FINITE ELEMENT MODELING OF ELECTROMAGNETIC SCATTERING PROBLEMS VIA HEXAHEDRAL EDGE ELEMENTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ASIM EGEMEN YILMAZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JULY 2007

Approval of the thesis:

FINITE ELEMENT MODELING OF ELECTROMAGNETIC SCATTERING PROBLEMS VIA HEXAHEDRAL EDGE ELEMENTS

submitted by **ASIM EGEMEN YILMAZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen ____________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen ____________
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Mustafa Kuzuoğlu ____________
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Gönül Turhan Sayan _____________
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Mustafa Kuzuoğlu _____________
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Adnan Köksal _____________
Electrical and Electronics Engineering Dept., Hacettepe University

Prof. Dr. Gülbin Dural _____________
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. Özlem Aydın Çivi _____________
Electrical and Electronics Engineering Dept., METU

Date: 20.07.2007

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Asım Egemen YILMAZ

Signature :

ABSTRACT

FINITE ELEMENT MODELING OF ELECTROMAGNETIC SCATTERING PROBLEMS VIA HEXAHEDRAL EDGE ELEMENTS

YILMAZ, Asım Egemen
Ph.D., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Mustafa KUZUOĞLU

July 2007, 233 pages

In this thesis, quadratic hexahedral edge elements have been applied to the three dimensional for open region electromagnetic scattering problems. For this purpose, a semi-automatic all-hexahedral mesh generation algorithm is developed and implemented. Material properties inside the elements and along the edges are also determined and prescribed during the mesh generation phase in order to be used in the solution phase. Based on the condition number quality metric, the generated mesh is optimized by means of the Particle Swarm Optimization (PSO) technique. A framework implementing hierarchical hexahedral edge elements is implemented to investigate the performance of linear and quadratic hexahedral edge elements. Perfectly Matched Layers (PMLs), which are implemented by using a complex coordinate transformation, have been used for mesh truncation in the software. Sparse storage and relevant efficient matrix ordering are used for the representation of the system of equations. Both direct and indirect sparse matrix solution methods are implemented and used.

Performance of quadratic hexahedral edge elements is deeply investigated over the radar cross-sections of several curved or flat objects with or without patches. Instead of the de-facto standard of 0.1 wavelength linear element size, 0.3-0.4 wavelength quadratic element size was observed to be a new potential criterion for electromagnetic scattering and radiation problems.

Keywords: All-Hexahedral Mesh Generation, Finite Element Method, Hierarchical Hexahedral Edge Elements, Optimization Based Mesh Smoothing, p-Extension.

ÖZ

ELEKTROMANYETİK SAÇILMA PROBLEMLERİNDE ALTIYÜZLÜ KENAR ELEMANLARI İLE SONLU ELEMAN MODELLEMESİ

YILMAZ, Asım Egemen
Doktora, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Mustafa KUZUOĞLU

Temmuz 2007, 233 sayfa

Bu tezde, ikinci derece altıyüzlü kenar elemanları açık bölge elektromanyetik saçılma problemlerinde uygulanmıştır. Bu amaçla, yarı otomatik bir tamamen altıyüzlü elemanlı ağ üretme algoritması geliştirilmiş ve gerçekleştirilmiştir. Elemanların içinde ve kenarlar boyunca materyal özellikleri, çözüm aşamasında kullanılmak üzere ağ üretimi esnasında belirlenmiş ve tanımlanmıştır. Üretilen ağ, Partikül Sürü Optimizasyon tekniği ile durum kalite metriğine dayalı olarak iyileştirilmiştir. Doğrusal ve ikinci derece altıyüzlü elemanların performanslarını incelemek amacıyla hiyerarşik kenar elemanlarını gerçekleyen bir yazılım çerçevesi geliştirilmiştir. Kompleks koordinat dönüşümü ile gerçekleştirilmiş olan Tamamen Eşlenmiş Katmanlar, bu yazılım kapsamında ağ sonlandırımı işlevini yerine getirmektedir. Denklem sistemini ifade etmek için seyrek matris depolama ve verimli matris düzenleme yöntemleri kullanılmıştır. Seyrek matris çözümü için doğrudan ve dolaylı matris çözüm yöntemleri uygulanmıştır.

İkinci derece altıyüzlü kenar elemanlarının performansı, üzerinde yama bulunan veya bulunmayan çeşitli düz veya kavisli cisimlerinin radar ara kesit yüzeyleri hesaplanarak incelenmiştir. Doğrusal elemanlar için bilinen 0.1 dalgaboyu eleman büyüklüğüne karşılık, ikinci derece elemanlar için 0.3-0.4 dalgaboyu büyüklüğünün elektromanyetik saçılım ve ışıma problemlerinde yeni bir kriter olabileceği değerlendirilmiştir.

Anahtar Kelimeler: Tamamen Altıyüzlü Elemanlı Ağ Üretme, Sonlu Elemanlar Metodu, Hiyerarşik Altıyüzlü Kenar Elemanları, Optimizasyon Tabanlı Ağ İyileştirme, *p*-Artırımı.

To My Family

ACKNOWLEDGEMENTS

I have to mention a couple of key people who had great influences on my academic life; and hence during the preparation of thesis directly and/or indirectly.

First of all, starting with my academic life in general; I would like to thank Assoc. Prof. Dr. Melek YÜCEL and Prof. Dr. Canan TOKER, from whom I took several graduate and undergraduate courses. In addition to their excellent teaching abilities; with their personalities and attitudes towards the students, they are the two major people who made me love electrical and electronics engineering first.

Another important figure is Prof. Dr. İsmail Ş. GÜLOĞLU from METU Department of Mathematics (who is currently retired but continuing to work at Doğuş University Department of Mathematics). I would like to thank him for providing me such a great mathematical background; from which I am not sure how much I could have absorbed.

Certainly, Prof. Dr. Kemal LEBLEBİCİOĞLU is the person who taught me to investigate the physical phenomena with a multidisciplinary point of view especially during my undergraduate term project. I would like to express my thankfulness to him not only for this fact, but also for directing and introducing me to Prof. Dr. Mustafa KUZUOĞLU for my graduate level research.

Regarding this thesis, I have to start with my dear friend and colleague B. Barış DÜNDAR, who has put the very initial effort to hexahedral edge element coding in the late 1990s. The codes that he developed are so robust and compact that, even today I still reuse some portions which he wrote. For about 8-9 years, I have not even experienced any crashes for the parts he developed. Without such a great infrastructure, the code development would inevitably be more painful for me.

I owe much gratitude to Prof. Dr. Adnan KÖKSAL (from Hacettepe University Department of Electronics Engineering) and Prof. Gönül TURHAN SAYAN for their participation to my Thesis Steering Committee. Even in my most inefficient and lazy semesters, they did not give up providing their encouraging and helpful comments at any moment.

Naturally, the greatest part of my gratitude goes to Prof. Dr. Mustafa KUZUOĞLU for his valuable supervision and support throughout the development and improvement of this thesis. In addition to his excellent guidance and infinite patience, he has been an idol to me with his wisdom and his multidisciplinary point of view. In my vita, being his student throughout the M. Sc. and Ph. D. theses is one of the most important items, of which I am very proud. I hope that we will have some other opportunities of working together after the completion of this thesis.

Finally, I have to express my gratefulness to my parents and my sisters for always supporting me and providing me such a comfortable life (sometimes even spoiling me) during my whole life of education. I would like to thank my wife for her support and tolerance to my static and negative energy radiating nights in front of the computer during the development of this thesis.

TABLE OF CONTENTS

ABSTRACT IV

ÖZ V

ACKNOWLEDGEMENTS VII

TABLE OF CONTENTS IX

LIST OF TABLES XIII

LIST OF FIGURES XIV

CHAPTER

1. INTRODUCTION 1

1.1. Electromagnetic Scattering 1
  1.1.1. Numerical Methods 1
  1.1.2. The Finite Element Method 2

1.2. Summary of the Present Work 4

2. HEXAHEDRAL MESHING 10

2.1. Hexahedral Meshing: Reasons and Challenges 10

2.2. Topological Existence of Hexahedral Mesh 13

2.3. Hexahedral Meshing Algorithms 14
  2.3.1. Sweep Methods 15
  2.3.2. Overlay Grid Methods 16
  2.3.3. Mesh Partitioning Methods 18
  2.3.4. Advancing Front Methods 19

2.4. Hexahedral Mesh Quality Improvement 22
  2.4.1. Mesh Refinement 22
  2.4.2. Mesh Smoothing 25

2.5. Hexahedral Mesh Quality Metrics 29

2.6. Curvilinearization - Extension of Quality Definitions for Curvilinear Mesh 31
  2.6.1. Mapping Functions 32
    2.6.1.1. Linear Mapping 32

    - 2.6.1.2. Quadratic Parametric Mapping....32
    - 2.6.1.3. Mapping by the Blending Function Method....34
    - 2.6.1.4. Bézier curves....36
  - 2.6.2. Extension of Mesh Quality in Curvilinear Elements....38
    - 2.6.2.1. Validity Rather Than Quality....39
    - 2.6.2.2. Effects of Element Distortion and Jacobian....40

3. HEXAHEDRAL EDGE ELEMENTS....42

3.1. Differential Forms and Algebraic Manifolds....42

3.2. Hilbert Spaces Related to Electromagnetic Quantities....45

3.3. General Idea of Edge Elements....48

3.4. The Linear and Quadratic Hexahedral Edge (van Welij's and Kameari's) Elements....50

3.5. Hierarchical Finite Element Methods....52

3.6. General Formulation of *p*-Hierarchical Hexahedral Edge Elements....54

4. FINITE ELEMENT FORMULATION OF ELECTROMAGNETIC SCATTERING PROBLEMS....60

4.1. Weak Formulation of the Electric Field....60

4.2. Mesh Generation....63

4.3. Mesh Quality Improvement....63

4.4. Mesh Truncation....63
  - 4.4.1. Analytical Investigation of PMLs....64
    - 4.4.1.1. PMLs for Cartesian Coordinate System....64
    - 4.4.1.2. Modelling of Edge and Corner Regions of the PML....66
    - 4.4.1.3. Conformal PMLs....68

4.5. Elemental Matrix Construction....68

4.6. Sparsity and Resource Requirements in the Finite Element Method....71
  - 4.6.1. Sparse Matrix Storage Schemes....71
  - 4.6.2. Sparsity Issues for *p*-Hierarchical Quadrilateral Edge Elements....73
  - 4.6.3. Sparsity Issues for *p*-Hierarchical Hexahedral Edge Elements....77
  - 4.6.4. Resource Requirements....80

4.7. Sparse Matrix Solvers....82
  - 4.7.1. General Assessment About Sparse Matrix Solvers....82
  - 4.7.2. Indirect Sparse Matrix Solvers....84
    - 4.7.2.1. Biconjugate Gradient Method....85
  - 4.7.3. Direct Sparse Matrix Solvers....88
    - 4.7.3.1. Multifrontal Method....88

4.8. Effects of Node/Edge Ordering During Matrix Solution....93

4.9. Radar Cross-Section and Huygens’ Equivalence Principle 101

5. NUMERICAL RESULTS...105

5.1. Results for Homogenous Scatterers 105
    5.1.1. Results for Uncurved Homogenous Scatterers...105
        5.1.1.1. Permeable Cube...105
        5.1.1.2. Composite Cube 106
    5.1.2. Results for Curved Homogenous Scatterers...107
        5.1.2.1. PEC Sphere...107
        5.1.2.2. Dielectric Sphere 110
        5.1.2.3. PEC Spheroid 111
        5.1.2.4. Dielectric Cylinder 114

5.2. Results for Scatterers with Patches 115
    5.2.1. Results for Uncurved Scatterers with Patches...115
        5.2.1.1. Unloaded Rectangular Microstrip Patch..116
        5.2.1.2. Singly-Loaded Rectangular Microstrip Patch 117
        5.2.1.3. Mutiply-Loaded Rectangular Microstrip Patch 118
    5.2.2. Results for Curved Scatterers with Patches..119
        5.2.2.1. Circular PEC Patch Above a Dielectric Cylinder.....................................119
        5.2.2.2. Circular PEC Patch Above a Dielectric Coated Sphere 120

5.3. General Discussions About the Results 123

6. AN OBJECT AND PATTERN ORIENTED APPROACH IN THE FINITE ELEMENT SOFTWARE DEVELOPMENT 125

6.1. Object Oriented Methodology and Software 125
    6.1.1. User’s Point Of View 125
    6.1.2. Design Considerations 127
    6.1.3. Object Oriented Languages and C++..128
    6.1.4. Standard Template Library of C++ 131
    6.1.5. Migration to Object Oriented Methodology in FEM: FEM++ ?......................133
    6.1.6. Design Patterns in FEM 133

6.2. An Object and Pattern Oriented Finite Element Software Proposal 134
    6.2.1. Architectural Decisions..135
    6.2.2. UML Analysis of the Proposed Architecture...137

7. CONCLUSIONS...145

REFERENCES..148

APPENDIX

A. SCIENTIFIC CONTRIBUTIONS 160

A.1. Great Contributors...160

A.2. Historical Milestones in the Finite Element Theory..163

B. EXPLICIT BASIS FUNCTIONS AND INTERPOLATION PROPERTIES OF VAN WELIJ AND KAMEARI ELEMENTS 164

B.1. The Hexahedral Edge Element Shape Functions 164

B.2. Interpolation Properties of the Linear Hexahedral Edge Elements 170

B.3. Interpolation Properties of the Quadratic Hexahedral Edge Elements....................173

C. DOMAIN DECOMPOSITION IN HEXAHEDRAL MESH GENERATION..........181

C.1. Domain Decomposition of Cylindrical Domains 181
    C.1.1. Domain Decomposition for the PEC Cylinder Problem.................................181
    C.1.2. Domain Decomposition for the Dielectric Cylinder Problem 183
    C.1.3. Domain Decomposition for the PEC Sphere Problem....................................184
    C.1.4. Domain Decomposition for the PEC Sphere Problem....................................187

C.2. Restrictions to Satisfy the “All-Hexahedra” Condition..192

D. A PARTICLE SWARM OPTIMIZATION APPROACH IN HEXAHEDRAL MESH SMOOTHING 193

D.1. An Algebraic Hexahedral Mesh Quality Metric 193

D.2. The Condition Number Based Objective Function 196

D.3. Particle Swarm Optimization (PSO) 197

D.4. Adaptation of PSO and Derivatives to Mesh Smoothing.......................................201

D.5. Speeding up the Convergence of PSO for High *D.O.F* Problems 221

D.6. Pareto Optimality in Mesh Smoothing and Application of MOPSO 225

E. BASIC NOTATION OF THE UNIFIED MODELING LANGUAGE (UML).........227

E.1. What is UML?..227

LIST OF TABLES

Table 2.1. Basic quality metrics for hexahedral elements. 30
Table 2.2. Comparison of Restricted and General Quality Metrics [73]. 31
Table 3.1. Domains and Ranges of Differential Operators. 45
Table 3.2. Properties of the *p*-forms. 47
Table 3.3. Comparison of *p*, *h*, and *hp*-versions 53
Table 3.4. Comparison of Hierarchical and Lagrange Elements. 54
Table 4.1. Gaussian Quadrature Evaluation Points and Their Weights. 70
Table 4.2. Numbers of Unknowns for the *NL*-Quadrilateral Element Mesh. 73
Table 4.3. Numbers of unknowns for the *NLM*-hexahedral element mesh. 77
Table 4.4. Comparison between linear and quadratic hexahedral edge element schemes. 82
Table 4.5. Overall View of the Sparse Solvers. 82
Table 4.6. Time and Memory Requirements for Direct Solvers [103]. 83
Table 4.7. Time Requirements for Some Indirect Solvers [103]. 84
Table 5.1. Comparison of Each Meshing Scheme (Meshing Parameters and Resource Requirements). 109
Table 5.2. Comparison (resource requirement, element size, solution accuracy, etc.) of linear and quadratic elements for PEC sphere problem. 110
Table 5.3. Backscatter RCS of an Oblate Spheroid (Maxor Axes Fixed at 2λ). 113
Table 5.4. Comparison (Resource Requirement, Element Size) of Linear and Quadratic Elements for Circular PEC Patch on Dielectric Coated PEC Sphere Problem. 122
Table B.1. The position functions ϕ. 166
Table B.2. The nodal shape functions. 169
Table B.3. Error norms for linear and quadratic elements of different sizes 179
Table D.1. Nodal ordering for a hexahedral element. 194
Table D.2. Performance Measures for PSO-Mesh Smoothing of Various Domains. 205
Table D.3. Reduction in the error norms after smoothing. 220
Table D.4. Main Properties of the Benchmark Functions Used in [172]. 222
Table E.1. Core Elements in Structural UML Modeling. 228
Table E.2. Core Relationships in Structural UML Modeling. 229
Table E.3. Core Definitions in UML Use Case Modeling 230

LIST OF FIGURES

Fig. 1.1. Finite Element Solution Process. 4

Fig. 2.1. A Simple Example for BLOB decomposition [13]. 13

Fig. 2.2. A tetrahedron partitioned to four hexahedra [22]. 14

Fig. 2.3. Mesh Generated by a Sweep Method [10]. 15

Fig. 2. 4. Mesh Generated by Overlay Grid Method; and Effect of Orientation in 3D [10]. 16

Fig. 2.5. Mesh Generated by Overlay Grid Method; and Effect of Orientation in 2D. 17

Fig. 2.6. Steps Followed in the Boundary Fit Method [30]. 18

Fig. 2.7. An Example Mesh Generated by the Cooper Tool Using the Mesh Partitioning Approach [10]. 19

Fig. 2.8. Paving [38]. 20

Fig. 2.9. Plastering. 20

Fig. 2.10. Spatial Twist Continuum. 21

Fig. 2.11. Propagation Requirement of Mesh Refinement for Quadrilateral/Hexahedral Mesh. 23

Fig. 2.12. An Example of a Pillow Region [45]. 24

Fig. 2.13. Example of Cleave-and-Fill Refinement [45]. 24

Fig. 2.14. Mesh quality improvement via modification of its connectivity [38]. 25

Fig. 2.15. A Successful Application of Laplacian Smoothing (Convex Region). 26

Fig. 2.16. An Unsuccessful Application of Laplacian Smoothing (Concave Region). 26

Fig. 2.17. Torsional Spring Analogy [38]. 28

Fig. 2.18. Nodal Numbering Scheme for Quadratic Quadrilateral Elements. 34

Fig. 2.19. A Quadrilateral with a Curved Side. 35

Fig. 2.20. Cubic Bézier Blending Functions and a Cubic Bézier Curve [76]. 37

Fig. 2.21. Bézier Curves of Various Orders [76]. 38

Fig. 3.1. Tonti Diagram of Maxwell's Equations. 46

Fig. 3.2. Hexahedral Elements for $H(\boldsymbol{\Omega},\mathrm{grad})$, $H(\boldsymbol{\Omega},\mathrm{curl})$, $H(\boldsymbol{\Omega},\mathrm{div})$ and $L^2(\boldsymbol{\Omega})$ (For simplicity, elements of their first kinds are shown). 48

Fig. 3.3. A Rectangular Edge Element. 49

Fig. 3.4. a) Linear hexahedral element in *xyz*-space; b) Linear hexahedral element transformed to *uvp*-space with its 12 edges shown. 51

Fig. 3.5. a) Quadratic hexahedral element in *xyz*-space; b) Quadratic hexahedral element transformed to *uvp*-space with its 12 edges along *u* shown. 52

Fig. 3.6. A Hexahedral Element in an Orthogonal Coordinate System..56

Fig. 3.7. Legendre Polynomials of Order 0 to 5 Plotted between -1 and 1. 58

Fig. 4.1. Scattering material enclosed by *S(Ω)* and an incident wave E^{inc}. 61

Fig. 4.2. Geometry of the PML region surrounding the scatterer. 67

Fig. 4.3. An *NL*-Quadrilateral Element Mesh of a Rectangular Region..73

Fig. 4.4. A Linear Quadrilateral Edge Element with 4 Nodes and 4 Edges. 74

Fig. 4.5. An Edge Shared by 2 Linear Quadrilateral Edge Elements. 74

Fig. 4.6. A Quadratic Quadrilateral Edge Element with 8 Nodes and 10 Edges............................75

Fig. 4.7. An Edge Shared by 2 Quadratic Quadrilateral Edge Elements..76

Fig. 4.8. An *NLM*-Hexahedral Element Mesh of a Rectangular Prismic Region...........................77

Fig. 4.9. An Edge Shared by 4 Linear Hexahedral Edge (van Welij’s) Elements. 78

Fig. 4.10. An Edge Shared by 4 Quadratic Hexahedral Edge (Kameari’s) Elements. 79

Fig. 4.11. Uniform Meshes in 2D and 3D with *n* elements...83

Fig. 4.12. Pictorial Description of Calculation of Frontal and Update (Contribution) Matrices for the Multifrontal Algorithm. 90

Fig. 4.13. Memory Usage of the Multifrontal Method...90

Fig. 4.14. Effect of the Elimination Tree Structure on Memory Usage [121]................................91

Fig. 4.15. Different Elimination Tree Structures..92

Fig. 4.16. A Comparison of the Matrix Reordering Methods [121]...92

Fig. 4.17. Different numbering schemes for a fixed quadrilateral mesh. 94

Fig. 4.18. Resultant Matrix for the Ordered Numbering Scheme. 94

Fig. 4.19. Resultant Matrix for the Spiral Numbering Scheme..95

Fig. 4.20. Resultant Matrix for the Irrelevant Numbering Scheme...96

Fig. 4.21. Effect of Fill-in for the matrices obtained by different ordering schemes. 97

Fig. 4.22. “Node Coloring” Numbering Scheme. 98

Fig. 4.23. Resultant Matrix After the Node Coloring Scheme. 99

Fig. 4.24. An example for the Linear Quadrilateral Edge Element..100

Fig. 4.25. Another example for the Quadratic Quadrilateral Edge Element. 101

Fig. 4.26. Pictorial Description of the Surface Integration Method. 103

Fig. 5.1. RCS of a Permeable Cube Calculated on Compared to [128]. 106

Fig. 5.2. RCS of a Composite Cube Calculated and Compared to [128]. 107

Fig. 5.3. Meshes of Different Levels of Density. 108

Fig. 5.4. Cross section of the mesh generated for the PEC sphere problem.................................109

Fig. 5.5. RCS of a Dielectric Sphere. 111

Fig. 5.6. Cross-Section of the Mesh Generated for a PEC Prolate Spheroid. 111

Fig. 5.7. Transformation from the PEC sphere mesh to the PEC spheroid mesh.........................112

Fig. 5.8. Sample All-Hexahedral Meshes Generated for...112

Fig. 5.9. Description of the Dielectric Cylinder Problem.....114
Fig. 5.10. RCS of a Dielectric Cylinder Calculated and Compared to [130]. 115
Fig. 5.11. RCS of an Unloaded Rectangular Microstrip Patch (Compared to [133]).....116
Fig. 5.12. RCS of the Same Geometry When Patch Removed (Compared to [133]).....116
Fig. 5.13. Backscatter RCS of a Singly Loaded Rectangular Patch (Compared to [133]).....117
Fig. 5.14. Backscatter RCS of Another Singly Loaded Rectangular Patch (Compared to [133]).....118
Fig. 5.15. Backscatter RCS of a Multiply Loaded Rectangular Patch (Compared to [133]). 119
Fig. 5.16. Circular PEC Disk Above a Dielectric Cylinder.....119
Fig. 5.17. RCS of a PEC Disk on a Dielectric Cylinder Calculated (Compared to [134]).....120
Fig. 5.18. Description of the Circular PEC Patch on Dielectric Coated PEC Sphere Problem.....121
Fig. 5.19. Cross Section of the Mesh Generated for the Circular PEC Patch on Dielectric Coated PEC Sphere Problem.....121
Fig. 5.20. 3D View of the Mesh Generated for the Circular PEC Patch on Dielectric Coated PEC Sphere Problem (Only Scatterer Shown). 122
Fig. 5.21. Radar Cross Section of the circular PEC patch on dielectric coated PEC sphere (Compared to [134]).....123
Fig. 6.1. Layered Architecture of the FEM Software.....136
Fig. 6.2. A Screen Snapshot of OpenGL based GUI.....137
Fig. 6.3. System Boundary of the FEM Software. 138
Fig. 6.4. Mesh Viewing Functionality Provided by the FEM Software.....138
Fig. 6.5. Mesh Manipulation Functionality Provided by the FEM Software. 139
Fig. 6.6. Direct FEM Functionality Provided by the FEM Software. 140
Fig. 6.7. Management Functionality Provided by the FEM Software.....140
Fig. 6.8. Mesh Data Structure inside the FEM Software.....141
Fig. 6.9. 2D Element Data Structure inside the FEM Software. 142
Fig. 6.10. 3D Element Data Structure inside the FEM Software. 142
Fig. 6.11. Curve Data Structure and Abstract Curve Factory inside the FEM Software.....143
Fig. 6.12. Polynomial Data Structure and Abstract Polynomial Factory inside the FEM Software. 144
Fig. 6.13. Quadrature Point Data Structure and Abstract Quadrature Point Factory inside the FEM Software. 144
Fig. B.1. The hexahedral edge element with its edges numbered. 165
Fig. B.2. The Hexahedral Edge Element with its Nodes Numbered. 167
Fig. B.3. Linear hexahedral edge element extending from (-1, -1, -1) to (1, 1, 1) in *uvp* space. 171
Fig. B.4. Effect of the first shape function for the linear hexahedral edge element. 171

Fig. B.5. Effect of the third shape function for the linear hexahedral edge element. 172
Fig. B.6. Effect of the summation of the first and the third shape functions for the linear hexahedral edge element. 172
Fig. B.7. Quadratic hexahedral edge element extending from (-1, -1, -1) to (1, 1, 1) in *uvp* space. 173
Fig. B.8. The disjoint effects and the effect of the summation of the first and second two edges along the *u* direction. 174
Fig. B.9. The disjoint effects and the effect of the summation of the first and second two edges along the *v* direction. 174
Fig. B.0.10. The disjoint effects and the effect of the summation of the first and second two edges along the *p* direction. 175
Fig. B.11. The effect of the ninth edge along the *v* direction. 175
Fig. B.12. The effect of the ninth edge along the *p* direction. 176
Fig. B.13. The effect of the summation of four edges. 176
Fig. B.14. The effect of the summation of the five edges. 177
Fig. B.15. Interpolation of the function $f(v) = 3v^2 + 4v + 2$ at $p = -1$ plane by means of the quadratic shape functions. 177
Fig. C.1. Top and Bottom Parts of the Mesh of PEC Cylinder. 182
Fig. C.2. Mesh Generated for PEC Cylinder Problem. 182
Fig. C.3. Mesh generation scheme for the dielectric cylinder. 183
Fig. C.4. Mesh Generation Scheme for the PEC Sphere Problem. 184
Fig. C.5. Top Hat Inner Part (A). 185
Fig. C.6. Top Hat Outer Shell (B). 185
Fig. C.7. Side Subvolume (C). 186
Fig. C.8. Whole Spherical Shell (2A+2B+8C). 186
Fig. C.9. Mesh generation scheme in the dielectric sphere problem. 187
Fig. C.10. Mesh defining parameters in the dielectric sphere problem. 188
Fig. C.11. The Inner Cylindrical Core (A). 189
Fig. C.12. Top Hats (B). 190
Fig. C.13. The Inner Cylindrical Core Combined with the Top Hats (A+B). 190
Fig. C.14. The Side Cover (C). 191
Fig. C.15. The Whole Sphere After All Parts Combined (A+B+C). 191
Fig. C.16. Top view of the top hat for the illustration of the restriction. 192
Fig. D.1. Pictorial Description of PSO in 2D. 198
Fig. D.2. Boundary Condition/Wall Concept in PSO. 200
Fig. D.3. 2D Pictorial Description of NPSO. 201
Fig. D.4. An all-hexahedral mesh with *NLM* elements: 203

Fig. D.5. PSO-smoothed version of the mesh in Fig. D.4. 207
Fig. D.6. 3D Views of the Circular Microstrip Patch Antenna. 210
Fig. D.7. Dimension Reduction by Means of Symmetry, and Domain Decomposition................211
Fig. D.8. Fixing Nodes, Imposing Node Dependencies and Setting Rules for Individual Node Movements. 211
Fig. D.9. Setting Boundary Conditions (Reflecting Walls) for Floating Nodes. 212
Fig. D.10. A 3D PSO Mesh Smoothing Scheme for Subdomain I of the Circular Microstrip Antenna Problem. 213
Fig. D.11. A 2D PSO Mesh Smoothing Scheme for Subdomain II of the Circular Microstrip Antenna Problem. 213
Fig. D.12. Performed Steps for a 2D+2D PSO Mesh Smoothing Scheme of the Circular Microstrip Antenna Problem. 214
Fig. D.13. Top Hat Sub-Domain to be Smoothed by PSO (Different 3D Views with Different Levels of Detail). 215
Fig. D.14. Improvement in the Mesh by Investigation of a Constant *R* Surface. 215
Fig. D.15. Improvement in the Mesh by 3D View Investigation of One Layer of Top Hat. 216
Fig. D.16. Cross Section of the Mesh Generated for the Perfectly Electric Conductor Sphere Problem. 216
Fig. D.17. Surface and Volume Definitions of the Top Hat. 217
Fig. D.18. Pictorial Description of the Behavior of 2D Griewank Function [176]. 222
Fig. D.19. Pictorial Description of the Behavior of 2D Rastrigin Function [176]. 223
Fig. D.20. Pictorial Description of the Behavior of 2D Rosenbrock Function [176]. 224
Fig. D.21. Pictorial Description of the Behavior of 2D Sphere Function [176]. 224
Fig. D.22. Dominance Relation in a Biobjective Space. 225
Fig. D.23. The Pareto Front of a Set of Solutions in a Biobjective Space. 226

CHAPTER 1

INTRODUCTION

1.1. Electromagnetic Scattering

Electromagnetic scattering has been one of the main subjects of interest especially for military, biomedical and communications research areas. The main aim is to find, or to predict the behavior of the scattered wave, when an electromagnetic wave from some direction at some specific frequency with a specific type of polarization, is incident on a given material.

1.1.1. Numerical Methods

For only a limited number of shapes and material types, it is possible to find an analytical expression for the scattered wave at any point in space. However, for most cases of concern, no analytical solution can be found. In such cases, some numerical methods are applied in order to find the approximate values of the scattered wave only at some space points, but not a functional expression yielding the field values throughout the whole space as in an analytical solution.

The main steps of the numerical solution methods can be summarized as:

i. Discretization of the geometric domain into small simple subdomains called meshes, such as triangles or quadrilaterals in two dimensions, and tetrahedra or hexahedra in three dimensions;
ii. Symbolic expression of the solution within each subdomain by a finite number of parameters;

iii. Combination of the local equations obtained for each subdomain;
iv. Construction of a set of equations describing the whole geometry;
v. Application of the boundary conditions;
vi. Solution of the global equation system to obtain the unknown function.

1.1.2. The Finite Element Method

The Finite Element Method is based on the solution of Maxwell's equations in their differential form. It is a very good technique for modeling complex, inhomogenous structures. The Finite Element Method has been originally developed for static problems of structural mechanics and initially used by mechanical and civil engineers. It was first formulated for use in electromagnetics in 1940s by Courant [1], who first discussed the versatility of piecewise approximations. In the 1950s, Argyris [2] began putting together the many mathematical ideas (domain partitioning, assembly, boundary conditions, etc.) that form the basis of the Finite Element Method for aircraft structural analysis.

The name 'finite element' results from the fact that the domain is represented by a set of 'elements' of fairly simple shape on which the unknown function is approximated. A major advantage of this technique is its flexibility, in other words, the possibility to match the elements to the geometry and physical characteristics of the solution.

In the numerical solution of a partial differential equation, one must approximately express the solution by a finite number of parameters. In other words, a problem with infinite degrees of freedom must be converted to one with finite degrees of freedom. In general, the solution is sought in a given class of functions; hence any function of this class must be expressed in terms of a finite number of parameters. As a second step, the differential operator must be transformed to expressions relating these parameters. If the differential equation is linear, then, in general, the relations among the parameters are also linear. That is, the process leads to a linear system of algebraic equations. However, in this process one has to deal with a large number of parameters, at least tens of thousands for practical cases. To avoid this complexity, it is preferable to implement a numerical method so that the resulting matrix is sparse. This is the case in the finite element method, since the equations preserve the local character of the differential equation, which implies that the system matrix is sparse. Some sparse matrix storage schemes result in O(n) memory requirements, implying that the memory needed for a solution of a Finite Element Method system is proportional to the number of unknowns n, whereas the memory requirement is O(n^2) for full matrix storage schemes in some other numerical methods. Certainly, the advantages of the sparse matrices are not limited to matrix storage. Some sparse matrix solution schemes have down to O(n) complexities (with some restrictions of course), instead of O(n^3) complexity of the Gaussian elimination.

Although the Finite Element Method is very flexible, it has certain drawbacks. In electromagnetic theory, fields extend to infinity in scattering and/or radiation problems. Hence, another issue while using the Finite Element Method in scattering applications is to apply it to open domain problems successfully. For this purpose, methods like absorbing boundary conditions, perfectly matched layers, hybridization with boundary integral methods have been developed.

Another situation in which the Finite Element Method encounters difficulties is the presence of corners in the system. At such corners, field quantities may become singular. Hence, approximate solutions cannot be adequately represented by locally based polynomial expansions.

Since the finite element method is only an approximation, one has to explain the effects of several error sources, some of which are:

- Choice of a finite number of trial functions (i.e. approximating a function in terms of basis functions of a finite dimensional subspace)
- Simplification of the geometry (polynomial approximation of boundaries or material non-uniformities)
- Modification of boundary conditions
- Numerical integration
- Iterative solution of the matrix equation (if applied)
- Roundoff errors due to finite precision arithmetics of the computer during the modeling and the solution of the discrete system

The overall process of finite element solution of electromagnetic scattering problems can be summarized and illustrated as in Fig. 1.1. The details of each task and step are discussed throughout the main text.

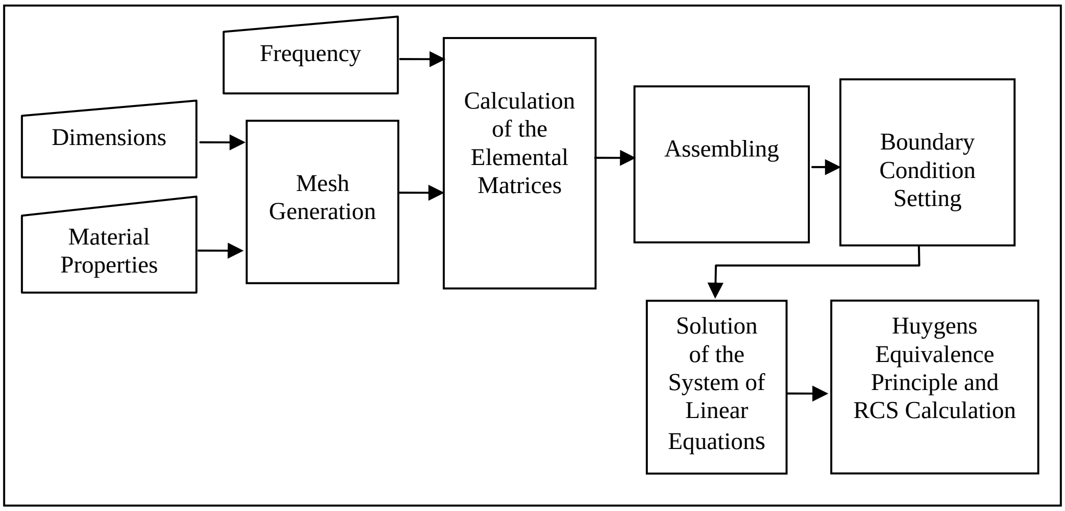

Frequency
Dimensions
Material Properties
Mesh Generation
Calculation of the Elemental Matrices
Assembling
Boundary Condition Setting
Solution of the System of Linear Equations
Huygens Equivalence Principle and RCS Calculation

Fig. 1.1. Finite Element Solution Process.

1.2. Summary of the Present Work

The aim of this thesis is to apply the higher order hexahedral edge elements to electromagnetic scattering problems together with a couple of generic implementations;

- one stand-alone software regarding the all-hexahedral mesh generation,
- another stand-alone software regarding 3D mesh viewing (not limited to hexahedral meshes),
- and the last stand-alone software regarding the finite element solution of three-dimensional electromagnetic scattering problems using the hexahedral edge elements.

Meanwhile, the effect of the hexahedral mesh quality is also investigated; a new topology preserving mesh improvement (i.e. mesh smoothing) technique is implemented and applied to typical meshes to be encountered in electromagnetic problems. This implementation is not a stand-alone software product, but a Matlab script instead.

The all-hexahedral mesh generation algorithm implemented in this thesis depends on the decomposition of the problem domain to subdomains so that each subdomain is homeomorphic to a rectangular prism (i.e. all-hexahedral meshable). Then, each subdomain is meshed with the constraint that the adjacent subdomains have the same quadrilateral surface meshes on the shared surfaces in order to preserve mesh continuity. Another important task performed during the mesh generation is the indication of the material properties inside each element, and along each edge. This activity can be considered as pre-processing for both mesh viewing and finite element solution, since it provides the necessary data for these tasks.

3D mesh viewing software in this thesis is a straightforward implementation which uses a generic graphic library; it requires no complex algorithms. The mesh can be viewed in a colorful manner where each color is assigned to a specific meaning. Principle activities such as zooming in/out, changing the camera angle and position can also be performed during viewing. Mesh viewing software is not limited to volume meshes or hexahedral meshes. Both surface/volume meshes of any element shape (triangular/quadrilateral elements in 2D, tetrahedral/prismic/hexahedral elements in 3D) are supported by this software.

Finite element solution software implemented in this thesis is focused to hexahedral edge elements. The nodal and edge basis functions of first and second order, and their curls are calculated; element matrices are calculated by Gaussian quadrature, assembled global stiffness matrix is stored with a sparse storage scheme and solved by means of a sparse solver (direct or indirect); radar cross-section is calculated by means of Huygens’ surface equivalence principle. During the finite element solution process, mesh truncation is achieved by means of the Perfectly Matched Layers, which are realized by complex coordinate transformation in this thesis. By means of this software, linear and quadratic hexahedral edge elements are compared in terms of resource requirements (CPU time, memory) and accuracy. For this purpose, scatterers of various basic/composite curved/uncurved shaped materials with/without patches are investigated.

The mesh quality is another important factor in the finite element method. The element size is a factor in solution accuracy, since the interpolation error increases as the element size increases; and the element shape is another factor in the solution accuracy, since bad-shaped elements might cause an ill-conditioned stiffness matrix. Mesh improvement can be performed either by changing or preserving the topological connectivity of the mesh. In this thesis, a condition number based combined hexahedral quality metric is used; and the mesh improvement is performed by optimization based mesh smoothing. The smoothing is performed by means of the Particle Swarm Optimization, which found wide application in the last decade.

To the author’s knowledge, there are two original contributions to the literature inside the scope of this thesis:

1. Application of the quadratic hexahedral edge elements to electromagnetic scattering problems,
2. Hexahedral mesh smoothing by means of Particle Swarm Optimization.

Quadratic hexahedral edge (Kameari’s) elements have so far been applied to various problems especially in magnetostatics. The application of such elements to electromagnetic scattering problems is new; research made throughout this thesis yielded:

1. [3], which was limited to uncurved single homogenous scatterers;
2. [4], which was focused to uncurved structures with patches;
3. [5], which was focused to curved single homogenous scatterers;
4. and [6], which was focused to comparison of linear and quadratic hexahedral edge elements in electromagnetic scattering problems.

Optimization based mesh smoothing is one of the popular research areas in the mesh generation society. However, the usage of Particle Swarm Optimization for this purpose is new. Research made throughout this thesis yielded [7] and [8]. Using several objective functions and performing multi-objective mesh smoothing can be considered to be a future work in this area.

In addition to these, several research areas, which are still premature and requiring extra work, are assessed to yield more publications in the future.

One of these is the investigation of the effects of edge ordering to the stiffness matrix storage and solution. Similar work can be found in the literature; but to the author's knowledge none of them specifically focuses on hexahedral edge elements.

Another one is the discussion of the object and pattern oriented finite element software. Object oriented approaches in the finite element software development have become popular in the last decade; but to the author's knowledge none of the available publications mention the usage of design patterns. Moreover, finite element software is a good and compact case study for "software sizing and cost estimation", which is a popular research area triggered by the new World Economy. To the author's belief, such a research and consequent publication(s) will find interest in societies of various disciplines.

The first chapter of this thesis gives a brief introduction to numerical methods and the Finite Element Method; it also gives an overall idea about the activities and the work products of the research carried out.

In the second chapter of this thesis, all-hexahedral meshing is discussed starting from the topological existence. It continues with the meshing algorithms, mesh quality measures, quality improvement techniques, and aspects of curvilinearization. This chapter can be considered as an overall literature survey in "hexahedral meshing" subject.

The third chapter is devoted to the theoretical background of the finite element method, specifically the hexahedral edge elements. It starts with the differential forms and the algebraic manifolds; continues with the definitions of Hilbert spaces containing various important quantities of electromagnetics; introduces the edge element concept; gives a brief discussion of hierarchical elements. The main aim of this chapter is to indicate the positioning of the hexahedral edge elements in the wide finite element universe. Again, this chapter can be considered as a literature survey of the mathematical background in finite element theory.

All formulations (starting from the weak formulation of the electric field, continuing with elemental matrix construction, stiffness matrix assembly, storage and solution algorithms, PML realization, RCS calculation), which are necessary for the finite element solution, are given in the fourth chapter. Analyses about the sparsity, effects of edge ordering, and resource requirements are also included in this chapter. This chapter can be considered as a compact summary of the implementations performed throughout this thesis.

The fifth chapter exhibits the results calculated with the hexahedral edge elements and compares them with the analytical results or measured values. Comparisons between linear and quadratic hexahedral edge elements are also given in this chapter.

The sixth chapter combines the new trends in software development, and discussed the possibilities of application of such new methodologies during the finite element software development.

Finally, the last chapter discusses the results obtained by the proposed technique(s); comments on the achieved results; and lists the potential future work.

In order to preserve the completeness and compactness of the main text, appendices are used for exhibition of some important topics. Appendix A gives a listing of great scientists (starting from Euclid) who could not be explicitly cited but referred throughout the thesis. It also gives the important milestones of the finite element theory and hierarchical hexahedral edge elements.

Appendix B gives the explicit basis functions of the hexahedral edge elements, and exhibits the interpolation properties of these functions inside the relevant elements.

Appendix C gives the methodology and the important factors during the domain decomposition in some problems solved in the thesis.

Appendix D outlines the definition Particle Swarm Optimization, its applicability to hexahedral mesh smoothing, application to some problems and the solution accuracy. This appendix also discusses the concept of Pareto optimality, and multi-objective hexahedral mesh smoothing.

Appendix E includes basic Unified Modeling Language notation in order to ease the reading of the sixth chapter.

CHAPTER 2

HEXAHEDRAL MESHING

2.1. Hexahedral Meshing: Reasons and Challenges

With the introduction of various numerical simulation techniques, analysis of partial differential equations describing complex physical systems became a practical reality. These techniques, including finite element method, finite difference time domain method, and finite volume method, rely on a discretization of the domain as the key to application of the numerical solution. This discretization provides a set, or mesh, of geometrically simple elements that as a whole approximate the complexity of the domain.

All-hexahedral meshes have proven to be desirable because of the facts that:

- A hexahedron provides shape functions with additional terms that may increase the accuracy of the solution; a study about the nodal element case can be found in [9].
- A hexahedron provides directional sizing without losing accuracy. For example, a very thin hexahedron within a boundary layer for fluid flow calculations performs far better than thin tetrahedron.
- A hexahedral mesh decreases the total element number; and the total number of unknowns consequently. A tetrahedral mesh usually increases the element number about 4 to 10 times compared to a hexahedral mesh.
- For especially man-made objects, a quadrilateral/hexahedral mesh provides better surface/volume representation compared to a triangular/hexahedral mesh.

Such conformity in decreases the local errors especially at the material interfaces.

Certainly, topological complexity and challenge in all-hexahedral mesh generation is another factor making it a center of attraction in the last decade. As stated by Blacker [10], automated all-hexahedral element meshing has been the *"Holy Grail"* of mesh generation research for years due to tight constraints such as connectivity and shape.

In general, a mesh generation scheme should have the following features (which are pair-wise contradictory in most cases):

1. *Geometric Generality:* Since the main aim is to have an automated use of a proposed algorithm, it should be able to handle as large a class of geometries as possible. Ideally, it should handle any geometry of arbitrary complexity and detail. It should also be sensitive to surface curvature and meshing domains with widely varying boundary proximity (i.e. long, thin regions versus blocky regions) adequately.

2. *Geometric Matching:* The mesh generated by the algorithm should contain the geometric features identified by the user. In practice, most meshing software defines this to be all the topological features of domains being meshed. This allows the user to control the mesh (e.g. for boundary conditions) by editing the topology, either manually or automatically [11] as needed.

3. *Boundary Sensitivity:* The boundary of the domain is often most important in an analysis, since most differential equations currently being solved in some engineering applications relate to stress/strain, flows or reactions. Thus, in order to define a meshing algorithm, which is going to be accepted as "good" by the mesh generation society, it should produce high quality elements close to the boundary and these elements should roughly follow the flow of the boundary. Element quality interior to the domain is usually less important.

4. *Orientation Insensitivity:* The orientation of the geometry should ideally not affect the generated mesh. This removes any dependency on volume placement

before meshing. Otherwise, good quality of the mesh and acceptable shape of the resulting elements will be chance driven; might not be achieved even by using trial-error method.

5. *Bad Geometry Tolerance:* The algorithms that can operate in spite of irritating gaps, overlaps, holes, and other problems in the geometry (often imported from various formats) save a lot of time and frustration for the user. Being able to de-feature or ignore insignificant detail would also be advantageous.

6. *Size Controllability:* The mesh should be able to match desired element sizing constraints throughout the domain. This is particularly important for adaptive analyses.

7. *Speed:* The algorithm should be able to generate reasonably large meshes in a reasonable "interactive" amount of time. Certainly, speeds obtained by tetrahedral meshing algorithms would be desirable; which are observed and accepted to be fast enough. With recent technology (considering CPU speed, operating system and hardware infrastructure and memory capacity), meshes with <1 million elements are considered to be small / mid sized.

Hexahedral mesh generation algorithms, which have been proposed so far, have usually failed especially due to requirements 1 and 5 above (geometric generality and bad geometry tolerance). Initial schemes used to suffer also from requirement 4 (orientation insensitivity), which used to be a big challenge more than 10 years ago.

Since geometric generality and bad geometry tolerance are the biggest hurdles, the general strategy in hexahedral meshing has become as follows:
1) Development/proposal of the hex meshing algorithm,
2) Investigation of the restrictions of the proposed algorithm (in terms of applicability to various geometries of different topological features)
3) Proposal of topological mapping/domain decomposition techniques in order to fit all (at least, most) geometries to the proposed hexahedral meshing algorithm [12].

Automated domain decomposition techniques attempt to decompose the volume being meshed into pieces, which themselves can be meshed using existing algorithms. Usually these pieces are recognizable primitive shapes or swept volumes. This approach is highly dependent on being able to identify and/or decompose the geometry appropriately. A very straightforward and practical method (due to easy applicability over the CAD modeled objects) is the extraction of Basic Logical Object Blocks (also known as BLOBs) [13]. Principally, the BLOBs are decomposed to Multiple Block Structures (MBSs), which will be considered separately during the mesh generation with the constraint that shared faces of MBSs will have the same surface mesh. This procedure is illustrated in Fig. 2.1. Since BLOB decomposition has limited applicability, more complicated topographic tools such as medial axis determination, Delaunay tessellation, embedded Voronoï diagrams/graphs are widely used for general purpose applications [14-17].

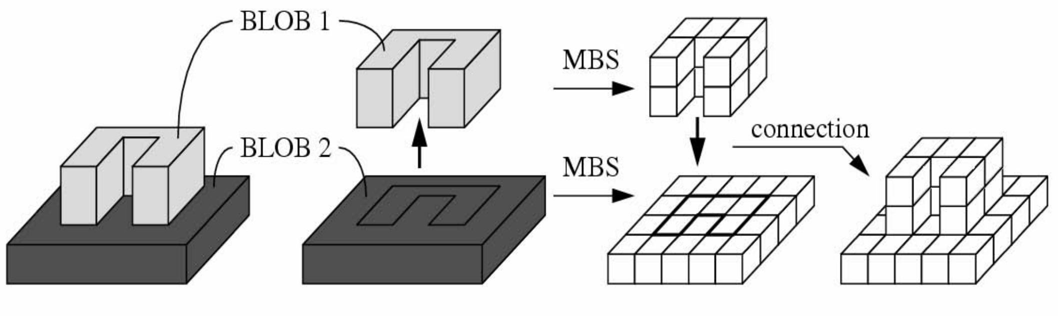


Fig. 2.1. A Simple Example for BLOB decomposition [13].

On the other hand, due to several difficulties and restrictions in the all-hexahedral meshing, some researchers thought that "all-hexahedral" requirement is not worth to try desperately; and hence migrated to "hex-dominant" meshes instead [18].

2.2. Topological Existence of Hexahedral Mesh

Up to 1996, the topological existence of hexahedral mesh was a big question mark. Without knowing the necessary and sufficient conditions for existence of an all-hexahedral mesh for an arbitrary volume, many researchers intuitively tried to start with a surface mesh on the boundary, and fill the volume by a hexahedral mesh touching to the boundary. Many of the researchers were mislead that there are complicated constraints on the surface mesh in order to be compatible with the volume mesh inside.

The answer(s) came from Thurston [19] and Mitchell [20-21] independently. Eppstein [22] stated their answer(s) in a clean and neat manner as: "*Any simply connected three-dimensional domain with an even number of quadrilateral boundary faces can be partitioned into a hexahedral mesh respecting the boundary*."

Starting from this fact, Eppstein claimed that any tetrahedron can be divided into four hexahedra as seen in Fig. 2.2; which yielded a straightforward generic hexahedral meshing algorithm. This algorithm is obviously inefficient; but more important thing about this work is that Eppstein proved that hexahedral mesh generation is a process of linear complexity.

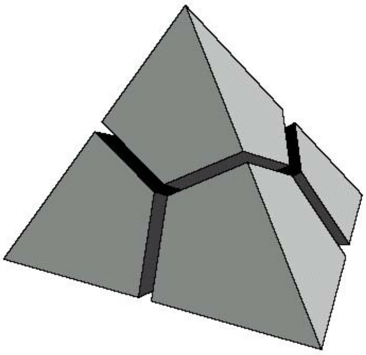


Fig. 2.2. A tetrahedron partitioned to four hexahedra [22].

2.3. Hexahedral Meshing Algorithms

Hexahedral meshing algorithms in the literature can be roughly classified into the following types:

- Sweep

- Overlay Grid
- Mesh Partitioning
- Advancing Front

2.3.1. Sweep Methods

General sweep algorithms [23-25] allow the sweeping of a surface quadrilateral mesh through an arbitrary path. This surface mesh maintains a constant topology cross-section during the sweep, but may deform as needed to match the path geometry. This generalized sweep is heavily used in practice as it allows more freedom since the cross-sectional mesh can be unstructured, as shown in Fig. 2.3. The advantages and disadvantages of this approach can be summarized as follows:

Advantages:

- No need for geometric decomposition,
- Speed,
- High element quality,
- Boundary sensitivity, and
- Orientation insensitivity.

Disadvantages:

- Relatively small applicable class of geometries.

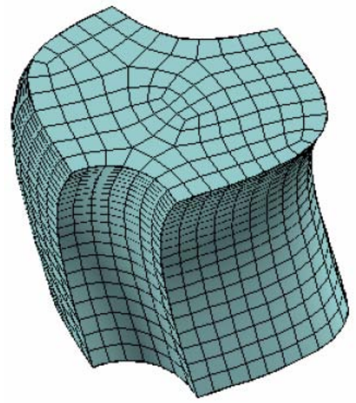


Fig. 2.3. Mesh Generated by a Sweep Method [10].

2.3.2. Overlay Grid Methods

The overlay grid techniques [26-29] use a grid that encompasses the volume to be meshed. Traditionally, this grid is structured and aligned with the coordinate axes. This background grid is then intersected with the geometry to determine elements that are:

- inside,
- on the boundary, and
- outside

the volume being meshed. The elements on the inside are retained, the ones on the outside are eliminated, and the ones on the boundary are then adjusted to fit the existing boundary.

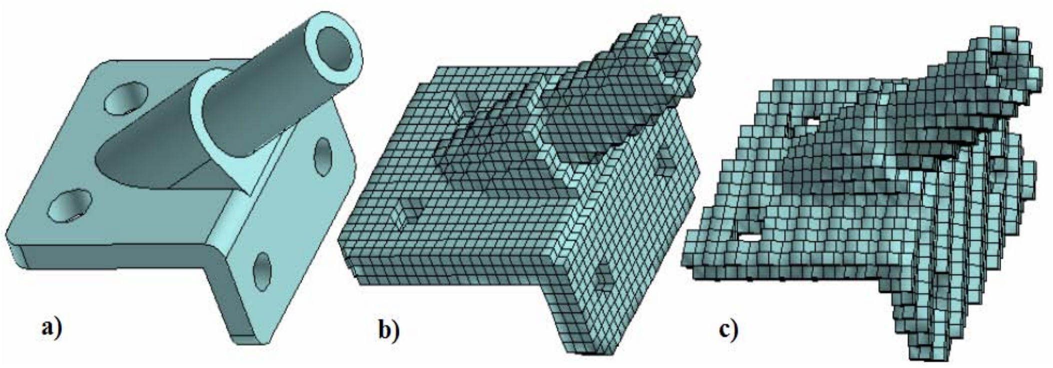


Fig. 2. 4. Mesh Generated by Overlay Grid Method; and Effect of Orientation in 3D [10].

Fig. 2.4 shows the interior and boundary elements generated within the volume shown, using this technique. The effect of grid alignment on the resulting mesh is obvious. An exaggerated illustration about orientation sensitivity in 2D is given in Fig. 2.5.

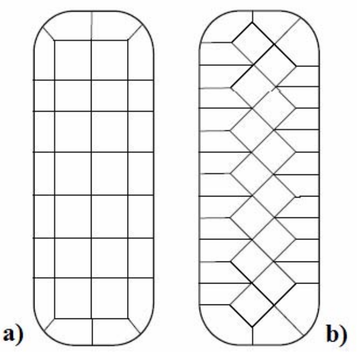


Fig. 2.5. Mesh Generated by Overlay Grid Method; and Effect of Orientation in 2D.

Another grid based approach, which has been successfully applied to mesh generation of CAD models, is the boundary fit method [30]. Between the decomposition and reassembly, it relies on several steps principally: edge detection performed on the boundaries, recognition depending on the angles between boundary surfaces, transformation to a space where the object will resemble a rectangular prism. The whole procedure is illustrated in Fig. 2.6.

The advantages and disadvantages of this approach can be summarized as follows:

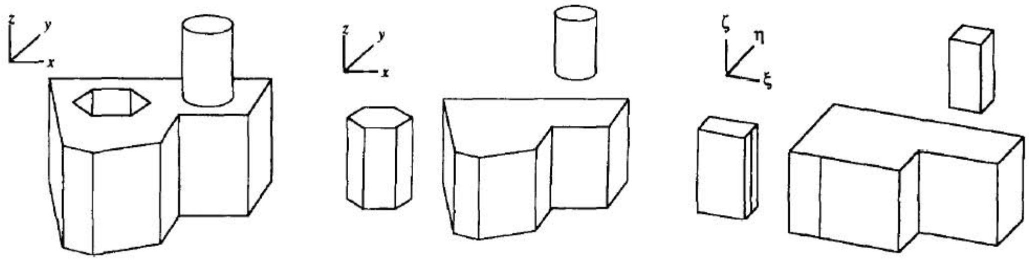
Advantages:

- High automation,
- Applicability to a broad class of geometries,
- Bad geometry tolerance (especially when the element sizes are comparable to gaps and holes),
- Ease of implementation (nice and simple data structures).

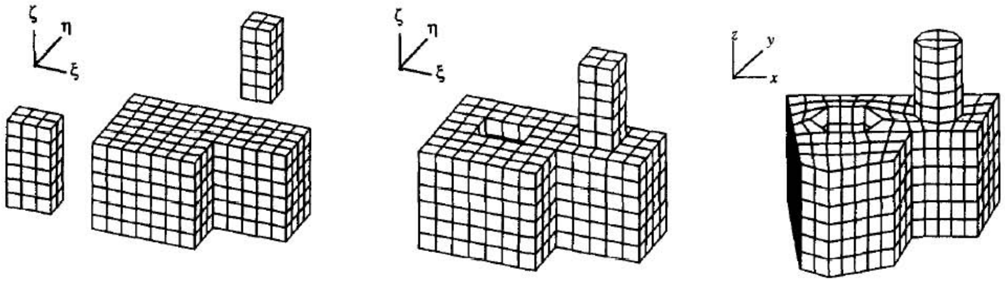
Disadvantages:

- Boundary insensitivity (requires more effort to improve the boundary elements, which are of worst quality),
- Surface mesh dependency to the algorithm (which limits the applicability of the algorithm to only volumes with no pre-existing surface mesh),
- Orientation sensitivity,

- Difficult/impossible conformal fine to coarse mesh transitions (requires non-conformal transitions).



a) Solid Model (from CAD file) b) Decomposed Solid Model c) Recognition Model



d) Meshed Recognition Model e) Reassembled Recognition Model f) Meshed Solid Model

Fig. 2.6. Steps Followed in the Boundary Fit Method [30].

2.3.3. Mesh Partitioning Methods

Mesh partitioning techniques [31-34] perform the decomposition integrally during the meshing process (not before as in the conventional approaches). These techniques tightly control the surface mesh to insure that decomposition is possible. An example cooper mesh, which uses mesh partitioning, is shown in Fig. 2.7. In mesh partitioning approaches, the decompositions occur integrally with the meshing process by using an interior mesh as the cutting mechanism (i.e. no geometric cuts).

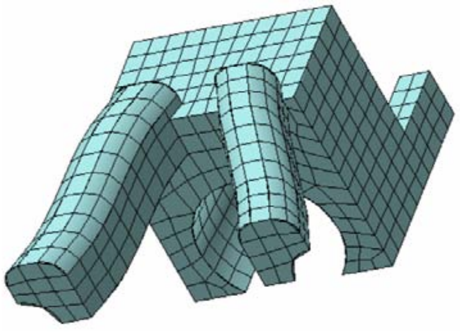


Fig. 2.7. An Example Mesh Generated by the Cooper Tool Using the Mesh Partitioning Approach [10].

The advantages and disadvantages of this approach can be summarized as follows:

Advantages:

- No geometric operator dependence,
- Boundary sensitivity,
- Orientation insensitivity.

Disadvantages:

- Applicability to a limited class of geometries,
- Necessity for much manual intervention of the user due to complexity,
- High dependency of the surface mesh to the algorithm (unexpected surface mesh).

2.3.4. Advancing Front Methods

Advancing front techniques are designed to work from the boundary of the mesh inward [35-36]. From the surface mesh, layers of elements are inserted to form the volume mesh in the interior of the volume. As the surfaces begin to intersect, they are connected in attempts to form a valid hexahedral mesh.

In the 2-D version of the advancing front technique, dealing with quadrilateral elements called paving by Blacker and Stephenson [37], rows of such elements are added along domain boundaries as seen in Fig. 2.8. Rows originating from different boundaries are matched to result in a conformal mesh. One of the most critical aspects is meshing the remaining void successfully when fronts from different boundaries converge.

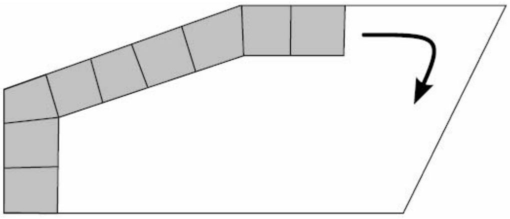


Fig. 2.8. Paving [38].

The method had been extended to 3-D surface meshing problems by Cass *et al* [39], which is referred to as plastering. In this method, layers of hexahedral elements are generated along boundaries as seen in Fig. 2.9. As in paving, fronts built on different surfaces are connected together to preserve mesh conformity. However, the problem of meshing the void which remains when different fronts converge to each other is generally not yet solved. Generally this task has proven to be intractable, and the use of mixed element types interior to the volume becomes necessary to fill the void [40]. While paving had successfully been proved to be robust and reliable, its three-dimensional counterpart lacks robustness in many classes of problems.

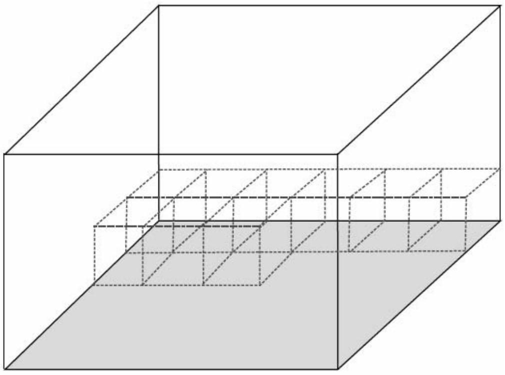


Fig. 2.9. Plastering.

