APPLICATION OF THE TIME-DOMAIN FINITE-ELEMENT METHOD TO ANALYSIS OF 3D ELECTRIC MACHINE PROBLEMS

BY

PENG CHEN

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Adviser:

Professor Jianming Jin
The motivation of this work is to apply the time-domain finite-element method (TDFEM) to the simulation of 3D electric machine problems. The features of the problems might include low-frequency excitation, high inhomogeneity in the material parameters, complex geometries and nonlinearity in the materials. The proposed formulations and algorithm aim at solving these problems.

In this work, starting from time-domain Maxwell’s equations, we firstly derive the A formulation of the time-domain finite-element method. This serves as the basic version of TDFEM which could be used to simulate the simplest linear machine problems. Then, by testing the convergence of a racetrack coil problem, the validity of the linear formulation is verified. Afterwards, the incomplete LU preconditioner and Cuthill-McKee reordering (RCM) technique are introduced to ameliorate the condition of the system matrix. The effects of the material parameters and the RCM algorithm on the system matrix condition are analyzed. Also, the tree-cotree splitting (TCS) technique is applied to solve low-frequency problems. Several examples are simulated and corresponding results are shown to demonstrate the performance of the algorithms. Finally, the model of nonlinear machine problems is shown, and the cubic spline interpolation is employed to obtain a continuous B-H curve from the tabulated measured data. Both the Newton-Raphson method and the fixed-point method are introduced and applied to solve nonlinear machine problems. Some examples are simulated and the preliminary results are shown and discussed.
ACKNOWLEDGMENTS

First of all, I must thank my adviser Prof. Jin who constantly provides me with the most informative and valuable instruction for my research and study. I benefit vastly both from his classes, ECE 520 and ECE 540, which are full of humor and enthusiasm, and doing research under his guidance. I also greatly appreciate his generous and broad help at my toughest time when I was studying here. He sets us a very good example of an ideal adviser who is not only excellent in both teaching and researching, but also cares about his students’ lives. The years I spent studying with him will be of great significance in my life. Then, I should also thank Dr. Rui Wang who gave me lots of advice in both research and life when he was studying at UIUC. I should also thank my labmate Dr. Su Yan who taught me a lot in research during the past years. I was very fortunate to have had the chance to be the labmate with him. I also want to thank XiaoLei Li, Wang Yao, Mingfeng Xue and Huanting Meng who inspired me extensively in my research; and I obtained many helpful ideas from our discussions. I should also thank every other member in Prof. Jin’s group. I had a very good time studying and researching with them. Finally, I would like to thank my parents and my fiancée for their permanent care and love. No matter what happens, they will always be the origin of the greatest support and strength to me!
TABLE OF CONTENTS

LIST OF ABBREVIATIONS ........................................... v

CHAPTER 1  INTRODUCTION ........................................... 1

CHAPTER 2  FINITE-ELEMENT FORMULATION AND IMPLEMENTATION ........................................... 4
  2.1 Introduction ........................................... 4
  2.2 Basic TDFEM Formulation for Electrical Machine Problems ........................................... 4
  2.3 Finite-Element Discretization: Nodal and Edge Elements ........................................... 7
  2.4 Temporal Discretization in TDFEM ........................................... 10
  2.5 Figures ........................................... 14

CHAPTER 3  MODELING OF LINEAR MACHINE PROBLEMS ........................................... 16
  3.1 Introduction ........................................... 16
  3.2 Modeling Racetrack Coil Excitation ........................................... 16
  3.3 Convergence of Linear System with RCM Reordering and ILU Preconditioner ........................................... 17
  3.4 Tree-Cotree Splitting Algorithm for Low-frequency Problems ........................................... 26
  3.5 Figures and Tables ........................................... 29

CHAPTER 4  MODELING OF NONLINEAR MACHINE PROBLEMS ........................................... 44
  4.1 Introduction ........................................... 44
  4.2 Cubic Spline Interpolation ........................................... 45
  4.3 TDFEM Formulation with Newton-Raphson Method ........................................... 47
  4.4 TDFEM Formulation with Fixed-Point Method ........................................... 49
  4.5 Numerical Examples and Results ........................................... 50
  4.6 Figures ........................................... 53

CHAPTER 5  CONCLUSIONS ........................................... 54

REFERENCES ......................................................... 56
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>Absorbing Boundary Condition</td>
</tr>
<tr>
<td>FDTD</td>
<td>Finite-Difference Time-Domain</td>
</tr>
<tr>
<td>FEM</td>
<td>Finite-Element Method</td>
</tr>
<tr>
<td>GCM</td>
<td>Globally Convergent Method</td>
</tr>
<tr>
<td>GMRES</td>
<td>Generalized Minimal Residual</td>
</tr>
<tr>
<td>ICS</td>
<td>International Compumag Society</td>
</tr>
<tr>
<td>ILU</td>
<td>Incomplete LU</td>
</tr>
<tr>
<td>ILUT</td>
<td>Threshold-based Incomplete LU</td>
</tr>
<tr>
<td>LCM</td>
<td>Locally Convergent Method</td>
</tr>
<tr>
<td>MoM</td>
<td>Method of Moments</td>
</tr>
<tr>
<td>MVP</td>
<td>Matrix-Vector Products</td>
</tr>
<tr>
<td>NR</td>
<td>Newton-Raphson</td>
</tr>
<tr>
<td>PEC</td>
<td>Perfect Electric Conductor</td>
</tr>
<tr>
<td>PMC</td>
<td>Perfect Magnetic Conductor</td>
</tr>
<tr>
<td>PML</td>
<td>Perfectly Matched Layer</td>
</tr>
<tr>
<td>TDFEM</td>
<td>Time-Domain Finite-Element Method</td>
</tr>
<tr>
<td>TEAM</td>
<td>Testing Electromagnetic Analysis Methods</td>
</tr>
<tr>
<td>SLES</td>
<td>Simplified Linear Equation Solver</td>
</tr>
<tr>
<td>PETSC</td>
<td>Portable Extensible Toolkit for Scientific Computation</td>
</tr>
<tr>
<td>RCM</td>
<td>Reverse Cuthill-McKee</td>
</tr>
</tbody>
</table>
Since the second Industrial Revolution, the discovery of electricity and its wide applications dramatically expedited the development of human civilizations. However, because of the limitation of the energy and the resources on our earth, the energy shortage has become an increasingly severe problem and how to use and deliver energy efficiently is one of the commonest concerns. Thus, the design of electric machines which are both energy-saving and eco-friendly arouses great concern and is a hot topic recently. Robust and efficient simulation tools are needed to simulate a variety of electric machines with complex geometries and materials.

The current simulation of electric machine problems has the following challenges. First, the electric machines normally contain different complex materials and geometries, like air gap, rotor, stator, etc. Both the dimensions and material parameters of these components may have large variations which result in a very ill-conditioned system matrix even for linear cases. Second, the electric machines usually work at low frequencies which range from 0 Hz to 1 kHz. This may also lead to an ill-conditioned system matrix which makes the solution computationally expensive. Third, as for some applications where ferromagnetic materials are present, the B-H curves are nonlinear. This requires a special algorithm, like the Newton-Raphson method or the fixed-point method, to solve the system of nonlinear equations. The desired simulation software package should be able to handle all these features.

With the emergence of digital computers, the real-world continuous problems which were previously intractable could now be discretized into a finite number of subproblems that could be solved readily by computers. As the computer technology has been developed very quickly, the computer has been able to deal with a huge number of variables. Numerical methods started to become increasingly popular and be applied in various fields. Nowadays, the finite-element method (FEM) has been widely used in many areas. The early
development of FEM could be traced back to the 1950s when it was first applied to structural analysis in aircraft engineering and aerospace technology [1], [2]. It was further developed by Ray W. Clough in the 1960s in the area of civil engineering and the terminology “finite-element method” was first used [3]. It then became popular in various fields in engineering [4]. There are other computational methods, such as the finite-difference time-domain (FDTD) method and the method of moment (MoM). FDTD is based on the time-domain Maxwell’s equations. It discretizes the computational domain into rectangular grids and approximates the partial differential equations (PDE) by using the finite-difference method. The construct of time-stepping formula is normally explicit and the solution of this time step could be updated from the values of the previous time steps readily. However, the stair-case problem resulting from using rectangular grids in FDTD limits its modeling accuracy [5]. Compared to FDTD, FEM has better modeling capability and it is powerful enough to model complicated problems with complex media and geometry because it uses an unstructured grid. As for MoM, the most important issue is to obtain Green’s function [6]. It may be readily obtained for open-region problems, like radiation or scattering. However, as for more complicated ones, where complex inhomogeneous materials are involved, it requires a significant amount of work to calculate Green’s function.

The time-domain finite-element method has been proved as a versatile numerical method, because of its modeling flexibility and broadband responses. By applying the Newmark-beta scheme, it could generate an unconditionally stable algorithm and possesses second-order accuracy. This shows its prominent merits, compared to other schemes, like the backward, forward or central difference schemes.

