CAD IMPLEMENTATION OF FINITE DIFFERENCE ALGORITHMS FOR THE ANALYSIS OF HIGH-SPEED CIRCUITS

BY

ZHICHAO DENG

B.Engr., Tsinghua University, 2000
M.S., University of Illinois at Urbana-Champaign, 2002

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2006

Urbana, Illinois

CERTIFICATE OF COMMITTEE APPROVAL

University of Illinois at Urbana-Champaign
Graduate College

March 1, 2006

We hereby recommend that the thesis by:

ZHICHAO DENG

Entitled:

CAD IMPLEMENTATION OF FINITE DIFFERENCE ALGORITHMS FOR THE ANALYSIS OF HIGH-SPEED CIRCUITS

Be accepted in partial fulfillment of the requirements for the degree of:

Doctor of Philosophy

Signatures:

Jose E. Schutt-Aine
Director of Research -

Head of Department -

Committee on Final Examination*

Jose E. Schutt-Aine
Chairperson - Jose E. Schutt-Aine

Committee Member - Andreas Cangellaris

Committee Member - Martin D. F. Wong

Committee Member - Naresh Shanbhag

Committee Member -

Committee Member -

* Required for doctoral degree but not for master's degree

ABSTRACT

This dissertation addresses a time-domain simulation method for use in high-speed circuit simulation. Efficient and very fast simulation is a requirement for today's high-density, high-speed circuit designs. Recently, a time-domain formulation, latency insertion method (LIM), was proposed that leads to the generation of update algorithms for the simulation of networks. The algorithms exhibit linear computational complexity and are scalable. Because of the time domain nature of the formulations, they can be extended to handle nonlinearities.

The basic goal of this dissertation is to extend the speed, accuracy, and application range of the LIM method. First, the formulation of the LIM method for the case of linear networks is presented by deriving the update algorithms. Next, the stability of these algorithms is addressed followed by an extension of the formulation to special elements and nonlinear networks. Then the possibility of extending unconditional stability to LIM is discussed. Finally, several networks are analyzed and simulated using the method for comparison with standard simulators. Trade-offs among speed, stability, and accuracy are examined throughout the comparisons

Another objective of this dissertation is to address the importance of computer-aided design (CAD) implementation of the latest techniques into the existing platform. Good research should lead to useful and repeatable results. This dissertation therefore briefs the implementation of an *n*-port transmission line model into SPICE3f4, which also enables the integration of external simulation programs for *n*-port devices. Next this dissertation also shows the integration of LIM into SPICE, which produce an easy-to-use, and powerful simulator, which retains the familiar interface of SPICE simulator and the fast speed of LIM into an entity.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Background

The need for an increased density of input-output (I/O) pin connections and the reduction in size of high-speed digital circuits have led to an increase in the complexity of interconnect schemes. At the board and package levels the implementation of multilayer interconnects has led to structures with a high density of passive components. As a result, the three-dimensional (3-D) nature of present-day networks has rendered their analysis more challenging.

These challenges are also emerging at the chip level. As outlined by the International Technology Roadmap for Semiconductors (ITRS), the acceleration in processor performance has led to an increasing gap between manufacturing technology and present-day design tools. Total interconnect lengths will lead to unprecedented wiring density which will require novel design methodologies placing more emphasis on interconnect issues.

For example, in sub-130-nm designs, inductive effects during switching must be taken into account in the computer-aided design (CAD) of on chip power grid. Furthermore with signal bandwidth above 60 GHz, the high-frequency electromagnetic interference effect becomes quite a big problem for power grid design.

The simulation of very large networks consisting of large numbers of nodes is a major problem in the CAD of integrated circuits. Circuits of this size can typically require several days of CPU time on a workstation. There are two directions for solving this problem. The first is to reduce the problem size by utilizing different techniques (model-order reduction). The second is building a fast solver that directly solves the problem.

For the first approach, several investigators have introduced algorithms and numerical techniques such as the asymptotic waveform evaluation (AWE) [1]-[3] method to approximate network transfer functions. The fundamental idea behind the first approach, model-order reduction, rests in the implementation of a circuit representation based on a smaller number of poles than the original network. These poles account for most of the behavior of the network over the frequency range of interest. The resulting macromodel equivalent circuits can be used in conjunction with standard circuit simulators.

Work was later introduced to reduce the number of spurious poles generated by the reduction process. This includes the complex frequency hopping techniques, and the Krylov subspace methods [4]-[7]. More recent work on model order reduction techniques have focused on the passivity of the reduced equivalent circuits [8], [9]. There is also some direct reduction techniques such as Y-Δ transformation technique [10]. And hierarchical model order reduction [11] and frequency domain analysis [12] are some other recently proposed methods.

The second path we can take is to build a fast simulator. In this approach finite difference method is a popular and suitable candidate due to its simplicity and possible linear computational complexity. Several techniques [13], [14] have been developed to speed up the analysis. The transmission matrix method [13] is based on a multi-input/multi-output transfer function, which enables the entire network to be computed as the product of several small individual sparse square matrices. The transmission matrix method is about 10 times faster than SPICE. The preconditioned conjugate gradient (PCG) simulator [14] is based on the preconditioned Krylov-subspace iterative method, which is significantly faster than traditional iterative methods without preconditioning. The hierarchical method [15], [16] is an alternative method. The finite difference method (finite difference, finite element or finite volume

methods) can simulate large problems through discretization of Maxwell's equations. Among these, the finite-difference time-domain (FDTD) method is a preferred method for regular network. Some similar technique like transmission-line-modeling (TLM) [17] was also developed, which can solve voltage and current directly on regular network.

There is transmission-line-modeling alternating-direction-implicit (TLM-ADI) methods [18]-[19], which has two-dimensional (2-D) and 3-D versions. The 2-D TLM-ADI method can be only be applied to regular 2-D mesh network. And 3-D TLM-ADI method has the limitation of the fixed ratio of resistor/inductor for the entire network.

1.2 Objective

In this dissertation, we use a time-domain formulation that leads to the generation of update algorithms for the simulation of networks. The algorithms exhibit linear computational complexity and are scalable. Because of the time domain nature of the formulations, they can be extended to handle nonlinearities.

First, we present a formulation of the method for the case of linear networks by deriving the update algorithms. Next, the stability of these algorithms is addressed followed by an extension of the formulation to special elements and nonlinear networks. Then the possibility of extending unconditional stability to latency insertion method (LIM) is discussed. Finally, several networks are analyzed and simulated using the method for comparison with standard simulators. Trade-offs between speed, stability and accuracy are examined throughout the comparisons

Another objective of this dissertation is to address the importance of CAD implementation of latest techniques into existing platform. Good research should lead to useful and repeatable results. For the above purpose, this dissertation details the implementation of an

n-port transmission line model into SPICE3f4, which also enables the integration of external simulation programs for n-port devices. Next this dissertation also shows the integration of LIM into SPICE, which produce an easy-to-use, and powerful simulator, which retains the familiar interface of SPICE simulator and the fast speed of LIM into an entity.

1.3 Organization

This document is organized as follows. Chapter 2 provides the basic information of SPICE device model installation and demonstrates it by installing an n-port transmission line model into SPICE. It also includes information on the general structure of a stamp-based circuit simulator such as data structure, device stamps, and the construction of circuit matrices. In Chapter 3, LIM is introduced in detail from several different aspects such as stability, speed, nonlinearity, frequency-dependent modeling, and unconditional stability. Chapter 4 gives the integration guide for LIM and SPICE. Chapter 5 shows the different numerical examples and comparison between SPICE and LIM. Chapter 6 is the conclusion.

1.4 SPICE3f4 Comments

The source code for SPICE3f4 is available from the University of California (UC), Berkeley free of charge. The version 3f4 is the final released version and technical support from UC Berkeley is no longer available. The program is written in C. Proficiency in C language is a prerequisite. The source was modified, compiled, and installed during the device installation procedure. In order to make the device installation procedure more comprehensible, code listings are provided throughout the dissertation, and only relevant portions of the code will be shown to conserve space. Copyright of the code listings is not shown. SPICE3f4 is copyright

1990 by the regents of the University of California. All rights reserved. SPICE was originally written in 1985 by Thomas L. Quarles.

2 N-PORT MULTICONDUCTOR TRANSMISSION LINE MODEL IMPLEMENTATION ON SPICE3F4

2.1 Introduction

Due to the rapid development of analog and digital electronics, computer-aided design becomes a very important part of the design of integrated circuits (ICs).

As process geometries continue to shrink, ever-thinner wires are packed more and more tightly together, producing unintended electrical effect—such as capacitance coupling and voltage drop—that impair signal integrity. These complex signal-integrity issues have profound effects on path delays and design timing. As process geometries move to 0.13 μm and below, signal-integrity timing issues have become a primary concern. They impede timing convergence, consume engineering resources, and result in needless overengineering, overly constrained designs, and costly schedule delays.

One of the fundamental problems in signal integrity is to simulate multiconductor transmission lines in a multifunctional simulator, or one that can perform several kinds of analyses, not just ordinary circuit analysis. For instance, as the speed of the clock increases and the IC becomes smaller, the delay of the interconnections becomes the dominant delay. The interconnections cannot be regarded as lumped element when the clock speed increased. Instead, they must be modeled as transmission lines to deal with the effects of the crosstalk noise, waveform distortion and signal attenuation effects [20]. This leads to the simulation of multiconductor transmission line in a circuit simulator.

The discussion of device installation and modification is based on the basic knowledge of how a circuit simulator functions. So we need to understand the structure and operation of the simulator.

Here we modify a standard circuit analysis package to fulfill the basic requirement of simulating n-port devices in a simple way. In this work, we will discuss the installation and modification of devices within a SPICE simulator. We use SPICE3f4 because it is a circuit simulation package that is widely used in many academic and commercial simulators and it is an open source code.

Previous work implemented an n-port dummy device in SPICE3f4. The n-port transmission line model can be implemented from that framework. The difficulty lies in two aspects: (a) interface between input and actual matrix filling; (b) the modification of device data structure and original independent algorithm to incorporate it into SPICE's variable timestep and order LMS (linear multistep) scheme. The first part involves the creation of appropriate parameters for user input. It also provides a mapping between the internal node and external terminals. The second part adds necessary contents into data structure and modifies the original algorithm to accommodate the variable time step environment of SPICE3f4.

2.1.1 General circuit simulation procedure

A general circuit simulation procedure is shown in Figure 2.1. First, the simulator needs the circuit description to get the netlist for that circuit. Then the information in the netlist is extracted out and refilled in an organized matrix form (this is the modified nodal form in SPICE). A matrix Equation (2.1) is also constructed, where $\mathbf{A}$ is an $n \times n$ matrix and $\mathbf{b}$ and $\mathbf{x}$ are $n \times 1$ vectors. This matrix equation contains almost all the information needed for the circuit

simulation including circuit structure information and the unknown nodal voltages and branch currents of the circuit.

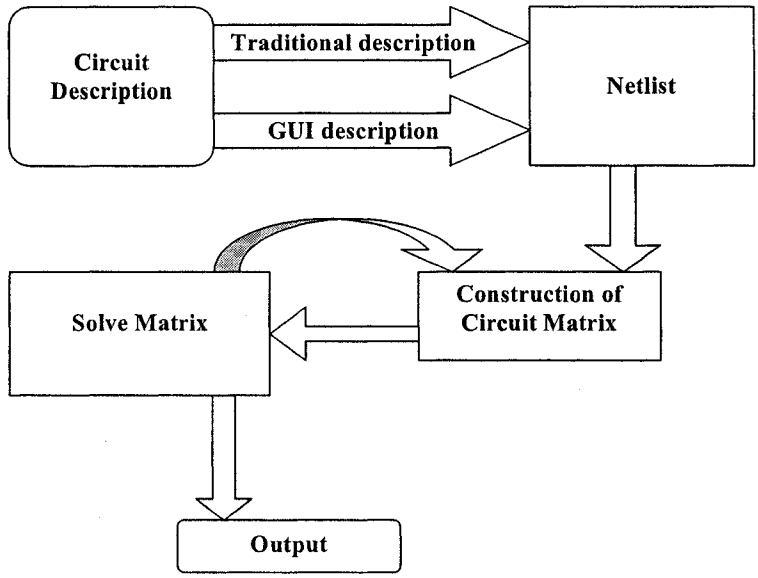


Figure 2.1 General procedure of circuit simulation

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{b} \tag{2.1}$$

The matrix **A** contains the information of the elements in the circuit, i.e., resistor, inductor, and capacitor. The vector **b** contains the independent voltage and current sources. The vector **x** contains the nodal voltages and branch currents that need to be determined.

The solution of Equation (2.1) may be repeated multiple times, depending on the type of analysis to be performed. Due to the excellent modularization of SPICE, we can examine each step without interrupting other parts. In the following sections, each step in Figure 2.1 will be presented. We will focus on the construction of circuit matrices, which deal for the most parts with the device installation process in SPICE.

2.1.2 Circuit description and netlist

The circuit simulator needs the information about the circuit to commence. There are usually two ways for users to describe the circuit and simulation information. The traditional way is a text file that contains the circuit element, topology, and simulation parameters. The simulator reads in the text file line by line and forms the netlist. Alternatively, users can draw the circuit through a graphical user interface (GUI), which is available in all of the commercial circuit simulation packages. The drawing will be converted to the netlist form as the traditional way. The netlist will be processed, and the information is stored in a SPICE-defined data structure, which will be discussed in later sections.

2.1.3 Data structure

The data structure is related to the algorithms of constructing the circuit matrix. There are two approaches to constructing a circuit matrix: node-by-node approach and device-by-device approach. The node-by-node approach is quite straightforward. The matrix is constructed by writing equations using Kirchoff's current law (KCL) node by node. This causes a data structure that is a linked list containing the information associated with each node. However, it is inefficient. For instance, let there be a device connected between nodes 1 and 2. The node-by-node approach will fill the device into the circuit matrix once at each node. So the constructing procedure will access the matrix twice for each device or element in the circuit.

To improve the efficiency of the node-by-node algorithm, SPICE introduce the device-by-device approach. It accesses the matrix only once per device and results in a data structure of a linked device list. The data structure shown in Figure 2.2 is a linked device element. Each device is an element, which has four pointers: two to point to the previous and following device

types, one to point to the parameter list and one to point to the node list. The detail of device-by-device filling routine is explained in the next section.

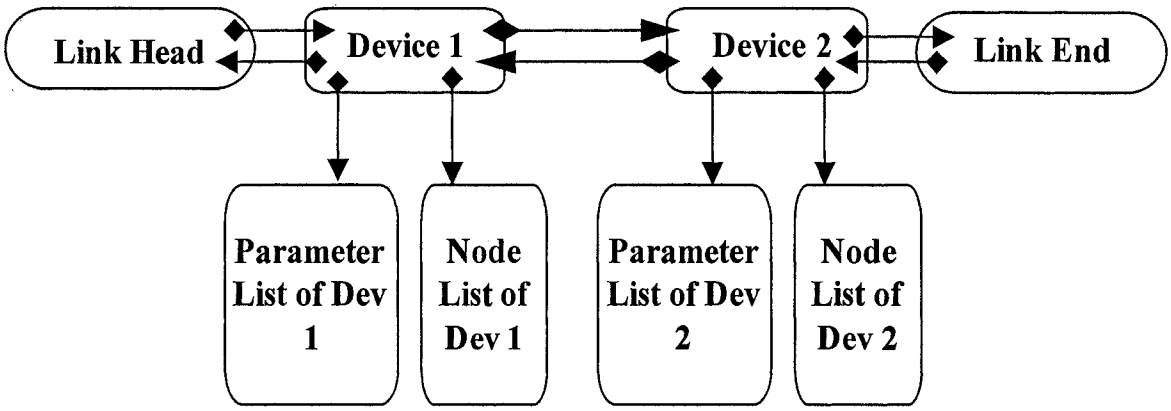


Figure 2.2 Data structure used in the device-by-device approach

2.1.4 Construction of circuit matrix

In SPICE, the construction of the circuit matrix step uses a device-by-device filling routine. The flow chart of the matrix construction is shown in Figure 2.3.

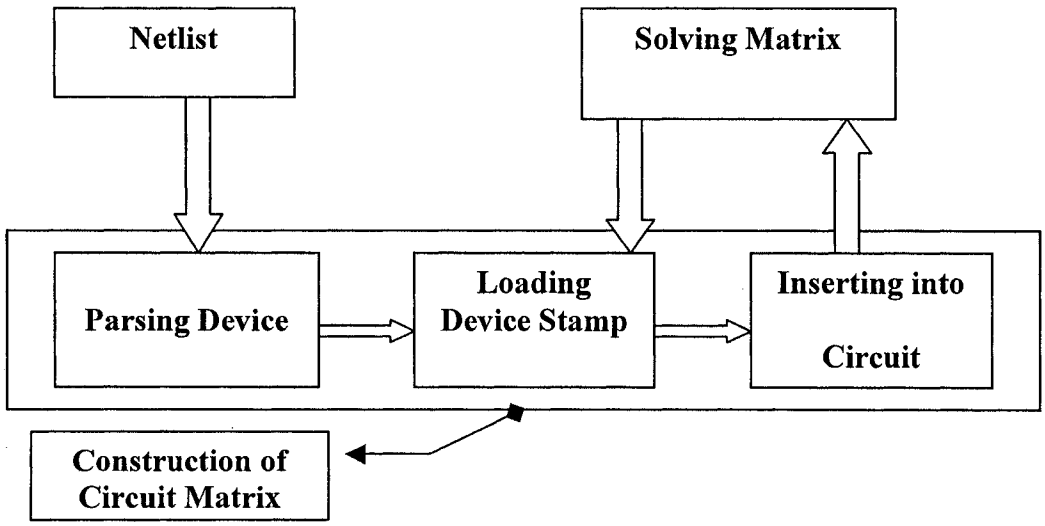


Figure 2.3 Detailed steps of construction of a circuit matrix

First, the netlist is parsed to get the corresponding device type and parameter. Then the routine turns to create the specified device type data structure and create a new device instance in the linked structure with the information of the element parameter. Parsing of a device is very important in the device installation procedure and will be covered in later section. The device stamp will be discussed in the next section.

After going through all the devices in the netlist, the data structure is formed and categorized by device type. This data structure facilitates the loading of the device stamp, which begins the actual matrix construction procedure. The procedure loads one device type and goes through all its device instances and fills the information to the matrix. Then it loads another device type and goes on until all the device types are loaded. At last the circuit matrix is ready to be solved by the simulator matrix solver.

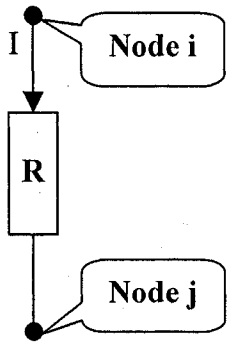
2.1.5 Device stamps

Device stamp is a technique used by SPICE and many other simulation programs to construct the modified nodal matrix. The basis of this technique is that each device makes an independent contribution to the circuit matrix. The detail is shown in [21]. Each stamp represents a pattern, which is the way that a device makes its own contribution to the matrix. We show a simple linear resistor stamp to reveal this concept.

A linear resistor is described by Equation (2.2):

$$V = iR \tag{2.2}$$

The stamp for linear resistor is the same for all analysis including dc, ac, and transient analysis. We assume that a linear resistor is connected to node i and j, which is shown in Figure 2.4(a). The linear resistor's stamp is shown in Figure 2.4(b).



