NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Finite Element Method Applied to Modelling Electromagnetic Interference and Compatibility Problems On Printed Circuit Boards

by

Raouf Lawrence Khan

A Thesis submitted to the
School of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute for
Electrical Engineering

Department of Electrical Engineering
Faculty of Engineering
University of Ottawa

ISBN 0-315-59998-7

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

I hereby declare that I am the sole author of this document. I authorize the University of Ottawa to lend this document to other individuals or institutions for the purposes of scholarly research.

I further authorize the University of Ottawa to reproduce this document by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

ABSTRACT

The finite element method is applied to compute the distributed capacitance and inductance matrices for multilayered dielectric, planar transmission media. The analysis involves the solution of the Laplace equation in two dimensions for multiconductor configurations in microstrip, triplate, or TEM Cell configurations of arbitrary cross section and utilizes a unique node allocation algorithm for automatic generation of the finite element mesh. The advantages, disadvantages, and difficulties encountered in application of the FEM for this problem are outlined and discussed. Several programs, based on multiconductor transmission line theory, are developed for application to Electromagnetic Interference and Compatibility (EMI/C) problems for crosstalk and radiated emission analysis. To verify the model experimentally, a simple printed circuit board consisting of five parallel conductors in microstrip configuration was constructed and measured in both time domain and frequency domains. Radiation model results are compared for a multiconductor configuration with numerical results obtained using the Numerical Electromagnetics Code (NEC) moment method program.

ACKNOWLEDGEMENTS

The author would like to acknowledge the support and guidance provided by Dr. G. I. Costache during my graduate studies at the University of Ottawa.

The author would also like to thank the assistance of G. Hartsgrove, S. Symons, B. Carraro, and M. Master for their assistance in use of the computer and laboratory facilities and to NSERC for their financial support in making this work possible.

In addition I would like to acknowledge Dr. G.I. Costache, Dr. M. Ney, and Dr. S.S. Stuchly for their lectures which developed my awareness and interest in the area of Electromagnetic Engineering.

Table of Contents

1 INTRODUCTION 1
1.1 Motivation. 1
1.2 EMI on Printed Circuit Boards. 1
1.3 Computer Modelling and Verification. 3

2 GENERAL THEORY 5
2.1 The Printed Circuit Board. 5
2.2 The Distributed Parameter Transmission Line Model. 6
2.3 Electrostatics and the Scalar Electric Potential. 7

3 FINITE ELEMENT METHOD APPLIED TO CALCULATE THE DISTRIBUTED PARAMETERS 10
3.1 Variational Approach. 10
3.2 Finite Elements. 11
3.2.1 First-Order Triangular Finite Elements. 13
3.2.2 Second Order Triangular Finite Elements. 15
3.2.3 Infinite Elements. 17
3.3 Assembling the Final System of Equations. 20
3.3.1 The Finite Element Energy Functional. 21
3.3.2 Minimization of the Energy Functional. 23
3.3.3 Finite Element Laplacian Matrices. 28
3.4 Capacitance/Inductance Matrix Computation. 37
3.5 Automatic Mesh Generation. 40
3.6 Finite Element Results. 48
3.7 Finite Element Method - Conclusions. 54

4 CROSSTALK ANALYSIS 57
4.1 Crosstalk on Printed Circuit Boards. 57
4.2 The Multiconductor Transmission Line Equations. 58
4.3 Solution of the MCTLE's. 61
4.4 Program Verification. 64
4.5 Experimental Results. 69
4.6 Crosstalk Conclusions. 78

5 RADIATED EMISSION ANALYSIS 79
5.1 Radiation from Printed Circuit Boards. 79
5.2 Radiation Model. 80
5.3 Radiation Results. 86
5.4 Radiation Model Conclusions. 95

6 SUMMARY AND CONCLUSIONS 96
6.1 Summary. 96
6.2 Discussion and Future Work. 96
6.3 Conclusions. 98

7 REFERENCES 102

Appendix A: Triangle Cotangent Identity 104

Appendix B: Computer Programs 108
Appendix B1: "MCTLE.FOR" 108
Appendix B2: "TDSTLE.FOR" 118
Appendix B3: "FEM.FOR" 126
Appendix B4: "A3CON.FOR" 150
Appendix B5: "ZTLE.FOR" 153
Appendix B6: "NEC INPUT FILE" 158

Table of Figures

Figure 2-1: Tracks on a Printed Circuit Board 8
Figure 3-1: Finite Elements 12
Figure 3-2: First/Second Order Finite/Infinite Elements 14
Figure 3-3: Algthm. for Assbl. the Final System of Equations 27
Figure 3-4: Triangle Coordinate System 31
Figure 3-5: Finite Element Division (Single Cond. Config.) 41
Figure 3-6: Finite Element Division (Multi Cond. Config.) 42
Figure 3-7: Creation of an Equilateral Triangle 46
Figure 3-8: Finite Element Method Convergence 49
Figure 3-9: Multi. Cond. Config. Used For Comparison 52
Figure 4-1: Dist. Parameter Trans. Line Model 59
Figure 4-2: Two Parallel Conductors Above Ground 66
Figure 4-3: Experimental Measurement Set-up 70
Figure 4-4: Frequency Domain Results - Near End Xtalk 72
Figure 4-5: Frequency Domain Results - Far End Xtalk 73
Figure 4-6: Time Domain Results - Excitation Conductor 75
Figure 4-7: Time Domain Results - Near End Xtalk 76
Figure 4-8: Time Domain Results - Far End Xtalk 77
Figure 5-1: Radiation Model 82
Figure 5-2: Multi Cond. Config. Used for Radiation Comp. 87
Figure 5-3a: Rad. Results - Variation With Angle |E| 90
Figure 5-3b: Rad. Results - Variation With Angle |Ex| 91
Figure 5-3c: Rad. Results - Variation With Angle |Ey| 92
Figure 5-3d: Rad. Results - Variation With Angle |Ez| 93
Figure 5-4: Rad. Results - Variation with Frequency 94
Figure A-1: Triangle Cotangent Identity 105

Table of Tables

Table 3-1: Convergence Data. 50
Table 3-2: Per Unit Length Parameters - Multi. Cond. Config. 53
Table 4-1: Frequency Domain Results - Two Conductor Problem 68
Table 4-2: Per-Unit Length Parameters - PCB 71
Table 5-1: Per-Unit Length Parameters - Radiation Example 88

1 INTRODUCTION

1.1 Motivation.

The fundamental rationale for the development of computer models and algorithms for the prediction of electromagnetic phenomena on printed circuit boards (PCB's) is to enable high speed digital circuits and similar microwave integrated circuits to be designed and marketed as quickly and as cost effectively as possible, such that the electronic system is electromagnetically compatible with its intended operating environment. Electromagnetic compatibility (EMC) of an electronic system refers to the systems ability to function properly and reliably within its operating environment such that the system itself is not an electromagnetic source which can cause interference with other electronic circuits and systems.

If EMC Computer Aided Design (CAD) tools are applied in the circuit design and layout stages, the EMC performance of the electronic system can be evaluated, and corrective changes made before the construction and testing stages. This serves to reduce the time required to take a product from its initial design stages to manufacturing and marketing by avoiding the costly delays and the added engineering costs which are often required when, during the testing stage, EMI/C problems are traditionally uncovered. The added costs are often the result of the engineering time and effort required to rework the design and layout. Thus, what is needed are the development of models and algorithms which can be used to predict EMC performance in the design and layout stages so corrective actions can be taken.

1.2 EMI on Printed Circuit Boards.

Two fundamental electromagnetic problems associated with electronic circuits on PCB's are, electromagnetic interference (EMI) due to crosstalk phenomena (introduced by cross-coupling) and radiation produced by the boards. Cross-coupling refers to the electromagnetic coupling of energy from one conductor to another. The voltage and current distributions associated with the signals on a conductor generate electric and magnetic fields which interact with other nearby conductors. It is this interaction of the electric fields (capacitive coupling) and magnetic fields (inductive coupling) which induce undesired voltages and currents on other conductors. These unwanted signals, termed crosstalk signals, can lead to degradation in circuit performance and system malfunctions. Electromagnetic Interference problems occurring in the commercial and residential environment has forced the Federal Communications Commission (FCC) and other regulatory authorities worldwide, to impose legislation which places restrictions on the electromagnetic emission levels radiated from electronic equipment [1], [2],[20]. This legislation acts to control the electromagnetic environment by placing restrictions on the radiated emission levels from equipment over the rf and communication frequency range 30 MHz to 1000 MHz. As a result the circuit board design engineer is faced with the task of minimizing both on and off-board electromagnetic interference.

With increasing clock and data rates and their associated faster rise/fall times, the higher frequency spectral content of these signals is increasing dramatically. In addition, the increase in circuit complexity has resulted in an increase in the number and density of components and interconnections, and has reduced the separations between signal paths. This has lead to an increase in cross-coupling effects and has increased the spectrum of the crosstalk signal. In time domain this transforms into higher amplitude of the time domain crosstalk waveform.

These effects have also lead to an increase in the amplitude and spectral content of the radiated electromagnetic fields. As a result the design and layout of printed circuits boards has become a crucial factor in control via minimization of crosstalk and radiation levels.

1.3 Computer Modelling and Verification.

Many Computer Aided Design (CAD) circuit design, routing, and simulation packages are available and the inclusion of crosstalk analysis is possible. Several models which have been investigated for implementation in CAD programs [15], require knowledge of the per-unit length transmission line parameters. The work presented in this thesis begins by applying the finite element method to calculate these parameters for various multiconductor configurations. A major task in application of the finite element method to electromagnetic problems involving complex geometries in inhomogeneous media is the enormous amount of user input and user input time required to generate the finite element mesh. The automatic mesh generation algorithm developed overcomes many of the difficulties encountered in generating the finite element mesh and is applicable to a wide variety of multiconductor configurations. After the per-unit length parameters are computed by the finite element method, the voltage and current distributions on the conductors are calculated through solution of the multiconductor transmission line equations. These equations are easily solved by modal analysis [14], and enforcement of the line boundary conditions - termination networks - in the frequency domain from which time domain analysis is performed by using a Fast Fourier Transform (FFT) algorithm. Finally, a simple radiation model based upon antenna theory is developed to compute the radiated electric field from the boards.

Several FORTRAN programs have been written for this analysis. These include a finite element analysis program, a frequency/time domain crosstalk analysis program, and a frequency domain radiation program. The source code and algorithms are presented, as well as a discussion of their use, capabilities, and present limitations. A simple multiconductor printed circuit board was constructed and both frequency and time domain measurements are compared with predicted results. Radiation model predictions are compared with those obtained using the Numerical Electromagnetics Code (NEC) package which uses the method of moments approach in modelling antenna problems.

2 GENERAL THEORY

2.1 The Printed Circuit Board.

Typical printed circuit boards (PCB's) fall into the category of quasi-planar technology and provide cost-effective, lightweight, durable means for circuit construction. They consist of rectangular conductors etched on a dielectric substrate. The conductors are commonly made out of copper and are of standard thickness of $35.56\,\mu m$ (1.4 mils), also referred to as one ounce copper, or $71.12\,\mu m$ (2.8 mils), also referred to as two ounce copper. The dielectric substrate is commonly glass-epoxy with a relative dielectric constant between 4.5 and 5.0 and thickness $1.5748\,mm$ (62.0 mils). The standard unit of measurement for PCB's is the "mil" (1 mil = 1/1000" = $25.4\,\mu m$). PCB's with tracks etched on only one side of the substrate are referred to as single sided boards while those with tracks etched on both sides are referred to as double sided boards. Interconnections on double sided boards are made by plated through holes called vias. Other commonly used configurations are the microstrip and triplate or stripline configurations. Microstrip configuration consists of tracks etched on one side of the substrate with a solid ground plane on the other side. Triplate configuration consists of the tracks located on a buried layer between two solid ground planes which are located on the top and bottom of the substrate.

These tracks on a PCB form the interconnections between components which serve as a medium to transmit electrical signals from one area to another. At higher frequencies these interconnections no longer act like simple short circuits and propagation and field effects must be considered. The electromagnetic fields, associated with these transmission media, are in general hybrid in nature and therefore have components of the electric and magnetic fields in

both the transverse and longitudinal directions. However, for the frequencies of interest to EMI/C engineers, which are below 1000 MHz, a quasi-static approximation can be used and the longitudinal components of the fields can be neglected. This allows the PCB structure to be replaced by a transmission line model in which propagation depends on both the PCB geometry and substrate properties.

2.2 The Distributed Parameter Transmission Line Model.

The distributed parameter transmission line model considers the mode of propagation along the structure to be a tranverse electromagnetic (TEM) wave. Although this is not entirely accurate when considering inhomogeneous media, it does provide a good approximation for PCB geometries for frequencies up to 1000 MHz. This approach models a system of conductors with a per-unit length circuit model. The conductor cross sectional area and conductivity determines the per-unit length series resistance. At low frequencies this parameter is inversely proportional to the conductor cross sectional area while at higher frequencies it increases due to skin effect. The conductor width, thickness, and separation determine the per-unit length inductance, while the separation with respect to ground and the permittivity of the dielectric determine the per-unit length capacitance. If real dielectric are considered the leakage current through the dielectric is represented by a shunt conductance. This allows propagation to be modelled using a per-unit length impedance $\hat{Z} = R + j\omega L$ and a shunt admittance $\hat{Y} = G + j\omega C$. At radio frequencies, the inductance reactance is much greater than the series resistance and the capacitive susceptance is much greater than the shunt conductance. The resulting model, neglecting the series resistance and shunt conductance, is referred to as the distributed parameter transmission line model for lossless lines. For multiconductor configurations each conductor is modelled with the per-unit length circuit model which is modified to include the additional

mutual capacitances and inductances between conductors. This is the model which will be used and is discussed in detail in Section 4. These per-unit length transmission line parameters can be calculated from the electrostatic cross-sectional fields associated with the PCB structure.

2.3 Electrostatics and the Scalar Electric Potential.

A typical PCB conductor configuration is illustrated in Figure 2-1. The structure is assumed infinitely long and homogeneous in the $\hat{z}$ direction. The problem is to evaluate the electrostatic field $\vec{E}(x,y)$ in the region, $\Re$, surrounding the conductors. The region $\Re$ is the region of space where the electrostaic field is non-zero. For a system of conductors above an infinite ground plane, the region $\Re$ is the entire region above the ground plane and outside the conductors. In this analysis, the conductors are assumed perfectly conducting ($\sigma = \infty$), and the region, $\Re$, surrounding the conductors characterized by permeability $\mu = \mu_o$ and lossless, charge free dielectric subregions with constant permittivity $\varepsilon(x,y) = \varepsilon_r(x,y)\varepsilon_o$.

Under these conditions, Maxwells equations for the static case ($\frac{\partial}{\partial t} = 0$) governing the electrostatic field solution are given by:

$$\nabla \times \vec{E} = 0 \tag{2.3.1a}$$

$$\nabla \cdot \vec{D} = 0 \tag{2.3.1b}$$

along with the constitutive relation:

$$\vec{D} = \varepsilon \vec{E} \tag{2.3.2}$$

where $\vec{D}(x,y)$ denotes the electric flux density.

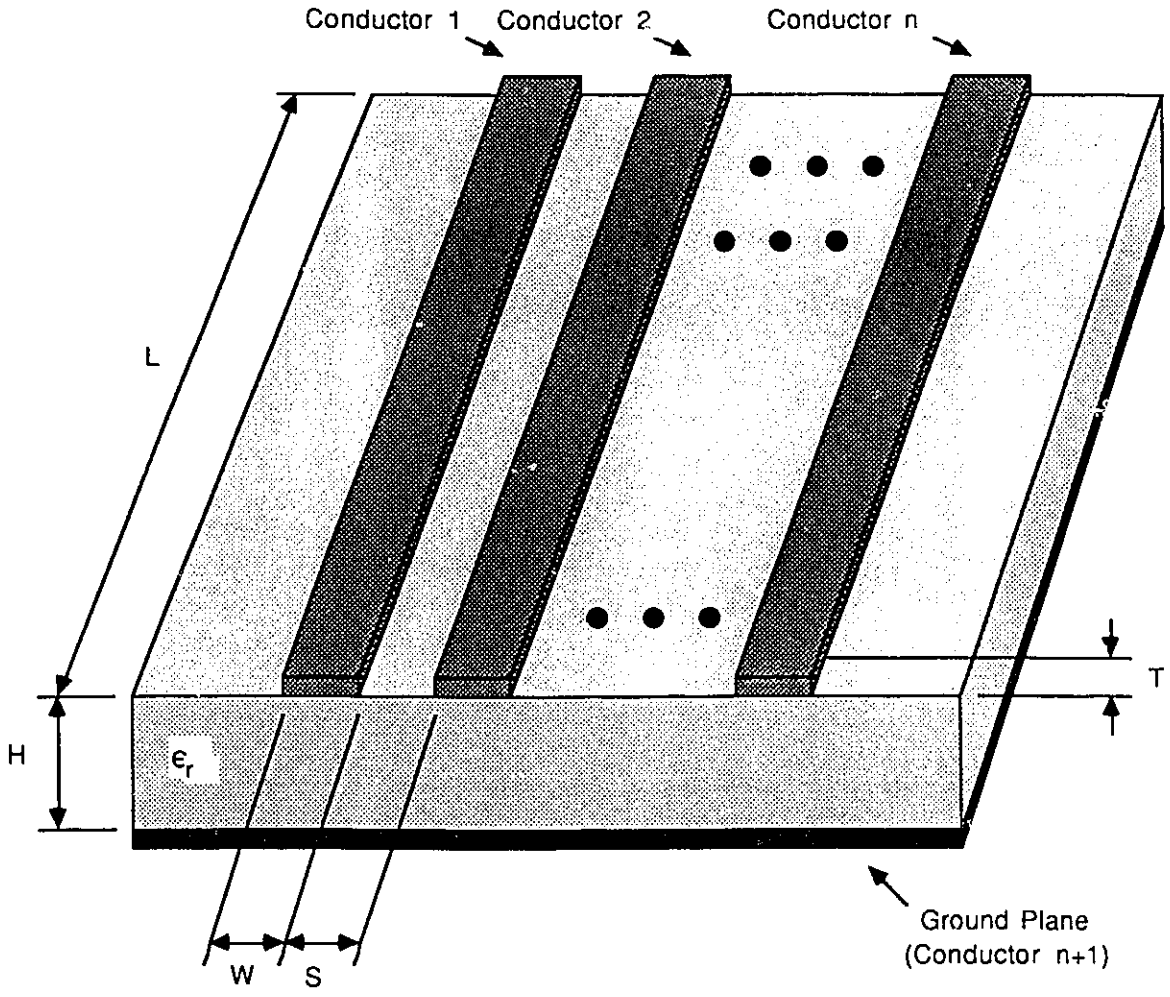


Figure 2-1: Tracks on a Printed Circuit Board

Since the electic field is conservative, equation (2.3.1a) can be expressed in terms of a scalar potential function $\Phi(x, y)$, as:

$$\vec{E} = -\nabla\Phi \tag{2.3.3}$$

Substitution of equation (2.3.2) followed by (2.3.3) in equation (2.3.1b), yields Laplace's equation:

$$\nabla^2\Phi = 0 \tag{2.3.4}$$

which governs the potential distribution throughout the region surrounding the conductors. In this region, Φ must satisfy Laplace's equation and any associated Dirichlet boundary conditions on electric walls and any homogeneous Neumann boundary conditions on magnetic walls. The Dirichlet boundary conditions are of the form $\Phi = \Phi_o$, where Φ_o is a specified constant on the equipotential conductor surfaces.

There exist many methods of solving Laplace's equation. These include analytical methods such as Conformal Mapping, relaxation methods such as the Finite Difference Method, integral equation methods such as the Method of Moments and the Boundary Element Method, and variational methods such as the Finite Element Method. The variational approach has been selected because of it is simplicity and accuracy in estimating the total energy.

3 FINITE ELEMENT METHOD APPLIED TO CALCULATE THE DISTRIBUTED PARAMETERS

3.1 Variational Approach.

In the variational approach applied to potential problems, the solution is found through determination of a potential function, satisfying the boundary conditions, which minimizes a certain integral. For electrostatic problems, which are to be solved to calculate the capacitance matrix for a system of conductors, the integral to be minimized is proportional to the energy stored in the electrostatic field. Let $u(x,y)$ represent all possible solutions satisfying the boundary conditions, and $U(x,y)=\Phi(x,y)$ the correct solution satisfying the boundary conditions and Laplace's equation. The correct solution, $U(x,y)=\Phi(x,y)$, is unique from all the other solutions satisfying the boundary conditions which do not satisfy Laplace's equation in that this solution minimizes the total energy stored in the electrostatic field. Thus instead of solving Laplace's equation:

$$\frac{\partial^2\Phi(x,y)}{\partial x^2}+\frac{\partial^2\Phi(x,y)}{\partial y^2}=0 \tag{3.1.1}$$

- the partial differential equation describing the two dimensional potential distribution - directly, the solution is being sought by minimizing the functional:

$$F(u)=\frac{1}{2}\int_{\Re}\varepsilon(x,y)(\nabla u(x,y))^2 dx\,dy \tag{3.1.2}$$

which is proportional to and has the dimension of system energy. Thus the solution $U(x, y)$, satisfying the boundary conditions, is the solution for $\Phi(x, y)$ if $F(u)$ is minimized and also will satisfy Laplace's equation.

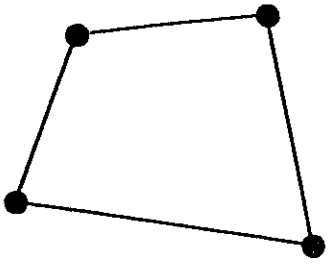
In the Finite Element Method, the studied region is divided into finite elements. Within each finite element, the potential distribution is assumed to vary as a function of position according to suitably chosen trial functions. These trial functions are often chosen as a linear combination of polynomials with unknown coefficients and minimization of the functional, $F(u)$, and enforcement of the boundary conditions determines these coefficients and an approximate solution for Φ [3].

3.2 Finite Elements.

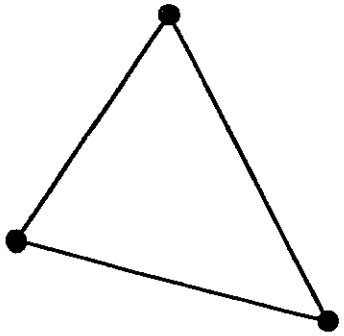
In the finite element approach an electromagnetic problem is solved by dividing the entire studied region into a connected set of arbitrary shapes, known as finite elements [3],[6]. Several types of finite elements are illustrated in Figure 3-1. For any finite element, a potential function is defined which varies over the finite element as a function of position. In this manner a piecewise approximation for the potential distribution throughout the entire studied region can be sought.

One of the most commonly used type of finite element in two dimensional finite element analysis is the triangular finite element because of its suitability in conforming to complicated surfaces and boundaries. The simplest triangular finite elements use first, second, or higher order polynomial functions to approximate the potential distribution over the finite element. These are the finite elements selected for use for evaluating the potential distribution surrounding the system of conductors and are described in further detail in the preceding sections.

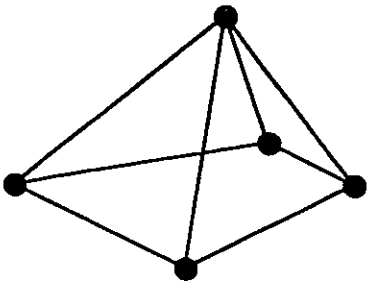
1-D Linear Finite Element



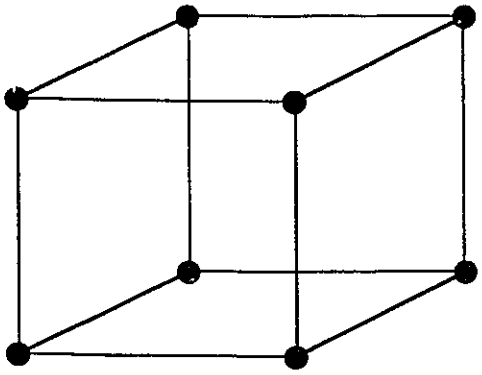
2-D Patch Finite Element



2-D Triangular Finite Element



3-D Tetrahedral Finite Element



3-D Rectangular Finite Element

Figure 3-1: Finite Elements.

3.2.1 First-Order Triangular Finite Elements.

Consider a triangular finite element occupying a portion of two dimensional space in the 'xy' plane. A first order variation for the potential distribution over the finite element requires [3]:

$$\Phi^{(e)}(x,y) = c_0 + c_1 x + c_2 y \tag{3.2.1}$$

where the superscript (e) denotes the potential function associated with a particular finite element. Figure 3-2a depicts one such finite element with three nodes, labelled n_1, n_2, and n_3, which are located at the three vertices which define the triangle. Each node, located at coordinates (x_{n_1}, y_{n_1}), (x_{n_2}, y_{n_2}), and (x_{n_3}, y_{n_3}), respectively is assigned a corresponding node potential. The corresponding node potentials are labelled Φ_{n_1}, Φ_{n_2}, and Φ_{n_3}, respectively. By evaluating equation (3.2.1) at the three nodes of the triangle, one obtains a linear system of equations:

$$\begin{aligned} \Phi_{n_1} &= c_0 + c_1 x_{n_1} + c_2 y_{n_1} \\ \Phi_{n_2} &= c_0 + c_1 x_{n_2} + c_2 y_{n_2} \\ \Phi_{n_3} &= c_0 + c_1 x_{n_3} + c_2 y_{n_3} \end{aligned} \tag{3.2.2}$$

which can be solved for the three constants c_0, c_1, and c_2. Substitution of the expressions for these constants back in equation (3.2.1) and rearranging terms, yields the expression for the first order variation of the potential distribution over the finite element in terms of the node potentials:

$$\Phi^{(e)}(x,y) = \alpha_{n_1}(x,y)\Phi_{n_1} + \alpha_{n_2}(x,y)\Phi_{n_2} + \alpha_{n_3}(x,y)\Phi_{n_3} \tag{3.2.3}$$

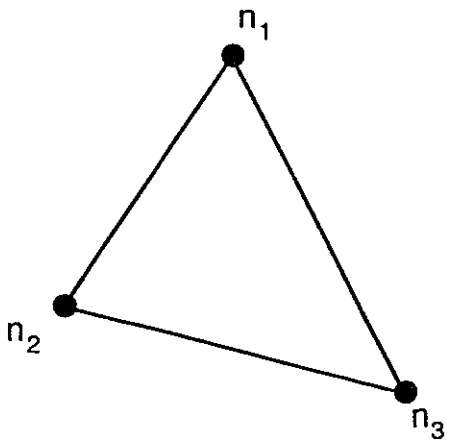


Figure 3-2a: First-Order Finite Element

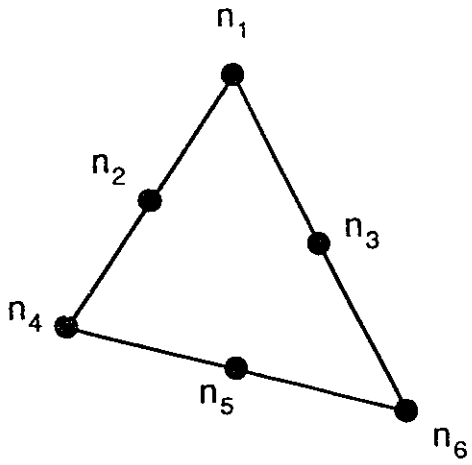


Figure 3-2b: Second Order Finite Element

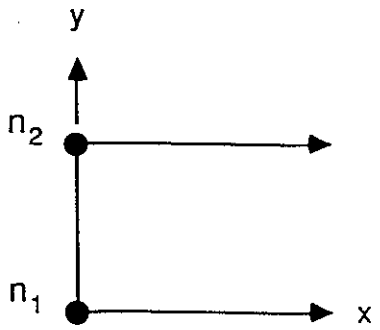


Figure 3-2c: First-Order Infinite Element

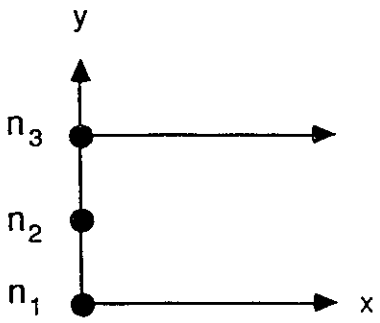


Figure 3-2d: Second-Order Infinite Element

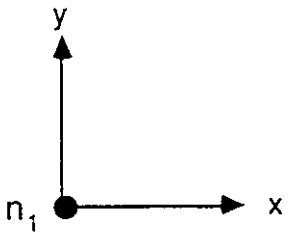


Figure 3-2e: Corner Infinite Element

Figure 3-2: Finite/Infinite Elements.

where $\alpha_{n_1}(x,y)$, $\alpha_{n_2}(x,y)$, and $\alpha_{n_3}(x,y)$ are interpolary shape functions dependant only on position within the finite element, and are given by:

$$\alpha_{n_1}(x,y)=\frac{1}{2A}\{(x_{n_2}y_{n_3}-x_{n_3}y_{n_2})+(y_{n_2}-y_{n_3})x+(x_{n_3}-x_{n_2})y\}$$

$$\alpha_{n_2}(x,y)=\frac{1}{2A}\{(x_{n_3}y_{n_1}-x_{n_1}y_{n_3})+(y_{n_3}-y_{n_1})x+(x_{n_1}-x_{n_3})y\} \qquad (3.2.4)$$

$$\alpha_{n_3}(x,y)=\frac{1}{2A}\{(x_{n_1}y_{n_2}-x_{n_2}y_{n_1})+(y_{n_1}-y_{n_2})x+(x_{n_2}-x_{n_1})y\}$$

where $|\,2A\,|$ denotes twice the area of the triangle given by:

$$2A=x_{n_b}y_{n_c}-x_{n_c}y_{n_b}+x_{n_c}y_{n_a}-x_{n_a}y_{n_c}+x_{n_a}y_{n_b}-x_{n_b}y_{n_a} \qquad (3.2.5)$$

and $n_a = n_1$, $n_b = n_2$, and $n_c = n_3$. This allows for the unknown node potentials to be used as variational parameters instead of the constants to specify continuity between finite elements.

3.2.2 Second Order Triangular Finite Elements.

A second order triangular finite element is pictured in Figure 3-2b. The general expression for the second order variation of the potential distribution over a finite element requires:

$$\Phi^{(e)}(x,y)=c_0+c_1x+c_2y+c_3xy+c_4x^2y+c_5xy^2 \qquad (3.2.6)$$

To recast this expression, consisting of the six constants c_0, c_1, c_2, c_3, c_4, and c_5, three additional nodes are defined. This second order finite element consists of $k = 6$ nodes, labelled n_1, n_2, n_3, n_4, n_5, and n_6, which are located at coordinates (x_{n_1}, y_{n_1}), $((x_{n_1}+x_{n_4})/2, (y_{n_1}+y_{n_4})/2)$,

$((x_{n_1}+x_{n_6})/2, (y_{n_1}+y_{n_6})/2)$, (x_{n_4}, y_{n_4}), $((x_{n_4}+y_{n_6})/2, (y_{n_4}+y_{n_6})/2)$, and (x_{n_6}, y_{n_6}), respectively. The corresponding node potentials are labelled Φ_1, Φ_2, Φ_3, Φ_4, Φ_5, and Φ_6, respectively. By evaluating equation (3.2.6) at the six nodes of the triangle and solving the resulting system of six equations for the constants c_0, c_1, c_2, c_3, c_4, and c_5, equation (3.2.6) can be rewritten in terms of the node potentials as:

$$\Phi^{(e)}(x,y) = \alpha_{n_1}(x,y)\Phi_{n_1} + \alpha_{n_2}(x,y)\Phi_{n_2} + \alpha_{n_3}(x,y)\Phi_{n_3} + \alpha_{n_4}(x,y)\Phi_{n_4} + \alpha_{n_5}(x,y)\Phi_{n_5} + \alpha_{n_6}(x,y)\Phi_{n_6} \tag{3.2.7}$$

where the interpolary shape functions $\alpha_{n_1}(x,y)$, $\alpha_{n_2}(x,y)$, $\alpha_{n_3}(x,y)$, $\alpha_{n_4}(x,y)$, $\alpha_{n_5}(x,y)$, and $\alpha_{n_6}(x,y)$ are given by:

$$\alpha_{n_1}(x,y)=\frac{1}{2A^2}\{(x_{n_4}y_{n_6}-x_{n_6}y_{n_4})+(y_{n_4}-y_{n_6})x+(x_{n_6}-x_{n_4})y\}^2$$

$$-\frac{1}{2A}\{(x_{n_4}y_{n_6}-x_{n_6}y_{n_4})+(y_{n_4}-y_{n_6})x+(x_{n_6}-x_{n_4})y\}$$

$$\alpha_{n_2}(x,y)=\frac{1}{A^2}\{(x_{n_4}y_{n_6}-x_{n_6}y_{n_4})+(y_{n_4}-y_{n_6})x+(x_{n_6}-x_{n_4})y\}$$

$$\{(x_{n_6}y_{n_1}-x_{n_1}y_{n_6})+(y_{n_6}-y_{n_1})x+(x_{n_1}-x_{n_6})y\}$$

$$\alpha_{n_3}(x,y)=\frac{1}{A^2}\{(x_{n_4}y_{n_6}-x_{n_6}y_{n_4})+(y_{n_4}-y_{n_6})x+(x_{n_6}-x_{n_4})y\}$$

$$\{(x_{n_1}y_{n_4}-x_{n_4}y_{n_1})+(y_{n_1}-y_{n_4})x+(x_{n_4}-x_{n_1})y\} \tag{3.2.8}$$

$$\alpha_{n_4}(x,y)=\frac{1}{2A^2}\{(x_{n_6}y_{n_1}-x_{n_1}y_{n_6})+(y_{n_6}-y_{n_1})x+(x_{n_1}-x_{n_6})y\}^2$$

$$-\frac{1}{2A}\{(x_{n_6}y_{n_1}-x_{n_1}y_{n_6})+(y_{n_6}-y_{n_1})x+(x_{n_1}-x_{n_6})y\}$$

$$\alpha_{n_5}(x,y)=\frac{1}{A^2}\{(x_{n_6}y_{n_1}-x_{n_1}y_{n_6})+(y_{n_6}-y_{n_1})x+(x_{n_1}-x_{n_6})y\}$$

$$\{(x_{n_1}y_{n_4}-x_{n_4}y_{n_1})+(y_{n_1}-y_{n_4})x+(x_{n_4}-x_{n_1})y\}$$

$$\alpha_{n_6}(x,y)=\frac{1}{2A^2}\{(x_{n_1}y_{n_4}-x_{n_4}y_{n_1})+(y_{n_1}-y_{n_4})x+(x_{n_4}-x_{n_1})y\}^2$$

$$-\frac{1}{2A}\{(x_{n_1}y_{n_4}-x_{n_4}y_{n_1})+(y_{n_1}-y_{n_4})x+(x_{n_4}-x_{n_1})y\}$$

where $2A$ is given by (3.2.5) with $n_a = n_1$, $n_b = n_4$, and $n_c = n_6$.

3.2.3 Infinite Elements.

For application to solving unbounded problems, one is faced with the task of subdividing an infinite space with finite elements unless some artificial boundary condition can be enforced or approximated. Although good results can often be obtained simply

through truncation of the finite element mesh far from the studied region (i.e. the system of conductors), the accuracy and rate of convergence can often be increased through the use of infinite elements [5], if suitably chosen trial functions are used to approximate the potential distribution outwards from the finite element region. In this approach the region of interest is enclosed in a closed region whose perimeter defines the finite element region boundary.

In solving Laplace's equation in the region surrounding a typical multiconductor configuration, the potential will decay with distance away from the system. The manner in which it will decay will depend on the geometry and electrical properties of system and surrounding medium. A simple infinite element is pictured in Figure 3-2c. This first order infinite element consists of $k = 2$ nodes. These two nodes, denoted n_1 and n_2, are located at coordinates (x_o, y_{n_1}) and (x_o, y_{n_2}) respectively. Φ_{n_1} and Φ_{n_2} are the associated node potentials. The second order infinite element (Figure 3-2d) consists of $k = 3$ nodes, denoted n_1, n_2, and n_3, which are located at coordinates (x_o, y_{n_1}), $(x_o, (y_{n_1} + y_{n_3})/2)$, and (x_o, y_{n_3}) respectively. Here Φ_{n_1}, Φ_{n_2}, and Φ_{n_3} represent the associated node potentials. If the potential distribution is chosen to vary as $1/x^n$ in the $\hat{x}$ direction, then the potential distribution over the infinite element can be written as:

$$\Phi^{(e)}(x,y) = \frac{1}{(y_{n_2} - y_{n_1})} \{(y_{n_2} - y)\Phi_{n_1} + (y - y_{n_1})\Phi_{n_2}\} \{\frac{x_o}{x}\}^n \qquad (3.2.9)$$

for the first order infinite element, and as:

$$\phi^{(e)}(x,y) = \{\frac{1}{(y_{n_2}-y_{n_1})^2}\{y_{n_2}(y_{n_1}+y_{n_2})-(y_{n_1}+3y_{n_2})y+2y^2\}\Phi_{n_1}$$

$$-\{y_{n_1}y_{n_2}-(y_{n_1}+y_{n_2})y+y^2\}\Phi_{n_2} \qquad (3.2.10)$$

$$+\{y_{n_1}(y_{n_1}+y_{n_2})-(3y_{n_1}+y_{n_2})y+2y^2\}\Phi_{n_3}\}\{\frac{x_o}{x}\}^n$$

for the second order infinite element. Although the expressions given in (3.2.9),(3.2.10) were derived for a $1/x^n$ variation in the $\hat{x}$ direction:

$$y_{n_1} \le y \le y_{n_2}$$

$$x_o \le |x| \le +\infty \qquad (3.2.11)$$

these expressions may be applied to an infinite element with a $1/y^n$ variation in the $\hat{y}$ direction:

$$x_{n_1} \le x \le x_{n_2}$$

$$y_o \le |y| \le +\infty \qquad (3.2.12)$$

by interchanging the 'x' and 'y' variables:

$$x_o = y_o$$

$$y_{n_1} = x_{n_1} \qquad (3.2.13)$$

$$y_{n_2} = x_{n_2}$$

If these infinite elements are used to approximate the potential distribution outwards from the finite element region boundary, it is necessary to define corner infinite elements (Figure 3-2e) at the corner of the finite element mesh boundary where the $1/x^n$ and $1/y^n$

infinite elements meet. To ensure continuity of the potential distribution along the infinite element boundary, the corner infinite elements require a $1/x^n$ decay in the $\hat{x}$ direction and a $1/y^n$ decay in the $\hat{y}$ direction. This requires the trial function for this corner infinite element to be defined by:

$$\Phi^{(e)}(x,y) = \{\frac{x_o}{x}\}^n \{\frac{y_o}{y}\}^n \Phi_o \tag{3.2.14}$$

where:

$$\begin{aligned} x_o \leq x < +\infty \quad ; x_o > 0 \\ \\ y_o \leq y < +\infty \quad ; y_o > 0 \end{aligned} \tag{3.2.15}$$

and Φ_o represents the corner node potential. The infinite elements derived in this section require the finite element region to be rectangular in shape and oriented parallel to the xy coordinate system. If this is not desirable, similar infinite elements with a $1/r^n$ decay can also be derived [5].

3.3 Assembling the Final System of Equations.

Consider a studied region, $\Re$, divided into finite elements. Let *NOFE* represent the total number of finite elements in the finite element mesh and *NPTL* represent the total number of node potentials which are numbered $1, 2, \ldots, NOFE$ and $1, 2, \ldots, NPTL$ respectively. Since each node is assigned a node potential, the total number of nodes is equal to the total number of node potentials. Any node may either be a free node or a fixed node. A free node is a node whose potential is unknown, whereas a fixed node is a node at a specified fixed potential, such as a

node lying on the surface of a conductor. The latter are referred to as Dirichlet nodes whose nodes are at fixed potentials and are forced boundary conditions. If $NDIR$ represents the total number of fixed nodes, then $NN = NPTL - NDIR$ represents the total number of free nodes with unknown potentials.

3.3.1 The Finite Element Energy Functional.

With the studied region, $\Re$, divided into finite elements, the potential distribution is defined everywhere throughout this region as a piecewise continuous function and, for any point $P(x, y) \in \Re$, can be evaluated from:

$$\Phi(x, y) = \Phi^{(e)}(x, y) \tag{3.3.1a}$$

where the superscript (e) denotes the finite element occupying the (x, y) coordinate in the region $\Re$. Within this finite element, consisting of a total of k nodes, the potential at any point is defined in terms of the associated node potentials and interpolatary shape functions as:

$$\Phi^{(e)}(x, y) = \{\alpha_{n_1}\Phi_{n_1} + \alpha_{n_2}\Phi_{n_2} + \ldots + \alpha_{n_k}\Phi_{n_k}\} \tag{3.3.1b}$$

where $\alpha_{n_i}(x, y)$ and Φ_{n_i} are the shape functions and node potentials, respectively, corresponding to the node n_i which represents the node number. Since the shape functions, which are a function of the characteristics of the finite element and its node coordinates, are specified (Section 3.2), the problem is to solve for the unknown node potentials which will minimize the energy functional.

The energy functional, equation (3.1.2), can now be expressed as:

$$F(\Phi) = \frac{1}{2}\sum_{e=1}^{NOFE} \varepsilon_r^{(e)}\varepsilon_o \int_{\Omega_{FE}} (\nabla\Phi^{(e)})^2 d\Omega_{FE}$$

$$= \frac{1}{2}\sum_{e=1}^{NOFE} \varepsilon_r^{(e)}\varepsilon_o \int_{\Omega_{FE}} \{\nabla\alpha_{n_1}\Phi_{n_1} + \nabla\alpha_{n_2}\Phi_{n_2} + \ldots + \nabla\alpha_{n_k}\Phi_{n_k}\} \cdot$$

$$\{\nabla\alpha_{n_1}\Phi_{n_1} + \nabla\alpha_{n_2}\Phi_{n_2} + \ldots + \nabla\alpha_{n_k}\Phi_{n_k}\} d\Omega_{FE} \qquad (3.3.2)$$

$$= \frac{1}{2}\sum_{e=1}^{NOFE} \varepsilon_r^{(e)}\varepsilon_o [\underline{\Phi}^{(e)}]^T [S^{(e)}] [\underline{\Phi}^{(e)}]$$

where $[\Phi^{(e)}]$ is a column vector of length k containing the node potentials corresponding to the finite element (e):

$$[\Phi^{(e)}] = \begin{pmatrix} \Phi_{n_1} \\ \Phi_{n_2} \\ \cdot \\ \cdot \\ \cdot \\ \Phi_{n_k} \end{pmatrix} \qquad (3.3.3)$$

, $[\Phi^{(e)}]^T$ denotes the transpose vector, and the matrix $[S^{(e)}]$, referred to as the Laplacian element matrix pertaining to the finite element (e), is defined as:

$$S_{ij}^{(e)} = \int_{\Omega_{FE}} \nabla\alpha_{n_i} \cdot \nabla\alpha_{n_j} d\Omega_{FE} \quad ; i, j = 1, 2, \ldots, k \qquad (3.3.4)$$

3.3.2 Minimization of the Energy Functional.

In Section 3.3.1, the finite element energy functional was expressed as a function of the node potentials:

$$F = F\{\Phi_1, \Phi_2, , \ldots, \Phi_{NPTL}\} \tag{3.3.5}$$

which can be written as a sum of the individual element energies as:

$$F = \sum_{e=1}^{NOFE} F^{(e)} \tag{3.3.6a}$$

where

$$F^{(e)} = \frac{1}{2}\varepsilon_r^{(e)}\varepsilon_o[\underline{\Phi}^{(e)}]^T[S^{(e)}][\underline{\Phi}^{(e)}] \tag{3.3.6b}$$

Recall that the solution for the potential distribution sought, is the one which minimizes the energy functional (3.1.2),(3.3.2). Thus by using the node potentials as variational parameters, the energy functional is minimized with respect to the node potentials. This minimizing condition can be written as:

$$\frac{\partial F(\Phi)}{\partial \underline{\Phi}} = 0 \tag{3.3.7a}$$

where

$$\left[\frac{\partial F}{\partial \underline{\Phi}}\right] = \begin{pmatrix} \frac{\partial F}{\partial \Phi_1} \\ \frac{\partial F}{\partial \Phi_2} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial F}{\partial \Phi_{NPTL}} \end{pmatrix} \tag{3.3.7b}$$

Applying this minimizing condition to the energy functional, one obtains a linear system of equations of the form:

$$[A]\underline{\Phi} = \underline{B} \tag{3.3.8}$$

where $[A]$ is a symmetrical matrix by virtue of the self-adjointness of the Laplacian operator describing the potential distribution, and the right-hand side contains information on the boundary conditions. This is a set comprised of $NN = NPTL - NDIR$ equations which is solved for the unknown node potentials. The equations resulting from the derivative of the functional with respect to the Dirichlet nodes are eliminated from computation as these nodes are at fixed potentials.

