KRYLOV'S METHODS IN FUNCTION SPACE FOR WAVEFORM RELAXATION

By

Wai-Shing Luk

A Thesis

submitted in partial fulfillment of the requirements

for the Degree of Doctor of Philosophy

Division of Computer Science and Engineering

The Chinese University of Hong Hong

June 1996

List of Symbols

$A, \ldots, Z$	matrices
$a, \ldots, z$	vectors
$\alpha, \beta, \ldots, \omega$	scalars
A^T	matrix transpose
A^{-1}	matrix inverse
$a_{i,j}$	matrix element
$\langle x, y \rangle$	inner product
$\|\cdot\|_2$	l_2-norm
$\|\mathcal{A}\|$	operator norm, $\sup_{x \neq 0} \|\mathcal{A}x\| / \|x\|$
$\dot{u}(t)$	first derivative with respect to t
$\mathbb{X}$	function space
$\mathcal{K}(\cdot)$	linear bounded operator
$\mathcal{K}^*(\cdot)$	adjoint operator of $\mathcal{K}(\cdot)$
$A \otimes B$	Kronecker product

Conventions Used in this thesis

D	diagonal matrix
L	lower triangular matrix
U	upper triangular matrix
$I, I^{n \times n}$	$n \times n$ identity matrix

Acknowledgements

First of all, I wish to express my gratitude to my advisors Omar Wing and Tony F. Chan. Prof. Wing provided much initial direction to my research. His expertise in circuit simulation is well-known and he has rich experiences in the area of waveform relaxation. Nevertheless, he has always given me great freedom in studying in other areas that seemed interesting to me. Prof. Tony Chan taught me the basics of parallel computing, Krylov subspace methods and domain decomposition when he was visiting at the Chinese University of Hong Kong.

I also wish to thank Prof. T.-C. Chen, my supervisor of my M.Phil research. I might not continue to pursue my Ph.D study without his encouragement.

The finite element package and the graph partitioning package that I used in this thesis were written respectively by Chun-Kit Nip and Wing-Kei Szeto under the supervision of Prof. Chan. I would like to thank both of them. I also wish to thank Prof. Franklin T. Luk, Prof. F.-Y. Chang and Prof. Y.-L. Jiang for their helpful suggestions to this work. I gratefully acknowledge the valuable assistance of many people on preliminary drafts of this thesis. In particular, I would like to thank my colleagues Man-Leung Wong, Yui-Wah Lee, Kei-Shiu Ho and Chi-Sing Leung for their insightful comments.

Wai-Shing Luk

May 28, 1996

Abstract

In this thesis, we analyze and illustrate the use of waveform methods for solving large sparse systems of ordinary differential equations (ODE). The methods can be viewed as an extension of classical methods to function spaces. The main algorithm that we focus on is waveform Krylov subspace method. This technique combines the idea of Krylov subspace, a method originally developed for solving general sparse linear algebraic systems, with waveform relaxation, an algorithm originally for solving very large differential equations arising from VLSI circuits. We show that waveform Krylov subspace methods give a better performance than waveform relaxation methods for tightly coupled systems.

The convergence behaviors of the waveform methods are investigated by numerical experiments on both sequential and parallel computers. The methods has been implemented and extensively tested on a MasPar massively parallel computer. In particular, we present the use of inexact ODE solver for waveform methods and show that it is well-suited for this type of machines.

We will also describe how to integrate the overlapped partitioning with the waveform Krylov subspace methods by domain decomposition technique. The subdomain solver acts as a preconditioner of waveform Krylov subspace methods. The resulting algorithm is the functional extension of an overlapping additive Schwarz method.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Many physical problems and engineering problems are modelled by differential equations. The numerical solution of these equations has been the subject of research activity long before the birth of digital computer. After the invention of digital computer, it plays a more important role in many applications. One of these applications that we will focus on is VLSI circuit simulation. Circuit simulation is a time-consuming task. In VLSI circuits, it is often that thousands of simultaneous differential equations needed to be solved in a reasonable time. Engineering requirement to the numerical algorithms is not just to be efficient, but also to be robust and reliable. In this thesis, we investigate the iterative approach for solving the circuit equations. One of the methods in this category is called waveform relaxation and has been studied for past ten years. We will address the convergence problems of this method for tightly coupled systems (which is well known) and suggest new methods to tackle this problem. The techniques that we develop will be applicable to not only circuit simulation, but can be also many applications that share the same numerical properties. The overview of the approach will be introduced step by step in the following sections in this chapter.