(a)

$$\begin{array}{c} \\ node\_i \\ node\_j \end{array} \begin{array}{c} \begin{array}{cc} V_i & V_j \end{array} \\ \begin{bmatrix} 1/R & -1/R \\ -1/R & 1/R \end{bmatrix} \end{array} \begin{bmatrix} V_i \\ V_j \end{bmatrix} = \begin{bmatrix} I_i \\ I_j \end{bmatrix}$$

(b)

Figure 2.4 (a) A simple linear resistor and (b) its device stamp

The Equations (2.3) and (2.4) at node i and j

$$\frac{V_i - V_j}{R} = I_i \tag{2.3}$$

$$\frac{V_j - V_i}{R} = I_j \tag{2.4}$$

are in matrix equations row 1 and 2 obtained by using Kirchoff's voltage law (KVL) and KCL. We can solve the matrix equation for V_i and V_j, if we know the value of R. From this simple resistor stamp, we can see that every element only takes effect on the nodes where it is connected. The more details on device stamps can be found in [21], [22]. In SPICE, each device has a unique device stamp. We can create stamps from the device's intrinsic equation, experimental data, or behavioral models.

Another point to note is that the circuit matrix is dynamically constructed to avoid the unknown size of the circuit. The simulator allocates memory for each new row and column of the circuit matrix. While filling a device instance's information, if the corresponding row and column in the circuit matrix is already created and occupied by some value, the new value will be added to the old value.

2.1.6 Matrix solution techniques and sparsity

If the circuit network is linear, the circuit matrix is a linear equation matrix. A nonlinear circuit network leads to a nonlinear equation matrix. But solving the nonlinear equation matrix linearizes them. So solving a linear equation matrix is a fundamental technique. Many techniques, both direct and iterative, are used to solve a linear equation matrix. The direct method is often used for operating point (op) and dc bias-point (dc) analysis, and the iterative method is used for nonlinear device analysis such as transient analysis. In transient analysis, the matrix is solved directly or iteratively at each time step, depending on the type of device in the simulation. The transient analysis will be discussed in the next section. The direct method of Gaussian elimination, such as lower-upper matrix (LU) factorization, is often used [22]. The indirect method, such as Jacobi iteration and Gauss-Seidel (GS) or successive over-relaxation (SOR) iteration, is based on the converge condition of the error of some chosen function.

The sparsity is the characteristic of sparse matrix, which is the normal case in usual circuits. Sparse matrix means the entries of matrix are mostly zero. There are special techniques to manipulate sparse matrices for time and space saving. In SPICE, there is a sparse matrix package named `sparse`. Sparse matrix algorithms neither store nor perform operations on zeros. More details can be reached in [22], [23].

2.1.7 Simulator output

After the matrix is solved, the result is stored in many data structures. Then the routines that can store, manipulate and plot the results begin operating.

2.1.8 Analysis

There are many analyses in a circuit simulator such as ac, dc, transient, pole-zero, and sensitivity analyses. Among them, dc, ac, and transient analyses are the key for simulating transmission lines.

An ac analysis computes a dc operating point and all of the necessary small signal parameters, and sweeps all ac sources through a set of frequencies, computing ac small signal response at these frequencies.

A dc analysis provides a simple dc analysis with all capacitances open-circuited and all inductances shorted. It does not require any parameter and always be called by other analysis as a preliminary step.

The transient analysis is the most complicated simulation. It permits the time dependent behavior of the circuit to be analyzed [24]. The whole procedure is shown in Figure 2.5.

2.1.9 Summary

In this chapter, we discussed the basic procedure of circuit simulation. The construction of the matrix is most important in the device installation procedure. In SPICE, the method of matrix construction uses stamps to insert devices into a circuit matrix. Any device must use its stamp to be modeled using voltages and currents. The stamp can be derived from the device's intrinsic equation, experimental data, or behavioral models. Transient analysis is very complicated and important in circuit simulation. The time step is a very important issue in some device installation.

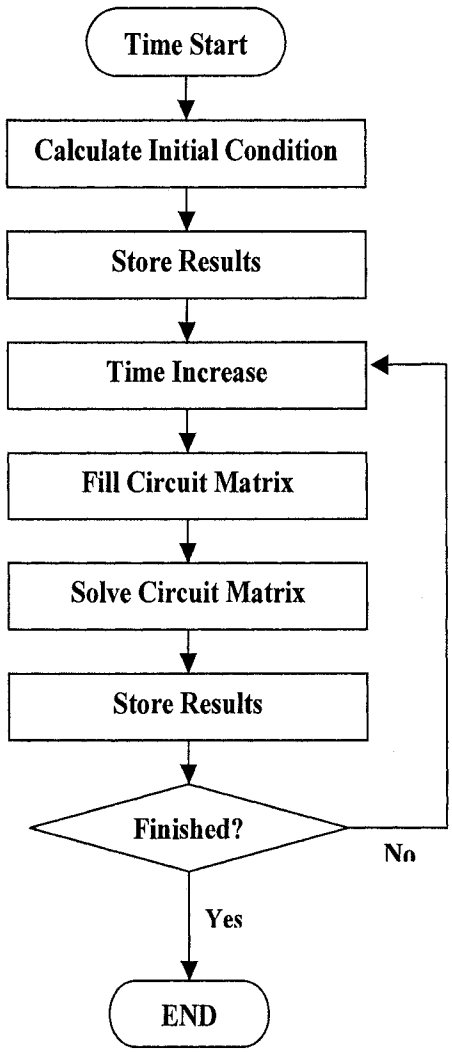


Figure 2.5 Flow chart of transient analysis

2.2 Detailed Implementation

The implementation has two stages. First a new model, NTRA, is modified the same way as in the dummy *n*-port device model PRES, which is implemented in [21]. Then the NTRA model can handle any number of ports. Second the data structure and the loading of the matrix process in ntraload.c and other files are modified to load in the appropriate current source and stamps. It also needs to record the previous timestep's information to be used to calculate future value.

2.3 User Manual

This is a simple guide for using the integrated *N*-line transmission line device NTRA in SPICE3f4.

2.3.1 Procedure for testing *n*-port transmission line model in SPICE

This new device can handle an arbitrary number of transmission lines with given RLGC parameters.

```
1.    run 'spice'
2.    In spice environment
      'source testnl.cir'
      'run'
      'plot v(3)'
      'plot v(4)'
      'plot v(5)'
      'plot v(7)'
3.    Quit SPICE using 'quit'
4.    To reset the display, use the 'reset' command
```

To edit the netlist and *N*-line parameter files, you can use "emacs filename &."

To plot a graph in SPICE, use X-win32 to login to a unix/linux machine. From there you can telnet into jsa4.ece.uiuc.edu, then you can plot the graph directly. Otherwise you will not be able to plot in SPICE. You have to use the "print" command.

2.3.2 New device "NTRA" for multiconductor transmission line

The *N*-line transmission line has $2N$ terminals. The numbering of the terminal should follow the rule shown in Figure 2.6, which will be discussed in detail later. The transmission line parameters and initial conditions are stored in a parameter file. Below is an example to illustrate the use of the new device NTRA in the spice netlist.

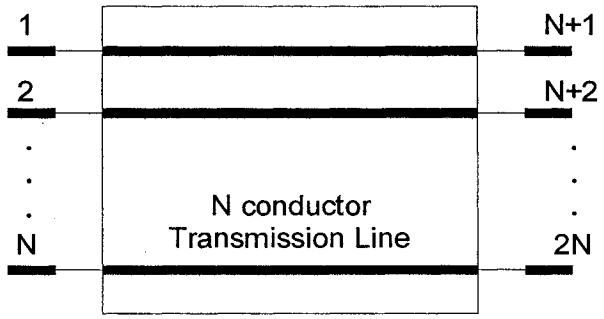


Figure 2.6 Port naming of N conductor transmission line

In the example circuit shown in Figure 2.7, there is a three-line transmission line device named ntra1. The format of the new *N*-line device NTRA is like:

```
NTRAxxx  # of terminals   node1  node2 ... node2N   nline=# of lines
nodes=# of terminals   userdata=RLGC datafile
```

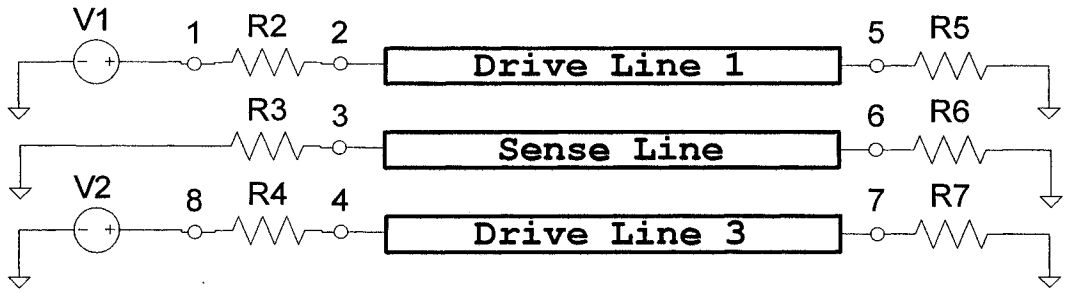


Figure 2.7 Test example of a three conductor transmission line

The netlist of this example circuit is listed as below:

```
Multiconductor Transmission Line (3 line) Test Circuit
v1 1 0  pulse(0 1 1ns 1ns 1ns 20ns 50ns)
R2 1 2 50
R3 0 3 50
R4 8 4 50
ntra1 6 2 3 4 5 6 7 nline=3 nodes=6 userdata=nlinedata1
R5 5 0 10000
R6 6 0 10000
R7 7 0 10000
v2 8 0  pulse(0 1 1ns 1ns 1ns 20ns 50ns)
.tran 0.1ns 40ns 0ns 0.1ns
.END
```

Now the details of numbering sequence of the NTRA device's terminals and the parameter file will be addressed in the following subsections.

(A) The node numbering sequence

The numbering follows the sequence top-left, bottom-left, top-right, bottom. You MUST name the terminals in a continuously increasing numbering sequence such as *4 5 6 7 8 9*.

An example of naming sequence is shown in Figure 2.8. Here we name the top-left terminal *4*, followed by *5*, and all the way to *9*.

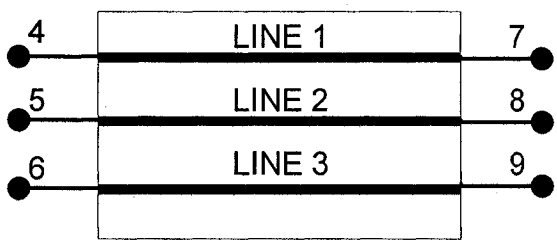


Figure 2.8 The naming example of a three-line device

(B) The nline and nodes parameters

In the semantics, *nline* is the number of lines and *nodes* is the total number of terminals. Normally, for an *N*-line device, *nline* = *N* and *nodes* = 2*N*.

(C) The format of the userdata file

The *userdata* file should be located in the directory, where you execute the command "*spice.*" The first line of this file has three parameters: "*mh,*" the number of time-points used for frequency determination for *tlmod()*; "*lmax,*" the order of approximation matrix; "*npar,*" some parameter used for parameter storage. This file contains the information about the *R, L, G, C* parameters of the *N*-line model and the initial voltage and current of each terminals and the length of each transmission line. Here the units of all the *L* and *C* are Nanohenry/meter or Nanofarad /meter. The unit of the length is in inches.

3 LATENCY INSERTION METHOD (LIM)

3.1 Introduction

The simulation of networks with large numbers of nodes is serious challenge in the computer-aided design of integrated circuits. For example, power grids are rapidly becoming a limiting factor in high-performance microprocessors, the ability to analyze power grids efficiently is a critical requirement to obtain a robust design [25]–[28]. Circuits of this size can typically require several days of CPU time on a workstation. The multigrid method [29], hierarchical method [15],[16], PCG [14], hierarchical model order reduction [26], and frequency domain analysis [27] are the latest methods proposed. For solving these large problems, there is also TLM-ADI methods [18],[19], which has 2-D and 3-D versions.

The latency insertion method was recently developed to simulate the high-frequency response of large networks in the time domain [30]. The method makes use of or introduces reactive latency in all branches and nodes of a circuit to generate update algorithms for the voltage and current quantities. The updating of branch currents and node voltages is performed in a leapfrog manner similar to the Yee algorithm used in the FDTD method [31]. Consequently, LIM has linear computational complexity and is thus substantially faster than the traditional matrix-vector product based methods such as the modified nodal analysis used in SPICE. The LIM formulation is enabled through the insertion of small inductors and capacitors throughout the network, which generate the latency at each node and branch and also impose conditions on the time step size and stability of the solution. In [32], the stability of the LIM solution is investigated in a manner analogous to the traditional FDTD schemes [33],[34].

The remainder of the chapter is organized as follows. In Section 2, we show the basic formulation of LIM and demonstrated the linear computational complexity *O(n)* of the scheme. Next, stability analysis is performed in Section 3. This is followed by Section 4, where LIM formulations are successfully developed for simple/general mutual inductors and the branch capacitor. In Section 5, the nonlinear simple bipolar junction transistor (BJT) model is demonstrated. In Section 6, the frequency dependent model such as skin-effect is demonstrated. Finally, the possibility of unconditionally stable LIM scheme is discussed in Section 7.

3.2 Basic Formulation

Distributed networks are often used to describe signal propagation on uniform transmission lines (Figure 3.1). This model is also a high-frequency representation of an interconnection. Figure 3.2 shows a more general interconnection topology in which signals can propagate in more than one direction. Such a model can be viewed as the high-frequency representation of an arbitrary network. In analyzing an arbitrary network, we can define a branch as a connection between two nodes (excluding the ground reference node). In defining the desired topology for such an analysis, the following requirements are made:

- Each branch of the network must contain an inductance; otherwise a small inductance is inserted into the branch to generate the latency.
- Each node of the network must provide a capacitive path to ground; otherwise, a small shunt capacitor is added to generate latency at that node.

In addition, it is also assumed that by using combinations of Thévenin and Norton transformations, all branches and nodes can be converted to this topology. After all augmentations and reductions are performed, network branches are represented with a voltage

source, a resistor, and an inductor in series. All connections to ground are represented by a parallel combination of a current source, a capacitance and a conductance to ground from every node.

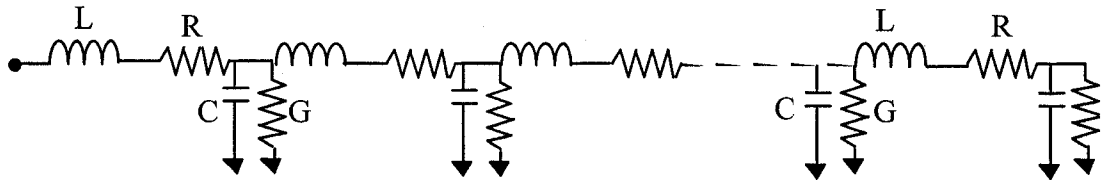


Figure 3.1 Discrete distributed model for uniform transmission line. *R*, *L*, *G*, and *C* are the resistance, inductance, conductance, and capacitance per unit length, respectively

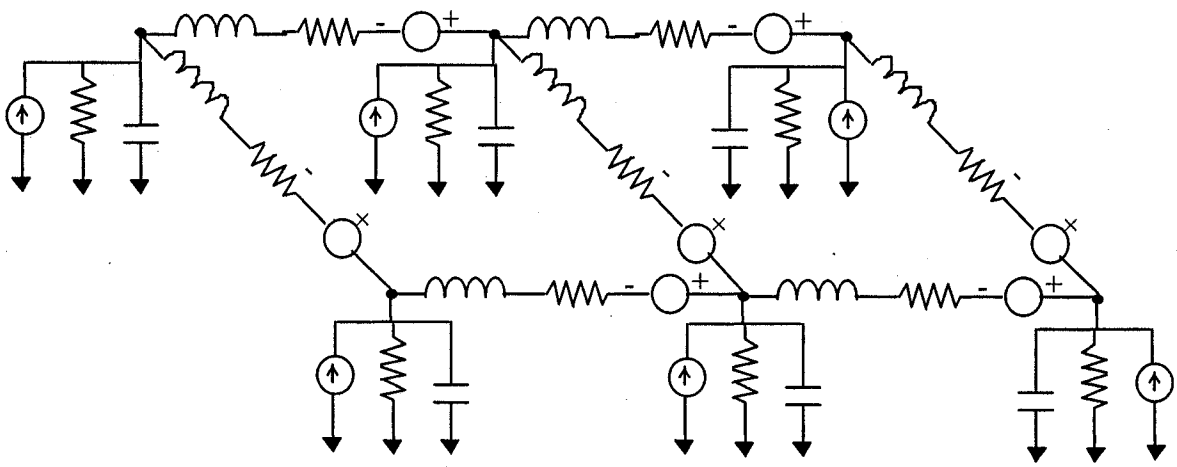


Figure 3.2 Network with interconnect topology

First, the time variable is discretized, next the voltage and current quantities are collocated in half time steps to generate sequences of the form $V^{n-1/2}$, $V^{n+1/2}$, and $V^{n+3/2}$ for voltages and I^{n}, I^{n+1}, and I^{n+2} for currents.

3.2.1 Branch algorithm

Each branch is represented as a combination of a voltage source, an inductor and a resistor in series. Current I_{ij} is assumed to be directed from node i at voltage V_i to node j at potential V_j (see Figure 3.3).

The discrete time equation reads

$$V_i^{n+1/2} - V_j^{n+1/2} = L_{ij}\left(\frac{I_{ij}^{n+1} - I_{ij}^{n}}{\Delta t}\right) + R_{ij} I_{ij}^{n} - E_{ij}^{n+1/2} \tag{3.1}$$

Solving for the unknown current leads to

$$I_{ij}^{n+1} = I_{ij}^{n} + \frac{\Delta t}{L_{ij}}\left(V_i^{n+1/2} - V_j^{n+1/2} - R_{ij} I_{ij}^{n} + E_{ij}^{n+1/2}\right) \tag{3.2}$$

At each time step, this operation is performed over all N_b branches of the network in order to update all the current values.

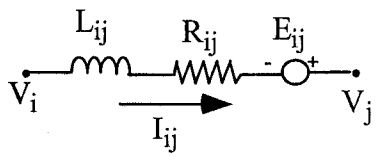


Figure 3.3 Branch equation formulation

3.2.2 Node algorithm

Each node is modeled as a parallel combination of a current source, a conductance and a capacitor to ground as shown in Figure 3.4. The equation reads:

$$C_i\left(\frac{V_i^{n+1/2} - V_i^{n-1/2}}{\Delta t}\right) + G_i V_i^{n+1/2} - H_i^{n} = -\sum_{k=1}^{M_i} I_{ik}^{n}\,, \tag{3.3}$$

where M_i is the number of branches connected to node i (excluding connections to ground). This yields

$$V_i^{n+1/2} = \frac{\dfrac{C_i V_i^{n-1/2}}{\Delta t} + H_i^n - \sum_{k=1}^{M_i} I_{ik}^n}{\dfrac{C_i}{\Delta t} + G_i}, \tag{3.4}$$

for $i = 1, 2, \ldots N_n$. At each time step, this operation is performed over all N_n nodes in order to update all the voltage quantities.

