A Computationally Economic Three Dimensional Magnetic Modelling System

by

Munna Mishra, B. Eng., M. Tech., M. Eng.

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the Requirements for the degree of Doctor of Philosophy

Department of Electrical Engineering
McGill University
Montréal, Canada
June, 1985

A Computationally Economic Three Dimensional Magnetic Modelling System

by

Munna Mishra, B. Eng., M. Tech., M. Eng
Department of Electrical Engineering

McGill University
Montréal, Canada
June, 1985

To

Cèline Bernatchez

Abstract

This thesis develops a computationally economic magnetic modelling system for three dimensional magnetostatic problems. An interactive mesh generator is developed to model geometries using irregular bricks which are finally divided into either tetrahedra or triangular prisms. A new algorithm is introduced for element resequencing to minimize wavefronts for the frontal solutions in $O(mMN)$ comparisons where m, M and N are, respectively, nodes in an element, number of nodes, and number of elements in a mesh. The algorithm which is tested on topologically different models also renumbers the nodes to reduce profiles and bandwidths of coefficient matrices and needs primary storage of $O(1)$. Two algorithms, one 'soft-failing' and the other 'fast converging and robust', for solving systems of linear algebraic equations are developed by combining the advantages of the frontal and the preconditioned conjugate gradient algorithms. A scan conversion technique is implemented to plot the equipotentials on the planes derived from the mesh. The developed system is applied to a terminal box model.

Résumé

Cette thése présente un approche intégrée et numeriquement économique á la solution de problémes de magnétostatique tridimensionnels. Un maillleur interactif a été développé afin de modéliser des géométries solides en utilisant des briques irréguliéres qui sont elles-mémes formées de tétrahédres ou encore de prismes triangulaires. Un nouvel algorithme permettant de réordonner les éléments afin de minimiser la largeur des fronts en vue dúne solution frontale et nécessitant $O(mMN)$ comparaisons a été developpé. Ici, m, M et N représentent respectivement le nombre de noeuds associés á un élément, le nombre total de noeuds et le nombre total d' éléments. Cet algorithme, qui été validé sur des modéles de topologies variées permet a aussi de renuméroter les noeuds d'interpolation de maniére á réduire la largeur de bande et la ligne de ciel de la matrice globale. Il requiert une quantité de mémoire vive d'order $O(1)$. Deux algorithmes de résolution de systémes d' équations algébriques linéaires, l' un de type "soft failing"et l'autre robuste et rapidement convergent ont été développés en combinant les avantages de la méthode frontale et de la méthode des gradients conjugués avec préconditionnement. Finalement, une technique dite de "conversion par balayage"a été appliquée au tracage des équipotentielles sur des plans de coup. L' approche intégrée décrite ci-haut a éte appliquée á un modéle de bornier á grande puissance.

Acknowledgements

I would like to thank Prof. D. A. Lowther, my thesis supervisor for his invaluable guidance and help throughout the period of this study. I am also grateful to Prof. P. P. Silvester for suggesting the terminal box problem and for introducing me to the preconditioned conjugate gradient method.

Gratitude is also extended to Geoff Stone for providing me with his element sorting programs, and to Greg Cambrell with whom I had numerous fruitful discussions.

Many of my colleagues helped a great deal in the lab and in putting this thesis together. Among them I thank Jean Francois Ostiguy, Robert Kotiuga, Nevine Nassif, Behzad Forghani, Yvan Leclerc and Wade Hong.

I am most grateful to my parents and the family for their constant encouragement and patience.

Financial support in the form of a research assistantship from the Natural Sciences and Engineering Research Council is gratefully acknowledged.

Table of Contents

	Page
Abstract	i
Résumé	ii
Acknowledgements	iii
Table of Contents	iv
List of Symbols	ix
Chapter 1: Introduction	1
1.1 General	1
1.2 Finite Elements for Three Dimensional Problems	4
1.2.1 Magnetic Vector Potential	6
1.2.2 Scalar Potentials	8
1.2.3 Integral Equations	11
1.3 Eddy Current Computations in Three Dimensions	13
1.3.1 The **T**-Ω and **A**-ϕ Methods	14
1.3.2 R-ψ Formulation	16
1.3.3 The Emson and Simkin Formulation	18
1.3.4 The Polak et al. Formulation	19
1.3.5 The Pillsbury Formulation	21
1.4 The Terminal Box - A Practical Problem	22

1.4.1 General Considerations 24

1.4.2 Which Potential ? 25

1.4.3 Pre-Processing: Mesh Generation in 3D 26

1.4.4 Solution of Algebraic Equations 28

1.4.4.1 The Frontal Method 28

1.4.4.2 The Element Sorting and Node Renumbering 29

1.4.4.3 Preconditioned Conjugate Gradients 30

1.4.5 Post-Processing 31

1.5 Original Contributions 32

1.6 Thesis Organization 34

Chapter 2: Mathematical Formulation 36

2.1 Introduction 36

2.2 The Scalar Potential Equation 37

2.3 Boundary Conditions 38

2.3.1 Current Carrying Conductors 38

2.3.2 Heavy Eddy Currents Flowing on the walls 39

2.3.3 The Walls Carry Eddy Currents of Finite Magnitude 41

2.4 Element Matrices for Sheet Elements 43

2.5 Treatment of the Sheet Edges 47

Chapter 3: Mesh Generation 50

3.1 Introduction 50

3.2 Desirable Features of an Automatic Mesh Generator 52

3.3 Mesh Generation Techniques in Structural Mechanics 53

3.4 3D Mesh Generators 56

3.5 Mesh Generators at the Research Level 56

3.6 Need for Developing a Mesh Generator 60

3.6.1 Choice of Elements 61

3.6.2 Mesh Generation Philosophy and Distinct Features 63

3.6.3 System Requirements 63

3.6.4 Input/Output Features (Data Structures) 63

3.6.5 Memory and CPU Time Requirements 69

3.7 Practical Experience 70

3.8 Limitations 71

Chapter 4: Element Reordering For Frontal Solutions 72

4.1 The Frontal Method 72

4.2 The Need for Element Sorting 76

4.3 A Summary of Existing Method 77

4.4 The New Method 81

4.4.1 The Algorithm 82

4.4.2 Comments on the Algorithm 85

4.4.3 Memory and Time Requirements 85

4.4.4 Data Structures 87

4.5 Numerical Experiments 87

4.6 On the Algorithm Performance 111

Chapter 5: Preconditioned Conjugate Gradient Frontal Method 113

5.1 Introduction 113

5.2 Conjugate Gradients 114

5.3 Preconditioned Conjugate Gradients 116

5.4 The Preconditioned Conjugate Gradient Algorithm 117

5.5 Preconditioned Conjugate Gradient Frontal Method 119

5.5.1 Frontal Gaussian Elimination 119

5.5.2 Incomplete Cholesky Decomposition 120

5.5.3 Implementation of PCGF Algorithm 121

5.5.4 The Soft-Failing Algorithm 122

5.5.5 Numerical Results 123

5.5.6 Discussion 124

5.6 Modified ICCG Frontal Method 124

5.6.1 General 124

5.6.2 A Modified Cholesky Decomposition 132

5.6.3 Computational Problems Involving Air-Iron Interfaces 135

5.7 Numerical Results 136

5.8 Conclusions 137

Chapter 6: Post-Processing 142

6.1 Introduction 142

6.2 Properties of the Data 143

6.3 Valid Post-Processing Operations 144

6.4 Displays 146

6.5 Post-Processor Structure 149

6.5.1 Algorithmic Features 150

6.5.2 Memory Requirements 152

6.5.3 Response Time 154

6.6 Derived Quantities 154

6.6.1 Surface Current Densities 155

6.6.2 Power Loss 156

6.6.3 Forces 157

6.7 Equipotential Contour Plots 157

6.7.1 General 157

6.7.2 The Scan Conversion Method 158

6.7.2.1 Formulation 160

6.7.2.2 Reduction from 3D to 2D 163

6.7.2.3 Plotting 164

6.8 An Application 166

Chapter 7: Conclusions 168

7.1 Recommendation for Future Work 173

References 175

Appendix I 188

Appendix II 190

List of Symbols

A	Magnetic Vector Potential
B	Magnetic Flux Density Vector
H	Magnetic Field Intensity Vector
E	Electric Field Intensity Vector
D	Electric Flux Density Vector
J	Electric Current Density Vector
ρ	Electric Charge Density
μ	Permeability of the Magnetic Material
μ_0	Permeability of Free Space
ε	Permitivitty of the material
σ	Conductivity of the material
ν	Reluctivity of the Magnetic Material
ν'	Lagrange Multiplier
φ	Reduced Scalar Potential or Electric Scalar Potential
Ω	Magnetic Scalar Potential or Total Scalar Potential
M	Induced Volume Magnetization
χ	Magnetic Susceptibility
T	Electric Vector Potential
$\mathbf{A}^*$	Vector Potential defined on page 18
ϕ_m	Scalar Potential defined on page 20
ϕ_c	Scalar Potential defined on page 20
φ_f	Scalar Potential defined on page 20
δ	Depth of Penetration
α	Interpolation Functions
R	Cartesian Product of Interpolation Functions

Chapter 1

Introduction

1.1 General

Modern power systems place stringent requirements on their basic generation and transmission equipment with regard to economical, efficient, and reliable operation. These machines operate at high power densities and it is of the utmost importance that their performance characteristics are accurately predicted at the design stage so that final designs can be optimized. The accurate prediction of machine parameters such as reactances, power losses, heating of various components due to these losses, and the forces on the various structural parts of the machine requires a knowledge of the electromagnetic field distribution in the entire geometry.

Over the last twenty five years the power densities in large turbogenerators have increased, approximately, by a factor of four, while there has been no significant increase in physical size. This can be attributed to the utilization of materials with better magnetic and electric properties, better insulation systems, and more efficient cooling systems. However, this rapid growth in generating capacities has placed considerable demands on equipment designs. Issues which were once considered insignificant for small machines have surfaced as critical problems in large ones, e.g. the surface current distribution, associated losses

and forces on the walls of the terminal boxes of the turbogenerators and transformer tanks, end region fields in large machines, and forces on the windings and other structures. Surface currents are induced on the walls of the terminal box because of the time varying fields established around the box by the load currents of several kilo-amps., flowing through the terminals.

Conventionally, field computations in magnetics have been performed using analytical and analogue methods. Application of these approaches was possible under simplified conditions and for idealized geometries. For electromagnetic devices, the geometries involved were such that it was impossible to provide analytical solutions for a wide range of problems without considerable simplifications. With the advent of high speed computers, an era of numerical methods began which made feasible the solution of some of these electromagnetic field problems. Earlier methods for the numerical solution of field problems employed either finite differences, integral equation methods, or variational methods.

In the finite difference method the differential equation is replaced by a system of simultaneous algebraic equations which are derived by approximating the derivatives with respect to one or more of the variables by difference quotients. The method of finite differences has been extensively used in electromagnetic applications even though three disadvantages are commonly cited against it. The first one arises if curved boundaries are to be modelled with a rectangular mesh. This problem can be overcome by using graded meshes, which in turn introduce a large number of variables to be solved for and the cost of solution grows enormously. The second limitation concerns the need to specify the boundary conditions at material interfaces and the boundaries by a set of equations. Also, finite difference equations resulting from the discretization of regions consisting of different material properties often result in poor convergence when solved iteratively, even if suitable accelerating schemes are used.

The integral equation method, on the other hand, is based on the fact that the field quantity at a point can be expressed as the sum of contributions from all sources. In this case only conducting regions are to be discretized, as opposed to the entire region in the case of differential method, thus there is considerable saving in the cost of input data preparation (it could be up to an order of magnitude). Nevertheless, integral equation methods result in unsymmetric and dense coefficient matrices, which not only have larger formation times but also the number of operations required to solve these linear or non-linear equations is of the order of N^3, where N is the number of unknowns.

The finite element method is a variational approach whose differential formulation frequently results in symmetric positive definite sparse matrices which can be solved in relatively fewer operations. Earlier engineering applications of the method were in the area of stress analysis but it was Silvester who first pioneered the FEM approach in electromagnetics in the late sixties [137], [144]. The finite element method permits easy modelling of curved and irregular geometries and detailed modelling for regions of high field intensity or of special interest. Furthermore, the standard formulations do not require imposition of homogeneous Neumann boundary conditions, as this occurs as a natural boundary condition in the formulation.

Over the last fifteen years, the use of the finite element method has grown considerably and it has been applied to solve linear, non-linear, and eddy current problems in magnetostatics using two dimensional representations in the $x-y$ or $r-z$ planes. Two dimensional solutions are easy to compute and are considered adequate for a wide range of field problems. There remains, however, a class of problems which is not amenable to two dimensional approximations of either the geometries or the fields, and needs three dimensional treatment. Two dimensional representations are inadequate if, for example, the effects of ventilating ducts and the end winding leakage fluxes are to be considered si-

multaneously, although some progress may be made by considering sets of two-dimensional solutions.

1.2 Finite Elements for Three Dimensional Problems

In general, solutions of field problems in two dimensions have been mostly obtained using a vector potential formulation which adequately treats regions containing sources and has only one component, i.e. it acts as a scalar. In three dimensions, however, there are various potential formulations, namely vector potential, total scalar potential, and hybrid formulations, which use a combination of the two potentials. These formulations arose due to the non-uniqueness of the vector potential alone and a requirement to reduce the number of degrees of freedom per node in order to make solutions feasible on the available computing hardware.

The basic numerical techniques for three dimensional problems have been either integral methods or differential methods as finite differences have not been considered satisfactory for general problems, although a few successful attempts have been made by Muller and Wolff [113], who employed a scalar potential representation for the magnetic field, and Carpenter [25], [27], who formulated the $\mathbf{T} - \Omega$ and $\mathbf{A} - \Phi$ methods for magnetostatics and eddy current problems. $\mathbf{T}$ and $\mathbf{A}$ are, respectively electric vector and magnetic vector potentials, whereas Ω and Φ should be understood as magnetic scalar and electric scalar potentials.

In this section, the major potential formulations currently in use will be discussed. These have been derived from Maxwell's equations and the constituting relations and can be summarized in their differential form as follows, with reference to quasi-static fields:

$$\nabla \times \mathbf{H} = \mathbf{J} \tag{1.2.1}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{1.2.2}$$

$$\nabla \cdot \mathbf{D} = \rho \tag{1.2.3}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{1.2.4}$$

The constituting relations are:

$$\mathbf{B} = \mu \mathbf{H} \tag{1.2.5}$$

$$\mathbf{D} = \varepsilon \mathbf{E} \tag{1.2.6}$$

$$\mathbf{J} = \sigma \mathbf{E} \tag{1.2.7}$$

Here:

H is the magnetic field intensity vector.

B is the magnetic flux density vector.

E is the electric field intensity vector.

D is the electric flux density vector.

J is the electric current density vector.

ρ is the electric charge density.

μ is the permeability of the magnetic material.

ε is the permittivity of the material.

σ is the conductivity of the material.

1.2.1 Magnetic Vector Potential

Choosing a vector potential function **A** such that

$$\mathbf{B} = \nabla \times \mathbf{A} \tag{1.2.1.1}$$

and substituting it in equation (1.2.1) with (1.2.5), one gets

$$\nabla \times \nu \nabla \times \mathbf{A} = \mathbf{J} \tag{1.2.1.2}$$

where $\nu = \frac{1}{\mu}$, is the reluctivity of the magnetic material. Equation (1.2.1.2) is known as the magnetic vector potential formulation or the curl-curl equation for three dimensional magnetostatic field problems with a three component vector potential **A**, and a three component source current density **J**. The energy related functional corresponding to (1.2.1.2), [36] is given by:

$$\mathcal{F} = \sum_{i=1}^{3} \int_{\Omega} \nu (\nabla A_i)^2 d\Omega - \int_{\Omega} \nu (\nabla \cdot \mathbf{A})^2 d\Omega - 2 \int_{\Omega} (\mathbf{J} \cdot \mathbf{A}) d\Omega - \int \int_{\Gamma} (\mathbf{A} \times \nu \nabla \times \mathbf{A}) \cdot d\mathbf{s}$$
$$- \int \int_{\Gamma} (A \cdot \nu \nabla) \mathbf{A} \cdot d\mathbf{s} + \int \int \mathbf{A} \nu \nabla \cdot \mathbf{A} \cdot d\mathbf{s} \tag{1.2.1.3}$$

The uniqueness of the vector potential solutions became a issue of controversy over the specification of $\nabla \cdot \mathbf{A}$. Mathematically, the uniqueness of a vector requires specification of

the curl, divergence and some boundary conditions. Kotiuga and Silvester [87] and Chari, Silvester et al. [36] argued that the choice of the Coulomb gauge, i.e. $div\mathbf{A} = 0$, guarantees the uniqueness of $\mathbf{A}$.

When $\nabla \cdot \mathbf{A} = 0$ is substituted in (1.2.1.3) and the surface integrals are set to zero, the functional becomes:

$$\mathcal{F} = \sum_{i=1}^{3} \int_{\Omega} \nu (\nabla A_i)^2 d\Omega - 2 \int \mathbf{J} \cdot \mathbf{A} d\Omega \tag{1.2.1.4}$$

which was used by Chari, Silvester et al.

Damerdash et al. [45] argued against using the Coulomb gauge and instead suggested solving the equation (1.2.1.2) with $\nabla \cdot \mathbf{A}$ non-zero and setting the boundary value of $\mathbf{A}$ such that the surface integral of its normal component is zero, which will ensure $\nabla \cdot \mathbf{A} = 0$. Their functional is given below in (1.2.1.5).

$$\mathcal{F} = \frac{\nu}{2} \int_{\Omega} (\nabla \times \mathbf{A})^2 d\Omega - \int_{\Omega} \mathbf{J} \cdot \mathbf{A} d\Omega \tag{1.2.1.5}$$

which satisfies (1.2.9) with $\nabla \cdot \mathbf{A}$ imposed.

Coulomb [42] suggested that $\nabla \cdot \mathbf{A}$ has to be used, and combines the energy functional given by (1.2.1.5) with the least square error functional for the divergence equation

$$\mathcal{F} = \int_{\Omega} (\nabla \cdot \mathbf{A})^2 d\Omega \tag{1.2.1.6}$$

and uses their sum

$$\mathcal{F} = \nu \int_{\Omega} (\nabla \times \mathbf{A})^2 d\Omega - 2 \int_{\Omega} \mathbf{J} \cdot \mathbf{A} d\Omega + \nu' \int_{\Omega} (\nabla \cdot \mathbf{A})^2 d\Omega \tag{1.2.1.7}$$

where ν' is a Lagrange multiplier .

There have also been other attempts to find a solution to (1.2.1.2) with or without specifying $\nabla \cdot \mathbf{A}$ and good results have been reported. Many of these implicitly set $div\mathbf{A}$ by their choice of finite element trial functions.

1.2.2 Scalar Potentials

Simkin and Trowbridge [145] are the main advocates of the use of total scalar potential for three dimensional magnetostatic problems, as the scalar potential solution is always computationally cheaper than the three component vector potential. Besides, several formulations can be developed by using a pair of scalar potentials and combining integral methods for linear regions with differential methods for permeable regions to further economize in the cost of solutions.

The total scalar potential Ω is defined for simply connected and source-free regions and is derived using (1.2.1) :

$$\nabla \times \mathbf{H} = 0 \tag{1.2.2.1}$$

then,

$$\mathbf{H} = -\nabla\Omega \tag{1.2.2.2}$$

It is often convenient to express the total field $\mathbf{H}$, as a sum of fields due to source currents $\mathbf{H}_s$ and due to induced magnetization $\mathbf{H}_m$, i.e.

$$\mathbf{H} = \mathbf{H}_s + \mathbf{H}_m \tag{1.2.2.3}$$

where $\mathbf{H}_s$ can be defined as:

$$\nabla \times \mathbf{H}_s = \mathbf{J} \tag{1.2.2.4}$$

Equation (1.2.2.4) implies that

$$\nabla \times \mathbf{H}_m = 0 \tag{1.2.2.5}$$

then

$$\mathbf{H}_m = \nabla \phi \tag{1.2.2.6}$$

The potential, ϕ, is known as reduced scalar potential and is valid everywhere.

Using equation (1.2.4), one gets

$$\nabla \cdot \mu(\mathbf{H}) = \nabla \cdot (\mu \mathbf{H}_s + \mu \mathbf{H}_m) = 0$$

i.e.

$$\nabla \cdot \mu \mathbf{H}_s = -\nabla \cdot \mu \mathbf{H}_m = \nabla \cdot \mu \nabla \phi \tag{1.2.2.7}$$

which is the main equation to be solved for. The reduced scalar potential forms the basis of algorithms for solving field problems which are reported in [78], [147], [177], [171]. The choice of using the reduced scalar potential for a highly permeable region often leads to numerical problems. For example, one can write equation (1.2.5) inside the iron as:

$$\mu(\mathbf{H}_s + \mathbf{H}_m) = (\mu \mathbf{H}_s - \mu \nabla \phi) \tag{1.2.2.8}$$

If $\mathbf{H}_s$ and $\nabla \phi$ are of the same order they may contribute to large errors in $\mathbf{H}$. This difficulty of numerical cancellation has been overcome by using a combination of ϕ and Ω. Upon taking the divergence of equation (1.2.2.2) i.e.

$$\nabla \cdot \mu \mathbf{H}_s = -\nabla \cdot \mu \nabla \Omega = 0 \tag{1.2.2.9}$$

a non-linear Laplace, equation is obtained.

The current sources are introduced either by taking suitable cuts to make the total scalar potential, Ω, single valued or using the combination of Ω and ϕ together.

For the sake of illustration, consider a problem domain consisting of two regions, in which region 1 has no sources, while region 2 has sources. Then region 1 can be expressed by Ω and region 2 by ϕ. The interface conditions of continuity of $\mathbf{B}_{normal}$ and $\mathbf{H}_{tangential}$ can be used to take care of any discontinuities in the potentials at the interfaces, i.e.

$$-\mu_1 \frac{\partial \Omega}{\partial n} = \mu_2 \left(\frac{-\partial \phi}{\partial n} + \mathbf{H}_{sn}\right) \tag{1.2.2.10}$$

and

$$-\frac{\partial \Omega}{\partial t} = -\frac{\partial \phi}{\partial t} + \mathbf{H}_{st} \tag{1.2.2.11}$$

where $\mathbf{H}_{sn}$ and $\mathbf{H}_{st}$ are the normal and tangential components of $\mathbf{H_s}$. $\mathbf{H}_{st}$ is related to Ω and ϕ by the following relation:

$$\Omega = \phi - \oint \mathbf{H}_{st} d\mathbf{l} \tag{1.2.2.12}$$

which defines a potential jump condition at any point on the surface. The 3D program package TOSCA [146] has been developed using these two potentials.

1.2.3 Integral Equations

Integral equation formulations based on the total scalar potential, Ω, and the reduced scalar potential, ϕ, have also been developed and used to solve 3D problems by Simkin and Trowbridge [147], Armstrong et al. [5]. These have been derived from equation (1.2.2.3) from which **H** can directly be represented in integral form:

$$\mathbf{H} = -\nabla \int \mathbf{M} \cdot \nabla(\frac{1}{R}) d\Omega + \mathbf{H}_s \tag{1.2.3.1}$$

where R is the distance between the source point $\mathbf{r}'$ and the field point $\mathbf{r}$ and **M** is the induced volume magnetization and is defined by:

$$\mathbf{M} = \frac{\mathbf{B}}{\mu_0} - \mathbf{H} \tag{1.2.3.2}$$

or,

$$\mathbf{M} = \chi \mathbf{H} = (\mu - 1)\mathbf{H} \tag{1.2.3.3}$$

where χ is the magnetic susceptibility . As $\mathbf{H}_s$ is the field due to the prescribed currents **J**, it can be directly found from

$$\mathbf{H}_s = \frac{1}{4\pi} \int_\Omega \frac{\mathbf{J} \times (\mathbf{r} - \mathbf{r}')}{|r - r'|^3} d\Omega \tag{1.2.3.4}$$

Using equations (1.2.3.1) and (1.2.3.2.), along with (1.2.2.6), the expression for total scalar potential is obtained :

$$\Omega = -\int_{\Omega} \chi \nabla\Omega \cdot \nabla(\frac{1}{R}) d\Omega + \phi_s \tag{1.2.3.5}$$

where ϕ_s is the scalar potential from prescribed currents and is related to Ω by

$$\Omega = \phi + \phi_s \tag{1.2.3.6}$$

The application of Green's theorem to equation (1.2.3.5) results in the integral equation for Ω, i.e.

$$\Omega = -\int_{\Gamma} \chi\Omega \nabla(\frac{1}{R}) \cdot d\Gamma + \int_{\Omega} \Omega \nabla\chi \cdot \nabla(\frac{1}{R}) d\Omega + \phi_s \tag{1.2.3.7}$$

where Γ is the surface of volume Ω. For a typical three dimensional problem discretized by finite elements the difference in the cost of the solution by integral and differential methods [148] are summarized below.

Operation	Integral	Differential
Matrix Set up	n^2	n
Equation Solution	$n^{2.7}$	$nlog(n)$ or $n^{1.5}$
Field Evaluation	n	1
N fold symmetry	N	1

From this table it is obvious why the differential formulations are preferred over integral ones.

The $\mathbf{T} - \Omega$ and $\mathbf{A} - \phi$ formulations of Carpenter [25] for calculating the magnetic field and eddy currents have been used to solve a host of practical field problems, including the

analysis of end-region fields in large turbogenerators. These formulations, especially the $\mathbf{T} - \Omega$ have been used extensively for eddy current computations in three dimensions. It is appropriate, therefore to discuss these formulations in the next section, where the current state of the art in 3D eddy current computations is reviewed.

1.3 Eddy Current Computations in Three Dimensions

In the past, eddy currents in electromagnetic devices were computed by analytical or analogue techniques using certain approximations to the material properties and simplifications in the geometries involved. These analytical methods proved inadequate for the treatment of the complex geometries associated with electrical machines, and this led to the development of numerical methods for engineering solutions to eddy current problems in linear as well non-linear media. However, applications were restricted to two dimensional representations of the geometries only. With the advent of new computers with powerful processors and large memories, three dimensional problems are now beginning to be solved. True three dimensional solutions, however, are still rare because the memory and time requirements for practical problems are prohibitive. New numerical formulations are being developed under certain approximations so that real problems can be solved within existing computing resources. These formulations have been developed, or are being developed, for specialized classes of problems, thus leading to a number of different formulations. Though these techniques provide meaningful solutions to these particular classes of problems, they are inadequate in treating most general problems and there has not, as yet, emerged a single technique which can handle a variety of problems. The recently published formulations [50], [122], [125] look promising but have yet to be demonstrated on a variety of practical problems.

Finite differences have been tried extensively both in one and two dimensions but their

use in three dimensions has been severely limited. Integral equation methods have also been investigated by several authors [12], [163], [13] and used to solve a host of practical problems. Despite several attractive features, discussed earlier, integral methods are not currently being favoured over differential methods because of the high cost involved in the solution of the resulting asymmetric and totally dense system of equations.

Instead of reviewing the entire literature on three dimensional eddy current computations, a task which has already been undertaken by several authors, namely Brown [14], Wolff [172], and extensive bibliography by Lari and Turner [88], some prominent current techniques will be discussed.

1.3.1 The $T - \Omega$ and $A - \phi$ Methods

In the work of Carpenter [25] are to be found the origins of $\mathbf{T} - \Omega$ and $\mathbf{A} - \phi$ methods, both of which have found extensive use in industrial applications, particularly the $\mathbf{T} - \Omega$ method. The $\mathbf{A} - \phi$ formulation has not been used to the same extent because of the difficulties in the treatment of interface conditions between conducting and non-conducting regions. Besides, $\mathbf{A}$ has three components to be solved for in all the regions, in addition to ϕ for conductors. On the other hand, the $\mathbf{T} - \Omega$ method has the attraction that $\mathbf{T}$ is only solved for in conducting regions as it is either constant or zero in non-conducting regions. For the general case of three current components, two components of $\mathbf{T}$ are sufficient, and for regions where current flows on planes, only one component of $\mathbf{T}$ is needed.

The electric vector potential $\mathbf{T}$ is defined as:

$$\nabla \times \mathbf{T} = \mathbf{J} \tag{1.3.1.1}$$

which is related to $\mathbf{H}$ by the following expression

$$\mathbf{H} = \mathbf{T} - \nabla\Omega \tag{1.3.1.2}$$

where Ω is the magnetic scalar potential. Use of (1.2.2) and (1.2.5) along with (1.3.1.1) and (1.3.1.2) finally lead to

$$\nabla \times \frac{1}{\sigma}\nabla \times \mathbf{T} = -\frac{\partial}{\partial t}(\mu\mathbf{T} - \mu\nabla\Omega) \tag{1.3.1.3}$$

Maxwell's equation (1.2.4) with equation (1.3.1.2) gives rise to

$$\nabla \cdot (\mu\nabla\Omega) = \nabla \cdot \mu\mathbf{T} \tag{1.3.1.4}$$

Despite several advantages, the possibility of cancellation between $\mathbf{T}$ and $\nabla\Omega$ in conducting regions exists.

Similarly, the $\mathbf{A} - \phi$ formulation results from the defining equations for the magnetic vector potential $\mathbf{A}$, i.e. equation (1.2.1.1), and

$$\mathbf{E} = -\frac{\partial \mathbf{A}}{\partial t} - \nabla\phi \tag{1.3.1.5}$$

where ϕ is the electric scalar potential.

The use of Maxwell's equation (1.2.1) and the constitutive relations (1.2.5), (1.2.7) together with (1.2.1.1) and (1.3.1.5) leads to

$$\sigma(-\frac{\partial \mathbf{A}}{\partial t} - \nabla\phi) = \nabla \times \frac{1}{\mu}\nabla \times \mathbf{A} \tag{1.3.1.6}$$

The equation which couples both potentials is derived using

$$\nabla \cdot \mathbf{J} = 0 \tag{1.3.1.7}$$

i.e

$$\nabla \cdot \mathbf{J} = \nabla \cdot \sigma \mathbf{E} = \nabla \cdot \{\sigma(\frac{-\partial \mathbf{A}}{\partial t} - \nabla \phi)\} = 0$$

or

$$\nabla \cdot \sigma \nabla \phi = -\nabla \cdot (\sigma \frac{\partial \mathbf{A}}{\partial t}) \tag{1.3.1.8}$$

The uniqueness of the solution is guaranteed for the $\mathbf{A} - \phi$ formulation by imposing $\nabla \cdot \mathbf{A} = 0$, and similarly for the $\mathbf{T} - \Omega$ formulation by imposing $\nabla \cdot \mathbf{J} = 0$. Nevertheless, there is still considerable discussion concerning uniqueness, much of which can be found in the published in the literature [14].

Most of the two potential formulations have their roots in either the $\mathbf{T} - \Omega$ or the $\mathbf{A} - \phi$ method and these other variants will be discussed now.

1.3.2 $R - \Omega$ Formulation

As discussed earlier, in the $\mathbf{T} - \Omega$ method it is difficult to handle the interface conditions because the normal component of $\mathbf{T}$ is not continuous. This issue may be resolved by using an alternate approach [13]. The vector $\mathbf{R}$ can be defined as

$$\mathbf{H} = \mathbf{R} - \nabla \Omega \tag{1.3.2.1}$$

for conducting region Ω_1, where Ω is the magnetic scalar potential. Using Maxwell's equations (1.2.1) and (1.2.2) with equation (1.3.2.1).

$$\nabla \times \mathbf{H} = \nabla \times \mathbf{R}$$

or

$$\sigma \mathbf{E} = \nabla \times \mathbf{R}$$

which, upon taking the curl of both sides becomes

$$\nabla \times \nabla \times \mathbf{R} = \sigma \nabla \times \mathbf{E} = -\sigma \frac{\partial \mathbf{B}}{\partial t} = -\sigma [\frac{\partial}{\partial t} \{\mu(\mathbf{R} - \nabla \Omega)\}] \tag{1.3.2.2}$$

Use of $\nabla \cdot \mathbf{B} = 0$, with (1.3.2.1) leads to

$$\nabla \cdot \mu \mathbf{H} = \nabla \cdot \mu(\mathbf{R} - \nabla \Omega) \tag{1.3.2.3}$$

For the non-conducting region Ω_2, which encloses the conducting region Ω_1, the magnetic scalar potential is defined as

$$\mathbf{H} = -\nabla \Omega \tag{1.3.2.4}$$

which, when used with (1.2.4), reduces to

$$\nabla \cdot \mu \nabla \Omega = 0 \tag{1.3.2.5}$$

The interface conditions are derived from the continuity of the normal component of $\mathbf{B}$ and the tangential component of $\mathbf{H}$ and can be written as:

$$\mu_1 (R_n - \frac{\partial \Omega}{\partial n})_{\Omega_1} = \mu_2 (\frac{\partial \Omega}{\partial n})_{\Omega_2} \tag{1.3.2.6}$$

and

$$(R_t - \frac{\partial \Omega}{\partial t})_{\Omega_1} = (\frac{\partial \Omega}{\partial t})_{\Omega_2} \tag{1.3.2.7}$$

This technique has been tested against analytical solutions and experimental results and provides correct solutions. Nevertheless, the possibility of cancellation in $\mathbf{H} = \mathbf{R} - \nabla\Omega$ still exists.

It has become clear by now that for eddy current problems conducting regions require two potentials, i.e. one vector with three components and one scalar to ensure that the solution is unique. In most engineering applications, the conducting region forms a major part of the discretized domain, and four degrees of freedom per node in this region may finally lead to a system of algebraic equations whose solution can become very demanding on the computing resources. It is, therefore, preferable to have a smaller number of variables without sacrificing the accuracy by making suitable approximations. Recently, three new formulations have been described which all have one goal in common, i.e. reducing the number of variables per node in the conducting regions. These are the results of the work done by Emson and Simkin [50], Polak et al. [125] and Pillsbury [122].

1.3.3 The Emson and Simkin Formulation

In this approach a new variable is defined, a vector potential $\mathbf{A}^*$, which is the combination of a vector potential and an electric scalar potential for the conducting regions. Non-conducting regions are treated by a magnetic scalar potential. It has been derived as follows:

The vector potential $\mathbf{A}$ is defined such that

$$\mathbf{B} = \nabla \times \mathbf{A} \tag{1.2.1.1}$$

then

$$\mathbf{E} = -(\frac{\partial \mathbf{A}}{\partial t} + \nabla\phi) \tag{1.3.1.5}$$

These two equations, together with Maxwell's equation (1.2.1) lead to

$$\nabla \times \frac{1}{\mu} \nabla \times \mathbf{A} = -\sigma \left(\frac{\partial \mathbf{A}}{\partial t} + \nabla \phi \right) \tag{1.3.3.1}$$

The new vector potential is defined as

$$\mathbf{A}^* = \mathbf{A} + \int \nabla \phi dt \tag{1.3.3.2}$$

which, when substituted in (1.3.3.1) leads to

$$\nabla \times \frac{1}{\mu} \nabla \times \mathbf{A}^* = -\sigma \frac{\partial \mathbf{A}^*}{\partial t} \tag{1.3.3.3}$$

The equation (1.3.3.3) has a unique solution with the conductivity gauge

$$\nabla \cdot \sigma \mathbf{A}^* = 0 \tag{1.3.3.4}$$

For non-conducting regions, the magnetic scalar potential Ω defined in equation (1.3.2.4) is used and the governing is (1.3.2.5).

1.3.4 The Polak et al. Formulation

In this formulation the domain of analysis is divided into three parts:

(1) Region I containing driving sources.

(2) Region II containing materials with high permeability.

(3) Region III where $\sigma \neq 0$ and eddy currents are expected to flow.

So, the corresponding governing equations are: for region I,

$$\nabla \cdot \mu (\nabla \phi_m + \mathbf{H}_c) = 0 \tag{1.3.4.1}$$

where $\mathbf{H}_c$ is the field from the prescribed currents and can be calculated explicitly, using Biot-Savart's Law. The scalar potential ϕ_m is defined as

$$\mathbf{H}_m = -\nabla\phi_m \tag{1.3.4.2}$$

which is governed by

$$\mathbf{H}_m = \mathbf{H} - \mathbf{H}_c \tag{1.3.4.3}$$

For region II, the main equation is

$$\nabla \cdot \mu\nabla\phi_f = 0 \tag{1.3.4.4}$$

with

$$\mathbf{H} = -\nabla(\phi_c + \phi_m) \tag{1.3.4.5}$$

where

$$\phi_f = \phi_c + \phi_m \tag{1.3.4.6}$$

The potential ϕ_f, ϕ_m and ϕ_c are obtained as follows. The equation

$$\nabla \times \mathbf{H}_c = \mathbf{J} \tag{1.3.4.7}$$

is solved for the entire region by replacing magnetic materials by air. The solution $\mathbf{H}_c$ has the property that for regions, where $\mathbf{J} = 0$, $\mathbf{H}_c = -\nabla\phi_c$ and for the regions involving magnetic parts with $(\mathbf{J} = 0)$, $\mathbf{H}$ is given by (1.3.4.5). The potential ϕ_f defined for highly permeable regions $\mathbf{H} = -\nabla\phi_f$ for $(\mathbf{J} = 0)$ is related to other potentials by (1.3.4.6).

For region III, the main equation is

$$\nabla \times \frac{1}{\mu}\nabla \times \mathbf{A} = -\sigma\frac{\partial \mathbf{A}}{\partial t} \tag{1.3.4.8}$$

This formulation has been tried on the test problems used by Emson and Simkin and good correlation has been obtained. Nonetheless, it needs to be applied to a wide range of problems before its validity can really be established.

1.3.5 The Pillsbury Formulation

The formulation proposed by Pillsbury uses two potentials, a magnetic vector potential **A** and a total magnetic scalar potential Ω. The electric scalar potential is required for the conducting regions to maintain the uniqueness of **A**. The main advantages of this formulation are

(a) that the final system of equations is symmetric, and

(b) that it can be used for magnetostatic problems as well.

The problem of numerical cancellation in various formulations is totally avoided. This has been accomplished through the imposition of the continuity requirements of the normal component of flux density and the tangential components of field intensity.

The problem region Ω is subdivided into two regions, region Ω_1 containing sources and conducting parts and region Ω_2 containing everything else. The **A** has been defined from $\nabla \times \mathbf{B}$, which, with the Coulomb gauge, results in a unique solution governed by

$$\nabla \times \frac{1}{\mu} \nabla \times \mathbf{A} = -\sigma \left(\frac{\partial \mathbf{A}}{\partial t} + \nabla \phi \right) \tag{1.3.3.1}$$

with **E** defined by equation (1.3.1.5). The scalar potential equation is the same as that derived by any other formulation and can be rewritten as

$$\mathbf{H} = -\nabla \Omega \tag{1.3.2.4}$$

and

$$\nabla \cdot \mu \nabla \Omega = 0 \tag{1.3.2.5}$$

The interface conditions are:

$$\mathbf{n} \cdot \mathbf{B} = 0$$

$$\mathbf{n} \times \mathbf{H} = 0$$

and,

$$\mathbf{n} \cdot \mathbf{J} = 0$$

$$\mathbf{n} \times \mathbf{E} = 0$$

Having discussed the issues involved with the numerical computations of eddy currents in three dimensions, and the importance of approximations in such computations in the context of current state of the art in the field, the terminal box problem can now be considered.

1.4 The Terminal Box- A Practical Problem

Fabricated of highly permeable sheet steel, a terminal box looks like a box through which the output terminals of the generator pass from the casing (Fig. 1.1). The magnitude of the currents flowing through these terminals is of the order of several kilo-amperes and when these currents continuously flow through the conductors strong magnetic fields are established in and around the box. These fields are time varying and induce eddy currents on the conducting magnetic surfaces such as the walls of the box, as well as the non-magnetic parts. The presence of eddy currents results in localized heating of the iron parts. In addition, the mechanical forces between conductors could become large enough to exert pressures on the box support structure such that it may need to be redesigned. Knowledge of the fields is required in the determination of the surface currents and with these two quantities known, a host of other additional quantities such as forces and surface impedances can be evaluated.

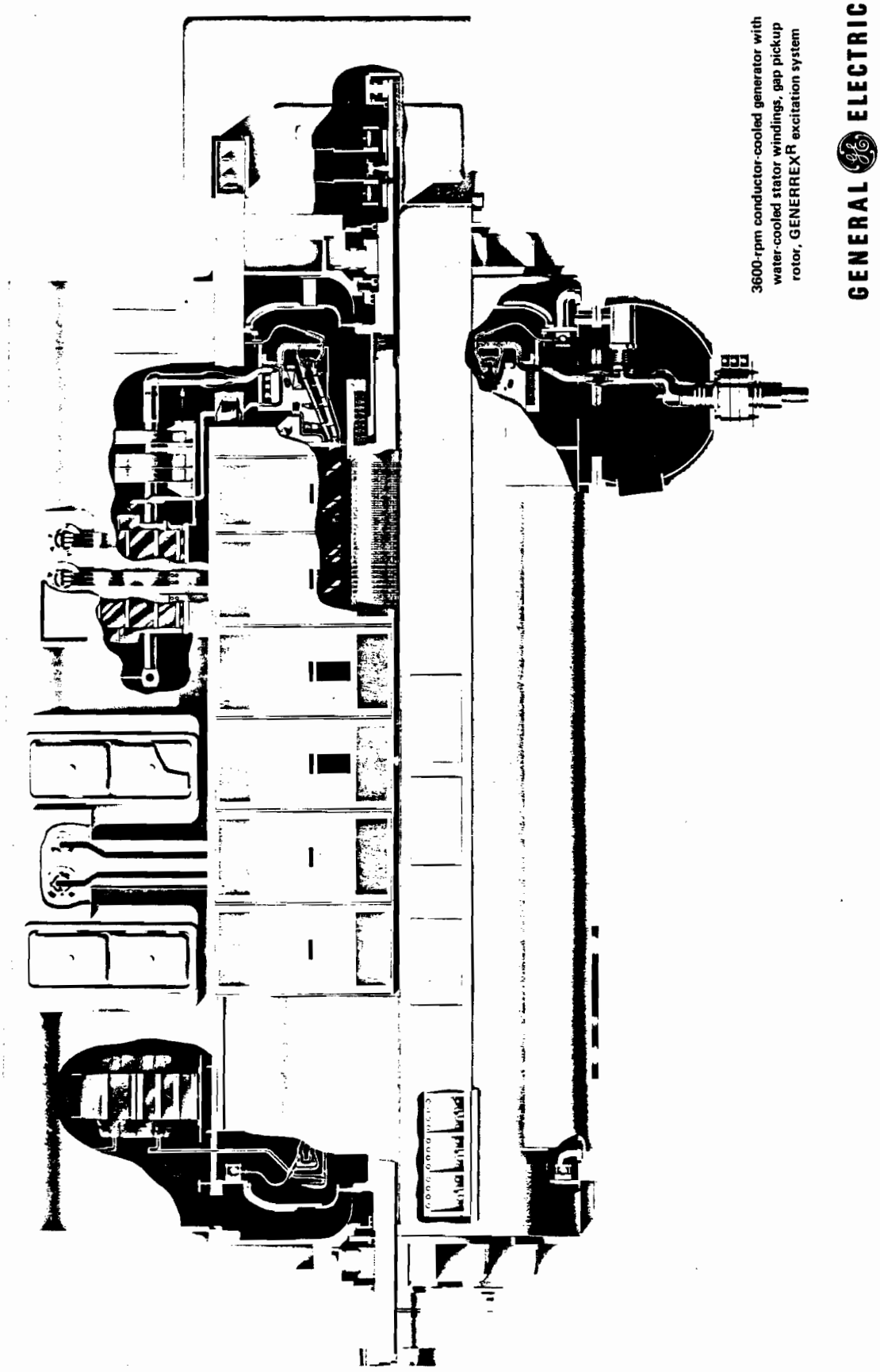


Figure 1.1 A turbogenerator with the terminal box

1.4.1 General Considerations

A cursory look at the geometry of the terminal box shown in Fig 1.1 reveals the non-existence of any translational symmetry, which could be exploited to reduce it from a three dimensional to an approximate two dimensional form. Owing to the complexity of the geometry of the terminal box, there does not seem to be any possibility of finding an analytical solution; only a numerical solution will model the problem and that, too, may require approximations and idealizations in the geometry.