If the free nodes are numbered from $1,2,\ldots,NN$, and the Dirichlet nodes from $NN+1, NN+2, \ldots, NPTL$, then a closed form expression for the numeric entries in the final system of equations can be written as:

$$A_{mn} = \sum_{e=1}^{NOFE} \varepsilon_r^{(e)} \left\{ \sum_{i=1}^{k} P_m(n_i^{(e)}) \cdot Q_m(n_i^{(e)}) \right\} \cdot \left\{ \sum_{j=1}^{k} P_n(n_j^{(e)}) \cdot Q_n(n_j^{(e)}) \cdot S_{ij}^{(e)} \right\}$$

$$B_m = -\sum_{e=1}^{NOFE} \varepsilon_r^{(e)} \left\{ \sum_{i=1}^{k} P_m(n_i^{(e)}) \cdot Q_m(n_i^{(e)}) \right\} \cdot \left\{ \sum_{j=1}^{k} (1 - P_n(n_j^{(e)})) \cdot Q_n(n_j^{(e)}) \cdot S_{ij}^{(e)} \cdot \Phi_{n_j} \right\} \tag{3.3.9}$$

where the subscripts m, n vary from $1, 2, \ldots, NN$, $\varepsilon_r^{(e)}$ represents the relative dielectric constant of the finite element (e), $n_i^{(e)}$ and $n_j^{(e)}$ represent the node number between $1, 2, \ldots, NPTL$ assigned to one of the k nodes of the finite element (e), Φ_{n_j} represents the fixed node potential assigned to the Dirichlet node n_j, and $S_{ij}^{(e)}$ represents an entry in the Laplacian matrix for the finite element (e). In equation (3.3.9) $P_m(n_i^{(e)}), P_n(n_j^{(e)})$ and $Q_m(n_i^{(e)}), Q_n(n_j^{(e)})$ are pulse functions defined to determine the contribution of a finite element, resulting from minimization of the energy functional with respect to the node potential Φ_{n_i}, to the final system of equations. These pulse functions are defined as:

$$P_m(n_i^{(e)}) = 1 \text{ if the node } n_i^{(e)} \le nn$$

$$P_m(n_i^{(e)}) = 0 \text{ if the node } n_i^{(e)} > nn \qquad (3.3.10a)$$

$$P_m(n_j^{(e)}) = 1 \text{ if the node } n_j^{(e)} \le nn$$

$$P_m(n_j^{(e)}) = 0 \text{ if the node } n_j^{(e)} > nn$$

and

$$Q_m(n_i^{(e)}) = 1 \text{ if the node } n_i^{(e)} = m$$

$$Q_m(n_i^{(e)}) = 0 \text{ if the node } n_i^{(e)} \ne m \qquad (3.3.10b)$$

$$Q_m(n_j^{(e)}) = 1 \text{ if the node } n_j^{(e)} = m$$

$$Q_m(n_j^{(e)}) = 0 \text{ if the node } n_j^{(e)} \ne m$$

A finite element consisting of a total of k nodes will generate up to k non zero equations which contribute to the final system. $Q_m(n_i^{(e)})$ is used to determine whether minimization

of the energy functional w.r.t the node potential Φ_{n_i}, i.e. $\partial F^{(e)}/\partial \Phi_{n_i} = 0$, will have a contribution to the mth row. $P_m(n_i^{(e)})$ is used to determine whether the node n_i is at a fixed potential. If so the equation is eliminated from the final system of equations. $Q_n(n_j^{(e)})$ is used to determine whether the equation has contribution to the nth column in the mth row. If so, $P_n(n_j^{(e)})$ determines whether the contribution is to the left hand side or right hand side of the equation. If node n_j is a free node, then the contribution will be to the left hand side in A_{mn}. If not, the node is a Dirichlet node and the contribution is to the right hand side in B_m.

Equations (3.3.9) and (3.3.10) represent closed form expressions for evaluating the numeric entries in the left hand side and right hand side matrices. These expressions are not recommended to be used for direct implementation into a computer algorithm because there exist much more efficient algorithms for assembling the final system of equations. What makes this algorithm so inefficient is that all the finite element nodes belonging to all the finite elements must be examined for each $m, n = 1, 2, \ldots, NN$. Since the only non-zero contribution from any one finite element will be from the equations resulting from the derivatives w.r.t the potentials of the k nodes of that finite element, i.e. $\partial F^{(e)}/\partial \Phi_{n_i}$ where $i = 1, 2, \ldots, k$, the algorithm in Figure 3-3 is recommended for assembling the final system of equations. This algorithm is much more efficient since it removes adding the zero contributions.

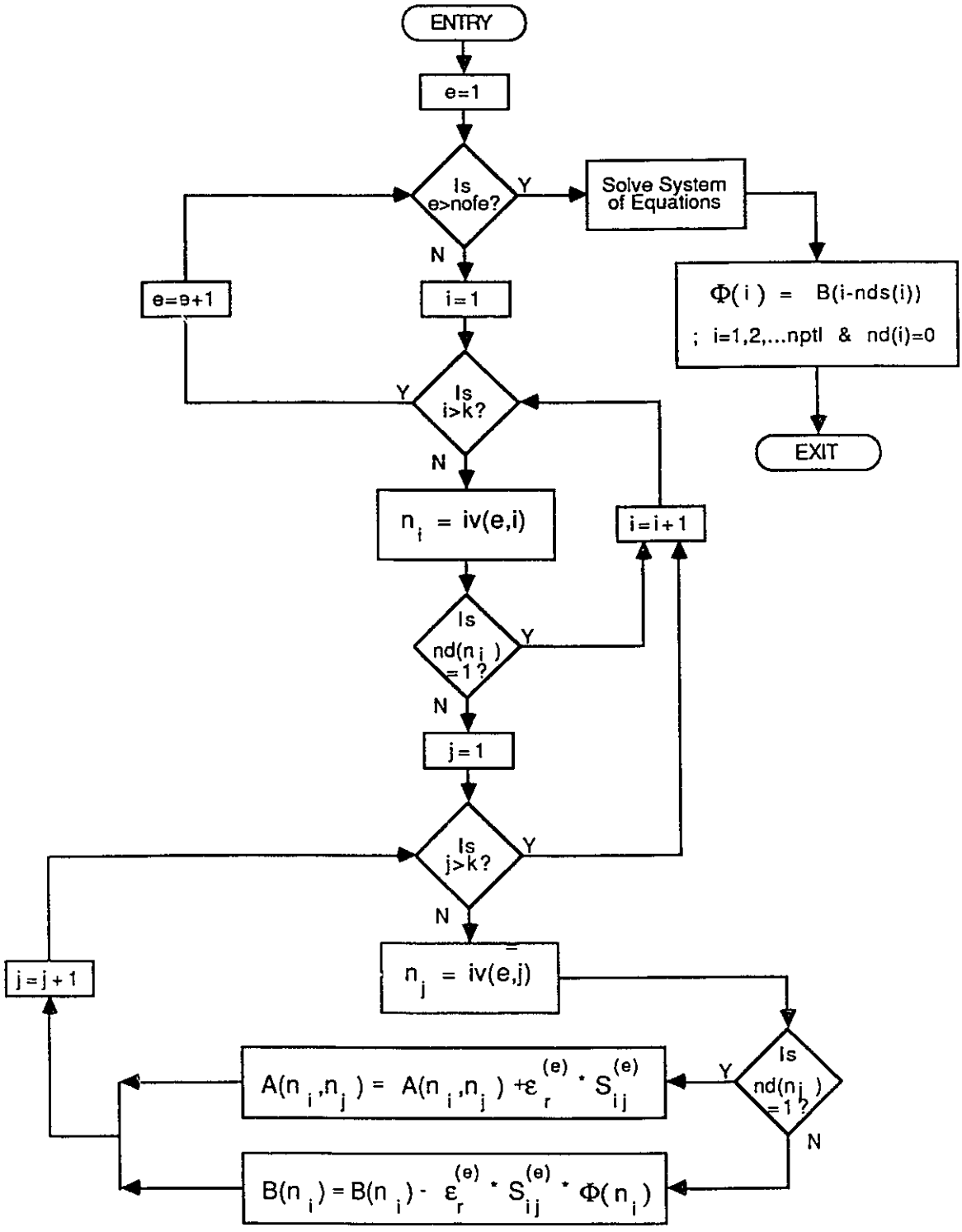


Figure 3-3: Algorithm for Assembling the Final System of Equations.

3.3.3 Finite Element Laplacian Matrices.

The finite element Laplacian matrix for a finite element consisting of a total of k nodes, is a $k X k$ symmetric matrix consisting of numeric entries which may be computed by evaluation of the integral:

$$[S_{ij}^{(e)}] = \int_{\Omega_{FE}} \nabla\alpha_{n_i}(x,y) \cdot \nabla\alpha_{n_j}(x,y) d\Omega_{FE} \quad ; (i,j = 1,2,\ldots,k) \tag{3.3.11}$$

once the potential distribution over the finite element has been defined and expressed in the form:

$$\Phi^{(e)}(x,y) = \alpha_{n_1}(x,y)\Phi_{n_1} + \alpha_{n_2}(x,y)\Phi_{n_2} + \ldots + \alpha(x,y)_{n_k}\Phi_{n_k} \tag{3.3.12}$$

These matrices are required for assembling the final system of equations and in computation of the total stored energy in the electrostatic field. Since the final system of equations is comprised of a summation of the individual contributions from each finite element, different types, orders, and shapes of finite elements (eg: first/second order, finite/infinite, triangular/patch) may be combined in a finite element mesh. However, continuity of the potential function defined in each finite element must be maintained between adjacent finite element edges. This is particularly significant as it allows various degrees and forms of approximations to be used throughout the studied region. Depending on the characteristics of the finite element, it is often possible to evaluate (3.3.11) analytically, yielding explicit expressions for determining the numeric entries in the Laplacian finite element matrix which is applicable to all finite elements of a given type, shape, and order as a function of its node coordinates. If this is not possible, other methods such as numerical integration can be used.

Alternatively, equation (3.3.11):

$$S_{ij} = \int_{\Omega_{FE}} \int \nabla \alpha_{n_i}(x,y) \cdot \nabla \alpha_{n_j}(x,y) d\Omega_{FE} \qquad (3.3.13a)$$

can be written as:

$$S_{ij} = \int_{\Omega_{FE}} \int \left\{ \frac{\partial \alpha_{n_i}(x,y)}{\partial x} \frac{\partial \alpha_{n_j}(x,y)}{\partial x} + \frac{\partial \alpha_{n_i}(x,y)}{\partial y} \frac{\partial \alpha_{n_j}(x,y)}{\partial y} \right\} d\Omega_{FE} \qquad (3.3.13b)$$

For the first order triangular finite element, the partial derivatives of the interpolatary shape functions (3.2.3),(3.2.4) are independent of position thus:

$$S_{ij} = \left\{ \frac{\partial \alpha_{n_i}(x,y)}{\partial x} \frac{\partial \alpha_{n_j}(x,y)}{\partial x} + \frac{\partial \alpha_{n_i}(x,y)}{\partial y} \frac{\partial \alpha_{n_j}(x,y)}{\partial y} \right\} \int_{\Omega_{FE}} \int d\Omega_{FE}$$

$$= A \left\{ \frac{\partial \alpha_{n_i}(x,y)}{\partial x} \frac{\partial \alpha_{n_j}(x,y)}{\partial x} + \frac{\partial \alpha_{n_i}(x,y)}{\partial y} \frac{\partial \alpha_{n_j}(x,y)}{\partial y} \right\} \qquad (3.3.14)$$

Evaluation of (3.3.14) for $i, j = 1,2,3$ yields explicit expressions for the calculating the numeric entries in the Laplacian element matrix for first-order triangular finite elements as:

$$S_{11} = \frac{1}{4A}\left\{(y_{n_2} - y_{n_3})^2 + (x_{n_3} - x_{n_2})^2\right\}$$

$$S_{12} = S_{21} = \frac{1}{4A}\{(y_{n_2} - y_{n_3})(y_{n_3} - y_{n_1}) + (x_{n_3} - x_{n_2})(x_{n_1} - x_{n_3})\}$$

$$S_{13} = S_{31} = \frac{1}{4A}\{(y_{n_2} - y_{n_3})(y_{n_1} - y_{n_2}) + (x_{n_3} - x_{n_2})(x_{n_2} - x_{n_1})\}$$

(3.3.15)

$$S_{22} = \frac{1}{4A}\left\{(y_{n_3} - y_{n_1})^2 + (x_{n_1} - x_{n_3})^2\right\}$$

$$S_{23} = S_{32} = \frac{1}{4A}\{(y_{n_3} - y_{n_1})(y_{n_1} - y_{n_2}) + (x_{n_1} - x_{n_2})(x_{n_2} - x_{n_1})\}$$

$$S_{33} = \frac{1}{4A}\left\{(y_{n_1} - y_{n_2})^2 + (x_{n_2} - x_{n_1})^2\right\}$$

For first order triangular finite elements the area integration in (3.3.13) and (3.3.14) was easily evaluated since the partial derivatives of the interpolatary shape functions were independent of position and thus could be removed from the integration. However, for second and higher order finite elements, this is not the case. For this reason it is easier to perform this integration using a transformation from rectangular to triangular coordinates. The transformation equations from rectangular to triangular coordinates are given by [4] (See Figure 3-4):

$$\zeta_{n_a} = \frac{1}{2A}\{(x_{n_b} y_{n_c} - x_{n_c} y_{n_b}) + (y_{n_b} - y_{n_c})x + (x_{n_c} - x_{n_b})y\}$$

$$\zeta_{n_b} = \frac{1}{2A}\{(x_{n_c} y_{n_a} - x_{n_a} y_{n_c}) + (y_{n_c} - y_{n_a})x + (x_{n_a} - x_{n_c})y\} \qquad (3.3.16)$$

$$\zeta_{n_c} = \frac{1}{2A}\{(x_{n_a} y_{n_b} - x_{n_b} y_{n_a}) + (y_{n_a} - y_{n_b})x + (x_{n_b} - x_{n_a})y\}$$

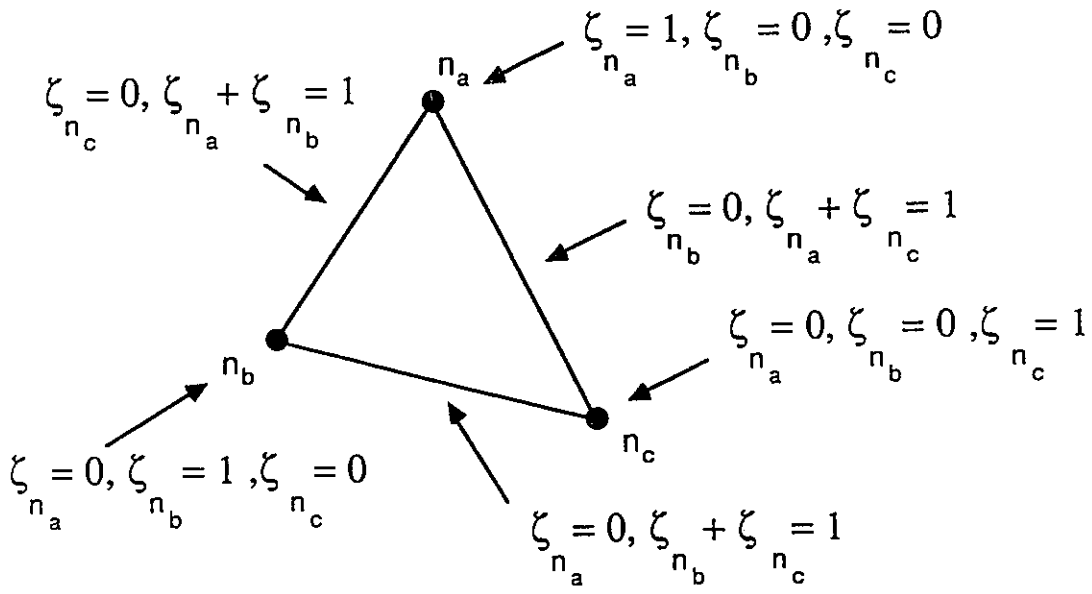


Figure 3-4: Triangular Coordinate System.

where $n_a = n_1$, $n_b = n_2$, and $n_c = n_3$ for the first order triangular finite element, and $n_a = n_1$, $n_b = n_4$, and $n_c = n_6$ for the second order finite element. Under this transformation, the potential distribution for the first order triangular finite element (3.2.3),(3.2.4) becomes:

$$\Phi^{(e)}(x,y) = \zeta_{n_a}\Phi_{n_1} + \zeta_{n_b}\Phi_{n_2} + \zeta_{n_c}\Phi_{n_3} \tag{3.3.17a}$$

and for second order triangular finite elements (3.2.7),(3.2.8) becomes:

$$\begin{aligned}\Phi^{(e)} = {} & \zeta_{n_a}(2\zeta_{n_a} - 1)\Phi_{n_1} + 4\zeta_{n_a}\zeta_{n_b}\Phi_{n_2} \\ & + 4\zeta_{n_a}\zeta_{n_c}\Phi_{n_3} + \zeta_{n_b}(2\zeta_{n_b} - 1)\Phi_{n_4} \\ & + 4\zeta_{n_b}\zeta_{n_c}\Phi_{n_5} + \zeta_{n_c}(2\zeta_{n_c} - 1)\Phi_{n_6}\end{aligned} \tag{3.3.17b}$$

where

$$\zeta_{n_a} + \zeta_{n_b} + \zeta_{n_c} = 1 \tag{3.3.18}$$

and the resulting area integration may be carried out as:

$$A = \int_{\Omega_{FE}}\int d\Omega_{FE} = 2A\int_0^1\int_0^{1-\zeta_{n_a}} d\zeta_{n_b}\, d\zeta_{n_a} \tag{3.3.19}$$

Applying the chain rule, the partial derivatives of $\alpha_{n_k}(x,y)$ required in (3.3.13) can be written as:

$$\begin{aligned}\frac{\partial\alpha_k}{\partial x} &= \frac{\partial\alpha_k}{\partial\zeta_a}\frac{\partial\zeta_a}{\partial x} + \frac{\partial\alpha_k}{\partial\zeta_b}\frac{\partial\zeta_b}{\partial x} + \frac{\partial\alpha_k}{\partial\zeta_c}\frac{\partial\zeta_c}{\partial x} \\ \frac{\partial\alpha_k}{\partial y} &= \frac{\partial\alpha_k}{\partial\zeta_a}\frac{\partial\zeta_a}{\partial y} + \frac{\partial\alpha_k}{\partial\zeta_b}\frac{\partial\zeta_b}{\partial y} + \frac{\partial\alpha_k}{\partial\zeta_c}\frac{\partial\zeta_c}{\partial y}\end{aligned} \tag{3.3.20}$$

Substitution of (3.3.20) in (3.3.13) yields:

$$S_{ij} = \int\int_{\Omega_{FE}} \left\{ \frac{\partial \alpha_{n_i}}{\partial \zeta_{n_a}} \frac{\partial \zeta_{n_a}}{\partial x} + \frac{\partial \alpha_{n_i}}{\partial \zeta_{n_b}} \frac{\partial \zeta_{n_b}}{\partial x} + \frac{\partial \alpha_{n_i}}{\partial \zeta_{n_c}} \frac{\partial \zeta_{n_c}}{\partial x} \right\}$$
$$\cdot \left\{ \frac{\partial \alpha_{n_j}}{\partial \zeta_{n_a}} \frac{\partial \zeta_{n_a}}{\partial x} + \frac{\partial \alpha_{n_j}}{\partial \zeta_{n_b}} \frac{\partial \zeta_{n_b}}{\partial x} + \frac{\partial \alpha_{n_j}}{\partial \zeta_{n_c}} \frac{\partial \zeta_{n_c}}{\partial x} \right\}$$
$$+ \left\{ \frac{\partial \alpha_{n_i}}{\partial \zeta_{n_a}} \frac{\partial \zeta_{n_a}}{\partial y} + \frac{\partial \alpha_{n_i}}{\partial \zeta_{n_b}} \frac{\partial \zeta_{n_b}}{\partial y} + \frac{\partial \alpha_{n_i}}{\partial \zeta_{n_c}} \frac{\partial \zeta_{n_c}}{\partial y} \right\} \tag{3.3.21}$$
$$\cdot \left\{ \frac{\partial \alpha_{n_j}}{\partial \zeta_{n_a}} \frac{\partial \zeta_{n_a}}{\partial y} + \frac{\partial \alpha_{n_j}}{\partial \zeta_{n_b}} \frac{\partial \zeta_{n_b}}{\partial y} + \frac{\partial \alpha_{n_j}}{\partial \zeta_{n_c}} \frac{\partial \zeta_{n_c}}{\partial y} \right\} d\Omega_{FE}$$

By evaluating the cotangent of the three interior angles of the triangle - θ_a, θ_b, and θ_c - and substitution of the ratio of lengths for the tangent of angles - θ'_a, θ''_a, θ'_b, θ''_b, θ'_c, θ''_c - (Appendix A) the following triangle identities can be obtained:

$$-2A\cot(\theta_{n_a}) = (y_{n_c} - y_{n_a})(y_{n_a} - y_{n_b}) + (x_{n_a} - x_{n_c})(x_{n_b} - x_{n_a})$$

$$= \left\{ \frac{\partial \zeta_{n_b}}{\partial x}\frac{\partial \zeta_{n_c}}{\partial x} + \frac{\partial \zeta_{n_b}}{\partial y}\frac{\partial \zeta_{n_c}}{\partial y} \right\} \qquad (3.3.22a)$$

$$-2A\cot(\theta_{n_b}) = (y_{n_b} - y_{n_c})(y_{n_a} - y_{n_b}) + (x_{n_c} - x_{n_b})(x_{n_b} - x_{n_a})$$

$$= \left\{ \frac{\partial \zeta_{n_a}}{\partial x}\frac{\partial \zeta_{n_c}}{\partial x} + \frac{\partial \zeta_{n_a}}{\partial y}\frac{\partial \zeta_{n_c}}{\partial y} \right\} \qquad (3.3.22b)$$

$$-2A\cot(\theta_{n_c}) = (y_{n_b} - y_{n_c})(y_{n_c} - y_{n_a}) + (x_{n_c} - x_{n_b})(x_{n_a} - x_{n_c})$$

$$= \left\{ \frac{\partial \zeta_{n_a}}{\partial x}\frac{\partial \zeta_{n_b}}{\partial x} + \frac{\partial \zeta_{n_a}}{\partial y}\frac{\partial \zeta_{n_b}}{\partial y} \right\} \qquad (3.3.22c)$$

$$2A\cot(\theta_{n_a}) + \cot(\theta_{n_b}) = (y_{n_a} - y_{n_b})^2 + (x_{n_b} - x_{n_a})^2$$

$$= \left(\frac{\partial \zeta_{n_c}}{\partial x}\right)^2 + \left(\frac{\partial \zeta_{n_c}}{\partial y}\right)^2 \qquad (3.3.22d)$$

$$2A\cot(\theta_{n_b}) + \cot(\theta_{n_c}) = (y_{n_b} - y_{n_c})^2 + (x_{n_c} - x_{n_b})^2$$

$$= \left(\frac{\partial \zeta_{n_a}}{\partial x}\right)^2 + \left(\frac{\partial \zeta_{n_a}}{\partial y}\right)^2 \qquad (3.3.22e)$$

$$= 2A\cot(\theta_{n_a}) + \cot(\theta_{n_c}) = (y_{n_c} - y_{n_a})^2 + (x_{n_a} - x_{n_c})^2$$

$$= \left(\frac{\partial \zeta_{n_b}}{\partial x}\right)^2 + \left(\frac{\partial \zeta_{n_b}}{\partial y}\right)^2 \qquad (3.3.22f)$$

Substitution of the identities in (3.3.22) in (3.3.21) yields the integral equation to be evaluated to determine the Laplacian element matrix for triangular finite elements:

$$S_{ij}^{(e)} = 4A^2\cot(\theta_a)\int_0^1\int_0^{1-\zeta_{n_a}}\left\{\frac{\partial\alpha_{n_i}}{\partial\zeta_{n_b}}-\frac{\partial\alpha_{n_i}}{\partial\zeta_{n_c}}\right\}\left\{\frac{\partial\alpha_{n_j}}{\partial\zeta_{n_b}}-\frac{\partial\alpha_{n_j}}{\partial\zeta_{n_c}}\right\}d\zeta_{n_b}d\zeta_{n_a}$$

$$+4A^2\cot(\theta_b)\int_0^1\int_0^{1-\zeta_{n_a}}\left\{\frac{\partial\alpha_{n_i}}{\partial\zeta_{n_a}}-\frac{\partial\alpha_{n_i}}{\partial\zeta_{n_c}}\right\}\left\{\frac{\partial\alpha_{n_j}}{\partial\zeta_{n_a}}-\frac{\partial\alpha_{n_j}}{\partial\zeta_{n_c}}\right\}d\zeta_{n_b}d\zeta_{n_a} \qquad (3.3.23)$$

$$+4A^2\cot(\theta_c)\int_0^1\int_0^{1-\zeta_{n_a}}\left\{\frac{\partial\alpha_{n_i}}{\partial\zeta_{n_a}}-\frac{\partial\alpha_{n_i}}{\partial\zeta_{n_b}}\right\}\left\{\frac{\partial\alpha_{n_j}}{\partial\zeta_{n_a}}-\frac{\partial\alpha_{n_j}}{\partial\zeta_{n_b}}\right\}d\zeta_{n_b}d\zeta_{n_a}$$

where $i,j = 1,2,3,4,5,6$ and from (3.3.18) $\zeta_{n_c} = 1-\zeta_{n_b}-\zeta_{n_a}$.

Evaluation of (3.3.23) yields the expressions for calculating the numeric entries in the Laplacian element matrix $[S^{(e)}]$ for first order finite elements:

$$S_{11} = \frac{1}{2}\{\cot(\theta_{n_b})+\cot(\theta_{n_c})\}$$

$$S_{12} = S_{21} = -\frac{1}{2}\{\cot(\theta_{n_c})\}$$

$$S_{13} = S_{31} = -\frac{1}{2}\{\cot(\theta_{n_b})\} \qquad (3.3.24)$$

$$S_{22} = \frac{1}{2}\{\cot(\theta_{n_a})+\cot(\theta_{n_c})\}$$

$$S_{23} = S_{32} = -\frac{1}{2}\{\cot(\theta_{n_a})\}$$

$$S_{33} = \frac{1}{2}\{\cot(\theta_{n_a})+\cot(\theta_{n_b})\}$$

and for second order finite elements:

$$S_{11} = \frac{1}{2}\{\cot(\theta_{n_b}) + \cot(\theta_{n_c})\}$$

$$S_{12} = S_{21} = S_{24} = S_{42} = -\frac{2}{3}\{\cot(\theta_{n_c})\}$$

$$S_{13} = S_{31} = S_{36} = S_{63} = -\frac{2}{3}\{\cot(\theta_{n_b})\}$$

$$S_{14} = S_{41} = -\frac{1}{6}\{\cot(\theta_{n_c})\}$$

$$S_{15} = S_{51} = S_{26} = S_{62} = S_{34} = S_{43} = 0$$

$$S_{16} = S_{61} = \frac{1}{6}\{\cot(\theta_{n_b})\}$$

$$S_{22} = S_{33} = S_{55} = \frac{4}{3}\{\cot(\theta_{n_a}) + \cot(\theta_{n_b}) + \cot(\theta_{n_c})\}$$

(3.3.25)

$$S_{23} = S_{32} = -\frac{4}{3}\{\cot(\theta_{n_a})\}$$

$$S_{25} = S_{52} = -\frac{4}{3}\{\cot(\theta_{n_b})\}$$

$$S_{35} = S_{53} = -\frac{4}{3}\{\cot(\theta_{n_c})\}$$

$$S_{44} = \frac{1}{2}\{\cot(\theta_{n_a}) + \cot(\theta_{n_c})\}$$

$$S_{45} = S_{54} = S_{56} = S_{65} = -\frac{2}{3}\{\cot(\theta_{n_a})\}$$

$$S_{46} = S_{64} = \frac{1}{6}\{\cot(\theta_{n_a})\}$$

$$S_{66} = \frac{1}{2}\{\cot(\theta_{n_a}) + \cot(\theta_{n_b})\}$$

where θ_a, θ_b, θ_c are derived in Appendix A, and nodes $n_a = n_1$, $n_b = n_2$, $n_c = n_3$ for the first order triangular finite elements and $n_a = n_1$, $n_b = n_4$, $n_c = n_6$ for the second order triangular finite elements.

3.4 Capacitance/Inductance Matrix Computation.

Consider a system of 'n+1' conductors such as that illustrated in Figure 2-1. Let the potential on the 'n+1' reference conductor, the ground plane, be chosen to be zero volts with respect to the potential on the other conductors which are charged to the potentials $\Phi_1 = V_1$, $\Phi_2 = V_2$, ...,$\Phi_n = V_n$. If the first conductor is charged up to a non zero potential of $\Phi_1 = V_1$, and the rest of the conductors grounded to the reference conductor (i.e. $V_2 = V_3 = ... = V_n = 0$ volts), then the electrostatic field and charge on each conductor is uniquely determined by the value of Φ_1. In this situation the charge on each conductor is proportional to Φ_1 by the constants:

$$Q_1 = C_{11}V_1, \quad Q_2 = C_{21}V_1, \quad \ldots, \quad Q_n = C_{n1}V_1 \tag{3.4.1}$$

where C_{i1} for $i = 2,\ldots,n$ are the constants of proportionality referred to as the capacitance between the *ith* and first conductor due to the potential on the first conductor.

If the conductors are infinitely long, the charge on each conductor is no longer finite, and is expressed as a per-unit length charge. Applying the principle of superpostion, the per-unit length capacitance matrix for a system of 'n+1' conductors relates charge to voltage as:

$$\begin{pmatrix} Q_1 \\ Q_2 \\ . \\ . \\ . \\ Q_n \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ . & . & & . \\ . & . & . & . \\ . & . & & . \\ C_{n1} & C_{n2} & \dots & C_{nn} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ . \\ . \\ . \\ V_n \end{pmatrix} \tag{3.4.2}$$

where Q_i is the per-unit length charge on the *ith* conductor, and V_j is voltage on the *jth*, i.e. non-zero potential, conductor. Thus to determine the entire per-unit length capacitance matrix for a system of 'n+1' conductors requires solution for the potential distribution 'n' times by using the Finite Element Method. Each time a static potential $\Phi_j = V_j = 1$ volt is applied to the *jth* conductor while grounding the rest. In this manner as $i = 1, n$, the *jth* column of the per-unit length capacitance matrix is determined by solving for the potential distribution and then computing the per-unit length charge on the *ith* conductor as:

$$C_{ij} = \frac{Q_i}{V_j}\Bigg]_{(V_1,\dots,V_{j-1},V_{j+1},\dots,V_n)=0} \tag{3.4.3}$$

where for $i = j, C_{ii}$ is the self-capacitance between the *ith* conductor and ground, and for $i \neq j, C_{ij}$ is the mutual capacitance between the *ith* and *jth* conductors. The total charge residing on the *ith* conductor is then calculated by applying Gauss's law and is given by:

$$\begin{aligned} Q_i &= \int_{S_i} \vec{D} \cdot \hat{n} dS_i \\ &= \int_{S_i} \varepsilon \vec{E} \cdot \hat{n} dS_i \\ &= \int_{S_i} -\varepsilon \nabla \Phi \cdot \hat{n} dS_i \end{aligned} \tag{3.4.4}$$

where S_i denotes the surface surrounding the *ith* conductor. In computing the per-unit length charge, this surface integral is effectively reduced to a line integral around the conductor boundary as the integrand, being uniform in $\hat{z}$, allows integration over a unit length.

An alternative methods exists which may be used to compute the self capacitance of a conductor with respect to ground. This is based on the total per-unit length stored electrical energy in the system which can be written as:

$$W_{(e)} = \frac{1}{2}[\underline{V_i}]\,[C]\,[\underline{V_j}] \tag{3.4.5}$$

where W_e is the energy per-unit length, $[\underline{V_i}]$ and $[\underline{V_j}]$ represent the potentials applied to the conductors, and $[C]$ is the per-unit length capacitance matrix. If a non zero static potential is applied to the *jth* conductor while the rest are grounded, then (3.4.5) reduces to:

$$C_{ij} = \frac{2W_e}{V_j^2} \tag{3.4.6}$$

where C_{ij} is the self capacitance of the *jth* conductor and V_j is the potential difference of the track with respect to ground used as a reference. Once the potential distribution is obtained using the finite element method, the total per-unit energy stored in the electrostatic field can be computed by a summation of the individual finite element energies via:

$$\begin{aligned} W_e &= \frac{1}{2}\int_{\Re} \varepsilon |\nabla\Phi|^2 dxdy \\ &= \frac{1}{2}\sum_{e=1}^{NOFE} \varepsilon_r^{(e)}\varepsilon_o [\underline{\Phi^{(e)}}]^T [S^{(e)}]\,[\underline{\Phi^{(e)}}] \end{aligned} \tag{3.4.7}$$

where $[\underline{\Phi}^{(e)}]$ and $[S^{(e)}]$ represents the solved potentials and Laplacian element matrix pertaining to the finite element (e).

Calculation of the per-unit length inductance matrix is obtained from the capacitance matrix calculated with the dielectric replaced by free space:

$$[L] = \varepsilon_o \mu_o [C_o]^{-1} \tag{3.4.8}$$

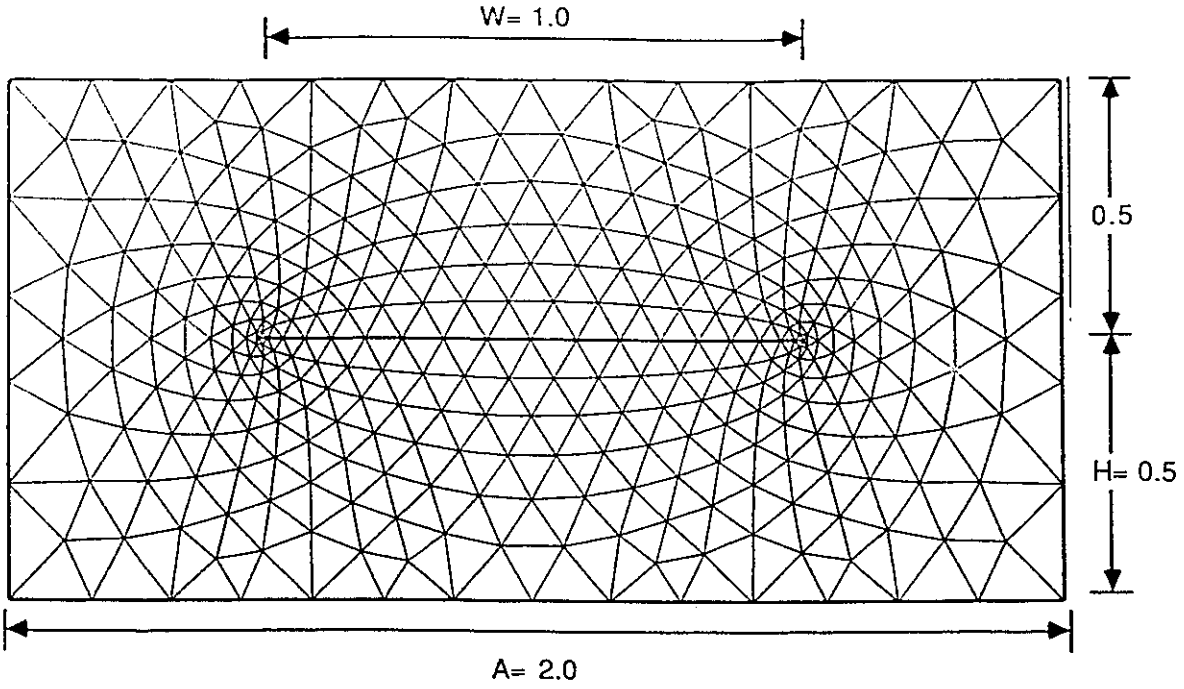
In the above expression ε_o and μ_o are the permittivity and permeability of free space, respectively, and $[C_o]$ is the per-unit length capacitance matrix in the absence of the dielectric.

3.5 Automatic Mesh Generation.

The use of two-dimensional finite elements for analysis of multiconductor configurations of arbitrary cross-section requires the division of the finite element region surrounding the conductors into finite elements (Figures 3-5 and 3-6). Of primary importance is an algorithm for automated mesh generation which allows maximum flexibility in the placement of the finite elements, a minimum amount of user time and input, and is adaptable for different multiconductor configurations. This is accomplished via a node placement algorithm which generates nodes throughout the finite element region and is followed by a triangulization algorithm which is used to connect the nodes to form a triangular finite element mesh.

In attempting to maximize the effectiveness of a finite element mesh in solving potential problems, both the manner in which the potential distribution to be approximated varies, and the characteristics of the finite elements used, should be examined. The equipotential lines of the electrostatic potential surrounding a conductor at a specified potential - $\Phi_c = \Phi_o$ - above an infinite ground plane at potential - $\Phi = 0$ - form a series of

TEM Cell Configuration



Microstrip Configuration
(ε_r = 9.5)

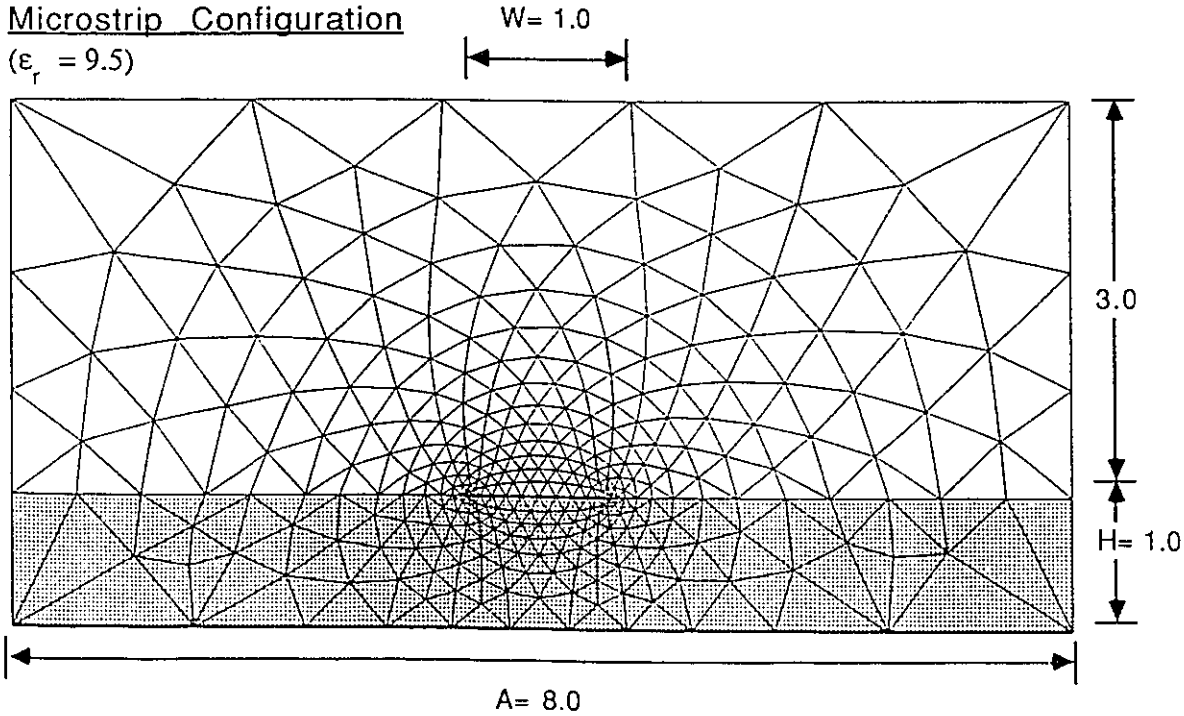


Figure 3-5: Finite Element Division.
(Single Conductor Configurations)

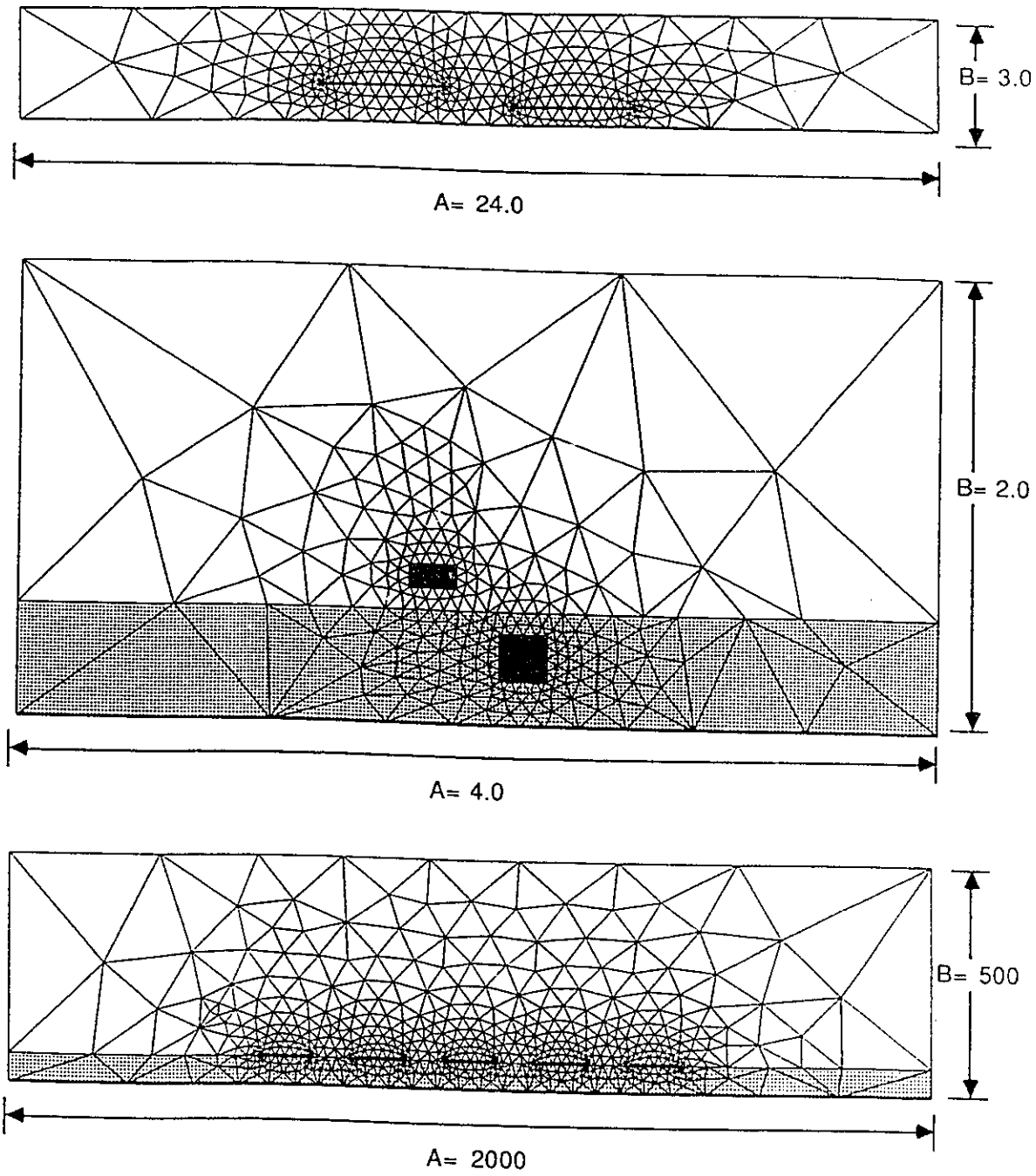


Figure 3-6: Finite Element Division.
(Multiconductor Configurations)

elliptically shaped contours. The corresponding lines of the electrostatic field and direction of maximum variation of the potential distribution are everywhere perpendicular to these elliptically shaped contours, and extend outward normal from the conductor and terminate on the ground plane. Thus a good finite element mesh used to approximate the potential distribution can be obtained by placing the triangular finite elements along elliptical contours surrounding the conductor. This allows the maximum xy variation of the potential distribution over the finite element, which extends from the perpendicular bisector of a side of the triangle to the opposing vertex, to be in the direction of maximum variation of the potential distribution. In addition, since the maximum variation of the potential distribution is in the vicinity directly surrounding the conductor particularly around the corners of the conductor, a more dense mesh, to better approximate the more rapid variation in this area is desired. Triangular finite elements, often used in two-dimensional finite element analysis, provide good flexibility in conforming to complicated surfaces and boundaries. In using triangular finite elements, the numerical accuracy of the solution can often be improved by maximizing the interior angles of the triangles [10]. In this respect, the equilateral triangle, which maximizes all interior angles to 60°, is an optimum triangular finite element. Thus the algorithm for automated mesh generation is faced with three major tasks. These are 1) placement of the triangular finite elements along elliptical contours around the conductors, 2) creation of a finite element mesh consisting of equilateral triangles, and 3) effective grading of the finite element mesh from a more dense mesh in the vicinity surrounding the conductors to a less dense mesh further away from the conductors. This is the rationale used in the development of the automated mesh generator.