In attempts to overcome the difficulties posed by plastering, an alternative algorithm called whisker weaving was developed [41-42]. The algorithm employs the so-called Spatial Twist Continuum (STC) or the dual of a volume mesh [43-44]. The idea behind whisker weaving can be described as follows: Let the dual of a surface quadrilateral mesh be generated. Let this dual be extruded into the volume bounded by this surface. This results in generation of a set of surfaces (twist planes), which intersect the surface mesh along the lines of its dual as seen in Fig. 2.10. Once a valid set of twist planes is found; hexahedral elements can be generated, wherever the three twist planes converge. As the intersection of twist planes is important in topological sense only, there is no need to compute any intersections. The method is relatively young, and must still prove itself as a reliable meshing tool for many classes of problems.

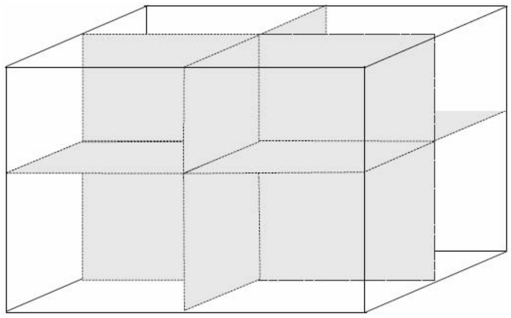


Fig. 2.10. Spatial Twist Continuum.

The advantages and disadvantages of this approach can be summarized as follows:

Advantages:

- Boundary sensitivity,
- Orientation insensitivity,
- Usability of existing known surface mesh,
- Independency to geometry (like most tetrahedral meshing algorithms).

Disadvantages:

- Necessity for technology maturity,
- Necessity for mixed element types (for plastering),
- Challenge in element quality (for whisker weaving),

- Time intensiveness.

2.4. Hexahedral Mesh Quality Improvement

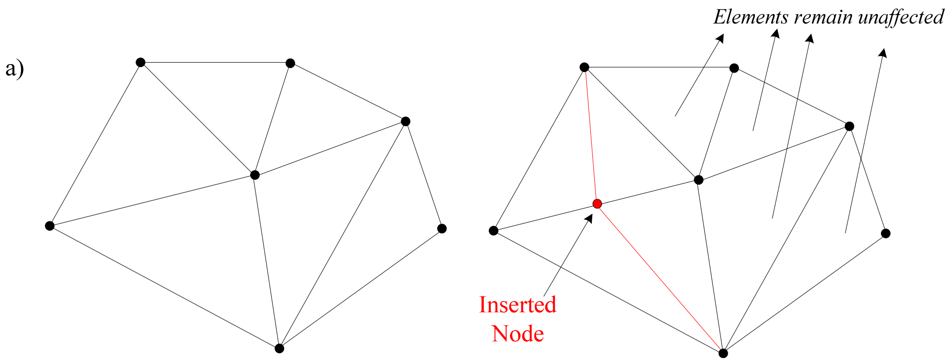
Generally, mesh quality is an important factor in accuracy. It can be said that qualitatively bad elements (especially in shape) cause the stiffness matrix to be ill-conditioned. This makes the system of equations to be more sensitive to any errors caused by the error sources mentioned in the first chapter. Mesh improvement can be performed either by modifying or preserving the node connectivity.

2.4.1. Mesh Refinement

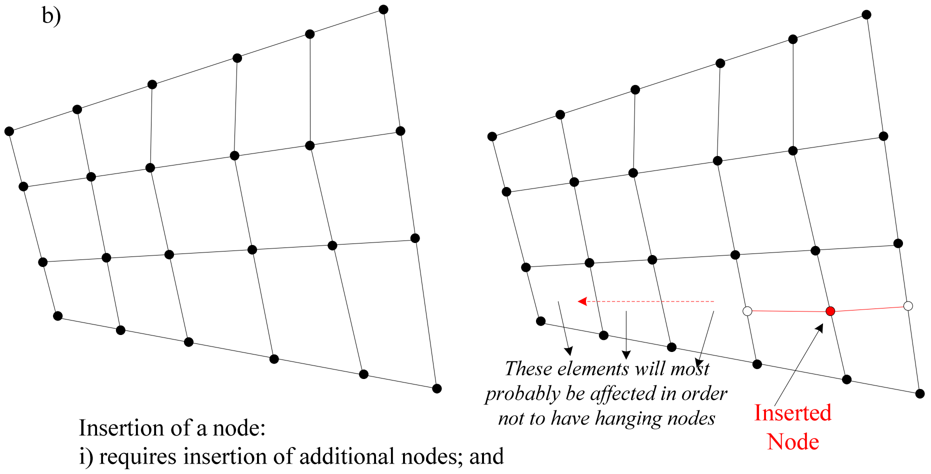
Refinement is performed by inserting additional nodes to the mesh; hence changing the connectivity. Refinement is not a popular method for quadrilateral/hexahedral elements as it is for triangular/tetrahedral elements. The reason is as follows: in order not to have a hanging[1] node or edge; propagation to neighboring elements is required. Or, some complicated methods shall be implemented to get rid of the propagation. What is meant by propagation is illustrated in Fig. 2.11 for 2D case.

---

[1] A node/edge is referred to as hanging, if it is not a node/edge of the neighboring element.



Insertion of a node:
i) does not require insertion of additional nodes; and
ii) only affects the directly related 2 elements (i.e. no propagation).



Insertion of a node:
i) requires insertion of additional nodes; and
ii) also affects the other elements (i.e. propagation to left).

Fig. 2.11. Propagation Requirement of Mesh Refinement for Quadrilateral/Hexahedral Mesh.

Meshes with hanging nodes/edges force the relevant finite element software to be more complicated. Methods using iterative octrees [48] result in such meshes. Refinement techniques based on pillowing, such as the cleave-and-fill tool [45], overcome the propagation requirement in a very clever manner; however, the control and scale of cleave and fill refinement is limited. A pillow region, and a cleave-and-fill refined mesh are given in Fig.s 2.12 and 2.13 respectively.

Other techniques insert non-hex elements that result in hybrid meshes or require uniform dicing to maintain a consistent element type [46]. Schneiders' directional refinement method [47] produces a conformal mesh by pillowing layers in alternating *i*, *j* and *k* directions but requires a Cartesian initial octree. The 3D anisotropic refinement scheme presented by Tchon *et al* [48] expands Schneiders' multi-directional refinement to initially unstructured meshes by pillowing layers of elements without the use of octrees. This method is quite robust but does not offer the capability to refine mesh regions around individual nodes, element edges or element faces.

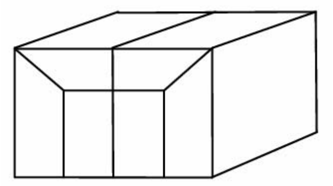


Fig. 2.12. An Example of a Pillow Region [45].

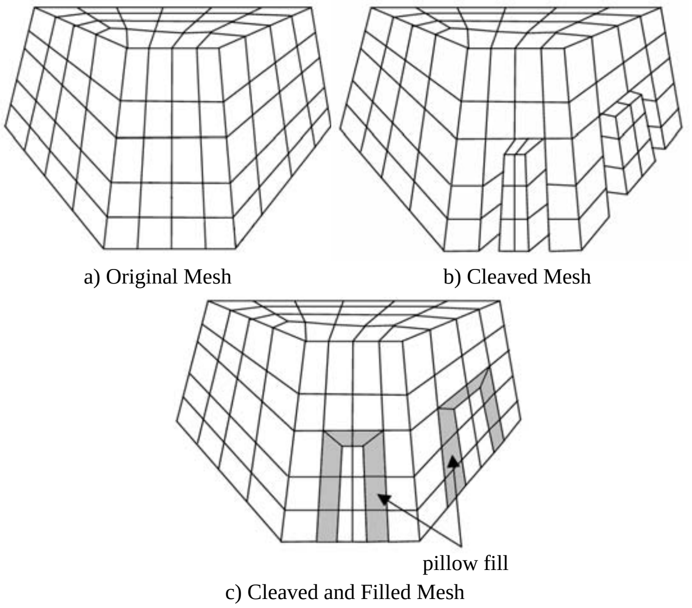


Fig. 2.13. Example of Cleave-and-Fill Refinement [45].

The term mesh connectivity denotes a system of edges, faces and cells which are formed based on a given set of vertices. It is also referred to as mesh topology. First of all, it is important to understand how topological modifications can affect mesh quality. Improvement of mesh quality via node movement is possible only up to a certain quality limit. This limit depends on topology of the mesh. Fig. 2.14 shows an example in which mesh quality cannot be improved unless its topology is modified. Namely, substitution of a five cell pattern by a four cell one results in an increase of the minimum angle of the mesh up to 90°.

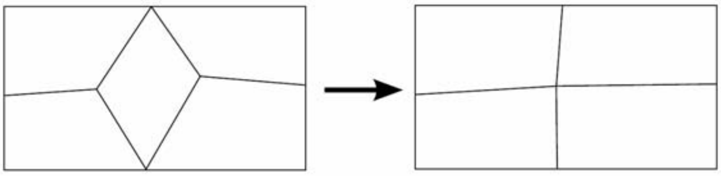


Fig. 2.14. Mesh quality improvement via modification of its connectivity [38].

Topological modifications are applied most successfully in triangular and tetrahedral mesh generation systems.

2.4.2. Mesh Smoothing

Smoothing can be defined as relocation of mesh nodes without changing their connectivity. Despite the recent evolution of methodology of unstructured mesh quality improvement, smoothing still remains the most common approach to quality improvement of non-simplicial meshes. There exist multiple ways of defining new positions for relocated nodes. Kovalev sorted these methods into the following categories [38]:

- Laplacian smoothing and its variations,
- Optimization-based smoothing,
- Physics-based smoothing,
- Hybrid techniques,
- Untangling.

Laplacian smoothing represents the most common family of methods among those mentioned above. The simplest form of Laplacian smoothing places each mesh node successively at the average position of nodes connected to it (i.e. to the centroid of all neighboring nodes). This has been a very popular method which was extended by many research groups [49-53] so far.

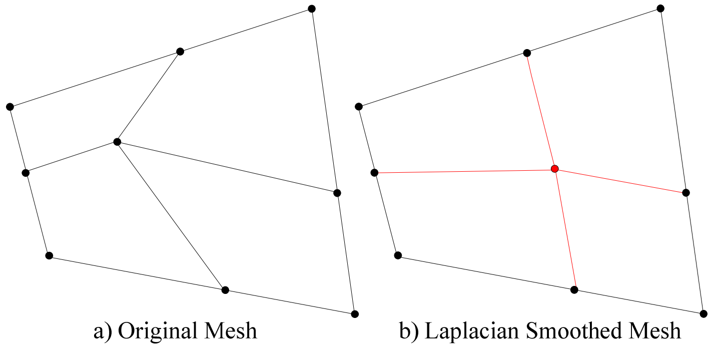

a) Original Mesh b) Laplacian Smoothed Mesh

Fig. 2.15. A Successful Application of Laplacian Smoothing (Convex Region).

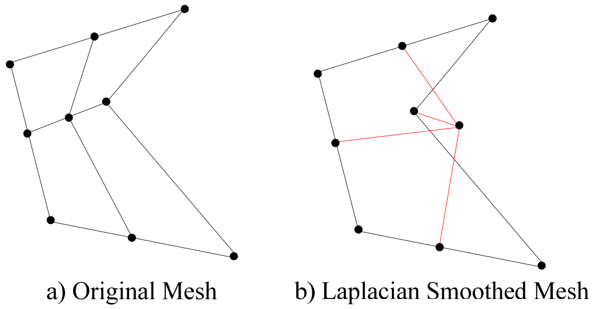

a) Original Mesh b) Laplacian Smoothed Mesh

Fig. 2.16. An Unsuccessful Application of Laplacian Smoothing (Concave Region).

This method works well in convex regions as seen in Fig. 2.15; on the other hand it may produce poorly-shaped or even inverted elements in concave regions as seen in Fig. 2.16. Typically, the mesh surrounding concave geometrical items is pulled outwards. This often leads to geometry-overlaying meshes.

In order to overcome such problems, more advanced methods have been proposed. For instance, the constrained Laplacian smoothing restricts nodal displacement to a certain limit in order to avoid element distortion. A simple form of this approach limits nodal movement equally in all directions: if the computed node displacement is smaller than a certain pre-defined threshold, this displacement is applied directly; otherwise, the node is moved by the threshold distance in the same direction.

In a more advanced implementation, quality of elements affected by node movement is computed using both old and new node locations. The node is then moved only if the quality of all affected elements does not decrease after the motion. This method is referred to as the “smart” Laplacian smoothing [50,54].

Another example of a modified Laplacian smoother can be found in [55]. Here, the advantage is taken from local action of Laplacian smoothing applied to unstructured meshes. Topology of a mesh subject to smoothing is analyzed and the most appropriate smoothing schemes are applied based on the result of the analysis. For instance, if exactly eight hexahedral cells surround a node of a hexahedral mesh (as in structured grids), then the so called equipotential (Winslow) smoothing, an approach for smoothing of structured grids, is applied. Earlier, Knupp [56] developed a 2D version of this smoother applied to quadrilateral unstructured meshes.

When applied to unstructured hexahedral meshes, a Laplacian smoother can be involved in a quality improvement process, but it cannot be a standalone quality improvement tool due to limited efficiency.

The optimization-based smoothing [51,54,57-61] moves mesh nodes to minimize/maximize a certain cell distortion/quality metric. Nodal positions are modified based on analysis of variations of local mesh quality, unlike in heuristic approach used in Laplacian smoothing. The optimization-based approach is more reliable, especially in concave regions of computational domain. However, the computational cost of this approach is obviously higher than that of the Laplacian methods.

The physics-based smoothing is based on the idea that a well-shaped mesh can be viewed as an analogue of certain mechanical systems that can be found in nature. Some authors have developed techniques that smooth unstructured meshes using principles which drive dynamics of such systems in nature.

A classical example of one of such techniques is the so-called spring analogy approach. In this method, edges of a mesh are considered as springs with stiffnesses depending on their target sizes. A mesh node is repositioned to a new location to bring the set of surrounding springs to equilibrium. All mesh nodes are successively repositioned until the entire spring system reaches equilibrium. Forces occurring in springs can be unidirectional or bi-directional. In the first case, magnitude of a force depends on spring’s length but its direction is always the same. In the second one, equilibrium spring length can be defined. A force changes its direction depending on the actual edge length with respect to its equilibrium length.

Another group of methods which has been developed recently and which employs the analogy with mechanical spring systems uses the so-called torsional springs [62-63]. While regular linear springs resist changes in internodal distances, the torsional springs resist changes in the angle between edges incident to same node. When the angle between such edges deflects from equilibrium, a corresponding torsional spring generates torque directed to bring the system back to equilibrium (see Fig. 2.17).

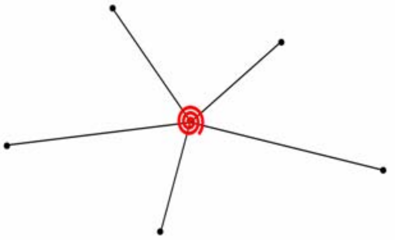


Fig. 2.17. Torsional Spring Analogy [38].

The combined approaches usually represent a blend of optimization-based and Laplacian smoothing methods. Development of such methods is aimed at combining the speed of

Laplacian approach with the efficiency of optimization-based methods. In these methods, it is important to develop a system of criteria to effectively distinguish whether a Laplacian or an optimization-based smoothing method should be applied to a mesh node. Examples of combined approaches can be found in [51, 54].

2.5. Hexahedral Mesh Quality Metrics

In the literature, numerous quality metrics for hexahedral elements have been defined. Most of these were originally defined for other element shapes, and later adapted to hexahedral elements. Hence, most of these might not be effective and meaningful although they exist in some availeable software packages. Table 2.1 is an overview of the basic hexahedral element quality metrics.

Among these metrics, Oddy metric [67] found broad application in the literature. Unlike other metrics, this metric was originally constructed for hexahedral elements. However, the metric was found to be excessively restrictive on element’s aspect ratio distortion and, on the contrary, insufficiently restrictive on element’s angle distortion.

In addition to these metrics, there have been a couple of other metric definitions in the literature. Robinson’s 2D metrics [68] have been generalized to 3D, but these metrics suffered from the same deficiencies in 2D. A metric definition for hexahedral and wedge elements was proposed by Kwok and Chen [69], as the product of aspect ratio, warpage, and volume ratio metrics. However the metric was neither continuous nor differentiable since the aspect ratio is a ‘step-function’. Its full and acceptable ranges were not clearly defined by the authors. In a series of papers about mesh generation for rotating machines by Noguchi *et al* [70-72]; the authors have defined various quality metrics for hexahedral elements. Ranges, dimensions, physical meanings of these definitions have not been explicitly defined in these papers.

Table 2.1. Basic quality metrics for hexahedral elements.

Metric Name	**Definition**	**Dimension**	**Full Range**	**Acceptable Range**	**Ref.**
Aspect Ratio	Maximum edge length ratios at hexahedron center	L^0	[1,∞)	[1,4]	[64]
Skew	Maximum $\mid\cos\theta\mid$ where θ is the angle between edges at hexahedron center.	L^0	[0,1]	[0,0.5]	[64]
Taper	Maximum ratio of lengths derived from opposite edges.	L^0	[0,∞)	[0,0.4]	[64]
Element Volume	Jacobian at hexahedron center.	L^3	(-∞,∞)	None	[64]
Stretch	$\sqrt{3}$ × minimum edge length / maximum diagonal length.	L^0	[0,1]	[0.25,1]	[65]
Diagonal Ratio	Minimum diagonal length / maximum diagonal length.	L^0	[0,1]	[0.65,1]	[64]
Dimension	Dimension of the element	L^1	[0,∞)	None	[64]
Condition Number	Maximum condition number of the Jacobian matrix at 8 corners.	L^0	[1,∞)	[1,8]	[66]
Jacobian	Minimum pointwise volume of local map at 8 corners & center of the hexahedron.	L^3	(-∞,∞)	None	[66]
Scaled Jacobian	Minimum Jacobian divided by the lengths of the 3 edge vectors.	L^0	[-1,1]	[0.5,1]	[66]
Shear	3/Mean Ratio of Jacobian Skew Matrix.	L^0	[0,1]	[0.3,1]	[66]
Shape	3/Mean Ratio of weighted Jacobian Matrix.	L^0	[0,1]	[0.3,1]	[66]
Relative Size	$\min(J, 1/J)$, where J is the determinant of weighted Jacobian matrix.	L^0	[0,1]	[0.5,1]	[66]
Shear & Size	Product of Shear and Size Metrics.	L^0	[0,1]	[0.2,1]	[66]
Shape & Size	Product of Shape and Size Metrics	L^0	[0,1]	[0.2,1]	[66]
Distortion	{min(|J|) / actual volume} × parent volume, where parent volume = 8 for a hexahedral element	L^0	[0,1]	[0.6,1]	-
Oddy		L^0	[0, ∞]		[67]

Being uncomfortable from the basic quality metrics (due to their restrictions), Knupp focused on trying to define general-purpose quality metrics especially after the year 1999. In [73], he described the properties of the general-purpose quality metrics as in Table 2.2.

Table 2.2. Comparison of Restricted and General Quality Metrics [73].

Property	**Restricted Metric**	**General Metric**
Dimension (n = 2 vs. n = 3)	Dimension-specific (e.g., only applies to n = 2)	Dimension-free (applies both to n = 2 and n = 3)
Element Type (e.g., triangular or quadrilateral)	Element-specific (e.g. only defined for quadrilateral elements)	Element-free (e.g. defined both for triangular and quadrilateral elements)
Domain (e.g. shape of quadrilateral element)	Domain-specific (e.g. rectangles only)	Domain-general (e.g. all quadrilateral elements)
Versatility (# of qualities measured)	Specialized (only one)	Versatile (e.g. volume-shape orientation)
Element Size (or volume)	Scale-sensitive (size dependent)	Scale-free (size invariant)
Orientation	Orientation-sensitive (orientation-dependent)	Orientation-free (orientation-invariant)
Units (of metric)	Has units (dimensional)	Unitless (nondimensional)
Reference (ideal element)	Unreferenced (implicit ideal)	Referenced (explicit ideal)

After these definitions, Knupp had a series of papers with lots of general purpose quality metrics [66,73-74]. His metric does not depend on element orientation and size, but only on shape distortion with respect to an ideal element. The ideal element can be chosen based on a certain set of requirements to element shape that must be satisfied in an optimized mesh. Details of his “condition number” based metric are given in Appendix D. This metric is also used in this thesis.

2.6. Curvilinearization - Extension of Quality Definitions for Curvilinear Mesh

Representation of curved edges and surfaces is another challenge in the finite element method. Simple mapping techniques might yield big errors since the objects are represented badly and distorted in such cases.

As well as the mapping, it can be claimed that the quality metrics, which are defined in previous sections, should be extended for curvilinear elements. However, as will be seen, this is not an easy task. Moreover, most of the research showed that this effort might not be worth to spend time; since adding more degrees of freedom handles much of the error reduction.

2.6.1. Mapping Functions

In two dimensions, the boundary of the solution domain generally consists of piecewise smooth curves, and it is necessary to consider mapping of the standard element so that these curves are closely approximated. In *p*-extensions (see Chapter 3; sections 3.5 and 3.6 for details), most elements remain large. Therefore, the use of accurate mapping techniques is much more important than in the case of *h*-extensions. Assume that, in 2D the vertex coordinates of quadrilateral elements in the *xy*-plane by capital letters X_i, Y_i, and the mapping functions for the *k*th quadrilateral element by:

$$x = Q_x^{(k)}(\xi,\eta) \tag{2.1}$$

2.6.1.1. Linear Mapping

When all sides of the mapped elements are straight lines, then generally linear mapping is used. However, certain kinds of nonlinear mapping are used in some cases. In the case of quadrilaterals, linear mapping is in the form:

$$\begin{aligned} x = Q_x^{(k)}(\xi,\eta) &= \frac{1}{4}(1-\xi)(1-\eta)X_1 + \frac{1}{4}(1+\xi)(1-\eta)X_2 + \\ &+ \frac{1}{4}(1+\xi)(1+\eta)X_3 + \frac{1}{4}(1-\xi)(1+\eta)X_4 \\ y = Q_y^{(k)}(\xi,\eta) &= \frac{1}{4}(1-\xi)(1-\eta)Y_1 + \frac{1}{4}(1+\xi)(1-\eta)Y_2 + \\ &+ \frac{1}{4}(1+\xi)(1+\eta)Y_3 + \frac{1}{4}(1-\xi)(1+\eta)Y_4 \end{aligned} \tag{2.2}$$

2.6.1.2. Quadratic Parametric Mapping

Quadratic parametric mapping permits representation of curved element sides by polynomials of degree 2. In the case of quadrilateral elements, the mapping functions are

$$x = \sum_{i=1}^{8} X_i N_i(\xi,\eta), \qquad y = \sum_{i=1}^{8} Y_i N_i(\xi,\eta) \tag{2.3}$$

where N_i are the shape functions given by

$$\begin{aligned}
N_1 &= \frac{1}{4}(1-\xi)(1-\eta)(-\xi-\eta-1)\\
N_2 &= \frac{1}{4}(1+\xi)(1-\eta)(\xi-\eta-1)\\
N_3 &= \frac{1}{4}(1+\xi)(1+\eta)(\xi+\eta-1)\\
N_4 &= \frac{1}{4}(1-\xi)(1+\eta)(-\xi+\eta-1)\\
N_5 &= \frac{1}{2}(1-\xi^2)(1-\eta)\\
N_6 &= \frac{1}{2}(1+\xi)(1-\eta^2)\\
N_7 &= \frac{1}{2}(1-\xi^2)(1+\eta)\\
N_8 &= \frac{1}{2}(1-\xi)(1-\eta^2)
\end{aligned} \tag{2.4}$$

and X_i, Y_i are the nodal coordinates corresponding to the numbering scheme in Fig. 2.18. When p=2, and the conventional shape functions are used, then the displacement vector components and the mapping are represented by the same shape functions. For this reason this mapping is called *iso*parametric mapping. Isoparametric mapping is used very extensively in conjunction with the finite element software. The accuracy increases with the number of elements.

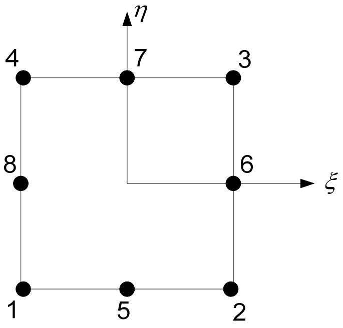


Fig. 2.18. Nodal Numbering Scheme for Quadratic Quadrilateral Elements.

2.6.1.3. Mapping by the Blending Function Method

In the *p*-version, generally large elements are used. It is important to represent curved boundary segments accurately with a few elements. The linear blending function proposed by Gordon and Hall [75] is well suited for this purpose.

To illustrate this method, consider a simple case where only one side of a quadrilateral element is curved, as shown in Figure 2.19. The curve $x = x_2(\eta)$, $y = y_2(\eta)$ is given in parametric form so that $x_2(-1) = X_2$, $y_2(-1) = Y_2$, $x_2(1) = X_3$, $y_2(1) = Y_3$. It is possible to write:

$$x = \frac{1}{4}(1-\xi)(1-\eta)X_1 + \frac{1}{4}(1+\xi)(1-\eta)X_2 + \frac{1}{4}(1+\xi)(1+\eta)X_3 + \frac{1}{4}(1-\xi)(1+\eta)X_4$$
$$+\left(x_2(\eta) - \frac{1-\eta}{2}X_2 - \frac{1+\eta}{2}X_3 \right)\frac{1+\xi}{2} \tag{2.5}$$

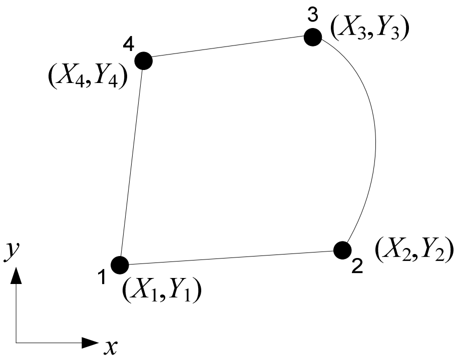


Fig. 2.19. A Quadrilateral with a Curved Side.

Clearly, the first four terms in this expression are the linear mapping terms of Equation(2.2). The fifth term is the product of two functions. One function, the bracketed expression, represents the difference between $x_2(\eta)$ and the x-coordinates of the chord that connects points (X_2, Y_2) and (X_3, Y_3).

The other function is the linear blending function $(1+\xi)/2$, which is unity along side 2 and zero along side 4. In this case, it can be claimed that:

$$x = \frac{1}{4}(1-\xi)(1-\eta)X_1 + \frac{1}{4}(1-\xi)(1+\eta)X_4 + x_2(\eta)\frac{1+\xi}{2} \tag{2.6}$$

Similarly:

$$y = \frac{1}{4}(1-\xi)(1-\eta)Y_1 + \frac{1}{4}(1-\xi)(1+\eta)Y_4 + y_2(\eta)\frac{1+\xi}{2} \tag{2.7}$$

In the general case, all sides may be curved. The curved sides can be written in the parametric form:

$$x = x_i(\xi), \quad y = y_i(\xi), \quad -1 \le \xi \le 1, \quad (i = 1,2,3,4) \tag{2.8}$$

where the subscripts represent the side numbers of the standard element. In this case, the mapping functions are:

$$
\begin{aligned}
x = {} & \frac{1}{2}(1-\eta)x_1(\xi)+\frac{1}{2}(1+\xi)x_2(\eta)+\frac{1}{2}(1+\eta)x_3(\xi)+\frac{1}{2}(1-\xi)x_4(\eta) \\
& -(1-\xi)(1-\eta)X_1-\frac{1}{4}(1+\xi)(1-\eta)X_2-\frac{1}{4}(1+\xi)(1+\eta)X_3-\frac{1}{4}(1-\xi)(1+\eta)X_4
\end{aligned}
\tag{2.9}
$$

where X_1, X_2, X_3, X_4 are the global x-coordinates of the four nodes of the quadrilateral element. Similarly, denoting the global y-coordinates by Y_i ($i = 1,2,3,4$):

$$
\begin{aligned}
y = {} & \frac{1}{2}(1-\eta)y_1(\xi)+\frac{1}{2}(1+\xi)y_2(\eta)+\frac{1}{2}(1+\eta)y_3(\xi)+\frac{1}{2}(1-\xi)y_4(\eta) \\
& -(1-\xi)(1-\eta)Y_1-\frac{1}{4}(1+\xi)(1-\eta)Y_2-\frac{1}{4}(1+\xi)(1+\eta)Y_3-\frac{1}{4}(1-\xi)(1+\eta)Y_4
\end{aligned}
\tag{2.10}
$$

The inverse mapping, that is $\xi = Q_\xi^{(k)}(x, y)$, $\eta = Q_\eta^{(k)}(x, y)$ cannot be given explicitly in general. However, (ξ, η) can be computed very efficiently for any given (x, y) by means of the Newton-Raphson method or a similar procedure. The quadratic (iso)parametric mapping, which is the commonly used mapping in finite element analysis, can be viewed as a special application of the blending function method in which the sides are represented by quadratic polynomial functions.

2.6.1.4. Bézier curves

Bézier curves are named after their inventor, Pierre Bézier, who was an engineer with the Renault car company and set out a curve formulation in the early 1960s which would lend itself to shape design. Engineers find it most understandable to think of Bézier curves in terms of the center of mass of a set of point masses. For example, consider the four masses m_0, m_1, m_2, and m_3 located at points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{P}_3$. The center of mass of these four point masses is given by the equation

$$
\mathbf{P} = \frac{m_0\mathbf{P}_0 + m_1\mathbf{P}_1 + m_2\mathbf{P}_2 + m_3\mathbf{P}_3}{m_0 + m_1 + m_2 + m_3} \tag{2.11}
$$

Next, imagine that instead of being fixed, constant values, each mass varies as a function of some parameter *t*. Specifically, let $m_0 = (1- t)^3$, $m_1 = 3t(1- t)^2$, $m_2 = 3t^2(1- t)$, and $m_3 = t^3$. The values of these masses as a function of *t* are shown in Fig. 2.20.

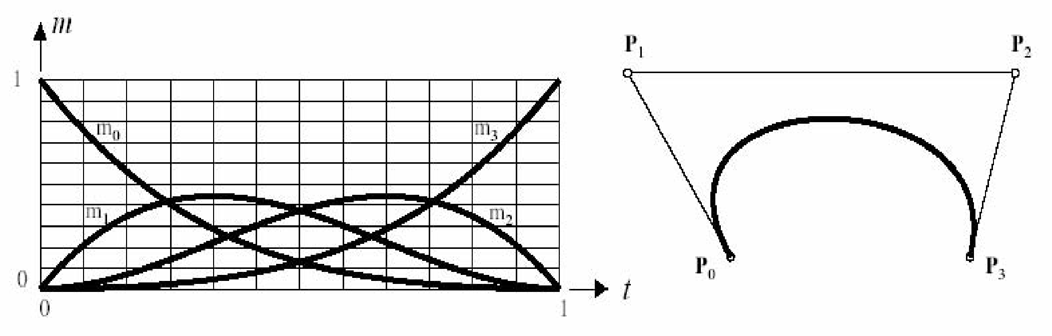


Fig. 2.20. Cubic Bézier Blending Functions and a Cubic Bézier Curve [76].

Now, for each value of *t*, the masses assume different weights and their center of mass changes continuously. In fact, as *t* varies between 0 and 1, a curve is swept out by the center of masses. This curve is a cubic Bézier curve – cubic because the mass equations are cubic polynomials in *t*. Notice that, for any value of *t*, $m_0 + m_1 + m_2 + m_3 = 1$, and so the equation of this Bézier curve can simply be written as $\mathbf{P} = m_0\mathbf{P}_0 + m_1\mathbf{P}_1 + m_2\mathbf{P}_2 + m_3\mathbf{P}_3$.

It should be noted that when $t = 0$, $m_0 = 1$ and $m_1 = m_2 = m_3 = 0$. This forces the curve to pass through $\mathbf{P}_0$. Also, when $t = 1$, $m_3 = 1$ and $m_0 = m_1 = m_2 = 0$, thus the curve also passes through point $\mathbf{P}_3$. Furthermore, the curve is tangent to $\mathbf{P}_0 – \mathbf{P}_1$ and $\mathbf{P}_3 – \mathbf{P}_2$. These properties make Bézier curves an intuitively meaningful means for describing free-form shapes. Here are some other examples of cubic Bézier curves, which illustrate these properties. These variable masses m_i are normally called blending functions and their locations $\mathbf{P}_i$ are known as control points or Bézier points. If straight lines between adjacent control points are drawn, as in a dot to dot puzzle, the resulting figure is known as a control polygon. The blending functions, in the case of Bézier curves, are known as Bernstein polynomials. Bézier curves of any degree can be defined. Fig. 2.21 shows sample curves of degree one through four.

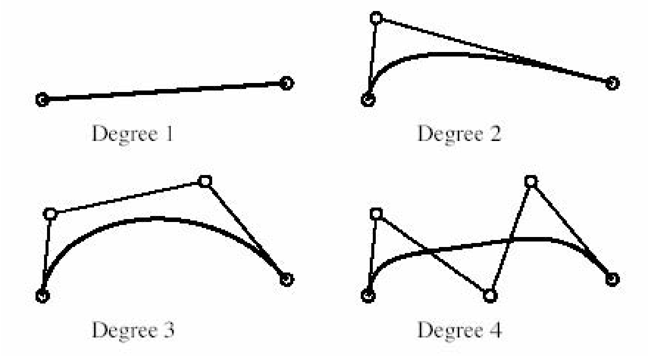


Fig. 2.21. Bézier Curves of Various Orders [76].

A degree *n* Bézier curve has *n* + 1 control points whose blending functions are denoted $B_i^n(t)$, where

$$B_i^n = \binom{n}{i}(1-t)^{n-i}t^i, \qquad i = 0,1,\ldots,n \tag{2.12}$$

In the introductory example, $n = 3$ and $m_0 = B_0^3 = (1- t)^3$, $m_1 = B_1^3 = 3t(1- t)^2$, $m_2 = B_2^3 = 3t^2(1- t)$, and $m_3 = B_3^3 = t^3$. $B_i^n(t)$ is also referred to as the *i*th Bernstein polynomial of degree *n*. The equation of a Bézier curve is thus:

$$\mathbf{P}(t) = \sum_{i=0}^{n}\binom{n}{i}(1-t)^{n-i}t^i\mathbf{P}_i \tag{2.13}$$

In summary, advantageous properties of Bézier polynomials can be listed as follows:

- They can be as high a degree as desired
- Convex hull provides smoother and more controllable approximation
- Better properties to allow more efficient intersection checks
- Derivatives and products of Béziers are also Béziers
- Efficient algorithms for degree elevation and subdivision can be found

2.6.2. Extension of Mesh Quality in Curvilinear Elements

To the author’s knowledge, several research about the extension of mesh quality metrics to curvilinear elements have ended up with the conclusion that for mesh validity is more important than mesh quality for such elements. There is not much published work about this subject, but Dey *et al* [77] proposed a procedure about curvilinear mesh generation. In this work, they start with linear mesh generation, improve its quality with available metrics, then curvilinearize the mesh as long as it remains valid (according to some validity constraint). The details are given in the following subsections.

2.6.2.1. Validity Rather Than Quality

The validity check is performed by means of the Jacobian. This means that:

i. as long as the corresponding linear mesh is of high quality;
ii. and the Jacobian inside the elements after curvilinearization is > 0 everywhere inside the element [77];

then the curvilinear mesh can be used with confidence. Naturally, there is no method to check the Jacobian everywhere inside the element. Traditional validation methods test the Jacobian at integration points. Increasing the check points require more CPU time, but certainly increases level of confidence.

Recently Luo *et al* [78] extended the validity to Bézier Curves and Regions.

- Jacobian is related to the region control points; and its minimum bound is determined
- A region is claimed to be valid globally if the minimum control point of the Jacobian determinant function is > 0

In this work, Luo *et al* [78] also listed the reasons for using “validity” rather than “additional quality” for curvilinear mesh:

- Past being valid, quality of curved mesh is not dictated by a-priori geometric measures
- Maximization of minimum element Jacobians is not likely to be superior to other options
- Even the mesh hardly passing the validation produces not too bad results
- A-priori quality metrics for curved elements are hard to define
- Adaptation based on the solution is more feasible

2.6.2.2. Effects of Element Distortion and Jacobian

Although the isoparametric elements are usually successful in representing the curved shapes, there is a limitation to the curvature, or namely to the element shape. In this section, the reason for the Jacobian based validity check is clarified with an example.

For simplicity, consider the one dimensional case, a quadratic isoparametric line element extending from 0 to h in x-coordinate system. As usual, the element is transformed to a unit element in ξ-coordinate system. The relation between x and ξ can be easily shown as

$$x(\xi) = h(4a-1)\xi + 2h(1-2a)\xi^2 \tag{2.14}$$

and the two coordinates have derivatives related by

$$\frac{\partial x}{\partial \xi} = h(4a-1) + 4h(1-2a)\xi \tag{2.15}$$

The Jacobian of the transformation is the inverse relation, that is

$$J = \frac{\partial \xi}{\partial x} \tag{2.16}$$

The mathematical principles require the Jacobian to be positive definite. Distortion of the elements can cause J to go to zero or become negative. This possibility is easily seen in the present 1-D example. If one locates the interior ($\xi = 1/2$) node at the standard midpoint position, then $a = 1/2$ so that $\partial x / \partial \xi = h$ and J is constant throughout the element. Such an element is generally well formulated.

However, if the interior node is distorted to any other position, the Jacobian will not be constant and the accuracy of the element may suffer. Generally, there will be points where $\partial x / \partial \xi$ goes to zero, so that the stiffness becomes singular due to division by zero.

For slightly distorted elements, say $0.4 < a < 0.6$, the singular points lie outside the element domain. As the distortion increases, the singularities move to the element boundary, e.g., $a = 1/4$ or $a = 3/4$. Eventually, the distortions cause singularities of J inside the element. Such situations can cause poor stiffness matrices and very bad estimates, unless the true solution has the same singularity. In that special case these distorted elements are known as the quarter point element. The effects of distortions of two- or three-dimensional elements are similar. For example, the edge of a quadratic element may have the non-corner node displaced in a similar way, or it may be moved normal to the line between the corners. Similar analytic singularities can be developed for such elements. However, the presence of singularities due to element distortions can easily be checked by numerical experiments, as stated in the previous subsection.

CHAPTER 3

HEXAHEDRAL EDGE ELEMENTS

3.1. Differential Forms and Algebraic Manifolds

This section is focused on concepts which are necessary to understand for the concept of edge elements; such as differential forms and manifolds.

A differential form is by definition any quantity that can be integrated, including differentials. In 1844, Hermann Günter Grassmann published his book *Die lineale Ausdehnungslehre, ein neuer Zweig der Mathematik* [79], in which he developed the idea of algebra in which the symbols representing geometric entities such as points, lines and planes are manipulated using certain rules. Grassmann introduced what is now called exterior algebra, based upon the exterior product. In the early 1900's, Elie Cartan developed an exterior calculus of differential forms. Since that time, differential forms have received widespread use in the physics and mathematics communities for many problems, including electrodynamics. After its early introduction into the engineering community by Deschamps [80], Engle [81], Baldomir [82] and others, the calculus of differential forms has been used in applications to numerical methods, boundary conditions, Green's functions, and anisotropic media.

Differential forms are a subset of a larger subject in mathematics called geometric calculus. The main concept of geometric calculus is its emphasis on the geometric interpretation of vectors, differential operators and all of the other ideas that combine to form a calculus. Both differential forms and geometric calculus are reinterpretations of vector calculus for metric free and higher order geometries. These concepts are useful in areas such as space time physics but can also be leveraged for use in described linear wave equations. The metric free nature of differential forms allows the construction of differential operators, gradient (∇), divergence (∇·) and curl (∇×) that are independent of the coordinate system. The importance of this property becomes apparent in the formulations.

The differential forms calculus is based on the concept of four entities called *p*-forms in three-dimensional space. The 0-form and 3-form are both scalar quantities in curvilinear geometry while the 1-form and 2-form are vector quantities in curvilinear geometry. The differential form takes a *p*-dimensional vector and gives a number. More information on differential forms can be found in the text by Burke [83].

To discuss differential forms, manifolds and their properties must first be discussed. Manifolds are descriptions of space which may be curved and have complicated topology; but they are spaces in which every point has a neighborhood resembling Euclidean space (i.e. R^n, the set of n-tuples $(x_1, x_2, \ldots, x_n)$). Locally manifolds look like Euclidean space and a general manifold is built by creating a set of locally Euclidean regions.

A simple example of manifolds is the surface of a sphere. The sum of the angles of a triangle is not exactly equal to 180° if the triangle is drawn on the surface of the sphere. Although the surface of the sphere is not a Euclidean space, locally the laws of the Euclidean geometry are good approximations (especially when the sphere is large, and the triangle is small).

On a manifold, structures such as vector, tangent and cotangent spaces can be created. Vector spaces are the set of vectors defined over a manifold and a tangent space is the set of all vectors at a single point three dimensional space. In R^3 a basis for the vector space of curvilinear coordinates can be defined by the vector (x_1, x_2, x_3).

At every point $y \in R^3$, a space of tangent vectors can be written by using the standard basis $\left\{ \frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \frac{\partial}{\partial x_3} \right\}$.

At every point $y \in R^3$, a space of cotangent vectors can be written by using the standard basis $\{dx_1, dx_2, dx_3\}$.

The 0-form takes a zero-dimensional vector, a point, and returns a scalar which corresponds to the evaluation of the scalar function at that point. These entities are useful for describing physical quantities that are continuous across a material interface such as potentials. Electric potential is a 0-form quantity.

1-forms correspond to quantities with tangential continuity across a material interface such as the electric field. Each component of a 1-form is a 0-form.

The 2-forms have normal continuity and represent fluxes such as the magnetic flux density.

The 3-forms are defined within a specific volume and therefore have no imposed continuity between adjacent volumes which allows them to represent discontinuous fields such as charge density.

More definitions and details can be found in Koning's thesis [84]; Warnick and Russer's [85] or Tonti's [86] works. As a summary, Table 3.2 lists the *p*-forms, their features; and which important quantities of electromagnetics belong to which class.

As can be understood, *p*-forms also construct Hilbert spaces (in fact Sobolev spaces) where the relevant discussion is given in the following section.

3.2. Hilbert Spaces Related to Electromagnetic Quantities

Let Ω be a conducting domain of interest; and Γ be its boundary. The symbols $L^2(\Omega)$ and $\mathbf{L}^2(\Omega)$ denote the spaces of all square integrable scalar and vector functions on Ω respectively. As usual, **n** denotes the normal vector outward Γ.

The vector spaces $H_0(\Omega,\mathrm{grad})$, $H_0(\Omega,\mathrm{curl})$, $H_0(\Omega,\mathrm{div})$ can be defined as:

$$H_0(\Omega,\mathrm{grad}) = \left\{\phi \in H(\Omega,\mathrm{grad}) \middle| \phi = 0 \text{ on } \Gamma\right\} \tag{3.1}$$

$$H_0(\Omega,\mathrm{curl}) = \left\{\mathbf{u} \in H(\Omega,\mathrm{curl}) \middle| \mathbf{u} \times \mathbf{n} = 0 \text{ on } \Gamma\right\} \tag{3.2}$$

$$H_0(\Omega,\mathrm{div}) = \left\{\mathbf{u} \in H(\Omega,\mathrm{div}) \middle| \mathbf{u} \cdot \mathbf{n} = 0 \text{ on } \Gamma\right\} \tag{3.3}$$

where

$$H(\Omega,\mathrm{grad}) = \left\{\phi \in L^2(\Omega) \middle| \nabla\phi \in L^2(\Omega)\right\} \tag{3.4}$$

$$H(\Omega,\mathrm{curl}) = \left\{\mathbf{u} \in \mathbf{L}^2(\Omega) \middle| \nabla \times \mathbf{u} \in \mathbf{L}^2(\Omega)\right\} \tag{3.5}$$

$$H(\Omega,\mathrm{div}) = \left\{\mathbf{u} \in \mathbf{L}^2(\Omega) \middle| \nabla \cdot \mathbf{u} \in L^2(\Omega)\right\} \tag{3.6}$$

With this information, the domains and ranges of the differential operators can be listed as in Table 3.1.

Table 3.1. Domains and Ranges of Differential Operators.

		Domain			
		$H(\Omega,\mathrm{grad})$	$H(\Omega,\mathrm{curl})$	$H(\Omega,\mathrm{div})$	$L^2(\Omega)$
Range	$H(\Omega,\mathrm{grad})$		$\nabla.$		
	$H(\Omega,\mathrm{curl})$	∇		$\nabla\times$	
	$H(\Omega,\mathrm{div})$		$\nabla\times$		∇
	$L^2(\Omega)$			$\nabla.$	

The four Hilbert spaces $H(\Omega,\text{grad})$, $H(\Omega,\text{curl})$, $H(\Omega,\text{div})$, $L^2(\Omega)$ and the three operators ∇, $\nabla\times$ and $\nabla.$ form a de Rham complex[2] relative to Γ. The dual complex can be introduced by using the adjoint differential operators ∇^*, $(\nabla\times)^*$ and $(\nabla.)^*$.

$$H(\Omega,\text{grad}) \xrightarrow{\nabla} H(\Omega,\text{curl}) \xrightarrow{\nabla\times} H(\Omega,\text{div}) \xrightarrow{\nabla\cdot} L^2(\Omega) \qquad (3.7)$$

The importance of this property stems from the fact that Maxwell's equations can be described in terms of a Tonti [86] diagram built up on this complex as seen in Fig. 3.1.

$$\begin{array}{ccccc}
 & \text{Ampère's Law} & & \text{Faraday's Law} & \\
H_0(\Omega,\text{grad}) & & & 0 & L_0^2(\Omega) \\
\nabla & & & \Uparrow & \nabla. \\
H_0(\Omega,\text{curl}) & \mathbf{H} & \Rightarrow \mu\mathbf{H} = \mathbf{B} \Rightarrow & \mathbf{B} & H_0^*(\Omega,\text{div}) \\
\nabla\times & \Downarrow & & \Uparrow & \nabla\times \\
H_0(\Omega,\text{div}) & \mathbf{J} & \Leftarrow \mathbf{J} = \varepsilon\mathbf{E} \Leftarrow & \mathbf{E} & H_0^*(\Omega,\text{curl}) \\
\nabla. & \Downarrow & & \uparrow & \nabla \\
L_0^2(\Omega) & 0 & & \phi & H_0^*(\Omega,\text{grad})
\end{array}$$

Fig. 3.1. Tonti Diagram of Maxwell's Equations.

Combining the definitions of the Hilbert spaces and the differential forms, we can list the properties of significant quantities of the electromagnetic theory as in Table 3.2.

---

[2] In mathematics, the de Rham complex is the cochain complex of exterior differential forms on some smooth manifold, with the exterior derivative as differential.

Table 3.2. Properties of the *p*-forms.

	0-form	**1-form**	**2-form**	**3-form**
Minimum Continuity	Total	Tangential	Normal	None
Integral	Point	Line	Surface	Volume
Derivative	Grad	Curl	Div	None
Physical Types	Scalar Potentials	Fields, Vector Potentials	Fluxes, Vector Densities	Scalar Densities
Specific Examples	ϕ	**A**, **E**, **H**	**B**, **D**, **J**	ρ
Hilbert Space	$H(\Omega,\text{grad})$	$H(\Omega,\text{curl})$	$H(\Omega,\text{div})$	$L^2(\Omega)$

For the classical Finite Element Method, the nodal elements are used in order to calculate the scalar quantities. In more general sense, this type of element correspond to 0-form element for $H(\Omega,\text{grad})$, which is shown in Fig. 3.2.

In 1986, Nédélec put the idea of a finite element to represent vector quantities belonging to $H(\Omega,\text{curl})$; which has been popularly known as edge elements since then [87]. This type of element correspond to 1-form element for $H(\Omega,\text{curl})$ in Fig. 3.2. They are also called as *H*(curl)-conforming elements.

In order to represent the vector quantities belonging to $H(\Omega,\text{div})$; Raviart-Thomas [88] and Brezzi-Douglas-Marini [89] have proposed the elements shown in Fig. 3.2. Unlike the Nédélec element (whose basis functions are defined along the edges); these elements have their basis functions normal to the element faces. Although Brezzi-Douglas-Marini element provides more degree of freedom, Raviart-Thomas element is more frequently used. These elements are facet elements, but usually miscalled as edge elements. They can be classified as *H*(div)-conforming elements.

Finally, in order to represent the scalar quantities belonging to $L^2(\Omega)$, required element type is certainly a volume element illustrated in Fig. 3.2. These elements are again usually miscalled as nodal elements.

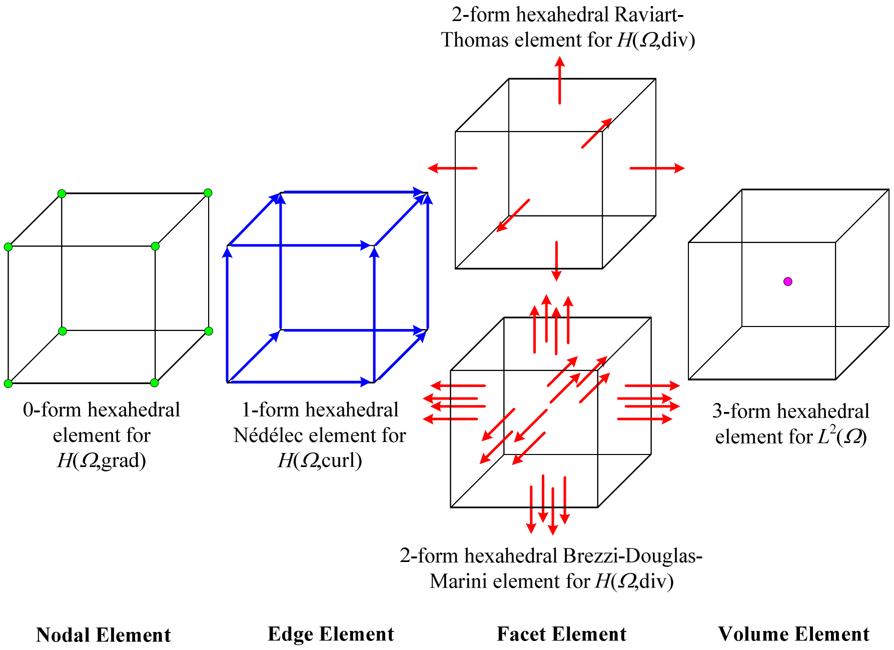

2-form hexahedral Raviart-Thomas element for $H(\Omega,\text{div})$
0-form hexahedral element for $H(\Omega,\text{grad})$
1-form hexahedral Nédélec element for $H(\Omega,\text{curl})$
3-form hexahedral element for $L^2(\Omega)$
2-form hexahedral Brezzi-Douglas-Marini element for $H(\Omega,\text{div})$
Nodal Element
Edge Element
Facet Element
Volume Element

Fig. 3.2. Hexahedral Elements for $H(\Omega,\text{grad})$, $H(\Omega,\text{curl})$, $H(\Omega,\text{div})$ and $L^2(\Omega)$ (For simplicity, elements of their first kinds are shown).

3.3. General Idea of Edge Elements

To present the main ideas associated with the edge elements, consider the hypothetical rectangular edge element shown in Fig. 3.3. Suppose that the vector variable **A** has to be defined on this rectangular element as follows:

$$\mathbf{A}(x,y)=\sum_{i=1}^{4}\mathbf{w}_i A_i \tag{3.8}$$

where $\mathbf{w}_i$ is the shape function related to one of the four edges (i.e. the *i*th edge) of the element.

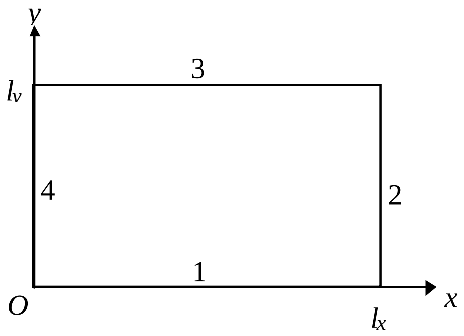


Fig. 3.3. A Rectangular Edge Element.

The shape functions of the element are

$$\mathbf{w}_1 = \mathbf{i}\left[1-\frac{y}{l_y}\right],\ \ \mathbf{w}_2 = \mathbf{j}\left[\frac{x}{l_x}\right],\ \ \mathbf{w}_3 = \mathbf{i}\left[\frac{y}{l_y}\right],\ \ \mathbf{w}_4 = \mathbf{j}\left[1-\frac{x}{l_x}\right] \tag{3.9}$$

We note that these shape functions have a direction (**i** or **j** depending on the edge) and also an expression depending on position.

Suppose that we need to obtain **A** at the centroid of the element ($x = l_x / 2$, $y = l_y / 2$). Applying these conditions to the shape functions, we obtain

$$\mathbf{w}_1 = \mathbf{i}1/2, \mathbf{w}_2 = \mathbf{j}1/2, \mathbf{w}_3 = \mathbf{i}1/2, \mathbf{w}_4 = \mathbf{j}1/2 \tag{3.10}$$

The value of **A** is, therefore,

$$\mathbf{A}(l_x/2, l_y/2) = \mathbf{i}\,A_1/2 + \mathbf{j}A_2/2 + \mathbf{i}\,A_3/2 + \mathbf{j}\,A_4/2. \tag{3.11}$$

The remaining question is the meanings of A_1, A_2, A_3, and A_4. To see this, take for example, $y = 0$. In this case we find the x component of **A**, namely A_x as

$$A_x(x,0) = A_1 \tag{3.12}$$

which means that A_1 is the projection of the vector field **A** on edge 1. Similarly, A_3 is the projection of the vector field **A** on edge 3. As y varies from 0 to l_y, the contribution of A_1 decreases and that of A_3 increases. Similarly, as x varies from 0 to l_x, the contribution of A_2 decreases and that of A_4 increases.

This example is a simple case in order to describe the idea of edge elements as vector elements. For arbitrarily shaped elements where the edges are in a direction not parallel to the Cartesian coordinate axes, the derivations may be more complicated.

3.4. The Linear and Quadratic Hexahedral Edge (van Welij's and Kameari's) Elements

A linear hexahedral edge element, which was defined by van Welij [90] with its 8 nodes and 12 edges, is illustrated in Fig. 3.4. Each hexahedral element can be defined in the local *uvp* coordinate system; and the coordinates u, v, and p range from -1 to 1 for each element. The vector basis functions (for the edges in u direction) for the linear edge elements can be written as follows:

$$\mathbf{w}_i = \frac{1}{8}(1 + v_i v)(1 + p_i p)\nabla u \tag{3.13}$$

where, $v_i = \pm 1$ and $p_i = \pm 1$, which are the v and p coordinates of the ith edge respectively. Basis functions for the edges in v and p directions can be found similarly.

Kameari's quadratic hexahedral edge element [91], with 20 nodes and 36 edges, is illustrated in Fig. 3.5. Again, for the local *uvp* coordinate system; the coordinates u, v, and p range from -1 to 1 for each element. The additional 12 nodes are defined close to the middle of physical edges of element. In each direction there are 12 edges, 8 of which are defined on the physical edges of the element, and the remaining 4 are defined on the surfaces.

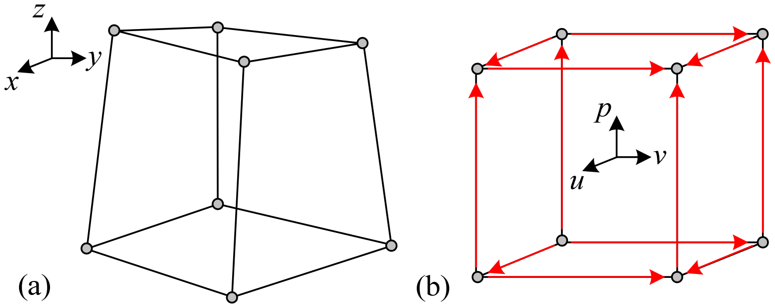


Fig. 3.4. a) Linear hexahedral element in *xyz*-space; b) Linear hexahedral element transformed to *uvp*-space with its 12 edges shown.

The vector basis functions (for the edges in *u* direction) for the quadratic edge elements can be written as:

$$\mathbf{w}_i = \frac{1}{8}(1 + v_i v)(1 + p_i p)(u_i u + v_i v + p_i p - 1)\nabla u \tag{3.14}$$

for the edges defined on the physical edges of the element, and

$$\mathbf{w}_i = \frac{1}{4}(1 + p_i p)(1 - v^2)\nabla u \tag{3.15}$$

for the edges defined on the surfaces. In equations (3.14) and (3.15), $u_i = \pm 1/2$, $v_i = \pm 1$ and $p_i = \pm 1$, which are the center point coordinates of *i*th edge. Basis functions for the edges in *v* and *p* directions can be found similarly.

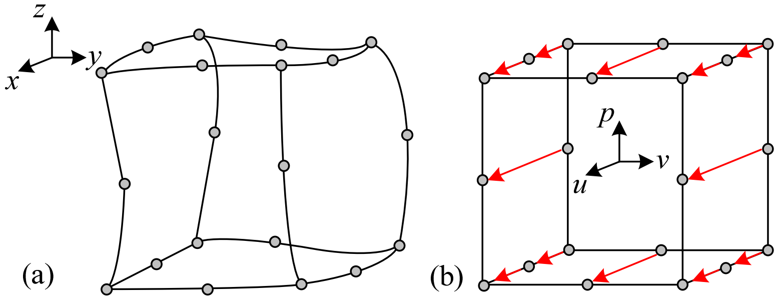


Fig. 3.5. a) Quadratic hexahedral element in *xyz*-space; b) Quadratic hexahedral element transformed to *uvp*-space with its 12 edges along *u* shown.

Physically, the values represented at each edge can be considered as the circulation (not exactly the projection) of the unknown vector function along that edge. In other words, a vector function **A** is expressed in terms of the basis functions as:

$$\mathbf{A}(x, y, z) = \sum_{i=1}^{k} l_i \mathbf{w}_i A_i \tag{3.16}$$

where k = 12 and 36 for linear and quadratic elements respectively. l_i, which is the length of an edge in *uvp*-space, is equal to 2 for all edges in case of linear elements. On the other hand, for the quadratic elements l_i = 1 for the edges defined on the physical edges of the element and l_i = 2 for the edges defined on the surfaces.

3.5. Hierarchical Finite Element Methods

In the finite element method, the discretization can be controlled two ways:

i) by varying the number of elements and/or

ii) by varying the polynomial order used to describe the displacements and the coordinates in the elements.

Note that the "*h*" in the terms "*h*-elements", "*h*-version", and "*h*-extension" refers to the length, width or height of the smallest element in the mesh. The "*p*" in the terms "*p*-elements", "*p*-version", "*p*-level", and "*p*-extension" refers to the maximum polynomial order "*p*" of the elements in the mesh.

What is an extension? Extensions are step by step changes in the FEM discretization (the mesh) that cause the number of degrees-of-freedom (DOFs) to increase at each step, with the goal of reducing numerical error in the solution.

Reduction of error can be accomplished by using any of these three extension techniques:

- In the '*h*-version," the extension is carried out by increasing the number or density of the finite elements while holding the polynomial order constant
- In the "*p*-version," the extension is carried out by increasing the polynomial level in the finite elements while maintaining the number and density of elements constant
- If the extension is carried out by increasing both the polynomial level and the number or density of elements in the mesh, the version is called the "*hp*-version." It turns out that the "*hp*-version" is the most efficient.

A general assessment and comparison of all techniques are given in Table 3.3.

Hierarchical elements are usually confused with Lagrange elements, which is another method for increasing the degree of freedom. In [92], Karanam clearly distinguished them as seen in Table 3.4.

Table 3.3. Comparison of *p, h*, and *hp*-versions.

***p*-version**	***h*-version**	***hp*-version**
Error controlled with polynomial level	Error controlled with number of elements	Error controlled with number of elements and polynomial level
Good numerical convergence for high order elements	Inferior numerical convergence (at best, quadratic)	Superior numerical convergence (exponential)
Hierarchic shape functions allows more accurate mapping of geometry shapes such as circles	Geometry shapes are mapped with quadratic functions	Hierarchic shape functions allows more accurate mapping of geometry shapes such as circles
Polynomial level in the elements can be variable	Polynomial level in the elements fixed, and restricted to linear (p=1) or quadratic (p=2)	Polynomial level can be variable; element grading recommended

Table 3.4. Comparison of Hierarchical and Lagrange Elements.

	Hierarchical Elements	**Lagrange Type Elements**
Number of Basis Functions	For a hierarchical element with *m* nodes, the number of required basis functions is calculated by cumulating the number of necessary modes. This number is $\geq m$.	For a Lagrange Type element of order *m*, the number of required basis functions is exactly *m*.
Basis Function Order	For a hierarchical element of order *n*, polynomial basis functions are of different orders ($\leq n$).	For a Lagrange type element of order *n*, all polynomial basis functions are of order *n*.
Basis Function Dependency	The basis functions of an hierarchical element of order *n* is a subset of the basis functions of an hierarchical element of higher order (i.e. It is easy and straightforward to increase the order and define higher order elements).	The basis functions of Lagrange Type elements of order *n* and *m* are totally independent if $n \neq m$.
Interpolation Properties	Basis function coefficients are actually related to higher order moments of the solution and its derivatives.	Basis function coefficients correspond to solution values at specific spatial locations.
Orientation	Definition of the basis functions depend on edge and face orientation.	Definition of the basis functions do not depend on edge and face orientation.

