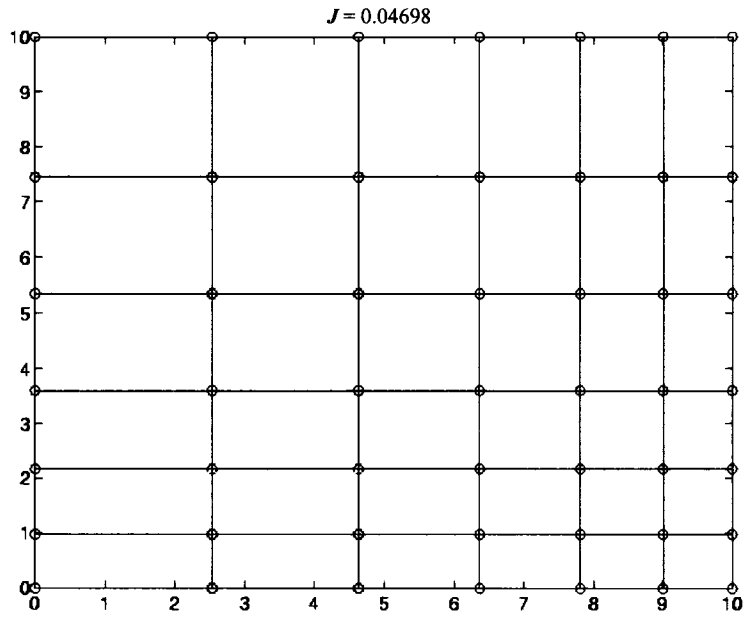


Figure 5.17. Optimal mesh for 36-element model: rectangular elements, biasing representation

Almost any function can be written in its full Fourier series form, which then can be used to describe arbitrary rectangular mesh, as shown in Figure 5.14, without losing generality. Noting that contribution of high frequency terms (sine and cosine) in Fourier series can be considered negligible for our purposes, it is reasonable to choose functional relations of line positions as truncated sine-cosine series. As an example, a constant and two sine-cosine terms are chosen along each direction:

$$L_i = A_0 + A_1 \cos(\omega_1 i) + A_2 \sin(\omega_2 i) \tag{5.2}$$

Here optimization parameters per direction are A_0, A_1, A_2, $\omega_1$and ω_2. Thus, there are 10 optimization parameters overall, and once again their number does not increase with problem size.

The same 36-element model is optimized with the GAs and the resulting optimal mesh is presented in Figure 5.18. As it can bee seen, the optimality criterion in this model is comparable to the optimality criterion for the general rectangular mesh model (Figure 5.15).

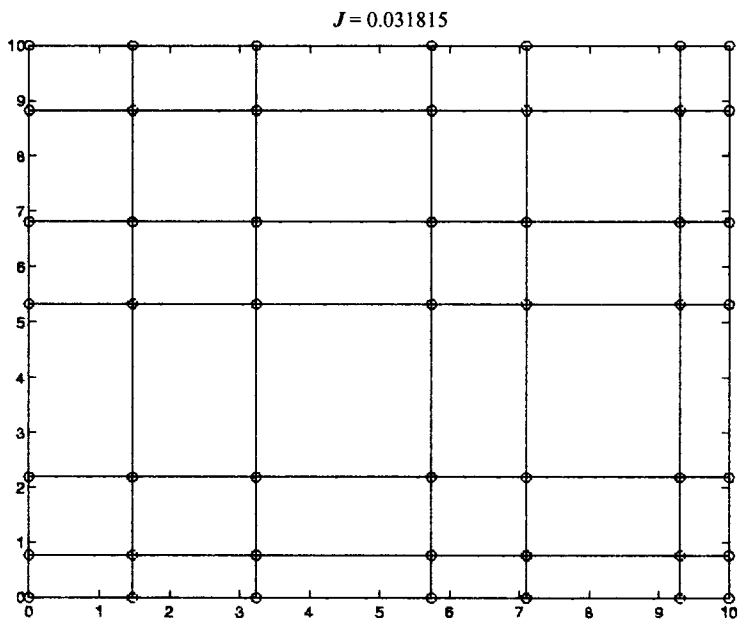


Figure 5.18. Optimal mesh for 36-element model: rectangular elements, sine-cosine representation

The progress of convergence for some considered meshes and cases are shown below (Figure 5.19). Even though the results represent single runs for each case, they can be considered as representatives for convergence behavior.

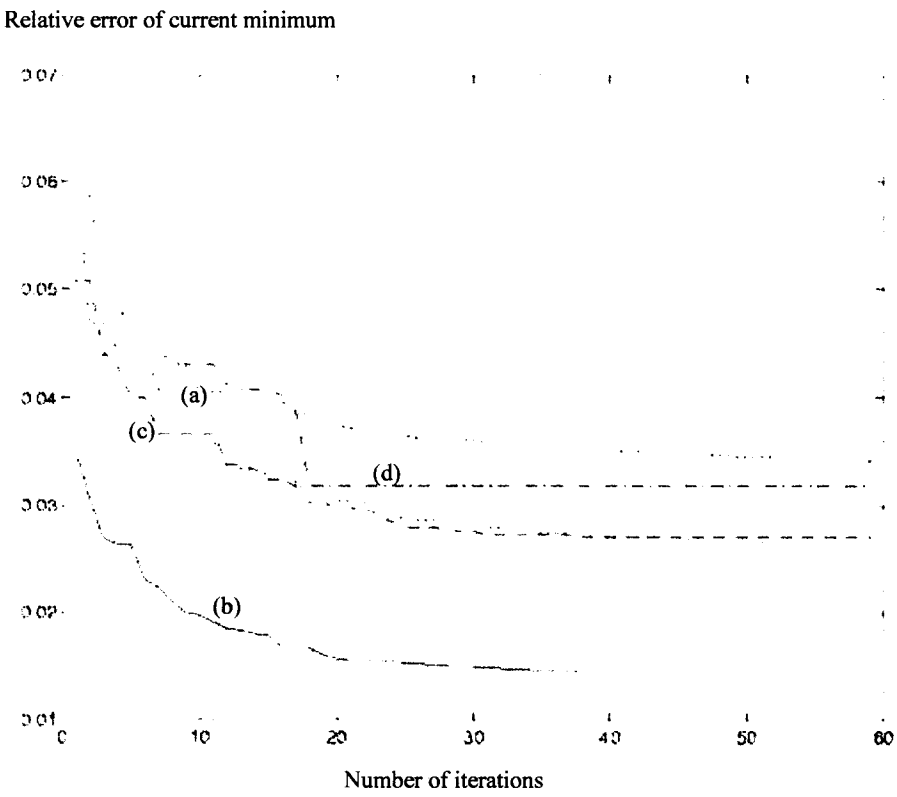


Figure 5.19. Convergence of Genetic Algorithms for: (a) 25-element mesh with quadrilateral elements (two separate runs); (b) 36-element mesh with quadrilateral elements; (c) 36-element mesh with general rectangular elements; and (d) 36-element mesh with rectangular elements, harmonic representation

Example 3. A thin plate with an opening

Real-life finite element applications frequently involve structures which have complex boundaries with, perhaps, stress concentrations. The application of structural optimization with openings and inner boundaries are of particular interest. The implementation of mesh generation involves certain difficulties in this type of multidimensional structural configurations. As an example, a thin plate with an opening under plane stress condition is explored.

Consider the same thin square plate from Example 2, but now with a square opening of $3^1/_3 \times 3^1/_3$ m in the middle of the plate (see Figure 5.20). As before, the plate size is 10×10 m with thickness of 4×10^{-4} m. One edge of the plate is fixed and a concentrated load of magnitude 10 kN is applied at a corner of the plate at a 45° angle, as shown below. The Young's modulus and Poisson ratio are 210 GPa and 0.3, respectively.

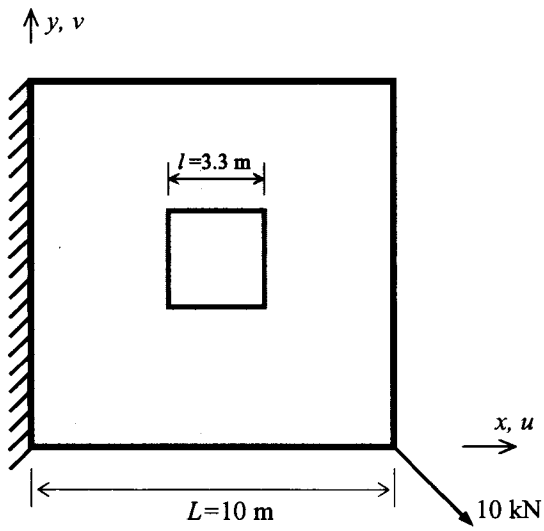


Figure 5.20. Square plate with opening under plane stress

As before, the problem is first solved using ABAQUS. The plate with opening is modeled with 2312 quadratic plane stress elements with reduced integration (CPS8R). After applying appropriate boundary conditions and loads, the problem is solved for static deformation and stresses are obtained. Banded and line contour plots of in-plane maximum principal stress, S_{max}, are shown in Figure 5.21 and Figure 5.22, respectively.

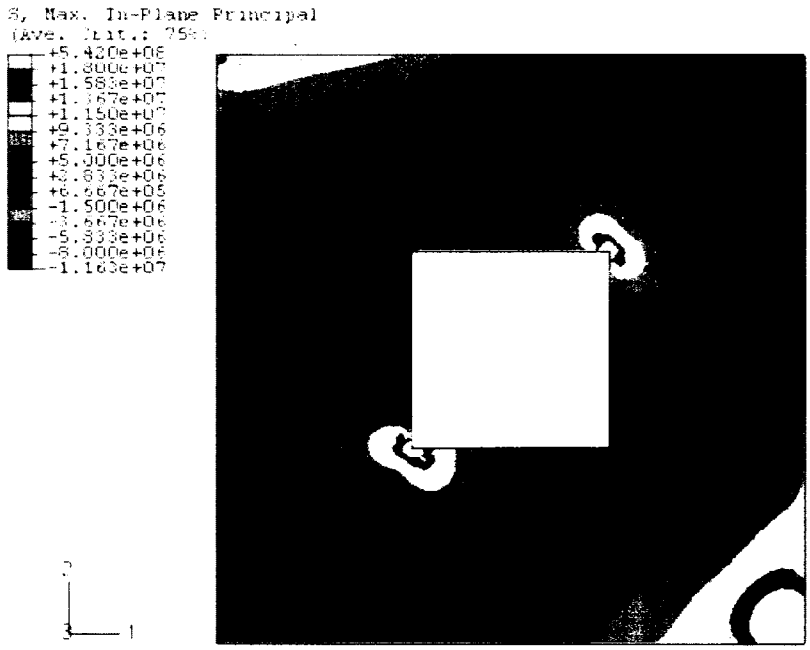


Figure 5.21. In-plane maximum principal stress for plate with opening: banded contour

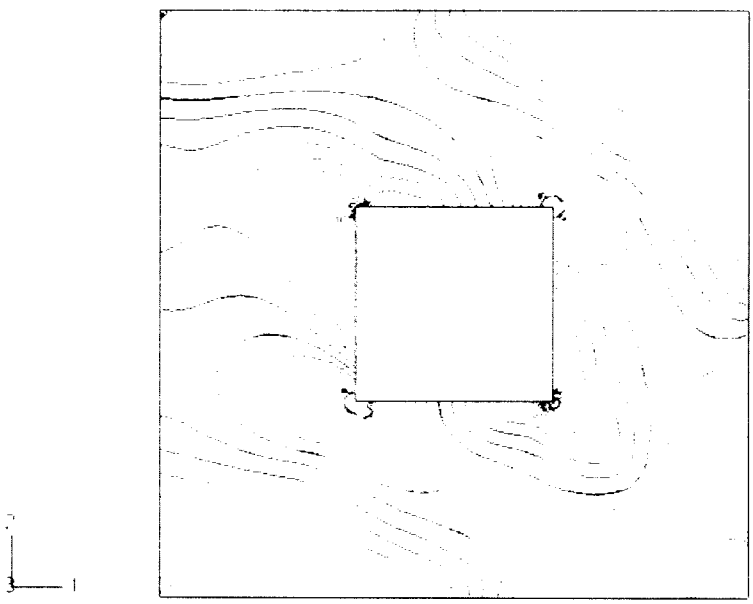


Figure 5.22. In-plane maximum principal stress for plate with opening: line contour

Next, for the finite element mesh optimization purposes, the plate is modeled with $(n-1)^2-(m-1)^2$ quadrilateral plane stress elements (see Figure 5.23), where n and m are numbers of nodes along outer and inner sides of the plate, respectively. For simplicity n and m are assumed to be the same along all sides.

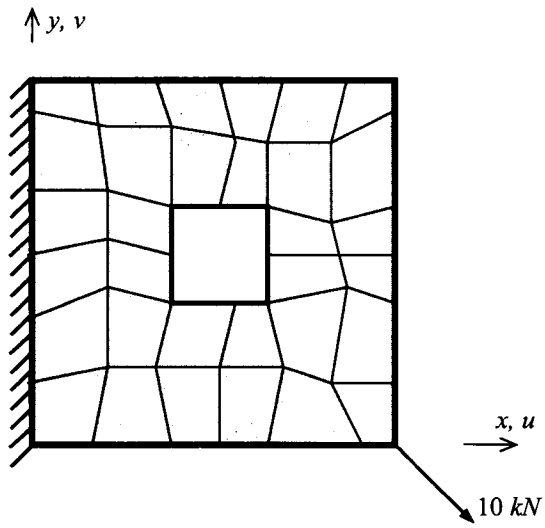


Figure 5.23. Finite element model of the plate with opening

For this case of optimization a 144-element model (n = 14, m = 6) with 280 optimization parameters is considered. The optimization parameters in this example are the x and y coordinates for the nodes located inside the domain, and either the x or the y coordinates for the nodes located on the boundary (outer and inner) of the domain. A 760-element model with uniform mesh ("fine mesh") is used as the "exact" solution and is calculated only once. Again, optimality criterion J is defined by (4.4), where $\mathbf{u}^M$ is (1612×1) displacement vector from the "fine mesh" model evaluated at the fine mesh nodes ("exact" solution) and $\mathbf{u}^{N\to M}(\boldsymbol{x})$ is a (1612×1) displacements vector interpolated

from each "coarse mesh" model, at the fine mesh nodes. The model is successfully tuned with the GAs with real-number coding.