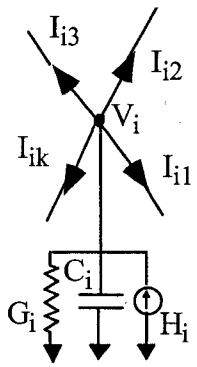


Figure 3.4 Node equation formulation

Computations of all branch currents then all node voltages are alternated as time progresses; thus a complete network simulation can be summarized by the following algorithm:

```
For time=1, Nt
    For branch=1, Nb
        Update current as per Equation (2);
    Next branch;
    For node=1, Nn
        Update voltage as per Equation (4);
    Next node;
Next time;
```

Altogether, $N_t(N_n+N_b)$ operations are performed to obtain $N_t(N_n+N_b)$ values yielding hence an optimally efficient algorithm.

3.2.3 System of equations

After reviewing the basic formulation of LIM, the systems of equations for both linear and nonlinear systems are shown below.

The linear elements in a branch linked to a node (Figure 3.5) are discretized according to the following relations:

$$L_{ij}\left(\frac{I_{ij}^{n+1}-I_{ij}^{n}}{\Delta t}\right)=V_i^{n+1/2}-V_j^{n+1/2}-R_{ij}I_{ij}^{n}+E_{ij}^{n+1/2} \tag{3.5}$$

$$C_i\left(\frac{V_i^{n+1/2}-V_i^{n-1/2}}{\Delta t}\right)=-\sum_{k=1}^{M_i}I_{ij}^{n}-G_iV_i^{n+1/2}+H_i^{n+1/2} \tag{3.6}$$

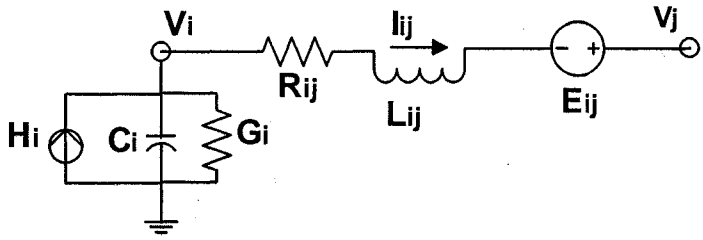


Figure 3.5 Linear branch and node

The nonlinear elements of a branch linked to a node (Figure 3.6) are discretized as shown below:

$$L_{ij}\left(\frac{I_{ij}^{n+1}-I_{ij}^{n}}{\Delta t}\right)=V_i^{n+1/2}-V_j^{n+1/2}-R_{ij}I_{ij}^{n+1}+E_{ij}^{n+1/2}-f^{-1}\left(I_{ij}^{n+1}\right) \tag{3.7}$$

$$C_i\left(\frac{V_i^{n+1/2}-V_i^{n-1/2}}{\Delta t}\right)=-\sum_{k=1}^{M_i}I_{ij}^{n}-G_iV_i^{n+1/2}+H_i^{n+1/2}-f(V_i^{n+1/2}) \tag{3.8}$$

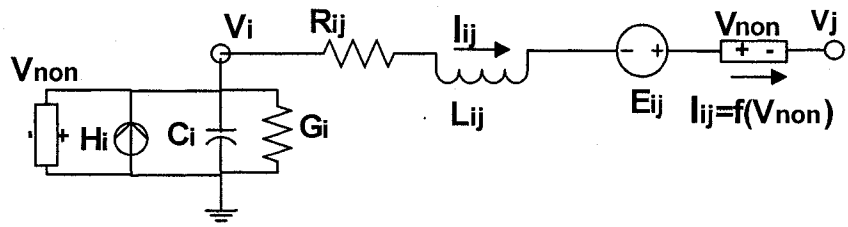


Figure 3.6 Nonlinear branch and node

3.3 Stability Analysis of LIM

In order to analyze the stability of these solutions, various difference approaches need to be compared. In [30], the stability condition is given for the lossless case, where R_{ij} and G_i are both zero and correspond to the same condition obtained for FDTD:

$$\Delta t \leq \sqrt{L_{ij}C_i} \tag{3.9}$$

In the linear case, [30] uses the forward and backward Euler schemes. When nonlinear elements are present, the backward Euler scheme is used. Without loss of generality, the independent sources can be ignored in the stability analysis. We first assume a homogeneously loaded grid with identical *R, L, G,* and *C* values at all branches and nodes. In addition, we assume that only two branches are connected to each node, which corresponds to a unidirectional network, labeled as x. This places the LIM algorithm in the same difference scheme category as the one-dimensional FDTD scheme used in [35]. The inhomogeneous case will be discussed later. The homogeneous case is shown in Figure 3.7. The corresponding simplified differential equations for linear branch and node are respectively given by,

$$L\frac{dI}{dt} = \frac{dV}{dx} - RI \tag{3.10}$$

$$C\frac{dV}{dt} = \frac{dI}{dx} - GV \tag{3.11}$$

For the circuit representing transmission lines, the *R*, *L*, *G*, and *C* values are p.u.l. values. Next, from (3.10)-(3.11), we use the stability criterion based on the Von Neumann method for FDTD [33],[35] to obtain similar stability constraints for several difference formulations. Further details about the Von Neumann method can be found in [34],[36].

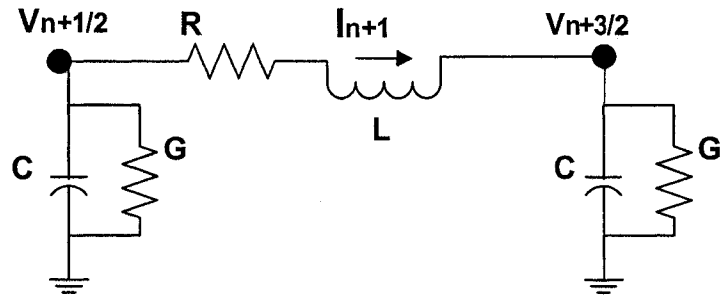


Figure 3.7 Simplified homogeneous branch and node for stability analysis

LIM uses a central difference formulation for *dV/dt* and *dI/dt* according to (3.5)-(3.6). However, the resistive drop terms, *RI* and *GV* can take an explicit, semi-implicit, or fully implicit form. The three cases will be discussed later.

If the current in the branches are in *x*-direction, the discretized forms of (3.10) and (3.11) are given by

$$L\frac{I_k^{n+1}-I_k^n}{\Delta t}=\frac{V_{k+1/2}^{n+1/2}-V_{k-1/2}^{n+1/2}}{\Delta x}-RI \tag{3.12}$$

$$C\frac{V_{k+1/2}^{n+1/2}-V_{k+1/2}^{n-1/2}}{\Delta t}=\frac{I_{k+1}^{n}-I_k^{n}}{\Delta x}-GV \tag{3.13}$$

The central difference formulation means that the time grids for the branch current and node voltage are interleaved. For example, $V(t = n+1/2)$ is between $I(t = n)$ and $I(t = n+1)$. The resistance/conductance term can have several elements. For linear device the resistance/conductance term can be expressed as a linear combination of the current/voltage at the previous time steps. For nonlinear device, it should be linearized at each time step.

$$RI \triangleq \frac{L}{\Delta t}(c_{-1}I^{n+1}+c_0I^n+c_1I^{n-1}+\cdots+c_nI^0) \tag{3.14}$$

$$GV \triangleq \frac{C}{\Delta t}(d_{-1}V^{n+\frac{1}{2}}+d_0V^{n-\frac{1}{2}}+d_1V^{n-\frac{3}{2}}+\cdots+d_{n-1}V^{\frac{1}{2}}) \tag{3.15}$$

Here c_i and d_i, $i=-1,0,1,\cdots,n$ are coefficients representing the linearized current/voltage variables of any nonlinear devices. These coefficients depend on the scheme used and will be discussed in the following subsections.

Substituting (3.14)-(3.15) into (3.12)-(3.13), the difference equations are shown in (3.16)-(3.17):

$$\frac{C}{\Delta t}V_{k+1/2}^{n+1/2}=\frac{C}{\Delta t}V_{k+1/2}^{n-1/2}+\frac{I_{k+1}^{n}}{\Delta x}-\frac{I_{k}^{n}}{\Delta x} -\frac{C}{\Delta t}(d_{-1}V_{k+1/2}^{n+1/2}+d_0V_{k+1/2}^{n-1/2}+d_1V_{k+1/2}^{n-3/2}+\cdots+d_{n-1}V_{k+1/2}^{1/2}) \tag{3.16}$$

$$-\frac{V_{k+1/2}^{n+1/2}}{\Delta x}+\frac{V_{k-1/2}^{n+1/2}}{\Delta x}+\frac{L}{\Delta t}I_{k}^{n+1}=\frac{L}{\Delta t}I_{k}^{n}-\frac{L}{\Delta t}(c_{-1}I_{k}^{n+1}+c_0I_{k}^{n}+c_1I_{k}^{n-1}+\cdots+c_nI_{k}^{0}) \tag{3.17}$$

To perform a stability analysis of the above system, the Von Neumann method used in [33],[35] is applied. A new vector $\boldsymbol{u}_k^n$ is introduced to transform the system into a two-level scheme. Each component of $\boldsymbol{u}_k^n$ is considered as the discrete Fourier transform of the time domain signal. For example,

$$V_{k+1/2}^{n+1/2}=\frac{1}{\sqrt{2\pi}}\int_{-\pi}^{\pi}e^{im\zeta}V_{k+1/2}^{n+1/2}(\zeta)d\zeta \quad I_{k+1}^{n+1}=\frac{1}{\sqrt{2\pi}}\int_{-\pi}^{\pi}e^{im\zeta}I_{k+1}^{n+1}(\zeta)d\zeta, \quad \zeta\in[-\pi,\pi] \tag{3.18}$$

$$\boldsymbol{u}_k^{n+1}=[V_{k+1/2}^{n+1/2} \quad I_k^{n+1} \quad \cdots \quad I_k^1 \quad V_{k+1/2}^{n-1/2} \quad \cdots \quad V_{k+1/2}^{3/2}]^T \tag{3.19}$$

$$\boldsymbol{U}_k^n(\zeta)\triangleq\frac{1}{\sqrt{2\pi}}\sum_{k=-\infty}^{\infty}e^{-ik\zeta}\boldsymbol{u}_k^n \tag{3.20}$$

$$\boldsymbol{U}_{k\pm1}^n(\zeta)=e^{\pm ik\zeta}\boldsymbol{U}_k^n(\zeta) \tag{3.21}$$

Using (3.18)-(3.21), (3.16)-(3.17) can be transformed into the Fourier domain as follows:

$$\boldsymbol{A}\cdot\boldsymbol{U}_k^{n+1}=\boldsymbol{B}\cdot\boldsymbol{U}_k^n \tag{3.22}$$

$$\boldsymbol{A}=\begin{pmatrix} (1+d_{-1})C/\Delta t & 0 & 0 & 0 & \cdots & 0 \\ \dfrac{e^{-i\zeta}-1}{\Delta x} & (1+c_{-1})L/\Delta t & 0 & 0 & \cdots & 0 \\ O_{(2n-1)\times 2} & & I_{(2n-1)\times(2n-1)} & & & \end{pmatrix} \tag{3.23}$$

Here, $O_{(2n-1)\times 2}$ represents the zero matrix of $(2n-1)\times 2$ and $I_{(2n-1)\times(2n-1)}$ is an identity matrix.

$$\boldsymbol{B}=\begin{pmatrix} \dfrac{C(1-d_0)}{\Delta t} & \dfrac{e^{i\zeta}-1}{\Delta x} & 0 & \cdots & 0 & \dfrac{-Cd_1}{\Delta t} & \dfrac{-Cd_2}{\Delta t} & \cdots & \dfrac{-Cd_{n-1}}{\Delta t} \\ 0 & \dfrac{L(1-c_0)}{\Delta t} & \dfrac{-Lc_1}{\Delta t} & \cdots & \dfrac{-Lc_n}{\Delta t} & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & & & & & & \\ \vdots & 0 & 1 & \ddots & & & & & \\ \vdots & \vdots & & \ddots & 0 & & & & \\ \vdots & \vdots & & & 1 & 0 & & & \\ \vdots & \vdots & & & & 1 & \ddots & & \\ \vdots & \vdots & & & & & \ddots & \ddots & \\ 0 & 0 & & & & & & 1 & 0 \end{pmatrix} \tag{3.24}$$

In order to insure stability, the magnitudes of all eigenvalues of the amplification matrix $A^{-1}B$ for the difference scheme have to be less than one for $\zeta \in [-\pi,\pi]$. This is investigated through the characteristic polynomial of $A^{-1}B$. For linear resistance and conductance, first- or second-order difference scheme will normally lead to only six nonzero terms of *RI* and *GV*: c_{-1},c_0,c_1 and d_{-1},d_0,d_1. For example, for forward Euler difference, $RI=RI^n$, $GV=GV^{n-1/2}$, which generates only two nonzero parameters: c_0,d_0. Then the characteristic polynomial of $A^{-1}B$ is

$$
\begin{aligned}
&P(\lambda) = \\
&\lambda^4 - \lambda^3[\frac{1-c_0}{1+c_{-1}} + \frac{1-d_0}{1+d_{-1}} - \frac{4\gamma^2 \sin^2(\zeta/2)}{(1+c_{-1})(1+d_{-1})}] \\
&+\lambda^2[\frac{c_1}{1+c_{-1}} + \frac{(1-c_0)(1-d_0)}{(1+c_{-1})(1+d_{-1})}] \\
&+\lambda[\frac{\gamma d_1(1-e^{-i\zeta})\sqrt{C/L}}{(1+c_{-1})(1+d_{-1})} - \frac{c_1(1-d_0)}{(1+c_{-1})(1+d_{-1})}]
\end{aligned}
\tag{3.25}
$$

with the Courant number $\gamma = \Delta t/(\Delta x\sqrt{LC})$, $(\Delta x = 1)$.

3.3.1 Explicit (forward Euler)

In the explicit case, the resistive voltage drop term RI is given by RI^n and the conductive current term GV is given by GV^n. This leads to $c_0 = \frac{R\Delta t}{L}$ $d_0 = \frac{G\Delta t}{C}$, $d_i \,\&\, c_i = 0,\ i \neq 0$. This is a first-order accurate scheme. Hence, the Courant number γ satisfies (see appendix for details) the following:

$$
\gamma \leq \frac{1}{2}\sqrt{4 - 2(\frac{R\Delta t}{L} + \frac{G\Delta t}{C}) + \frac{RG\Delta t^2}{LC}} \tag{3.26}
$$

According to $\gamma = \Delta t/(\Delta x\sqrt{LC})$, $(\Delta x = 1)$, the above inequality (3.26) is used to solve for the time step leading to the following stability condition:

$$
\begin{aligned}
&\Delta t \leq K_1\sqrt{LC} \\
&K_1 = \frac{\left(R\sqrt{\frac{C}{L}} + G\sqrt{\frac{L}{C}}\right) - \sqrt{\left(R\sqrt{\frac{C}{L}} - G\sqrt{\frac{L}{C}}\right)^2 + 16}}{RG - 4}
\end{aligned}
\tag{3.27}
$$

If $RG > 8$, $K_1 \geq 1$; the time step is larger than that of the unloaded FDTD case. For other cases, K_1 can be any value. A special condition is when $G = 0$, the stability condition becomes

$$
\gamma \leq \frac{1}{2}\sqrt{4 - 2\frac{R\Delta t}{L}} \tag{3.28}
$$

$$\Delta t \le \sqrt{\left(\frac{RC}{4}\right)^2 + LC} - \frac{RC}{4} = \sqrt{LC}\left[\sqrt{\left(\frac{R}{4}\sqrt{\frac{C}{L}}\right)^2 + 1} - \left(\frac{R}{4}\sqrt{\frac{C}{L}}\right)\right] \tag{3.29}$$

When R is increased, the time step is decreased.

3.3.2 Semi-implicit (leap frog)

In the semi-implicit case, the resistive voltage drop term RI is expressed by $R(I^n + I^{n+1})/2$ and the conductive current term GV is given by $G(V^{n+1/2} + V^{n-1/2})/2$. This is a second-order accurate scheme. We obtain $c_{-1} = c_0 = \frac{R\Delta t}{2L}$ $d_{-1} = d_0 = \frac{G\Delta t}{2C}$, $d_i \,\&\, c_i = 0, i \neq 0, -1$. Hence, from (3.25), the Courant number γ satisfies the condition:

$$\gamma \le \frac{1}{2}\sqrt{\frac{ac + bd + abcd + 1}{cd}}$$
$$a \triangleq 1 - c_0,\ b \triangleq 1 - d_0,\ c \triangleq \frac{1}{1 + c_{-1}},\ d \triangleq \frac{1}{1 + d_{-1}} \tag{3.30}$$

There are three cases, where the inequality in (3.30) can be satisfied:

(1) a and $b \in (0,1]$

(2) $ab < 0$ and $|b|d < 1$

(3) $a \,\&\, b < 0$ and $abcd < 1$ and $\begin{Bmatrix} |a|c > 1 \,\&\, |b|d < 1 \\ |a|c < 1 \,\&\, |b|d > 1 \end{Bmatrix}$

Then the time step Δt is limited by

$$\Delta t \le K_2 \sqrt{LC}$$
$$K_2 = \frac{1}{2}\sqrt{(1-c_0)(2+d_{-1}+d_0)+(1+c_{-1})(2+d_{-1}-d_0)} \tag{3.31}$$

For a special case where $c_{-1} = c_0$, d_{-1} & $d_0 = 0$, the stability condition is

$$\Delta t \le \sqrt{LC} \tag{3.32}$$

This is the same criterion as in the lossless case. The time step Δt is not related to the value of the resistor.

3.3.3 Fully implicit (backward Euler)

In the fully implicit case, the resistive voltage drop term RI is given by RI^{n+1} and the conductive current term GV is given by $GV^{n+1/2}$. This yields $c_{-1} = \frac{R\Delta t}{L}$ $d_{-1} = \frac{G\Delta t}{C}$, d_i & $c_i = 0$, $i \neq -1$. This has a first-order accuracy with Courant number γ satisfying the following conditions:

$$\gamma \le \frac{1}{2}\sqrt{4+2(c_{-1}+d_{-1})+c_{-1}d_{-1}} \tag{3.33}$$

$$\Delta t \le K_3\sqrt{LC}, \quad K_3 = \frac{1}{2}\sqrt{4+2(c_{-1}+d_{-1})+c_{-1}d_{-1}} \tag{3.34}$$

For the case $c_{-1} \neq 0$, $d_{-1} = 0$, the stability condition becomes

$$\gamma \le \frac{1}{2}\sqrt{4+2\frac{R\Delta t}{L}} \tag{3.35}$$

$$\Delta t \le \frac{RC}{4}+\sqrt{\left(\frac{RC}{4}\right)^2+LC} = \sqrt{LC}\left[\left(\frac{R}{4}\sqrt{\frac{C}{L}}\right)+\sqrt{\left(\frac{R}{4}\sqrt{\frac{C}{L}}\right)^2+1}\right] \tag{3.36}$$

This stability limit is larger than that of the lossless case, because the factor $\sqrt{M^2+1}+M$, $(M=\sqrt{C/L}\cdot R/4)$ is always greater than one.

A similar result can be derived for the case where $G\neq 0$ and $R=0$. This stability condition shows that the backward Euler case could have larger time step but lower accuracy. Thus, for more accurate results, the semi-implicit scheme is the most appropriate.

At each time step, the nonlinear function can be linearized; in fact, the stability analysis becomes more complex due to the time dependent parameters. For a branch diode, the equivalent local resistance $R_{eq}=V_T/(I+I_S)$ is used in the above stability analysis.

For inhomogeneous cases, where the L and C vary throughout network while R and G remain constant, the smallest LC product obtained at any given node determines the stability of the system.