The objective of this research is to develop a robust and efficient numerical method to analyze three-dimensional electrical machine problems. The method should have modeling flexibility and be capable of handling the problems where nonlinear materials might be present. Moreover, in most machine applications, the machines are operating at a low frequency, so the method should be able to deal with this kind of problems where the frequency of the excitation may range from 0 Hz to 1 kHz. Among all the numerical methods, the time-domain finite-element method (TDFEM) has been proven as a versatile numerical method, because of its power of handling complex geometries and materials and convenience for modeling nonlinearity. In this
thesis, starting from Maxwell’s equations, we firstly derive the A-V formulation for the TDFEM. Then, the Newton-Raphson method is introduced and employed to solve the system of nonlinear equations. Because of the low-frequency excitation, tree-cotree splitting (TCS) technique is applied to make the singular system of matrix nonsingular so that unique solutions can be obtained. Also, several numerical examples are shown to verify and validate the proposed formulations and algorithms. Finally the conclusions are drawn.
CHAPTER 2

FINITE-ELEMENT FORMULATION AND IMPLEMENTATION

2.1 Introduction

This chapter mainly describes the time-domain finite-element method (TD-FEM) formulation and its implementation for the electrical machine problem. Starting from Maxwell’s equations, the A formulation for electrical machine problems is derived first. Then, two FEM discretization methods are introduced. The characteristics of nodal and edge elements are studied and compared. This is followed by the introduction of the temporal discretization where the Newmark-beta method is applied to generate an unconditionally stable algorithm. The formulation and implementation described in this chapter will be used as a basic version of TDFEM for the simulation of electrical machine problems in the following chapters.

2.2 Basic TDFEM Formulation for Electrical Machine Problems

Maxwell’s equations in the time-domain can be written as

\[ \nabla \times \mathbf{E} = -\frac{\partial}{\partial t}\mathbf{B} \]  \hspace{1cm} (2.1)

\[ \nabla \times \mathbf{H} = \frac{\partial}{\partial t}\mathbf{D} + \mathbf{J}_e + \mathbf{J}_{im} \]  \hspace{1cm} (2.2)
where \( J_c \) and \( J_{im} \) are the conduction current and the impressed current, respectively. The constitutive relations are given by

\[
B = \mu_0 \mu_r H \tag{2.3}
\]
\[
D = \epsilon_0 \epsilon_r E \tag{2.4}
\]
\[
J = \sigma E \tag{2.5}
\]

By noting that

\[
B = \nabla \times A \tag{2.6}
\]

and substituting the above expression into (2.1) and (2.2), we have

\[
\nabla \times (E + \frac{\partial}{\partial t} A) = 0 \tag{2.7}
\]
\[
\nabla \times \left( \frac{1}{\mu_r} \nabla \times A \right) = \mu_0 \epsilon_0 \epsilon_r \frac{\partial}{\partial t} E + \mu_0 \sigma E + \mu_0 J_{im} \tag{2.8}
\]

Since for our simulations, we only care about the magnetic flux density \( B \) instead of \( A \) itself, the curl of \( B \) is uniquely defined, if we assume

\[
E = -\frac{\partial}{\partial t} A \tag{2.9}
\]

Substituting (2.6) and (2.9) into (2.8), we have

\[
\nabla \times \left( \frac{1}{\mu_r} \nabla \times A \right) + \frac{\epsilon_r}{\epsilon_0} \frac{\partial^2}{\partial t^2} A + \frac{z_0}{\epsilon_0} \sigma \frac{\partial}{\partial t} A = \mu_0 J_{im} \tag{2.10}
\]

where \( z_0 \) is the intrinsic impedance in the free space and is equal to \( \sqrt{\frac{\mu_0}{\epsilon_0}} \).

Expanding \( A \) in terms of vector basis functions yields

\[
A = \sum_{j=1}^{N} \phi_j N_j \tag{2.11}
\]

where \( \phi_j \) are the expansion coefficients that need to be determined. Testing both sides of (2.10) with \( N_i \) and integrating the whole equation over the
entire computational domain $V$ gives rise to

$$\iiint_V \mathbf{N}_i \cdot \nabla \times \frac{1}{\mu_r} \nabla \times \sum_{j=1}^{N} \phi_j \mathbf{N}_j \, dV$$

(2.12)

$$+ \iiint_V \frac{\varepsilon_r}{c_0} \frac{\partial^2}{\partial t^2} \mathbf{N}_i \cdot \sum_{j=1}^{N} \phi_j \mathbf{N}_j \, dV$$

(2.13)

$$+ \iiint_V \frac{\varepsilon_0}{c_0} \frac{\partial}{\partial t} \phi_j \mathbf{N}_i \cdot \sum_{j=1}^{N} \phi_j \mathbf{N}_j \, dV$$

(2.14)

$$= \iiint_V \mu_0 \mathbf{N}_i \cdot \mathbf{J}_{im} \, dV$$

(2.15)

By using Gauss’s theorem

$$\iiint_V \nabla \cdot \mathbf{A} \, dV = \oiint_S \mathbf{A} \cdot d\mathbf{S}$$

(2.16)

and some mathematical manipulations, we obtain the following equation

$$\sum_{j=1}^{N} \phi_j \iiint_V \frac{1}{\mu_r} (\nabla \times \mathbf{N}_i) \cdot (\nabla \times \mathbf{N}_j) \, dV$$

$$+ \frac{1}{\varepsilon_0} \sum_{j=1}^{N} \frac{\partial^2}{\partial t^2} \phi_j \iiint_V \varepsilon_r \mathbf{N}_i \cdot \mathbf{N}_j \, dV$$

$$+ \frac{\varepsilon_0}{c_0} \sum_{j=1}^{N} \frac{\partial}{\partial t} \phi_j \iiint_V \sigma \mathbf{N}_i \cdot \mathbf{N}_j$$

(2.17)

$$- \sum_{j=1}^{N} \oiint_S \phi_j \frac{1}{\mu_r} (\nabla \times \mathbf{N}_j) \cdot (\hat{n} \times \mathbf{N}_i)$$

$$= \mu_0 \iiint_V \mathbf{N}_i \cdot \mathbf{J}_{im} \, dV$$

On the perfect electric conductor (PEC) and the perfect magnetic conductor (PMC), the boundary conditions are given by

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

(2.18)

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

(2.19)

respectively. Thus, the surface integral term in (2.17) vanishes on either PEC.
or PMC. Equation (2.17) can be rewritten in a compact matrix form as

\[
[S] \{\phi\} + \frac{1}{c_0^2} [M] \frac{\partial^2}{\partial t^2} \{\phi\} + \frac{1}{c_0} ([G] + [B]) \frac{\partial}{\partial t} \{\phi\} = \{b\} \tag{2.20}
\]

where the matrix elements are given by

\[
S_{ij} = \iiint_V \nu_r (\nabla \times N_i) \cdot (\nabla \times N_j) \, dV \tag{2.21}
\]

\[
M_{ij} = \iiint_V \epsilon_r N_i \cdot N_j \, dV \tag{2.22}
\]

\[
G_{ij} = z_0 \iint_V \sigma N_i \cdot N_j \, dV \tag{2.23}
\]

\[
B_{ij} = - \iint_S \frac{1}{\mu_r} (\nabla \times N_j) \cdot (\hat{n} \times N_i) \tag{2.24}
\]

\[
b_i = \mu_0 \iiint_V N_i \cdot J_{im} \, dV \tag{2.25}
\]

where \(z_r = \sqrt{\frac{\mu_r}{\epsilon_r}}\) and \(\nu_r\) is the relative reluctivity of the materials.

### 2.3 Finite-Element Discretization: Nodal and Edge Elements

After the finite-element discretization, a set of basis functions are employed to obtain the approximate solution within each element. There are various methods to define the basis functions. Taking a tetrahedral element as an example, the solution within one element can be expressed as the linear combination of the basis functions \(N_j^e(x, y, z)\). The basis function is a scalar and defined on the four nodes of the tetrahedral element, so it is referred to as a nodal element [7].

\[
\phi_e = \sum_{j=1}^{4} N_j^e(x, y, z) \phi_j^e \tag{2.26}
\]

The first-order basis functions’ expressions are given by

\[
N_j^e(x, y, z) = \frac{V_j}{V_e}, \quad j = 1, 2, 3, 4 \tag{2.27}
\]
where

$$V_1 = \frac{1}{6} \begin{vmatrix} 1 & x & y & z \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}$$  \hspace{1cm} (2.28)$$

$$V_2 = \frac{1}{6} \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x & y & z \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}$$  \hspace{1cm} (2.29)$$

$$V_3 = \frac{1}{6} \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x & y & z \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}$$  \hspace{1cm} (2.30)$$

$$V_4 = \frac{1}{6} \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x & y & z \end{vmatrix}$$  \hspace{1cm} (2.31)$$

$$V^e = \frac{1}{6} \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix}$$  \hspace{1cm} (2.32)$$

By scrutinizing (2.28)-(2.32) we could find that $V^e$ is actually the volume of the tetrahedral element. Moreover, $V_j$ is the volume of the tetrahedral element specified by the node $(x,y,z)$ and the other three nodes of $V^e$ except for node $j$. Apparently, if node $(x,y,z)$ is within the tetrahedral element, we
have

\[ V^e = \sum_{j=1}^{4} V_j \]  

(2.33)

and then

\[ \sum_{j=1}^{4} N_j^e = 1 \]  

(2.34)

which is consistent with the definition of the basis functions. The magnitude distributions of the nodal basis functions are shown in Figure 2.1(a) - 2.1(d) (all figures are at the end of chapters).