3.6. General Formulation of *p*-Hierarchical Hexahedral Edge Elements

There are three requirements for vector functions defined on a hierarchic edge element, i.e. they should be:

i) tangentially continuous,

ii) hierarchic (i.e. the lower polynomial order terms are used to construct the higher order polynomial terms), and

iii) *H*(curl) conforming.

A general formulation for construction of these functions for tetrahedral and hexahedral elements is given by Wang [93]. According to this paper, the *p*-hierarchic hexahedral element is defined as follows. For the general formulation of *p*-hierarchic hexahedral elements, we start with the 20-noded isoparametric mapping

$$\mathbf{r} = \sum_{j=1}^{20} N_j(\xi, \eta, \zeta)\mathbf{r}_j \tag{3.17}$$

where (ξ,η,ζ) (pronounced *xi*, *eta*, *zeta*) are local coordinates and N_j's are shape functions. Based on the mapping, the unitary triple vectors can be defined:

$$\mathbf{a}_\xi = \frac{\partial \mathbf{r}}{\partial \xi}, \quad \mathbf{a}_\eta = \frac{\partial \mathbf{r}}{\partial \eta}, \quad \mathbf{a}_\zeta = \frac{\partial \mathbf{r}}{\partial \zeta} \tag{3.18}$$

These vectors correspond to an oblique coordinate system as shown in Fig. 3.6. The gradient vectors along the three local coordinates can be easily found as:

$$\nabla \xi = \frac{1}{J}\mathbf{a}_\eta \times \mathbf{a}_\zeta, \quad \nabla \eta = \frac{1}{J}\mathbf{a}_\zeta \times \mathbf{a}_\xi, \quad \nabla \zeta = \frac{1}{J}\mathbf{a}_\xi \times \mathbf{a}_\eta \tag{3.19}$$

where J is the Jacobian determinant given by $J = (\mathbf{a}_\xi \,.\, \mathbf{a}_\eta \times \mathbf{a}_\zeta)$.

These gradient vectors naturally serve as the basis for edge elements since the tangential continuity requirement (*i*) can be easily met.

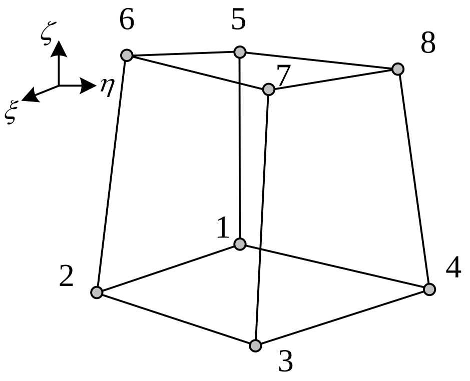


Fig. 3.6. A Hexahedral Element in an Orthogonal Coordinate System.

The family of hierarchic shape functions has initially been defined by Szabo and Babuška in [94]. For a *p*th order element:

1) There are 8 node modes given by:

$$N^{(1)} = \frac{1}{8}(1-\xi)(1-\eta)(1-\zeta), etc. \tag{3.20}$$

2) There are (*p*-1) edge modes on each edge for *p*≥2. For edge (1,2) in Fig. 3.6, these are

$$N_e^{(p1)} = \frac{1}{4}(1-\eta)(1-\zeta)\phi_{p-1}(\xi), \tag{3.21}$$

where

$$\phi_i(\xi) = \sqrt{\frac{2i-1}{2}\int_{-1}^{\xi} P_i(t)dt} \tag{3.22}$$

and $P_i(t)$ is the Legendre polynomial (see Fig. 3.7) of order i, which can be expressed by means of Rodrigues rotation formula[3] as:

$$P_i(t) = \frac{1}{2^i i!} \frac{d^i}{dt^i} \left[(t^2 - 1)^i \right] \tag{3.23}$$

3) There are (p-2) (p-3)/2 face modes on each face for $p \geq 4$. For face (1,2,3,4) in Fig. 3.6, these are given by:

$$N_f^{(p1)} = \frac{1}{2}(1-\zeta)\phi_i(\xi)\phi_j(\eta) \tag{3.24}$$

4) There are (p-3) (p-4) (p-5)/6 body modes on each face for $p \geq 6$. These are given by:

$$N_b^{(p1)} = \phi_i(\xi)\phi_j(\eta)\phi_k(\zeta) \tag{3.25}$$

Using these hierarchic functions, the requirement (*ii*) is met. To meet the requirement (*iii*), the polynomial order associated with $\nabla\xi$ is picked less one in ξ as compared to η and ζ. The construction for a pth order (H_{p-1}(curl) conforming) elements is given as follows:

1. There is one vector function per edge for p=1. For edge (1,2) it is given by

$$\mathbf{W}_e^{(1)} = \frac{1}{4}(1-\eta)(1-\zeta)\nabla\xi \tag{3.26}$$

This is nothing but van Welij’s element [90].

---

[3] More generally, Legendre polynomials are the solutions to the Legendre differential equation: $\frac{d}{dt}\left[(1-t^2)\frac{d}{dt}P(t)\right] + i(i+1)P(t) = 0$.

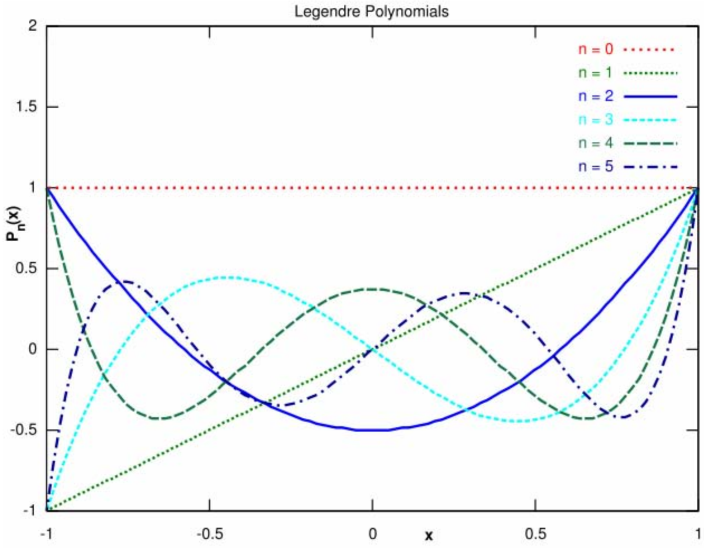


Fig. 3.7. Legendre Polynomials of Order 0 to 5 Plotted between -1 and 1.

2. There are two vector modes per edge associated with corner node functions for $p \geq 2$. These are:

$$\mathbf{W}_e^{(1)} = N^{(1)} \nabla \xi, \quad \mathbf{W}_e^{(2)} = N^{(2)} \nabla \xi \tag{3.27}$$

Setting $p=2$, we obtain exactly the Kameari's element [91].

3. There are (p-2) tangential vector functions per edge associated with edge modes for $p \geq 3$. For edge (1,2), it is given by:

$$\mathbf{W}_e^{(p1)} = N_e^{(p1)} \nabla \xi \tag{3.28}$$

4. There are 2(p-1) vector functions per face associated with edge functions (normal to edges) for $p \geq 2$. For face (1,2,3,4) in Fig. 3.6, these are given by:

$$\mathbf{W}_f^{(N1)} = \frac{1}{2}(1-\zeta)\phi_{n-1}(\xi)\nabla \eta, \quad \mathbf{W}_f^{(N2)} = \frac{1}{2}(1-\zeta)\phi_{n-1}(\eta)\nabla \xi \tag{3.29}$$

5. There are two vector functions for each face mode. For face (1,2,3,4) in Fig. 3.6, these are given by:

$$\mathbf{W}_f^{(p1)} = N_f^{(p1)}\nabla\xi,\ \ \mathbf{W}_f^{(p2)} = N_f^{(p1)}\nabla\eta \tag{3.30}$$

6. There are 3(*p*-1) vector functions per element associated with face modes (normal to faces for *p*≥4. These are:

$$\begin{aligned}\mathbf{W}_b^{(N1)} &= \phi_i(\eta)\phi_i(\zeta)\nabla\xi,\\ \mathbf{W}_b^{(N2)} &= \phi_i(\xi)\phi_i(\zeta)\nabla\eta,\\ \mathbf{W}_b^{(N3)} &= \phi_i(\xi)\phi_i(\eta)\nabla\zeta.\end{aligned} \tag{3.31}$$

7. There are three vector functions for each body mode. These are given by

$$\mathbf{W}_b^{(p1)} = N_b^{(p1)}\nabla\xi,\ \ \mathbf{W}_b^{(p2)} = N_b^{(p1)}\nabla\eta,\ \ \mathbf{W}_b^{(p3)} = N_b^{(p1)}\nabla\zeta \tag{3.32}$$

Setting *p*=2 in above equations, we obtain exactly the Kameari’s element [91].

More interesting fact is that, both van Welij and Kameari defined their elements (linear and quadratic respectively) intuitively by using the completeness condition and tangential continuity; much before these general rules for *p*-hierarchic elements were defined.

One of the key factors in the *p*-hierarchic element construction is the usage of Legendre polynomials, which is an orthogonal polynomial family. Fig. 3.7 shows the interpolation properties of Legendre polynomials of different orders.

Recently, Zaglmayr [95] defined a more general framework to define *p*-hierarchic quadrilateral, triangular, tetrahedral, hexahedral and prismatic *H*(*Ω*,curl) and *H*(*Ω*,div) elements. In this work, although not shown and verified explicitly, it is claimed that other orthogonal polynomial families, such as Gegenbauer, Hermite, or Jacobi, might also be used for similar purpose.

CHAPTER 4

FINITE ELEMENT FORMULATION OF ELECTROMAGNETIC SCATTERING PROBLEMS

4.1. Weak Formulation of the Electric Field

The most general three-dimensional electromagnetic scattering problem can be stated as an electromagnetic wave ($\mathbf{E}^{\mathbf{inc}}$, $\mathbf{H}^{\mathbf{inc}}$) with any type of polarization at any frequency, incident on a scattering material of arbitrary material properties and shape occupying a volume Ω_{int}. This geometry is illustrated in Fig. 4.1, where Ω_{ext} is the exterior domain outside the scatterer. $\Omega = \Omega_{\text{int}} \cup \Omega_{\text{ext}}$ is the computational domain. $S(\Omega)$ is the boundary of Ω_{int}.

Starting from the Maxwell's equations in differential form we have

$$\nabla \times \mathbf{E}^{\mathbf{tot}} = -j\omega\mu_0\mu_r\mathbf{H}^{\mathbf{tot}} \tag{4.1}$$

$$\nabla \times \mathbf{H}^{\mathbf{tot}} = j\omega\varepsilon_0\varepsilon_r\mathbf{E}^{\mathbf{tot}} \tag{4.2}$$

where the total fields $\mathbf{E}^{\mathbf{tot}}$ and $\mathbf{H}^{\mathbf{tot}}$ are the sum of the incident and the scattered fields (i.e. $\mathbf{E}^{\mathbf{tot}} = \mathbf{E}^{\mathbf{inc}} + \mathbf{E}^{\mathbf{sct}}$ and $\mathbf{H}^{\mathbf{tot}} = \mathbf{H}^{\mathbf{inc}} + \mathbf{H}^{\mathbf{sct}}$) and it is assumed that the materials are isotropic but inhomogenous with the relative permittivity ε_r and permeability μ_r. In Ω_{ext}, ε_r and μ_r are assumed to be unity.

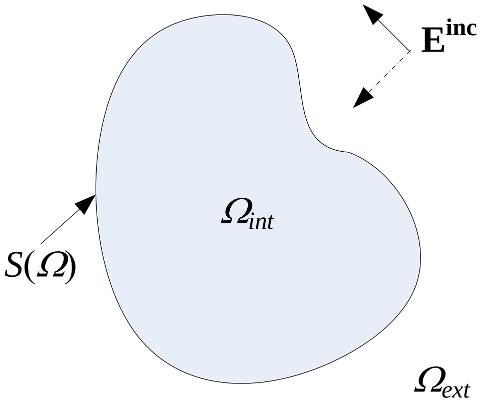


Fig. 4.1. Scattering material enclosed by $S(\Omega)$ and an incident wave $\mathbf{E^{inc}}$.

Substituting Equation (4.2) into Equation (4.1) we get the wave equation as

$$\nabla \times \frac{1}{\mu_r} \nabla \times \mathbf{E^{tot}} - k_0^2 \varepsilon_r \mathbf{E^{tot}} = 0 \tag{4.3}$$

subject to a given set of boundary conditions on the surface $S(\Omega)$. Choosing a vector-valued function φ defined on Ω, the inner product of Equation (4.3) with φ gives

$$\int_{\Omega} \left(\nabla \times \frac{1}{\mu_r} \nabla \times \mathbf{E^{tot}} \right) \cdot \varphi d\Omega - \int_{\Omega} \left(k_0^2 \varepsilon_r \mathbf{E^{tot}} \right) \cdot \varphi d\Omega = 0 \tag{4.4}$$

To reduce this integral representation to a simplified form, we utilize the following vector identities for any arbitrary vector functions **P** and **Q**:

$$\int_{\tilde{\Omega}} (\nabla \times \mathbf{P}) \cdot \mathbf{Q} d\Omega = \int_{\tilde{\Omega}} \mathbf{P} \cdot (\nabla \times \mathbf{Q}) d\Omega + \int_{\tilde{\Omega}} \nabla \cdot (\mathbf{P} \times \mathbf{Q}) d\Omega \tag{4.5}$$

$$\int_{\tilde{\Omega}} \nabla \cdot (\mathbf{P} \times \mathbf{Q}) d\Omega = \int_{S(\tilde{\Omega})} (\mathbf{P} \times \mathbf{Q}) \cdot d\mathbf{s} \tag{4.6}$$

where $\tilde{\Omega}$ is a domain enclosed by $S(\tilde{\Omega})$ .Using Equation (4.5) and Equation (4.6) and assuming that Ω_{int} is occupied by a perfectly conducting scatterer we obtain the following weak form:

$$\int_{\Omega_{ext}} \left(\nabla\times\mathbf{E}^{\mathbf{tot}}\right)\cdot\left(\nabla\times\varphi\right)d\Omega - \int_{S(\Omega)} \left(\nabla\times\mathbf{E}^{\mathbf{tot}}\right)\times\varphi\cdot d\mathbf{s} - \int_{\Omega_{ext}} k_0^2\mathbf{E}^{\mathbf{tot}}\cdot\varphi d\Omega = 0 \tag{4.7}$$

The surface integral term vanishes because of the boundary condition on the surface *S(Ω)*.

$$\mathbf{n}\times\mathbf{E}^{\mathbf{tot}} = 0 \tag{4.8}$$

Since the total electric field vanishes in Ω_{int} we obtain the following weak form for the scattered field $\mathbf{E}^{\mathbf{sct}}$

$$\mathbf{E}^{\mathbf{sct}} = -\mathbf{E}^{\mathbf{inc}} \text{ in } \Omega_{\text{int}} \tag{4.9}$$

$$\int_{\Omega_{ext}} \left(\nabla\times\mathbf{E}^{\mathbf{sct}}\right)\cdot\left(\nabla\times\varphi\right)d\Omega - \int_{\Omega_{ext}} \left(k_0^2\mathbf{E}^{\mathbf{sct}}\right)\cdot\varphi d\Omega = 0 \tag{4.10}$$

If the material is a dielectric with relative permittivity ε_r, we have the following weak form

$$\int_{\Omega} \left(\frac{1}{\mu_r}\nabla\times\mathbf{E}^{\mathbf{sct}}\right)\cdot\left(\nabla\times\varphi\right)d\Omega - \int_{\Omega} \left(k_0^2\varepsilon_r\mathbf{E}^{\mathbf{sct}}\right)\cdot\varphi d\Omega = \int_{\Omega_{\text{int}}} \left(k_0^2\left(\varepsilon_r - 1\right)\mathbf{E}^{\mathbf{inc}}\right)\cdot\varphi d\Omega \tag{4.11}$$

Or more generally, if the material is a composite structure with relative permittivity ε_r and relative permeability μ_r, then we have the following weak form

$$\begin{aligned}
&\int_{\Omega}\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}^{\mathbf{sct}}\right)\cdot(\nabla\times\varphi)d\Omega-\int_{\Omega}\left(k_0^2\varepsilon_r\mathbf{E}^{\mathbf{sct}}\right)\cdot\varphi d\Omega= \\
&\int_{\Omega_{\mathrm{int}}}\left(\left(1-\frac{1}{\mu_r}\right)\nabla\times\nabla\times\mathbf{E}^{\mathbf{inc}}\right)\cdot\varphi d\Omega+\int_{\Omega_{\mathrm{int}}}\left(k_0^2(\varepsilon_r-1)\mathbf{E}^{\mathbf{inc}}\right)\cdot\varphi d\Omega
\end{aligned} \tag{4.12}$$

4.2. Mesh Generation

The mesh generation in this thesis depends on the decomposition of the problem to subdomains so that each subdomain is homeomorphic (topologically equivalent) to a rectangular prism. Each subdomain is divided to hexahedra with the constraint that adjacent subdomains will have equivalent quadrilateral surface meshes in order to preserve mesh continuity.

Details about the implementation of the domain decomposition and hexahedral mesh generation (together with the constraints) are given in Appendix C.

4.3. Mesh Quality Improvement

In this thesis, optimization based hexahedral mesh smoothing was applied. The objective function was chosen to be depending on a condition number based metric; and Particle Swarm Optimization was applied for smoothing. Effect of smoothing on solution accuracy was also investigated. Theoretical and implementation level details are given in Appendix D.

4.4. Mesh Truncation

The concept of using a lossy material to absorb an outgoing wave to simulate an infinite region of free space for finite methods is not a new one [96]. However, this method of truncation has not gained widespread use because of the reflections, which occur at the free space/material interface. One idea to minimize the reflections at this interface is to choose a low loss absorbing material. Unfortunately, the lossy region must be sufficiently large to attenuate the wave. This can significantly reduce the computational efficiency.

Berenger [97] introduced a modification to Maxwell’s equations to allow for the specification of material properties which result in a reflectionless lossy material. The material is reflectionless in the sense that a plane wave propagating through an infinite free space/material interface has no reflection for all angles of incidence. Berenger refers to this material as a “perfectly matched layer (PML)”. Although Berenger demonstrates the validity of his approach with numerical experiments, the physical meaning of his modifications to the Maxwell’s equations is not very clear. Later, Chew and Weedon [98] provided a systematic analysis of the PML in terms of the concept of “coordinate stretching”. They demonstrated that Berenger’s modifications to Maxwell’s equations can be derived from a more generalized form of Maxwell’s equations employing complex coordinates.

Later, it has also been discovered by Sacks *et al* [99] that the reflectionless properties of a material can be achieved if the material is assumed to be anisotropic. Unlike Berenger’s approach, this one does not require a modification of Maxwell’s equations, making it easier to analyze in the general framework of electromagnetics.

4.4.1. Analytical Investigation of PMLs

The ideal PML comprises an anisotropic medium, whose complex permittivity and permeability matrices are chosen such that it absorbs an arbitrary incident electromagnetic wave with no reflection. Initially, the PMLs have been designed to absorb planar electromagnetic waves, of arbitrary frequency and incident angle, that are incident from free space onto the PML half-space. Later it has been shown by Kuzuoğlu and Mittra [100] that, under certain conditions, the PMLs can effectively absorb spherical and cylindrical waves as well.

4.4.1.1. PMLs for Cartesian Coordinate System

A thorough discussion of PMLs suitable for Cartesian geometries has been presented in [99], where the transmission and reflection characteristics of a planar free-space PML has been detailed. One begins by dividing the three-dimensional (3D) space into two half-spaces with free space for $z < 0$ and an anisotropic medium for $z > 0$, where z is normal to the interface. The constitutive parameters of the anisotropic medium are given in terms of the complex permittivity and permeability tensors

$$\begin{aligned} \overset{=}{\varepsilon} &= \varepsilon_0 [\Lambda_z] \\ \overset{=}{\mu} &= \mu_0 [\Lambda_z] \end{aligned} \tag{4.13}$$

and $[\Lambda_z]$ is a diagonal matrix defined as

$$\Lambda_z = \begin{bmatrix} a & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a^{-1} \end{bmatrix} \tag{4.14}$$

where

$$a = 1 - j\frac{\sigma}{\omega\varepsilon_0} \tag{4.15}$$

and σ is the constant conductivity of the medium.

For the case where the three-dimensional (3D) space is divided into two half-spaces with free space for $y < 0$ and an anisotropic medium for $y > 0$, where y is normal to the interface, the constitutive parameters of the anisotropic medium are

$$\begin{aligned} \overset{=}{\varepsilon} &= \varepsilon_0 [\Lambda_y] \\ \overset{=}{\mu} &= \mu_0 [\Lambda_y] \end{aligned} \tag{4.16}$$

where

$$\Lambda_y = \begin{bmatrix} a & 0 & 0 \\ 0 & a^{-1} & 0 \\ 0 & 0 & a \end{bmatrix} \tag{4.17}$$

Similarly, by dividing the three-dimensional (3-D) space into two half-spaces with free space for $x < 0$ and an anisotropic medium for $x > 0$, where y is normal to the interface, the constitutive parameters of the anisotropic medium are obtained as

$$\begin{aligned} \overset{=}{\varepsilon} &= \varepsilon_0 [\Lambda_x] \\ \overset{=}{\mu} &= \mu_0 [\Lambda_x] \end{aligned} \tag{4.18}$$

where

$$\Lambda_x = \begin{bmatrix} a^{-1} & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & a \end{bmatrix} \tag{4.19}$$

In this work, the coordinate stretching technique is used. In the coordinate stretching technique, the spatial variable u is replaced by the complex spatial variable u' given by

$$u' = u\left(1 - j\frac{\sigma}{\omega\varepsilon_0}\right) = u\left(1 - j\frac{\alpha}{k}\right) \tag{4.20}$$

assuming that the wave is propagating in the u-direction. The spatial variable u can be either x, y, or z. The variable α is the parameter determining how fast the field decays inside the Perfectly Matched Layer.

4.4.1.2. Modelling of Edge and Corner Regions of the PML

The theory for the anisotropic PML is based on the assumption that the plane wave is propagating through a planar interface of infinite extent as explained above. However, for the cases where the main interest is to use the PML in order to absorb the scattered field from a finite object in free space, the PML absorber must totally surround the scattering material.

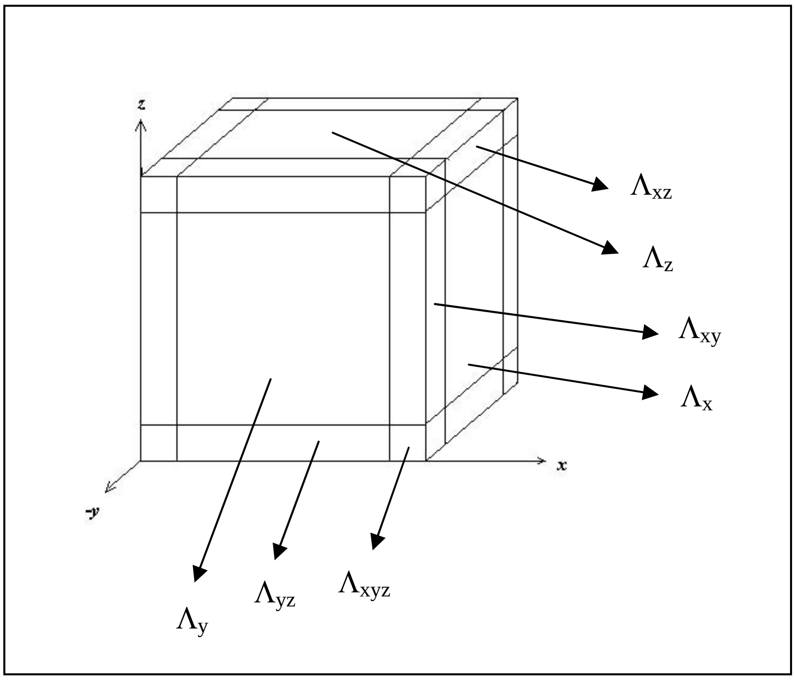


Fig. 4.2. Geometry of the PML region surrounding the scatterer.

The PML material is usually placed in the shape of a prism to best approximate the reflectionless properties of the PML as shown in Fig. 4.2.

The choice for the material properties of the side regions occupied by the PML is straightforward and the Λ_x, Λ_y, Λ_z matrices above can be used. However, the method for determining the material properties at the edge and the corner regions of the box is not so clear.

One approximate approach for the edge region is to choose the edge properties such that they are perfectly matched to the adjacent side regions when the edge-side interface is of infinite extent.

One can use a similar approach for the corner region by matching the corner properties to the adjacent edges. If we repeat the corresponding analysis for a plane wave propagating through an interface for the edges and corners, we arrive the relationship that Λ_{ij}, which is the corresponding matrix, is equal to the matrix multiplication of the corresponding matrices, namely Λ_i and Λ_j. Similarly, for the corner regions, Λ_{ijk} is found to be the multiplication of Λ_i, Λ_j, and Λ_k.

4.4.1.3. Conformal PMLs

The anisotropic PML has also been realized by applying a complex coordinate transformation in the direction normal to the PML-free space boundary by Chew *et al* [101], which yields the design of conformal PMLs. The approach can be considered and called as locally conformal PML, since at every point in the PML-free space interface, the normal vector is considered during the complex coordinate transformation. This approach is easy to implement and realize.

4.5. Elemental Matrix Construction

In the derivation of the FEM formulation, the weighting function φ has been used in the inner products. To obtain the matrix equation, the weighting function φ is chosen to be identical to the vector basis functions, namely

$$\int_{\Omega_{ext}} \left(\frac{1}{\mu_r} \nabla \times \mathbf{E}^{\mathbf{sct}} \right) \cdot \left(\nabla \times \mathbf{w}_m \right) d\Omega - \int_{\Omega_{ext}} \left(k_0^2 \varepsilon_r \mathbf{E}^{\mathbf{sct}} \right) \cdot \mathbf{w}_m d\Omega = 0 \tag{4.21}$$

The scattered electric field inside an element is also expressed as

$$\mathbf{E}^{\mathbf{sct}} = \sum_{i=1}^{12} \mathbf{w}_i (\mathbf{r}) E_i \tag{4.22}$$

The left hand side of Equation (4.21) will be the following

$$\int_{V_i}\left(\frac{1}{\mu_r}\nabla\times\sum_{n=1}^{12}\mathbf{w}_n E_n\right)\cdot\left(\nabla\times\mathbf{w}_m\right)dV-\int_{V_i}\left(k_0^2\varepsilon_r\sum_{n=1}^{12}\mathbf{w}_n E_n\right)\cdot\mathbf{w}_m dV \tag{4.23}$$

where V_i is the volume of the ith element. This implies that for an element, a generic term of the elemental matrix E is given as

$$E(m,n)=\int_{V_i}\left(\frac{1}{\mu_r}\nabla\times\mathbf{w}_n\right)\cdot\left(\nabla\times\mathbf{w}_m\right)dV-\int_{V_i}\left(k_0^2\varepsilon_r\mathbf{w}_n\right)\cdot\mathbf{w}_m dV \tag{4.24}$$

Similarly, a generic term for the right hand side matrix R of the corresponding element will be

$$R(m,1)=\int_{V_i}\left(k_0^2\left(\varepsilon_r-1\right)\mathbf{E}^{\mathbf{inc}}\right)\cdot\mathbf{w}_m dV \tag{4.25}$$

where ε_r is the relative permittivity of the medium.

As shown in the previous section, the remaining problem is to carry out the integration in the elemental matrix term. Since the shape functions are sufficiently smooth, a low order Gaussian Integration scheme is assumed to be accurate for this purpose. By using Gaussian Integration, Equation (4.24) becomes

$$\begin{aligned}E(m,n)=\sum_{i=1}^{N_G}&W_i\left(\frac{1}{\mu_r}\nabla\times\mathbf{w}_n\left(u_i,v_i,p_i\right)\right)\cdot\left(\nabla\times\mathbf{w}_m\left(u_i,v_i,p_i\right)\right)\\&-W_i\left(k_0^2\varepsilon_r\mathbf{w}_n\left(u_i,v_i,p_i\right)\right)\cdot\mathbf{w}_m\left(u_i,v_i,p_i\right)\end{aligned} \tag{4.26}$$

where N_G is the number of Gaussian integration points inside an element, (u_i, v_i, p_i) are the possible N_G combinations of the Gaussian integration points, and W_i's are the weightings of the corresponding points. Details about the Gaussian quadrature are given in the following sub-section.

4.5.1. Gaussian Quadrature

The integral is computed by means of *n*-point Gaussian quadrature[4], which can be described as follows. For an arbitrary function *f*(*x*),

$$\int_{-1}^{1} f(x)dx \cong \sum_{i=1}^{n} w_i f(x_i) \tag{4.27}$$

where the evaluation points and the weights are defined as in Table 4.1. It can be shown that the evaluation points are just the roots of a polynomial belonging to a class of orthogonal polynomials; and the values in Table 4.1 are the roots of Legendre polynomials.

Table 4.1. Gaussian Quadrature Evaluation Points and Their Weights.

Number of Quadrature Points (n)	Points (x_i)	Weights (w_i)
1	0	2
2	$\pm\sqrt{1/3}$	1
3	0	8/9
	$\pm\sqrt{3/5}$	5/9
4	±0.339981044	0.652145155
	±0.861136312	0.347854845
5	0	0.568889
	±0.538469	0.478629
	±0.906180	0.236927

---

[4] In numerical analysis, a quadrature rule is an approximation of the definite integral of a function, usually stated as a weighted sum of function values at specified points within the domain of integration.

Increasing the number n will increase the accuracy of the integration. Certainly, the rule described above can be generalized to any arbitrary interval [a,b] different than [-1,1].

$$\int_a^b f(x)dx \cong \frac{b-a}{2}\sum_{i=1}^{n} w_i f\left(\frac{b-a}{2}x_i + \frac{b+a}{2}\right) \tag{4.28}$$

For double and triple integrals, all combinations of the Gaussian quadrature evaluation points and their multiplied weights should be considered in the summation.

Recently another quadrature, Clenshaw-Curtis quadrature has become popular. The quadrature points are calculated from the Chebyshev polynomials. The computation of the Clenshaw-Curtis quadrature points is computationally cheaper than the computation of those of Gaussian quadrature; although in terms of accuracy Gaussian is better. In applications where it is required to compute the points during the run-time, Clenshaw-Curtis is preferred. In this thesis, since the look-up tables are constructed initially and loaded to the memory in the compile-time; Gaussian quadrature is used.

4.6. Sparsity and Resource Requirements in the Finite Element Method

4.6.1. Sparse Matrix Storage Schemes

Since the global system matrix is sparse, instead of full storage of complexity O(N^2), the row indexed sparse storage scheme described by Bentley [102] of complexity O(N) is chosen in this work. The storage scheme is as follows:

To represent a matrix **A** of dimension $N \times N$, two one-dimensional arrays *sa* and *ija* are set. The former stores the matrix element values in the desired precision, and the latter stores integer values. The storage rules are listed below:

- The first N locations of *sa* store the diagonal matrix elements of **A** in order. This implies that zero diagonal elements are also stored.
- Each of the first N locations of *ija* stores the index of the array *sa* that contains the first off-diagonal element of the corresponding row of the matrix. If there are no off-

diagonal elements for that row, it is one greater than the index in *sa* of the most recently stored element of a previous row.

- Location 1 of *ija* is always equal to $N + 2$, which can be used to determine N.
- Location $N + 1$ of *ija* is one greater than the index in *sa* of the last off-diagonal element of the last row. It can be used to determine the number of nonzero elements in the matrix, of the number of elements in the arrays *sa* and *ija*. Location $N + 1$ of *sa* is not used and can be set arbitrarily.
- Entries in *sa* at locations $\geq N + 2$ contain off-diagonal entries of **A**, ordered by rows and, ordered by columns within each row.
- Entries in *ija* at locations $\geq N + 2$ contain the column number of the corresponding element in *sa*.

As an example, the following 5×5 matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & 0 & a_{13} & 0 & 0 \\ 0 & a_{22} & 0 & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 \\ 0 & 0 & 0 & 0 & a_{45} \\ 0 & 0 & 0 & a_{54} & a_{55} \end{bmatrix} \tag{4.29}$$

will be stored in the arrays

$$\begin{aligned} ija &= \begin{bmatrix} 7 & 8 & 8 & 10 & 11 & 12 & 3 & 2 & 4 & 5 & 4 \end{bmatrix} \\ sa &= \begin{bmatrix} a_{11} & a_{22} & a_{33} & 0 & a_{55} & x & a_{13} & a_{32} & a_{34} & a_{45} & a_{54} \end{bmatrix} \end{aligned} \tag{4.30}$$

Another sparse storage scheme is chosen in this work for the Multifrontal Algorithm. The same matrix **A**, will be represented by the arrays

$$\begin{aligned} ia &= \begin{bmatrix} 1 & 3 & 4 & 7 & 8 & 10 \end{bmatrix} \\ ja &= \begin{bmatrix} 1 & 3 & 2 & 2 & 3 & 4 & 5 & 4 & 5 \end{bmatrix} \\ a &= \begin{bmatrix} a_{11} & a_{13} & a_{22} & a_{32} & a_{33} & a_{34} & a_{45} & a_{54} & a_{55} \end{bmatrix} \end{aligned} \tag{4.31}$$

4.6.2. Sparsity Issues for *p*-Hierarchical Quadrilateral Edge Elements

Assume that the computational domain in 2D is divided into N and L segments in x and y directions respectively, yielding a mesh with totally NL quadrilateral elements as seen in Fig. 4.3.

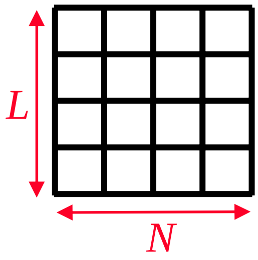


Fig. 4.3. An NL-Quadrilateral Element Mesh of a Rectangular Region.

Obviously, the number of resulting unknowns would be different for the same mesh if the types of the elements (node, edge, facet, or volume) change. Table 4.2 lists the number of unknowns for the same mesh for each element type where all the elements are assumed to be linear.

Table 4.2. Numbers of Unknowns for the NL-Quadrilateral Element Mesh.

	Linear Quadrilateral Node Element	Linear Quadrilateral Edge Element	Linear Quadrilateral Facet Element	Linear Quadrilateral Volume Element
Number of Unknowns	$(N+1)(L+1)$	$2NL+N+L$	$2NL+N+L$	NL

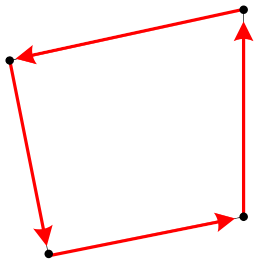


Fig. 4.4. A Linear Quadrilateral Edge Element with 4 Nodes and 4 Edges.

A linear quadrilateral edge element is defined with 4 nodes and 4 edges as seen in Fig. 4.4. From Table 4.2, it can be seen that for linear quadrilateral edge elements, the total number of edges (unknowns) is

$$N_{linear\_quadrilateral\_edge\_element} = 2NL + N + L \tag{4.32}$$

As seen in Fig. 4.5, topologically, an edge can be shared by two quadrilateral elements inside a 2D mesh. This means that: for linear quadrilateral edge elements, the global stiffness matrix will have at most 7 nonzero entries at each row; since the number of total edges in a 2-element mesh is 7.

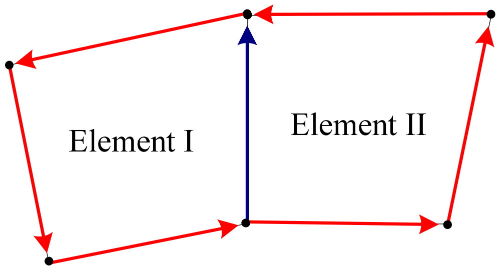


Fig. 4.5. An Edge Shared by 2 Linear Quadrilateral Edge Elements.

The sparsity of the global stiffness matrix for the NL-linear quadrilateral edge element mesh can be calculated as follows (where the sparsity of a matrix is defined as the ratio of zero entries to the whole):

$$
\begin{aligned}
& sparsity_{linear\_quadrilateral\_edge\_element} = \frac{\text{number of zero entries}}{\text{number of total entries}} \\
& = 1 - \frac{7(2NL+N+L)}{(2NL+N+L)^2} = 1 - \frac{7}{(2NL+N+L)}
\end{aligned}
\tag{4.33}
$$

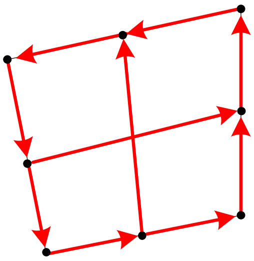


Fig. 4.6. A Quadratic Quadrilateral Edge Element with 8 Nodes and 10 Edges.

On the other hand, a quadratic quadrilateral edge element is defined with 8 nodes and 10 edges as seen in Fig. 4.6. If the elements in the mesh are quadratic, then the total number of edges (unknowns) becomes

$$
N_{quadratic\_quadrilateral\_edge\_element} = 6NL + 2N + 2L \tag{4.34}
$$

As seen in Fig. 4.7: for quadratic quadrilateral edge elements, the global stiffness matrix will have at most 18 nonzero entries at each row; since the number of total edges in a 2-element mesh is 18.

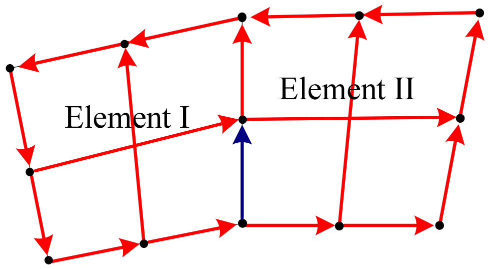


Fig. 4.7. An Edge Shared by 2 Quadratic Quadrilateral Edge Elements.

Hence, the sparsity of the global stiffness matrix for the *NL*-quadratic quadrilateral edge element mesh is:

$$sparsity_{quadratic\_quadrilateral\_edge\_element} = 1 - \frac{18(6NL + 2N + 2L)}{(6NL + 2N + 2L)^2} = 1 - \frac{18}{(6NL + 2N + 2L)} \tag{4.35}$$

As a specific numerical example, let the computational domain be a square of size $\lambda \times \lambda$.

i) If linear elements are to be used, the element size should be chosen about 0.1λ due to accuracy concerns. This means that $N = L = 10$; which yields 100 elements and 220 unknowns. For this configuration, the sparsity of the global stiffness matrix is 1-(7/220) = 0.9682.

ii) If quadratic elements are to be used, the element size should be chosen about 0.33λ due to same accuracy concerns. This means that $N = L = 3$; which yields 9 elements and 66 unknowns. For this configuration, the sparsity of the global stiffness matrix is 1-(18/66) = 0.727.

This shows that in *p*-version finite element analysis, the sparsity of the global stiffness matrix dramatically decreases with increasing *p*. This analysis can be expanded for p=3 or more; and other elements of different shapes and types.

4.6.3. Sparsity Issues for *p*-Hierarchical Hexahedral Edge Elements

The same steps carried out for quadrilateral elements will be repeated for hexahedral elements. Again, assume that the computational domain in 3D is divided into N, L and M segments in x, y and z directions respectively, yielding a mesh with totally NLM hexahedral elements as seen in Fig. 4.8.

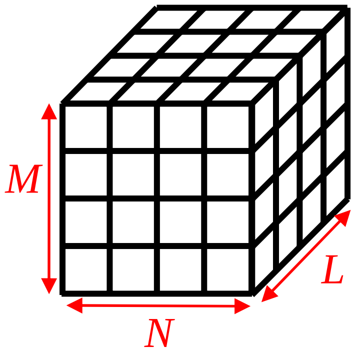


Fig. 4.8. An *NLM*-Hexahedral Element Mesh of a Rectangular Prismic Region.

Again, the number of resulting unknowns would be different for the same mesh if the types of the elements (node, edge, facet, or volume) change. Table 4.3 lists the number of unknowns for the same mesh for each element type where all the elements are assumed to be linear.

Table 4.3. Numbers of unknowns for the *NLM*-hexahedral element mesh.

	Number of Unknowns
Linear Hexahedral Node Element	$(N+1)(L+1)(M+1)$
Linear Hexahedral Edge Element	$M(N+1)(L+1)+(M+1)(2LN+N+L)$
Linear Hexahedral Facet Element	$M(N+1)(L+1)+(M+1)(2LN+N+L)$
Linear Hexahedral Volume Element	NLM

A linear hexahedral edge (van Welij's) element is defined with 8 nodes and 8 edges. From Table 4.3, it can be seen that for van Welij's elements, the total number of edges (unknowns) is

$$N_{linear\_hexahedral\_edge\_element} = M(N+1)(L+1) + (M+1)(2LN+N+L) \tag{4.36}$$

As seen in Fig. 4.9, topologically, an edge can be shared by four hexahedral elements inside a 3D mesh. This means that: for van Welij's elements, the global stiffness matrix will have at most 33 nonzero entries at each row; since the number of total edges in a 4-element mesh is 33.

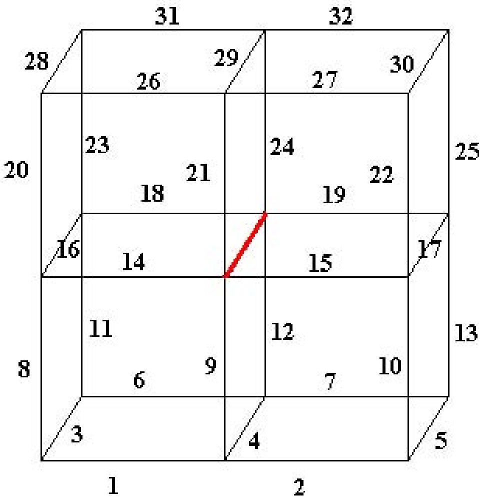


Fig. 4.9. An Edge Shared by 4 Linear Hexahedral Edge (van Welij's) Elements.

Hence, the sparsity of the global stiffness matrix for the *NLM*-van Welij's element mesh is:

$$\begin{aligned} sparsity_{linear\_hexahedral\_edge\_element} &= sparsity_{van\_Welij's\_element} = \\ &= 1 - \frac{33\left[M(N+1)(L+1)+(M+1)(2LN+N+L)\right]}{\left[M(N+1)(L+1)+(M+1)(2LN+N+L)\right]^2} \\ &= 1 - \frac{33}{\left[M(N+1)(L+1)+(M+1)(2LN+N+L)\right]} \end{aligned} \tag{4.37}$$

A quadratic hexahedral edge (Kameari's) element is defined with 20 nodes and 38 edges. For Kameari's elements, it can be calculated the total number of edges (unknowns) is

$$N_{quadratic\_hexahedral\_edge\_element} = M(2N^2+6NL+N+L+2)+(M+1)(6LN+2N+2L) \tag{4.38}$$

Since topologically an edge can be shared by four hexahedral elements inside a 3D mesh; for Kameari's elements, the global stiffness matrix will have at most 106 nonzero entries at each row; since the number of total edges in a 4-element mesh is 106 as seen in Fig. 4.10.

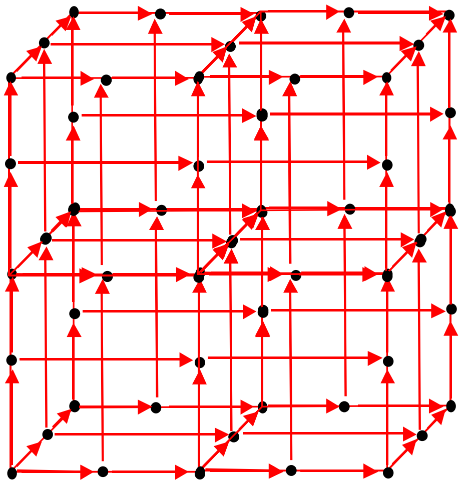


Fig. 4.10. An Edge Shared by 4 Quadratic Hexahedral Edge (Kameari's) Elements.

Hence, the sparsity of the global stiffness matrix for the *NLM*-Kameari's element mesh is:

$$\begin{aligned}
& sparsity_{quadratic\_hexahedral\_edge\_element} = sparsity_{Kameari's\_element} = \\
& = 1 - \frac{106\left[M(2N^2 + 6NL + N + L + 2) + (M+1)(6LN + 2N + 2L)\right]}{\left[M(2N^2 + 6NL + N + L + 2) + (M+1)(6LN + 2N + 2L)\right]^2} \\
& = 1 - \frac{106}{\left[M(2N^2 + 6NL + N + L + 2) + (M+1)(6LN + 2N + 2L)\right]}
\end{aligned} \tag{4.39}$$

As a specific numerical example, let the computational domain be a cube of size $\lambda\times\lambda\times\lambda$.

i) If van Welij's elements are to be used, the element size should be chosen about 0.1λ due to accuracy concerns. This means that $N = L = M = 10$; which yields 1000 elements and 3630 unknowns. For this configuration, the sparsity of the global stiffness matrix is 1-(33/3630) = 0.9909.

ii) If Kameari's elements are to be used, the element size should be chosen about 0.33λ due to same accuracy concerns. This means that $N = L = M = 3$; which yields 27 elements and 504 unknowns. For this configuration, the sparsity of the global stiffness matrix is 1-(106/504) = 0.7897.

This again shows that in *p*-version finite element analysis, the sparsity of the global stiffness matrix dramatically decreases with increasing *p*. Certainly this analysis can also be expanded for *p*=3 or more; and other elements of different shapes and types.

4.6.4. Resource Requirements

Consider our volume of interest (i.e. the problem domain), which is a rectangular prism, is divided into $N \times L \times M$ hexahedral elements as in the previous subsection.

It is obvious that the total number of elements is $N \times L \times M$, and the significant term is the total number of edges, which is closely related to the number of unknowns. For this case, if linear hexahedral edge elements are used, the total number of edges (N_{edges}) is found as follows:

$$N_{edges\_linear} = M(N+1)(L+1) + (M+1)(2LN + N + L) \tag{4.40}$$

For the case of quadratic edge elements, the total number of edges is given by

$$N_{edges\_quadratic} = M(2N^2 + 6NL + N + L + 2) + (M+1)(6LN + 2N + 2L) \tag{4.41}$$

We know that, Bentley's sparse storage scheme for a $K \times K$ matrix by requires an array of size (K + 1 + $N_{off\_diagonal}$) where $N_{off\_diagonal}$ is the total number of nonzero off-diagonal terms in the matrix.

For the finite element solution, an edge can be shared by 4 elements. This means that, a row can have at most 33 nonzero entries (one of which is on the diagonal) for the linear elements. Similarly, a row can have at most 106 nonzero entries (one of which is on the diagonal) for the quadratic elements.

Hence, the array size in the linear hexahedral finite element solution is

$$N_{array\_linear} = N_{edges\_linear} + 1 + (105 \times N_{edges\_linear}) = 106 \times N_{edges\_linear} + 1 \tag{4.42}$$

whereas the array size in the quadratic hexahedral finite element solution is

$$N_{array\_quadratic} = N_{edges\_quadratic} + 1 + (105 \times N_{edges\_quadratic}) = 106 \times N_{edges\_quadratic} + 1 \tag{4.43}$$

Table 4.4 gives some numerical values for the linear and the quadratic cases. It exhibits that, for such a problem 512-quadratic-element scheme is computationally as expensive as the 8,000-linear-element scheme. In order to use the quadratic element scheme, it should give as good results as the corresponding (i.e. computationally having the same price) linear element scheme.

Table 4.4. Comparison between linear and quadratic hexahedral edge element schemes.

N	*L*	*M*	**Total Number of Elements**	**Linear**		**Quadratic**	
				Total Number of Edges	**Array Size**	**Total Number of Edges**	**Array Size**
8	8	8	512	1,944	**64,153**	7,984	**846,305**
10	10	10	1,000	3,630	**119,791**	15,260	**1,617,561**
12	12	12	1,728	6,084	**200,773**	25,992	**2,755,153**
15	15	15	3,375	11,520	**380,161**	50,040	**5,304,241**
18	18	18	5,832	19,494	**643,303**	85,644	**9,078,265**
20	20	20	8,000	26,460	**873,181**	116,920	**12,393,521**

4.7. Sparse Matrix Solvers

Sparse matrices, which are eventually encountered, allow the developers and programmers only to store and perform the operations over only nonzero entries in most cases. A system of equations with a sparse matrix can be solved by direct or indirect methods.

4.7.1. General Assessment About Sparse Matrix Solvers

Table 4.5 gives an overall assessment about the sparse solvers.

Table 4.5. Overall View of the Sparse Solvers.

	Direct	**Iterative**	
Non-Symmetric	Pivoting LU	GMRES, QMR, etc .	More General ↕
Symmetric Positive Definite	Cholesky	Conjugate Gradient	More Robust

More Storage ↔ Less Storage

For evaluation the complexity of the direct solvers, the following figures of merit can be considered in the following scenario: Assume that we have uniform (well shaped) finite element meshes in 2D and 3D, each with *n* elements totally as seen in Fig. 4.11.

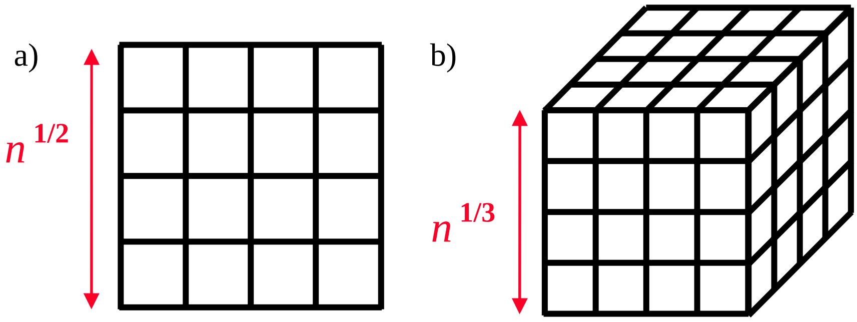


Fig. 4.11. Uniform Meshes in 2D and 3D with *n* elements.

Then, for the case of node elements, the storage and the time (FLoating point OPerations) complexities for direct solvers are as seen in Table 4.6 [103].

Table 4.6. Time and Memory Requirements for Direct Solvers [103].

	2D	**3D**
Memory	$O(n \log n)$	$O(n^{1.33})$
Time (FLOPs)	$O(n^{1.5})$	$O(n^{2})$

Again, for the case of node elements, the storage and the time complexities for several iterative solvers are as seen in Table 4.7 [103]. For element types other than the node elements (edge, facet, or volume) not exactly same but similar figures of merit might be found.

Table 4.7. Time Requirements for Some Indirect Solvers [103].

	2D	**3D**
Sparse Cholesky	O($n^{1.5}$)	O(n^{2})
CG, exact arithmetic	O(n^{2})	O(n^{2})
CG, no preconditioning	O($n^{1.5}$)	O($n^{1.33}$)
CG, modified	O($n^{1.25}$)	O($n^{1.17}$)
CG, support trees	O($n^{1.2}$)	O($n^{1.75}$)
Multigrid	O(n)	O(n)

4.7.2. Indirect Sparse Matrix Solvers

Iterative methods can also be employed for the solution of matrix equations. Among various iterative methods, the conjugate gradient method receives more attention because, in principle, it yields an exact solution (of the matrix equation) when the number of iterations reaches the number of equations or unknowns. For this reason, conjugate gradient method is also referred to as a semi-direct method.

The conjugate gradient method was originally developed by Hestenes and Stiefel [104].

The biconjugate gradient method was developed by Lanczos [105], and it can also be applied to general (both symmetric and non-symmetric) matrix equations.

In addition to the conjugate gradient (CG) and biconjugate gradient (BCG) methods, there are several other closely related iterative methods. These include the generalized minimal residual (GMRES) [106], quasi-minimal residual (QMR) [107], conjugate gradient squared (CGS) [108], biconjugate gradient stabilized (BCGSTAB) [109], and transpose-free quasi-minimal residual (TFQMR) [110] methods. The algorithms implementing these methods can be found in public literature and software packages [111]. Their main features can be summarized as follows:

GMRES computes a sequence of orthogonal vectors that minimizes the residual norm in a least squares manner. Hence, the method leads to the smallest residual for a fixed number of iterations: However, it requires storing the entire sequence so that an increasingly large amount of storage is needed as the number of iterations increases. This difficulty is alleviated by restarting the algorithm after a certain number of iterations. The method is useful for general non-symmetric matrices.

QMR applies least squares to minimize a quantity that is closely related to the BCG residuals, thereby smoothing out the irregular convergence of the BCG, which may lead to more reliable approximations. It has a look-ahead strategy, which avoids BCG breakdown. In fact, even without this look-ahead strategy QMR largely avoids the breakdown. On the other hand, while it converges smoothly, it often does not improve on the BCG algorithm in terms of the iteration number.

CGS is the transpose-free variant of BCG. Although it converges faster than BCG, it exhibits more irregular convergence behavior with wilder oscillations in residual norm than does BCG; and sometimes does not guarantee convergence.

BCGSTAB uses local steepest descents to obtain more smooth convergence. While this method seems to work well in many cases, it still exhibits the irregular convergence behavior in some difficult problems. Also, its convergence is considerably slower than that of CGS.

TFQMR can be easily implemented by changing only a few lines in the standard CGS algorithm. However, unlike CGS, the iterations of TFQMR are characterized by quasi-minimization of the residual norm. This leads to smooth convergence with a convergence rate similar to CGS. Therefore, it can be considered as the new version of the CGS, which quasi-minimizes the residual in the space spanned by the vectors generated by the CGS iterations.

CGS, BCGSTAB, TFQMR share the feature that their implementations do not require any matrix transpose.

4.7.2.1. Biconjugate Gradient Method

A group of iterative solution algorithms, known under the name conjugate gradient methods, provide a quite general means for solving the $N \times N$ linear system

$$\mathbf{A}\cdot\mathbf{x}=\mathbf{b} \tag{4.44}$$

The attractiveness of these methods for large sparse systems is that they reference **A** only through its multiplication of a vector, or the multiplication of its transpose and a vector. These operations are very efficient for the row indexed sparse storage scheme. The simplest ordinary conjugate gradient algorithm [112-114] is effective only in the case that **A** is symmetric and positive definite. It is based on the idea of minimizing the function

$$f(\mathbf{x})=\frac{1}{2}\mathbf{x}'\cdot\mathbf{A}\cdot\mathbf{x}-\mathbf{b}\cdot\mathbf{x} \tag{4.45}$$

The function is minimized when its gradient

$$\nabla f=\mathbf{A}\cdot\mathbf{x}-\mathbf{b} \tag{4.46}$$

is zero, which is equivalent to Equation (4.44). The minimization is carried out by generating a succession of search directions $\mathbf{p}_k$, and improved minimizers $\mathbf{x}_k$. At each stage a parameter α_k is found that minimizes $f(\mathbf{x}_k+\alpha_k\mathbf{p}_k)$, and $\mathbf{x}_{k+1}$ is set equal to the new point $\mathbf{x}_k+\alpha_k\mathbf{p}_k$. The $\mathbf{p}_k$ and $\mathbf{x}_k$ are built up in such a way that $\mathbf{x}_{k+1}$ is also the minimizer of f over the vector space spanned by the directions already evaluated, namely $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k\}$. After N iterations, the minimizer over the entire space is arrived, which is the solution of Equation (4.46).

A generalization of the conjugate gradient method, where the matrix is not necessarily symmetric or positive definite, is the biconjugate gradient method. The method does not, in general, have a simple connection with function minimization. It constructs four set of vectors $\mathbf{r}_k$, $\mathbf{r}'_k$, $\mathbf{p}_k$, $\mathbf{p}'_k$, $k = 1, 2, \ldots$. The initial vectors $\mathbf{r}_1$ and $\mathbf{r}'_1$ are supplied and the equations $\mathbf{p}_1=\mathbf{r}_1$ and $\mathbf{p}'_1=\mathbf{r}'_1$ are set. The following recurrent relations are carried out:

$$\begin{aligned}
&\alpha_k = \frac{\mathbf{r}'_k \cdot \mathbf{r}_k}{\mathbf{p}'_k \cdot \mathbf{A} \cdot \mathbf{p}_k} \\
&\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A} \cdot \mathbf{p}_k \\
&\mathbf{r}'_{k+1} = \mathbf{r}'_k - \alpha_k \mathbf{A}^T \cdot \mathbf{p}_k \\
&\beta_k = \frac{\mathbf{r}'_{k+1} \cdot \mathbf{r}_{k+1}}{\mathbf{r}'_k \cdot \mathbf{r}_k} \\
&\mathbf{p}_{k+1} = \mathbf{r}_k + \beta_k \mathbf{p}_k \\
&\mathbf{p}'_{k+1} = \mathbf{r}'_k + \beta_k \mathbf{p}'_k
\end{aligned} \tag{4.47}$$

This sequence of vectors satisfies the biorthogonality condition

$$\mathbf{r}'_i \cdot \mathbf{r}_j = \mathbf{r}_i \cdot \mathbf{r}'_j = 0 \qquad j < i \tag{4.48}$$

And the biconjugacy condition

$$\mathbf{p}'_i \cdot \mathbf{A} \cdot \mathbf{p}_j = \mathbf{p}_i \cdot \mathbf{A}^T \cdot \mathbf{p}'_j = 0 \quad j < i \tag{4.49}$$

There is also a mutual orthogonality

$$\mathbf{r}'_i \cdot \mathbf{p}_j = \mathbf{r}_i \cdot \mathbf{p}'_j = 0 \qquad j < i \tag{4.50}$$

The proof of these properties proceeds by induction [115]. As long as the recurrence does not break down earlier because one of the denominators is zero, it must terminate at $m \leq N$ steps with $\mathbf{r}_{m+1} = \mathbf{r}'_{m+1} = 0$. This is basically because after at most N steps, new orthogonal directions to the vectors are run out.

To use the algorithm, an initial guess $\mathbf{x}_1$ is made for the solution. The residual $\mathbf{r}_1$ is chosen

$$\mathbf{r}_1 = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_1 \tag{4.51}$$

and $\mathbf{r}'_1 = \mathbf{r}_1$ is taken. Then the following sequence of improved estimates are formed

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{4.52}$$

while the recurrence in Equation (4.47) is carried out. Equation (4.52) guarantees that $\mathbf{r}_{k+1}$ from the recurrence is in fact the residual $\mathbf{b} - \mathbf{A} \cdot \mathbf{x}_{k+1}$ corresponding to $\mathbf{x}_{k+1}$. Since $\mathbf{r}_{m+1} = 0$, $\mathbf{x}_{m+1}$ is the solution to Equation (4.44). While there is no guarantee that this whole procedure will not break down, or will not become unstable for general **A**, this is rare in practice. More importantly, the exact termination in at most N iterations occurs only with exact arithmetic. Round-off error means that the procedure is to be halted when some appropriate error criterion is met.

4.7.3. Direct Sparse Matrix Solvers

Direct solvers can simply be summarized as methods depending on Gaussian elimination rather than iterative solution of the matrix equations.

One of the most popular direct solver families, frontal methods have their origins in the solution of finite element problems of structural analysis. One of the earliest implementations was that of Irons [116]. In this work, only the symmetric positive definite systems were considered. Later, the method was extended to unsymmetric systems [117], and also problems other than finite elements [118].

4.7.3.1. Multifrontal Method

The direct solution of the sparse matrix can be obtained by the Multifrontal Method, which belongs to the class of frontal methods and is defined by Liu [119]. The main advantages of this scheme are as follows:

1. Most arithmetic operations are performed on dense matrices, which reduces indexing efforts while addressing the entries,
2. In each front, parallelism can be achieved.

The algorithm works over the elimination tree, where its notion is initially defined by Duff [120]. For the multifrontal algorithm, necessary steps can be summarized as follows: For each node of T(**A**) from leaves to root:

- Sum own row/col of **A** with children's *Update* ($\mathbf{U}_i$) matrices into *Frontal* ($\mathbf{F}_i$) matrix
- Eliminate current variable from *Frontal* matrix, to get *Update* matrix
- Pass *Update* matrix to parent

The algorithm is best understood by an illustration. Fig. 4.12 illustrates a simple example. Several observations can be made over this method:

- All arithmetic happens on dense square matrices
- Needs extra memory for a stack of pending update matrices
- There exists potential parallelism:
    1. Between independent tree branches
    2. Parallel dense operations on the frontal matrix

The performance of algorithm depends on the structure of the elimination tree (i.e. the positioning of the nonzero entries) of the matrix. Guermouche *et al* had several publications [121-122] on the memory usage of the method.

As seen in Fig. 4.13, the total memory required by the algorithm is not limited to matrix storage as in the case of iterative solvers. The total required memory can be considered as the factor storage area (factor memory); and also the current frontal matrix storage area and the contribution storage area (active memory).