For inhomogeneous cases, where the R, G, L, and C are all different over the overall domain, a stability criterion cannot be formulated analytically. But for linear system, the stability condition can be limited by the smallest product of L and C connected to the same node for fully implicit difference scheme.

For multidimensional case, in which n pairs of branches are connected to a node i, the L_{ik} and C_i are the inductance of the kth branch and capacitance of node i, respectively. Then the stability condition becomes

$$\Delta t \le P_{ik}\cdot \min_{i,k}\{\sqrt{L_{ik}C_i}\}/\sqrt{n}\ ,\quad k=1,2,\cdots 2n \tag{3.37}$$

Here P_{ik} represents the coefficient from a given difference scheme (i.e., $P_{ik}=\sqrt{M^2+1}+M$ for backward Euler). Thus for an inhomogeneous medium, the stability condition will vary from one location to another, the overall stability condition is determined by

the smallest LC product which can limit the computational speed. Automatic partitioning algorithms may be used to subdivide the entire network into smaller blocks with different time step requirements.

3.4 Mutual Inductor and Branch Capacitor Modeling

3.4.1 Mutual inductance formulation (forward Euler)

The mutual inductor formulation used in [30] is an approximation. A more accurate formulation with branch resistance is derived below:

$$\begin{cases} V_{1ij} = L_1 \dfrac{dI_1}{dt} + M_{12} \dfrac{dI_2}{dt} + R_1 I_1 \\ V_{2ij} = L_2 \dfrac{dI_2}{dt} + M_{12} \dfrac{dI_1}{dt} + R_2 I_2 \end{cases} \tag{3.38}$$

Equations (3.38) are discretized as per the forward Euler method to give the following updating equations

$$\begin{cases} I_1^{n+1} = I_1^n + \dfrac{1}{L_1 L_2 - M_{12}^2}\left(M_{12} R_2 \Delta t I_2^n - L_2 R_1 \Delta t I_1^n + L_2 \Delta t V_1^{n+1/2} - M_{12} \Delta t V_2^{n+1/2}\right) \\ I_2^{n+1} = I_2^n + \dfrac{1}{L_1 L_2 - M_{12}^2}\left(M_{12} R_1 \Delta t I_1^n - L_1 R_2 \Delta t I_2^n - M_{12} \Delta t V_1^{n+1/2} + L_1 \Delta t V_2^{n+1/2}\right) \end{cases} \tag{3.39}$$

The model is shown in Figure 3.8. This model is difficult to expand to arbitrary number of coupling inductors. So another expandable form is proposed in the next section.

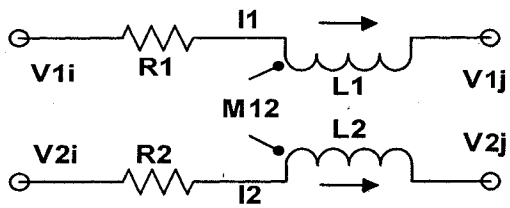


Figure 3.8 Mutual inductors (forward Euler)

3.4.2 Mutual inductor model (K element)

In general when more than two mutual inductors are present, mutual inductors are handled by using L matrix element, which cannot be discretized to an arbitrary L matrix. This means that the previous method cannot be extended. A new approach is developed to handle the general coupling of mutual inductors using the K element, which was introduced in [37]. K is the inverse of the partial inductance matrix L; it has a higher degree of locality as does the capacitance matrix. Furthermore, [37] shows that the K matrix is diagonally dominant and hence positive definite. The off-diagonal terms are negative and can be removed without sacrificing the stability of the modified K matrix. From the improved locality property of the K matrix, mutual inductors can be handled by inverting the L matrix into a K matrix if K is not directly available.

From the definition of $K = inv(L)$, we have the general form of the inductance inductance equations given by: For mutual inductor modeling, semi-implicit scheme is used for the case without resistors. Combined explicit and semi-implicit scheme is used for the case with resistors.

$$\begin{cases} V_{1ij} = L_{11}\dfrac{dI_1}{dt} + L_{12}\dfrac{dI_2}{dt} + \cdots + L_{1N}\dfrac{dI_N}{dt} \\ V_{2ij} = L_{21}\dfrac{dI_1}{dt} + L_{22}\dfrac{dI_2}{dt} + \cdots + L_{2N}\dfrac{dI_N}{dt} \\ \quad\vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\qquad \vdots \\ V_{Nij} = L_{N1}\dfrac{dI_1}{dt} + L_{N2}\dfrac{dI_2}{dt} + \cdots + L_{NN}\dfrac{dI_N}{dt} \end{cases} \tag{3.40}$$

$$\Rightarrow \begin{cases} \dfrac{dI_1}{dt} = K_{11}V_{1ij} + K_{12}V_{2ij} + \cdots + K_{1N}V_{Nij} \\ \dfrac{dI_2}{dt} = K_{21}V_{1ij} + K_{22}V_{2ij} + \cdots + K_{2N}V_{Nij} \\ \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ \dfrac{dI_N}{dt} = K_{N1}V_{1ij} + K_{N2}V_{2ij} + \cdots + K_{NN}V_{Nij} \end{cases} \tag{3.41}$$

From (3.41), the simple branch update equation for general inductance coupling is deduced as shown below:

$$I_j^{n+1} = I_j^n + \Delta t \sum_{m=1}^{N} K_{jm} V_{mij}^{n+1/2} \tag{3.42}$$

which is the updating equation for branch j.

For branches with resistor and voltage source, the update equation of ith branch becomes:

$$\begin{aligned} I_i^{n+1} = \; & I_i^n \frac{2 - dtrk}{2 + dtrk} \\ & + \frac{\Delta t}{1 + dtrk/2} \left(\sum_{m=1}^{N} K_{im} V_{mij}^{n+1/2} \right) \\ & - \Delta t \left(\sum_{m=1}^{N} K_{1m} R_m I_m^n \right) \end{aligned} \tag{3.43}$$

here, $dtrk = \Delta t K_{ii} R_i$, R_m is the resistance of the mth branch. The numerical validation of this formulation is found in Chapter 5.

3.4.3 Branch capacitor formulation

When a capacitor is present within a branch, the standard LIM formulation must be modified. For this purpose, a small inductor is added in series with the branch capacitor between nodes i and j, as shown in Figure 3.9.

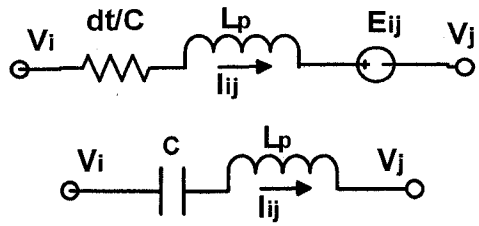


Figure 3.9. Branch capacitor model: the top diagram is standard approach; the bottom diagram is the proposed scheme

The standard approach consists of representing the capacitor as a resistor of value *dt/C*, which is derived from companion model of capacitor. A linear branch update equation can then be used as in (3.1).

$$I_{ij} = C\frac{dV_c}{dt} \tag{3.44}$$

where V_C and V_L are the voltages across the capacitor and inductor, respectively. With the presence of the inductor L_P, the updated voltage difference V_i-V_j does not equal V_C. This limitation can be removed by using an efficient direct finite difference formulation for branch current and voltage as follows:

$$\begin{aligned} & I_{ij} = C\frac{dV_c}{dt} \text{ and } V_L = L_p\frac{dI_{ij}}{dt} \\ & \Rightarrow \begin{cases} V_c^{n+1/2} = \sum_{k=0}^{n}\left(\frac{I_{ij}^k \Delta t}{C}\right) \equiv K_{ij}^n \\ V_L^{n+1/2} = L_p\frac{I_{ij}^{n+1} - I_{ij}^n}{\Delta t}, \; K_{ij}^n = K_{ij}^n + \frac{I_{ij}^n \Delta t}{C} \end{cases} \\ & \Rightarrow \left\{ I_{ij}^{n+1} = I_{ij}^n + \frac{\Delta t}{L_p}\left(V_{ij}^{n+1/2} - K_{ij}^n\right) \right. \end{aligned} \tag{3.45}$$

3.5 Bipolar Junction Transistor Model

The BJT model used in LIM is based on the Ebers-Moll representation [38],[39]. It is the basic model used in SPICE. The complete large signal model of the BJT used in SPICE also includes nonlinear capacitances and three terminal resistances used to simulate dynamic operation as well as the resistances for the three terminal regions. The Ebers-Moll model consists of two diodes and two current-controlled current sources (CCCS). Thus in general, the simple static transistor model has four branches, when none of the terminals is connected to ground. This model is shown in Figure 3.10. When one of the terminals is connected to ground, the LIM model will have either two or zero branches. If the emitter (E) is connected to ground, the branch from B to E becomes grounded. It leads to only two branches. If base (B) is connected to ground, there will be no branch.

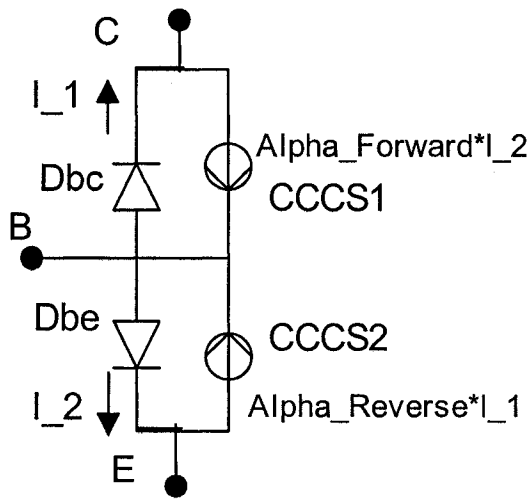


Figure 3.10 Simple BJT transistor model for LIM

In this work, the Ebers-Moll model is implemented to verify the feasibility of this approach. For each branch a small resistor and small inductor are introduced in series as required by LIM. The updating sequence for the branch current and node voltage must be

respected; due to the presence of the CCCS, the reference branch current must be updated first in order to ensure correct result.

The updating procedure for the simple BJT model in LIM is shown below:

```
For  time=1, Nt {
      For branch#=1, Nb {
       If (branch_type == linear)
           Update current as linear formulation;
       Else if (branch_type == diode)
           Update current as nonlinear diode;
       Else if (branch_type == voltage/current controlled current source)
           Update current from the reference node/branch;
       /* for BJT transistor model */
       /* First:     update branch#1 for diode BC
          Second: update branch#2 for diode BE
          Third:    update branch#3 for CCCS BC
          Fourth:  update branch#4 for CCCS BE
           */
      Next branch;
     }
    For node#=1, Nn {
     If (node_type == linear)
         Update voltage as linear formulation;
     Else if (node_type == diode)
         Update voltage as nonlinear diode;
     Else if (node_type == voltage/current controlled voltage source)
         Update voltage from the reference node/branch;
    Next node;
    }
Next time;
}
```

It must be noted that for step 2, a user specified value determines the size of the inserted inductance and capacitance values. To minimize the impact of the inserted component, the inserted inductance and capacitance is limited to the one tenth of the smallest inductance/capacitance.

Using the above integration scheme, LIM can be invoked as a new command within SPICE and simulate the high-frequency transient response with speeds far superior than the standard SPICE.

The above LIM-SPICE integration enables an efficient simulation method for large networks. The detailed implementation is skipped here to save space.

3.6 Frequency-Dependent Model

The frequency-dependent behavior of circuit element is a critical parameter in the simulation of integrated circuits. With higher processing speeds, digital signals have spectrum components that easily extend into the gigahertz range. The simulation of such circuits can be prohibitive and often requires several days of CPU time on a traditional workstation. In order to handle the skin effect of the conductors, a simple formulation is derived here and verified by the measurement result in Chapter 5.

The generalized branch and nodes LIM equations showing space dependence can be written in the S domain as below:

$$Z(s)I = -\frac{\partial V}{\partial x}, \quad Z(s) = R(s) + sL(s) \tag{3.46}$$

$$(sC + G)V = \frac{\partial I}{\partial x} \tag{3.47}$$

In this paper, the frequency dependent effects of dielectric loss G and capacitance C are ignored. It means that C and G are independent of frequency. When the frequency dependent losses of the conductor are taken into account, the resistance R and the inductance L vary with frequency. This dependence can be analytically described using various methods. In particular, a rational function expansion can be used to model the frequency-dependent of conductors as can be observed in [40], [41]. This expansion could be used to model the frequency dependent C and G.

$$Z(s) = R_0 + sL_e + \sum_{q=1}^{Q} \frac{R_q}{s - P_q} \tag{3.48}$$

For modeling skin effect, the above formulation is replaced by a simpler and more efficient approximation used in [42] in which the frequency dependences of the resistance R and inductance L are shown as in (3.49)-(3.52).

$$Z(s) = R(s) + L(s) \tag{3.49}$$

$$R(s) \triangleq R_0 + R_s\sqrt{s}\,, \quad L(s) \triangleq L_i(s) + L_e \tag{3.50}$$

In the above equations, R_0 is the constant dc resistance per unit length (p.u.l.); L_e is the constant external inductance p.u.l. and L_i is the internal inductance p.u.l., which decrease with $\sqrt{s}$. The skin effect can then be represented by $Z_i(s)$, which is defined as in (3.51)-(3.52).

$$Z(s) = Z_i(s) + sL_e \tag{3.51}$$

$$Z_i(s) \triangleq R(s) + sL_i(s) \tag{3.52}$$

Then the frequency-domain representation is given by

$$Z_i(s) = R_0 + R_s(s) + sL_i(s) \tag{3.53}$$

Since $L_i \propto 1/\sqrt{s}$, the frequency dependent impedance can be expressed as in [42]

$$Z_i(s) = A + B\sqrt{s} \tag{3.54}$$

$$A = R_0\,, \quad R_s(f) = B\sqrt{\pi}\sqrt{f}\,, \quad L_i(f) = B/2\sqrt{\pi f} \tag{3.55}$$

Equations (3.46) and (3.47) are transformed to yield

$$sL_eI(s) + Z_i(s)I(s) = -\frac{\partial V(s)}{\partial x} \tag{3.56}$$

$$sCV(s) + GV(s) = \frac{\partial I(s)}{\partial x} \tag{3.57}$$

The term $Z_i(s)I(s)$ translates to a convolution in the time domain because $1/\sqrt{s} \leftrightarrow \frac{1}{\sqrt{\pi}\sqrt{t}}$.

$$Z_i(s)I(s) = AI(s) + \frac{B}{\sqrt{s}} sI(s) \tag{3.58}$$

$$\Rightarrow Z_i(t) * I(t) = AI(t) + \frac{B}{\sqrt{\pi}} \int_0^t \frac{1}{\sqrt{\tau}} \frac{\partial I(t-\tau)}{\partial(t-\tau)} d\tau \tag{3.59}$$

The integral in (3.59) can be evaluated using piecewise constant convolution techniques such as in [43], which have second-order accuracy:

$$Z_i(t) * I(t)\big|_{t=n\Delta t} \cong AI^n + \frac{B\sqrt{\Delta t}}{\sqrt{\pi}} \left[\left.\frac{\partial I}{\partial t}\right|_{t=n} \int_0^{1/2} \frac{1}{\sqrt{\tau}} d\tau + \sum_{m=0}^{n-1} \left.\frac{\partial I}{\partial t}\right|_{t=n-1-m} \int_{(m+1/2)}^{(m+3/2)} \frac{1}{\sqrt{\tau}} d\tau \right] \tag{3.60}$$

After substitution of (3.60) into (3.56) and discretization, the branch current update equation takes the form

$$\begin{aligned} I_{k+1/2}^{n+1/2} = & \left(\frac{L_e}{\Delta t} + A/2 + \frac{B}{\sqrt{\pi \Delta t}} \Psi_0 \right)^{-1} \\ & \left[\left(\frac{L_e}{\Delta t} - A/2 + \frac{B}{\sqrt{\pi \Delta t}} \Psi_0 \right) I_{k+1/2}^{n-1/2} + \frac{1}{\Delta x} \left(V_{k+1}^n - V_k^n \right) - \frac{B}{\sqrt{\pi \Delta t}} \Psi_{k+1/2}^n \right] \end{aligned} \tag{3.61}$$

where the variables Ψ_0 and $\Psi_{k+1/2}^n$ are defined as

$$\Psi_0 \triangleq \int_0^{1/2} \frac{1}{\sqrt{\tau}} d\tau, \quad \Psi_{k+1/2}^n \triangleq \sum_{m=0}^{n-1} \left(I_{k+1/2}^{n-1/2-m} - I_{k+1/2}^{n-3/2-m} \right) \int_{(m+1/2)}^{(m+3/2)} \frac{1}{\sqrt{\tau}} d\tau \tag{3.62}$$

We next transform the integral into a recursive convolution form. This is done by approximating with a series of exponentials [40]:

$$Pz(m) \triangleq \int_{(m+1/2)}^{(m+3/2)} \frac{1}{\sqrt{\tau}} d\tau \cong \sum_{i=1}^{P} a_i \exp(b_i m), \quad \Psi_0 = \int_0^{1/2} \frac{1}{\sqrt{\tau}} d\tau \cong \sum_{i=1}^{P} a_i \tag{3.63}$$

By substituting (3.63) into (3.62), we have a recursive convolution updating equation

$$\Phi_{k+1/2,i}^{n+1} = a_i \sum_{m=0}^{n} \left(I_{k+1/2}^{n+1/2-m} - I_{k+1/2}^{n-1/2-m} \right) \exp(b_i m)$$
$$= \Phi_{k+1/2,i}^{n} \exp(b_i) + a_i \left(I_{k+1/2}^{n+1/2} - I_{k+1/2}^{n-1/2} \right), \quad i = 1, \cdots, P \tag{3.64}$$

and $\Phi_{k+1/2,i}^{0} = \Phi_{k+1/2,i}^{1} = 0$ for any i and k.

This process discretizes Equation (3.57), which then provides final LIM update equations implementing the skin effect. The branch currents and node voltages are updated by using

$$I_{k+1/2}^{n+1/2} = \left(\frac{L_e}{\Delta t} + A/2 + \frac{B}{\sqrt{\pi \Delta t}} \Psi_0 \right)^{-1}$$
$$\left[\left(\frac{L_e}{\Delta t} - A/2 + \frac{B}{\sqrt{\pi \Delta t}} \Psi_0 \right) I_{k+1/2}^{n-1/2} + \frac{1}{\Delta x} \left(V_{k+1}^{n} - V_{k}^{n} \right) - \frac{B}{\sqrt{\pi \Delta t}} \sum_{i=1}^{P} \Phi_{k+1/2,i}^{n} \right] \tag{3.65}$$

$$V_k^{n+1} = \left(\frac{C}{\Delta t} + \frac{G}{2} \right)^{-1} \left[\left(\frac{C}{\Delta t} - \frac{G}{2} \right) V_k^n + \frac{1}{\Delta x} \left(I_{k+1/2}^{n+1/2} - I_{k-1/2}^{n+1/2} \right) \right] \tag{3.66}$$

where $\Phi_{k+1/2,i}^{n+1} = \Phi_{k+1/2,i}^{n} \exp(b_i) + a_i \left(I_{k+1/2}^{n+1/2} - I_{k+1/2}^{n-1/2} \right)$, $i = 1, \cdots, P$.

Using the above updating equations, LIM can handle conductors with skin effect efficiently.