Simplification is not only required for the geometries involved but also for the phenomena being analyzed. A significant reduction in computational cost is achieved if the function modelling it is chosen with the physics of the problem in consideration. In the context of the terminal box, the phenomenon being modelled is the magnetic field, which is more often than not solved in terms of potentials.

The finite element solution of a field problem is obtained by passing it through three distinct, sequential phases, all of which are interlinked. These phases are:

(i) Pre-processing: which includes the modelling and the discretization of the geometry and the definition of the problem in terms of boundary conditions and source specifications.

(ii) Analysis: which includes formation of the coefficient matrix and the solution of a system of algebraic equations.

(iii) Post-processing: which is concerned with the evaluation of desired parameters and may involve graphic displays.

Each phase forms a basic unit, although changes in the first phase affect the subsequent phases, for example the choice of element types, i.e. triangular, tetrahedral or prism, and their order of approximation, affects the second and third phases. In the next sections,

these three phases will be described particularly in the view of the terminal box and each of them will then be treated, in general, in subsequent chapters.

1.4.2 Which Potential ?

The choice of a potential function is governed by two fundamental requirements: one is that it should be able to model the fields without sacrificing any accuracy and the second is that it should offer computational savings in terms of reducing the number of unknowns to be solved for. The choice of the magnetic vector potential **A** is most natural for problems with translational or rotational symmetry, as it is assumed to have only one component and may be treated as a scalar. The same thing is not true in three dimensions. For eddy current problems at least a pair of potentials is required to treat the prescribed currents, the conducting regions and the free space. As discussed earlier this pair may be either magnetic vector potential with one electric scalar potential or an electric vector potential with a magnetic scalar potential or total scalar potential with reduced scalar potential.

A scalar potential, though it has advantages in terms of computational costs, is inadequate to model the general volume distribution of sources because it is non-unique. However, using Carpenter's example of magnetic shells [30], it can be shown to be unique under the assumptions that the volume distributions of the induced sources are surface distributions. For most of the engineering problems involving ferro-magnetic materials, this assumption is not far from reality, as the skin depth of the field for these materials is less than a millimeter at power frequencies. Moreover, non-uniqueness due to the sources carrying prescribed currents can be removed by taking suitable cuts through the geometry [146].

The solution of the linear algebraic equations resulting from the use of a two potential formulation, i.e four degrees of freedom per node for the conducting region, may well be

higher by an order of magnitude when compared to the total magnetic scalar potential formulation and may thus need a main-frame computer because of memory and time requirements. Three dimensional problems invariably result in large systems of equations, of the order of several thousands, even when there is only one degree of freedom per node. Solutions of these equations on small machines, in general, are not possible using conventional techniques because of limitations in address space and slower numerical processing. Nevertheless, the solution of large three dimensional problems can be made possible on machines of reasonably small storage if new algorithms are developed and suitable data-structures are employed.

1.4.3 Pre-Processing - Mesh Generation in Three Dimensions

The pre-processing phase consists of modelling and the discretization with finite elements of the geometric domain in which the solution is sought. There are two basic approaches currently in use. In the first approach, the entire geometric model is produced with the help of analytic functions, e.g. surface and solid modelling in a CAD/CAM system, and then automatic triangulation of the domain using suitable elements. The other approach, which is more common, consists of building the model element by element, by synthesis. Several distinct parts can be meshed separately and then finally joined together. The first approach, though very promising in 2D is yet to be fully exploited in three dimensions, as automatic triangulation with well shaped elements, with little or no user control, sometimes becomes difficult. On the other hand, the second approach makes the task of mesh generation easier as the user has full control while building the mesh and, therefore, can define the elements and refine the mesh as desired; but display is difficult.

The terminal box has a very complicated geometry and is designed to house the output terminals of the generator. The presence of bent conductors, fastners, holes, etc. add to

the complexity of the box shape. The entire model will consist of interior structural details of the box with the air, fastners, ribs, etc., and the conductors coming out of the box along with the surrounding air outside the box. The facilities to model the box in its entirety possibly do not exist and, therefore, approximations have to be made by considering the fact that ribs, fastners and the associated holes form a very small part of the entire problem region and their effect on overall accuracy of the solution can be considered insignificant. In any case this is a preliminary study of the box and will not be used directly for the terminal box design in its present form, hence certain liberties can be taken in simplifying the geometry. In this form, the problem region will still be comprised of the walls of the box, the air inside and outside of the box, and the current carrying conductors.

Tetrahedral elements offer both geometric flexibility in modelling almost any shape and faster matrix assembly times as well, and are well-suited to model the air both inside and outside the box. But they are not particularly suited to model the comparatively thin walls of the box as they not only result in poorly shaped elements, but also require a large number of elements. Triangular prisms can avoid both of these problems, and are compatible with tetrahedra. In order to be able to model the terminal box geometry, therefore, a three dimensional mesh generator is required which has in its element library at least two different kind of elements, i.e. triangular prisms and tetrahedra.

There are quite a number of powerful, commercially available mesh generators which can be used to model the terminal box geometry; but, first, they are very expensive to buy, and second, they only run on main frames or super minis such as the Digital Equipment's VAX. These two constraints pave the way for the development of an interactive three dimensional mesh generator which has triangular prisms and tetrahedra as the basic mesh building elements and runs on machines with finite memories, in the range of 28 Kbytes to 3/4 Megabytes, such as PDP-11, Perq, Codata and similar machines.

1.4.4 Solution of the Algebraic Equations

The second phase in the solution of a problem is known as the analysis phase. Here the coefficient matrix is formed and the resulting equations are solved. As discussed earlier, three dimensional regions have a large number of variables and as a result lead to large sparse matrices. The time required to solve these equations is considerable and far exceeds the time required for mesh generation and post-processing. Both the memory requirements and the execution time are at stake and the chosen method should offer a reasonable compromise. More often than not, the number of equations cannot be reduced without sacrifices in the accuracy of the solution. This in turn places a stringent requirement on computer memory. In almost any reasonably sized problem the analyst is faced with this situation. The only possible cure is to use or develop memory-economic algorithms which trade memory for time.

Direct methods such as Gaussian elimination, and iterative methods such as Gauss-Seidel, SOR and conjugate gradients are the basic family of methods which may be used for solving large sparse systems of equations. For large sparse matrices, iterative methods are very competitive and are preferred over direct methods because they require less storage (no fill-ins), and the use of a good starting solution vector can be made to speed up convergence. Secondly, iterative methods are considered to be self-correcting from round-off errors as they use the original coefficient matrix in obtaining the solution and the matrix is preserved throughout the solution.

1.4.4.1 Frontal Method

One of the major difficulties limiting the size of the problem to be solved is the need to hold the entire coefficient matrix in core during the matrix formation phase. Irons [76], who circumvented this difficulty by a technique widely known as the frontal method, avoids

the formation of complete matrix at one time. Instead a frontwidth (f) by frontwidth submatrix is stored in core. Variables are eliminated as soon as their last occurrence is noticed while forming the matrix, and the rows or columns corresponding to these variables are transferred on-to a back-up disk. As the only space requirement is $O(f^2)$ rather than $O(N^2)$, where f is $O(N^{\frac{2}{3}})$, larger problems can be solved even on smaller machines, provided these machines have large backup disks and enough core to store f^2 elements. The efficiency of the frontal algorithm, however, is critically dependent on how the elements of the mesh are sequenced. Otherwise, it may lose to a good band solver.

1.4.4.2 The Element Sorting and Node Renumbering

The frontwidth of a mesh is defined in terms of the number of active variables while the current variable is being processed. It is not at all unusual to encounter three dimensional problems where f may be several hundreds or thousands, and it is increasingly important to develop algorithms which can minimize the frontwidths of meshes. It will not only facilitate the solution of problems with large frontwidths, but also make the execution of a frontal solver efficient. There currently exist algorithms which perform the element sorting in the same memory space as that required by frontal solvers.

Node renumbering schemes are conventionally used to reduce the bandwidth of the resulting matrices, so that fill-in in the profile during matrix factorization can be reduced. Node renumbering schemes are basically profile minimization procedures and are frequently used with solution algorithm. For the frontal solution technique, node renumbering is not as important as the element resequencing. It has been observed, however, that a good element sorting algorithm also indirectly induces a good node renumbering.

Fill-in in the factorized matrix may be reduced by a reduction in the bandwidth, but fill-in inside the band cannot be avoided. For problems having bandwidths in the several

hundreds, the amount of fill-in can be enormous. It is the amount of fill-in within the band that makes the solution of equations expensive both in terms of time and space and therefore plays a decisive role in the selection of a solution algorithm on machines of limited core.

A new element sorting algorithm has been developed in this thesis which works analogous to a Gaussian elimination and could be termed as a natural resequencing algorithm. It is a one-step, memory economic algorithm which also renumbers the nodes in the same pass.

1.4.4.3 Preconditioned Conjugate Gradients

As has been discussed above, iterative methods which preserve the sparsity of the matrix and have a fast guaranteed convergence are considered to be better than direct methods on current computing hardware, and the conjugate gradient method is the most favoured of them. It can be considered as a semi-iterative method: in principle it is an iterative method, but it has the basic characteristic of direct methods in the sense that, apart from round-off errors, the solution is achieved in no more than N steps, for N equations. It has been shown [128], that the solution could converge in very much less than the upper bound of N iterations if the eigenvalues of the coefficient matrix are clustered with many close to unity. This is possible by preconditioning the system of equations with an easily derived approximate inverse to the coefficient matrix.

One of the well known preconditioned conjugate gradient methods, ICCG(0), uses an incomplete Cholesky decompose as a preconditioning matrix [101]. The preconditioning matrix is computed by retaining the same sparsity pattern in the decompose as the coefficient matrix had. This algorithm has the convergence characteristics of $O(\sqrt{N})$, determined experimentally.

A new algorithm, which combines the frontal method for assembling the coefficient matrix and its factorization and the preconditioned conjugate gradient algorithm ICCG(0) for solving the equations, was recently developed [112] and has the basic characteristics of both algorithms. The new algorithm has the advantage of modifying the 'incompleteness' of the decomposition to use as much memory as is available. It behaves as a 'soft-failing' algorithm in which memory is traded-off against solution time.

The convergence of iterative methods for symmetric positive definite matrices depends on the condition number of the coefficient matrix. The condition number is defined as the ratio $(\frac{\lambda_{max}}{\lambda_{min}})$, i.e. the ratio of largest and smallest eigenvalues. The basic ICCG(0) method, in which all the fill-in entries are discarded, results in a preconditioned matrix whose eigenvalues are not very close and their condition number is large. The condition number of preconditioned system of equations could be improved if the Cholesky factor L is computed by keeping exact equality between the elements of the matrices (LL^T) and A. Moreover, by maintaining row sum equality, the effect of round-off errors in matrix decomposition can also be suppressed and this matrix is a better approximation to the complete Cholesky factor. When this modified preconditioning matrix is used with the conjugate gradient algorithm for the system of equations assembled using the frontal method, faster convergence is achieved when compared to the 'soft-failing' algorithm.

1.4.5 Post-Processing

This is the phase in which the validity of the solution is examined and the required output quantities are computed. There is a growing tendency among CAD analysts to examine the validity of numerical computations by looking at the plots of the solution, before any further post-processing is performed. The display of the potentials and field distributions by means of contours is well established in two dimensions. However, its

logical extension into three dimensions is somewhat complicated. Difficulties arise due to the limitations of display devices in displaying three dimensional fields or objects onto two dimensional screens. Even if the plots can be displayed, it is often not easy to comprehend them since a perspective view of the plots of scalar field is provided over the geometrical model. The general convention is to provide plots of these fields or potentials on a plane cutting through the discretized geometry. These plots are both comprehensible and easy to produce. The plots of potentials or the field on a number of cut-planes provide detailed distribution of the potential or the field in the domain of analysis.

From the point of view of the terminal box, the most important quantities which must be derived from the analysis are the graphical representation of equipotential contours of the scalar potential on the wall surfaces and the numerical estimates of losses and forces. These plots will provide a description of the eddy current distribution and their surface concentrations.

1.5 Original Contributions

To the best of the author's knowledge, the following are the original contributions contained in this thesis:

(1) Development of a methodology for the approximate analysis of turbogenerator terminal boxes and related housings, in three dimensions, using magnetic scalar potential only, developed under the following approximations:

   (a) regions carrying prescribed currents occupy so little space that these can be excluded from the domain of analysis and are replaced by boundary conditions;

(b) the depth of penetration of the magnetic field in ferromagnetic sheets is so low that the eddy currents flowing in them may be treated as surface currents.

(2) Development of two preconditioned conjugate gradient frontal solvers for three dimensional electromagnetic field problems to solve the large sparse system of linear algebraic equations on machines of limited address spaces:

(a) a 'soft-failing' algorithm in which memory is traded off against solution time by modifying the 'incompleteness' of the decomposition;

(b) a robust and fast converging algorithm which can handle several regions of high contrasting material properties by employing a modified preconditioning matrix.

(3) Development of an interactive mesh generator using irregular hexahedral bricks to model three dimensional geometries with a choice of discretization of the region either by first or second order tetrahedral elements or triangular prism elements to run on small machines such as PDP-11, Perq, or Codata systems.

(5) Development of a new memory-economic element and node sorting algorithm for the reduction of frontwidths for the frontal solutions and the minimization of profile and bandwidths of the resulting matrices, designed to run on small machines.

(6) Development of a three dimensional analysis system running on a mini-computer using (2), (3), (5) and (8).

(7) Analysis of a turbogenerator terminal box model using the developed system and the display of equipotential contours.

(8) Improvements in a contour plotting algorithm which is based on a scan-conversion technique for geometries modelled either by tetrahedra or triangular prisms.

1.6 Thesis Organization

The prominent numerical approaches for the solution of three dimensional magnetostatic or eddy current problems have been reviewed in Chapter I. These are based mostly on two potential formulations, one vector and the other scalar. The problem of the terminal box is introduced and is subsequently examined for its pre-processing, analysis and post-processing requirements. These requirements have been particularly addressed from the point of view of memory economy.

In Chapter II the mathematical formulation of the terminal box analysis is developed in terms of a magnetic scalar potential alone on the assumption that the current carrying conductors occupy comparatively little space, and the eddy currents, which flow on the ferromagnetic walls of the box, have a small skin depth, i.e. are surface currents. Element matrices, both Dirichlet and Metric, for tetrahedra compatible sheet elements (triangular prisms) for the scalar Helmholtz equation are derived.

A survey of the state of the art in the field of mesh generation in three dimensions is presented in Chapter III. A brief survey of commercially available 3D mesh generators in mechanics as well as magnetics is given and basic techniques are highlighted. Finally, the need to develop a mesh generator and its distinct features are presented. Results in the form of a few discretized geometries are also included.

In Chapter IV a new algorithm of element and node sorting for frontal solutions is developed. The importance of element resequencing for the reduction of frontwidths of

the meshes and the effects of bandwidth reduction due to node labelling techniques are presented. The new algorithm is a memory economic, one step algorithm in which both the element and the node sorting are done in the same pass. The performance results of the algorithm when tested extensively on various topologically different three dimensional geometries are also summarized.

Chapter V is concerned with the preconditioned conjugate gradient method and its combination with the frontal algorithm, which in turn, led to the development of two preconditioned conjugate gradient frontal methods. One of the attractive features of the first algorithm is its 'soft-fail' nature, while the second algorithm is fast and robust. Both incomplete Cholesky preconditioned conjugate gradient algorithms are discussed and their performances are compared.

In Chapter VI post-processing aspects of a finite element solution are presented. Difficulties encountered in post-processing solutions on three dimensional geometries are touched upon and a scan-conversion technique for plotting the scalar potential contours on user defined planes is presented. The technique is then applied on to a terminal box model.

Chapter VII presents and relevant conclusions with regard to solving a three dimensional problem on machines of limited memories are made in the light of a terminal box. Areas which need furthur exploration in relation to the terminal box are discussed. This chapter is followed by References and Appendices.

Chapter 2

Mathematical Formulation

2.1 Introduction

In general, eddy current problems in electromagnetic devices have been solved using analytical techniques, analog techniques, and numerical techniques. Most of these methods are restricted to a one component solution of the eddy currents, which is orthogonal to the problem geometry, both in one and two dimensions. In many applications, this approach to the treatment of eddy currents is inadequate, and a true three dimensional analysis is required. This is not only computationally very expensive but the computed field distribution is very difficult to comprehend. Under these circumstances, numerical solutions of three dimensional eddy current problems are obtained under simplifying assumptions and approximations with regard to the phenomenon itself and the geometries involved. In this chapter an approximate method for finding the eddy current distribution on box shape geometries, fabricated from ferromagnetic plates, is presented. The formulation, in terms of magnetic scalar potential, is developed under the following assumptions:

(1) The eddy currents are restricted to surface currents.

(2) The permeability of iron sheet is very high.

(3) The source region is very small as compared to the total domain of analysis and thus the sources may be considered as filamentary in nature.

(4) The field due to eddy currents is very small.

There are two cases which are of interest for the terminal box analysis:

(a) Strong eddy currents flowing on the wall surfaces of the box preventing flux penetration into the iron.

(b) Moderate currents flowing on the wall surfaces.

2.2 The Scalar Potential Equation

For all the current-free regions, except the current carrying conductors and the box walls, the vector **H** is irrotational, i.e.

$$\nabla \times \mathbf{H} = 0 \tag{2.2.1}$$

Therefore, **H** can be written as,

$$\mathbf{H} = -\nabla\Omega \tag{2.2.2}$$

Where Ω is the magnetic scalar potential. Making use of constitutive relation

$$\mathbf{B} = \mu\mathbf{H} \tag{1.2.6}$$

one can write

$$\mathbf{B} = -\mu(\nabla\Omega) \tag{2.2.4}$$

Since the $\nabla \cdot \mathbf{B}$ is always zero, (2.2.4) can be written as

$$\nabla \cdot (\mu\nabla\Omega) = 0 \tag{2.2.5}$$

which is a nonlinear Laplace's equation. Basically, Laplace's equation has to be solved in linear as well as ferromagnetic media subject to suitable boundary conditions, which are to be derived from the physical model of the terminal box. The first set of boundary conditions are those which arise due to current carrying conductors. The second set of boundary conditions originate from the wall surfaces and depend on the magnitude of the eddy currents flowing in them and the material properties.

2.3 Boundary Conditions

In the following, the boundary conditions arising due to current carrying conductors and those arising due to the material properties of sheet iron and the magnitude of the eddy currents will be discussed.

2.3.1 Current Carrying Conductors

It is reasonable to assume that the current carrying conductors inside the terminal box are thin and infinitely long. The assumption of thinness reduces the source region in the domain of analysis, as these wires may be totally excluded and treated as boundary conditions. If it is assumed that, when balanced, three phase currents are flowing through the conductors, there cannot be any magnetic flux lines linking all three conductors; the line integral of the field must vanish

$$\oint \mathbf{H} \cdot d\mathbf{s} = 0 \qquad (2.3.1.1)$$

around any path which encloses either all three conductors or none.

The scalar potential is definable for the whole region provided that suitable cuts are established which prevent a circuit from being closed in such a manner as to link a current.

Fig 2.1 depicts three conductors carrying three phase balanced currents. The lines joining the middle conductor with the one on the left and the other on the right are the imaginary barriers. Thus, at points lying on either side and infinitely close to the barrier, the values of Ω differ by the current enclosed, I. There is thus a discontinuity in the potential which is defined as

$$\Omega^+ - \Omega^- = I \tag{2.3.1.2}$$

2.3.2 Heavy Eddy Currents Flowing on the Walls

If strong eddy currents flow on the metal parts of the terminal box, the surface eddy currents will prevent flux penetration into the metal. Heavy eddy currents also imply that the conductivity of the walls is very high and that the walls form Neumann boundaries for the scalar potential, i.e the field is tangential. The mathematical description of this case in terms of field quantities would be

(a) $\mathbf{H}$ and $\mathbf{B}$ are zero inside the iron.

(b) $\mathbf{H} = -\nabla\Omega$

(c) $\mathbf{J_S} = \mathbf{n} \times \mathbf{H} = -\mathbf{n} \times \nabla\Omega$

(2.3.2.1)

(d) The normal component of $\mathbf{B}$ should be zero, i.e.,

$$\mathbf{n} \cdot \mathbf{B} = -\mathbf{n} \cdot \nabla\Omega = 0$$

i.e.

$$\frac{\partial\Omega}{\partial n} = 0 \tag{2.3.2.2}$$

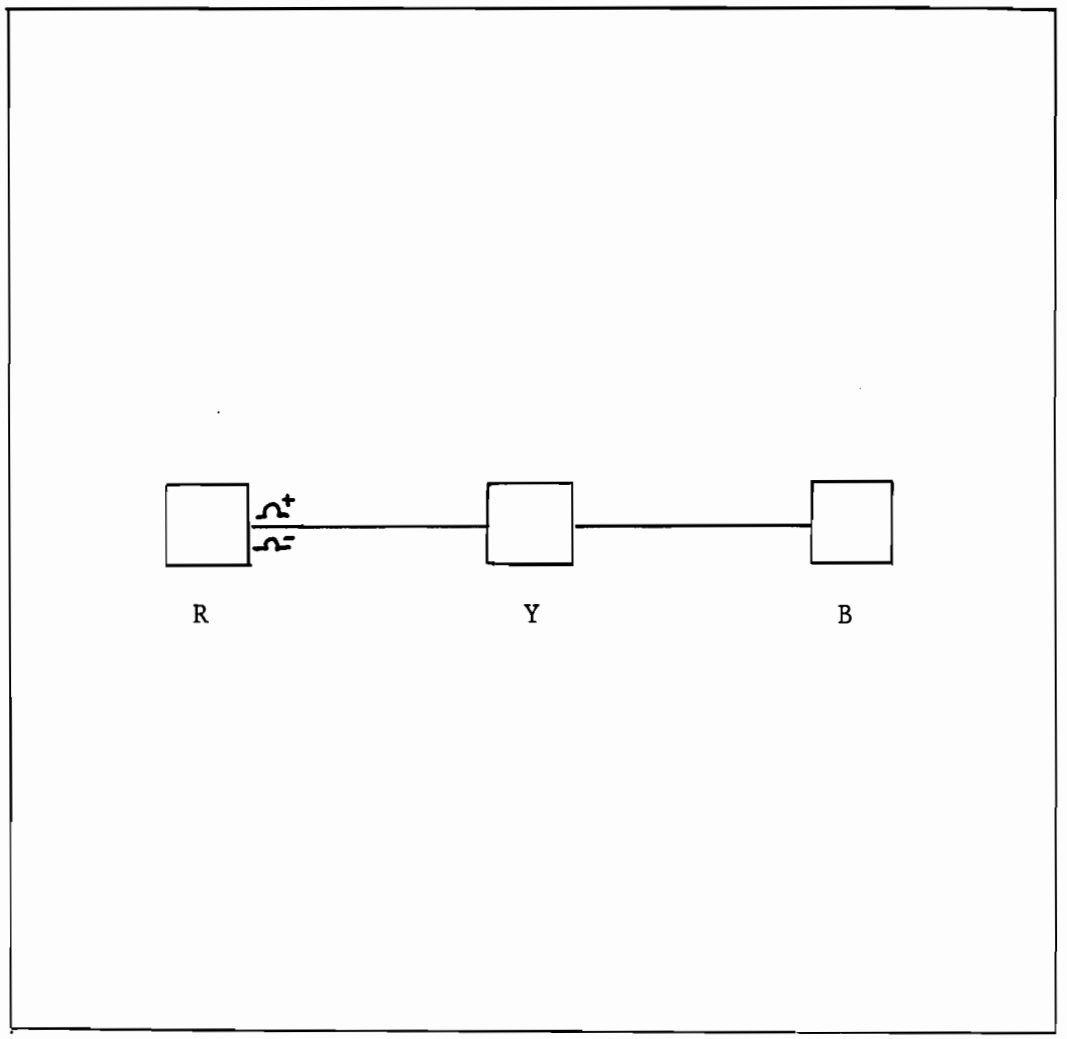


Figure 2.1 Terminals with the barrier

This interface condition is implemented as a natural boundary condition in the finite element approach being used.

2.3.3 The Walls Carry Eddy Currents of Finite Magnitude

This is the general case which can be described as:

(a) conductivity of iron, σ is finite.

(b) the permeability of iron, μ_{iron} is very high but finite.

(c) The first two conditions mean that:

$$\delta = \sqrt{\frac{2}{\omega\sigma\mu}} \ll t$$

where δ is the depth of penetration of the field and t is the thickness of the sheet.

This assumption is necessary in order to make the eddy currents appear as surface currents which can be adequately treated by a scalar potential. Otherwise, the eddy currents will be volume currents and a scalar potential alone is not appropriate. Fig. 2.2 depicts an air-iron interface and the variation of Ω across this interface.

(d) The tangential component of $\mathbf{H}$ across the interface will be given as

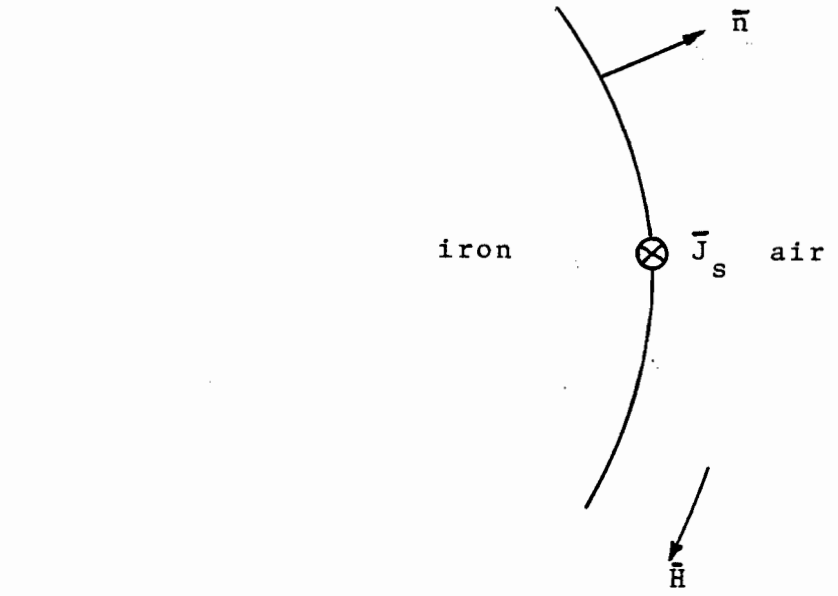
$$\mathbf{n} \times \mathbf{H}|_{air} - \mathbf{n} \times \mathbf{H}|_{iron} = \mathbf{J}_s \qquad (2.3.3.1)$$

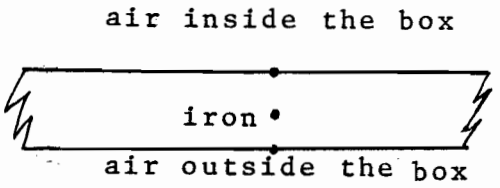
and, if $\mathbf{H} = -\nabla\Omega$, then

$$\mathbf{n} \times \nabla\Omega|_{iron} - \mathbf{n} \times \nabla\Omega|_{air} = \mathbf{J}_s \qquad (2.3.3.2)$$

This condition will be satisfied on each interface.

(e) Similarly, the normal component of $\mathbf{B}$ should also be continuous





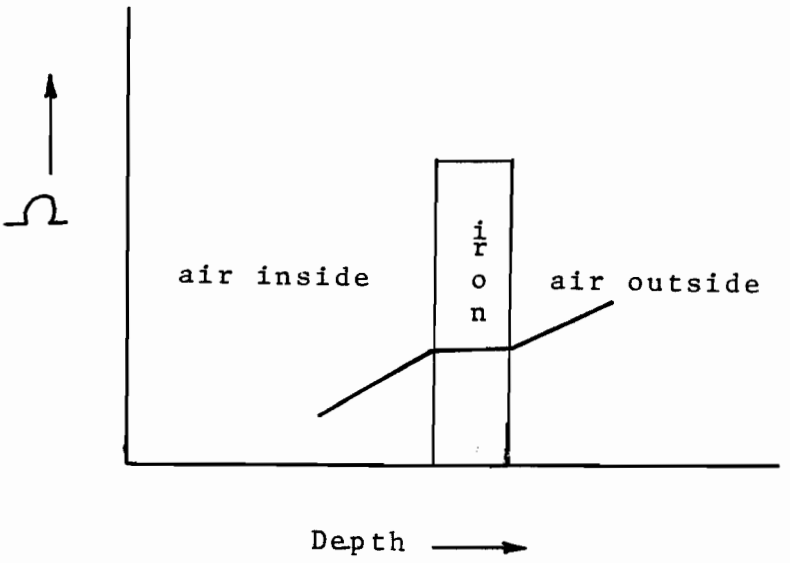


Figure 2.2 Air-Iron Interface

$$\mathbf{n} \cdot \mathbf{B}|_{air} = \mathbf{n} \cdot \mathbf{B}|_{iron}$$

i.e.

$$\mathbf{n} \cdot \mu_0 \nabla \Omega|_{air} = \mathbf{n} \cdot \mu_{iron} \nabla \Omega|_{iron}$$

$$\mu_0 \frac{\partial \Omega}{\partial n}|_{air} = \mu_{iron} \frac{\partial \Omega}{\partial n}|_{iron}$$

or,

$$\frac{\frac{\partial \Omega}{\partial n}|_{air}}{\frac{\partial \Omega}{\partial n}|_{iron}} = \frac{\mu_{iron}}{\mu_{air}} \qquad (2.3.3.3)$$

2.4 Element Matrices for Sheet Elements

Triangular prisms have been referred to as sheet elements, as these are convenient for modelling thin sheets or walls. In this section, the formulation for sheet elements for deriving the element Dirichlet matrix S and the element Metric T will be presented. The element matrices, S and T for tetrahedral elements, for the scalar Helmholtz equation are already reported [139] and therefore will not be derived here. Fig. 2.3 depicts a typical sheet element in which the surface in the x-y plane is modelled by a high order polynomial, and the thickness in the z direction is modelled only by first order. This is due to the fact that iron permeability is very high and therefore the flux is confined to the surface and does not penetrate in the iron. Thus, detailed modelling of the field in the direction of thickness is not necessary whilst the field external to iron needs to be modelled with precision.

On a triangle of order N, the potential variation can be written in terms of

$$M = (N+1)(N+2)/2 \tag{2.4.1}$$

interpolation polynomials, defined as

$$\phi = \sum_{k=1}^{M} \phi_k \alpha_k^N(x,y) \tag{2.4.2}$$

In order to ensure the potential continuity between triangular prisms and tetrahedral elements, a form of potential variation similar to that of a tetrahedron is to be employed for the triangular prism. As discussed earlier, detailed modelling of the field is not required in the z direction. Interpolation functions, therefore, may be chosen to be the cartesian products of first order, one dimensional interpolation functions in the z direction and higher order, two dimensional interpolation functions for the $x-y$ plane, defined by equation (2.4.2):

$$R = \sum_{i=1}^{2} \sum_{k=1}^{M} R_{ik} \alpha_k^N(x,y) \alpha_i^1(z) \tag{2.4.3}$$

For the sake of simplicity in derivation, assume $i = 1$ denotes the nodes on the top face of the elements, and $i = 2$ denotes the nodes on the bottom face of the elements. Therefore, R can be written as:

$$R = \sum_{m} R_m \alpha_m(x,y,z) \tag{2.4.4}$$

where, for $m \leq M$:

$$\alpha_m = \alpha_1^1(z) \alpha_m^N(x,y)$$

and, for $m > M$:

$$\alpha_m = \alpha_2^1(z) \alpha_{m-M}^N(x,y) \tag{2.4.5}$$

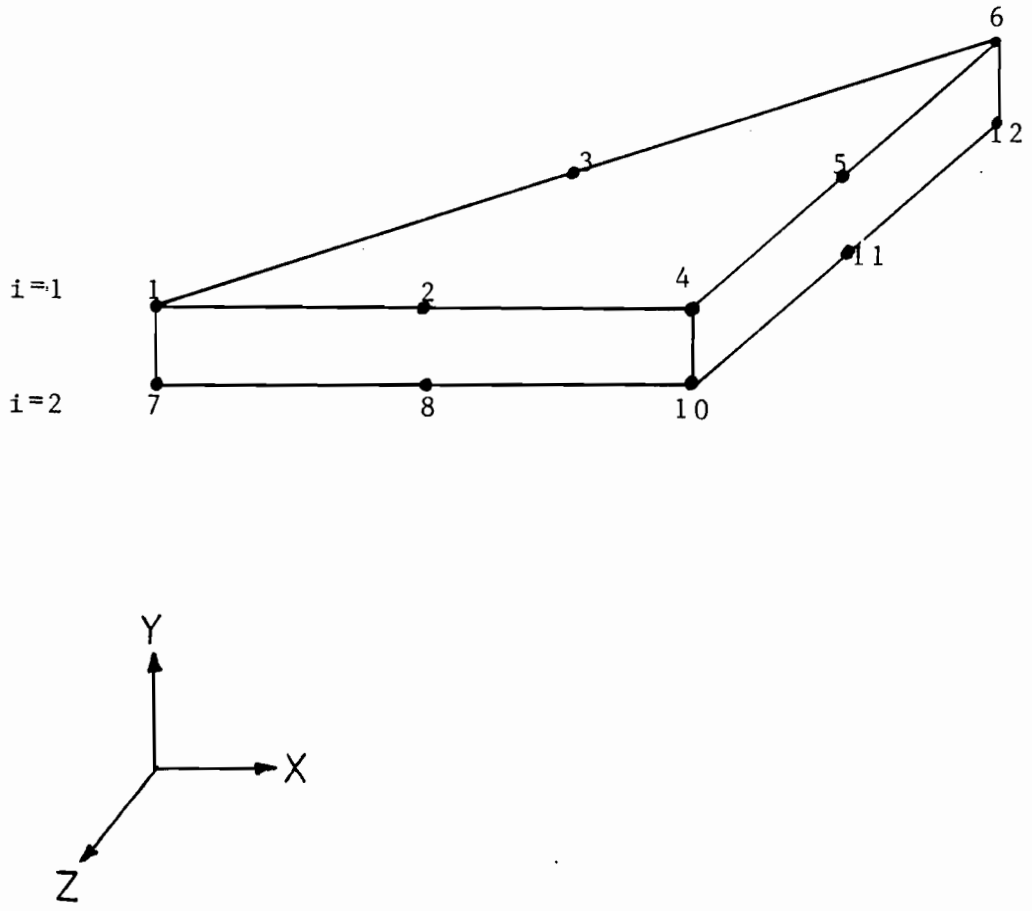


Figure 2.3 A Triangular Prism

The element Dirichlet matrix S for the triangular prism element is obtained by evaluating

$$S_{kl}^{(3)} = \int \nabla \alpha_k \cdot \nabla \alpha_l d\Omega \tag{2.4.6}$$

If $i - j$ are considered to be one dimensional, and $m - n$ to be two dimensional indices then, with the help of (2.4.3), $S_{kl}^{(3)}$ can be expanded as:

$$\begin{aligned} S_{kl}^{(3)} = & \int \nabla \alpha_m^N \cdot \nabla \alpha_n^N dS \int \alpha_i^1 \alpha_j^1 dz \\ & + \int \alpha_m^N \alpha_n^N dS \int \frac{\partial \alpha_i^1}{\partial z} \frac{\partial \alpha_j^1}{\partial z} dz \end{aligned} \tag{2.4.7}$$

Here, the differential dS refers to the plane of the triangular element and dz is the line element in the direction of the thickness. The gradient operator in equation (2.4.7), should be considered to operate on the x-y plane only. If integration is performed using simplex coordinates, the integrands finally reduce to

$$S_{kl}^{(3)} = t T_{ij}^{(1)} S_{mn}^{(2)} + A S_{ij}^{(1)} T_{mn}^{(2)} \tag{2.4.8}$$

where

$$S_{mn}^{(2)} = \int_S \nabla \alpha_m^N \cdot \nabla \alpha_n^N dS \tag{2.4.9}$$

and,

$$T_{mn}^{(2)} = \frac{1}{A} \int_S \alpha_m^N \alpha_n^N dS \tag{2.4.10}$$

are the Dirichlet matrix and the Metric for a triangular element, whose area is A. Also,

$$S_{ij}^{(1)} = \int_l \frac{\partial \alpha_i^1}{\partial z} \frac{\partial \alpha_j^1}{\partial z} dz \tag{2.4.11}$$

and

$$T_{ij}^{(1)} = \frac{1}{t} \int_l \alpha_i^1 \alpha_j^1 dz \tag{2.4.12}$$

are the Dirichlet and Metric matrices for a line element, where t is the distance between two nodes. In this case, it is the thickness of the sheet elements.

Similarly, Metric for a triangular prism is obtained using the same product functions:

$$T_{kl}^{(3)} = \int \alpha_m^N \alpha_n^N dS \int \alpha_i^1 \alpha_j^1 dz$$

which can be simplified by using (2.4.10) and (2.4.12) to

$$T_{kl}^{(3)} = V T_{mn}^{(2)} T_{ij}^{(1)} \tag{2.4.13}$$

where $V = tA$, is the volume of the triangular prism.

2.5 Treatment of the Sheet Edges

The accurate accounting of the flux passing through the sheet edges requires a special numerical treatment. Fig. 2.4. illustrates the situation where sheet elements and tetrahedra form an interface. The nodes 1, 2 and 3, 4 belong to different tetrahedra which have been

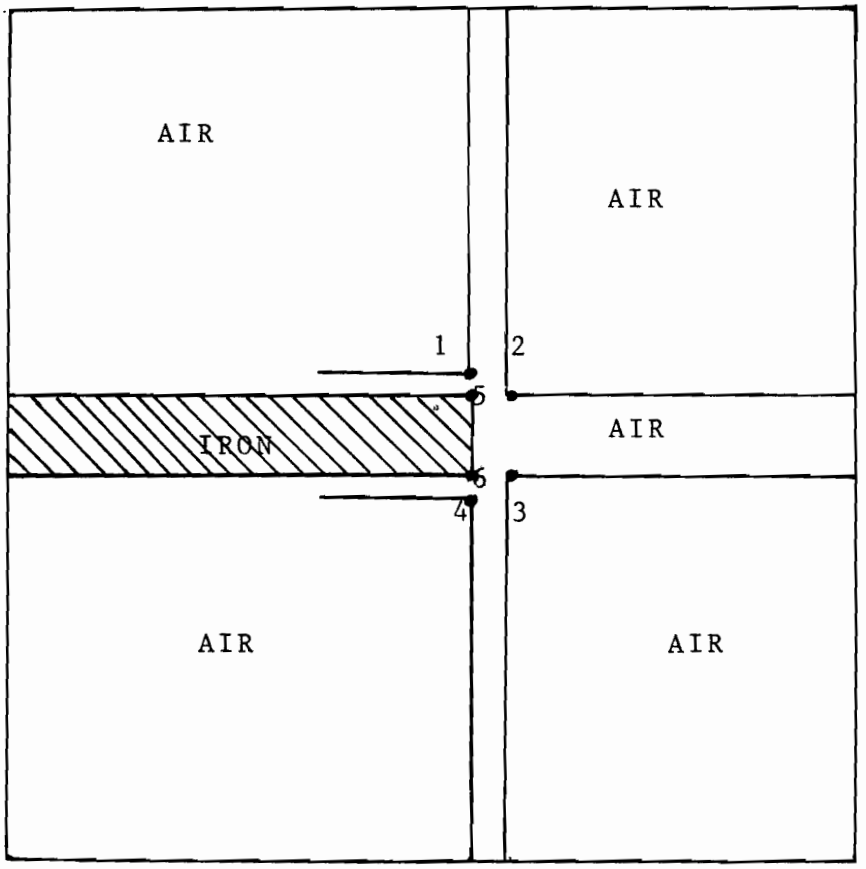


Figure 2.4 Treatment of Sheet Edges

chosen to model the air; and the nodes 5, 6 belong to the sheet element. Since the sheet is very thin when compared to the other lateral dimensions, very little flux will enter through the edges as most of the flux entering the sheet will either come from above or below the sheet. The small flux which passes through the edges, means a very small scalar potential difference between the nodes 5 and 6. The difference in potential is small enough that these nodes can be treated as being the same potential, i.e. no flux is allowed to pass through the edges. Thus, all the nodes inside the square in figure 2.4 will have the same scalar potential, i.e. these nodes are magnetically equal but geometrically different.