Although the implementation of the above-mentioned nodal elements are pretty clear and straightforward, it suffers from some severe problems that undermine its applications [7]. First, it is not very convenient for the nodal elements to enforce the boundary conditions at the materials interfaces. It requires specifying the relations among each vector field component at the interface, which would be troublesome to implement. Second, the solutions obtained from the node-based FEM may contain spurious components due to the lack of the divergence condition enforcement. Third, it is difficult to deal with the edges or corners on the conducting and dielectric objects because of the singularity in the fields.

The alleviation of the problems was achieved after the edge element was proposed by J. C. Nedelec in 1986 [8]. The edge basis functions are defined on the six edges of the tetrahedral element. Equation (2.35) shows one of the edge basis functions defined on the tetrahedron edge with two endpoints, node 1 and 2. It can be seen that the edge basis function is constructed based on the information of nodal basis functions.

\[ N_{12} = N_1^e \nabla N_2^e - N_2^e \nabla N_1^e \]  

(2.35)

It can be easily verified that

\[ \nabla \cdot N_{12} = 0 \]  

(2.36)

The vector field distribution of \( N_{12} \) is shown in Figure 2.2 with node numberings. The highlighted edge with endpoints of node 1 and node 2 is the
edge where $\mathbf{N}_{12}$ is defined on. From the vector fields, we could see that $\mathbf{N}_{12}$ is normal to the surfaces (2,3,4) and (1,3,4). It only has tangential field components on the surfaces that contain edge (1,2). Moreover, $\mathbf{N}_{12}$ has trivial components on the edge (3,4) which is opposite to it. Finally, it can be proved that along edge (1,2), $\mathbf{N}_{12}$ has a constant tangential component which is equal to $1/L_{12}^t$, where $L_{12}^t$ is the length of edge (1,2). The edge element guarantees the tangential continuity condition between the adjacent elements. In the following analysis, only the first-order linear edge element is applied in the simulation.

2.4 Temporal Discretization in TDFEM

The equation in (2.20) has been discretized spatially, which is necessary since the computer could only deal with a finite amount of discrete information. Since the expression is still continuous in time, we need to employ some schemes to discretize (2.20) in the time domain. There are several time-domain discretization schemes and most of them can be derived based on the Taylor series expansion. Considering a temporal function $\phi(t)$, the Taylor series expansion can be expressed as [7]

$$
\phi(t \pm \Delta t) = \phi(t) \pm \frac{d\phi}{dt} \Delta t + \frac{d^2\phi}{dt^2} \frac{\Delta t^2}{2!} + O(\Delta t^3)
$$

By some mathematical manipulations and introducing the notation noting $\phi(n\Delta t) = \phi^n$, we could obtain different approximations to the time derivatives $d\phi/dt$ and $d^2\phi/dt^2$. Taking the backward difference scheme as an example, the approximations are given by

$$
\frac{d\phi}{dt} = \frac{\phi^n - \phi^{n-1}}{\Delta t}
$$

$$
\frac{d^2\phi}{dt^2} = \frac{\phi^n - 2\phi^{n-1} + \phi^{n-2}}{\Delta t^2}
$$

By substituting these temporal approximations into (2.20), we have the following different time-marching schemes, resulting from the forward, back-
ward and central difference schemes.

\[
\frac{1}{\Delta t^2} [M] \{\phi\}^{n+1} = \left\{ \frac{2}{\Delta t^2} [M] - \frac{1}{\Delta t} ([G] + [B]) \right\} \{\phi\}^n \\
- \left\{ \frac{1}{\Delta t} ([G] + [B]) + [S] \right\} \{\phi\}^{n-1}
\]  

(2.40)

\[
\left\{ \frac{1}{\Delta t^2} [M] + \frac{1}{\Delta t} ([G] + [B]) \right\} \{\phi\}^{n+1} \\
= \left\{ \frac{2}{\Delta t^2} [M] + \frac{1}{\Delta t} ([G] + [B]) \right\} \{\phi\}^n \\
- \frac{1}{\Delta t^2} [M] \{\phi\}^{n-1}
\]  

(2.41)

\[
\left\{ \frac{1}{\Delta t^2} [M] + \frac{1}{2 \Delta t} ([G] + [B]) \right\} \{\phi\}^{n+1} \\
= \left\{ \frac{2}{\Delta t^2} [M] - [S] \right\} \{\phi\}^n \\
- \left\{ \frac{1}{\Delta t^2} [M] - \frac{1}{2 \Delta t} ([G] + [B]) \right\} \{\phi\}^{n-1} \\
- \left\{ \frac{1}{\Delta t^2} [M] - \frac{1}{\Delta t} ([G] + [B]) + [S] \right\} \{\phi\}^{n-1}
\]  

(2.42)

Not all the three time-marching schemes can be used practically. It has been shown in [9] that the forward difference scheme is numerically unstable. It cannot be used in practical simulations. The time-marching scheme derived from backward difference is an implicit scheme and shown to be unconditionally stable, which means the scheme is always stable no matter which \(\Delta t\) is chosen. By contrast, the central difference scheme is explicit since the left-hand side system matrix does not involve the \([S]\). The scheme is conditionally stable which requires \(\Delta t\) to be small enough. Compared to the backward scheme which has only first-order accuracy, the central difference scheme is second-order accurate. A more useful temporal discretization scheme is the Newmark method [10]. By applying the Newmark-beta method for the temporal discretization, we obtain the following time-marching for-
\[ \{ [M] + \frac{1}{2} c_0 \Delta t ([G] + [B]) + c_0^2 \Delta t^2 \beta [S] \} \{ \phi \}^{n+1} \]
\[ = 2 \left\{ [M] - c_0^2 \Delta t^2 \left( \frac{1}{2} - \beta \right) [S] \right\} \{ \phi \}^n + \left\{ - [M] + \frac{1}{2} c_0 \Delta t ([G] + [B]) - c_0^2 \Delta t^2 \beta [S] \right\} \{ \phi \}^{n-1} + c_0^2 \Delta t^2 \left\{ \beta \{ b \}^{n+1} + (1 - 2 \beta) \{ b \}^n + \beta \{ b \}^{n-1} \right\} \]

It has been proved that if \( \beta \geq \frac{1}{4} \), the time-marching scheme is always unconditionally stable. Usually, we choose \( \beta = \frac{1}{4} \) and (2.43) becomes
\[ \{ [M] + \frac{1}{2} c_0 \Delta t ([G] + [B]) + \frac{1}{4} c_0^2 \Delta t^2 \beta [S] \} \{ \phi \}^{n+1} \]
\[ = 2 \left\{ [M] - \frac{1}{4} c_0^2 \Delta t^2 [S] \right\} \{ \phi \}^n + \left\{ - [M] + \frac{1}{2} c_0 \Delta t ([G] + [B]) - \frac{1}{4} c_0^2 \Delta t^2 \beta [S] \right\} \{ \phi \}^{n-1} + \frac{1}{4} c_0^2 \Delta t^2 \left\{ \{ b \}^{n+1} + 2 \{ b \}^n + \{ b \}^{n-1} \right\} \]

In each time step, we need to solve the following system of equations
\[ [K] \{ \phi \} = \{ b \} \] (2.45)
where
\[ [K] = [M] + \frac{1}{2} ([G] + [B]) + \frac{1}{4} [S] \] (2.46)
and
\[ \{ b \} = \left\{ [M] - \frac{1}{4} c_0^2 \Delta t^2 [S] \right\} \{ \phi \}^n + \left\{ - [M] + \frac{1}{2} ([G] + [B]) - \frac{1}{2} c_0^2 \Delta t ([G] + [B]) \right\} \{ \phi \}^{n-1} + \frac{1}{4} c_0^2 \Delta t^2 \left\{ \{ b \}^{n+1} + 2 \{ b \}^n + \{ b \}^{n-1} \right\} \]

It can be seen from (2.43) that the solution vectors \( \{ \phi \}^{n+1} \) in the current time step can be obtained based on the information of the previous two time steps.
steps, $\{\phi\}^{n-1}$ and $\{\phi\}^n$. The updating process continues in the time period of interest. This formulation of the time-domain finite-element method provides the most basic version of TDFEM.
2.5 Figures