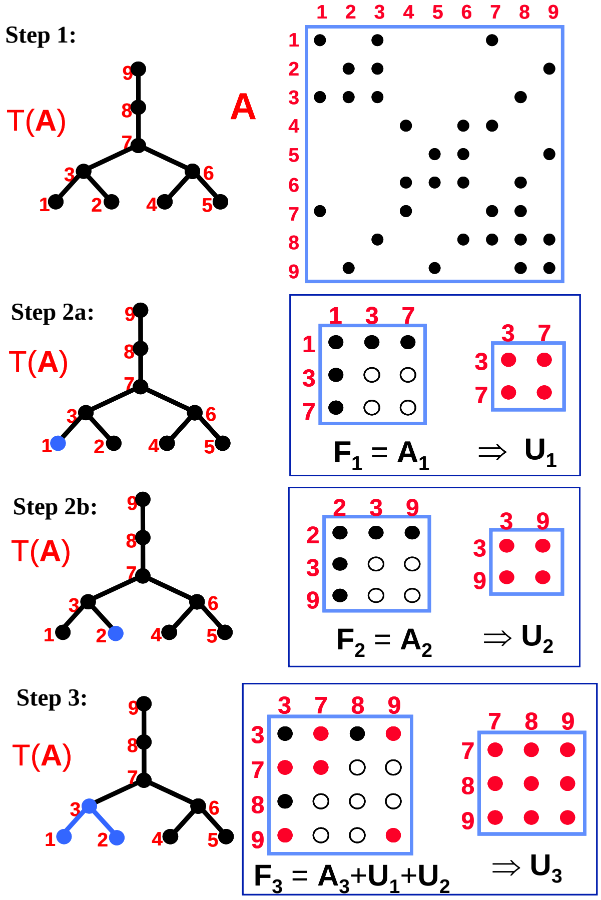


Fig. 4.12. Pictorial Description of Calculation of Frontal and Update (Contribution) Matrices for the Multifrontal Algorithm.

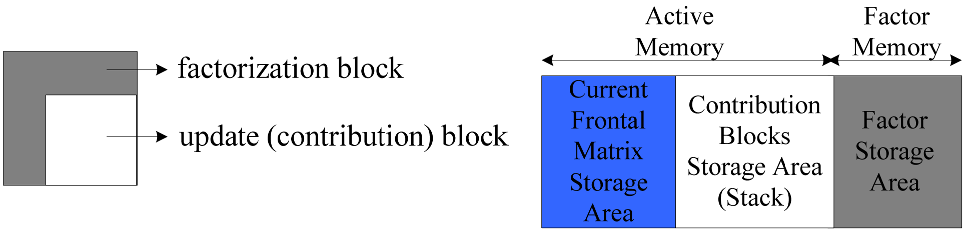

factorization block
update (contribution) block
Active Memory
Factor Memory
Current Frontal Matrix Storage Area
Contribution Blocks Storage Area (Stack)
Factor Storage Area

Fig. 4.13. Memory Usage of the Multifrontal Method.

Guermouche *et al* also deeply investigated the relationship between the structure of the elimination tree and active memory usage. As seen in Fig. 4.14, a wide elimination tree causes a higher peak in the active memory usage.

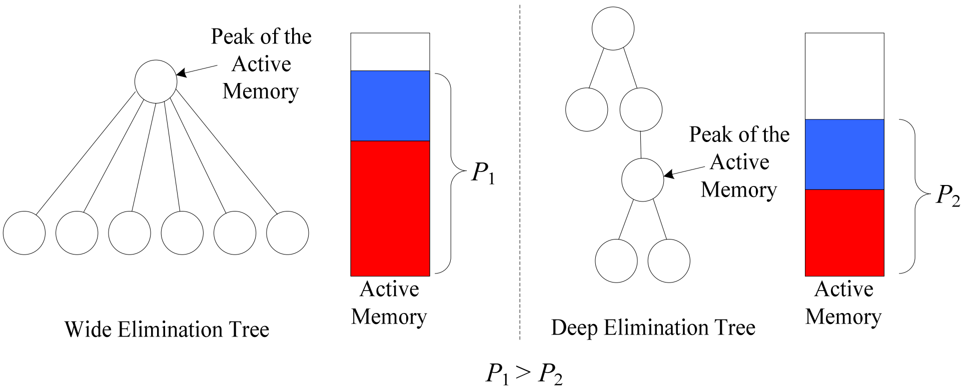


Fig. 4.14. Effect of the Elimination Tree Structure on Memory Usage [121].

Depending on the positioning of the non zero entries (simply called and known as matrix ordering), the elimination tree might have different structures as seen in Fig. 4.15. Several observations and comments can be made:

- Each branch of the elimination tree can be distributed to another processor. A very narrow tree might cause inefficient parallelism, since some of the processors might not be allocated. On the other hand, a very wide tree might cause that the number of processors become insufficient.
- A deep tree might reduce active memory usage, but increases CPU time.
- An unbalanced tree might cause inefficient parallelism; some processors can finish their tasks earlier and wait, but the dominant term is the CPU time of the busiest processor.

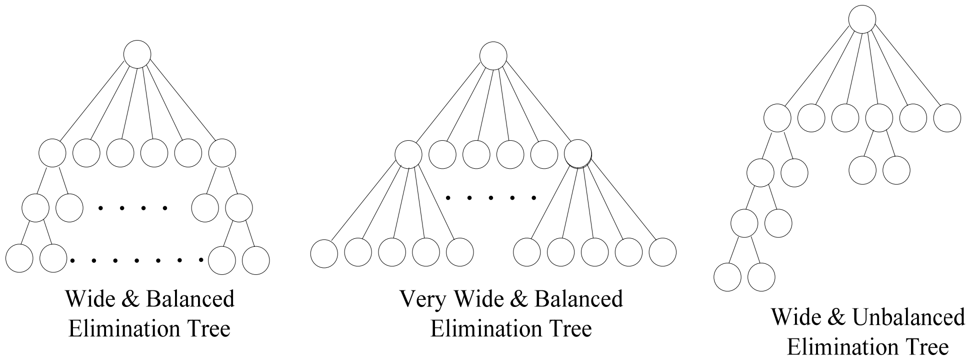


Fig. 4.15. Different Elimination Tree Structures.

There are several matrix reordering methods available in the literature (not only in published form but also inside compileable/linkable or executable software packages [123-127], which structures the elimination tree of a matrix according to the needs. A rough comparison of these methods and resultant elimination trees are given in Fig. 4.16. Of course, there is not any certain statement that "such reordering method comes first"; the one which is most appropriate for the needs should be selected.

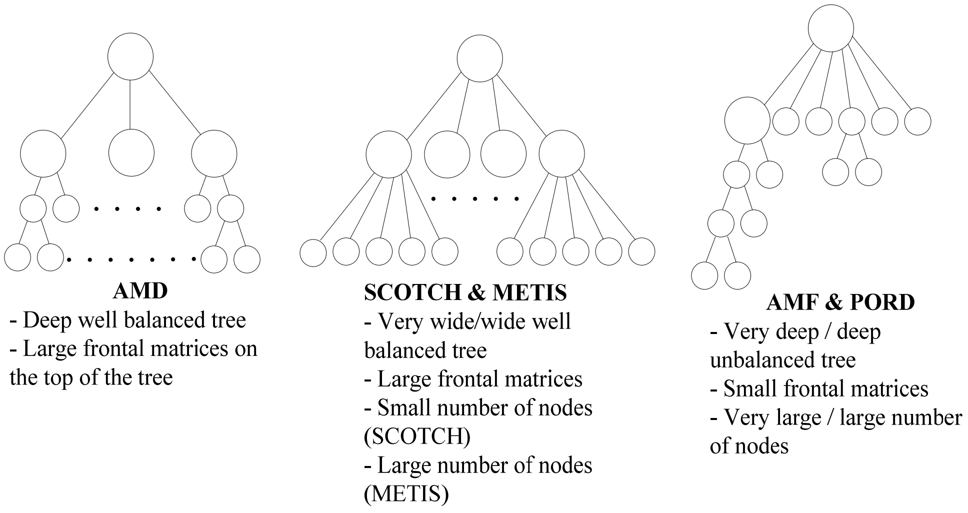

AMD
- Deep well balanced tree
- Large frontal matrices on the top of the tree
SCOTCH & METIS
- Very wide/wide well balanced tree
- Large frontal matrices
- Small number of nodes (SCOTCH)
- Large number of nodes (METIS)
AMF & PORD
- Very deep / deep unbalanced tree
- Small frontal matrices
- Very large / large number of nodes

Fig. 4.16. A Comparison of the Matrix Reordering Methods [121].

4.8. Effects of Node/Edge Ordering During Matrix Solution

One of the main steps of the finite element solution is the matrix assembly process, which is the construction of the global system matrix from the elemental matrices. For this purpose, during the mesh generation, while the elements are being constructed, a mapping relating the local edge and the element numbers to a global edge number should also be constructed. The assembly process assigns the contribution of the element matrices to the corresponding entries of the global matrix via this mapping.

For example, if the *i*th and *j*th local edges of an element *e* are mapped to *I*th and *J*th global edges, then the entry (*i*, *j*) of the *e*th elemental matrix should have a contribution to the entry (*I*, *J*) of the global system matrix.

Similarly, if an edge with a global number *I* is shared by 4 elements e_1, e_2, e_3, e_4 (which is the usual case except the edges on the boundaries), and if it has the local numbers i_1, i_2, i_3, i_4 in these elements respectively, then

$$G(I,I) = e_1(i_1,i_1) + e_2(i_2,i_2) + e_3(i_3,i_3) + e_4(i_4,i_4) \tag{4.53}$$

where *G* is the global system matrix.

The method of edge ordering determines the structure of the global stiffness matrix. For simplicity, consider the example of quadrilateral node elements. Naturally, in this example node ordering determines the structure of the global stiffness matrix. As seen in Fig. 4.17, the nodes same mesh can be numbered in different manners.

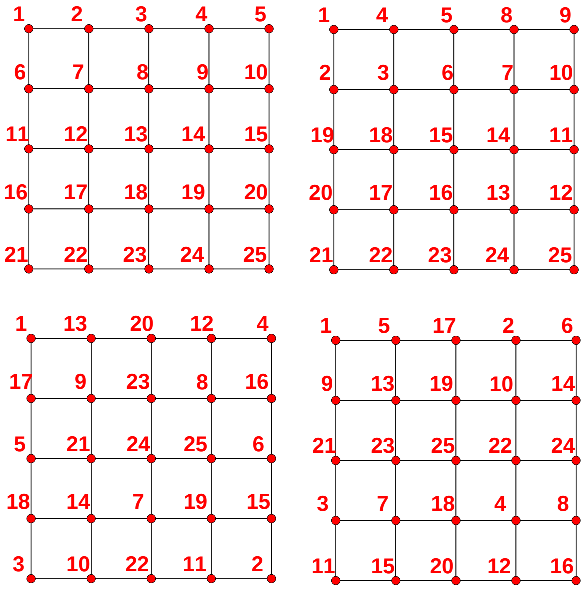


Fig. 4.17. Different numbering schemes for a fixed quadrilateral mesh.

If the nodes are numbered in an ordered manner, the resultant global stiffness matrix will be banded as seen in Fig. 4.18.

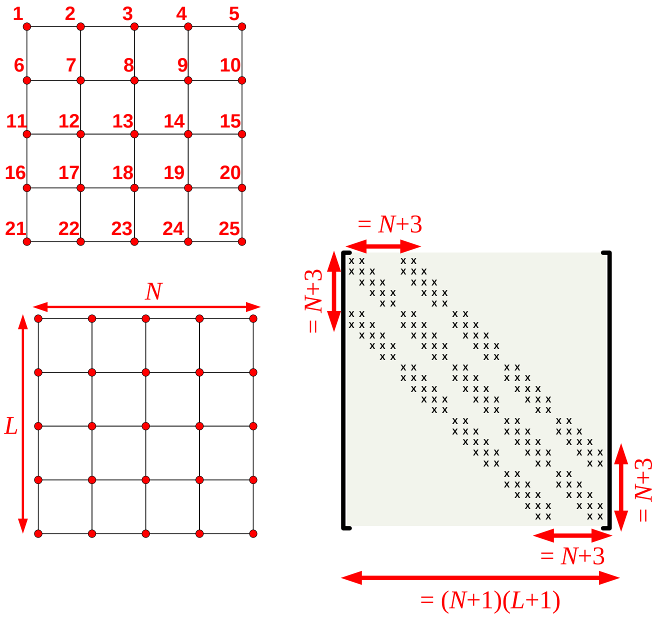


Fig. 4.18. Resultant Matrix for the Ordered Numbering Scheme.

If the nodes are numbered in a spiral manner, the resultant global stiffness matrix will not be banded, but still structured this time, as seen in Fig. 4.19.

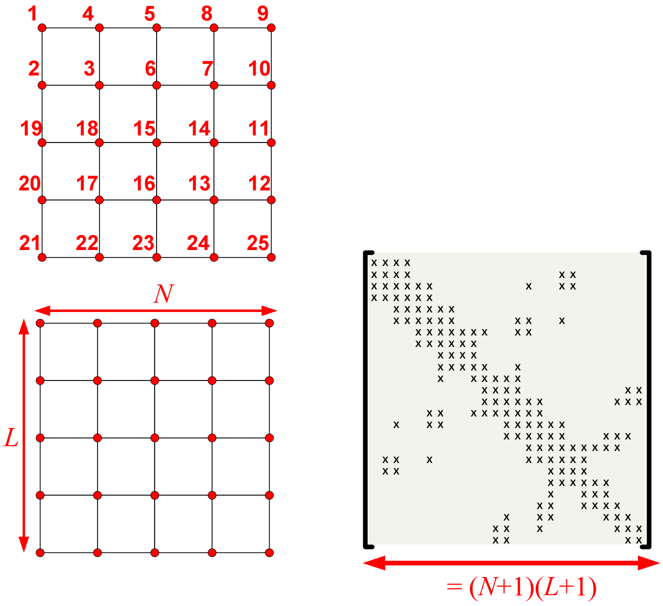


Fig. 4.19. Resultant Matrix for the Spiral Numbering Scheme.

If the nodes are numbered in an irrelevant (seems like random; but intentionally maximizing the global node number difference between adjacent nodes) manner, the resultant global stiffness matrix will be very unstructured this time, as seen in Fig. 4.20.

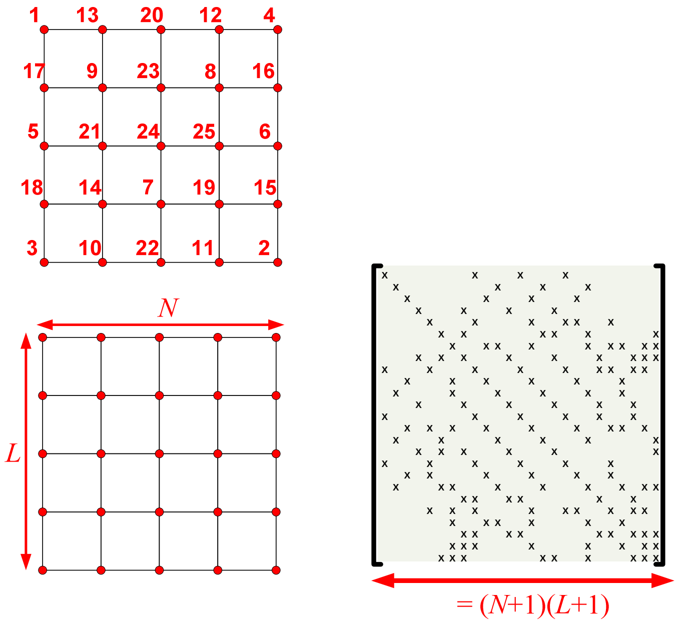


Fig. 4.20. Resultant Matrix for the Irrelevant Numbering Scheme.

The structure of the global stiffness matrix is very important especially in the multifrontal algorithm. Although all the matrices will have the same sparsity, the amount of fill-ins (i.e. the number of probable nonzero entries after the Gaussian elimination starts) differs. The fill-ins are illustrated for all matrices (their upper diagonal parts) in Fig. 4.21.

As seen from this figure, ordered and spiral numbering schemes yield less fill-ins compared to irrelevant numbering scheme. This exaggerated experiment shows that the global node number difference of adjacent nodes should be kept as minimum as possible in order to minimize the memory requirements if the multifrontal method will be used. When ordered and spiral numbering schemes are compared, it can be seen that the amount of fill-ins is nearly equal. However, the positions of the potential filled-in entries are more predictable for the ordered scheme, since the fill-ins reside inside the band.

The conclusion of this analysis can be summarized as: Ordered node/edge numbering scheme should be preferred in order to minimize the memory usage of the multifrontal algorithm.

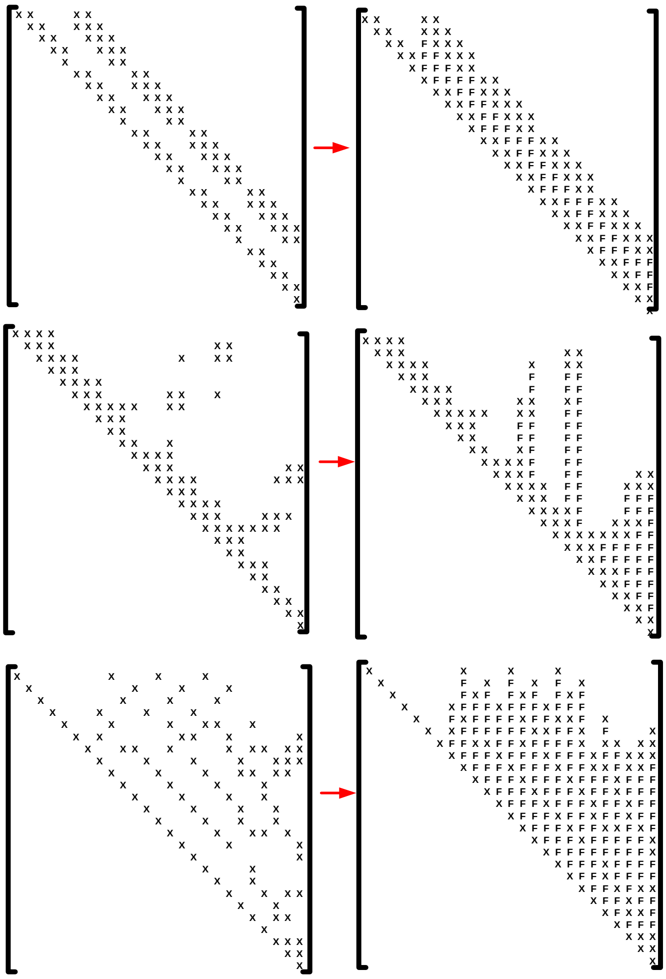


Fig. 4.21. Effect of Fill-in for the matrices obtained by different ordering schemes.

The situation is different if the conjugate gradient method (or any of its derivatives) is used. For this purpose, the procedure called "node coloring" might be followed. The nomenclature "coloring" comes from the duality of this problem to the famous "map coloring" problem of the graph theory. The procedure is a two-step algorithm:

i) First, the nodes are colored so that adjacent and related nodes will have different colors. The coloring is performed by assignment of integers (i.e. color codes) to each node.

ii) Second, starting from the smallest color code (0 in this example), the nodes of the same color are numbered in order. When the nodes in one color code finishes, the numbering operation continues after incrementing the color code.

The procedure is repeated for the same example as seen in Fig. 4.22. The main aim of the procedure is to maximize the distance of the nodes with successor numbers. This is similar to maximizing the distance of two countries with same colors in the map coloring problem.

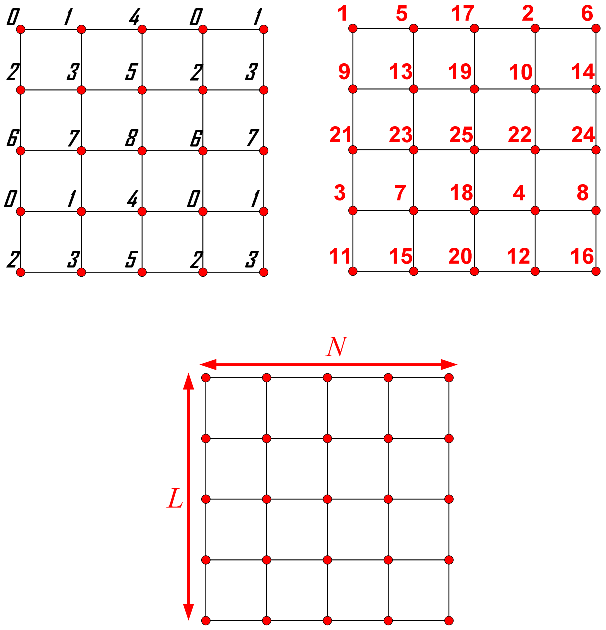


Fig. 4.22. “Node Coloring” Numbering Scheme.

The resultant matrix after the node coloring scheme is as seen in Fig. 4.23. Since this matrix can be subdivided to submatrices, which are mostly diagonal; parallelization can be achieved during the iterative solution of this matrix.

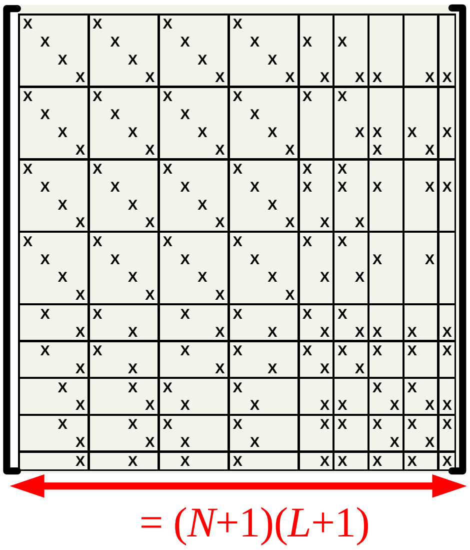


Fig. 4.23. Resultant Matrix After the Node Coloring Scheme.

After such an analysis, the lessons learnt can be applied to the numbering of edge elements. In Fig. 4.24, ordered edge numbering for linear quadrilateral edge elements is given. The band of the matrix can be calculated in such a numbering scheme.

Similarly, ordered edge numbering for quadratic quadrilateral edge elements can be performed as seen in Fig. 4.25.

The analysis can similarly be extended and effects of any numbering scheme can be investigated for hexahedral edge elements of any order.

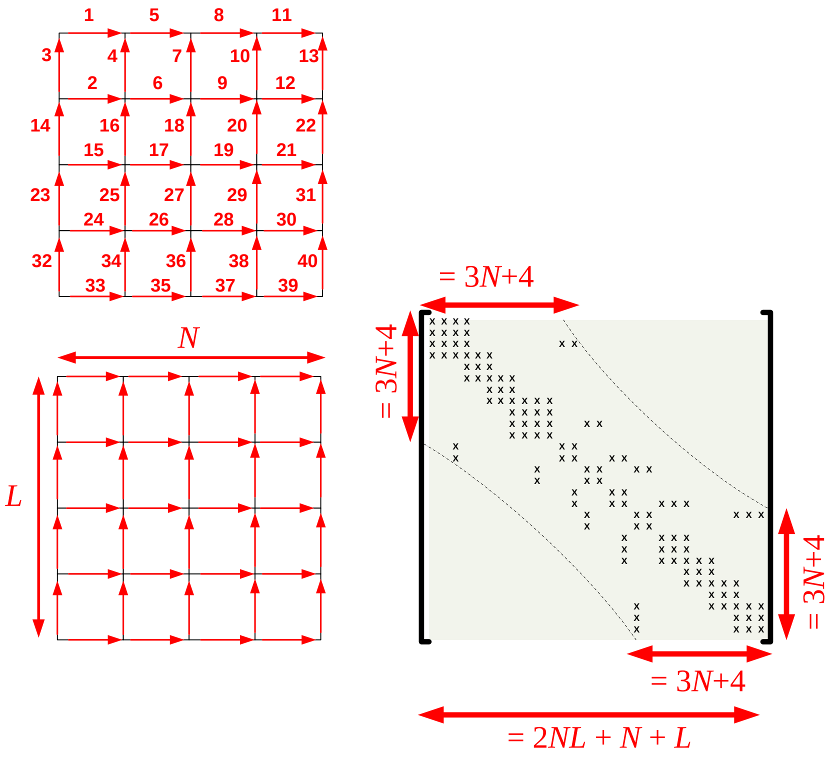


Fig. 4.24. An example for the Linear Quadrilateral Edge Element.

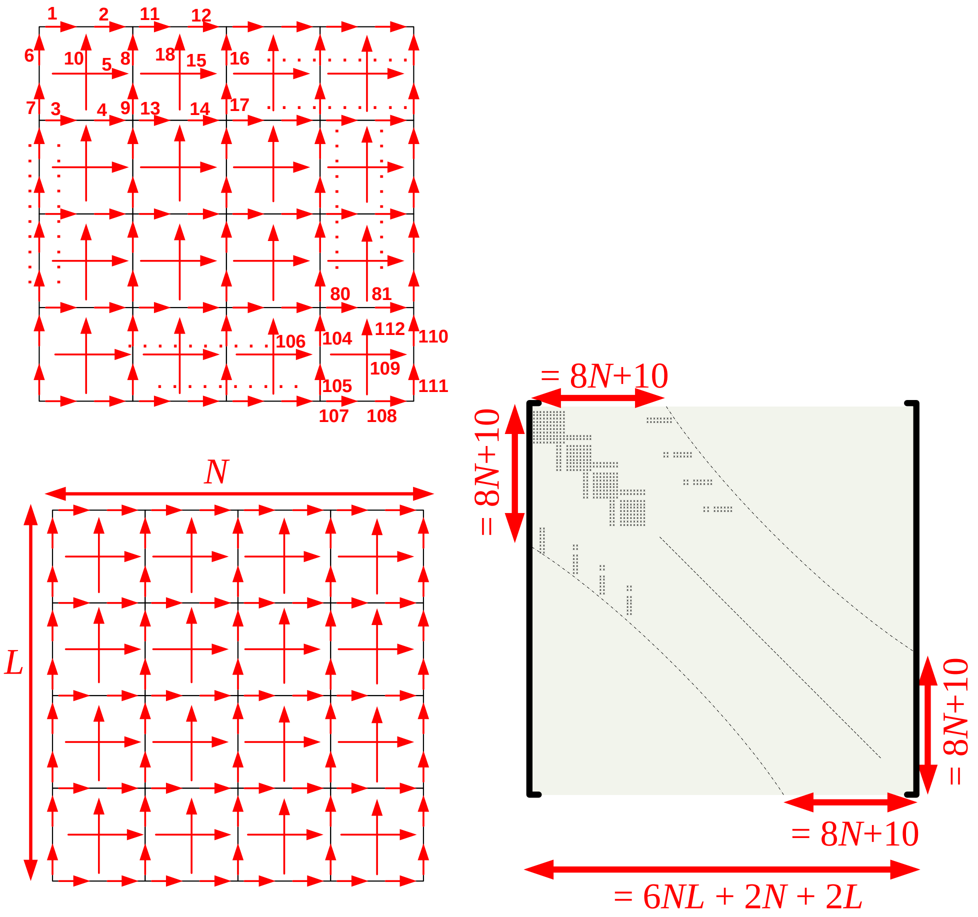


Fig. 4.25. Another example for the Quadratic Quadrilateral Edge Element.

4.9. Radar Cross-Section and Huygens' Equivalence Principle

One of the particular interests in scattering is the evaluation of the radar echo area (for the two-dimensional case) or scattering cross-section (for the three-dimensional case). The latter is given by

$$\sigma_{3D} = \lim_{r\to\infty} 4\pi r^2 \frac{\left|\mathbf{E}^{\mathrm{sct}}\right|^2}{\left|\mathbf{E}^{\mathrm{inc}}\right|^2} \tag{4.54}$$

where $\mathbf{E}^{\mathbf{sct}}$ is the far zone scattered field. In order to calculate the far zone scattered field, we utilize the surface equivalence principle, which reduces the calculation to the computation of a surface integral.

The surface equivalence principle states that the field exterior (or interior) to a given surface may be exactly represented by equivalent currents on that surface and allowed to radiate into the region external (or internal) to that surface. The appropriate currents representing the fields are given as

$$\begin{aligned} \mathbf{n} \times \mathbf{H} &= \mathbf{J} \\ \mathbf{E} \times \mathbf{n} &= \mathbf{M} \end{aligned} \tag{4.55}$$

The far field expression for the electric field due to these equivalent currents is approximately given by the expression

$$\mathbf{E}(\mathbf{r}) \approx jk_0 \frac{e^{-jk_0 r}}{4\pi r} \oiint_{S_m} \left[\mathbf{r} \times \mathbf{M}(\mathbf{r}') + Z_0 \mathbf{r} \times (\mathbf{r} \times \mathbf{J}(\mathbf{r}'))\right] \; e^{-jk_0(\mathbf{r}' \cdot \mathbf{r})} dS' \tag{4.56}$$

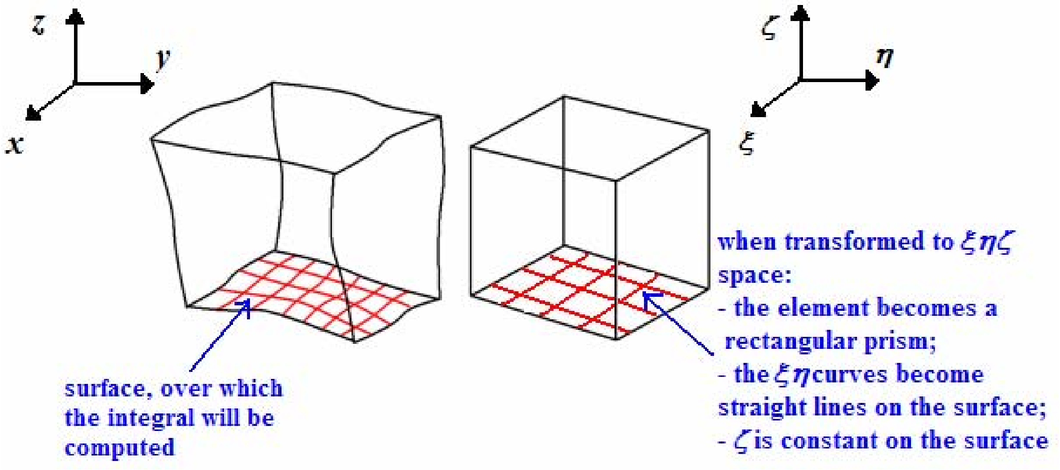
where $\mathbf{r}$ and $\mathbf{r}'$ denote the observation and source points respectively, and Z_0 is the free space impedance. The far zone scattered field $\mathbf{E}^{\mathbf{sct}}$ is calculated by Equation (4.56) and this value is substituted in Equation (4.54). An acceptable criterion for using Equation (4.56) conveniently is Rayleigh’s criterion stated as

$$r \geq \frac{2D^2}{\lambda_0} \tag{4.57}$$

where D is the largest dimension of the scattering material.

In practical applications the scattering cross-section, which is calculated by using Equation (4.54), is normalized with the square of the wavelength, and its characteristics is observed in logarithmic scale. This quantity is denoted by ‘RCS dBSW’.

The computation of the surface integral for conformal meshes with curved elements is not a straightforward task. The following formulation should be used for this purpose.



surface, over which the integral will be computed

when transformed to $\xi\eta\zeta$ space:
- the element becomes a rectangular prism;
- the $\xi\eta$ curves become straight lines on the surface;
- ζ is constant on the surface

Fig. 4.26. Pictorial Description of the Surface Integration Method.

As seen in Fig. 4.26., by using the isoparametric hexahedral elements (i.e. assuming that each hexahedral element is transformed to a cube in $\xi\eta\zeta$-space extending from (-1,-1,-1) to (1,1,1)); for any function $G'(x,y,z)$, the surface integral on the surface of an element

$$\iint_{S_e} G^{'}(x,y,z)ds \tag{4.58}$$

in the *xyz*-space can be stated as

$$\int_{-1}^{1}\int_{-1}^{1} G(\xi,\eta,\zeta)\left|\frac{\partial(x,y,z)}{\partial(\xi\eta)}\right| d\xi d\eta\,, \qquad \zeta\,\text{constant} \tag{4.59}$$

in the $\xi\eta\zeta$-space. In Equation (4.60),

$$\left|\frac{\partial(x,y,z)}{\partial(\xi\eta)}\right| = \left[\left(\frac{\partial x}{\partial\xi}\frac{\partial y}{\partial\eta} - \frac{\partial x}{\partial\eta}\frac{\partial y}{\partial\xi}\right)^2 + \left(\frac{\partial z}{\partial\xi}\frac{\partial x}{\partial\eta} - \frac{\partial z}{\partial\eta}\frac{\partial x}{\partial\xi}\right)^2 + \left(\frac{\partial y}{\partial\xi}\frac{\partial z}{\partial\eta} - \frac{\partial z}{\partial\eta}\frac{\partial y}{\partial\xi}\right)^2\right]^{1/2} \tag{4.60}$$

Or in other words,

$$ds = \left|\frac{\partial(x,y,z)}{\partial(\xi\eta)}\right| d\xi d\eta, \qquad \text{where } \zeta \text{ is constant} \tag{4.61}$$

CHAPTER 5

NUMERICAL RESULTS

5.1. Results for Homogenous Scatterers

Throughout this section, results obtained for homogenous scatterers are given. Comparisons and comments on the solutions are given; and the solution accuracy is discussed. Unless otherwise stated, in all examples the incident field is chosen as $e^{-jkz}\mathbf{a_y}$. The results are investigated for different θ values in the $\phi = 0$ and $\phi = \pi / 2$ planes.

5.1.1. Results for Uncurved Homogenous Scatterers

5.1.1.1. Permeable Cube

As a first example, RCS of a permeable cube with a sidelength of 0.5λ, and a relative permeability of $\mu_r = 2.2$ is considered. The results obtained for 125 quadratic hexahedral edge elements and 2,080 unknowns (indicated with triangles), are compared to those of Sertel [128] (indicated with line) in Fig. 5.1. The same accuracy level was achieved about 1,000 linear hexahedral edge elements.

For this problem, in [128] Sertel used the Multilevel Fast Multipole Method (MLFMM) together with a Volume Integral Equation (VIE) formulation. He also used Finite Element Boundary Integral (FE-BI) Method for comparison. Unfortunately, he has not explicitly stated the details (number of elements, number of unknowns, etc.) of the FE-BI solution, although he clearly stated that he used hexahedral elements.

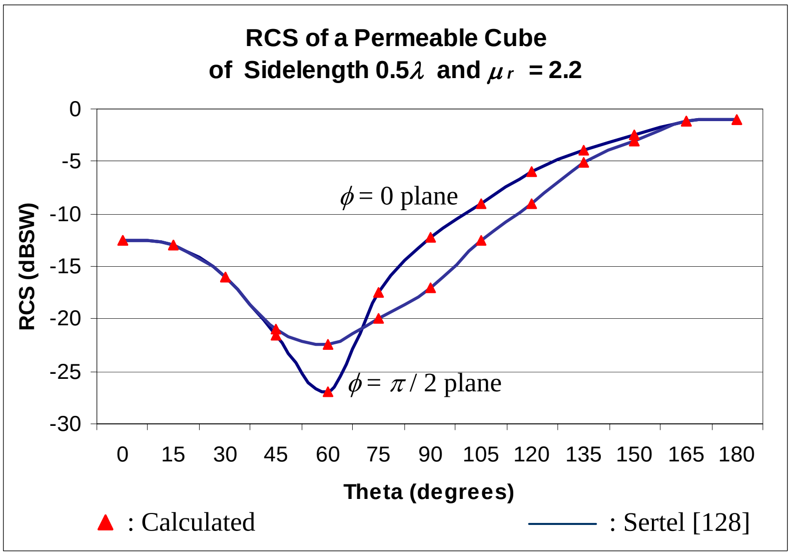


Fig. 5.1. RCS of a Permeable Cube Calculated on Compared to [128].

5.1.1.2. Composite Cube

As a second example, RCS of a composite cube with a sidelength of 0.2λ, a relative permeability of μ_r = 2.2, and a relative permittivity of ε_r = 2.2 is considered. Again, the results obtained for 125 quadratic hexahedral edge elements and 2,080 unknowns (indicated with triangles), are compared to those of Sertel [128] (indicated with line) in Fig. 5.2. Similar to the permeable cube case, the same accuracy level were achieved about 1,000 linear hexahedral edge elements.

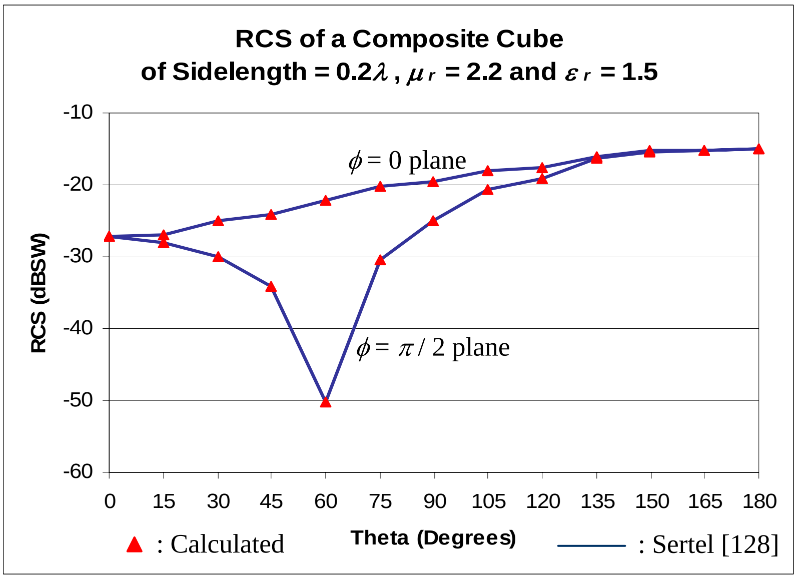


Fig. 5.2. RCS of a Composite Cube Calculated and Compared to [128].

As in the permeable cube problem, in [128] Sertel also used the Multilevel Fast Multipole Method (MLFMM) together with a Volume Integral Equation (VIE) formulation. He also used Finite Element Boundary Integral (FE-BI) Method for his comparisons. Unfortunately, again he has not explicitly stated the details (number of elements, number of unknowns, etc.) of the FE-BI solution, although he clearly stated that he used hexahedral elements.

5.1.2. Results for Curved Homogenous Scatterers

5.1.2.1. PEC Sphere

In this problem, the scatterer is chosen to be a PEC sphere with a radius of 0.5λ. The mesh generated for this problem corresponds to a mesh of a spherical shell, since the volume (and the elements) inside the PEC sphere is thrown out. The domain decomposition and the details of the mesh generation are given in Appendix C. Meshes with different levels of density are given in Fig. 5.3.

As expected, varying the parameters of meshing (mentioned in Appendix C) is a driving factor in the element and edge sizes as well as the number of elements and edges (unknowns). In order to perform a comparison, each meshing scheme is given a name (pecsph00x). In summary, the parameters and total number of elements achieved in each scheme is given in Table 5.1.

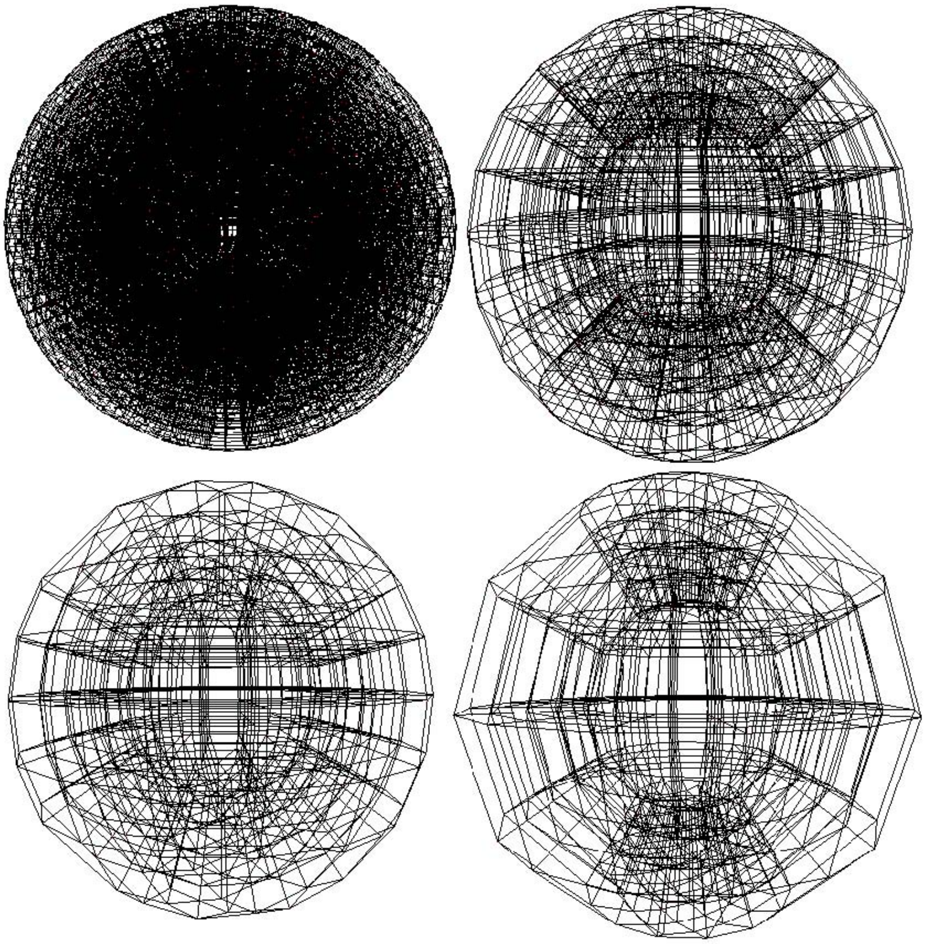


Fig. 5.3. Meshes of Different Levels of Density.

Table 5.1. Comparison of Each Meshing Scheme (Meshing Parameters and Resource Requirements).

	θ_c (degrees)	θ_d (degrees)	Mesh Resolution in R direction (unitless)	Mesh Resolution in θ direction (degrees)	Mesh Resolution in ϕ direction (degrees)	Number of Elements
pecsph001	45	40	1	5	5	20,880
pecsph002	45	40	2	9	9	3,400
pecsph003	45	40	2.5	15	12.85	1,288
pecsph004	30	25	2.5	30	22.5	512

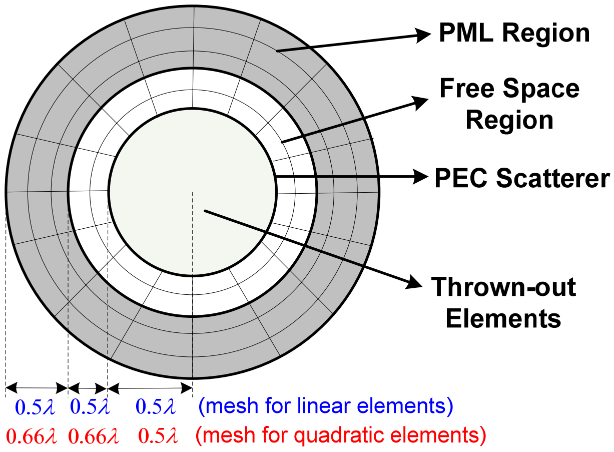

PML Region
Free Space Region
PEC Scatterer
Thrown-out Elements
0.5λ 0.5λ 0.5λ (mesh for linear elements)
0.66λ 0.66λ 0.5λ (mesh for quadratic elements)

Fig. 5.4. Cross section of the mesh generated for the PEC sphere problem.

The cross-section of the mesh generated for this problem is given in Fig. 5.4. For this problem, a percentage error measure $err(\mathbf{E})$ for the magnitude of the scattered field is defined as

$$err(\mathbf{E}) = \frac{1}{K}\sum_{i=1}^{K}\frac{\left\|\mathbf{E}_{exact}(P_i)-\mathbf{E}_{comp}(P_i)\right\|}{\left\|\mathbf{E}_{exact}(P_i)\right\|} \tag{5.1}$$

where $\mathbf{E}_{exact}(P_i)$ is the exact electric field calculated via the Mie series at the centroid (P_i) of an element lying in free space; whereas $\mathbf{E}_{comp}(P_i)$ is the value calculated by FEM at the same point. Certainly, the summation traces all elements lying in free space; K is the number of such elements; and $err(\mathbf{E})$ is therefore the mean normalized error over the free space portion of the computational domain.

A summary of the resources required for the execution of different numerical experiments is given in Table 5.2. It is clear that the usage of quadratic hexahedral elements yields better accuracy compared to the usage of linear hexahedral elements.

Table 5.2. Comparison (resource requirement, element size, solution accuracy, etc.) of linear and quadratic elements for PEC sphere problem.

Element Type	**Element Size (λ)**					**Number of Elements**	**Number of Unknowns**	$err(\mathbf{E})$
	along R	**along θ**		**along ϕ**				
	fixed	**min**	**max**	**min**	**max**			
Linear	0.1	0.092	0.270	0.137	0.404	7,040	23,380	0.0531
Quadratic	0.33	0.262	0.785	0.393	1.178	640	9,870	0.0397

5.1.2.2. Dielectric Sphere

Next, the RCS of a dielectric sphere of radius 0.5λ is considered. This time, the mesh to be generated is a full sphere, not a spherical shell. This is because of the fact that the volume of the scatterer should also be considered. The details of the domain decomposition and mesh generation are given in Appendix C.

The RCS of the dielectric sphere is calculated and compared to Mie series as seen in Fig. 5.5. This accuracy level was achieved by 804 quadratic or 8,640 linear elements (corresponding to 8,230 and 26,840 unknowns respectively).

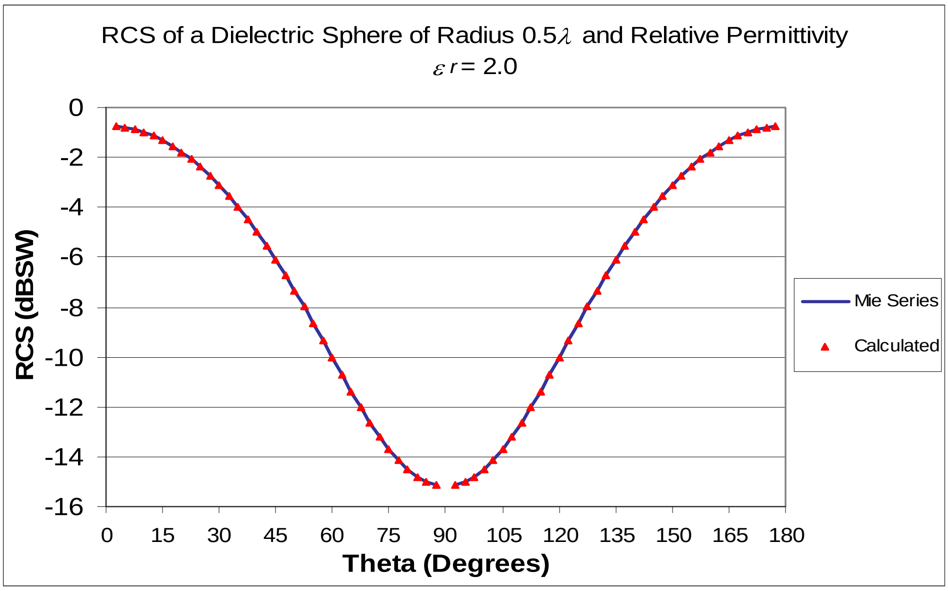


Fig. 5.5. RCS of a Dielectric Sphere.

5.1.2.3. PEC Spheroid

The next solved problem of this class is RCS of the oblate/prolate PEC spheroids. The cross-section of the mesh generated for a PEC sphere problem is given in Fig. 5.6.

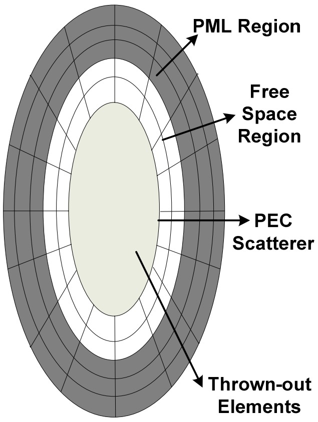


Fig. 5.6. Cross-Section of the Mesh Generated for a PEC Prolate Spheroid.

As seen in Fig. 5.6, the problem is similar to PEC sphere problem: The elements inside the scatterer should be thrown out, and there exists a transformation from the PEC sphere mesh to PEC spheroid mesh (i.e. two meshes are homeomorphic), where the tranformation is pictorially described in Fig. 5.7.

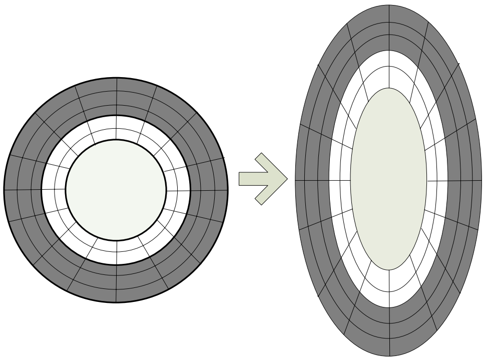


Fig. 5.7. Transformation from the PEC sphere mesh to the PEC spheroid mesh.

By using this method, all-hexahedral meshes for oblate and prolate spheroids have been generated as seen in Fig. 5.8.

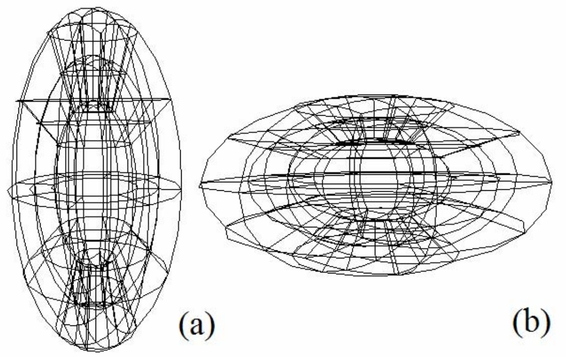


Fig. 5.8. Sample All-Hexahedral Meshes Generated for

(a) Prolate, (b) Oblate PEC Spheroids.

To the author's knowledge, unfortunately bistatic RCS results for PEC spheroids do not exist in the literature. Only, backscatter RCS values for oblate PEC spheroids are given in [129]. Hence, the backscatter RCS problem is solved by using locally conformal PML formulation [101] and both van Welij's and Kameari's elements; and the obtained results are compared as seen in Table 5.3. Throughout this analysis, the major axes of the spheroid are kept fixed as 2λ; and the minor axis length is varied between 0.2λ and 0.9λ.

Table 5.3. Backscatter RCS of an Oblate Spheroid (Maxor Axes Fixed at 2λ).

Sph. Minor Axis Length (λ)	**Element Type**	**Element Size in Radial Dir. (λ)**	**Number of Elements**	**Back-scatter RCS (dB) [129]**	**Calc. Back-scatter RCS (dB)**	**Error (%)**
0.2	Linear	0.1	7,040	43	44.0	2.33
	Quadratic	0.4	640		43.3	0.70
0.3	Linear	0.1	10,560	37	37.6	1.62
	Quadratic	0.4	960		37.5	1.35
0.4	Linear	0.1	14,080	31	32	3.23
	Quadratic	0.4	1,280		31.8	2.58
0.5	Linear	0.1	17,600	24	25.0	4.17
	Quadratic	0.4	1,600		24.9	3.75
0.6	Linear	0.1	21,120	20	21.1	5.50
	Quadratic	0.4	1,920		20.9	4.50
0.7	Linear	0.1	24,640	19	20.8	9.47
	Quadratic	0.4	2,240		20.4	7.37
0.8	Linear	0.1	28,160	18	20.4	13.33
	Quadratic	0.4	2,560		19.3	7.22
0.9	Linear	0.1	31,680	18	20.2	12.22
	Quadratic	0.4	2,880		19.1	6.11

As seen from Table 5.3, 0.4λ-sized quadratic elements give accuracy comparable with 0.1λ-sized linear elements. In [129], the solution is performed also by means of FEM but with spheroid PML formulation; but the authors have not specified the element type or the number of elements/unknowns. It is not possible to compare the method in this work with theirs in terms of resource usage (CPU time, memory, etc.). Nevertheless, the solutions seem to be matching with each other.

Backscatter RCS values for prolate PEC spheroids for comparison could not be found in the literature.

5.1.2.4. Dielectric Cylinder

As another example, the dielectric cylinder ($\varepsilon_r = 4$, d=4λ, h=4λ) as shown in Fig. 5.9 is considered. The details of the domain decomposition and mesh generation are given in Appendix C. In order to get benefit of the cylindrical coordinates during the mesh generation, the problem is transferred to a $x'y'z'$-space, in which the main axis of the cylinder will be along the z' axis. After the mesh is generated, the following back transformation is carried out: $x' \to z$; $y' \to x$; $z' \to y$

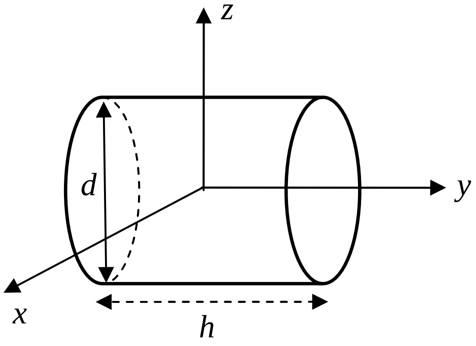


Fig. 5.9. Description of the Dielectric Cylinder Problem.

Since the scatterer in this problem is electrically large, only solution with quadratic elements is considered. Solution with 5,860 elements and 86,400 unknowns is illustrated in Fig. 5.10 and compared to [130]. In [130], the solution was found by using the Method of Moments and the so-called Precorrected Fast Fourier Transform. The authors have not explicitly stated the resource requirements (CPU time, memory) of the set-up they have used for the solution of this problem.

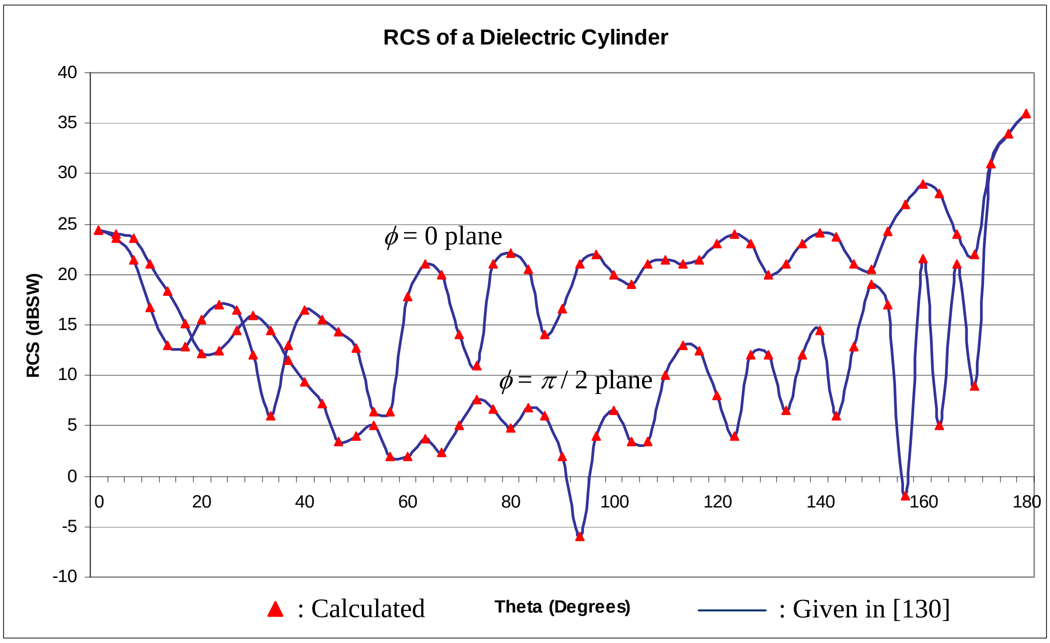


Fig. 5.10. RCS of a Dielectric Cylinder Calculated and Compared to [130].

5.2. Results for Scatterers with Patches

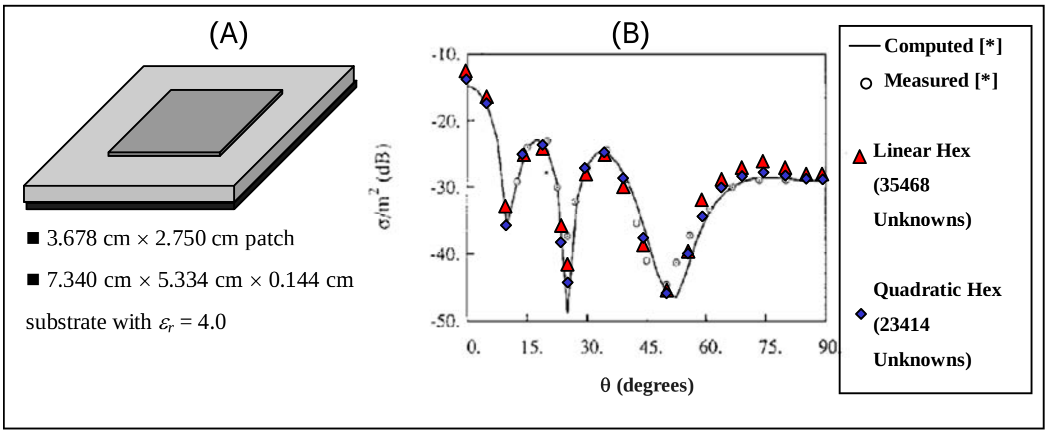
In this section, RCS (either bistatic or backscatter) of various scatterers with patches are investigated.

5.2.1. Results for Uncurved Scatterers with Patches

Among the scatterers with patches, first the ones with uncurved sides and faces are considered. RCS of microstrip patch antennas was a popular research area in the early 90s since they are dominant on the RCS of the platforms that they are mounted on. Volakis *et al* had a series of papers about the RCS of the rectangular patch microstrip patch antennas [131-133]. In these papers, the authors used the Finite Element – Boundary Integral (FE-BI) Method. They have not specified the element type, and the number of unknowns; hence it is not possible to compare the results in terms of resource requirements (CPU time, memory, etc.).

5.2.1.1. Unloaded Rectangular Microstrip Patch

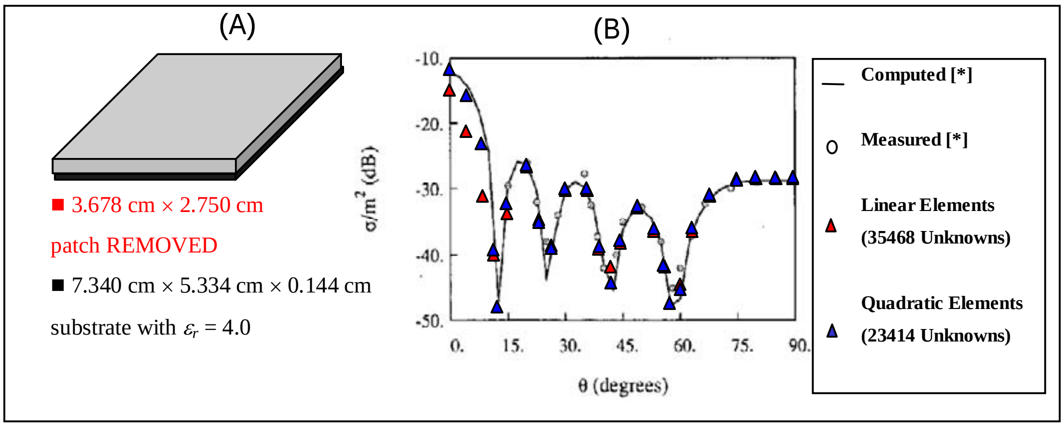
The first problem in this category is the RCS of a rectangular patch. The geometry and the results related to this problem are given in Fig. 5.11; comparison are performed against [133]. In the first part, the RCS is observed when the patch is appearing.



[*]: Jin and Volakis

Fig. 5.11. RCS of an Unloaded Rectangular Microstrip Patch (Compared to [133]).

In the second part, the patch is removed and the RCS is observed. The results are given in Fig. 5.12.

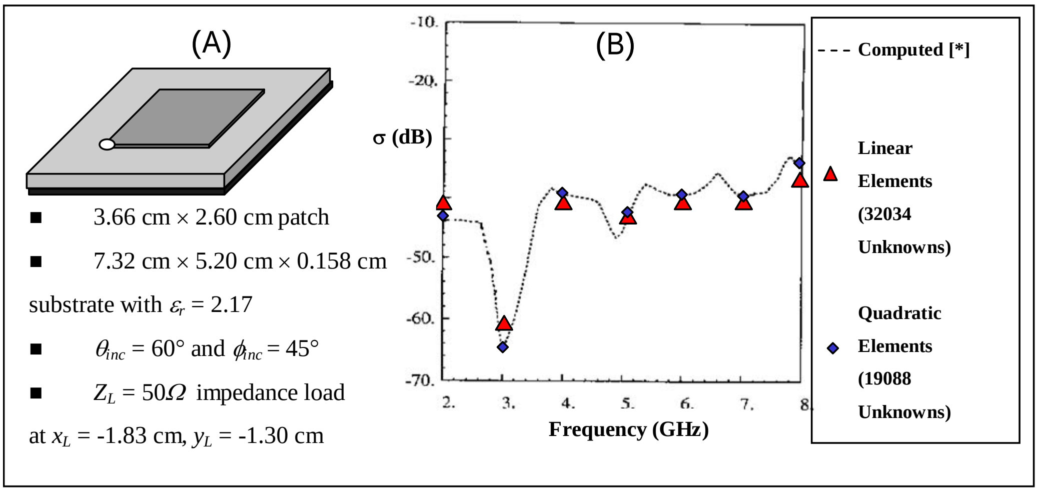


[*]: Jin and Volakis

Fig. 5.12. RCS of the Same Geometry When Patch Removed (Compared to [133]).