The algorithm begins by placing all mandatory nodes at the four corners of the rectangular finite element region and at the four corners of the dielectric subregions. After placement of all mandatory nodes, a series of nodes are placed on the boundaries of each conductor and an

additional series of nodes are placed in an elliptical pattern which encloses each conductor. The focii and major/minor lengths of the elliptical contour on which the nodes are placed is chosen such to optimize the triangles to be placed within the elliptical contour to be as close to equilateral triangles as possible and yet still conforming to the elliptical contour. These parameters of the ellipse are based upon the user input of the width and thickness of the conductor as well as the number of finite elements to be placed on the conductor boundary. The nodes placed on the elliptical boundary are stored sequentially as they are generated for later use.

The algorithm then recalls the original sequence of nodes surrounding the first conductor and selects the first two nodes. These two nodes represent an adjacent node pair located on the elliptical boundary surrounding the first conductor. It then attempts to generate a new node, based on the distance between the node pair, in the direction outward normal from the ellipse containing the conductor. After this, the second node and third node on the original elliptical boundary are selected and a new node is attempted to be placed in a similar fashion. This process continues on until one complete revolution around the original elliptical boundary is completed and all the nodes on the it have been used to attempt to generate a new node. During this process, all the new nodes generated are sequentially stored as they are generated to form the new boundary node placement. If both nodes selected lie on the outer boundary of the finite element region, no new node is generated, and both nodes in the node pair are stored as part of the new boundary node placement. If no new node is generated because it would be too close to a previously generated node, then only the first node in the node pair is stored. The algorithm then proceeds to each of the remaining conductors, one at a time, and repeats this procedure. This whole process is repeated, and each time, a new series of nodes is attempted to be generated based upon the last node placement, until the entire finite element region is filled and no more nodes can be placed.

When a node pair is selected, the new node is attempted to be placed at a distance $c_1\sqrt{\frac{3}{2}}$ times the distance between the adjacent node pair on the perpendicular bisector of the line segment joining this node pair in the direction outward normal from the node boundary. Let the first and second node in the node pair be labelled n_i, located at coordinates (x_{n_i}, y_{n_i}), and n_{i+1}, located at coordinates $(x_{n_{i+1}}, y_{n_{i+1}})$, respectively. The new node is attempted to be placed at the coordinates (x_{new}, y_{new}) given by:

$$x_{new} = \sqrt{\frac{3}{2}} c_1 (y_{n_i} - y_{n_{i+1}}) + \left(\frac{x_{n_i} + x_{n_i+1}}{2} \right)$$

$$y_{new} = \sqrt{\frac{3}{2}} c_1 (x_{n_{i+1}} - x_{n_i}) + \left(\frac{y_{n_i} + y_{n_i+1}}{2} \right) \qquad (3.5.1)$$

If this new nodes coordinates lie outside the finite element region boundary, the new node is extrapolated along the perpendicular bisector to lie on the finite element boundary, where it will be attempted to be placed. If this is not the case, then the new node will be attempted to be placed at a distance $\sqrt{\frac{3}{2}}$ times the distance between the node pair to form an equilateral triangle (Figure 3-7):

$$x_{new} = \sqrt{\frac{3}{2}} (y_{n_i} - y_{n_{i+1}}) + \left(\frac{x_{n_i} + x_{n_i+1}}{2} \right)$$

$$y_{new} = \sqrt{\frac{3}{2}} (x_{n_{i+1}} - x_{n_i}) + \left(\frac{y_{n_i} + y_{n_i+1}}{2} \right) \qquad (3.5.2)$$

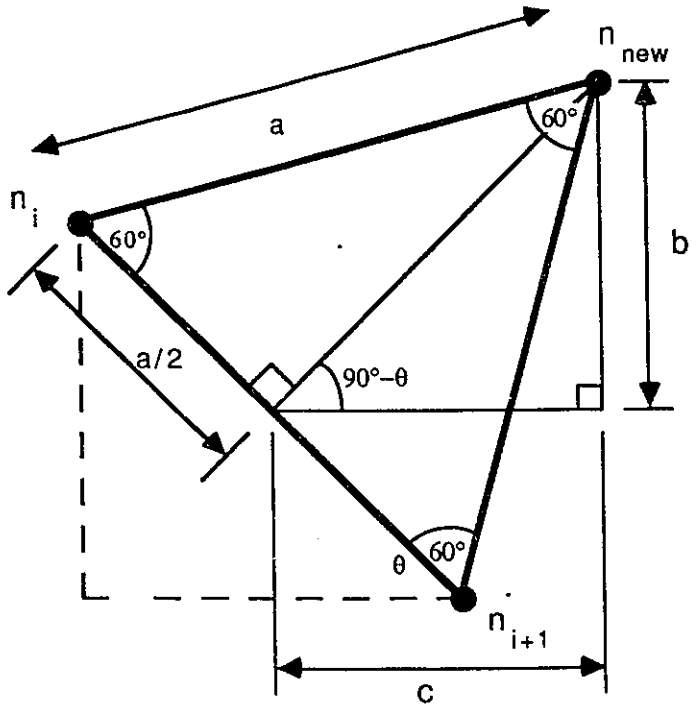


Figure 3-7: Creation of an Equilateral Triangle.

Equation (3.5.2) is identical to (3.5.1) with the constant $c_1 = 1$. Before this new node is accepted, it must pass a minimum distance rule which places a restriction on how far the new node must be from any other previously generated node. Under this minimum distance rule:

$$r_{new} \leq \frac{r_c}{c_2} \tag{3.5.3a}$$

where

$$r_c = \sqrt{\left\{xnew - \left(\frac{x_{n_i} - x_{n_{i+1}}}{2}\right)\right\}^2 + \left\{ynew - \left(\frac{y_{n_i} - y_{n_{i+1}}}{2}\right)\right\}^2}$$

$$r_{new} = \sqrt{\{xnew - x_{n_k}\}^2 + \{ynew - y_{n_k}\}^2} \tag{3.5.3b}$$

In (3.5.3) r_c represents the distance along the perpendicular bisector of the line connecting the node pair and the new node, and r_{new} represents the distance between the new node and a previously generated node denoted by n_k. If this condition is met then the node is rejected. The constants c_1 and c_2 can be selectively chosen but to avoid poorly shaped finite elements it is recommended to select $c_1 > 1$ and $c_2 > \sqrt{3}\, c_1/\sqrt{1+3c_1^2}$. The values chosen for c_1 and c_2 in the automated mesh generation algorithm were 1.5 and $\sqrt{2}$ respectively.

Once all nodes are placed an excellent triangulization algorithm is used to construct the finite element mesh [9]. This algorithm will triangulate a simply connected finite element region given the number of closed boundaries, a list of the sequentially stored nodes on each boundary, and the total number of nodes and their respective (x, y) coordinates. The resultant finite element mesh is then passed through a node relocation algorithm which ensures that all finite elements conform to the dielectric interface boundaries. Next, a node smoothing algorithm was written

based upon an idea suggested in [10] which, through an iterative approach, relocates all nodes at the centroid of all the nodes by which it is directly connected through a finite element edge. Finally, if second order finite elements are desired the mesh is passed through an algorithm which translates any first order triangular finite element mesh to a second order triangular mesh.

To evaluate the performance any finite element mesh generator requires consideration of CPU time required to create the mesh, amount of user input and user input time required, and quality of the resultant mesh. The automatic mesh generator described has the advantages of creating a finite element mesh consisting of near equilateral triangles which to a certain degree conform along the equipotentials of the potential distribution, a minimum amount of user input, and effective grading of mesh from areas were the potential distribution varies most rapidly and a denser mesh is desired, to areas were it varies more slowly and a less dense mesh is desired.

3.6 Finite Element Results.

Figure 3-8 illustrates the results obtained using the finite element method and automated mesh generator to evaluate the capacitance of a TEM Cell configuration - C - and impedance of a microstrip - Z_o -, as the number of finite elements on the conductor boundary is increased. This Figure plots the results obtained for C and Z_o as a function of the number of finite elements in the mesh and as a function of the CPU time to run on a microvax workstation. The numeric data plotted, as well as the percent error from the final converged value are tabulated in Table 3-1. The CPU time for the microstrip configuration can effectively be doubled since the program was executed twice due to the presence of the dielectric substrate. The characteristic

FIGURE 3-8: FEM CONVERGENCE TEM CELL & MICROSTRIP CONFIGURATIONS

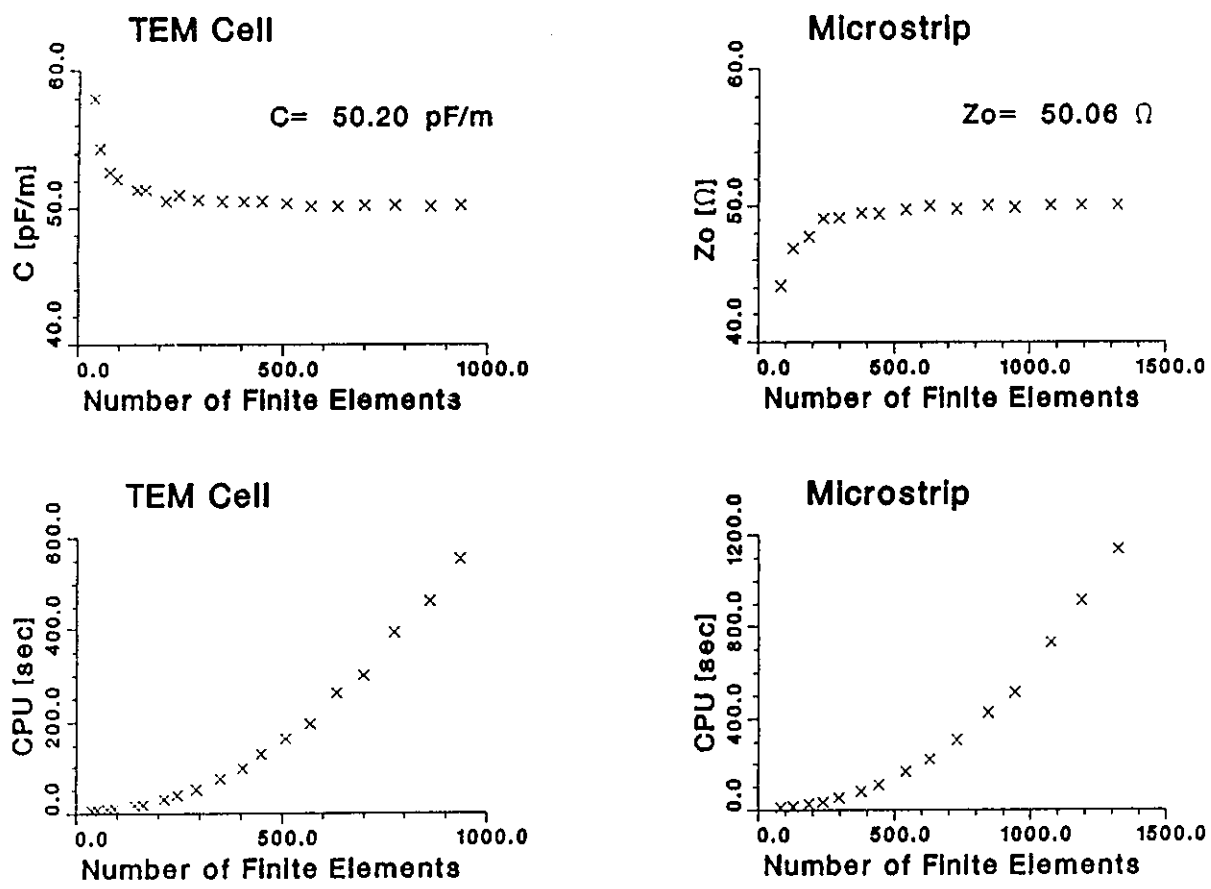


Table 3-1: Convergence Data.

TEM Cell Configuration				Microstrip Configuration			
Number of of Finite Elements	Capacitance (pF/m)	CPU Time (Seconds)	Error* (%)	Number of of Finite Elements	Impedance (ohms/m)	CPU Time (Seconds)	Error** (%)
38	58.0	7.9	+15.5	83	44.1	10.8	-12.0
52	54.3	8.3	+8.2	129	46.8	16.2	-6.6
76	52.6	9.9	+4.8	187	47.7	26.7	-4.8
94	52.1	11.5	+3.8	238	49.1	36.8	-2.0
142	51.3	18.0	+2.2	298	49.1	53.5	-2.0
162	51.3	19.1	+2.2	378	49.5	81.6	-1.2
214	50.5	31.3	+0.6	443	49.4	110.2	-1.4
246	50.9	40.2	+1.4	541	49.7	166.8	-0.8
292	50.6	53.1	+0.8	630	50.0	221.4	-0.2
350	50.5	75.4	+0.6	730	49.8	309.7	-0.6
404	50.4	97.3	+0.4	846	50.0	427.2	-0.2
448	50.5	130.3	+0.6	945	49.9	514.1	-0.4
508	50.3	164.7	+0.2	1077	50.0	729.9	-0.2
568	50.1	196.3	-0.2	1190	50.1	918.1	0.0
634	50.2	262.4	0.0	1324	50.1	1140.9	0.0
700	50.2	301.5	0.0				
774	50.2	392.9	0.0				
862	50.2	463.0	0.0				
934	50.2	554.0	0.0				

* For the TEM CELL Capacitance, the error from the final converged value of 50.2 pF/m is computed from:

$$\text{Error (\%)} = \frac{\text{Value - 50.2}}{50.2} \quad \text{X 100}$$

** For the Microstrip Impedance, the error from the final converged value of 50.1 ohms/m is computed from:

$$\text{Error (\%)} = \frac{\text{Value - 50.1}}{50.1} \quad \text{X 100}$$

impedance was calculated using:

$$Z_c = \sqrt{\frac{\mu_o \varepsilon_o}{C C_o}} \tag{3.6.1}$$

where C and C_o represent the per unit length capacitance in the presence and absence of the dielectric substrate, respectively. In both cases first order triangular finite elements were used. The geometry for these two problems is illustrated in Figure 3-4. For the TEM Cell results the converged value of $C = 50.2\,\rho F/m$ was obtained with finite element mesh's consisting of more 600 finite elements and for the microstrip configuration $Z_o = 50.1\,\Omega$ was obtained with finite element mesh's consisting of upwards of 1000 finite elements. The latter result can be compared with that published results [12], which computed $Z_o = 50.8097\,\Omega$ using a method of moments algorithm.

Results obtained using for two types of multiconductor configurations, illustrated in Figure 3-9, are tabulated in Table 3-2. Comparisons are made with published results obtained using the method of moments [12]. In evaluation of the per-unit length parameters, the energy definition, given in equations (3.4.6) and (3.4.7), was used to calculate the self capacitance values, and the charge definition, given in equations (3.4.3) and (3.4.4), was used to calculate the mutual capacitance values. This results from the observation that, as more finite elements were added, the rate of convergence of the self-capacitance solution obtained via using the energy definition is much more rapid than those obtained using the charge definition. Recall that in application of the finite element method, the solution for the potential distribution is obtained via minimization of the energy functional, which is proportional to the total energy in the finite element mesh. Thus the estimation of the total system energy, requiring a summation of all the element energies, is inherently more accurate than the estimation of the total charge residing on a conductor which limits itself to the variation of the potential distribution as

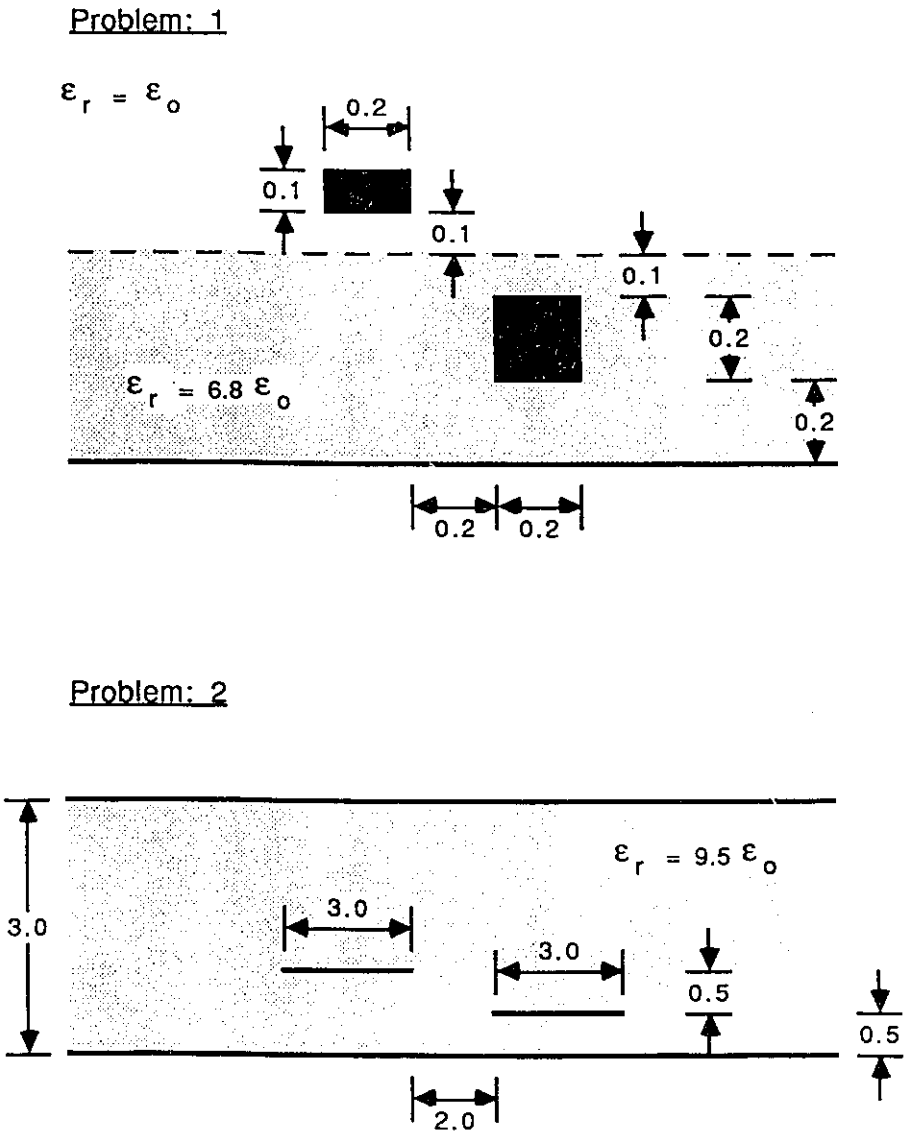


Figure 3-9: Multiconductor Configurations Used For Comparison.

Table 3-2

PER-UNIT LENGTH PARAMETERS: FEM AND MOM COMPARISON.

Problem 1	FEM	MOM
C_{11}	0.3654E-10	0.3651E-10
C_{12}	-0.1538E-10	-0.1562E-10
C_{21}	-0.1599E-10	-0.1562E-10
C_{22}	0.2146E-09	0.2099E-09
L_{11}	0.5763E-06	0.5315E-06
L_{12}	0.1351E-06	0.1241E-06
L_{21}	0.1404E-06	0.1241E-06
L_{22}	0.3251E-06	0.3235E-06

Problem 2	FEM	MOM
C_{11}	0.5399E-09	0.5356E-09
C_{12}	-0.8287E-11	-0.9250E-11
C_{21}	-0.8670E-11	-0.9250E-11
C_{22}	0.7861E-09	0.7834E-09
L_{11}	0.1972E-06	0.2033E-06
L_{12}	0.2069E-08	0.2401E-08
L_{21}	0.2165E-08	0.2401E-08
L_{22}	0.1345E-06	0.1390E-06

Units : Capacitance - F/m
Inductance - H/m

approximated by the finite elements located directly on the boundary of the conductor. Better results using the charge definition may well be obtained by integrating along the elliptical contour, where the normal derivatives are better approximated, to compute the total charge enclosed. However this will require additional computational effort.

In addition the placement of the finite elements, limited by the algorithm for automated mesh generation, was found to be a crucial factor in the speed of convergence and numerical accuracy of the solution. Adding finite elements in regions where the potential distribution is adequately represented by a fewer number of finite elements will not substantially increase the accuracy of the solution as compared to placing the additional finite elements in regions where the potential varies more rapidly and is insufficiently approximated. A common solution to these types of problems is through the use of finite element optimization methods such as adaptable meshing algorithms [8,11] which begin with an initial finite element division, solve the problem, and then use some analysis criteria for judging where more finite elements are required. Several other approaches to improve numerical accuracy for a given amount of computational effort include the use of 1) singular elements [7], 2) a denser mesh at the expense of additional nodes, and/or 3) the use of higher-order elements [4]. This applies particularly at the sharp corners of the conductors, where the charge density is infinite, and in the direct vicinity around them, in which the potential varies most rapidly.

3.7 Finite Element Method - Conclusions.

The results presented in this section indicate that the Finite Element Method can be used to calculate the distributed parameters for both single and multiconductor systems. The two-dimensional solution of Laplace's equation used in calculation of these parameters was found to be stable and convergent. Although the FEM was found to be simple in mathematical

derivation, a significant amount of time and computer code was required to generate the finite element mesh. This is reflected in the size of the subroutine for automated mesh generation which consisted of about 1000 lines of Fortran source code. It was also observed that 1) the CPU time required to compute the distributed parameters increases exponentially as the number of finite elements is linearly increased, 2) that a large system of equations (i.e. 600-1000 equations for single conductor configurations, upwards of 1000 equations for multiconductor configurations) is required to obtain suitably converged results, and 3) as finite elements are added, the rate of convergence is rapid at first, and then slows down. This suggests that in applying this method of solution, there exists a point where continuing to increase the number of finite elements does not yield a beneficial increase in convergence for the additional computational time required.

These findings indicate that because of the large system of equations required, this method of solution is 1) not practical for real-time CAD applications because of the CPU time required, and 2) not feasible for use on the vast majority of present day home Personal Computers due to the amount of memory required for program execution. In a real-time CAD application, the user would ideally receive feedback from his or her input immediately. From the single convergence results it was observed that it required approximately 15 CPU minutes to set up and solve a system of 1000 equations on a microvax workstation. Thus to compute the entire per-unit length capacitance and inductance matrices for a system of five conductors will require the solution of 1000 unknowns 10 times (two hours and thirty minutes). Thus to compute the per-unit length transmission line parameters using the finite element method, real-time analysis is clearly not achievable on present day computers. For CAD and analysis of EMI/C on printed circuit boards, this is not necessarily a requirement. For example, since most CAD layout packages and manufacturing processes have fixed configurations, substrate thicknesses,

dielectric constants, and allowable track widths and separations, the transmission line parameters could be stored in a look up table and then used in real-time EMC analysis programs. Also, in the near future, with faster computers and PC's with expanded memory becoming available, these present disadvantages will become less significant.

4 CROSSTALK ANALYSIS

4.1 Crosstalk on Printed Circuit Boards.

Crosstalk on a printed circuit board results from the coupling of the electromagnetic fields between nearby circuits. The electromagnetic sources of these fields are predominantly due to the line voltages and currents propagating on the conductors (i.e. signal tracks). At higher frequencies these signal tracks act as antennas and become more efficient radiators as well as receptors of electromagnetic energy.

For a system of conductors immersed in a homogeneous medium, the electromagnetic fields lie in the plane transverse to the direction of propagation. This mode of propagation is known as the TEM mode of propagation. Consider now a simple printed circuit board configuration such as that illustrated in Figure 2-1. This multiconductor configuration consists of 'n' conductors on a dielectric substrate above a solid ground plane (the 'n+1' reference conductor). Let the conductors be of total length 'L' and oriented in the 'x' direction. In this situation, although the conductor cross section is inhomogeneous and the pure TEM mode does not completely describe the physical situation, the assumption is made that the dominant mode of propagation is the TEM mode with the electromagnetic field components lying in the 'yz' (i.e. transverse) plane. This approximation is referred to as the "Quasi-TEM" mode of propagation. Under this approximation the currents and voltages vary in time as a function of position along its length and are represented by $V(x,t)$ and $I(x,t)$. To model this mode of propagation and include the effects of electromagnetic coupling, the distributed parameter multiconductor transmission line model is used (Section 2.2). Here C'_{ii} represents the per-unit

length capacitance between the *ith* conductor and the reference conductor, and C'_{ij} the mutual capacitance between the *ith* and *jth* conductors. Similarly L_{ij} represents the self inductance of the *ith* conductor and L_{ij} the mutual inductance between the *ith* and *jth* conductors.

In this section, the multiconductor configurations are assumed uniform along their length, and the medium surrounding the conductors characterized by real permeability $\mu = \mu_o$ and real permittivity $\varepsilon(x,y) = e_r(x,y)\varepsilon_o$. The conductors are assumed lossless and the media surrounding the conductors are both linear and isotropic such that the per unit length parameters are real and symmetric (i.e. $L_{ij} = L_{ji}$ and $C_{ij} = C_{ji}$).

4.2 The Multiconductor Transmission Line Equations.

Consider the configuration of 'n+1' conductors of total length 'L' such as that illustrated in Figure 2-1. By dividing the line along its length into electrically small segments, each longitudinally homogeneous subsection can be modelled with a lumped circuit model. For one such sub-section (Figure 4-1), the voltage and current equations are given by:

$$\begin{aligned}
V_1(x,t) &= V_1(x+\Delta x,t) + L_{11}\Delta x\frac{\partial I_1(x,t)}{\partial t} + L_{12}\Delta x\frac{\partial I_2(x,t)}{\partial t} + \cdots + L_{1n}\Delta x\frac{\partial I_n(x,t)}{\partial t} \\
V_2(x,t) &= V_2(x+\Delta x,t) + L_{21}\Delta x\frac{\partial I_1(x,t)}{\partial t} + L_{22}\Delta x\frac{\partial I_2(x,t)}{\partial t} + \cdots + L_{2n}\Delta x\frac{\partial I_n(x,t)}{\partial t} \\
&\;\;\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
V_n(x,t) &= V_n(x+\Delta x,t) + L_{n1}\Delta x\frac{\partial I_1(x,t)}{\partial t} + L_{n2}\Delta x\frac{\partial I_2(x,t)}{\partial t} + \cdots + L_{nn}\Delta x\frac{\partial I_n(x,t)}{\partial t}
\end{aligned} \qquad (4.2.1a)$$

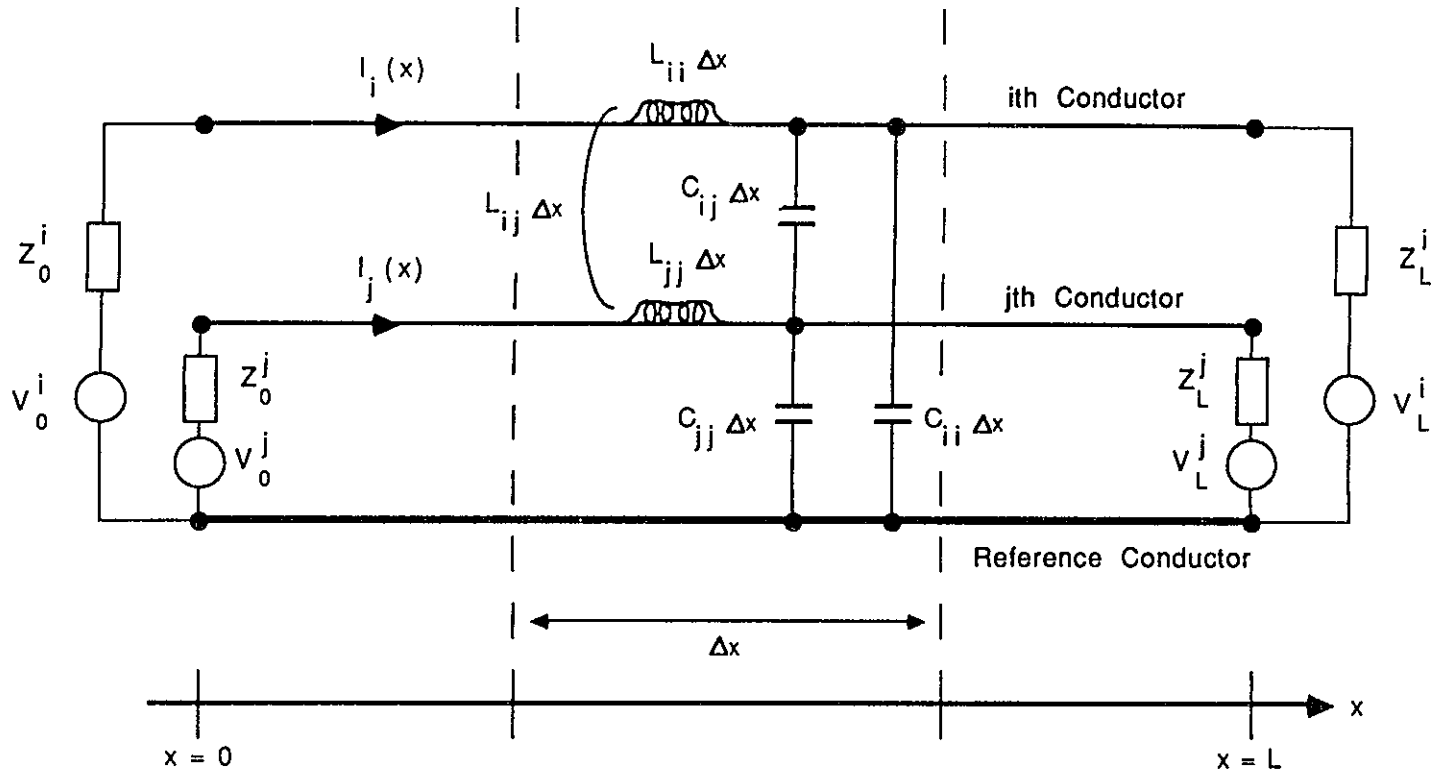


Figure 4-1: Distributed Parameter Multiconductor Transmission Line Model.

and

$$I_1(x,t) = I_1(x+\Delta x,t) + C'_{11}\Delta x\frac{\partial V_1(x,t)}{\partial t} - C'_{12}\Delta x\left\{\frac{\partial V_2(x,t)}{\partial t} - \frac{\partial V_1(x,t)}{\partial t}\right\} - \cdots - C'_{1n}\Delta x\left\{\frac{\partial V_n(x,t)}{\partial t} - \frac{\partial V_1(x,t)}{\partial t}\right\}$$

$$I_2(x,t) = I_2(x+\Delta x,t) - C'_{21}\Delta x\left\{\frac{\partial V_1(x,t)}{\partial t} - \frac{\partial V_2(x,t)}{\partial t}\right\} + C'_{22}\Delta x\frac{\partial V_2(x,t)}{\partial t} - \cdots - C'_{2n}\Delta x\left\{\frac{\partial V_n(x,t)}{\partial t} - \frac{\partial V_2(x,t)}{\partial t}\right\}$$

. . .

. . . (4.2.1b)

. . .

$$I_n(x,t) = I_n(x+\Delta x,t) - C'_{n1}\Delta x\left\{\frac{\partial V_1(x,t)}{\partial t} - \frac{\partial V_n(x,t)}{\partial t}\right\} - C'_{n2}\Delta x\left\{\frac{\partial V_2(x,t)}{\partial t} - \frac{\partial V_n(x,t)}{\partial t}\right\} - \cdots + C'_{nn}\Delta x\frac{\partial V_n(x,t)}{\partial t}$$

Alternatively, equation (4.2.1b) can be written in the form of (4.2.1a) as:

$$I_1(x,t) = I_1(x+\Delta x,t) + C_{11}\Delta x\frac{\partial V_1(x,t)}{\partial t} + C_{12}\Delta x\frac{\partial V_2(x,t)}{\partial t} + \cdots + C_{1n}\Delta x\frac{\partial V_n(x,t)}{\partial t}$$

$$I_2(x,t) = I_2(x+\Delta x,t) + C_{21}\Delta x\frac{\partial V_1(x,t)}{\partial t} + C_{22}\Delta x\frac{\partial V_2(x,t)}{\partial t} + \cdots + C_{2n}\Delta x\frac{\partial V_n(x,t)}{\partial t}$$

. . .

. . . (4.2.1c)

. . .

$$I_n(x,t) = I_n(x+\Delta x,t) + C_{n1}\Delta x\frac{\partial V_1(x,t)}{\partial t} + C_{n2}\Delta x\frac{\partial V_2(x,t)}{\partial t} + \cdots + C_{nn}\Delta x\frac{\partial V_n(x,t)}{\partial t}$$

if the per-unit length transmission line parameters are defined as:

$$C_{ii} = \sum_{k=1}^{n} C'_{ik}$$

(4.2.2)

$$C_{ij} = -C'_{ij}$$

In the limit as the length of the segment tends to zero (i.e. $\Delta x \rightarrow 0$):

$$\frac{\partial V_i(x,t)}{\partial x} = \lim_{\Delta x \rightarrow \infty}\left\{\frac{V_i(\Delta x + x) - V_i(x)}{\Delta x}\right\}$$

$$\frac{\partial I_i(x,t)}{\partial x} = \lim_{\Delta x \rightarrow \infty}\left\{\frac{I_i(\Delta x + x) - I_i(x)}{\Delta x}\right\} \tag{4.2.3}$$

equations (4.2.1a) and (4.2.1c) can be written in matrix notation as:

$$\frac{\partial \underline{V}(x,t)}{\partial x} = -[L]\frac{\partial \underline{I}(x,t)}{\partial t} \tag{4.2.4a}$$

$$\frac{\partial \underline{I}(x,t)}{\partial x} = -[C]\frac{\partial \underline{V}(x,t)}{\partial t} \tag{4.2.4b}$$

Equation (4.2.4) is a system of '2n' coupled partial differential equations which must be solved to determine the voltage and current distributions on the conductors. These equations are referred to as the Multiconductor Transmission Line equations. In (4.2.4) the $n \times n$ matrices $[C]$ and $[L]$ are the per-unit length multiconductor transmission line matrices whose numeric entries may be computed directly by using the Finite Element Method as outlined in Section 3.

4.3 Solution of the MCTLE's.

Solution of the multiconductor transmission line equations (4.2.4) can be accomplished in the time domain using, for example, the finite difference time domain method, but this technique has the disadvantage of increasing both the complexity and computation time required when compared with frequency domain solutions. The solution outlined in this Section is applied

to the multiconductor transmission line equations in the frequency domain by using modal analysis to uncouple the transmission line equations, followed by enforcement of the line termination networks as boundary conditions. This limits the solution to circuits with linear termination networks where simple terminal constraint equations can be written. As a last step the time domain solution is obtained using Fast Fourier Transform (FFT) Techniques.

Under steady-state excitation, with the line voltages and currents given by:

$$\underline{V}(x,t) = \underline{V}(x)\exp^{j\omega t} \tag{4.3.1a}$$

$$\underline{I}(x,t) = \underline{I}(x)\exp^{j\omega t} \tag{4.3.1b}$$

the multiconductor transmission line equations (4.2.4) are written as a system of '2n' coupled differential equations:

$$\frac{d\underline{V}(x)}{dx} = -j\omega[L]\underline{I}(x) \tag{4.3.2a}$$

$$\frac{d\underline{I}(x)}{dx} = -j\omega[C]\underline{V}(x) \tag{4.3.2b}$$

The voltage variable can be eliminated and a system of 'n' coupled differential equations in terms of the line currents is obtained:

$$\frac{d^2\underline{I}(x)}{dx^2} = -j\omega[C]\frac{d\underline{V}(x)}{dx} = -\omega^2[C]\,[L]\underline{I}(x) \tag{4.3.3}$$

To decouple these equations one can define a change of variables to transform the line currents in terms of their modal values [14]:

$$\underline{I}(x) = [T]\underline{I}_m(x) \tag{4.3.4}$$

As a result of this operation we obtain a set of 'n' coupled second-order differential equations given by:

$$\frac{d^2 I_m(x)}{dx^2} = -\omega^2 [T]^{-1} [C] [L] [T] \underline{I}_m(x) \tag{4.3.5}$$

These equations are uncoupled via finding the non-singular matrix $[T]$ which under the transformation:

$$[T]^{-1} [C] [L] [T] = \begin{pmatrix} \Lambda_{11}^2 & 0 & \dots & 0 \\ 0 & \Lambda_{22}^2 & \dots & 0 \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & & \cdot \\ 0 & 0 & \dots & \Lambda_{nn}^2 \end{pmatrix} = [\Lambda]^2 \tag{4.3.6}$$

results in a matrix $[\Lambda]^2$ which is diagonal. It can be shown that the columns of $[T]$ are composed of the eigenvectors of the product matrix $[C]$ $[L]$, [14].

This results in 'n' uncoupled equations with the general solution:

$$\underline{I}_m(x) = e^{-[\gamma]x} \underline{I}^+ - e^{+[\gamma]x} \underline{I}^- \tag{4.3.7}$$

where the propagation constants $[\gamma]$ are associated to the eigenvalues $[\Lambda]$ via:

$$[\gamma]^2 = -\omega^2[\Lambda]^2$$

$$[\gamma] = j\omega[\Lambda] \tag{4.3.8}$$

$$e^{\pm[\gamma]x} = \begin{pmatrix} e^{\pm[\gamma]x} & 0 & \dots & 0 \\ 0 & e^{\pm[\gamma]x} & \dots & 0 \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & & \cdot \\ 0 & 0 & \dots & e^{\pm[\gamma]x} \end{pmatrix}$$

The column vectors corresponding to the current and voltages on the lines can then be expressed in terms of the modal currents and are given by:

$$\underline{I}(x) = [T]\underline{I}_m(x) = [T]\{e^{-[\gamma]x}\underline{I}^+ - e^{+[\gamma]x}\underline{I}^-\} \tag{4.3.9a}$$

$$\underline{V}(x) = -\frac{1}{j\omega}[C]^{-1}\frac{d\underline{I}(x)}{dx} = [C]^{-1}[T][\Lambda]\{e^{-[\gamma]x}\underline{I}^+ + e^{+[\gamma]x}\underline{I}^-\} \tag{4.3.9b}$$

The unknown constants $\underline{I}^+$ and $\underline{I}^-$ are determined by enforcing the line termination conditions:

$$\underline{V}(0) = \underline{V}_0 - [Z_0]\underline{I}_0 \tag{4.3.10a}$$

$$\underline{V}(L) = \underline{V}_L + [Z_L]\underline{I}_L \tag{4.3.10b}$$

resulting in the final system of equations composed of '2n' equations.

The above approach has been implemented in a computer program which calculates the current and voltage distribution along the lines in the frequency domain and calculates the time domain values for the crosstalk currents and voltages using a Fast Fourier Transform algorithm. This is accomplished by multiplication in the frequency domain of the FFT of the excitation signal with the near and far-end voltage transfer functions computed using the multiconductor

transmission line solution. The time domain crosstalk waveforms are then computed by taking inverse FFT's. In this analysis solution of the multiconductor transmission line equations is required at each frequency but, when considering lossless lines and perfect dielectrics, modal decomposition is independent of frequency [14].

4.4 Program Verification.

The multiconductor transmission line model outlined in Section 4.2 was implemented in a FORTRAN program (i.e. "MCTLE.FOR, Appendix B1). For three parallel circular conductors (i.e. two conductors and one reference conductor) under the TEM approximation, explicit expressions for the near and far end crosstalk voltages can be obtained. These expressions, derived in reference [13], assume that the conductors are lossless with uniform cross-section along their length, and are immersed in a homogeneous, lossless, linear, isotropic medium. To verify the multiconductor transmission line program, results obtained for the near and far end crosstalk voltages from the program "MCTLE.FOR" were compared with results obtained using the explicit expressions derived in [13] and implemented in the Fortran program "A3CON.FOR" (Appendix B4) for the two coupled conductor configuration illustrated in Figure 4-2.

Pictured in Figure 4-2 are two perfect parallel circular conductors of radius '$r = 1mm$' and total length '$L = 10cm$', separated by a distance '$s = 1cm$', and located a distance '$h = 1cm$' above a perfect ground plane. This configuration is modelled with the distributed parameter transmission line model and is represented by two equivalent circuits, termed the generator and receptor circuit, which share the common reference conductor. The generator is excited at the near end ,$x = 0$, by a Thevenin equivalent circuit consisting of a steady state voltage source ,V_s, in series with an impedance Z_{0G}. The generator circuit is terminated at the far end ,$x = L$, in an impedance Z_{LG}. The receptor circuit, for which the near and far end crosstalk voltages will be

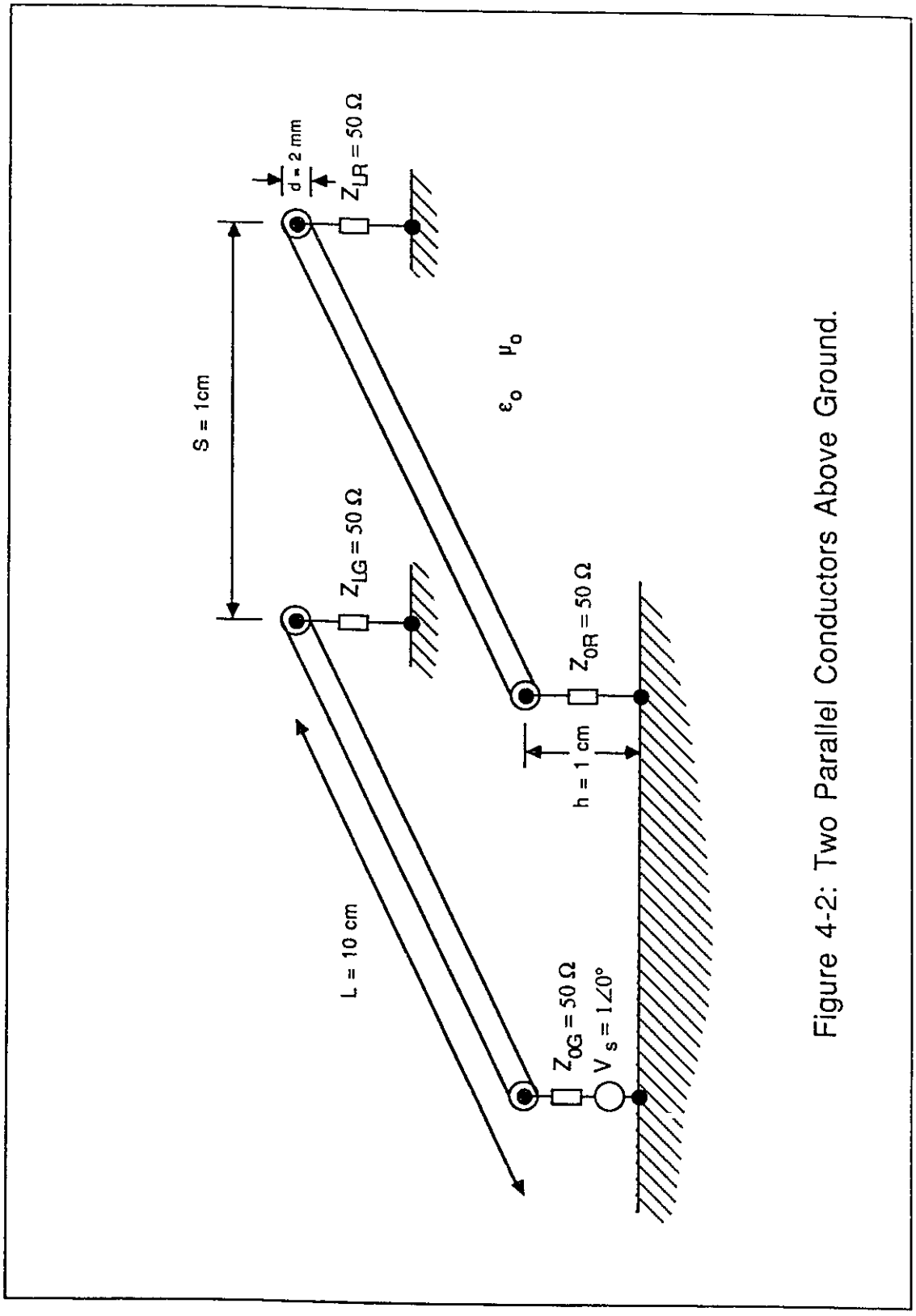


Figure 4-2: Two Parallel Conductors Above Ground.

compared, is terminated at $x = 0$ and $x = L$ by impedances Z_{0R} and Z_{LR} respectively. For this problem the excitation source is a normalized voltage source $V_s = 1\angle 0°$. All terminations are chosen as resistive impedances with Z_{0G}=Z_{0R}=Z_{LG}=Z_{LR}=50Ω and the medium surrounding the conductor as that of free space which is characterized by real permittivity, $\varepsilon = \varepsilon_o = 8.8542X\,10^{-12}F/m$, and permeability, $\mu = \mu_o = 4\pi X\,10^{-7}H/m$.