Furthermore, other aspects of GAs are explored for this type of finite element applications. This implementation of GAs involves choice of number of active members (individual meshes), N_{Active}, in a current population and number of newly created members (children), N_{Adapt}, to be considered as replacements in every generation (iteration). In order to assess the sensitivity of GAs to these parameters in this optimization problem, several cases of N_{Active} and N_{Adapt} are considered:

	N_{Active} population size	N_{Adapt} offspring
Case 1	50	25
Case 2	50	50
Case 3	200	25
Case 4	200	50

One initial choice in GAs which could affect the progress, and possibly even the outcome, of the optimization process is the selection of the initial population for the algorithm. As discussed in Chapter 2 above, the initial population should represent a random sample from the search space and any available knowledge about possible solution can and must be used so that members of population include as much information about the solution as possible. Knowing that undistorted elements (rectangular or square) perform better in most of the finite element applications and

taking into account randomness requirements for the choice of members (meshes) in the initial population, the following two types of initial mesh generations were considered:

- quadrilateral - each node position is chosen randomly and fully independent of others (most general mesh), then elements are generated as described in Example 2 (see examples in Figure 5.12 and Figure 5.23);
- rectangular - the nodes lie along vertical and horizontal lines, positions of which are chosen randomly. This forces the elements to have rectangular shapes (see example in Figure 5.14).

Both types of meshes were used in initial populations with different weights to examine the effect of various initial mesh types on the progress of the solution and the final result. For each of the above mentioned cases (Case 1 through Case 4), several combinations of quadrilateral / rectangular mesh ratios (total 100%) were considered in the initial population:

Mesh / Ratios	Quadrilateral	Rectangular
q0.00 r1.00	0 %	100 %
q0.03 r0.97	3 %	97 %
q0.50 r0.50	50 %	50 %
q0.80 r0.20	80 %	20 %
q1.00 r0.00	100 %	0 %

Thus, a total of 20 mesh optimization problems were considered and set up for this example. The optimal meshes for the 144-element model of the plate with a square opening for Case 1 (N_{Active} = 50 and N_{Adapt} = 25) and all of the above mentioned variations for distribution in the initial population are shown in Figures 5.24 through 5.28. Similarly, the optimal meshes for the same model are shown as follows: for Case 2 (N_{Active} = 50 and N_{Adapt} = 50) in Figures 5.29 through 5.33; for Case 3 (N_{Active} = 200 and N_{Adapt} = 25) in Figures 5.34 through 5.38; and for Case 4 (N_{Active} = 200 and N_{Adapt} = 50) in Figures 5.39 through 5.43.

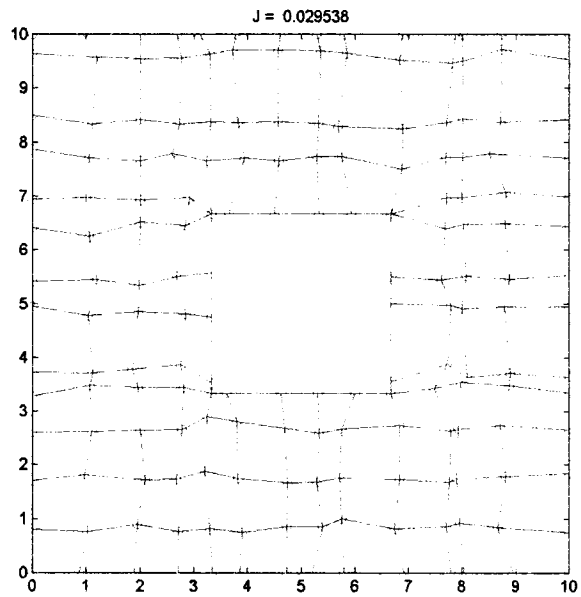


Figure 5.24. Optimal meshes for N_{Active} = 50 and N_{Adapt} =25 case with initial population quadrilateral / rectangular mesh ratio of 0 / 100

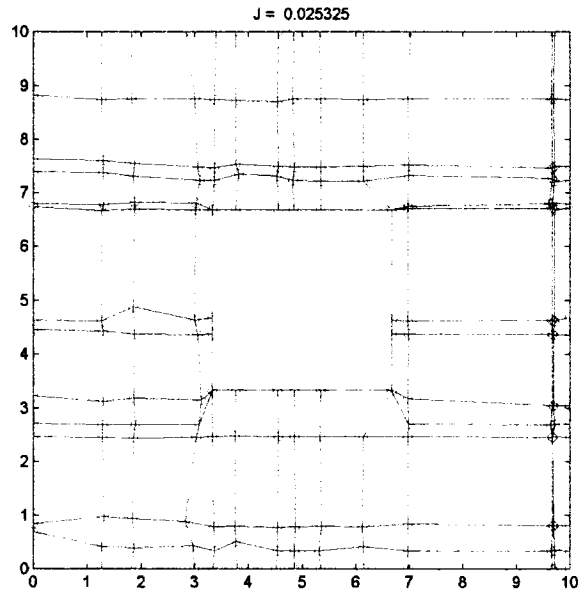


Figure 5.25. Optimal meshes for N_{Active} = 50 and N_{Adapt} =25 case with initial population quadrilateral / rectangular mesh ratio of 3 / 97

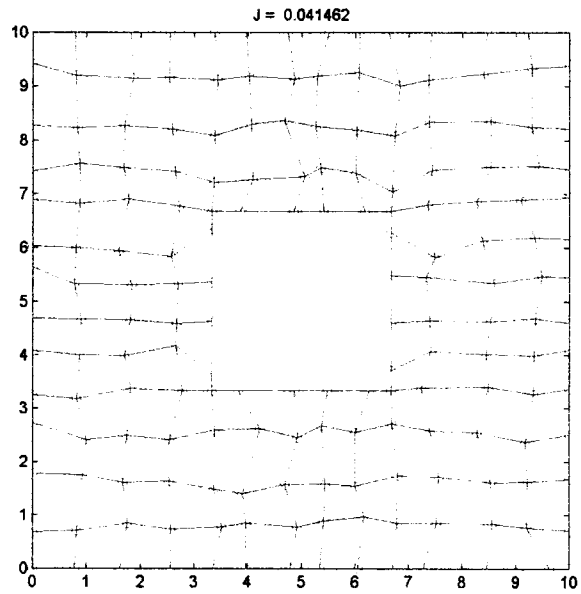


Figure 5.26. Optimal meshes for N_{Active} = 50 and N_{Adapt} =25 case with initial population quadrilateral / rectangular mesh ratio of 50 / 50

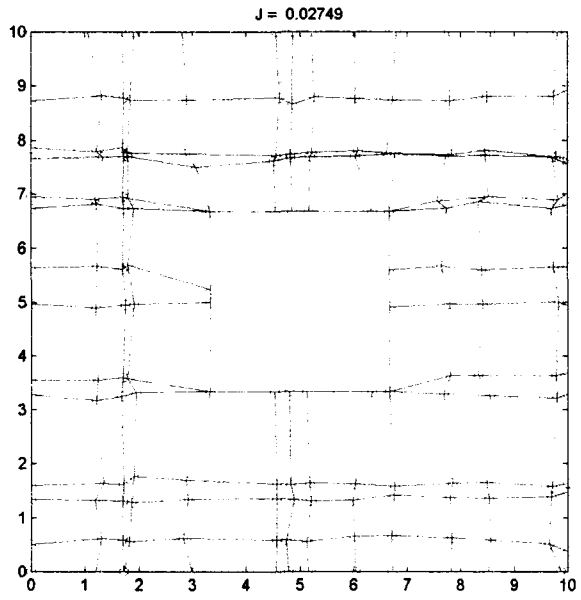


Figure 5.27. Optimal meshes for N_{Active} = 50 and N_{Adapt} =25 case with initial population quadrilateral / rectangular mesh ratio of 80 / 20

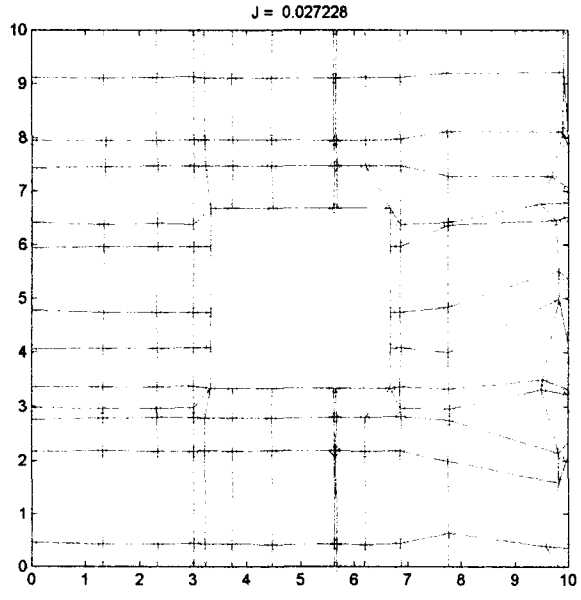


Figure 5.28. Optimal meshes for N_{Active} = 50 and N_{Adapt} =25 case with initial population quadrilateral / rectangular mesh ratio of 100 / 0

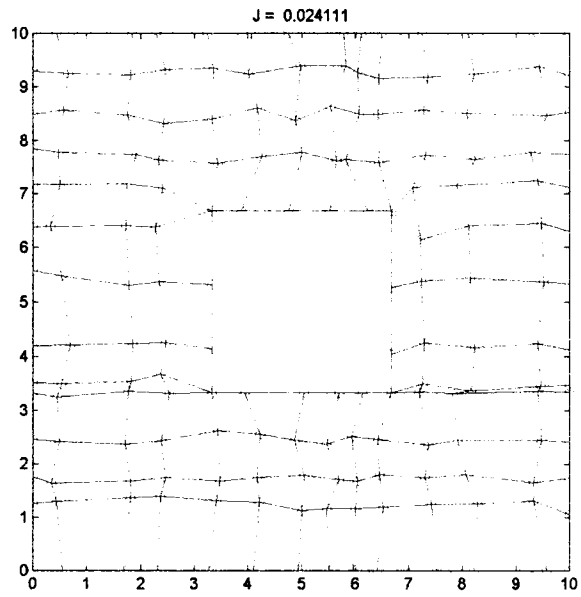


Figure 5.29. Optimal meshes for N_{Active} = 50 and N_{Adapt} =50 case with initial population quadrilateral / rectangular mesh ratio of 0 / 100

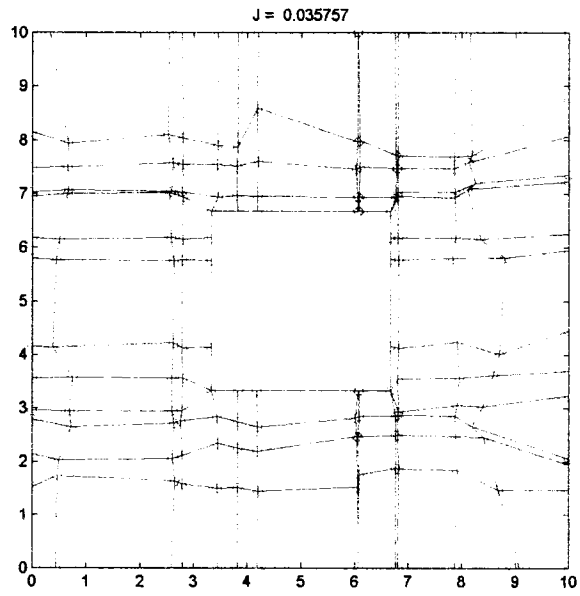


Figure 5.30. Optimal meshes for N_{Active} = 50 and N_{Adapt} =50 case with initial population quadrilateral / rectangular mesh ratio of 3 / 97

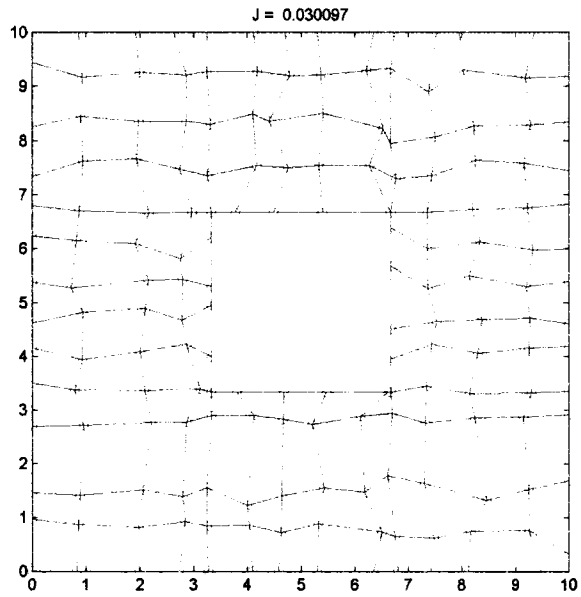


Figure 5.31. Optimal meshes for N_{Active} = 50 and N_{Adapt} =50 case with initial population quadrilateral / rectangular mesh ratio of 50 / 50

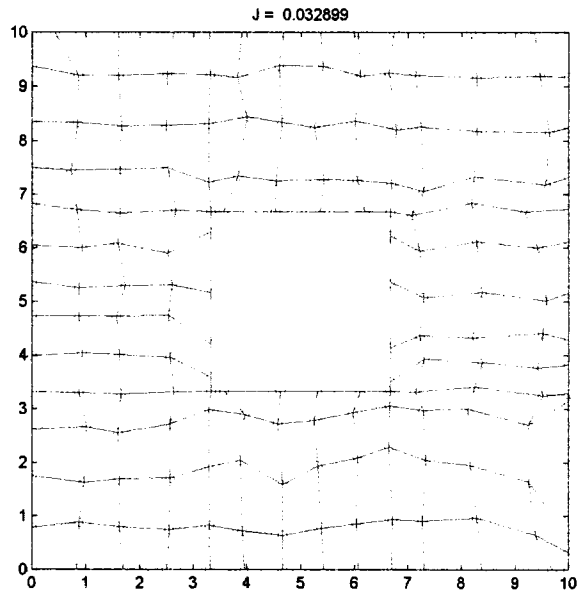


Figure 5.32. Optimal meshes for N_{Active} = 50 and N_{Adapt} =50 case with initial population quadrilateral / rectangular mesh ratio of 80 / 20

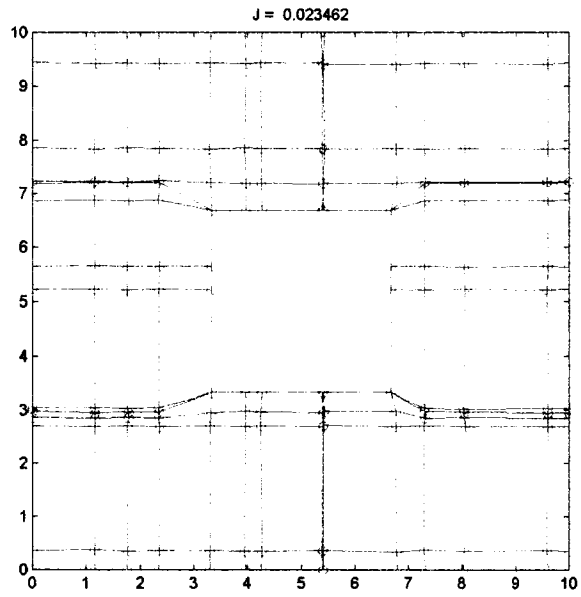


Figure 5.33. Optimal meshes for N_{Active} = 50 and N_{Adapt} =50 case with initial population quadrilateral / rectangular mesh ratio of 100 / 0