5.2.1.2. Singly-Loaded Rectangular Microstrip Patch

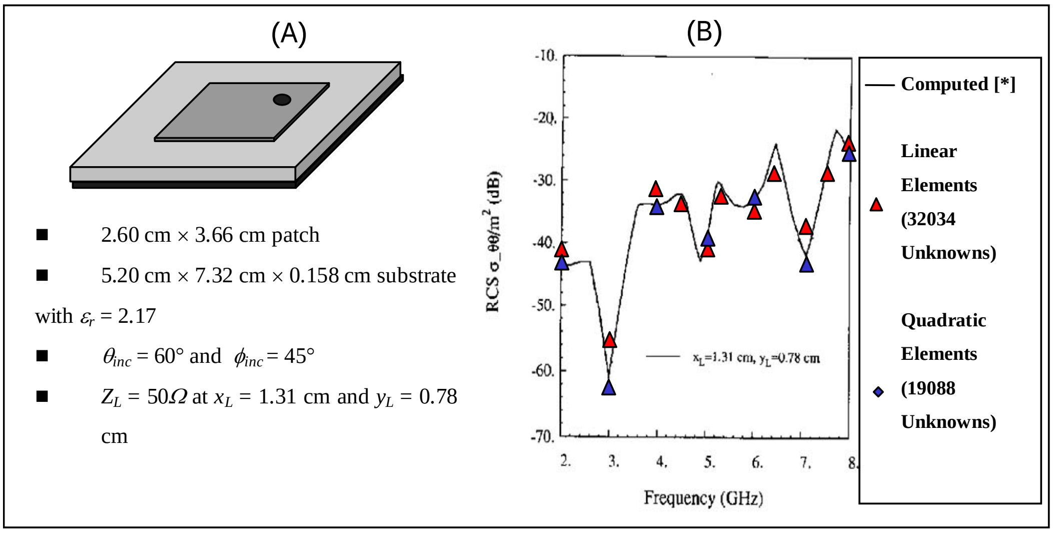
In this problem, the backscatter RCS of a loaded rectangular patch is analyzed. For this problem, the software is executed once for each frequency to compute the relevant backscatter RCS value. Results are given in Fig. 5.13.



[*]: Jin and Volakis

Fig. 5.13. Backscatter RCS of a Singly Loaded Rectangular Patch (Compared to [133]).

As a second example for this type, a different configuration -a differently located load- is analyzed. The corresponding results are given in Fig. 5.14.



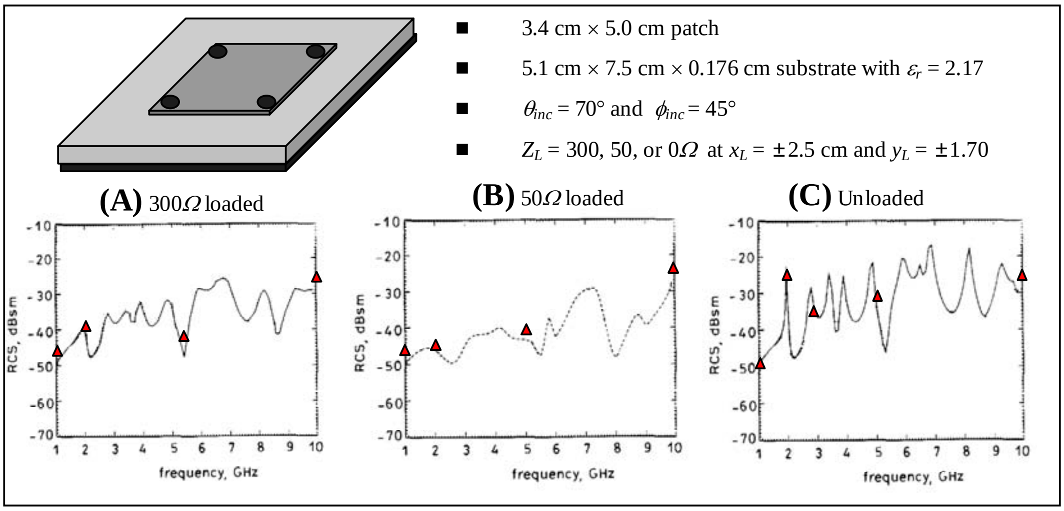
- 2.60 cm × 3.66 cm patch
- 5.20 cm × 7.32 cm × 0.158 cm substrate with $\varepsilon_r = 2.17$
- $\theta_{inc} = 60°$ and $\phi_{inc} = 45°$
- $Z_L = 50\Omega$ at x_L = 1.31 cm and y_L = 0.78 cm

[*]: Jin and Volakis

Fig. 5.14. Backscatter RCS of Another Singly Loaded Rectangular Patch (Compared to [133]).

5.2.1.3. Mutiply-Loaded Rectangular Microstrip Patch

In this problem, the backscatter RCS of a multiply-loaded rectangular patch is analyzed. For each control group, the load impedance (not the location) is changed this time. Again, the software is executed once for each frequency to compute the relevant backscatter RCS value. Results are given in Fig. 5.15.



- 3.4 cm × 5.0 cm patch
- 5.1 cm × 7.5 cm × 0.176 cm substrate with $\varepsilon_r = 2.17$
- $\theta_{inc} = 70°$ and $\phi_{inc} = 45°$
- $Z_L = 300, 50$, or 0Ω at $x_L = \pm 2.5$ cm and $y_L = \pm 1.70$

(A) 300Ω loaded

(B) 50Ω loaded

(C) Unloaded

Fig. 5.15. Backscatter RCS of a Multiply Loaded Rectangular Patch (Compared to [133]).

5.2.2. Results for Curved Scatterers with Patches

In this subsection rather than flat objects, RCS of the scatterers with curved edges and faces are considered.

5.2.2.1. Circular PEC Patch Above a Dielectric Cylinder

As a patched structure, this time a structure with curved faces is chosen. The scatterer is illustrated in Fig. 5.16 with relevant electrical dimensions and material properties.

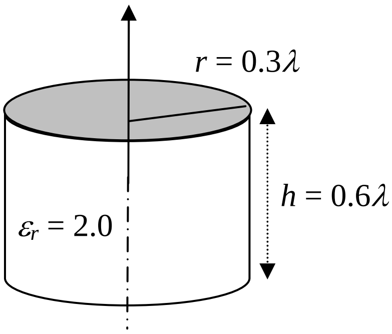


Fig. 5.16. Circular PEC Disk Above a Dielectric Cylinder.

1,152 quadratic hexahedra and 17,640 unknowns result in the solution illustrated Fig. 5.17. The results are compared to those of [134]. In [134], the solution was found by using the Method of Moments and the so-called Precorrected Fast Fourier Transform.

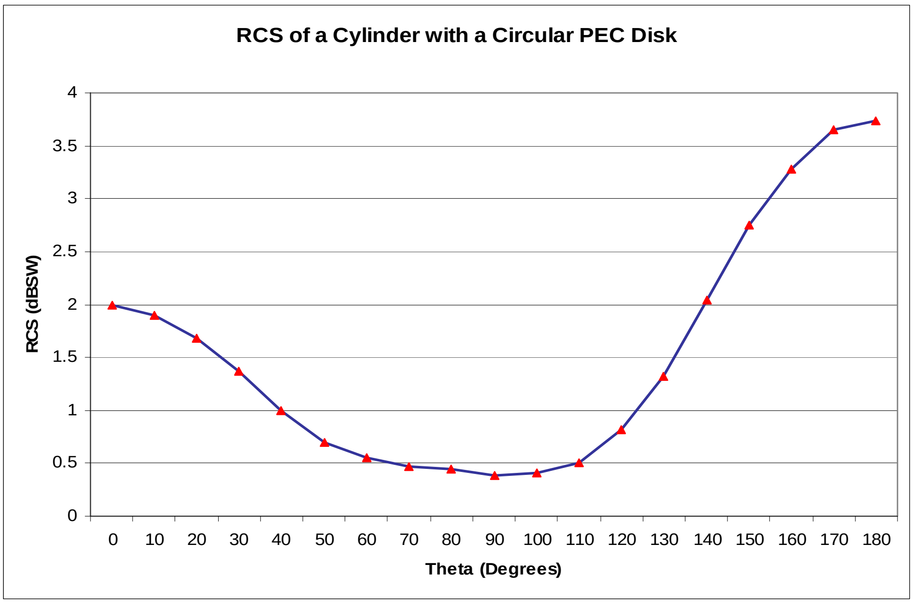


Fig. 5.17. RCS of a PEC Disk on a Dielectric Cylinder Calculated (Compared to [134]).

5.2.2.2. Circular PEC Patch Above a Dielectric Coated Sphere

As a second example of this kind, the RCS of a circular PEC patch on a dielectric spherical shell located on a PEC sphere is considered. The geometry is illustrated and the dimensions are given in Fig. 5.18. Again, the problem is solved with linear and quadratic hexahedral elements separately, where the incident field is a uniform plane wave of $\theta_i = \pi$. The solution approach and a cross-section of the generated mesh are illustrated in Fig. 5.19.

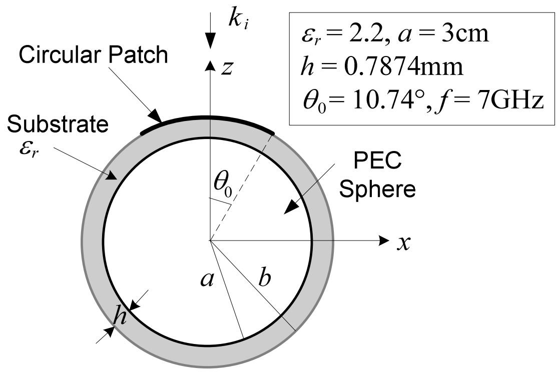
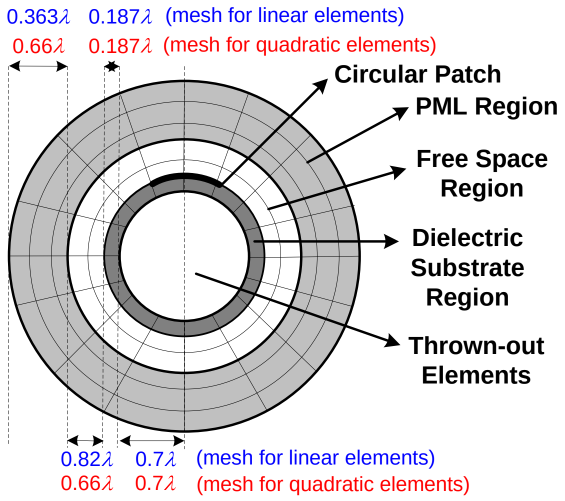


Fig. 5.18. Description of the Circular PEC Patch on Dielectric Coated PEC Sphere Problem.

A highly dense mesh generated for this problem is given in Fig. 5.20; where only the scatterer part of the mesh is illustrated for better visualization. In Fig. 5.20, the edges corresponding to the patch are illustrated via a different color/tone.



0.363λ
0.187λ
(mesh for linear elements)
0.66λ
0.187λ
(mesh for quadratic elements)
Circular Patch
PML Region
Free Space Region
Dielectric Substrate Region
Thrown-out Elements
0.82λ
0.7λ
(mesh for linear elements)
0.66λ
0.7λ
(mesh for quadratic elements)

Fig. 5.19. Cross Section of the Mesh Generated for the Circular PEC Patch on Dielectric Coated PEC Sphere Problem.

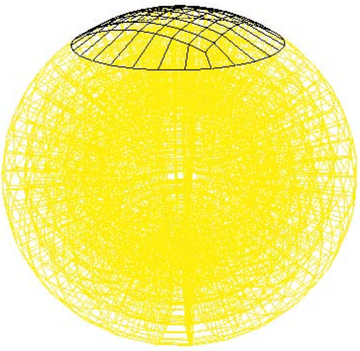


Fig. 5.20. 3D View of the Mesh Generated for the Circular PEC Patch on Dielectric Coated PEC Sphere Problem (Only Scatterer Shown).

Table 5.4. Comparison (Resource Requirement, Element Size) of Linear and Quadratic Elements for Circular PEC Patch on Dielectric Coated PEC Sphere Problem.

Element Type	**Element Size (λ)**					**Number of Elements**	**Number of Unknowns**
	along R	**along θ**		**along ϕ**			
	fixed	**min**	**max**	**min**	**max**		
Linear	0.090865	0.065	0.196	0.098	0.294	10,560	34,634
Quadratic	0.33*	0.183	0.576	0.262	0.863	960	13,480

*: Element Size 0.1873λ for the first level (dielectric substrate region);
033λ for further levels (free space and PML regions).

The results for this problem are given in Fig. 5.21, and compared with the results in the literature [134-135]. Moreover, resource requirements of each approach are summarized in Table 5.4. Again, quadratic hexahedral elements (up to size of 0.4λ) are proven to be successful in RCS calculation. In [134], the solution was found by using the Method of Moments and the so-called Precorrected Fast Fourier Transform. Hence it is not possible to make a resource usage (CPU time, memory, etc.) comparison with [134].

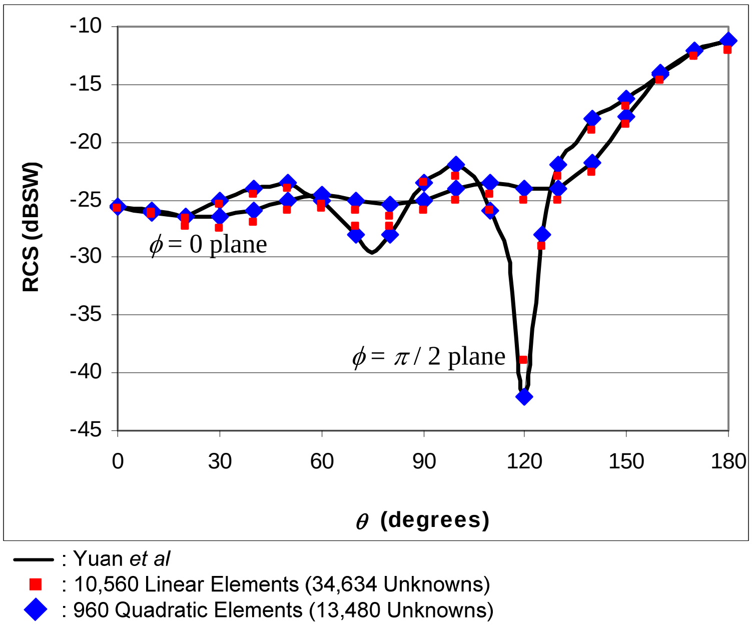


Fig. 5.21. Radar Cross Section of the circular PEC patch on dielectric coated PEC sphere (Compared to [134]).

5.3. General Discussions About the Results

Unfortunately, framework defining a set of benchmark scattering problems does not exist in the literature. In such a case, it would be possible to compare the proposed method to all other existing methods in the literature one-to-one in all aspects (accuracy, speed, memory, code complexity, etc).

Hence for various problems, the results have been compared to the ones (in terms of solution accuracy) in the literature [128-135] particularly. In each of these works, the authors have implemented and demonstrated the results of various methods, such as:

- Finite Element Boundary Integral (FE-BI) Method [128, 131-133],
- Multilevel Fast Multipole Method (MLFMM) [128],
- Method of Moments (MoM) with Precorrected Fast Fourier Transform (PFFT) [130-134].

Regardless of the method used in the compared publication, the method used in this work seems to be working properly and accurately (i.e. usage of Kameari’s element in the scattering problems is appropriate, and it is a promising technique in terms of accuracy and resource usage).

On the other hand, it could not be possible to compare the method with other methods for other aspects (CPU usage, memory usage, solution speed, etc) due to lack of information given about the other methods.

In [129], Finite Element Method has been applied together with “spheroid PML” formulation. Among the others, the method is the closest one to the method used in this work. However, since the authors did not give any information about the implementation details (element type, element shape, number of elements, number of unknowns, etc), again it could not be possible to perform a comparison in all aspects.

CHAPTER 6

AN OBJECT AND PATTERN ORIENTED APPROACH IN THE FINITE ELEMENT SOFTWARE DEVELOPMENT

6.1. Object Oriented Methodology and Software

In this section, the reasons of migration to the object oriented software are discussed. Unlike the other chapters, the contents of this chapter are not physical facts, or exact mathematical expressions (i.e. the content is just “a” correct approach; not “the” correct one, since there doesn’t exist a unique one). Two factors are important to develop all products; the user’s expectations, and the developer’s considerations. Naturally, these will apply to software products.

6.1.1. User’s Point Of View

From the user’s point of view, modern finite element software should have the following features:

- **Ease of use:** The finite element software should provide a graphical user interface from which the user can investigate the mesh, the boundary conditions, and also the results. It should also provide the user to select his/her preferences during the finite element analysis in an ergonomic manner.
- **Platform independence:** Considering that the end users of the finite element software are mostly academic researchers (and also some large institutes and companies), it is more acceptable to have the finite element software deployable to various platforms. A researcher might want to execute the software in his

Apple Macintosh or Windows PC at home or his office; as well as at the UNIX based workstation in his laboratory.

- **Adaptability to elements of various shape and various types:** The spectrum of the finite element software should be very wide in order to get attention both from the academic society and the commercial market. For example, software covering and targeting only tetrahedral node elements with 1 degree of freedom will be commented to be ordinary and old fashioned in the 21st century. The software should be capable of handling node, edge, facet, and volume elements of various shapes in 2D and 3D.
- **Inclusion of various algorithms for a specific purpose:** In almost every step (element selection and basis function construction, mesh quality measurement, mesh quality improvement, numerical integration during element matrix construction, stiffness matrix ordering, stiffness matrix reordering/ preconditioning, matrix solution, etc.) of the finite element analysis, there are numerous alternative competing algorithms and methods, and there are always trade-offs (speed, accuracy, memory, etc.) while deciding to use one of them. Modern software should allow the user to set the preferences at each step enabling the researchers also to perform comparative analyses.
- **Concurrency/Parallelization:** Adaptability to concurrent/parallel execution in single/multi-processor environments respectively is one of the key factors in modern numeric software. Since some steps in the finite element software are time consuming, it is nearly compulsory for the software to be deployable to distributed/parallel computing environments.
- **Interoperability:** Interoperability with the available mesh generation software in the literature or market is another key factor. Mesh generated by widely accepted and well known mesh generation software should be importable by the finite element software. This fact will spread the usage of the software in academic and industrial societies. Similarly, interoperability with the available matrix software/libraries in the literature or market is also important with the same reasons and arguments.

6.1.2. Design Considerations

On the other hand, the design aspects of modern finite element software should be as follows from the developer's point of view:

- **Ease of development:** The software development environment (computer aided design and engineering tools) should provide encouraging and comfortable rapid prototyping and development opportunities to the developer(s).
- **Modularity and adaptability:** Since the finite element world is a continuously evolving universe, adding new functionality and features is nothing but the nature of it. The software architecture should be so modular and adaptable that adding new functionality would not cause reinvention of the wheel, or one of the nightmares on Elm Street.
- **Understandability and Maintainability:** Considering that the evolution of such large software would take 5-10 years, the developer(s) might change. It is even not uncommon for a developer to forget his/her own development activities a couple of months later. Hence understandability of the code is important both for the ongoing developers and new-comers, not only during the development but also during the maintenance phase. Maintenance is usually the mostly ignored but the most trouble causing phase in the software life cycle. Considering the trends in the finite element analysis; it seems that adaptive, corrective, preventive and perfective maintenance activities (for a clean definition of maintenance activities see [136]) might be inevitable and crucial for the modern finite element software.
- **Testability:** Bottom-up integration and test, although underestimated and ignore usually, is as important as top-down analysis, design and development in software life cycle. From the very atomic portions of the code to the highest level, there should be systematic methods to perform the test activities in the code. Earlier the bugs are found, it is cheaper to fix them.
- **Good Documentation:** Another feature of the software language to be used for development (together with the available tools certainly) should be self-documented. Documentation is a headache and a big overhead for the developers due to its time consumption; but lack of documentation is the nightmare of new-comers and maintenance personnel on the other hand. Hence, the language should be self-descriptive; and there should be tools & methods to extract the

documentation in an elegant manner directly from the code itself. In such a case, there will not be an additional boring overhead of documentation impacting the developers.

- **Language Appropriateness:** Since the finite element analysis is depending on numerical methods, and it requires high speed in run-time; the language should be also appropriate for such purposes. Some high level languages like Java do not provide high speed run-time results due to their natures.
- **Platform independence:** "*Write once, run everywhere*" is the main aim of most of the software developers. Although the main reason can be specified as increasing the customer/consumer/end user spectrum of the end product; it is a well-known fact that it increases the self-satisfaction of the developers.

6.1.3. Object Oriented Languages and C++

Considering the design aspects and the compulsory/preferred features listed above, it is clear that the modern finite element software should be developed by using object oriented methodology. Object oriented methodology (analysis, design, development and test), which has aroused in the 1990s during the big software crisis after the end of the Cold War, is the trend in the software development world. Due to their strengths and advantages in terms of various factors, object oriented languages such as Lisp, Java and C++ have found very wide usage in the last one and a half decade. In a couple of years after their inventions, object oriented languages found broad usage also in the numeric methods software.

In 1994, almost 99% percent of the available numeric method software had been developed by structure oriented languages with the domination of Fortran and C. In the last decade, especially after the development of object oriented wrappers around popular libraries such as PetSc [137], BLAS [138], and LAPACK [139] (which were originally developed with structure oriented languages); researchers got rid of their hesitations to use object oriented methodology during the development of numeric method software. An up-to-date list of available object oriented numeric software of any kind can be found in [140].

Due to lack of support (in terms of computer aided engineering tools), Lisp might not be the right choice for the development of the finite element software. On the other hand, due to run-time inefficiency, Java should not be preferred in the finite element application, which requires hard and heavy run-time computing effort. For this purpose, C++ seems to be the ideal choice; and detailed information is in the following paragraphs.

C++, which is invented by Stroustrup [141] over a robust procedure oriented language C [142], is recently used by hundreds of thousands of programmers in essentially every application domain. This use is supported by about a dozen independent implementations, hundreds of libraries, hundreds of textbooks, several technical journals, many conferences, and innumerable consultants. Training and education at a variety of levels are widely available. Early applications tended to have a strong systems programming flavor. For example, several major operating systems have been written in C++ and many more have key parts done in C++. During the development of C++, Stroustrup considered uncompromising low level efficiency essential. This allows developers to use C++ to write device drivers and other software that rely on direct manipulation of hardware under real time constraints. In such code, predictability of performance and compactness are as important as raw speed. For most of the code development efforts, the important factors are

- maintainability,
- ease of extension, and
- ease of testing.

C++'s support for these concerns has led to its widespread use where reliability is a must and in areas where requirements change significantly over time. Examples are banking, trading, insurance, telecommunications, and military applications. For years, the central control of the U.S. long distance telephone system has relied on C++ and every 800 call (that is, a call paid for by the called party) has been routed by a C++ program. Many such applications are large and long lived. As a result, stability, compatibility, and scalability have been constant concerns in the development of C++. Million line C++ programs are not uncommon in practice. Like C, C++ wasn't specifically designed with numerical computation in mind. However, much numerical, scientific, and engineering computation is done in C++. A major reason for this is that traditional numerical work must often be combined with graphics and with computations relying on data structures that don't fit into the traditional Fortran mold. Graphics and user interfaces are areas in which C++ is heavily used. Anyone who has used either an Apple Macintosh or a Windows PC has indirectly used C++ because the primary user interfaces of these systems are C++ programs. In addition, some of the most popular libraries supporting X for UNIX are written in C++. Thus, C++ is a common choice for the vast number of applications in which the user interface is a major part. All of these points to what may be C++'s greatest strength: its ability to be used effectively for applications that require work in a variety of application areas. It is quite common to find an application that involves local and widearea networking, numerics, graphics, user interaction, and database access. Traditionally, such application areas have been considered distinct, and they have most often been served by distinct technical communities using a variety of programming languages. However, C++ has been widely used in all of those areas.

Furthermore, C++ is able to coexist with code fragments and programs written in other languages. C++ is widely used for teaching and research. This has surprised some who – correctly – point out that C++ isn't the smallest or cleanest language ever designed. It is, however

- clean enough for successful teaching of basic concepts,
- realistic, efficient, and flexible enough for demanding projects,
- available enough for organizations and collaborations relying on diverse development and execution environments,
- comprehensive enough to be a vehicle for teaching advanced concepts and techniques, and
- commercial enough to be a vehicle for putting what is learned into nonacademic use.

In summary, as its inventor Stroustrup mentioned: "*C++ is a language that you can grow with*."

6.1.4. Standard Template Library of C++

Another important feature of the C++ is that it comes with a standard library handling most of the boring and complicated low level operations. This yields the developers to focus on its own work and forget any indirect business.

The C++ standard library:

1. Provides support for language features, such as memory management and runtime type information.
2. Supplies information about implementation defined aspects of the language, such as the largest *float* value.
3. Supplies functions that cannot be implemented optimally in the language itself for every system, such as *sqrt*() and *memmove*().
4. Supplies nonprimitive facilities that a programmer can rely on for portability, such as lists, maps, sort functions, and I/O streams.
5. Provides a framework for extending the facilities it provides, such as conventions and support facilities that allow a user to provide I/O of a userdefined type in the style of I/O for builtin types.
6. Provides the common foundation for other libraries.

In addition, a few facilities – such as random number generators – are provided by the standard library simply because it is conventional and useful to do so. The design of the library is primarily determined by the last three roles. These roles are closely related. For example, portability is commonly an important design criterion for a specialized library, and common container types such as lists and maps are essential for convenient communication between separately developed libraries.

The heart of the C++ standard library, the part that influenced its overall architecture, is the standard template library (STL). The STL is a generic library that provides solutions to managing collections of data with modern and efficient algorithms. It allows programmers to benefit from innovations in the area of data structures and algorithms without needing to learn how they work. From the programmer's point of view, the STL provides a bunch of collection classes that meet different needs, together with several algorithms that operate on them. All components of the STL are templates, so they can be used for arbitrary element types. But the STL does even more: It provides a framework for supplying other collection classes or algorithms for which existing collection classes and algorithms work. All in all, the STL gives C++ a new level of abstraction. There is no need for programming dynamic arrays, linked lists, and binary trees; or programming different search algorithms. To use the appropriate kind of collection, one simply defines the appropriate container and calls the member functions and algorithms to process the data. The STL's flexibility, however, has a price, chief of which is that it is not self-explanatory. Therefore, the subject of the STL fills several chapters in many books. An introductory reading about STL is [143], and more advanced topics for efficient usages in complex components can be found in [144].

In summary, the STL is based on different well-structured components, which are containers, iterators, and algorithms.

- Containers are used to manage collections of objects of a certain kind. The containers may be implemented as arrays or as linked lists, or they may have a special key for every element.

- Iterators are used to step through the elements of collections of objects. These collections may be containers or subsets of containers. For example, one operation lets the iterator step to the next element in the collection. This is done independently of the internal structure of the collection. Regardless of whether the collection is an array or a tree, it works.

- Algorithms are used to process the elements of collections. For example, they can search, sort, modify, or simply use the elements for different purposes. Algorithms use iterators. Thus, an algorithm has to be written only once to work with arbitrary containers because the iterator interface for iterators is common for all container types.

Even at first glance, it can be stated that STL provides tools & methods which are very suitable to the finite element analysis. By using the templates, it is possible to define and use a structure regardless of its type. By means of the templates, only the definition of “vector” is sufficient; then it is possible to use this template for a vector of real numbers, or for a vector of nonnegative integers, or even for a vector of a custom type, etc. Iterators and algorithms also provide infrastructure to very complicated operations which are encountered in the finite element analysis. For example, during the imposure of the boundary conditions, some matrix rows/columns or entries should be deleted and the remaining entries should be shifted accordingly. Implementation of such an algorithm in Fortran or C might be a mess; however STL handles most of it.

6.1.5. Migration to Object Oriented Methodology in FEM: FEM++ ?

Regarding all the information given above, the current trend in the finite element software is naturally migration to object oriented architectures. Various papers focused on different subjects have been published [145-152]. The successfully leading and mature example of work products are:

- OOFEM developed by Patzak [153-154], which is a full product but only limited to node elements,
- FEMSTER developed by Castillo *et al* [155-156], which is not an end product but an object oriented library and framework providing components (finite elements of various shapes and types) to finite element software researchers and developers,
- deal.II developed by Bangerth *et al* [157], which is again not an end product but an object oriented library and framework providing components (not only finite elements of various shapes and types; but also error estimators) to finite element software researchers and developers.

6.1.6. Design Patterns in FEM

Another methodology providing a common understanding and improving the readability and reusability of object oriented codes is the usage of design patterns. Gamma *et al* (also known as GoF standing for "*Gang of Four*") encyclopedically listed 23 such patterns [158], classified as creational, structural and behavioral patterns; which are widely used and specifically known in the object oriented design world. In summary, design patterns have been defined by GoF in order to bring more standardization, make people live and feel *déjà-vu*'s during the design and carry their experiences by means of analogies.

Examples about the usage of design patterns in the modern finite element software can be listed as follows:

1. For the finite element software, there are numerous ways of implementing an operation. For example, basis functions can be defined by means of different orthogonal polynomials. Any creational pattern, such as “Abstract Factory”, can be used for the creation of different orthogonal polynomials of different orders. Such factories can be defined for other operations such as curve production, quadrature point production etc.
2. The global stiffness matrix can be implemented by means of the creational “Singleton” pattern, which guarantees the uniqueness of the matrix.
3. As another example, there are numerous algorithms for the solution of the matrix equation. Among the structural ones, the “Strategy” pattern can be used to implement various matrix solvers with a single unique interface. It can also be used where multi-algorithm alternatives exist (such as mesh quality measurement, improvement, etc).
4. For *hp*-version finite element method, some elements might have additional attributes (some other modes and more basis functions) than the others. “Decorator” pattern, which is a behavioral pattern, seems to be suitable to use for such purposes.

More cases about the usage of design patterns can certainly be found after more a detailed analysis and design phase.

6.2. An Object and Pattern Oriented Finite Element Software Proposal

The modern finite element software should have a layered architecture in order to decrease the dependency between components requiring different type of expertise, and to enable parallel development of different components by several development teams. Conventionally, graphical user interface has always been distinguished from the business logic portion of the code, which is considered as the application layer. Especially after the object oriented era, utilities (either reused from commercial/free libraries; or newly developed) providing special purpose infrastructure (e.g. mathematical function libraries in a numerical method software, or coordinate conversion and projection library in a geographical information software) and hardware/operating system accessing code (e.g. communication code making use of operating system calls and network cards) are considered to be as framework. Such layering mechanism lets developers concentrate only on their component; not necessarily know the details but just the interfaces/services of other components.

6.2.1. Architectural Decisions

A proposal of the modern finite element software architecture is given in Fig. 6.1. The graphical user interface layer consists of components:

i) Window Element Handler: handling general window elements (any action on menus, tabular displays, buttons etc.);

ii) Mesh Handler: illustrating and manipulating the mesh in a 3D environment (zooming in/out; rotation; adjusting camera position and angle etc.); and

iii) FEM Data Handler: handling the whole geometry and the preferences during the finite element analysis; interfacing with the necessary components of the application layer.

Regarding the platform independency design consideration mentioned in previous sections;

i) QT [159], which is an advanced platform independent library for widget and other user interface element generation and manipulation, might be a useful tool during the development of the Window Element Handler.

ii) OpenGL [160-161], which is an advanced platform independent library for 3D graphics manipulation, seems to be suitable for the development of the Mesh Handler.

Application layer consists of business logic components:

i) Mesh Manager: importing the mesh from a formatted file or a set of formatted files , handling it throughout the execution, and exporting/saving it to the file(s) of same format(s).

ii) Resource Manager: allocating the resources (CPU, memory, threads) to necessary operations; i.e. handling the concurrency and parallelization issues during the execution,

iii) FEM Manager: performing the necessary actions in the finite element analysis in relevant order.

Again regarding the platform independency design consideration; the mesh file to be parsed by the Mesh Manager can be chosen to be in XML format [162]; which is a structured, well organized, human-readable format appropriate for hierarchical data structures.

The framework consists of:

i) Utilities like a general purpose matrix library and sparse matrix utilities (either to be developed from the scratch; or object oriented wrapped versions of available libraries such as PETSc, BLAS, LAPACK, UMFPACK etc.),

ii) Communication infrastructure especially to be used for parallel/distributed environment (where ACE [163] and CORBA [164] can be effectively used).

Similarly, the framework tools are proposed in a manner that the platform independency is preserved.

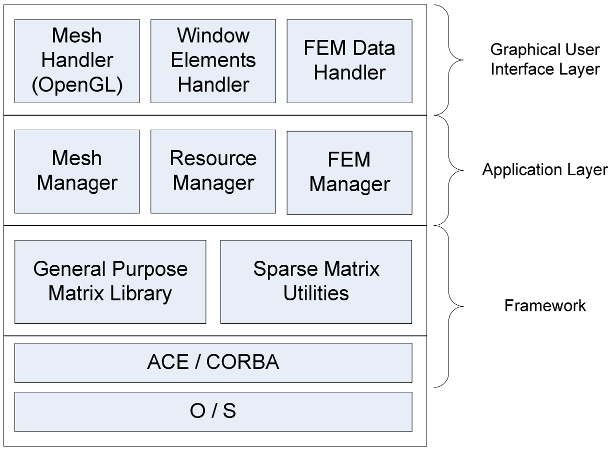


Fig. 6.1. Layered Architecture of the FEM Software.

The mesh viewing software developed in this thesis is an OpenGL application, and it can be considered as a prototype for the GUI Layer. A screen snapshot of it is given in Fig. 6.2.

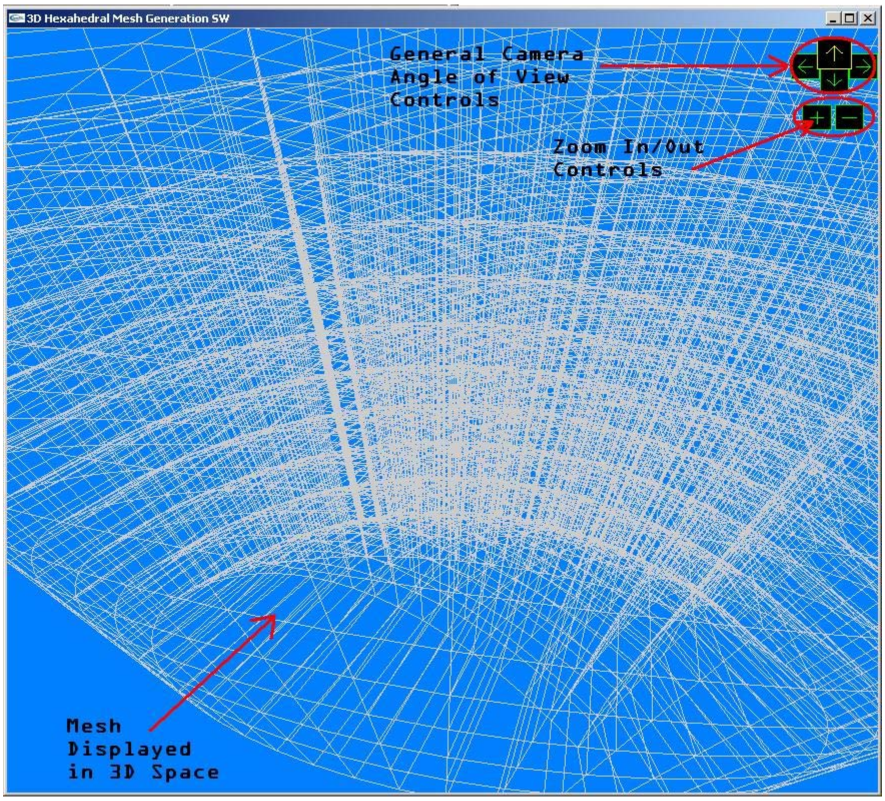

3D Hexahedral Mesh Generation SW
General Camera Angle of View Controls
Zoom In/Out Controls
Mesh Displayed in 3D Space

Fig. 6.2. A Screen Snapshot of OpenGL based GUI.

6.2.2. UML Analysis of the Proposed Architecture

Another aspect of the design is the user’s point of view (i.e. user’s expectations about the functionality of the program). Generally, UML use case diagrams [165] are used for this purpose. Readers, who are not familiar with UML, might proceed to Appendix E for detailed information about it and its notation.

The modern finite element software proposed here has the system boundaries as seen in Fig. 6.3 (i.e. According to user’s request, it imports a mesh created by an external mesh generation software and performs finite element analysis). Namely mesh generation, which is considered as another world or universe, is out of the scope of the proposed software.

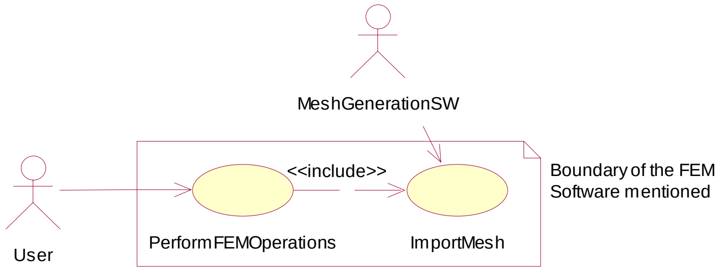


Fig. 6.3. System Boundary of the FEM Software.

The use cases of the software might certainly be detailed and extended. Fig.s 6.4, 6.5, 6.6 and 6.7 illustrate some more use cases in order to give the complete functionality set provided by the software.

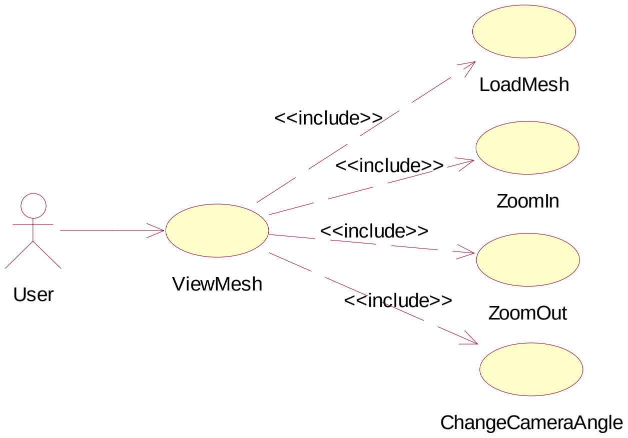


Fig. 6.4. Mesh Viewing Functionality Provided by the FEM Software.

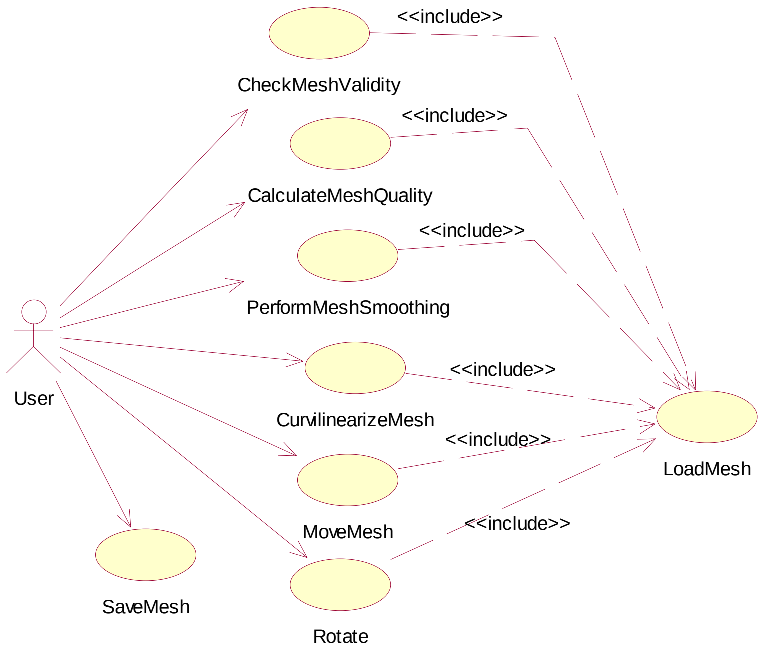


Fig. 6.5. Mesh Manipulation Functionality Provided by the FEM Software.

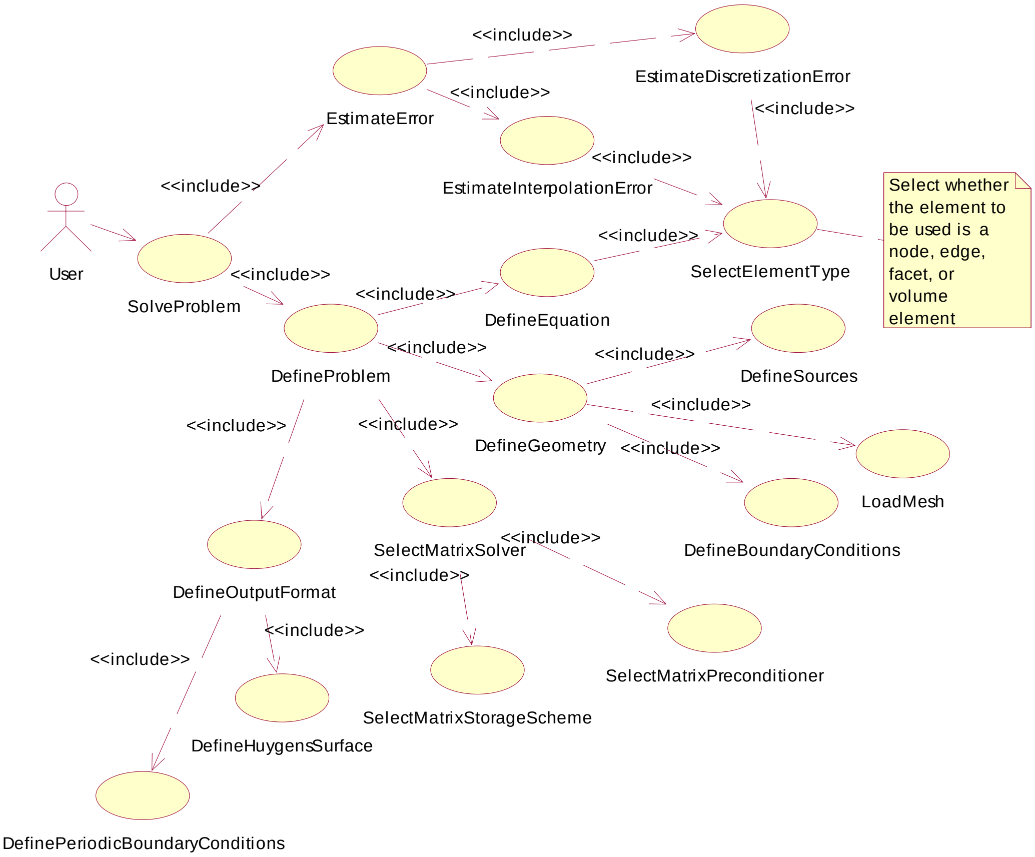
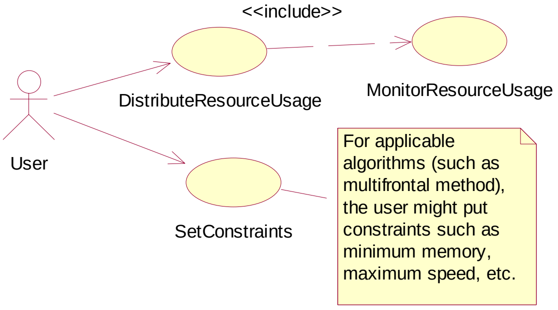


Fig. 6.6. Direct FEM Functionality Provided by the FEM Software.



<<include>>
DistributeResourceUsage
MonitorResourceUsage
User
SetConstraints
For applicable algorithms (such as multifrontal method), the user might put constraints such as minimum memory, maximum speed, etc.

Fig. 6.7. Management Functionality Provided by the FEM Software.

Another point of view is construction and analysis of the data structures in order to have an idea of the interfaces in mind. Class diagrams are used for this purpose. For example, class diagram illustrating the mesh data structure is given in Fig. 6.8. Moreover, the data structures of the elements (showing their relationships and hierarchy) are given in Fig.s 6.9 and 6.10.

Patterns used in the design can also be addressed and specified in the class diagrams. Abstract factory pattern used in polynomial, curve and quadrature point generation is illustrated in Fig.s 6.11, 6.12, and 6.13 respectively.

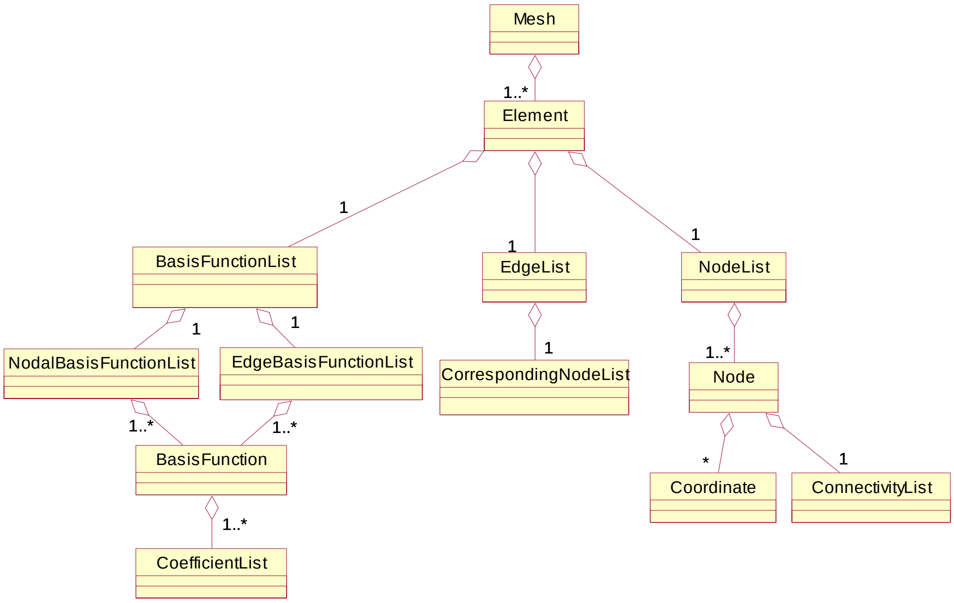


Fig. 6.8. Mesh Data Structure inside the FEM Software.

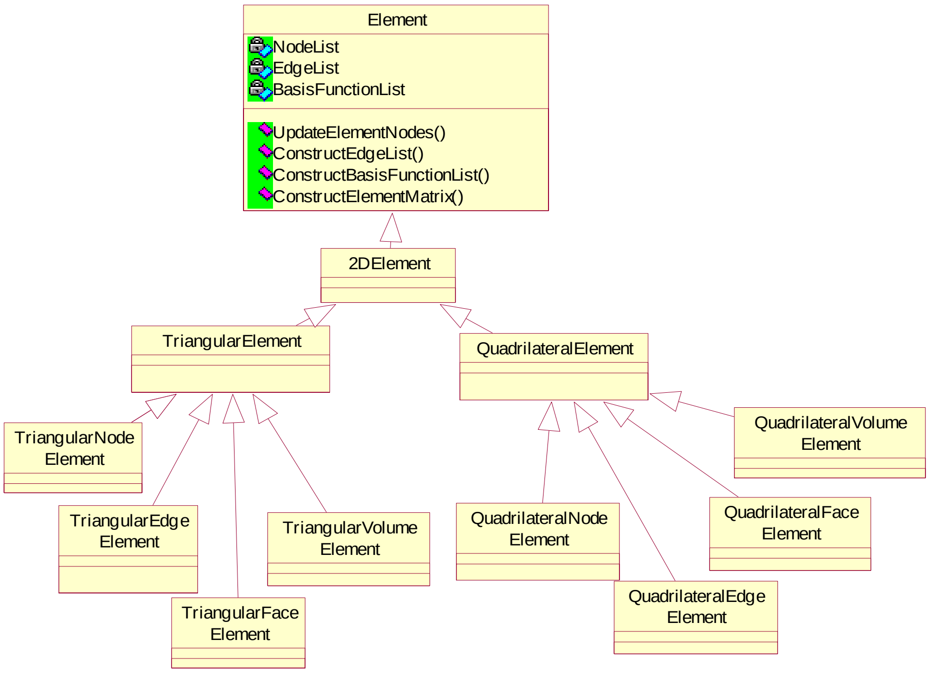


Fig. 6.9. 2D Element Data Structure inside the FEM Software.

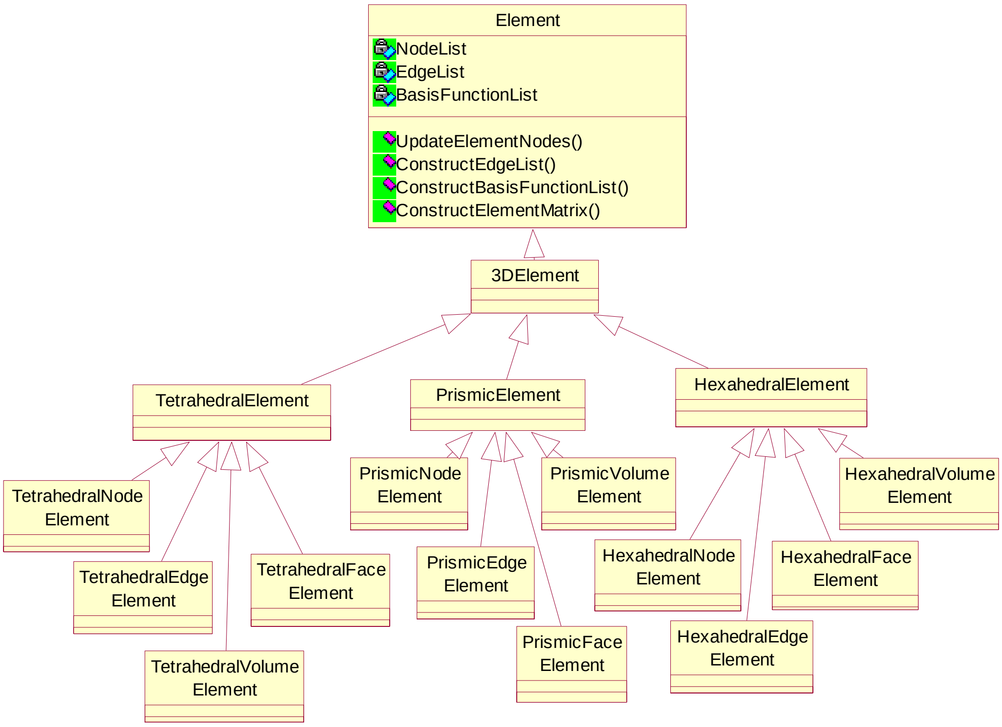


Fig. 6.10. 3D Element Data Structure inside the FEM Software.

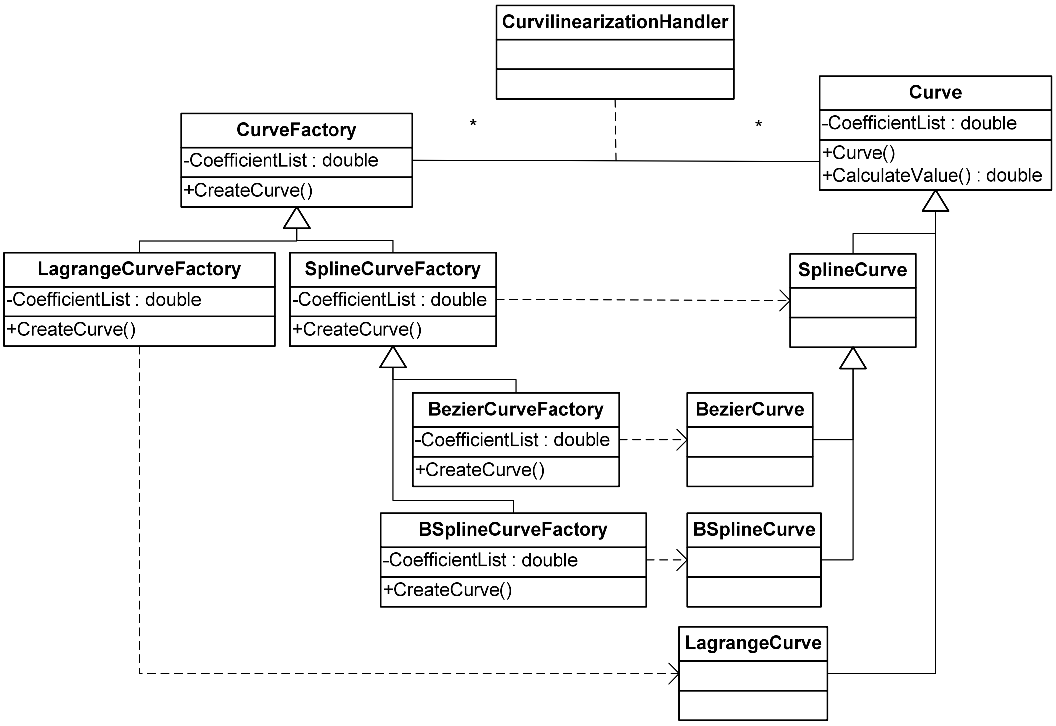


Fig. 6.11. Curve Data Structure and Abstract Curve Factory inside the FEM Software.

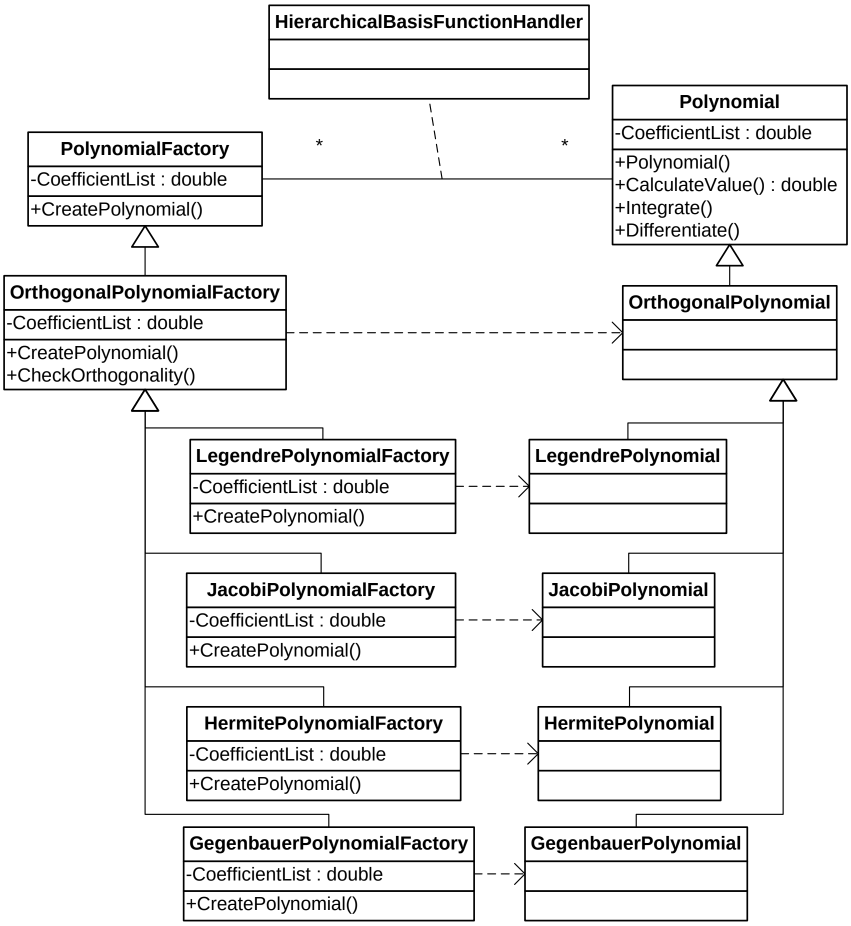


Fig. 6.12. Polynomial Data Structure and Abstract Polynomial Factory inside the FEM Software.

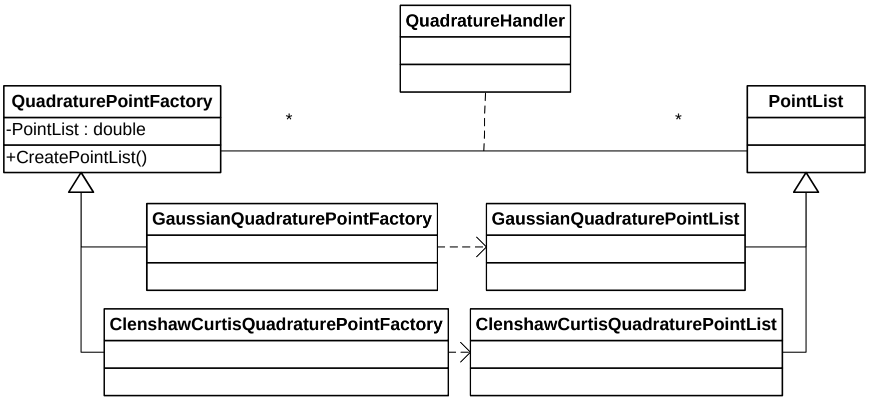

QuadratureHandler
QuadraturePointFactory
-PointList : double
+CreatePointList()
*
*
PointList
GaussianQuadraturePointFactory
GaussianQuadraturePointList
ClenshawCurtisQuadraturePointFactory
ClenshawCurtisQuadraturePointList

Fig. 6.13. Quadrature Point Data Structure and Abstract Quadrature Point Factory inside the FEM Software.

CHAPTER 7

CONCLUSIONS

In many engineering disciplines, approximate solutions of differential or integral equations play an important role to analyze or design complicated engineering systems. Several examples can be found from various branches such as Computational Fluid Dynamics (CFD), Computational Electromagnetics (CEM), heat transfer applications, Structural Mechanics, etc. In all such applications, the spatial domain must be discretized by generating a mesh, which is a collection of elements with simple shapes. Then the operator equations (i.e. partial differential equations or integral equations) are solved by using the well-known methods such as finite differences, finite elements or method of moments.

Among these, the Finite Element Method is a powerful and useful tool employed in the numerical solution of partial differential equations that arise in different applications. The technique allows for the solution of practical problems that would otherwise be intractable for analytical methods because of non-linearities or complex geometries. However, to achieve the full benefits of considering arbitrary geometries, there must exist simple and efficient means to generate the required meshes.

From the viewpoint of applications, three-dimensional problems are much more important (but more difficult as well) than their two-dimensional counterparts. Mesh generation is crucial in the application of FEM in three-dimensional problems. On the other hand all-hexahedral meshing, which yields the most accurate finite element solutions, is the most challenging topic in the mesh generation era.

The aim of this thesis is to apply the higher order hexahedral edge elements to electromagnetic scattering problems together with generic implementations. For this purpose, three separate stand-alone software products (all-hexahedral mesh generation; 3D mesh viewing, and finite element solver by means of hierarchical hexahedral edge elements) were developed. Moreover, a separate Matlab script was developed for hexahedral mesh smoothing with Particle Swarm Optimization in order to investigate the effects of mesh quality on the solution accuracy.

Perfectly Matched Layers (PMLs) which are implemented by using a complex coordinate transformation, have been successfully used for mesh truncation in this software. Material uniformities have been handled during the very initial all-hexahedral mesh generation in order to support both the mesh visualization and the finite element solution.

In the three-dimensional Finite Element Analysis, the number of elements are typically very large (in the order of tens of thousands of unknowns), so the resulting matrices for those systems are large but fortunately sparse. In the finite element solver different sparse storage schemes, each of which is appropriate for a different solver, have been used. The row-indexed sparse storage mode is optimized for multiplication of the matrix (or the transpose of the matrix) with a vector from the left. This is a very good property since the sparse matrices need to be operated over other matrices to construct the system of equations and matrix-vector multiplications are needed during the solution of the system of equations using the biconjugate gradient method. The other sparse storage scheme is optimized for Gaussian elimination like operations, which are performed during the multifrontal method.

To the author’s belief, two original contributions have been made throughout this thesis:

Performance of quadratic hexahedral edge elements has been deeply investigated over the radar cross-sections of several curved or flat objects with or without patches. Instead of the widely known and accepted “0.1λ linear element size” criterion, it has been observed and concluded that “0.3-0.4 λ quadratic element size” is a new potential criterion for electromagnetic scattering and radiation problems. Analyses have shown that the usage of quadratic elements is not only more confident, but also computationally cheaper.

The second original research topic in this thesis is the mesh improvement performed by optimization based mesh smoothing. The smoothing has been performed by means of the Particle Swarm Optimization, which found wide application in the last decade. During the smoothing, a condition number based combined hexahedral quality metric was used.

There remain several avenues for further research in this work. Deeper investigation of the effects of edge ordering to the stiffness matrix storage and solution (focused on hexahedral edge elements) might be one of these. Object and pattern oriented finite element software development, software size and cost estimation are other potential research areas.

Certainly, application of higher order (third, fourth, fifth and even more) hierarchical hexahedral edge elements to electromagnetic scattering problems might be an extension to this work. Similarly, multiobjective hexahedral mesh smoothing can be considered as another near-future term work.

REFERENCES

[1] R. Courant, “Variational methods for a solution of problems of equilibrium and vibrations”, *Bull. Amer. Math. Soc.*, vol. 49, pp. 1-23, 1943.

[2] J. H. Argyris, “Energy theorems and structural analysis”, *Aircraft Engineering*, vol. 26, pp. 347-356, 1954.

[3] A. E. Yılmaz and M. Kuzuoğlu, “Elektromanyetik Sınır Değer Problemlerinin İkinci Dereceden Altı Yüzlü Sonlu Elemanlar İle Modellenmesi”, in *Proc. URSI-Türkiye 2004 2. Ulusal Kongresi*, pp. 78-80, 2004.

[4] A. E. Yılmaz and M. Kuzuoğlu, “Mikroşerit Yama Antenlerin Sonlu Elemanlar Yöntemi ile Modellenmesi”, in *Proc. URSI-Türkiye 2006 3. Ulusal Kongresi*, pp. 146-148, 2006.