Alternatively, the sheet edges can also be treated numerically by grading the finite element mesh around the edges. This method of treating the edges has the advantage of simplicity, and can allow for cases where long edges are alone or where two or three edges are joined together, as in the case of a terminal box. In cases where there may be an unknown amount of flux passing through the edges, mesh grading is the only possible choice available to account for it. The mesh grading approach, however, is not that simple to implement in three dimensions, as it can introduce a considerable number of elements in all the three coordinate directions, extending up to the boundaries.

Chapter 3

Mesh Generation

3.1 Introduction

Application of the finite element method in magnetics is extensive and well developed. Although the computational efficiency of general finite element programs has steadily improved over the past years, the time consuming and error prone process of accurate data preparation is still a burden for most finite element users. For this reason, a great deal of effort has been expended in the development of finite element pre-processing systems.

It is a fact that growth in pre-processing systems has been largely due to the fast technological developments in the field of interactive graphics, starting from refresh vector graphics to the present state of raster graphics. The use of raster graphics in pre-processing systems has helped convert the operation of geometric modelling and discretization from manual to semi-automatic. Also, graphics echo is used for error checking, thereby simplifying the coding. The emphasis has always been to minimize the manual input in the modelling phase, making the system as automatic as possible. Currently the mesh generation in 2D systems (e.g. MagNet-11 [100], PE2D [120], FLUX [134] etc.) is almost automatic, and manual input is required only for the boundaries of the geometries.

Whilst geometric modelling and mesh generation have matured relatively in two dimensions, they are still under development in three dimensions. Recent developments in 3D modelling have been attributed largely to the current advances in display systems which have resulted from the use of powerful microprocessors (National 32016, Motorola 68000, etc.), high capacity disks, and custom designed VLSI chips. Incorporation of these developments has resulted in intelligent raster display terminals which have high resolution colour screens, multiple bit-planes and colour maps and allow display of lines in different styles.

These developments, though they have enabled two dimensional pre-processors to be established on a strong footing, do not provide enough help in three dimensional pre-processor development. The mathematics behind 3D transformations such as scaling, translation, and rotation about arbitrary axes using matrix methods have been known for many years. On 2D displays, these transformations are performed by software, so that when a view of 3D model is altered, its 2D projection must be recomputed and retransmitted to the display. This process takes many seconds and as a result rules out any hope of real-time rotations as a useful aid to visualization. However, some newer displays provide this functionality in the terminal itself. If many rotations and translations are involved, some information is also lost and then the display and its interpretation become difficult. Real progress in three dimensional modelling will only be possible if problems involved with the input/output images of 3D bodies are solved, both at the hardware and the software levels.

The pre-processor, in essence, consists of two distinct phases. In the first phase, a model is created which defines to an appropriate accuracy the shape or the geometry of the component to be analyzed whilst the second phase creates a suitable finite element mesh. Ideally, these two phases should be independent in the sense that the model should be allowed to be manipulated, edited, and adjusted before the discretization process begins.

However, in many systems, these two operations have been combined into one.

3.2 Desirable Features of an Automatic Mesh Generator

Features desired in an automatic mesh generator can be summarized as:

(1) Precise modelling of the geometry: the meshing should not introduce any error beyond discretization error in the finite element model.

(2) Good matching between the interior mesh and the information on the mesh boundary: the interior of the mesh should reflect the boundary curvature and the node spacings; this facilitates the control of shapes of the elements and mesh refinement, if required.

(3) Minimal manual input: this will reduce the time and the effort required to obtain a finite element solution and, therefore, the cost of the solution; at the same time chances of human error will also be reduced.

(4) Topological properties: the mesh generator should be capable of handling a broad range of geometric topologies without imposing any restrictions on the topology of the mesh within a region. The element connectivities should also be created by the mesh generator without any user intervention.

(5) Favourable element shapes: the elements created by the mesh generator should have good aspect ratios; ill-shaped elements contribute to the ill-conditioning of the coefficient matrix and thus instability.

(6) Ordered database: the numbering of the nodes and elements within the model should be such that it will result in a lower bandwidth, frontwidth, and profile of

the coefficient matrix. These features are necessary for reducing the cost of the solution of equations.

(7) Response time: the mesh generator should provide reasonably fast response time.

(8) Resource optimization: the method of mesh generation should optimize the use of computer resources, so that the expenses involved in the input data preparation can be minimized.

3.3 Mesh Generation Techniques in Structural Mechanics

For over fifteen years investigators have been developing mesh generation techniques and designing finite element pre-processing systems to reduce the effort required in producing finite element models. Although these efforts have reduced drastically the cost of generating a finite element model, there are still major efforts underway which are concerned both with developing flexible three dimensional mesh generators for use in interactive pre-processing, and in integrating these procedures with CAD systems. Currently available mesh generators both in 2D and 3D can be classified as employing one or more of the following techniques:

1. Automatic triangulation
2. Coordinate transformations
3. Smoothing procedures
4. Blending functions

Although some mesh generating techniques are generally better than others, none have proven superior or entirely satisfactory for a broad range of applications.

Coordinate transformations and blending function techniques are currently very popular because of their ability to produce well conditioned meshes. The most popular transformation technique is based on the isoparametric coordinate concept [179]. Isoparametric mappings require that each side be represented as a specific order polynomial and that opposite sides have an equal number of node points. Some of the restrictions can be relaxed by using the Schwarz-Christoffel transformation [16].

Much of the current emphasis in the development of mesh generating techniques is on the use of blending functions in the form of transfinite mappings [69], [64], [10], [70]. This is a family of methods, useful for the approximation of complex surfaces and volumes, which describes an approximate surface that matches a true surface at all points lying on a series of interpolation curves that are embedded in the true surface. These methods do suffer from general restrictions of a fixed number of regions and equal number of node points on opposite sides, but the definition of what constitutes a region side is much less restrictive than for the isoparametric mapping approach. These mesh generators have proven extremely effective in interactive graphic pre-processors.

Smoothing procedures [72], [94] are normally used in conjunction with one of the other techniques to improve the shape of the elements by repositioning the nodal locations within the mesh generated by that technique.

The only approaches that show any promise of automatically generating a mesh for any geometry are the automatic triangulation techniques [20], [32], [135], [161]. Although algorithms based on this approach have been developed that will work for general two dimensional shapes, they have not proven popular because they often produce ill-conditioned meshes and, in many cases, still require a substantial amount of user input. Recently, Coulomb et al. [43] and Cendes and Shenton [35] have successfully implemented this 'Delaunay Triangulation' technique for meshing three dimensional domains in magnetics.

Although there are several different approaches used in the automatic triangulation algorithms, there are basically two different classes [161] and [135] that have proven popular in the literature. In both classes the boundary of the domain is discretized by placing nodes along the boundary curves. From that point on one technique triangulates by using the boundary polygon and cutting off sharp corners. Selected points on the boundary are replaced with new ones on the interior near the replaced point. Each of the areas removed are one or more triangular shape and are stored as elements. One such approach first removes each vertex of the polygon with an angle less than 90 degrees, thus creating an element for each vertex removed. After all those vertices are removed, the ones with angles less than 180 degrees are replaced by the introduction of new nodes placed in the interior based on the coordinates of the removed node and its two neighbours. The area removed is then broken into two triangular elements. If this step creates any vertices less than 90 degrees, the vertices are immediately removed. The process of introducing new nodes and cutting vertices is continued until three points remain in the polygon which then defines the last element.

In the other technique, additional interior information is incorporated into the triangulation procedure. For example, Bykat's algorithm [20] first breaks the domain of interest into simple convex subregions and then triangulates those subregions into elements. Cavendish's algorithm [32], on the other hand, places node points throughout the interior of the domain and then proceeds to triangulate the domain using the boundary nodes. These algorithms could be classified as an outside-inside procedure in that they are strongly based on boundary information but use additional interior information. It is common to apply some form of smoothing procedure with any of these algorithms to improve the shape of its resulting elements.

3.4 3D Mesh Generators

There are a number of mesh generators, for example FEMGEN, ANSYS, CALMA, NISA, PATRAN-G, MOVIE, SUPERSAP, MENTAT, PADDY, and SCARPIA which are commercially available and have been developed primarily for use in mechanics, with the exception of the last two which are for magnetics. Excellent reviews/descriptions of these are available in [9], [55], [74], and hence will not be repeated here. Instead, the principal attributes of only a few prominent ones will be summarized in table 3.1

Even though there are many versatile mesh generators commercially available, three dimensional mesh generation is still an active area of research, mainly for the following reasons:

(a.) to improve the communication between analyst and machine.

(b.) to automatize triangulation of solids with well shaped elements and user controlled mesh densities.

(c.) to minimize manual input.

(d.) to make the system general purpose and simple to use.

In the next section some of the mesh generators which are currently at the research level will be discussed briefly.

3.5 Mesh Generators at the Research Level

Some of the more recently developed three dimensional mesh generators are by Nguyen [119], Perrucchio, Abel et al. [121], Pissanetzky [124], Bryant and Freeman [17], [18], and Coulomb et al. [43]. The first three were developed or are being developed for use

Attributes		FEMGEN	Display/ Digit	Patran-G	Medusa	SCARPIA	ANSYS
Data Display	Interactive	*	*	*	*	*	*
Geometry Definition	Implicit		*	*	*	*	*
	Shape Description Language	*	*	*	*		
Automatic Mesh Generation	2D	*	*	*	*		*
	3D	*	*	*	*	*	*
Data Checking and Viewing	Complete Model	*	*	*	*	*	*
	Part of Model	*	*	*	*	*	*
	Rotation etc.	*	*	*	*		*
	Perspective, Isometric views	*	*	*	*		*
Data Editing	Batch	*				*	
	Interactive	*	*	*	*		*
Operational on	CDC	*	*	*			*
	DEC		*	*			*
	PRIME		*	*	*	*	*
	IBM	*	*				*
	Univac	*	*				*
	Other	*	*	*			*

Table 3.1

in structural mechanics while the last two are intended for use in magnetics. The mesh generator by Nguyen is a non-interactive mesh generator developed for analyzing network structures. It requires manual definition of all the nodes in the model before mesh generation can begin. The mesh generation procedure connects all these nodes to a continuous network structure consisting of tetrahedral elements. Mesh refinement is available in the form of splitting each tetrahedron into eight smaller tetrahedra.

The KUBIK-system developed by Pissanetzky is also non-interactive and requires manual entry of the node coordinates. It starts by generating a set of modules (planes) and then the modules are linked according to the connection matrix which stores the connectivities. The system uses irregular bricks, called cuboids, which in turn can be divided into tetrahedra or triangular prisms. A mesh refinement facility is also available. The program runs on IBM-360 and requires a storage of 128 Kbytes to generate a 1500 noded model.

The mesh generator developed by Perrucchio et al. is interactive. It uses a digitizing tablet and menus for the input data to reduce the manual entries from the keyboard. The plane cross-sectional meshes are generated by combining discrete transfinite mappings and cubic spline blending algorithms. These plane cross-sectional meshes are then used to generate a three dimensional geometry by interpolation between the cross-sections. The mesh is made of first order hexahedral elements and it is possible to examine the interior of the mesh by displaying the corresponding layer. The major limitations are that the defining cross-sections must be planar and must show a degree of compatibility. For non-planar geometries manual input of data is required. Multi-view dynamic display is also employed, which helps in element and node selection while editing the mesh. It runs on a VAX 11/780 connected with an Evans and Sutherland Picture System 2 vector display.

Bryant and Freeman's mesh generator is a highly interactive system which minimizes the input from the keyboard and is essentially an extension to 3D of interactive 2D techniques.

It requires the definition of all the nodes forming the outline of the model in a plane. The nodes lying between two points (already defined) on a straight line or an arc can be generated automatically. As many planes as desired can be defined. Then two planes are joined to form a segment and by joining the segments, a geometry is modelled. The mesh can be generated using tetrahedra, pentahedra, hexahedra or their combinations. The hardware requirements of this system are a LSI-11/23 with a KEF-11 floating point processor, 128 Kwords of random access memory, two 20 cm floppy drives, a 20 Mbyte hard disc, a digitizing tablet, and colour graphics support. This system was considered for the work reported in this thesis but some difficulties were experienced whilst building the meshes using hexahedral bricks. It was observed that if the model required 20 or more elements then it became difficult to differentiate between the already defined elements and the elements yet to be defined with the remaining nodes. It is apparent that the situation will be worse when tetrahedra or pentahedra are used as the main elements in constructing the complex models. This situation is not unique to this particular generator but is inherent to any highly interactive system and is caused by the limitations of 2D graphics displays.

Coulomb et al. have proposed two solutions for 3D automatic mesh generation. One is a parameterized topological mesh generator and the other is an automatic mesh generator with tetrahedral elements which can also be parameterized. Their second mesh generator is better suited for general applications and the ideas behind it can be summarized as follows:

1. Define the domain by parameters and geometric points.
2. Apply Delaunay triangulation over all the points.
3. If the geometry consists of topologically different parts, then model them separately.
4. Apply a coarse triangulation and then group them together.
5. Generate elements.
6. Finally, regularize and refine the mesh.

3.6 Need For Developing A Mesh Generator

The mesh generator used in this thesis has been developed out of a necessity to overcome the complexity and generality of available systems. It was primarily developed for modelling the turbogenerator terminal box and the surrounding air space. While investigating the possibility of using one of the existing, commercially available three dimensional mesh generators, the following limitations needed consideration:

(1) Most of the mesh generators which are capable of accomplishing the task can only run on main-frame or super minicomputers.

(2) Some packages require system dependent software which makes them untransportable.

(3) Most of the packages require dedicated machines and special graphics hardware.

(4) The cost of leasing or buying these software packages is high.

(5) Some of the packages may cause heavy system loading.

These considerations had a significant impact on the decision to develop a new mesh generator having the following desirable attributes:

(i) is simple to use.

(ii) runs on small machines.

(iii) has primitive graphics.

(iv) is portable.

(v) is not costly to run.

3.6.1 Choice of Elements

The mesh generator developed for this thesis is capable of modelling relatively complicated geometrical domains, including the terminal box, and has features required by the potential formulation chosen, such as creation of potential barriers, etc. The main restriction on the mesh generator, however, is that it be capable of running on machines with relatively limited address spaces.

Irregular bricks were chosen as the basic building blocks because they offer simplicity in modelling, while simultaneously handling complicated boundary shapes providing relatively uncluttered displays. In addition, these bricks can be divided easily into tetrahedral elements which have faster global matrix assembly times.

The second important reason for choosing hexahedral bricks as basic building blocks is that they can easily be divided into triangular prisms. Triangular prisms are ideally suited for modelling walls or thin sheets of high permeability ferromagnetic materials often encountered when electromagnetic devices are analyzed. In the analysis of the terminal box, the walls forming the box were modelled using triangular prisms.

3.6.2 Mesh Generation Philosophy and Distinct Features

Geometries of most electromagnetic devices possess irregularities in one direction or another and cannot be modelled adequately using regular hexahedral bricks. In fact, they require a mesh generator which can model irregular shapes. Therefore, a mesh generator using irregular bricks has been developed as an extension of one based on regular bricks [54]. The developed system is an interactive, semi-automatic mesh generator developed to

run on small machines of limited memory, i.e. machines with 32Kbyte to 1/2 Megabyte of memory.

The model is constructed from hexahedral bricks of regular or irregular shape. Each brick lives within a three dimensional lattice and may be described by joining together two dimensional slices through the lattice. Once a basic set of bricks has been created, the user is allowed to refine interactively the mesh within any brick by creating smaller bricks, the only rule being that the subdivision on adjacent faces of large bricks must match. Finally, when the mesh is complete, an automatic mesh generator is used to create six second order tetrahedra within each hexahedron. The subdivision into six tetrahedra per hexahedron was chosen, rather than the minimum set of five, because of the simplicity in matching cuts along adjacent hexahedron faces.

The task of generating complex models has been simplified by modelling each topologically different part of the geometry separately and then discretizing them into tetrahedra using the condition that the adjoining faces should be compatible with each other. The facility of interactive display of the geometry on a user defined arbitrary cut plane provides a visual response of the mesh and is a powerful tool for checking the validity of the mesh.

After each part has been discretized and checked by displaying one or several cross-sections, all the parts can be automatically grouped together to produce the combined database for the entire model. Once again the user can check the mesh by taking slices on arbitrary planes and displaying them. Thus, overlapping elements or ill-shaped elements can often be visually detected. Generating the mesh separately for topologically different parts not only assures a valid mesh at the end but also makes the task of localized editing much easier and faster as the database involved is kept small. In addition, the system response time is improved by generating and editing the individual parts separately.

In order to minimize input from the keyboard, the user is provided with powerful geometric operations (e.g..copy and join) to reduce the manual effort involved in modelling geometries which are repetitive or have symmetry in any or all three coordinate at a certain space interval, in any or all the three coordinate axes. The program can automatically generate the parts, join them, and finally discretize them. The only input required is the space interval(s) and the respective coordinate axes.

The mesh generator offers a choice of two elements, tetrahedra or triangular prisms. Triangular prisms are automatically produced by cutting the hexahedral bricks into two triangular prisms instead of six tetrahedra. In order to keep the triangular prisms and the tetrahedra compatible with each other, the direction of the cut in the case of prisms is kept the same as in the case of tetrahedra.

The Fig. 3.1 depicts the cutting of one hexahedral brick into six tetrahedra; Figs. 3.2 and 3.3 show, respectively, the division of hexahedra into subhexahedra and the direction of cuts. Some of the geometries which have been modelled using this program are shown in figures 3.4 and 3.5.

3.6.3 System Requirements

This mesh generator requires a LSI-11/23 with 32Kbyte of memory, an alphanumeric keyboard, a graphics support (black and white or colour) and two 20 cm floppy disk drives. The program has been written in standard Fortran and also runs on Perq and VAX computers. No digitizing tablet is required.

3.6.4 Input/Output Features (Data Structures)

The x, y, and z coordinates of the eight vertices in each hexahedron are required if the

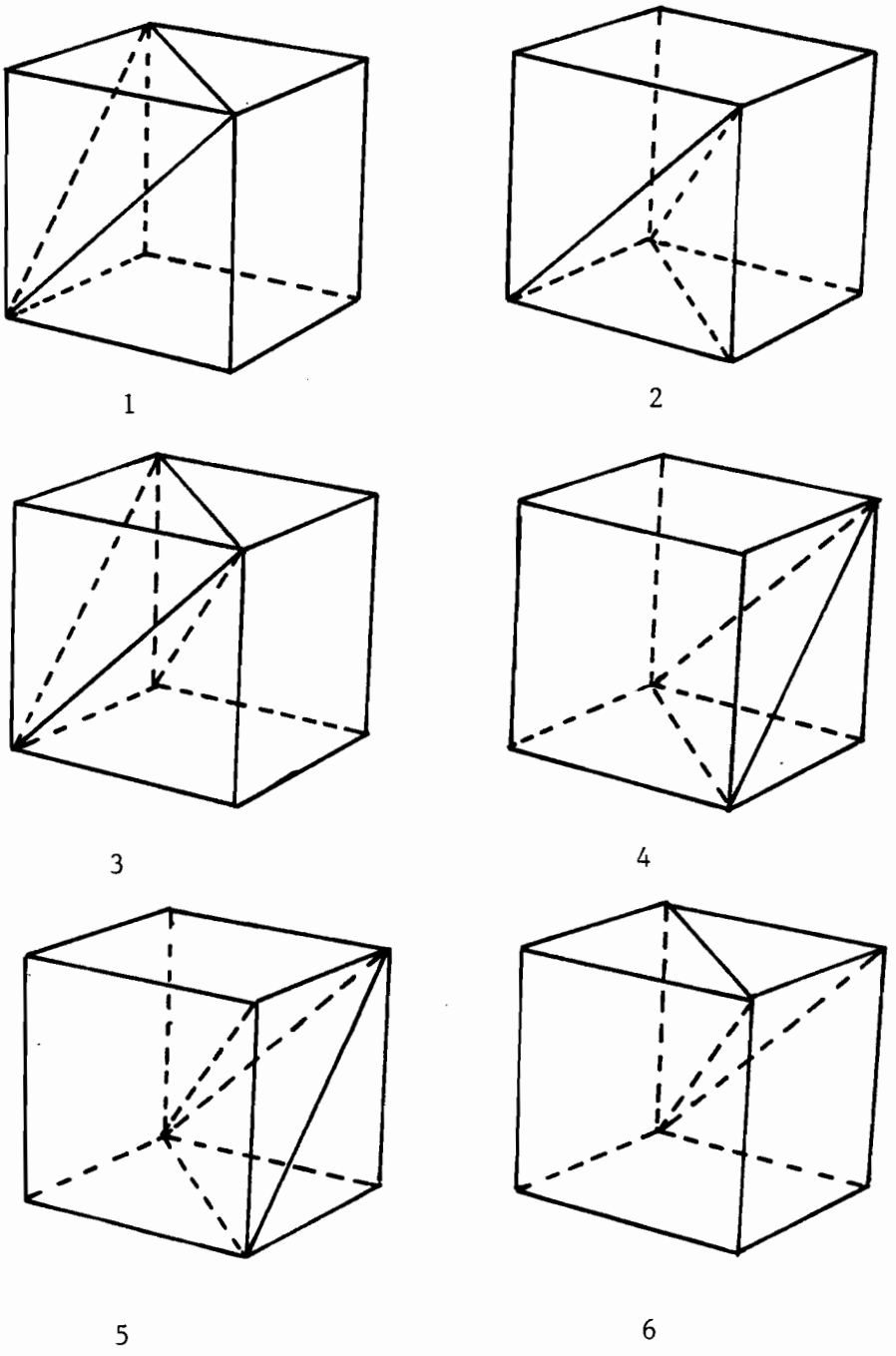


Figure 3.1 Subdivision of one hexahedron into six tetrahedron

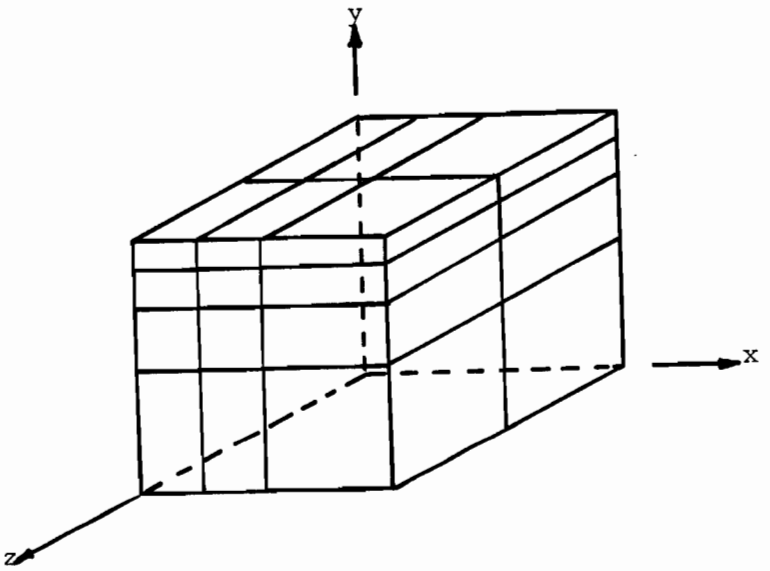


Figure 3.2 Division of a hexahedron into subhexahedra

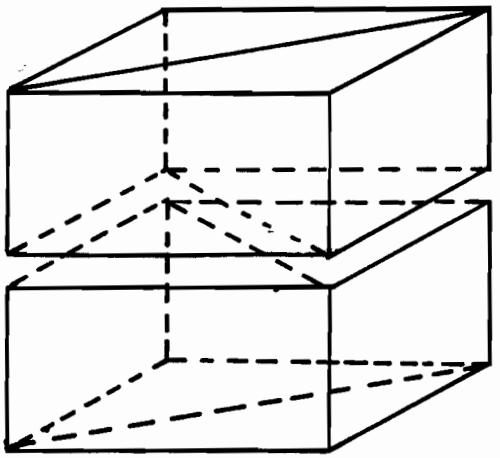


Figure 3.3 Direction of cuts

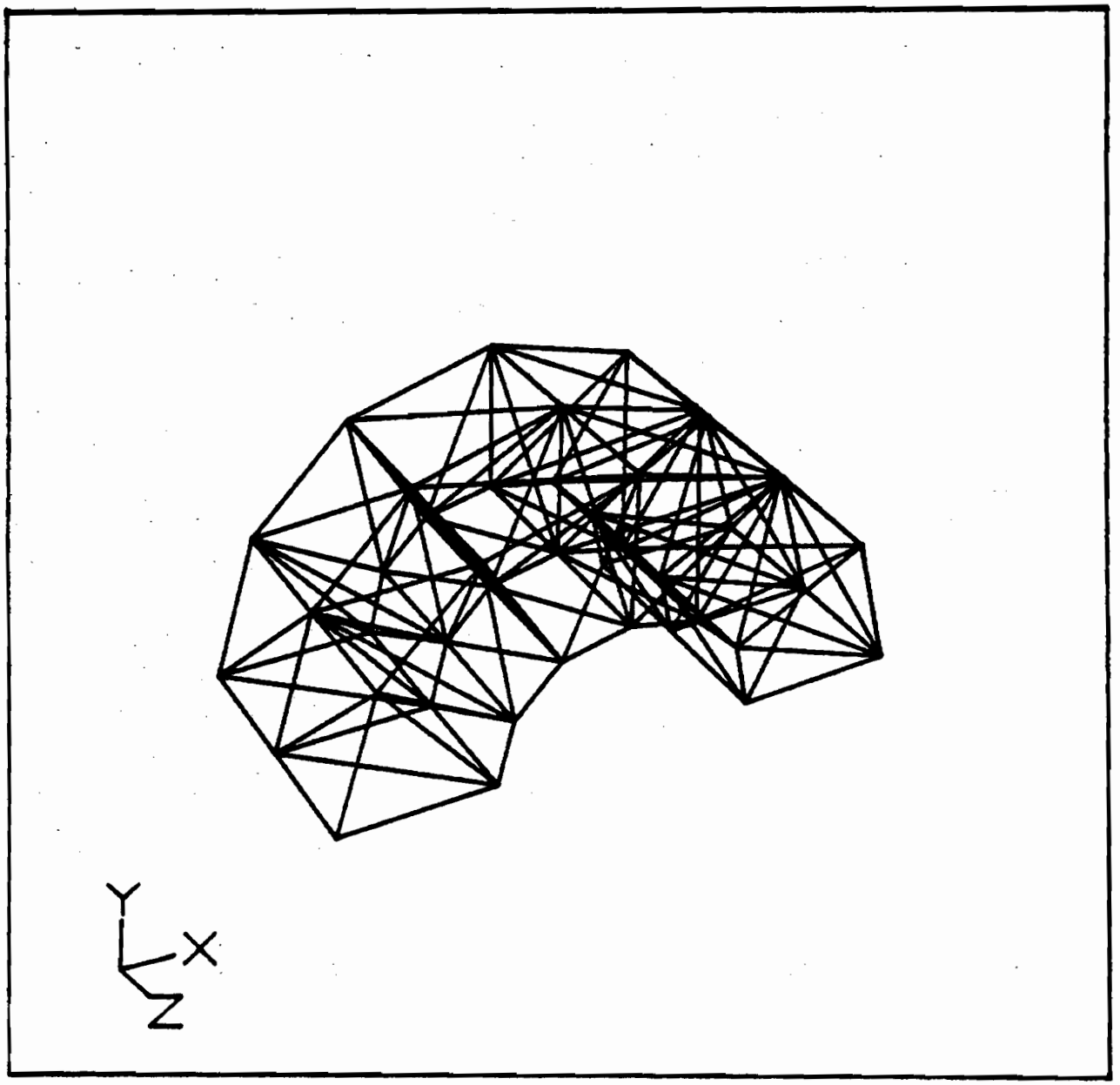


Figure 3.4 A discretized model

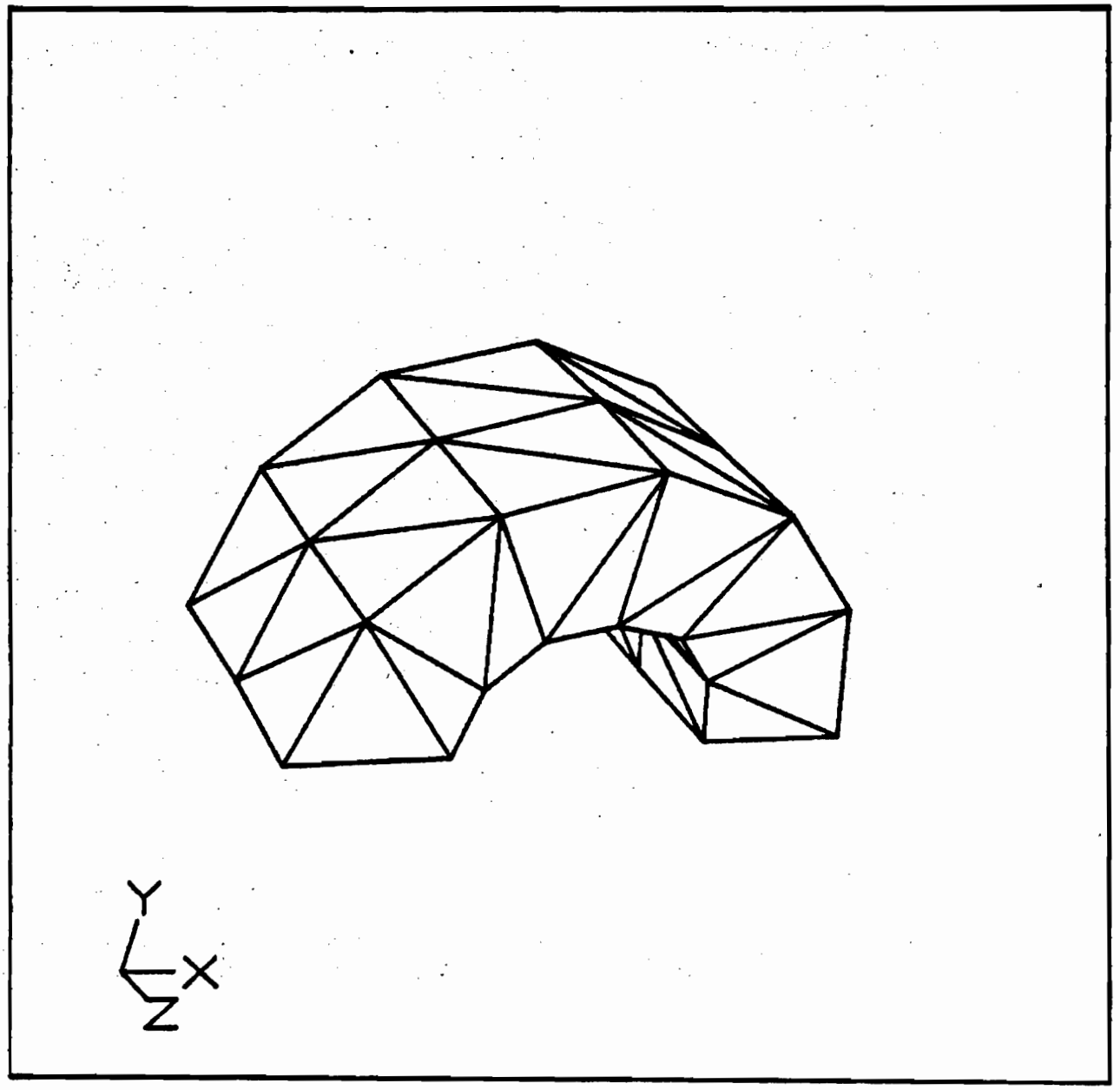


Figure 3.5 Model with hidden lines removed

brick is of irregular shape; only four need be given if it is of regular shape as input via the keyboard. Also required are the number of subdivisions (subhexahedra) along each axis and an alphabetic character to identify the material property of the element involved. The program generates two output files, the first contains the node numbers of each tetrahedron sequentially, element by element, along with the label specifying the material property of the element. The other file stores the x, y, and z coordinates of each node listed sequentially, in increasing order of their numbers.

3.6.5 Memory and CPU Time Requirements

The memory requirements for carrying out the triangulation, i.e. the division of each hexahedron into subhexahedra and then the subsequent division of subhexahedra into tetrahedra, is $O(n^3)$, where n is the number of subdivisions along each axis. Other memory requirements are dependent on the number of nodes (N) the model will have. Three real vectors are required to store the x, y and z coordinate values associated with each node. In addition, a pointer vector of length $(N, 4)$ is required to store the node label and three pointers to the locations of x, y, and z coordinate values in the database.

In order to have a fast response time while building fairly large models (limited only by machine memory), the choice of suitable data structures becomes critical. The approach used in this mesh generator is that of an octal tree for maintaining the node coordinate lists. The octal tree is, essentially, a development of the data structures used in some of the two dimensional mesh generators. The geometric database is kept in an extremely ordered fashion such that all searches for nodes may run in approximately 'binary' (nlogn) time. Thus, the geometric modeller provides fast access to any particular node and the system is node dependent rather than element dependent. This approach has another advantage

in that the entire node list need not be in memory at any one time. This gain in memory requirement can be used for modelling larger problems.

3.7 Practical Experience

The method of mesh building in the new mesh generator is by synthesis, i.e. the entire solid is modelled by filling in the elements one by one. This approach is currently prevalent in most of the two dimensional and some of the three dimensional mesh generators, e.g. KUBIK, SCARPIA, and the mesh generator developed by Bryant and Freeman. One of the advantages of this approach is that it results in a unique mesh for the model being discretized, as opposed to the analytical approach in which many triangulations of the model are possible.

As described earlier, in this approach first the boundaries of the model are constructed with hexahedral elements and then the entire region is filled with similar elements to complete the model. The filling of the elements inside the boundaries is also done element by element. While filling these blocks in presence of several other blocks, care has to be taken so that the faces or sides common to other blocks match in space.

Experience with the mesh generator suggests that it is preferable to work with a few larger blocks rather than many smaller blocks - if the bigger blocks in no way affect the shape of the boundaries - and then subdivide the bigger blocks in as many smaller blocks as desired. Following this procedure, relatively large and complex models can be built quickly. While there is no restriction on the size of the blocks it is preferable, however, that the ratios of lengths in all the three coordinate axes be as close to unity as possible to prevent generation of deformed tetrahedra when the blocks are subdivided.

Simple geometries which do not involve cylindrical or curved surfaces can be very well treated by the mesh generator with minimum input. But should they involve such surfaces, a considerably large number of blocks may be required to model the curves or the circumference with piecewise linear segments and the manual input may then become large. The interesting point is that it can be done.

The use of interactive graphics is not extensive and, in fact, can be totally dispensed with if desired. Graphics come into play only if the finished model or a cross-section of it is to be displayed for error checking or for some other reason. The total independence from the graphics hardware and the ability to run on small machines makes it a remarkably low cost system.

Using this mesh generator, a variety of topologically different models consisting of 120 to 12000 elements have been built. The most complex model built was that of a terminal box which had 11772 elements and 3010 nodes. It took at least 36 hours to build such a model, which includes manual data preparation and feeding of data into the computer. Models which are relatively simple or possess symmetries can be built in a few hours.

3.8 Limitations

Although the mesh generator provides a low cost approach to three dimensional mesh generation of relatively complex models, it does have some limitations.

The main limitation of the system is that once the number of subdivisions in each of the coordinate axes for the principal hexahedra are decided, the same number of subdivisions are to be kept in adjoining hexahedra and, as a consequence, it becomes difficult to choose different discretization densities at different regions in the same model. At times this restriction leads to a large number of elements in the model.

Chapter 4

Element Reordering For Frontal Solutions

The frontal method is usually associated with direct methods for solving large sparse systems of linear algebraic equations with symmetric positive definite coefficient matrices. The advantage of the frontal method is that it allows the solution of such problems with a minimum of computer memory and, as a result, fairly large problems can be solved on machines with limited address spaces. In the following section a brief description of the frontal method will be given; more details are available elsewhere [76].

4.1 The Frontal Method

The frontal method is considered to be the most natural solution scheme, since it operates directly on the underlying structure of the finite element mesh. With this approach, the finite elements are assembled and entered into the solution one at a time. A nodal variable is eliminated as soon as all the neighbouring finite elements connected to it are available. The remaining, partially assembled nodal variables which are not yet ready for elimination are retained in the 'front'. Fig. 4.1 illustrates a typical situation while a problem is being solved using the frontal procedure. It is customary to divide all the nodal variables into three parts:

x_d: Those variables which are already decomposed.

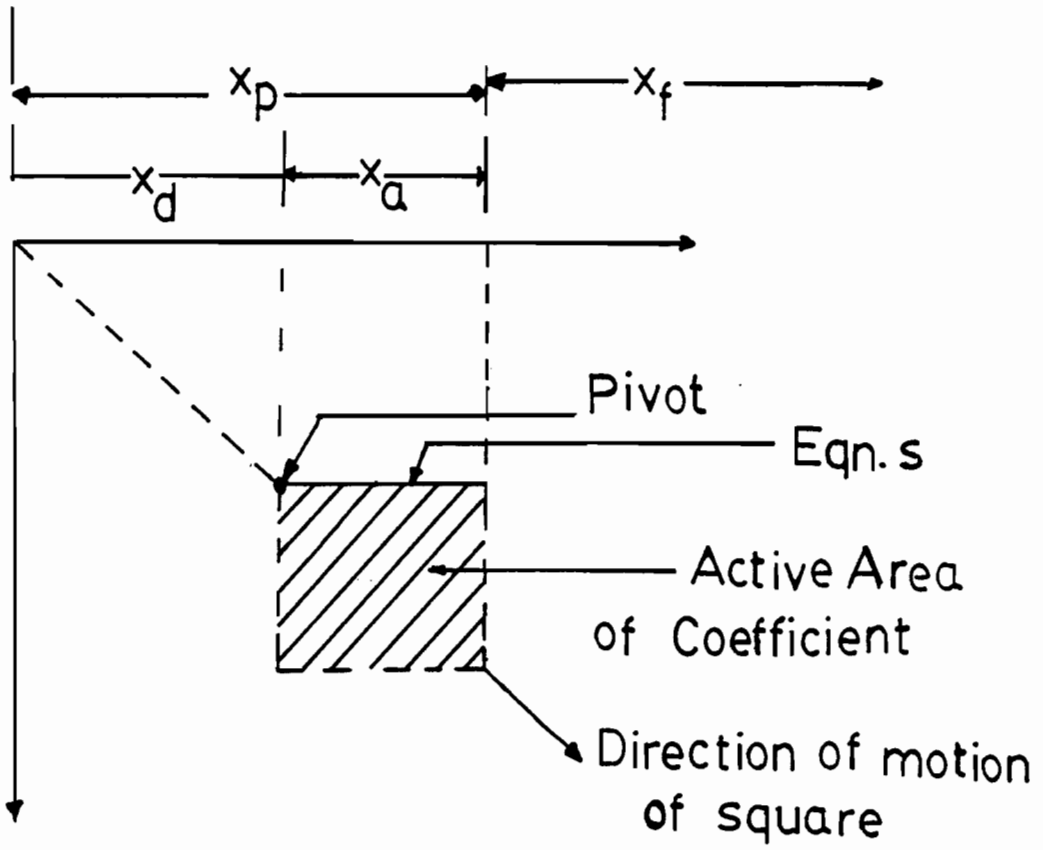


Figure 4.1 Frontal Procedure

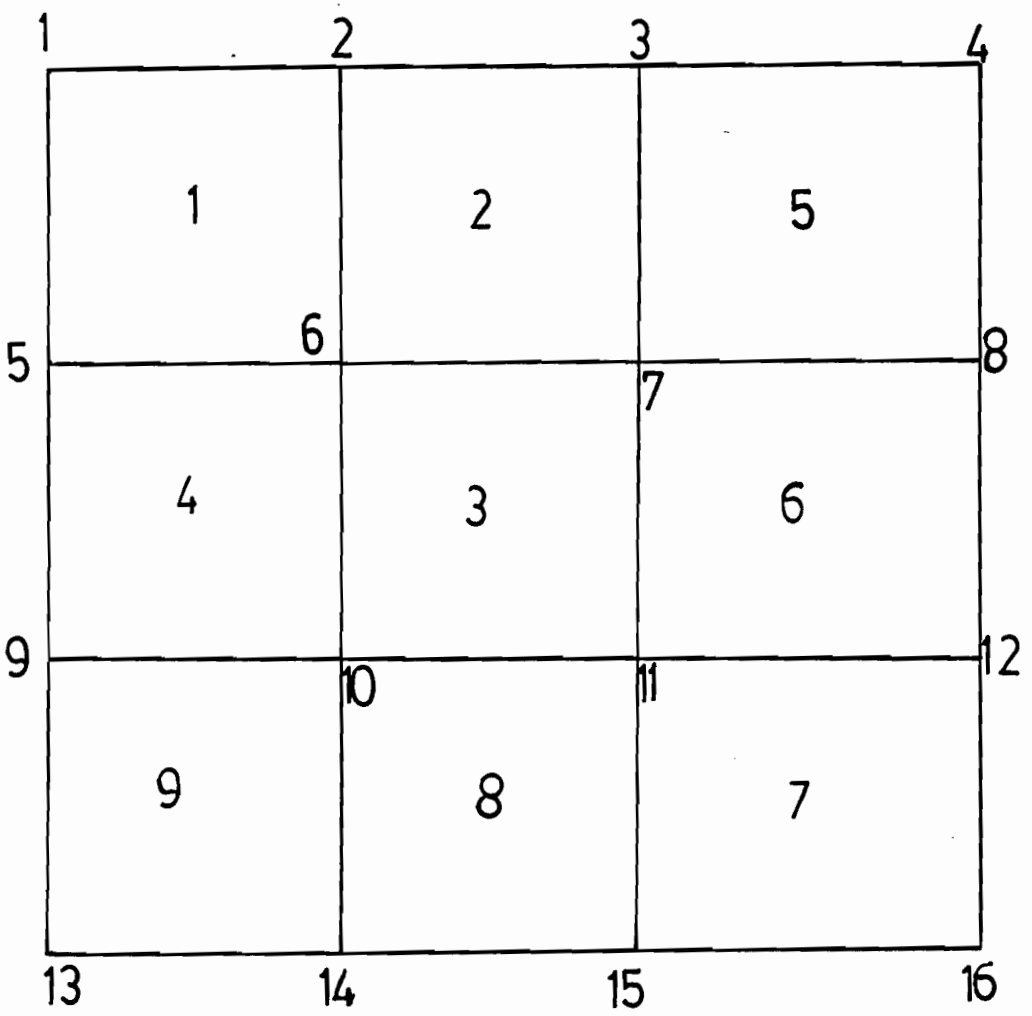


Figure 4.2 The Quadrilateral Mesh

Elements Assembled	Active Variables x_a	Eliminated Variables x_d	Variables Not Yet Processed x_f
1	1,2,5,6	1	3,4,7,8,9,10,11,12,13,14,15,16
2	2,3,5,6,7	1,2	4,8,9,10,11,12,13,14,15,16
3	3,5,6,7,19,11	1,2	4,8,9,12,13,14,15,16
4	3,7,10,11,9,6,5	1,2,5,6	4,8,12,13,14,15,16
5	3,4,7,8,9,10,11	1,2,3,4,5,6	12,13,14,15,16
6	7,8,9,10,11,12	1,2,3,4,5,6,7,8	13,14,15,16
7	9,10,11,12,15,16	1,2,3,4,5,6,7,8,12,16	13,14
8	9,10,11,14,15	1,2,3,4,5,6,7,8,11,12,15,16	13
9	9,10,13,14	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16	

Table 4.1 Assembly and Elimination

x_a: Those variables which are currently active and are still awaiting elimination (This area signifies the activity of the front), and

x_f: Those variables which are not yet assembled, i.e are to be processed in future.