To determine the per-unit length parameters, approximate formulae for the self and mutual inductance for a system of '*n*' weakly coupled conductors (i.e. $h_i \gg r_i$ and $s_i \gg r_i$) above a perfect ground plane were used. These approximate formulae, derived in reference [17] using image theory, are given by:

$$L_{ii} = \frac{\mu}{2\pi}\ln\left[\frac{2h_i}{r_i}\right] \qquad (4.4.1a)$$

$$L_{ij} = \frac{\mu}{2\pi}\ln\left[\frac{d_{ij}}{d_{ij}^*}\right] \qquad (4.4.1b)$$

where h_i and r_i represent the height above ground and radius of the *ith* conductor, d_{ij} represents the center to center separation between the *ith* and *jth* conductors, and d_{ij}^* represents the center to center distance between the *ith* conductor and the image of the *jth* conductor in the ground plane. Evaluation of (4.4.1) for the geometry illustrated in Figure 4-2, yields the self and mutual per-unit length inductance parameters:

$$L_{11} = L_{22} = 0.5991\ \mu H/m \qquad (4.4.2a)$$

$$L_{12} = L_{21} = 0.1609\ \mu H/m \qquad (4.4.2b)$$

The per-unit length capacitance matrix is then evaluated using the relation given by (3.4.8) which yields:

Table 4-1

CROSSTALK PROGRAM RESULTS: THREE CONDUCTOR CONFIGURATION

	Analytic Solution		MCTLE Solution	
Frequency [MHz]	V_{NEXT} [mV]	V_{FEXT} [mV]	V_{NEXT} [mV]	V_{FEXT} [mV]
3.0	1.643∠88.80°	1.390∠–91.40°	1.643∠88.80°	1.390∠–91.40°
9.0	4.923∠86.39°	4.165∠–94.20°	4.923∠86.39°	4.165∠–94.20°
30.0	16.21∠78.04°	13.73∠–104.0°	16.21∠78.04°	13.73∠–104.0°
90.0	43.87∠55.51°	37.57∠–130.5°	43.87∠55.51°	37.57∠–130.5°
300.0	68.98∠4.308°	66.92∠162.4°	68.98∠4.308°	66.92∠162.4°
900.0	45.29∠10.49°	62.68∠69.97°	45.29∠10.51°	62.69∠69.98°

$$C_{11} = C_{22} = 20.01\ pF/m \qquad (4.4.3a)$$

$$C_{12} = C_{21} = -5.376\ pF/m \qquad (4.4.3b)$$

The results obtained using the multiconductor transmission line program, "MCTLE.FOR", and three coupled conductor program, "A3CON.FOR" for several frequencies are tabulated in Table 4-1. In Table 4-1, V_{NEXT} and V_{FEXT} represent predicted values for the near end, $x = 0$, and far end, $x = L$, crosstalk voltages on the receptor circuit across the termination impedances Z_{0R} and Z_{LR} respectively. It can be seen that the multiconductor solution conforms to the three conductor solution.

4.5 Experimental Results.

To verify the model experimentally a simple printed circuit board was constructed and measured in frequency and time domains. The circuit consists of five parallel conductors 24 cm long on a glass epoxy substrate (h~62 $mils$, ε_r~4.75) above a solid ground plane. The board dimensions are 24 cm by 16 cm. The conductors have equal widths and separations of 100 $mils$ and thickness 1.4 $mils$. The per-unit length parameters for this conductor configuration computed using the FEM are tabulated in Table 2. All lines are terminated in 50 Ω impedances except for conductor #3 at x=0, the excitation point. Frequency domain measurements were conducted using an HP 8505A Network Analyzer with an HP 8503A S-Parameter test set (Figure 4-3a). Port 1 of the Network Analyzer was connected directly to conductor #3 at x=0, the excitation point. Port 2 of the network analyzer was then connected to the line under test. All other lines were terminated with 50 Ω resistors. The transmission coefficient S_{21} was then measured to determine the crosstalk voltage transfer functions. Numerical and experimental results for the frequency domain transfer functions are illustrated in Figures 4-4 and 4-5.

Figure 4-3a: Frequency Domain Measurements

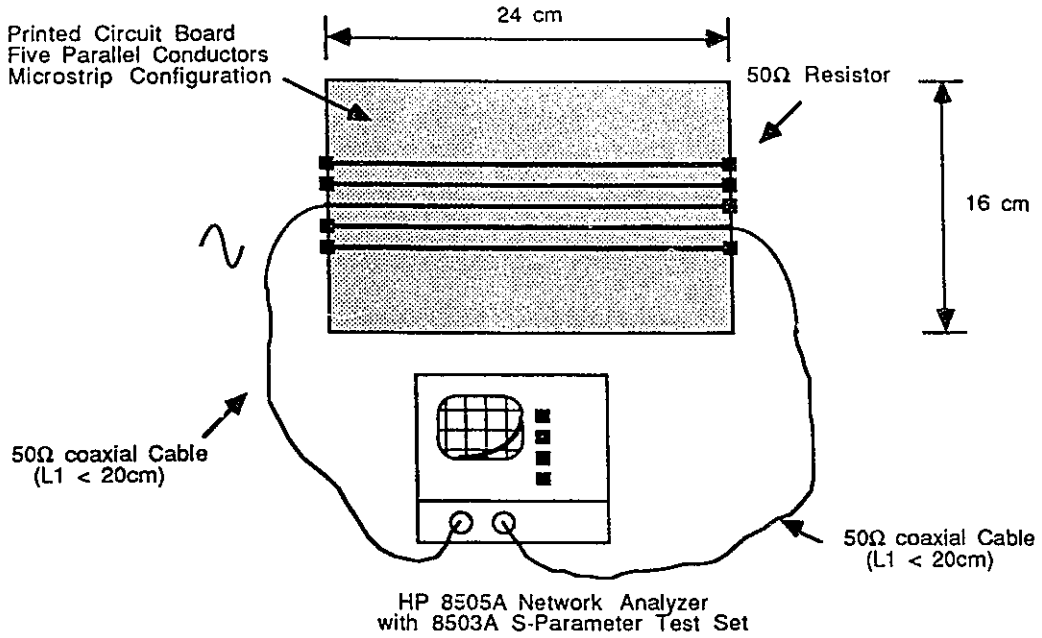


Figure 4-3b: Time Domain Measurements

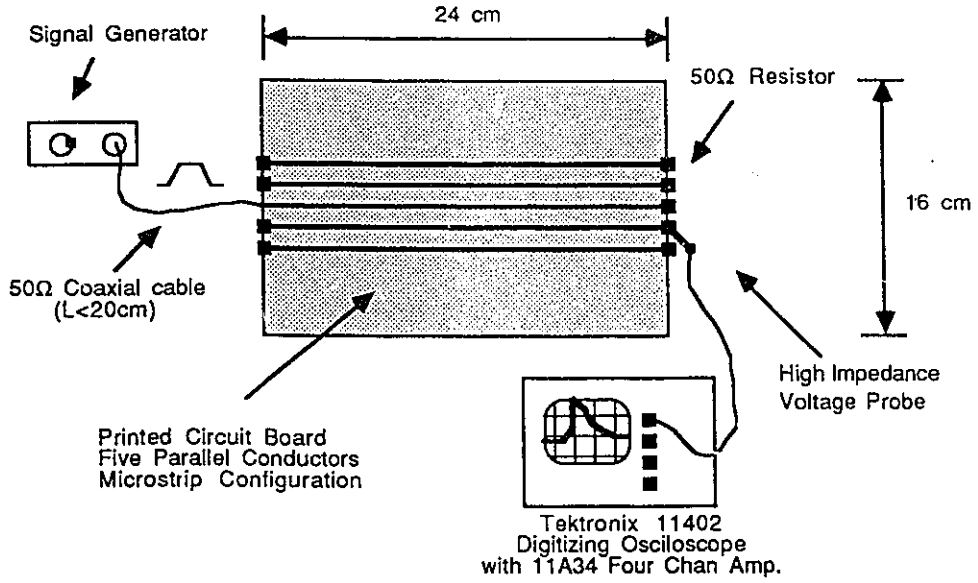


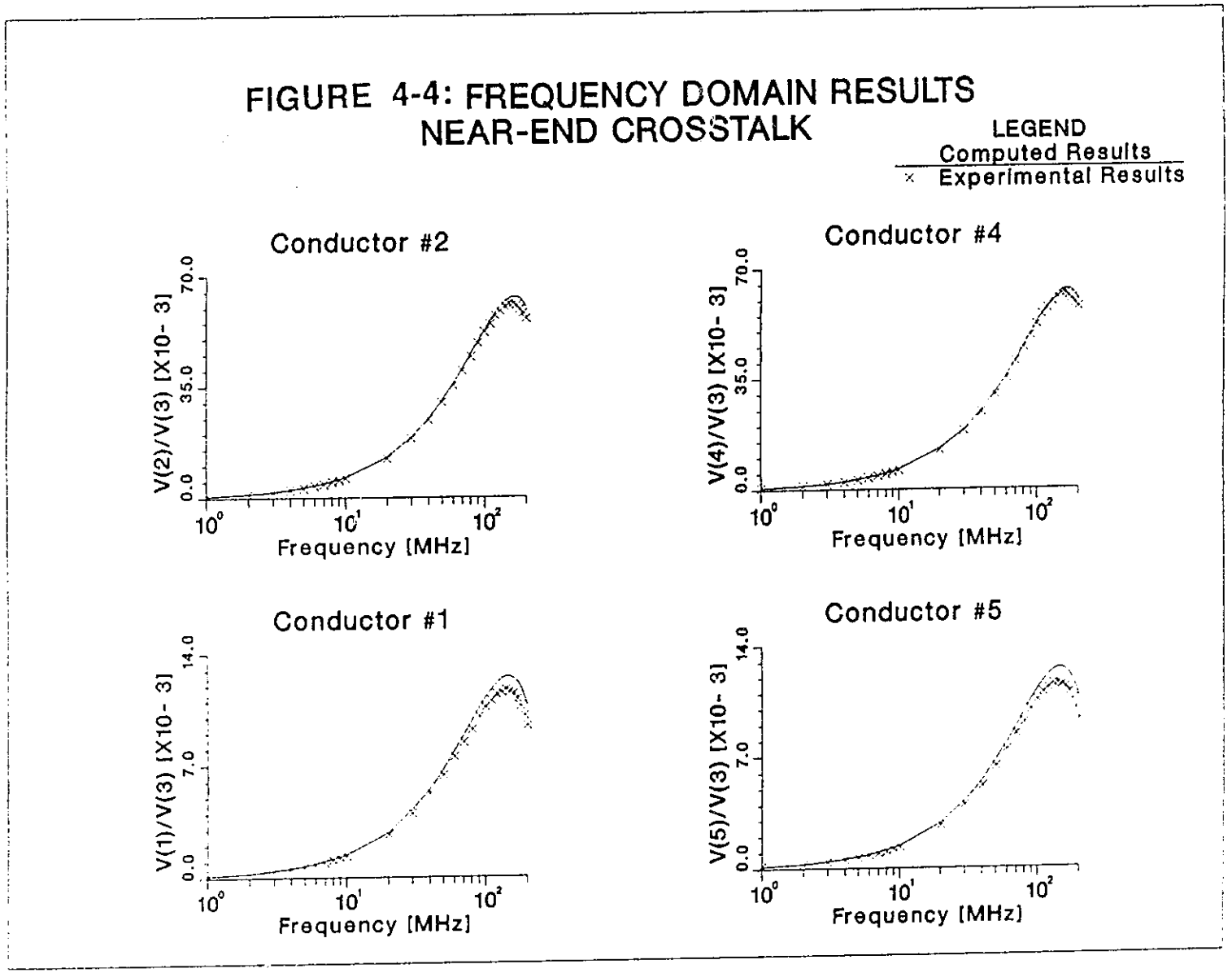
Figure 4-3: Experimental Measurement Set-up

Table 4-2

PER-UNIT LENGTH PARAMETERS: PRINTED CIRCUIT BOARD.

CAPACITANCE MATRIX (F/m)		INDUCTANCE MATRIX (H/m)	
C_{11}, C_{55}	0.1173E-09	L_{11}, L_{55}	0.3263E-06
$C_{12}, C_{21}, C_{45}, C_{54}$	-0.4306E-11	$L_{12}, L_{21}, L_{45}, L_{54}$	0.3225E-07
$C_{13}, C_{31}, C_{35}, C_{53}$	-0.4391E-12	$L_{13}, L_{31}, L_{35}, L_{53}$	0.8441E-08
$C_{14}, C_{41}, C_{25}, C_{52}$	-0.1847E-12	$L_{14}, L_{41}, L_{25}, L_{52}$	0.3625E-08
C_{15}, C_{51}	-0.1054E-12	L_{15}, L_{51}	0.2030E-08
C_{22}, C_{44}	0.1184E-09	L_{22}, L_{44}	0.3225E-06
$C_{23}, C_{32}, C_{34}, C_{43}$	-0.4281E-11	$L_{23}, L_{32}, L_{34}, L_{43}$	0.3173E-07
C_{24}, C_{42}	-0.4294E-12	L_{24}, L_{42}	0.8292E-08
C_{33}	0.118.4E-09	L_{33}	0.3224E-06

FIGURE 4-4: FREQUENCY DOMAIN RESULTS
NEAR-END CROSSTALK



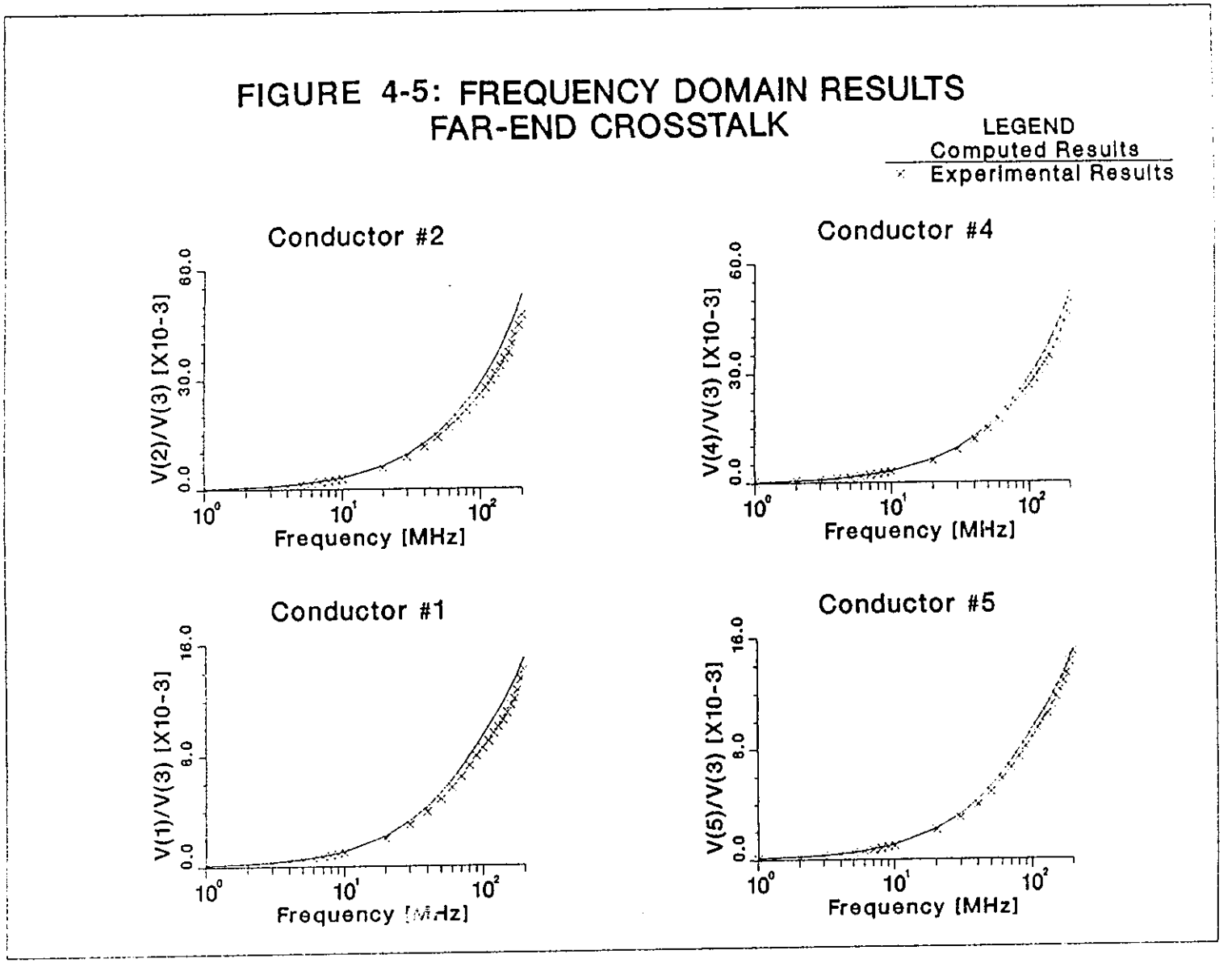


FIGURE 4-5: FREQUENCY DOMAIN RESULTS
FAR-END CROSSTALK
LEGEND
Computed Results
Experimental Results
Conductor #2
V(2)/V(3) [X10-3]
60.0
30.0
0.0
10^0
10^1
10^2
Frequency [MHz]
Conductor #4
V(4)/V(3) [X10-3]
60.0
30.0
0.0
10^0
10^1
10^2
Frequency [MHz]
Conductor #1
V(1)/V(3) [X10-3]
16.0
8.0
0.0
10^0
10^1
10^2
Frequency [MHz]
Conductor #5
V(5)/V(3) [X10-3]
16.0
8.0
0.0
10^0
10^1
10^2
Frequency [MHz]

Time domain measurements were made using a high impedance voltage probe and a Tektronics 11402 Digitizing Oscilloscope (Figure 4-3b). An HP waveform generator was used to excite conductor #3 at x=0 with a 50/50 duty cycle, 10MHz trapezoidal waveform with approximately equal rise/fall times of 3 ns. The amplitude of the waveform was adjusted to give a 0-5V amplitude signal at x=0. The measured points of the time domain waveform on conductor #3 at x=0 were used to generate the input waveform in computing the numerical results. The numerical and experimental excitation and crosstalk voltages on conductor#3, and on conductors #1,2,4 and 5 at x=0, V_{NEXT}, and at x=24 cm, V_{FEXT}, are illustrated in Figures 4-6 to 4-8.

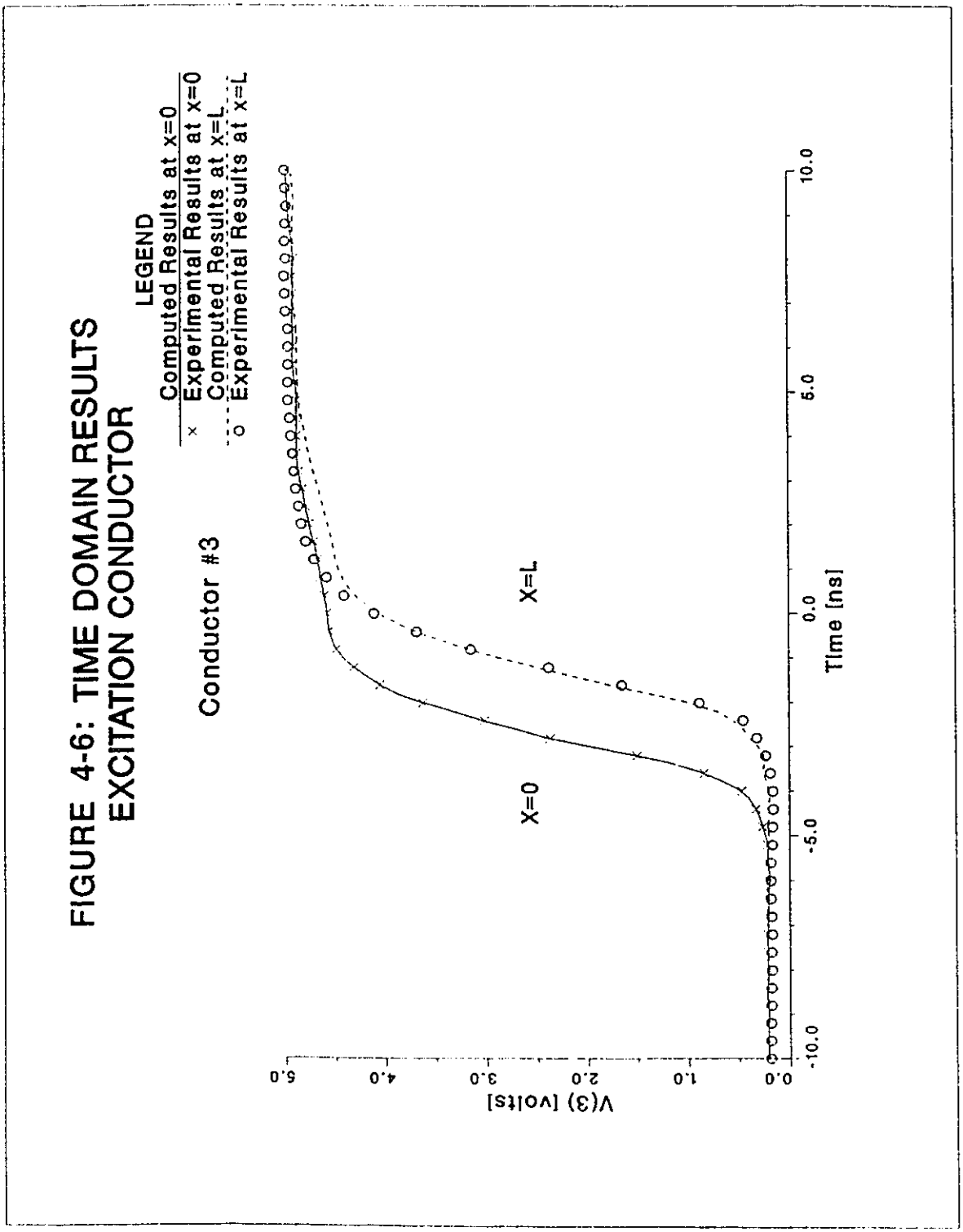


FIGURE 4-6: TIME DOMAIN RESULTS
EXCITATION CONDUCTOR
LEGEND
Computed Results at x=0
Experimental Results at x=0
Computed Results at x=L
Experimental Results at x=L
Conductor #3
X=0
X=L
V(3) [volts]
Time [ns]
0.0
1.0
2.0
3.0
4.0
5.0
-10.0
-5.0
0.0
5.0
10.0

FIGURE 4-7: TIME DOMAIN RESULTS NEAR-END CROSSTALK

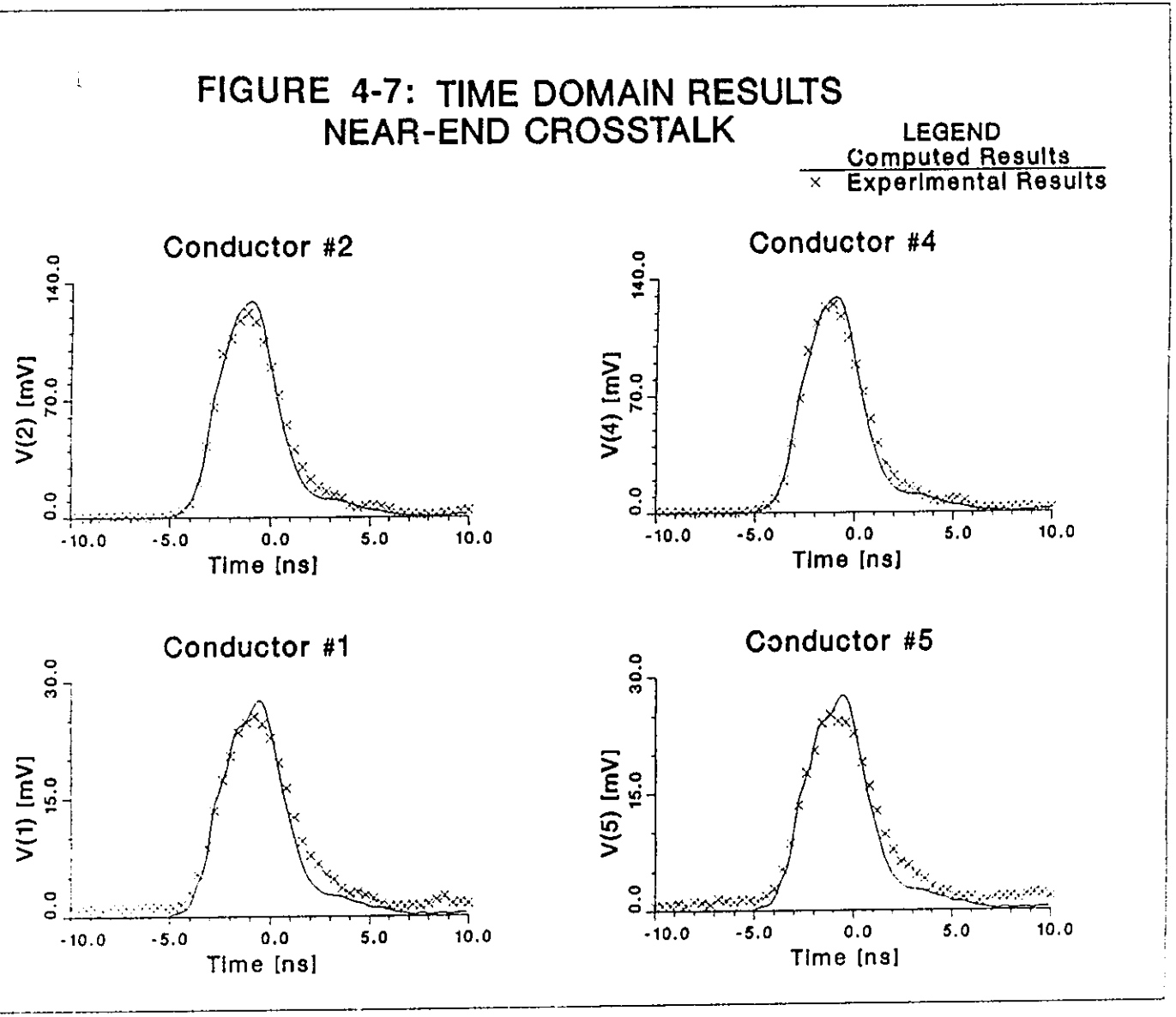
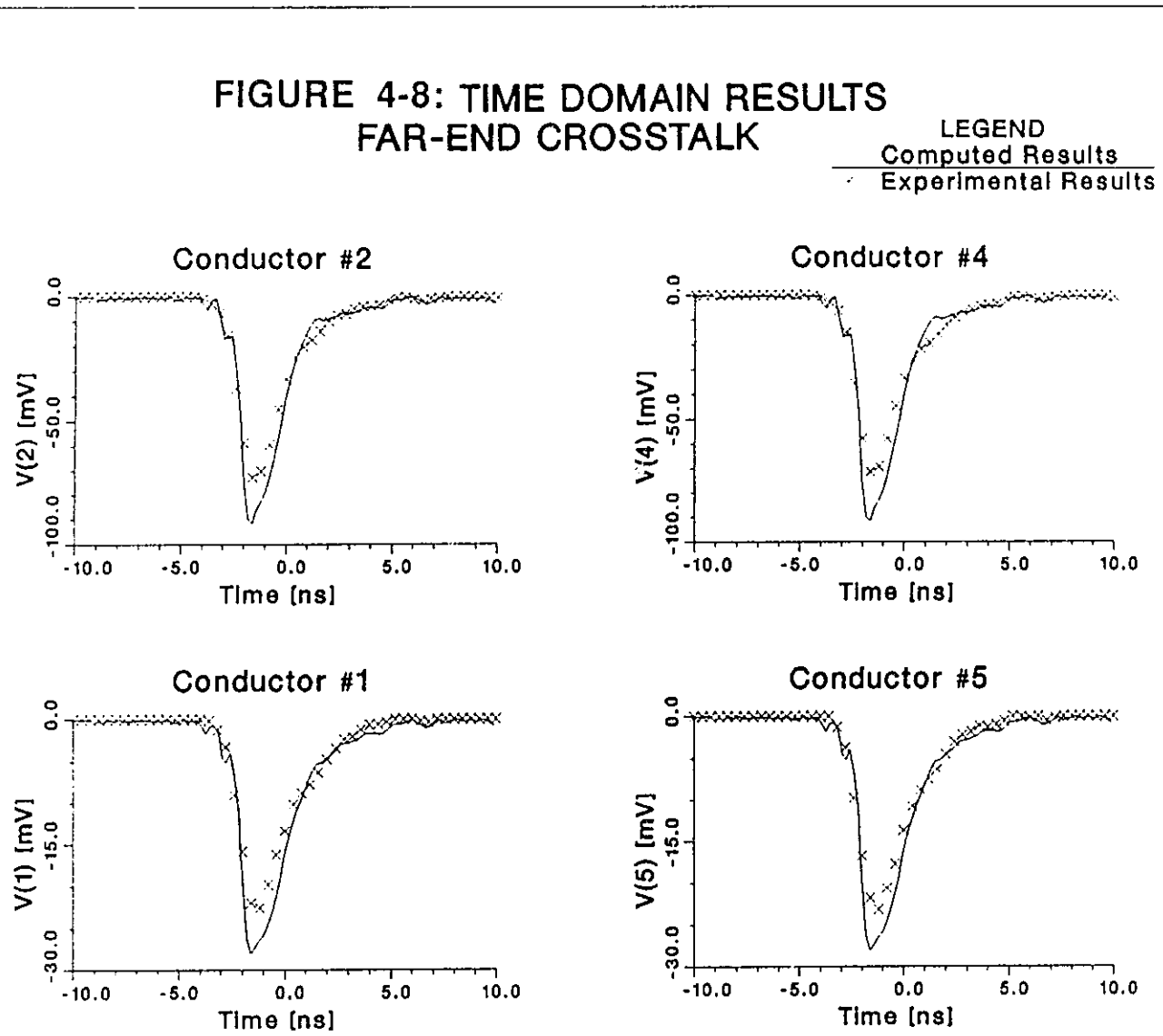


FIGURE 4-8: TIME DOMAIN RESULTS FAR-END CROSSTALK



4.6 Crosstalk Conclusions.

The results of this section show that the use of the distributed parameters computed using the Finite Element Method and modelled under the Quasi-TEM approximation with Multiconductor Transmission Line Theory can provide good results for prediction of coupling between conductors for the signal rise times and repetition rates associated with present day digital circuits. This is particularly useful as it enables variation of the coupling factors such as conductor width, separation, and length in conjunction with signal rise time and repetition rate to be analyzed for crosstalk performance. This type of analysis will aid in the EMC analysis of data and address busses as well as coupling between high risk, high frequency clock lines and adjacent parallel conductors. However for analysis of very high speed circuits, where the track widths and thickness of the substrate are no longer small compared to the wavelength, a full wave analysis which takes into account the hybrid mode of propagation is required.

5 RADIATED EMISSION ANALYSIS

5.1 Radiation from Printed Circuit Boards.

The characteristics of the radiated field from a printed circuit board depend on the source characteristics, the medium surrounding the source, and the distance and orientation of the observation point [2]. In close proximity to the source (i.e. at distances $d < \lambda/2\pi$) the wave impedance $Z = E/H$ is not constant and both the electric and magnetic fields decay at different rates with distance from the source. This region is termed the near field region and for the purposes of field measurements, both the electric and magnetic fields must be specified. In the far field ($d \gg \lambda/2\pi$), the radiated field propagates as a plane wave and the ratio E/H is a constant. For a plane wave propagating in free space, the wave impedance is given by $Z \sim 377\Omega$). In this region either the electric or magnetic field may be specified to define the radiated field. Radiated emission requirements for digital electronic equipment [1],[20] presently regulate the magnitude of the radiated electric field over the 30 – 1000 MHz frequency range. In this range the limits are specified in electric field units $dB\mu V/m$ at 3m, 10m or 30m measurement distances. For small electronic equipment, this puts radiated emission measurements at 3m in the far field region.

Radiation from a printed circuit board is often classified as differential mode and common mode radiation [2]. The major contribution to differential mode radiation from a PCB results from the signal currents which flow around the loops formed by the tracks and ground return. A major contributor to common mode radiation results from undesired voltage drops on the board which form a common mode potential difference between two areas on a printed circuit board. This potential difference often occurs between points on the board ground due to the

finite impedance (ground inductance) of the ground which may be significant at higher frequencies. Connection of a cable leaving the board at this point can give rise to common mode radiation which can be modelled like a monopole above ground. In both situations the use of a solid or gridded ground plane (i.e microstrip or stripline configuration will help reduce radiation. The ground aids in minimization of loop area by providing many signal return paths as well as minimization of ground inductance. In this section a radiation model is developed to compute the differential mode radiation produced from the signals propagating on tracks on a printed circuit board.

5.2 Radiation Model.

A simple radiation model is used to compute the radiated electric field produced by the board. This model computes the electric field produced by the steady state current distribution on the conductors [18]. The coordinate system is set up such that the ground plane is located at '$z = 0$' in the 'xy' plane. The system of parallel conductors of total length 'L' is centered about the 'yz' plane defined by '$x = 0$' and oriented parallel to the 'x' axis such that the conductors extend between '$x = -\frac{L}{2}$' and '$x = +\frac{L}{2}$'. Each conductor of length 'L', located a distance 'h' above the ground plane, and centered about the 'xz' plane defined by '$y = 0$' is modelled as an equivalent line filament of current. The conductors are replaced by 'x' oriented current filaments, with current distribution '$I(x')$' calculated in the crosstalk model, and the termination ends with a constant current distribution. The value of the current distribution on the termination ends is given by the near and far end values as:

$$I(z') = I\left(x' = \pm\frac{L}{2}\right) \tag{5.2.1}$$

and represents a good approximation provided $h \ll \lambda$..

Consider now a single conductor modelled in this manner as illustrated in Figure 5-1 in which the ground plane is replaced by the image current distribution below ground. The field produced by this current distribution may be calculated by:

$$\vec{E} = -j\omega\mu\vec{A} - \frac{j}{\omega\varepsilon}\nabla(\nabla \cdot \vec{A}) \tag{5.2.2}$$

where the magnetic vector potential is given by:

$$\vec{A} = A_x\hat{x} + A_z\hat{z} \tag{5.2.3}$$

which yields the following expressions for the electric field components:

$$\begin{aligned} E_x &= -j\omega\mu A_x - \frac{j}{\omega\varepsilon}\left[\frac{\partial^2 A_x}{\partial x^2} + \frac{\partial}{\partial x}\left(\frac{\partial A_z}{\partial z}\right)\right] \\ E_y &= -\frac{j}{\omega\varepsilon}\left[\frac{\partial}{\partial y}\left(\frac{\partial A_x}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial A_z}{\partial z}\right)\right] \\ E_z &= -j\omega\mu A_z - \frac{j}{\omega\varepsilon}\left[\frac{\partial}{\partial z}\left(\frac{\partial A_x}{\partial x}\right) + \frac{\partial^2 A_z}{\partial z^2}\right] \end{aligned} \tag{5.2.4}$$

The contribution to the 'x' component of the magnetic vector potential from the 'x' oriented current filament located at '$z = +h$' and its image current distribution located at '$z = -h$' is given by:

$$A_x = \frac{1}{4\pi}\int_{-\frac{L}{2}}^{+\frac{L}{2}} I(x')\frac{\exp(-j\beta R_{x_1})}{R_{x_1}}dx' - \frac{1}{4\pi}\int_{-\frac{L}{2}}^{+\frac{L}{2}} I(x')\frac{\exp(-j\beta R_{x_2})}{R_{x_2}}dx' \tag{5.2.5a}$$

and the contribution to the 'z' component of the magnetic vector potential from the termination ends located at $x = -\frac{L}{2}$ and $x = +\frac{L}{2}$ by:

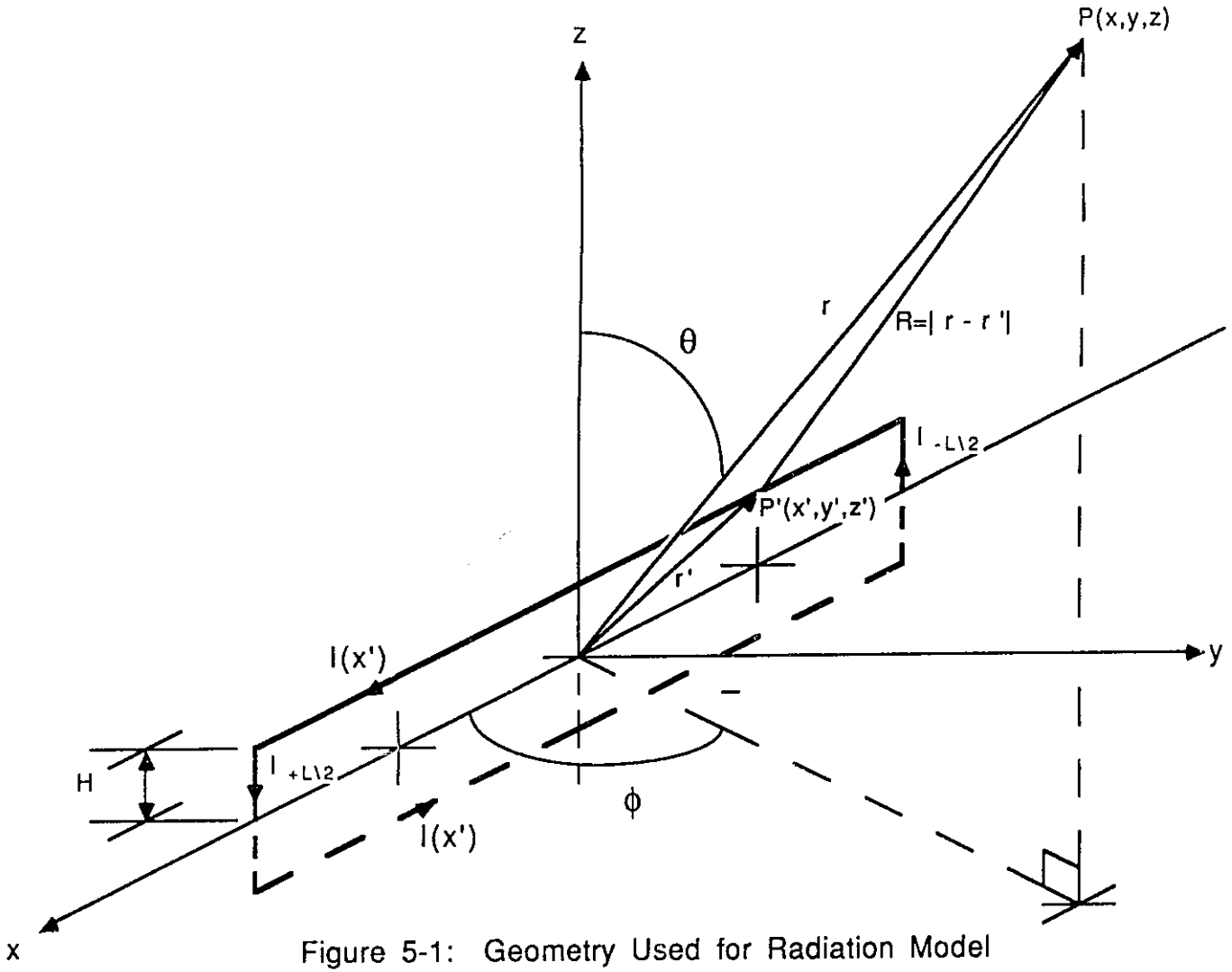


Figure 5-1: Geometry Used for Radiation Model

$$A_z = \frac{1}{4\pi} I\left(x' = -\frac{L}{2}\right) \int_{-h}^{+h} \frac{\exp(-j\beta R_{z_1})}{R_{z_1}} dz'$$

$$-\frac{1}{4\pi} I\left(x' = +\frac{L}{2}\right) \int_{-h}^{+h} \frac{\exp(-j\beta R_{z_2})}{R_{z_2}} dz' \qquad (5.2.5b)$$

In equation (5.2.5) the quantities $R_{x_1}, R_{x_2}, R_{z_1}, R_{z_2}$ represent the distance between a source point $P'(x', y', z')$ and the field computation point $P(x, y, z)$ and are given by:

$$R_{x_1} = \sqrt{(x - x')^2 + (y - y')^2 + (z - (+h))^2}$$

$$R_{x_2} = \sqrt{(x - x')^2 + (y - y')^2 + (z - (-h))^2} \qquad (5.2.6)$$

$$R_{z_1} = \sqrt{\left(x - \left(-\frac{L}{2}\right)\right)^2 + (y - y')^2 + (z - z')^2}$$

$$R_{z_2} = \sqrt{\left(x - \left(+\frac{L}{2}\right)\right)^2 + (y - y')^2 + (z - z')^2}$$

By dividing the line along the 'x' and 'z' directions into electrically small segments, these integrals may be approximated by a summation over which the current distribution $I(x')$ is assumed constant over each segment. If '*nsx*' and '*nsz*' represent the number of segments a line is divided along its length and height respectively, then the length of each segment is given by:

$$\Delta x = \frac{L}{nsx}$$

(5.2.7)

$$\Delta z = \frac{h}{nsz}$$

Under this approximation the integrals for computing the components of the vector magnetic potential given in (5.2.5) may be written as:

$$A_x = \frac{\Delta x}{4\pi} \sum_{n=1}^{nsx} I(x_n) \left[\frac{\exp(-j\beta R_{x_1})}{R_{x_1}} - \frac{\exp(-j\beta R_{x_2})}{R_{x_2}} \right]$$

(5.2.8)

$$A_z = \frac{\Delta z}{4\pi} \sum_{n=1}^{nsz} \left[I\left(x' = -\frac{L}{2}\right) \frac{\exp(-j\beta R_{z_1})}{R_{z_1}} - I\left(x' = +\frac{L}{2}\right) \frac{\exp(-j\beta R_{z_2})}{R_{z_2}} \right]$$

Evaluation of the spatial derivatives of equation (5.2.8) and collecting terms yields contributions from the scalar potential term in (5.2.2) to the electric field components as:

$$\frac{\partial^2 A_x}{\partial x^2} = \frac{\beta^2}{4\pi} \Delta x \sum_{n=1}^{nsx} I(x_n) \left\{ \left[\frac{e^{-j\beta R_{x_1}}}{R_{x_1}} \right] \left[\frac{1}{j\beta R_{x_1}} - \frac{1}{\beta^2 R_{x_1}} - \frac{(x-x')^2}{R_{x_1}^2} - \frac{3(x-x')^2}{j\beta R_{x_1}^3} + \frac{3(x-x')^2}{\beta^2 R_{x_1}^4} \right] \right.$$
$$\left. - \left[\frac{e^{-j\beta R_{x_2}}}{R_{x_2}} \right] \left[\frac{1}{j\beta R_{x_2}} - \frac{1}{\beta^2 R_{x_2}} - \frac{(x-x')^2}{R_{x_2}^2} - \frac{3(x-x')^2}{j\beta R_{x_2}^3} + \frac{3(x-x')^2}{\beta^2 R_{x_2}^4} \right] \right\}$$

(5.2.9a)

$$\frac{\partial}{\partial x}\left(\frac{\partial A_z}{\partial z}\right) = \frac{\beta^2}{4\pi} \Delta z \sum_{n=1}^{nsz} (z-z') \left\{ I\left(x = -\frac{L}{2}\right)(x-x') \left[\frac{e^{-j\beta R_{z_1}}}{R_{z_1}} \right] \left[-\frac{1}{R_{z_1}^2} - \frac{3}{j\beta R_{z_1}^3} + \frac{3}{\beta^2 R_{z_1}^4} \right] \right.$$
$$\left. - I\left(x = +\frac{L}{2}\right)(x-x') \left[\frac{e^{-j\beta R_{z_2}}}{R_{z_2}} \right] \left[-\frac{1}{R_{z_2}^2} - \frac{3}{j\beta R_{z_2}^3} + \frac{3}{\beta^2 R_{z_2}^4} \right] \right\}$$

$$\frac{\partial}{\partial y}\left(\frac{\partial A_x}{\partial x}\right)=\frac{\beta^2}{4\pi}(y-y')\Delta x\sum_{n=1}^{nsx}I(x_n)(x-x')\left\{\left[\frac{e^{-j\beta R_{x_1}}}{R_{x_1}}\right]\left[-\frac{1}{R_{x_1}^2}-\frac{3}{j\beta R_{x_1}^3}+\frac{3}{\beta^2 R_{x_1}^4}\right]\right.$$

$$\left.-\left[\frac{e^{-j\beta R_{x_2}}}{R_{x_2}}\right]\left[-\frac{1}{R_{x_2}^2}-\frac{3}{j\beta R_{x_2}^3}+\frac{3}{\beta^2 R_{x_2}^4}\right]\right\} \tag{5.2.9b}$$

$$\frac{\partial}{\partial y}\left(\frac{\partial A_z}{\partial z}\right)=\frac{\beta^2}{4\pi}(y-y')\Delta z\sum_{n=1}^{nsz}(z-z')\left\{I\left(x=-\frac{L}{2}\right)\left[\frac{e^{-j\beta R_{z_1}}}{R_{z_1}}\right]\left[-\frac{1}{R_{z_2}^2}-\frac{3}{j\beta R_{z_2}^3}+\frac{3}{\beta^2 R_{z_2}^4}\right]\right.$$

$$\left.-I\left(x=+\frac{L}{2}\right)\left[\frac{e^{-j\beta R_{z_2}}}{R_{z_2}}\right]\left[-\frac{1}{R_{z_2}^2}-\frac{3}{j\beta R_{z_2}^3}+\frac{3}{\beta^2 R_{z_2}^4}\right]\right\}$$

$$\frac{\partial^2 A_z}{\partial z^2}=\frac{\beta^2}{4\pi}\Delta z\sum_{n=1}^{nsz}\left\{I\left(x=-\frac{L}{2}\right)\left[\frac{e^{-j\beta R_{z_1}}}{R_{z_1}}\right]\left[\frac{1}{j\beta R_{z_1}}-\frac{1}{\beta^2 R_{z_1}}-\frac{(z-z')^2}{R_{z_1}^2}-\frac{3(z-z')^2}{j\beta R_{z_1}^3}+\frac{3(z-z')^2}{\beta^2 R_{z_1}^4}\right]\right.$$

$$\left.-I\left(x=+\frac{L}{2}\right)\left[\frac{e^{-j\beta R_{z_2}}}{R_{z_2}}\right]\left[\frac{1}{j\beta R_{z_2}}-\frac{1}{\beta^2 R_{z_2}}-\frac{(z-z')^2}{R_{z_2}^2}-\frac{3(z-z')^2}{j\beta R_{z_2}^3}+\frac{3(z-z')^2}{\beta^2 R_{z_2}^4}\right]\right\}$$

(5.2.9c)

$$\frac{\partial}{\partial z}\left(\frac{\partial A_x}{\partial x}\right)=\frac{\beta^2}{4\pi}(z-z')\Delta x\sum_{n=1}^{nsx}I(x_n)(x-x')\left\{\left[\frac{e^{-j\beta R_{x_1}}}{R_{x_1}}\right]\left[-\frac{1}{R_{x_1}^2}-\frac{3}{j\beta R_{x_1}^3}+\frac{3}{\beta^2 R_{x_1}^4}\right]\right.$$

$$\left.-\left[\frac{e^{-j\beta R_{x_2}}}{R_{x_2}}\right]\left[-\frac{1}{R_{x_2}^2}-\frac{3}{j\beta R_{x_2}^3}+\frac{3}{\beta^2 R_{x_2}^4}\right]\right\}$$

For computing the radiated field from a typical printed circuit board, where a dielectric is inherently present, a far-field approximation is necessary thus neglecting the scalar potential term in (5.2.2).