[5] A. E. Yılmaz and M. Kuzuoğlu, “Elektromanyetik Sınır Değer Problemlerinin Düzgün Olmayan Ağlar ve Altı Yüzlü Sonlu Kenar Elemanları ile Modellenmesi”, in *Proc. URSI-Türkiye 2006 3. Ulusal Kongresi*, pp. 125-127, 2006.

[6] A. E. Yilmaz and M. Kuzuoglu, “Comparison of Linear And Quadratic Hexahedral Edge Elements in Electromagnetic Scattering Problems”, to appear in *AEÜ – International Journal of Electronics and Communications*.

[7] A. E. Yilmaz and M. Kuzuoglu, “A Particle Swarm Optimization Approach in Hexahedral Mesh Smoothing”, submitted to *Communications in Numerical Methods in Engineering*.

[8] A. E. Yilmaz and M. Kuzuoglu, “Parçacık Sürü Optimizasyonu ile Altı Yüzlü Eleman Ağlarının İyileştirilmesi - Hexahedral Mesh Smoothing by Means of Particle Swarm Optimization”, in *CD-ROM Proc. IEEE 15. Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SİU-2007)*, Eskişehir, Türkiye.

[9] S. E. Benzley, E. Perry, K. Merkley and B. Clark, “A Comparison of All Hexagonal and All Tetrahedral Finite Element Meshes for Elastic and Elastic-Plastic Analysis”, *Proc. 4th Int. Meshing Roundtable*, pp. 179-191, 1995.

[10] T. Blacker, “Meeting the Challenge for Automated Conformal Hexahedral Meshing”, *Proc. 9th Int. Meshing Roundtable*, 2000.

[11] A. Sheffer, T. Blacker and M. Bercovier, “Clustering: Automated Detail Suppression Using Virtual Topology”, *Joint ASME/ASCE/SES Summer Meeting*, June 1997.

[12] A. Sheffer, T. Blacker, J. Clements and M. Bercovier, ``Virtual Topology Operators for Meshing", *Proc. 6th Int. Meshing Roundtable*, pp. 49-65, 1997.

[13] S. S. Liu and R. Gadh, “Basic LOgical Bulk Shapes (BLOBs) for Finite Element Hexahedral Mesh Generation”, *Proc. 5th Int. Meshing Roundtable*, 1996.

[14] C. G. Armstrong, D. J. Robinson, R. M. McKeag, T. S. Li, S. J. Bridgett, R. J. Donaghy and C. A. McGleenan, “Medials for Meshing and More”, *Proc. 4th Int. Meshing Roundtable*, 1995.

[15] T. D. Blacker, M. B. Stephenson, J. L. Mitchiner, L. R. Phillips and Y. T. Lin, “Automated Quadrilateral Mesh Generation: A Knowledge System Approach”, *ASME*, Paper No. 88-WA/CIE-4.

[16] S. Liu and R. Gadh, “Automatic Hexahedral Mesh Generation by Recursive Convex and Swept Volume Decomposition”, *Proc. 6th Int. Meshing Roundtable*, pp. 217-231, 1997.

[17] A. Sheffer, M. Etzion, A. Rappoport and M. Bercovier, “Hexahedral Mesh Generation using the Embedded Voronoi Graph”, *Proc. 7th Int. Meshing Roundtable*, 1998.

[18] R. J. Meyers, T. J. Tautges and P. M. Tuchinsky, “The “Hex-Tet” Hex-Dominant Meshing Algorithm as Implemented in CUBIT”, *Proc. 7th Int. Meshing Roundtable*, pp. 151-158, 1998.

[19] W. P. Thurston, “Hexahedral decomposition of polyhedra”, http://www.ics.uci.edu/~eppstein/gina/Thurston-hexahedra.html, *Posting to sci.math, 25 October 1993*, (last accessed Jul 1, 2007).

[20] S. A. Mitchell, “A Characterization of the Quadrilateral Meshes of a Surface Which Admits a Compatible Hexahedral Mesh of Enclosed Volume”,*5th MSI WS.Comp. Geometry*, 1995.

[21] S. Mitchell, “A Characterization of the Quadrilateral Meshes of a Surface Which Admit a Compatible Hexahedral Mesh of the Enclosed Volume”, *Proc. 13th Annual Symposium on Theoretical Aspects of Computer Science*, Springer, pp. 465-476, 1996.

[22] D. Eppstein, “Linear Complexity Hexahedral Mesh Generation”, *Computational Geometry '96*, ACM (1996).

[23] M. Stephenson and T. Blacker, “Using Conjoint Meshing Primitives to Generate Quadrilateral and Hexahedral Elements in Irregular Regions”, *ASME*, G0502B, 1989.

[24] P. M. Knupp, “Next-Generation Sweep Tool: A Method for Generating All-Hex Meshes on Two-and-One-Half Dimensional Geometries”, *Proc. 7th Int. Meshing Roundtable*, pp. 505-514, 1998.

[25] M. L. Staten, S. A. Canann and S. J. Owen, “BMSweep: Locating Nodes During Sweeping”, *Proc. 7th Int. Meshing Roundtable*, pp. 7-17, 1998.

[26] R. Schneider, “Automatic Generation of Hexahedral Finite Element Meshes”, *Proc. 4th Int. Meshing Roundtable*, pp. 103-114, 1995.

[27] J. Zhu and T. Blacker, “Overcoming Cartesian Grid Generation Obstacles”, *Proc. 7th Int. Conference on Numerical Grid Generation in Computational Field Simulations*, 2000.

[28] R. Taghavi, “Hexar: Automatic, Parallel and Fault Tolerant Mesh Generation from CAD on Cray Research Supercomputers”, *Proc. 3rd Int. Meshing Roundtable*, 1994.

[29] R. Smith, “A Novel Cartesian Grid Method for Complex Aerodynamic CFD Applications”, *Proc. 5th Int. Conference on Numerical Grid Generation in Computational Field Simulations*, pp. 709-718, 1996.

[30] N. Chiba, I Nishigaki, Y. Yamashita, C. Takizawa, and K. Fujishiro, “A flexible automatic hexahedral mesh generation by boundary-fit method”, *Comp. Meth. in Appl. Mech. And Eng.* Vol. 161, pp. 145-154, 1998.

[31] K. Miyoshi and T. D. Blacker, “Hexahedral Mesh Generation Using Multi-Axis Cooper Algorithm”, *Proc. 9th Int. Meshing Roundtable*, 2000.

[32] M. E. Hohmeyer and W. Christopher, “Fully-Automatic Object-Based Generation of Hexahedral Meshes”, *Proc. 4th Int. Meshing Roundtable*, pp.129-138, 1995.

[33] T. D. Blacker, “The Cooper Tool”, *Proc. 5th Int. Meshing Roundtable*, pp. 13-29, 1996.

[34] D. R. White, L. Mingwu, S. E. Benzley and G. D. Sjaardema, “Automated Hexahedral Mesh Generation by Virtual Decomposition”, *Proc. 4th Int. Meshing Roundtable*, pp. 165-176, 1995.

[35] M. B. Stephenson, S. A. Canann, T. D. Blacker and R. J. Meyers, “Plastering Progress Report I”, *SAND89-2192, Sandia National Laboratories*. 1992.

[36] T. D. Blacker and R. J. Meyers, “Seams and Wedges in Plastering: A 3-D Hexahedral Mesh Generation Algorithm”, *Engineering with Computers*, vol. 9, pp. 83-93, 1993.

[37] T. Blacker and M. Stephenson. Paving: A New Approach to Automated Quadrilateral Mesh Generation. *International Journal for Numerical Methods in Engineering*, vol. 32, pp. 811–847, 1991.

[38] K. Kovalev, “Unstructured Hexahedral Non-conformal Mesh Generation”, *Ph. D. Thesis*, Vrije Universtiteit Brussel, December 2005.

[39] R. Cass, S. Benzley, R. Meyers, and T. Blacker, “Generalized 3D Paving: An Automated Quadrilateral Surface Mesh Generation Algorithm”, *International Journal for Numerical Methods In Engineering*, vol. 39, pp. 1475–1489, 1996.

[40] R. J. Meyers, T. J. Tautges and P. M. Tuchinsky, “The “Hex-Tet” Hex-Dominant Meshing Algorithm as Implemented in CUBIT”, *Proc. 7th Int. Meshing Roundtable*, pp. 151-158, 1998.

[41] T. Tautges, T. Blacker and S. Mitchell, “The Whisker Weaving Algorithm: A Connectivity-Based Method for Constructing All-Hexahedral Finite Element Meshes”, *Int. J. Numerical Methods in Engineering*, vol. 39, pp. 3327-3349, 1996.

[42] N. T. Folwell and S. A. Mitchell, “Reliable Whisker Weaving via Curve Contraction”, *Proc. 7th Int. Meshing Roundtable*, pp. 365-378, 1998.

[43] P. Murdoch and S. Benzley, “The Spatial Twist Continuum”, *Proc. 4th International Meshing Roundtable*, pp. 243–251, 1995.

[44] P. Murdoch, S. Benzley, T. Blacker, and S. A. Mitchell, "The spatial twist continuum: a connectivity based method for representing all-hexahedral finite element meshes", *Finite Elements in Analysis and Design*, vol. 28, no. 2, pp. 137-149, Dec. 1997.

[45] M. Borden, S. Benzley, S. Mitchell, D. White and R. Meyers, “The cleave and fill tool: An all-hexahedral refinement algorithm for swept meshes,” *Proc. 9th International Meshing Roundtable*, pp. 69-76, 2000.

[46] L. Marechal, “A new approach to octree-based hexahedral meshing,” *Proc. 10th International Meshing Roundtable*, pp. 209-221, 2001.

[47] R. Schneiders, “Octree-based hexahedral mesh generation,” *Int. Journal of Comp. Geom. & Applications*, vol. 10, no. 4, pp. 383-398, 2000.

[48] K. Tchon, C. Hirsch, and R. Schneiders, "OctreeBased Hexahedral Mesh Generation for Viscous Flow Simulations", *Proc. 13th AIAA Computational Fluid Dynamics Conference*, AIAA-971980, 1997.

[49] D. Field, “Laplacian Smoothing and Delaunay Triangulations”, *Communications and Applied Numerical Methods*, vol. 4, pp. 709–712, 1988.

[50] L. Freitag, “On Combining Laplacian and Optimization-Based Mesh Smoothing Techniques”, *Trends In Unstructured Mesh Generation*, vol. 220, pp. 37–43, ASME, July 1997.

[51] L. Freitag and C. Ollivier-Gooch, “Tetrahedral Mesh Improvement Using Swapping and Smoothing”, *International Journal for Numerical Methods in Engineering*, vol. 40, pp. 3979–4002, 1997.

[52] P. Hansbo, “Generalized Laplacian Smoothing Of Unstructured Grids”, *Communications in Numerical Methods in Engineering*, vol. 11, pp. 455–464, 1995.

[53] N. Jones and S. Wright “Algorithm For Smoothing Triangulated Surfaces”, *Journal of Computing in Civil Engineering*, ASCE vol. 1, pp. 85–102, 1991.

[54] S. A. Canann, J. R. Tristano, and M. L. Staten, “An Approach to Combined Laplacian and Optimization-Based smoothing for Triangular, Quadrilateral, and Quad-Dominant Meshes”, *Proc. 7th International Meshing Roundtable*, pp. 479–494, 1998.

[55] N. Mukherjee, "A Hybrid, Variational 3D Smoother for Orphaned Shell Meshes", *Proc. 11th International Meshing Roundtable*, pp. 379–390, 2002.

[56] P. Knupp, “Winslow Smoothing on Two Dimensional Unstructured Meshes”, *Proc. 7th International Meshing Roundtable*, pp. 449–457, 1998.

[57] S. A. Canann, M. B. Stephenson and T. D. Blacker, “Optismoothing: An Optimization-Driven Approach to Mesh Smoothing”, *Finite Elements in Analysis and Design*, vol. 13, pp. 185–190, 1993.

[58] L. Freitag and P. Knupp, “Tetrahedral Element Shape Optimization via The Jacobian Determinant and Condition Number”, *Proc. 8th International Meshing Roundtable*, pp. 247–258, 1999.

[59] L. Freitag and P. Knupp, “Tetrahedral Mesh Improvement via Optimization of The Element Condition Number”, *International Journal for Numerical Methods in Engineering*, vol. 53, pp. 1377–1391, 2002.

[60] O. P. Jacquotte and G. Coussement, “Structured Mesh Adaptation: Space Accuracy and Interpolation Methods”, *Computational Methods in Applied Mechanics and Engineering*, vol. 101, pp. 397–432, 1992.

[61] V. Parthasarathy and S. Kodiyalam, “A Constrained Optimization Approach to Finite Element Mesh Smoothing”, *Journal of Finite Elements in Analysis and Design*, vol. 9, pp. 309–320, 1991.

[62] C. Farhat, C. Degand, B. Koobus and M. Lesoinne, “Torsional springs for two-dimensional unstructured fluid meshes”, *Computer Methods in Applied Mechanics and Engineering*, vol. 163, pp. 231–245, 1998.

[63] C. Degand and C. Farhat, “A three-dimensional torsional spring analogy method for unstructured dynamic meshes”, *Computers and Structures*, vol. 80, pp. 305–316, 2002.

[64] Sandia National Labotartories, “CUBIT 10.2 User Documentation”, http://cubit.sandia.gov/help-version10.2/cubithelp.htm, Oct11, 2006 (last accessed Jul 1, 2007).

[65] National University of Singapore, “A Simple Guide To FIDAP/FIMESH - A Fluid Dynamics Analysis Package”, http://www.nus.edu.sg/comcen/svu/techinfo/fidap_userguide.html, 2006 (last accessed Jul 1, 2007).

[66] P. M. Knupp, "Algebraic mesh quality metrics for unstructured initial meshes", *Finite Elements in Analysis and Design*, vol. 39, pp. 217–241, 2003.

[67] A. Oddy, J. Goldak, M. McDill and M. Bibby, “A Distortion Metric for Isoparametric Finite Elements”, *Trans. CSME*, nr. 38-CSME-32, 1988.

[68] J. Robinson, “CRE method of element testing and the Jacobian shape parameters”, *Eng. Comput*., vol. 4, pp. 113–118, 1987.

[69] W. Kwok and Z. Chen, “A simple and effective mesh quality metric for hexahedral and wedge elements”, *Proc. 9th International Meshing Round Table*, pp. 325–333, 2000.

[70] S. Nagakura, S. Noguchi, H. Yamashita and V. Cingoski, “Automatic Hexahedral Mesh Generation for FEM Using Shape Recognition Technique and Tree Method”, *IEEE Transactions on Magnetics*, vol. 38, no. 2, pp. 417-420, March 2002.

[71] T. Maeda, S. Noguchi, H. Yamashita and V. Cingoski, “Automatic Hexahedral Mesh Generation for Rotating Machine”, *IEEE Transactions on Magnetics*, vol. 40, no. 2, pp. 973-976, March 2004.

[72] T. Maeda, S. Noguchi, H. Yamashita, and V. Cingoski, “An automatic hexahedral mesh generation method for hexahedral elements towards rotating machine”, *Journal of Materials Processing Technology*, vol. 161, no. 1-2, pp. 101–106, Apr. 2005.

[73] P. M. Knupp, "Algebraic mesh quality metrics", *SIAM J. Sci. Comput.*, vol. 23, no. 1, pp. 193-218, 2001.

[74] P. M. Knupp, "A method for hexahedral mesh shape optimization", *Int. J. Numer. Meth. Eng.*, vol. 58, pp. 319–332, 2003.

[75] W. J. Gordon and C. A. Hall, “Transfinite Element Methods: Blending Function Interpolation over Arbitrary Curved Element Domains”, *Numer. Math.*, vol. 21, pp. 109-129, 1973.

[76] T. Sederberg, “Bézier Curves”, http://www.tsplines.com/resources/, 2007 (last access Jul 1, 2007).

[77] S. Dey, R. M. O’Bara, and M. S. Shephard, “Curvilinear Mesh Generation in 3D”, *Proc. 8th International Meshing Roundtable*, 1999.

[78] X. J. Luo, M. S. Shephard, J. F. Remacle, R. M. O’Bara, M. W. Beall, and B. Szabo, “p-Version Mesh Generation Issues”, *Proc. 11th International Meshing Roundtable*, 2002.

[79] H. Grassmann and L. Kannenberg, *A New Branch of Mathematics: The “Ausdehnungslehre" of 1844 and Other Works*. Chicago: Open Court Publishing, 1995.

[80] G. A. Deschamps, “Electromagnetics and differential forms," *IEEE Proc.*, vol. 69, pp. 676-696, June 1981.

[81] W. L. Engl, “Topology and geometry of the electromagnetic field," *Radio Sci.*, vol. 19, pp. 1131-1138, Sept. Oct. 1984.

[82] D. Baldomir, “Differential forms and electromagnetism in 3-dimensional Euclidean space R^3”, *IEE Proc.*, vol. 133, pp. 139-143, May 1986.

[83] W. L. Burke, *Applied Differential Geometry*. Cambridge, UK: Cambridge University Press, 1985.

[84] J. M. Koning, “An Object Oriented, Finite Element Framework For Linear Wave Equations”, *PhD Thesis*, Engineering - Applied Science, University Of California Davis, January 2004.

[85] K. F. Warnick, P. Russer, “Two, Three and Four-Dimensional Electromagnetics Using Differential Forms”, *Turk J Elec Engin*, vol. 14, no.1, pp. 153-172, 2006.

[86] E. Tonti, "Finite Formulation of the Electromagnetic Field", *Progress in Electromagnetics Research, PIER 32*, pp. 1-44, 2001.

[87] J. C. Nédélec, "A new family of mixed finite elements in R^3", *Numer. Math.*, vol. 50, no. 1, pp. 57-81, 1986.

[88] P. A. Raviart and J. M. Thomas, "A mixed finite element method for second order elliptic problems", *In I. Galligani and E. Magenes, editors, Mathematical Aspects of the 17 Finite Element Method*, pp. 292-315, Berlin-Heilderberg-New York, 1977.

[89] F. Brezzi, J. Douglas Jr. and L. D. Marini, "Two families of mixed finite elements for second order elliptic problems", *Numer. Math.*, vol. 47, pp. 217-235, 1985.

[90] J. S. van Welij, “Calculation of eddy current in terms of H on hexahedra”, *IEEE Trans Mag*, vol. 21, pp. 2239-2241, 1985.

[91] A. Kameari, “Calculation of Transient 3D Eddy Current Using Edge Elements”, *IEEE Trans Mag*, vol. 26, pp. 466-469, 1990.

[92] A. K. Karanam, "Hierarchical Hexahedral Elements For Fluid Dynamic Simulations Using Stabilized Finite Element Methods", *M.Sc. Thesis*, Rensselaer Polytechnic Institute Dept of Mechanical Engineering, New York, 2000.

[93] J.S. Wang, “Hierarchic Edge Elements for High-Frequency Problems”, *IEEE Transactions on Magnetics*, vol. 33, no. 2, pp. 1536-1539, March 1997.

[94] B. Szabo and I. Babuška, *Finite Element Analysis*, Wiley, NY: 1991.

[95] S. Zaglmayr, “High Order Finite Element Methods for Electromagnetic Field Computation”, *Ph.D. Thesis*, Institut für Numerische Mathematik, Johannes Kepler Universität Linz, July 2006.

[96] R. Holland and J. W. Williams, “Total-field versus scattered field finite-difference codes: A comparative asessment,” *IEEE Trans. Nucl. Sci.*, vol. NS-30, pp. 4583-4588, Dec. 1983.

[97] J. P. Berenger, “A perfectly matched layer for the absorption of electromagnetic waves”, *J. Comp. Phys.*, vol. 114, pp. 185-200, Oct. 1994.

[98] W. C. Chew and W. H. Weedon, “A 3-D perfectly matched medium from modified Maxwell’s equations with stretched coordinates”, *Microwave Opt. Tech. Lett.*, pp. 599-604, Sept. 1994.

[99] Z. S. Sacks, D. M. Kingsland, R. Lee, and J. F. Lee, “A perfectly matched anisotropic absorber for use as an absorbing boundary condition”, *IEEE Trans. Antennas Propagat.*, vol.43, pp. 1460-1463, Dec. 1995.

[100] M. Kuzuoğlu and R. Mittra, “Investigation of Nonplanar Perfectly Matched Absorbers for Finite-Element Mesh Truncation”, *IEEE Trans. Antennas Propagat.*, vol 45, pp. 474-486, Mar. 1997.

[101] W. C. Chew, J. M. Jin, and E. Michielssen, "Complex coordinate stretching as a generalized absorbing boundary condition", *Microwave Opt. Technol. Lett.*, vol. 15, pp. 363–369, 1997.

[102] J. Bentley, *Programming Pearls*, MA: Addison-Wesley, 1986.

[103] J. R. Gilbert, “Sparse Matrices”, *3-Day Seminar at Sparse Matrix Days in MIT*, http://www.cs.ucsb.edu/~gilbert/talks/talks.htm, (last accessed Jul 1, 2007).

[104] M. R. Hestenes and E. Steifel, “Method of conjugate gradients for solving linear systems”, *J. Res. Natl. Bur. Stand.*, vol. 49, pp. 409-436, Dec. 1952.

[105] C. Lanczos, “Solution of systems of linear equations by minimized iterations”, *J. Res. Natl. Bur. Stand.*, vol. 49, pp. 33-53, 1952.

[106] Y. Saad, “GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems”, *SIAM J. Sci. Statist. Comput.*, vol. 7, pp. 856-869, 1986.

[107] R. W. Freund and N. M. Nachtigal, “QMR: A quasi-minimal residual method for non-Hermitian linear systems”, *Numerische Mathematik*, vol. 60, pp. 315-339, 1991.

[108] P. Sonneveld, “CGS: A fast Lanczos-type solver for nonsymmetric linear systems”, *SIAM J. Sci. Statist. Comput.*, vol. 10, pp. 36-52, 1989.

[109] H. A. van der Vorst, “BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems”, *SIAM J. Sci. Statist. Comput.*, vol. 13, pp. 631-644, 1992.

[110] R. W. Freund, “A transpose-free quasi-minimal residual algorithm for non Hermitian linear systems”, *SIAM J. Sci. Statist. Comput.*,, vol. 14, no. 2, pp. 470-482, 1993.

[111] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Philadelphia: SIAM, 1997.

[112] G. H. Golub, C.F. van Loan, *Matrix Computations*, Baltimore: Johns Hopkins University Press, 1989.

[113] J. Stoer, R. Bulirsch, *Introduction to Numerical Analysis*, New York: Springer-Verlag, 1980.

[114] L. Baker, *More C Tools for Scientists and Engineers*, New York: McGraw-Hill, 1991.

[115] R. Fletcher, “Numerical Analysis Dundee 1975”, *Lecture Notes in Mathematics*, Springer-Verlag, vol. 506, pp. 73-89, 1976.

[116] B. M. Irons, “A frontal solution program for finite-element analysis”, *Int. J. Num. Meth. Eng.*, vol. 2, pp. 5-32, 1970.

[117] P. Hood, “Frontal solution program for unsymmetric matrices”, *Int. J. Num. Meth. Eng.*, vol. 10, pp. 379-400, 1976.

[118] I. S. Duff, “MA32 – A package for solving sparse unsymmetric systems using the frontal method”, *Technical Report AERE R11009*, Her Majesty’s Stationery Office, London, 1981.

[119] J. W. H. Liu, “The role of elimination trees in sparse factorization”, *SIAM J. Matrix Analysis and Applications*, vol. 11, pp. 134-172, 1990.

[120] I. S. Duff, “Parallel implementation of multifrontal schemes”, *Parallel Computing*, vol. 3, pp. 193-204, 1986.

[121] A. Guermouche, J. Y. L'Excellent, and G. Utard, "Some memory issues in the multifrontal method", *SIAM Conf. on Parallel Processing for Scientific Computing (PP04)*, February 2004.

[122] A. Guermouche, J. Y. L'Excellent, and G. Utard, "Impact of reordering on the memory of a multifrontal solver", *Parallel Computing*, vol. 29, no. 9, pp. 1191-1218, 2003.

[123] F. Pellegrini, "SCOTCH 3.4 user's guide", *Technical Report RR 1264-01*, LaBRI, Université Bordeaux I, November 2001.

[124] F. Pellegrini, Jean Roman, and Patrick Amestoy, "Hybridizing Nested Dissection and Halo Approximate Minimum Degree for Efficient Sparse Matrix Ordering", *Proc. 11th IPPS/SPDP'99 Workshops, Held in Conjunction with the 13th International Parallel Processing Symposium and 10th Symposium on Parallel and Distributed Processing*, pp. 986-995, 1999.

[125] G. Karypis, V. Kumar, "METS - a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices-Version 4.0", University of Minnesota, September 1998.

[126] P. R. Amestoy, T. A. Davis, and I. S. Duff, "An Approximate Minimum Degree Ordering Algorithm", *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 4, pp. 886-905, Oct. 1996.

[127] CEC-ESPRIT IV Fund, "MUMPS: a MUltifrontal Massively Parallel sparse direct Solver", http://mumps.enseeiht.fr/index.html, May 3, 2007 (last accessed Jul 1, 2007).

[128] K. Sertel, "Multilevel Fast Multipole Method For Modeling Permeable Structures Using Conformal Finite Elements", *Ph. D. Thesis*, University of Michigan, 2003.

[129] Y. Xiao and Y. Lu, "The Prolate and Oblate Spheroid Perfectly Matched Layers", *IEEE Transactions on Magnetics*, vol. 38, no. 2, pp. 669-672, Mar 2002.

[130] X. C. Nie, L. W. Li, N. Yuan, T. S. Yeo, and Y. B. Gan, "A Fast Analysis of Electromagnetic Scattering by Arbitrarily Shaped Homogeneous Dielectric Objects", *Microwave and Optical Technology Letters*, vol. 38, no. 1, pp. 30-35, July 2003.

[131] J. L. Volakis, A. Alexanian, and J. M. Jin, "Broadband RCS Reduction of Rectangular Patch by Using Distributed Loading," *IEEE Elect Letters*, vol. 28, no. 25, pp. 2322-2323, Dec. 1992.

[132] J. L. Volakis, A. Alexanian, J. M. Jin, and C. L. Yu, "Radar Cross Section Analysis and Control of Microstrip Patch Antennas," Draft paper, *IEEE*, 1992.

[133] J. M. Jin and J. L. Volakis, "A Hybrid Finite Element Method for Scattering and Radiation by Microstrip Patch Antennas ad Arrays Residing in a Cavity," *IEEE Trans. Antennas Propagat*. vol. 39, no. 11, pp. 1598-1604, Nov. 1991.

[134] N. Yuan, T.S. Yeo, X.C. Nie, L.W. Li, and Y.B. Gan, Efficient analysis of electromagnetic scattering and radiation from patches on finite, arbitrarily curved, grounded substrates, *Radio Science*, vol. 39, no.3, Art. No. RS3003, May 11 2004.

[135] J. Shin, A. W. Glisson, and A. A. Kishk, "Analysis of combined conducting and dielectric structures of arbitrary shapes using an E-PMCHW integral equation formulation", *Proc IEEE Antennas and Propagation Society International Symposium 2000*, pp. 2282–2285, IEEE Antennas and Propag. Soc., New York.

[136] Hsiang-Jui Kung, "Quantitative Method to Determine Software Maintenance Life Cycle", *Proc. 20th IEEE International Conference on Software Maintenance (ICSM'04)*, 2004.

[137] Argonne National Laboratory – Mathematics and Computer Science Division, "PETSc: Portable, Extensible Toolkit for Scientific Computation", http://www-unix.mcs.anl.gov/petsc/petsc-as/index.html, May 23, 2007 (last accessed on Jul 1, 2007).

[138] Department of Energy – National Science Foundation, "BLAS (Basic Linear Algebra Subprograms)", http://www.netlib.org/blas/index.html, Jul 25, 2005 (last accessed on Jul 1, 2007).

[139] Department of Energy – National Science Foundation, "LAPACK – Linear Algebra Package", http://www.netlib.org/lapack/index.html, Feb 26, 2007 (last accessed on Jul 1, 2007).

[140] Indiana University Computer Science Department - Open Systems Laboratory, “oonumerics: Scientific Computing in Object Oriented Languages”, http://www.oonumerics.org/, Jul 8, 2005 (last accessed on Jul 1, 2007).

[141] B. Stroustrup, *The C++ Programming Language, third edition*, Addison-Wesley Publications, Massachusetts 1997.

[142] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, PrenticeHall, Englewood Cliffs, New Jersey. 1978.

[143] N. M. Josuttis, *The C++ Standard Library: A Tutorial and Reference*, Addison-Wesley Publications, Massachusetts 1999.

[144] U. Breymann, *Designing Components with the C++ STL, revised edition*, Addison-Wesley Publications, Massachusetts 2002.

[145] J. T. Cross, I. Masters and R. W. Lewis, “Why you should consider object-oriented programming techniques for finite element methods”, *International Journal of Numerical Methods for Heat & Fluid Flow*, vol: 9, no. 3, pp. 333 – 347, May 1999.

[146] B. Henz and D. Shires, “Parallel Finite Element Software Development and Performance Analysis in an Object-Oriented Programming Framework”, *Journal of Mathematical Modelling and Algorithms*, vol. 4, no.1, pp. 17-34 (18), March 2005.

[147] J. Mackerle, “Object-oriented techniques in FEM and BEM a bibliography (1996-1999)”, *Finite Elements in Analysis and Design*, vol. 36, pp. 189-196, 2000.

[148] E. J. Silva, R. C. Mesquita, R. R. Saldanha and P. F. M. Palmeira, “An object-oriented finite-element program for electromagnetic field computation”, *IEEE Transactions on Magnetics*, vol. 30, no. 5(2), pp. 3618 – 3621, Sep 1994.

[149] E. J. Silva and R. C. Mesquita, “Data management in finite element analysis programs using object-oriented techniques”, *IEEE Transactions on Magnetics*, vol. 32, no. 3(1), pp. 1445 - 1448, May 1996.

[150] J. Kangas, T. Tarhasaari and L. Kettunen, “Maxwell equations and finite element software systems: object-oriented coding needs well defined objects”, *IEEE Transactions on Magnetics*, vol. 36, no. 4(1), pp. 1645 - 1648, July 2000.

[151] W. Mai and G. Henneberger, “Object-oriented design of finite element calculations with respect to coupled problems”, *IEEE Transactions on Magnetics*, vol. 36, no. 4(1), pp. 1677-1681, July 2000.

[152] H. M. Chen and G. C. Archer, “A Distributed Object-Oriented Finite-Element Analysis Program Architecture”, *Computer-Aided Civil and Infrastructure Engineering*, vol. 16, no. 5, pp. 326, September 2001.

[153] B. Patzák, “OOFEM: Free Object Oriented Finite Element Code”, http://www.oofem.org/, Apr 19, 2007 (last accessed on Jul 1, 2007).

[154] B. Patzák and Z. Bittnar, “OOFEM: An object oriented framework for finite element analysis”, *Acta Polytechnica*, vol. 44, no. 5-6, pp. 54--60, 2004.

[155] Lawrence Livermore National Laboratory, “FEMSTER Main Page”, http://www-eng.llnl.gov/emsolve/DOC/html/index.html, March 2, 2006 (last accessed on Jul 1, 2007).

[156] P. Castillo, R. Rieben and D. White, "FEMSTER: An Object Oriented Class Library of High-Order Discrete Differential Forms", *ACM Transactions on Mathematical Software*, vol. 31, no. 4, pp 425-457, 2006.

[157] W. Bangerth, G. Kanschat and R. Hartmann, “deal.II: A Finite Element Differential Equations Analysis Library”, http://www.dealii.org/, Jun 30, 2007 (last accessed on Jul 1, 2007).

[158] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, *Design Patterns: Elements of Reusable Object Oriented Software*, Addison-Wesley Publications, Massachusetts 1998.

[159] Trolltech, “QT – Code Less, Create More”, http://www.trolltech.com/products/qt, Jun 26, 2007 (last accessed on Jul 1, 2007).

[160] SGI, “OpenGL – The Industry Standard for High Performance Graphics”, http://www.opengl.org/, Jun 28, 2007 (last accessed on Jul 1, 2007).

[161] J. Neider and T. Davis, *Redbook: OpenGL Programming Guide*, Reading: Addison-Wesley Publishing Company, 1997.

[162] OASIS International Standards Consortium, “Applying XML and Web Services Standards in Industry”, http://www.xml.org/, Jun 28, 2007 (last accessed on Jul 1, 2007).

[163] D. C. Schmidt, “The Adaptive Communication Environment (ACE)”, http://www.cs.wustl.edu/~schmidt/ACE.html, Jun 25, 2007 (last accessed on Jul 1, 2007).

[164] Object Management Group, “CORBA (Common Object Request Broker Architecture) Specifications”, http://www.omg.org/technology/documents/corba_spec_catalog.htm, April 3, 2007 (last accessed on Jul 1, 2007).

[165] Object Management Group, “Catalog of OMG Modeling and Metadata Specifications”, http://www.omg.org/technology/documents/modeling_spec_catalog.htm, April 3, 2007 (last accessed on Jul 1, 2007).

[166] N. Calvo and S. Idelsohn, “All-hexahedral mesh smoothing with a node-based measure of quality”, *International Journal for Numerical Methods in Engineering*, vol. 50, no. 8, pp. 1957–1967, 2001.

[167] P. Knupp, “Hexahedral and tetrahedral mesh untangling”, *Engineering with Computers*, vol. 17, no. 3, pp. 261–268, 2001.

[168] R. C. Eberhart and Y. Shi, “Evolving artificial neural networks”, *Proceedings of 1998 International Congress on Neural Networks and Brain*, Beijing, P.R.C. 1998.

[169] J. Kennedy amd W. M. Spears, “Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on multi modal problem generator”, *Proceedings of IEEE International Congress on Evolutionary Computation*, 1998.

[170] J. Kennedy and R. C. Eberhart, “Particle swarm optimization”, *Proceedings of IEEE Congress on Neural Networks IV*, 1995.

[171] J. Robinson and Y. Rahmat-Samii, “Particle swarm optimization in electromagnetics”, *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 2, pp. 397-407, 2004.

[172] C. Yang and D. Simon, “A new particle swarm optimization technique”, *Proceedings of 18th International Conference on Systems Engineering*, pp. 164-169, 2005.

[173] C. A. Coello Coello and M. S. Lechunga, “MOPSO: A proposal for multiple objective particle swarm optimization”, *Proceedings of the Congress on Evolutionary Computation*, pp. 1051-1056, 2002.

[174] M. Reyes-Sierra and C. A. Coello Coello, "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art", *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287-308, 2006.

[175] M. Clerc and J. Kennedy, “The Particle Swarm – Explosion, Stability, and Convergence”, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp.58-73, Feb 2002.

[176] M. A. M. de Oca Roldán, On the Performance of Particle Swarm Optimizers, *M. Sc. Thesis*, Université Libre de Bruxelles Faculté des Sciences Appliquées, IRIDIA - Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle, 2006.

APPENDIX A

SCIENTIFIC CONTRIBUTIONS

A.1. Great Contributors

Here is a chronological list of scientists who could not be explicitly cited, but mentioned throughout the flow of thesis due to their contributions:

- Euclid (a.k.a. Euclid of Alexandria) (c.325-c.265BC), *Greek mathematician of Hellenistic Egypt*, "*Father of Geometry*" and "*Uncle of Number Theory*", mentioned via *Euclidean spaces*.
- René Descartes (1596-1650), also known as “Cartesius”, *French philosopher, mathematician, and scientist; "Father of Modern Mathematics”* , mentioned via *Cartesian coordinates*.
- Christiaan Huygens (1629-1695), *Dutch mathematician and physicist*, mentioned via *Huygens equivalence principle*.
- Sir Isaac Newton, (1643-1727), *English physicist, mathematician, astronomer, alchemist, and natural philosopher, regarded by many as "the greatest figure in the history of science"*, mentioned via *Newton-Raphson method*.
- Joseph Raphson (ca. 1648-ca. 1715), *English mathematician*, mentioned via *Newton-Raphson method.*
- Joseph-Louis Lagrange (1736-1813), *Italian-French mathematician-physicist*, mentioned via *Lagrange polynomials*.
- Pierre-Simon, Marquis de Laplace (1749-1827), *French mathematician and astronomer*, mentioned via *Laplacian smoothing*.

- Adrien-Marie Legendre (1752-1833), *French mathematician*, mentioned via *Legendre polynomials*.
- André-Marie Ampère (1775-1836), *French physicist*, mentioned via *Ampère's Law*.
- Karl Friedrich Gauss (Gauß) (1777-1855), *German mathematician and physicist*, mentioned via *Gauss quadrature* and *Gauss eliminitation*.
- Michael Faraday (1791-1867), *English chemist-physicist*, mentioned via *Faraday's Law*.
- Benjamin Olinde Rodrigues (1795-1851), *French mathematician and social reformer*, mentioned via *Rodrigues rotation formula*.
- Karl Gustav Jacob Jacobi (1804-1851), *German mathematician*, mentioned via *Jacobi determinant* and *Jacobi polynomials*.
- Charles Hermite (1822-1901), *French mathematician*, mentioned via *Hermite polynomials*.
- Pafnuty Lvovich Chebyshev (also Romanized in various ways, e. g. as Chebychev, Chebyshov, Tchebycheff or Tschebyscheff in French and German transcriptions) (1821-1894), *Russian mathematician*, mentioned via *Chebyshev polynomials*.
- James Clerk Maxwell (1831-1879), *Scottish mathematical physicist*, mentioned via *Maxwell equations*.
- John William Strutt, 3rd Baron Rayleigh (1842-1919), *English physicist and chemist*, mentioned via *Rayleigh's criterion*.
- Vilfredo Federico Damaso Pareto (1848-1923), *Italian sociologist, economist and philosopher*, mentioned via *Pareto optimality*.
- Leopold Gegenbauer (1849-1903), *Austrian mathematician*, mentioned via *Gegenbauer polynomials*.
- Ferdinand Georg Frobenius (1849-1917), *German mathematician*, mentioned via *Frobenius norm*.
- David Hilbert (1862-1943), *German mathematician, the most influential mathematician of the 20th century*, mentioned via *Hilbert spaces*.
- Georgy Voronoï (1868-1908), *Russian mathematician of Ukrainian descent*, mentioned via *Voronoï graph/diagram*.
- Gustav Mie (1869-1957), *German physicist*, mentioned via *Mie series*.

- André-Louis Cholesky (1875-1918), *French mathematician*, mentioned via *Cholesky decomposition*.
- Sergei Natanovich Bernstein (sometimes Romanized as Bernshtein) (1880-1968), *Ukrainian mathematician*, mentioned via *Bernstein polynomials*.
- Boris Nikolaevich Delaunay (1890-1980), *Soviet/Russian mathematician*, mentioned via *Delaunay tesselation/triangulation*.
- Georges de Rham (1903-1990), *Swiss mathematician*, mentioned via *de Rham complex*.
- Sergei L'vovich Sobolev (1908-1989), *Russian mathematician*, mentioned via *Sobolev spaces*.
- Pierre Étienne Bézier (1910-1999), *French engineer*, mentioned via *Bézier curves* and *Bézier surfaces*.

A.2. Historical Milestones in the Finite Element Theory

1844 – Differential Forms by Grassman

1900s – Definition of Exterior Calculus of Differential Forms by Cartan

1943 – Piecewise Approximations by Courant

1954 – Domain Partitioning, Assembly and Boundary Conditions (i.e. basic FEM) by Argyris

1970s – Application of the Finite Element Method to Electromagnetics by Numerous Researchers

1985 – Intuitive Definition of the Linear Hexahedral Edge Element by van Welij

1986 – Introduction of the Edge Element Concept by Nédélec

1990 – Intuitive Definition of the Quadratic Hexahedral Edge Element by Kameari

1991 – Hierarchical Finite Element Concept by Szabo and Babuška

1994 – Definition of the Perfectly Matched Layers by Berenger

1996 – Proof of Topological Existence of Hexahedral Mesh by Thurston and Mitchell

1997 – Methodological Construction of Hierarchical Hexahedral Edge Elements by Wang

2006 – Methodological Construction of Hierarchical Hexahedral Edge and Facet Elements by Zaglmayr

APPENDIX B

EXPLICIT BASIS FUNCTIONS AND INTERPOLATION PROPERTIES OF VAN WELIJ AND KAMEARI ELEMENTS

In this section, the basis functions and the interpolation properties of van Welij and Kameari elements are given.

B.1. The Hexahedral Edge Element Shape Functions

The hexahedral edge element has six faces, eight nodes and twelve edges as shown in Fig. B.1.

A vector quantity is given by

$$\mathbf{A} = \sum_{i=1}^{12} \mathbf{w}_i(\mathbf{r}) A_i \tag{B.1}$$

where $\mathbf{w}_i$ is the shape function related to edge i and **A** is, for instance, the magnetic vector potential or any other vector field. A_i represents the projection of **A** along edge i.

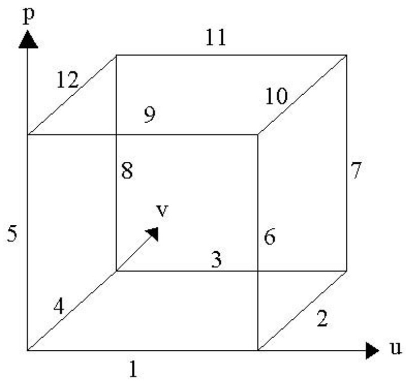


Fig. B.1. The hexahedral edge element with its edges numbered.

The shape function is described by the following product

$$\mathbf{w}_i(\mathbf{r}) = \phi_i(u, v, p)\mathbf{q}_i(\mathbf{r}) \tag{B.2}$$

where

$$\mathbf{r} = \mathbf{i}x + \mathbf{j}y + \mathbf{k}z \tag{B.3}$$

which represents the position vector of a generic point M (x, y, z). The function $\phi_i(u, v, p)$ depends on the reference coordinates u, v, p and it is the placement function of the edge element; it is given in reference coordinates since the numerical integration and other algebraic operations are performed in this coordinate system. The function $\mathbf{q}_i(\mathbf{r})$, which is responsible for the direction of the edge, is given in global coordinates, since only these coordinates take into account the actual geometry of the elements. From now on, instead of $\mathbf{w}_i(\mathbf{r})$, $\mathbf{q}_i(\mathbf{r})$, $\phi_i(u, v, p)$, a simplified notation, namely $\mathbf{w}_i$, $\mathbf{q}_i$, ϕ_i will be used. Table B.1. shows the position functions ϕ for all edges.

The vector quantity $\mathbf{q}_i$ depends on the direction of the edge. The direction is ∇a for an edge parallel to a direction where a is either u, v or p. To obtain ∇u, ∇v and ∇p we define the following vectors:

$$
\begin{aligned}
\mathbf{v}_u &= \frac{\partial \mathbf{r}}{\partial u} = \frac{\partial}{\partial u}(\mathbf{i}x + \mathbf{j}y + \mathbf{k}z) \\
\mathbf{v}_v &= \frac{\partial \mathbf{r}}{\partial v} = \frac{\partial}{\partial v}(\mathbf{i}x + \mathbf{j}y + \mathbf{k}z) \\
\mathbf{v}_p &= \frac{\partial \mathbf{r}}{\partial p} = \frac{\partial}{\partial p}(\mathbf{i}x + \mathbf{j}y + \mathbf{k}z)
\end{aligned}
\tag{B.4}
$$

Table B.1. The position functions ϕ.

EDGE NUMBER	ϕ_{EDGE_NO}
1	$(1-v)(1-p)$
2	$(1-p)$
3	$v(1-p)$
4	$(1-u)(1-p)$
5	$(1-u)(1-v)$
6	$u(1-v)$
7	uv
8	$(1-u)v$
9	$(1-v)p$
10	up
11	vp
12	$(1-u)p$

As an example, $\mathbf{v}_u$ can be written as

$$
\mathbf{v}_u = \begin{bmatrix} \dfrac{\partial x}{\partial u} \\ \dfrac{\partial y}{\partial u} \\ \dfrac{\partial y}{\partial u} \end{bmatrix}
\tag{B.5}
$$

Using this notation, the three vectors give

$$\begin{bmatrix}\mathbf{v}_u & \mathbf{v}_v & \mathbf{v}_p\end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} & \frac{\partial x}{\partial p} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} & \frac{\partial y}{\partial p} \\ \frac{\partial z}{\partial u} & \frac{\partial z}{\partial v} & \frac{\partial z}{\partial p} \end{bmatrix} \tag{B.6}$$

Transposing the matrix, we obtain a matrix $[J_1]$,

$$[J_1] = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial y}{\partial u} & \frac{\partial z}{\partial u} \\ \frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} & \frac{\partial z}{\partial v} \\ \frac{\partial x}{\partial p} & \frac{\partial y}{\partial p} & \frac{\partial z}{\partial p} \end{bmatrix} \tag{B.7}$$

This is a Jacobian matrix which expresses a vector in the (x, y, z) system in terms of the (u, v, p) system. Now, let us calculate the vectors $\mathbf{v}_u$, $\mathbf{v}_v$, and $\mathbf{v}_p$ in terms of the coordinates of the nodes of the elements. Fig. B.2 shows the hexahedral element with its nodes numbered.

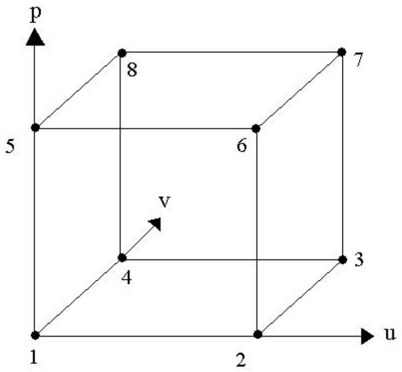


Fig. B.2. The Hexahedral Edge Element with its Nodes Numbered.

To achieve the calculation, we use the nodal shape functions in order to define the coordinates x, y, and z as functions of u, v, and p.

$$\mathbf{r} = \mathbf{i}x + \mathbf{j}y + \mathbf{k}z = \sum_{j=1}^{8} N_j(u,v,p)\mathbf{r}_j = \mathbf{i}\sum_{j=1}^{8} N_j x_j + \mathbf{j}\sum_{j=1}^{8} N_j y_j + \mathbf{k}\sum_{j=1}^{8} N_j z_j \tag{B.8}$$

Recalling that

$$\mathbf{v}_u = \frac{\partial}{\partial u}(\mathbf{i}x + \mathbf{j}y + \mathbf{k}z) \tag{B.9}$$

we can get the simplified notation

$$\mathbf{v}_u = \mathbf{i}\frac{\partial N}{\partial u}x + \mathbf{j}\frac{\partial N}{\partial u}y + \mathbf{k}\frac{\partial N}{\partial u}z \tag{B.10}$$

where $(\partial N/\partial u)x$ is given as

$$\begin{bmatrix} \frac{\partial N_1}{\partial u} & \frac{\partial N_2}{\partial u} & \cdots & \frac{\partial N_8}{\partial u} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_8 \end{bmatrix} \tag{B.11}$$

The terms $(\partial N/\partial u)y$ and $(\partial N/\partial u)z$ are obtained in a similar manner. The nodal shape functions are given in Table B.2.

Table B.2. The nodal shape functions.

NODE	N
1	$a_2\, b_2\, c_2 / 8$
2	$a_1\, b_2\, c_2 / 8$
3	$a_1\, b_1\, c_2 / 8$
4	$a_2\, b_1\, c_2 / 8$
5	$a_2\, b_2\, c_1 / 8$
6	$a_1\, b_2\, c_1 / 8$
7	$a_1\, b_1\, c_1 / 8$
8	$a_2\, b_1\, c_1 / 8$

The parameters appearing in Table B.2 are defined as

$$\begin{aligned} a_1 &= (1+u), \quad a_2 = (1-u), \\ b_1 &= (1+v), \quad b_2 = (1-v), \\ c_1 &= (1+p), \quad c_2 = (1-p). \end{aligned} \tag{B.12}$$

On the other hand, we have

$$\begin{aligned} \nabla u &= \mathbf{i}\frac{\partial u}{\partial x} + \mathbf{j}\frac{\partial u}{\partial y} + \mathbf{k}\frac{\partial u}{\partial z}, \\ \nabla v &= \mathbf{i}\frac{\partial v}{\partial x} + \mathbf{j}\frac{\partial v}{\partial y} + \mathbf{k}\frac{\partial v}{\partial z}, \\ \nabla p &= \mathbf{i}\frac{\partial p}{\partial x} + \mathbf{j}\frac{\partial p}{\partial y} + \mathbf{k}\frac{\partial p}{\partial z}. \end{aligned} \tag{B.13}$$

From these equations we can write that

$$\nabla u = \frac{\mathbf{v}_v \times \mathbf{v}_p}{vol}, \quad \nabla v = \frac{\mathbf{v}_p \times \mathbf{v}_u}{vol}, \quad \nabla p = \frac{\mathbf{v}_u \times \mathbf{v}_v}{vol} \tag{B.14}$$

where *vol* is the volume of the element. Hence, the twelve vector shape functions can be written as

$$\begin{aligned}
\mathbf{w}_i &= \phi_i \nabla u, && \text{for} \quad i = 1,3,9,11,\\
\mathbf{w}_i &= \phi_i \nabla v, && \text{for} \quad i = 2,4,10,12,\\
\mathbf{w}_i &= \phi_i \nabla p, && \text{for} \quad i = 5,6,7,8.
\end{aligned} \tag{B.15}$$

In working with Maxwell's equations, we often need to evaluate the curl of the vector shape functions. Writing the curl operator explicitly we can obtain

$$\begin{aligned}
\nabla \times \mathbf{w}_i &= \frac{1}{vol}\left(\frac{\partial \phi_i}{\partial p}\mathbf{v}_v - \frac{\partial \phi_i}{\partial v}\mathbf{v}_p\right) && \text{for} \quad i = 1,3,9,11,\\
\nabla \times \mathbf{w}_i &= \frac{1}{vol}\left(\frac{\partial \phi_i}{\partial u}\mathbf{v}_p - \frac{\partial \phi_i}{\partial p}\mathbf{v}_u\right) && \text{for} \quad i = 2,4,10,12,\\
\nabla \times \mathbf{w}_i &= \frac{1}{vol}\left(\frac{\partial \phi_i}{\partial v}\mathbf{v}_u - \frac{\partial \phi_i}{\partial u}\mathbf{v}_v\right) && \text{for} \quad i = 5,6,7,8.
\end{aligned} \tag{B.16}$$

B.2. Interpolation Properties of the Linear Hexahedral Edge Elements

Consider the linear hexahedral element, whose nodes and edges are numbered as shown in Fig. B.3:

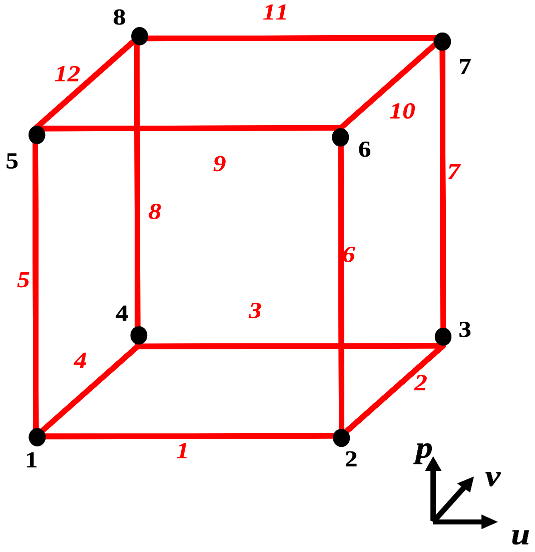


Fig. B.3. Linear hexahedral edge element extending from (-1, -1, -1) to (1, 1, 1) in *uvp* space.

The effect of the first edge is shown in Fig. B.4.

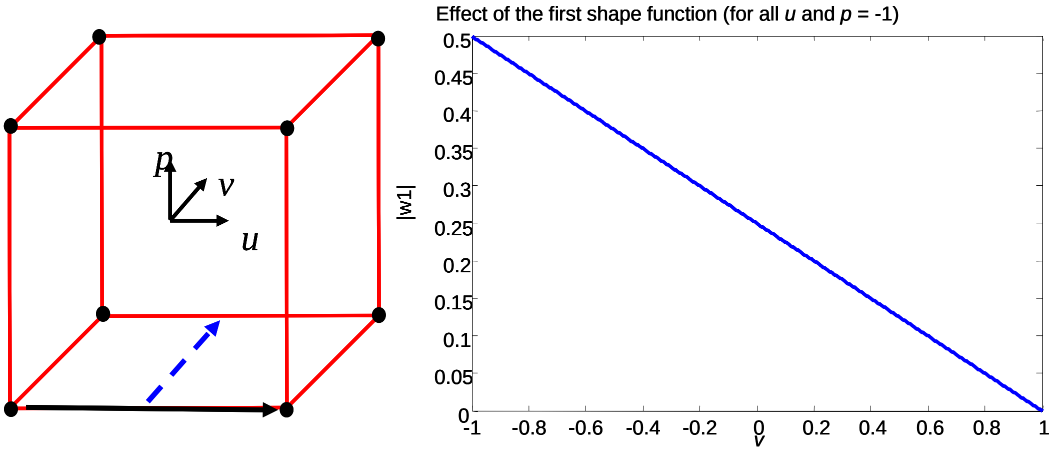


Fig. B.4. Effect of the first shape function for the linear hexahedral edge element.

The effect of the second edge is shown in Fig. B.5.

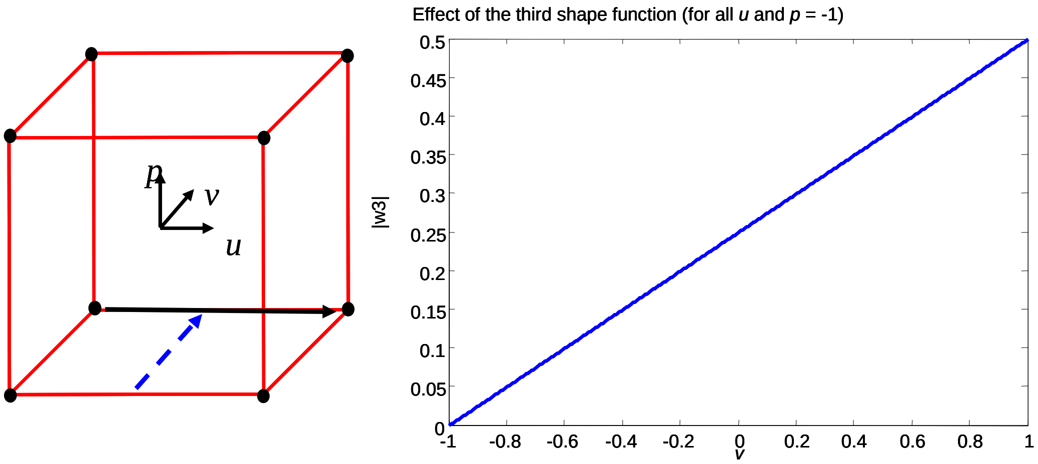


Fig. B.5. Effect of the third shape function for the linear hexahedral edge element.

The effect of the summation of the first and the third functions is shown in Fig. B.6.

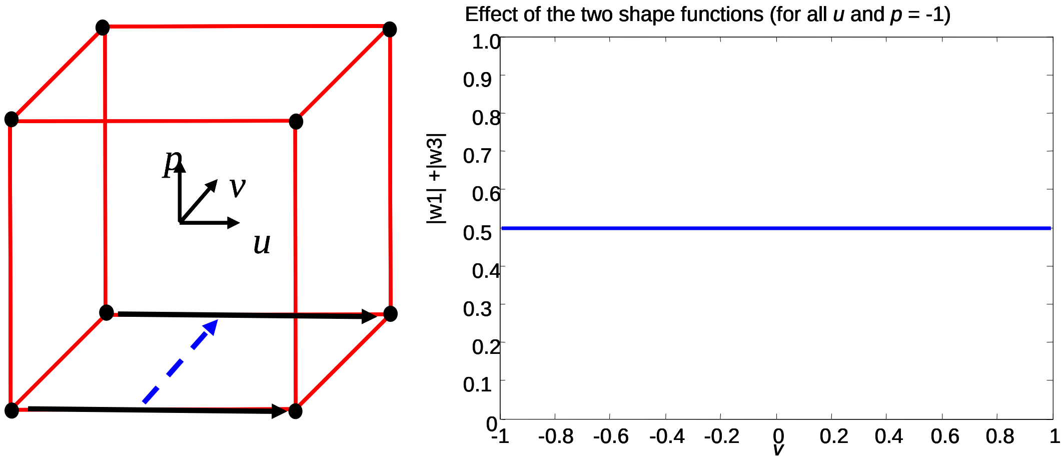


Fig. B.6. Effect of the summation of the first and the third shape functions for the linear hexahedral edge element.

As a conclusion, the effect of the shape functions is linear as seen in the figures above.

B.3. Interpolation Properties of the Quadratic Hexahedral Edge Elements

Consider the quadratic hexahedral element, whose nodes and edges are numbered as shown in Fig. B.7:

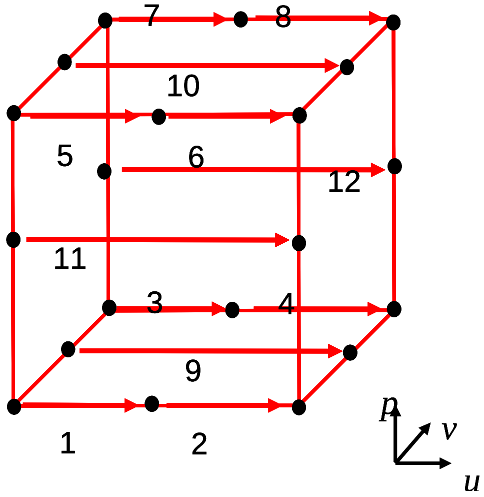


Fig. B.7. Quadratic hexahedral edge element extending from (-1, -1, -1) to (1, 1, 1) in *uvp* space.

The disjoint effects and the effect of the summation of the first and second two edges are shown in Fig. B.8.

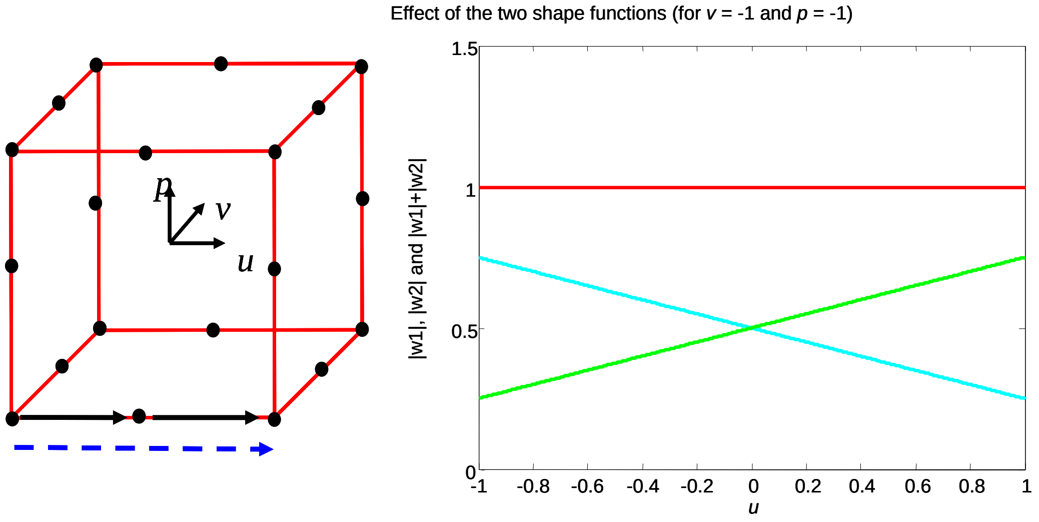


Fig. B.8. The disjoint effects and the effect of the summation of the first and second two edges along the *u* direction.

The effects of the first two edges along the *v* and *p* directions are shown in Fig. B.9 and Fig. B.10, respectively.

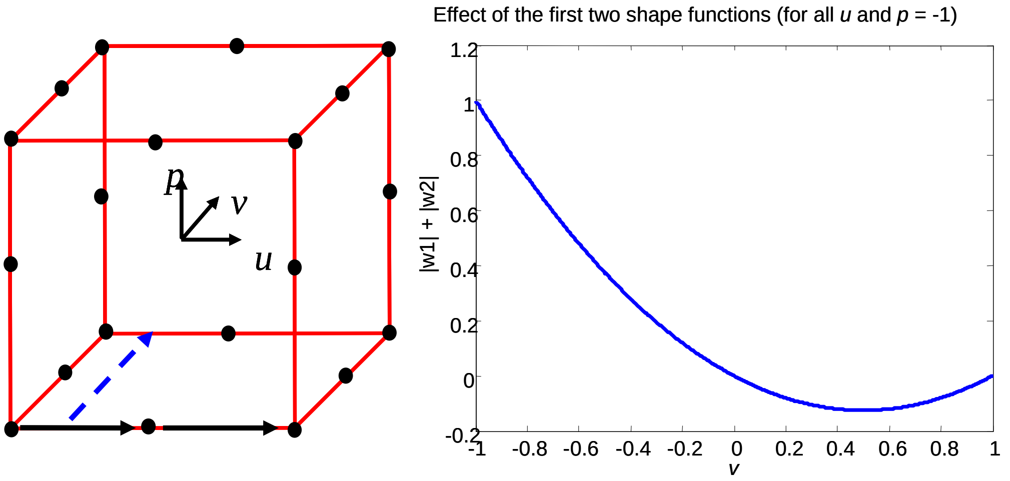


Fig. B.9. The disjoint effects and the effect of the summation of the first and second two edges along the *v* direction.