There are two additional, frequently encountered terms in the context of the frontal method; these are the frontwidth and the destination. The frontwidth of a variable associated with a finite element mesh is defined as the total number of variables currently active. The last occurrence of any variable in an element is known as its destination, thus any variable is considered to be active only after being encountered for the first time and until it reaches its destination. Appendix I contains the basic definitions of the terms used in the context of the frontal method.

For the sake of illustration, the method has been applied to a mesh consisting of nine quadrilateral elements shown in Fig. 4.2 and the details are summarized in table 4.1.

4.2 The Need for Element Sorting

The ordering of the nodes, which is very important for a band solver, is immaterial to the frontal solver. In band solvers, all the matrix entries inside a band are stored and operated on; the nodes, therefore, are labelled in an attempt to achieve small bandwidths. The efficiency of the frontal method, on the other hand, is solely dependent on the element ordering, since this procedure assembles the matrix elements, element by element and removes variables node by node. The size of the largest problem which can be solved depends on the frontwidth (f) of its mesh, because the minimum storage requirements for the element assembly are $O(f^2)$. To be able to solve large problems, means are sought which can reduce the frontwidth. This becomes even more critical when frontal solutions are attempted on computers with restricted address spaces. Here, a solution is only possible if the available memory is utilized economically.

4.3 A Summary of Existing Methods

There are basically two distinct approaches for renumbering elements described in the current literature. The first method is known as the 'direct approach' and requires that the elements are ordered first and the nodes subsequently, if desired. The second approach is known as 'indirect', and here the nodes are renumbered first to minimize the bandwidth and then the elements are resequenced according to the new node numbers.

The indirect methods, in which nodes are renumbered first to minimize the bandwidth, and then the elements are resequenced, draw heavily on the existing node numbering techniques used with band solvers, where reduction in bandwidth and/or profile is of prime concern. One of the oldest and most popular methods belonging to this class is attributable to Cuthill and McKee [44], and has been modified subsequently by George [60], Collins [41], and King [86]. All these methods suffer from a weakness of being sensitive to the choice of a starting node, a problem which was later overcome by Gibbs, Poole and Stockmeyer [63]. Node renumbering techniques based on the 'graph theoretic' approach (e.g. the minimum degree algorithm and its variant, the nested dissection method) to reduce the profile and subsequent fill-in in the solution phase were pioneered by Liu and George [58], [61], [62].

The earliest attempt to derive a direct procedure for minimizing the frontwidth of a mesh appears to be that of King. This algorithm, which has the ability to minimize the frontwidth and is well suited to frontal elimination, was originally developed for reordering the nodes in problems arising in water distribution networks. King divided the total variables in the mesh into four distinct groups:

1. Those rows of the matrix which are already eliminated.

2. Those nodes for which the equations are fully constructed and ready for elimination.

3. Those nodes for which the equations are partly constructed.

4. Those nodes for which no equations are formed.

The main goal of his algorithm was to minimize the number of nodes in group 3 by scanning all the nodes in group 3 for that particular node which, when eliminated, would add the lowest number of new active nodes in group 3 from those in group 4. From the context of a frontal solution it would mean that the elements belonging to nodes in group 4 should be chosen such that the active front, i.e. stage 3, is a minimum. As mentioned earlier, though the method has a frontal like element sorting strategy, it was primarily developed for node reordering. The principal drawback of the method is that it is highly sensitive to the location of the starting node.

An alternate method for resequencing finite element meshes is Levy's [91], and was also originally developed to reorder nodes. His method has a strong resemblance to that of King, but is based on an expanded minimum front-growth criteria. When searching for the next node to be renumbered at each stage, all nodes which have yet to be renumbered are considered, whereas in King's method nodes are considered for possible renumbering only if they are currently in the front. As a result, Levy's method is slower than that of King and also shares the weakness of being unable to select the starting node.

Akin and Pardue [3] described two algorithms, one direct, the other indirect, in their paper. The procedure begins by identifying an element of minimum degree (i.e. having a minimum number of elements connected to it) and this is numbered first. The elements adjoining this element are then numbered in increasing order of their current degrees. Then, for each numbered element, the adjoining elements are searched using a criterion based on first renumbering the nodes by a Cuthill-McKee algorithm [44] followed by an element resequencing based on the concept of element degrees.

Liu's [93] method of element resequencing relies on minimizing the 'edge-front', which is defined as the set of edges connecting the renumbered and un-numbered nodes, based on the 'dual-graph'. In a dual-graph, each element is represented by a node, and two nodes are considered connected by an edge if two elements have one side in common. The direct strategies of Liu and Akin and Pardue require that the entire element and node lists be stored in core so that adjacency lists can be generated. Thus a heavy demand is placed on the available memory.

Bykat [21] orders the elements according to a procedure very similar to that of the Cuthill-McKee algorithm, resulting in a renumbering scheme which proceeds in a layer by layer fashion. Bykat chooses, as the next element for assembly, that which has the greatest number of neighbouring elements already assembled, starting with those having more nodes in the front. The first element is chosen as the one with the smallest number of neighbours.

An alternate approach to resequencing elements has been described by Pina [123]. His theory is based on a minimum front growth principle, but has a provision for searching ahead which is mainly useful for higher order elements. The method also has problems in selecting a starting node.

More recently, Fenves and Law [53] have come up with a strategy which they call 'a two step approach'. Their scheme involves ordering of the finite elements by first using the Cuthill-McKee algorithm and then renumbering the nodes, element by element, according to decreasing order of their valency, defined as the number of elements incident on that node. After the nodes are renumbered, the node ordering is reversed. The object behind this scheme is that the nodes with the lowest valency will be eliminated first. The algorithm needs the full element and node sets in core to produce the adjacency list and the starting node must be specified.

One of the more prominent indirect methods, which recently came to the fore, is that of Razzaque [127], which first renumbers the node for bandwidth minimization using either the Collins or Grooms [65] algorithms (which are essentially Cuthill-McKee like methods) then reorders the elements in ascending order of their lowest numbered nodes. More recently, another version has been expounded by Sloan and Randolph [150], where the nodes are renumbered using a modified King's algorithm, and the elements are then renumbered using the same strategy as Razzaque's.

Since indirect methods use node numbering first, either by Cuthill-McKee or similar techniques, they do not quite match with the frontal concept, as the reordering phase alone needs the entire node and element lists in core. Direct methods, however, do have the potential of producing element renumbering algorithms in a true frontal fashion but, surprisingly, were not investigated from a memory economy point of view until recently.

Auda [4], in 1981, developed an algorithm for constructing an element level structure of maximum depth within frontal storage limitations as the number of memory resident elements at any stage were proportional to the active variables in the frontal procedure. The algorithm needs $O(m^2N^2)$ comparisons to complete the first iteration, where m and N are respectively the number of nodes per element and total number of elements in the mesh. The storage requirements are of $O(\sqrt{N})$ for two dimensional, and $O(N^{\frac{2}{3}})$ for three dimensional meshes. The algorithm, hereinafter referred to as the SAS algorithm, was later modified [141] to reduce the number of disk accesses at the cost of more memory, since the time spent in input-output operations involving three sequential files tended to be long.

4.3.1 Comments on the SAS Algorithm

The SAS algorithm, which is related to both the minimum degree algorithm and the Cuthill-McKee method for renumbering the nodes, was developed to resequence the ele-

ments within the frontal frame-work so as to reduce the frontwidth. The algorithm was tested on several topologically different two dimensional models consisting of between 128 and 1024 first order triangular elements. The examination of the performance results leads to the following conclusions:

(a) The algorithm performs poorly in reducing the frontwidth. In almost 70 percent of the cases there is either an increase in the frontwidth from the original, or no improvement, or only a marginal improvement.

(b) The algorithm is very sensitive to the starting element.

(c) The algorithm reduces both the bandwidth and the profile remarkably.

(d) At least two or more iterations are required.

These conclusions provided the major motivation for developing a new algorithm which is capable of alleviating some or all of the shortcomings involved, whilst at the same time operating in the frontal framework of restricted space requirements. In the next sections, the new algorithm will be discussed, and its performance will be compared with that of SAS.

4.4 The New Method

The frontal solver has often been referred to as the 'natural' way of solving finite element equations. On the same basis, the new method can be described as a 'natural' approach to element renumbering. In this algorithm, the resequencing of elements is done using a theory similar to that of the Gaussian elimination algorithm, in which equations are eliminated one by one. Along the same lines, element reordering can proceed by sorting and arranging the elements as a group for each node as it occurs in a natural fashion. The nodes

are then eliminated one by one in the same natural order in the assembly and elimination phase. This procedure has two distinct advantages: one, there is no need to renumber separately the nodes, as the first node to be eliminated can be renumbered immediately as number one and so on in increasing order; second, the task of destination allotment for every node is complete when the last element sharing this node is sorted out. This is a better way of handling destination allotment than the approach used in [4], where all the elements in a level defined as a superelement have the same destination, which means that all the nodes in the superelement will be eliminated simultaneously; a procedure which is in direct contrast to the frontal spirit.

As the new algorithm automatically renumbers the nodes while sorting the elements, both the element sorting and node sorting are completed in one pass. The numerical results, reported in the sections to follow, confirm that this procedure leads to considerable reductions in both the frontwidths and the bandwidths. However, in the absence of any optimal element and node sorting algorithm, definite conclusions regarding absolute minimality cannot be established.

4.4.1 The Algorithm

The new element and node resequencing algorithm can be described as follows:

N = Number of elements.

M = Number of nodes.

m = Nodes per element.

A = Integer pointer array.

E = Input file containing element lists.

P = Output file containing element lists.

Allnodes = Vector storing the nodes as they occur.

FW = Frontwidth

S = Nodes which can be eliminated.

mp = Number of nodes. which have destinations lower than the current node.

nodelabel = Array into which the input is read.

fwm = Minimum frontwidth.

step = 1:

Initialize A=0; P=0; S=0; FW=0; Allnodes=0;

$P = \{firstelement\}$

step 2:

$S := \{Empty\}$; mp := 0;

BEGIN

FOR k := 1 to m **DO**

in := nodelabel (k)

$if(in \notin Allnodes)$ **THEN** $Allnodes := Allnodes + in$;

step 3:

FOR j := 1 to N **DO**

$if(j \neq A)$ **THEN**

BEGIN

FOR i := 1 to m **DO**

$if(in \in E_{(j,i)})$ $A := A + j$;

$P := P + j$;

END

step 4:

BEGIN

FOR I := j to N **DO**

FOR II := 1 to m **DO**

$if(in \notin E_{(l.ll)})$ **THEN** $S := S + 1$:

mp := mp + 1;

END

step 5:

$if\ S := \{Empty\}$ **THEN** Compute FW

ELSE

BEGIN

FOR i := 1 to mp **DO**

$fwm := min|FW|$

END

UNTIL $\{P = E\}$:

END.

4.4.2 Comments on the Algorithm

The algorithm presented in the last section is simple and self explanatory. However, in order to enhance the readability and understanding, a brief description seems necessary.

In step 1, all the variables are initialized and the first element is read. The first node of the node-list is chosen as the starting node. In step 2, all the elements of E are searched and those elements sharing the starting node are read into main memory. All the nodes of the node-list are transferred to the vector Allnodes and the first element is written to the output file P. In step 3 pointer vector A is updated and the elements sharing the common node are written onto P. In step 4, E is searched to find out if other nodes which belong to elements in the core occur for the last time. The search for a node is discontinued upon the first encounter of an element containing the node being searched for. The search effort is saved in a pointer vector A, which stores this element number. The same procedure is repeated for all the remaining nodes of the element. If there is more than one node which is occurring for the last time, i.e. has a destination lower than the first node, then these nodes are sorted out according to their frontwidths in step 5. The nodes are arranged according to ascending frontwidths, i.e. the node contributing a minimum to the front is numbered first.

In an element for which there are no nodes occurring for the last time, the node being checked is renumbered in the usual way; its frontwidth is determined and the last element sharing this node is tagged. Also, in order to keep the growth of the front low, the elements in core are resequenced according to increasing order of their highest numbered node.

4.4.3 Memory and Time Requirements

One of the best features of this algorithm is that at any one time only the element list corresponding to one node is in core. Unlike the SAS algorithm, in which a level structure

of maximal depth is constructed, with a memory size of $O(\sqrt{N})$ for two dimensional and $O(N^{\frac{2}{3}})$ for three dimensional meshes, the memory requirements of the present algorithm are $O(k)$ for constructing an element level-structure for one node where $k << N$, and depends on the mesh topology and the order of the elements.

A bound on k can be arrived at easily by considering the fact that an algorithm based on constructing $O(N^{\frac{1}{3}})$ numbers of level structures each comprising of $O(N^{\frac{2}{3}})$ requires $O(N^{\frac{2}{3}})$ main memory; a similar algorithm, therefore, based on constructing N level structures will need $O(1)$ storage locations. Supposing M and N are to be of the same order for first order elements, then the number k is $O(1)$. Thus savings in space are resulting not only due to working with $O(k)$ spaces, but also due to the fact that no adjacency or linked lists are generated.

Two working vectors, each of length M, are needed to accomplish the task of node numbering at the same time as the element numbering is in progress. The disk storage is required for E and P is mN integer words each.

The time requirements of the algorithm are $O(mNM)$ as these are the total number of operations executed in the sorting. In addition, time is required for accessing the input and output files. Therefore, the execution time can be longer due to a large number of disk accesses. A great reduction in I/O time is possible, however if large part of the database is brought into the core depending on the availability of the core while sorting.

The algorithm is primarily written for small computer systems where the main storage is always a major restriction. This restriction can only be overcome at the cost of execution time, as the exploitation of one generally contributes to the deterioration of the other. Therefore, a possibility of slower execution for large models is inherent in the algorithm.

4.4.4 Data Structures

The input file to the element sorting routines is a direct access file. Each record in the file contains two alphabetic characters and the node labels belonging to this element. The first character is reserved for specifying the material property and the other for identifying the element type. There are as many records as there are elements in the model.

The output file produced by the new algorithm has essentially the same structure, with the exception that each node is now either followed by a blank or a '$\star$'. The '$\star$' indicates that this node occurs for the last time in this element. In the matrix assembly routine, the '$\star$' will indicate that this row can be decomposed, as soon as the current element is assembled. Typical input and output records are shown below:

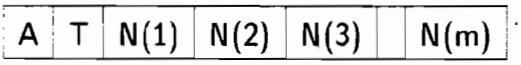


Figure 4.3 Data Structure-Input file

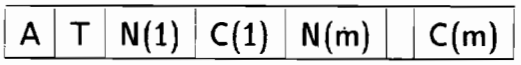


Figure 4.4 Data Structure- Output File

4.5 Numerical Experiments

In order to evaluate the performance of the new algorithm, a series of numerical experiments were performed on meshes which were different in both topology and size. The six topologically different models used in the evaluation process were a square, a square with a hole in the middle, an U-shape, a L-shape, a '+'-shape and an H-shape. These shapes,

shown in Figs. 4.5-4.10, were chosen because they were easy to construct using the mesh generator described in Chapter III.

All six models were then used to compare the SAS algorithm and the new algorithm with regard to the frontwidths determined by each of them. In tables 4.2 and 4.3, the original statistics, i.e. before element sorting, are given. Both algorithms were tried with different starting nodes and the resulting frontwidths were recorded. The results are summarized in tables 4.4 and 4.9. After establishing that the new algorithm performs better than the SAS algorithm in returning the minimum frontwidth, several larger models were constructed. The topological structures of the models were kept the same as the smaller ones. The parameters monitored were, (a) the bandwidth, (b) the frontwidth, (c) the root mean square frontwidth, (d) the profile and (e) the number of comparisons required to resequence the elements. Both first order and second order elements were used. All the results on each of the models are presented in tables 4.10-4.21.

For the sake of convenience, the numerical experiments were performed on a VAX-11/780 computer and had to be run as batch jobs under lowest priority. As a result, accurate monitoring of the execution time became difficult.

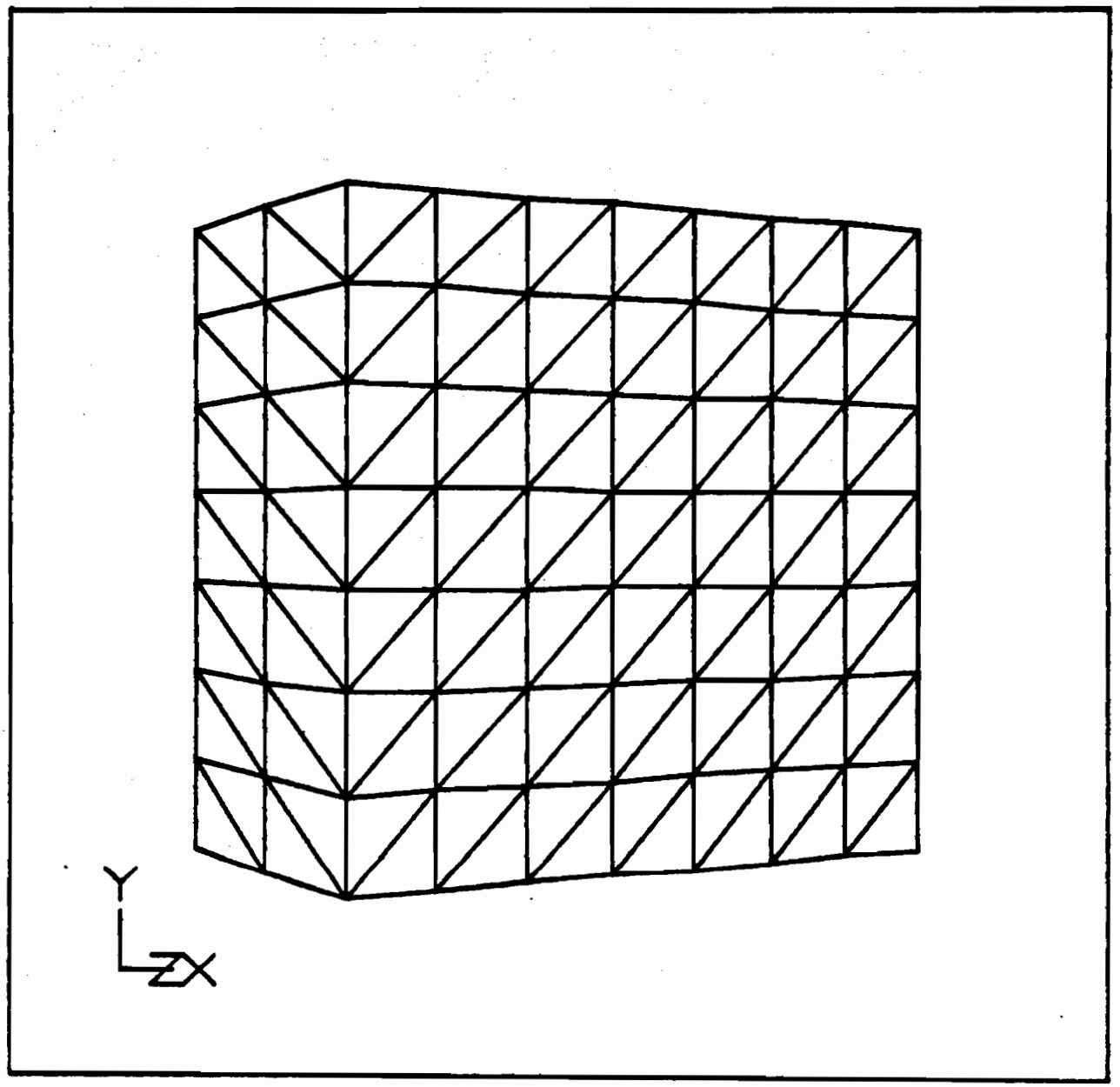


Figure 4.5 Model Square

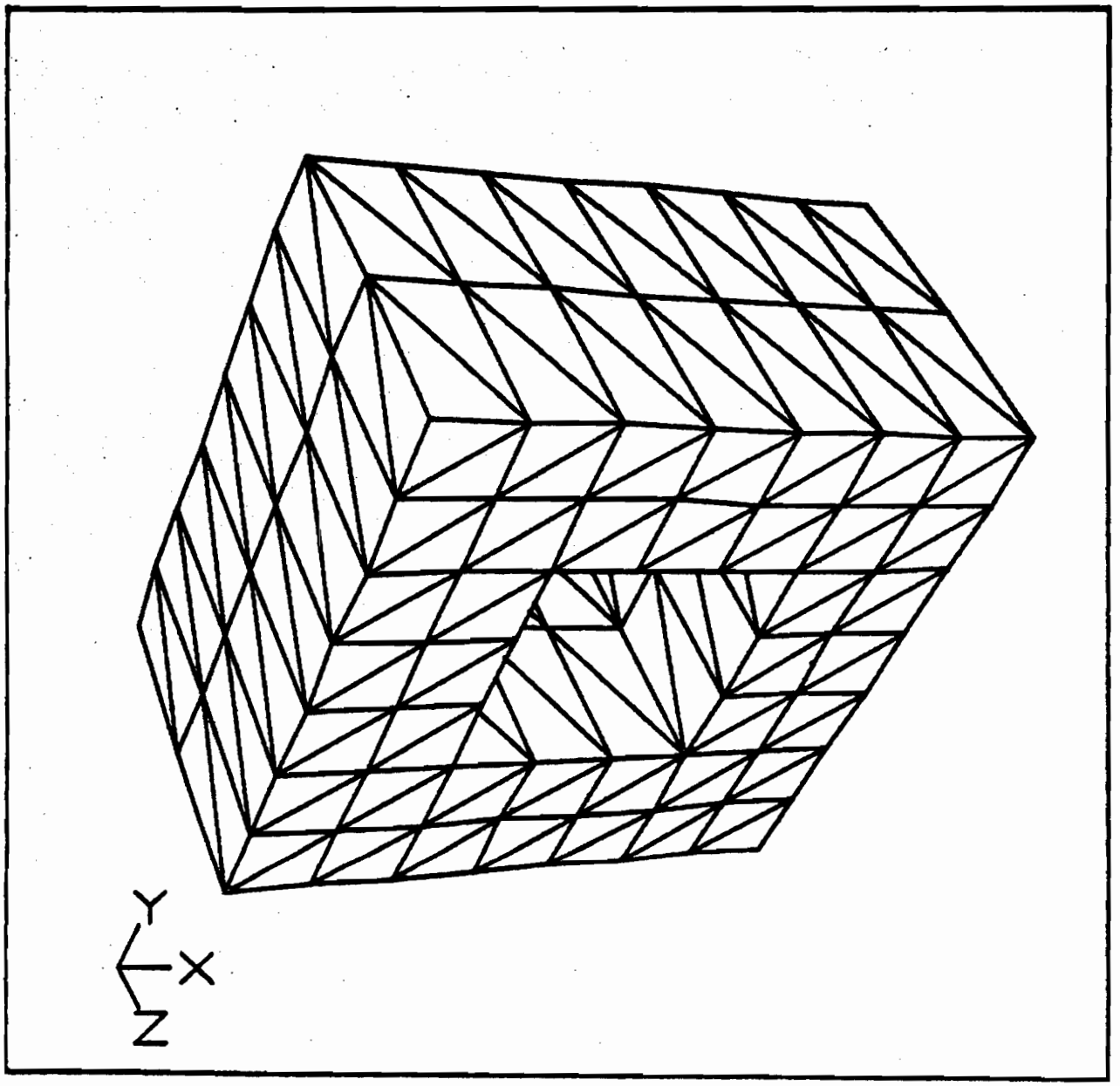


Figure 4.6 Model Square with Hole

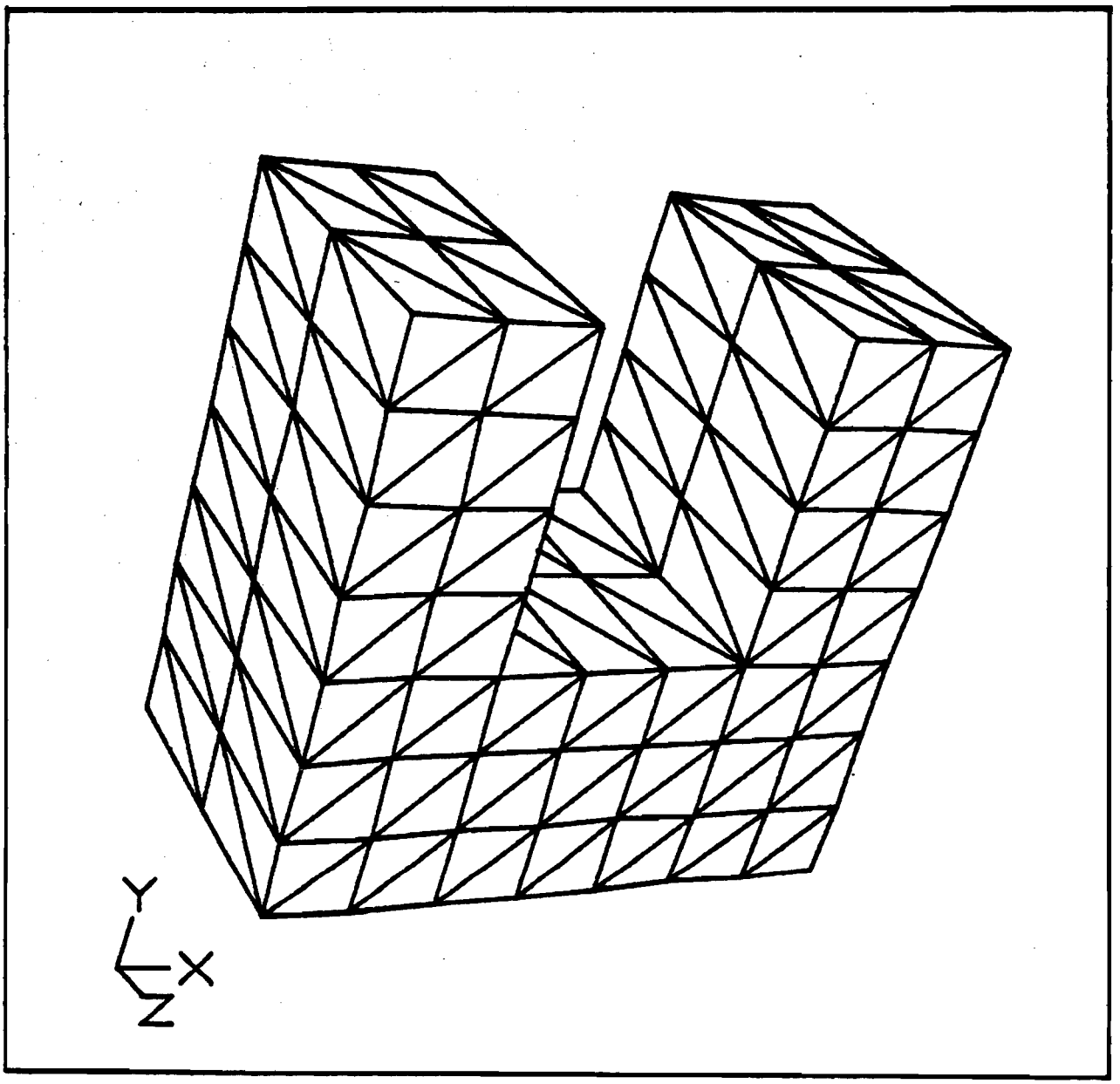


Figure 4.7 Model U-Shape

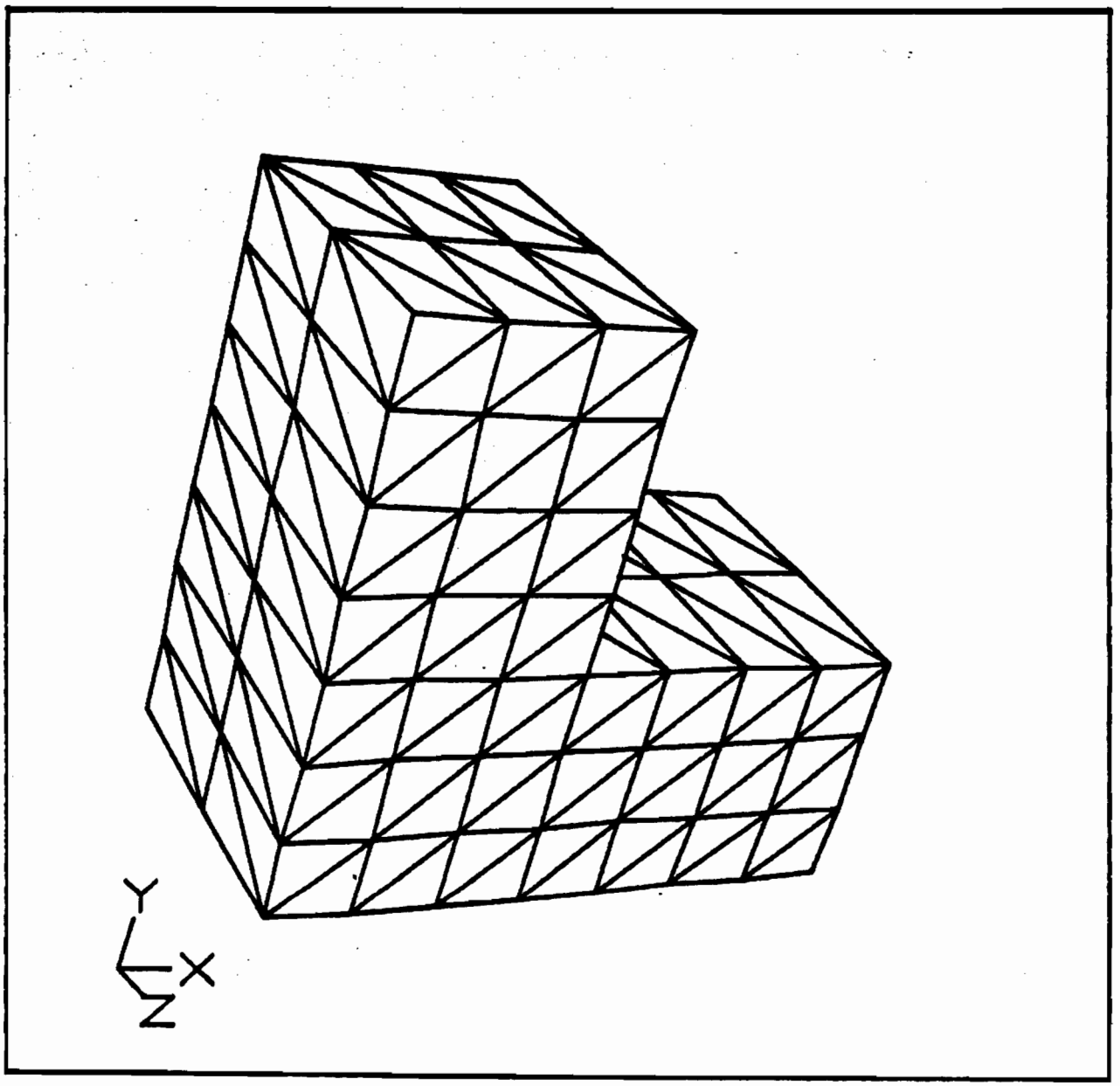


Figure 4.8 Model L-Shape

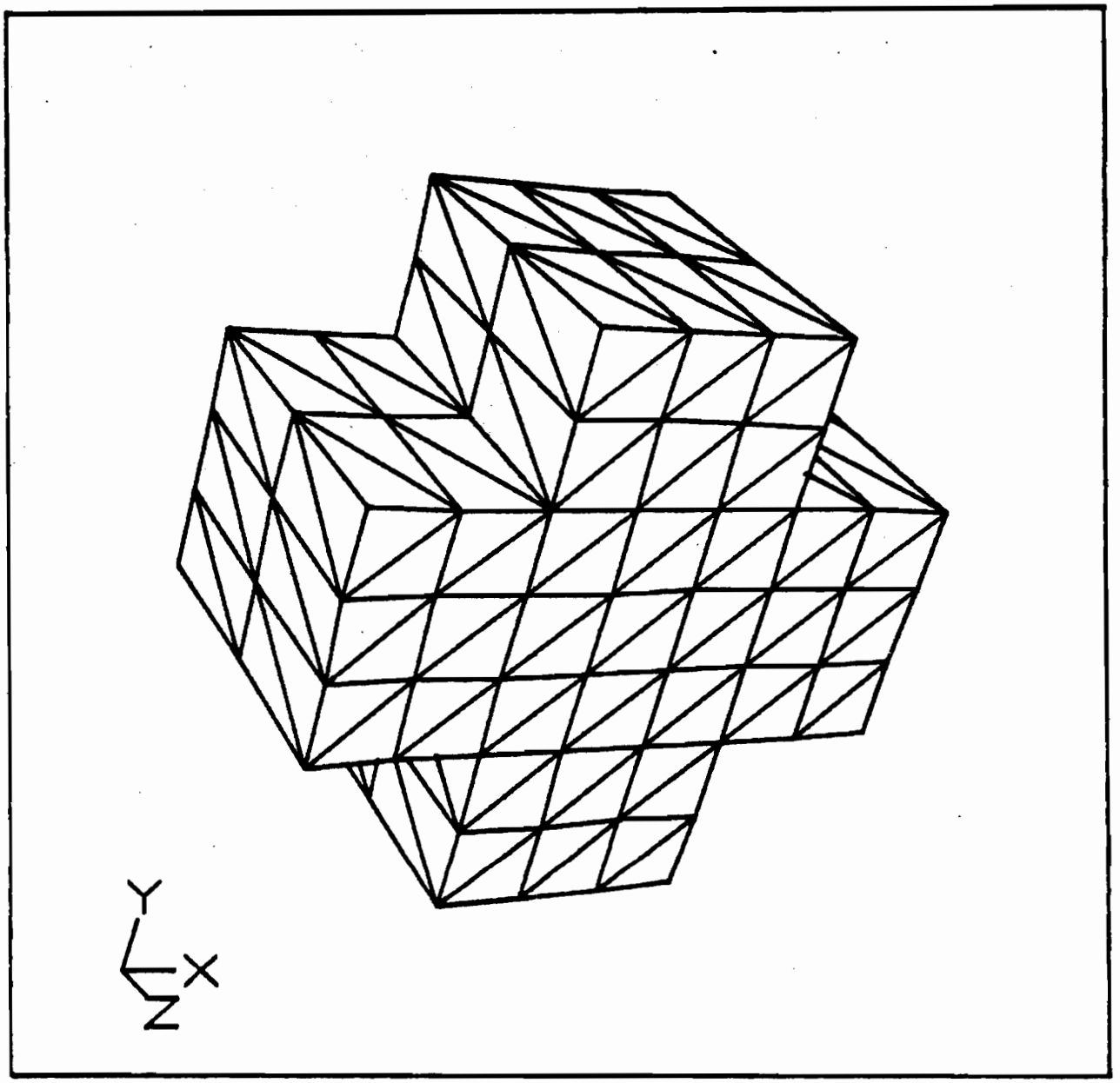


Figure 4.9 Model '+'-Shape

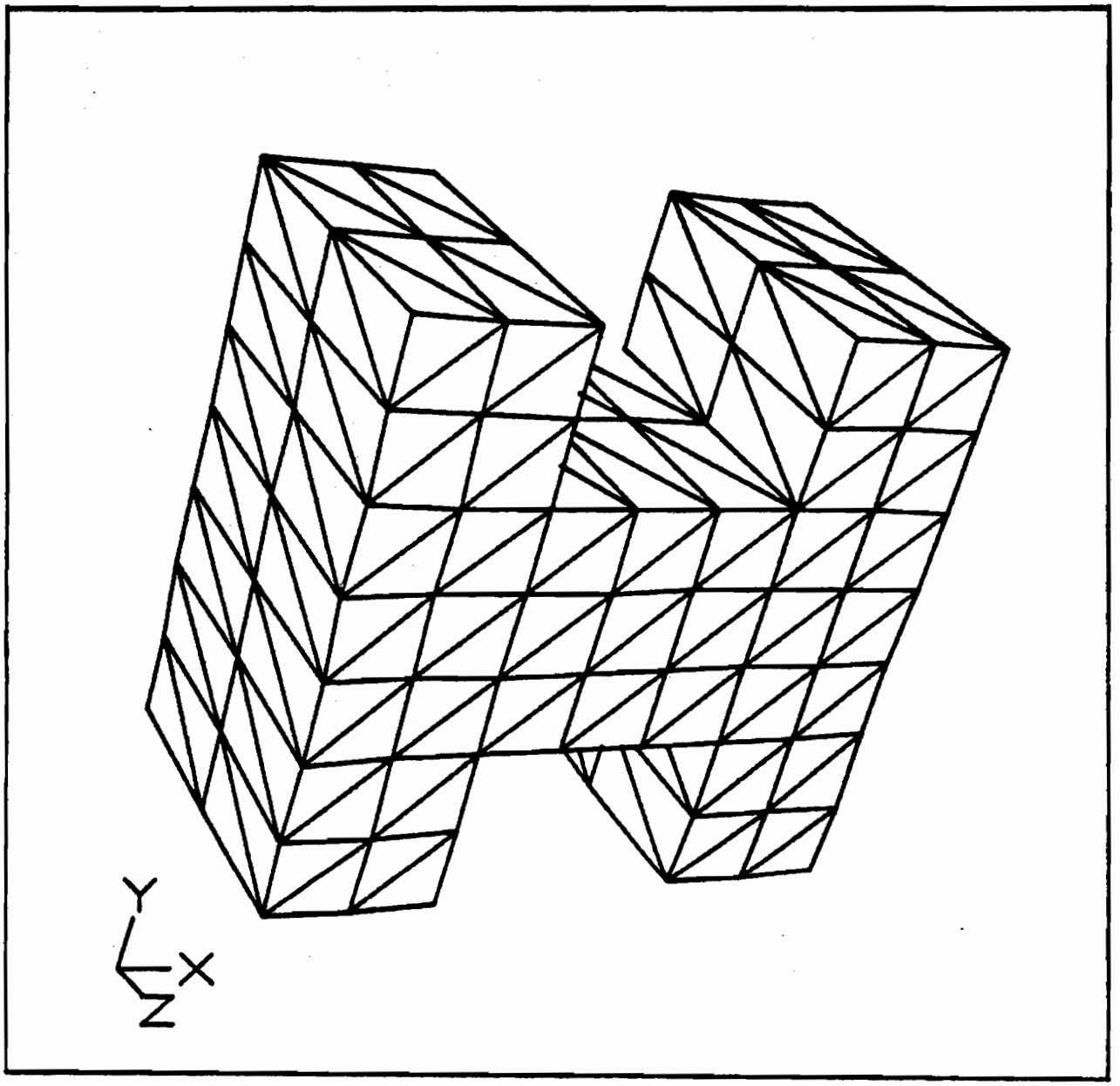


Figure 4.10 Model H-Shape

Model	Elements	Nodes	Nodes Per Element	Frontwidth
Square	294	128	4	17
	588	192	4	65
Square With Hole	240	120	4	17
	480	180	4	61
U-Shape	204	108	4	27
	408	162	4	55
L-Shape	144	78	4	24
	288	117	4	40
+-Shape	198	96	4	17
	396	144	4	49
H-Shape	222	112	4	20
	444	168	4	57

Table 4.2 Test Problem Statistics: First Order Elements

Model	Elements	Nodes	Nodes Per Element	Frontwidth
Square	294	675	10	52
	588	1125	10	232
Square With Hole	240	600	10	52
	480	1000	10	207
U-Shape	204	525	10	52
	408	875	10	182
L-Shape	144	375	10	51
	288	625	10	132
+-Shape	198	483	10	52
	396	805	10	168
H-Shape	222	555	10	17
	444	925	10	192

Table 4.3 Test Problem Statistics: Second Order Elements

Elements	Nodes	SAS Algorithm			New Algorithm	
		Starting Element	Iterations	Frontwidth	Iterations	Frontwidth
294	128	1	1	15	1	15
		1	2	15	1	15
		100	1	28	1	15
		100	2	15	1	15
		150	1	40	1	15
		150	2	15	1	15
		200	1	24	1	15
		200	2	16	1	15
588	192	1	1	47	1	22
		1	2	36	1	22
		98	1	42	1	22
		98	2	36	1	22
		301	1	40	1	22
		301	2	40	1	22
		588	1	45	1	22
		588	2	36	1	22

Table 4.4 Model Square

Elements	Nodes	SAS Algorithm			New Algorithm	
		Starting Elements	Iterations	Frontwidth	Iterations	Frontwidth
240	120	1	1	24	1	12
		1	2	28	1	12
		81	1	24	1	12
		81	2	28	1	12
		161	1	27	1	12
		161	2	27	1	12
		235	1	21	1	12
		235	2	17	1	12
480	180	1	1	47	1	18
		1	2	35	1	18
		200	1	44	1	18
		200	2	35	1	18
		405	1	39	1	18
		405	2	39	1	18
		479	1	41	1	18
		479	2	40	1	18

Table 4.5 Model Square with Hole

Elements	Nodes	SAS Algorithm			New Algorithm	
		Starting Element	Iterations	Frontwidth	Iterations	Frontwidth
204	108	1	1	27	1	12
		1	2	18	1	12
		101	1	27	1	12
		101	2	20	1	12
		153	1	24	1	12
		153	2	20	1	12
		203	1	20	1	12
		203	2	21	1	12
408	162	1	1	34	1	18
		1	2	24	1	18
		205	1	33	1	18
		205	2	24	1	18
		304	1	33	1	18
		304	2	24	1	18
		407	1	33	1	18
		407	2	24	1	18