3.7 Possible Unconditionally Stable LIM

In order to expedite the simulation of a large circuit network such as power grid or interconnect network, using unconditionally stable difference updating scheme such as alternating-direction implicit (ADI) method to eliminate the stability condition is a preferred technique.

There are two current unconditionally stable simulation methods: the 2-D and 3-D TLM-ADI method. Two-dimensional TLM-ADI has the limitation of a maximum of four branches connected to any node, which limits the range of the network it can simulate. Three-dimensional TLM-ADI can have up to 6 branches. But it has a critical assumption different from the 2-D case: The ratio of *R/L* for every branch must be the same constant in all the branches connected to a node. This assumption ensures the difference equations from KCL and KVL can be transformed into a second-order equation:

$$3lc\frac{\partial^2 v}{\partial t^2}+3rc\frac{\partial v}{\partial t}-(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2})=0 \tag{3.67}$$

The reason behind the above assumption is the basic theorem behind general ADI method or other operator splitting based unconditionally stable method. For a two-level difference equation system given by

$$(I+A)v^{n+1}+Bv^n=g^n \tag{3.68}$$

Equation (3.68) can be transformed into the linear system shown as below:

$$\begin{aligned}
&(I+A_i)\beta_{(1)}^{n+1}+\sum_{j=2}^{q}A_j\beta^n+B\beta^n=g^n,\\
&(I+A_i)\beta_{(i)}^{n+1}-\beta_{(i-1)}^{n+1}-A_i\beta^n=g^n,\quad i=2,\cdots,q,\\
&A=\sum_{i=1}^{q}A_i,\quad \beta^{n+1}=\beta_{(q)}^{n+1}
\end{aligned} \tag{3.69}$$

The new set of equations (3.69) is convergent to the original Equation (3.68) (consistent and stable) if

(1) A_i is positive semi definite, $i = 1, \ldots, q$,

(2) B is Hermitian, and

(3) A_i and B commute.

The operator splitting new scheme (3.69) is then convergent to the original one (3.68). In general q = the spatial dimension.

Then there is a special case for $q = 2$, the third condition of A_i and B commute can be eliminated. This good property ensures the 2-D TLM-ADI can work as it split each time step into two subiterations. However for 3-D TLM-ADI, the critical assumption is essential to satisfy the third condition of the above theorem.

Now we transform the LIM equations into following equations under the assumption that there are six branches connected to a node:

$$C\frac{\partial v}{\partial t} = -(\frac{\partial i_x}{\partial x} + \frac{\partial i_y}{\partial y} + \frac{\partial i_z}{\partial z}) - vG \tag{3.70}$$

$$L_x\frac{\partial i_x}{\partial t} = -\frac{\partial v}{\partial x} - R_x i_x \tag{3.71}$$

$$L_y\frac{\partial i_y}{\partial t} = -\frac{\partial v}{\partial y} - R_y i_y \tag{3.72}$$

$$L_z\frac{\partial i_z}{\partial t} = -\frac{\partial v}{\partial z} - R_z i_z \tag{3.73}$$

Then we got the matrix form of the above equations:

$$\frac{\partial u}{\partial t} = Au \tag{3.74}$$

$$A = \begin{bmatrix} \frac{-G}{C} & \frac{-1}{C}\frac{\partial}{\partial x} & \frac{-1}{C}\frac{\partial}{\partial y} & \frac{-1}{C}\frac{\partial}{\partial z} \\ \frac{-1}{L_x}\frac{\partial}{\partial x} & \frac{-R_x}{L_x} & & \\ \frac{-1}{L_y}\frac{\partial}{\partial x} & & \frac{-R_y}{L_y} & \\ \frac{-1}{L_z}\frac{\partial}{\partial x} & & & \frac{-R_z}{L_z} \end{bmatrix} \tag{3.75}$$

Using the previous theorems, there are two possible ways of achieving the unconditionally stable method. For the 2-D network, the approach can use the similar fractional step method in the 2-D TLM-ADI to gain unconditionally stable method. For the 3-D network, we need to find a way to split the operator A and guarantee the commutative condition. From (3.75), we can see that it is really difficult to decompose A into three submatrix and satisfy all three conditions of the above theorem.

So the tentative conclusion is that we can achieve unconditionally stable LIM for 2-D case and is quite impossible to get such nice result for 3-D case.

4 SPICE-LIM INTEGRATION

4.1 Introduction

SPICE is a well-established platform for circuit simulation. LIM is a fast simulation method for large, high-frequency circuit simulation. The latency insertion nature of the method implies a high-frequency approach to the simulation of large networks with strong parasitic effects. LIM-SPICE integration provides a way to exploit the fast speed of LIM method in the familiar SPICE environment. The objective of the LIM-SPICE integration is to replace/bypass the modified nodal analysis (MNA) formulation within SPICE.

SPICE has data structures, input and output format, that are incompatible with LIM. So the integration requires three major modifications to SPICE: the input (front end), solver, and output. For the input part, a translation module is needed to parse the input netlist and translate the information into the appropriate data structure for LIM simulation. The inserted values are normally one-tenth of the smallest value of the circuit capacitor/inductor. The solver part is the LIM algorithm. The output part is to save the required information into SPICE's raw file data format. The new flow of LIM-SPICE is shown in Figure 4.1.

The dark black boxes represent added/modified sections. More specifically, for integration into SPICE (version is 3f5) the following steps are required:

1. Create a new command "lim" in SPICE.
2. Implement a new parser based on SPICE's own parser to translate the SPICE netlist into suitable LIM circuit topology.
3. Optimize the input circuit and create reduced LIM branches.
4. Perform DC analysis (nonlinear circuit only).

5. Initiate the time loop using the translated LIM input, DC simulation results and the transient simulation parameters.
6. Output the selected node voltage information into SPICE's raw file format, which can be accessed by "nutmeg" command in SPICE.

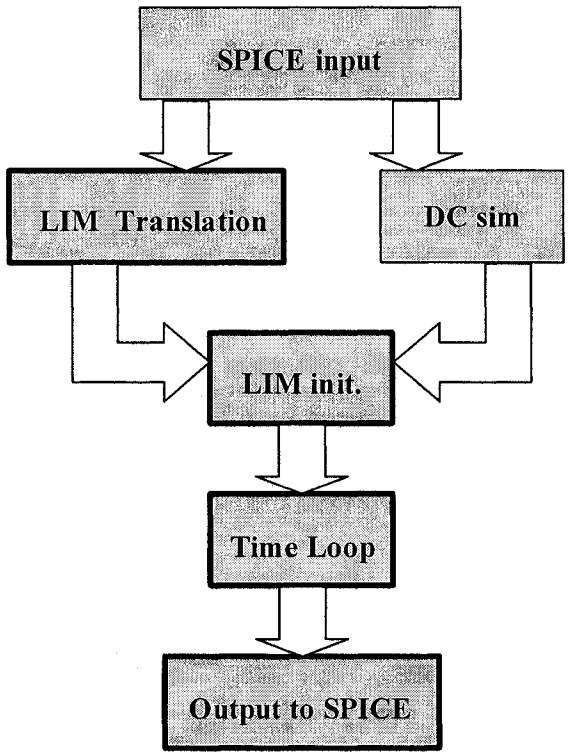


Figure 4.1 Integration flow chart of LIM and SPICE

It must be noted that for step 2, a user-specified value determines the size of the inserted inductance and capacitance values. This value is also related to the highest frequency of the input signal.

Using the above integration scheme, LIM can be invoked as a new command within SPICE and simulate the high-frequency transient response with speeds far superior than the standard SPICE.

An efficient method for the simulation of large networks has been presented. The method introduces or uses latency behavior in the network to generate update equations for the branch

currents and node voltages. First simulation algorithms were derived for linear passive networks; the formulation was then extended to handle nonlinear elements. Comparisons with SPICE have shown speedups of several orders of magnitude.

This chapter is organized as follows. In Section 4.2, we show the basic structure of the parsing and optimization method of the new LIM-SPICE. This is followed by Section 4.3, where the integration of LIM and SPICE is illustrated in detail.

4.2 Basic Parsing and Optimization Structure of LIM-SPICE

The new LIM-SPICE modified several files and created some new files.

- **Modified files**: *src/lib/fte/inpcom.c*, *src/lib/fte/makedefs*, *src/lib/fte/spcmdtab.c, src/include/fteext.h*
- **New files**: *main_lim_new.h, cell.h, matrix.h, com_lim.c, parsing_new.c, matrix.c, inp2rnew.c inp2cnew.c inp2lnew.c inp2dotnew.c*

The basic step is to modify and create the above files and re-build SPICE using command *util/build linux.*

4.2.1 Modified part

1. *src/lib/fte/spcmdtab.c:* Add the new command "lim" into SPICE's command list, which is shown in Figure 4.2.
2. *src/include/fteext.h:* Add the new main function "*com_lim( )*" into the SPICE's front end. It is shown in Figure 4.3.
3. *src/lib/fte/makedefs:* Add new .c files into the compilation list shown in Figure 4.4.
4. *src/lib/fte/inpcom.c:* Modify the original SPICE function *inp_readall*( ) and create a new function *inp_readall_new*( ). The procedure is to copy the function of

inp_readall( ) to a new function *inp_readall_new*( ) in the same file and change several places, which is shown in Figure 4.5.

```
struct comm spcp_coms] = {

    { "let", com_let, false, false, true,
      { 040000, 040000, 040000, 040000 }, E_DEFHMASK, 0, LOTS,
      arg_let,
"varname = expr : Assign vector variables." } ,
... ...
    { "source", com_source, false, false, true,
      { 1, 1, 1, 1 }, E_DEFHMASK, 1, LOTS,
      (int (*)()) NULL,
"file : Source a %s file." } ,
    { "lim", com_lim, false, false, true,
      { 1, 1, 1, 1 }, E_DEFHMASK, 1, LOTS,
      (int (*)()) NULL,
      "lim : LIM analyzes the file." } ,
... ...
}
```

Figure 4.2 Experts from spcmdtab.c after modification

```
... ...
extern int com_lim();
... ...
```

Figure 4.3 Excerpt from fteext.h after modification

```
CFILES      =  ... ...
        parsing_new.c inp2rnew.c inp2cnew.c inp2lnew.c inp2dotnew.c
com_lim.c matrix.c
COBJS =    ... ...
        parsing_new.o inp2rnew.o inp2cnew.o inp2lnew.o inp2dotnew.o
com_lim.o matrix.o
... ...
breakp2.o: breakp2.c
com_lim.o: com_lim.c
matrix.o: matrix.c
parsing_new.o: parsing_new.c
inp2rnew.o: inp2rnew.c
inp2cnew.o: inp2cnew.c
inp2lnew.o: inp2lnew.c
inp2dotnew.o: inp2dotnew.c
```

Figure 4.4 Excerpt from makedefs after modification

```
void
inp_readall_new(fp, data, numline)
    FILE *fp;
    struct line **data;
int *numline;
{
... ...
     if (!(newfp = inp_pathopen(s, "r"))) {
          perror(s);
          continue;
     }
     inp_readall(newfp, &newcard, numline);
     (void) fclose(newfp);
... ...
            default:
                prev = working;
                working = working->li_next;
                *numline = *numline + 1;
... ...
}
```

Figure 4.5 Excerpt from inpcom.c after modification

4.2.2 New function part

The main function for parsing and simulation is *com_lim*( ) in *src/lib/fte/com_lim.c*.

A. *src/lib/fte/com_lim.c*

- *com_lim*( ) and related functions *freeOPTlist( ), freeLClist( ), freeKlist( ), freeNodelist( ), freeDevlist( );*

LIM simulation related functions: *viterate( ), citerate( ), cct( ), compsource( ), BranchVectorSpace( ), NodeVectorSpace( ), FreeBranchVector( ), FreeNodeVector( ).*

B. *src/lib/fte/parsing_new.c*

- *hash*( ): calculate the hash key of device/node name string and used in indexing their own hash tables.
- *SymtabInit*( ): symbol table initialization for storing device & node name tables.
- *Devtab_insert*( ): insert a device name into device name symbol table.

- *Nodgnd_Insert*( ): insert ground node into the node name symbol table and create its corresponding node instance using *mkgndNode*( ).
- *mkgndNode*( ): create the ground node instance.
- *Nodtab_Insert*( ): insert a device name into node name symbol table and create its corresponding node instance using *insertnewNode*( ).
- *insertnewNode*( ): create the node instance.
- *OPTupdate*( ): update the optimization list during parsing.
- *Opt_RL*( ): perform the optimization after parsing.
- *RLC_insert*( ): create new device instances during parsing used in *inp2rnew*(), *inp2cnew*(), and *inp2lnew*( ).
- *gndR_update*( ): update the node's information when a resistor connects to ground used in *inp2rnew*().
- *gndC_update*( ): update the node's information when a capacitor connects to ground used in *inp2cnew*().

C. *src/lib/fte/matrix.c*

Some matrix related manipulation functions.

D. *src/lib/fte/inp2rnew.c*

Parsing for resistor line.

E. *src/lib/fte/inp2cnew.c*

Parsing for capacitor line.

F. *src/lib/fte/inp2lnew.c*

Parsing for inductor line.

G. *src/lib/fte/inp2dotnew.c*

Parsing for dot line.

H. *src/lib/fte/main_lim_new.h*

Header file for parsing and optimization related functions.

I. *src/lib/fte/cell.h*

Header file for LIM simulation related functions.

J. *src/lib/fte/matrix.h*

Header file for *matrix.c*.

4.2.3 Flow of parsing and optimization

The new parser utilizes the similar functions in SPICE. The old parser is generated by LEX and YACC. The parser of the LIM-SPICE uses the following flow in *com_lim*( ) as shown below:

(1) *inp_pathopen*( ): Open the configuration file, which contains the voltage source, save node name, and LIM related parameters. It is a SPICE function.

(2) *inp_readall_new*( ): Read in the circuit netlist file and store each line as a separate string. All the lines are stored in linked list structure: *card* or *line*. It is a modified SPICE function; I added the feature of counting the number of lines when read in the netlist files. This information will be used in the later function: *hash*( ) to calculate the key for the hash-table.

(3) *SymtabInit*( ): Initialize the symbol table for device name and node symbol table for node names.

(4) *Nodgnd_Insert*( ): Insert ground node name into node name table and create the corresponding node instance.

(5) Read in the configuration file and store LIM related parameters: *scale, grdG, grdC, KinsR, RinsL, CinsL* into the structure *pparam*.

- *scale*: scale for different unit conversion, normally set to 1
- *grdG:* the default ground conductance for node
- *grdC:* the default ground capacitance for node
- *KinsR:* the default inserted resistance for inductor branch
- *RinsL:* the default inserted inductance for resistor branch
- *CinsL:* the default inserted inductance for serial capacitor branch

(6) Insert the voltage source nodes and save nodes into node name hash table and create the corresponding node instance using *Nodtab_Insert*( ).

(7) Initialize device arrays: *Vol_R, Vol_L, Vol_C, Vol_K, Vol_ckt*.

(8) Parse the whole netlist line by line using different functions according to the first character of each logic line: *INP2Rnew*( ) for Rxxx; *INP2Cnew*( ) for Cxxx.

(9) Start optimization process to eliminate the redundant nodes and branches for two types of scenarios: *R-L* and *R-L-R*.

(10) Build the node and branch data structures for LIM simulation.

(11) Start LIM simulation.

(12) At each time step, save the wanted node voltages into array.

(13) Write the saved node voltages into SPICE's raw data file format, which can be processed by "*nutmeg*" command of SPICE.

The above gives a brief guideline for the LIM-SPICE integration. The detailed implementation is shown in [44].

5 NUMERICAL EXAMPLES

To illustrate the validity of the stability conditions for the LIM method derived in the previous parts, several different stability criteria for different schemes are shown here.

The object under study is a 1-D *RLGC* circuit, as shown in Figure 5.1. The numerical spatial region consists of 22 segments in one direction. Each segment is loaded with the same *R*, *L*, *G*, and *C* value. The exciting source is a voltage pulse with rise and fall time of 0.1 ns and the width is 5 ns. All three difference schemes are tested.

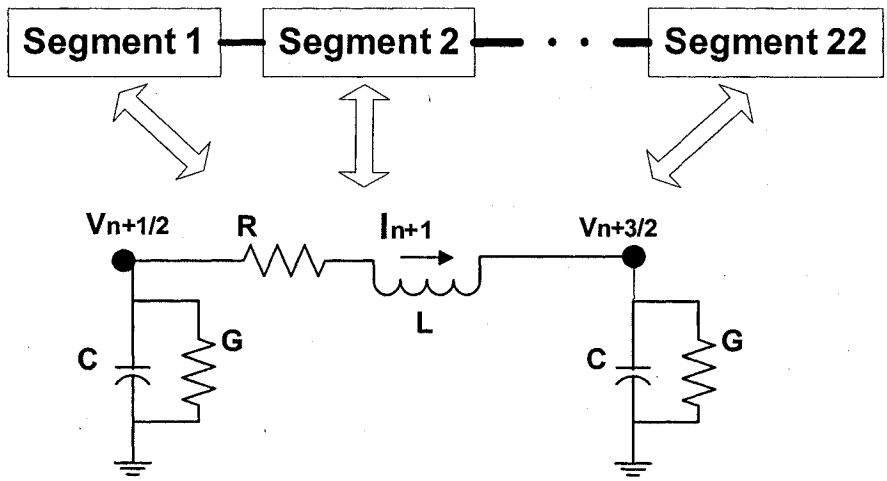


Figure 5.1 Test setup for validating derived stability conditions

(1) Explicit (forward Euler) case: R = 20.0 Ω, L = 0.01 nH, G = 0, and C = 0.02 nF. According to the stability criteria in (3.29), the maximum time step is

$$\Delta t = \sqrt{LC}\left[\sqrt{1+\left(\frac{R}{4}\sqrt{\frac{C}{L}}\right)^2}-\left(\frac{R}{4}\sqrt{\frac{C}{L}}\right)\right] = 0.00099505 \text{ ns} \tag{5.1}$$

The node voltages of nodes 1, 2, and 3 are simulated at $0.99\,\Delta t_{\max}$ and $1.01\,\Delta t_{\max}$ are shown in Figures 5.2 and 5.3. Obviously, the algorithm is stable when the time step is

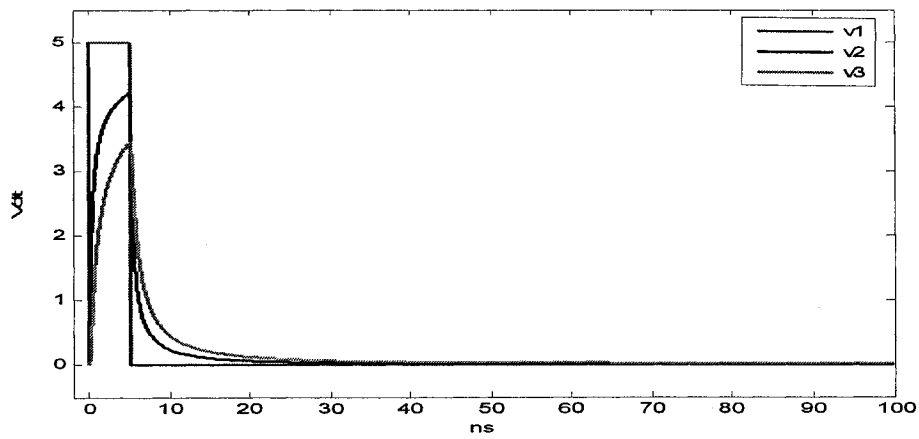


Figure 5.2 Node voltages at nodes 1, 2, and 3 with 0.99 $\Delta t_{\max}$ for explicit case

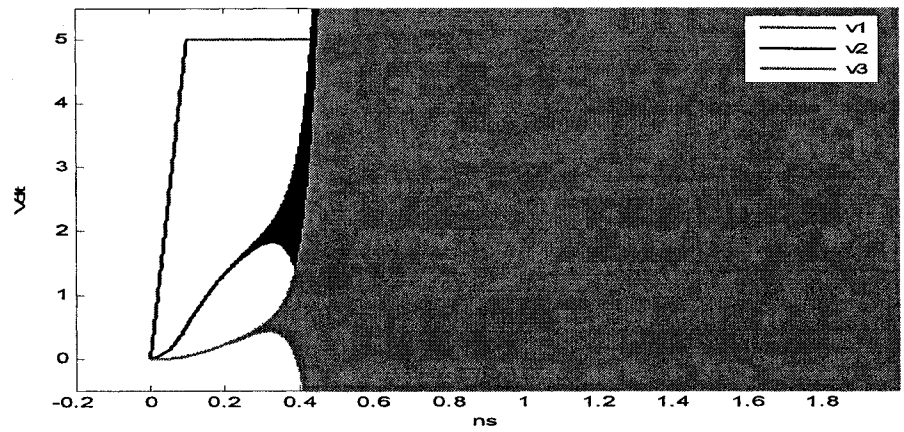


Figure 5.3 Node voltages at nodes 1, 2, and 3 with 1.01 $\Delta t_{\max}$ for explicit case

$0.99\,\Delta t_{\max}$, but it is unstable when the time step is $1.01\,\Delta t_{\max}$. Therefore, the stability condition of (3.29) is valid.