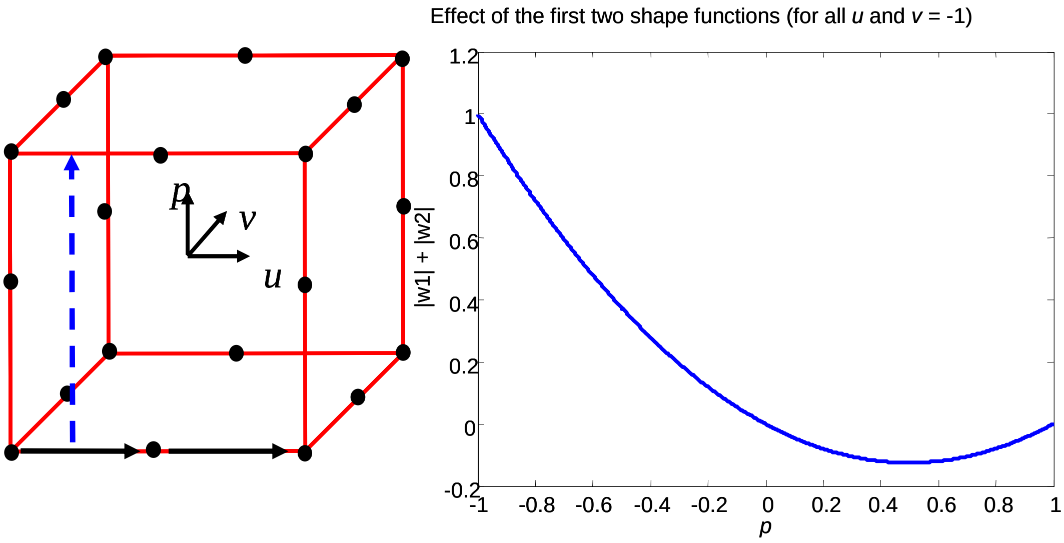


Fig. B.0.10. The disjoint effects and the effect of the summation of the first and second two edges along the *p* direction.

The effects of the ninth edge along *v* and *p* directions are shown in Fig. B.11 and Fig. B.12, respectively.

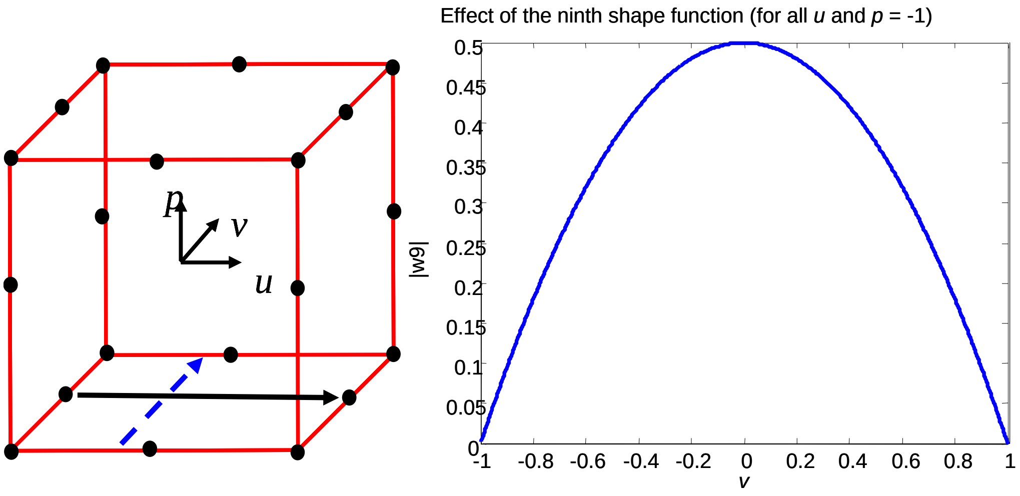


Fig. B.11. The effect of the ninth edge along the *v* direction.

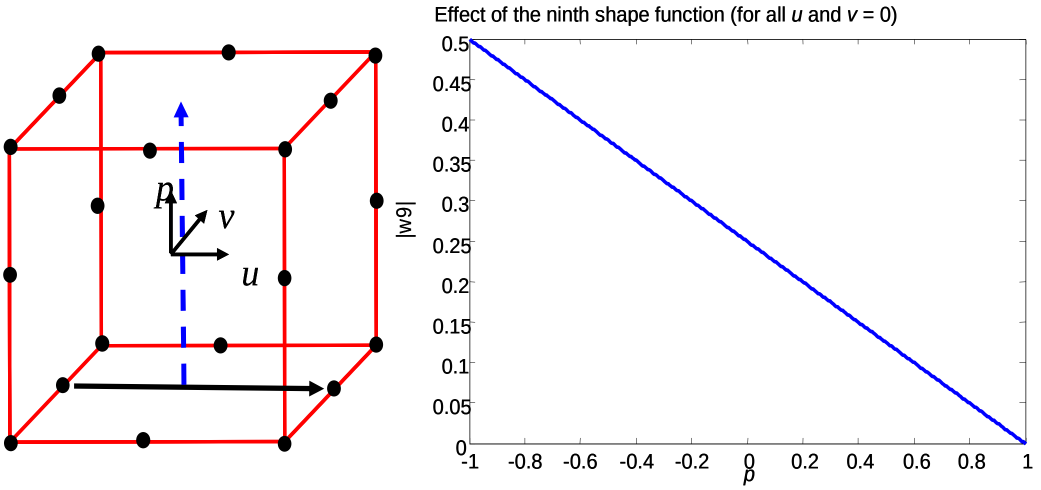


Fig. B.12. The effect of the ninth edge along the *p* direction.

The effect of the summation of four shape functions (the first, the second, the third and the fourth) is shown in Fig. B.13.

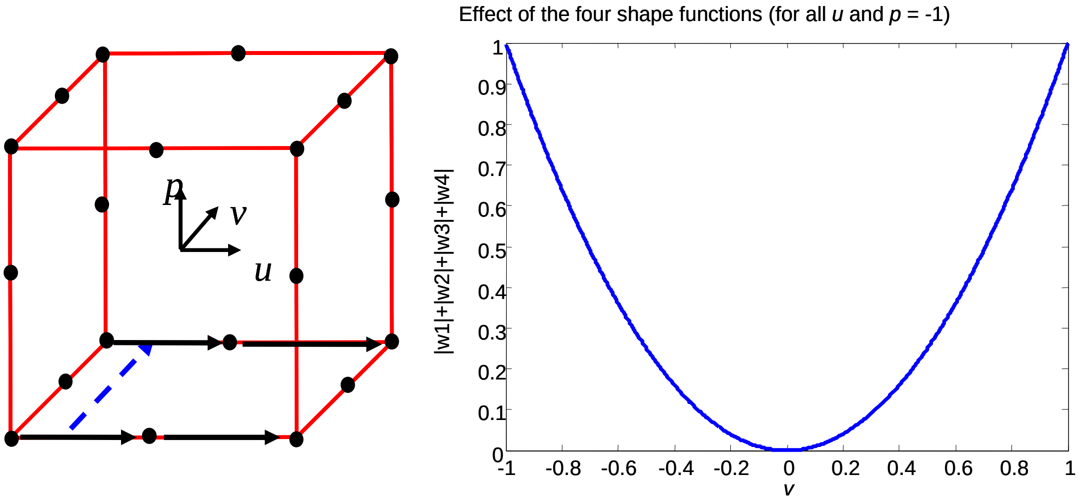


Fig. B.13. The effect of the summation of four edges.

The effect of the summation of five shape functions (the first, the second, the third and the fourth and the ninth) is shown in Fig. B.14. Here, it should be noted that since the circulation (i.e. the projection times the length of the edge) of the electric field is represented on each edge, the effect of the ninth edge is twice the others.

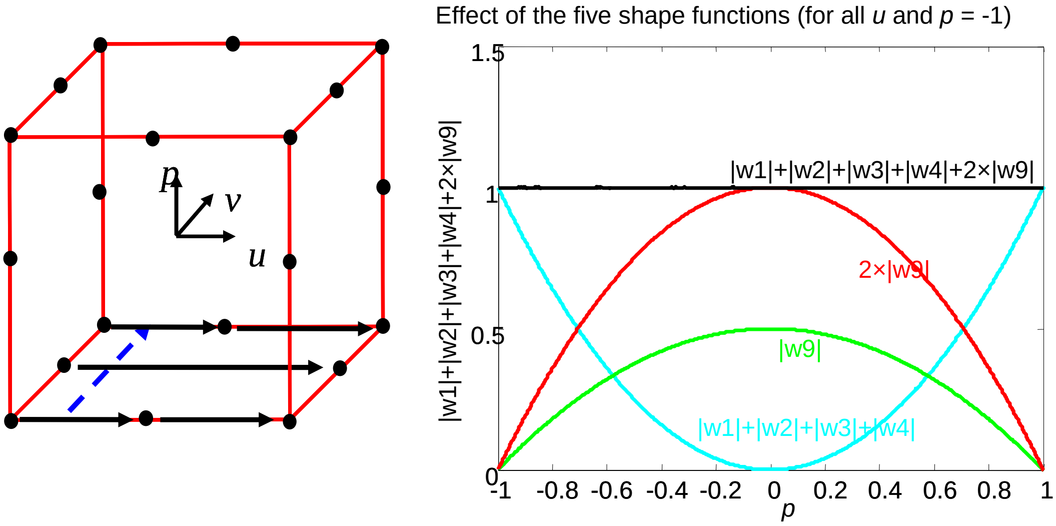


Fig. B.14. The effect of the summation of the five edges.

Consider that we want to evaluate the function $f(v) = 3v^2 + 4v + 2$ by means of the shape functions at $p = -1$ plane inside the element.

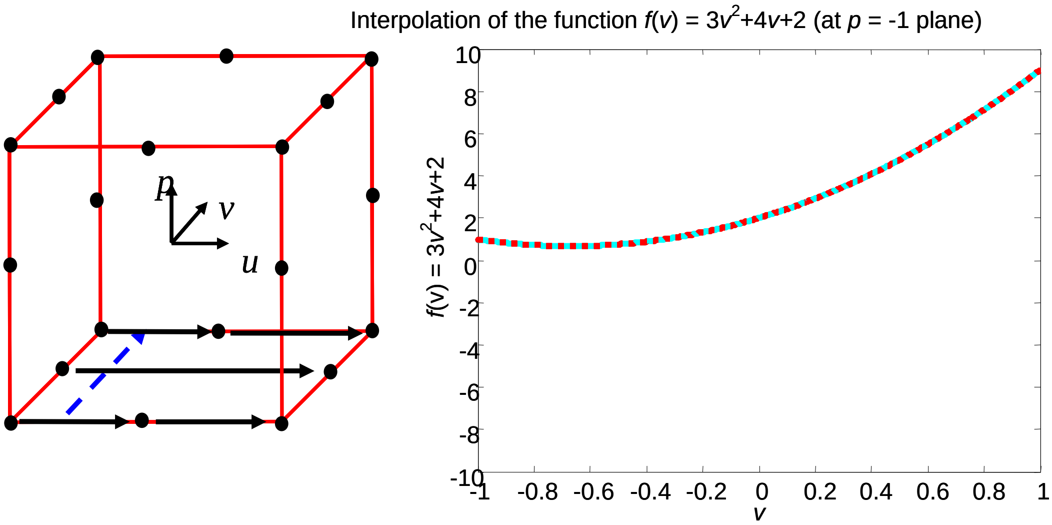


Fig. B.15. Interpolation of the function $f(v) = 3v^2 + 4v + 2$ at $p = -1$ plane by means of the quadratic shape functions.

As seen in Fig. B.15, the quadratic shape functions are sufficient to represent the function, which is also quadratic. This yields the conclusion that quadratic hexahedral edge elements are better while representing quadratic functions, and they give more accurate results than the linear hexahedral edge elements.

In electromagnetic wave propagation applications, the field variations are compared on the basis of time-harmonic uniform plane waves due to the fact that an arbitrary propagating field can be represented by means of a plane wave spectrum. For linear elements, these functions are to be represented by means of linear interpolation. On the other hand, for quadratic elements, the representation is performed via quadratic interpolation.

For this comparison, an ideal hexahedral element, namely a cube denoted as ε, is taken as a test element for investigation of the interpolation properties. The function, which will be approximated inside this element, is taken as a uniform plane wave $\mathbf{A}=A\exp[-j(kx+\phi_i)]\hat{\mathbf{a}}_z$, where $\phi_i=(2\pi i/360k)$ and $i=0,1,\ldots,359$. By this procedure, the effect of phase variation on interpolation accuracy is included in the analysis.

For a fixed phase angle of ϕ_i, the exact value of the function is denoted as $\mathbf{A}_{exact\_i}$, and its approximated form (as the linear combination of basis functions in z-direction like in equation (B.1)) is denoted as $\mathbf{A}_{app\_i}$. Two error norms are defined as

$$e_\infty = \max_{i=0,\ldots,359}\left[\frac{\max\limits_{\varepsilon}\left\|\mathbf{A}_{exact\_i}-\mathbf{A}_{app\_i}\right\|}{\max\limits_{\varepsilon}\left\|\mathbf{A}_{exact\_i}\right\|}\right] \tag{B.17}$$

and

$$e_1 = \max_{i=0,\ldots,359}\left[\frac{\int_{\varepsilon}\left\|\mathbf{A}_{exact\_i}-\mathbf{A}_{app\_i}\right\|}{\int_{\varepsilon}\left\|\mathbf{A}_{exact\_i}\right\|}\right] \tag{B.18}$$

where e_∞ stands for the maximum normalized interpolation error inside the element maximized over all investigated phase angles; and e_1 stands for the average normalized interpolation error throughout the element, maximized over all investigated phase angles.

A qualitative assessment about these definitions can be as follows: e_∞ is a metric of the point-wise accuracy, whereas e_1 is a metric of overall accuracy inside an element ε. Overall accuracy inside an element is an important and dominant Fig. of merit especially in the computation of the surface and volume integrals during FEM analysis.

For a λ/10-size linear hexahedral element, by following the analysis given above, it is calculated that e_∞ is 0.0490; and e_1 is about 0.0331.

For the interpolation of a uniform plane wave by means of quadratic elements, in order to have a reasonable accuracy level, it can be intuitively claimed that the upper limit for the element size should be about 0.5λ. The same procedure is repeated for quadratic hexahedral elements having different sizes. The results of this analysis are summarized in Table B.3.

Table B.3. Error norms for linear and quadratic elements of different sizes.

Element Type	**Element Size (λ)**	e_∞	e_1
Linear	0.1	0.049	0.0331
Quadratic	0.33	0.055	0.0077
Quadratic	0.4	0.095	0.0170
Quadratic	0.5	0.151	0.0472

In this analysis, a specific plane wave polarized in the z-direction and propagating in the *x*-direction is considered. In general, a uniform plane wave can be expressed as $\exp[-j\mathbf{k}.\mathbf{r}]\hat{\mathbf{a}}_u$, where $\mathbf{k}=k_x\hat{\mathbf{a}}_x+k_y\hat{\mathbf{a}}_y+k_z\hat{\mathbf{a}}_z$, $\mathbf{r}=x\hat{\mathbf{a}}_x+y\hat{\mathbf{a}}_y+z\hat{\mathbf{a}}_z$ and $\hat{\mathbf{a}}_u$ is a unit vector perpendicular to $\mathbf{k}$. Certainly, $\hat{\mathbf{a}}_u$ can be written as a linear combination of $\hat{\mathbf{a}}_x$, $\hat{\mathbf{a}}_y$, and $\hat{\mathbf{a}}_z$. Moreover, the wave number is $k=\|\mathbf{k}\|=(k_x^2+k_y^2+k_z^2)^{1/2}$; which implies that the variation of this plane wave in each Cartesian direction will be smaller than the variation in the direction of propagation.

In other words, the plane wave **A** of the form exp[-*jkx*]$\hat{\mathbf{a}}_z$ in the analysis above serves as a worst case condition (i.e. corresponding to maximum phase variation in one direction inside an element).

As a conclusion, while dealing with the electromagnetic scattering problems, quadratic elements will give better results in case the electric field has quadratic dependency to the coordinate variables.

APPENDIX C

DOMAIN DECOMPOSITION IN HEXAHEDRAL MESH GENERATION

C.1. Domain Decomposition of Cylindrical Domains

The mesh generation in this thesis depends on the decomposition of the problem to subdomains so that each subdomain is homeomorphic (topologically equivalent) to a rectangular prism. Each subdomain is divided to hexahedra with the constraint that adjacent subdomains will have equivalent quadrilateral surface meshes in order to preserve mesh continuity.

C.1.1. Domain Decomposition for the PEC Cylinder Problem

For the RCS calculation of a Perfect Electric Conductor (PEC) cylinder, it is not required to consider the region inside the cylinder, since it is certainly known that the total electric field vanishes. Hence, this part can be taken out of consideration during the whole process starting from the mesh generation.

The mesh generation for this problem is straightforward. The parameters defining the mesh are:

1. Mesh Resolution in r direction
2. Mesh Resolution in ϕ direction
3. Mesh Resolution in z direction

For the PML regions, which are placed at the top and bottom parts, it is obvious that the central part should also be filled with elements. The way to do this, is to put rectangular prisms in the centers and to extend these prisms to cylinders. The top and bottom figures are illustrated in Fig. C.1.

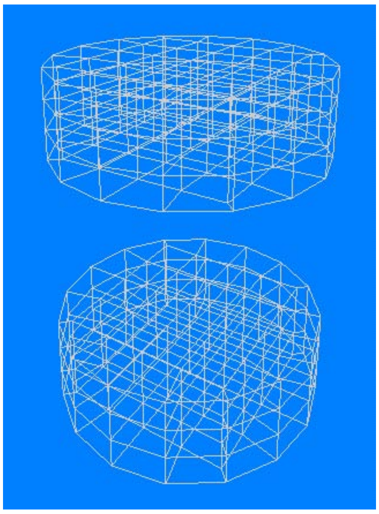


Fig. C.1. Top and Bottom Parts of the Mesh of PEC Cylinder.

Fig. C.2 illustrates the whole mesh generated for the PEC Cylinder problem.

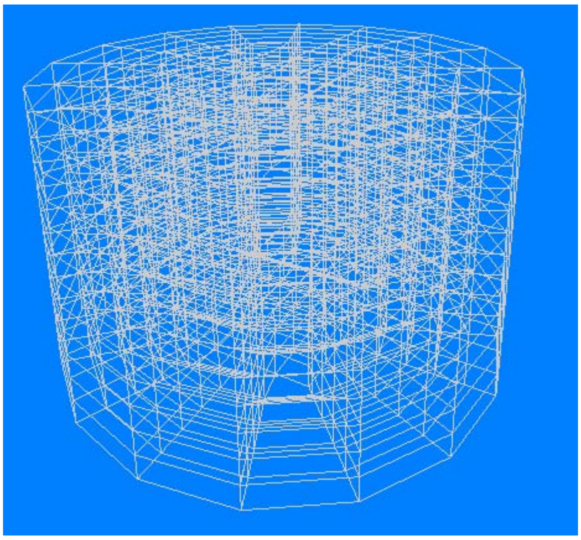


Fig. C.2. Mesh Generated for PEC Cylinder Problem.

See further subsections of this Appendix for the restrictions in mesh generation for this problem.

C.1.2. Domain Decomposition for the Dielectric Cylinder Problem

For the RCS calculation of a dielectric cylinder; unlike the PEC case, it is required to consider the region inside the cylinder. Namely, this part cannot be taken out of consideration during the whole process starting from the mesh generation.

The mesh generation for this problem is slightly different than the PEC cylinder case. Fig. C.3 illustrates the main idea of mesh generation in this problem.

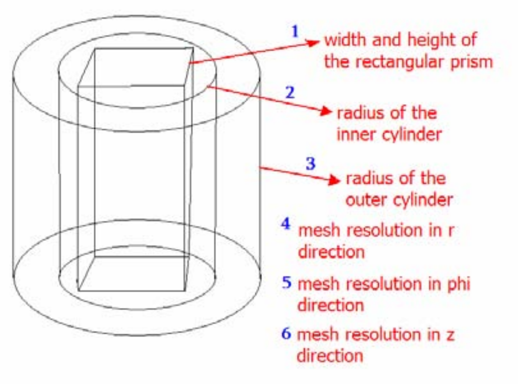

1 width and height of the rectangular prism
2 radius of the inner cylinder
3 radius of the outer cylinder
4 mesh resolution in r direction
5 mesh resolution in phi direction
6 mesh resolution in z direction

Fig. C.3. Mesh generation scheme for the dielectric cylinder.

The parameters defining the mesh can be listed as follows:

i) Width and height of the core rectangular prism

ii) Radius of the core cylinder (of same height with the rectangular prism)

iii) Radius of the outer cylinder (the Volume of Interest)

iv) Mesh Resolution in *r* direction

v) Mesh Resolution in ϕ direction

vi) Mesh Resolution in *z* direction

Another parameter (dependent to wavelength), which will be effective in accuracy during the FEM solution, is as follows:

- Radius and height of the cylinder in terms of wavelength

See further subsections of this Appendix for the restrictions in mesh generation for this problem.

C.1.3. Domain Decomposition for the PEC Sphere Problem

For the RCS calculation of a Perfect Electric Conductor (PEC) sphere, it is not required to consider the region inside the sphere, since it is certainly known that the total electric field vanishes. Hence, this part can be taken out of consideration during the whole process starting from the mesh generation.

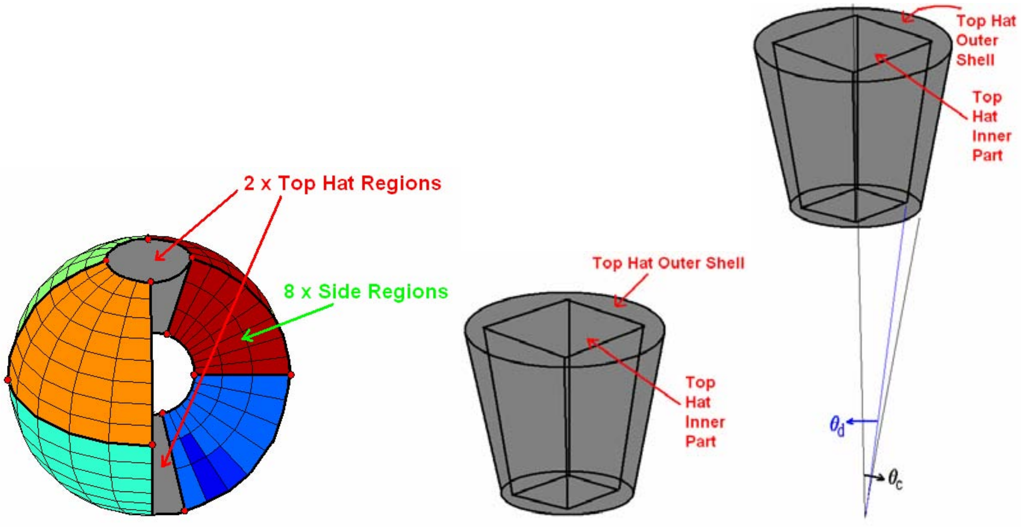


Fig. C.4. Mesh Generation Scheme for the PEC Sphere Problem.

In order not to have a non-hexahedral element, the mesh generation shall be performed as follows:

- The spherical shell volume is divided into 3 main volumes (2 top hat, and side volumes) as seen in Fig. C.4.

- Side volume is divided further into 8 side sub volumes, and each top hat volume is divided to subvolumes called top hat outer shell and top hat inner part respectively.

This scheme is directly applicable for linear and quadratic element mesh generation. The generated meshes are illustrated in the following figures (Fig.s C.5, C.6, C.7, and C.8).

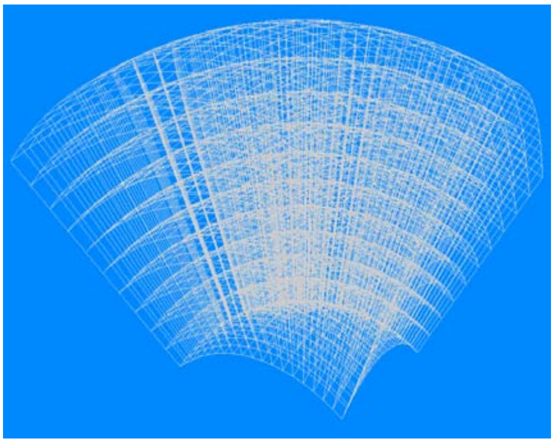


Fig. C.5. Top Hat Inner Part (A).

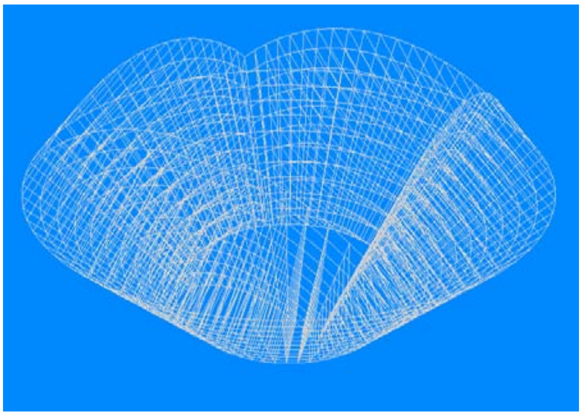


Fig. C.6. Top Hat Outer Shell (B).

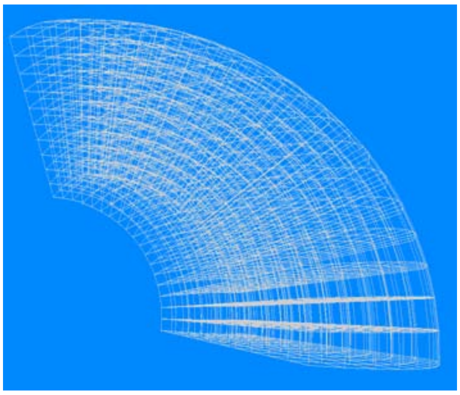


Fig. C.7. Side Subvolume (C).

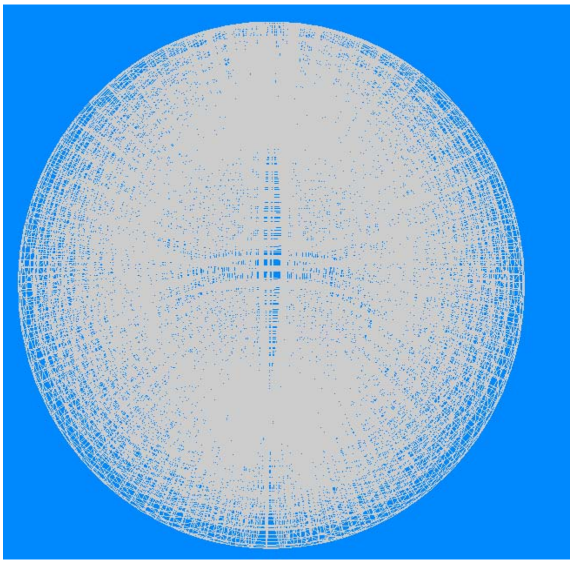


Fig. C.8. Whole Spherical Shell (2A+2B+8C).

The parameters affecting the shape and structure of the generated mesh are as follows:

1. θ_c (see Fig. C.4 for the definition)
2. θ_d (see Fig. C.4 for the definition)
3. Mesh Resolution in R direction
4. Mesh Resolution in θ direction
5. Mesh Resolution in ϕ direction

Two other parameters (dependent to wavelength), which will be effective in accuracy during the FEM solution, are as follows:

1. Radius of the PEC Sphere in terms of wavelength
2. Radius of the total spherical Volume of Interest (VoI)

See further subsections of this Appendix for the restrictions in mesh generation for this problem.

C.1.4. Domain Decomposition for the PEC Sphere Problem

For the RCS calculation of a dielectric sphere; unlike the PEC case, it is required to consider the region inside the sphere. Namely, this part cannot be taken out of consideration during the whole process starting from the mesh generation.

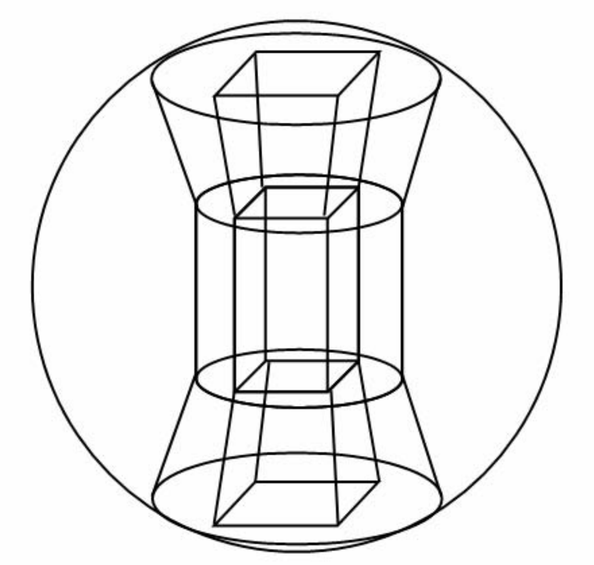


Fig. C.9. Mesh generation scheme in the dielectric sphere problem.

In order not to have a non-hexahedral element, the mesh generation shall be performed as follows:

- The spherical shell volume is divided into 3 main volumes
    - 1 cylindrical core consisting of

        - a rectangular prism,
        - and a cover completing it to a cylinder
    - 2 top hats consisting of
        - inner part,
        - and outer shell
    - and side volume

  as seen in Fig. C.9.

- Side volume is divided further into 8 side sub volumes, and each top hat volume is divided to subvolumes called top hat outer shell and top hat inner part respectively.

This scheme is directly applicable for linear and quadratic element mesh generation.

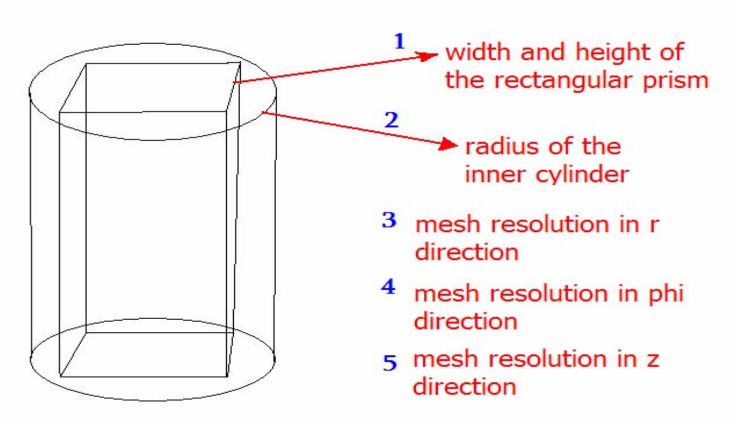

1
width and height of the rectangular prism
2
radius of the inner cylinder
3
mesh resolution in r direction
4
mesh resolution in phi direction
5
mesh resolution in z direction

Fig. C.10. Mesh defining parameters in the dielectric sphere problem.

The parameters affecting the shape and structure of the generated mesh (seen in Fig. C.10) are as follows:

1. Width and height of the core rectangular prism
2. Radius of the core cylinder (of same height with the rectangular prism)
3. Mesh Resolution in r direction (which is also implicitly defining the mesh resolution in R direction)
4. Mesh Resolution in ϕ direction

5. Mesh Resolution in z direction (which is also implicitly defining the mesh resolution in θ direction)

Two other parameters (dependent to wavelength), which will be effective in accuracy during the FEM solution, are as follows:

1. Radius of the sphere in terms of wavelength
2. Radius of the total spherical Volume of Interest (VoI)

The relevant generated subvolume meshes are given in the following figures (Fig.s C.11, C.12, C.13, C.14, and C.15).

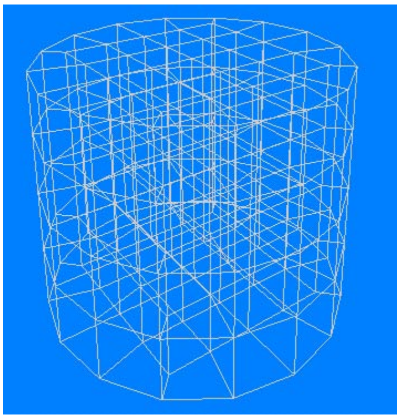


Fig. C.11. The Inner Cylindrical Core (A).

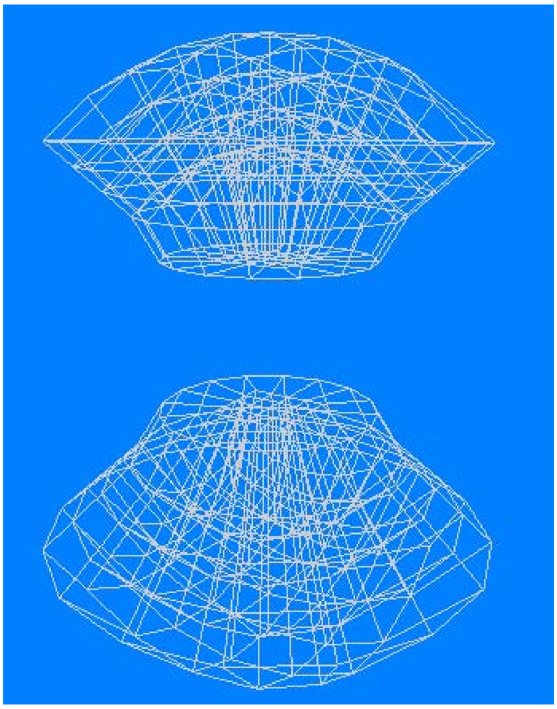


Fig. C.12. Top Hats (B).

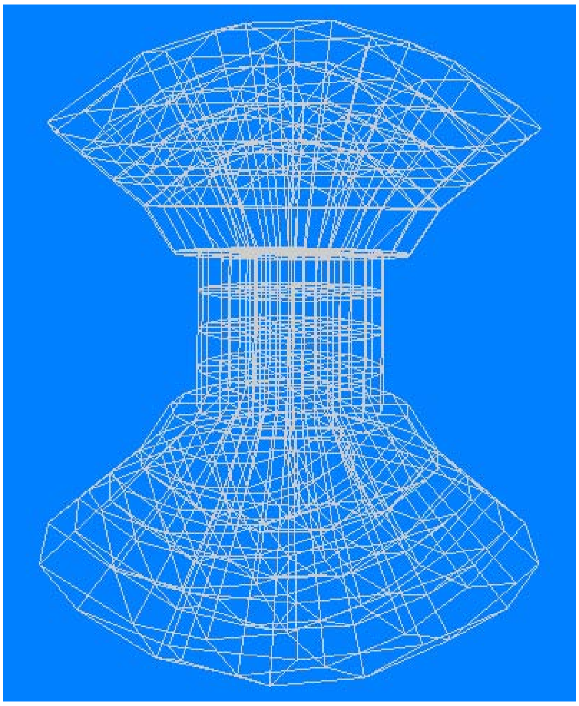


Fig. C.13. The Inner Cylindrical Core Combined with the Top Hats (A+B).

See further subsections of this Appendix for the restrictions in mesh generation for this problem.

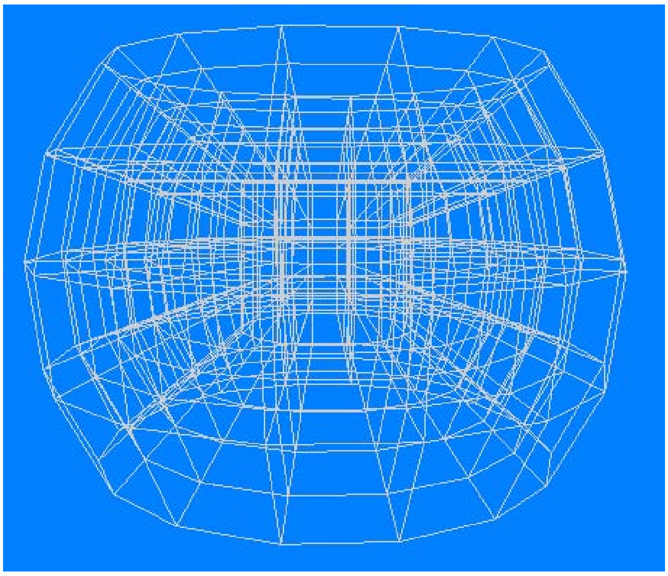


Fig. C.14. The Side Cover (C).

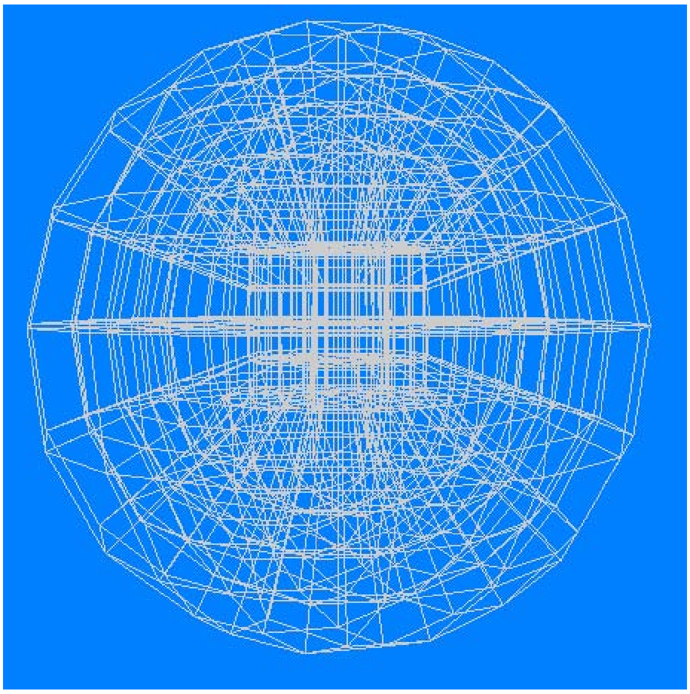


Fig. C.15. The Whole Sphere After All Parts Combined (A+B+C).

See further subsections of this Appendix for the restrictions in mesh generation for this problem.

C.2. Restrictions to Satisfy the “All-Hexahedra” Condition

During the all-hexahedral mesh generation process, the mesh defining parameters should be chosen carefully. This restriction is valid and similar for both the cylindrical and spherical geometries. Fig. C.16 illustrates the reason for this restriction.

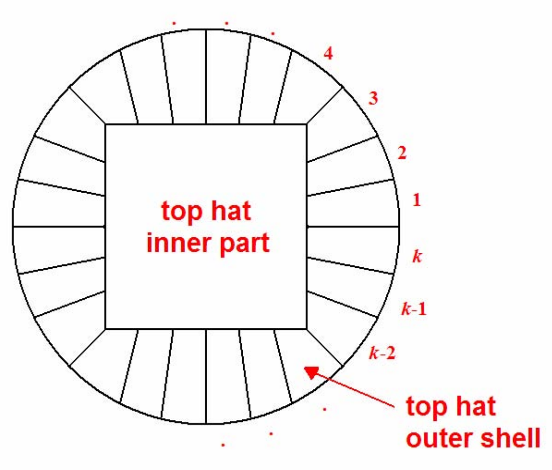


Fig. C.16. Top view of the top hat for the illustration of the restriction.

The number k, should be divisible by 8 in order not to have a singularity (a non-hexahedral element). k is obviously determined by the mesh resolution in ϕ direction. Also, the mesh density in the inner part is closely related with k. Surface-wise there are:

- m^2 elements in the inner part of the top hat, and
- 4×m elements in the top hat outer shell,

where m=((k+4)/4)-1. This yields a very dense mesh in the top hat, if θ_c and θ_d values are not chosen very large.

As a numerical example, a 22.5° mesh resolution in ϕ direction implies that k = 16; and m = ((k+4)/4)-1 = 4; which means that there will be surface-wise m^2=16 elements in the inner part of the top hat, and 4×m=16 elements in the top hat outer shell; with a total of 32 elements.

APPENDIX D

A PARTICLE SWARM OPTIMIZATION APPROACH IN HEXAHEDRAL MESH SMOOTHING

This Appendix is devoted to the Particle Swarm Optimization method and the application of it to the all-hexahedral mesh smoothing.

D.1. An Algebraic Hexahedral Mesh Quality Metric

Many papers have been devoted to the topic of unstructured mesh smoothing and optimization [53–63, 74, 166]. However, there are only a limited number of papers which specifically address unstructured hexahedral mesh smoothing [74, 166]. The method described in [166] is based on maximizing a variant of the scaled-Jacobian metric. This approach does not guarantee that the improved mesh will consist only of untangled elements; and actually have improved shape-quality according to the user definition. The method in [74] achieves these criteria. The aim of this paper is to propose another method, which is as successful as [74].

For finite element meshing three geometric qualities of elements are almost always important: invertibility, size, and shape. An element is invertible if it has positive local volume. If a hexahedral mesh contains inverted elements the hexahedral mesh untangling algorithm [167] is recommended to automatically remove the inverted elements by node repositioning. Assuming a mesh contains no inverted elements, element ‘size’ becomes the next most important metric. Element size must be small enough that discretization error is small, yet large enough that computer memory is not exceeded and the application can be solved in a reasonable amount of time.

Invertibility, size, and shape are the three important factors of the element quality:

- If an element has positive local volume (everywhere in the element, not just at the 8 corners), then it is considered to be *invertible*.

- For a mesh without any inverted elements, element *size* becomes the factor under consideration. An element must be small enough to have small discretization errors, and on the other hand, it should be sufficiently large such that available computer resources (CPU time, memory) are efficiently used.

- The last metric is element *shape*, which is a function of element aspect ratios and skew [74]. Skew gives information about the angles within an element regardless of the aspect ratio. Accuracy decreases if an element contains very large and small angles (i.e. close to 0 or 180°). In [73], Knupp showed that the shape can be expressed as a function of the *condition number*, which can be improved by means of:

  i) aspect ratio improvement, and/or
  ii) element skew reduction.

Definitions about element quality are directly taken from [74]. For clarification, these definitions are repeated in this section. The eight nodes of a hexahedral element (k =0, 1,…, 7) can be numbered such that the nodes 0, 1, 2, 3 are at the bottom, and the nodes 4, 5, 6, 7 are at the top surface; this numbering scheme and the node coordinates of hexahedral element transformed to the $\xi\eta\zeta$-space are listed in Table D.1.

Table D.1. Nodal ordering for a hexahedral element.

k	(ξ, η, ζ)	x_1	x_2	x_3
0	(0, 0, 0)	1	3	4
1	(1, 0, 0)	2	0	5
2	(1, 1, 0)	3	1	6
3	(0, 1, 0)	0	2	7
4	(0, 0, 1)	7	5	0
5	(1, 0, 1)	4	6	1
6	(1, 1, 1)	5	7	2
7	(0, 1, 1)	6	4	3

Let **x** be the position vector of one of the eight nodes; and $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ be the coordinates of the three neighbor nodes. Their order for proper orientation is again listed in Table D.1. Three edge vectors $\mathbf{e}_1 = \mathbf{x}_1 - \mathbf{x}$, $\mathbf{e}_2 = \mathbf{x}_2 - \mathbf{x}$, $\mathbf{e}_3 = \mathbf{x}_3 - \mathbf{x}$ and the matrix **A** can be formed from the three column vectors, $\mathbf{A} = [\mathbf{e}_1 \mid \mathbf{e}_2 \mid \mathbf{e}_3]$. This means that if the vector **x** is (x, y, z); and for $i = 1, 2, 3$ if the vectors $\mathbf{x}_i$ are (x_i, y_i, z_i), then **A** can be written as follows:

$$\mathbf{A} = \begin{bmatrix} x_1 - x & x_2 - x & x_3 - x \\ y_1 - y & y_2 - y & y_3 - y \\ z_1 - z & z_2 - z & z_3 - z \end{bmatrix} \tag{D.1}$$

For each node of the hexahedron, eight such matrices, $\mathbf{A}_k$ can be constructed. It is assumed that the element is untangled; namely $det(\mathbf{A}_k) \geq 0$ for all k. If there is a tangled element inside the mesh, then prior to the optimization it should be untangled. For this purpose, the method described in [167] can be used.

Knupp also has defined weight matrices $\mathbf{W}_k$ in order to specify the ideal element shape in [73]. By means of $\mathbf{A}_k$ and $\mathbf{W}_k$, the matrices $\mathbf{T}_k = \mathbf{A}_k \mathbf{W}_k^{-1}$ are formed and used in order to represent the shape metrics. The shape metric is defined as follows: the matrices $\mathbf{T}_k$ resemble an orthogonal matrix if the objective function gets optimized. In case that the element becomes an ideal element (most probably with a different orientation), $\mathbf{T}_k$ becomes orthogonal, and eventually $\mathbf{A}_k = \mathbf{T}_k \mathbf{W}_k$,. For hexahedra, the ideal shape is a cube; and the weight matrix is the identity matrix.

Let the condition number be $\kappa(\mathbf{T}) = |\mathbf{T}|\,|\mathbf{T}^{-1}|$, where the matrix norm is the Frobenius norm. $f = 8/\sum_k (\kappa(\mathbf{T}_k)/3)^2$ is an algebraic shape metric for hexahedral elements. The proof is given in [74]. With this definition, f is a non-simplicial algebraic shape metric since it has the following properties:

- the domain of f is the set of matrices $\mathbf{T}_k = \mathbf{A}_k \mathbf{W}_k^{-1}$, $k = 0, 1, \ldots, K - 1$, with $det(\mathbf{T}_k) \geq 0$,
- f is size invariant,
- f is orientation invariant,

- For all $\mathbf{T}_k$, $0 \le f(\{\mathbf{T}_k\}) \le 1$,
- $f(\{\mathbf{T}_k\}) = 1$ if and only if $\mathbf{T}_k$ is a scalar multiple of an orthogonal matrix for all k,
- $f(\{\mathbf{T}_k\}) = 0$ if and only if $det(\mathbf{T}_k) = 0$ for some k; i.e. the three edges having a common node are coplanar.
- It should be noted that for tangled elements, shape is not defined.

(The notation $f(\{\mathbf{T}_k\})$ stands for f being a function of all $\mathbf{T}_k$ matrices.)

It is obvious that other functions satisfying the definition of a shape metric can also be defined and used in order to define an objective function for element quality improvement. For example, shape can be defined by modified Winslow

$$(f = 3/(\tau^{2/3}|\mathbf{T}^{-1}|^2)), \tag{D.2}$$

mean ratio

$$(f = 3/(\tau^{-2/3}|\mathbf{T}|^2)), \tag{D.3}$$

and inverse condition number

$$(f = 3/\kappa(\mathbf{T})). \tag{D.4}$$

The present work focuses on the condition number shape metric.

D.2. The Condition Number Based Objective Function

Based on the shape quality metric mentioned in the previous section, an objective function is described in this section. Again, the definition is reused from [74]. The objective function considers the shape quality of all elements in a hexahedral mesh. Let $\mathbf{T}_{n,k}$ be the matrix corresponding to the kth node of the nth hexahedral element ε_n. We can define

$$f_n = f(\varepsilon_n) = \frac{1}{\frac{1}{8}\sum_k (\kappa(\mathbf{T}_{n,k})/3)^2} \tag{D.5}$$

as the quality metric of the *n*th hexahedral element in the mesh (where $n = 1, \ldots, N$).

For the definition of the objective function, it is better to work with $1 / f_n$ instead of f_n because it provides a greater numerical range and a steeper gradient; as well as a metric creating a barrier. The objective function is the sum over all elements (ε_n's) inside the hexahedral mesh (ε)

$$F = \frac{1}{N}\sum_{n}(1/f_n) - 1 = \frac{1}{8N}\sum_{n}\sum_{k}(\kappa(\mathbf{T}_{n,k})/3)^2 - 1 \tag{D.6}$$

This is nothing but the sum of the squares of the element condition numbers. The objective function is scaled so that the minimum value of F is 0.

D.3. Particle Swarm Optimization (PSO)

The particle swarm optimization (PSO) method is an effective optimization algorithm, which has been applied successfully to some difficult multidimensional continuous/discontinuous problems in various fields so far [168]. Moreover, this technique has been shown to be outperforming other optimization methods such as genetic algorithms (GAs) [169].

PSO, which has been developed in 1995 by Kennedy and Eberhart [170], can be described through its leading example: Assume that there is a swarm of bees whose main aim is to find the location with the highest density of flowers in a field without any a priori knowledge; starting at random locations with random velocities. Each bee can remember its previsited successful locations (cognitive behavior), and also it can feel the best locations found by the swarm (social behavior). When a bee finds a better place than previously found places, then it would have tendency to go to this new location in addition to the best location found by the swarm. Eventually, the whole swarm would be attracted towards that location.

Each member of the swarm is referred to as a *particle*; which corresponds to a solution candidate. All the particles accelerate toward the best personal and best overall location; meanwhile they continuously check their value of their current locations.

Each member of the swarm remembers the best location of own discovery. This location is called as the personal best or *pbest*. On the other hand, each member feels the best location discovered by the swarm. This is called as the global best or *gbest* of the swarm.

The necessary steps for the PSO algorithm are certainly as follows: After the definition of the fitness/objective function; and the definition of the solution space; the particles (i.e. locations and velocities) in the swarm are initialized. Then the particles are moved inside the solution space. For each particle, the fitness is evaluated at the relevant particle's location. If this value is greater than the value calculated at *pbest* of the relevant particle, or the global *gbest* of the swarm, then these values are updated.

The velocity manipulation is the main key to convergence in PSO. Locations of *pbest* and *gbest* are major factors for the new velocity value of a particle during this step. A particle gets accelerated in the directions of *pbest* and *gbest* as follows:

$$v_n = w.v_n + c_1.u_1.(p_{best,n} - x_n) + c_2.u_2.(g_{best,n} - x_n) \tag{D.7}$$

where x_n is the particle's coordinate in the *n*th dimension and v_n is the velocity of the particle in the *n*th dimension. This operation is performed at each dimension in an *N*-dimensional problem. A pictorial description in 2-dimensions can be seen in Fig. D.1.

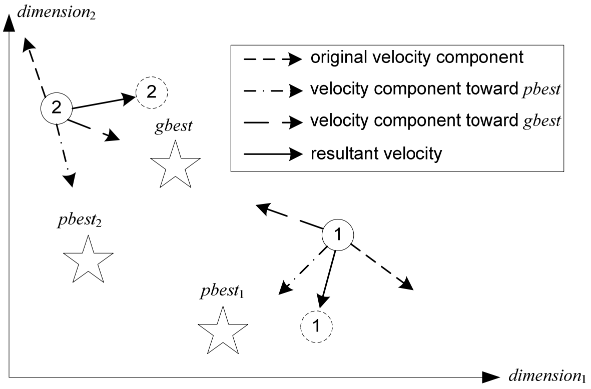

dimension$_2$
original velocity component
velocity component toward pbest
velocity component toward gbest
resultant velocity
2
2
gbest
pbest$_2$
1
pbest$_1$
1
dimension$_1$

Fig. D.1. Pictorial Description of PSO in 2D.

It can clearly be seen in this equation that the new velocity is the summation of the current velocity scaled by w and increased in the direction of *gbest* and *pbest* for that dimension.

c_1 and c_2 are scaling factors representing the attraction powers of *pbest* and *gbest*. c_1 is a factor showing the memory/history influence on a particle's movement (i.e. a metric of cognitivity), and c_2 is a factor showing the swarm's influence on a particle's movement (i.e. a metric of sociality). Increasing c_1 increases a particle's tendency to its own *pbest*; whereas increasing c_2 increases a particle's tendency to the assumed global maximum.

u_1 and u_2 are random numbers between 0.0 and 1.0 obeying uniform distribution. In most PSO implementations, two independent random numbers are used in order to control the attraction powers of *gbest* and *pbest*. The main reason for this is to add a flavor of unpredictability to the behavior of the swarm. w is known as the inertial weight, and this number (chosen to be between 0.0 and 1.0) determines how much the particle remains along its original direction regardless of the *gbest* and *pbest* attraction. This is a factor adding diversity, and setting up a balance between exploration and exploitation. Detailed discussions about the ideal choices of c_1, c_2 and w can be found in [171].

After the velocity has been calculated, the movement of the particle is straightforward. The velocity is applied during a given time-step, which is usually chosen to be unity; and the new coordinate is calculated at each dimension as follows:

$$x_n = x_n + \Delta t . v_n \tag{D.8}$$

After these operations are completed for all particles in the swarm, the whole movement and fitness evaluation process is repeated. Hence, the particles are moved for discrete time intervals as if their snapshots are taken at the end of each time-interval. This is carried on until the termination criterion (criteria) is (are) met. There might be several termination criteria, such as maximum iteration number, achievement of target fitness, saturation in improvement of *gbest* etc.

In most applications, it is usually desired to put constraints on the search domain. Due to the movements of the particles, there is always a possibility that particles fall outside the solution space during the iterations. In order to prevent/avoid this problem, three different boundary conditions can be imposed [171] as seen in Fig. D.2:

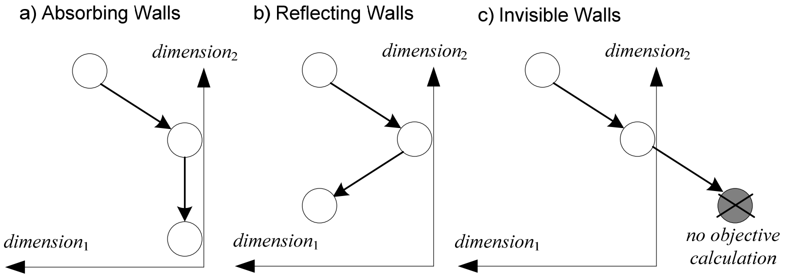


Fig. D.2. Boundary Condition/Wall Concept in PSO.

1) If a particle exceeds the boundary of the solution space at one dimension, the velocity in that dimension is set to zero; and the relevant particle is implicitly pulled back toward the allowed solution space. This case can be considered as an *absorbing boundary condition*.

2) If a particle exceeds the boundary of the solution space at one dimension, the velocity is reversed in that dimension; and hence the particle is directly reflected back; which can be considered as a *reflecting boundary condition*.

3) Without any constraints, the particles are moved to everywhere; but for a, particle falling outside, fitness is not evaluated; which is interpreted as an *invisible boundary condition*.

Another idea is to push the particles away from the worst solutions, instead of pulling them towards the best solutions. This idea is proposed by Yang and Simon [172], and named as NPSO standing for New Particle Swarm Optimization. In NPSO, the conventional PSO equation (D.7) is modified as in (D.9); where *pworst* and *gworst* are used instead of *pbest* and *gbest*.

$$v_i = w.v_i + c_1.u_1.(x_i - p_{worst,i}) + c_2.u_2.(x_i - g_{worst,i}) \qquad \text{(D.9)}$$

A more detailed comparison of NPSO with the classical PSO (in terms of convergence etc.) can be found in [172]. A pictorial description of NPSO in 2 dimensions can be seen in Fig. D.3. More derivatives and hybridizations of PSO have been proposed since the first proposal in 1995.

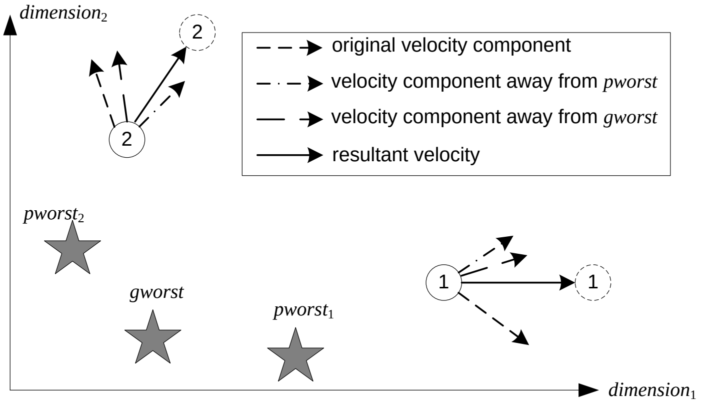

dimension$_2$
2
2
original velocity component
velocity component away from pworst
velocity component away from gworst
resultant velocity
pworst$_2$
gworst
pworst$_1$
1
1
dimension$_1$

Fig. D.3. 2D Pictorial Description of NPSO.

D.4. Adaptation of PSO and Derivatives to Mesh Smoothing

Adaptation of PSO to the mesh smoothing will fall into the optimization-based smoothing category. Certainly, optimization based smoothing methods are computationally expensive compared to methods like Laplacian methods; on the other hand, they do not have any restrictions like Laplacian methods. The main motivations for the usage of PSO in this work can be summarized as follows:

- First of all, PSO is a young but promising, flexible, easy-to-implement global optimization algorithm which is suitable for multidimensional continuous optimization problems by its definition. Due to these facts, it is appropriate for the mesh smoothing problems.
- The method is open to modifications, variations, and hybridizations for performance improvement purposes.

- Unlike some other optimization algorithms, it does not require any a priori information about the gradient of the objective function. Evaluation of the objective function at a given point is sufficient for the implementation.
- So far, it has been demonstrated that PSO outperforms (in terms of accuracy, convergence speed, CPU & memory requirements) most of the nature-inspired optimization algorithms.
- By its well known rapid convergence feature, it can provide quick results from highly distorted mesh. Moreover, unlike most mesh smoothing techniques, it provides global optimization rather than yielding local extrema; which are highly probable to be encountered in the mesh smoothing problems.
- Since it is a population based search method, PSO provides not only the best solution, but also a large set of good solutions. There might be some applications for mesh smoothing, where one would like to get advantage of this property.
- The composite nature of PSO makes it especially conducive to implementation on parallel processors.

For the adaptation of PSO (and its derivatives such as NPSO [172]) to mesh smoothing problems, in the most general case it should be noted that the objective function F is a function of n variables, where n is the total number of node coordinates to be adjusted for mesh quality improvement. More specifically, if the nodes to be adjusted in the problem are P_1, P_2, …, P_k where $P_i=(x_i, y_i, z_i)$; then the objective function F will be a function of $n=3k$ variables, and it can be written as $F(x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_k, y_k, z_k)$.

Considering that the number of nodes in a mesh is usually very large, at first glance it can be said that the problem will yield a function F with a large number of variables. The following paragraphs give an idea about how large the dimension or the degree of freedom ($D.O.F$) is.

The mesh seen in Fig. D.4 has a total of $(N+1)(L+1)(M+1)$ nodes assuming that all elements are of first order. During the smoothing of this mesh; if all nodes are allowed to move, then the degree-of-freedom ($D.O.F$) of this problem will be:

$$D.O.F_{max} = 3(N+1)(L+1)(M+1) \tag{D.10}$$

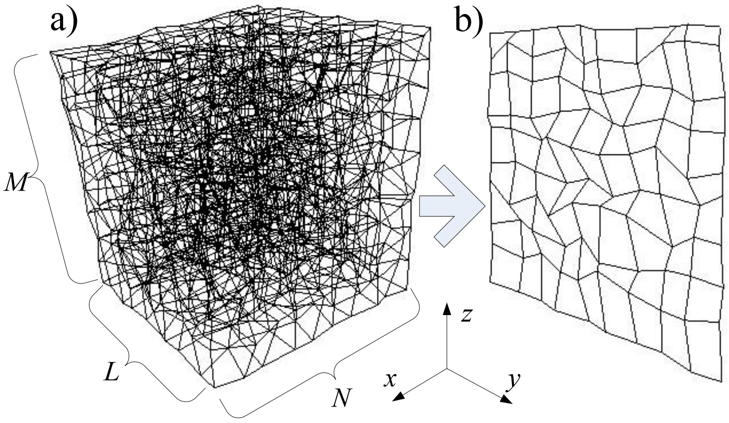


Fig. D.4. An all-hexahedral mesh with *NLM* elements:
a) 3D Isometric View. b) Isometric View of One Sample Layer.

The value in (D.10) represents the worst case; and hence it is called as $D.O.F_{max}$. Obviously by allowing all the nodes to move, the shape of the volume of interest will most probably change after mesh smoothing. However in most cases, the shape of the volume of interest is preserved, which means:

- The corner nodes are fixed,
- The surface nodes are allowed only to move along the surface, and
- The edge nodes are allowed only to move along the edges.

With such restrictions, it can be shown that the *D.O.F* might reduce down to:

$$D.O.F_{nom} = 3(N\text{-}1)(L\text{-}1)(M\text{-}1) + 4M(L\text{-}1) + 4L(N\text{-}1) + 4N(M\text{-}1) \qquad \text{(D.11)}$$

The proof is straightforward depending on the brute force count of the nodes. This value represents the typical case, and it is a nominal value; hence it is called as $D.O.F_{nom}$. This means that $D.O.F_{max}$ is only a theoretical upper bound, which is not encountered in practice.

Another extreme case is to preserve the surface mesh completely. In such a case, the $D.O.F$ reduces to:

$$D.O.F_{min} = 3(N\text{-}1)(L\text{-}1)(M\text{-}1) \tag{D.12}$$

In summary, $D.O.F_{min}$ is a lower bound, whereas $D.O.F_{max}$ is an upper bound; and $D.O.F_{nom}$ is a typical value for the smoothing of such volume of interest.