Table 4.6 Model U-Shape

Elements	Nodes	SAS Algorithm			New Algorithm	
		Starting Elements	Iterations	Frontwidth	Iterations	Frontwidth
144	78	1	1	15	1	6
		1	2	9	1	6
		76	1	15	1	6
		76	2	7	1	6
		105	1	19	1	6
		105	2	7	1	6
		141	1	16	1	6
		141	2	8	1	6
288	117	1	1	21	1	9
		1	2	13	1	9
		77	1	22	1	9
		77	2	13	1	9
		226	1	21	1	9
		226	2	13	1	9
		283	1	20	1	9
		283	2	13	1	9

Table 4.7 Model L-Shape

Elements	Nodes	SAS Algorithm			New Algorithm	
		Starting Element	Iterations	Frontwidth	Iterations	Frontwidth
198	96	52	1	16	1	15
		52	2	17	1	15
		101	1	24	1	15
		101	2	17	1	15
		151	1	17	1	15
		151	2	29	1	15
		190	1	28	1	15
		190	2	17	1	15
396	144	1	1	33	1	20
		1	2	24	1	20
		75	1	32	1	20
		75	2	24	1	20
		155	1	32	1	20
		155	2	24	1	20
		370	1	33	1	20
		370	2	24	1	20

Table 4.8 Model '+'-Shape

Elements	Nodes	SAS Algorithm			New Algorithm	
		Starting Elements	Iterations	Frontwidth	Iterations	Frontwidth
222	112	1	1	21	1	17
		1	2	18	1	17
		60	1	26	1	17
		60	2	18	1	17
		150	1	26	1	17
		150	2	20	1	17
		210	1	18	1	17
		210	2	18	1	17
444	168	1	1	46	1	26
		1	2	31	1	26
		109	1	39	1	26
		109	2	30	1	26
		250	1	48	1	26
		250	2	31	1	26
		400	1	31	1	26
		400	2	32	1	26

Table 4.9 Model H-Shape

Elements	Nodes	Nodes Per Element	Original				After Sorting and Renumbering			
			Band Width	Profile	Front Width	RMS Front Width	Band Width	Profile	Front Width	RMS Front Width
294	128	4	29	1657	17	14.899	21	1116	15	12.86
294	675	10	138	52581	52	44.72	33	4170	34	21.26
588	192	4	136	8326	65	45.12	29	1465	22	16.37
588	1125	10	748	329671	232	159.71	188	97555	47	23.77
726	288	4	131	10199	47	34.71	51	5360	17	11.59
726	1590	10	697	333400	131	103.0	201	113250	59	26.31
1542	432	4	350	43814	159	102.77	107	10801	26	15.73
1542	2651	10	2040	1867540	578	375.86	265	130969	131	49.18
2904	720	4	350	87748	165	128.63	83	3774	26	16.42
2904	4773	10	2040	4289700	614	484.43	281	140352	102	61.34

Table 4.10 Model Square

Elements	Nodes	Nodes Per Elements	Original				After Sorting and Renumbering			
			Band Width	Profile	Front Width	RMS Front Width	Band Width	Profile	Front Width	RMS Front Width
240	120	4	29	1428	17	12.87	22	1216	12	8.35
240	600	10	138	39965	52	36.85	102	237250	56	33.19
480	180	4	128	7192	61	41.52	61	3163	18	11.29
480	1000	10	673	256497	207	139.5	346	107676	92	64.452
630	270	4	99	8275	44	29.4	102	8544	44	29.71
630	1443	10	505	256088	131	87.75	89	26209	63	43.64
1260	405	4	316	37487	142	94.18	171	18868	71	52.7
1260	2406	10	1765	1506960	503	334.96	369	72431	131	67.89
2520	675	4	445	74394	148	110.73	189	7577	47	36.54
2520	4332	10	1765	3485480	565	433.71	353	43658	132	72.11

Table 4.11 Model Square with Hole

Elements	Nodes	Nodes Per Elements	Original				After Sorting and Renumbering			
			Band Width	Profile	Front Width	RMS Front Width	Band Width	Profile	Front Width	RMS Front Width
204	108	4	21	1215	27	12.12	21	1115	12	7.9
204	525	10	122	31917	56	33.18	104	30458	23	17.34
408	162	4	112	5871	55	37.67	42	27.71	18	13.00
408	875	10	568	198585	182	123.45	300	916732	92	60.95
558	252	4	99	6890	38	26.09	54	3990	17	11.34
558	1314	10	505	202015	110	75.73	147	19951	34	21.77
1116	378	4	298	32056	133	87.23	166	15177	64	45.27
1116	2190	10	1636	1228280	461	303.28	407	55078	83	53.15
2232	630	4	298	65122	144	110.14	83	6303	51	33.21
2232	3942	10	1636	2853420	509	393.553	332	270832	119	57.64

Table 4.12 Model U-Shape

Elements	Nodes	Nodes Per Elements	Original				After Sorting and Renumbering			
			Band Width	Profile	Front Width	RMS Front Width	Band Width	Profile	Front Width	RMS Front Width
144	78	4	26	615	24	9.51	11	417	6	3.10
144	375	10	129	16754	51	27.82	29	1118	15	10.44
288	117	4	76	3009	40	27.29	31	804	9	4.36
288	625	10	389	100816	132	89.39	57	3270	32	26.47
342	160	4	99	2713	29	18.46	23	270.1	9	5.73
342	819	10	505	78210	84	52.5	114	3611	37	24.73
684	240	4	206	13051	87	56.40	161	6086	51	31.10
684	1365	10	1141	482677	296	192.53	276	21709	160	96.02
1368	400	4	206	26565	93	70.75	121	12.28	26	20.16
1368	2457	10	1141	1121970	325	248.227	473	107837	67	36.47

Table 4.13 Model L-Shape

Elements	Nodes	Nodes Per Element	Original				After Sorting and Renumbering			
			Band Width	Profile	Front Width	RMS Front Width	Band Width	Profile	Front Width	RMS Front Width
198	96	4	21	1037	17	12.06	22	1130	15	10.43
198	483	10	117	30520	52	36.06	43	12200	26	16.81
396	144	4	96	4768	49	34.12	35	2620	20	14.60
396	805	10	508	172883	168	116.33	322	92519	79	45.9
510	216	4	113	5993	37	27.42	86	4275	35	24.54
510	1158	10	589	188234	111	81.44	141	16369	64	34.77
1020	324	4	269	25022	123	77.91	178	11893	68	47.43
1020	1931	10	1536	1009040	434	277.52	456	46358.3	218	151.66
2040	540	4	269	49850	125	97.14	133	4747	52	31.34
2040	3477	10	1536	2303810	455	355.90	461	217161	167	84.36

Table 4.14 Model '+'-Shape

Elements	Nodes	Nodes Per Elements	Original				After Sorting and Renumbering			
			Band Width	Profile	Front Width	RMS Front Width	Band Width	Profile	Front Width	RMS Front Width
222	112	4	21	1345	20	13.10	27	1368	17	12.84
222	555	10	114	37900	52	38.23	61	19546	32	24.18
444	168	4	116	6394	57	39.38	42	3178	32	22.38
444	925	10	598	224941	192	131.81	311	111235	102	70.82
582	252	4	107	7755	38	29.36	24	742.8	9	5.14
582	1337	10	553	235654	112	86.33	123	22151	16	10.75
1164	378	4	302	33297	141	89.41	32	1629	39	21.13
1164	2230	10	1691	1318390	493	314.97	117	60982	127	83.26
2328	630	4	302	66737	144	112.083	107	64285	41	23.37
2328	4016	10	1691	3029130	517	40606700	241	78598	96	63.76

Table 4.15 Model H-Shape

Elements N	Nodes M	Nodes Per Element m	Comparisons O	Ratio $\frac{O}{mNM}$
294	128	4	0.148764E06	0.988
294	675	10	0.197127E07	0.993
588	192	4	0.448056E06	0.992
588	1125	10	0.658854E07	0.996
726	288	4	0.831998E06	0.994
726	1590	10	0.115107E08	0.997
1542	432	4	0.250935E07	0.938
1542	2651	10	0.335544E08	0.820
2904	720	4	0.834610E07	0.9979
2904	4773	10	0.335544E08	0.2420

Table 4.16 Model Square

Elements N	Nodes M	Nodes Per Element m	Comparisons O	Ratio $\frac{O}{mNM}$
240	120	4	0.113760E06	0.987
240	600	10	0.142920E07	0.992
480	180	4	0.342720E06	0.991
480	1000	10	0.477840E07	0.995
630	270	4	0.676622E06	0.994
630	1443	10	0.906256E07	0.996
1260	405	4	0.203364E07	0.996
1260	2406	10	0.335544E08	1.10
2520	675	4	0.678889E07	0.997
2520	4332	10	0.335544E08	0.307

Table 4.17 Model Square with Hole

Elements N	Nodes M	Nodes Per Element m	Comparisons O	Ratio $\frac{O}{nNM}$
204	108	4	0.869040E05	0.986
204	525	10	0.106182E07	0.991
408	162	4	0.261936E06	0.990
408	875	10	0.355164E07	0.994
558	252	4	0.559116E06	0.994
558	1314	10	0.730701E07	0.996
1116	378	4	0.168070E07	0.996
1116	2190	10	0.320031E08	1.309
2232	630	4	0.561125E07	0.997
2232	3942	10	0.335544E08	0.3813

Table 4.18 Model U-Shape

Elements N	Nodes M	Nodes Per Element m	Comparisons O	Ratio $\frac{O}{mNM}$
144	78	4	0.440640E05	0.980
144	375	10	0.533520E06	0.988
288	117	4	0.133056E06	0.987
288	625	10	0.178704E07	0.992
342	160	4	0.216828E06	0.984
342	819	10	0.278559E07	0.994
684	240	4	0.652536E06	0.993
684	1365	10	0.930582E07	0.993
1368	400	4	0.218059E07	0.996
1368	2475	10	0.335544E08	0.998

Table 4.19 Model L-Shape

Elements N	Nodes M	Nodes Per Element m	Comparisons O	Ratio $\frac{O}{mNM}$
198	96	4	0.740844E05	0.984
198	483	10	0.947430E06	0.990
396	144	4	0.225720E06	0.989
396	805	10	0.316998E07	0.994
510	216	4	0.437582E06	0.993
510	1158	10	0.588286E07	0.996
1020	324	4	0.131580E07	0.995
1020	1931	10	0.225234E08	1.143
2040	540	4	0.439417E07	0.997
2040	3477	10	0.335544E08	0.473

Table 4.20 Model '+'-Shape

Elements N.	Nodes M	Nodes Per Element m	Comparisons O	Ratio $\frac{O}{mNM}$
222	112	4	0.981240E05	0.984
222	555	10	0.122211E07	0.991
444	168	4	0.295704E06	0.991
444	925	10	0.408702E07	0.995
582	252	4	0.583166E06	0.994
582	1337	10	0.775516E07	0.996
1164	378	4	0.175299E07	0.996
1164	2230	10	0.335544E08	1.29
2328	630	4	0.585260E07	0.997
2328	4016	10	0.335544E08	0.358

Table 4.21 Model H-Shape

4.6 On the Algorithm Performance

The analysis of results presented in tables 4.10 to 4.21 lead to the following conclusions:

(1) The new algorithm consistently reduced the bandwidth, frontwidth, profile, and the root-mean-square frontwidth for almost all the cases considered, and the frontwidths obtained after resequencing the elements were always lower than the bandwidths obtained after renumbering the nodes.

(2) The new algorithm is insensitive to the choice of starting element and always returns the same maximum frontwidth. This is achieved because of the fact that, for a given model, the number of elements shared by each node is fixed and, therefore, the maximum frontwidth also remains the same. However, it may be possible that the rms frontwidths may be different for a different choice of starting node. Since the storage allocation for a frontal solver is always dependent on the maximum frontwidth, changes in rms frontwidth are not of any concern.

(3) The new algorithm requires only one iteration to the elements with near minimum frontwidths. The subsequent iterations tried on all the cases did not further reduce the maximum frontwidth.

(4) The new algorithm behaved strictly as per the operation count estimates, except for a few models comprising of second order elements for which it is lower. This is possibly due to the fact that for bigger models a large number of midside nodes are sorted out with their vertices and need not be searched individually. The number of comparisons was always $O(mMN)$ for all the cases analyzed (See tables 4.16-4.21). The factor mMN seems to be a better way of assessing the performance of element or node resequencing algorithms than using N^2, because the former can

take care of the exact relationship between the node and the element numbers for higher order elements.

(5) No extra work is required to renumber the nodes after element sorting is finished, since the new algorithm accomplishes both the tasks together in one pass. This feature adds to the efficiency of the algorithm by reducing the additional number of comparisons required if the node numbering is to be done separately.

(6) The algorithm needs $O(k)$ memory locations for element resequencing where ($k << N$). For example, a mesh consisting of 10000 nodes and as many elements as possible would need 11K floating point words of memory to resequence the elements and the nodes. However, if only element sorting is to be done, then the space requirements are halved.

Chapter 5

Preconditioned Conjugate Gradient Frontal Method

5.1 Introduction

The finite element solution of a partial differential equation leads to an algebraic system of linear equations whose coefficient matrix is sparse, symmetric, and positive definite. There are two general classes of methods for solving such systems of linear algebraic equations: direct and iterative. Recently, there have been significant developments in both classes to make the methods computationally more efficient by manipulating the sparsity structures of the resulting matrices. Also, advancements in both classes have been brought about by exploiting the properties of the matrices, i.e. property M, diagonal dominance, etc.

In the class of direct methods, the usage of fast Fourier transform techniques, marching methods, and nested dissection techniques have provided most of the improvements and these methods have become more competitive for two dimensional problems, including single or multiple right hand sides. Generally, for three dimensional problems involving more than a thousand variables, iterative methods have always provided better performance and, with major advances in the strongly implicit procedure and the use of preconditioning tech-

niques with the conjugate gradient algorithm, the power of iterative methods has increased significantly.

For the sake of illustration, consider a problem for which the coefficient matrix, A,is full. The solution of the matrix equation, based on Gaussian elimination would need $O(N^3)$ operations with $O(N^2)$ storage. Also, if band solvers are employed, the operation count can be further reduced to $O(\beta^2 N)$ for a matrix with semi-bandwidth β, which is still higher than $O(N^{1.5})$ for the preconditioned conjugate gradient method. On the other hand, one iteration of an iterative method which calculates the residual requires $O(N^2)$ operations, and if the method converges in less than N iterations, the number of operations required is considerably less than for the direct methods. Additionally, the storage is always less, as no decomposition is required . Iterative methods usually exploit the sparsity of the algebraic equations from two angles; the first is the reduction in the amount of storage needed to solve the system, and the second is the reduction in the number of operations required to obtain a solution of specified accuracy. In the sections to follow, it will be explained how these two objectives are achieved by using the conjugate gradient method, and how a memory-economic and fast converging preconditioned conjugate gradient frontal algorithm can be developed.

5.2 Conjugate Gradients

The conjugate gradient (CG) method was developed by Hestenes and Stiefel in 1952 [73] for solving sets of linear algebraic equations. Essentially, the method is N step iterative, i.e. the solution is achieved in no more than N steps, if there are no roundoff errors. The following advantages are principally associated with the conjugate gradient algorithm:

(1) The simplicity of the computational procedure.

(2) The preservation of the original matrix of coefficients and its sparsity during the computations.

(3) Small storage.

(4) The ability to start anew at any point in the computations.

(5) A progressively refined solution at every step.

Using exact arithmetic, CG calculates the solution to a system of N equations in $O(N^3)$ operations, each made up of N steps of $O(N^2)$ computations.

The convergence of the conjugate gradient algorithm depends on the number of distinct eigenvalues and their spread, i.e how closely the eigenvalues are clustered together. This phenomenon was first observed by Reid [128], and later Jennings [82], and leads to the conclusion that in the absence of roundoff the number of CG steps for the convergence is equal to the number of clusters.

However, for the algebraic equations derived from finite element or finite difference models of partial differential equations, the clustering property does not hold, as the eigenvalues are generally well distributed and, in some circumstances, are uniformly distributed. Thus, the conjugate gradient method was not considered to be suitable for such systems.

In 1977, Meijerink and Van der Vorst [102] proposed the use of a system of equations, preconditioned by an approximate inverse of the coefficient matrix, with the conjugate gradient algorithm. The preconditioning operation resulted in a clustering of the eigenvalues, and thus conjugate gradients could be applied to finite element or finite difference equations. This procedure turned out to be very efficient and gave rapid convergence to the conjugate gradient algorithm, which has resulted in increased popularity for preconditioned conjugate gradients.

5.3 Preconditioned Conjugate Gradients

The basic conjugate gradient method yields the solution of N linear equations

$$A * x = b \tag{5.1}$$

in less than or equal to N iterations, in the absence of roundoff errors. It could converge in far less than N iterations, if the eigenvalues of the matrix A are clustered together with many close to unity. Theoretically, the number of iterations is equal to the number of distinct eigenvalues. The central idea of the preconditioned conjugate gradient algorithm is to transform the matrix equation (5.1) to

$$(L^{-1}AL^{-T})(L^{T}x) = (L^{-1})b \tag{5.2}$$

or,

$$Cy = z$$

where

$$L^{-1}AL^{-T} = C$$

$$L^{T}x = y$$

$$L^{-1}b = z$$

and L is a positive definite preconditioning matrix and is usually computed by performing either LU, LDL^{T}, an incomplete Cholesky or any other factorization of the matrix A. Meijerink and Van der Vorst proposed two kinds of incomplete Cholesky preconditioning matrices; one, type ICCG(0), is computed by keeping the same sparsity pattern as that of the matrix A whilst the other, type ICCG(3), has three extra diagonals of non-zero elements. The idea behind choosing these additional diagonals is that it can be demonstrated that the exact Cholesky factor of a five diagonal matrix A has almost total fill-in between the non-zero diagonals of the matrix A, and that these fill-in elements decrease in size further from

the non-zero diagonals in the positions defined by those of the matrix A. The incomplete Cholesky factors for both ICCG(0) and ICCG(3) are calculated by enforcing the condition that the preconditioning matrix LL^T has the same elements as the matrix A on those diagonals which correspond to the non-zero diagonals of L and L^T.

Meijerink and Van der Vorst demonstrated the existence of incomplete LU factorization for 'M-matrices', with the sparsity of the approximate factors predetermined, and on the basis of which an iterative method of solution would be convergent. They also showed that approximate factors obtained using both classes of sparsity for the matrices L and U are non-singular and the incomplete LU factorization process is at least as stable as the construction of the complete LU factorization without pivoting.

If L is the exact Cholesky factor of the matrix A, then C is the identity matrix and the conjugate gradient method would yield the exact solution in just one step. However, since L is only an approximate decomposition of the matrix A, several steps of the conjugate gradient algorithm will be required. Since the preconditioning will have clustered the eigenvalue spectrum of the matrix C when compared to that of the matrix A, the residual of the preconditioned system should reduce much more rapidly in the first few steps.

5.4 The Preconditioned Conjugate Gradient Algorithm

There are basically three ways of preconditioning a system of equations if the approximate factors of $A = LL^T$ are known [79]; these may be defined as split, left, and right preconditioning. However, it is the split preconditioning which has been used extensively and therefore, the algorithm stated in an Algol-like notation below is based on a split preconditioning i.e. equation (5.2).

Real vectors x, b, r, p, q

Matrices A, L

$\gamma := 1.0$;

$p := 0.$;

$x := L^{-1}b$;

$x := L^{-T}$;

WHILE $norm(r) \leq \epsilon$ **DO**

BEGIN

$q := L^{-1}r$;

$\theta := q^t q$;

$\beta := \theta/\gamma$;

$\gamma := \theta$;

$q := L^{-t}r$;

$p := q + \beta * p$;

$q := A * p$;

$\delta := q^t p$;

$\alpha := \frac{\gamma}{\delta}$;

$x := x + \alpha * p$;

$r := r - \alpha * p$;

END

Generally, both A and L are stored in compacted form in actual computing practice. In fact, memory space for the finite element coefficient matrix A may be dispensed with entirely, since it is only required for a matrix vector multiplication which can be performed quite efficiently on an element by element basis. A may thus be stored implicitly as a set

of individual element matrices, and the only high speed memory needed is that required for one element matrix. Backup memory (disk space) requirements, however, are $O(N)$.

5.5 Preconditioned Conjugate Gradient Frontal Method

The PCCG frontal approach [112] is a new method for solving the large sparse systems of linear equations which arise in three dimensional finite element problems. This method has been developed by combining the frontal method (described in Chapter IV) for matrix assembly and incomplete factorization with the preconditioned conjugate gradient technique. The resulting hybrid method makes the solution of large, three dimensional problems feasible even on machines with limited primary memory, since it automatically trades memory for execution time. The concept is that as problems become large, so computing time increases dramatically but solution is still possible, i.e the algorithm has a 'soft-failing' tendency.

5.5.1 Frontal Gaussian Elimination

The frontal algorithm of Irons made the solution of large two and three dimensional finite element problems feasible on computers of limited memory by eliminating the need to store the entire coefficient matrix in memory at one time. This method is mathematically indistinguishable from conventional Gaussian elimination. Computationally, however, it is very different, for the variables are renumbered (or, what is equivalent, rows and columns of the coefficient matrix are permuted in such a sequence that (1) only a minimal number of matrix rows and columns are needed at any given moment and (2) rows and columns are made active exactly once. In other words, an 'activity front' passes through the ensemble of variables, and each variable is in turn considered (a) not yet active, (b) active, (c) no longer active. Since only the active rows and columns need be housed in core, considerable memory economy is achieved, the active memory required being $O(f^2)$, where f is the

frontwidth, i.e. the largest number of variables active at any time. The computing time is almost entirely spent in triangular decomposition and is $O(f^2 N)$, where N is the total number of variables to be solved for.

For the sake of comparison between the frontal method based on Gaussian elimination and the frontal method based on the preconditioned conjugate gradient algorithm, reference will be made to a model problem. This will be a finite element model of a cuboidal space filled with $(n-1) * (n-1) * (n-1)$ hexahedral first order finite elements. The number of variables is then $N = n^3$. With optimal variable numbering, $f = n^2$. For Irons frontal technique, the total computing time for the model problem is therefore $O(N^{\frac{7}{3}})$, and the high speed memory requirement is $O(N^{\frac{4}{3}})$.

5.5.2 Incomplete Cholesky Decomposition

The incomplete Cholesky decomposition of type ICCG(0) has been used in the preconditioned conjugate gradient frontal algorithm. Incomplete in this context means that a Cholesky decomposition is computed but, whenever the element A_{ij} in the original matrix is zero and $i > j$, the corresponding entry L_{ij} in the Cholesky factor is discarded and set to zero also. The sparsity structure of the decompose and the original matrix are therefore identical. In all cases, the total (disk) storage requirement is kN, where k is the average number of non-zero elements per half-row of A. The computing time required for the conventional ICCG is not analytically known, but it is very close to $O(N^{\frac{3}{2}})$, for a broad variety of problems of stress, thermal, and electromagnetic field analysis. For the model problem, therefore, this technique requires $O(N)$ disk space, $O(f)$ random access memory, and $O(N^{\frac{3}{2}})$ time, better than the corresponding values for the frontal Gaussian decomposition method. Consequently, the ICCG method has become the method of choice for most new software in the electromagnetics area.

5.5.3 Implementation of PCGF Algorithm

An examination of the preconditioned conjugate gradient algorithm indicates that L enters the computation in only two ways:

(a) Classical forward elimination.

(b) Backsubstitution.

Both operations are ideally suited to a sequential work organization and both require at least one row of L at a time to reside in random access memory, along with part of the vector or vectors r or q currently being treated. The part of r and q which must reside in random access memory is $O(f)$, i.e. it is proportional to the frontwidth of the matrix L. The vector inner products and scalar vector multiplications essentially require constant small amounts of memory and may be ignored. The disk space required by L and by the vectors is $O(N)$.

The incomplete decomposition itself can also be organized more easily on a frontal basis than the full decomposition, because the fill-in is strictly controlled and known in advance. Consequently, a frontal work organization is practical, provided that the frontwidth can be held to a reasonable size. This, however, is perfectly feasible using the element resequencing technique described in the last chapter. Frontally organized partial decomposition requires $O(fk)$ memory, since f rows of the matrix L need to be accessed simultaneously, and each row contains k nonzero values. Thus, the decomposition requires $O(N)$ disk space, and $O(f)$ random access memory. Its time requirement is $O(N^{\frac{5}{3}})$, since $O(k)$ operations need to be performed for each of the f matrix rows associated with each of the N variables, and $O(fkN) = O(N^{\frac{5}{3}})$ for the model problem.

5.5.4 The Soft-Failing Algorithm

The random access memory requirements of the ICCG algorithm are dictated primarily by the frontwidth of the preconditioning matrix L. But, since the preconditioning matrix in the ICCG method represents only a rough decomposition of the coefficient matrix A, certain liberties may be taken in computing it. For example, the requirement that L have identical sparsity to A is arbitrary in any case; a different sparsity pattern might be imposed on L if desired. This observation leads to a memory-economic modification of the ICCG algorithm which reduces the frontwidth of L as required, so as to fit within a prescribed amount of memory.

Preconditioning matrices may be more or less arbitrary in many cases. For instance, the preconditioned conjugate gradient method reduces to the ordinary conjugate gradient algorithm if the unit matrix is used for preconditioning, $L = I$. According to Manteuffel [103] the convergence rate is invariant under diagonal scaling, and convergence is always guaranteed in at most N steps, if roundoff error accumulation is negligible. This fact suggests that a gradual transition from ICCG to straight conjugate gradient solution is possible. To do so, the preconditioning matrix L is computed by a slightly different incomplete Cholesky factorization: L_{ij} is forced to zero if either $A_{ij} = 0$, or if the maximum permissible frontwidth f_{max} is exceeded, $(i - j) > f_{max}$. In practical computing, f_{max} is chosen to suit available computer memory.

As problem size grows very large, this procedure slowly degrades the standard ICCG algorithm into an ordinary conjugate gradient method without preconditioning. The disk space required grows linearly with N, and solution time gradually moves from $O(N^{\frac{3}{2}})$ value of ICCG to $O(N^2)$ as preconditioning is lost. However, the random access memory requirement remains fixed at $O(f_{max})$. The effect is that solver performance degrades

markedly – for 10^4 equations, the $O(N^2)$ solver will take 100 times as long as the $O(N^{\frac{3}{2}})$ solver– but there is no longer any 'brick wall' constraint on the number of equations to be solved and the algorithm acquires a 'soft-fail' nature.

Extensive details of the implementations of the 'soft-fail' algorithm are given in [110] and therefore will not be repeated here.

5.5.5 Numerical Results

Numerical experiments were performed which reduced the completeness of the pre-conditioning matrix by gradually reducing the memory available for decomposition. As expected, the algorithm was found to be numerically stable. However, the number of conjugate gradient steps increased as the decomposition became more approximate.

The problem chosen for numerical experimentation was Laplace's equation for the air region surrounding three conductors carrying balanced three phase currents. It was discretized using first order tetrahedra, Fig. 5.1. Two discretizations were used, the first containing 1368 elements and 556 nodes, and the second one 5472 elements and 1390 nodes. The conjugate gradient iterations were stopped when the Euclidian norm of the residue became less than or equal to a specified tolerance. The numerical experiments reported here were performed on a Perq Computer System's PERQ-1 computer.

The performance of the solver with a gradual reduction of the buffer size is summarized in tables 5.1 and 5.2 for each model respectively. The variation in the number of conjugate gradient steps with respect to the Cholesky buffer and with respect to $log(\frac{\sqrt{N}}{Buffer})$ is shown in Figs. 5.2, 5.3 and 5.4.

5.5.6 Discussion

In the preconditioned conjugate gradient routine the behavior of the Euclidian norm of the residue vector r was monitored for both the models. The typical variation of $log(r)$ with respect to CG steps for both problems is shown in Fig. 5.5. Only one run was chosen for plotting because, although there will be a variation with different buffer sizes, this will be reflected in varying convergence rates and the overall nature of the curves will be preserved. Since, the Cholesky decompositions were stable, a pronounced convergence was observed near the solutions for both the models.

Examination of the results of both test problems reveals that a minimum number of CG steps are needed if the preconditioning matrix has been computed with a buffersize of $O(f_{max}^2)$, and are constant until the buffer is reduced from $O(f_{max}^2)$ to $O(f_{max})$. In these cases, they increase gradually from $O(N^{\frac{1}{2}})$ to $O(N)$ as is evident from the curves shown in Figs. 5.2, 5.3, and 5.4.

5.6 Modified ICCG Frontal Method

5.6.1 General

The methods of Meijerink and Van der Vorst were restricted to symmetric M matrices and, in the case of ICCG(0), the sparsity pattern of the incomplete Cholesky factors was forced to be identical to that of the matrix A. For ICCG(3), three extra diagonals were allowed to have non-zero elements in the factor matrices over and above those in the corresponding positions of the matrix A. In both cases, the values of the non-zero elements of the approximate factors were determined by forcing the equality of the elements of the preconditioning matrix LL^T and the original matrix A. The numerical stability during the incomplete factorization and the non-singularity of L was proven for the special case of symmetric M matrices.

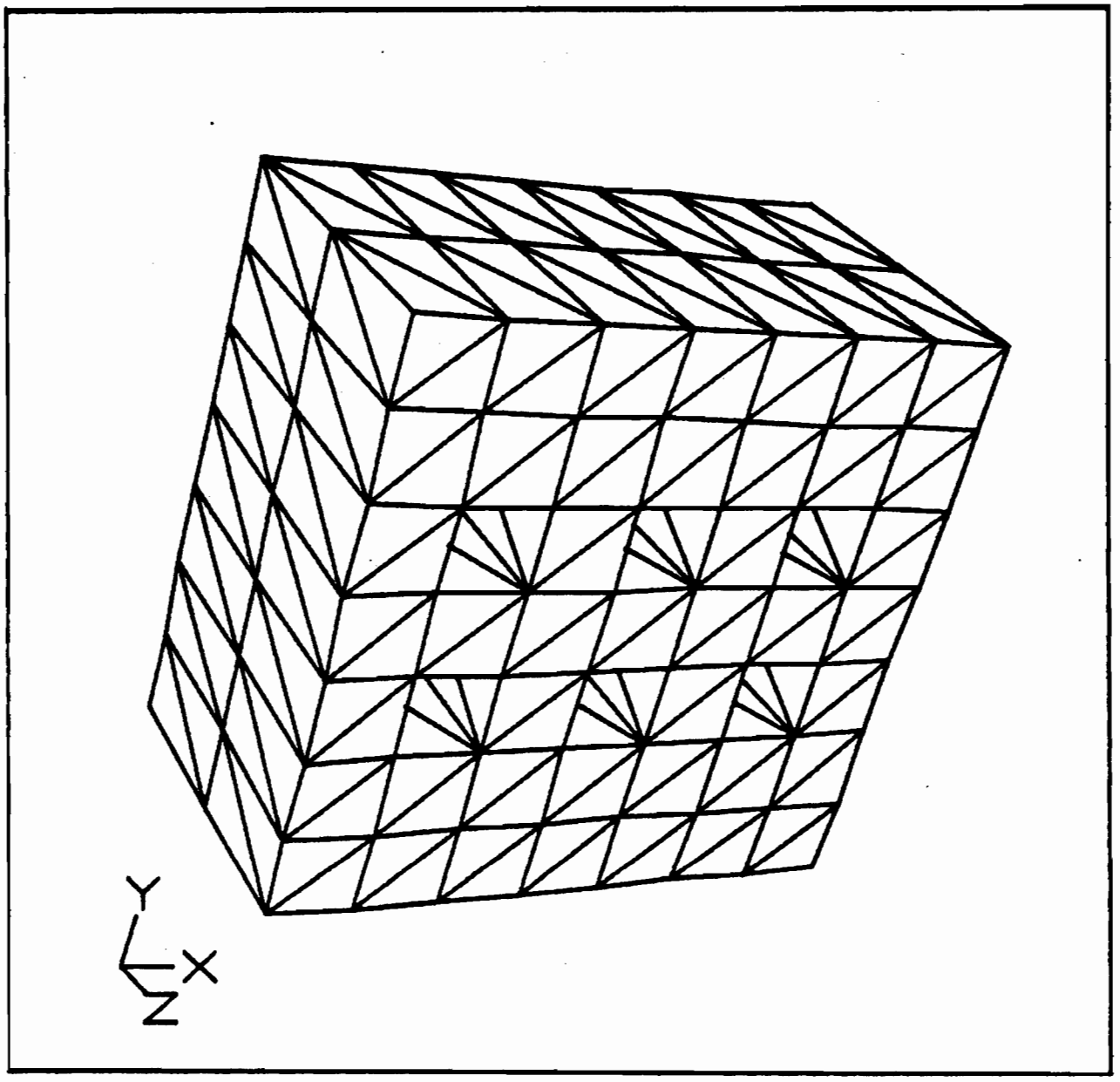


Figure 5.1 A Discretized Model

S.N.	Buffer for Precond. Matrix	CG Steps	Disk Accesses in assembly and elimin.	Disk Accesses in solving	Time in assembly and elimination Seconds	Time in solving Seconds
1	3000	19	3036	43923	10607	2577
2	2500	19	3036	43923	10607	2577
3	2000	19	3036	43923	10607	2577
4	1500	19	3036	43923	10607	2577
5	1000	19	3036	43923	10607	2577
6	500	19	3036	43923	10607	2577
7	250	19	3036	43923	10607	2577
8	200	23	3036	53171	10607	3120
9	150	27	3036	62418	10607	3662
10	100	34	3036	78600	10607	4612
11	50	44	3036	101718	10607	5968
12	10	133	3036	307467	10607	18041

Table 5.1 Three Dimensional Model-1368 Elements

S.N.	Buffer for Precond. Matrix	CG Steps	Disk Accesses in assembly and elimin.	Disk Accesses in solving	Time in assembly and elimination Seconds	Time in solving Seconds
1	8000	37	9642	209890	42484	14350
2	6000	37	9642	209890	42484	14352
3	4000	37	9642	209890	42484	14351
4	2000	37	9642	209890	42484	14351
5	1000	37	9642	209890	42484	14350
6	500	37	9642	209890	42484	14350
7	430	41	9642	232580	42484	15901
8	360	46	9642	260944	42484	17840
9	290	51	9642	289307	42484	19779
10	220	58	9642	329016	42484	22494
11	150	67	9642	380071	42484	25985
12	80	83	9642	470834	42484	32190
13	10	251	9642	1423484	42484	97347