(2) Semi-implicit (leap frog) case: R = 20.0 Ω, L = 0.01 nH, G = 0, and C = 0.02 nF. According to the stability criteria in (3.32), the maximum time step is

$$\Delta t = \sqrt{LC} = 0.01414213 \text{ ns} \tag{5.2}$$

The node voltages of nodes 1, 2, and 3 are shown in Figure 5.4 for $0.99\,\Delta t_{\max}$ and in Figure 5.5 for at $1.01\,\Delta t_{\max}$. Figure 5.4 shows that the solution is stable when the maximum time step condition is met. Figure 5.5 shows that the node voltage becomes unstable after about 28000th time step when the stability condition is not satisfied. Thus, the stability condition of (3.32) obtained in the above sections are verified.

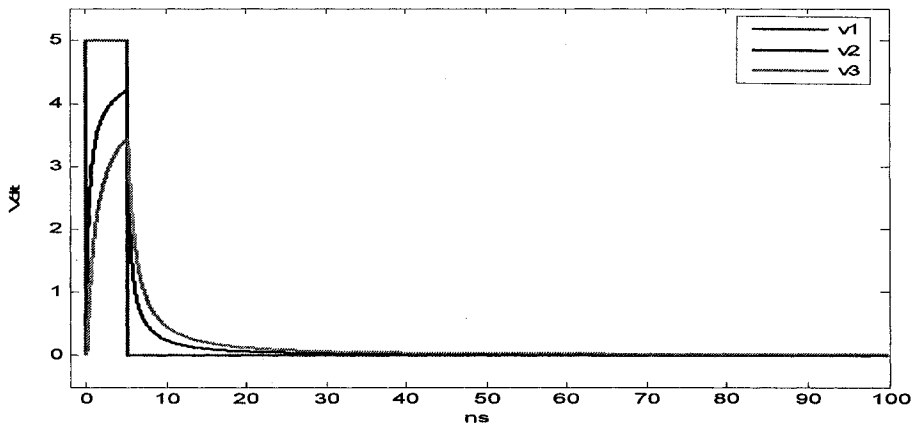


Figure 5.4 Node voltages at nodes 1, 2, and 3 with 0.99 $\Delta t_{\max}$ for semi-implicit case

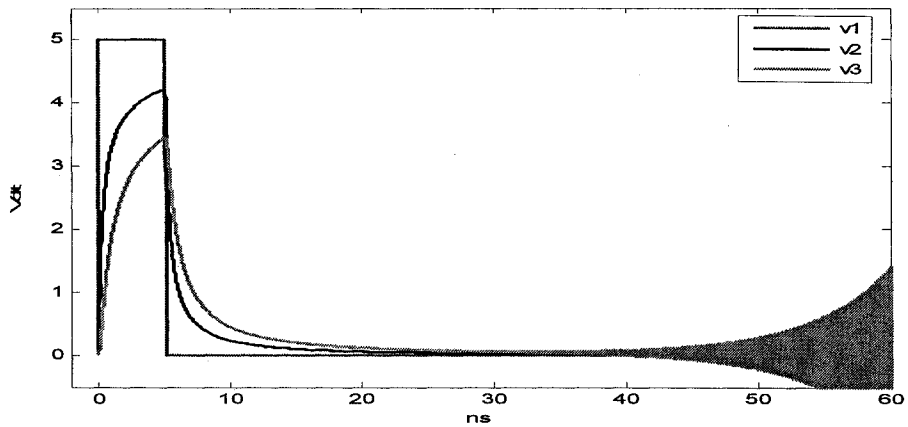


Figure 5.5 Node voltages at nodes 1, 2, and 3 with 1.01 $\Delta t_{\max}$ for semi-implicit case

(3) Fully implicit (backward Euler) case: R = 20.0 Ω, L = 0.01 nH, G = 0, and C = 0.02 nF. According to the stability criteria in (3.36), the maximum time step is

$$\Delta t = \sqrt{LC}\left[\sqrt{1+\left(\frac{R}{4}\sqrt{\frac{C}{L}}\right)^2}+\left(\frac{R}{4}\sqrt{\frac{C}{L}}\right)\right] = 0.20099505 \text{ ns} \tag{5.3}$$

The node voltages of nodes 1, 2, and 3 are simulated at $0.99\,\Delta t_{\max}$ and $1.01\,\Delta t_{\max}$ are shown in Figures 5.6 and 5.7. The node voltage in Figure 5.6 is stable while the voltage in Figure 5.7 starts to increase from around 40th time step. The low accuracy of voltage waveform in Figure 5.7 is due to the very large time step: 0.2 ns.

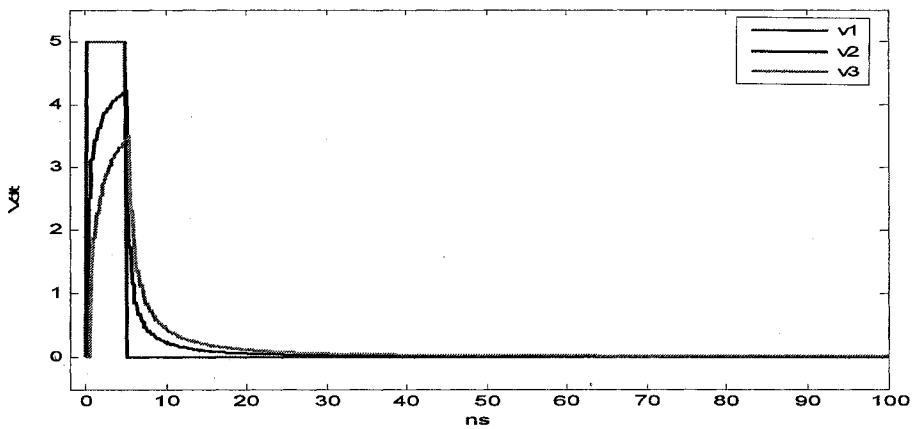


Figure 5.6 Node voltages at nodes 1, 2, and 3 with 0.99 $\Delta t_{\max}$ for fully implicit case

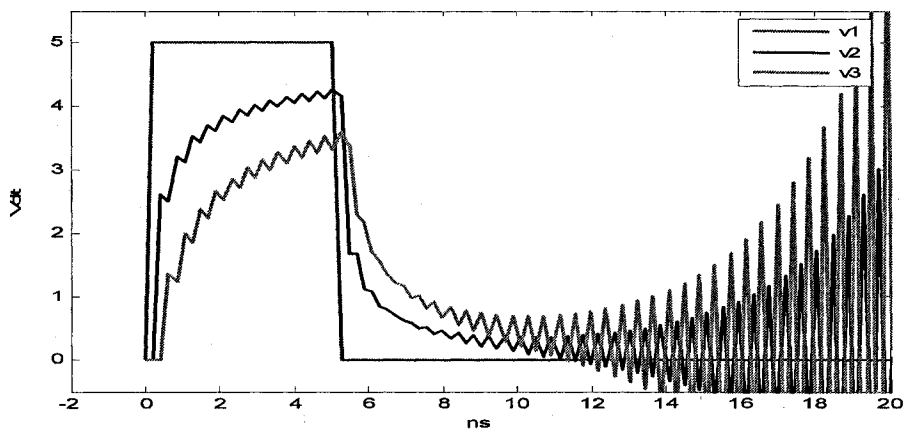


Figure 5.7 Node voltages at nodes 1, 2, and 3 with 1.01 $\Delta t_{\max}$ for fully implicit case

Next, the linear computational cost of the LIM method is demonstrated with the ease of use in SPICE environment. Simulations were performed in the new LIM-SPICE simulator. To demonstrate the speed advantage of the LIM-SPICE simulator, several large-sized linear networks consisting of only resistors, self-inductors and ground capacitors were analyzed. The benchmark results for SPICE and LIM-SPICE are shown in Table 5.1. From Table 5.1, we can see that SPICE's computational complexity is at best superlinear, but not linear. LIM's speed does scale linearly from 4 s to 44 s for the cases time stop = 20 ns and 200 ns.

Table 5.1. Simulation speed comparison of LIM-SPICE and SPICE

No. of Nodes		2002	3988	70 000 timestop = 20 ns	70 000 timestop = 60 ns	70 000 timestop = 120 ns	70 000 timestop = 160 ns	70 000 timestop = 200 ns
SPICE (sec)	Parsing	< 1	2	**914**	914	914	914	914
	Simulation	69	19	**461**	2200	**6116**	10812	**16836**
	Total	69	21	1375	3114	7028	11726	17750
LIM–SPICE (sec)	Parsing	< 1	2	**90**	90	90	90	90
	Simulation	49	2	**4**	13	**26**	35	**44**
	Total	49	4	94	103	116	125	134
Speedup	Total/Total	1.41	5.25	***14.63***	30.23	***60.59***	93.81	***132.46***

Results from Table 5.1 clearly indicate the superiority of the LIM method. The speedup factor in favor of LIM-SPICE goes up as the size of the network increases. Figure 5.8 shows a plot of the speedup factor versus problem size. The x-axis is logarithmic value of the equivalent problem size normalized to the smallest example. The problem size is proportional to the number of unknowns such as the node voltage and branch current at N time points. The y-axis shows the ration of the LIM-SPICE's total runtime to the SPICE's total runtime. The waveform verification for the test case of 2000 nodes is shown in Figure 5.9.

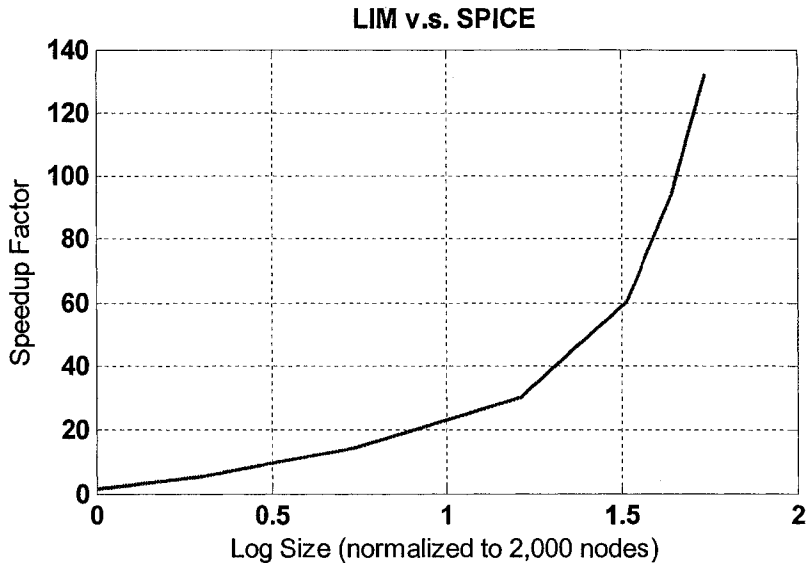


Figure 5.8 Plot of speedup factor of LIM-SPICE over SPICE versus network size

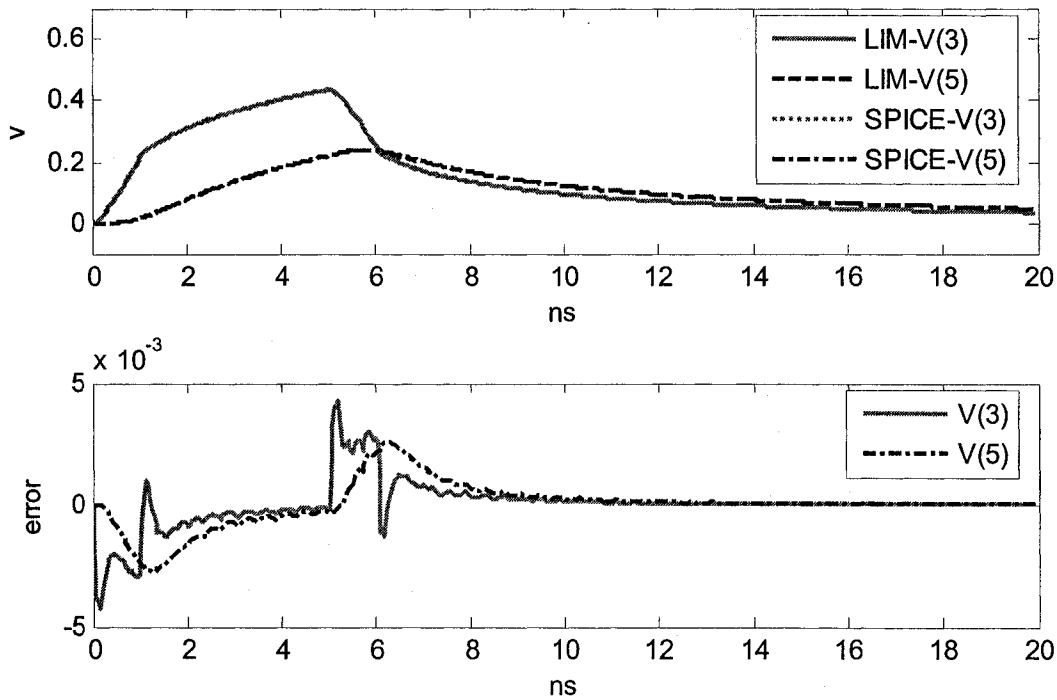


Figure 5.9 Simulation results of a test circuit with 2000 nodes;
the above graph: simulation results of LIM and SPICE;
the bottom graph: error between LIM and SPICE

Figure 5.10 indicates an interconnect network with lots of resistors, inductors, and capacitors. The exciting source is a pulse input which is has rise time = 1 ns, fall time = 1 ns, width = 4 ns, and rises from 0 V to 1 V. The node voltages have good agreement with SPICE simulation result. In Figure 5.10 the network consists of resistors, inductors, and capacitors, which has about 70K nodes. The exciting source is the same as the one in Figure 5.9. Good agreement with SPICE simulation is shown through error estimator LIM−SPICE.

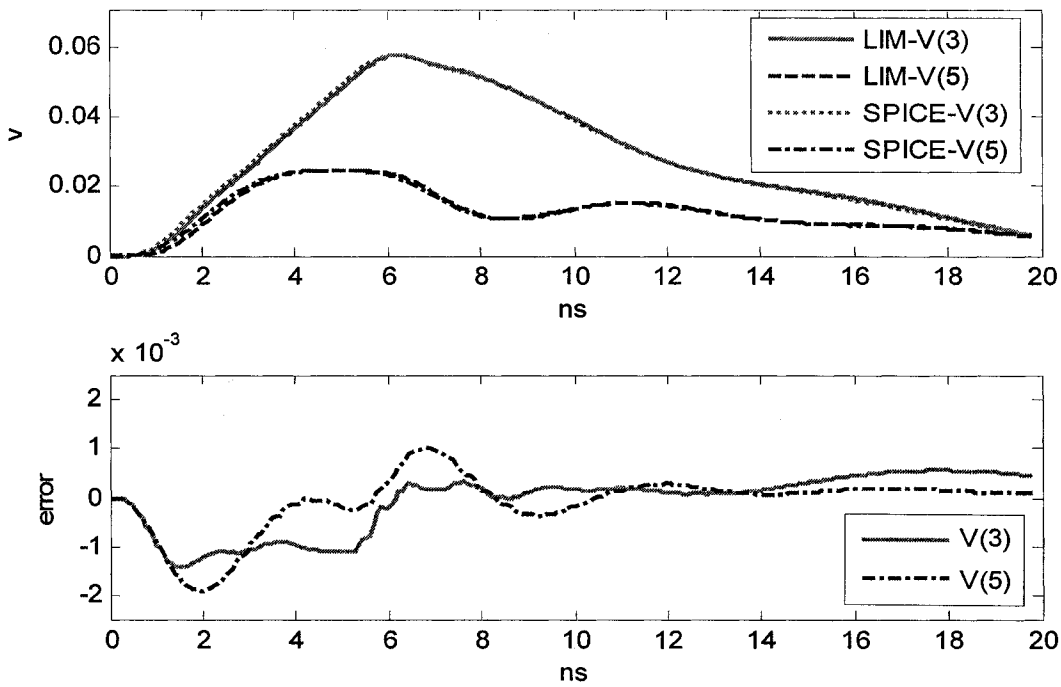
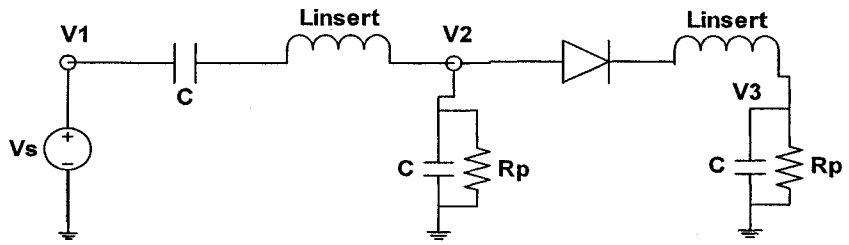


Figure 5.10 Simulation results of a test circuit with 70 000 nodes and timestop = 20ns;
the above graph: simulation results of LIM and SPICE;
the bottom graph: error between LIM and SPICE

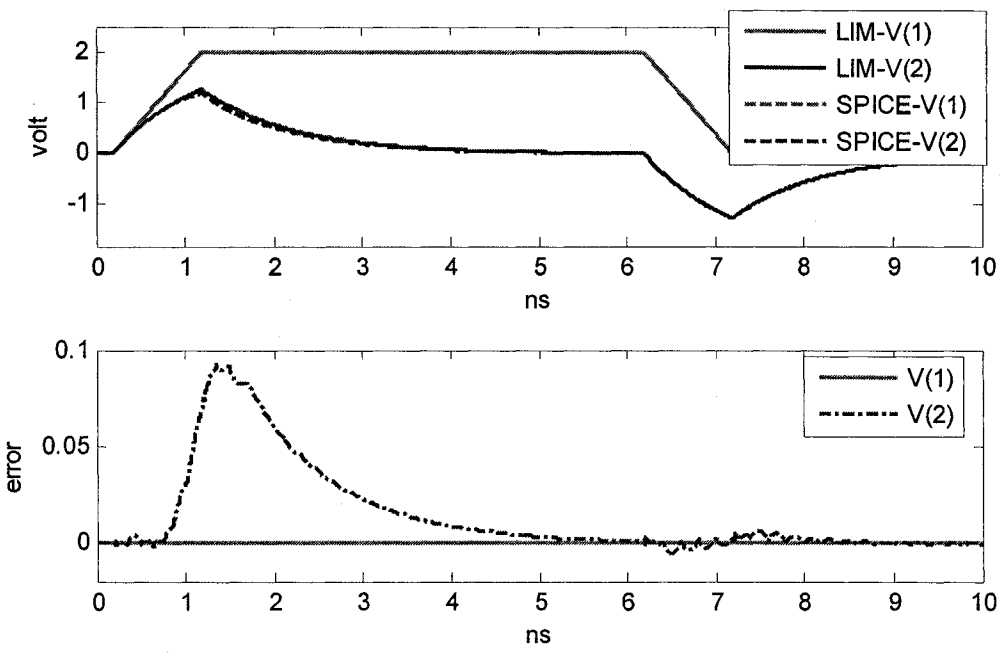
Next, results for the new branch capacitor model and mutual (forward Euler) inductor models were obtained and are shown in Figures 5.11 and 5.12, respectively. The test circuit in Figure 5.11 (a) has a branch capacitor of 0.1 nF, an inserted inductor of 0.001 nH, grounded

capacitor of 0.001 nF, grounded resistor of 10 Ω, and a diode with resistance of 0.001 Ω. The exciting voltage source is a pulse, which has rise/fall time = 1 ns, width = 5 ns, delay = 0.2 ns. Figure 5.11 (b) shows the simulation result for node 1 and 2, which is also compared with SPICE by error estimator LIM−SPICE.