It is sure that high $D.O.F$ will increase the computation efforts during the mesh smoothing. The following paragraphs discuss some techniques in order to reduce the $D.O.F$, and to have a more efficient PSO solution:

1. *Domain Decomposition (The Divide and Conquer Method):* During the improvement of the mesh quality, there might be opportunities to decompose the main domain into r independent subdomains. Via this manipulation, instead of trying to solve a PSO problem with n-$D.O.F$, one can try to solve r independent problems with n_i-$D.O.F$ where $\Sigma n_i = n$.

Assume that the volume of interest (i.e. the problem domain) is divided into subdomains. For mesh continuity, two adjacent subdomains (say V_1 and V_2) should have the same surface mesh at their shared surface (S_{12}). Assume that the mesh of V_1 is smoothed initially; and assume that the surface mesh at S_{12} is preserved during the smoothing of V_2. This means that in such a case, the expected degree-of-freedom (say $D.O.F_{exp}$) for the smoothing operation of V_2 mesh will be even lower than $D.O.F_{nom}$. Moreover, if a volume is surrounded by other volumes in all directions; and if all of its surface meshes have already been smoothed, then $D.O.F_{exp}$ will be reduced down to $D.O.F_{min}$. Consequently, for a non-isolated volume (i.e. surrounded by other volumes) the following can be said about $D.O.F_{exp}$:

$$D.O.F_{min} \leq D.O.F_{exp} \leq D.O.F_{nom} \tag{D.13}$$

The optimum computation size (in terms of number of elements) is tried to be investigated by means of a simple analysis. A PSO setup with 20-particle population and

50 iterations is executed. The degree-of-freedom is chosen to be worst case; i.e. all nodes are allowed to be floating, which yields $D.O.F_{max}$. Table D.2 shows the elapsed time (both total and per element) during the PSO-smoothing for meshes with for various N, L, and M values. All preprocessing (memory allocations, PSO population setup, etc.) and postprocessing (memory deallocations, result displays, etc.) operations are included in the total elapsed time.

Table D.2. Performance Measures for PSO-Mesh Smoothing of Various Domains.

					PC-1		**PC-2**	
N	***L***	***M***	**Number of Elements**	**Maximum Degree of Freedom**	**Total Elapsed Time (sec)**	**Elapsed Time Per Element (sec)**	**Total Elapsed Time (sec)**	**Elapsed Time Per Element (sec)**
2	2	2	8	81	0.06	0.00750	0.048	0.00600
3	3	3	27	192	0.09	0.00333	0.065	0.00241
4	4	4	64	375	0.17	0.00266	0.113	0.00177
5	5	5	125	648	0.27	0.00216	0.193	0.00154
6	6	6	216	1029	0.461	0.00213	0.321	0.00149
7	7	7	343	1536	0.711	0.00207	0.513	0.00150
8	8	8	512	2187	1.122	0.00219	0.786	0.00154
9	9	9	729	3000	1.753	0.00240	1.268	0.00174
10	10	10	1000	3993	2.644	0.00264	2.022	0.00202
11	11	11	1331	5184	3.925	0.00295	3.146	0.00236
12	12	12	1728	6591	5.888	0.00341	4.414	0.00255
13	13	13	2197	8232	8.472	0.00386	5.838	0.00266
14	14	14	2744	10125	12.067	0.00440	8.015	0.00292
15	15	15	3375	12288	16.424	0.00487	10.997	0.00326
16	16	16	4096	14739	22.342	0.00545	14.894	0.00364
17	17	17	4913	17496	29.923	0.00609	20.274	0.00413

Number of Particles = 20
Number of Iterations = 50
PC1: Intel Pentium M 1.4 GHz 512MB RAM Laptop
PC2: Intel Pentium 4 3.4 GHz 1,00GB RAM Desktop

The performance measurement is performed by means of PSO mesh generation script executed at Matlab 6.5 on two separate PCs, where:

- PC1 is a laptop Windows PC with Intel Pentium M 1.4GHz CPU and 512MB RAM,
- PC2 is a desktop Windows PC with Intel Pentium 4 3.4GHz CPU and 1GB RAM.

Investigation of Table D.2 yields the following observations:

- The elapsed time per element is minimum (about 0.00150 seconds on PC-2) for 6×6×6 or 7×7×7-element sized domains.
- For domains smaller than 6×6×6, the overhead for the problem setup (operations like swarm generation, and updates) seem to be dominant in terms of CPU time. Hence, due to such overheads, elapsed time per element is high for small domains.
- For domains larger than 7×7×7, PSO related operations seem to be dominant. As the allocated memory and the particle sizes increase, the updates and objective function evaluations take longer times.

By using the results of this analysis, the following divide-and-conquer strategy can be proposed:

- For the most efficient PSO mesh smoothing, a mesh can be considered as a collection of 7×7×7-element (or comparable sized) subdomains.
- Mesh smoothing can be performed at each subdomain one by one.

By using the divide-and-conquer strategy, a mesh of 100,000 elements is smoothed by PSO (with 50 iterations, 20 particles) about 84 seconds on PC-2 by using 7×7×7-element subdomains. During this experiment;

- The shape of the main domain is preserved; i.e. the nodes along the outer surfaces are allowed to be moving along the surfaces. There is no other specific restriction.
- For the interior subdomains, eventual *D.O.F* reduction is performed by getting use of the shared surface mesh, if the relevant adjacent subdomain is already smoothed. For such subdomains, the *D.O.F* is $D.O.F_{exp}$, where its lower and upper bounds are given in (D.13).

The optimality of 7×7×7-element subdomain size can be observed in the same problem numerically. By using the same setup described above, the solution of the same problem takes 123 seconds on PC-2 if 10×10×10-element subdomains are used.

Such a strategy will dramatically reduce the complexity and the computation time. Certainly, the number of iterations necessary for convergence highly depends on the

D.O.F. Moreover, the population size should be increased for high *D.O.F* problems. Nevertheless, the results of this analysis give an idea of optimum subdomain size (which is 7×7×7 or equivalent) for fixed population size and fixed number of iterations.

Fig. D.5 shows the PSO-smoothed version (via divide-and-conquer method) of the mesh seen in Fig. D.4.

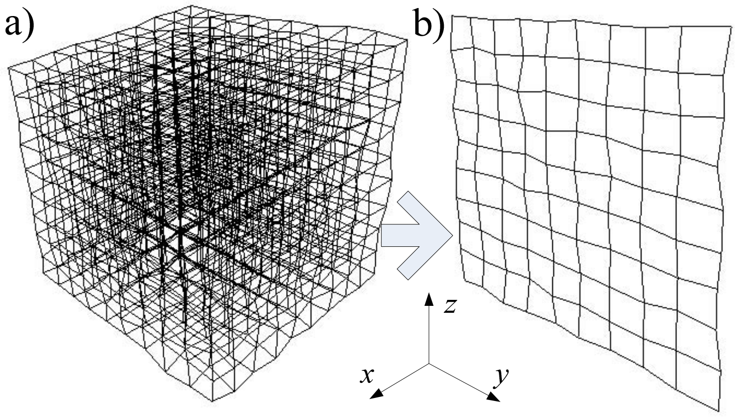


Fig. D.5. PSO-smoothed version of the mesh in Fig. D.4.
a) 3D Isometric View. b) Isometric View of One Sample Layer.

2. *Fixing Some Nodes:* Instead of trying to move all the nodes, some nodes (especially the ones on the boundaries) might be considered to be fixed. As an example, if the *i*th and *j*th nodes are defined to be fixed; then the function $F(x_1, y_1, z_1, \dots, x_i, y_i, z_i, \dots, x_j, y_j, z_j, \dots, x_k, y_k, z_k)$ will be simplified to $F(x_1, y_1, z_1, \dots, x_k, y_k, z_k)$.

3. *Imposing Nodes Dependencies:* The movement of some nodes can be defined to be dependent to each other. For example, if the movement of the *i*th node in *x* direction is set to be dependent to the movement of the *j*th node in *x* direction; then the function $F(x_1, y_1, z_1, \dots, x_i, y_i, z_i, \dots, x_j, y_j, z_j, \dots, x_k, y_k, z_k)$ will be simplified to $F(x_1, y_1, z_1, \dots, x_i, y_i, z_i, \dots, y_j, z_j, \dots, x_k, y_k, z_k)$.

4. *Setting Rules to Individual Node Movements:* The movement of a node inside a mesh might be defined to be in some specific direction; to yield a dependent movement in two directions. For example, the movement of a node might be defined to be in r direction of the cylindrical coordinates where $r=(x^2+y^2)^{1/2}$. If the movement of the ith node is set to be in r direction; then the function $F(x_1, y_1, z_1, \ldots, x_i, y_i, z_i, \ldots, x_k, y_k, z_k)$ will be simplified to $F(x_1, y_1, z_1, \ldots, r_i, z_i, \ldots, x_k, y_k, z_k)$.

5. *Reduction by Means of Symmetry:* For symmetric problems, instead of trying to optimize the whole mesh, only a subset can be optimized and the whole mesh can be reconstructed. For some problems, this might cause the dimension of F to reduce to 1/8 or even 1/16 of the original; if a solution in an octant or half octant is sufficient.

Certainly, manipulations as fixing some nodes, imposing some nodes to be dependent, setting rules to individual node movement reduce the level of quality improvement. There is a trade-off between the quality of the final mesh and the computation time. On the other hand, increasing the *D.O.F* does not always guarantee better improvement. Moreover, manipulations as reduction by means of symmetry might not be applicable in most of the problems in practice. Practically, methods other than domain decomposition (divide-and-conquer) might not be applicable in most cases.

The adaptation of PSO to mesh smoothing is slightly different than the normal PSO procedure. Instead of initializing all the particles in n-dimensional space in a totally random manner, an automatically generated mesh is used for this purpose. All particles are positionally initialized by superimposing Gaussian noise (with zero mean and a user defined variance) to the automatically generated mesh at each dimension. The initial velocities of the particles at each dimension are generated as in the ordinary PSO. The Δt value is chosen to be unity; and the initial random velocities of the particles are assigned so that a node can move a distance of at most l_i along one direction due to this velocity component at first iteration. Here, l_i is a user-defined parameter; chosen to be comparable to average edge length along one direction.

Obviously, the choice of the step size (i.e. both Δt and v_n) in the optimization has great impact on the convergence. So far, the effects of the step size and its selection/computation have not been specifically investigated in the mesh smoothing problem. This is a potential subject of further research.

The fitness evaluation throughout the algorithm is achieved by means of minimization of *F*; i.e. the fitness of a mesh increases as *F* decreases. All *pbest*, *gbest* computations are performed by using this definition.

With manipulations, initializations and definitions described above, it is possible to apply PSO and its derivatives to the mesh smoothing problems.

For all examples in the present work, a population size of 15 is chosen (for most applications it is shown that a population size of 10-20 is efficient); and the number of iterations is taken as 50. c_1 and c_2 are chosen to be 2.0 as usual, and w is chosen to be 1.0. Moreover, for all nodes reflecting walls are defined.

As a practical application of above manipulations, two problems in engineering electromagnetics are given as examples. As will be noticed, there are so numerous manipulations in the examples that; one can think whether these reductions are worth to apply rather than solving the problem with high *D*.O.*F*. Certainly, it is wiser to solve these problems directly rather than spending effort to decrease the *D*.O.*F*; but the examples are just given to demonstrate the application of the suggested reduction methods.

First, the circular microstrip patch antenna, which is a well-known structure both for scattering or radiation problems in engineering electromagnetics, is considered. The mesh generated for this problem should be conformal to the circular patch internally; and it should be conformal to the rectangular prism substrate externally. An automatically generated all-hexahedra mesh for this structure can be seen in Fig. D.6.

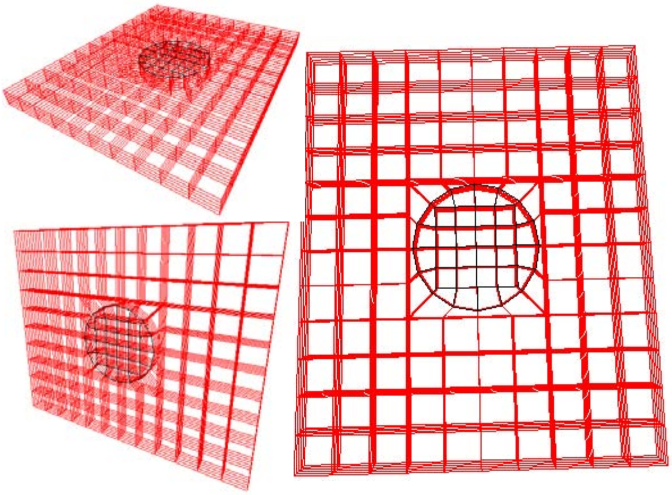


Fig. D.6. 3D Views of the Circular Microstrip Patch Antenna.

First, the problem domain can be reduced by means of symmetry; where the problem can be solved in a quadrant. Then domain decomposition can be performed by considering the sub-domain inside the circular patch; and the sub-domain between the circular patch and the outer boundary of the substrate separately. More dimension reduction can be achieved if further symmetry is considered in each sub-domain. These manipulations are illustrated in Fig. D.7. Moreover in each sub-domain, the nodes at the boundaries can be defined to be fixed; and the movements of some nodes can be defined in specific directions only. These operations can be seen in Fig. D.8.

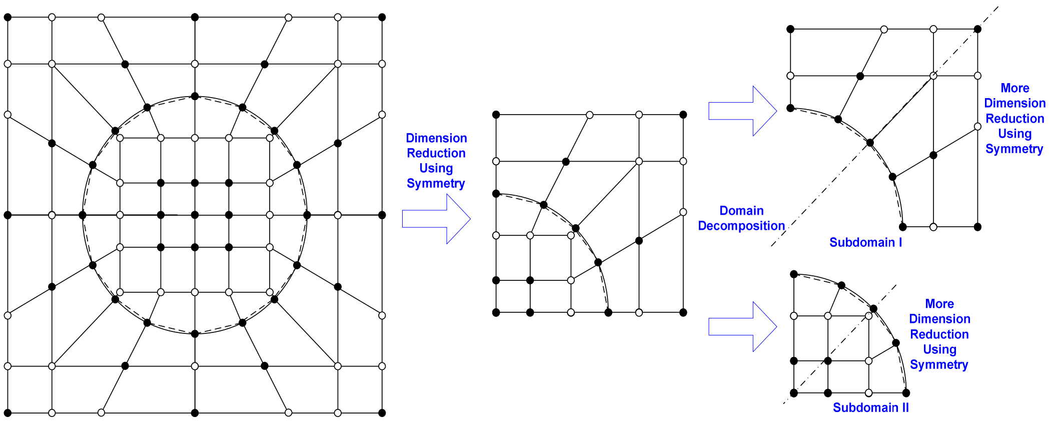
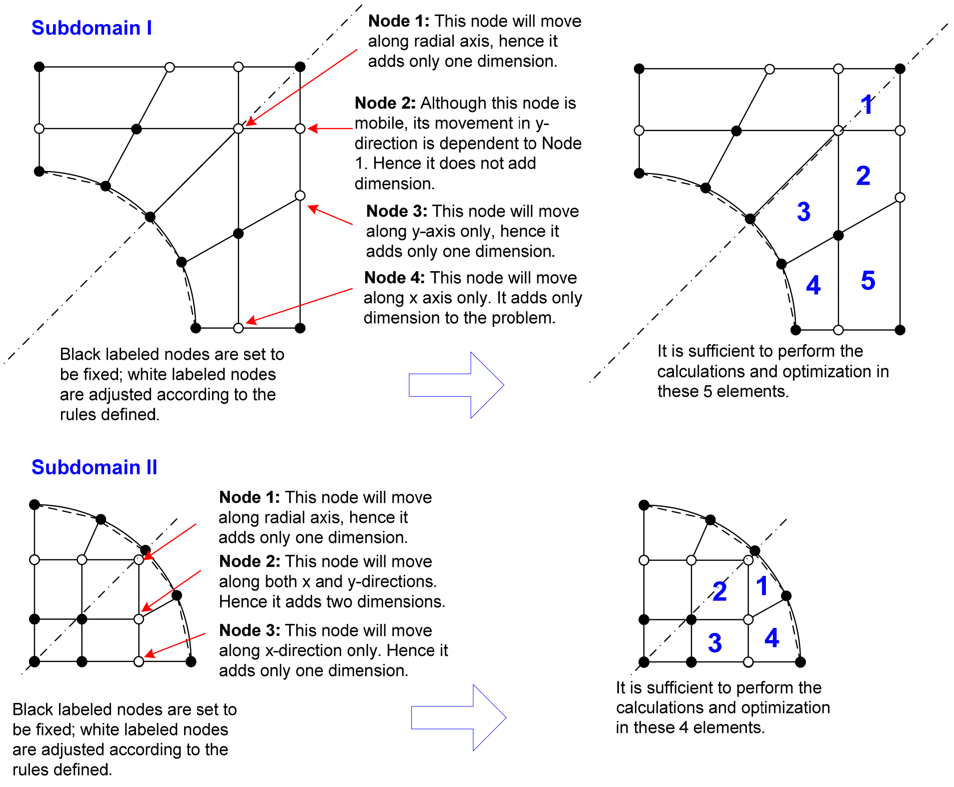

Dimension Reduction Using Symmetry
Domain Decomposition
More Dimension Reduction Using Symmetry
Subdomain I
More Dimension Reduction Using Symmetry
Subdomain II

Fig. D.7. Dimension Reduction by Means of Symmetry, and Domain Decomposition.



Node 1: This node will move along radial axis, hence it adds only one dimension.

Node 2: Although this node is mobile, its movement in y-direction is dependent to Node 1. Hence it does not add dimension.

Node 3: This node will move along y-axis only, hence it adds only one dimension.

Node 4: This node will move along x axis only. It adds only dimension to the problem.

Black labeled nodes are set to be fixed; white labeled nodes are adjusted according to the rules defined.

It is sufficient to perform the calculations and optimization in these 5 elements.

Node 1: This node will move along radial axis, hence it adds only one dimension.

Node 2: This node will move along both x and y-directions. Hence it adds two dimensions.

Node 3: This node will move along x-direction only. Hence it adds only one dimension.

Black labeled nodes are set to be fixed; white labeled nodes are adjusted according to the rules defined.

It is sufficient to perform the calculations and optimization in these 4 elements.

Fig. D.8. Fixing Nodes, Imposing Node Dependencies and Setting Rules for Individual Node Movements.

Another advantage of PSO, when applied to mesh smoothing problem, is the imposure of the boundary conditions. For a moving node, a reflecting boundary condition (wall) can be applied so that the relevant element is kept untangled. In order to keep the nodes inside a boundary (e.g. the boundary of the computational domain), absorbing or invisible walls can be defined on the boundaries as well. For the circular microstrip patch antenna problem, the reflecting walls defined for all floating nodes are illustrated in Fig. D.9.

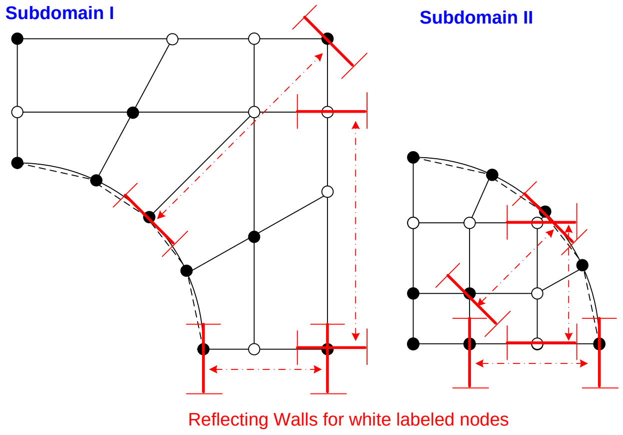
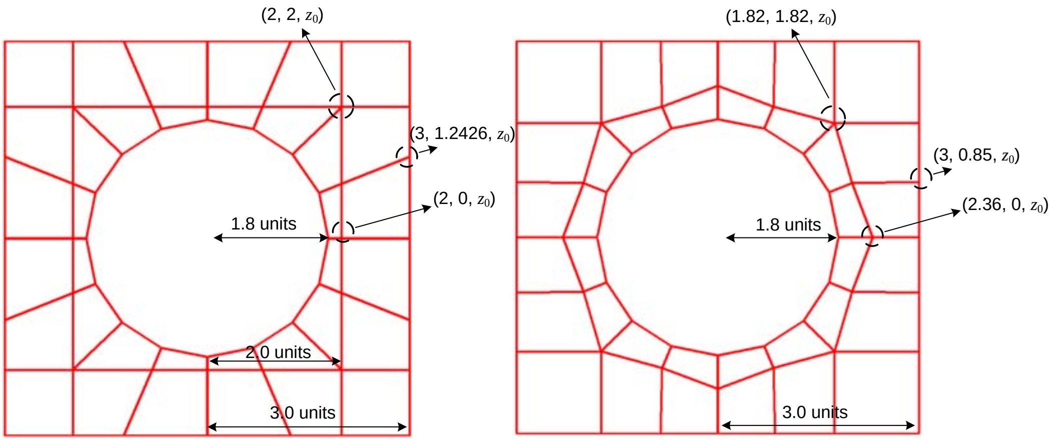


Fig. D.9. Setting Boundary Conditions (Reflecting Walls) for Floating Nodes.

The overall quality improvement in the circular microstrip patch antenna mesh improvement is achieved by means of two separate PSO schemes; a 2D scheme for Subvolume I (Fig. D.10), and a 2D scheme for Subvolume II (Fig. D.11). Finally, the whole procedure is summarized and illustrated in Fig. D.12.



F = 0.7560
$f_{average}$ = 0.7202

Fixed element size in z direction : 1.0 unit

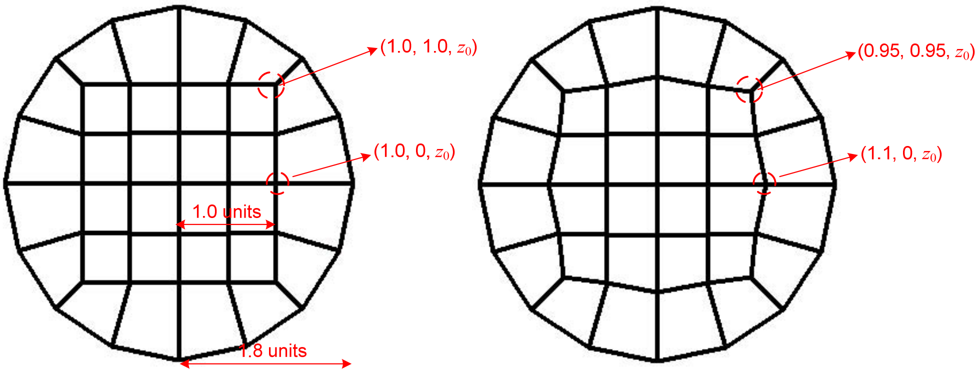
a) Automatically Generated Mesh

F = 0.3277
$f_{average}$ = 0.7967

Fixed element size in z direction : 1.0 unit

b) PSO Smoothed Mesh

Fig. D.10. A 3D PSO Mesh Smoothing Scheme for Subdomain I of the Circular Microstrip Antenna Problem.



F = 0.5543
$f_{average}$ = 0.6573

a) Automatically Generated Mesh

F = 0.5128
$f_{average}$ = 0.6692

b) PSO Smoothed Mesh

Fig. D.11. A 2D PSO Mesh Smoothing Scheme for Subdomain II of the Circular Microstrip Antenna Problem.

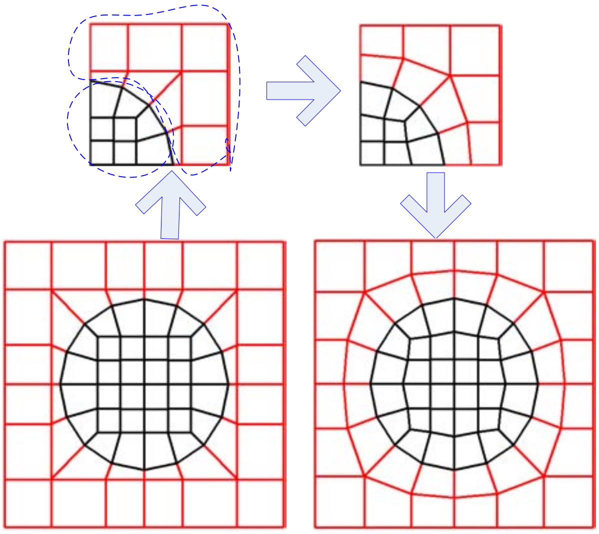


Fig. D.12. Performed Steps for a 2D+2D PSO Mesh Smoothing Scheme of the Circular Microstrip Antenna Problem.

Another problem dealt during the present work is the scattering of a perfectly conducting sphere with a radius of one wavelength (λ), which is an engineering electromagnetics application again. For this problem, it is certainly known that the total electric field inside the perfectly electric conducting sphere is zero. Hence, there is no need to consider the sphere; and there is no need to generate mesh for this volume. This means that the computational domain is a very thick spherical shell. For automatic all-hexahedral mesh generation, the computational domain shall be decomposed into three sub-domains; two top hats and the remaining surrounding region.

Automatically generated mesh for the top hat is usually in poor quality; of which the cross section for a constant R surface can be seen in Fig. D.13. Again, by getting use of symmetry the dimension of the objective function F can be reduced. Moreover, instead of trying to improve the whole top hat mesh, only one layer can be improved and then whole top hat can be reconstructed.

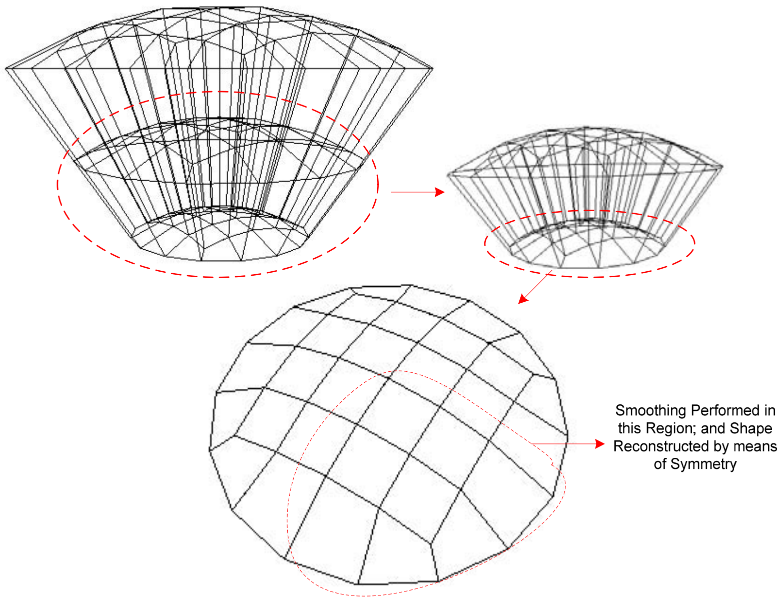
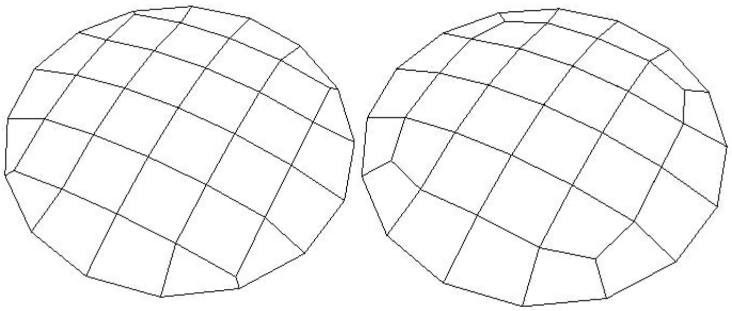


Fig. D.13. Top Hat Sub-Domain to be Smoothed by PSO (Different 3D Views with Different Levels of Detail).

The improvement in the top hat mesh can be seen in Fig. D.14 and Fig. D.15 with different views. It should be noted that this improvement is achieved by means of only a 3D PSO scheme.



a) Automatically Generated Mesh (3D Isometric View of a Constant *R* Surface)

b) PSO Smoothed Mesh (3D Isometric View of a Constant *R* Surface)

Fig. D.14. Improvement in the Mesh by Investigation of a Constant *R* Surface.

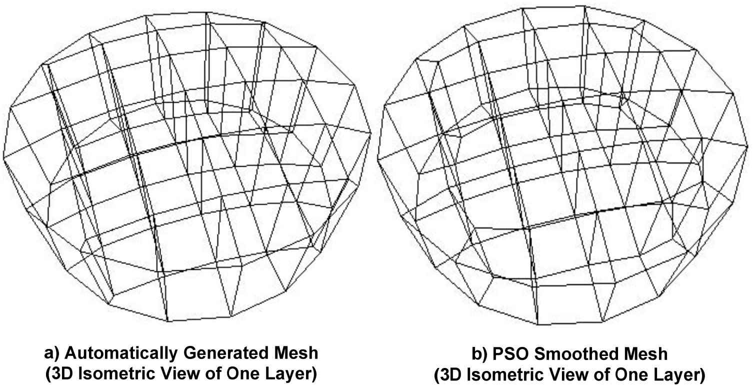

a) Automatically Generated Mesh
(3D Isometric View of One Layer)
b) PSO Smoothed Mesh
(3D Isometric View of One Layer)

Fig. D.15. Improvement in the Mesh by 3D View Investigation of One Layer of Top Hat.

Since this problem is an open domain problem, in order to be able to apply the Finite Element Method, an artificial absorber shall be defined for the simulation of infinity and mesh truncation. Perfectly Matched Layers (PMLs) defined by Berenger [97] is implemented in this work for this purpose by means of the complex coordinate stretching [98]. Hence, the cross section of the mesh is classified into regions as free space and PML as seen in Fig. D.16. The calculated electric field for the PML region is physically meaningless, and hence ignored throughout the error norm analysis described in the following paragraphs.

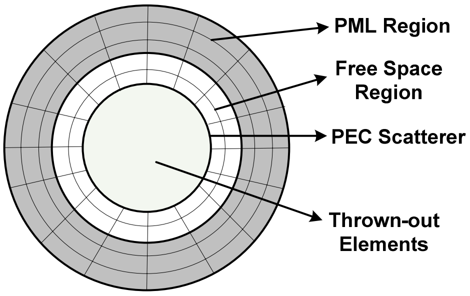


Fig. D.16. Cross Section of the Mesh Generated for the Perfectly Electric Conductor Sphere Problem.

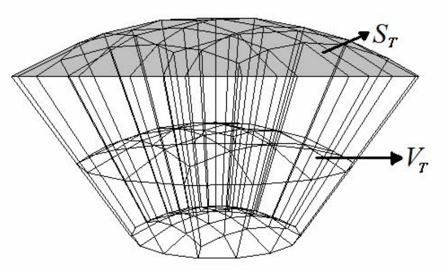


Fig. D.17. Surface and Volume Definitions of the Top Hat.

For this problem, the effect of the mesh quality on the Finite Element Solution accuracy is investigated. First, the exact area of the surface S_T of the top hat (as seen in Fig. D.17) has been compared to the calculated areas of the automatically generated and PSO smoothed meshes. In other words, for $\mathbf{G}'(x,y,z) = G'(x,y,z)\mathbf{a}_R$ where $G'(x,y,z) = 1$, the following surface integral is computed.

$$S_{exact} = \iint\limits_{S_T} \mathbf{G}'(x,y,z) \cdot \mathbf{ds} = \iint\limits_{S_T} (\mathbf{a}_R G'(x,y,z)) \cdot (\mathbf{a}_R ds) = \iint\limits_{S_T} ds \qquad \text{(D.14)}$$

By using the isoparametric hexahedral elements (i.e. assuming that each hexahedral element is transformed to a cube in $\xi\eta\zeta$-space extending from (-1,-1,-1) to (1,1,1)); for any function $G'(x,y,z)$, the surface integral on the surface of an element

$$\iint\limits_{S_e} G'(x,y,z)ds \qquad \text{(D.15)}$$

in the xyz-space can be stated as

$$\int_{-1}^{1}\int_{-1}^{1} G(\xi,\eta,\zeta)\left|\frac{\partial(x,y,z)}{\partial(\xi\eta)}\right| d\xi d\eta\,, \qquad \zeta\,\text{constant} \qquad \text{(D.16)}$$

in the $\xi\eta\zeta$-space. In (D.16),

$$\left|\frac{\partial(x,y,z)}{\partial(\xi\eta)}\right| = \left[\left(\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\right)^2 + \left(\frac{\partial z}{\partial \xi}\frac{\partial x}{\partial \eta} - \frac{\partial z}{\partial \eta}\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial z}{\partial \eta}\frac{\partial y}{\partial \xi}\right)^2\right]^{1/2} \tag{D.17}$$

Or in other words,

$$ds = \left|\frac{\partial(x,y,z)}{\partial(\xi\eta)}\right| d\xi d\eta, \qquad \text{where } \zeta \text{ is constant} \tag{D.18}$$

Certainly, $S_{calculated}$ can be stated as $\sum_n \iint_{S_e} G^{'}(x,y,z)ds$ where the summation traces all elements on the surface of the top hat. For the error in *S*, we define the following error norm:

$$err(S) = \frac{\left|S_{calculated} - S_{exact}\right|}{S_{exact}} \tag{D.19}$$

Second, the exact volume of the top hat (V_T as seen in Fig. D.17) has been compared to the calculated volumes of the automatically generated and PSO smoothed meshes. In other words, for $H'(x,y,z) = 1$, the following volume integral is computed.

$$V_{exact} = \iiint_{V_T} H'(x,y,z)dv = \iiint_{V_T} dv \tag{D.20}$$

By using the isoparametric hexahedral elements; for any function $H'(x,y,z)$, the volume integral in an element

$$\iiint_{V_e} H'(x,y,z)dv \tag{D.21}$$

in the *xyz*-space can be stated as

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} H(\xi,\eta,\zeta)\det(\mathbf{J})\,d\xi d\eta d\zeta \qquad \text{(D.22)}$$

in the $\xi\eta\zeta$-space, where **J** is the Jacobian matrix of the *xyz* to $\xi\eta\zeta$ transformation. Certainly, $V_{calculated}$ can be stated as $\sum_n \iiint_{V_e} H'(x,y,z)dv$ where the summation traces all elements inside the top hat. For the error in *V*, we define the following error norm:

$$err(V) = \frac{\left|V_{calculated} - V_{exact}\right|}{V_{exact}} \qquad \text{(D.23)}$$

A low quality mesh usually implies that the relevant surface mesh is also of low quality. This implies that both the surface and volume representations are bad; i.e. *err*(*S*) and *err*(*V*) values are high. On the other hand, having low *err*(*S*) and *err*(*V*) values does not guarantee mesh quality. These are just indicators about the low quality but not the high quality of a mesh. Hence, more reliable error norms should be defined if possible.

Finally, solution of the scattered electric field results obtained by both the automatically generated and the PSO smoothed meshes are compared to the analytical results, which can be found by using Mie Series. For the electric field, we define the following error norm:

$$err(\mathbf{E}) = \frac{1}{K}\sum_{i=1}^{K} \frac{\left\|\mathbf{E}_{calculated}(P_i) - \mathbf{E}_{exact}(P_i)\right\|}{\left\|\mathbf{E}_{exact}(P_i)\right\|} \qquad \text{(D.24)}$$

where $\mathbf{E}_{exact}(P_i)$ is the exact electric field calculated via the Mie Series at the centroid (P_i) of an element lying in free space; whereas $\mathbf{E}_{calculated}(P_i)$ is the value calculated by FEM at the same point. Certainly, the summation traces all elements lying in free space; *K* is the number of such elements; and *err*(**E**) is therefore the mean normalized error over the free space portion of the computational domain. Table D.3 demonstrates the improvement in the solutions (i.e. reduction in the error norms) after PSO smoothing.

Table D.3. Reduction in the error norms after smoothing.

		err(*S*)	*err*(*V*)	*err*(**E**)
3,400 total elements	Automatically Generated Mesh	0.0398	0.0336	0.0968
	PSO Smoothed Mesh	0.0367	0.0302	0.0881
28,800 total elements	Automatically Generated Mesh	0.0026	0.0022	0.0131
	PSO Smoothed Mesh	0.0024	0.0020	0.0117

A method for mesh improvement by means of local node repositioning based on the condition number related quality metric and Particle Swarm Optimization is proposed in this thesis. Meshes that can be encountered in practical situations are smoothed with this method. As an example, the impact of smoothing to the finite element solution accuracy is observed when H(*curl*)-conforming hexahedral elements are used. On the other hand, the method puts no restriction on the type of the hexahedral element; i.e. quality of any other hexahedral element type (H(*grad*)-conforming, H(*div*)-conforming) can be improved by means of this method.

Mesh improvement by means of PSO might be extended to other types of finite elements (e.g. triangular and quadrilateral elements in 2D, tetrahedral, prismatic elements in 3D). The method can also be extended to any type of element with higher order if applied to appropriate quality metrics or combined to appropriate validity criteria.

The application of PSO to mesh generation problem yields several research topics:

1. Development of methods for improving the convergence in this problem might be an attractive avenue. Since the *D.O.F* is very high in the mesh smoothing problem; and since getting the global extremum is not crucial as in other optimization problems, trying to improve the convergence (trading the convergence to accuracy where necessary) might be a good choice. The following section D.5 is dedicated to an introductory discussion about this topic.
2. Definition of several objective functions depending on various quality metrics; and application of multiobjective mesh smoothing by means of Multi Objective Particle

Swarm Optimization (MOPSO) [173-174] might be another further research topic; this will be introduced in section D.6.

Unfortunately, it could not be possible to obtain mature and meaningful results in either topic yet.

D.5. Speeding up the Convergence of PSO for High *D.O.F* Problems

It has been shown several times that the Particle Swarm Optimization method works well for challenging problems. In most of these works, the authors have not investigated the behavior of the particles, although they have tested the overall performance of the swarm by means of benchmark functions.

However recently in [175], Clerk and Kennedy discussed and investigated the success of the PSO with the particle's point of view; and they have proposed some modifications in the original algorithm in order to guarantee the stability and convergence in multidimensional problems. In this work, the authors modified the formulation of PSO in order to make the adjusted parameters controllable for guaranteeing convergence.

An idea might be:

1. To try to find a PSO derivative, which outperforms to the original one;
2. And then to modify the formulation of this derivative just as Clerk and Kennedy performed.

For the first step, the proposed PSO derivative should be first tested against the original PSO for some challenging problems. Since 1996, IEEE Congress on Evolutionary Computing has been trying to provide a standard test bench for the performance testing and evaluation of the proposed optimization schemes. The last set has been defined in 2005 at IEEE Congress on Evolutionary Computing. These benchmark functions provide challenges (with either numerous local optima, or wide plateaus) to optimization methods and researches.

Our proposal for the outperforming PSO derivative is NPSO of Yang and Simon [172]. As stated before, it basically tries to push each particle away from the worst location and bad locations. Therefore, for initial rapid convergence it seems to be reasonable according to the results given in [172], in which the authors used four benchmark functions (Sphere, Griewank, Rastrigin and Rosenbrock). The main properties of these functions are given in Table D.4.

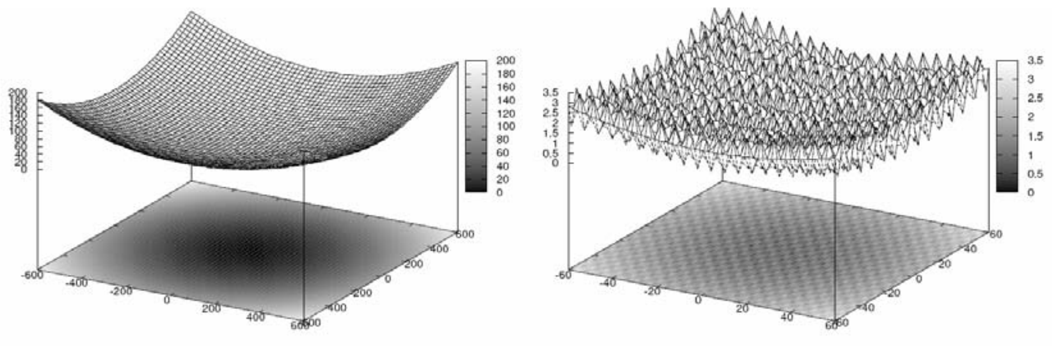
Table D.4. Main Properties of the Benchmark Functions Used in [172].

Function	**Multimodal**	**Separable**
Griewank	√	×
Rastrigin	√	√
Rosenbrock	×	×
Sphere	×	√

Among these, the Griewank function is defined as:

$$f(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad \text{(D.25)}$$

where $\mathbf{x} = [x_1 \ldots x_i \ldots x_n]$, and $x_i \in [-600.0, 600.0]$. The behavior of the function in 2D can be seen in Fig. D.18.



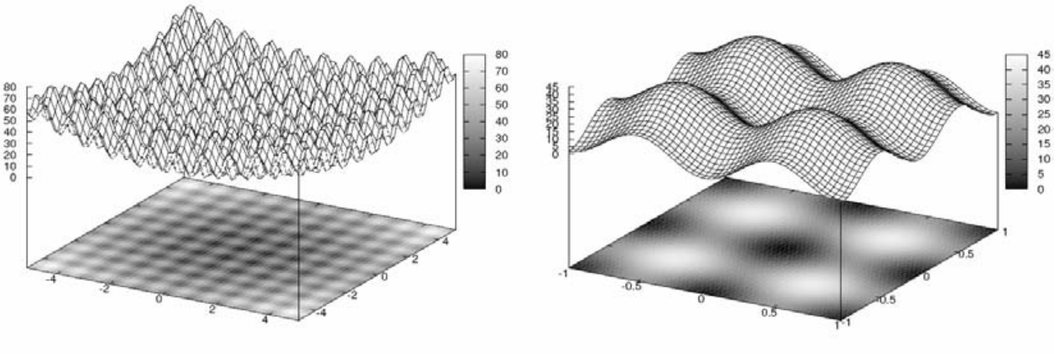
(a) Griewank (whole range) (b) Griewank (zoom)

Fig. D.18. Pictorial Description of the Behavior of 2D Griewank Function [176].

And the Rastrigin function is defined as:

$$f(\mathbf{x}) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)) \tag{D.26}$$

where $\mathbf{x} = [x_1 \ldots x_i \ldots x_n]$, and $x_i \in [-5.12, 5.12]$. The behavior of the function in 2D can be seen in Fig. D.19.



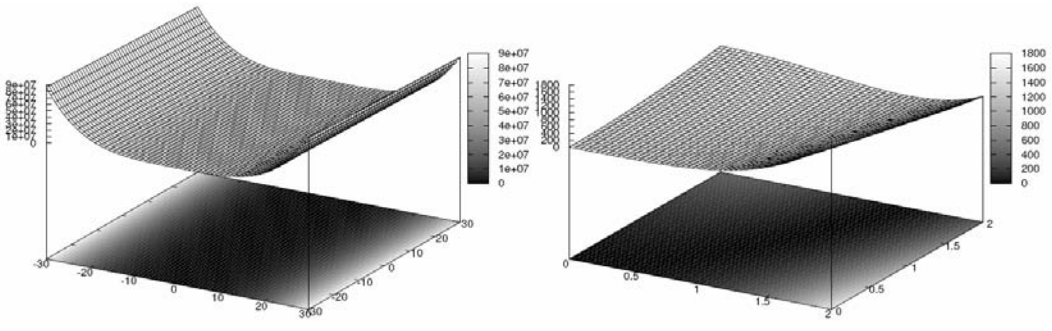
(a) Rastrigin (whole range)   (b) Rastrigin (zoom)

Fig. D.19. Pictorial Description of the Behavior of 2D Rastrigin Function [176].

The Rosenbrock function is defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \tag{D.27}$$

where $\mathbf{x} = [x_1 \ldots x_i \ldots x_n]$, and $x_i \in [-30.0, 30.0]$. The behavior of the function in 2D can be seen in Fig. D.20.



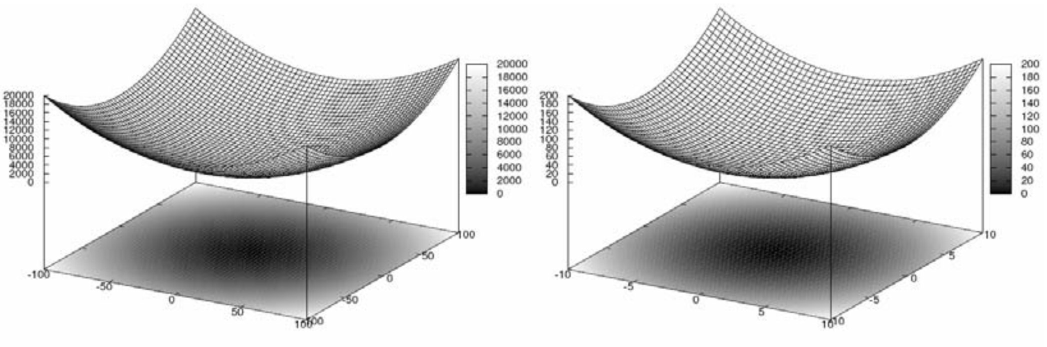
(a) Rosenbrock (whole range) (b) Rosenbrock (zoom)

Fig. D.20. Pictorial Description of the Behavior of 2D Rosenbrock Function [176].

Finally, the Sphere function is defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \qquad \text{(D.28)}$$

where $\mathbf{x} = [x_1 \ldots x_i \ldots x_n]$, and $x_i \in [-100.0, 100.0]$. The behavior of the function in 2D can be seen in Fig. D.21.



(a) Sphere (whole range) (b) Sphere (zoom)

Fig. D.21. Pictorial Description of the Behavior of 2D Sphere Function [176].

According to [172], for *D.O.F*s of 2, 5, and 10; NPSO outperforms (in both convergence and accuracy) to PSO for these 4 benchmark functions. Higher *D.O.F*s (1,000-10,000) should also be tested by using these functions. The comparison should also be performed for more challenging benchmark functions (e.g. Schaffer's *F*6); and benchmark functions with wide plateaus (e.g. Easom) for which PSO is well known to be suffering.

D.6. Pareto Optimality in Mesh Smoothing and Application of MOPSO

Definition 1. Given two vectors $\mathbf{x}; \mathbf{y} \in R^k$, we say that $\mathbf{x} \le \mathbf{y}$ if $x_i \le \cdot y_i$ for $i = 1, ..., k$, and $\mathbf{x}$ dominates $\mathbf{y}$ (denoted by $\mathbf{x} \prec \mathbf{y}$) if $\mathbf{x} \le \mathbf{y}$ and $\mathbf{x} \ne \mathbf{y}$.

Fig. D.22 shows a particular case of the dominance relation in the presence of two objective functions.

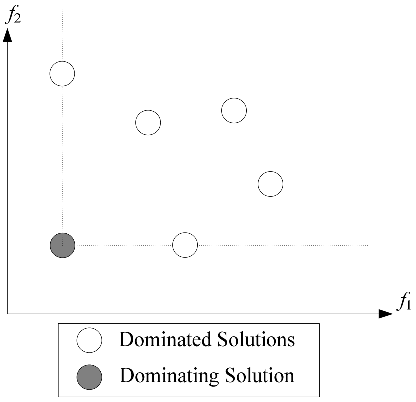


Fig. D.22. Dominance Relation in a Biobjective Space.

Definition 2. We say that a vector of decision variables $\mathbf{x} \in X \subset R^k$ is nondominated with respect to X, if there does not exist another $\mathbf{y} \in \chi$ such that $\boldsymbol{f}(\mathbf{y}) \prec \boldsymbol{f}(\mathbf{x})$.

Definition 3. We say that a vector of decision variables $\mathbf{x}^* \in F \subset R^k$ (F is the feasible region) is Pareto-optimal if it is nondominated with respect to F.

Definition 4. The Pareto Optimal Set P^* is defined by: $P^* = \{\boldsymbol{f}(\mathbf{x}) \in \mathrm{R}^k \mid \mathbf{x}$ is Pareto-optimal$\}$.

Definition 5. The Pareto Front PF^* is defined by: $PF^* = \{\boldsymbol{f}(\mathbf{x}) \in \mathrm{R}^k \mid \mathbf{x} \in P^*\}$.

Fig. D.23 shows a particular case of the Pareto front in the presence of two objective functions.

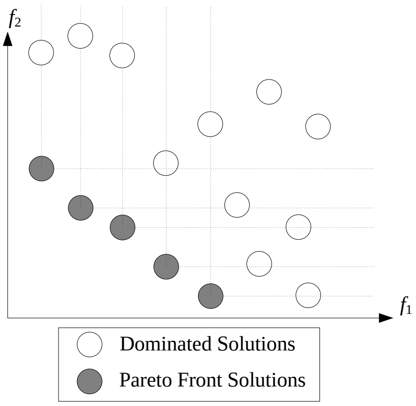


Fig. D.23. The Pareto Front of a Set of Solutions in a Biobjective Space.

Hexahedral mesh smoothing can be extended by defining another metric (i.e. another objective function). Since the element size is a dominating factor in interpolation accuracy, the second objective function can be defined as the maximum deviation from the “ideal element volume”; where the maximization is over the all elements inside the mesh. Keeping the condition number based objective function as the first objective, putting a second objective like this, and using the Pareto definitions above; a Pareto front can be found. This might be the subject of further research.

APPENDIX E

BASIC NOTATION OF THE UNIFIED MODELING LANGUAGE (UML)

In this Appendix, basic notation of Unified Modelling Language is given to the readers who are not familiar with the concept.

E.1. What is UML?

According to the OMG specification: "*The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components."*

The important point to note here is that UML is a 'language' and not a method or procedure. The UML is used to define a software system; to detail the artifacts in the system, to document and construct - it is the language that the blueprint is written in. UML defines the notation and semantics for the following domains:

- The User Interaction or Use Case Model - describes the boundary and interaction between the system and users. Corresponds in some respects to a requirements model.
- The Interaction or Collaboration Model - describes how objects in the system will interact with each other to get work done.

- The Dynamic Model - State charts describe the states or conditions that classes assume over time. Activity graphs describe the workflows the system will implement.
- The Logical or Class Model - describes the classes and objects that will make up the system.
- The Physical Component Model - describes the software (and sometimes hardware components) that make up the system.
- The Physical Deployment Model - describes the physical architecture and the deployment of components on that hardware architecture.

Table E.1. Core Elements in Structural UML Modeling.

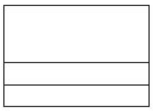
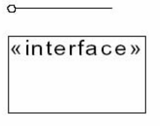
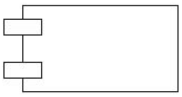
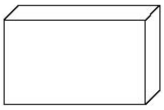
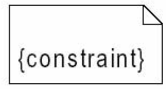
Construct	**Description**	**Syntax**
Class	A description of a set of objects that share the same attributes, operations, methods, relationships and semantics.	
Interface	A named set of operations that characterize the behavior of a software element.	
Component	A modular, replaceable and significant part of a system that packages implementation and exposes a set of interfaces.	
Node	A run-time physical object that represents a computational resource.	
Constraint	A semantic condition or restriction.	

Table E.2. Core Relationships in Structural UML Modeling.

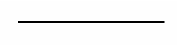
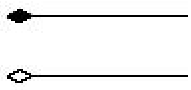
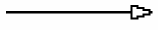
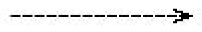
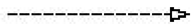
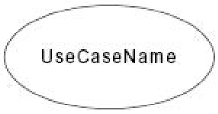
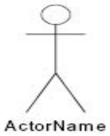
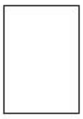
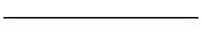
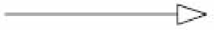
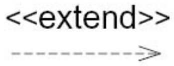
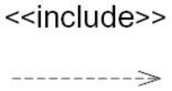
Construct	**Description**	**Syntax**
Association	A relationship between two or more classifiers that involves connections among their instances.	
Aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	
Generalization	A taxonomic relationship between a more general and a more specific software element.	
Dependency	A relationship between two modeling elements, in which a change to one modeling element (the independent software element) will affect the other modeling element (the dependent software element).	
Realization	A relationship between a specification and its implementation.	

Table E.3. Core Definitions in UML Use Case Modeling.

Construct	**Description**	**Syntax**
Use Case	A sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system.	
Actor	A coherent set of roles that users of use cases play when interacting with these use cases.	
System Boundary	Represents the boundary between the physical system and the actors who interact with the physical system.	
Association	The participation of an actor in a use case. i.e., instance of an actor and instances of a use case communicate with each other.	
Generalization	A taxonomic relationship between a more general use case and a more specific use case.	
Extend	A relationship from an *extension* use case to a *base* use case, specifying how the behavior for the extension use case can be inserted into the behavior defined for the base use case.	
Include	An relationship from a *base* use case to an *inclusion* use case, specifying how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.	

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name	: Yılmaz, Asım Egemen
Nationality	: Turkish (TC)
Date and Place of Birth	: 18 October 1975, Adıyaman
Marital Status	: Married
Phone	: +90 312 219 57 87 / 3120
Fax	: +90 312 219 57 97
e-mail	: aeyilmaz@havelsan.com.tr, asimegemenyilmaz@yahoo.com

EDUCATION

Degree	**Institution**	**Year of Graduation**
MS	METU Electrical and Electronics Engineering	2000
BS	METU Electrical and Electronics Engineering	1997
BS	METU Mathematics	1997
High School	Ankara Fen Lisesi	1993

TEACHING EXPERIENCE

Year	**Place**	**Enrollment**
September 2004- Present	Ankara University Department of Electronics Engineering	Guest Instructor (EM206 Electromagnetics I) (EM311 Electromagnetics II) *Formerly* *(EM206 Elektromanyetik I)* *(EM311 Elektromanyetik II)*

WORK EXPERIENCE

Year	**Place**	**Enrollment**
July 2003- Present	HAVELSAN A.Ş. 737 AEW&C Peace Eagle Program	Mission Computing Subsystem IPT Manager
July 1999- July 2003	STM A.Ş. HERİKKS Project	R&D Engineer
February 1999- July 1999	Vestel Information Appliances	R&D Engineer
July 1997- January 1999	ASELSAN A.Ş. Communications Division Communication Security Group	R&D Engineer
September 1996- June 1997	ASELSAN A.Ş. Communications Division Communication Security Group	Part Time R&D Engineer

FOREIGN LANGUAGES

Advanced English, Fluent German

PUBLICATIONS

1. A.E. Yilmaz, M. Kuzuoglu, “Calculation of Optimized Parameters of Rectangular Microstrip Patch Antenna Using Particle Swarm Optimization”, to appear in *Microwave and Optical Technology Letters* (submitted in May 2007).
2. A.Ö. Bozdoğan, M. Efe, A.E. Yılmaz, “Genel Atama Probleminde Koloni Optimizasyon Yaklaşımları”, in *CD-ROM Proc. IEEE 15. Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SİU-2007)*, Eskişehir, Türkiye.
3. A.E. Yılmaz, M. Kuzuoğlu, “Parçacık Sürü Optimizasyonu ile Altı Yüzlü Eleman Ağlarının İyileştirilmesi”, in *CD-ROM Proc. IEEE 15. Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SİU-2007)*, Eskişehir, Türkiye.
4. M.B. Akbulut, A.E. Yılmaz, “A Modified Genetic Algorithm for the Generalized Assignment Problems”, submitted to *IU-JEEE (Istanbul University – Journal of Electrical and Electronics Engineering)* (Feb. 2007).
5. A.E. Yilmaz, M. Kuzuoglu, “Comparison of Linear And Quadratic Hexahedral Edge Elements in Electromagnetic Scattering Problems”, to appear in *AEÜ – International Journal of Electronics and Communications* (submitted in Dec. 2006).
6. A.E. Yilmaz, M. Kuzuoglu, “A Particle Swarm Optimization Approach in Hexahedral Mesh Smoothing”, submitted to *Communications in Numerical Methods in Engineering* (Dec. 2006 – rev. May 2007).
7. A.E. Yılmaz, M. Kuzuoğlu, “Mikroşerit Yama Antenlerin Sonlu Elemanlar Yöntemi ile Modellenmesi”, in *Proc. URSI-Türkiye 2006 3. Ulusal Kongresi*, pp. 146-148, 6-8 Eylül 2006, Ankara, Türkiye.
8. A.E. Yılmaz, M. Kuzuoğlu, “Elektromanyetik Sınır Değer Problemlerinin Düzgün Olmayan Ağlar ve Altı Yüzlü Sonlu Kenar Elemanları İle Modellenmesi”, in *Proc. URSI-Türkiye 2006 3. Ulusal Kongresi*, pp. 125-127, 6-8 Eylül 2006, Ankara, Türkiye.
9. İ. Kılınç, A.A. Diri, A.E. Yılmaz, “Komuta Kontrol Sistemlerinde Üç Boyutlu Hava Resmi Görüntüleme Teknolojileri”, in *Proc. 3. Savunma Teknolojileri Kongresi (SAVTEK-2006)*, vol. 2, pp. 231-241, 29-30 Haziran 2006, Ankara, Türkiye.
10. A.R. Ünal, G. Karaca, Ö. Dura, A.E. Yılmaz, “Taktik Veri İşleme ve Dağıtım Sistemi”, in *Proc. 3. Savunma Teknolojileri Kongresi (SAVTEK-2006)*, vol. 1, pp. 613-623, 29-30 Haziran 2006, Ankara, Türkiye.
11. M. Efe, A.E. Yilmaz, O. Donmez Dura, “Data Fusion for a Surveillance System: Addressing Some Practical Problems”, in *Proc. 18th International Conference on Systems Engineering (ICSENG-05)*, pp. 342-247, August 16-18, 2005, Las Vegas, Nevada, USA.
12. E. Çağlav, H.G. İlk, R.M. Özel, A.E. Yılmaz, “Karar Destek Mekanizmalarında Özellik Füzyonu için Çoklu Sensor Entegrasyonu” , in *CD-ROM Proc. IEEE 13. Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SİU-2005)*, Kayseri, Türkiye.
13. A.E. Yılmaz, M. Kuzuoğlu, “Elektromanyetik Sınır Değer Problemlerinin İkinci Dereceden Altı Yüzlü Sonlu Elemanlar İle Modellenmesi”, in *Proc. URSI-Türkiye 2004 2. Ulusal Kongresi*, pp. 78-80, 8-10 Eylül 2004, Ankara, Türkiye.
14. A.E. Yılmaz, R.M. Özel, H.G. İlk, “Çoklu Taktik Ses ve Veri Haberleşme Linklerinin Analizi ve Modellenmesi”, in *Proc.IEEE 12. Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SİU-2004)*, pp. 763-766, 28-30 Nisan 2004, Kuşadası, Türkiye.
15. E.H. Kök, A.E. Yılmaz, Ö. Yıldız, M. Efe, R.M. Özel, “Dağınık Konuşlandırılmış Sensör İzlerinin Birleştirilmesinde Kayıtlanma Problemleri ve Çözümleri”, in *Proc. IEEE 12. Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SİU-2004)*, pp. 755-758, 28-30 Nisan 2004, Kuşadası, Türkiye.

16. E.H. Kök, A.E. Yılmaz, Ö. Yıldız, M. Efe, R.M. Özel, “Heterojen Radarlardan Gelen İzlerin Birleştirilmesi: Pratik Problemler ve Çözümleri”, in *Proc. IEEE 12. Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SİU-2004)*, pp. 35-40, 28-30 Nisan 2004, Kuşadası, Türkiye.

PATENTS

M.A. Yazıcı and A.E. Yılmaz, inventors; ASELSAN A.Ş., assignee. “X İlk Koşullu Artan/Azalan Örüntü Eleyici Yapma Rassal Sayı Üretme Metodu”, Turkish Patent Institute No: 1998 00950, 2001/12/21.

HONORS & AWARDS

- 3rd, Student Proceedings Award, URSI-2004, Ankara, Turkey.
- 1st patent holder of ASELSAN; 1st software patent holder in Turkey.

MEMBERSHIPS

- Member (’05) – International Council on Systems Engineering (INCOSE)
- Founding Member (’05) – International Council on Systems Engineering (INCOSE) Turkey Chapter
- Student Member (’07) – Institute of Electrical and Electronics Engineers (IEEE)

OTHER ACADEMIC ACTIVITIES

- Guest Referee, Turkish Journal of Electrical Engineering and Computer Sciences (ELEKTRİK) – *upon Invitation or Request.*
- Speaker on Career & Training Days of various universities such as Ankara University, TOBB-ETU, Selçuk University, and Bilkent University – *upon Invitation or Request.*
- Organization and Chair of the Special Session “Tactical Environment Modelling, Target Tracking and Applications” in SİU-2006, Antalya, Turkey (co-organized and co-chaired with M. Efe).
- Organization and Chair of the Special Session Series “Evolutionary Algorithms: Theory and Applications (I–II)” in SİU-2007, Eskişehir, Turkey (co-organized and co-chaired with M. Efe).

NON PROFESSIONAL ACTIVITIES

- Amateur Photography:
  Photos published in various media such as AFSAD Monthly Bulletin - Kontrast, AFSAD Web Page, Cumhuriyet-Ankara, and Ankara Chamber of Dentists Bulletin
  Personal Galleries:
  www.treklens.com/members/egemenyilmaz/,
  www.trekearth.com/members/egemenyilmaz/

- Performing Arts:
  (Mar 2007 – present) Back vocalist, harmonica player, and keyboardist/pianist of the Pop/Rock cover band *A Broad Band*™
  (Jun 2004 – Mar 2007) Lead singer and keyboardist/pianist of the Pop/Rock cover band *Plug N’ Play*™
  (Jul 1999 – May 2001) Keyboardist/pianist of the Ethnic Jazz experimental band *The Eye Queue Project*™

PERSONAL INTERESTS

History of Music (Romantic Era), History of Art (Impressionist & Post-Impressionist Era)