Figure 2.1: Nodal basis functions: (a) $N_1(x, y, z)$, (b) $N_2(x, y, z)$, (c) $N_3(x, y, z)$ and (d) $N_4(x, y, z)$. 
Figure 2.2: Vector field distribution of edge basis function.
CHAPTER 3

MODELING OF LINEAR MACHINE PROBLEMS

3.1 Introduction

In practice, there are several challenges in solving linear machine problems. For instance, a linear system of equations needs to be solved after discretizations in both space and time. It can be solved either iteratively or directly. In some linear machine problems, the variations of the dimensions of objects or material parameters are very large across the computational domain. This highly nonuniform feature in either meshes or materials might lead to an ill-conditioned system matrix which is time consuming for an iterative solver to solve. Another challenge is caused by the low operation frequency of the machines, which leads to a similar ill-conditioned matrix which is nearly singular. For both cases, special treatments are required to resolve the problems so that solution time is reduced to an acceptable amount. This chapter firstly describes the application of the TDFEM to simulation of electromagnetic fields excited by a racetrack coil. This is the most basic model and can be used to verify the fundamental formulation of TDFEM by checking the convergence of the solution. Then, the effects of the ILU preconditioner and Cuthill-McKee reordering on the convergence of a linear solver are analyzed. Also, the tree-cotree splitting algorithm is introduced to solve low-frequency problems. Finally, several linear examples are shown to demonstrate the validity of the current formulation in solving linear machine problems.

3.2 Modeling Racetrack Coil Excitation

The first problem simulated is a coil in free space as shown in Figure 3.1(a). The whole coil is boxed by an absorbing boundary, which is a cubic with the
side length of 1 m. In most machine problems, the coil usually resembles a racetrack shown in Figure 3.1(b). The magnetic flux densities in the line with two end point coordinates of (0.7, 0.7, 0.0) and (0.7, 0.7, 1.0) within the computational domain are calculated. The simulated results obtained from the TDFEM are compared to the converged results calculated by FEM in the frequency domain. The relative root mean square (RMS) error is given by

$$\sqrt{\frac{\sum_{i=1}^{N} (x_{1,i} - x_{2,i})^2}{N \sum_{i=1}^{N} (x_{2,i})^2}}$$

(3.1)

where \(N\) is the number of the sampling points and \(x_{1,i}\) and \(x_{2,i}\) are the values of the simulated and referenced results for the \(i\)th sampling point, respectively. By varying the mesh size, the relative RMS error is shown in Figure 3.2, which demonstrates the convergence of the simulated results.

3.3 Convergence of Linear System with RCM Reordering and ILU Preconditioner

In the finite element method, we need to solve a linear system of equations after assembling the system matrix. Generally, there are two ways to solve the system of equations. One of them is using a direct solver where LU factorization can be applied to factorize the system matrix as the product of one upper triangular matrix and one lower triangular one. Then, the solution of the resulting system of equations can be obtained by either forward or backward substitutions. The other way to solve the linear system of equations is to apply an iterative solver, where we try to obtain a successive approximation to the solution by adding a correction term to it in each iterative step. The efficiency of the solution to the linear system benefits from the high sparsity of the matrix to a great extent. As for a problem with a size that is not extremely large, a direct solver could be used to solve it efficiently, since the system matrix is prefactorized before the time-marching process begins and there is no need to repeat this work once it is done. The solution can always be obtained by using the direct solver so long as the system matrix
is nonsingular. Even if the system matrix is singular, it will still provide a solution and the user needs to check its accuracy. However, as the size of the problem becomes extremely large, the application of a direct solver to solve the system of equation would be prohibitively expensive. This is because the LU factorization would contribute to most of the computational expenses that are proportional to the cube of the total number of unknowns. Under such a circumstance, an iterative solver seems to be a wiser choice, since its major work lies in the repeated evaluations of the matrix-vector products (MVPs) which are highly parallelizable and very fast for sparse matrices. The efficiency of an iterative solver depends on many factors, such as the condition number of the system matrix and the preconditioner used. In this chapter, the effect of the material parameters on the condition of the matrix is shown and analyzed. In certain problems where the materials are highly inhomogeneous, the system matrix would be so ill-conditioned that it requires a significant number of iterative steps to get converged results. Then, the reverse Cuthill-McKee reordering technique and ILUT preconditioner are employed to resolve this issue.

3.3.1 Iterative Methods and Krylov Subspace Methods

The basic idea of an iterative method in solving a linear system in (3.2) is trying to find a nearby system matrix, say \( K \), of the system matrix \( A \) so that the modified system is much easier to solve [11].

\[
Ax = b
\]  

(3.2)

With a proper initial guess \( x_0 \), the residual \( r_0 \) of the equation can be calculated as (3.3).

\[
r_0 = Ax_0 - b
\]  

(3.3)

The correction term \( dx \) which is supposed to satisfy (3.4) can be evaluated as (3.5), where we replace the original system matrix \( A \) with the nearby matrix
Then, $x_{i+1}$ can be derived as (3.6)

$$x_{i+1} = x_i + \delta x_i = x_i + \tilde{b} - \tilde{A}\delta x_0$$  \hspace{1cm} (3.6)

where

$$\tilde{A} = K^{-1}A \hspace{1cm} \text{(3.7)}$$

$$\tilde{b} = K^{-1}b \hspace{1cm} \text{(3.8)}$$

It is important to note that the matrix $K$ is an approximated system matrix, which is much simpler compared to $A$. Thus, the solution to (3.5) would be more efficient than (3.4). Since the direct evaluation of the inverse of $K$ is rather expensive, it is more usual to evaluate the product of $K^{-1}$ and $b$ by solving $\tilde{b}$ from (3.8).

If we let $K = I$, (3.6) reduces to

$$x_{i+1} = x_i + \delta x_i = x_i + b - A\delta x_i = x_i + r_i$$  \hspace{1cm} (3.9)

which is the well-known Richardson iteration. Equation (3.9) can be further written as

$$x_{i+1} = x_i + r_i = r_0 + r_1 + \cdots + r_i = \sum_{j=1}^{i} (I - A)^j r_0$$  \hspace{1cm} (3.10)

where we assume an initial $x_0$ to be a zero vector. It can be seen from (3.9) that, $x_{i+1}$ belongs to the space spanned by a group of vectors: $r_0, Ar_0, A^2r_0, \ldots$
\[ x_{i+1} \in \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^ir_0\} \triangleq K_{i+1} \] 

(K) in (3.9) is called the \( i+1 \) dimensional Krylov subspace that is generated by \( A \) and \( r_0 \). The so-called Krylov subspace methods are a group of methods that attempt to obtain the approximated \( x_i \in K_{i+1} \) on the ground of the Krylov subspace; they are categorized into different classes based on different approaches of searching for \( x_i \) [11]. For instance, one of the most popular Krylov subspace methods is the generalized minimal residual method (GMRES). In GMRES, \( x_i \) is obtained by minimizing the Euclidean norm of the residual of (3.2) over the \( K_{i+1} \), namely [12]

\[
\text{minimize}_{x \in K_{i+1}} \| b - Ax \|_2
\]

The above leads to a least square problem which usually can solved by the QR factorization.

3.3.2 Preconditioning and ILU Preconditioner

The aforementioned Krylov subspace methods alone do not guarantee that the solution process can be completed within a reasonable amount of time and computer storage. The iterative process could be very inefficient if the problem involves high inhomogeneity in geometry or materials so that the resulting assembled matrix has a very large condition number. The condition number of a matrix is defined as the ratio of the maximum singular value \( \sigma_{\text{max}} \) to its minimum singular value \( \sigma_{\text{min}} \) as shown in (3.13).

\[
\text{cond}(A) = \frac{\sigma_{\text{max}}}{\sigma_{\text{min}}}
\]

A preconditioner is a matrix \( P \) which is applied to ameliorate the condition of the system matrix. Generally, it is close to the original system matrix \( A \) but is easier to solve. There are different implementations of preconditioners, such as left-preconditioning, right-preconditioning and two-sided preconditioning. Equation (3.14) shows a typical linear system of equations with

\[
\therefore, A^ir_0, \text{i.e.}
\]
left-preconditioning.

\[ P^{-1}Ax = P^{-1}b \]  

(3.14)

Because \( P \) is set to be close to \( A \), the preconditioned system has a smaller condition number than the original one. Instead of solving the original system, the preconditioned system is solved. Ideally, if \( P = A^{-1} \), all the Krylov subspace methods would converge in one single step. In practice, however, is nearly impossible to evaluate inverse of \( P \), which would turn out to be computational expensive. Thus, there is a tradeoff between the generation of the preconditioner and the solution to the preconditioned system.