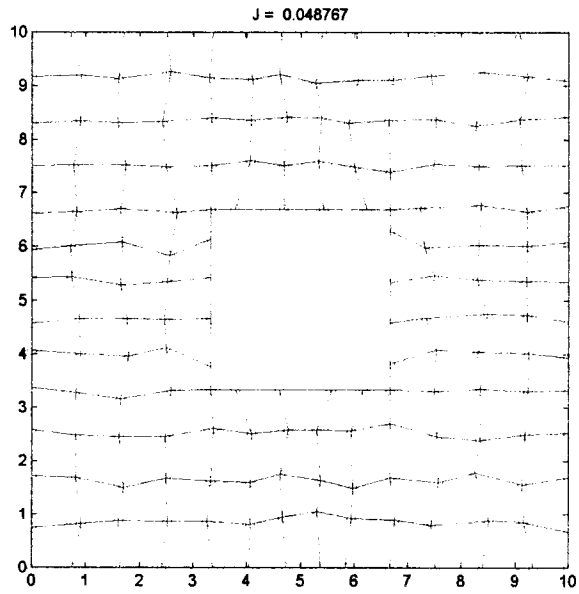


Figure 5.34. Optimal meshes for N_{Active} = 200 and N_{Adapt} =25 case with initial population quadrilateral / rectangular mesh ratio of 0 / 100

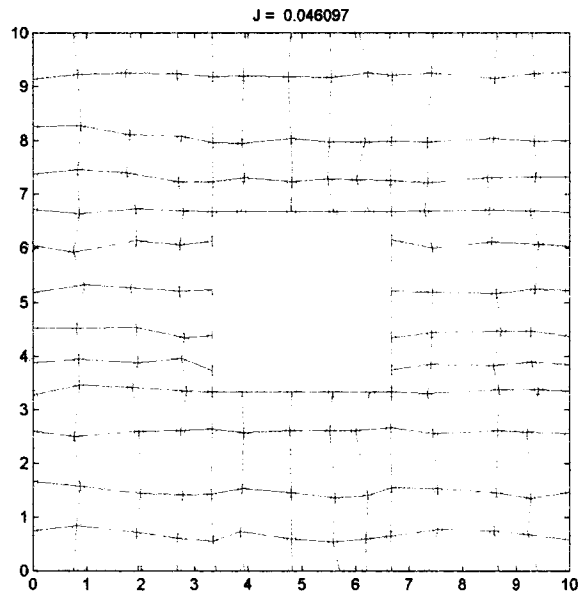


Figure 5.35. Optimal meshes for N_{Active} = 200 and N_{Adapt} =25 case with initial population quadrilateral / rectangular mesh ratio of 3 / 97

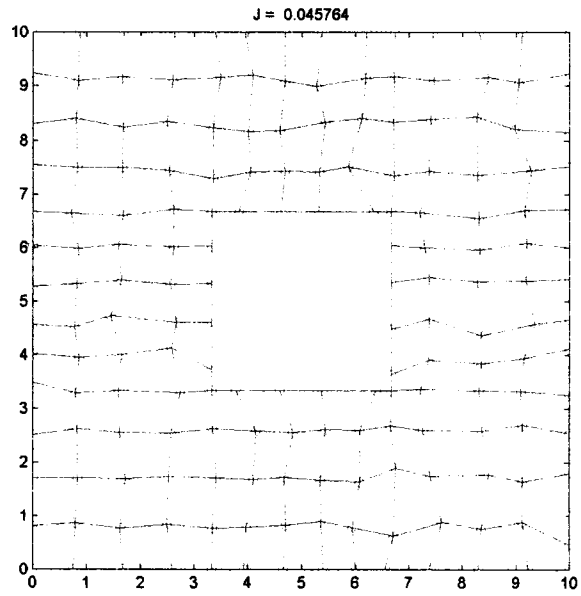


Figure 5.36. Optimal meshes for N_{Active} = 200 and N_{Adapt} =25 case with initial population quadrilateral / rectangular mesh ratio of 50 / 50

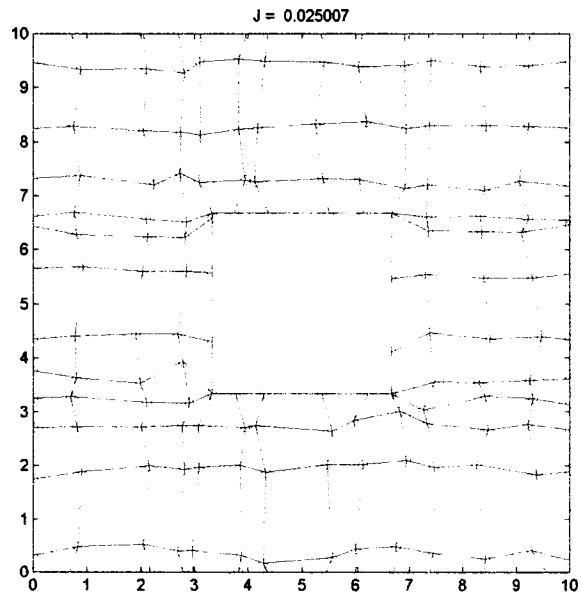


Figure 5.37. Optimal meshes for N_{Active} = 200 and N_{Adapt} =25 case with initial population quadrilateral / rectangular mesh ratio of 80 / 20

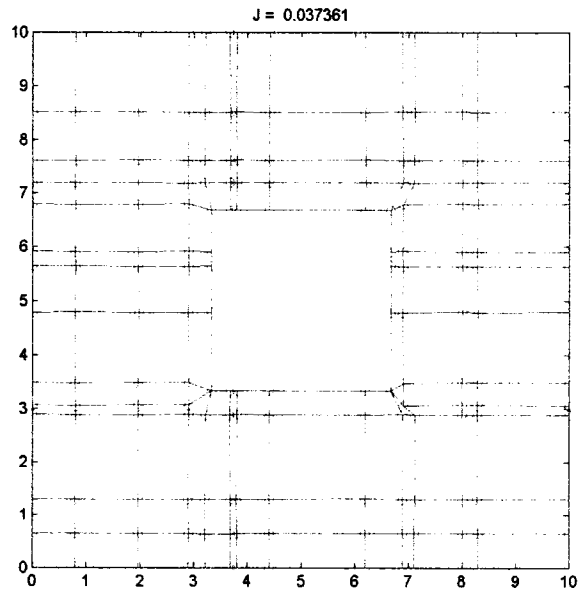


Figure 5.38. Optimal meshes for N_{Active} = 200 and N_{Adapt} =25 case with initial population quadrilateral / rectangular mesh ratio of 100 / 0

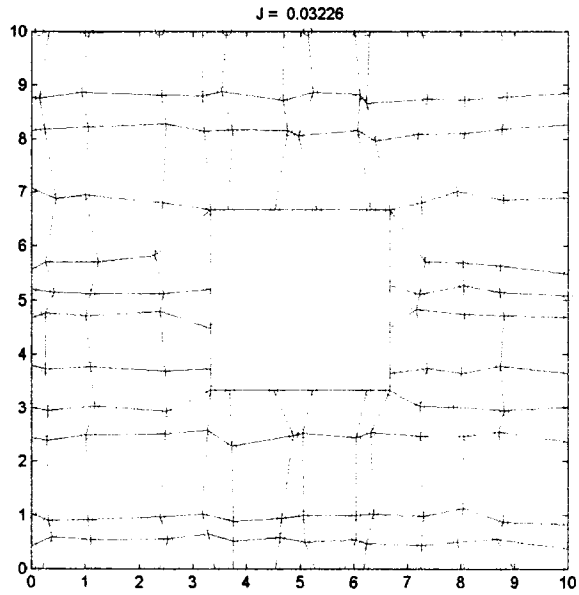


Figure 5.39. Optimal meshes for N_{Active} = 200 and N_{Adapt} =50 case with initial population quadrilateral / rectangular mesh ratio of 0 / 100

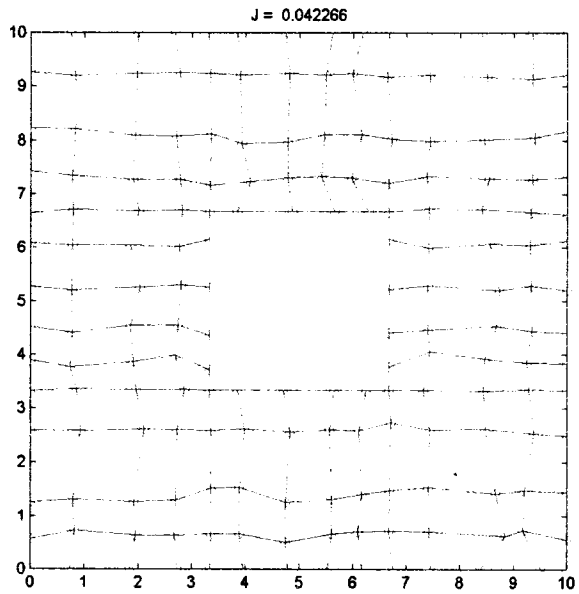


Figure 5.40. Optimal meshes for N_{Active} = 200 and N_{Adapt} =50 case with initial population quadrilateral / rectangular mesh ratio of 3 / 97

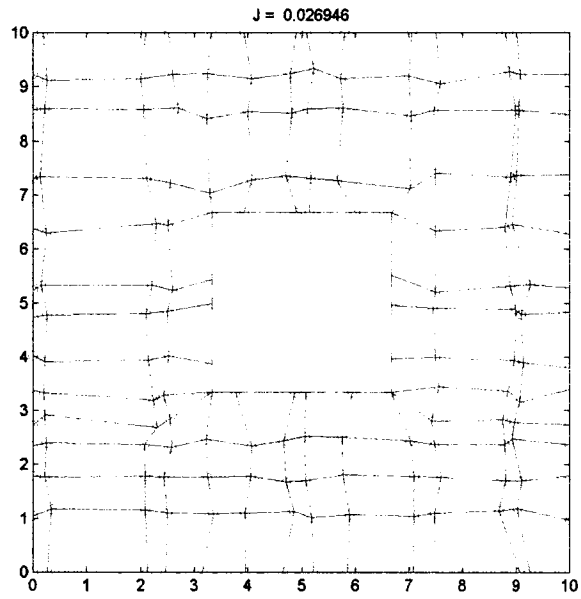


Figure 5.41. Optimal meshes for N_{Active} = 200 and N_{Adapt} =50 case with initial population quadrilateral / rectangular mesh ratio of 50 / 50

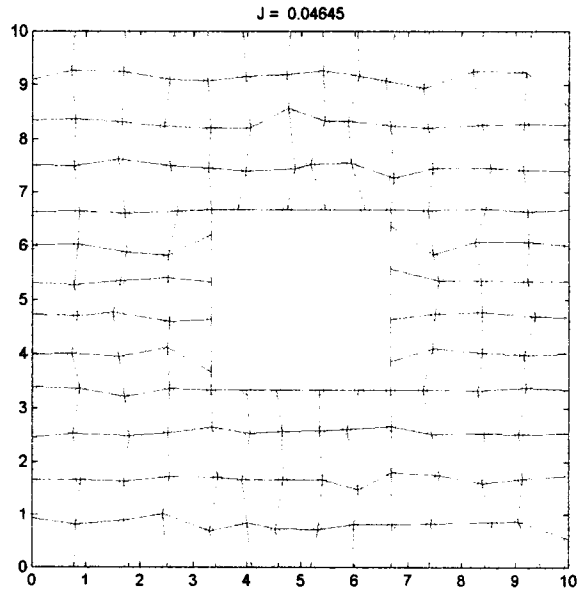


Figure 5.42. Optimal meshes for N_{Active} = 200 and N_{Adapt} =50 case with initial population quadrilateral / rectangular mesh ratio of 80 / 20

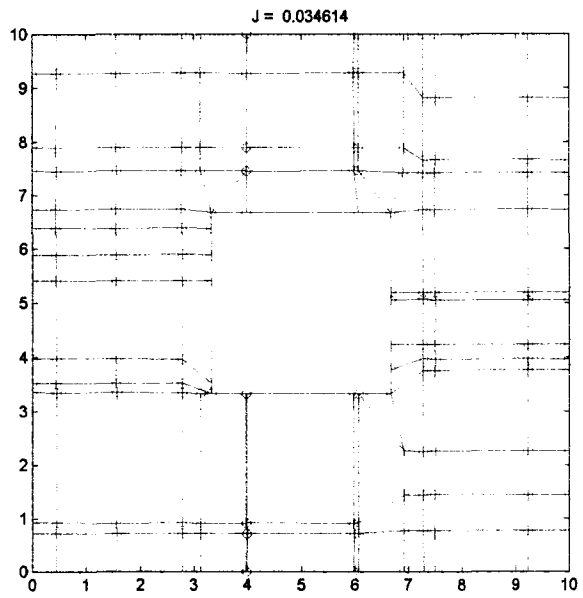


Figure 5.43. Optimal meshes for N_{Active} = 200 and N_{Adapt} =50 case with initial population quadrilateral / rectangular mesh ratio of 100 / 0

As it can be seen from Figures 5.24 through 5.43, the initial mesh choice does not affect much the final optimal mesh quality or the value of final optimality criterion, *J*, as long as randomness of possible node positions is satisfied. Though, it is worth to mention that it does influence the convergence progress of the optimization algorithm. In addition, introducing possible "good" (rectangular) meshes in the initial population helps the convergence to the final mesh.

On the other hand, the choice of the number of active members in a population (size of the population) along with the number of children (new meshes) in a generation has a stronger effect on convergence. The larger these numbers, the lower the number of iterations required for convergence, and, of course, the more expensive each iteration.

The progress of GAs convergence for the considered cases 1 through 4 is shown below in Figures 5.44 through 5.47. These figures show that a larger ratio of quadrilateral (more general) to rectangular initial mesh shapes ensures faster convergence rate in the beginning of the optimization. However, as the iterations progress, this choice of the initial population distribution does not necessarily lead to a faster convergence or better final optimality criterion. This demonstrates the ability of GAs to bring new information not selected in the random initialization into the population.

Nonetheless, it is more practical to improve initial mesh quality quickly in a few optimization iterations rather than get the "best" possible mesh after many iterations, thus saving both time and computational resources. Therefore, using a larger number of quadrilateral elements with random nodal locations in the mesh initialization can be a more effective and beneficial choice.