The test circuit for mutual inductor (F.E.) model in Figure 5.12 has very small grounded resistor of 1.0e-6 Ω and grounded capacitors of 1 nF for all nodes. The voltage source is a pulse with rise/fall time = 1 ms, width = 5 ms, delay = 0.2 ms. The voltage waveform of node 2 is compared with SPICE using error estimator LIM−SPICE.

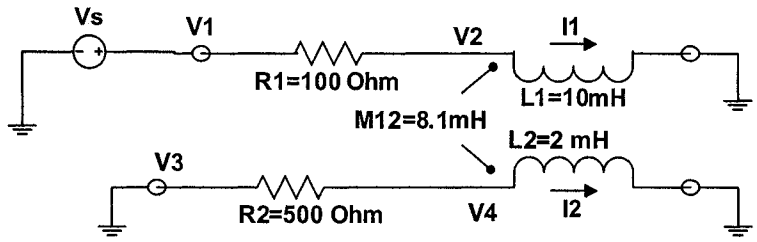


(a)

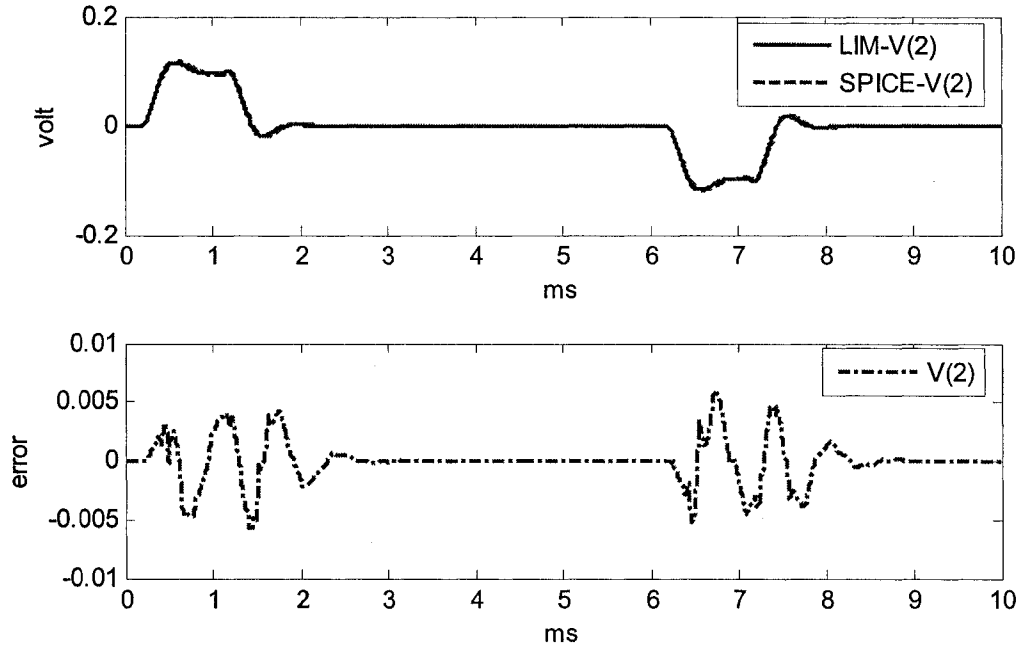


(b)

Figure 5.11 Branch capacitor: (a) test circuit setup and (b) LIM and SPICE simulation results



(a)



(b)

Figure 5.12 Mutual inductor (F.E.):
(a) test circuit setup and (b) LIM & SPICE simulation results of node 2

To verify the accuracy and efficiency of the mutual inductor (*K* element) model, a power grid network with only self/mutual inductors and capacitors is used as test circuit. The resistors of the circuit are kept small. The circuit consists of 150 self-inductors, 4500 mutual inductors, and 1700 capacitors. The results are shown in Figure 5.13. The simulation time for LIM-SPICE is 45 s against 1252 s for SPICE. This translates to a speedup factor of 28.

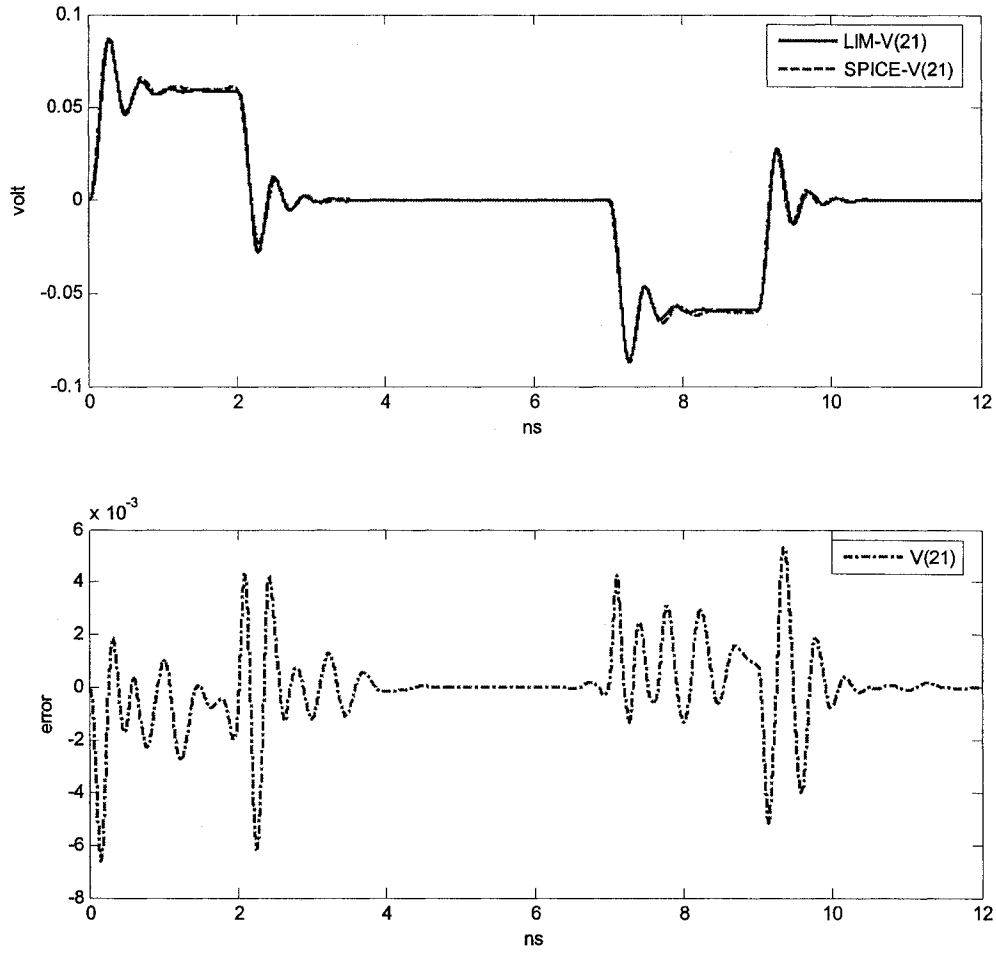
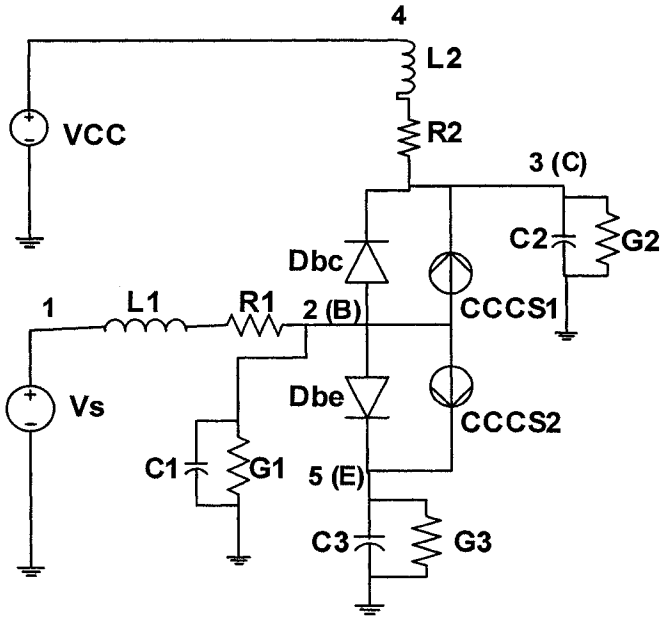
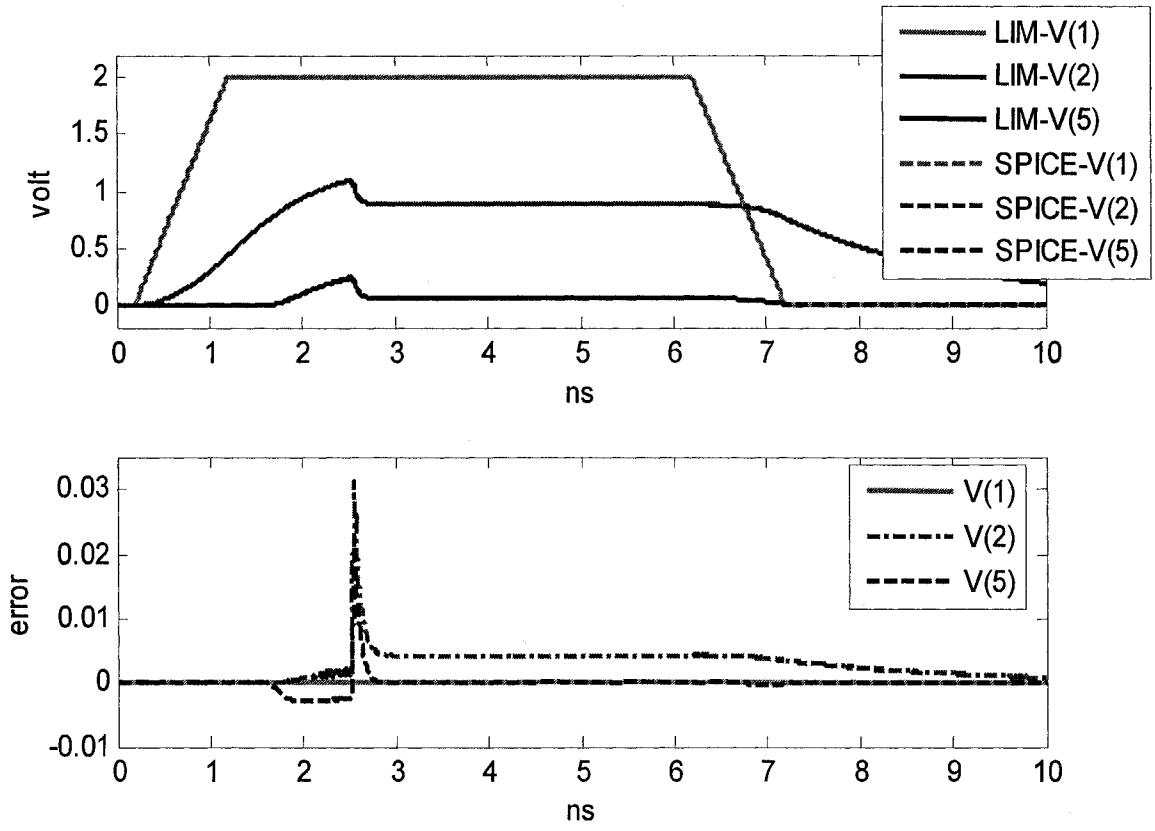


Figure 5.13. Mutual inductor (K element): simulation results;
the above graph: simulation results of LIM and SPICE;
the bottom graph: error between LIM and SPICE

Nonlinear capability of LIM is shown through the following two test examples. First a simple BJT inverter is simulated. Figure 5.14 shows the inverter test circuit setup and the LIM and SPICE simulation results of node voltages. The nodes shown in the Figure 5.14 are nodes 1, 2, and 5 in the model. This simple circuit is defined as a cell. Four such cells are cascaded to define a second test structure. The simulation results are shown in Figure 5.15, which shows the voltage of internal node 5 of cells 1 and 3.



(a)



(b)

Figure 5.14 A simple BJT transistor model:
(a) test circuit setup and (b) LIM and SPICE simulation results

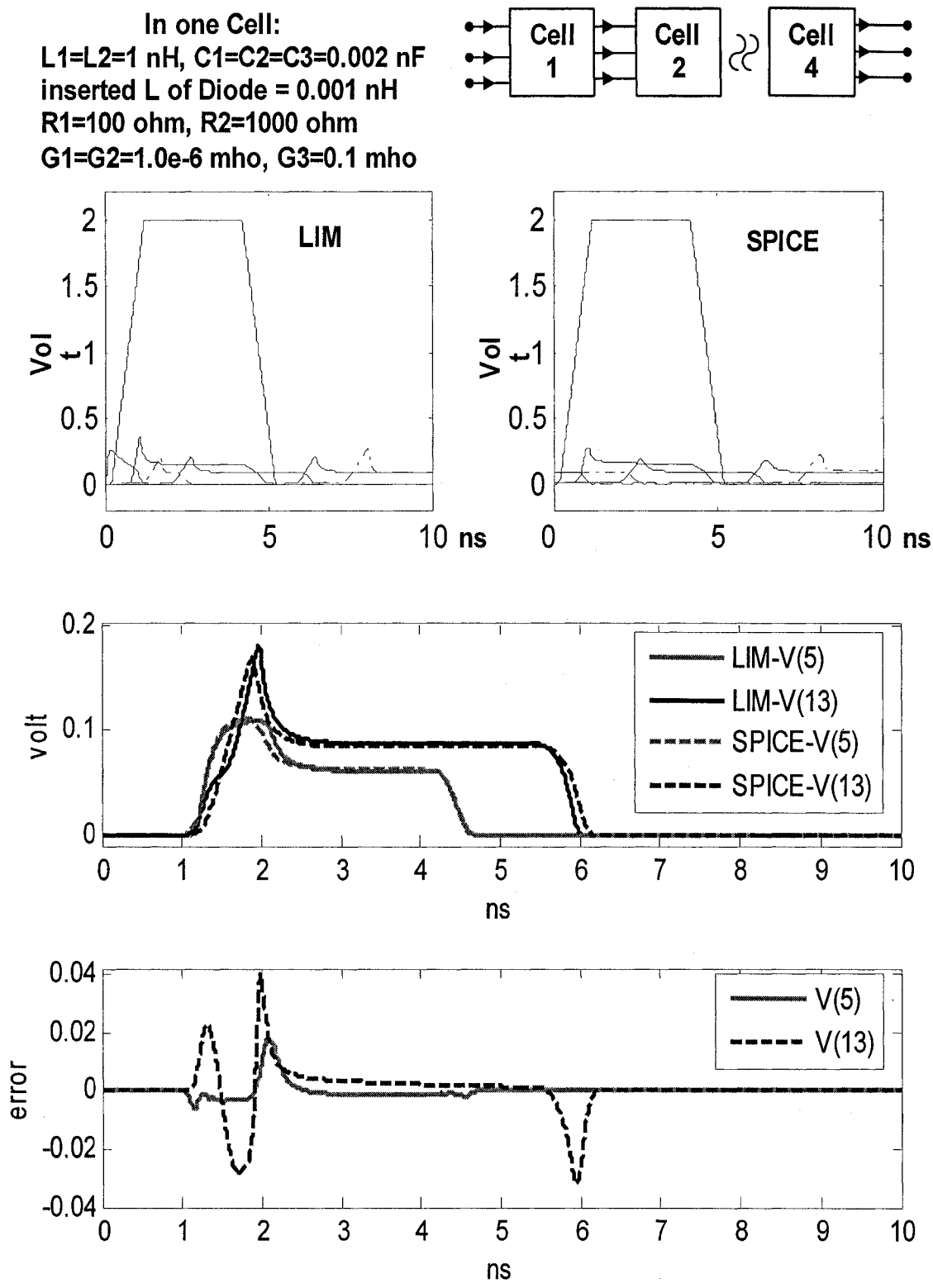


Figure 5.15 Cascading BJT transistors test setup and simulation results

To validate the above general formulation for skin effect in LIM, we use an example in [45], in which a 100-m twisted-pair cable is considered with the parameters shown in Table 5.2.

Table 5.2 Twisted-pair cable parameters

Z_0 (Ω)	L_e (nH/m)	C (nF/m)	$v_{relative}$ (m/ns)	R_0 (Ω/m)	R_S ($\Omega/m\text{-}GHz^{\rho}$)	ρ	f_{max} (GHz)
100	476.19047619	0.047619048	0.7	0	16	0.5	0.2

The circuit configuration is shown in Figure 5.16. The cable in Figure 5.16 is a 100-m twisted-pair cable with far end open circuited. The magnitude of the excitation pulse is 2.7 V, the rise and fall times are 1 ns, and the pulse width is 80 ns. The cable is modeled using 1000 *RLC* blocks, as shown in Figure 5.17. The simulation results using our formulation and the measurement results from [45] are shown in Figures 5.18 and 5.19.