Among all the preconditioners, the incomplete LU (ILU) preconditioner has been proved as an efficient preconditioner for solving a sparse linear system. The ILU decomposition generates one lower and one upper triangular matrix represented by \( L \) and \( U \), respectively. The decomposition process is very similar to the LU factorization and the only difference is that the product of \( L \) and \( U \) may not exactly be \( A \) here. The difference between \( LU \) and \( A \) depends on the degree of fill-in during the ILU decomposition. Basically, there are two sorts of ILU algorithms. One is threshold-based and the other is level-based. Here, we only discuss the former one, which is also referred to as ILUT.

In ILUT, the threshold \( \delta_{tol} \), which is also called drop tolerance, is given to control the degree of fill-in. More specifically, if the magnitude of the fill-in entries satisfies the following condition (3.15), the entries can be ignored.

\[ \| A_{ij} \| < \delta_{tol} \| A_j \| \]  

(3.15)

where \( \| A_j \| \) is the norm of the \( j \)th column vector of matrix \( A \). If \( \delta_{tol} = 0 \), the ILUT reduces to the conventional LU decomposition where no term is discarded. Apparently, the smaller the drop tolerance \( \delta_{tol} \) is, the closer ILUT will be to the LU decomposition. The decomposition of ILUT is given in Algorithm 1 [13]. \( S \) is the index set of zero entries in the system matrix (3.16).

\[ S = \{(i,j)|a_{ij} \neq 0\} \]  

(3.16)
Algorithm 1: ILU Decomposition

\begin{algorithm}
\begin{algorithmic}
\State \textbf{for} $k = 1, 2, \cdots, n$ \textbf{do}
\State \hspace{1em} \textbf{for} $i = k + 1, k + 2, \cdots, n$ \textbf{do}
\State \hspace{2em} \textbf{if} $(i, k) \in S$ \textbf{then}
\State \hspace{3em} $a_{ik} = a_{ik} / a_{kk}$;
\State \hspace{2em} \textbf{for} $j = k + 1, k + 2, \cdots, n$ \textbf{do}
\State \hspace{3em} \textbf{if} $(i, j) \in S$ \textbf{and} $(k, j) \in S$ \textbf{then}
\State \hspace{4em} $a_{ij} = a_{ij} - a_{kj} / a_{ik}$;
\end{algorithmic}
\end{algorithm}

3.3.3 Reverse Cuthill-McKee Reordering

The system matrix generated by the finite-element discretization is highly sparse. In order to minimize the profile storage of the matrix, it is necessary to make full use of the advantages of the banded matrix. There is more than one way to define the bandwidth of a matrix. One definition of the bandwidth of a matrix $A$ proposed by Cuthill and McKee is given by (3.17) [14].

$$\text{bandwidth}(A) = \max \{|i - j| : A_{ij} \neq 0\} \quad (3.17)$$

The bandwidth of a matrix greatly depends on the connectivity graph and the numbering of the degrees of freedom (DOF). Taking the connectivity graph in Figure 3.3(a) as an example, the nodes in the graph can represent the unknowns of any type of elements, such as nodal or edge elements. It can be easily calculated that the bandwidth of the matrix is 4. Its corresponding system matrix is given in Figure 3.3(b). The reordering technique provides a strategy to renumber the unknowns in a way such that the bandwidth of the matrix can be minimized in some sense. One of the most widely used reordering techniques is proposed by Cuthill and McKee [14]. The steps are described as follows. First, we choose a node as our starting node. It is tricky to pick a starting node, since different choices may lead to different final bandwidths of the matrix and the results could vary significantly. The rule of thumb as stated in [14] is choosing the node which has the minimum connections to other nodes in the connectivity graph. For the convenience of
explanation, we assign a level to each node. We assume the starting node has a level of 1. Then, the nodes connected directly to the starting nodes have a level of 2. These nodes are numbered subsequently. In fact, nodes in level 2 are all the neighboring nodes of node 1. We use the notation Neighbor[i][j] to denote the jth neighboring node of node i. All the nodes connected to the nodes in level 2 excluding all the nodes in previous levels, which is level 1, are of level 3. This process is repeated in the same manner until all the nodes are processed. The algorithm for Cuthill-McKee reordering is given in Algorithm 2.

Algorithm 2: Cuthill-KcMee Reordering

Choose a starting node and number it as node 1

Number of nodes \( N = 1 \)

Add node 1 to node set \( P \)

while \( P \neq \emptyset \) do

   Extract the first node \( i \) from \( P \)

   for \( j = 1, 2, \ldots \) do

      if Neighbor[i][j] \( \notin P \) then

         \( N = N + 1 \)

         Add node Neighbor[i][j] to \( P \) and number it as \( N \)

By applying Cuthill-McKee reordering, the reordered connectivity graph is shown in Figure 3.3(b). Figure 3.4(b) shows its system matrix. It can be seen from Figure 3.4(b) that nonzero entries cluster more around the diagonal of the matrix and the bandwidth is 3 after reordering. It was proven that the reverse Cuthill-McKee (RCM) reordering algorithm, which simply reverses the numbering generated by the Cuthill-McKee algorithm, usually has improvement in the storage and the workload of arithmetic operations in factorization [15]. The connectivity graph generated by RCM reordering is shown in Figure 3.5. For this case, because of the symmetry of the problem, the corresponding matrix after RCM reordering is exactly the same as Figure 3.4(b). Figures 3.6(a) and 3.6(b) show the nonzeros entries of the system matrix without and with RCM reordering.
3.3.4 Convergence Discussion

The condition of the system matrix depends on many factors. We investigate the effects of the parameters on the condition of the system matrix by testing a benchmark problem. The problem simulated is a modified version of benchmark problem 10 of the Testing Electromagnetic Analysis Methods (TEAM) workshop, where three steel plates are around a coil in the free space. The side view and top view of the model are shown in Figures 3.7(a) and 3.7(b), respectively. For this case, all the materials are linear. The three steel plates have a relative permeability as high as 2000. Their conductivity is $7.505 \times 10^6$ S/m. Because of the symmetry of the problem, only one-eighth of the problem is simulated. The linear solver, the simplified linear equation solver (SLES) in the package Portable Extensible Toolkit for Scientific Computation (PETSC), is used to solve the linear system of equations. We employ the GMRES method and ILUT preconditioner. By changing the parameters of the problem, we will see how these parameters influence the convergence of the linear solver.

First, we explore the effects of the material parameters on the convergence of the iterative solver. The model and mesh of the modified version of problem 10 are generated by the software Cubit13.0 and are shown in Figures 3.8(a) and 3.8(b), respectively. The time step size $\Delta t$ is chosen as 1500 $\mu$s. The total number of degrees of freedom is 57327. The drop tolerance is set to $1 \times 10^{-6}$. Figure 3.9 shows the convergence history on the linear solver for different relative permeabilities of the steel plates. The permeability ranges from 1, which is the same as that of the surrounding air region, to 2000, which is possible if the steel plates are composed of nonlinear materials. It can be seen from Figure 3.9 that, as the relative permeability of the steel plates increases, it takes more iterative steps for the linear solver to converge. This is because the condition of the system matrix becomes worse as the materials in the computational domain become highly inhomogeneous, which leads to abnormally large and tiny terms in the system matrix. This results in a large matrix condition number, which is defined in (3.13), making the system of equations very time consuming to solve. In the worst case when the relative permeability is as high as 2000, as shown in the Figure 3.9, the linear solver may fail to converge even with 10000 iterative steps.

Then, we investigate the relation between the time step size and the con-
dition of the system matrix. Figure 3.9 shows the convergence history of the iterative solver under the circumstances of different relative permeabilities. The time step size $\Delta t$ is 1500 $\mu$s. The results shown in Figure 3.10 are obtained based on the same parameters, like drop tolerance and mesh, as those in Figure 3.9, except that the time step sizes for these cases are all chosen to be 150 $\mu$s which is the one-tenth of the previous one. We can see from Figure 3.10 that the diminished time step size generally accelerates the convergence of the iterative solver, especially for the cases of $\mu_r = 500$ and 2000, where the amelioration in convergence is very prominent. This is because the curl-curl matrix, $[S]$ in (2.44), is singular [16], [17]. The smaller the $\Delta t$ is, the less the contribution of the singular matrix to the system matrix would be. However, a decrease in the time step size, though it improves the convergence to some extent, would result in many more time steps needed in the time-domain simulation, provided the total simulation time is fixed. For this case, the total number of time steps would be ten times that of the original one. As a consequence, the total amount of time in the simulation may not be reduced, or may even increase, after decreasing the time step size.