The legend names in Figures 5.44 through 5.47 below are chosen to identify the ratio of quadrilateral to rectangular FE mesh shapes in the initial population. The curve with legend "q0.00 r1.00" is plotted for the case with 0/100 ratio of quadrilateral to rectangular initial mesh. Similarly, the curves with the following legends are plotted for the ratios of quadrilateral to rectangular meshes in the initial population as follows: legend "q0.03 r0.97" stands for ratio 3/97; legend "q0.50 r0.50" stands for ratio 50/50; legend "q0.80 r0.20" stands for ratio 50/50; and legend "q1.00 r0.00" stands for ratio 100/0.

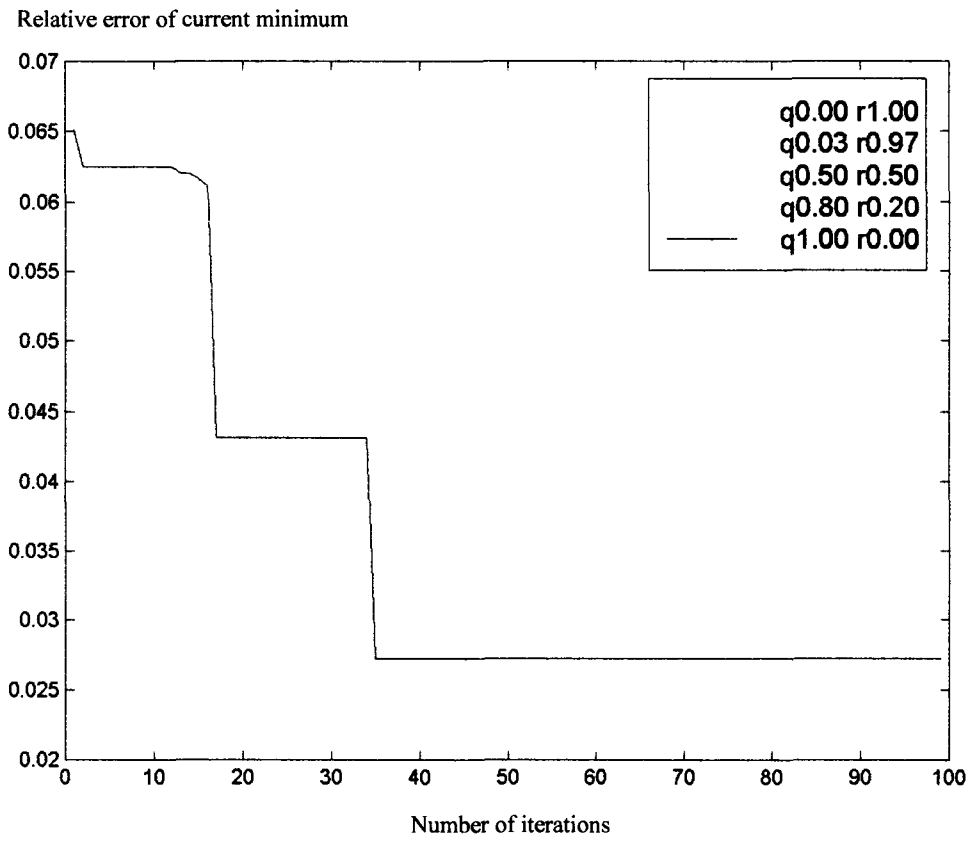


Figure 5.44. Convergence of GAs for N_{Active} = 50 and N_{Adapt} = 25, and various ratios of quadrilateral / rectangular meshes in initial population

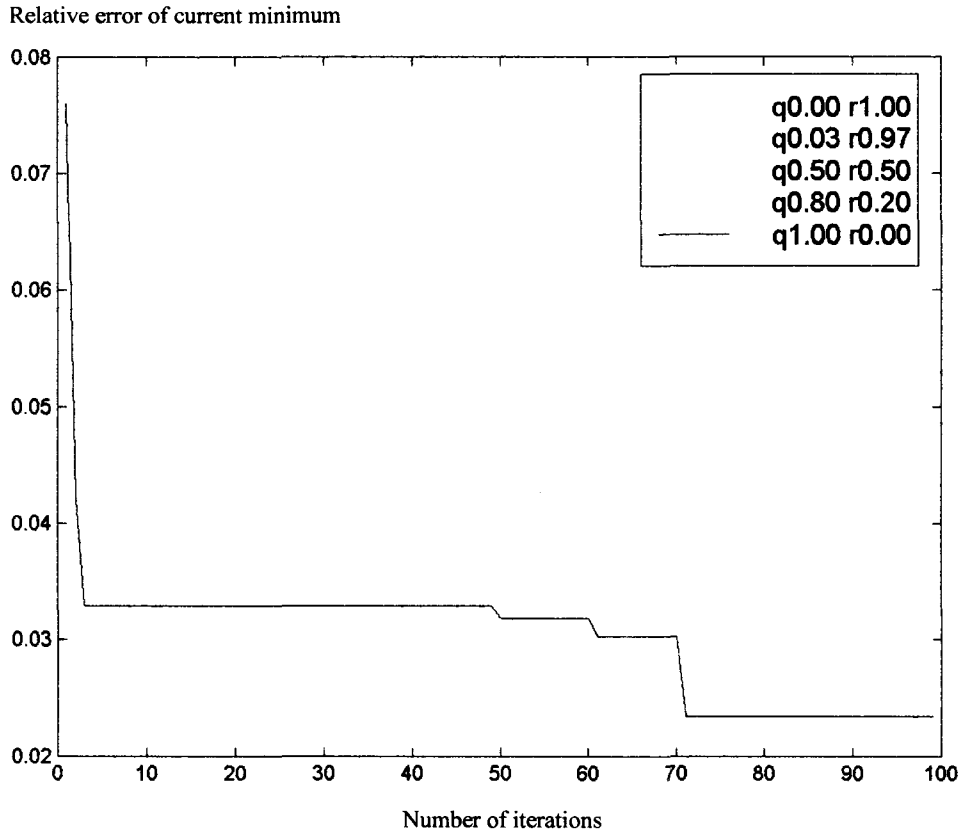


Figure 5.45. Convergence of GAs for N_{Active} = 50 and N_{Adapt} = 50, and various ratios of quadrilateral / rectangular meshes in initial population

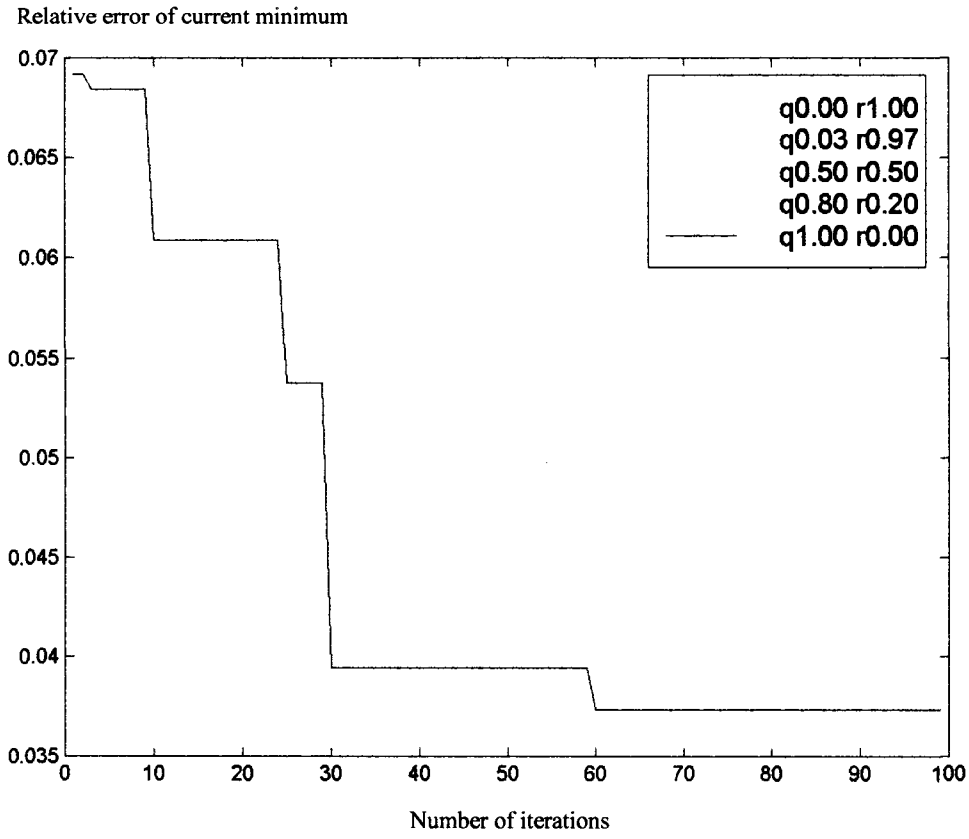


Figure 5.46. Convergence of GAs for $N_{Active} = 200$ and $N_{Adapt} = 25$, and various ratios of quadrilateral / rectangular meshes in initial population

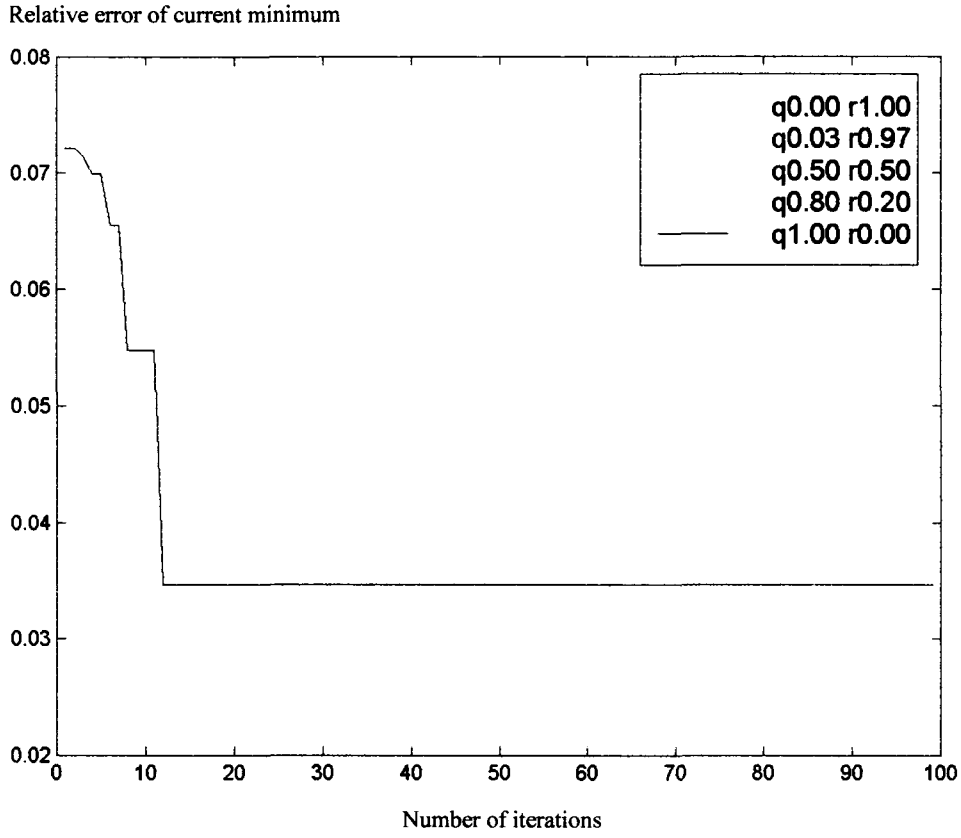


Figure 5.47. Convergence of GAs for N_{Active} = 200 and N_{Adapt} = 50, and various ratios of quadrilateral / rectangular meshes in initial population

And finally, Figures 5.48 and 5.49 depict, respectively, the minimum / maximum convergence envelope and the mean value over all ratios of quadrilateral / rectangular meshes in the initial population for each case considered above. It can be seen from the figures that initial convergence is faster for the cases when more new members (N_{Adapt} = 50) are chosen in every generation for forming of new population. However, the envelope also shows that in all cases the convergence is approximately the same, and thus it is little affected by the choice of population size (N_{Active}) and number of children in a

generation (N_{Adapt}) for considered ranges. This actually demonstrates the robustness of GAs for finite element mesh optimization applications.

The legend names in Figures 5.48 and 5.49 below are chosen to identify the number of active members ($N_{Active} \equiv$ pop) in a population and the number of newly created children ($N_{Adapt} \equiv$ child). Thus, the curve with legend "pop50 child25" is plotted for the Case 1 ($N_{Active} = 50$ and $N_{Adapt} = 25$). Similarly, the curves with the following legends are plotted as follows: legend "pop50 child50" for Case 2 ($N_{Active} = 50$ and $N_{Adapt} = 50$); legend "pop200 child25" for Case 3 ($N_{Active} = 200$ and $N_{Adapt} = 25$); and legend "pop200 child50" for Case 4 ($N_{Active} = 200$ and $N_{Adapt} = 50$).