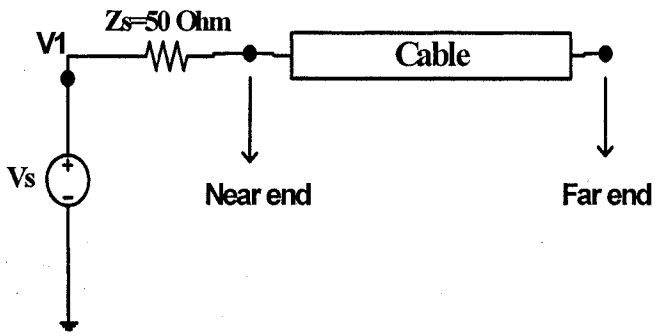


Figure 5.16 Transmission line

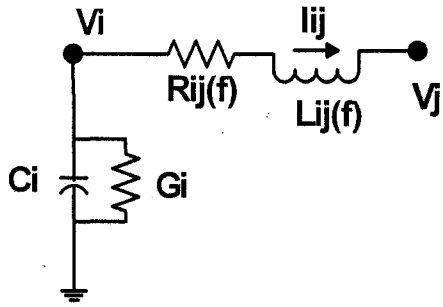
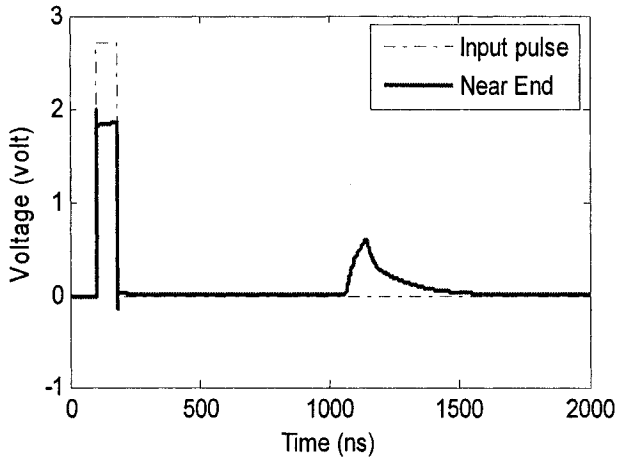


Figure 5.17 One segment of the cable model



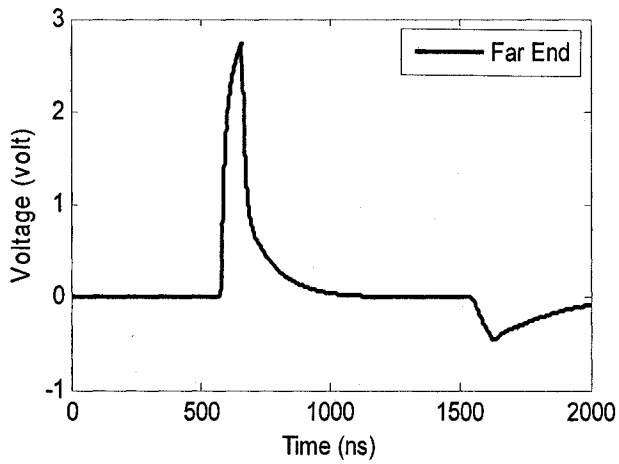
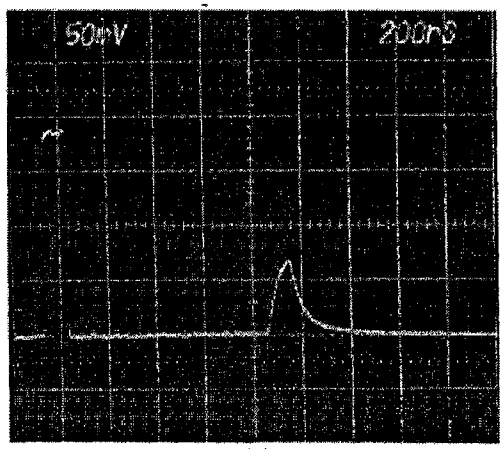
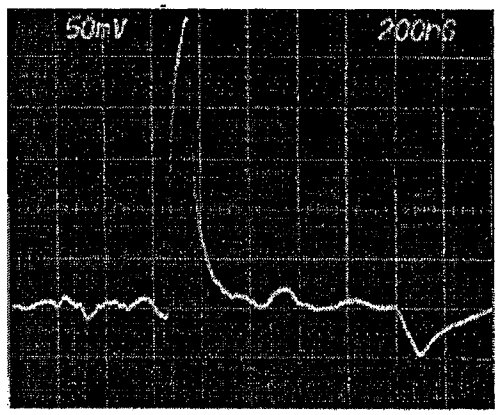


Figure 5.18 Simulation results using LIM skin-effect formulations



(a)



(b)

Figure 5.19 Measurement result: (a) near end (b) far end

6 CONCLUSION

This dissertation gave brief guidelines for implementing an n-port transmission line model in SPICE, which facilitates integrating LIM into SPICE. Then the stability analysis of the LIM simulation method for explicit, semi-explicit, and fully implicit difference schemes was presented. With the proper constraints, the largest possible time step could optimize the algorithm. Improved formulations for branch series capacitor and mutual inductors were proposed and verified through a power ground plane test circuit. The frequency-dependent modeling, such as skin effect, was proposed and implemented. Measurement results are used to verify the accuracy of the skin effect modeling. The integration guide for the LIM-SPICE simulator exploits the speed advantage of LIM in the familiar SPICE shell by integrating LIM as a new built-in command in SPICE3f5. The speed acceleration is verified through the testing of several large linear networks. Several orders of magnitude in simulation speedup has been observed. A BJT model is also developed to show the capability of simulating a general nonlinear circuit. These improvements help make the LIM algorithm more suitable for a larger class of simulations including power distribution networks and general nonlinear circuits. Possible unconditionally stability version of LIM is also addressed. Comparisons with SPICE have shown speedups of several orders of magnitude.

APPENDIX A – STABILITY ANALYSIS FOR EXPLICIT CASE

This appendix shows the detail derivation for the stability analysis of the explicit case:

From the definition of the explicit scheme: $RI = \frac{L}{\Delta t} c_0 I^n$, $GV = \frac{C}{\Delta t} d_0 V^{n-\frac{1}{2}}$, the updating equation is transformed into the Fourier domain as below:

$$\frac{C}{\Delta t} V_{k+1/2}^{n+1/2} = \frac{C}{\Delta t} V_{k+1/2}^{n-1/2} + \frac{I_{k+1}^n}{\Delta x} - \frac{I_k^n}{\Delta x} - \frac{C}{\Delta t} d_0 V_{k+1/2}^{n-1/2} \tag{A.1}$$

$$-\frac{V_{k+1/2}^{n+1/2}}{\Delta x} + \frac{V_{k-1/2}^{n+1/2}}{\Delta x} + \frac{L}{\Delta t} I_k^{n+1} = \frac{L}{\Delta t} I_k^n - \frac{L}{\Delta t} c_0 I_k^n \tag{A.2}$$

Using (3.18) and $I_{k\pm1}^n(\zeta) = e^{\pm ik\zeta} I_k^n(\zeta), V_{k\pm1}^n(\zeta) = e^{\pm ik\zeta} V_k^n(\zeta)$, (A.1) becomes

$$\frac{C}{\Delta t} \int_{-\pi}^{\pi} V_{k+1/2}^{n+1/2}(\zeta) e^{im\zeta} d\zeta = \int_{-\pi}^{\pi} [\frac{C}{\Delta t}(1-d_0) V_{k+1/2}^{n-1/2}(\zeta) + \frac{1}{\Delta x}(e^{i\zeta} - 1) I_k^n(\zeta)] e^{im\zeta} d\zeta \tag{A.3}$$

$$\Rightarrow \quad \frac{C}{\Delta t} V_{k+1/2}^{n+1/2}(\zeta) = \frac{C}{\Delta t}(1-d_0) V_{k+1/2}^{n-1/2}(\zeta) + \frac{1}{\Delta x}(e^{i\zeta} - 1) I_k^{n-1/2}(\zeta) \tag{A.4}$$

Similarly, (A.2) becomes

$$\frac{1}{\Delta x}(e^{-i\zeta} - 1) V_{k+1/2}^{n+1/2}(\zeta) + \frac{L}{\Delta t} I_k^{n+1}(\zeta) = \frac{L}{\Delta t}(1-c_0) I_k^n(\zeta) \tag{A.5}$$

Then $u_k^{n+1} = [V_{k+1/2}^{n+1/2} \quad I_k^{n+1} \quad I_k^n \quad V_{k+1/2}^{n-1/2}]^T$,

$$A \cdot U_k^{n+1} = B \cdot U_k^n \tag{A.6}$$

$$A=\begin{pmatrix} C/\Delta t & 0 & 0 & 0 \\ \dfrac{e^{-i\zeta}-1}{\Delta x} & L/\Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.7}$$

$$B=\begin{pmatrix} \dfrac{C(1-d_0)}{\Delta t} & \dfrac{e^{i\zeta}-1}{\Delta x} & 0 & 0 \\ 0 & \dfrac{L(1-c_0)}{\Delta t} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{A.8}$$

So the characteristic polynomial of $A^{-1}B$ is

$$P(\lambda)=\lambda^2-\lambda[2-(c_0+d_0)-4\gamma^2\sin^2(\zeta/2)]+(1-c_0)(1-d_0) \tag{A.9}$$

with the Courant number $\gamma=\Delta t/(\Delta x\sqrt{LC})$, $(\Delta x=1)$.

From the requirement of stability condition: $|\lambda|\le 1$, for any , the stability condition becomes (3.26):

$$\gamma\le\frac{1}{2}\sqrt{4-2(\frac{R\Delta t}{L}+\frac{G\Delta t}{C})+\frac{RG\Delta t^2}{LC}}$$

The above gives the stability condition for explicit scheme of LIM.

REFERENCES

[1] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Computer-Aided Design*, vol. CAD-9, pp. 352-366, April 1990.

[2] J. Bracken, V. Raghavan, and R. Rohrer, "Interconnect simulation with asymptotic waveform evaluation (AWE)," *IEEE Transactions on Circuits and Systems*, vol. CAS-39, no. 11, pp. 869-878, November 1992.

[3] T. Tang and M. Nakhla, "Analysis of lossy multiconductor transmission lines using the asymptotic wave-form evaluation technique," *IEEE Trans. Microwave Theory Tech.*, vol. 39, no. 12, pp. 2107-2116, December 1991.

[4] E. Chiprout and M. S. Nakhla, "Analysis of interconnect networks using complex frequency hopping (CFH)," *IEEE Trans. Computer-Aided Design*, vol. 14, no. 2, pp. 186-200, February 1995.

[5] P. Feldmann and R. W. Freund, "Efficient linear circuit analysis by Padé via Lanczos process," *IEEE Trans. Computer-Aided Design*, vol. 14, no. 5, pp. 639-649, May 1995.

[6] M. Celik and A. Cangellaris, "Simulation of dispersive multiconductor transmission lines by Padé approximation via Lanczos Process," *IEEE Trans. Microwave Theory Tech.*, vol. 44, no. 12, pp. 2525-2535, December 1996.

[7] W. T. Beyene and J. E. Schutt-Ainé, "Accurate frequency-domain modeling and efficient circuit simulation of high-speed package interconnects," *IEEE Trans. Microwave Theory Tech.*, vol. MTT-45, pp.1941-1947, October 1997.

[8] A. Odabasioglu, M. Celik, and L. Pillegi, "PRIMA: Passive reduced-order interconnect macromodeling algorithm," in *Proc. IEEE/ACM International Conf. on Computer-Aided Design*, Nov. 1997, pp. 58-65.

[9] Q. Yu, J. Wang, and E. Kuh, "Passive multipoint moment matching model order reduction algorithm on multi-port distributed interconnect networks," *IEEE Trans. Circuit Syst.*, vol. CAS-46, pp. 140-160, January 1999.

[10] Z. Qin and C. Cheng, "Realizable parasitic reduction using generalized Y-Δ transformation," in *Proc. ACM/IEEE Design Automation Conf.*, June 2003, pp. 220-225.

[11] Y. Cao, Y. Lee, T. Chen, and C. C. Chen, "HiPRIME: Hierarchical and passivity reserved interconnect macromodeling engine for RLKC power delivery," in *Proc. ACM/IEEE Design Automation Conf.*, June 2002, pp. 379-384.

[12] G. Bai, S. Bobba, and I. N. Hajj, "Simulation and optimization of the power distribution network in VLSI circuits," in *Proc. IEEE/ACM International Conf. on Computer-Aided Design*, 2000, pp. 481-486.

[13] J.-H. Kim, M. Swaminathan, and Y. Suh, "Modeling of power distribution networks for mixed signal applications," in *Proc. 2001 IEEE EMC Int. Symp.*, vol. 2, 2001, pp. 1117-1122.

[14] T. Chen, and C. C. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative method," in *Proc. ACM/IEEE Design Automation Conf.*, June 2001, pp. 559-562.

[15] H.-H. Su, K. H. Gala, and S. S. Sapatnekar, "Fast analysis and optimization of power/ground networks," in *Proc. IEEE/ACM International Conf. on Computer-Aided Design*, 2000, pp. 477-482.

[16] M. Zhao, R. V. Panda, S. S. Sapatnekar, T. Edwards, R. Chaudhry, and D. Blaauw, "Hierarchical analysis of power distribution networks," in *Proc. ACM/IEEE Design Automation Conf.*, 2000, pp. 150-155.

[17] C. Christopoulos, *The Transmission-Line Modeling Method: TLM.* New York, NY: Oxford Univ. Press-USA, 1995.

[18] Y-M. Lee and C. C.-P.Chen, "Power grid transient simulation in linear time based on transmission-line-modeling alternating-direction-implicit method," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 11, pp. 1343-1352, Nov. 2002.

[19] Y.-M. Lee and C. C.-P.Chen, "The power grid transient simulation in linear time based on 3-D alternating-direction-implicit method," *IEEE Trans. Computer-Aided Design*, vol. 22, no. 11, pp. 1545-1550, Nov. 2003.

[20] J. E. Schutt-Aine and R. Mittra, "Modeling and simulation of high-speed digital circuit interconnection," University of Illinois at Urbana-Champaign, Technical Report No. 88-2, April 1988.

[21] Z. Deng, "N-port dummy device installation in SPICE," M.S. thesis, University of Illinois at Urbana-Champaign, 2002.

[22] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design* New York: Chapman & Hall, 1994.

[23] I. S. Duff, Ed., *Sparse Matrices and Their Uses*. London: Academic Press, 1981.

[24] T. L. Quarles, "The Spice3 implementation guide," University of California at Berkeley, Memorandum No. UCB/ERL M89/44, April 1989.

[25] H. H. Chen and D. D. Ling, "Power supply noise analysis methodology for deep-submicron VLSI chip design," in *Proc. Design Automation Conf.*, 1997, pp. 638-643.

[26] M. K. Gowan, L. L. Biro, and D. B. Jackson, "Power considerations in the design of the Alpha 21 264 microprocessor," in *Proc. Design Automation Conf.*, 1998, pp. 726-731.

[27] A. Dharchoudhury, R. Panda, D. Blaauw, and R. Vaidyanathan, "Design and analysis of power distribution networks in PowerPC™ microprocessors," in *Proc. Design Automation Conf.*, 1998, pp. 738-743.

[28] Y.-M. Jiang and K.-T. Cheng, "Analysis of performance impact caused by power supply noise in deep submicron devices," in *Proc. Design Automation Conf.*, 1999, pp. 760-765.

[29] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "A multigridlike technique for power grid analysis," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 10, pp. 1148-1160, Oct. 2002.

[30] J. E. Schutt-Ainé, "Latency insertion method (LIM) for the fast transient simulation of large networks," *IEEE Trans. Circuits Syst.*, vol. 48, pp. 81-89, Jan. 2001.

[31] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equation in isotropic media," *IEEE Trans. Antennas Propagat.*, vol. AP-14, pp. 302-307, May 1966.

[32] Z. Deng and J. E. Schutt-Aine, "Stability analysis of latency insertion method (LIM)," in *Proc. IEEE 13th EPEP Conf.*, October 25-27, 2004, pp. 167-170.

[33] J. A. Pereda, O. Garcia, A. Vegas, and A. Prieto, "Numerical dispersion and stability analysis of the FDTD technique in lossy dielectrics," *IEEE Microwave and Guided Wave Letters*, vol. 8, no. 7, pp. 245-247, July 1998.

[34] J. W. Thomas, *Numerical Partial Differential Equations, Finite Difference Methods*. New York: Springer-Verlag, 1995.

[35] W. Thiel and L. P. B. Katehi, "Some aspects of stability and numerical dissipation of the finite-difference time-domain (FDTD) technique including passive and active lumped elements," *IEEE Trans. Microwave Theory Tech.*, vol. 50, no. 9, pp. 2159-2165, September 2002.

[36] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*. Belmont, California: Wadsworth Books, 2002.

[37] A. Devgan, H. Ji, and W. Dai, "How to efficiently capture on-chip inductance effects: Introducing a new circuit element k," in *Proc. IEEE/ACM International Conf. on Computer-Aided Design*, Nov. 2000, pp. 150-155.

[38] J. Ogrodzki, *Circuit Simulation Methods and Algorithms*. Boca Raton, Florida: CRC Press Inc., 1994.

[39] A. S. Sedra and K. C. Smith, *Microelectronic Circuits,* 4th ed. New York: Oxford University Press Inc., 1998.

[40] J. H. Beggs, R. J. Luebbers, K. S. Yee, and K. S. Kunz, "Finite-difference time-domain implementation of surface impedance boundary conditions," *IEEE Trans. Antennas Propagat.*, vol. 40, no. 1, pp. 49-56, Jan. 1992.

[41] K. M. Coperich, J. Morsey, V. I. Okhmatovski, A. C. Cangellaris, and A. E. Ruehli, "Systematic development of transmission-line models for interconnects with frequency-dependent losses," *IEEE Trans. MTT*, vol. 49, no. 10, part 1, pp. 1677-1685, Oct. 2001.

[42] N. S. Nahman and D. R. Holt, "Transient analysis of coaxial cables using the skin effect approximation $A + B\sqrt{s}$," *IEEE Trans. Circuit Theory*, vol. 19, no. 5, pp. 443-451, Sept. 1972.

[43] J. W. Schuster and R. J. Luebbers, "An accurate FDTD algorithm for dispersive media using a piecewise constant recursive convolution technique," in *Proc. IEEE Int. Symposium AP-S*, vol. 4, June 21-26, 1998, pp. 2018-2021.

[44] Z. Deng, "LIM-SPICE integration," internal report, University of Illinois at Urbana-Champaign, 2005.

[45] J. E. Schutt-Ainé, "High-frequency characterization of twisted-pair cables," *IEEE Trans. Communications*, vol. 49, no. 4, pp. 598-601, April 2001.

AUTHOR'S BIOGRAPHY

Zhichao Deng was born on August 8, 1977, in Beijing, China. From August 1995 to July 2000 he attended the Tsinghua University at Beijing and received the Bachelor of Engineering degree in Electronic Engineering.

For his undergraduate thesis, Mr. Deng built some models for an oscillator library in the VHDL-AMS language (IEEE 1076.1). The library is used for debugging and testing the VHDL-AMS development platform: VeriasHDL of Analogy. In summer of 1999, he worked for Tsinghua Tong Fang Electronic Co., accomplishing a proposal of network design for a residential community security system.

In January 2001, Mr. Deng began his graduate studies at the University of Illinois at Urbana-Champaign. He has held a research assistantship with the Center for Computational Electromagnetics under the supervision of Professor Jose E. Schutt-Aine. He worked on the SPICE simulator and multiconductor transmission line modeling. His main research thrust has been the simulation and macromodeling of the large high-speed circuits.

He also held a teaching assistantship for four semesters, involving an automated microwave measurement course. From July to December 2004, Mr. Deng worked full time at Analog Devices Inc. He was in the corporate CAD R&D group and worked on the macro modeling part of the internal circuit simulator, which is used by all the designers in the company. After graduation he is going to join Synopsys, Inc., in Portland, Oregon.