Figure 3.11 shows the comparisons of the convergence history of the iterative solver with and without RCM reordering. Two cases of different permeabilities are shown here. The drop tolerance for the ILUT preconditioner is $1 \times 10^{-6}$. The maximum number of nonzeros allowed in a row is set to 80. It can be seen from Figure 3.11 that the RCM reordering technique has a significant effect on the convergence of the iterative solver for a given drop tolerance and maximum number of nonzeros allowed in a row. These better convergence results are attributed to the characteristics of the banded matrix where there are many fewer potential fill-ins during the generation of the ILU preconditioner by using the Gaussian elimination. Therefore, for a given fill-in degree, the banded matrix resulting from RCM reordering would have a much better ILU preconditioner than the matrix without RCM reordering. This contributes to the better convergence of the preconditioned matrix. Moreover, even if a better preconditioner could be obtained by allowing more fill-ins during the ILU decomposition, the increased number of fill-ins will require much more storage and operations in the matrix-vector products, making the iterative solution expensive both in memory and time.

To better understand this fact, the detailed simulation time for each part
is recorded and compared under different situations. Table 3.1 shows that, without RCM reordering, as the maximum number of nonzero entries allowed in a row increases from 80 to 500, the number of iterations needed to obtain converged results is reduced from 4770 to 18, which indicates that a better preconditioner is obtained. Nevertheless, the time consumed to generate the ILU preconditioner soars from 502.2 to 4048.4 seconds due to more arithmetic operations in the Gaussian elimination. Moreover, even if there are fewer iterations in case 5, where no RCM reordering is applied, the iteration time in case 1 is less as a result of RCM reordering. This demonstrates that without RCM reordering, more nonzero entries would be introduced so that more operations and time are needed to perform one iteration. Moreover, the unordered matrix requires as much as 25 times more time to generate the ILU preconditioner than the reordered matrix. Since RCM reordering is very cheap to implement, there is much less total simulation time in case 1 than case 5. Therefore, the implementation of the RCM reordering technique could obviously minimize the bandwidth of the matrix and then improve the convergence of the system matrix when the ILUT preconditioner is applied.

3.4 Tree-Cotree Splitting Algorithm for Low-frequency Problems

Most problems in the electric machine area usually operate at a low frequency band which ranges from 0 Hz to 1 kHz, which commonly requires a larger time step size $\Delta t$ in the time-domain simulation. However, it is well-known that, for the edge-based finite-element method, the discretization of the differential curl-curl operator results in a singular matrix. The nullity of the rank-deficient matrix equals the number of the tree edges in the finite-element mesh [16], [18]. As the $\Delta t$ becomes larger, the stiffness matrix $[S]$ plays a more important role in the construction of the system matrix, making it more ill-conditioned. This leads to a significant increase in the number of iterations, when an iterative solver is applied. One remedy is to eliminate the singularity of the matrix by removing the tree edges while introducing the nodal unknowns at the same time. The procedure is the so-called tree-cotree splitting (TCS). To introduce the algorithm, we first define some variables which will be used subsequently:
• int N_{\text{node}}: Total number of nodes in the mesh.
• int N_{\text{tree}}: Total number of tree edges in the mesh.
• int \text{RootNode}[i]: A 1D array which stores the indexes for all the nodes residing on PEC.
• int \text{TreeNode}[i]: A 1D array which stores the indexes for all the nodes that have been connected by the tree edges.
• int \text{NeighborNode}[i][j]: A 2D array with N_{\text{node}} rows by where ith row stores the indexes of all the nodes adjacent to node i.
• int \text{Tree}[i][j]: A N_{\text{tree}} by 2 array where ith row stores the indexes of the two nodes for the ith tree edge.

Then, the TCS algorithm can be described in Algorithm 3.

**Algorithm 3: Tree-Cotree Splitting**

\begin{verbatim}
Algorithm 3: Tree-Cotree Splitting
for \text{ith node in RootNode}[i] do
    Get the ith node index: \textit{k} = \text{RootNode}[i];
    for \text{jth neighboring node of node k in NeighborNode}[i][j] do
        Get the neighboring node index: \textit{p} = \text{NeighborNode}[i][j];
        if \textit{p} \notin \text{RootNode} and \textit{p} \notin \text{TreeNode} then
            Add node \textit{p} into \text{TreeNode};
            Add nodes pairs (k,p) into \text{Tree};
\end{verbatim}

Then, given a triangular finite-element mesh, all the tree edges can be found in this way. Figure 3.12(a) shows a typical two-dimensional triangular mesh and its corresponding tree-cotree splitting pattern. The outmost boundary is a perfectly electric conductor (PEC). The bold lines in Figure 3.12(b) represent tree edges and the lines which are not highlighted are cotree edges. It is important to note that the number of the tree edges is equal to the number of the free nodes in the mesh. Then, the new basis functions are the combination of edge-based basis functions associated with the cotree edges and the nodal basis functions on the free nodes.

Shielding is widely used to reduce the magnetic field outside the transformer. It is meaningful to research on the influence of different structural parameters, such as electric properties or geometry, on the shielding effect. A set of problems has been designed and put forward by the International Compumag Society (ICS) for Testing Electromagnetic Analysis Methods.
The first problem analyzed is a three-dimensional TEAM Problem 21a-2, which is a member of the benchmark family [19], [20]. The model simulated is shown in Figure 3.13. This model contains a non-magnetic steel plate with two parallel slits in the middle. The exciting currents in the two exciting coils nearby are flowing in the opposite directions and take the forms of modulated Gaussian pulses with the central frequency of 50 Hz. The magnitude of the currents is 3000 A. The conductivity of the non-magnetic steel plate is \(1.3889106 \times 10^6\) S/m and the relative permeability \(\mu_r\) is 1.0. The magnetic flux densities at the points where the magnetic flux enters the conducting steel plate and the points where it exits the plate are computed. The Fourier transform is employed to convert the simulated time-domain results to the frequency-domain results at each sampled point. Figures 3.14 and 3.15 show the comparisons between the simulated results and the measured results. It can be seen from the figures that the simulated results agree very well with the measured results, which demonstrates the validity of present formulations and implementations for linear models. Figures 3.16 and 3.17 show the corresponding results after TCS is applied. The identical physical quantities are computed and compared to the measured results. The simulated results with TCS also coincide with the measured results very well. Hence, the implementation of tree-cotree splitting technique is verified.
3.5 Figures and Tables

Figure 3.1: Racetrack coil model: (a) coil boxed by ABC and (b) The top view of a racetrack coil.
Figure 3.2: Relative error with respect different mesh sizes.
Figure 3.3: Connectivity graph and system matrix without RCM reordering: (a) connectivity graph and its numbering and (b) nonzero entries distribution of the system matrix.
Figure 3.4: Connectivity graph and system matrix with CM reordering: (a) connectivity graph and its numbering and (b) nonzero entries distribution of the system matrix.
Figure 3.5: Connectivity graph after RCM reordering.
Figure 3.6: Nonzero entries of the system matrix: (a) without RCM reordering and (b) with RCM reordering.
Figure 3.7: Side view and top view of benchmark problem 10: (a) side view and (b) top view.
Figure 3.8: Model and mesh of the modified problem 10 in Cubit: (a) model and (b) mesh.
Figure 3.9: Comparison of the effects of the relative permeability on the iterative solver convergence when $\Delta t = 1500 \mu s$.

Figure 3.10: Comparison of the effects of the relative permeability on the iterative solver convergence when $\Delta t = 150 \mu s$. 

37
Figure 3.11: Comparison of the iterative solver with or without RCM re-ordering for $\mu_r = 500$ and $\mu_r = 2000$ when $\Delta t = 1500 \mu s$. 
Figure 3.12: Tree-cotree splitting of a 2D finite element mesh: (a) mesh and (b) tree and cotree edges.
Figure 3.13: Model of Problem 21a-2.
Figure 3.14: Simulated and measured results of magnetic flux density $B_x$ at $x = 5.76$ mm, $y = 0.0$ mm.

Figure 3.15: Simulated and measured results of magnetic flux density $B_x$ at $x = -5.76$ mm, $y = 0.0$ mm.
Figure 3.16: Simulated and measured results of magnetic flux density $B_x$ with TCS at $x = 5.76$ mm, $y = 0.0$ mm.

Figure 3.17: Simulated and measured results of magnetic flux density $B_x$ with TCS at $x = -5.76$ mm, $y = 0.0$ mm.
Table 3.1: Comparisons of simulation time and convergence for different time step sizes, maximum number of nonzeros in a row and with or without RCM reordering.