Using equations (5.2.8) and (5.2.9) the radiated electric field produced by the current distribution for a single conductor can be computed at any point $P(x,y,z)$ where $x \neq x'$, $y \neq y'$, and $z \neq z'$ using equation (5.2.4). For multiconductor configurations the total electric field is computed at the point $P(x,y,z)$ by a superposition of the field components radiated by the current distributions on each conductor. In equations (5.2.8) and (5.2.9) the constant value assigned to each 'x' oriented current segment is evaluated in the crosstalk model at the midpoint of the source segment at the point:

$$I(n) = I(x_n) = I\left(x = \left(n - \frac{1}{2}\right)\Delta x - \frac{L}{2}\right) \tag{5.2.10}$$

and for the each 'z' oriented segment, the approximation given in equation (5.2.1) is used. Translation of the electric field components at the point $P(x,y,z)$ from the rectangular to spherical coordinate system can be determined using:

$$\begin{aligned} E_r &= E_x \sin(\theta)\cos(\phi) + E_y \sin(\theta)\sin(\phi) + E_z \cos(\theta) \\ E_\theta &= E_x \cos(\theta)\cos(\phi) + E_y \cos(\theta)\sin(\phi) - E_z \sin(\theta) \\ E_\phi &= -E_x \sin(\theta) \end{aligned} \tag{5.2.11}$$

5.3 Radiation Results.

The radiation model outlined in Section 5.2 was used to compute the radiated electric field strength for five lossless conductors above a perfect ground plane. The geometry, termination impedances, and excitation are illustrated in Figure 5-2 and the per-unit length parameters used for computation of the current distribution on the conductors are tabulated in Table 5-1.

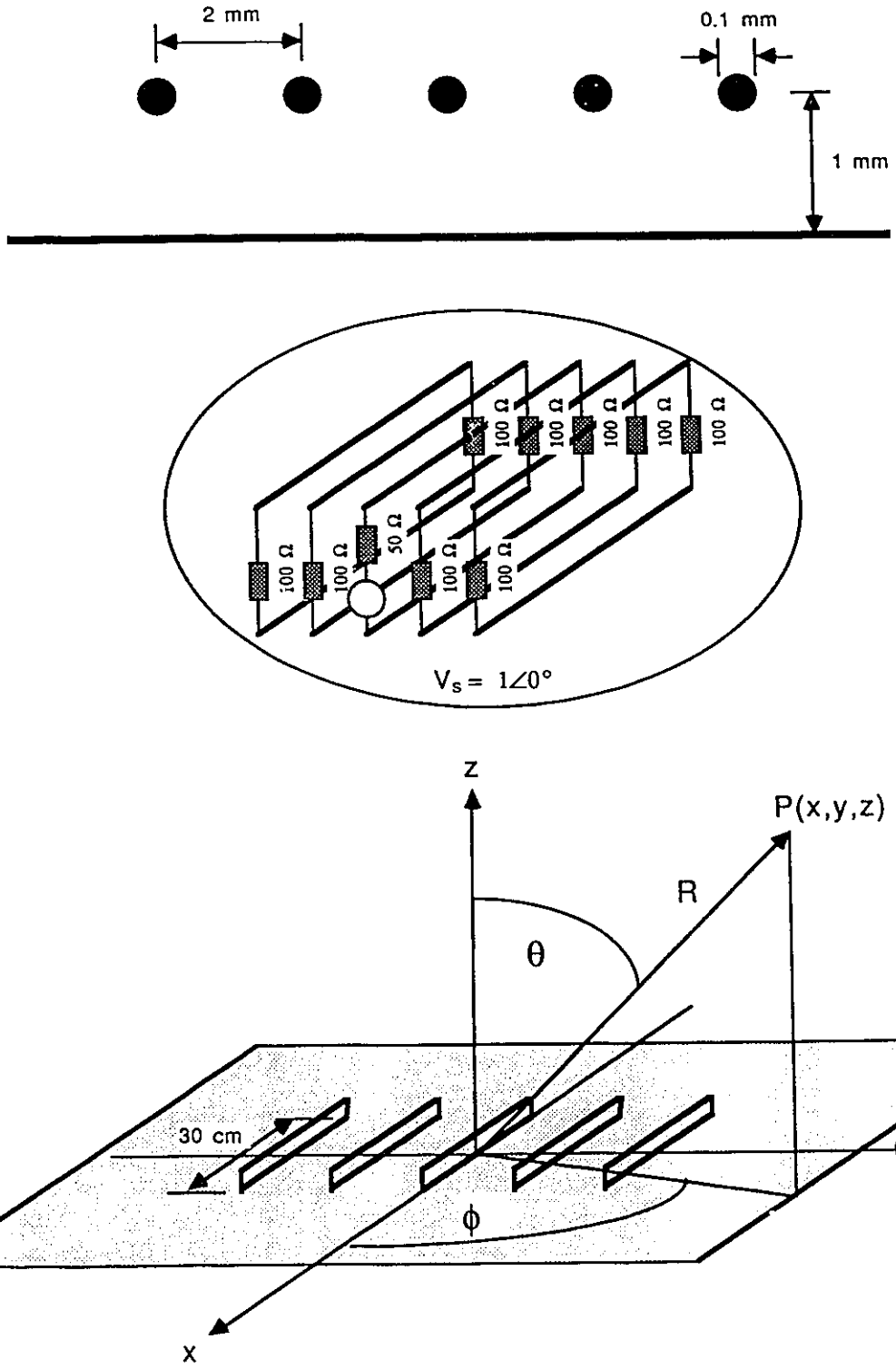


Figure 5-2: Multiconductor Configuation for Radiation Comparison.

Table 5-1

PER-UNIT LENGTH PARAMETERS: RADIATION EXAMPLE.

CAPACITANCE MATRIX (F/m)		INDUCTANCE MATRIX (H/m)	
C_{11}, C_{55}	0.1522E-10	L_{11}, L_{55}	0.7378E-06
$C_{12}, C_{21}, C_{45}, C_{54}$	-0.1396E-11	$L_{12}, L_{21}, L_{45}, L_{54}$	0.6931E-07
$C_{13}, C_{31}, C_{35}, C_{53}$	-0.3139E-12	$L_{13}, L_{31}, L_{35}, L_{53}$	0.2231E-07
$C_{14}, C_{41}, C_{25}, C_{52}$	-0.1379E-12	$L_{14}, L_{41}, L_{25}, L_{52}$	0.1054E-07
C_{15}, C_{51}	-0.8272E-13	L_{15}, L_{51}	0.6062E-08
C_{22}, C_{44}	0.1535E-10	L_{22}, L_{44}	0.7378E-06
$C_{23}, C_{32}, C_{34}, C_{43}$	-0.1367E-11	$L_{23}, L_{32}, L_{34}, L_{43}$	0.6931E-07
C_{24}, C_{42}	-0.3030E-12	L_{24}, L_{42}	0.2231E-07
C_{33}	0.1536E-10	L_{33}	0.7378E-06

Results obtained for this multiconductor configuration are plotted in Figures 5-3a, 5-3b, 5-3c, 5-3d and 5-4 at a distance three meters from the center of the coordinate system. Figure 5-3a to 5-3d plots the variation of the magnitude of the electric field and its components with angle theta in the vertical plane defined by $\phi = 0°$, $\phi = 45°$, and $\phi = 90°$ for a steady state excitation at 30 MHz. Figure 5-4 plots the variation of the magnitude of the electric field as a function of theta in the fixed vertical plane defined by $\phi = 45°$ for the excitation frequencies 30 MHz, 90 MHz, 300 MHz, and 900 MHz. These results are compared by modelling the identical problem using the Numerical Electromagnetics Code (NEC2D) package.

The NEC computer code allows modelling of wire or surfaces in free space or above a ground plane and numerically solves integral equations for the induced currents due to an applied voltage source or incident plane wave. For application to this problem, a voltage excitation was used, and NEC solves for the induced current distribution on the conductors, replaces the ground plane with the image current distribution, and uses a thin wire approximation to evaluate the near electric fields [19]. The input computer code for modelling this problem using NEC is provided in Appendix B6.

FIGURE 5-3a: RADIATION RESULTS
FREQUENCY = 30.0 MHz , R = 3.0 m

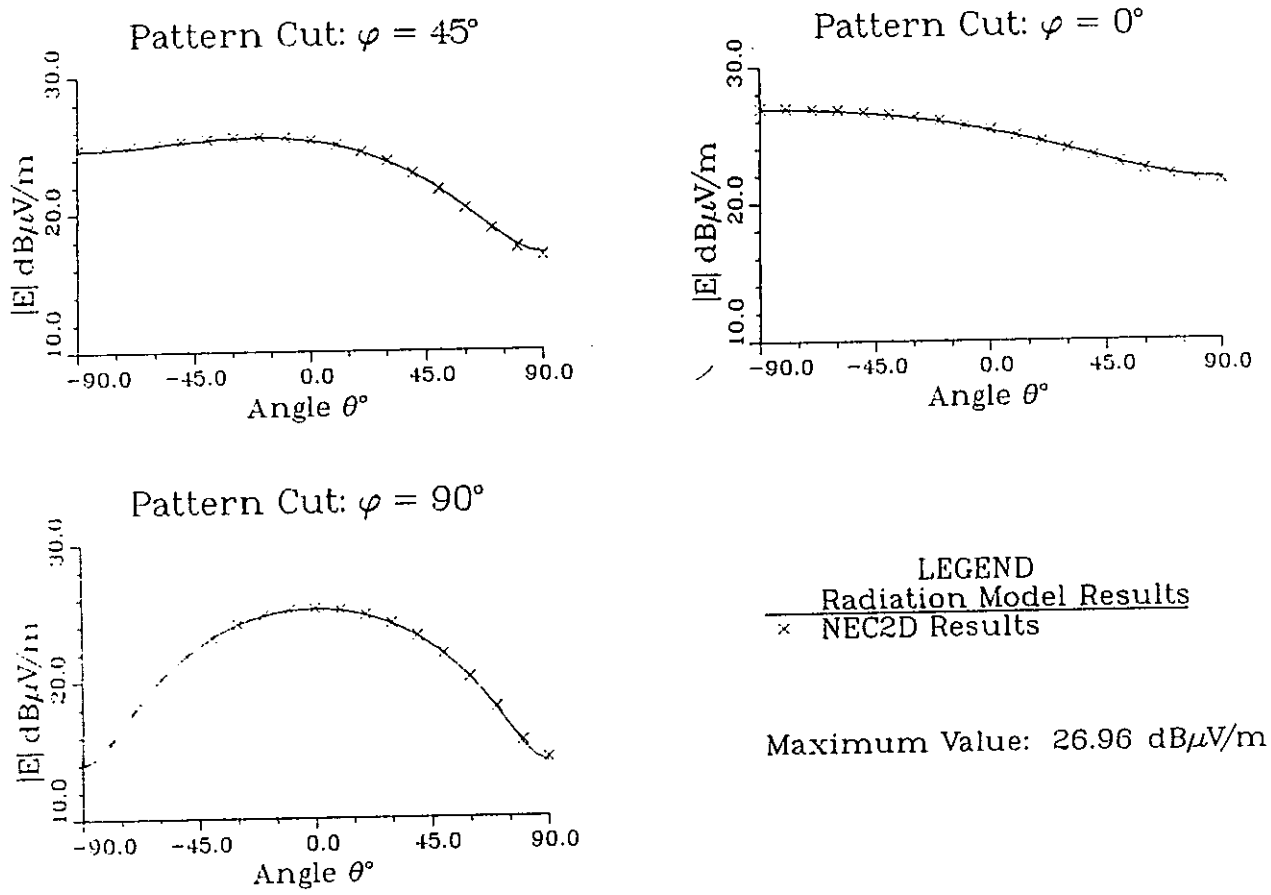
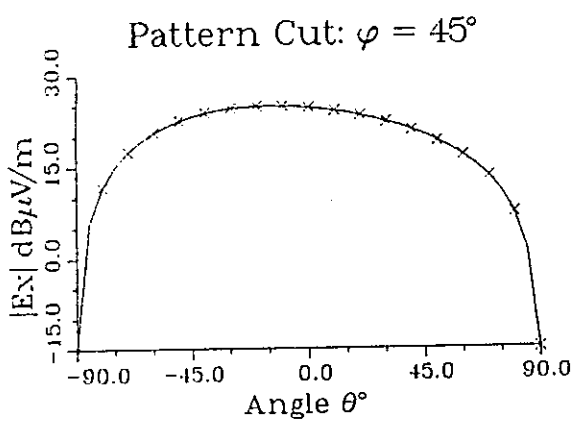
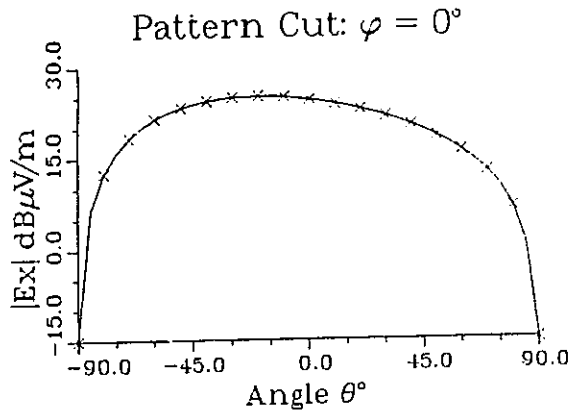
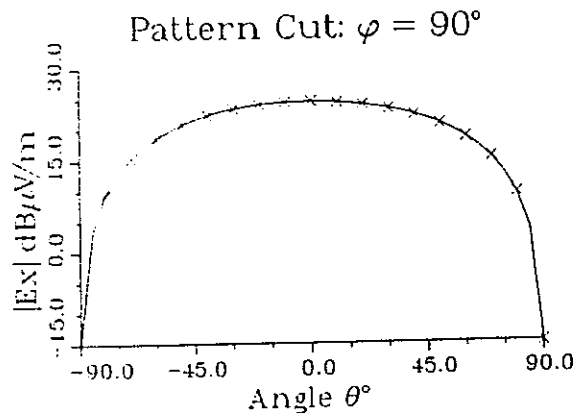


FIGURE 5-3b: RADIATION RESULTS
FREQUENCY = 30.0 MHz , R = 3.0 m





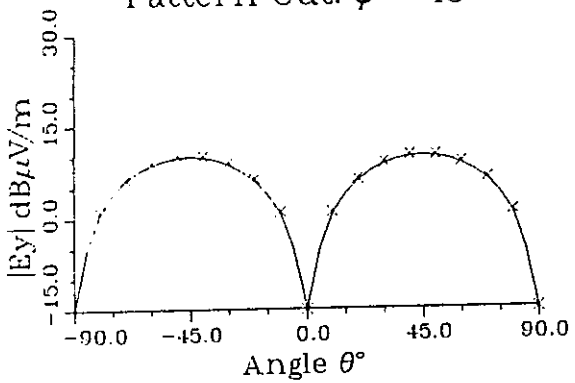


LEGEND
Radiation Model Results
× NEC2D Results

Maximum Value: 25.26 dBμV/m

FIGURE 5-3c: RADIATION RESULTS
FREQUENCY = 30.0 MHz , R = 3.0 m

Pattern Cut: $\varphi = 45°$



LEGEND
Radiation Model Results
× NEC2D Results

Maximum Value: 13.31 dBμV/m

Pattern Cut: $\varphi = 90°$

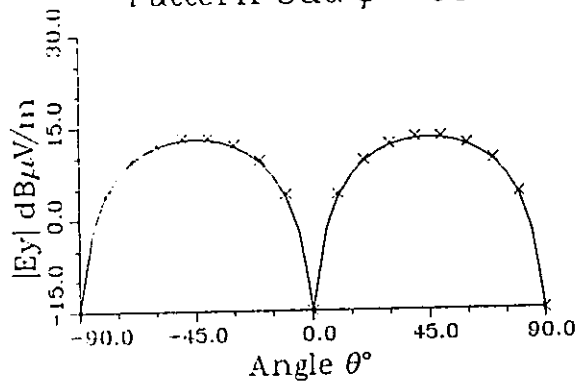
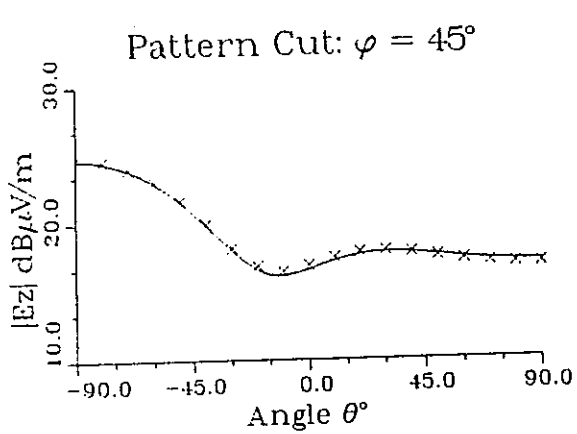
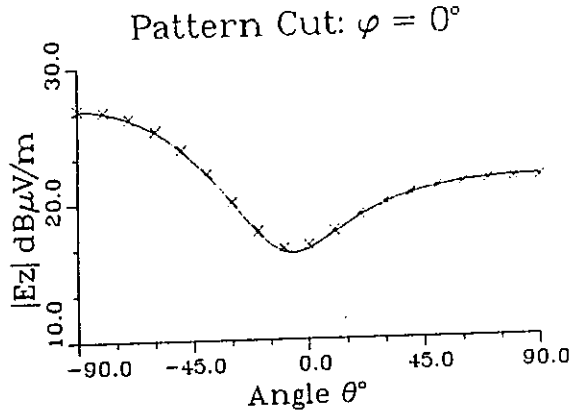
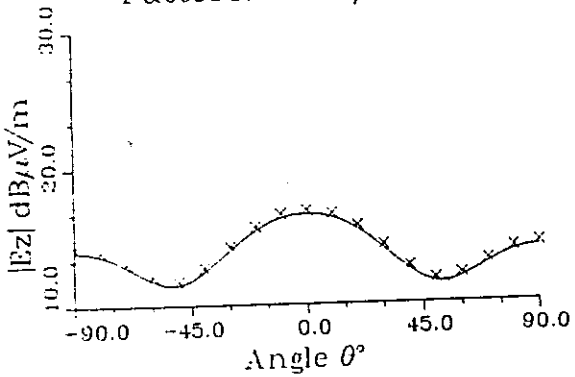


FIGURE 5-3d: RADIATION RESULTS
FREQUENCY = 30.0 MHz , R = 3.0 m







LEGEND
Radiation Model Results
× NEC2D Results

Maximum Value: 26.96 dBμV/m

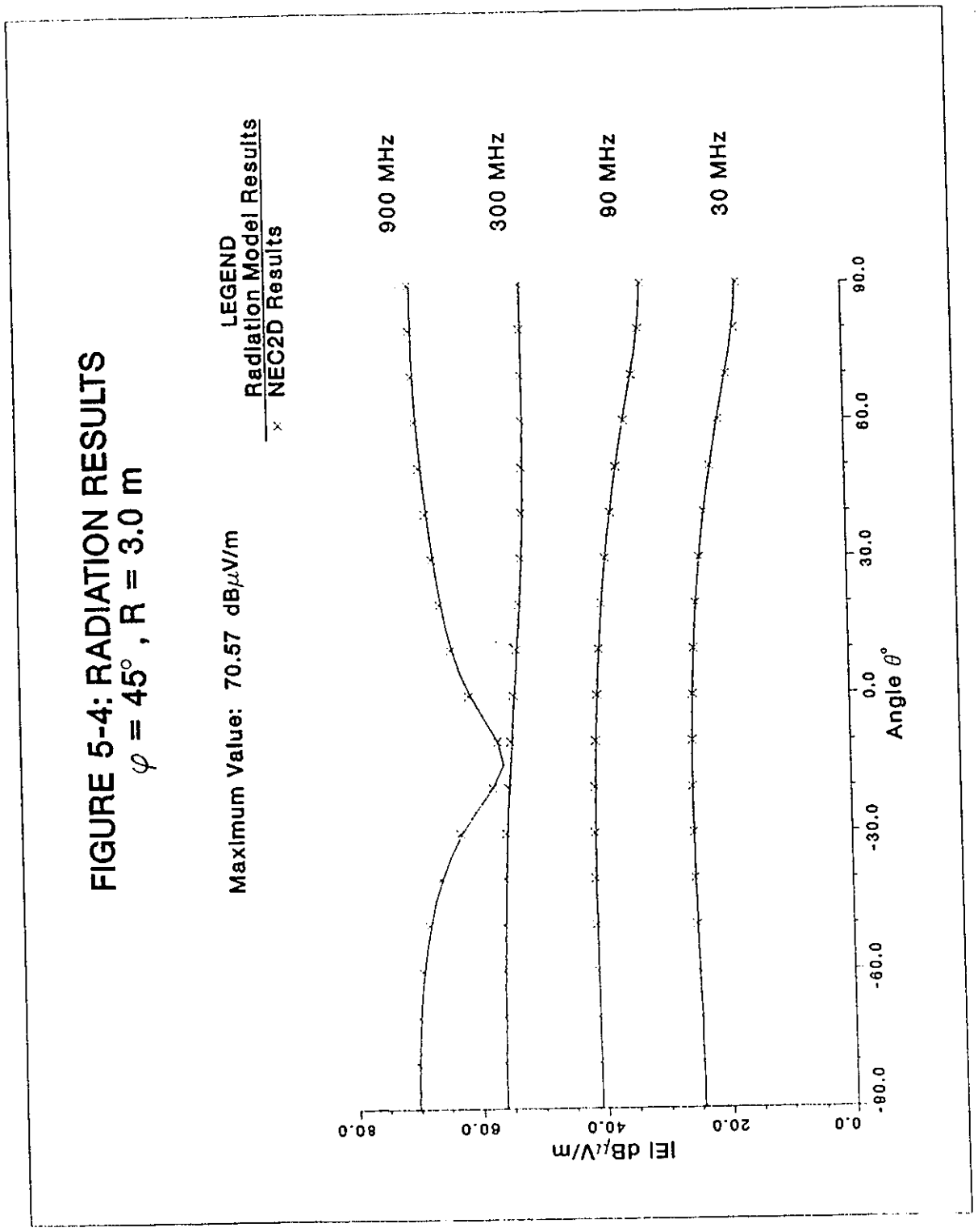


FIGURE 5-4: RADIATION RESULTS
φ = 45°, R = 3.0 m
Maximum Value: 70.57 dBμV/m
LEGEND
Radiation Model Results
× NEC2D Results
900 MHz
300 MHz
90 MHz
30 MHz
|E| dBμV/m
80.0
60.0
40.0
20.0
0.0
-90.0
-60.0
-30.0
0.0
30.0
60.0
90.0
Angle θ°

5.4 Radiation Model Conclusions.

The work in this section presented a simple radiation model for computing the radiated electric field produced by the differential mode current distribution. The model assumed a thin wire approximation for the conductor current distribution which was computed by solving the multiconductor transmission line equations. This analysis was applied to model a simple multiconductor configuration consisting of five parallel tracks above ground and compared results with the identical problem modelled using the NEC moment method program. The results using these two methods were found to be in very good agreement.

The radiation model presented in this section has the advantage that once the distributed parameters are computed for a given conductor configuration, changes to various parameters in the circuit design and layout be analyzed quickly and used as an interactive tool to provide positive feedback in aiding the designer or layout engineer in reducing emission levels. One method by which this could be accomplished is to compute the maximum radiated field at 3m for each of the spectral components of the time domain signal and use this spectral profile as a reference. Then comparisons of the spectral profile computed for variations of the signal rise times, conductor lengths, and termination impedances can be compared with that of the reference to identify means which lower the emission levels. Another method would be to run an analysis of the computed electric field for all the high frequency signal lines on the board individually. From this analysis, the signal lines which are significant contributors can be identified and corrective action taken to lower their contribution to emissions.

6 SUMMARY AND CONCLUSIONS

6.1 Summary.

The work presented in this thesis deals with the theoretical development and application of the Finite Element Method to compute the distributed parameters for the purpose of modelling EMI/C phenomena on printed circuit boards. A finite element analysis program was developed for computing the distributed transmission line parameters for multilayerd dielectric, planar transmission media. In addition, programs implementing multiconductor transmission line theory were written and extended for time domain crosstalk analysis and frequency domain analysis of differential mode emissions.

6.2 Discussion and Future Work.

The accuracy of a finite element mesh depends on the type and order of the trial functions used over the finite elements and the number, shape, and placement of the finite elements. The finite element analysis program aids in minimizing the computational effort required by the algorithms developed for automated mesh generation (Section 3.5) which determines the shape, order, and placement of the finite elements and through the use of infinite elements.

The single conductor convergence results can also be used to compare the number of finite elements required to reach convergence for a bounded problem (i.e. conductor in TEM Cell configuration) with that of an unbounded problem (i.e. conductor in microstrip configuration). These results show that the unbounded problem required about twice as many finite elements in the studied region as that for the bounded problem. In solving the unbounded problem, the number of finite elements may well be reduced if a better approximation for the variation of the potential distribution outwards from the finite element boundary is found. One method by which

this may be accomplished is to derive a better infinite element. The infinite elements used, assumed a $1/r^n$ variation. It seems that an infinite element consisting of a summation of $1/r, 1/r^2, \ldots, 1/r^n$ would provide a better approximation. The potential use of combining the finite element solution with the another numerical technique which may allow a better approximation for the variation of the potential distribution is identified as an alternative approach to improve accuracy for a given amount of computational effort. In particular, the Boundary Element Method, in which integral equations can be defined around the finite element region boundary, is a viable and compatible method.

Even with the use of infinite elements, the major disadvantage in applying the finite element method to derive the transmission line parameters is the enormous amount of computer CPU time used which is the result of the large systems of equations (i.e >1000 unknowns) which are often required to reach convergence. One possible approach which would help reduce the CPU time is through the use of matrix preconditioning algorithms or sparse matrix equation solving algorithms which are applicable since the resulting finite element matrices are sparse matrices consisting of mainly zero elements.

Comparisons between the calculated and measured time domain results for the five conductor configuration (Section 4.5) for the 10 MHz trapezoidal waveform yielded good results for the near end crosstalk waveform but predicted higher peak values for the far-end waveforms. These differences are very likely due to radiation lossess which become prevalent at higher frequencies. Although for the signal frequencies, track lengths, and PCB construction materials used on typical PCB's, conductor and dielectric losses are generally negligible, more accurate results for the far end crosstalk may well be obtained, retaining the quasi-static approximation, by extending the distributed parameter multiconductor transmission line model to include losses (Section 2.2, Appendix B-5).

Comparisons between the computed results for the radiated electric field at 3m for the five conductor configuration calculated using the radiation model and the NEC Moment Method program (Section 5.3) were found to be in good agreement. It should be noted that a significant contribution to the radiated emission levels from electronic equipment results from the electromagnetic coupling of high frequency energy from the high frequency signals tracks onto external cables leaving the equipment. Thus although the radiation model is useful in identifying tracks with high emission levels and can provide valuable insight for reducing emission levels from these tracks, the radiated field may be significantly altered due to contributions from the cables attached to the equipment.

6.3 Conclusions.

In Section 3, the mathematical formulation and computer implementation of the Finite Element Method was found to be simple with the exception of the algorithm for automated mesh generation which required a significant amount of specialized code (i.e. of the order of 800-1000 lines of Fortran source code) for computer implementation.

The method and algorithms developed for automated mesh generation overcome many of the difficulties encountered in generating a finite element mesh. It reduces the user input time and effort by automatically generating a mesh given the conductor geometry, configuration, and relative permittivity of the dielectric. It offers flexibility to the user as it is 1) applicable to a wide variety of 2-D problems with inhomogeneous planar geometries, and 2) allows the user to vary the complexity of the mesh by specifying the number of segments on each conductor, first or second order triangular finite elements, and the minimum distance between any two nodes or between an interior node and the finite element region boundary. Other features which serve to improve the convergence for a given number of finite elements are 1) the creation of

a mesh consisting of near-equilateral finite elements, 2) placement of finite elements on elliptical contours around the conductors, and 3) effective grading of the mesh from a more dense mesh in the vicinity of the conductors where the potential varies most rapidly, to a less dense mesh away from the conductors where the potential varies more slowly.

The Finite Element solution of Laplace's equation used to compute the distributed parameters was found to be stable and convergent, however a very large system of equations was required to reach convergence. As a rule of thumb, it was found that for single conductor configurations a final system of about 600 unknowns was required and upwards of 1000 unknowns for multiconductor configurations. This is a significant finding and as a result it is concluded that this method of solution is 1) not suitable for real time use in present day CAD programs due to the CPU times required to solve the final system of equations, and 2) not suitable for use on the majority of present day Personal Computers due to the memory required to set up and solve these large systems. This disadvantage of the CPU time associated with solving the large systems of equations is further compounded as the number of conductors is increased. This results from the fact that for a multiconductor configuration consisting of 'n' conductors, the system of equations must be solved 'n' times for conductors in homogeneous media and '2n' times for conductors in inhomogeneous media.

In a real-time CAD application, the user would ideally receive feedback from his or her input immediately. From the data on the convergence results (section 3.6), it required approximately 15 CPU minutes to set up and solve a system of 1000 equations on a microvax workstation. This means that to compute the entire per-unit length capacitance matrix for a system of five conductors with 1000 unknowns would take approximately one hour and fifteen CPU minutes. To compute the per-unit length inductance matrix would require an additional one hour and fifteen CPU minutes, or two hours and thirty CPU minutes in total. Thus to

compute the per-unit length transmission line parameters using the finite element method, this is clearly not achievable in real-time on present day computers. However, for CAD and analysis of EMI/C on printed circuit boards, this is not necessarily a requirement. For example, since most CAD layout packages and manufacturing processes have fixed configurations, substrate thickness, dielectric constants, and allowable track widths and separations, the transmission line parameters could be stored in a look up table and then used in real-time EMC analysis programs. Also, in the near future, with faster computers and PC's with expanded memory becoming available, these present disadvantages will become less significant.

In Section 4 and 5 two models, based on Multiconductor Transmission Line Theory, were used to determine the extent to which this method of analysis can be applied to the design and layout of printed circuit boards. In Section 4, numerical results for the near-end and far-end crosstalk voltages were compared for a five parallel conductor configuration with experimental measurements in time and frequency domains. The numerical results were found to agree well with the experimental results and demonstrate the applicability of this model in predicting the crosstalk for the multiconductor geometries and signal rise times associated with present day digital circuits. In this respect, the 10 MHz excitation with 2-3 ns risetimes is representative of the waveform characteristics associated with the clock signals used in digital circuits. In Section 5, a simple radiation model was used to compute the electric field produced by a thin wire approximation for the current distribution computed in the crosstalk model and compared with numerical results computed by modelling the identical problem using the NEC moment method program. Although these radiation results (Section 5.3) are in very good agreement, the accuracy of this model in quantifying the field level has not been verified experimentally.

The conclusions from application of these two models are that they are suitable for inclusion into CAD programs and can be used as a tool in optimizing the design and layout to

minimize both on-board electromagnetic coupling and off board emissions. As most computer aided design and routing algorithms already contain the geometrical and physical characteristics of a printed circuit board, specialized code, which would allow the user to easily select parallel tracks and signals, and send these parameters to the prediction programs is required. This would allow identification and analysis of areas or tracks where cross coupling and emission levels are high (i.e data and address bus analysis and conductors running in proximity with clock lines), as well as an assessment of the impact of potential fixes such as alteration track width, length, and separation, or of signal characteristics (i.e. risetime, drive levels, filtering, etc...). In addition, this analysis could be used to determine the impedance characteristics of tracks in microstrip or stripline configuration which provide a constant impedance.

7 REFERENCES

[1] Federal Communication Commission, "Radio Frequency Devices," *Part 15B*, Docket 89-103.

[2] H. W. Ott, *Noise Reduction Techniques in Eletronic Systems*. John Wiley & Sons, 1988, ch. 1, pp. 6-189.

[3] P. Silvester and R. L. Ferrari, *Finite Elements for Electrical Engineers*. New York: Cambridge University Press, 1983, ch. 1-2, pp. 1-72.

[4] P. Silvester, "High-Order Polynomial Triangular Finite Elements for Potential Problems," *Int. J. Engng. Sci.*, Vol. 7, pp.849-861, 1969.

[5] S. Pissanetzky, "A Simple Infinite Element," *The International Journal for Computation and Mathematics in Electrical and Electronics Engineering*, Vol. 3, No. 2, pp. 107-114, 1984.

[6] Z. Pantic and R. Mittra, "Quasi-TEM Analysis of Microwave Transmission Lines by the Finite Element Method," *IEEE Transactions on Microwave Theory and Techniques*, Vol. MTT-34, No. 11, pp. 1096-1103, November 1986.

[7] D. M. Tracey and T. S. Cook, "Analysis of Power Type Singularities Using Finite Elements," *International Journal for Numerical Methods in Engineering*, Vol. 11, pp. 1225-1233, 1977.

[8] S. P. Edwards, K. De Meter, and Ph. De Wilde, "Adaptive Meshing Applied to a Two-Dimensional Device Simulator," *The International Journal for Computation and Mathematics in Electrical and Electronics Engineering*, Vol. 6, No. 2, pp. 65-70, 1987.

[9] J. M. Nelson, "A Triangulization Algorithm for Arbitrary Planar Domains," *Appl. Math. Modelling*, Vol. 2, pp. 151-159, September 1978.

[10] M.C. Rivara, "Algorithms for Refining Triangular Grids Suitable for Adaptive and Multigrid Techniques," *International Journal for Numerical Methods in Engineering*, Vol. 20, pp. 745-756, 1984.

[11] A. Liniecki and J. Yun, "Finite Element Triangular Meshing Optimization for Pure Torsion," *International Journal for Numerical Methods in Engineering*, Vol. 19, pp. 929-942, 1983.

[12] C. W. Wei, R. F. Harrington, J. R. Mautz, and T. K. Sarkar, "Multiconductor Transmission Lines in Multilayered Dielectric Media," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 32, No. 4, April 1984.

[13] C. R. Paul, "Solution of the Transmission-Line Equations for Three Conductor Lines in Homogeneous Media," *IEEE Transactions on Electromagnetic Compatibility*, Vol. EMC-20, No. 1, February 1978.

[14] C. R. Paul, "Applications of Multiconductor Transmission Line Theory to the Prediction of Cable Coupling: Vol.1," *Technical Report, RADC-TR-76-101*, Rome Air Development Center, Griffiss AFB, NY, April 1976.

[15] R. C. Paul and W. W. Everett, "Modelling Crosstalk on Printed Circuit Boards," *Phase Report, RADC-TR-85-107*, Rome Air Development Center, Griffiss AFB, NY, July 1985.

[16] G. I. Costache, "Finite Elements Method Applied to Skin-Effect Problems in Strip Transmission Lines," *IEEE Transactions on Microwave Theory and Techniques*, Vol. MTT-35, pp. 1009-1013, November 1987.

[17] C. R. Paul and A. E. Feather, "Computation of the Transmission Line Inductance and Capacitance Matrix from the Generalized Capacitance Matrix," *IEEE Transactions on Electromagnetic Compatibility*, Vol. EMC-18, No. 4, pp. 175-183, 1976.

[18] W. L. Stutzman and G. A. Theile, *Antenna Theory and Design*. John Wiley & Sons, 1981, ch. 1, pp. 1-29.

[19] G. J. Burke and A. J. Poggio, *Numerical Electromagnetics Code (NEC) - Method of Moments*. Part III: User's Guide, Technical Document 116, 18th July 1977, Revised 2 January 1980.

[20] International Special Committee on Radio Interference, "Limits and Methods of Measurement of Radio Interference Characteristics of Information Technology Equipment," *C.I.S.P.R Publication 22*, First Edition 1985.

Appendix A: Triangle Cotangent Identity

Appendix A derives explicit expressions for determining the cotangent of the three interior angles of a triangle in terms of the coordinates of the three nodes located at the three vertices which define the triangle. Evaluation of these angles can be used to construct the Laplacian element matrix for first, second, or higher order triangular finite elements (Section 3.3.3).

Consider a triangular finite element located in the xy plane inscribed in the smallest possible rectangle who's sides are parallel to the coordinate axis (Figure A-1) [3]. Let the node coordinates of these three nodes, labelled n_a, n_b, and n_c, be given by (x_{n_a}, y_{n_a}), (x_{n_b}, y_{n_b}), and (x_{n_c}, y_{n_c}), respectively. For the first order triangular finite element (Section 3.2.1) these three nodes are given by the nodes $n_a = n_1$, $n_b = n_2$, and $n_c = n_3$ while for the second order triangular finite element the node $n_a = n_1$, $n_b = n_4$, and $n_c = n_6$. Further, let the three interior angles of the triangle associated with the nodes n_a, n_b, and n_c be labelled θ_a, θ_b, and θ_c the other six angles, external to the triangle be labelled θ'_a, θ''_a, θ'_b, θ''_b, θ'_c, and θ''_c as illustrated in Figure A-1.

The three interior angles can now be written as:

$$\begin{aligned} \theta_a &= \pi - (\theta'_a + \theta''_a) \\ \theta_b &= \pi - (\theta'_b + \theta''_b) \\ \theta_c &= \frac{\pi}{2} - (\theta'_c + \theta''_c) \end{aligned} \qquad (A1.1)$$

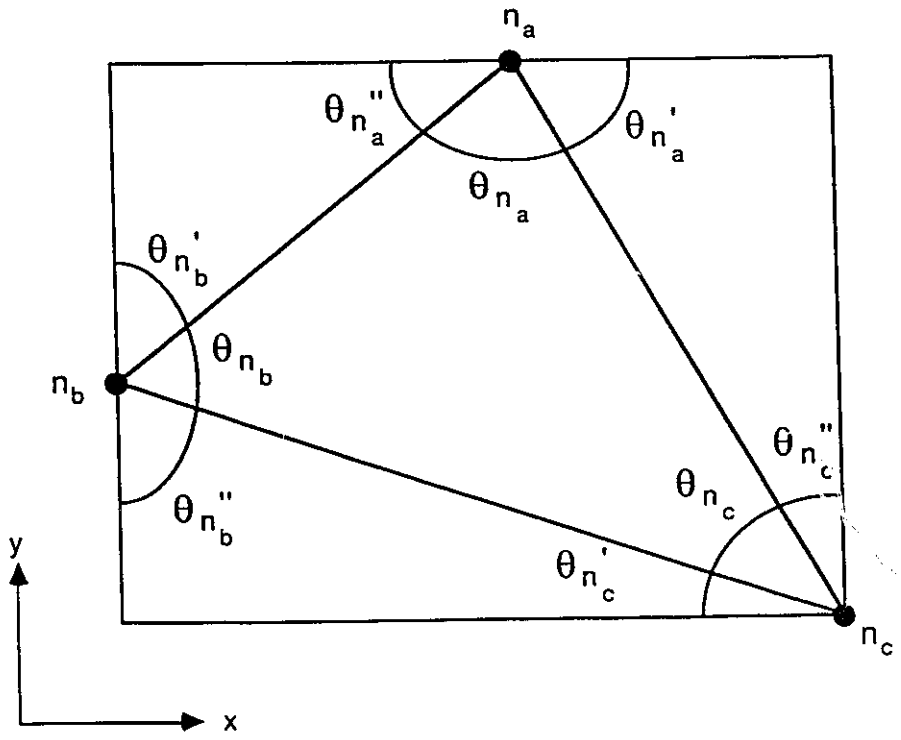


Figure A-1: Triangle Cotangent Identity.