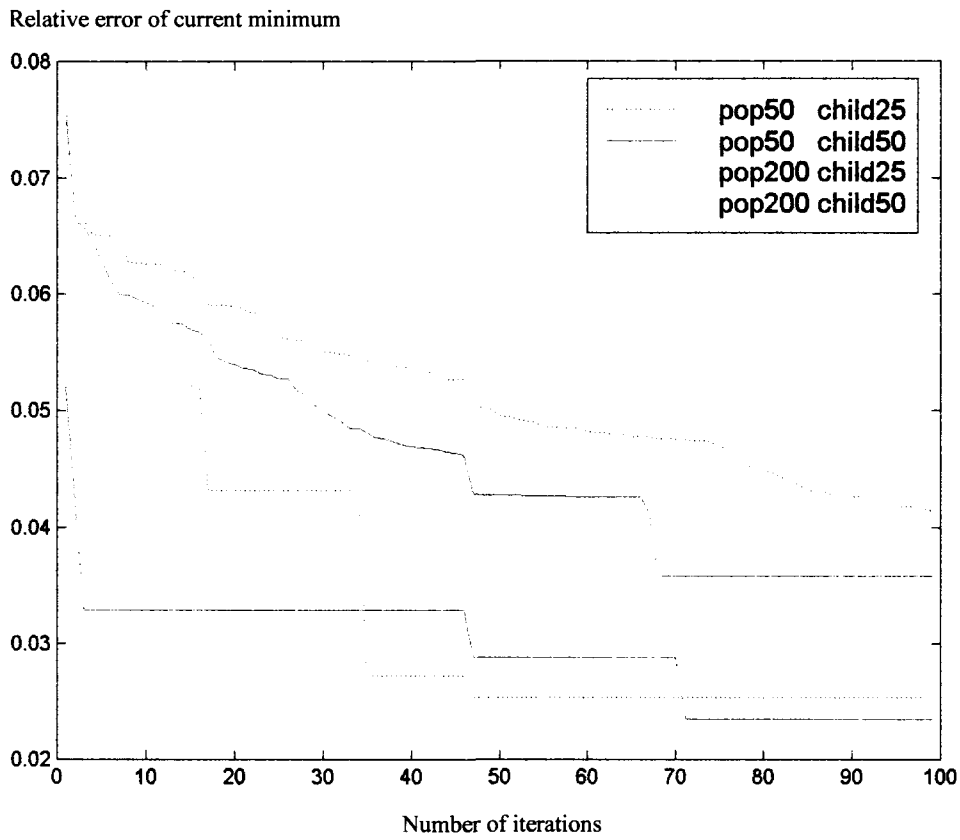


Figure 5.48. Minimum / maximum envelope of GAs convergence over choice of initial population for various N_{Active} and N_{Adapt}

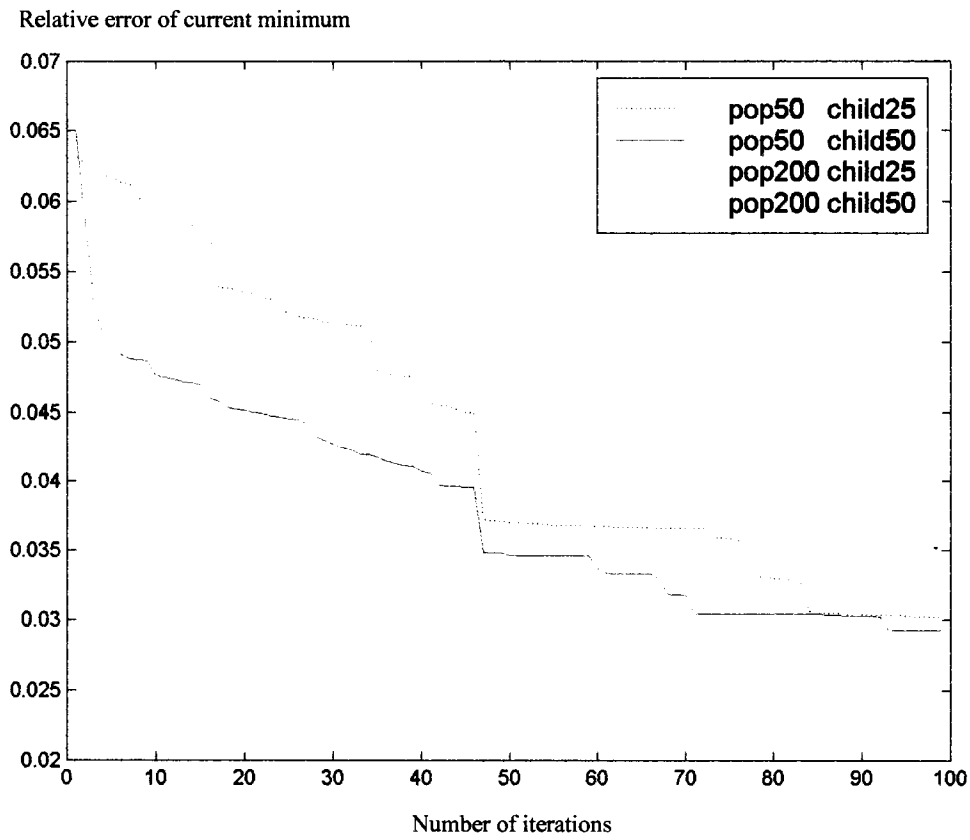


Figure 5.49. Mean value of GAs convergence over choice of initial population for various N_{Active} and N_{Adapt}

Chapter 6

Mesh Generation in Commercial Finite Element Analysis Software

In recent years, advances in commercial finite element analysis (FEA) software and the rapid rise in the processing power of computers have encouraged wider use of FEA codes across all engineering disciplines. With increasing size and complexity of finite element models, FEA software provider companies are focusing on optimizing solver packages for speed as well as developing and improving parallelization techniques for various applications and computing platforms. However, solution-based mesh optimization is yet to be developed as the companies prioritize more on robust automatic mesh generation algorithms.

As an example, mesh generation in ABAQUS is examined. Consider the same thin square plate with square opening from Example 3 above. All meshes in this section were created in ABAQUS/CAE [35], a complete modeling and visualization environment for ABAQUS analysis products, and all models were analyzed with ABAQUS/Standard [36-40], a solver for traditional implicit finite element analysis. The plots were created in

ABAQUS/Viewer, the visualization module of ABAQUS/CAE, and the post processing calculations for the optimality criterion were implemented using the Python scripting interface to the ABAQUS results database [41]. The Python script used for the calculation of J is brought in the Appendix.

First, the plate with opening is modeled with 800 first-order plane stress elements with reduced integration (CPS4R). After applying appropriate boundary conditions and loads, the problem was solved for static deformation. To obtain comparable results with Example 3, this model was used as the "fine mesh" in the optimality criterion defined by (4.4), where now $\mathbf{u}^M$ is a (1760×1) displacement vector from "fine mesh" model evaluated at its nodes and $\mathbf{u}^{N \to M}(\boldsymbol{x})$ is a (1760×1) displacements vector interpolated from each node of "coarse mesh" (FE mesh being optimized) model at the "fine mesh" nodal locations. Banded contour plots of displacement magnitude, U, and in-plane maximum principal stress, S_{max}, are shown in Figure 6.1 and Figure 6.6, respectively.

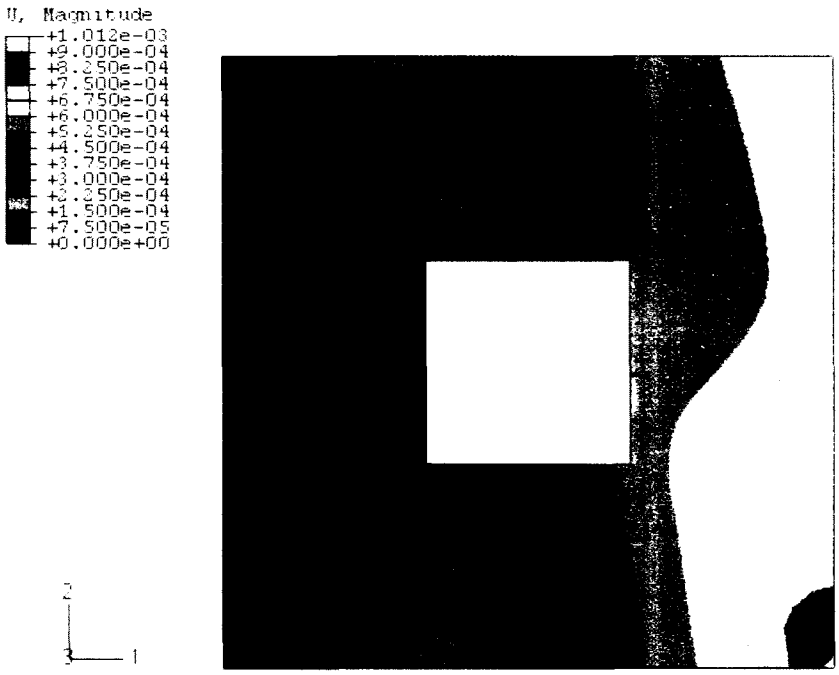


Figure 6.1. Contour of displacement magnitude for 'fine mesh' plate with opening

The "coarse mesh" plate problem is solved with several meshing techniques available in ABAQUS/CAE. Various levels of automation are available in the `Mesh` module of ABAQUS/CAE providing a variety of tools that allow specification of different mesh characteristics, such as mesh density, element shape, and element type.

The `Mesh` module uses a variety of techniques to generate meshes allowing different levels of control over the mesh. In addition, the following tools allow further tuning of the finite element mesh:

- The `seeding` tools allow adjusting the mesh density in selected regions.
- The `Partition` toolset allows partitioning complex models into simpler subregions.
- The `Virtual Topology` toolset allows simplification of the model by combining small faces and edges with adjacent faces and edges.
- The `Edit Mesh` toolset allows for minor adjustments to the mesh.

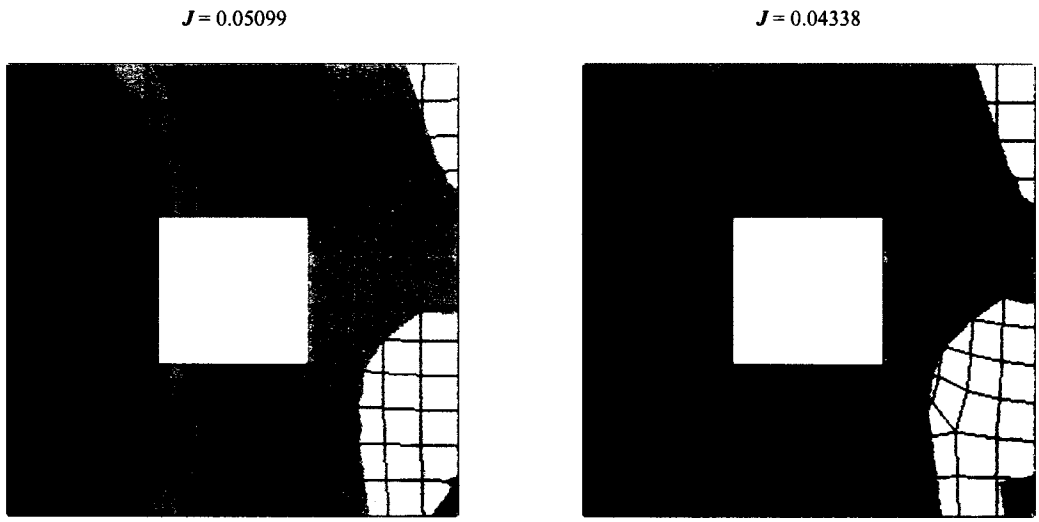
Finally, the verification tools provide information concerning the quality of the elements used in a mesh.

First, comparison solutions were obtained using the two free meshing techniques appropriate for the problem on hand:

1. The *medial axis* algorithm first decomposes the region to be meshed into a group of simpler regions. The algorithm then uses structured meshing techniques to fill each simple region with elements.
2. The *advancing front* algorithm generates quadrilateral elements at the boundary of the region and continues to generate quadrilateral elements as it moves systematically to the interior of the region.

The geometries for both meshing techniques were seeded such that they produced 144 elements to match the GAs optimization performed in Example 3. However, the advancing front algorithm produced 162 elements because the seeding locations are only

honored on the boundary (vertices, edges, surfaces) of the geometry and internal nodes and elements are generated as the algorithm progresses. The displacement magnitude contour plots for both cases are shown in Figure 6.2 with the same contour limits as the "fine mesh" contour shown in Figure 6.1.



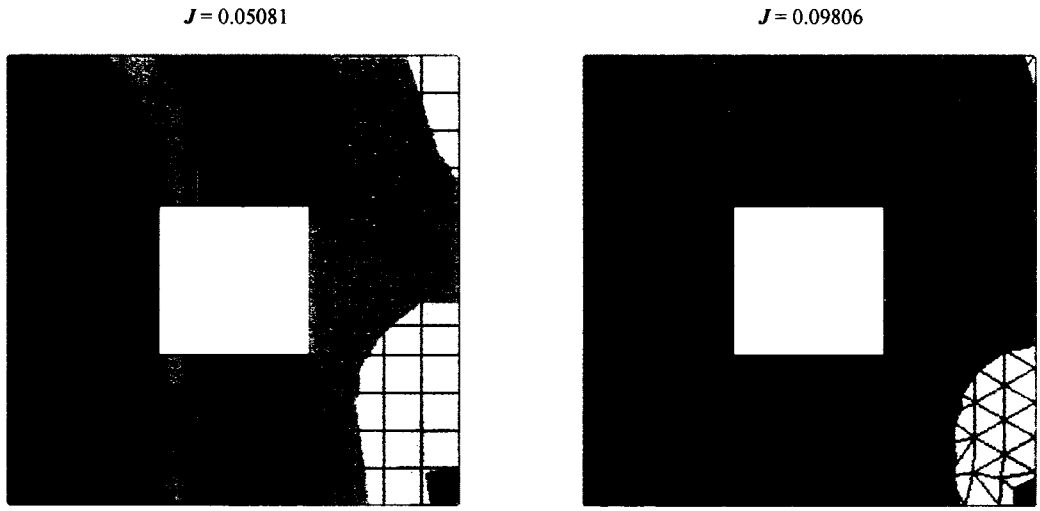
(a) medial axis: 144 elements

(b) advancing front: 162 elements

Figure 6.2. Contour of displacement magnitude using medial axis and advancing front meshing techniques

Next, two additional finite element solutions for the "coarse mesh" were obtained by meshing the geometry with rectangular and triangular elements. The rectangular mesh was obtained using structured meshing technique by first partitioning the geometry into eight simple rectangular regions and then seeding the geometry boundary such that the

resulting mesh has 144 elements. The triangular mesh was obtained by choosing the triangular element formulation for the model, and the same seed as in medial axis and advancing front cases was used. This created a finite element model with 366 elements. The contour plots for displacement magnitude for both these cases are shown in Figure 6.3 below.



(c) rectangular meshing: 144 elements

(d) triangular meshing: 366 elements

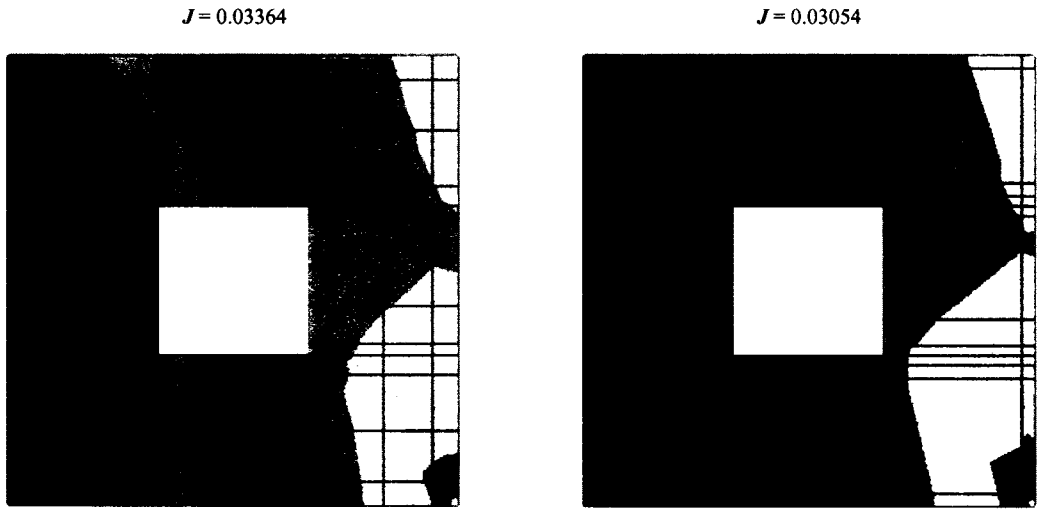
Figure 6.3. Contour of displacement magnitude using rectangular and triangular meshing techniques

It is evident from the results above (cases *a*, *b*, and *c*) that the values of optimality criterion, *J*, for all three quadrilateral meshing cases are comparable to each other and higher than that obtained for "best" optimal mesh using GAs in Example 3. The slightly

lower value of J in the case of advancing front meshing algorithm can be explained by the larger number of elements in the finite element model. Also, as expected, the optimality criterion from the triangular mesh (case *d*) is much higher, even though over 2.5 times more elements are used in the model. Triangular elements in general are known to have much lower solution accuracy compared to quadrilateral elements.