<table>
<thead>
<tr>
<th>Case #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Step Size $\Delta t(\mu s)$</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
<td>1500</td>
<td>150</td>
</tr>
<tr>
<td>Time Step Number</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>Nonzeros Number(^1)</td>
<td>80</td>
<td>80</td>
<td>160</td>
<td>240</td>
<td>500</td>
<td>80</td>
</tr>
<tr>
<td>Iterations Number(^2)</td>
<td>55</td>
<td>4770</td>
<td>171</td>
<td>58</td>
<td>18</td>
<td>181</td>
</tr>
<tr>
<td>Iteration Time(^2) (sec)</td>
<td>20.6</td>
<td>1825.8</td>
<td>93.2</td>
<td>45.2</td>
<td>22.4</td>
<td>67.0</td>
</tr>
<tr>
<td>ILU Time (sec)</td>
<td>159.8</td>
<td>502.2</td>
<td>1271.1</td>
<td>1954.0</td>
<td>4048.4</td>
<td>555.4</td>
</tr>
<tr>
<td>RCM Time (sec)</td>
<td>2.2</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Total Time(^3) (hour)</td>
<td>0.62</td>
<td>50.8</td>
<td>2.94</td>
<td>1.80</td>
<td>1.75</td>
<td>18.8</td>
</tr>
</tbody>
</table>

\(^1\) Maximum number of nonzeros in a row.
\(^2\) Iteration number and time for one time step.
\(^3\) Total estimated simulation time = RCM time + ILU time + iteration time $\times$ iteration number.
CHAPTER 4

MODELING OF NONLINEAR MACHINE PROBLEMS

4.1 Introduction

In a nonlinear medium, the constitutive parameters, such as permittivity or permeability, usually depend on the strength of the electric or the magnetic field. Taking the nonlinear materials in the optics area as an example, the permittivity of the materials is a quadratic or high-order polynomial function of the magnitude of the electric field [21]. Second and third harmonics are generated due to the nonlinear effect if a laser pulse is incident towards the materials. FDTD is widely applied in the nonlinear optics region because the medium under study is normally electrically long. It is possibly as long as hundreds of thousands of wavelengths, which results in a tremendous number of unknowns that is very time consuming to solve if the FEM is applied.

Compared to FDTD in the optics area, the FEM gains much more popularity in the simulation of nonlinear materials in the electric machine area where complicated geometry and complex materials are involved. This is due to the fact that most of the machines work at a low frequency ranging from 0 Hz to 1 kHz. Thus, the discretization density is determined more by the resolution of the geometry than by the variation of the wave propagation in the computational domain. These two features in the general machine problems explain the wide usage of FEM in the electric machine area. The nonlinearity is usually caused by the existence of the ferromagnetic materials where the ratio between the magnetic flux density $B$ and the magnetic field strength $H$ is no longer constant. The ratio depends on the magnitude of the flux density. The magnetization curve, or B-H curve, is nonlinear in this case. Moreover, the magnetizing current in the coil will generate a magnetic field in the ferromagnetic core material. If the current is turned off, the core materials may retain some magnetic field inside and a magnetic
hysteresis loop will be produced. As for soft ferromagnetic materials, the width of the loop is small and usually can be considered as a single nonlinear curve. Efficient algorithms for the simulation of nonlinear materials are still an open question today. The Newton-Raphson method is widely applied in solving nonlinear problems due to its fast convergence around the solution. It also has been applied in solving nonlinear machine problems with the polarization technique [22], [23], [24]. However, this algorithm mainly deals with nonlinear static magnetic field problems. The convergence of the Newton-Raphson method is usually the biggest concern because it greatly relies on a good initial guess. The Newton-Raphson method may be unstable or diverge if the initial guess is out of the convergent region. Several types of relaxation techniques have been implemented to improve the convergence of the Newton-Raphson method [25], [26], [27], [28]. It still limits its applications in solving nonlinear problems if hysteretic field problems are involved [29]. Recently, the fixed-point method (FPM) with global and local convergence has been proposed [29], [30]. The nonlinear term which needs to be updated in constructing the Jacobian matrix in the Newton-Raphson method is now moved to the right-hand side. In each iteration, only a column vector instead of the whole system matrix needs be updated. This saves a significant amount of time in the time-domain simulations. The fixed-point methods with global and local convergence have been applied to solve 1D or 2D nonlinear machine problems [31], [32].

This chapter mainly describes the methodology of the simulation of nonlinear materials by using the time-domain finite-element method. The cubic spline interpolation is introduced to interpolate the discrete measured data points. Then, both the Newton-Raphson method and the fixed-point method with global convergence are formulated and applied in the simulation. Finally some preliminary results are shown and discussed.

4.2 Cubic Spline Interpolation

The B-H curve relations are given by a set of discrete measured data. In order to obtain the values between the discrete sampled points, interpolation is needed to interpolate the tabulated data. There are various interpolation methods. Linear interpolation is the simplest one, where the interpo-
lating function is constructed by connecting every two consecutive sampled points. As for this interpolating function, the first and second derivatives are constants within each interval and they are undefined at the junction points. The cubic spline interpolation constructs an interpolating function that guarantees the smoothness of the first derivative and continuity of the second derivative at all the junction points [9]. The continuity of derivatives of the B-H curve is ensured when the evaluation of the Jacobian matrix is involved in the Newton-Raphson method.

The detailed cubic spline interpolation is defined as follows. In each interval, the cubic polynomial takes the form of

\[ P(x) = c_0 + c_1x + c_2x^2 + c_3x^3 \] (4.1)

Assuming that the polynomial function \( P(x) \) is defined within the interval \([0,1]\), the corresponding coefficients of the \( P(x) \) can be evaluated from

\[
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  1 & 1 & 1 & 1 \\
  0 & 1 & 2 & 3 \\
\end{bmatrix}
\begin{bmatrix}
  c_0 \\
  c_1 \\
  c_2 \\
  c_3 \\
\end{bmatrix} =
\begin{bmatrix}
  p_0 \\
  \frac{p_0'}{3} \\
  p_1 \\
  \frac{p_1'}{3} \\
\end{bmatrix}
\] (4.2)

where \( p_0 \) and \( p_1 \) are the polynomial values at \( x = 0 \) and \( x = 1 \), respectively, and \( p_0' \) and \( p_1' \) are the values of the first derivatives of the polynomial at \( x = 0 \) and \( x = 1 \), respectively. The coefficients can be calculated from (4.2) as

\[
\begin{bmatrix}
  c_0 \\
  c_1 \\
  c_2 \\
  c_3 \\
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  -3 & -2 & 3 & -1 \\
  2 & 1 & -2 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  p_0 \\
  \frac{p_0'}{3} \\
  p_1 \\
  \frac{p_1'}{3} \\
\end{bmatrix}
\] (4.3)

By substituting (4.3) into (4.1), we have [33]

\[
P(x) = (2x^3 - 3x^2 + 1)p_0 + (-2x^3 + 3x^2)p_1 \\
+ (x^3 - 2x^2 + x)p_0' + (x^3 - x^2)p_1'
\] (4.4)

which is the interpolating function of a third-order polynomial. By applying
the following variable substitution

\[ x = \frac{b - a}{L} \]  

(4.5)

where \( L = b - a \), we could map any interval \([a, b]\) to the interval \([0,1]\) and the above-mentioned interpolation function could then be applied. There are \( n \) segments in total with \( 4n \) undetermined coefficients. Given a set of data points, \( x_0, x_1, x_2, ..., x_n \) and the corresponding values \( y_0, y_1, y_2, ..., y_n \), we could obtain \( 2n \) equations by setting

\[ p^0_i = y_{i-1}, \quad i = 1, ..., n \]  

(4.6)

\[ p^1_i = y_i, \quad i = 1, ..., n \]  

(4.7)

for each interval \( i \). Other \( 2(n - 1) \) equations could be obtained by making the first and second derivatives continuous at the junctions of the intervals, namely

\[ \frac{dp^0_i}{dx^2} = \frac{dp^1_{i+1}}{dx^2}, \quad i = 1, ..., n - 1 \]  

(4.8)

\[ \frac{d^2p^i(x)}{dx^2} = \frac{d^2p^{i+1}(x)}{dx^2}, \quad i = 1, ..., n - 1 \]  

(4.9)

Thus, so far we have \( 4n \) unknowns and \( 4n - 2 \) equations in total. The last two equations can be obtained by setting the second derivatives at the two end points to zeros, i.e.

\[ \frac{d^2p^1(x)}{dx^2} = 0 \]  

(4.10)

\[ \frac{d^2p^n(x)}{dx^2} = 0 \]  

(4.11)

Now, by solving this system of equations, all the coefficients can be computed and the interpolating functions at each segment are determined.

4.3 TDFEM Formulation with Newton-Raphson Method

As for many practical applications in the machine area, the relation between the magnetic field \( \mathbf{H} \) and the magnetic flux density \( \mathbf{B} \) is characterized as a
nonlinear function $H(B)$. The reluctivity $\nu$ in the system matrix $[S]$ shown in (2.21) depends on the strength of the magnetic flux density and this results in a nonlinear system of equations that need to be solved in every time step. Several methods have been proposed to solve such kinds of nonlinear problems. Among them, the Newton-Raphson (NR) method is recognized as one of the most popular methods and is widely applied to seek the solutions to a nonlinear system of equations [34].