Applying the trigonometric identity:

$$\cot(\pi \pm x) = \pm\cot(x) \tag{A 1.2a}$$

$$\cot\left(\frac{\pi}{2} \pm x\right) = \mp\tan(x)$$

followed by:

$$\cot(x \pm y) = \frac{1 \mp \tan(x)\tan(y)}{\tan(x) \pm \tan(y)} \tag{A 1.2b}$$

to the three interior angles yields:

$$\begin{aligned}
\cot(\theta_a) &= \cot(\pi - (\theta'_a + \theta''_a)) \\
&= -\cot(\theta'_a + \theta''_a) \\
&= \frac{\tan(\theta'_a)\tan(\theta''_a) - 1}{\tan(\theta'_a) + \tan(\theta''_a)} \\
\cot(\theta_b) &= \cot(\pi - (\theta'_b + \theta''_b)) \\
&= \frac{\tan(\theta'_b)\tan(\theta''_b) - 1}{\tan(\theta'_b) + \tan(\theta''_b)} \\
\cot(\theta_c) &= \cot(\pi/2 - (\theta'_c + \theta''_c)) \\
&= \frac{\tan(\theta'_c) + \tan(\theta''_c)}{1 - \tan(\theta'_c)\tan(\theta''_c)}
\end{aligned} \tag{A 1.3}$$

Substitution of the ratio of lengths for the tangent of the six exterior angles θ'_a, θ''_a, θ'_b, θ''_b, θ'_c, and θ''_c (Figure A-1) given by:

$$\tan\left(\theta'_a = \frac{(y_{n_a} - y_{n_c})}{(x_{n_c} - x_{n_a})}\right)$$

$$\tan\left(\theta''_a = \frac{(y_{n_a} - y_{n_b})}{(x_{n_a} - x_{n_b})}\right)$$

$$\tan\left(\theta'_b = \frac{(x_{n_a} - x_{n_b})}{(y_{n_a} - y_{n_b})}\right)$$

(A1.4)

$$\tan\left(\theta''_b = \frac{(x_{n_c} - x_{n_b})}{(y_{n_b} - y_{n_c})}\right)$$

$$\tan\left(\theta'_c = \frac{(y_{n_b} - y_{n_c})}{(x_{n_c} - x_{n_b})}\right)$$

$$\tan\left(\theta''_c = \frac{(x_{n_c} - x_{n_a})}{(y_{n_a} - y_{n_c})}\right)$$

in equation (A1.3) yields expressions for the cotangent of the three interior angles of the triangle as:

$$\cot(\theta_a) = \frac{(y_{n_a} - y_{n_c})(y_{n_a} - y_{n_b}) - (x_{n_c} - x_{n_a})(x_{n_a} - x_{n_b})}{2A}$$

$$\cot(\theta_b) = \frac{(x_{n_a} - x_{n_b})(x_{n_c} - x_{n_b}) - (y_{n_a} - y_{n_b})(y_{n_b} - y_{n_c})}{2A} \qquad (A1.5)$$

$$\cot(\theta_c) = \frac{(y_{n_b} - y_{n_c})(y_{n_a} - y_{n_c}) + (x_{n_c} - x_{n_a})(x_{n_c} - x_{n_b})}{2A}$$

where the quantity $2A$ is given by:

$$2A = x_{n_b}y_{n_c} - x_{n_c}y_{n_b} + x_{n_c}y_{n_a} - x_{n_a}y_{n_c} + x_{n_a}y_{n_b} - x_{n_b}y_{n_a} \qquad (A1.6)$$

Appendix B: Computer Programs

Appendix B lists the source code for several computer programs which implement the models and algorithms presented in Sections 3 thru 5, together with a brief description of each program, their input/output variables, and required subroutines. These include a frequency domain crosstalk/radiation program, a time domain crosstalk program, and a finite element analysis program.

Appendix B1: "MCTLE.FOR"

Description: This program solves the frequency domain multicondutcor transmission line equations for a configuration of 'n+1' lossless parallel conductors (i.e. 'n' conductors, 1 reference conductor) of total length 'L'. The required input parameters are the excitation amplitude, phase, and frequency, the length of the conductors, the per-unit length capacitance and inductance matrices, and the line termination impedances. The output parameters are the near and far end voltages across each of the line terminations. If the subroutine radiation is called, the reference conductor is assumed to be a perfect ground plane located in the xy plane. The conductors height above ground and the separation between each conductor must then be specified, and the computed current distribution is used to compute the radiated electric field at a fixed distance 'r' from the center of the conductor configuration with either angle ϕ or θ fixed. The unfixed angle may be varied in fixed steps.

Calling Subroutines:

- "MPRT": Prints out the entires in a matrix.
- "SQMINV": Inverts a real matrix.
- "SQMUX": Multiplies two real matrices.
- "CDSIMQ": Solves a complex system of equations.
- "DIAG": Computes the Eigenvalues/Eigenvectors of a real symmetric matrix.
- "Radiation": Computes the Radiated Electric Field.

Source Listing:

```
      IMPLICIT NONE
      real*8 c(9,9),cinv(9,9),l(9,9)
      real*8 t(9,9),cinvt(9,9),lamda(9)
      real*8 freq,length
      real*8 conv,pi,omega,value,sc
      real*8 im,re,pho(9),phl(9)
c-----
      complex*16 vo(9),vl(9)
      complex*16 zt(18),zgen(324),zz
      complex*16 vt(18),io(9),il(9)
c-----
      integer*4 ncon
      integer*4 nunit,ns,nn,nc,nrad
      integer*4 i,j,k,ks
c-----
      common/real/ t,cinvt,lamda
      common/rea2/ freq,length
c-----
      common/cmpl/ vt,io,il
c-----
      common/intl/ ncon
c-----
      CHARACTER*1 ESC
      ESC=CHAR(27)
c-----
      open (5,file='TLE.OUT',status='NEW')
c-----
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-------------------------------------------------------------------
c     NUMBER OF CONDUCTORS.
c-------------------------------------------------------------------
      write(*,900)
      read(*,*) ncon
      if((ncon.lt.1).or.(ncon.gt.9)) stop
c-------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-------------------------------------------------------------------
c     INPUT: CAPACITANCE/INDUCTANCE VALUES.
c-------------------------------------------------------------------
      write(*,902)
      read(*,*) nunit
      if((nunit.lt.1).or.(nunit.gt.3)) stop
```

```
      if(nunit.eq.1) conv=1.0d0
      if(nunit.eq.2) conv=1.0d-2
      if(nunit.eq.3) conv=(1.0d-2)/2.54d0
c-----
      do 10 nc=1,ncon
        do 20 k=nc,ncon
          if(nunit.eq.1) write(*,910) nc,k
          if(nunit.eq.2) write(*,912) nc,k
          if(nunit.eq.3) write(*,914) nc,k
          read(*,*) value
          if((nc.eq.k).and.(value.le.0.0d0)) stop
          if((nc.ne.k).and.(value.ge.0.0d0)) stop
          c(nc,k)=conv*1.0d-12*value
          if(nc.ne.k) c(k,nc)=c(nc,k)
20      continue
10    continue
c-----
      do 30 nc=1,ncon
        do 40 k=nc,ncon
          if(nunit.eq.1) write(*,920) nc,k
          if(nunit.eq.2) write(*,922) nc,k
          if(nunit.eq.3) write(*,924) nc,k
          read(*,*) value
          if(value.le.0.0d0) stop
          l(nc,k)=conv*1.0d-6*value
          if(nc.ne.k) l(k,nc)=l(nc,k)
40      continue
30    continue
c-----------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     INPUT: TERMINATION IMPEDANCES/VOLTAGE EXCITATIONS.
c-----------------------------------------------------------------------
      write(*,930)
      do 50 nc=1,ncon
        write(*,932) nc
        read(*,*) zt(nc)
        write(*,934) nc
        read(*,*) vt(nc)
        write(*,936) nc
        read(*,*) zt(nc+ncon)
        write(*,938) nc
        read(*,*) vt(nc+ncon)
50    continue
c-----------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     INPUT: FREQUENCY AND LENGTH.
c-----------------------------------------------------------------------
      write(*,940)
      read(*,*) freq
      if((freq.lt.1.0d0).or.(freq.gt.1000.0d0)) stop
      freq=1.0d6*freq
      write(*,942)
      read(*,*) length
      if(length.lt.0.0d0) stop
c-----------------------------------------------------------------------
      write(5,800) ncon,1.0D-6*freq,length
c-----
      sc=1.0d6
      call mprt(l,ncon,ncon,'INDUCTANCE [uH/m]',17,sc)
c-----
      sc=1.0d12
      call mprt(c,ncon,ncon,'CAPACITANCE [pF/m]',18,sc)
c-----
      write(5,810)
      do 812 nc=1,ncon
        write(5,814) nc,zt(nc),zt(nc+ncon)
812   continue
      write(5,820)
      do 822 nc=1,ncon
        write(5,824) nc,vt(nc),vt(nc+ncon)
822   continue
c-----------------------------------------------------------------------
      call sqminv(ncon,c,cinv)
```

```
c-----
      nn=2*ncon
      pi=dacos(-1.0d0)
      omega=2.0d0*pi*freq
c-----
      call diag(ncon,l,cinv,lamda,t)
c-----
      call sqmux(ncon,cinv,t,cinvt)
c-----
      do 100 i=1,ncon
        do 110 j=1,ncon
          cinvt(i,j)=cinvt(i,j)*dsqrt(lamda(j))
110     continue
100   continue
c-----
      do 200 i=1,ncon
        do 210 j=1,ncon
          zz=dcmplx(0.0d0,omega*dsqrt(lamda(j))*length)
          zgen((j-1)*nn+i)=(cinvt(i,j)-zt(i)*t(i,j))
          zgen((j+ncon-1)*nn+i)=(zt(i)*t(i,j)+cinvt(i,j))
          zgen(j*nn+i-ncon)=(zt(i+ncon)*t(i,j)+cinvt(i,j))*cdexp(zz)
          zgen((j+ncon)*nn+i-ncon)=(cinvt(i,j)-zt(i+ncon)*t(i,j))
     & *cdexp(-zz)
210     continue
200   continue
c-----
      call cdsimq(zgen,vt,nn,ks)
c-----
      if(ks.ne.0) write(*,950) ks
c-----
      do 300 i=1,ncon
         io(i)=dcmplx(0.0d0,0.0d0)
         vo(i)=dcmplx(0.0d0,0.0d0)
         do 310 j=1,ncon
           io(i)=t(i,j)*(vt(j+ncon)-vt(j))+io(i)
           vo(i)=cinvt(i,j)*(vt(j+ncon)+vt(j))+vo(i)
310      continue
         il(i)=dcmplx(0.0d0,0.0d0)
         vl(i)=dcmplx(0.0d0,0.0d0)
         do 320 j=1,ncon
           zz=dcmplx(0.0d0,omega*dsqrt(lamda(j))*length)
           il(i)=t(i,j)*(vt(j+ncon)*cdexp(-zz)-
     &                 vt(j)*cdexp(zz))+il(i)
           vl(i)=cinvt(i,j)*(vt(j+ncon)*cdexp(-zz)+
     &                 vt(j)*cdexp(zz))+vl(i)
320      continue
300   continue
c-----------------------------------------------------------------------
      do 400 i=1,ncon
        re=dreal(vo(i))
        im=dimag(vo(i))
        pho(i)=datan2(im,re)
        re=dreal(vl(i))
        im=dimag(vl(i))
        phl(i)=datan2(im,re)
400   continue
c-----------------------------------------------------------------------
      write(5,830)
      do 832 i=1,ncon
        write(5,834) i,1.0d3*cdabs(vo(i)),180.0d0*pho(i)/pi
     &              ,1.0d3*cdabs(vl(i)),180.0d0*phl(i)/pi
832   continue
c-----------------------------------------------------------------------
      do 500 i=1,ncon
        re=dreal(io(i))
        im=dimag(io(i))
        pho(i)=datan2(im,re)
        re=dreal(il(i))
        im=dimag(il(i))
        phl(i)=datan2(im,re)
500   continue
c-----------------------------------------------------------------------
      write(5,840)
      do 842 i=1,ncon
        write(5,844) i,1.0d3*cdabs(io(i)),180.0d0*pho(i)/pi
```

```
     &                ,1.0d3*cdabs(il(i)),180.0d0*phl(i)/pi
842   continue
c-----------------------------------------------------------------------
c     COMPUTE RADIATED ELECTRIC FIELD.
c-----------------------------------------------------------------------
      write(*,850)
      read(*,*) nrad
      if((nrad.lt.1).or.(nrad.gt.2)) stop
      if(nrad.eq.1) call radiation
c-----------------------------------------------------------------------
c     INPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
900   FORMAT(' ENTER: "Number of Conductors"')
902   FORMAT(' PART A: LINE PARAMETERS',/,/
     & ,' Input Units Desired [1,2,3]',/,/
     & ,' 1) Capacitance [pF/m]  & Inductance [uH/m]',/
     & ,' 2) Capacitance [pF/cm] & Inductance [uH/cm]',/
     & ,' 3) Capacitance [pF/in] & Inductance [uH/in]',/)
910   FORMAT(' ENTER: "C(',I1,',',I1,')" [pF/m]')
912   FORMAT(' ENTER: "C(',I1,',',I1,')" [pF/cm]')
914   FORMAT(' ENTER: "C(',I1,',',I1,')" [pF/in]')
920   FORMAT(' ENTER: "L(',I1,',',I1,')" [uH/m]')
922   FORMAT(' ENTER: "L(',I1,',',I1,')" [uH/cm]')
924   FORMAT(' ENTER: "L(',I1,',',I1,')" [uH/in]')
930   FORMAT(' PART B: TERMINATION NETWORKS - Input Format
     & "(real,imag)"',/)
932   FORMAT(/,' ENTER: "Impedance at Z=0 on Conductor# ',I1,'" ')
934   FORMAT(' ENTER: "Voltage at Z=0 on Conductor# ',I1,'" ')
936   FORMAT(/,' ENTER: "Impedance at Z=L on Conductor# ',I1,'" ')
938   FORMAT(' ENTER: "Voltage at Z=L on Conductor# ',I1,'" ')
940   FORMAT(' PART C: INPUT PARAMETERS',/,/,
     & ' ENTER: "Frequency  of Operation" [MHz]')
942   FORMAT(/,' ENTER: "Length of Conductors" [m]')
950   FORMAT(/,' SOLUTION ERROR! KS=',I4,/)
c-----------------------------------------------------------------------
c     OUTPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
800   format(/,' ncon:',i4,/,' freq:',1pd12.3,' MHz',/,
     &       ' length:',1pd12.3,' m')
810   FORMAT(/,' Termination Impedances: [ohms]',/,/,
     & '                                z=0                              z=l')
814   FORMAT(' Conductor#',I1,' :  ',' (',2(1PD10.2,')      ,
     &                          ,' (',2(1PD10.2,')      ')
820   FORMAT(/,' Termination Sources: [volts]',/,/,
     & '                                z=0                              z=l')
824   FORMAT(' Conductor#',I1,' :  ',' (',2(1PD10.2,')      ,
     &                          ,' (',2(1pd10.2,')      ')
830   format(/,/,'                          Vnext',
     &       '                    Vfext ',/
     &   ,'                 Mag. [mV]   Phase [deg]'
     &   ,'    Mag. [mV]   Phase [deg]')
834   format(' Conductor(',i1,') ',2(1pd12.3),'   ',2(1pd12.3))
840   format(/,/,'                              Inext',
     &       '                    Ifext ',/
     &   ,'                 Mag. [mA]   Phase [deg]'
     &   ,'    Mag. [mA]   Phase [deg]')
844   format(' Conductor(',i1,') ',2(1pd12.3),'   ',2(1pd12.3))
850   format(/,' CROSSTALK ANALYSIS COMPLETED.',/,/,
     & ' Radiaton Analysis: 1) Yes',/,
     & '                    2) No')
c-----------------------------------------------------------------------
      close(5)
c-----
      stop
      end

      subroutine sqmux(m,a,b,ab)
      IMPLICIT NONE
      real*8 a(9,9),b(9,9),ab(9,9)
c-----
      integer*4 i,j,k,m
c-----
      do 10 i=1,m
        do 20 j=1,m
          ab(i,j)=0.0d0
```

```
          do 30 k=1,m
            ab(i,j)=a(i,k)*b(k,j)+ab(i,j)
30        continue
20      continue
10    continue
c-----
      return
      end
      subroutine radiation
      IMPLICIT NONE
      real*8 t(9,9),cinvt(9,9),lamda(9)
      real*8 freq,length
      real*8 cs(8),height,value,dx,dz,cy(9)
      real*8 eo,uo,beta,pi,omega,lam,re,im
      real*8 phx,phy,phz,phr,phth,phph
      real*8 xr,xphi,xtheta,R,THETA,PHI
      real*8 xx1,yy1,zz1,rx1,rz1
      real*8 xx2,yy2,zz2,rx2,rz2
      real*8 fx(400),fy(400),fz(400)
      real*8 fr(400),ftheta(400),fphi(400)
c-----
      complex*16 vt(18),io(9),il(9)
      complex*16 curr(9,1000),zz
      complex*16 Ax,dxAxdx,dyAxdx,dzAxdx,cfx1,cfx2
      complex*16 Az,dxAzdz,dyAzdz,dzAzdz,cfz1,cfz2
      complex*16 Ex(400),Ey(400),Ez(400)
      complex*16 Er(400),Etheta(400),Ephi(400)
c-----
      integer*4 ncon
      integer*4 i,j,nt,nc,nd
      integer*4 nsx,nsz,ndegtheta,iff
c-----
      common/real/ t,cinvt,lamda
      common/rea2/ freq,length
c-----
      common/cmpl/ vt,io,il
c-----
      common/intl/ ncon
c-----
      CHARACTER*1 ESC
      ESC=CHAR(27)
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
      open(6,file='RADIATION.OUT',status='new')
c-----------------------------------------------------------------------
      pi=dacos(-1.0d0)
      eo=8.8542d-12
      uo=4.0d-7*pi
      omega=2.0d0*pi*freq
      beta=omega*dsqrt(uo*eo)
      lam=1.0d0/(dsqrt(eo*uo)*freq)
c-----------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     INPUT: CONDUCTOR HEIGHT/SEPARATION.
c-----------------------------------------------------------------------
      write(*,900)
      read(*,*) height
      write(*,910)
      read(*,*) i
      if((i.lt.1).or.(i.gt.2)) stop
      if(i.eq.1) then
        write(*,912)
        read(*,*) value
        do 10 i=1,ncon-1
          cs(i)=value
10      continue
      else
        do 20 i=1,ncon-1
          write(*,914) i,i+1
          read(*,*) cs(i)
          if(cs(i).le.0.0d0) stop
20      continue
      end if
```

```
c-----------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     INPUT: CONDUCTOR SEGMENTATION.
c-----------------------------------------------------------------------
      write(*,920)
      read(*,*) nsx
      if((nsx.lt.1).or.(nsx.gt.1000)) stop
      dx=length/dble(nsx)
      write(*,922)
      read(*,*) nsz
      if((nsz.lt.1).or.(nsz.gt.100)) stop
      dz=2.0d0*height/dble(nsz)
c-----------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     INPUT: R,THETA,PHI.
c-----------------------------------------------------------------------
      write(*,930)
      read(*,*) iff
      if((iff.lt.1).or.(iff.gt.2)) stop
      write(*,932)
      read(*,*) xr
      if((iff.eq.1).and.(xr.le.5.0d0*length)) stop
      if((iff.eq.1).and.(xr.le.1.6d0*lam)) stop
      if((iff.eq.1).and.(xr.le.2.0d0*length*length/lam)) stop
      if((iff.eq.2).and.(xr.le.height)) stop
c-------
      write(*,934)
      read(*,*) xphi
      if((xphi.lt.0.0d0).or.(xphi.gt.360.0d0)) stop
      write(*,936)
      read(*,*) i
      if((i.lt.1).or.(i.gt.2)) stop
      if(i.eq.1) then
        write(*,938)
        read(*,*) xtheta
        ndegtheta=0
      else
        write(*,940)
        read(*,*) xtheta
        if((xtheta.lt.1.0d0).or.(xtheta.gt.180.0d0)) stop
        ndegtheta=180.0d0/xtheta
        if((180.0d0/xtheta).ne.ndegtheta) stop
      end if
c-----------------------------------------------------------------------
      write(6,800) ncon,1.0d-6*freq,length,height
      value=0.0d0
      do 30 i=1,ncon-1
        value=value+cs(i)
30    continue
      cy(1)=-value/2.0d0
      write(6,802) 1,cy(1)
      do 40 i=2,ncon
        cy(i)=cy(i-1)+cs(i-1)
        write(6,802) i,cy(i)
40    continue
c-----------------------------------------------------------------------
c     COMPUTE CURRENT AT MIDPOINT OF EACH SEGMENT.
c-----------------------------------------------------------------------
      do 100 i=1,ncon
        do 110 nd=1,nsx
          curr(i,nd)=dcmplx(0.0d0,0.0d0)
          do 120 j=1,ncon
            zz=dcmplx(0.0d0,omega*dsqrt(lamda(j))*(dble(nd)-0.5d0)*dx)
            curr(i,nd)=t(i,j)*(vt(j+ncon)*cdexp(-zz)-
     &                 vt(j)*cdexp(zz))+curr(i,nd)
120       continue
110     continue
100   continue
c-----------------------------------------------------------------------
c     COMPUTE RADIATED ELECTRIC FIELD.
c-----------------------------------------------------------------------
      R=XR
      PHI=XPHI*pi/180.0d0
```

```
c-----
      do 200 nt=1,ndegtheta+1
        THETA=(-90.0d0+dble(nt-1)*(XTHETA))*pi/180.0d0
        if(ndegtheta.eq.0) THETA=XTHETA*pi/180.0d0
c-------
        Ax=dcmplx(0.0d0,0.0d0)
        dxAxdx=dcmplx(0.0d0,0.0d0)
        dyAxdx=dcmplx(0.0d0,0.0d0)
        dzAxdx=dcmplx(0.0d0,0.0d0)
        Az=dcmplx(0.0d0,0.0d0)
        dxAzdz=dcmplx(0.0d0,0.0d0)
        dyAzdz=dcmplx(0.0d0,0.0d0)
        dzAzdz=dcmplx(0.0d0,0.0d0)
c-------
        do 210  nc=1,ncon
          do 220 nd=1,nsx
            xx1=(R*dsin(THETA)*dcos(PHI)-
     &          ((dble(nd)-0.5d0)*dx-length/2.0d0))
            yy1=(R*dsin(THETA)*dsin(PHI)-cy(nc))
            zz1=(R*dcos(THETA)-height)
            xx2=(R*dsin(THETA)*dcos(PHI)-
     &          ((dble(nd)-0.5d0)*dx-length/2.0d0))
            yy2=(R*dsin(THETA)*dsin(PHI)-cy(nc))
            zz2=(R*dcos(THETA)+height)
            rx1=dsqrt((xx1*xx1)+(yy1*yy1)+(zz1*zz1))
            rx2=dsqrt((xx2*xx2)+(yy2*yy2)+(zz2*zz2))
            cfx1=xx1*dcmplx(((3.0d0/(beta*beta*rx1*rx1*rx1*rx1))
     &       -(1.0d0/(rx1*rx1))),(3.0d0/(beta*rx1*rx1*rx1)))
            cfx2=xx2*dcmplx(((3.0d0/(beta*beta*rx2*rx2*rx2*rx2))
     &       -(1.0d0/(rx2*rx2))),(3.0d0/(beta*rx2*rx2*rx2)))
            Ax=Ax+curr(nc,nd)*dx*(
     &          (cdexp(dcmplx(0.0d0,-beta*rx1))/rx1)
     &        -(cdexp(dcmplx(0.0d0,-beta*rx2))/rx2))
            if(iff.eq.1) go to 220
            dxAxdx=dxAxdx+curr(nc,nd)*dx*(
     &          ((xx1*cfx1
     &        -dcmplx(1.0d0/(beta*beta*rx1*rx1),1.0d0/(beta*rx1)))
     &        *cdexp(dcmplx(0.0d0,-beta*rx1))/rx1)
     &        -((xx2*cfx2
     &        -dcmplx(1.0d0/(beta*beta*rx2*rx2),1.0d0/(beta*rx2)))
     &        *cdexp(dcmplx(0.0d0,-beta*rx2))/rx2))
            dyAxdx=dyAxdx+curr(nc,nd)*dx*(
     &          (yy1*cfx1*cdexp(dcmplx(0.0d0,-beta*rx1))/rx1)
     &        -(yy2*cfx2*cdexp(dcmplx(0.0d0,-beta*rx2))/rx2))
            dzAxdx=dzAxdx+curr(nc,nd)*dx*(
     &          (zz1*cfx1*cdexp(dcmplx(0.0d0,-beta*rx1))/rx1)
     &        -(zz2*cfx2*cdexp(dcmplx(0.0d0,-beta*rx2))/rx2))
 220      continue
 210    continue
 c-------
        do 250  nc=1,ncon
          do 260 nd=1,nsz
            xx1=(R*dsin(THETA)*dcos(PHI)+length/2.0d0)
            yy1=(R*dsin(THETA)*dsin(PHI)-cy(nc))
            zz1=(R*dcos(THETA)-
     &          ((dble(nd)-0.5d0)*dz-height))
            xx2=(R*dsin(THETA)*dcos(PHI)-length/2.0d0)
            yy2=(R*dsin(THETA)*dsin(PHI)-cy(nc))
            zz2=(R*dcos(THETA)-
     &          ((dble(nd)-0.5d0)*dz-height))
            rz1=dsqrt((xx1*xx1)+(yy1*yy1)+(zz1*zz1))
            rz2=dsqrt((xx2*xx2)+(yy2*yy2)+(zz2*zz2))
            cfz1=zz1*dcmplx(((3.0d0/(beta*beta*rz1*rz1*rz1*rz1))
     &       -(1.0d0/(rz1*rz1))),(3.0d0/(beta*rz1*rz1*rz1)))
            cfz2=zz2*dcmplx(((3.0d0/(beta*beta*rz2*rz2*rz2*rz2))
     &       -(1.0d0/(rz2*rz2))),(3.0d0/(beta*rz2*rz2*rz2)))
            Az=Az+((io(nc)*dz*
     &          (cdexp(dcmplx(0.0d0,-beta*rz1))/rz1))
     &        -(il(nc)*dz*
     &          (cdexp(dcmplx(0.0d0,-beta*rz2))/rz2)))
            if(iff.eq.1) go to 260
            dxAzdz=dxAzdz+((io(nc)*dz*
     &          (xx1*cfz1*cdexp(dcmplx(0.0d0,-beta*rz1))/rz1))
     &        -(il(nc)*dz*
     &          (xx2*cfz2*cdexp(dcmplx(0.0d0,-beta*rz2))/rz2)))
```

```
            dyAzdz=dyAzdz+((io(nc)*dz*
     &          (yy1*cfz1*cdexp(dcmplx(0.0d0,-beta*rz1))/rz1))
     &        -(il(nc)*dz*
     &          (yy2*cfz2*cdexp(dcmplx(0.0d0,-beta*rz2))/rz2)))
            dzAzdz=dzAzdz
     &        +(io(nc)*dz*((zz1*cfz1
     &        -dcmplx(1.0d0/(beta*beta*rz1*rz1),1.0d0/(beta*rz1)))
     &        *cdexp(dcmplx(0.0d0,-beta*rz1))/rz1))
     &        -(il(nc)*dz*((zz2*cfz2
     &        -dcmplx(1.0d0/(beta*beta*rz2*rz2),1.0d0/(beta*rz2)))
     &        *cdexp(dcmplx(0.0d0,-beta*rz2))/rz2))
260       continue
250     continue
c-------
        Ex(nt)=dcmplx(0.0d0,-omega*1.0d-7)*(Ax+dxAxdx+dxAzdz)
        Ey(nt)=dcmplx(0.0d0,-omega*1.0d-7)*(dyAxdx+dyAzdz)
        Ez(nt)=dcmplx(0.0d0,-omega*1.0d-7)*(Az+dzAxdx+dzAzdz)
        Er(nt)=+Ex(nt)*dsin(THETA)*dcos(PHI)
     &         +Ey(nt)*dsin(THETA)*dsin(PHI)
     &         +Ez(nt)*dcos(THETA)
        Etheta(nt)=+Ex(nt)*dcos(THETA)*dcos(PHI)
     &             +Ey(nt)*dcos(THETA)*dsin(PHI)
     &             -Ez(nt)*dsin(THETA)
        Ephi(nt)=-Ex(nt)*dsin(PHI)
     &           +Ey(nt)*dcos(PHI)
        fr(nt)=R
        ftheta(nt)=THETA
        fphi(nt)=PHI
        fx(nt)=R*dsin(THETA)*dcos(PHI)
        fy(nt)=R*dsin(THETA)*dsin(PHI)
        fz(nt)=R*dcos(THETA)
c-------
200   continue
c-----------------------------------------------------------------------
c       OUTPUT RECTANGULAR COMPONENTS.
c-----------------------------------------------------------------------
        write(6,810)
        write(6,812)
        do 300 nt=1,ndegtheta+1
c---------
          re=dreal(Ex(nt))
          im=dimag(Ex(nt))
          phx=180.0d0*datan2(im,re)/pi
          re=dreal(Ey(nt))
          im=dimag(Ey(nt))
          phy=180.0d0*datan2(im,re)/pi
          re=dreal(Ez(nt))
          im=dimag(Ez(nt))
          phz=180.0d0*datan2(im,re)/pi
c---------
          write(6,814) fx(nt),fy(nt),fz(nt),cdabs(Ex(nt)),phx,
     &                 cdabs(Ey(nt)),phy,cdabs(Ez(nt)),phz
300     continue
c-----------------------------------------------------------------------
c       OUTPUT SPHEREICAL COMPONENTS.
c-----------------------------------------------------------------------
        write(6,820)
        write(6,822)
        do 400 nt=1,ndegtheta+1
c---------
          re=dreal(Er(nt))
          im=dimag(Er(nt))
          phr=180.0d0*datan2(im,re)/pi
          re=dreal(Etheta(nt))
          im=dimag(Etheta(nt))
          phth=180.0d0*datan2(im,re)/pi
          re=dreal(Ephi(nt))
          im=dimag(Ephi(nt))
          phph=180.0d0*datan2(im,re)/pi
c---------
          write(6,824) fr(nt),ftheta(nt),fphi(nt),cdabs(Er(nt)),phr,
     &                 cdabs(Etheta(nt)),phth,cdabs(Ephi(nt)),phph
400     continue
c-----------------------------------------------------------------------
c       INPUT FORMAT STATEMENTS.
```

```
c-----------------------------------------------------------------------
900   format(/,' ENTER: "Height of the Conductors
     & Above Ground Plane" [m]')
910   format(/,' ENTER: [ 1,2 ]',/,/,
     & ' 1) All Conductors are Equally Spaced.',/,
     & ' 2) Conductor Separation Varies.')
912   format(/,' ENTER: "Separation Between Conductors" [m]')
914   format(/,' ENTER: "Separation Between Conductors
     & #',il,' & ',il,' [m]')
920   format(/,' ENTER: "Number of Segments Along Length" [1-1000]')
922   format(/,' ENTER: "Number of Segments Along Height" [1-100]')
930   format(/,' ENTER: [ 1,2 ]',/,/,
     & ' 1) Far-Field Approximation.',/,
     & ' 2) Near-Field Approximation.')
932   format(/,' ENTER: "Radial Distance R" [m]')
934   format(/,' ENTER: "PHI" [degrees]')
936   format(/,/,' ENTER: "THETA" [1 or 2]',/,/,
     & ' 1) Fixed Value',/,
     & ' 2) Vary Theta')
938   format(/,' ENTER: "THETA" [degrees]')
940   format(/,' ENTER: "THETA Step [degrees]')
c-----------------------------------------------------------------------
c     OUTPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
800   format(/,' ncon',i4,' Freq',1pd12.3,' [MHz]',/,/,
     & ' Conductor Length',1pd12.3,' [m]',/,
     & ' Conductor Height',1pd12.3,' [m]',/,/)
802   format(' Conductor #',il,' Y - Location',1pd12.3,' [m]')
810   format(/,/,' Rectangular Coordinate Components',/)
812   format(/,/,'    X [m]    Y [m]     Z [m] ',
     & '                Ex [V/m]                      Ey [V/m]
     &                Ez [V/m] ',/)
814   format(3(1pd10.2),'     (',1pd12.3,',',1pd12.3,')  ',
     & '  (',1pd12.3,',',1pd12.3,')  ',
     & '  (',1pd12.3,',',1pd12.3,')  ')
820   format(/,/,' Spereical Coordinate Components',/)
822   format(/,/,'      Er [m]   Theta [deg]  Phi [deg]  ',
     & '           Er [V/m]                  Etheta [V/m]   ',
     & '          Ephi [V/m]     ',/)
824   format(3(1pd12.2),'      (',1pd10.2,',',1pd10.2,') ',
     & '  (',1pd10.2,',',1pd10.2,')  ',
     & '  (',1pd10.2,',',1pd10.2,')  ')
c-----------------------------------------------------------------------
      close(6)
c-----
      return
      end
```

Appendix B2: "TDSTLE.FOR"

Description: This program computes the near and far end time domain crosstalk voltage waveforms for a system of 'n+1' lossless parallel conductors of total length 'L' for a single periodic time domain excitation. Program input is identical to that of "MCTLE.FOR" with the exception that the time domain voltage waveform of the excitation is constructed by specifying voltage/time points over one period of the waveform. In the analysis, the time domain input waveform is sampled using linear interpolation between the user specified points according to the order chosen for the FFT. The multiconductor transmission line equations are then solved at each frequency and the program output consists of the inverse FFT's, i.e. time domain crosstalk voltage waveforms, across each of the conductor terminations.

Calling Subroutines:

- "TRAPWAVE": Reads in Points of Input Waveform.
- "RGFFT": FFT algorithm.
- "MPRT": Prints out entries in a Matrix.
- "SQMINV": Inverts a real Matrix.
- "DIAG": Computes the Eigenvectors/Eigen values of a real matrix.
- "SQMUX": Multiplies two real matrices.
- "CDSIMQ": Solves a complex system of Equations.

Program Listing:

```
      IMPLICIT NONE
      real*8 c(9,9),cinv(9,9),l(9,9)
      real*8 cinvt(9,9),t(9,9),lamda(9)
      real*8 fclk,length,time(4096)
      real*8 conv,pi,omega,value,sc,freq
      real*8 im,re,pho(9),phl(9)
      real*8 sign,phto,phtl
c-----
      complex*16 vo(9),vl(9),io(9),il(9)
      complex*16 zt(18),zgen(324),zz
      complex*16 vt(18),vex(4096)
      complex*16 tvo(9,4096),tvl(9,4096)
      complex*16 stvt(18),stvo(4096),stvl(4096)
c-----
      integer*4 ncon,nft,mft
      integer*4 nunit,ns,nn,nc
      integer*4 i,j,k,ks
c-----
      common/real/ fclk,length,time
c-----
      common/intl/ ncon,nft,mft
c-----
      common/cmpl/ vt,vex
c-----
      CHARACTER*1 ESC
      ESC=CHAR(27)
c-----
      open (5,file='TDSTLE.OUT',status='NEW')
      open (6,file='TDSTLE.DAT',status='NEW')
c-----
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-------------------------------------------------------------------
c     NUMBER OF CONDUCTORS.
c-------------------------------------------------------------------
      write(*,900)
      read(*,*) ncon
      if((ncon.lt.1).or.(ncon.gt.9)) stop
c-------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-------------------------------------------------------------------
c     INPUT: CAPACITANCE/INDUCTANCE VALUES.
c-------------------------------------------------------------------
      write(*,902)
      read(*,*) nunit
      if((nunit.lt.1).or.(nunit.gt.3)) stop
      if(nunit.eq.1) conv=1.0d0
      if(nunit.eq.2) conv=1.0d-2
      if(nunit.eq.3) conv=(1.0d-2)/2.54d0
c-----
      do 10 nc=1,ncon
        do 20 k=nc,ncon
          if(nunit.eq.1) write(*,910) nc,k
          if(nunit.eq.2) write(*,912) nc,k
          if(nunit.eq.3) write(*,914) nc,k
          read(*,*) value
          if((nc.eq.k).and.(value.le.0.0d0)) stop
          if((nc.ne.k).and.(value.ge.0.0d0)) stop
          c(nc,k)=conv*1.0d-12*value
          if(nc.ne.k) c(k,nc)=c(nc,k)
20      continue
10    continue
c-----
      do 30 nc=1,ncon
        do 40 k=nc,ncon
          if(nunit.eq.1) write(*,920) nc,k
          if(nunit.eq.2) write(*,922) nc,k
          if(nunit.eq.3) write(*,924) nc,k
```

```
          read(*,*) value
          if(value.le.0.0d0) stop
          l(nc,k)=conv*1.0d-6*value
          if(nc.ne.k) l(k,nc)=l(nc,k)
40      continue
30    continue
c-----------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     INPUT: TERMINATION IMPEDANCES.
c-----------------------------------------------------------------------
      write(*,930)
      do 50 nc=1,ncon
        write(*,932) nc
        read(*,*) zt(nc)
        write(*,936) nc
        read(*,*) zt(nc+ncon)
50    continue
c-----------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     INPUT: LENGTH.
c-----------------------------------------------------------------------
      write(*,940)
      read(*,*) length
      if(length.lt.0.0d0) stop
c-----------------------------------------------------------------------
c     INPUT TRAPEZOIDAL WAVEFORM AND LOCATION.
c-----------------------------------------------------------------------
      call trapwave
c-----------------------------------------------------------------------
c     TAKE FOURIER TRANSFORM OF EXCITATION.
c-----------------------------------------------------------------------
      sign=-1.0d0
      call rgfft(vex,mft,nft,sign)
c-----------------------------------------------------------------------
c     OUTPUT TRANSMISSION LINE DATA.
c-----------------------------------------------------------------------
      write(5,800) ncon,length
c-----
      sc=1.0d6
      call mprt(l,ncon,ncon,'INDUCTANCE [uH/m]',17,sc)
c-----
      sc=1.0d12
      call mprt(c,ncon,ncon,'CAPACITANCE [pF/m]',18,sc)
c-----
      write(5,810)
      do 812 nc=1,ncon
        write(5,814) nc,zt(nc),zt(nc+ncon)
812   continue
      write(5,820)
      do 822 nc=1,ncon
        write(5,824) nc,vt(nc),vt(nc+ncon)
822   continue
c-----------------------------------------------------------------------
      call sqminv(ncon,c,cinv)
c-----
      nn=2*ncon
      pi=dacos(-1.0d0)
c-----
      call diag(ncon,l,cinv,lamda,t)
c-----
      call sqmux(ncon,cinv,t,cinvt)
c-----
      do 100 i=1,ncon
        do 110 j=1,ncon
          cinvt(i,j)=cinvt(i,j)*dsqrt(lamda(j))
110     continue
100   continue
c-----------------------------------------------------------------------
c     SOLVE MULTICONDUCTOR TLE AT EACH FREQUENCY.
c-----------------------------------------------------------------------
      do 200 k=1,(nft/2)+1
        freq=dble(k-1)*fclk
        omega=2.0d0*pi*freq
```

```
c-------
        do 300 i=1,ncon
          stvt(i)=vt(i)
          stvt(i+ncon)=vt(i+ncon)
300     continue
c-------
        do 400 i=1,ncon
          do 410 j=1,ncon
            zz=dcmplx(0.0d0,omega*dsqrt(lamda(j))*length)
            zgen((j-1)*nn+i)=(cinvt(i,j)-zt(i)*t(i,j))
            zgen((j+ncon-1)*nn+i)=(zt(i)*t(i,j)+cinvt(i,j))
            zgen(j*nn+i-ncon)=(zt(i+ncon)*t(i,j)+cinvt(i,j))*
     &           cdexp(zz)
            zgen((j+ncon)*nn+i-ncon)=(cinvt(i,j)-zt(i+ncon)*t(i,j))*
     &           cdexp(-zz)
410       continue
400     continue
c-------
        call cdsimq(zgen,stvt,nn,ks)
c-------
        if(ks.ne.0) write(*,950) ks
c-------
        do 500 i=1,ncon
            io(i)=dcmplx(0.0d0,0.0d0)
            vo(i)=dcmplx(0.0d0,0.0d0)
            do 510 j=1,ncon
              io(i)=t(i,j)*(stvt(j+ncon)-stvt(j))+io(i)
              vo(i)=cinvt(i,j)*(stvt(j+ncon)+stvt(j))+vo(i)
510         continue
c-----------
            tvo(i,k)=vex(k)*vo(i)
c-----------
            il(i)=dcmplx(0.0d0,0.0d0)
            vl(i)=dcmplx(0.0d0,0.0d0)
            do 520 j=1,ncon
              zz=dcmplx(0.0d0,omega*dsqrt(lamda(j))*length)
              il(i)=t(i,j)*(stvt(j+ncon)*cdexp(-zz)-
     &                    stvt(j)*cdexp(zz))+il(i)
              vl(i)=cinvt(i,j)*(stvt(j+ncon)*cdexp(-zz)+
     &                    stvt(j)*cdexp(zz))+vl(i)
520         continue
c-----------
            tvl(i,k)=vex(k)*vl(i)
c-----------
500     continue
c-------
        if(k.eq.2) then
          do 550 i=1,ncon
            re=dreal(vo(i))
            im=dimag(vo(i))
            pho(i)=datan2(im,re)
            re=dreal(vl(i))
            im=dimag(vl(i))
            phl(i)=datan2(im,re)
550       continue
          write(5,830)
          do 832 i=1,ncon
          write(5,834) i,1.0d3*cdabs(vo(i)),180.0d0*pho(i)/pi
     &                  ,1.0d3*cdabs(vl(i)),180.0d0*phl(i)/pi
832       continue
        end if
200   continue
c-----------------------------------------------------------------------
      do 600 i=1,ncon
        do 650 j=(nft/2)+2,nft
          tvo(i,j)=dconjg(tvo(i,nft-j+2))
          tvl(i,j)=dconjg(tvl(i,nft-j+2))
650     continue
600   continue
c-----------------------------------------------------------------------
c     COMPUTE INVERSE FOURIER TRANSFORMS.
c-----------------------------------------------------------------------
      sign=1.0d0
      do 700 i=1,ncon
        do 725 j=1,nft
```

```
          stvo(j)=tvo(i,j)
          stvl(j)=tvl(i,j)
725     continue
c-------
        call rgfft(stvo,mft,nft,sign)
        call rgfft(stvl,mft,nft,sign)
c-------
        do 775 j=1,nft
          tvo(i,j)=stvo(j)
          tvl(i,j)=stvl(j)
775     continue
700   continue
c-----------------------------------------------------------------------
c     OUTPUT: TIME DOMAIN SIGNALS.
c-----------------------------------------------------------------------
      write(6,850)
      do 852 i=1,ncon
        write(6,854) i
        do 856 j=1,nft
          re=dreal(tvo(i,j))
          im=dimag(tvo(i,j))
          phto=180.0d0*datan2(im,re)/pi
          re=dreal(tvl(i,j))
          im=dimag(tvl(i,j))
          phtl=180.0d0*datan2(im,re)/pi
          write(6,858) j,1.0d9*time(j),1.0d3*cdabs(tvo(i,j)),phto,
     &                                 1.0d3*cdabs(tvl(i,j)),phtl
856     continue
852   continue
c-----------------------------------------------------------------------
c     INPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
900   FORMAT(' ENTER: "Number of Conductors"')
902   FORMAT(' PART A: LINE PARAMETERS',/,/
     & ,' Input Units Desired [1,2,3]',/,/
     & ,' 1) Capacitance [pF/m]  & Inductance [uH/m]',/
     & ,' 2) Capacitance [pF/cm] & Inductance [uH/cm]',/
     & ,' 3) Capacitance [pF/in] & Inductance [uH/in]',/)
910   FORMAT(' ENTER: "C(',I1,',',I1,')" [pF/m]')
912   FORMAT(' ENTER: "C(',I1,',',I1,')" [pF/cm]')
914   FORMAT(' ENTER: "C(',I1,',',I1,')" [pF/in]')
920   FORMAT(' ENTER: "L(',I1,',',I1,')" [uH/m]')
922   FORMAT(' ENTER: "L(',I1,',',I1,')" [uH/cm]')
924   FORMAT(' ENTER: "L(',I1,',',I1,')" [uH/in]')
930   FORMAT(' PART B: TERMINATION NETWORKS - Input Format
     & "(real,imag)"',/)
932   FORMAT(/,' ENTER: "Impedance at Z=0 on Conductor# ',I1,'" ')
936   FORMAT(/,' ENTER: "Impedance at Z=L on Conductor# ',I1,'" ')
940   FORMAT(' PART C: INPUT PARAMETERS',/,/,
     & ' ENTER: "Length of Conductors" [m]')
950   FORMAT(/,' SOLUTION ERROR! KS=',I4,/)
c-----------------------------------------------------------------------
c     OUTPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
800   format(/,' ncon:',i4,/,' length:',1pd12.3,' m')
810   FORMAT(/,' Termination Impedances: [ohms]',/,/,
     & '                                z=0                          z=l')
814   FORMAT(' Conductor#',I1,' :  ',' (',2(1PD10.2),')      '
     &                          ,' (',2(1PD10.2),')      ')
820   FORMAT(/,' Termination Sources: [volts]',/,/,
     & '                                z=0                          z=l')
824   FORMAT(' Conductor#',I1,' :  ',' (',2(1PD10.2),')      '
     &                          ,' (',2(1pd10.2),')      ')
830   format(/,/,'                                Vnext',
     &       '                    Vfext ',/,/
     &    ,'                 Mag. [mV]  Phase [deg]'
     &    ,'     Mag. [mV]  Phase [deg]',/)
834   format(' Conductor(',i1,') ',2(1pd12.3),'   ',2(1pd12.3))
850   format(/,/,' TIME DOMAIN SIGNALS')
854   format(/,'     Conductor#',i2,'             Vnext',
     & '                    Vfext',/,/,'    i    Time [ns] ',
     & '   Mag. [mV]  Phase [Deg]   Mag. [mv]  Phase [deg]',/)
858   format(' ',i4,' ',1pd12.3,' ',2(1pd12.3),' ',2(1pd12.3))
c-----------------------------------------------------------------------
      close(5)
```

```
      close(6)
c-----
      stop
      end
      subroutine sqmux(m,a,b,ab)
      IMPLICIT NONE
      real*8 a(9,9),b(9,9),ab(9,9)
c-----
      integer*4 i,j,k,m
c-----
      do 10 i=1,m
        do 20 j=1,m
          ab(i,j)=0.0d0
          do 30 k=1,m
            ab(i,j)=a(i,k)*b(k,j)+ab(i,j)
30        continue
20      continue
10    continue
c-----
      return
      end
      subroutine trapwave
      IMPLICIT NONE
      real*8 fclk,length,time(4096)
      real*8 vmax,vmin,tr,tf
      real*8 vbp(6),tbp(6)
c-----
      integer*4 i,j,nbpts
      integer*4 ncon,nft,mft
c-----
      complex*16 vt(18),vex(4096)
      complex*16 value
c-----
      common/real/ fclk,length,time
      common/int1/ ncon,nft,mft
      common/cmpl/ vt,vex
c-----------------------------------------------------------------------
      CHARACTER*1 ESC
      ESC=CHAR(27)
c-----------------------------------------------------------------------
      do 10 i=1,ncon
        vt(i)=dcmplx(0.0d0,0.0d0)
        vt(i+ncon)=dcmplx(0.0d0,0.0d0)
10    continue
c-----------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     INPUT: LOCATION OF EXCITATION.
c-----------------------------------------------------------------------
      write(*,900) 1,ncon
      read(*,*) i
      if((i.lt.1).or.(i.gt.ncon)) stop
      write(*,902) i,0.0d0,i,length
      read(*,*) j
      if((j.lt.1).or.(j.gt.2)) stop
      if(j.eq.1) write(5,800) i,0.0d0
      if(j.eq.2) write(5,800) i,length
      vt(i+(j-1)*ncon)=dcmplx(1.0d0,0.0d0)
c-----------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     INPUT: Construct Waveform.
c-----------------------------------------------------------------------
      write(*,910)
      read(*,*) vbp(1),tbp(1)
      tbp(1)=1.0d-9*tbp(1)
      vmax=vbp(1)
      vmin=vbp(1)
      write(*,912)
      read(*,*) nbpts
      if((nbpts.lt.1).or.(nbpts.gt.24)) stop
      nbpts=nbpts+1
      do 20 i=2,nbpts
        write(*,914) i
```

```
        read(*,*) vbp(i),tbp(i)
        tbp(i)=1.0d-9*tbp(i)
        if(tbp(i).le.tbp(i-1)) stop
        if(vbp(i).lt.vmin) vmin=vbp(i)
        if(vbp(i).gt.vmax) vmax=vbp(i)
20    continue
      fclk=1.0d0/(tbp(nbpts)-tbp(1))
      tr=0.0d0
      tf=0.0d0
      do 30 i=1,nbpts-1
        if((vbp(i).eq.vmin).and.(vbp(i+1).eq.vmax)) tr=tbp(i+1)-tbp(i)
        if((vbp(i).eq.vmax).and.(vbp(i+1).eq.vmin)) tf=tbp(i+1)-tbp(i)
30    continue
      write(5,810) 1.0d-6*fclk,1.0d9/fclk,vmax,vmin
      if(tr.ne.0.0d0) then
        write(5,820) 1.0d9*tr,4.0d9*tr/5.0d0
      else
        write(5,822)
      end if
      if(tf.ne.0.0d0) then
        write(5,824) 1.0d9*tf,4.0d9*tf/5.0d0
      else
        write(5,826)
      end if
c-----------------------------------------------------------------------
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     LEVEL FOR FFT.
c-----------------------------------------------------------------------
      write(*,920)
      read(*,*) mft
      if((mft.lt.2).or.(mft.gt.12)) stop
      nft=2.0d0**(dble(mft))
      write(5,830) mft,nft
c-----------------------------------------------------------------------
c     DISCRETIZE WAVEFORM.
c-----------------------------------------------------------------------
      do 100 i=1,nft
        time(i)=tbp(1)+dble(i-1)/(fclk*dble(nft))
        do 200 j=2,nbpts
          if(time(i).lt.tbp(j)) then
            vex(i)=dcmplx(((vbp(j)-vbp(j-1))*(time(i)-tbp(j-1))/
     &                    (tbp(j)-tbp(j-1)))+vbp(j-1),0.0d0)
            go to 100
          end if
200     continue
100   continue
c-----------------------------------------------------------------------
      write(6,840)
      value=vex(1)
      write(6,850) 1,1.0d9*time(1),vex(1)
      do 300 i=2,nft
        if(vex(i).eq.value) go to 300
        value=vex(i)
        write(6,850) i,1.0d9*time(i),vex(i)
300   continue
c-----------------------------------------------------------------------
c     INPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
900   format(/,' ENTER: "Excitation on Conductor#" [',
     &           i1,' to ',i1,']')
902   format(/,' ENTER: "Location of Excitation"',/,/,
     &        ' 1) On Conductor#',i2,' at x=',1pd12.3,' [m].',/,
     &        ' 2) On Conductor#',i2,' at x=',1pd12.3,' [m].')
910   format(/,' ENTER: "Left edge Reference" [volts,ns]')
912   format(/,' ENTER: "Number of Additional Breakpoints" [1 to 24]')
914   format(/,' ENTER: "Breakpoint#',i2,' [volts,ns]')
920   format(/,' ENTER: "Order of FFT" [ 2 to 12 ]')
c-----------------------------------------------------------------------
c     OUPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
800   format(/,' Trapezoidal Waveform Characteristics:',/,/,
     &        ' Excitation on Conductor#',i2,' at x=',1pd12.3,
     &        ' [m]')
810   format(/,' Exciation - Frequency:',1pd12.3,' [MHz]',/,
```

```
     &          '            - Period   :',1pd12.3,' [ns]',/,
     &          '            - Vmax     :',1pd12.3,' [volts]',/,
     &          '            - Vmin     :',1pd12.3,' [volts]')
820   format(/,' Rise Time: 1) Vmin/Vmax - ',1pd12.3,' [ns]',/,
     &          '             2) 10%/90%   - ',1pd12.3,' [ns]')
822   format(/,' Rise Time: Not Found')
824   format(/,' Fall Time: 1) Vmax/Vmin - ',1pd12.3,' [ns]',/,
     &          '             2) 10%/90%   - ',1pd12.3,' [ns]')
826   format(/,' Fall Time: Not Found')
830   format(/,' For Fast Fourier Transform: mft=',i2,' nft=',i5)
840   format(/,' INPUT TRAPEZOIDAL WAVEFORM:',/,/,
     &          '     i      Time [ns]         Vex [volts]',/)
850   format('  ',i4,'  ',1pd12.3,'  ',2(1pd12.3))
c-----------------------------------------------------------------------
      return
      end
```