In order to get a better value for the optimality criterion, user intervention is required to assign finer distribution of nodes in certain areas of FE discretization. Several other partitioning, seeding and meshing techniques were considered. As a first approach rectangular meshing was used again along with multiple bias seeding ratios along the edges so that the resulting mesh had a higher density near the corners of the opening and the plate. In an attempt to force the automatic mesh generator to produce rectangular elements, the model was again partitioned into several simple rectangular regions and the seeds were fully constrained on the edges. Figure 6.4 shows two models with different bias ratios so that the mesh in one model is denser around the corners than in the other model.

Both mesh models show considerable improvements in the optimality criterion values. The error has been reduced from 5.1% to 3.05%. The model with higher mesh density around the corners, shown in Figure 6.4 (f), is quite similar to the optimal mesh shapes obtained from some of the GAs optimization problems considered in Example 3 (compare to Figures 5.25, 5.27 and 5.33). However, in all these cases the GAs approach generated better values of the optimality criterion.

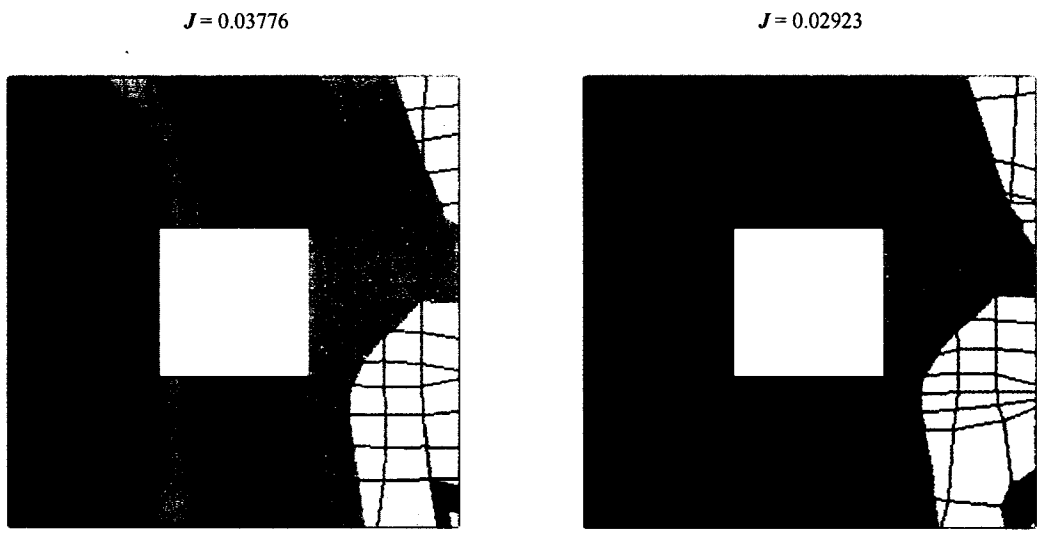


(e) lower bias ratios: 144 elements

(f) higher bias ratios: 144 elements

Figure 6.4. Contour of displacement magnitude using rectangular meshing with various bias seeding ratios

Finally, more general quadrilateral element meshing was considered with starting seed locations similar to the nodal locations of optimal meshes from Example 3 above. These seeds were again fully constrained on the edges to prevent the automatic meshing algorithm from redefining their locations. The calculations of locations for internal nodes, however, were left to the ABAQUS/CAE mesh generator. The results for these mesh models are shown in Figure 6.5.



(g) more inform seeding: 144 elements

(h) more localized seeding: 170 elements

Figure 6.5. Contour of displacement magnitude using quadrilateral meshing with various boundary seeding locations

Again, here both mesh models show similar improvements in the optimality criterion compared to the rectangular meshing with biasing (cases *e* and *f*). The model with more localized seeding, shown in Figure 6.5 (h), has the best optimality criterion value, 2.92%, obtained using ABAQUS. It is close but still higher than the lowest error value obtained using GAs optimization, 2.35%. As for previous case *f*, the mesh in case *h* is quite similar to the optimal mesh shapes obtained from some of the GAs optimization problems (see Figures 5.29, 5.37 and 5.41), which actually was the intent of the intervention in the mesh generation process.

The last four examples (cases *e*, *f*, *g*, and *h*) also show that meshes with higher node densities around the corners of the plate and the square opening yield better values of the optimality criterion. As stated above (see also Figures 6.4 and 6.5), the optimality criterion has lower values for the models with the higher bias seeding ratio (case *f*) and more localized seeding (case *h*), even though they have poorer quality of mesh: higher aspect ratios and larger angle distortions of the elements. The poor quality of the mesh effects can be realized from the stress contour plots shown in Figures 6.7 through 6.10 shown below in the same order as the displacement contour plots shown above. The model for the rectangular mesh with the higher bias seeding ratio (case *f*) and the model for quadrilateral mesh with more localized seeding locations (case *h*) exhibit discontinuity in the stress field. These models also differ more from the stress field of the "fine mesh" solution (Figure 6.6) than the other models which have higher values of optimality criterion. This is easily explained by the fact that the optimality criterion used for the mesh optimization (Equation 4.4) is chosen to operate on the displacement field, **u**, and not the stress field. This demonstrates the importance of the objective function formulation both in terms of its variables choice and on the form of their functional dependence.

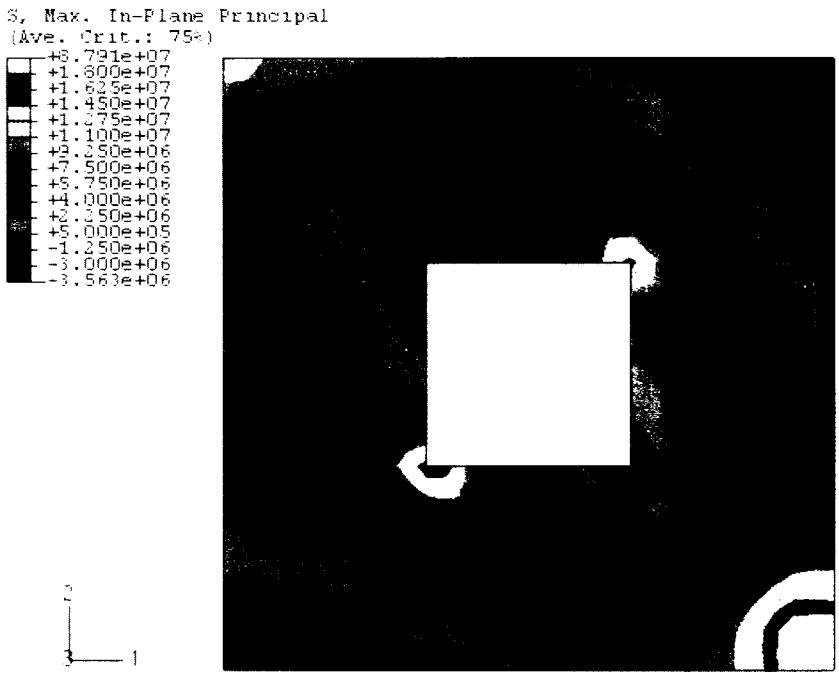
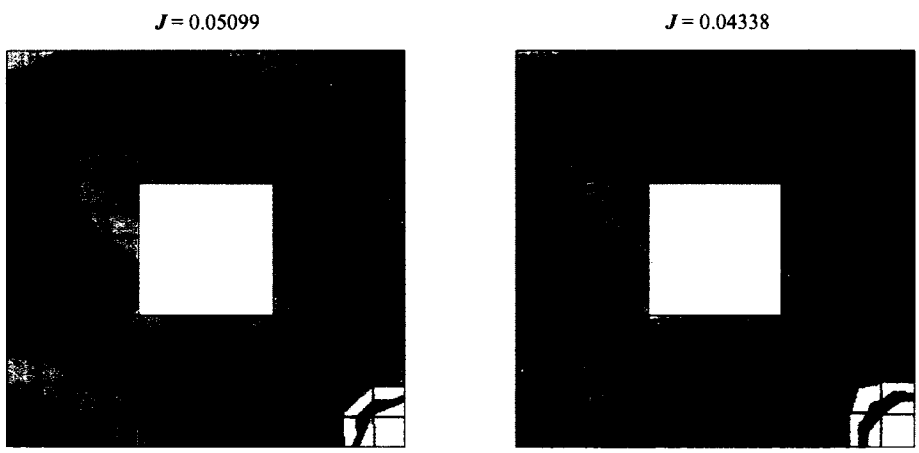


Figure 6.6. Contour of in-plane maximum principal stress for 'fine mesh' plate with opening

The in-plane maximum principal stress contour plots for the models using the meshing techniques considered in ABAQUS/CAE are shown in the following figures:

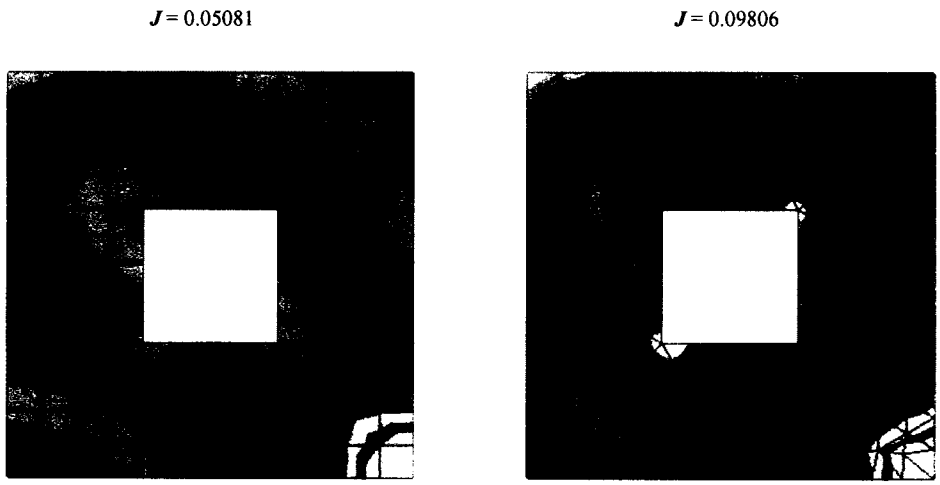
- medial axis and advancing front algorithms – Figure 6.7,
- uniform rectangular and triangular meshing – Figure 6.8,
- rectangular meshing with various bias seeding ratios – Figure 6.9,
- quadrilateral meshing with various boundary seeding – Figure 6.10.

The plots for all these cases are displayed with the same contour limits as the "fine mesh" in-plane maximum principal stress contour plot shown in Figure 6.6.



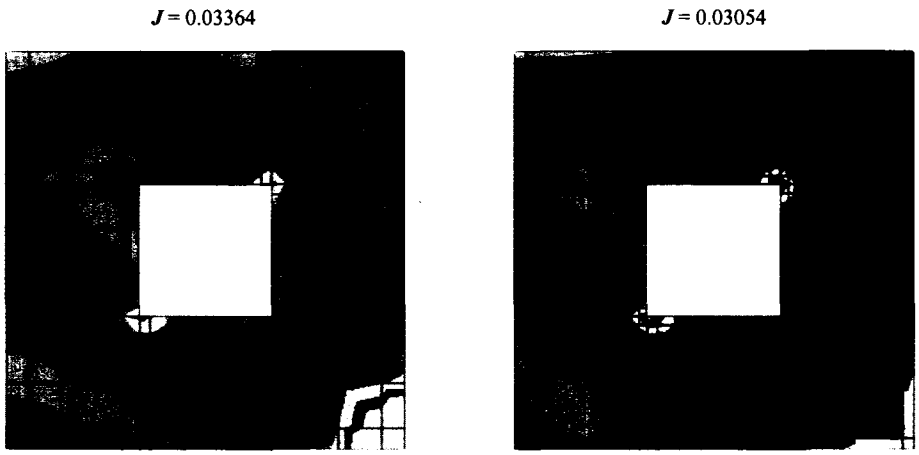
(a) medial axis: 144 elements (b) advancing front: 162 elements

Figure 6.7. Contour of in-plane maximum principal stress using medial axis and advancing front meshing techniques



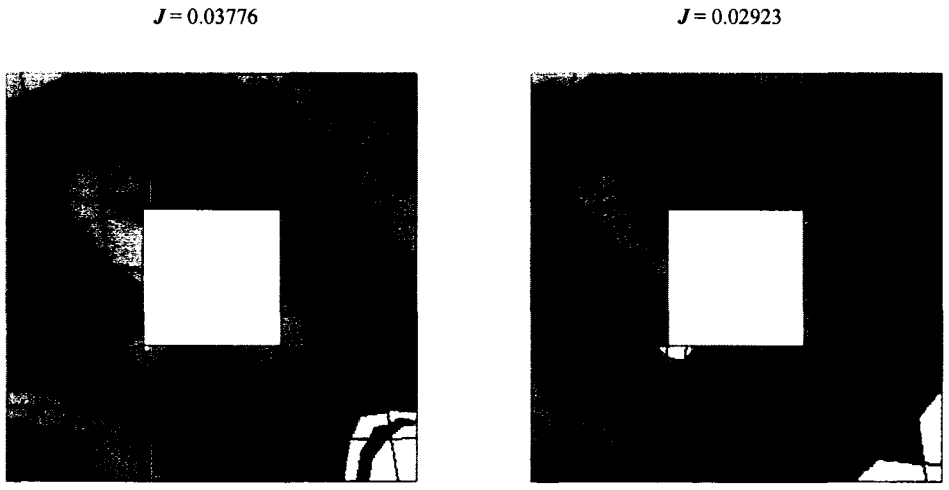
(c) rectangular meshing: 144 elements (d) triangular meshing: 366 elements

Figure 6.8. Contour of in-plane maximum principal stress using rectangular and triangular meshing techniques



(e) lower bias ratios: 144 elements (f) higher bias ratios: 144 elements

Figure 6.9. Contour of in-plane maximum principal stress using rectangular meshing with various bias seeding ratios



(g) more inform seeding: 144 elements (h) more localized seeding: 170 elements

Figure 6.10. Contour of in-plane maximum principal stress using quadrilateral meshing with various boundary seeding locations

The choice of an objective function in an optimization problem should greatly depend on the structure under consideration and the final goal of FE analysis being performed. For example, in some civil structures the concern might be in allowable deformations (displacements) in a structure, or the deformed shapes resulting from fuselage or wing shell buckling might be of interest to aircraft designers. For these cases the error in displacement field might be a good choice for finite element optimization. In medical applications engineers look for maximum strain regions in a device being analyzed. In other structural applications the limitations can be placed, for instance, on bearing capacity and thus on the stress field in a structure, or material plasticity limits could be of relevance. For these types of FE analyses an alternative formulation of an objective function needs to be considered. It could be the error in the stress field either in the energy norm sense or one which imposes uniform error distribution for all elements in the structure (local criteria).