Here, the system matrix $[K]$ in (2.45) can be firstly decomposed into a linear part and a nonlinear part as follows

$$([K_l] + [K_{nl}])\{\phi\} = \{b\} \quad (4.12)$$

where $[K_l]$ and $[K_{nl}]$ are given by

$$[K_l] = [M] + \frac{1}{2}c_0\Delta t([G] + [B]) \quad (4.13)$$
$$[K_{nl}] = \frac{1}{4}c_0^2\Delta t^2[S] \quad (4.14)$$

The Jacobian matrix can be calculated by

$$J^K_{ij} = K_{ij} + \frac{1}{4}c_0^2\Delta t^2\left\{\frac{\partial S_i}{\partial \phi_j}\right\}^T \{\phi\} \quad (4.15)$$

where $T$ denotes the transpose of the vector. In the derivation of the above expression, the following identities are applied

$$\frac{\partial \nu(\|B\|)}{\partial \phi_j} = \frac{\partial \nu(\|B\|)}{\|B\|} \frac{\partial \|B\|}{\partial \phi_j} \quad (4.16)$$
$$B = \nabla \times A = \sum_j \phi_j \nabla \times N_j = \sum_j \phi_j N_j^{\text{curl}} \quad (4.17)$$
$$\frac{\partial \|B\|}{\partial \phi_j} = N_j^{\text{curl}} \cdot \frac{B}{\|B\|} \quad (4.18)$$

The values of $\frac{\partial \nu(\|B\|)}{\partial \|B\|}$ can be calculated from the B-H curve obtained from the cubic spline interpolation of the measured data. Then, with a proper initial guess at the beginning of the time-marching scheme, a set of nonlinear equations is solved iteratively in each time step. The nonlinearity-related term $[K_{nl}]$ will be updated in the iterations based on the information of
previous iterative step. The converged solution vector obtained from the current step is then applied as the new initial guess for the iteration in the next time step. This process is repeated in the range of interest.

4.4 TDFEM Formulation with Fixed-Point Method

Though the aforementioned Newton-Raphson method converges quadratically around the true solution, it largely relies on a good initial guess. Sometimes, it may fail to converge if the initial guess is not within the vicinity of the solution. In such a case, a relaxation factor needs to be implemented to modify the size of the Newton updating step so that the decrease of the residual in each iterative step is guaranteed. However, for each iterative step, the Jacobian matrix needs to be recalculated and reassembled, which will be repeatedly done during each time step. This is very computationally expensive if the size of the problem becomes large. Moreover, the necessity of repeatedly updating the system matrix impedes the prefactorization and preassembly of the system matrix in the direct solver and requires regenerating the preconditioner in each iterative step if an iterative solver is used. Thus, this method, although it seems to be promising at first glance, could hardly be implemented practically in the current context.

An alternative remedy is employing the fixed-point method. The magnetic field $\mathbf{H}$ is first expressed as

$$\mathbf{H} = \nu_{opt} \nu_0 \mathbf{B} + \mathbf{R}$$

(4.19)

where $\nu_{opt}$ represents the linearized reluctivity and $\mathbf{R}$ is the nonlinear term.
as a function of \( B \). By plugging (4.19) into (2.10), we have

\[
\begin{align*}
\iint_V N_i \cdot \nabla \times \nu_{opt} \nabla \times \sum_{j=1}^{N} \phi_j N_j \, dV \\
+ \iint_V \frac{\varepsilon_r}{c_0} \frac{\partial^2}{\partial t^2} N_i \cdot \sum_{j=1}^{N} \phi_j N_j \, dV \\
+ \iint_V \frac{\varepsilon_0 \sigma}{c_0} \frac{\partial}{\partial t} N_i \cdot \sum_{j=1}^{N} \phi_j N_j \, dV \\
= \iint_V \mu_0 N_i \cdot \mathbf{J}_{im} \, dV - \iint_V \mu_0 N_i \cdot \nabla \times \mathbf{R} \, dV 
\end{align*}
\] (4.20)

Compared to (2.12), we only need to modify the previous formulation by adding an extra nonlinearity term on the right-hand side of equation. The good feature of this method is that the \( \nu_{opt} \) is fixed during the time-marching process, only the nonlinearity term on the right-hand side of the equation needs to be updated instead of the whole system matrix. Such a method saves a significant amount of time by avoiding reassembling and refactorizing the system matrix or regenerating the preconditioner repeatedly in each time step. Normally, \( \nu_{opt} \) is chosen as

\[
\nu_{opt} = \frac{\nu_{\min} + \nu_{\max}}{2} 
\] (4.21)

where \( \nu_{\min} \) and \( \nu_{\max} \) are the minimum and maximum relative reluctivities, respectively, obtained from the nonlinear B-H curve. This method is the so-called global convergence method (GCM) as described in [35].

### 4.5 Numerical Examples and Results

The example simulated here is the benchmark problem 10 [36], where three steel plates composed of nonlinear materials surround a coil. Figures 3.7(a) and 3.7(b) show the side and top views of the simulated model, respectively. The reluctivity of the nonlinear materials (steel plates) depends on the strength of the magnetic flux density, and their relation is characterized by a H-B curve obtained by using the cubic spline interpolation of the measured data. Figure 4.1 shows the measured H-B relation and interpolated...
curve. The conductivity of the nonlinear materials is $7.505 \times 10^6$ S/m. The excitation current inside the coil takes the form of

$$I = \begin{cases} 
0, & x < 0, \\
I_m(1 - e^{-t/\tau}), & x \geq 0.
\end{cases} \quad (4.22a)$$

where $I_m = 5.64$ A is chosen to ensure that the plates are sufficiently saturated and $\tau$ denotes the time relaxation factor which is chosen to be 0.05 s so that the eddy current inside the coil can be neglected.

Figure 4.2 shows the simulated results of problem 10 by using the fixed-point method with the global convergence method (GCM). The time variations of the average magnetic flux densities at the cross sections of three steel plates are recorded and compared with those obtained from measurement [36]. It can be seen from Figure 4.2 that the simulated results are generally consistent with the measured results when nonlinear plates are sufficiently saturated. However, some deviation happens in the unsaturated region, which may be because that in the unsaturated region, the variation of the fields between two consecutive time steps might be large as the time-marching process proceeds. The time step is not small enough to accurately capture the variations of fields within the whole computational domain. Moreover, problem 10 contains fine geometries, such as air gap and thin steel plates, which require a fine mesh to characterize the geometry and field variation inside. However, refining the mesh in these regions will lead to a much smaller mesh size compared to most of other mesh elements in the computational region. This highly nonuniform mesh combined with a highly inhomogeneous medium results in a very ill-conditioned system matrix which is very time consuming for either a direct solver or an iterative solver to solve. The convergence of a linear iterative solver can be very slow and impractical to solve. On the other hand, a uniform discretization in the entire computational domain results in an unnecessarily dense mesh in the vast air region, making the number of degrees of freedom unnecessarily huge. Moreover, the convergence of the current approach has severe problems. In many cases, it would fail to converge after the time-marching process, proceeding about one-third of the whole simulation. Once it diverges at one step, it would never converge in the subsequent time steps. Some techniques are being applied to improve the condition of the system matrix. Future research should aim at
the improvement of the convergence of the nonlinear solver.
4.6 Figures

Figure 4.1: Nonlinear B-H curve.

Figure 4.2: Simulated and measured results of benchmark problem 10.
This work first derives the formulation of the time-domain finite-element method for analyzing three-dimensional linear electrical machine problems. The verification of the linear formulation for the TDFEM is carried out by modeling and testing a simple racetrack coil problem. The convergence of the solution demonstrates the validity of the linear TDFEM formulation. This is followed by introducing the ILU preconditioner and the Cuthill-McKee reordering technique to improve the condition of the system matrix. The influence of their implementation on the convergence of the linear solver is shown and analyzed. We can see from the results that the implementation of RCM reordering could significantly minimize the bandwidth of the system matrix and then improve its condition if the same ILU preconditioner is used. Then, the tree-cotree splitting algorithm is applied to remove the singularity of the system matrix if a low-frequency excitation is present. A linear benchmark problem 21^n-2 is simulated to validate the algorithm. Consistent results are obtained, which demonstrates the correctness of implementation of the algorithm.

In the last chapter, the model of nonlinear machine problems is introduced and analyzed. In order to obtain a continuous B-H curve which would be used in the iterations, cubic spline interpolation is employed to interpolate the values between the discrete measured data points. Two nonlinear solver algorithms, the Newton-Raphson method and the fixed-point method, are incorporated into the TDFEM formulation. The benchmark problem 10 is shown and simulated by the proposed algorithm. Some deviation is observed, although the simulated results roughly follow the trend of the measured results. The possible reason for such a discrepancy is that although the mesh in the linear region is already dense enough, it may not be small enough in the nonlinear region which has fine geometry to model. The lack of dense mesh in the nonlinear region leads to the loss of accuracy in modeling the
fields in the nonlinear region. Further investigation is needed to improve the convergence of the nonlinear solver and enhance its efficiency in solving larger-size nonlinear systems.
REFERENCES