Table 5.2 Three Dimensional Model-5472 Elements

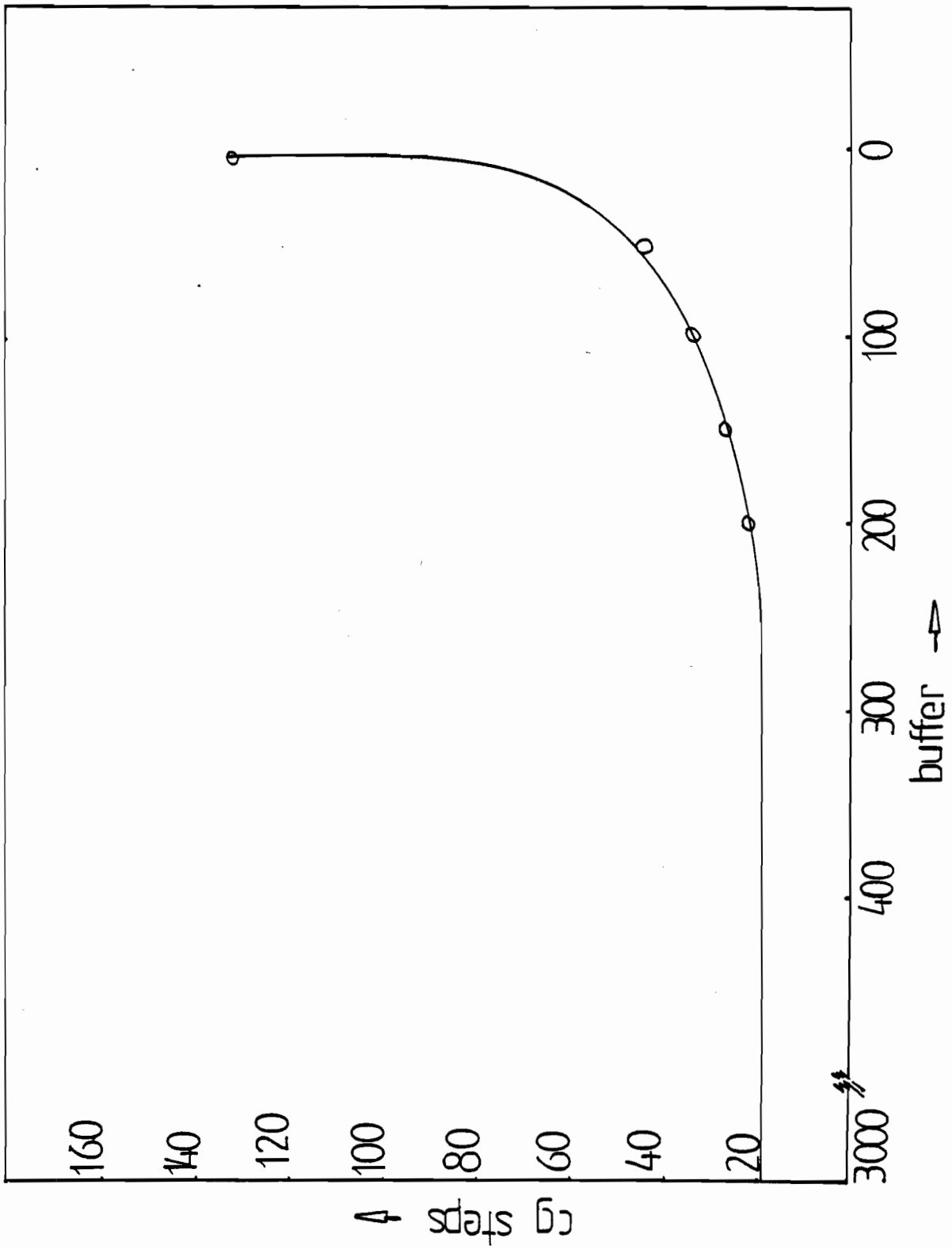


Figure 5.2 Variation of CG steps with buffer - Model with 1368 elements

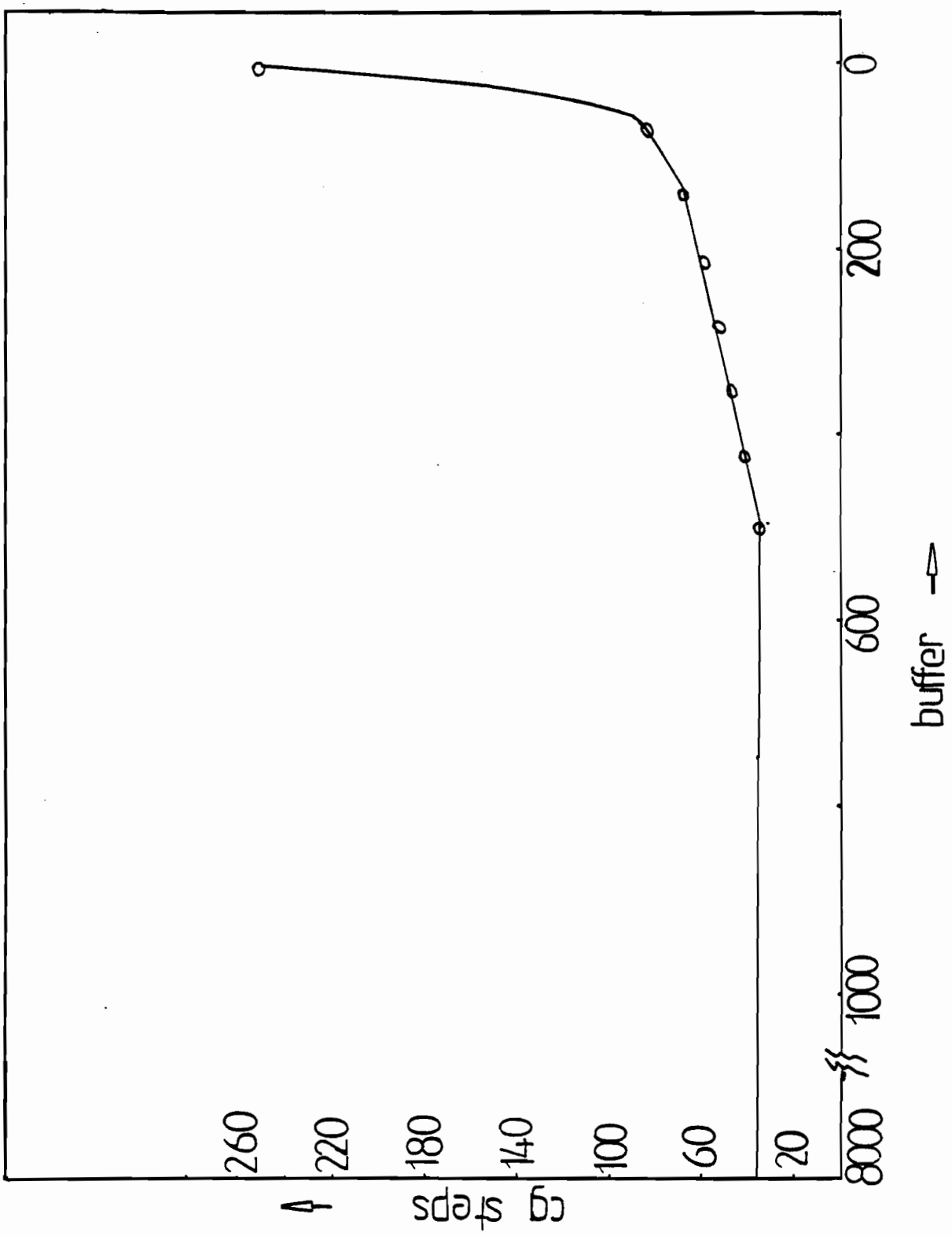


Figure 5.3 Variation of CG steps with buffer - Model with 5472 elements

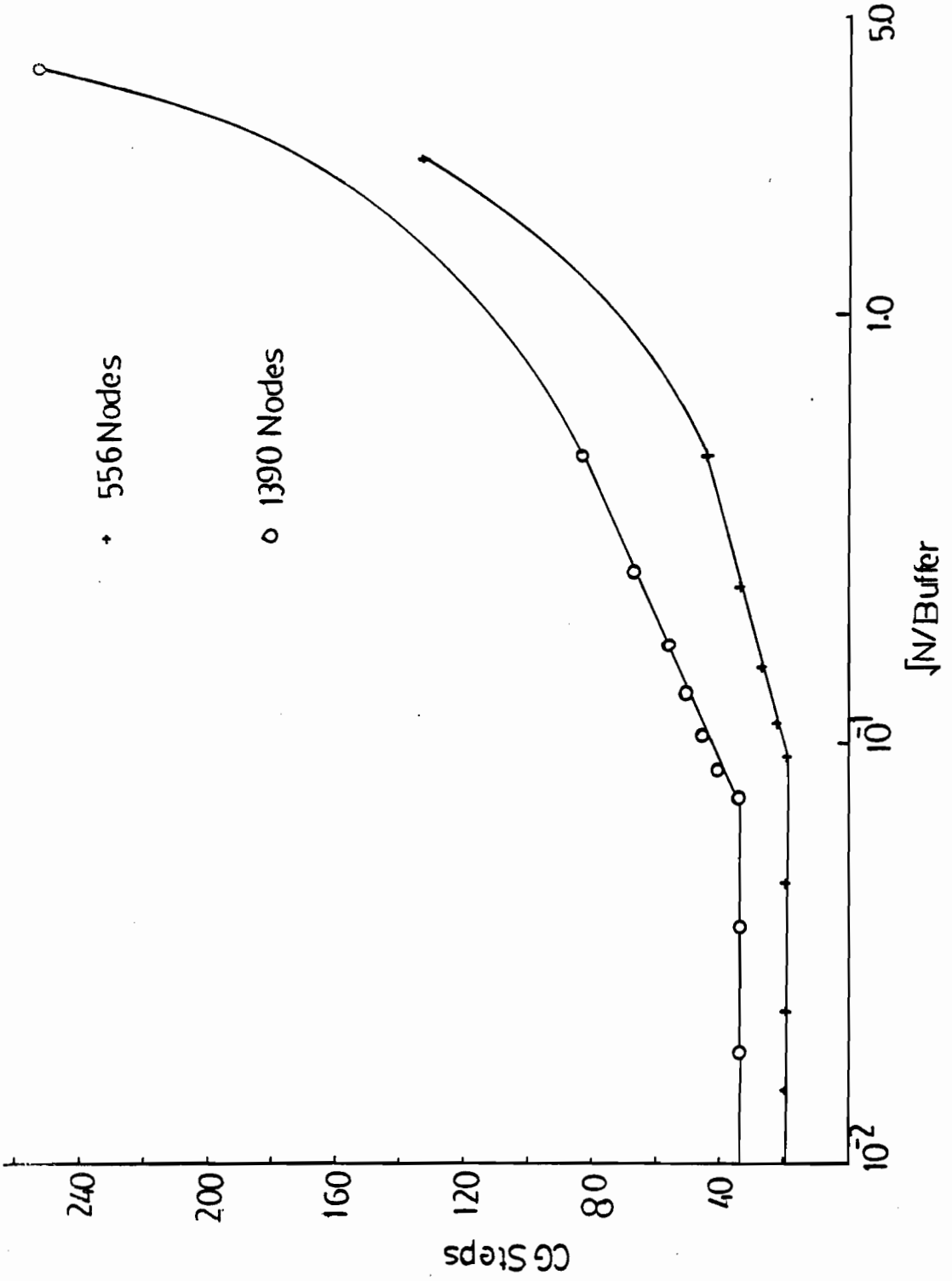


Figure 5.4 Variation of CG Steps with $log(\frac{\sqrt{N}}{Buffer})$

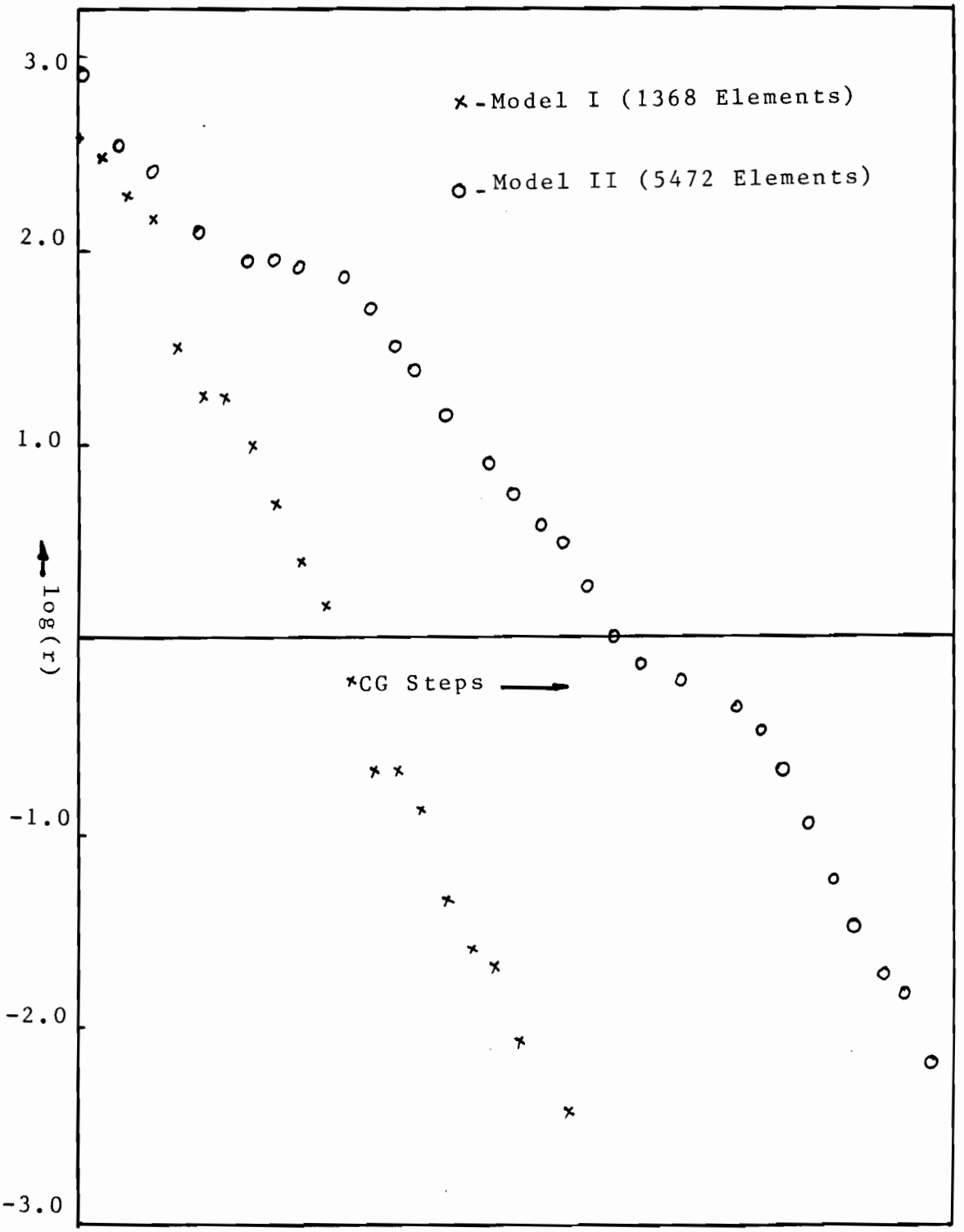


Figure 5.5 Variation of $log(r)$ with CG steps

In their published studies of 1977 and 1979, Meijerink and Von der Vorst [101], [102] showed that both the ICCG(0) and the ICCG(3) methods belong to a class of methods for symmetric M matrices for which the incomplete Cholesky decomposition are derived either by ignoring fill-ins altogether or by allowing fill-ins if a particular acceptance criteria is met. The symmetric successive over-relaxation method of Axelsson [7] also belongs to this class. The conclusion to which Meijerink and Van der Vorst reached is that there is an optimal threshold of fill-in which balances the extra computations needed with the faster rate of convergence obtained with better factors. It was shown that the threshold depends on the diagonal structure of the matrix as well as on the values of the matrix elements.

Kershaw [85] later extended the method of Meijerink and Van der Vorst for a wider class of systems for which the coefficient matrix is symmetric positive definite, but not an M matrix. For an M matrix, the diagonal terms of the Cholesky factor are positive but the same is not true for a general symmetric positive definite matrix. In order to ensure that the preconditioning matrix be positive definite as well as non-singular, Kershaw suggested modifying the main diagonal elements to a suitable positive value, if they happen to be non-positive. Munksgarrd [114], developed a similar algorithm in which fill-ins are controlled by a user defined threshold factor. The fill-in is ignored if it is smaller in magnitude than the tolerance times the diagonal element.

5.6.2 A Modified Cholesky Decomposition

The convergence of iterative methods for symmetric positive definite matrices depends on the condition number of the matrix, which is defined as the ratio of maximum to minimum eigenvalues ($\frac{\lambda_{max}}{\lambda_{min}}$). The basic ICCG(0) algorithm for a symmetric positive definite matrix, in which all the fill-in entries are discarded, results in a preconditioned matrix whose eigenvalues are not very close and as a result their condition number is large. Gustafsson

[65] showed that the condition number can be reduced if the row sum of the preconditioning matrix LL^T be equal to that of A. Munksgarrd used this idea for modifying only the diagonal terms, as the off-diagonal terms were computed by maintaining strict equality between elements of matrices LL^T and A.

The main object of preconditioning a coefficient matrix is to obtain a system which is as close to identity as the imposition of the sparsity on the approximate factor matrices will allow. The product of the preconditioning matrix LL^T is therefore to be as close to as possible to the matrix A. The measure of closeness for a symmetric positive definite matrix is the spread of eigenvalues. The Raleigh quotient bounds the maximum and minimum eigenvalues of a matrix A and can be written as:

$$\frac{(x, Ax)}{(x, x)} \tag{5.6.1}$$

for any non-zero vector x. Thus, the preconditioning matrix LL^T will be close to the matrix A, if the two quadratic forms are approximately equal for all vectors x, i.e.

$$(x, Ax) \simeq (x, LL^T x) \tag{5.6.2}$$

Therefore the Raleigh quotient of the preconditioned matrix $L^{-1}AL^{-T}$ is given by:

$$\frac{(x, Ax)}{(x, LL^T x)} \tag{5.6.3}$$

which is desired to be as close to unity as possible for any vector x and this is possible if the equality of row sums of the preconditioning matrix LL^T and the original matrix A is ensured, i.e. for every row i

$$\sum_{j=1}^{n} |A_{ij}| = \sum_{j=1}^{n} |(LL^T)_{ij}| \tag{5.6.4}$$

Therefore, if the equation (5.1) is preconditioned by a matrix which has been computed by enforcing the row sum equality, then a system results whose condition number $(\frac{\lambda_{max}}{\lambda_{min}})$ is much smaller [173] than the condition number of the preconditioned system of equations (5.2).

The modified incomplete Cholesky factorization is performed in the same manner as the incomplete Cholesky factorization ICCG(0) with exact equality between the off-diagonal elements of matrices A and LL^T, but the diagonal elements are modified to ensure the row sum equality between the rows of the matrix A and the matrix LL^T. It has been observed that the enforcement of these two conditions makes the decomposition both less sensitive to roundoff errors and makes it a closer approximation to that of a complete Cholesky factor.

For a matrix which has k non-zero elements per row, the extra computations beyond those required by the ICCG(0) type preconditioning are of $O(k^2N)$ for forming the product LL^T; $O(N)$ operations for computing the row-sum and $O(N)$ for modifying the diagonals. Thus, the extra work required is $O(k^2N)$, which is small when compared to the total operations of $O(4kN+5N)$ for each CG iteration in the preconditioned conjugate gradient algorithm.

Incorporation of a modified incomplete Cholesky decomposition with the preconditioned conjugate gradient method, along with the frontal algorithm, results in a modified ICCG frontal method whose performance is much better than that of the ICCG(0) frontal method. The comparison of both the algorithms on a set of experimental problems is presented in section 5.6.4.

5.6.3 Computational Problems Involving Air-Iron Interfaces

In general, the elements of the coefficient matrix characterizing three dimensional problems are higher by an order of magnitude than those in coefficient matrices resulting from two dimensional problems for the same medium. The presence of an air-iron interface presents computational difficulties even in 2D, but the problem becomes more aggravated in three dimensions for the following reasons:

(1) In most problems involving electromagnetic devices, regions comprised of air-iron interfaces are made up of thin and/or ill-shaped elements. More often than not the geometries involved are such that the mesh-grading techniques which are commonly employed in 2D are not easily applicable to 3D models, as the size of the mesh grows by $O(N^3)$. As a result, the core requirements in the solution phase can be prohibitive.

(2) The heavy contrast in material properties of iron, with relative permeability values in the thousands and air with relative permeability equal to 1, results in a wider spread of eigenvalues $\lambda_{max} - \lambda_{min}$. In addition, on occasion, it can lead to the loss of numerical stability during incomplete Cholesky factorization due to the generation of negative pivots for the rows which are in, or close to, the interface. The magnitude of these negative pivots is much larger than those arising in 2D problems.

There is no easy cure for the first problem other than always modelling the geometry with well shaped elements and also grading the mesh in regions of intense variation of the field, subject to memory limitations. The magnitudes of the negative pivots in the incomplete Cholesky factors are quite large and the suggestion of Manteuffel [103] of adding small positive real numbers to the diagonals does not always work. Because in the Manteuffel approach the diagonal element is replaced by a small positive number, matrix elements which are computed by the division of this small number can grow to an extent that

exponent overflow problems may be encountered during incomplete Cholesky factorization if the computations are being performed in single precision. On the other hand, if double precision is used and/or the negative pivots are replaced by sufficiently large numbers, the incomplete Cholesky decomposition can be stabilized. Nevertheless, when this matrix is used as a preconditioning matrix there can be convergence problems. It is also interesting to note that the concepts of arbitrary diagonal scaling of the preconditioning matrix or the use of arbitrary preconditioning matrices often do not achieve the CG convergence. Also, the conjugate gradient algorithm without preconditioning does not converge due to accumulation of roundoff errors.

The problems of growth in the magnitude of numbers in the Cholesky factor and negative pivots are related but need to be addressed separately. It has been found, while searching for a cure, that if the coefficient matrix is normalized by a suitable norm of any row which belongs to an air-iron interface, i.e. an interface along which there is a large change in material property, the inherent growth of the matrix elements due to large values of material properties and the use of three dimensional elements can be controlled. The normalization of the coefficient matrix also makes a positive contribution in lowering the magnitude of the negative pivots. The difficulties associated with negative pivots can be solved by replacing them by real numbers which are equal to the sum of the absolute values of all the off-diagonal elements in the row. Incorporation of these two measures appears to lead to an unconditionally stable preconditioned conjugate gradient algorithm which can provide a solution in $O(\sqrt{N})$ steps, even in the presence of several air-iron interfaces, and is computationally cheaper than the algorithm of Aziz and Jennings [2] in which the incomplete factor is computed at every CG steps.

5.7 Numerical Results

Laplace's equation was solved for the different geometries shown in Figs. 4.5 - 4.10,

which were all discretized by second order tetrahedral elements. The models consisted of 243 to 11259 nodes. The comparison of performance of both type of solvers based on ICCG(0) preconditioning and the MIC preconditioning for these models is tabulated in table 5.6. Before starting the solution phase, the elements were sorted to minimize the frontwidth using the algorithm described in Chapter IV, and were renumbered subsequently to minimize the bandwidth. Maximum bandwidth and frontwidth for each mesh is also included in table 5.6. The gains in execution speed for the MIC preconditioned solver are considerable over the ICCG(0)type solver, particularly when the problem size grows, as is evident from Fig. 5.6, which has been plotted between $(\sqrt{N})$ and CG steps, where N is the number of nodes in the model. The variation of the residue with the CG steps for both type of algorithms has been plotted in Fig. 5.7. These results are in close conformity with those of Jacobs [80] and Lavers [89].

5.8 Conclusions

Using the frontal technique, the total number of operations in forming the coefficient matrix A with an average k number of non-zero elements per row will be $O(k^2N)$ and consequently may be performed in $O(N)$ times. Its disk requirements, however, will be $O(N)$. On the assumption that the extra work required in computing the modified incomplete Cholesky decomposition is negligible and is comparable to that of a simple ICCG(0) type incomplete Cholesky decomposition, the ICCG and MICCG algorithms with frontally organized storage and computation sequences, can be implemented at the following asymptotic computing costs:

S.N.	Nodes	Elements	Maximum Front	Maximum Band	CG Steps	
					ICCG(0)	MICCG
1	243	96	41	173	12	8
2	567	288	50	206	16	12
3	891	480	52	248	20	14
4	1215	672	55	266	24	16
5	1539	864	61	270	26	17
6	2187	1248	79	312	31	19
7	2835	1632	107	477	35	21
8	3483	2016	138	584	37	22
9	4779	2784	165	726	43	24
10	6075	3552	175	922	48	25
11	8667	5088	360	975	57	27
12	11259	6624	721	1138	64	39

Table 5.3 Comparison

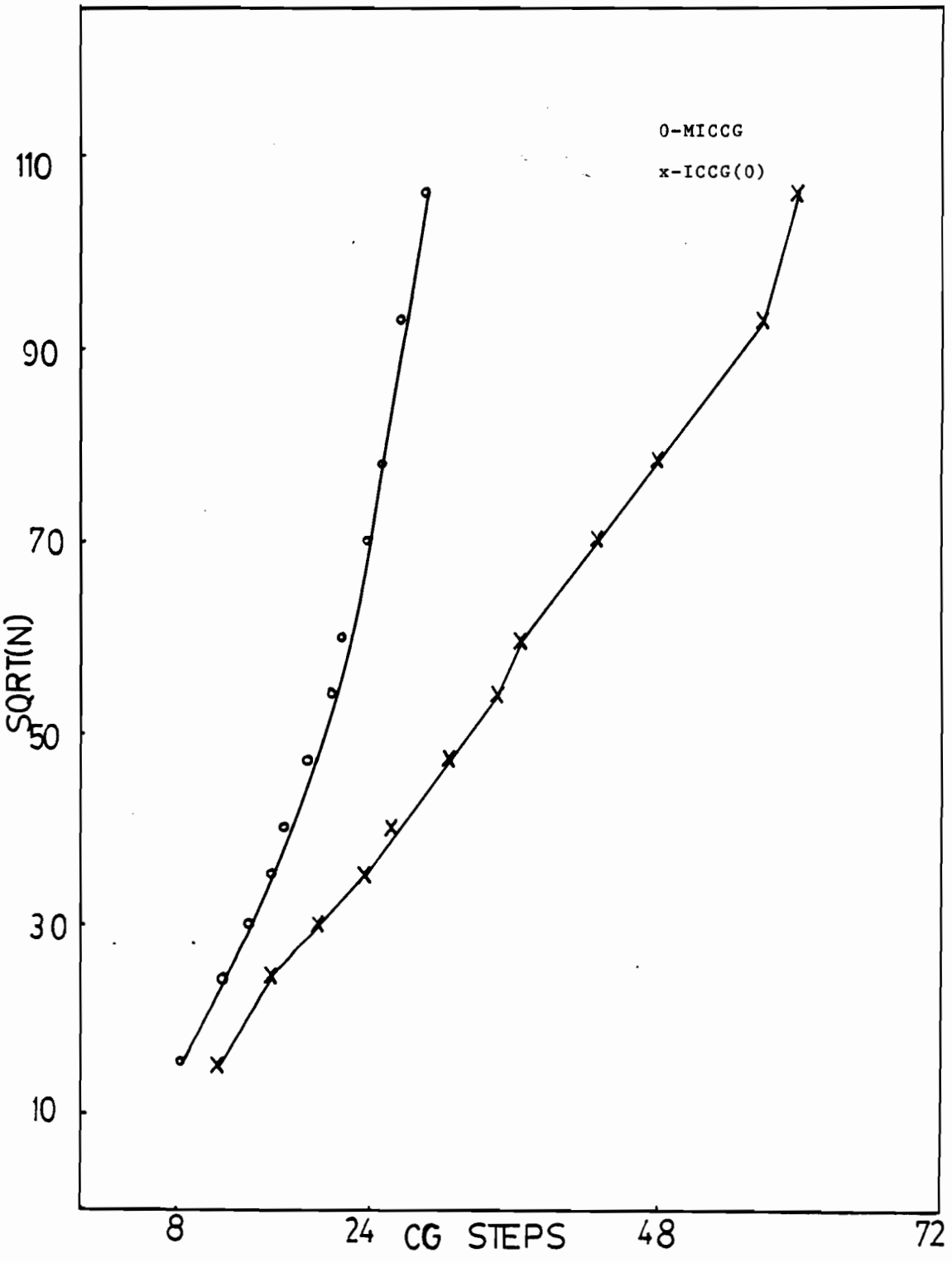


Figure 5.6 Variation of $(\sqrt{N})$ with CG Steps

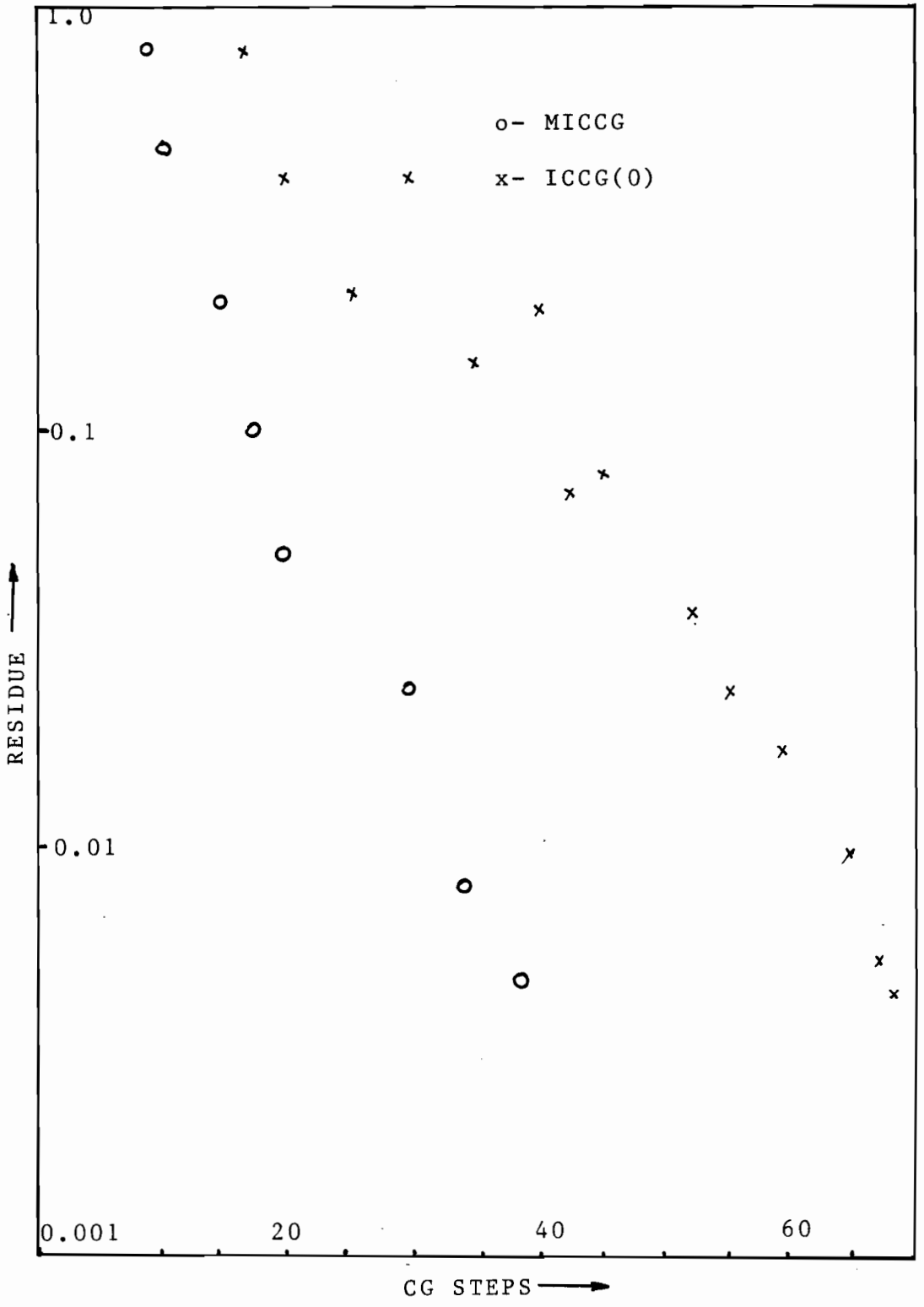


Figure 5.7 Variation of residue with CG steps

Working Step	Time	Disk	Memory
Matrix Setup	$O(N)$	$O(N)$	
Decomposition	$O(N^{\frac{5}{3}})$	$O(N)$	$O(N^{\frac{2}{3}})$
Solution	$O(N^{\frac{3}{2}})$	$O(N)$	$O(N^{\frac{2}{3}})$

From the table it is clear that the time required for the decomposition is a little longer than the time for the solution. But this difference between the exponents $\frac{5}{3}$ and $\frac{3}{2}$ is very small. For example, for a system of equations with 10000 equations the ratio $\frac{(N^{\frac{5}{3}})}{N^{\frac{3}{2}}}$ is approximately four. The exponent $\frac{5}{3}$ for the decomposition is dependent on problem topology to some extent, while the exponent $\frac{3}{2}$ is well established experimentally.

Both algorithms described above provide memory-economic approaches to the solution of equations. While the first algorithm is soft-failing in nature and the other is robust and faster, both are suitable for solving large sets of linear algebraic equations, such as those that arise from the use of finite elements in the solution of three dimensional electromagnetic field problems. These algorithms were devised initially to be used on small computer systems or engineering work-stations. The frontal characteristics, however, coupled with the semi-iterative approach of the conjugate gradient method, probably make them suitable for computer architectures in which there are several interconnected processors, or over a set of networked computers.

Chapter 6

Post-Processing

6.1 Introduction

Post-processing is the last, but probably the most important, phase in any system in which quantities of engineering significance such as impedances, losses and field distributions are derived from the numerical solution provided by the solver and are displayed, if possible. In an industrial environment a newly designed machine is always subjected to rigorous testing to check its compliance with the design specifications. This is done by performing a series of tests on the machine and then comparing them with the design data. Similarly, a post-processor can be considered as a test-bed on which any numerical design of a magnetic device could be tested by performing a series of operations on the solution provided by the solver. Thus, the different tests which are performed on actual machines on a test-bed are simulated in a post-processor by a number of operations. However, the end result - 'to check the validity of the design'- is the same. In the machine it is the actual design but in the post-processor it is the numerical design. In a post-processor, generally, an examination consists of inspecting graphical plots of potentials, fluxes or flux density, eddy current distributions, etc., rather than reading a pile of numeric output, and evaluating some global parameters, e.g. iron losses, inductances, forces.

The post-processors specifically developed for 3D magnetics are still rare and so is the literature on them, although there could be several in use in various research laboratories developed for specific applications. All of these processors lack generality, for example the post-processor which comes with the program package TOSCA is very much limited as it allows the display of potentials and the field components on 2D slices only and the geometries on which these are plotted or computed cannot be displayed thus making comprehension of the results difficult. Also, the processor cannot be used with any other analysis system where the potential function in the formulation is a vector rather than a scalar.

In this chapter, the various problems in designing a general purpose post-processor for three dimensional magnetics will be examined. Also, a scan conversion method for equipotential plotting will be described and its performance will be compared with the other contour plotting techniques, e.g. a conventional contour plotting method using linear first order elements. At the end of the chapter the scan conversion technique will be used to plot the contours on the walls of the terminal box.

6.2 Properties of the Data

The finite element method essentially transforms a continuum physical problem into a discretized problem. Both the process of transformation and the characteristics of the physical problem contribute certain attributes to the discretization which are reflected in the solution produced by the analysis phase. Identification of these attributes and their characteristics will eventually not only govern the structure of the processor but will also help in defining a range of operations which are mathematically or otherwise meaningful and valid.

In magnetics the field is modelled in terms of potentials which have certain restrictions imposed on them by their defining equations and these restriction must be remembered when the potentials are processed. Also, the finite element method being a numerical technique produces a numerical solution, i.e. data produced by the solver are numerical. A numerical solution is inherently an approximate solution because of the idealizations made in the mathematical modelling of the field, in the geometrical modelling, and in the discretization of the device. Also, computational errors result from the nature of the algorithm and the roundoff affects the accuracy of the numerical solution. Thus, further processing of the data has to be performed whilst respecting these approximations which may have significant effect on the final results.

Another important characteristic of the numerical data, which results from the the use of piecewise linear shape functions for modelling the potentials, is that piecewise linear interpolation functions are C^0 continuous only and, consequently, derivatives higher than the first order are meaningless.

6.3 Valid Post-Processing Operations

With the knowledge of the properties of the numerical data provided by the solver, the task of scrutinizing the operations which are valid on them becomes straightforward and easier and will be described in this section.

Essentially, Maxwell's equations and the constituting relations govern all the phenomena taking place in magnetic materials and devices. Because Maxwell's equations have mostly vectors and, occasionally, tensor variables, extraction of a desired quantity from the potential solution involves applications of vector or tensor calculus and algebra in addition to scalar algebra and calculus. Assuming that the application of tensors is relatively rare

in day-to-day analysis and post-processing, the basic operations in a post-processor are those which involve vectors and scalars [99] :

(i) Vector Operations: dot and cross products.

(ii) Scalar Operations: addition, subtraction, multiplication, and division.

(iii) Differential Operations: curl, gradient, and divergence.

(iv) Integral Operations: Line, surface, and volume integration.

These are the operations which may be needed in extracting any desired quantity and it is quite possible that not all the combinations operations may be valid. A valid operation can be defined as one which is mathematically correct and at the same time does not violate the data properties. For example, any operation involving second or higher order derivatives is invalid because the result is zero. Also, if the scalar potential function is used in the formulation, then trying to compute $\nabla(\nabla\Omega)$ is invalid and meaningless.

There is another property of the data which is very important from the point of view of valid operations and that is the dimensionality of the data. Data may be zero dimensional, i.e. a scalar, or a two or a three dimensional vector. A vector operation on a one dimensional scalar may lead to a vector and situations can arise which will lead to invalid operations involving two non-compatible data items.

As long as the mathematical operations performed are valid and the characteristics of the data items on which these are performed are respected it does not matter to what end application the post-processor will be put. In other words, it is quite possible to design a post-processor without *a priori* knowledge of its ultimate application, provided only valid operations are performed.

6.4 Displays

The key to any interactive operation is the ability of the system to provide feedback to the user by displaying the results, and the same is also true for post-processing. The results may vary from pure numerics to three dimensional scalar or vector functions. Before delving into the specifics of displaying three dimensional quantities, it would be ideal to concentrate on the displaying of numerical data in complete generality.

Generally a display of n dimensional data involves $(n+1)$ variables, i.e. it has $n+1$ dimensions. For example, a zero dimensional pure scalar or a numeric requires one dimensional representation which can be displayed either as an alphanumeric string or as a bar chart. Similarly, a one dimensional quantity will represent a two dimensional variation. The typical example of this class can be functions of single variable and will need at least two columns, one for the function and the other for the variable, if the tabular mode of displaying is chosen. Alternatively, this information can be displayed graphically in the form of a curve. The function will be plotted on one axis and the variable on the other.

The task of representation of zero dimensional or one dimensional data which are simple and easily comprehensible becomes more complex when variables of higher dimensionality are encountered. The dimensions of variations grow with the addition of extra space dimensions or if variables change from a scalar to a vector.

Scalar fields of two independent variables occur frequently and are generally represented as isothermals or equipotentials, depending on the field. Colours have also long been used for depiction of such fields, as in the geographical descriptions of the continents in an atlas. Now with the availability of inexpensive colour monitors the use of colours for displaying regions of different intensities is also becoming popular. In the absence of colour

monitors numbered potentials or gray scaling can be used on monochrome display terminals to produce the same effect. Alternatively, such fields may be depicted by using isometric projections. It is a common practice in engineering design offices to provide isometric views of various components on drawings before these components are manufactured. Although this practice is well established and quite old in engineering, it apparently only has been seen in post-processing applications for the last two decades.

While displays of scalar fields have become integral parts of post-processing systems, the same thing cannot be said about vector fields. This is primarily due to the fact that a vector field has one variable more than a scalar field, i.e. its direction, and is, essentially, a four variable field in two dimensions. This makes two dimensional displays difficult. Secondly, the vector field in general does not provide any important engineering information which can be of any use to a designer. Generally a designer is more interested in the components of the vector fields which then become scalars and can be plotted in the form of curves. Nevertheless a two component vector can be represented by arrows. The arrowhead represents the direction of the field whereas the length of the tip corresponds to the magnitude. Examples of such plots are more frequent in flow problems and structural mechanics [90], [40].

The addition of a third space dimension complicates displays of scalar or vector fields further as the scalar and vector fields now have four or five dimensional variations respectively. The degree of complexity of a two component vector field and a three dimensional scalar field, both of which have identically four degrees of freedom, are not the same because in the latter an extra space dimension is involved. The directional component of a planar two component vector field can be treated by arrowheads without any ambiguity on a two dimensional display with no recourse to any means of approximations or appropriate reductions in the dimensionality of the variations, but for a three dimensional scalar field

some form of auxiliary means are a must.

The options available for displaying a scalar field in three dimensions can be:

(1) Plotting of equipotential contours on planar cross-sections which are derived from the geometrical model.

(2) Using the colours for depicting zones of different intensities or to using numbered potentials.

(3) Plotting the contours on the isometric views of the geometric model.

There does not seem to be any way of displaying a three component vector field other than displaying it on a plane. The field is depicted by either arrows in the form of pyramids [8], hockey pucks [117], or stereograms [174].

The field on arbitrary cut-planes is finally projected onto the display screen. In the first approach the sense of direction is conveyed by the direction of arrowheads, whereas in the second approach orientation of the axes of hockey pucks (short cylinders) communicates the same. The magnitude, however, is communicated in both cases by the size of the object.

In the last case, the standard object for representing a vector is also an arrow but stereographics are used for displaying the field. In a stereographic display system two perspective views, one from the left eye and the other from the right eye, are needed. Once both the views are displayed on a cathode ray tube a stereoimage is generated. The right view is displayed by the colour red and the left view by green. The viewer wears a pair of stereoglasses having a red lens in the right and green lens in the left eye. The stereograms can also be used for displaying three dimensional scalar fields.

6.5 Post-Processor Structure

In the preceding sections, the characteristics of numerical data have been examined and valid post-processing operations have been defined. The dimensionality of the data and the complexities with regard to displaying them have also been considered. What is left now is to build a structure which will embody these parts and at the same time preserve the generality of the system.

In three dimensions there is not a single post-processing system which can be said to be general purpose. However, in two dimensions a truly general purpose post-processor MagPost [98] exists which has its origins in the the well known stack-configured Ruthless post-processor [99].

MagPost is a highly structured post-processing system in which three distinct stacks, namely 'field', 'curve', and 'numeric', have been created according to the geometric dimensionality of the data. Operations which use or create two dimensional data are treated in the field stack whereas operations involving one space variable are done in the curve stack. All the operations on pure numerics are performed in the numeric stack. Since the basic operations in each stack involve calculations for extracting desired parameters, these stacks are nothing but three calculators. Each stack handles valid operations on the data, performs stack management operations, and is also responsible for the data display.

Since these stack-configured processors are now well established and have advantages such as a simple command language and efficient data management it seems quite logical that a stack-configured structure will also suit a three dimensional post-processor. The three dimensional processor will have an extra stack to deal with the variables involving three space dimensions in addition to the usual three stacks. This stack will be the largest

in terms of memory and will have more features to take care of applications arising due to three dimensional data. The operations which will be carried out in this main stack will be all the operations of vector calculus, algebra, and arithmetic beside the stack management functions.

A typical structure of a three dimensional post-processor is shown in figure 6.1

6.5.1 Algorithmic Features

A glance through the basic operations needed in a post-processor reveals that the differentiation is the key in forming the gradient, divergence, and curl operators. The best way to perform the differential operations is by using universal finite element matrices [142]. Using these primitive matrices efficient directional differentiation and generation of element S and T matrices is possible. In this approach, the differentiation is performed over the interpolation polynomials and, as a consequence, it is independent of the size and shape of the elements. The gradients, divergences, and curls of various vectors can be accomplished easily. The computation of differentials using universal matrices results in memory economic procedures as only four primitive matrices need to be stored, which are of size m, where m is the order of approximating functions.

Scalar and vector operations are purely algebraic in nature and do not need any special consideration because computing machines are at their best in handling them. Some suggestions for the integration operations, however, can be made. It appears that the best way to perform integration is element by element. At times, while performing contour or surface integrations, the elements involved in the path or space of integration need to be extracted from the entire geometrical database. In such instances it will be preferable if possible, to work with hexahedral bricks or triangular elements, rather than tetrahedra be-

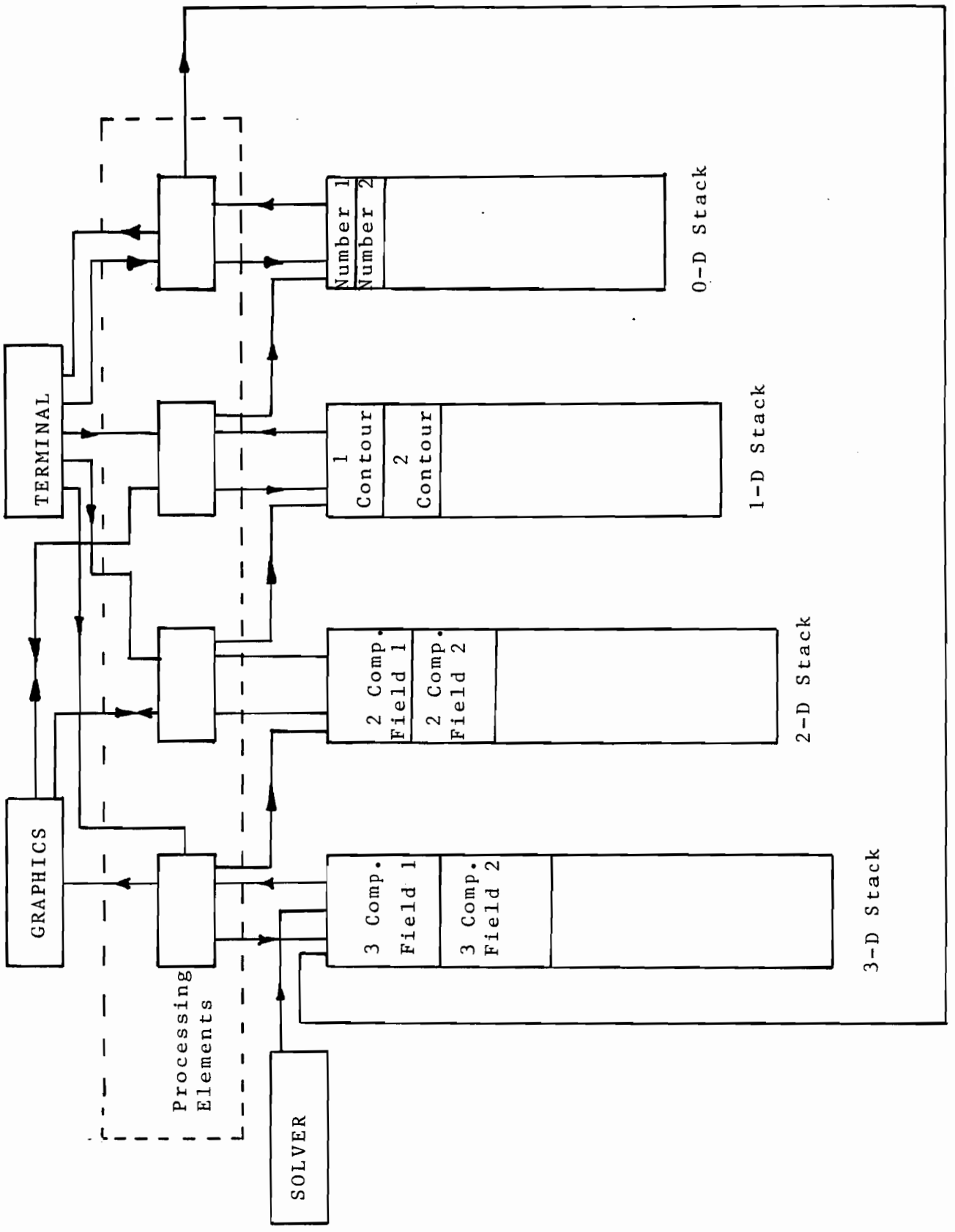


Figure 6.1 Post Processor Structure

cause intersections with the elements could be computed faster and result comprehension may be easier if the accompanying geometry is also to be displayed.

The efficiency of the post-processor is dependent on the choice of suitable data structures. Since most of the operations in the processor will be performed on the entire model or part of the model which is to be extracted from the geometrical database, it is obvious that the data structure of the post-processor should be compatible with the geometrical database of the model.

6.5.2 Memory Requirements

The storage in a post-processor is basically required for three purposes: (a) to store the software which constitutes the various operators; (b) to carry out the operations, and (c) to carry out interstack transfers and to store the results which will constitute the working space.

The storage for the software is fixed for any given post-processor and is dependent on its sophistication and the number of operations it is supposed to perform. This does not mean that the memory size is in any way directly proportional to the number of operations it can perform. By providing a basic set of primitive operators in the post-processor, e.g. universal matrices which need minimal storage, a more complex set of operators can be constructed by the user, depending on his needs. In fact, any operator can be constructed from the basic operators defined in the system as long as valid operations are performed. The ability to generate operators when required totally eliminates the need for storing the generated operators explicitly and, as a result, no extra space other than for storing the primitive operators is required for different operators.

Savings in memory locations are not the only consideration for choosing such a software

structure. The other major consideration is to preserve the flexibility of usage by keeping an open-ended design. By an open ended design is meant that the post-processing system is free from any assumptions with regard to the physical phenomenon and the choice of potential function. As a result an open-ended system can be used with any analysis system, a task which cannot be done by application-restricted post-processors designed primarily for specific applications.