All of these various objective function formulations will result in different optimal solutions using the GAs paradigm, satisfying one or another criterion in the optimization process. If the latter objective function (local stress formulation) is used in a FE mesh optimization, the individuals with more uniform stress distribution in all elements will be chosen for regeneration more often and pass on their genes to next generations. However, the individuals not suited well to the local criteria would eventually drop out as more and more "fit" individuals enter the population. This process will ensure that the final optimal solution has "almost" uniform error distribution for the whole structure.

Chapter 7

Concluding Remarks

In this chapter, possible directions for enhancements are discussed after some concluding remarks.

The prime objective of this study was to introduce a method to minimize discretization error in finite element analysis by changing the nodal locations in a mesh without changing the number of nodes (*r*-refinement). The optimization method used here is based on the recently developed technique of GAs which are better capable of dealing with complex multidimensional problems and finding global minima of complicated objective functions with numerous local minima.

The examples considered clearly show that GAs give very reasonable results even though in the implementation of the optimization the properties of the objective function are completely unknown (i.e. there is no information about the neighborhood of the global minimum). This definitely makes the GAs methodology one of the best alternatives for this type of optimization problems.

Several difficulties and limitations are observed when dealing with two-dimensional structures. The first difficulty is the initialization of a population, that is, the creation of a valid mesh with consistent node numbering from a random set of parameters, preserving the boundary. Second, the dimension of the optimization space rapidly increases as the size of the problem grows, which increases the computation time significantly. The use of functional representation of nodal positions helps to alleviate this problem; however obtaining a valid mesh from a set of parameters is still a significant issue.

Next, the problem of creating a finite element mesh from a set of parameters results in a formulation that defines meshes only of a particular prescribed form. For instance, the meshes constructed in the two-dimensional plate example (see Figure 5.12 and Figure 5.23) can only have fixed predefined number of nodes along each boundary. Also, all those mesh models have the same fixed number of nodes. This clearly limits different mesh types which can be used in finite element mesh optimization problems. As an illustration, two 16-element models are shown in Figure 7.1: (a) a mesh model used in the first of two-dimensional examples in Chapter 5 with fixed number of nodes along each boundary; and (b) a mesh model without a restriction on number of nodes along boundaries. The second mesh type has the same number of elements and, if possible to use, it allows more general form of discretization, thus giving more chances of obtaining optimal and better solutions.

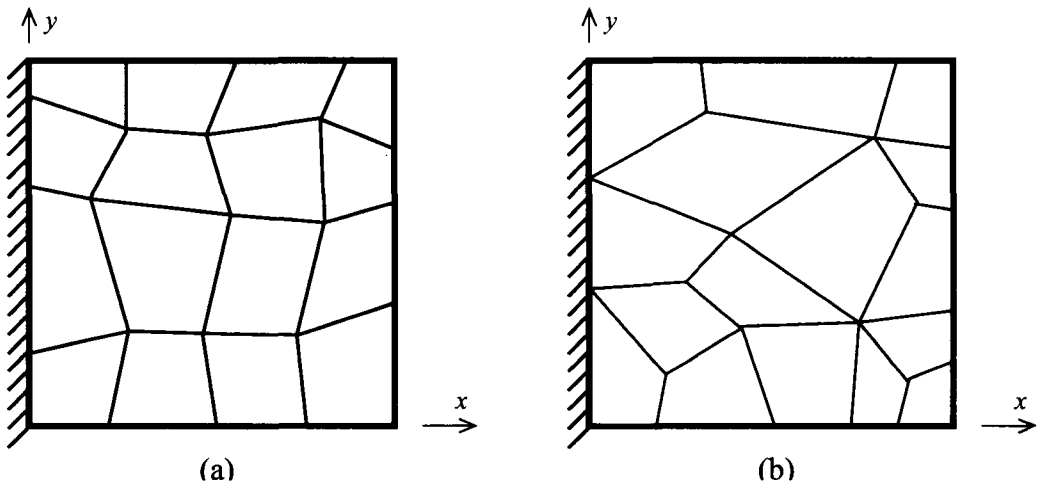


Figure 7.1. Finite element models of a plate: (a) a 16-element mesh with fixed number of nodes along each boundary; (b) a 16-element mesh without a restriction on number of nodes along boundaries

In the case of optimal mesh generation for the purpose of learning or studying the behavior of a particular class of finite element analysis problems, it could be more beneficial to use "fine mesh" solution in objective function calculations. In the process of GAs optimization many hundreds and even thousands of finite element mesh problems are analyzed which covers the bulk of the computational cost. Therefore, replacing the same number of error estimation computations with one single "fine mesh" model analysis and simple error calculations using that solution can result in comparable and even lower computational cost. However, if this proves not to be the case for a particular structural analysis problem, the calculation of the error in the optimality criterion can be approached using Z^2 error estimator (discussed in Chapter 3), thus further reducing computational time.

Finally, the choice of the objective function (optimality criterion) and its variables most definitely influences the resultant optimal mesh. In the considered example of a plate with a square opening, the dependence of the optimality criterion on the displacement field results in optimal meshes which are not obvious even to an "experienced" engineer. These non-obvious optimal finite element meshes prove to be not the better choice in terms of the stress field, but show lower error values in displacement solution which is used in the goal function.

All these issues and their effects on the optimization of a finite element mesh can be considered separately or as a whole. In either case, as this study shows, GAs can be used (and are recommended for use) in finite element mesh optimization of structures.

Suggested Directions for Enhancements

There are several issues that deserve further study in finite element mesh optimization and the GAs methodology. Some of these issues determine possible enhancement directions and applications which can focus on theoretical and practical aspects of the implementation of optimization and mesh generation.

- One of the biggest challenges in the optimization of multidimensional structures is the creation of a valid mesh from a set of optimization parameters. The method of mesh generation used in this study involves mapping a set of parameters to a set of node positions in the structure which are used to create a valid finite element mesh. This

results in the limitations mentioned earlier (Figure 7.1b), and reduce the chances of obtaining the "best" possible mesh.

To avoid this type of limitations, it is suggested that instead of creating a mesh directly from nodal positions the mesh be generated using positions of the centers of elements and their relative size. The size of an element can be associated with length parameters in principal directions (e.g. directions of coordinate axes), area, volume, or any other measure deemed feasible for a reasonable mesh generation. Here it is assumed that the elements have "nice" shapes and forms, i.e. there are no severe shape distortions or large aspect ratios. For these types of elements it is well known that that the location, size and density of elements play quite important role in accuracy of finite element approximation. This method can allow the generation of any type of mesh with the same number of elements.

As an example of this method consider discretization of a two-dimensional plate. First, the given set of optimization parameters is mapped onto the positions of element centers (Figure 7.2a). This mapping can be a "natural" one to one mapping or any other functional correspondence between those two sets as discussed in this work. Then, the size associated with an element center can be decided from the density of other element centers and boundaries around the center. The larger the number of elements (element centers) around the element center being considered and the closer it is to a boundary (external or internal), the higher the density, and thus the smaller the size of that element. The size of an element in this example can be associated with its area.

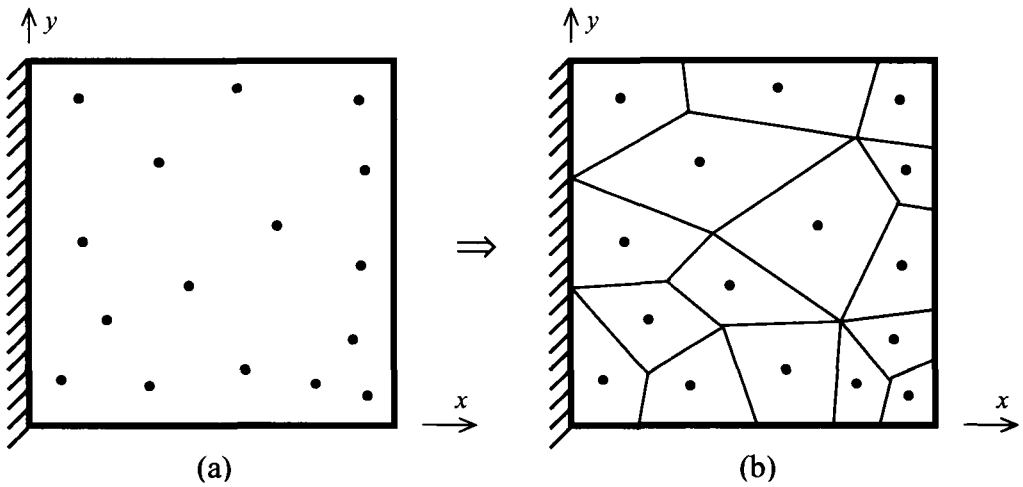


Figure 7.2. Mesh generation using positions of element centers

After the positions of the centers of the elements and their sizes are obtained, the elements can be created by using an existing meshing technique, e.g. medial axis or advancing front meshing algorithms in ABAQUS/CAE. An example of a finite element mesh represented by its element centers is shown in Figure 7.2b.

Even though many different meshes can be generated for the same distribution of element centers using this process, it is believed that the finite element approximation errors for all those meshes are close to each other. Moreover, the more the number of elements in the approximation, the closer the errors are. The only requirement for GAs in this case is that optimization parameters (element center locations) define a unique mesh and thus have only one measure for optimality criterion.

To summarize, the suggested mesh generation method includes the following steps:

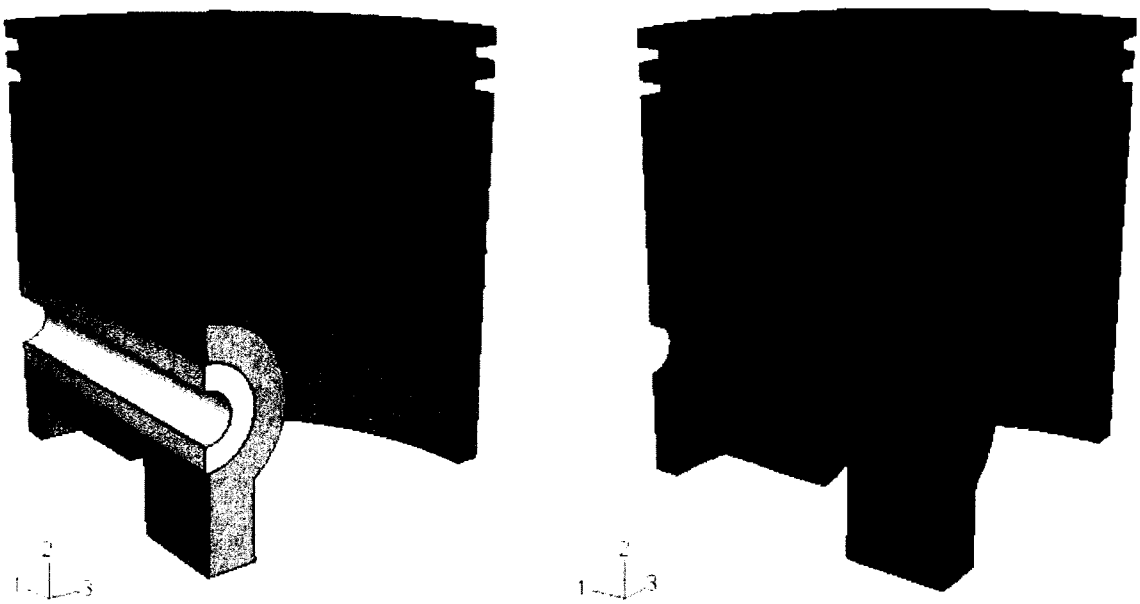
(a) the positions of element centers are obtained from a set of optimization parameters;

(b) the sizes of elements are chosen according to the density of elements and boundaries in the area;

(c) the elements are created in sequence by a meshing algorithm until the finite element mesh is generated.

Though there are numerous alternative ways of generating a mesh, it is the user's choice to implement one or another method depending on a given class of structural geometries.

- The implementation of mesh optimization with GAs in commercial FEA suites can become a powerful tool for solution-based optimal mesh generators. It can also be very advantageous to incorporate existing automatic meshing tools into applications involving GAs optimization. This can be utilized by considering various types of meshes that can be created by commercially available meshing algorithms. Several meshing tools and, in particular, partitioning tools, available for example in ABAQUS/CAE, can be used to partition a complex structural domain (geometry) into several simpler subdomains that can be meshed easily. An example of a piston assembly quarter symmetry model is shown in Figure 7.3. A finite element mesh of the model is generated by first utilizing manual partitioning tools, followed by seeding tools and finally automatic meshing tools in ABAQUS/CAE.



(a) partitions (b) mesh

Figure 7.3. Quarter-symmetry model of piston assembly

The following steps can be used in applying GAs methodology for this type of mesh generation. First, the optimization parameters are defined as seeding locations for the boundary by either a "natural" or a reasonable functional representation. Then the automatic meshing tool generates a mesh with a selected meshing technique, thus ensuring the uniqueness of a mesh defined by a set of parameters, and then the optimization problem is solved. In this process a user can divide the structure into simpler regions in the beginning of the analysis with the help of partitioning tools or let the automatic meshing algorithm make the choice based on user-defined seeding on the external boundary.

- The representation of a finite element mesh by a set of optimization parameters (chromosome) is another very important issue. The implementation of different functional correspondence between a mesh and a chromosome allows the use of fewer optimization parameters in GAs and can possibly ensure that their number does not increase with the increase in problem size. This feature can help to reduce computational time which is very important when dealing with large structural optimization problems.

- Higher computational speeds and lower total run times can be achieved by either increasing computer speeds and/or parallelizing the application. The latter approach seems to be the most promising direction for GAs applications as they are explicitly parallel and substantial computational gains can be expected. The parallelization shows even more potential in view of fast improvements in computing technology, availability of large multiprocessor systems, and most recently, the development of high-performance cluster computing.

The parallelization of a finite element mesh optimization problem can be applied to different parts of the computations: finite element equation solvers and/or generational loop of optimization algorithm itself. These parallelizations can be utilized independently of each other or, with the availability of large number of processors, even simultaneously in the same optimization problem.