Appendix B3: "FEM.FOR"

Description: This FORTRAN program applies the Finite Element Method to compute the per-unit length multiconductor transmission line capacitance matrix for a system of 'n' conductors in microstip, triplate, or TEM Cell configuration. Program input is the cross-sectional geometry of the multiconductor configuration. The program accepts both zero thickness and finite thickness conductors and, for the analysis, either first or second order triangular finte elements and corresponding infinite elements may be selected. Dielectric subregions are modelled as dieletric layers with real permittivity $\varepsilon_r\varepsilon_o$. The program output consists of the per-unit length multiconductor transmission line capacitance matrix using the charge definition. The self capacitances (i.e. C_{ii}'s) are also computed using the energy definition.

Calling Subroutines:

- "INPUT": Reads in Geometry of Problem.
- "NODELEM": Generates the finite element mesh.
- "SECIV": Translates any first order triangular finite element mesh to a second order finite element mesh.
- "FILTER": Filters the nodes in the finite element mesh.
- "INFE": Creates infinite elements on finite element region boundary.
- "FPMESH", "FPCNTR", "FPSSOL": Triangulizes a simply connected region of nodes into finite elements.
- "QMAT": Generates the Q-matrices for first and second order triangular finite elements.

- "SOLVE": Sets up the finial system of equations.

- "ZGECO", "ZGESL": Solves a complex system of equations.

- "ENERGY": Computes the per-unit length capacitance of a conductor based on energy in the finite element mesh.

- "CHARGE": Computes the per-unit length capacitance between two conductors by computing the per-unit length charge residing on a conductor.

Program Listing:

```
      IMPLICIT NONE
      real*8 top,bottom,right,left
      real*8 cl(5),cr(5),ct(5),cb(5)
      real*8 dl(5),er(4)
      real*8 x(1160),y(1160),pot(1160)
c-----
      integer*4 ncon,nord,nogp,nod
      integer*4 nofe,nptl,ndir,nn
      integer*4 iv(1500,6),ieps(1500)
      integer*4 int(5,50),icn(5,50),nt(5),cn(5)
      integer*4 ql(6,6),q2(6,6),q3(6,6),iden
      integer*4 i,j,k,l,m,n,nbl,ndbl,icon
      integer*4 nds(1160)
      integer*4 noie,ivinf(500,3),epsinf(500)
c-----
      logical nd(1160)
c-----
      common /real/ top,bottom,right,left
      common /rea2/ cl,cr,ct,cb
      common /rea3/ dl,er
      common /rea4/ x,y,pot
c-----
      common /intl/ ncon,nord,nogp,nod
      common /int2/ nofe,nptl,ndir,nn
      common /int3/ iv,ieps
      common /int4/ int,icn,nt,cn
      common /int5/ ql,q2,q3,iden
      common /int6/ nds
      common /int7/ noie,ivinf,epsinf
c-----
      common /logl/ nd
c---------------------------------------------------------------------
c---------------------------------------------------------------------
      open(5,file='FEMINF.OUT',status='NEW')
c---------------------------------------------------------------------
c     READ INPUT DATA.
c---------------------------------------------------------------------
      call input
c---------------------------------------------------------------------
c     GENERATE NODES AND MESH.
c---------------------------------------------------------------------
      call nodelem
```

```
c-----------------------------------------------------------------------
c     SET UP Q1,Q2,Q3 MATRICIES.
c-----------------------------------------------------------------------
      call qmat(nord)
c-----------------------------------------------------------------------
      do 100 icon=1,ncon
c-----------------------------------------------------------------------
c       INITIALIZE NODE POTENTIALS.
c-----------------------------------------------------------------------
        do 110 j=1,nptl
          pot(j)=0.0d0
110     continue
        do 120 j=1,cn(icon)
          pot(icn(icon,j))=1.0d0
120     continue
c-----------------------------------------------------------------------
c       SET UP AND SOLVE FINAL SYSTEM OF EQUATIONS.
c-----------------------------------------------------------------------
        call solve
c-----------------------------------------------------------------------
c       CALCULATE CAPACITANCE USING ENERGY DEFINITION.
c-----------------------------------------------------------------------
        call energy(icon)
c-----------------------------------------------------------------------
c       CALCULATE CAPACITANCE USING CHARGE DEFINITION.
c-----------------------------------------------------------------------
        call charge(icon)
c-----------------------------------------------------------------------
100   continue
c-----
      close(5)
c-----
      stop
      end

      subroutine input
      IMPLICIT NONE
      real*8 top,bottom,right,left
      real*8 cl(5),cr(5),ct(5),cb(5)
      real*8 dl(5),er(4)
      real*8 save
c-----
      integer*4 ncon,nord,nogp,nod
      integer*4 i,j,k,l,m,n
c-----
      common /rea1/ top,bottom,right,left
      common /rea2/ cl,cr,ct,cb
      common /rea3/ dl,er
c-----
      common /int1/ ncon,nord,nogp,nod
c-----
      CHARACTER*1 ESC
      ESC=CHAR(27)
c-----
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
c     FINITE ELEMENT BOUNDARY.
c-----------------------------------------------------------------------
      write(*,900)
      read(*,*) left,right
      if(right.lt.left) then
        save=left
        left=right
        right=save
      end if
      write(*,902)
      read(*,*) bottom,top
      if(top.lt.bottom) then
        save=bottom
        bottom=top
        top=save
      end if
c-----
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----------------------------------------------------------------------
```

```
c     NUMBER OF CONDUCTORS, WIDTH, THICKNESS, CENTROIDS.
c-------------------------------------------------------------------
      write(*,910)
      read(*,*) ncon
      if((ncon.lt.1).or.(ncon.gt.5)) stop
c-----
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----
      write(*,912)
      do 10 i=1,ncon
        write(*,914) i
        write(*,916)
        read(*,*) cl(i),cr(i)
        if(cl(i).gt.cr(i)) then
          save=cl(i)
          cl(i)=cr(i)
          cr(i)=save
        end if
        write(*,918)
        read(*,*) ct(i),cb(i)
        if(cb(i).gt.ct(i)) then
          save=ct(i)
          ct(i)=cb(i)
          cb(i)=save
        end if
        if((ct(i)-cb(i)).gt.(cr(i)-cl(i))) stop
10    continue
c-----
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-------------------------------------------------------------------
c     DIELECTRIC LAYERS.
c-------------------------------------------------------------------
      write(*,920)
      read(*,*) n
      if((n.lt.0).or.(n.gt.3)) stop
c-----
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----
      dl(1)=bottom
      write(*,922)
      do 20 i=1,n
        write(*,924) i
        write(*,926)
        read(*,*) dl(i+1)
        if(dl(i+1).le.dl(i)) stop
20    continue
      if(dl(n+1).ge.top) stop
      dl(n+2)=top
c-----
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-----
      nod=n+1
      do 30 i=1,nod
        write(*,928) i,dl(i),dl(i+1)
        read(*,*) er(i)
30    continue
c-----
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-------------------------------------------------------------------
c     NUMBER OF GROUND PLANES & ORDER OF FINITE ELEMENTS.
c-------------------------------------------------------------------
      write(*,930)
      read(*,*) nogp
      if((nogp.lt.1).or.(nogp.gt.4).or.(nogp.eq.3)) stop
      write(*,932)
      read(*,*) nord
      if((nord.lt.1).or.(nord.gt.2)) stop
c-----
      WRITE(*,' (1X,A,A)') ESC,'[2J'
c-------------------------------------------------------------------
c     OUTPUT.
c-------------------------------------------------------------------
      write(5,800) ncon,nogp,nod,nord
      write(5,802)
      do 40 i=1,ncon
```

```
        write(5,804) i,(cr(i)-cl(i)),(ct(i)-cb(i))
40    continue
      write(5,806)
      do 50 i=1,nod
        write(5,808) i,dl(i),dl(i+1),er(i)
50    continue
      write(5,810) top,bottom,right,left
c-----------------------------------------------------------------------
c     OUTPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
800   format(/,' ncon',i2,'    nogp',i2,'    nod',i2,'    nord',i2)
802   format(/,/,' Conductor      Width       Thickness',/)
804   format('     ',i1,'       ',1pd12.4,'  ',1pd12.4)
806   format(/,/,'  Layer          y1    < Y <    y2            er',/)
808   format('     ',i1,'       ',1pd12.4,'   ',1pd12.4,'  ',1pd12.4)
810   format(/,/,' Finite Element Boundary: Top    ',1pd12.4,/
     &          '                          Bottom',1pd12.4,/
     &          '                          Right  ',1pd12.4,/
     &          '                          Left   ',1pd12.4,/)
c-----------------------------------------------------------------------
c     INPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
900   format(/,' ENTER: "Finite Element Boundary" [ x1,x2 ]')
902   format(/,' ENTER: "Finite Element Boundary" [ y1,y2 ]')
910   format(/,' ENTER: "Number of Conductors"    [ 1,2,3,4,5 ]')
912   format(/,' LOCATION OF CONDUCTOR(S)',/,/)
914   format(/,' Conductor #',i2,':')
916   format(/,' ENTER: "x1,x2"')
918   format(/,' ENTER: "y1,y2"')
920   format(/,' ENTER: "Number of Horizontal Layer Cuts" [ 0,1,2,3 ]')
922   format(/,' HORIZONTAL LAYER CUT LOCATION(S)',/,/)
924   format(/,' Layer Cut #',i2,':')
926   format(/,' ENTER: "y"')
928   format(/,' DIELECTRIC LAYER #',i2,
     &         ' {',1pd12.3,' > y > ',1pd12.3,'}',/,/,
     &         ' ENTER: "Relative Dielectric Constant of Layer"')
930   format(/,' ENTER: "Number of Ground Planes"',/,/,
     &  ' 1) Single Ground Plane  ( Microstrip Configuration )',/,
     &  ' 2) Two Ground Planes    ( Triplate Configuration )',/,
     &  ' 4) Four Ground Planes   ( TEM Cell )')
932   format(/,' ENTER: "Order of Finite Elements"  [ 1,2 ]')
c-----
      return
      end

      subroutine nodelem
      IMPLICIT NONE
      real*8 top,bottom,right,left
      real*8 cl(5),cr(5),ct(5),cb(5)
      real*8 dl(5),er(4)
      real*8 x(1160),y(1160),pot(1160)
      real*8 xs(1160),ys(1160)
      real*8 w,t,xc,yc,dx,dy,rc,xnew,ynew,rnew
      real*8 kw,uw,dw,kt,ut,dt,v
c-----
      integer*4 ncon,nord,nogp,nod
      integer*4 nofe,nptl,ndir,nn
      integer*4 iv(1500,6),ieps(1500)
      integer*4 int(5,50),icn(5,50),nt(5),cn(5)
      integer*4 nscw(5),nsct(5),nsi(5),np(6),iwac(5)
      integer*4 ibound(5,60),ns(5,200),nss(200)
      integer*4 nds(1160)
      integer*4 mw,mt,n1,n2,n3,ndnew,nb,dimiv,ifil
      integer*4 i,j,k,l,m,n
      integer*4 noie,ivinf(500,3),epsinf(500)
c-----
      logical nd(1160)
c-----
      common /real/ top,bottom,right,left
      common /rea2/ cl,cr,ct,cb
      common /rea3/ dl,er
      common /rea4/ x,y,pot
c-----
      common /int1/ ncon,nord,nogp,nod
      common /int2/ nofe,nptl,ndir,nn
```

```
      common /int3/ iv,ieps
      common /int4/ int,icn,nt,cn
      common /int6/ nds
      common /int7/ noie,ivinf,epsinf
c-----
      common /log1/ nd
c-----
      dimiv=1500
c-----------------------------------------------------------------------
c     GENERATE ALL MANDITORY NODES ON FINITE ELEMENT BOUNDARY.
c-----------------------------------------------------------------------
      npt1=0
      do 10 i=1,nod+1
        xs(npt1+1)=left
        ys(npt1+1)=dl(i)
        nds(npt1+1)=1
        xs(npt1+2)=right
        ys(npt1+2)=dl(i)
        nds(npt1+2)=1
        npt1=npt1+2
10    continue
c-----------------------------------------------------------------------
c     GENERATE ELLIPTICAL NODE BOUNDARY AROUND EACH CONDUCTOR.
c-----------------------------------------------------------------------
      do 100 i=1,ncon
        w=(cr(i)-cl(i))
        t=(ct(i)-cb(i))
        xc=(cl(i)+cr(i))/2.0d0
        yc=(ct(i)+cb(i))/2.0d0
        write(*,900) i,cl(i),cr(i)
        read(*,*) nscw(i)
        if((nscw(i).lt.1).or.(nscw(i).gt.25)) stop
        nsct(i)=jidnnt(dble(nscw(i))*t/w)
        mw=nscw(i)+3
        dw=dacos(-1.0d0)/dble(mw)
        dx=w/(dcos(dw)+dcos(dw+dw))
        dy=dsqrt(3.0d0)*w*(dcos(dw)-dcos(dw+dw))/
     &     ((dcos(dw)+dcos(dw+dw))*(dsin(dw)+dsin(dw+dw)))
        kw=dsqrt(dx*dx-dy*dy)
        dx=w/(kw*(dcos(dw)+dcos(dw+dw)))
        uw=dlog(dx+dsqrt(dx*dx-1.0d0))
c-------
        if(nsct(i).eq.0) then
          mt=0
          if(t.eq.0) then
            dx=kw*dcosh(uw)*dcos(dw)
            dy=kw*dsinh(uw)*dsin(dw)
            rc=kw*dcosh(uw)
          else
            nsct(i)=1
            rc=kw*dsinh(uw)*dsin(dw+dw)+(t/2.0d0)
            dx=kw*dcosh(uw)*dcos(dw+dw)+(rc/dsqrt(2.0d0))
            dy=rc/dsqrt(2.0d0)
            rc=kw*dcosh(uw)*dcos(dw+dw)+rc
          end if
        else
          mt=nsct(i)+3
          dt=dacos(-1.0d0)/dble(mt)
          dx=t/(dcos(dt)+dcos(dt+dt))
          dy=dsqrt(3.0d0)*t*(dcos(dt)-dcos(dt+dt))/
     &       ((dcos(dt)+dcos(dt+dt))*(dsin(dt)+dsin(dt+dt)))
          kt=dsqrt(dx*dx-dy*dy)
          dx=t/(kt*(dcos(dt)+dcos(dt+dt)))
          ut=dlog(dx+dsqrt(dx*dx-1.0d0))
          dx=(kt*dsinh(ut)*dsin(dt+dt)/dsqrt(2.0d0))+(w/2.0d0)
          dy=(kw*dsinh(uw)*dsin(dw+dw)/dsqrt(2.0d0))+(t/2.0d0)
        end if
c-------
        n=npt1
        xs(n+1)=xc+dx
        ys(n+1)=yc-dy
        n=n+1
        do 110 j=3,mw-1
          v=2.0d0*dacos(-1.0d0)-dble(j-1)*dw
          xs(n+1)=xc+kw*dcosh(uw)*dcos(v)
```

```
          ys(n+1)=yc-(t/2.0d0)+kw*dsinh(uw)*dsin(v)
          n=n+1
110     continue
        xs(n+1)=xc-dx
        ys(n+1)=yc-dy
        n=n+1
        if(mt.eq.0) then
          xs(n+1)=xc-rc
          ys(n+1)=yc
          n=n+1
        else
          do 120 j=3,mt-1
            v=2.0d0*dacos(-1.0d0)-dble(j-1)*dt
            xs(n+1)=xc-(w/2.0d0)+kt*dsinh(ut)*dsin(v)
            ys(n+1)=yc-kt*dcosh(ut)*dcos(v)
            n=n+1
120       continue
        end if
        xs(n+1)=xc-dx
        ys(n+1)=yc+dy
        n=n+1
        do 130 j=3,mw-1
          v=dacos(-1.0d0)-dble(j-1)*dw
          xs(n+1)=xc+kw*dcosh(uw)*dcos(v)
          ys(n+1)=yc+(t/2.0d0)+kw*dsinh(uw)*dsin(v)
          n=n+1
130     continue
        xs(n+1)=xc+dx
        ys(n+1)=yc+dy
        n=n+1
        if(mt.eq.0) then
          xs(n+1)=xc+rc
          ys(n+1)=yc
          n=n+1
        else
          do 140 j=3,mt-1
            v=dacos(-1.0d0)-dble(j-1)*dt
            xs(n+1)=xc+(w/2.0d0)+kt*dsinh(ut)*dsin(v)
            ys(n+1)=yc-kt*dcosh(ut)*dcos(v)
            n=n+1
140       continue
        end if
c-------
        m=n-nptl
        np(i+1)=m
        nsi(i)=m
        do 150 j=1,m
          ns(i,j)=nptl+j
          ibound(i,j)=nptl+j
          nds(nptl+j)=i+1
150     continue
c-------
        nptl=n
        iwac(i)=1
100   continue
c-----------------------------------------------------------------------
c     GENERATE INTERIOR NODES AND NODES ON FINITE ELEMENT BOUNDARY.
c-----------------------------------------------------------------------
180   do 200 i=1,ncon
        if(iwac(i).eq.0) go to 200
        iwac(i)=0
        n=0
c-------
        do 210 j=1,nsi(i)
          nl=ns(i,j)
          if(j.eq.nsi(i)) then
            n2=ns(i,1)
          else
            n2=ns(i,j+1)
          end if
          if((nds(nl).eq.1).or.(nds(n2).eq.1)) go to 230
          xc=(xs(nl)+xs(n2))/2.0d0
          yc=(ys(nl)+ys(n2))/2.0d0
          dx=dsqrt(3.0d0)*(ys(nl)-ys(n2))/2.0d0
```

```
          dy=dsqrt(3.0d0)*(xs(n2)-xs(n1))/2.0d0
          xnew=xc+1.5d0*dx
          ynew=yc+1.5d0*dy
          if((xnew.gt.left).and.(xnew.lt.right).and.
     &       (ynew.gt.bottom).and.(ynew.lt.top)) then
            xnew=xc+dx
            ynew=yc+dy
            ndnew=0
          else
            if(xnew.le.left) then
              xnew=left
              ynew=yc+dy*(left-xc)/dx
            end if
            if(xnew.ge.right) then
              xnew=right
              ynew=yc+dy*(right-xc)/dx
            end if
            if(ynew.le.bottom) then
              xnew=xc+dx*(bottom-yc)/dy
              ynew=bottom
            end if
            if(ynew.ge.top) then
              xnew=xc+dx*(top-yc)/dy
              ynew=top
            end if
            ndnew=1
          end if
          if((ndnew.lt.0).or.(ndnew.gt.1)) then
            write(*,*) ' Node Generation Error'
            stop
          end if
          rc=dsqrt((xnew-xc)*(xnew-xc)+(ynew-yc)*(ynew-yc))
          do 220 k=1,nptl
            rnew=dsqrt((xnew-xs(k))*(xnew-xs(k))+
     &                 (ynew-ys(k))*(ynew-ys(k)))
            if(rnew.le.rc/dsqrt(2.0d0)) go to 230
220       continue
          xs(nptl+1)=xnew
          ys(nptl+1)=ynew
          nds(nptl+1)=ndnew
          nss(n+1)=nptl+1
          n=n+1
          nptl=nptl+1
          iwac(i)=1
          go to 210
c---------
230       if(n.eq.0) then
            nss(n+1)=n1
            nss(n+2)=n2
            n=n+2
          else
            if(n1.ne.nss(n)) then
              nss(n+1)=n1
              n=n+1
            end if
            if(n2.ne.nss(1)) then
              nss(n+1)=n2
              n=n+1
            end if
          end if
c---------
210     continue
        if(n.gt.200) stop
        nsi(i)=n
        do 250 j=1,200
          ns(i,j)=0
          if(j.le.n) ns(i,j)=nss(j)
          nss(j)=0
250     continue
c-------
200   continue
      i=0
      do 300 j=1,ncon
```

```
        if(iwac(j).eq.1) i=1
300   continue
      if(i.eq.1) go to 180
c-----------------------------------------------------------------------
c     FIND AND SORT NODES LOCATED ON FINITE ELEMENT BOUNDARY.
c-----------------------------------------------------------------------
      nsi(1)=0
      nsi(2)=0
      nsi(3)=0
      nsi(4)=0
      do 400 i=1,npt1
        if(nds(i).eq.1) then
          if(((xs(i).ge.left).and.(xs(i).le.right)).and.
     &       (ys(i).eq.bottom)) then
            ns(1,nsi(1)+1)=i
            nsi(1)=nsi(1)+1
          end if
          if(((ys(i).gt.bottom).and.(ys(i).lt.top)).and.
     &       (xs(i).eq.right)) then
            ns(2,nsi(2)+1)=i
            nsi(2)=nsi(2)+1
          end if
          if(((xs(i).le.right).and.(xs(i).ge.left)).and.
     &       (ys(i).eq.top)) then
            ns(3,nsi(3)+1)=i
            nsi(3)=nsi(3)+1
          end if
          if(((ys(i).lt.top).and.(ys(i).gt.bottom)).and.
     &       (xs(i).eq.left)) then
            ns(4,nsi(4)+1)=i
            nsi(4)=nsi(4)+1
          end if
        end if
400   continue
c-----
      do 500 i=1,4
        k=1
        m=nsi(i)-1
510     if(k.eq.1) then
          k=0
          do 520 j=1,m
            if((xs(ns(i,j))).ge.xs(ns(i,j+1))).and.
     &         (ys(ns(i,j))).ge.ys(ns(i,j+1)))) then
              l=ns(i,j)
              ns(i,j)=ns(i,j+1)
              ns(i,j+1)=l
              k=1
            end if
520       continue
          m=m-1
          go to 510
        end if
500   continue
c-----
      m=0
      do 550 i=1,4
        do 580 j=1,nsi(i)
          k=j
          if(i.gt.2) k=nsi(i)-j+1
          x(m+1)=xs(ns(i,k))
          y(m+1)=ys(ns(i,k))
          nd(m+1)=.false.
          if((i.eq.1).or.((nogp.eq.2).and.(i.eq.3)).or.(nogp.eq.4))
     &      nd(m+1)=.true.
          m=m+1
580     continue
550   continue
c-----
      np(1)=m
      do 600 i=1,ncon
        do 610 j=1,np(i+1)
          x(m+1)=xs(ibound(i,j))
          y(m+1)=ys(ibound(i,j))
          ibound(i,j)=m+1
```

```
          nd(m+1)=.false.
          m=m+1
610     continue
600   continue
c-----
      do 650 i=1,nptl
        if(nds(i).eq.0) then
          x(m+1)=xs(i)
          y(m+1)=ys(i)
          nd(m+1)=.false.
          m=m+1
        end if
650   continue
      if(m.ne.nptl) stop
c-----
      nb=ncon+1
c-----
      call fpmesh(iv,x,y,nptl,nofe,nb,np,dimiv)
c-----
      m=0
      do 680 i=1,nb
        m=m+np(i)
680   continue
      m=2*(nptl-2+nb)-m
      if(nofe.ne.m) then
        write(*,*) ' FPMESH ERROR'
        stop
      end if
c-----------------------------------------------------------------------
c     CREATE CONDUCTOR NODES.
c-----------------------------------------------------------------------
      do 700 i=1,ncon
        m=0
        n=0
        x(nptl+1)=cr(i)
        y(nptl+1)=cb(i)
        nd(nptl+1)=.true.
        icn(i,m+1)=nptl+1
        ns(i,n+1)=nptl+1
        ns(i,n+2)=nptl+1
        nptl=nptl+1
        m=m+1
        n=n+2
        do 710 j=2,nscw(i)
          x(nptl+1)=(x(ibound(i,n))+x(ibound(i,n+1)))/2.0d0
          y(nptl+1)=cb(i)
          nd(nptl+1)=.true.
          icn(i,m+1)=nptl+1
          ns(i,n+1)=nptl+1
          nptl=nptl+1
          m=m+1
          n=n+1
710     continue
        x(nptl+1)=cl(i)
        y(nptl+1)=cb(i)
        nd(nptl+1)=.true.
        icn(i,m+1)=nptl+1
        ns(i,n+1)=nptl+1
        ns(i,n+2)=nptl+1
        nptl=nptl+1
        m=m+1
        n=n+2
c-------
        if(ct(i).eq.cb(i)) then
          if(m.gt.50) stop
          cn(i)=m
          do 750 j=1,n
            ns(i,n+j)=ns(i,n-j+1)
750       continue
          go to 700
        end if
c-------
        do 720 j=2,nsct(i)
          x(nptl+1)=cl(i)
```

```
          y(nptl+1)=(y(ibound(i,n))+y(ibound(i,n+1)))/2.0d0
          nd(nptl+1)=.true.
          icn(i,m+1)=nptl+1
          ns(i,n+1)=nptl+1
          ns(i,n+2)=nptl+1
          nptl=nptl+1
          m=m+1
          n=n+1
  720     continue
          x(nptl+1)=cl(i)
          y(nptl+1)=ct(i)
          nd(nptl+1)=.true.
          icn(i,m+1)=nptl+1
          ns(i,n+1)=nptl+1
          ns(i,n+2)=nptl+1
          nptl=nptl+1
          m=m+1
          n=n+2
          do 730 j=2,nscw(i)
            x(nptl+1)=(x(ibound(i,n))+x(ibound(i,n+1)))/2.0d0
            y(nptl+1)=ct(i)
            nd(nptl+1)=.true.
            icn(i,m+1)=nptl+1
            ns(i,n+1)=nptl+1
            nptl=nptl+1
            m=m+1
            n=n+1
  730     continue
          x(nptl+1)=cr(i)
          y(nptl+1)=ct(i)
          nd(nptl+1)=.true.
          icn(i,m+1)=nptl+1
          ns(i,n+1)=nptl+1
          ns(i,n+2)=nptl+1
          nptl=nptl+1
          m=m+1
          n=n+2
          do 740 j=2,nsct(i)
            x(nptl+1)=cr(i)
            y(nptl+1)=(y(ibound(i,n))+y(ibound(i,n+1)))/2.0d0
            nd(nptl+1)=.true.
            icn(i,m+1)=nptl+1
            ns(i,n+1)=nptl+1
            nptl=nptl+1
            m=m+1
            n=n+1
  740     continue
          if(m.gt.50) stop
          cn(i)=m
  700   continue
  c-------------------------------------------------------------------
  c     CREATE MESH AROUND CONDUCTOR(S).
  c-------------------------------------------------------------------
        do 760 i=1,ncon
          m=0
          do 770 j=1,np(i+1)
            n1=ibound(i,j)
            if(j.eq.np(i+1)) then
              n2=ibound(i,1)
              n3=ns(i,1)
            else
              n2=ibound(i,j+1)
              n3=ns(i,j+1)
            end if
            iv(nofe+1,1)=n1
            iv(nofe+1,2)=n2
            iv(nofe+1,3)=n3
            nofe=nofe+1
            n3=n1
            n1=ns(i,j)
            if(j.eq.np(i+1)) then
              n2=ns(i,1)
            else
              n2=ns(i,j+1)
            end if
```

```
            if(n1.eq.n2) go to 770
            iv(nofe+1,1)=n1
            iv(nofe+1,2)=n2
            iv(nofe+1,3)=n3
            int(i,m+1)=nofe+1
            nofe=nofe+1
            m=m+1
770       continue
          if(m.gt.50) stop
          nt(i)=m
760     continue
c-----------------------------------------------------------------------
c       ASSIGN RELATIVE DIELECTRIC CONSTANT TO EACH ELEMENT.
c-----------------------------------------------------------------------
        do 780 i=1,nofe
          n1=iv(i,1)
          n2=iv(i,2)
          n3=iv(i,3)
          do 790 j=2,nod
            if(((y(n1).gt.dl(j)).and.(y(n2).lt.dl(j))).or.
     &         ((y(n2).gt.dl(j)).and.(y(n1).lt.dl(j)))) then
            ynew=dl(j)
            xnew=(ynew*(x(n1)-x(n2))+x(n2)*y(n1)-x(n1)*y(n2))
     &           /(y(n1)-y(n2))
            rc=((xnew-x(n1))*(xnew-x(n1))+(ynew-y(n1))*(ynew-y(n1)))
            rnew=((xnew-x(n2))*(xnew-x(n2))+(ynew-y(n2))*(ynew-y(n2)))
            if(nd(n1).and.nd(n2)) stop
            k=2
            if(dsqrt(rc).lt.dsqrt(rnew)) k=1
            if(nd(n1)) k=2
            if(nd(n2)) k=1
            if(k.eq.1) then
              x(n1)=xnew
              y(n1)=ynew
            end if
            if(k.eq.2) then
              x(n2)=xnew
              y(n2)=ynew
            end if
          end if
          if(((y(n1).gt.dl(j)).and.(y(n3).lt.dl(j))).or.
     &       ((y(n3).gt.dl(j)).and.(y(n1).lt.dl(j)))) then
            ynew=dl(j)
            xnew=(ynew*(x(n1)-x(n3))+x(n3)*y(n1)-x(n1)*y(n3))
     &           /(y(n1)-y(n3))
            rc=((xnew-x(n1))*(xnew-x(n1))+(ynew-y(n1))*(ynew-y(n1)))
            rnew=((xnew-x(n3))*(xnew-x(n3))+(ynew-y(n3))*(ynew-y(n3)))
            if(nd(n1).and.nd(n3)) stop
            k=3
            if(dsqrt(rc).lt.dsqrt(rnew)) k=1
            if(nd(n1)) k=3
            if(nd(n3)) k=1
            if(k.eq.1) then
              x(n1)=xnew
              y(n1)=ynew
            end if
            if(k.eq.3) then
              x(n3)=xnew
              y(n3)=ynew
            end if
          end if
          if(((y(n2).gt.dl(j)).and.(y(n3).lt.dl(j))).or.
     &       ((y(n3).gt.dl(j)).and.(y(n2).lt.dl(j)))) then
            ynew=dl(j)
            xnew=(ynew*(x(n2)-x(n3))+x(n3)*y(n2)-x(n2)*y(n3))
     &           /(y(n2)-y(n3))
            rc=((xnew-x(n2))*(xnew-x(n2))+(ynew-y(n2))*(ynew-y(n2)))
            rnew=((xnew-x(n3))*(xnew-x(n3))+(ynew-y(n3))*(ynew-y(n3)))
            if(nd(n2).and.nd(n3)) stop
            k=3
            if(dsqrt(rc).lt.dsqrt(rnew)) k=2
            if(nd(n2)) k=3
            if(nd(n3)) k=2
            if(k.eq.2) then
```

```
              x(n2)=xnew
              y(n2)=ynew
            end if
            if(k.eq.3) then
              x(n3)=xnew
              y(n3)=ynew
            end if
          end if
790     continue
        k=0
        do 795 j=1,nod
          if(((y(n1).ge.dl(j)).and.(y(n1).le.dl(j+1))).and.
     &       ((y(n2).ge.dl(j)).and.(y(n2).le.dl(j+1))).and.
     &       ((y(n3).ge.dl(j)).and.(y(n3).le.dl(j+1)))) k=j
795     continue
        if(k.eq.0) stop
        ieps(i)=k
780   continue
c-------------------------------------------------------------------------
c     FILTER NODES.
c-------------------------------------------------------------------------
      write(*,904)
      read(*,*) ifil
      if(ifil.lt.0) stop
      if(ifil.gt.0) call filter(ifil,np,ibound)
c-------------------------------------------------------------------------
      if(nord.eq.2) then
        call seciv
c-------
        do 800 i=1,ncon
          m=0
          if(ct(i).eq.cb(i)) then
            do 810 j=1,nt(i)/2
              icn(i,m+1)=iv(int(i,j),1)
              icn(i,m+2)=iv(int(i,j),2)
              m=m+2
              if(j.ne.(nt(i)/2)) go to 810
              icn(i,m+1)=iv(int(i,j),4)
              m=m+1
810         continue
          else
            do 820 j=1,nt(i)
              icn(i,m+1)=iv(int(i,j),1)
              icn(i,m+2)=iv(int(i,j),2)
              m=m+2
820         continue
          end if
          if(m.gt.50) stop
          cn(i)=m
800     continue
c-------
      end if
c-----
      ndir=0
      do 850 i=1,nptl
        nds(i)=ndir
        if(nd(i)) ndir=ndir+1
850   continue
      nn=nptl-ndir
c-----
      write(5,910) nptl,ndir,nofe,nn
      write(*,910) nptl,ndir,nofe,nn
      write(5,912)
      write(5,914) (i,cn(i),nt(i),i=1,ncon)
      if((nn.gt.1070).or.(nptl.gt.1160).or.(nofe.gt.dimiv)) then
        write(*,*) nn,nptl,nofe
        stop
      end if
c-----
      if(nogp.eq.1) call infem
c-----
      write(5,916) noie
      write(*,916) noie
      if(noie.gt.500) stop
c-------------------------------------------------------------------------
```

```
c     INPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
900   format(/,' ENTER: No. of Seg. on Conductor#',i2,' Width:',
     &        ' {',1pd12.3,' > x > ',1pd12.3,'}')
902   format(/,' ENTER: No. of Seg. on Conductor#',i2,' Thickness:',
     &        ' {',1pd12.3,' > y > ',1pd12.3,'}')
904   format(/,' ENTER: "Filter Iteration"')
c-----------------------------------------------------------------------
c     OUTPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
910   format(/,' npt1',i4,' ndir',i4,' nofe',i4,' nn',i4,/)
912   format(/,' Conductor    # Nodes    # Triangles',/)
914   format('     ',i2,'           ',i2,'           ',i2)
916   format(/,' noie',i4,/)
c-----------------------------------------------------------------------
      return
      end

      subroutine qmat(nord)
      IMPLICIT NONE
      integer*4 q1(6,6),q2(6,6),q3(6,6),iden
      integer*4 pq2(2,6),pq3(2,6)
      integer*4 nord,n,i1,i2,i3,j1,j2,j3
c-----
      common /int5/ q1,q2,q3,iden
c-----
      data pq2/3,6,1,3,2,5,0,1,0,2,0,4/
      data pq3/2,4,3,5,1,2,0,6,0,3,0,1/
c-----
      n=((nord+1)*(nord+2))/2
      do 10 i1=1,n
        do 20 j1=1,n
          q1(i1,j1)=0
          q2(i1,j1)=0
          q3(i1,j1)=0
20      continue
10    continue
c-----
      if(nord.eq.1) then
        iden=2
        q1(2,2)=1
        q1(3,3)=1
        q1(3,2)=-1
      else
        iden=6
        q1(3,2)=-8
        q1(5,4)=-4
        q1(6,4)=1
        q1(6,5)=-4
        do 30 i1=2,n
          q1(i1,i1)=8
          if((i1.eq.4).or.(i1.eq.6)) q1(i1,i1)=3
30      continue
      end if
c-----
      do 40 i1=1,n
        do 50 j1=i1+1,n
          q1(i1,j1)=q1(j1,i1)
50      continue
40    continue
c-----
      do 60 i1=1,n
        do 70 j1=1,n
          q2(i1,j1)=q1(pq2(nord,i1),pq2(nord,j1))
          q3(i1,j1)=q1(pq3(nord,i1),pq3(nord,j1))
70      continue
60    continue
c-----
      return
      end
```

```
      subroutine solve
      IMPLICIT NONE
      real*8 top,bottom,right,left
      real*8 dl(5),er(4)
      real*8 x(1160),y(1160),pot(1160)
      real*8 a(1070,1070),b(1070),s(6,6)
      real*8 sinfa1(2,2),sinfb1(2,2),dena1,denb1,delta,ref
      real*8 sinfa2(3,3),sinfb2(3,3),dena2,denb2
      real*8 112,113,123,cosa1,cosa2,cosa3,cota1,cota2,cota3
c-----
      real*8 work(1070)
      integer*4 pivot(1070),dimb/1070/,job/0/
      real*8 rcond
c-----
      integer*4 iv(1500,6),ieps(1500)
      integer*4 q1(6,6),q2(6,6),q3(6,6),iden
      integer*4 ncon,nord,nogp,nod
      integer*4 nofe,nptl,ndir,nn
      integer*4 nds(1160)
      integer*4 i,j,k,l,m,n
      integer*4 itr,n1,n2,n3,nb1,ndb1,i1,j1
      integer*4 noie,ivinf(500,3),epsinf(500)
c-----
      logical nd(1160)
c-----
      common /real/ top,bottom,right,left
      common /rea3/ dl,er
      common /rea4/ x,y,pot
c-----
      common /int1/ ncon,nord,nogp,nod
      common /int2/ nofe,nptl,ndir,nn
      common /int3/ iv,ieps
      common /int5/ q1,q2,q3,iden
      common /int6/ nds
      common /int7/ noie,ivinf,epsinf
c-----
      common /log1/ nd
c-----
      data sinfa1/2,1,1,2/
      data sinfb1/1,-1,-1,1/
      data sinfa2/4,2,-1,2,16,2,-1,2,4/
      data sinfb2/7,-8,1,-8,16,-8,1,-8,7/
c-----
      dena1=18
      denb1=1
      dena2=90
      denb2=3
c-----
      n=((nord+1)*(nord+2))/2
      do 10 i=1,nn
        b(i)=0.0d0
        do 20 j=1,nn
          a(i,j)=0.0d0
20      continue
10    continue
c-----
      do 100 itr=1,nofe
        n1=iv(itr,1)
        n2=iv(itr,n-nord)
        n3=iv(itr,n)
        112=(x(n1)-x(n2))*(x(n1)-x(n2))
     &     +(y(n1)-y(n2))*(y(n1)-y(n2))
        113=(x(n1)-x(n3))*(x(n1)-x(n3))
     &     +(y(n1)-y(n3))*(y(n1)-y(n3))
        123=(x(n2)-x(n3))*(x(n2)-x(n3))
     &     +(y(n2)-y(n3))*(y(n2)-y(n3))
        cosa1=(112+113-123)/(2.0d0*dsqrt(112*113))
        cosa2=(112+123-113)/(2.0d0*dsqrt(112*123))
        cosa3=(113+123-112)/(2.0d0*dsqrt(113*123))
        cota1=cosa1/dsqrt(1.0d0-cosa1*cosa1)
        cota2=cosa2/dsqrt(1.0d0-cosa2*cosa3)
        cota3=cosa3/dsqrt(1.0d0-cosa2*cosa3)
c-------
        do 110 i=1,n
          do 120 j=1,n
```

```
            s(i,j)=(cota1*q1(i,j)+cota2*q2(i,j)+cota3*q3(i,j))
     &             /dble(iden)
120       continue
110     continue
c-------
        do 150 i=1,n
          i1=iv(itr,i)
          if(nd(i1)) go to 150
          i1=i1-nds(i1)
          do 160 j=1,n
            j1=iv(itr,j)
            if(nd(j1)) then
              b(i1)=b(i1)-pot(iv(itr,j))*er(ieps(itr))*s(i,j)
            else
              j1=j1-nds(j1)
              a(i1,j1)=a(i1,j1)+er(ieps(itr))*s(i,j)
            end if
160       continue
150     continue
c-------
100   continue
c-----
      if(nogp.eq.1) then
        do 165 i=1,npt1
          if((x(i).eq.left).and.(y(i).eq.top)) n1=i
          if((x(i).eq.right).and.(y(i).eq.top)) n2=i
165     continue
        i1=n1-nds(n1)
        a(i1,i1)=a(i1,i1)+(dabs(x(n1)/y(n1))+dabs(y(n1)/x(n1)))/3.0d0
        i1=n2-nds(n2)
        a(i1,i1)=a(i1,i1)+(dabs(x(n2)/y(n2))+dabs(y(n2)/x(n2)))/3.0d0
      end if
      do 170 itr=1,noie
        n1=ivinf(itr,1)
        n2=ivinf(itr,nord+1)
        delta=0.0d0
        ref=0.0d0
        if(x(n1).eq.x(n2)) then
          ref=dabs(x(n1))
          delta=dabs(y(n1)-y(n2))
        end if
        if(y(n1).eq.y(n2)) then
          ref=dabs(y(n1))
          delta=dabs(x(n1)-x(n2))
        end if
c-------
        if(nord.eq.1) then
          do 171 i=1,nord+1
            do 172 j=1,nord+1
              s(i,j)=(sinfa1(i,j)*delta/(ref*dena1))
     &               +(sinfb1(i,j)*ref/(delta*denb1))
172         continue
171       continue
        end if
        if(nord.eq.2) then
          do 173 i=1,nord+1
            do 174 j=1,nord+1
              s(i,j)=(sinfa2(i,j)*delta/(ref*dena2))
     &               +(sinfb2(i,j)*ref/(delta*denb2))
174         continue
173       continue
        end if
c-------
        do 180 i=1,nord+1
          i1=ivinf(itr,i)
          if(nd(i1)) go to 180
          i1=i1-nds(i1)
          do 190 j=1,nord+1
            j1=ivinf(itr,j)
            if(nd(j1)) then
              b(i1)=b(i1)-pot(iv(itr,j))*er(epsinf(itr))*s(i,j)
            else
              j1=j1-nds(j1)
              a(i1,j1)=a(i1,j1)+er(epsinf(itr))*s(i,j)
            end if
```

```
190       continue
180     continue
c-------
170   continue
c-----
      call zgeco (a,dimb,nn,pivot,rcond,work)
      write(*,*) 'Matrix condition number (estimate) = ',rcond
      call zgesl (a,dimb,nn,pivot,b,job)
c-----
      do 200 i=1,nptl
        j=i
        if(nd(j)) go to 200
        j=j-nds(j)
        pot(i)=b(j)
200   continue
c-----
      return
      end
      subroutine energy(icon)
      IMPLICIT NONE
      real*8 top,bottom,right,left
      real*8 dl(5),er(4)
      real*8 x(1160),y(1160),pot(1160)
      real*8 s(6,6),c(5)
      real*8 112,113,123,cosa1,cosa2,cosa3,cota1,cota2,cota3
      real*8 sinfa1(2,2),sinfb1(2,2),dena1,denb1,delta,ref
      real*8 sinfa2(3,3),sinfb2(3,3),dena2,denb2
c-----
      integer*4 iv(1500,6),ieps(1500)
      integer*4 q1(6,6),q2(6,6),q3(6,6),iden
      integer*4 ncon,nord,nogp,nod
      integer*4 nofe,nptl,ndir,nn
      integer*4 i,j,k,l,m,n,icon
      integer*4 itr,n1,n2,n3
      integer*4 noie,ivinf(500,3),epsinf(500)
c-----
      common /real/ top,bottom,right,left
      common /rea3/ dl,er
      common /rea4/ x,y,pot
c-----
      common /int1/ ncon,nord,nogp,nod
      common /int2/ nofe,nptl,ndir,nn
      common /int3/ iv,ieps
      common /int5/ q1,q2,q3,iden
      common /int7/ noie,ivinf,epsinf
c-----
      data sinfa1/2,1,1,2/
      data sinfb1/1,-1,-1,1/
      data sinfa2/4,2,-1,2,16,2,-1,2,4/
      data sinfb2/7,-8,1,-8,16,-8,1,-8,7/
c-----
      dena1=18
      denb1=1
      dena2=90
      denb2=3
c-----
      c(icon)=0.0d0
      n=((nord+1)*(nord+2))/2
      do 100 itr=1,nofe
        n1=iv(itr,1)
        n2=iv(itr,n-nord)
        n3=iv(itr,n)
        112=(x(n1)-x(n2))*(x(n1)-x(n2))
     &     +(y(n1)-y(n2))*(y(n1)-y(n2))
        113=(x(n1)-x(n3))*(x(n1)-x(n3))
     &     +(y(n1)-y(n3))*(y(n1)-y(n3))
        123=(x(n2)-x(n3))*(x(n2)-x(n3))
     &     +(y(n2)-y(n3))*(y(n2)-y(n3))
        cosa1=(112+113-123)/(2.0d0*dsqrt(112*113))
        cosa2=(112+123-113)/(2.0d0*dsqrt(112*123))
        cosa3=(113+123-112)/(2.0d0*dsqrt(113*123))
        cota1=cosa1/dsqrt(1.0d0-cosa1*cosa1)
        cota2=cosa2/dsqrt(1.0d0-cosa2*cosa3)
        cota3=cosa3/dsqrt(1.0d0-cosa2*cosa3)
```

```
c-------
        do 110 i=1,n
          do 120 j=1,n
            s(i,j)=(cota1*q1(i,j)+cota2*q2(i,j)+cota3*q3(i,j))
     &             /dble(iden)
120       continue
110     continue
c-------
        do 130 i=1,n
          do 140 j=1,n
            c(icon)=c(icon)+er(ieps(itr))*
     &              pot(iv(itr,i))*s(i,j)*pot(iv(itr,j))
140       continue
130     continue
c-------
100   continue
c-----
      write(5,900)
      write(5,902) icon,icon,8.8542d0*c(icon)
c-----
      if(nogp.eq.1) then
        do 165 i=1,nptl
          if((x(i).eq.left).and.(y(i).eq.top)) n1=i
          if((x(i).eq.right).and.(y(i).eq.top)) n2=i
165     continue
        s(1,1)=(dabs(x(n1)/y(n1))+dabs(y(n1)/x(n1)))/3.0d0
        c(icon)=c(icon)+pot(n1)*s(1,1)*pot(n1)
        s(1,1)=(dabs(x(n2)/y(n2))+dabs(y(n2)/x(n2)))/3.0d0
        c(icon)=c(icon)+pot(n2)*s(1,1)*pot(n2)
      end if
      do 170 itr=1,noie
        n1=ivinf(itr,1)
        n2=ivinf(itr,nord+1)
        delta=0.0d0
        ref=0.0d0
        if(x(n1).eq.x(n2)) then
          ref=dabs(x(n1))
          delta=dabs(y(n1)-y(n2))
        end if
        if(y(n1).eq.y(n2)) then
          ref=dabs(y(n1))
          delta=dabs(x(n1)-x(n2))
        end if
c-------
        if(nord.eq.1) then
          do 171 i=1,nord+1
            do 172 j=1,nord+1
              s(i,j)=(sinfa1(i,j)*delta/(ref*dena1))
     &               +(sinfb1(i,j)*ref/(delta*denb1))
172         continue
171       continue
        end if
        if(nord.eq.2) then
          do 173 i=1,nord+1
            do 174 j=1,nord+1
              s(i,j)=(sinfa2(i,j)*delta/(ref*dena2))
     &               +(sinfb2(i,j)*ref/(delta*denb2))
174         continue
173       continue
        end if
c-------
        do 180 i=1,nord+1
          do 190 j=1,nord+1
            c(icon)=c(icon)+er(epsinf(itr))*
     &              pot(ivinf(itr,i))*s(i,j)*pot(ivinf(itr,j))
190       continue
180     continue
c-------
170   continue
c-----
      write(5,900)
      write(5,902) icon,icon,8.8542d0*c(icon)
c-----
900   format(/,' Energy Formulation',/)
902   format(' C(',i1,',',i1,') ',1pd12.4,' pF/m')
```

```
c-----
      return
      end
      subroutine charge(icon)
      real*8 dl(5),er(4)
      real*8 x(1160),y(1160),pot(1160)
      real*8 c(5,5),x13,x21,x32,y13,y21,y32
      integer*4 iv(1500,6),ieps(1500)
      integer*4 ncon,nord,nogp,nod
      integer*4 int(5,50),icn(5,50),nt(5),cn(5)
      integer*4 n1,n2,n3,n4,n5,n6
      integer*4 i,j,k,l,m,n,icon
c-----
      common /rea3/ dl,er
      common /rea4/ x,y,pot
c-----
      common /int1/ ncon,nord,nogp,nod
      common /int3/ iv,ieps
      common /int4/ int,icn,nt,cn
c-----
      write(5,900)
      n=(nord+1)*(nord+2)/2
      do 10 i=1,ncon
        c(icon,i)=0.0d0
        do 20 j=1,nt(i)
          k=int(i,j)
          n1=iv(k,1)
          n2=iv(k,n-nord)
          n3=iv(k,n)
          x13=x(n1)-x(n3)
          x21=x(n2)-x(n1)
          x32=x(n3)-x(n2)
          y13=y(n1)-y(n3)
          y21=y(n2)-y(n1)
          y32=y(n3)-y(n2)
c-----
          n1=iv(k,1)
          n2=iv(k,2)
          n3=iv(k,3)
          n4=iv(k,4)
          n5=iv(k,5)
          n6=iv(k,6)
c-----
          if(y21.eq.0.0d0) then
            if(dabs(y13).ne.dabs(y32)) stop
            if(nord.eq.1) then
              c(icon,i)=c(icon,i)-er(ieps(k))*(
     &            x32*pot(n1)+x13*pot(n2)+x21*pot(n3))/y32
            else
              c(icon,i)=c(icon,i)-er(ieps(k))*(
     &            x32*pot(n1)+x13*pot(n4)-x21*pot(n6)
     &            +2.0d0*x21*(pot(n3)+pot(n5)-pot(n2)))/y32
            end if
          end if
c-----
          if(x21.eq.0.0d0) then
            if(dabs(x13).ne.dabs(x32)) stop
            if(nord.eq.1) then
              c(icon,i)=c(icon,i)+er(ieps(k))*(
     &            y32*pot(n1)+y13*pot(n2)+y21*pot(n3))/x32
            else
              c(icon,i)=c(icon,i)+er(ieps(k))*(
     &            y32*pot(n1)+y13*pot(n4)-y21*pot(n6)
     &            +2.0d0*y21*(pot(n3)+pot(n5)-pot(n2)))/x32
            end if
          end if
c-----
20      continue
        c(icon,i)=8.8542d0*c(icon,i)
        write(5,902) icon,i,c(icon,i)
10    continue
c-----
900   format(/,' Charge Formulation',/)
902   format(' C(',i1,',',i1,') ',1pd12.4,' pF/m')
```

```
c-----
      return
      end
      subroutine seciv
      IMPLICIT NONE
      real*8 x(1160),y(1160),pot(1160)
c-----
      integer*4 nofe,nptl,ndir,nn
      integer*4 iv(1500,6),ieps(1500)
      integer*4 itr,i,j,n1,n2,nc,in1(3),in2(3),inc(3)
c-----
      logical nd(1160)
c-----
      common /rea4/ x,y,pot
c-----
      common /int2/ nofe,nptl,ndir,nn
      common /int3/ iv,ieps
c-----
      common /log1/ nd
c-----
      data in1/1,4,6/
      data in2/4,6,1/
      data inc/2,5,3/
c-----
      do 10 itr=1,nofe
        iv(itr,4)=iv(itr,2)
        iv(itr,6)=iv(itr,3)
        iv(itr,2)=0
        iv(itr,3)=0
        iv(itr,5)=0
10    continue
c-----
      do 100 itr=1,nofe
        do 200 i=1,3
          n1=iv(itr,in1(i))
          n2=iv(itr,in2(i))
          nc=inc(i)
          if(iv(itr,nc).ne.0) go to 200
          iv(itr,nc)=nptl+1
          x(nptl+1)=(x(n1)+x(n2))/2.0d0
          y(nptl+1)=(y(n1)+y(n2))/2.0d0
          nd(nptl+1)=.false.
          if((nd(n1)).and.(nd(n2))) nd(nptl+1)=.true.
          nptl=nptl+1
          if(itr.eq.nofe) go to 200
c---------
          do 300 j=itr+1,nofe
            if(((iv(j,1).eq.n1).and.(iv(j,4).eq.n2)).or.
     &         ((iv(j,1).eq.n2).and.(iv(j,4).eq.n1))) then
              iv(j,2)=nptl
              go to 200
            end if
            if(((iv(j,1).eq.n1).and.(iv(j,6).eq.n2)).or.
     &         ((iv(j,1).eq.n2).and.(iv(j,6).eq.n1))) then
              iv(j,3)=nptl
              go to 200
            end if
            if(((iv(j,4).eq.n1).and.(iv(j,6).eq.n2)).or.
     &         ((iv(j,4).eq.n2).and.(iv(j,6).eq.n1))) then
             iv(j,5)=nptl
             go to 200
            end if
300       continue
c---------
200     continue
100   continue
c-----
      return
      end
```

```
      subroutine filter(ifil,np,ibound)
      IMPLICIT NONE
      real*8 top,bottom,right,left
      real*8 dl(5),er(4)
      real*8 x(1160),y(1160),pot(1160)
      real*8 xnew,ynew
c-----
      integer*4 ncon,nord,nogp,nod
      integer*4 nofe,nptl,ndir,nn
      integer*4 iv(1500,6),ieps(1500)
      integer*4 int(5,50),icn(5,50),nt(5),cn(5)
      integer*4 i,j,k,l,m,n
      integer*4 ifil,np(6),ibound(5,60)
      integer*4 filt(40),n1,n2,n3
c-----
      common/real/ top,bottom,right,left
      common/rea3/ dl,er
      common/rea4/ x,y,pot
c-----
      common/int1/ ncon,nord,nogp,nod
      common/int2/ nofe,nptl,ndir,nn
      common/int3/ iv,ieps
      common/int4/ int,icn,nt,cn
c-----
      do 100 m=1,ifil
        do 200 k=1,nptl
          do 300 i=1,ncon
            do 310 j=1,np(i+1)
              if(k.eq.ibound(i,j)) go to 200
310         continue
300       continue
          do 330 i=1,ncon
            do 340 j=1,cn(i)
              if(k.eq.icn(i,j)) go to 200
340         continue
330       continue
          do 360 i=1,nod+1
            if((x(k).eq.left).and.(y(k).eq.dl(i))) go to 200
            if((x(k).eq.right).and.(y(k).eq.dl(i))) go to 200
360       continue
          n=0
          do 400 i=1,nofe
            if(iv(i,1).eq.k) then
              n2=0
              n3=0
              do 430 j=1,n
                if(filt(j).eq.iv(i,2)) n2=1
                if(filt(j).eq.iv(i,3)) n3=1
430           continue
              if(n2.eq.0) then
                filt(n+1)=iv(i,2)
                n=n+1
              end if
              if(n3.eq.0) then
                filt(n+1)=iv(i,3)
                n=n+1
              end if
            end if
            if(iv(i,2).eq.k) then
              n1=0
              n3=0
              do 460 j=1,n
                if(filt(j).eq.iv(i,1)) n1=1
                if(filt(j).eq.iv(i,3)) n3=1
460           continue
              if(n1.eq.0) then
                filt(n+1)=iv(i,1)
                n=n+1
              end if
              if(n3.eq.0) then
                filt(n+1)=iv(i,3)
                n=n+1
              end if
            end if
            if(iv(i,3).eq.k) then
```

```
              n1=0
              n2=0
              do 490 j=1,n
                if(filt(j).eq.iv(i,1)) n1=1
                if(filt(j).eq.iv(i,2)) n2=1
490           continue
              if(n1.eq.0) then
                filt(n+1)=iv(i,1)
                n=n+1
              end if
              if(n2.eq.0) then
                filt(n+1)=iv(i,2)
                n=n+1
              end if
            end if
400       continue
          if(n.gt.40) stop
          xnew=0.0d0
          ynew=0.0d0
          do 500 i=1,n
            xnew=xnew+x(filt(i))
            ynew=ynew+y(filt(i))
500       continue
          if((x(k).eq.right).or.(x(k).eq.left)) then
            y(k)=ynew/dble(n)
            go to 200
          end if
          do 600 i=1,nod+1
            if(y(k).eq.dl(i)) then
              x(k)=xnew/dble(n)
              go to 200
            end if
600       continue
          x(k)=xnew/dble(n)
          y(k)=ynew/dble(n)
200     continue
100   continue
c-----
      return
      end

      subroutine infem
      IMPLICIT NONE
      real*8 top,bottom,right,left
      real*8 dl(5),er(4)
      real*8 x(1160),y(1160),pot(1160)
c-----
      integer*4 ncon,nord,nogp,nod
      integer*4 nofe,nptl,ndir,nn
      integer*4 iv(1500,6),ieps(1500)
      integer*4 noie,ivinf(500,3),epsinf(500)
      integer*4 i,j,k,l,m,n,n1,n2,n3
c-----
      common /real/ top,bottom,right,left
      common /rea3/ dl,er
      common /rea4/ x,y,pot
c-----
      common /int1/ ncon,nord,nogp,nod
      common /int2/ nofe,nptl,ndir,nn
      common /int3/ iv,ieps
      common /int7/ noie,ivinf,epsinf
c-----
      noie=0
      do 100 i=1,nofe
        n=(nord+1)*(nord+2)/2
        n1=iv(i,1)
        n2=iv(i,n-nord)
        n3=iv(i,n)
        if((x(n1).eq.left).and.(x(n2).eq.left)) then
          if(nord.eq.1) then
            ivinf(noie+1,1)=n1
            ivinf(noie+1,2)=n2
          else
            ivinf(noie+1,1)=n1
            ivinf(noie+1,2)=iv(i,2)
```

```
      ivinf(noie+1,3)=n2
    end if
    noie=noie+1
  end if
  if((x(n1).eq.left).and.(x(n3).eq.left)) then
    if(nord.eq.1) then
      ivinf(noie+1,1)=n1
      ivinf(noie+1,2)=n3
    else
      ivinf(noie+1,1)=n1
      ivinf(noie+1,2)=iv(i,3)
      ivinf(noie+1,3)=n3
    end if
    noie=noie+1
  end if
  if((x(n2).eq.left).and.(x(n3).eq.left)) then
    if(nord.eq.1) then
      ivinf(noie+1,1)=n2
      ivinf(noie+1,2)=n3
    else
      ivinf(noie+1,1)=n2
      ivinf(noie+1,2)=iv(i,5)
      ivinf(noie+1,3)=n3
    end if
    noie=noie+1
  end if
  if((x(n1).eq.right).and.(x(n2).eq.right)) then
    if(nord.eq.1) then
      ivinf(noie+1,1)=n1
      ivinf(noie+1,2)=n2
    else
      ivinf(noie+1,1)=n1
      ivinf(noie+1,2)=iv(i,2)
      ivinf(noie+1,3)=n2
    end if
    noie=noie+1
  end if
  if((x(n1).eq.right).and.(x(n3).eq.right)) then
    if(nord.eq.1) then
      ivinf(noie+1,1)=n1
      ivinf(noie+1,2)=n3
    else
      ivinf(noie+1,1)=n1
      ivinf(noie+1,2)=iv(i,3)
      ivinf(noie+1,3)=n3
    end if
    noie=noie+1
  end if
  if((x(n2).eq.right).and.(x(n3).eq.right)) then
    if(nord.eq.1) then
      ivinf(noie+1,1)=n2
      ivinf(noie+1,2)=n3
    else
      ivinf(noie+1,1)=n2
      ivinf(noie+1,2)=iv(i,5)
      ivinf(noie+1,3)=n3
    end if
    noie=noie+1
  end if
  if((y(n1).eq.top).and.(y(n2).eq.top)) then
    if(nord.eq.1) then
      ivinf(noie+1,1)=n1
      ivinf(noie+1,2)=n2
    else
      ivinf(noie+1,1)=n1
      ivinf(noie+1,2)=iv(i,2)
      ivinf(noie+1,3)=n2
    end if
    noie=noie+1
  end if
  if((y(n1).eq.top).and.(y(n3).eq.top)) then
    if(nord.eq.1) then
      ivinf(noie+1,1)=n1
      ivinf(noie+1,2)=n3
    else
```

```
            ivinf(noie+1,1)=n1
            ivinf(noie+1,2)=iv(i,3)
            ivinf(noie+1,3)=n3
          end if
          noie=noie+1
        end if
        if((y(n2).eq.top).and.(y(n3).eq.top)) then
          if(nord.eq.1) then
            ivinf(noie+1,1)=n2
            ivinf(noie+1,2)=n3
          else
            ivinf(noie+1,1)=n2
            ivinf(noie+1,2)=iv(i,5)
            ivinf(noie+1,3)=n3
          end if
          noie=noie+1
        end if
100   continue
c-----
      do 200 i=1,noie
        n1=ivinf(i,1)
        n2=ivinf(i,nord+1)
        k=0
        do 300 j=1,nod
          if((y(n1).ge.dl(j)).and.(y(n1).le.dl(j+1)).and.
     &       (y(n2).ge.dl(j)).and.(y(n2).le.dl(j+1))) k=j
300     continue
        if(k.eq.0) stop
        epsinf(i)=k
200   continue
c-----
      return
      end
```