The working space requirements are dependent on the size of the geometric models. Apart from the geometrical database, the solution vector is also required in core in post-processing operations and needs to be saved if any subsequent usage other than the current one is envisaged. Also, in certain instances data has to be transferred from one stack to the other, depending on the dimensionality of the data produced as a result of an operation, e.g. integration operations on a one, two or three dimensional domain result in a zero dimensional pure numeric which need to be transferred to the appropriate stack. These interstack transfers which will usually occur between a stack of higher dimension to one of lower dimension are to be done through a space separately reserved only for data transfer purposes and can be a part of the working space.

The post-processing operations generally need access to one or more of the following input data which are usually stored in different files:

(1.) Geometrical data of the mesh in terms of element lists and node coordinates.

(2.) Material data comprised of magnetization curves, loss curves, etc.

(3.) Potential solutions provided by the solver.

An efficient management of different types of data which may involve extracting a part of the model, a particular loss curve etc. in a post-processor becomes very important if

the memory economy has to be achieved in storing them. A part of the working space has to be specifically assigned for this purpose. Thus the bound on the working space will be $O(N)$, where N is the number of elements in the model and cannot be reduced without sacrificing the response time.

6.5.3 Response Time

It is important to have fast response from the machine to the user's commands if the post-processing is to be done with a real sense of interaction. Response time is dependent on the size of the problem being analyzed as well as on the availability of the data to be processed. Obviously, it will be much better if the problem is small and all the data is in core. More often than not the designer is more interested in processing information pertaining to a local part of the model and this can be done relatively quickly.

The local post-processing involves identification of the zone in terms of space coordinates on which processing is to be done. Once the zone is identified all the elements belonging to the zone are to be brought in the space reserved for the geometrical attributes which is usually a part of the working space. The potential solution pertaining to the nodes belonging to the elements in the zone is also to be extracted and stored in the working space. The desired operations will be performed in the appropriate stacks depending on the dimensionality of the geometric data. Interstack transfers may also be needed if there is a change in the dimensionality of the output data subsequent to the operations. It is always cheaper and faster to solve a few smaller sub-problems than to solve a large one.

6.6 Derived Quantities

The quantities which can be derived from the numerical solution can be many depending on the characteristics of the device being analyzed. In one application it may be the losses

which are of concern; in another it might be forces or temperature rise. Therefore, instead of elaborating on the means of deriving important parameters of electromagnetic devices from the numerical solution which are already discussed elsewhere [96], quantities which are of importance to a terminal box and can be obtained from the scalar potential solution will be considered in this section.

6.6.1 Surface Current Densities

The surface currents on the box walls are related to the scalar potential by the following relation:

$$\oint \mathbf{H} \cdot d\mathbf{s} = i_{enc} \tag{6.6.1.1}$$

But as per the definition of the magnetic scalar potential equation (2.2.2), this enclosed current is equal to the difference in scalar potential between the two sides of the sheet

$$\delta\Omega = -i_{enc} \tag{6.6.1.2}$$

Consider the sheet shown in fig. 6.2 to which a local coordinate system (x,y,z) is attached. Assuming z is normal to the sheet and the origin of the coordinate system is placed at the point where the contour of integration contacts the sheet then,

$$\nabla \times \mathbf{H} = \nabla \times (\delta\Omega) = \mathbf{J}$$

$$\nabla \times (\mathbf{k}\delta\Omega) = \mathbf{i}\frac{\partial}{\partial y}\delta\Omega - \mathbf{j}\frac{\partial}{\partial x}\delta\Omega \tag{6.6.1.3}$$

Therefore,

$$\mathbf{J} = -\nabla \times (\mathbf{k}\delta\Omega) \tag{6.6.1.4}$$

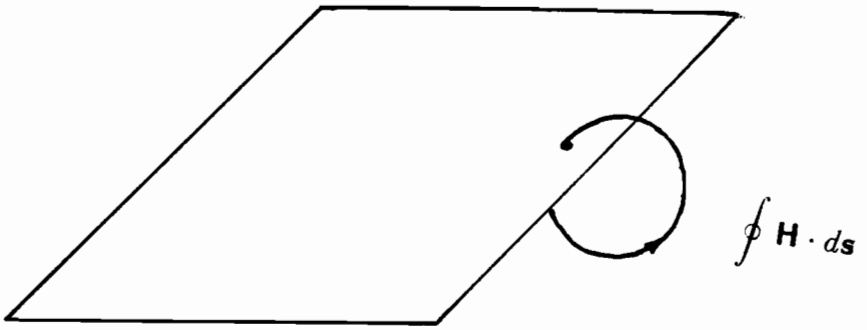


Figure 6.2

Equation (6.6.1.4) gives the local current density and also makes clear that lines of constant Ω are indeed the eddy current paths.

6.6.2 Power Loss

Once the surface current density is known the power loss density can be obtained by using the following expression:

$$W = \mathbf{J} \cdot \mathbf{E} \tag{6.6.2.1}$$

and since $\mathbf{J} = \sigma \mathbf{E}$, the expression for power density becomes,

$$W = \frac{J^2}{\sigma} \tag{6.6.2.2}$$

The integration of W over the wall surfaces will give numerical estimates of the power loss.

6.6.3 Forces

Forces on the iron parts can be computed by various methods based on surface integration [29]. The use of the Maxwell stress tensor appears to be a good choice for the terminal box. The stresses consist of a tension along the 'lines of force' of magnitude $\frac{1}{2}\mu_0 H^2$, and an equal pressure of magnitude $\frac{1}{2}\mu_0 H^2$, at right angles to them. When resolved in normal and tangential directions over a surface, the component of stress directed away from the surface is

$$F_n = \frac{1}{2}\mu_0(H_n^2 - H_t^2) \tag{6.6.3.1}$$

and the tangential component is

$$F_t = \mu_0 H_n H_t \tag{6.6.3.2}$$

6.7 Equipotential Contour Plots

6.7.1 General

The equipotential plots of a scalar or a vector function are generally the first part of any post-processing session because these plots provide the qualitative feeling for the correctness of the numerical solution and, as a result, the equipotential plotting facilities have become one of the standard features of post-processing systems. Some systems have

this facility as a part of the analysis phase also and can immediately provide the plots once the solution to the algebraic equations is known. Consequently, decisions regarding the need to process the results any further can be taken much earlier without having to enter the post-processing phase.

Equipotentials which are the curves joining points having the same function value are desired to be plotted for the entire model. In three dimensions the process takes a very long time even for small models, let alone large ones, and becomes a batch job rather than an interactive process. Moreover, these plots over the entire geometry are, in most cases, totally incomprehensible because of the massive clutter on the screen. Therefore, it has become a general convention to plot the contours on user defined two dimensional slices passing through the geometry which are not only comprehensible but also can be plotted relatively quickly even on small machines. A series of such plots on different planes then provide a three dimensional variation of the solution.

Methods which can be used for contour plotting are:

(i) Linear first order

(ii) Scan conversion

(iii) Higher order curves

The linear first order method is the most popular method because it is fast as well as reliable. In this method the intersections of the contour with the element edges are easily calculable and the intersections can then be joined by straight lines without any ambiguity, whereas in higher order curves the nature of the contour between the intersections is unknown and can be of any shape.

In practice, higher order curves are rarely plotted because of the difficulties associated in finding them inside any element on which they are being plotted. To plot contours on

higher order elements there are only two reliable choices. Either divide the higher order elements to first order elements and then plot linear first order curves, or use the scan conversion method.

The scan conversion method is related to raster displays in which the contour to be plotted on an element is scanned by horizontal lines and is plotted pixel by pixel. The method of contour plotting using scan conversion is relatively new and will be discussed in subsequent sections.

Before the details of the scan conversion method are elaborated it is worth comparing its speed of plotting with the linear first order method as the scan conversion technique is generally considered to be slow [118]. The bound on the number of operations for the linear first order method is $O(N)$ whereas for the scan conversion algorithm it is $O(No.\ of\ pixels)$. Therefore for the first order elements there is no ambiguity about the superiority of the linear first order method. But for higher order elements where the number of elements grows as the square of the order of approximating polynomial, when broken down to first order elements, the scan conversion method can be comparable with the linear first order method because the number of operations increases by an order of magnitude for third and fourth order elements.

6.7.2 The Scan Conversion Method

The method of scan conversion was first proposed by Forghani [54], and is implemented in this thesis. The formulation, which involves the derivation of the expressions of the coefficients of the polynomial being solved, is presented in the next section. The expressions are derived in terms of simplex coordinates and are matched with the mesh produced by the mesh generator described in Chapter III. In the present form it can handle first as well as second order elements.

6.7.2.1 Formulation

In finite element analysis the potential function which models the field is approximated by interpolation functions. The knowledge of these approximating functions is required in determining equipotential points in the model. If second order approximating functions are used for modelling the potential functions, then, for a three dimensional element, it is given by

$$V_{(x,y,z)} = Ax^2 + By^2 + Cz^2 + Dxy + Exz + Fyz + Gx + Hy + Pz + Q \quad (6.7.2.1.1)$$

To obtain this equation for a particular tetrahedron the ten coefficients A, B, C, D, E, F, G, H, P, Q must be determined. Once these coefficients are found the potential value at any node inside the tetrahedron can be determined by specifying its coordinate values. One very expensive possibility for determining the coefficients is to solve a system of 10 by 10 equations for every tetrahedron. The alternate approach is to use interpolation polynomials, which will provide expressions for the coefficients in terms of the coordinate of the nodes and their potential values. In this section, the latter approach which is adopted.

For each tetrahedron, the potential function is represented as a linear combination of approximating functions:

$$u(\varsigma_1, \varsigma_2, \varsigma_3, \varsigma_4) = \sum_{ijkl} u_{ijkl}\alpha_{ijkl}(\varsigma_1, \varsigma_2, \varsigma_3, \varsigma_4) \quad (6.7.2.1.2)$$

where u_{ijkl} are the potential values at the nodes and α_{ijkl} are the approximating functions, expressed in terms of simplex coordinates ς_1, ς_2, ς_3 and ς_4. For a second order tetrahedron using single subscripted notations such that $i + j + k + l = n$, the expression (6.7.2.1.2) can be written as:

$$u = \sum_{n=1}^{10} u_n \alpha_n \tag{6.7.2.1.3}$$

The interpolation polynomial is defined in terms of auxiliary polynomials:

$$\alpha_{ijkl} = R_i(n, \varsigma_1) R_j(n, \varsigma_2) R_k(n, \varsigma_3) R_l(n, \varsigma_4) \tag{6.7.2.1.4}$$

where the auxiliary polynomials are defined as:

$$R_m(n, \varsigma) = \frac{1}{m!} \prod_{k=0}^{m-1} (n\varsigma - k) \tag{6.7.2.1.5}$$

and,

$$R_0(n, \varsigma) = 1 \tag{6.7.2.1.6}$$

The expanded version of (6.7.2.1.3) will be :

$$U = \alpha_1 u_1 + \alpha_2 u_2 + \alpha_3 u_3 + \alpha_4 u_4 + \alpha_5 u_5 + \alpha_6 u_6 + \alpha_7 u_7 + \alpha_8 u_8 + \alpha_9 u_9 + \alpha_{10} u_{10} \tag{6.7.2.1.7}$$

where

$$\alpha_1 = R_2(n, \varsigma_1)$$

$$\alpha_2 = R_1(n, \varsigma_1) R_1(n, \varsigma_2)$$

$$\alpha_3 = R_1(n, \varsigma_1) R_1(n, \varsigma_3)$$

$$\alpha_4 = R_1(n, \varsigma_1) R_1(n, \varsigma_4)$$

$$\alpha_5 = R_2(n, \varsigma_2)$$

$$\alpha_6 = R_1(n, \varsigma_2) R_1(n, \varsigma_3)$$

$$\alpha_7 = R_1(n, \varsigma_2) R_1(n, \varsigma_4)$$

$$\alpha_8 = R_2(n, \varsigma_3)$$

$$\alpha_9 = R_1(n, \varsigma_3) R_1(n, \varsigma_4)$$

$$\alpha_{10} = R_2(n, \varsigma_4)$$

These expressions for the $\alpha's$ can be transformed in terms of $\varsigma's$ by using (6.7.2.1.5) and (6.7.2.1.6) and, when finally substituted in (6.7.2.1.7), the resulting expression is:

$$\begin{aligned} u = (2\varsigma_1^2 - \varsigma_1)u_1 + 4(\varsigma_1\varsigma_2)u_2 + 4(\varsigma_1\varsigma_3)u_3 + 4(\varsigma_1\varsigma_4)u_4 + \\ (2\varsigma_2^2 - \varsigma_2)u_5 + 4(\varsigma_2\varsigma_3)u_6 + (4\varsigma_2\varsigma_4)u_7 \\ +(2\varsigma_3^2 - \varsigma_3)u_8 + (4\varsigma_3\varsigma_4)u_9 + (2\varsigma_4^2 - \varsigma_4)u_{10} \end{aligned} \qquad (6.7.2.1.8)$$

The simplex coordinates $\varsigma's$ can be expressed in terms of Cartesian coordinates by using the relation

$$\varsigma_m = \frac{\sigma(s_m)}{\sigma(s)} \qquad (6.7.2.1.9)$$

where $\sigma(s_m)$ is the subsimplex of simplex $\sigma(s)$, which can be defined as:

$$\sigma(s) = \frac{\frac{1}{3!}}{\begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}} \qquad (6.7.2.1.10)$$

Any point p inside a simplex, whose coordinates are x, y, z, divides it into $n+1$ subsimplexes, where n is the dimension of the space and each of these simplexes have $n+1$ vertices. Therefore, with the help of (6.7.2.1.9), the expression for ς_1 can be written as:

$$\varsigma_1 = \frac{\begin{vmatrix} 1 & x & y & z \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}}{\begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}} \qquad (6.7.2.1.11)$$

which finally becomes:

$$\begin{aligned}\varsigma_1 = \frac{1}{\Delta}\{&x_1(y_4z_3 - y_3z_4 + y_2z_4 - z_2y_4 - y_2z_3 + y_3z_2)\\ &+y_1(x_3z_4 - x_4z_3 - x_2z_4 + x_4z_2 + x_2z_3 - x_3z_2)\\ &+z_1(x_2y_4 - x_4y_2 - x_3y_4 + x_4y_3 - x_2y_3 + x_3y_2)\\ &+x_2(y_3z_4 - y_4z_3) - y_2(x_3z_4 - x_4z_3) + z_2(x_3y_4 - x_4y_3)\}\end{aligned} \qquad (6.7.2.1.12)$$

where Δ is the determinant of the denominator. Similarly, expressions for ς_2, ς_3 and ς_4 are derived and are contained in Appendix II.

Once the values of ς_1, ς_2, ς_3 and ς_4 are substituted in (6.7.2.1.8) and the coefficients of x^2, y^2, z^2, xy, xz, yz, x, y, z and the constant are collected then, upon equating this expression with (6.7.2.1.1), the coefficients A, B, C, D, E, F, G, H, P and Q are determined. The expressions for these coefficients are given in Appendix II.

6.7.2.2 Reduction from 3D to 2D

If the potentials at the nodes of a tetrahedron are known and the coefficients of the polynomials (6.7.2.1.8) are also known in terms of the Cartesian coordinates, then the plotting operation requires finding a set of x, y ,z coordinates which satisfy equation (6.7.2.1.1) for a given potential. The problem can be simplified if the y and z coordinates are kept constant and the equation is solved for x. This particular approach is similar to that of FINPLT [34].

If the cut-plane is parallel to the X-Y plane then it has a constant z value. If not, it can be rotated to become parallel to the X-Y plane. Now, if the cut plane is translated to $z = 0$, the problem is reduced from three dimensions to two dimensions and, as a result only 6 coefficients of equation (6.7.2.1.1) need to be determined instead of 10. To be able

to plot the contours on a cut-plane both the cut-plane and its intersections with the finite element mesh, must be defined. Also, the solution for each of the nodes belonging to the plane is needed. The intersection may be either triangular or quadrilateral. In the latter case the quadrilateral is divided into two triangles.

6.7.2.3 Plotting

Once the information required to plot the contours on a cut-plane is available, plotting is started. One of the attractive features of the plotting is the use of the structure of the graphics system. The method is based on a form of scan-conversion using the fact that a raster display consists of a set of horizontal lines. These horizontal lines, which cover the cross-sections of elements, are found and then in turn are examined to check if any of the required potential values occur on them. The task of examining whether the lines lie within a triangle is done by solving the polynomial equation as explained below.

The second order function to be plotted inside the triangle can be written as:

$$u_{(x,y)} = Ax^2 + By^2 + Cxy + Dx + Ey + F \qquad (6.7.2.3.1)$$

For a row of pixels as shown in Fig. 6.3, the y-coordinate value, e.g. y_1, is known. Also, the solution for which the plot is to be made is known, say v_1. Then

$$v_1 = Ax^2 + By_1^2 + Cxy_1 + Dx + Ey_1 + F \qquad (6.7.2.3.2)$$

which is reduced to

$$Ax^2 + (Cy_1 + D)x + (By_1^2 + Ey_1 + F - v_1) = 0 \qquad (6.7.2.3.3)$$

The details of projections from one Z plane to another, reducing the problem from 3D to 2D, and the procedures for generating the intersections of the mesh with the cut-plane along with the corresponding solutions are available in [54] and, hence, will not be repeated here.

A three dimensional post-processing system having attributes similar to that described in preceding sections can be used to examine the terminal box. The contour plotting method based on scan conversion technique has been applied for plotting the contours on one of the terminal box wall which are shown in the next section.

6.8 An Application

The equipotentials which are shown in Fig. 6.5 were plotted on a plane using the scan conversion method. The plane was derived from the model shown in Fig. 5.1 which was constructed using the interactive three dimensional mesh generator described in Chapter III. The discretized model consisted of 11772 first order elements and 3010 nodes. The elements of the mesh were first sorted to minimize the frontwidth and subsequently renumbered to minimize the bandwidth of the resulting coefficient matrix using the new algorithm presented in Chapter IV. The solution to the problem was obtained by the preconditioned conjugate gradient frontal solver described in Chapter V.

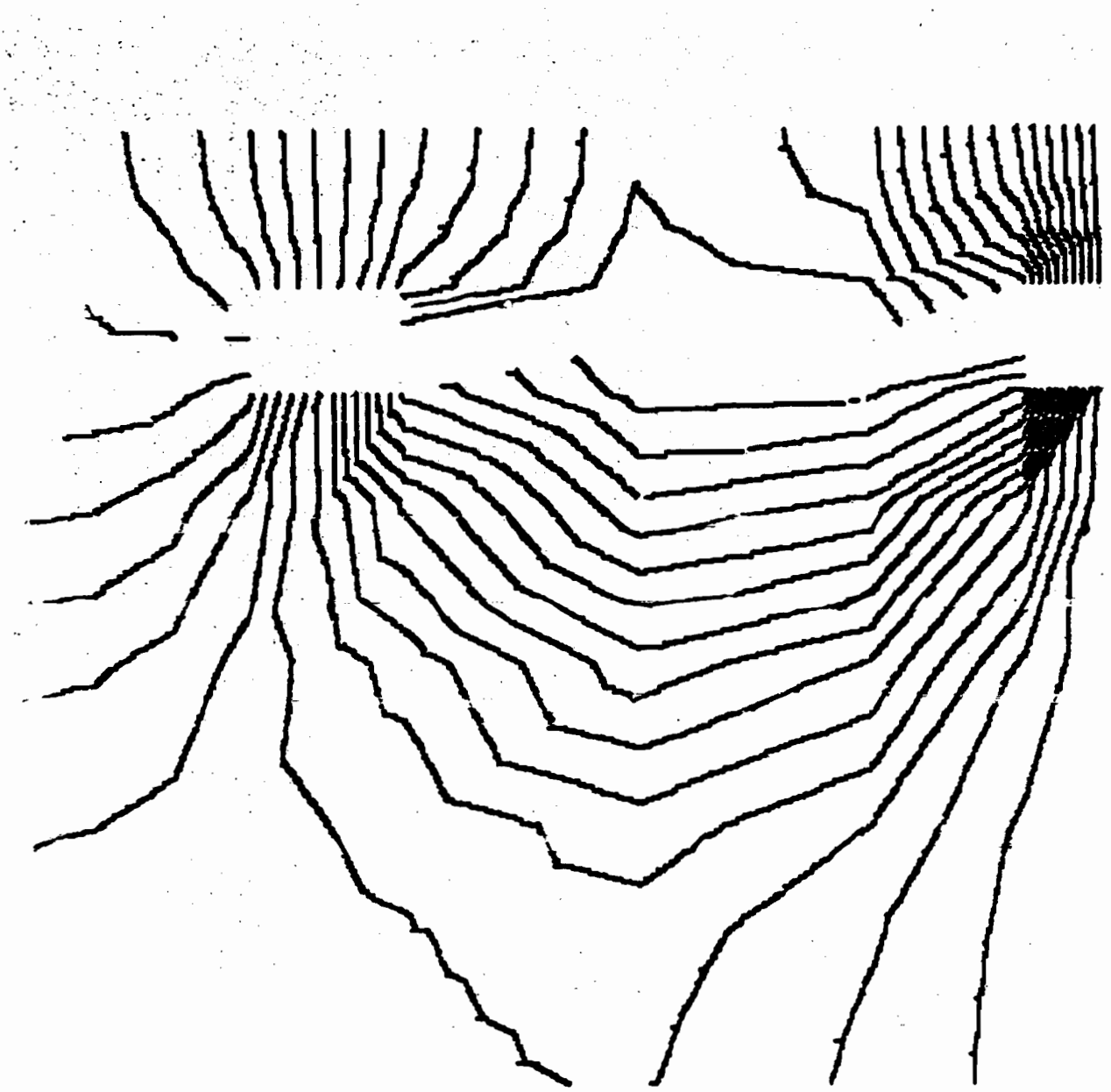


Figure 6.5 Equipotential plot on a wall of the terminal box

Chapter 7

Conclusions

This thesis has developed a computationally economic magnetic modelling system for three dimensional magnetostatic field problems. The whole system has evolved from the need to determine the surface current distributions on the walls of a turbogenerator terminal box- a problem which belongs to the class of problems characterized by low skin depth. It has been shown that such problems can be modelled using magnetic scalar potential alone, instead of pairs of a scalar and a vector potentials minimally required. A significant computational economy is achieved by reducing the number of degrees of freedoms per node from four to one as a direct consequence of the low skin depth approximation. The solution of a large number of variables which are characteristic of a three dimensional domain, generally leads to high computational costs due to the space and time requirements. Therefore, it is very important that in a finite element analysis system, the overall reduction in computing cost should stem not only from memory economic mesh generators, fast converging algebraic equation solvers, and efficient post-processors, but also from the potential function which has been chosen to model the phenomenon. Only then can such problems be solved on machines of limited address spaces.

As part of the system, an interactive three dimensional mesh generator has been developed to model geometries using irregular hexahedral bricks. The model could be refined

by subdividing each hexahedron into subhexahedra-in all three space directions. Once the model is ready, the mesh is automatically generated by dividing each brick into either six tetrahedra or two triangular prisms. Either first order or second order elements can be generated. The mesh generator runs on an LSI -11/23 with 32 Kbyte of memory and can be transported to other machines. The geometric database is highly organized in the form of an octal tree and, as a result, all the searches for nodes run in 'binary' $(nlogn)$ time.

A new algorithm for element resequencing to minimize the wavefront for the frontal solver has been introduced. The algorithm is a natural method of element sorting and is similar in theory to a Gaussian elimination algorithm. The elements are sorted and arranged as a group for each node in a natural fashion and they are eliminated one by one in the same order in the matrix assembly and the elimination phase. This procedure has two distinct advantages; first, there is no need to renumber the nodes separately and, second, the task of destination allotment for every node is completed when the last element sharing this node is sorted out.

One of the best features of this algorithm is that the element list corresponding to one node is in core at one time. Unlike the SAS algorithm in which a level structure of maximal depth is constructed with a memory size of $O(N^{\frac{1}{2}})$ for two dimensional and $O(N^{\frac{2}{3}})$ for three dimensional meshes, the memory requirements of the present algorithm are $O(k)$, where k is the number of elements shared by one node and generally k is $O(1)$. Other significant advantages of the algorithm are that it is insensitive to the choice of starting node (at least on all the experimental models on which it has been tried) and needs only one iteration.

The performance of the new algorithm has been compared with the SAS algorithm on topologically different models and in most cases it gave better frontwidths than the SAS

algorithm. Further experimental testing on larger models of similar topologies resulted in considerable reductions in the frontwidths, bandwidths, profiles, and root-mean-square frontwidths. Moreover, the algorithm respected the upper bound for the number of comparisons $O(mMN)$ for all the runs. Here m is the number of nodes in an element of a mesh consisting of N elements and M nodes.

Two memory economic algorithms, one 'soft-failing' and the other 'fast converging and robust', for the solution of systems of linear algebraic equations are introduced. Both the algorithms are developed by combining the advantages of the frontal algorithm for the matrix assembly and its subsequent factorization, and the preconditioned conjugate gradient algorithm for solving the resulting matrix equations iteratively.

The soft-failing algorithm works for any size problem but gradually deteriorates in efficiency as the memory size is reduced. This behaviour results because the quality of the preconditioning matrix decreases with reduced computer memory, resulting in increased conjugate gradient iterations. Problems which can fit in machine memory are solved with a true incomplete Cholesky preconditioning matrix, while others which do not fit are solved with more incomplete preconditioning matrices. So, there is a gradual transition in the algorithm from a preconditioned conjugate gradient state towards a pure conjugate gradient state. But now there is no longer any constraint on the number of equations to be solved.

The second algorithm, which is faster than the first one, also employs the incomplete Cholesky decomposition matrix as a preconditioning matrix, but is computed with two restrictions in addition to those in the first algorithm such that:

(a) all the off-diagonal non-zero elements of matrix (LL^T) be equal to that of the coefficient matrix A, and

(b) the row-sum of matrix LL^T be equal to that of A.

With these two restrictions, the condition number of the preconditioned system is much smaller than that of the first algorithm and, as a result, the second algorithm has a faster convergence than the first.

It has been observed that the elements of the coefficient matrix characterizing 3D problems are larger in magnitude than those for a 2D section of the same problem. This fact, combined with the material property differences encountered at the air-iron interfaces, can make the difficulties normally encountered in two dimensional problems even worse in three dimensions. The problem is further compounded because air-iron interfaces in most electromagnetic devices often consist of, and are surrounded by, thin and badly shaped elements. Although most mesh generation systems attempt to avoid this situation, its complete removal seems difficult. The combined effect of these two factors is that negative pivots are created during the incomplete factorization for rows which represent areas inside or close to the interface. It has been found that many of the traditional cures, such as diagonal scaling or Manteuffel shifts, do not seem to be totally successful in overcoming these difficulties and an alternate approach, that of normalization, is a better solution for such situations.

By normalizing the coefficient matrix by a suitable norm of a row which belongs to the interface, the difficulties associated with the excessively large numbers which can cause overflow can be overcome. The replacement of a negative pivot by a suitable number which has been found to be equal to the sum of absolute values of the off-diagonal elements is very crucial for the convergence of the CG algorithm. The incorporation of both the measures makes the second algorithm numerically stable and it generally converges in $O(N^{\frac{1}{2}})$ iterations, even in the presence of a wide range of material properties and element shapes. The asymptotic computing costs of both the algorithms can be summarized as:

Working Step	Time	Disk	Memory
Matrix Setup	$O(N)$	$O(N)$	
Decomposition	$O(N^{\frac{5}{3}})$	$O(N)$	$O(N^{\frac{2}{3}})$
Solution	$O(N^{\frac{3}{2}})$	$O(N)$	$O(N^{\frac{2}{3}})$

Extraction of engineering information in terms of local and global quantities from the numerical solution is accomplished in the post-processing phase, which is probably the most important phase in the finite element analysis. A design of a three dimensional post-processor has been developed by considering the characteristics and limitations of the various attributes which constitute such a processor. Identification of the attributes such as properties of data, valid post-processing operations, and displays and their characteristics is important because these factors not only govern the structure of the processor but also helpful in defining a range of operations which is mathematically valid and meaningful on the numerical data.

Displays are the key to interactive post-processing and, as a result, have been treated by classifying the data to be displayed according to its dimensionality. Various possible means of displaying data ranging from a zero dimensional scalar to scalars or vectors which are functions of upto three space dimensions are suggested. General considerations related to the structure of the processor such as algorithmic features, space requirements and response time have been examined also. It has been concluded that a post-processor embodying the features discussed above will be potential formulation independent and its design will not require *a priori* knowledge of ultimate future applications.

A new contour plotting technique for plotting higher order curves using the scan conversion feature of raster displays on the user defined cut-planes has been developed and its performance speed has been compared with the linear first order method of contour

plotting. This developed system has been used for displaying the solution on the terminal box walls.

7.1 Recommendation for Future Work

There are many areas of research relating to this thesis that were not explored or implemented due to the limitations of time and scope. The areas which need further exploration/implementation are:

(1) In the present analysis of the terminal box a linear model for the ferromagnetic sheets has been used: further search should extend to the use of a nonlinear one.

(2) The general purpose three dimensional post-processor whose design has been proposed in chapter VI should be fully implemented so that desired quantities of interest can be derived from the potential solution and displayed if desired.

(3) It will be interesting if the computed results are compared with the experimental results performed on an actual machine.

(4) The influence of the geometrical dimensions on the losses and the forces should be studied by solving the problem with different geometrical models. Studies of this kind may finally lead to design optimizations.

(5) Suitability of sheet iron of particular magnetic characteristics from the point of view of losses should also be examined.

(6) The method of analysis presented in this thesis should be applied to other devices, such as transformer tanks, which belong to the same class of problems as the terminal box.

(7) Improvements in the scan conversion method of contour plotting should be done to make it run faster. One of the possible means of accomplishing this is by solving the polynomial equation on pixels which are just above and below the current pixels.

References

[1] Abbas, S. F., 'Some Novel Applications of the Frontal Concepts', *Int. Jour. for Num. Methods in Engg.*, Vol. 15, 1980, pp. 519-536.

[2] Ajiz, M. A. and Jennings, A., 'A Robust Incomplete Cholesky-Conjugate Gradient Algorithm', *Int. Jour for Num. Methods in Engg.*, Vol. 20, 1984, pp. 949-966.

[3] Akin, J. E., and Pardue, R. M., 'Element Resequencing for Frontal Solutions', in *The mathematics of Finite Elements and Applications*, edited by J. R. Whiteman, pp. 535-541, Academic Press, New York, 1974.

[4] Auda, H. A., 'Memory Economic Finite Element and Node Renumbering', M. Eng. Thesis, Dept. of Electrical Engineering, McGill University, 1981.

[5] Armstrong, A.G.A.M., Collie, C. J. et al., 'New Developments in the Magnet Design Program GFUN', Rutherford Appelton Laboratory Report RL-78-088.

[6] Axelsson, O. and Munksagaraad, N., 'A Class of Preconditioned Conjugate Gradient Methods for the Solution of a Mixed Finite Element Discretizations of the Biharmonic Operators', *Int. Jour. for Num. Methods in Engg.*, Vol. 14, 1979, pp. 1001 - 1009.

[7] Axelsson, O., 'A Generalized SSOR Method', BIT 12, 1972, pp. 443-467.

[8] Baden Fuller, A. J. and dos Santos, M. L., 'Computer Generated Display of 3D Vector Fields', *Computer Aided Design*, Vol. 12, No. 6, March 1980, pp. 61-66.

[9] Baer, A., Eastman, C., Henrion, M., 'Geometric Modelling: A Survey', *Computer Aided Design*, Vol. 11, No. 5, September 1979, pp. 253-272.

[10] Barnhill, E. E., Birkloff, T., and Gordon, W. J., 'Smooth Interpolation in Triangles', *Jour. of Approximation Theory*, Vol. 8, 1973, pp. 114-477.

[11] Beer, G., 'Recent Developments in Finite Element Analysis Using Minicomputers', Proc. of Third Int. Conf. in Australlia on Finite Element Methods, The University of New South Wales, July 1979.

[12] Biddlecombe, C. S., Collie, C. J., Simkin, J. and Trowbridge, C. W., 'The Integral Equation Method Applied to Eddy Currents', Proc. of Compumag Conference, Oxford, 1976, pp. 363-372.

[13] Biddlecombe, C. S., Heighway E. A., Simkin J., and Trowbridge, C. W., 'Methods of Eddy Current Computations in Three Dimensions', *IEEE Trans. on Magnetics*, Vol. MAG-18, No. 2, March 1982, pp. 492-497.

[14] Brown, M. L., 'Calculation of 3D Eddy Currents at Power Frequencies', *Proc. IEE*, Vol. 129, Pt. A, No. 1, Jan. 1982, pp. 46-53.

[15] Brown, M. L., 'Scalar Potentials in Multiply Connected Regions', *Int. Jour. for Num. Methods in Engg.*, Vol. 20, 1984, pp. 665-680.

[16] Brown, R. P., 'A non-interactive Method for the Automatic Generation of Finite Element Meshes Using the Schwarz-Christoffel Transformations', *Comp. Methods in Applied Mech. and Engg.*, Vol. 25, 1981, pp. 101-126.

[17] Bryant, C. and Freeman, E. M., 'A Highly Interactive Three Dimensional Mesh Generator', *IEEE Trans. on Magnetics*, Vol. MAG-19, No. 6, 1983, pp. 2531-2534.

[18] Bryant, C. and Freeman, E. M., 'Three Dimensional Finite Element Preprocessing of Magnetic Field Problems', *IEEE Trans. on Magnetics*, Vol. MAG-20, No. 5, 1984, pp. 1897-1899.

[19] Bunch, R. J. and Rose, D. J., *Sparse Matrix Computations*, Academic Press, New York, 1976.

[20] Bykat, A., 'Automatic Generation of Triangular Grid: I-Subdivision of General Polygon into Convex Subregions. II-Triangulation of convex Polygons', *Int. Jour. for Num. Methods in Engg.*, Vol. 10, 1976, pp. 1329-1342.

[21] Bykat, A., 'A Note on an Element Ordering Scheme', *Int. Jour. for Num. Methods in Engg.*, Vol. 11, 1977, pp. 194-198.

[22] Cambrell, G., - Private Communication.

[23] Campbell, P., Chari, M. V. K. and D'Angelo J., '3D Finite Element Solution of Permanent Magnet Machines', *IEEE Trans. on Magnetics*, Vol. MAG-17, No. 6, 1981, pp. 2997-2999.

[24] Cantin, G., 'An Equation Solver of Very Large Capacity', *Int. Jour. for Num. Methods in Engg.*, Vol. 3, 1971, pp. 379-388.

[25] Carpenter, C. J., 'Comparisons of Alternative Formulations of Three Dimensional Magnetic Field and Eddy Current Problems at Power Frequencies', *Proc. IEE*, Vol. 124, No. 11, 1977, pp. 1026-1034.

[26] Carpenter, C. J. and Wyatt, E. A., 'Efficiency of Numerical Techniques for Computing Eddy Currents in Two and Three Dimensions', Proc. of the Compumag Conference, Oxford, 1976, pp. 242-250.

[27] Carpenter, C. J. and Djurovic, M., 'Three Dimensional Numerical Solution of Eddy Currents in Thin Plates', *Proc. IEE*, Vol. 122, No. 6, 1975, pp. 681-688

[28] Carpenter, C. J., 'Numerical Solution of Magnetic Fields in the Vicinity of Current Carrying Conductors', *Proc. IEE*, Vol. 114, No. 11, 1967, pp. 1793-1800.

[29] Carpenter, C. J., 'Surface Integral Methods of Calculating Forces on Magnetized Iron Parts', *Proc. IEE*, Vol. 107, Part C, No. 11, 1960, pp. 19-28.

[30] Carpenter, C. J., 'Theory and Application of Magnetic Shells', *Proc. IEE*, Vol. 114, No. 7, 1967, pp. 995-1000.

[31] Carter, G. W., *The Electromagnetic Fields and its Engineering Aspects*, Longmans Green and Co., London, 1954.

[32] Cavendish, J. C., 'Automatic Triangulation of Arbitrary Planar Domains for the Finite Element Method', *Int. Jour. for Num. Methods in Engg.*, Vol. 8, 1974, pp. 679-697.

[33] Cendes, Z. J., 'Finite Element Methods in Electromagnetics', Lecture Notes, Dept. of Electrical Engineering, McGill University, 1981.

[34] Cendes, Z. J. and Silvester, P. P., 'FINPLT: A Finite Element Field Plotting Program', Dept. of Electrical Engineering, McGill University.

[35] Cendes, Z. J. and Shenton, D., 'Delaunay Triangulation on Three Dimensional Domains', Proc. Intermag Conference Minnesota, 1985.

[36] Chari, M. V. K., Csendes, Z. J., Silvester, P.,et al., 'Three Dimensional Magnetostatic Field Analysis of Electrical Machinery by the Finite Element Method', *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-100, 1981, pp. 4007-4019.

[37] Chari, M. V. K. and Silvester, P. P., eds. *Finite Elements in Electric and Magnetic Field Problems*, John Wiley, New York, 1980.

[38] Chari, M. V. K., 'Finite Element Analysis of Nonlinear Magnetic Fields in Electric Machines', Ph. D. Dissertation, McGill University, 1970.

[39] Chari, M. V. K., 'Finite Element Solution of the Eddy Current Problem in Magnetic Structures', *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-93, 1974, pp. 62-72.

[40] Christiansen, H. N. and Stephenson, M. B., 'MOVIE.BYU - A Computer Graphics Software System', *Jour. of the Technical Councils*, ASCE, Vol. 105, No. TC1, Proc. paper 14484, April 1979, pp. 3-12.

[41] Collins, R. J., 'Bandwidth Reduction by Automatic Renumbering', *Int. Jour. for Num. Methods in Engg.*, Vol. 6, 1973, pp. 345-356.

[42] Coulomb, J. L., 'Finite Elements 3-D Magnetic Field Computations', *IEEE Trans. on Magnetics*, Vol. MAG-17, 1981, pp. 3241-3246.

[43] Coulomb, J. L., Du Terrail and G. Meunier, 'Two Three Dimensional Parametrized Mesh Generators for the Magnetic Field Computations', *IEEE Trans. on Magnetics*, Vol. MAG-20, No. 5, September 1984, pp. 1900-1902.

[44] Cuthill, E. and McKee, J., 'Reducing the Bandwidth of Sparse Symmetric Matrices', ACM Nat. Conf., San Francisco, 1969, pp. 157-172.

[45] Damerdash, N. A. and Nehl T. W., 'Three Dimensional Finite Element Vector Potential Formulation of Magnetic Fields in Electrical Apparatus', *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-100, 1981, pp. 4105-4111.

[46] de Beer, A., Polak, S. J., Wachters, A. J. H. and van Welij, J. S., 'The Use of PADDY for the Solution of 3D Magnetostatic Problems', *IEEE Trans. on Magnetics*, Vol. MAG-18, No. 2, 1982, pp. 617-619.

[47] Duff, I. S., 'Recent Developments in the Solution of Large Sparse Linear Equations', in *Computing Methods in Applied Sciences and Engineering*, edited by R. Glowinski and J. L. Lions, North Holland Publishing Company, 1980.

[48] Duff, I. S. and Reid, J. K., 'The Multifrontal Solution of Indefinite Sparse Symmetric Linear Equations', *ACM Trans. on Mathematical Software*, Vol. 9, No. 3, September 1983, pp. 302-325.

[49] Duff, I. S., 'Design Features of a Frontal Code for Solving Sparse Unsymmetric Linear Systems Out of Core', *SIAM Jour. Sci. Stat. Comput.*, Vol. 5, No. 2, June 1984, pp. 270-280.

[50] Emson C. R. I. and Simkin, J., 'An optimal Method for 3D Eddy Currents', *IEEE Trans. on Magnetics*, Vol. MAG-19, No. 6, November 1983, pp. 2450-2452.

[51] Everstine, G. C., 'A Comparison of Three Resequencing Algorithms for the Reduction of Matrix Profile and Wavefront', *Int. Jour. for Num. Methods in Engg.*, Vol. 14, 1979, pp. 837-853.

[52] Fellipa, C. A., 'Solution of Linear Equations With Skyline Stored Symmetric Matrix', *Comp. and Structures*, Vol. 5, 1975, pp. 13-29.

[53] Fenves, S. J. and Law, H. K. 'A Two Step Approach to Finite Element Ordering' *Int. Jour. for Num. Methods in Engg.*, Vol. 19, 1983, pp. 891-911.

[54] Forghani, B., 'Mesh Generation and Equipotential Plotting in Three Dimensions', M. Eng. Thesis, Dept. of Electrical Engg., McGill University, 1981.

[55] Fredriksson, B., Mackkerle, J. and Persson, B. G., 'A Finite Element Programs in Integrated Software for Structural Mechanics and CAD', *Computer Aided Design*, Vol. 13, No. 1, 1981, pp. 27-39.

[56] Fried, I., 'A Gradient Computational Procedure for the Solution of Large Problems Arising from Finite Element Discretization Method', *Int. Jour. Num. Methods in Engg.*, Vol. 2, 1970, pp. 477-494.

[57] Garcia de Jalon, J. and Muguerza, R., 'On the Mini-Computer Solution of Large Systems of Linear Equations Arising from the Finite Element Method', *Appl. Math. Modelling*, Vol. 4, October 1980, pp. 381-388.

[58] George, A. and Liu, J., *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall Inc., Englewood Cliffs, N. J., 1981.

[59] George, A., 'Nested Dissection of a Regular Finite Element Mesh', *SIAM Jour. Num. Analysis*, Vol. 10, 1973, pp. 345-363.

[60] George, A., 'Computer Implementation of the Finite Element Method', Ph. D. Thesis, Stanford University, 1971.

[61] George, A., and Liu, J. W. H., 'An Automatic Nested Dissection Algorithm for Irregular Finite Element Problems', *SIAM J. Num. Analysis*, Vol. 15, 1978, pp. 1053-1069.

[62] George, A. and Liu, J. W. H., 'An Implementation of Pseudoperipheral Node Finder, '*ACM Trans. on Math. Software*, Vol. 5, 1979, pp. 284-295.

[63] Gibbs, N. E., Poole, W. G., and Stockmeyer, P. K., 'An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix', *SIAM Jour. Num. Anal.*, Vol. 13, April 1976, pp. 236-250.

[64] Gordon, W. J. and Hall, C. A., 'Construction of Curvilinear Coordinate Systems and Applications to Mesh Generation', *Int. Jour. for Num. Methods in Engg.*, Vol. 7, 1973, pp. 461-477.

[65] Grooms, H. R., 'Algorithm for Matrix Bandwidth Reduction', *Jour. of Structural Div.*, ASCE, Vol. 98, (ST1), 1972, pp. 203-214.