In recent years much work has been done in parallel implementations of the equation solvers in commercial FEA software. Parallel execution in ABAQUS/Explicit,

an explicit time integration solver in the ABAQUS FEA suite, is available with the following parallel implementations: a thread-based for shared memory architecture platforms, and an MPI-based for distributed memory architecture platforms. Threads are lightweight processes that can perform different tasks simultaneously (i.e. in parallel) within the same application. Thread-based parallelization in ABAQUS/Explicit is implemented in two ways: the loop-level method which parallelizes low-level loops that are responsible for most of the computational cost; and the domain-level method which splits the model into a number of topological domains that are distributed evenly among the available processors. The Message Passing Interface, or MPI [43], is a middleware used to facilitate interprocessor communication and is implemented using the domain decomposition approach mentioned above. Additionally, there are two parallelization methods implemented for the direct solver in ABAQUS/Standard: "tree" and "supernode" parallelization. The tree parallel solver processes multiple fronts in parallel, in addition to parallelizing the solution of individual fronts. The supernode parallel solver parallelizes the solution of individual fronts. Though these methods produce reasonable speedups in FEA solutions, the full power of available processors is not fully utilized and they almost always leave some of the processors waiting for other processors or I/O communications.

In contrast, the GAs like other techniques of evolutionary computation are highly adaptable to parallelization at essentially 100% efficiency. This creates a potential of achieving considerable gains in computational speed with the use of parallel cluster computers with hundreds and even thousands of processors.

The GAs methodology offers the potential for very high speedup factors with perfect scaling. The speedup factor is referred to the ratio of the total time required to run on a single processor (serial run) to the total time required to run on multiple processors (parallel run). The speedup factor equals the number of processors used for the parallel run in the case of perfect parallelization. Scaling refers to the behavior of the speedup factor as the number of processors is increased. Perfect scaling indicates that the speedup factor increases linearly with the number of processors.

In the case of GAs optimization for finite element mesh generation, the fitness evaluation of each individual in each generation, involving finite element analysis solutions, is the dominant component of computational cost (in terms of computing time), and relatively little computer time is spent on the execution of the Darwinian selection and genetic operations on each generation during the run. These observations give rise to one of the most commonly used approach to parallelization of genetic programming, namely the "asynchronous island" approach to parallelization [42].

In the "island" approach to parallelization of genetic programming, the population for a given run is divided into semi-isolated subpopulations which are assigned to separate processors of the parallel computing system. In the main generational loop of genetic programming, the task of measuring the fitness of each individual is first performed locally at each processor. Then the Darwinian selection step and, subsequently, the genetic operations are performed. Finally, upon completion of a generation, a relatively small percentage of the individuals in each subpopulation are probabilistically selected (based on fitness) for emigration from each processor to various

neighboring processors, and the procedure is repeated again. Since each of these tasks is performed independently and because the processors are not synchronized (each generation starts and ends independently at each processor), this asynchronous island approach to parallelization efficiently uses all the computing power of multi processors.

The parallelization of the Genetic Algorithms, in particular the asynchronous island model for parallelization, delivers an overall increase in the total amount of work performed that is nearly linear with the number of processors, resulting in nearly 100% efficiency. This near perfect efficiency is in marked contrast to the efficiency achieved in parallelizing finite element equation solvers discussed above.

- Knowledge-based rules can be applied to eliminate an invalid or "bad" mesh from the population even before the finite element analysis is attempted for that mesh. These rules can include, but are not limited to:
  (a) element feasibility, severe shape distortion or large aspect ratio;
  (b) tests of zero-energy deformation modes, lack of invariance, and absence of rigid-body motion capability;
  (c) ill-conditioning of the stiffness matrix;
  (d) any prior knowledge about the type of elements (mathematical formulation), their arrangement and the behavior (stiff regions or supports, etc.) of the structure being analyzed.

Appendix. Python Script for Error Calculation in ABAQUS

The Python script for the optimality criterion, J, calculation (Equation 4.4) for FE solutions obtained using ABAQUS is brought below. The filenames point to appropriate ABAQUS analysis output database files. The script opens the results databases and operates on the displacement variables automatically retrieving data for all the nodes in both "fine" and "coarse" mesh models. The computed value of the optimality criterion (`error`) is printed in the message area of ABAQUS/CAE window.

```
from abaqus import *
from abaqusConstants import *
import visualization

fineOdbName = 'fine-mesh.odb'
coarseOdbName = 'course-mesh.odb'

fineOdb = session.openOdb(fineOdbName)

inst = fineOdb.rootAssembly.instances.values()[0]

pointCoordList = []
fineSolution1 = []
fineSolution2 = []
```

```
lastFrame = fineOdb.steps.values()[-1].frames[-1]
disp = lastFrame.fieldOutputs['U']

for val in disp.values:
    node = inst.getNodeFromLabel(label=val.nodeLabel)
    pointCoordList += [node.coordinates]
    fineSolution1 += [val.data[0]]
    fineSolution2 += [val.data[1]]

fineSolution = fineSolution1 + fineSolution2

myPath = session.Path(name='finePoints',
                      type=POINT_LIST,
                      expression=pointCoordList)

coarseOdb = session.openOdb(coarseOdbName)
vp = session.viewports[session.currentViewportName]
vp.setValues(displayedObject=coarseOdb)
vp.odbDisplay.setFrame(step=0, frame=1)
vp.odbDisplay.setPrimaryVariable(variableLabel='U',
                                 outputPosition=NODAL,
                                 refinement=(COMPONENT, 'U1'))
coarseXY = session.XYDataFromPath(name='XYData-1',
                                  path=myPath,
                                  includeIntersections=FALSE,
                                  shape=UNDEFORMED,
                                  labelType=TRUE_DISTANCE)
coarseSolution1 = []
for xy in coarseXY.data:
```

```
    coarseSolution1 += [xy[1]]

vp.odbDisplay.setPrimaryVariable(variableLabel='U',
                                 outputPosition=NODAL,
                                 refinement=(COMPONENT, 'U2'))
coarseXY = session.XYDataFromPath(name='XYData-2',
                                  path=myPath,
                                  includeIntersections=FALSE,
                                  shape=UNDEFORMED,
                                  labelType=TRUE_DISTANCE)
coarseSolution2 = []
for xy in coarseXY.data:
    coarseSolution2 += [xy[1]]

coarseSolution = coarseSolution1 + coarseSolution2

#fineSolsq = map(lambda x: x**2, fineSolution)
denom = 0.
num = 0.
for i in range(len(coarseSolution)):
    denom += fineSolution[i]**2
    num += (fineSolution[i]-coarseSolution[i])**2
error = (num/denom)**0.5
print error

fineOdb.close()
coarseOdb.close()
```

References

[1] Arabyan, A., Chemishkian, S. and Tonoyan, A., (1996). Genetic Algorithms for Finite Elements Model Optimization: A Study of Beam with Static Loads. *Proceedings 3rd APEIE Conference*, 10 (Design of Automatic Devices and Control Systems), Russia, Novosibirsk, 56-57.

[2] Zienkiewicz, O.C. and Zhu, J.Z., (1991). Adaptivity and Mesh Generation. *International Journal for Numerical Methods in Engineering*, 32, 783-810.

[3] Zienkiewicz, O.C. and Zhu, J.Z., (1987). A simple Error Estimator and Adaptive Procedure for Practical Engineering Analysis. *International Journal for Numerical Methods in Engineering*, 24, 337-357.

[4] Zhu, J.Z. and Zienkiewicz, O.C., (1988). Adaptive Techniques in the Finite Element Method. *Commun. Appl. Numer. Methods*, 4, 197-204.

[5] Ainsworth, M., Zhu, J.Z., Craig, A.W. and Zienkiewicz, O.C., (1989). Analysis of the Zienkiewicz-Zhu *a-posteriori* Error Estimator in the Finite Element Method. *International Journal for Numerical Methods in Engineering*, 28, 2161-2174.

[6] Jimack, P.K., (1994). An Optimal Finite Element Mesh for Linear Elastic Structural Analysis. *Advances in Post and Preprocessing for Finite Element Technology,* CIVIL-COMP Ltd., Edinburgh, Scotland, 157-167.

[7] Jung-Ho Cheng, (1993). Adaptive Grid Optimization for Structural Analysis – Geometry-Based Approach. *Comput. Meth. Appl. Mech. Eng.*, 107, 1-22.

[8] Fellipa, C.A., (1976). Optimization of Finite Element Grids by Direct Energy Search. *App. Math. Modeling*, 1, 93-96.

[9] Fellipa, C.A., (1977). Numerical Experiments in Finite Element Grid Optimization by Direct Energy Search. *App. Math. Modeling*, 1, 239-244.

[10] Tang, W.J. and Turke, D.J., (1977). Characteristics of Optimal Grids. *Comput. Meth. Appl. Mech. Eng.*, 11, 31-37.

[11] Babuska, I., Szabo, B. and Katz, I.N., (1981). The *p*-version of the Finite Element Method. *SIAM J. Num. Anal.*, 18, 515-545.

[12] Holland, J.H., (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, Mich.

[13] David E. Goldberg, (1989). *Genetic Algorithms in Search, Optimization & Machines Learning.* Reading, MA: Addison-Wesley Publishing Company, Inc.

[14] Goldberg, D. E., and Richardson, J., (1987). Genetic Algorithms and their Applications. *2nd International Conference on Genetic Algorithms.* Hillsdale, NJ, 151-169.

[15] *Handbook on Genetic Algorithms*, L. Davis, ed., Van Nostrand Reinhold, New York, 1991.

[16] Grefenstette, J.J., (1993). Genetic Algorithms. *IEEE Expert.*

[17] Michalewicz, Z., (1992). *Genetic Algorithms + Data Structures = Evolution Programming.* Springer Verlag.

[18] Cook, R.D., Malkus, D.S. and Plesha, M.E., (1989). *Concepts and Applications of Finite Element Analysis.* Third edition, John Wiley & Sons, Inc.

[19] Zienkiewicz, O.C. and Taylor, R.L., (1989). *The Finite Element Method, Vol 1.* 4th edition, McGraw Hill, New York.

[20] Brauchli, H.J. and Oden, J.T., (1971). On the Calculation of Consistent Stress Distribution in Finite Element Applications. *Int. J. Numer. Methods Eng.*, 3, 317-325.

[21] Hinton, E. and Campbell, J., (1974). Local and Global Smoothing of Discontinuous Finite Element Functions Using a Least Square Method. *Int. J. Numer. Methods Eng.*, 8, 461-480.

[22] Zhu, J.Z., Hinton, E. and Zienkiewicz, O.C., (1993). Mesh Enrichment Against Mesh Regeneration using Quadrilateral Elements. *Communications in Numerical Methods in Engineering*, 9, 547-554.

[23] Babuska, I. and Rheinboldt, W.C., (1980). Reliable Error Estimation and Mesh Adaptation for Non-Linear Mechanics. *Computational Method in Nonlinear Mechanics*, J. T. Oden (ed.), North Holland, Amsterdam, 67-108.

[24] Bank, R. E., (1983). The Efficient Implementation of Local Mesh Refinement Algorithm. *Adaptive Computational Methods for Partial Differential Equations*, I. Babuska *et al.* (Eds.), SIAM, Philadelphia, 74-85.

[25] Rheinboldt, W.C. and Mesztenyi, Ch. K., (1980). On a Data Structure for Adaptive Finite Element Mesh Refinements. *ACM Trans. Mathematical Software*, 6(2), 166-187.

[26] Gago, J. P. R., Kelly, D. W., Zienkiewicz, O.C. and Babuska, I., (1983). A Posteriori Error Analysis and Adaptive Process in the Finite Element Method, Part II: Adaptive Mesh Refinement *Int. J. Numer. Methods Eng.*, 19, 1621-1656.

[27] Zhu, J.Z., Zienkiewicz, O.C., Hinton, E. and Wu, J., (1991). A New Approach to the Development of Automatic Quadrilateral Mesh Generation. *Int. J. Numer. Methods Eng.*, 32, 849-868.

[28] Zhu, J.Z. and Zienkiewicz, O.C., (1990). The Three R's of Engineering Analysis and Error Estimation and Adaptivity. *Comput. Methods Appl. Mech. Eng.*, 82, 95-113.

[29] Zienkiewicz, O.C. and Zhu, J.Z., (1992). Superconvergent Patch Recovery Techniques and a Posteriori Error Estimation, Part I: The Recovery Technique. *Int. J. Numer. Methods Eng.*, 33, 1331-1364.

[30] Zienkiewicz, O.C. and Zhu, J.Z., (1992). Superconvergent Patch Recovery Techniques and a Posteriori Error Estimation, Part II: Error Estimates and Adaptivity. *Int. J. Numer. Methods Eng.*, 33, 1365-1382.

[31] Chapman, C. D., Saitou, K. and Jakiela, M. J., (1994) Genetic Algorithms as an Approach to Configuration and Topology Design. *Journal of Mechanical Design*, ASME, 116, 1005-1012.

[32] Adeli, H. and Cheng, N. T., (1993). Integrated Genetic Algorithm of Space Structures. *Journal of Aerospace Engineering*, ASCE, 6(4), 315-328.

[33] Adeli, H. and Cheng, N. T., (1994). Concurrent Genetic Algorithms for Optimization of Large Structures. *Journal of Aerospace Engineering*, ASCE, 7(3), 276-295.

[34] Rajeev, S. and Krishnamoorthy, C. S., (1992). Discrete Optimization of Structures Using Genetic Algorithms. (Discussion by L. Schmid), ASCE, 118(5), 2494-2496.

[35] *ABAQUS/CAE User's Manual.* ABAQUS Inc., 2003

[36] *ABAQUS Analysis User's Manual. Volume I: Introduction, Spatial Modeling, Execution & Output.* ABAQUS Inc., 2003

[37] *ABAQUS Analysis User's Manual. Volume II: Analysis*. ABAQUS Inc., 2003

[38] *ABAQUS Analysis User's Manual. Volume III: Materials*. ABAQUS Inc., 2003

[39] *ABAQUS Analysis User's Manual. Volume IV: Elements*. ABAQUS Inc., 2003

[40] *ABAQUS Analysis User's Manual. Volume V: Prescribed Conditions, Constraints & Interactions*. ABAQUS Inc., 2003

[41] *ABAQUS Scripting User's Manual*. ABAQUS Inc., 2003

[42] John R. Koza, Forrest H. Bennett III, Forrest H. Bennett, David Andre, (1999). *Genetic Programming III: Darwinian Invention and Problem Solving*. Morgan Kaufmann Publishers.

[43] Gropp, W., Lusk, E., and Skejellum, A., (1994). *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. The MIT Press, Cambridge, Mass.

[44] *Users Guide to the PGAPack Parallel Genetic Algorithm Library*. D. Levine, Argonne National Laboratory, 1996.