Appendix B4: "A3CON.FOR"

Description: This Fortran program implements the analytic expressions derived in [13] which solve the transmission line equations for three conductor lines in homogeneous media. Program input consists of the per-unit length parameters, the length of the conductors, the permeablity and permittivity of the surrounding medium, the line termination impedances, and the amplitude and frequency of the excitation on the generator conductor. Program output are the near and far end crosstalk voltages across the termination impedances on the receptor conductor.

Calling Subroutines: - none.

Source Listing:

```
      IMPLICIT NONE
      real*8 freq,length
      real*8 cg,cm,cr,lg,lm,lr
      real*8 zog,zlg,zor,zlr
      real*8 beta,lamda,omega,pi,vel,eo,uo,cf,sf
      real*8 kcou,zcg,zcr,tr,tg
      real*8 aor,aog,alr,alg
      real*8 denr,deni
      real*8 pho,phl
c-----
      complex*16 vs,den,vgdc,igdc,vrl,vro
      complex*16 coml,com2,com3,com4
c-----
      open (23,file='a3con.out',status='new')
c-----------------------------------------------------------------
c     INITIALIZATIONS: FREQUENCY, LENGTH.
c-----------------------------------------------------------------
      freq=100.0d06
      length=0.24d0
c-----------------------------------------------------------------
c     INITIALIZATIONS: CAPACITANCE/INDUCTANCE VALUES.
c-----------------------------------------------------------------
      cg=10.0d-12
      cm=-1.0d-12
      cr=10.0d-12
c-----
      lg=1.1239d-6
```

```
      lm=0.11239d-6
      lr=1.1239d-6
c-----------------------------------------------------------------------
c     INITIALIZATIONS: RESISTIVE TERMINATION VALUES.
c-----------------------------------------------------------------------
      zog=0.0d0
      zlg=50.0d0
      zor=50.0d0
      zlr=50.0d0
c-----------------------------------------------------------------------
      pi=dacos(-1.0d0)
      eo=8.8542d-12
      uo=4.0d-7*pi
c-----
      vel=1.0d0/dsqrt(eo*uo)
      lamda=vel/freq
      beta=(2.0d0*pi)/lamda
      omega=2.0d0*pi*freq
c-----
      vs=dcmplx(1.0d0,0.0d0)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      kcou=lm/dsqrt(lg*lr)
      zcg=vel*lg*dsqrt(1.0d0-(kcou*kcou))
      zcr=vel*lr*dsqrt(1.0d0-(kcou*kcou))
c-----
      aor=zor/zcr
      aog=zog/zcg
      alr=zlr/zcr
      alg=zlg/zcg
c-----
      tr=(1.0d0+aor*alr)*(lr*length)/(zor+zlr)
      tg=(1.0d0+aog*alg)*(lg*length)/(zog+zlg)
c-----
      cf=dcos(beta*length)
      sf=dsin(beta*length)/(beta*length)
c-----
      deni=omega*cf*sf*(tr+tg)
      denr=((cf*cf)-((sf*sf*omega*omega*tr*tg)*
     &  (1.0d0-(kcou*kcou*((1.0d0-aog*alr)*(1.0d0-alg*aor))/
     &              ((1.0d0+aor*alr)*(1.0d0+aog*alg))))))
      den=dcmplx(denr,deni)
c-----
      vgdc=zlg*vs/(zog+zlg)
      igdc=vs/(zog+zlg)
c-----
      com1=dcmplx(0.0d0,zlr*omega*lm*length/(zor+zlr))
      com2=dcmplx(0.0d0,-zor*zlr*omega*cm*length/(zor+zlr))
      vrl=sf*(com2*vgdc-com1*igdc)/den
      denr=dreal(vrl)
      deni=dimag(vrl)
      phl=datan2(deni,denr)
c-----
      com1=dcmplx(0.0d0,zor*omega*lm*length/(zor+zlr))
      com2=dcmplx(cf,beta*length*alg*sf/dsqrt(1.0d0-(kcou*kcou)))
      com3=dcmplx(0.0d0,-zor*zlr*omega*cm*length/(zor+zlr))
      com4=dcmplx(cf,beta*length*sf/(alg*dsqrt(1.0d0-(kcou*kcou))))
      vro=sf*(com1*com2*igdc+com3*com4*vgdc)/den
      denr=dreal(vro)
      deni=dimag(vro)
      pho=datan2(deni,denr)
c-----------------------------------------------------------------------
c     OUTPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
      write(23,fmt=100) 1.0d-6*freq,length
      write(23,fmt=102) 1.0d12*cg,1.0d12*cm,1.0d12*cr
      write(23,fmt=104) 1.0d6*lg,1.0d6*lm,1.0d6*lr
      write(23,fmt=106) zog,zlg
      write(23,fmt=108) zor,zlr
      write(23,fmt=110) 1.0d3*cdabs(vro),180.0d0*pho/pi,
     &                  1.0d3*cdabs(vrl),180.0d0*phl/pi
c-----
100   format(/,' Frequency=',1PE12.3,' MHz',/,
     &         ' Length   =',1PE12.3,' m')
102   format(/,' Capacitance Values:',' Cg',1pd12.3,' pF/m',/,
```

```
     &          '                      ',' Cm',1pd12.3,' pF/m',/,
     &          '                      ',' Cr',1pd12.3,' pF/m')
104   format(/,' Inductance  Values:',' lg',1pd12.3,' uH/m',/,
     &          '                      ',' lm',1pd12.3,' uH/m',/,
     &          '                      ',' lr',1pd12.3,' uH/m')
106   format(/,' Generator  Circuit:',' Zg(0)',1pd12.3,' ohms',/,
     &          '                      ',' Zg(L)',1pd12.3,' ohms')
108   format(/,' Receptor   Circuit:',' Zg(0)',1pd12.3,' ohms',/,
     &          '                      ',' Zg(L)',1pd12.3,' ohms')
110   format(/,/,'          MAGNITUDE [mV]        PHASE [Deg]',/,/,
     &        ' Vnext:',1PD12.3,'         ',1pd12.3,/,
     &        ' Vfext:',1PD12.3,'         ',1pd12.3,/)
c-----
      close (23)
c-----
      stop
      end
```

Appendix B5: "ZTLE.FOR"

Description: This program solves the frequency domain multiconductor transmission line equations for a configuration of 'n+1' lossy parallel conductors (i.e. 'n' conductors, 1 reference conductor) of total length 'L'. The required input parameters are the excitation amplitude, phase, and frequency, the length of the conductors, the per-unit length capacitance, inductance matrices, conductance, and resistance matricies, and the line termination impedances. The output parameters are the near and far end voltages across each of the line terminations.

Calling Subroutines:

- "MPRT": Prints out the entries in a matrix.
- "CSQMINV": Inverts a complex matrix.
- "CSQMUX": Multiplies two complex matrices.
- "CDSIMQ": Solves a complex system of equations.
- "CG": Computes the Complex Eigenvalues/Vectors of a complex matrix.

Source Listing:

```
      IMPLICIT LOGICAL (A-Z)
      complex*16 z(6,6),y(6,6),yinv(6,6),yz(6,6)
      complex*16 t(6,6),yinvt(6,6),gamma(6)
      complex*16 zo(6,6),zl(6,6),vo(6),vl(6)
      complex*16 zot(6,6),zlt(6,6)
c-----
      real*8 ar(6,6),ai(6,6),wr(6),wi(6),zr(6,6),zi(6,6)
      real*8 fv1(6),fv2(6),fv3(6)
      integer*4 nm,n,matz,ierr
c-----
      real*8 c(6,6),l(6,6)
      real*8 g(6,6),r(6,6)
```

```
      real*8 freq,length,dx
      real*8 pi,omega,value,sc
      real*8 im,re,magv,phv,magi,phi
c-----
      complex*16 b(12),a(144),zz
      complex*16 a11(6,6),a12(6,6),a21(6,6),a22(6,6)
      complex*16 vx(9,101),ix(9,101)
c-----
      integer*4 ncon
      integer*4 nunit,ns,nn,nc,nrad
      integer*4 i,j,k,ks,nsx
c-----
      open(55,file='ZTLE.DAT',status='OLD')
      open(66,file='ZTLE.OUT',status='NEW')
c-----------------------------------------------------------------------
c     INPUT: - NUMBER OF CONDUCTORS "NCON".
c-----------------------------------------------------------------------
      read(55,*) ncon
      if((ncon.lt.1).or.(ncon.gt.9)) stop
c-----------------------------------------------------------------------
c     INPUT: - CAPACITANCE MATRIX "[C]".
c            - INDUCTANCE MATRIX  "[L]".
c-----------------------------------------------------------------------
      do 10 nc=1,ncon
        do 15 k=nc,ncon
          read(55,*) value
          c(nc,k)=1.0d-12*value
          if(nc.ne.k) c(k,nc)=c(nc,k)
15      continue
10    continue
c-----
      do 20 nc=1,ncon
        do 25 k=nc,ncon
          read(55,*) value
          l(nc,k)=1.0d-6*value
          if(nc.ne.k) l(k,nc)=l(nc,k)
25      continue
20    continue
c-----------------------------------------------------------------------
c     INPUT: - RESISTANCE MATRIX  "[R]".
c            - CONDUCTANCE MATRIX "[G]".
c-----------------------------------------------------------------------
      do 30 nc=1,ncon
        do 35 k=nc,ncon
          read(55,*) value
          r(nc,k)=value
          if(nc.ne.k) r(k,nc)=r(nc,k)
35      continue
30    continue
      do 40 nc=1,ncon
        do 45 k=nc,ncon
          read(55,*) value
          g(nc,k)=1.0d-6*value
          if(nc.ne.k) g(k,nc)=g(nc,k)
45      continue
40    continue
c-----------------------------------------------------------------------
c     INPUT: - TERMINATION IMPEDANCES MATRIX "[Z0]".
c            - VOLTAGE EXCITATION MATRIX     "[ZL]".
c-----------------------------------------------------------------------
      do 50 nc=1,ncon
        do 55 k=nc,ncon
          read(55,*) zo(nc,k)
          if(nc.ne.k) zo(k,nc)=zo(nc,k)
55      continue
50    continue
      do 60 nc=1,ncon
        do 65 k=nc,ncon
          read(55,*) zl(nc,k)
          if(nc.ne.k) zl(k,nc)=zl(nc,k)
65      continue
60    continue
c-----
      do 70 nc=1,ncon
        read(55,*) vo(nc)
```

```
70    continue
      do 75 nc=1,ncon
        read(55,*) vl(nc)
75    continue
c-----------------------------------------------------------------------
c     INPUT: - FREQUENCY OF EXCITATIONS "FREQ".
c            - LENGTH OF LINE "LENGTH".
c            - NUMBER OF SEGMENTS "NSX".
c-----------------------------------------------------------------------
      read(55,*) freq
      freq=1.0d6*freq
      read(55,*) length
      read(55,*) nsx
c-----------------------------------------------------------------------
c     OUTPUT: - INPUT DATA.
c-----------------------------------------------------------------------
      write(66,800) ncon,1.0D-6*freq,length,nsx
c-----
      sc=1.0d6
      call mprt(l,ncon,ncon,'INDUCTANCE [uH/m]',17,sc)
c-----
      sc=1.0d12
      call mprt(c,ncon,ncon,'CAPACITANCE [pF/m]',18,sc)
c-----
      sc=1.0d0
      call mprt(r,ncon,ncon,'RESISTANCE [ohms/m]',19,sc)
c-----
      sc=1.0d6
      call mprt(g,ncon,ncon,'CONDUCTANCE [S/m]',17,sc)
c-----
      write(66,820)
      do 80 i=1,ncon
        do 82 j=1,ncon
          write(66,810) i,j,zo(i,j)
82      continue
80    continue
c-----
      write(66,804)
      do 84 i=1,ncon
        do 86 j=1,ncon
          write(66,810) i,j,zl(i,j)
86      continue
84    continue
c-----
      write(66,806)
      do 90 i=1,ncon
        write(66,812) i,vo(i)
90    continue
c-----
      write(66,808)
      do 95 i=1,ncon
        write(66,812) i,vl(i)
95    continue
c-----------------------------------------------------------------------
c     COMPUTE: - IMPEDANCE MATRIX  "[Z]".
c              - ADMITTANCE MATRIX "[Y]".
c-----------------------------------------------------------------------
      pi=dacos(-1.0d0)
      omega=2.0d0*pi*freq
      do 100 i=1,ncon
        do 110 j=1,ncon
          z(i,j)=dcmplx(r(i,j),omega*l(i,j))
          y(i,j)=dcmplx(g(i,j),omega*c(i,j))
110     continue
100   continue
c-----------------------------------------------------------------------
c     COMPUTE: - INVERSE OF ADMITTANCE MATRIX "[YINV]".
c-----------------------------------------------------------------------
      call csqminv(ncon,y,yinv)
c-----------------------------------------------------------------------
c     COMPUTE: - PRODUCT MATRIX "[YZ]".
c-----------------------------------------------------------------------
      call csqmux(ncon,y,z,yz)
c-----------------------------------------------------------------------
c     COMPUTE: - EIGENVALUES AND EIGEN VECTORS OF PRODUCT MATRIX.
```

```
c-----------------------------------------------------------------------
      nm=6
      n=ncon
      do 120 i=1,ncon
        do 125 j=1,ncon
          ar(i,j)=dreal(yz(i,j))
          ai(i,j)=dimag(yz(i,j))
125     continue
120   continue
      matz=1
c-----
      call cg(nm,n,ar,ai,wr,wi,matz,zr,zi,fv1,fv2,fv3,ierr)
c-----------------------------------------------------------------------
c     COMPUTE: - PROPAGATION CONSTANTS "[GAMMA]".
c              - TRANSFORMATION MATRIX "[T]".
c-----------------------------------------------------------------------
      do 130 i=1,ncon
        gamma(i)=cdsqrt(dcmplx(wr(i),wi(i)))
        do 135 j=1,ncon
          t(i,j)=dcmplx(zr(i,j),zi(i,j))
135     continue
130   continue
c-----------------------------------------------------------------------
c     ASSEMBLE FINAL SYSTEM OF EQUATIONS.
c-----------------------------------------------------------------------
      call csqmux(ncon,yinv,t,yinvt)
      call csqmux(ncon,zo,t,zot)
      call csqmux(ncon,zl,t,zlt)
c-----
      do 140 i=1,ncon
        do 145 j=1,ncon
          yinvt(i,j)=yinvt(i,j)*gamma(j)
145     continue
140   continue
c-----
      do 150 i=1,ncon
        do 155 j=1,ncon
          a11(i,j)=yinvt(i,j)+zot(i,j)
          a12(i,j)=yinvt(i,j)-zot(i,j)
          a21(i,j)=(yinvt(i,j)-zlt(i,j))*cdexp(-gamma(j)*length)
          a22(i,j)=(yinvt(i,j)+zlt(i,j))*cdexp(+gamma(j)*length)
155     continue
150   continue
c-----
      nn=2*ncon
      do 160 i=1,ncon
        b(i)=vo(i)
        b(i+ncon)=vl(i)
        do 165 j=1,ncon
          a((j-1)*nn+i)=a11(i,j)
          a((j+ncon-1)*nn+i)=a12(i,j)
          a(j*nn+i-ncon)=a21(i,j)
          a((j+ncon)*nn+i-ncon)=a22(i,j)
165     continue
160   continue
c-----------------------------------------------------------------------
c     SOLVE FINAL SYSTEM OF EQUATIONS.
c-----------------------------------------------------------------------
      call cdsimq(a,b,nn,ks)
c-----
      write(*,950) ks
950   format(/,' ks=',i4,/)
c-----------------------------------------------------------------------
c     COMPUTE: LINE VOLTAGES AND CURRENTS.
c-----------------------------------------------------------------------
      do 200 k=1,nsx+1
        dx=dble(k-1)*length/dble(nsx)
        do 210 i=1,ncon
          ix(i,k)=dcmplx(0.0d0,0.0d0)
          vx(i,k)=dcmplx(0.0d0,0.0d0)
          do 220 j=1,ncon
            zz=gamma(j)*dx
            ix(i,k)=t(i,j)*(b(j)*cdexp(-zz)-
     &                b(j+ncon)*cdexp(zz))+ix(i,k)
            vx(i,k)=yinvt(i,j)*(b(j)*cdexp(-zz)+
```

```
     &              b(j+ncon)*cdexp(zz))+vx(i,k)
220       continue
210     continue
200   continue
c-----------------------------------------------------------------------
c     OUTPUT: LINE VOLTAGES AND CURRENTS.
c-----------------------------------------------------------------------
      do 300 i=1,ncon
        write(66,820) i
        write(66,822)
        do 310 k=1,nsx+1
          dx=dble(k-1)*length/dble(nsx)
          magv=cdabs(vx(i,k))
          re=dreal(vx(i,k))
          im=dimag(vx(i,k))
          if((re.eq.0.0d0).and.(im.eq.0.0d0)) phv=0.0d0
          if((re.eq.0.0d0).and.(im.gt.0.0d0)) phv=+90.0d0
          if((re.eq.0.0d0).and.(im.lt.0.0d0)) phv=-90.0d0
          if(re.ne.0.0d0) phv=180.0d0*datan2(im,re)/pi
          magi=cdabs(ix(i,k))
          re=dreal(ix(i,k))
          im=dimag(ix(i,k))
          if((re.eq.0.0d0).and.(im.eq.0.0d0)) phi=0.0d0
          if((re.eq.0.0d0).and.(im.gt.0.0d0)) phi=+90.0d0
          if((re.eq.0.0d0).and.(im.lt.0.0d0)) phi=-90.0d0
          if(re.ne.0.0d0) phi=180.0d0*datan2(im,re)/pi
          write(66,824) dx,magv,phv,magi,phi
310     continue
300   continue
c-----------------------------------------------------------------------
c     OUTPUT FORMAT STATEMENTS.
c-----------------------------------------------------------------------
800   format(/,' ncon:',i4,/,' freq:',1pd12.3,' MHz',/,
     &         ' length:',1pd12.3,' m',/,' nsx:',i6)
802   FORMAT(/,' Termination Impedance Matrix at (x=0): [ohms]',/,/)
804   FORMAT(/,' Termination Impedance Matrix at (x=L): [ohms]',/,/)
806   FORMAT(/,' Voltage Excitation Matrix at (x=0): [volts]',/,/)
808   FORMAT(/,' Voltage Exciation Matrix at (x=L): [volts]',/,/)
810   format(' Z(',i1,',',i1,') = (',1pd12.3,',',1pd12.3,') ')
812   format(' V(',i1,') = (',1pd12.3,',',1pd12.3,') ')
c-----
820   format(/,' Voltage/Current Distribution on Condutor #',i1)
822   format(/,'   Position ',
     &         '           Voltage       ',
     &         '           Current       ',/,
     &         '       [m]     ',
     &         ' Mag. [V]    Phase [deg]',
     &         ' Mag. [A]    Phase [deg]')
824   format('  ',1pd12.3,'  ',2(1pd12.3),'   ',2(1pd12.3))
c-----------------------------------------------------------------------
      close(55)
      close(66)
c-----
      stop
      end
      subroutine csqmux(m,a,b,ab)
      IMPLICIT LOGICAL (A-Z)
      complex*16 a(6,6),b(6,6),ab(6,6)
c-----
      integer*4 i,j,k,m
c-----
      do 10 i=1,m
        do 20 j=1,m
          ab(i,j)=dcmplx(0.0d0,0.0d0)
          do 30 k=1,m
            ab(i,j)=a(i,k)*b(k,j)+ab(i,j)
30        continue
20      continue
10    continue
c-----
      return
      end
```

Appendix B6: "NEC INPUT FILE"

Description: The following is the NEC input file used for computing the radiation results in Section 5.

```
CM  RADIATION FROM FIVE CONDUCTORS
CM
CM  5   10        20        30        40        50        60        70        80
CM
CE
GW  1   50      -0.15       0.0     0.001      0.15       0.0     0.001   0.00005
GW  2    5      -0.15       0.0       0.0     -0.15       0.0     0.001   0.00005
GW  3    5       0.15       0.0     0.001      0.15       0.0       0.0   0.00005
GW  4   50      -0.15    -0.002     0.001      0.15    -0.002     0.001   0.00005
GW  5    5      -0.15    -0.002       0.0     -0.15    -0.002     0.001   0.00005
GW  6    5       0.15    -0.002     0.001      0.15    -0.002       0.0   0.00005
GW  7   50      -0.15     0.002     0.001      0.15     0.002     0.001   0.00005
GW  8    5      -0.15     0.002       0.0     -0.15     0.002     0.001   0.00005
GW  9    5       0.15     0.002     0.001      0.15     0.002       0.0   0.00005
GW 10   50      -0.15    -0.004     0.001      0.15    -0.004     0.001   0.00005
GW 11    5      -0.15    -0.004       0.0     -0.15    -0.004     0.001   0.00005
GW 12    5       0.15    -0.004     0.001      0.15    -0.004       0.0   0.00005
GW 13   50      -0.15     0.004     0.001      0.15     0.004     0.001   0.00005
GW 14    5      -0.15     0.004       0.0     -0.15     0.004     0.001   0.00005
GW 15    5       0.15     0.004     0.001      0.15     0.004       0.0   0.00005
GE  1
GN  1
LD  0    2    3    3     50.0
LD  0    3    3    3    100.0
LD  0    5    3    3    100.0
LD  0    6    3    3    100.0
LD  0    8    3    3    100.0
LD  0    9    3    3    100.0
LD  0   11    3    3    100.0
LD  0   12    3    3    100.0
LD  0   14    3    3    100.0
LD  0   15    3    3    100.0
EX  0    2    3    0     1.00       0.0       0.0       0.0       0.0       0.0
FR  0    0    0    0    900.0       0.0       0.0       0.0       0.0       0.0
NE  1    1    1   37      3.0      45.0     -90.0       0.0       0.0       5.0
EN
```