[66] Gustafsson, I., 'Stability and Rate of Convergence of Modified Incomplete Cholesky Factorization Methods', Research Report 79.02R, Department of Computer Sciences, Chalmers University of Technology, Goteborg, Sweden, 1979.

[67] Gustafsson, I., 'A Class of First Order Factorization Methods', BIT 18, 1978, pp. 142-156.

[68] Gustafsson, I., 'On First and Second Order Symmetric Factorization Methods For the Solution of Elliptic Difference Equations', Report 78.01R, Department of Computer Sciences, Chalmers University of Technology, Goteborg, Sweden, 1978.

[69] Haber, Robert, Shepherd, S. Mark and Abel, J. F. et al, 'A General Two Dimensional Graphical Finite Element Preprocessor Utilizing Discrete Transfinite Mappings', *Int. Jour. for Num. Methods in Engg.*, Vol. 17, 1981, pp. 1015-1044.

[70] Haber, Robert, 'Discrete Transfinite Mappings for the Description and Meshing of Three Dimensional Surfaces Using Interactive Computer Graphics, *Int. Jour. for Num. Methods in Engg.*, Vol. 18, 1982, pp. 41-666

[71] Hellen, T. K., 'A Front Solution for Finite Element Techniques', CEGB Report No. RD/B/N 1469, 1969.

[72] Herman, L. R., 'Laplacian-Isoparametric Grid Generation Scheme', *J. of Eng. Mech. Div.*, ASCE 102, 1976, pp. 749-756.

[73] Hestens, M. R. and Stiefel, E., 'Method of Conjugate Gradients for Solving Linear Systems', National Bureau of Standards, Report 1969, Washington, D. C., 1952.

[74] Hillyard, R., 'The Build Group of Solid Modellers', *IEEE Computer Graphics and Applications*, Vol. 2, No. 2, March 1982, pp. 43-52.

[75] Ida, N. and Lord, W., 'Solution of Linear Equations for Small Computer Systems', *Int. Jour. for Num. Methods in Engg.*, Vol. 20, 1984, pp. 625-641.

[76] Irons, B. M., 'A Frontal Solution Program for Finite Element Analysis', *Int. Jour. for Num. Methods in Engg.*, Vol. 2, 1970, pp. 5-32.

[77] Irons, B. M. and Ahemad, S., *Techniques of Finite Elements*, E. Horwood; Halsted Press, 1980.

[78] Iselin, Ch., 'A Scalar Integral Equation for Magnetostatic Fields', Proc. Compumag Conference, Oxford, 1976, pp. 15-18.

[79] Jacobs, D. A. H., 'Generalizations of the Conjugate Gradient Method for Solving Non-Symmetric and Complex System of Equations', CERL Laboratory Note No. RD/L/N/70/80, 1980.

[80] Jacobs, D. A. H., 'Preconditioned Conjugate Gradient Methods for Solving Systems of Algebraic Equations', CERL Report No. RD/L/N/193/80, 1981.

[81] Jennings, A. and Malik, G. M., 'The Solution of Sparse Linear Equations by the Conjugate Gradient Method', *Int. Jour. for Num. Methods in Engg.*, Vol. 12, 1978, pp. 141-158.

[82] Jennings, A., *Matrix Computations for Engineers and Scientists*, John Wiley and Sons, Chichester, 1977.

[83] Jennings, A., 'Influence of Eigenvalue Spectrum on the Convergence Rate of the Conjugate Gradient Method', *Jour. Inst. Math. Applics.*, 20, 1977, pp. 61-72.

[84] Jensen, H. G. and Parks, G. A., 'Efficient Solutions for Linear Matrix Equations', *Jour. of Structural Division*, ASCE, Vol. 96, No. ST1, Proc. Paper 7007, Jan. 1970, pp. 49-64.

[85] Kershaw, D. S., 'The Incomplete Cholesky Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations', Jour. of Comp. Physics, 26, 1978, pp. 43-65.

[86] King, I. P.,'An Automatic Reordering Scheme for Simultaneous Equations Derived from Network Systems,' *Int. Jour. for Num. Methods in Engg.*, Vol. 2, 1970, pp. 523-533.

[87] Kotiuga, P. R., and Silvester, P. P., 'Vector Potential Formulation for Three Dimensional Magnetostatics', *Jour. Applied Physics*, Vol. 54, No. 11, 1982, pp. 8399-8401.

[88] Lari, R. J. and Turner, L. R. ,'Survey of Eddy Current Programs', *IEEE Trans. on Magnetics*, Vol. MAG-19, No. 6, November 1983, pp. 2474-2477.

[89] Lavers, J. D., 'Finite Element Solution of Nonlinear Two Dimensional TE-Mode Eddy Current Problems', *IEEE Trans. on Magnetics*, Vol. MAG-19, No. 5, September 1983, pp. 2201-2203.

[90] Lee, L. H. and Clark J. A.,'Flow Visualization in Complex Turbulent Flows', Jour. of the Hydraulics Division, Proc. of ASCE, Vol. 106, No. HY2, 1980, pp. 247-268.

[91] Levy, R., 'Resequencing of the Structural Stiffness Matrix to Improve Computational Efficiency', *JPL Quarterly Tech. Rev.*, Vol. 1, 1971, pp. 61-65.

[92] Liu, Wai-Hung and Sherman, A. H., 'Comparative Analysis of the Cuthill- Mckee and Reverse Cuthill-McKee Algorithms for Sparse Matrices', *SIAM Jour. Num. Anal.*, Vol. 13, No. 2, April 1976, pp. 198-213.

[93] Liu, J. W. H., 'On Reducing the Profile of Sparse Symmetric Matrices', Ph. D. Thesis, Dept. of Computer Science, University of Waterloo, 1976.

[94] Lorensen, W., 'Grid Generation Tools for the Finite Element Analyst', *First Chautauqua on Finite Element Modelling*, edited by J. H. Conway, Schaffer Analysis Inc., 1980, pp. 119-136.

[95] Lowther, D. A., Silvester, P. P. and Freeman, E. M., 'The Use of Interactive Post-Processing in the Design of Electromagnetic Devices', *IEEE Trans. on Magnetics*, Vol. MAG-16, 1980, pp. 803-805.

[96] Lowther D. A. and Silvester, P. P., 'Computer Aided Design in Magnetics', Lecture Notes, Department of Electrical Engineering, McGill University, Montreal, 1982.

[97] Lowther, D. A., 'A Microprocessor Based Electromagnetic Field Analysis System', *IEEE Trans. on Magnetics*, Vol. MAG-18, No. 2, March 1982, pp. 351-356.

[98] Lowther, D. A. and Forghani, B., 'Interactive Post-Processing Technique for Electromagnetic Field Analysis', *IEEE Trans. on Magnetics*, Vol. MAG-19, September 1983, pp. 2168-2170.

[99] Lowther, D. A., Silvester, P. P. et al.; 'RUTHLESS-A General Purpose Finite Element Post-Processor', *IEEE Trans. on Magnetics*, Vol. MAG-17, 1981, pp. 3402-3404.

[100] MagNet-11, User Manual, Infolytica Corporation, Montreal, 1983.

[101] Meijerink, J. A. and Van der Vorst, H. A., 'An Iterative Solution Method for Linear Systems of which the Coefficient Matrix is a Symmetric M-matrix', *Math. of Comp.*, 31, 137, Jan. 1977, pp. 148-162.

[102] Meijerink, J. A. and Van der Vorst, H. A., 'Incomplete Decompositions as Preconditioning for the Conjugate Gradient Algorithm', in *Conjugate Gradient Methods and Similar Techniques*, edited by I. S. Duff, UKAEA Report No. R9636, 1979.

[103] Mantueffel, T. A., 'The Shifted Incomplete Cholesky Factorization', Report Sand 78-8226, Sandia Laboratory, 1978.

[104] Mantueffel, T. A., 'Solving Structures Problems Iteratively with a Shifted Incomplete Cholesky Preconditioning', in *Computing Methods in Applied Sciences and Engineering*, R. Glowinski, J. L. Lions, eds., North Holland Publishing Company, New York, 1980, pp. 427-434.

[105] McWhirter, J. H. and Duffin, R. J. et al 'Computational Methods for Solving Static Field and Eddy Current Problems via Fredholm Integral Equations', *IEEE Trans. on Magnetics*, Vol. MAG-15, pp. 1075-1084, 1979.

[106] Melosh, R. J. and Bamford, R. M., 'Efficient Solution of Load Deflection Equations', *Jour. of Structural Division*, ASCE, Vol. 97, No. ST2, Proc. Paper 7867, Feb. 1971, pp. 713-715.

[107] Meyer, C., 'Special Problems Related to Linear Equation Solvers', *Jour. of Structural Division*, ASCE, Vol. 101, No. ST4, April 1975, pp. 869-890.

[108] Meyer, C., 'Solution of Linear Equations - State of the Art', *Jour. of Structural Divisioh*, Vol. 99, No. ST7, July 1973, pp. 1507-1526.

[109] Mishra, M., 'Saturation Effects in Alternators', M. Tech. Thesis, Department of Electrical Engineering, Indian Institute of Technology Kharagpur, India, 1975.

[110] Mishra, M., 'A Preconditioned Conjugate Gradient Frontal Solver', M. Eng. Thesis, Dept. of Electrical Engineering, McGill University, 1981.

[111] Mishra, M., Forghani, B., Lowther, D. A. and Silvester, P. P., 'Solution of Three Dimensional Electromagnetic Field Problems on a Mini-Computer', *IEEE Trans. on Magnetics*, Vol. MAG-19, No. 6, November, 1983, pp. 2663-2666

[112] Mishra, M., Lowther, D. A. and Silvester, P. P., 'A Preconditioned Conjugate Gradient Frontal Solver for Three Dimensional Electromagnetic Field Problems', *IEEE Trans. on Magnetics*, Vol. MAG-20, No. 5, September 1984, pp. 1909-1911.

[113] Muller, W. and Wolff, W., 'Numerisch Berechnung Dreidimensionaler Magnet felder Fur Grosse Turbogeneratoren Bei Feldabhangiger Permeability Und Beliebiger Strondichte Verteilung', ETZ-A, 94, pp. 1973, pp. 276-282.

[114] Munksagard, N., 'Solving Sparse Symmetric Set of Linear Equations by Preconditioned Conjugate Gradients', *ACM Trans. on Mathematical Software*, Vol. 6, No. 2, June 1980, pp. 206-219.

[115] Moro, P. and Verite, J. C., 'Study of the Current Density on the Power Stations Output Conductors and their Sheaths', *Bulletin de la direction des etudes et recherches, Electricite de France*, No. 1, 1982, pp. 75-98.

[116] Nakamae, E. et al, 'Color Computer Graphics in Magnetic Field Analysis by Means of the Finite Element Method', *Computer and Graphics*, Vol. 7, No. 3-4, 1983, pp. 295-306.

[117] Nassif, N. and Silvester, P. P., 'Graphics Representation of Three-Component Vector Fields', *Computer Aided Design*, Vol. 12, No. 6, 1980, pp. 289-294.

[118] Newman, W. M. and Sproull R. F., *Principles of Interactive Computer Graphics*, McGraw Hill Book Company, 1979.

[119] Nguyen-Van-Phai, 'Automatic Mesh Generation with Tetrahedron Elements', *Int. Nat. Jour. for Num. Methods in Engg.*, Vol. 18, 1982, pp. 273-289.

[120] PE2D User Manual, Rutherford Appelton Laboratories., England, 1983.

[121] Perucchio, Renato et al, 'Interactive Computer Graphic Preprocessing for Three Dimensional Finite Element Analysis', *Int. Jour. for Num. Methods in Engg.*, Vol. 18, 1982, pp. 909-926.

[122] Pillsbury, R. D. 'A Three Dimensional Eddy Current Formulation Using Two Potentials-The Magnetic Vector Potential and Total Magnetic Scalar Potential', *IEEE Trans. on Magnetics*, Vol. MAG-19, No. 6, November 1983, pp. 2284-2287.

[123] Pina, H. L. G., 'An Algorithm for Frontwidth Reduction', *Int. Jour. for Num. Methods in Engg.*, Vol. 17, 1981, pp. 1539-1549.

[124] Pissanetzky, S., 'KUBIK: An Automatic Three Dimensional Finite Element Mesh Generator', *Int. Jour. for Num. Methods in Engg.*, Vol. 17, 1981, pp. 255-269.

[125] Polak, S. J., Watchers, A. J. H. and van Welij, J. S.,'A New 3D Eddy Current Model', *IEEE Trans. on Magnetics*, Vol. MAG-19, No. 6, November 1983, pp. 2447-2449.

[126] Preston, T. W. and Reece, A. B. J., 'Solution of Three Dimensional Eddy Current Problems: The $T - \Omega$ Method', *IEEE Trans. on Magnetics*, Vol. MAG-18, No. 2, March 1982, pp. 486-491.

[127] Razzaque, A., 'Automatic Reduction of Front-width for Finite Element Analysis', *Int. Jour. for Num. Methods in Engg.*, Vol. 15, 1980, pp. 889-910.

[128] Reid, J. K., 'On the Method of Conjugate Gradients for the Solution of Sparse System of Linear Equations', in *Large Sparse sets of Linear Equations*, edited by J. K. Reid, Academic Press, London, 1971.

[129] Requicha, A. A. G. and Voelcker, H. B., 'Solid Modelling', A Historical Summary and Contemporary Assessment', *IEEE Computer Graphics and Applications*, Vol. 2, No. 2, March 1982, pp. 43-52.

[130] Roger, D., 'Finite Element Method for Calculating Power Frequency Three Dimensional Electromagnetic Field Distributions', *Proc. IEE*, Vol. 130, Pt. A, No. 5, July 1983, pp. 233-238.

[131] Rose, D. J. and Willoughby, R. A., *Sparse Matrices and Their ApplicationD*, Plenum Press, New York, 1972.

[132] Rose, D. J., 'A Graph Theoretic Study of The Numerical Solution of Sparse Symmetric Positive Definite Systems of Linear Equations', in *Graph Theory and Computing*, edited by R. C. Read, Academic Press, New York, 1972, pp. 183-217.

[133] Rosen, R., 'Matrix Bandwidth Minimization', Proceedings, National Conference, ACM, Publication P-68, Brandon Systems Press, Princeton, 1968, pp. 585 -595.

[134] Sabonnadiere, J. C., Meunier, G. and Morel, B., 'FLUX: A General Purpose Finite Elements Package for Two Dimensional Electromagnetic Fields', *IEEE Trans. on Magnetics*, Vol. MAG-18, No. 2, March 1982, pp. 624-626.

[135] Sadek, E. A., 'A Scheme for the Automatic Generation of Triangular Finite Elements', *Int. Jour. for Num. Methods in Engg.*, Vol. 15, 1980, pp. 1813-1822.

[136] Schneider, J. M., Chaudhury, K., and Salon, S., 'The Use of Interactive Computer Graphics in Electromagnetic Problems', *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-102, No. 1, Jan. 1983, pp. 91-95.

[137] Silvester, P., 'High Order Polynomial Triangular Finite Elements for Potential Problems', *Int. Jour. Engg. Sci.*, 7, 1969, pp. 849-861.

[138] Silvester, P. P., 'Analysis of Turbogenerator Terminal Boxes', CAD Lab Report, Department of Electrical Engineering, McGill Univbresity, April 1980.

[139] Silvester, P., 'Tetrahedral Polynomial Finite Elements for the Helmholtz Equation', *Int. Jour. for Num. Methods in Engg.*, Vol. 4, 1972, pp. 405-413.

[140] Silvester, P. P. and Ferrari, R. L., 'Finite Elements for Electrical Engineers', Cambridge University Press, 1983.

[141] Silvester, P. P., Auda, H. A. and Stone, G. D., 'A Memory-Economic Frontwidth Reduction Algorithm', *Int. Jour. for Num. Methods in Engg.*, Vol. 20, 1984, pp. 733-743.

[142] Silvester, P., 'Universal Finite Elements Matrices for Tetrahedra', *Int. Jour. for Num. Methods in Engg.*, Vol. 12, 1978, pp. 237-344.

[143] Silvester, P. P., 'Interactive Computer Aided Design in Magnetics', *IEEE Trans. on Magnetics*, Vol. MAG-17, No. 6, November, 1981, pp. 3388-3392.

[144] Silvester, P., 'A General High Order Finite Element Waveguide Analysis Program ', *IEEE Trans. on Microwave Theory and Techniques*, Vol. MTT-17, 1969, pp. 489-495.

[145] Simkin , J. and Trowbridge, C. W., 'Which Potential ?, A Comparison of the Various Scalar and Vector Potentials for the Numerical Solution of Non-linear Poisson Problem', Rutherford Appleton Laboratories., Oxford, Report RL-78-009/13, 1978.

[146] Simkin, J. and Trowbridge, C. W., '3D-Nonlinear Electromagnetic Field Computations Using Scalar Potentials', *Proc. IEE*, Vol. 127B, 1980, pp. 368-374.

[147] Simkin, J. and Trowbridge, C. W., 'Magnetostatic Fields Computed Using an Integral Equation Derived From Green's Theorems', Proc. Compumag Conference, Oxford, 1976, pp. 5-14.

[148] Simkin, J.,'A Comparison of Integral and Differential Equation Solutions for Field Problems', *IEEE Trans. on Magnetics*, Vol. MAG-18, No. 2, March 1982, pp. 401-405.

[149] Simkin, J. and Trowbridge, C. W., 'On the Use of Total Scalar Potential in the Numerical Solution of Field Problems in Electromagnetics', *Int. Jour. of Num. Methods in Engg.*, Vol. 14, 1979, pp. 423-440.

[150] Sloan, S. W. and Randolph, M. F., 'Automatic Reordering for Finite Element Analysis with Frontal Solution Schemes', *Int. Jour. for Num. Methods in Engg.*, Vol. 19, 1983, pp. 1153-1181.

[151] Stakgold, I., *Boundary Value Problems of Mathematical Physics*, McMillan, New York, 1968.

[152] Stewart, G. W., *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[153] Stoll, R. L., *The Analysis of Eddy Currents*, Clarendon Press, 1974.

[154] Strang, G. and Fix, G., *An Analysis of Finite Element Method*, Prentice Hall Inc., Englewood Cliffs, N. J., 1973.

[155] Stratton, J. A., *Electromagnetic Theory*, McGraw Hill, New York, 1941.

[156] Tewarson, R. P., *Sparse Matrices*, Academic Press, New York, 1976.

[157] Thompson, E., and Shimazaki, Y., 'A Frontal Procedure Using Skyline Storage', *Int. Jour. for Num. Methods in Engg.*, Vol. 15, 1980, pp.889-910

[158] Tortschanoff, T., 'Survey of Numerical Methods in Field Calculations', *IEEE Trans. on Magnetics*, Vol. MAG-20, No. 5, September 1984, pp. 1912-1917.

[159] TOSCA-3D, User Guide, Rutherford Appleton Laboratory, UK, May 1982.

[160] Tozani, O. V., *Mathematical Models for the Evaluation of Electric and Magnetic Fields*, Iliffe Books Ltd., London, 1968.

[161] Tracy, F. T., 'Graphics Pre- and Post-Processor for Two-Dimensional Finite Element Programs', SIGGRAPH 77, Vol. 11, No. 2,1977, pp. 8-12.

[162] Trowbridge, C. W., 'Three Dimensional Field Computation', *IEEE Trans. on Magnetics*, Vol. MAG-18, 1981, pp. 293-297.

[163] Trowbridge, C. W., 'Application of Integral Equation Methods for the Numerical Solutions of Magnetostatic and Eddy Current Problems', in *Finite Elements in Electric and Magnetic Field Problems*, eds. M. V. K. Chari and P. P. Silvester, John Wiley and Sons, New York, pp. 191-214.

[164] Tuff, A. D. and Jennings, A., 'An Iterative Method for Large Systems of Linear Structural Equations', *Int. Jour. Num. Methods in Engg.* , Vol. 7 1973, pp. 175-183.

[165] Van der Vorst H. A., 'A Vectorizable Variant of Some ICCG Methods', *SIAM J. Sci. Stat. Comp.*, Vol. 3, No. 3, Sept. 1982, pp. 251-257.

[166] Varga, R. S., *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs, 1962.

[167] Verite, J. C., 'Computation of Eddy Currents on the Alternator Output Conductors by a Finite Element Method', *Electrical Power and Energy Systems*, Vol. 1, No. 3, October, 1979, pp. 193-198.

[168] Wilkinson, J. H. and Reinsch, R., *Hand Book for Automatic Computations*, Vol. 2, 'Linear Algebra', Springer, Berlin, 1971.

[169] Wilkinson, J. H., *The Algebraic Eigenvalue Problem*, Clarendon Press, 1965.

[170] Wilson, E. L., Bathe, K. J. and Doherty, W. P., 'Direct Solution of Large Systems of Linear Equations', *Computers and Structures*, Vol. 4, 1974, pp. 363-372.

[171] Wolff, W. and Muller, W., 'General Numerical Solution of the Magnetostatic Equations', *Wiss Ber AEG-TELEFUNKEN 49*, 3, pp. 77-86, 1979.

[172] Wolff, W. 'Three Dimensional Eddy Current Calculations', Proc. Compumag Conference, Oxford, 1976, pp. 231-240.

[173] Wong, Y. S., 'Iterative Method for Problems in Numerical Analysis', D. Phil. Thesis, University of Oxford, 1978.

[174] Yamashita, H. et al, 'Stereographic Display on Three Dimensional Magnetic Fields of Electromagnetic Machines', *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-100, No. 11, November 1981, pp. 4962-4967.

[175] Young, D., *Iterative Solution of Large Sparse Systems*, Academic Press, New York, 1971.

[176] Zienkiewicz, O. C., *The Finite Element Method*, McGraw Hill, 1979.

[177] Zienkiewicz, O. C., Lyness, J. and Owens, D. R. J., 'Three Dimnesional Magnetic Field Determination using a Scalar Potential - A Finite Element Solution', *IEEE Trans. on Magnetics*, Vol. MAG-13, 1977, pp. 1649-1656.

[178] Zienkiewicz, O. C., 'The Basic Concepts and an Application to 3-D Magnetostatics Problems', in *Finite Elements in Electrical and Magnetic Field Problems*, M. V. K. Chari and P. P. Silvester eds. John Wiley, New York, 1980, pp. 11-32.

[179] Zienkiewicz, O. C. and Phillips, D. V., 'An Automatic Mesh Generation Scheme for Plane and Curves Surface by Isoparametric Coordinates', *Int. Jour. Num. Meth. Engg.*, Vol. 3, 1971, pp. 519-528.

[180] Zlatev Zaheri et al., 'Comparison of Two Algorithms for Solving Large Linear Systems', SIAM J. Sci. Stat. Comp. Vol. 3, No. 4, December 1982, pp. 486-501.

Appendix I

For a symmetric positive definite matrix A, the bandwidth for $i-th$ row can be defined as

$$\beta_i = i - f_i(A) \tag{A.1}$$

where $f_i(A)$ is the leftmost nonzero element in row i and can be given as

$$f_i(A) = min\{j : a_{ij} \neq 0\} \tag{A.2}$$

Thus the maximum bandwidth $\beta_{max}(i)$ for the matrix A can be defined as

$$\beta_{max}(i) = max\{\beta_i(A) : 1 \leq i \leq N\} \tag{A.3}$$

The profile of the matrix A is the region from the leftmost nonzero element in any row to the main diagonal, i.e.

$$Profile(A) = \sum_{i=1}^{N} \beta_i(A) \tag{A.4}$$

Similarly, the frontwidth for the $i-th$ row is defined as the number of rows in the profile of A which intersect column i

$$fw_i(A) = \{|j : j > i \ and \ a_{jl} \neq 0 \ for \ l \leq i|\} \tag{A.5}$$

and then the maximum frontwidth or wavefront can be defined as

$$fw_{max} = max\{fw_i(A) : 1 \leq i \leq N\} \tag{A.6}$$

The root mean square (rms) frontwidth for the coefficient matrix A is defined as

$$fw_{rms} = \sqrt{\{\frac{(\sum_{i=1}^{N} fw_i(A))}{N}\}} \tag{A.7}$$

The fill-in which occurs in the triangular factors L and U when a Gaussian elimination is applied to the matrix A is defined as the number of nonzeros created in L or U in locations where A had zeros. Formally

$$Fill(A) = \{\{i,j\} : a_{ij} = 0, (L+U)_{ij} \neq 0\} \tag{A.8}$$

Appendix II

The second order function to be plotted inside the triangle and the expression for simplex coordinate ς_1 are defined as:

$$u(x,y) = Ax^2 + By^2 + Cxy + Dx + Ey + F \qquad (B.1)$$

$$\begin{aligned}\varsigma_1 = \frac{1}{\Delta}\{&x_1(y_4z_3 - y_3z_4 + y_2z_4 - z_2y_4 - y_2z_3 + y_3z_2)\\ &+ y_1(x_3z_4 - x_4z_3 - x_2z_4 + x_4z_2 + x_2z_3 - x_3z_2)\\ &+ z_1(x_2y_4 - x_4y_2 - x_3y_4 + x_4y_3 - x_2y_3 + x_3y_2)\\ &+ x_2(y_3z_4 - y_4z_3) - y_2(x_3z_4 - x_4z_3) + z_2(x_3y_4 - x_4y_3)\}\end{aligned} \qquad (B.2)$$

where Δ is the determinant of the denominator (equation 6.7.2.1.11). Similarly, expressions for ς_2, ς_3 and ς_4 are derived and are listed below:

$$\begin{aligned}\varsigma_2 = \frac{1}{\Delta}\{&x(y_3z_4 - y_4z_3) - y(x_3z_4 - x_4z_3) + z(x_3y_4 - x_4y_3)\\ &- x_1(y_3z_4 - y_4z_3) + y_1(x_3z_4 - x_4z_3) - z_1(x_3y_4 - x_4y_3)\\ &+ x_1(yz_4 - zy_4) - y_1(xz_4 - zx_4) + z_1(xy_4 - yx_4)\\ &- x_1(yz_3 - zy_3) + y_1(xz_3 - zx_3) - z_1(xy_4 - x_4y)\}\end{aligned} \qquad (B.3)$$

$$\begin{aligned}\varsigma_3 = \frac{1}{\Delta}[&\{x_1(y_2z_3 - y_3z_2) - y_1(x_2z_4 - x_4z_3) + z_1(x_2y_4 - x_4y_2)\}\\ &+ \{(x_2z_4 - z_2x_4 - x_1z_4 + x_4z_1 + x_1z_2 - x_2z_1)\}y\\ &+ \{(-y_2z_4 + z_2y_4 + y_1z_4 - z_1y_4 - y_1z_2 + y_2z_1)\}x\\ &+ \{(-x_2y_4 + y_2x_4 + x_1y_4 - y_1x_4 + y_1x_2 - x_1y_2)z\}]\end{aligned} \qquad (B.4)$$

and

$$\begin{aligned}\varsigma_4 = \frac{1}{\Delta}[&\{-x_1(y_2z_3 - y_3z_2) + y_1(x_2z_3 - x_3z_2) - z_1(x_2y_3 - x_3y2)\}\\ &+ x(y_2z3 - z2y_3 - y_1z_3 + y_3z_1 + y_1z_2 - z_1y_2)\\ &+ y(-x_2z_3 + z_2x_3 + x_1z_3 - z_1x_3 - x_1z_2 + x_2z_1)\\ &+ z(x_2y3 - x_3y_2 - x_1y_3 + x_3y_1 + x_1y_2 - x_2y_1)\}]\end{aligned} \qquad (B.5)$$

The procedure to determine the expressions for the constants A, B, C, D, E, and F has already been described in Chapter VI. In this appendix, the expressions for these

constants in terms of the coordinates of the vertices and the potential values of the nodes are given. Let the constants be defined as

$$A_1 = y_3z_4 - y_4z_3 \qquad F_1 = y_3z_4 + y_4z_3 \qquad A_2 = y_2z_3 - y_3z_2$$
$$F_2 = y_2z_3 + y_3z_2 \qquad A_3 = y_2z_4 - y_4z_2 \qquad F_3 = y_2z_4 + y_4z_2$$
$$A_4 = y_1z_4 - y_4z_1 \qquad F_4 = y_1z_4 + y_4z_1 \qquad A_5 = y_1z_2 - y_2z_1$$
$$F_5 = y_1z_2 + y_2z_1 \qquad A_6 = y_1z_3 - y_3z_2 \qquad F_6 = y_1z_3 + y_3z_1$$

$$B_1 = x_3 - x_2 \qquad B_2 = x_4 - x_3 \qquad B_3 = x_4 - x_1$$
$$B_4 = x_2 - x_4 \qquad B_5 = x_1 - x_3 \qquad B_6 = x_2 - x_1$$

$$D_1 = z_4A_1 \qquad D_2 = z_3A_1 \qquad D_3 = z_2A_1 \qquad D_4 = z_4A_3$$
$$D_5 = z_3A_3 \qquad D_6 = z_2A_3 \qquad D_7 = z_4A_2 \qquad D_8 = z_3A_2$$
$$D_9 = z_2A_4 \qquad D_{10} = -z_1A_2 \qquad D_{11} = -z_1A_3 \qquad D_{12} = -z_1A_2$$
$$D_{13} = z_4A_4 \qquad D_{14} = z_4A_6 \qquad D_{15} = z_3A_4 \qquad D_{16} = z_3A_6$$
$$D_{17} = z_2A_4 \qquad D_{18} = z_2A_6 \qquad D_{19} = -z_1A_4 \qquad D_{20} = -z_1A_6$$
$$D_{21} = z_4A_5 \qquad D_{22} = z_3A_5 \qquad D_{23} = -z_1A_5 \qquad D_{24} = -z_2A_5$$

$$H_1 = x_2B_1 \qquad H_2 = x_2B_4 \qquad H_3 = x_2B_2 \qquad H_4 = -x_3B_1$$
$$H_5 = -x_3B_4 \qquad H_6 = -x_3B_2 \qquad H_7 = x_4B_1 \qquad H_8 = x_4B_4$$
$$H_9 = x_4B_2 \qquad H_{10} = x_2B_5 \qquad H_{11} = x_2B_3 \qquad H_{12} = -x_3B_5$$
$$H_{13} = -x_3B_3 \qquad H_{14} = x_4B_5 \qquad H_{15} = x_4B_3 \qquad H_{16} = -x_1B_1$$
$$H_{17} = -x_1B_4 \qquad H_{18} = -x_1B_2 \qquad H_{19} = -x_1B_5 \qquad H_{20} = -x_1B_3$$
$$H_{21} = x_3B_6 \qquad H_{22} = -x_4B_6 \qquad H_{23} = x_1B_6 \qquad H_{24} = -x_2B_6$$

$$G_1 = x_2 + x_3 \qquad G_2 = x_2 + x_4 \qquad G_3 = x_3 + x_4$$
$$G_4 = x_1 + x_2 \qquad G_5 = x_1 + x_4 \qquad G_6 = x_1 + x_3$$

$$\Delta = A_1B_6 + A_2B_3 + A_3B_5 + A_4B_1 + A_5B_2 + A_6B_4$$

$$\begin{aligned}A = 2.0[&u_1\{A_1(A_1 - 2A_3) + A_3(A_3 - 2A_2) + A_2(A_2 + 2A_1)\}\\&+ 2(-A_1 + A_4 - A_6)(u_2\{A_1 - A_3 + A_2) + u_4(A_3 - A_4 + A_5)\}\\&+ u_5(A_3(A_3 - 2A_4) + A_4(A_4 - 2A_5) + A_5(A_5 + 2A_3)\}\\&+ 2(A_1 + A_2 - A_3)\{(u_6(A_3 - A_4 + A_5) + u_7(-A_2 + A_6 \quad A_5) + u_8(A_2 - 2A_6)\\&+ A_6(A_6 - 2A_5) + A_5(A_5 + 2A_2)\} + 2(-A_2 + A_6 - A_5)\{(u_9(A_3 - A_4 + A_5)\\&- u_{10}(A_1 - A_4 + A_6) + u_3\{(A_1(A_1 - 2A_4) + A_4(A_4 - 2A_6) + A_6(A_6 + 2A_1)\}\}]/\Delta^2\end{aligned}$$

$$B = BB_1 + BB_2$$

$$\begin{aligned}BB_1 = 2.0[&u_1\{z_4B_1(z_4B_1 + 2z_3B_4) + z_3B_4(z_3B_4 + 2z_2B_2)\\&+ z_2B_2(z_2B_2 + 2z_4B_1)\} + 2(z_4B_5 + z_3B_3 - z_1B_2)\{u_2(z_4B_1 + z_3B_4 + z_2B_2)\\&- u_4(-z_4B_6 + z_2B_3 + z_1B_4)\} + u_3\{z_4B_5(z_4B_5 + 2z_3B_3)\\&+ z_3B_3(z_3B_3 - 2z_1B_2) + z_1B_2(z_1B_2 - 2z_4B_5)\}\\&+ u_5\{z_4B_6(z_4B_6 - 2z_2B_3) + z_2B_3(z_2B_3 + 2z_1B_4)\\&+ z_1B_4(z_1B_4 - 2z_4B_6)\}]/\Delta^2\end{aligned}$$

$$\begin{aligned}BB_2 = 2.0[&2z_4B_1 + z_3B_4 + z_2B_2)\{u_6(z_4B_6 - z_2B_3 - z_1B_4)\\&- u_7(z_3B_6 + z_2B_5 + z_1B_1)\} + u_8\{z_3B_6(z_3B_6 + 2z_2B_5)\\&+ z_2B_5(z_2B_5 + 2z_1B_1) + z_1B_1(z_1B_1 + 2z_3B_6)\}\\&+ 2(z_3B_6 + z_2B_5 + z_1B_1)\{u_9(-z_4B_6 + z_2B_3 + z_1B_4) - u_{10}(z_4B_5 + z_3B_3 - z_1B_2)\}]/\Delta^2\end{aligned}$$

$$C = CC_1 + CC_2$$

$$\begin{aligned} CC_1 = 4.0[&(-D_1 + D_4 - D_7)(u_1B_1 + u_2B_5 + u_6B_6) \\ &+ (-D_2 + D_5 - D_8)(u_1B_4 + u_2B_3 - u_7B_6) \\ &+ (D_3 - D_6 + D_9)(-u_1B_2 + u_6B_3 + u_7B_5) \\ &+ (D_1D_{13} + D_{14})(u_2B_1 + u_3B_5 + u_4B_6) \\ &+ (D_2 - D_{15} + D_{16})(u_2B_4 + u_3B_3 - u_{10}B_6)]/\Delta^2 \end{aligned}$$

$$\begin{aligned} CC_2 = &[(D_{10} - D_{19} + D_{20})(u_3B_2 + u_4B_4 + u_{10}B_1) \\ &+ (-D_4 + D_{13} - D_{21})(u_4B_5 + u_5B_6 + u_6B_1) \\ &+ (-D_{11} + D_{19} - D_{23})(u_4B_2 + u_5B_4 + u_9B_1) \\ &+ (D_6 - D_{17} - D_{24})(u_5B_3 - u_6B_2 + u_9B_5) \\ &+ (D_8 - D_{16} + D_{22})(u_7B_4 - u_8B_6 + u_{10}B_3) \\ &+ (-D_9 + D_{18} + D_{24})(-u_7B_2 + u_8B_5 + u_9B_3) \\ &+ (D_{12} - D_{20} + D_{23})(u_8B_1 + u_9B_4 + u_{10}B_2) \\ &+ (D_7 - D_{14} + D_{21})(u_7B_1 + u_9B_6 + u_{10}B_5) \\ &+ (D_3 - D_{17} + D_{18}(u_2B_2 - u4B3 - u_{10}B_5) \\ &+ (-D_5 + D_{15} - D_{22})(u_6B_4 - u_9B_6 + u_4B_3) \\ &+ (-D_{10} + D_{11} - D_{12})(u_2B_2 + u_6B_4 + u_7B_1)]/\Delta^2 \end{aligned}$$

$$D = DD_1 + DD_2 + DD_3$$

$$\begin{aligned} DD_1 = 4.0[&u_1\{A_1(-x_2A_1 + G_1A_3) + A_3(-x_3(A_3 + G_3A_2) - A_2(x_4A_2 + G_2A_1)\} \\ &+ (G_4A_1 - G_6A_3 + G_5A_2)(u_2A_1 - u_6A_3 + u_7A_2) + (G_1A_1 - 2.0x_3A_3 + G_3A_2) \\ &(-u_2A_4 + u_7A_5) + (G_2A_1 - G_3A_3 + 2.0x_4A_2)(u_2A_6 - u_6A_5) + u_3\{A_1(-x_1A_1 + G_6A_4) + \\ &A_4(-x_3A_4 + G_3A_6) - (A_6(x_4A_6 + G_5A_1)\} + (2.0x_1A_3 - G_4A_4 + G_5A_5)(u_4A_1 + u_9A_2)]/\Delta \end{aligned}$$

$$
\begin{aligned}
DD_2 = {} & [G_6A_3 - G_1A_4 + G_3A_5)(-u_4A_4 + u_9A_5) \\
& + (u_4A_6(G_5A_3 - G_2A_4 + 2.0x_4A_5) - u_5(A_3(x_1A_3(x_1A_3 - G_4A_4) \\
& + A_4(x_2A_4 - G_2A_5) + A_5(x_4A_5 + G_5A_3)) \\
& + (2.0x_2A_1 - G_1A_3 + G_2A_2)(u_6A_4 - u_7A_6) \\
& + u_8(A_2(-x_1A_2 + G_4A_6) + A_6(-x_2A_6 + G_1A_5) \\
& - A_5(x_3A_5 + G_6A_2)) + u_9(A_6(-G_4A_3 + 2.0x_2A_4 - G_2A_5))]/\Delta
\end{aligned}
$$

$$
\begin{aligned}
DD_3 = {} & [u_{10}(A_1(-2.0x_1A_2 + G_4A_6 - G_6A_5) + A_4(G_6A_2) \\
& - G_1A_6 + 2.0x_3A_5) + A_6(-G_5A_2 + G_2A_6 - G_3A_5))]/\Delta^2 \\
& + [u_1(A_1 + A_2 - A_3) + u_3(-A_1 + A_4 - A_6) \\
& + u_5(A_3 - A_4 + A_5) + u_8(-A_2 + A_6 - A_5)]/\Delta
\end{aligned}
$$

$$E = EE_1 + EE_2$$

$$
\begin{aligned}
EE_1 = {} & 4.0[u_7(D_2H_{24} - D_3H_{10} + H_1(D_{10} + D_{14}) \\
& + D_5H_{21} - D_6H_{12} + H4(D_{11} + D_{21}) + D_8(H_{22} + H_{17}) \\
& + D_9(-H_{14} + H_{18}) + D_{12}H_7 + D_7H_{16} + D_{16}H_2 \\
& + D_{22}H_5 + D_{18}H_3 - D_{24}H_6) + u_8(H_{23}D_8 \\
& + H_{24}D_{16} + H_{21}D_{22} - H_{19}D_9 - H_{10}D_{18} + H_{12}D_{24} \\
& + H_{16}D_{12} + H_1D_{20} + H_4D_{23}) + u_9(-H_{23}(D_7 + D_5) \\
& - H_{24}(D_{14} + D_{15}) - H_{21}D_{21} + H_{19}D_6 - H_{16}D_{11} - H_{20} \\
& + D_9 - H_{11}D_{18} + D_{24}(H_{13} + H_{14}) + H_{10}D_{17} - H_1D_{19}]/\Delta
\end{aligned}
$$

$$
\begin{aligned}
EE_2 = [&(H_{17}D_{12} + H_2D_{20} + D_{23}(H_5 + H_7) + H_{22}D_{22}) \\
&+ u_{10}(D_2H_{23} + H_{19}(-D_3 + D_7) + D_{10}H_{16} + D_{14}H_{10} \\
&+ H_{12}(D_{21} + D_{17}) + D_8H_{20} + D_{22}H_{13} + D_{16}(H_{11} \\
&- H_{22}) - D_{19}H_4 + D_{18}H_{14} + D_{20}(-H_7 + H_3) \\
&+ D_{12}H_{18} + D_{23}H_6 - D_{15}H_{21}))/\Delta^2 \\
&+ (-u_1(z4B_1 + z_3B4 + z_2B_2) - u_3(z4B_5 \\
&+ z_3B_3 - z_1B_2) + u_5(-z4B_6 + z_2b_3 + z_1B_4) \\
&+ u_8(z_3B_6 + z_2B_5 + z_1B_1)]/\Delta
\end{aligned}
$$

$$
\begin{aligned}
F = 2.0[&u_1(x_2A_1(x_2A_1 - 2.0x_3A_3) + x_3A_3(x_3A_3 - 2.0x_4A_2) \\
&+ x_4A_2(x_4A_2 + 2.0x_2A_1)) + 2.0(-x_2A_1 + x_3A_3 \\
&- x_4A_2)(x_1(u_2A_1 - u_6A_3 + u_7A_2) \\
&+ x_3A_3 - x_4A_2)(x_1(u_2A_1 - u_6A_3 + u_7A_2) \\
&+ x_3(-u_2A_4 + u_7A_5) + x_4(u_2A_6 - u_6A_5) \\
&+ x_2(u_6A_4 - u_7A_6) + \Delta u_1/4.0)] \\
&+ [u_3(x_1A_1)) + 2.0x_4A_6) + x_4A_6(x_4A_6 \\
&+ 2.0x_1A_1)) + 2.0(x_1A_1 - x_3A_4 + x_4A_6) \\
&- u_4(x_1A_3 + u_4x_2A_4 - u_4x_4A_5 + \Delta u_3/4.0)] \\
&+ [u_5(x_1A_3(x_1A_3 - 2.0x_2A_4) + x_2A_4 \\
&+ (x_2A_4 - 2.0x_4A_5) + x_4A_5(x_4A_5 + 2.0x_1A_3)) \\
&+ u_8(x_1A_2(x_1A_2 - 2.0x_2A_6) \\
&+ x_2A_6(x_2A_6 - 2.0x_3A_5) + x_3A_5(x_3A_5 \\
&+ 2.0x_1A_2)) + 2.0(x_1A_2 - x_2A_6 \\
&+ x_3A_5)(u_9(-x_1A_3 + x_2A_4 - x_4A_5) + u_{10}(x_1A_1 - x_3A_4 \\
&+ x_4A_6) + u_8/4.0))]/\Delta^2[-u_5(x_1A_3A_4 + x_4A_5)]/\Delta
\end{aligned}
$$