1.1 Functional Extension of Iterative Methods

There have been many iterative methods developed for solving large sparse linear and non-linear equations in recent years (see [38, 32] for reference). However, studies of iterative approaches to solution of a system of *ordinary differential equations* (ODE) or *differential-algebraic equations* (DAE) have been comparatively scarce. This is not really surprising because the standard numerical treatment takes the approach of discretizing the system in the time domain (we assumed that the independent variable is time for convenience of discussion) in the first step by the implicit integration methods, and the resulting algebraic equations at each time-step are solved by well-developed methods such as Gaussian elimination or LU decomposition. This approach is called *time-marching algorithm.* However, this approach becomes inefficient when the problem size is large and is not easy to be parallelized. Another approach, which this thesis will focus on, is to extend classical iterative methods to function spaces. This idea dates back in the early '50s in the study of iterative methods in Banach and Hilbert spaces; see the survey by Patterson [54]. Many classical methods were generalized to function spaces around and after this period. They include the Jacobi method, the Gauss-Seidel method, the successive over-relaxation method, the alternating direction method, the steepest descent method and the conjugate gradient method [54].

1.2 Applications in Circuit Simulation

Since the '80s, iterative methods in function spaces have found an important application, namely circuit simulation. In VLSI circuit simulation, the amount of work increases dramatically with the circuit size, and one has to solve a system of thousands of equations efficiently. In this case, the matrices are sparse and

well-suited to iterative approach. More important, in the problem of circuit simulation, the step-sizes are often kept small in time-marching approach in order to achieve accurate results. To tackle this problem, *waveform relaxation* methods (WR) was proposed for circuit simulation in 1982 [40], and are sometimes also referred to in the literature as *dynamic iteration* (for example [74, 72, 74]) and *Picard-Lindelöf iteration* (for example [49, 50, 51, 52, 47]). Those who are familiar with the classical iterative methods can imagine that the extended methods work similarly as the traditional ones, but the unknown is now a function with a bounded time interval (or simply called a *waveform*) instead of a single value, and the updating needs to solve an individual ODE equation instead of simply algebraic arithmatic. Compared with the time-marching approach, this approach can exploit the multi-rate behavior[1] of the circuit variables [79], since the individual ODE's can have their own time-steps. The theoretical analysis of waveform relaxation methods can be based on the contraction mapping and fixed point theorems in Banach spaces. In Chapter 2, we will review the methods in more detail. Waveform relaxation has been certified to be applicable to the simulation of MOS circuits, in which the Jacobian matrices are often diagonally dominant. However, for tightly coupled systems such as bipolar circuits, the convergence of waveform relaxation methods can be extremely slow. This thesis aims to search for the more robust techniques to tackle this problem.

1.3 Multigrid Acceleration

Recently there have been many publications on the technique of multigrid acceleration of waveform relaxation for solving parabolic partial differential equations (PDE) [75, 67, 71, 72, 70, 73, 68, 69, 74, 33]. Since the multigrid technique have been successfully used to solve elliptic problems, the generalized version

[1]That is, the circuit variables are changing at very different rates

is expected to have good performance for parabolic problems. However, the performance for applying this technique to the circuit problems is in question. Although some test problems in this thesis are in fact the parabolic problems, we are more interested in developing other techniques that are also applicable for circuit problems.

1.4 Why Hilbert Space?

In recent years, Krylov subspace methods have become popular over the relaxation methods in solving large linear algebraic systems [5]. It is natural to apply these techniques to the waveform relaxation methods. The idea is to extend Krylov subspace methods to function space. To do this, we need to look at a more specific space, namely the Hilbert space, where the concepts of *inner product* and *orthogonality* are introduced. For solving linear system of ODE's, new iterates on one waveform relaxation iteration are treated as a result of a linear operator applied to the old iterates [43]. The sequences of iterates are then accelerated by the generalized Krylov subspace methods. Recently, Lumsdaine *et al.* extended the *generalized minimal residual* method (GMRES) to accelerate the waveform relaxation methods [43]. The method was implemented for semiconductor device simulation. Three times faster than waveform relaxation was reported. This motivates us to consider this method seriously for circuit simulation. However, a drawback to the waveform GMRES method is that the amount of storage is huge. This is due to the fact that the whole discretized waveforms have to be stored in the waveform methods, and to the requirement that all computed orthogonal vectors have to be kept in the GMRES method. In Chapter 3, a range of waveform Krylov subspace methods are further developed. They use less amount of storage than the waveform GMRES method. The adjoint operator is derived for the waveform bi-conjugate gradient method and

the waveform quasi-minimal residual method by using the reverse ODE solving. The methods are then applied to solve tightly coupled circuit problems and the unstructured grid partial differential equations (Section 3.5).

1.5 Parallel Implementation

One of the advantages of waveform relaxation is that it can be implemented efficiently on parallel computers. In fact, there have been many parallel waveform relaxation programs implemented on this type of machines, for example [59, 71]. Since the Krylov subspace methods can be implemented efficiently on parallel computers, their functional counterparts are expected to be also suitable for parallel implementation. In this thesis, we give a parallel implementation of these new methods. In Chapter 4, we present the parallel implementation issues on a MasPar massively parallel computer. The use of inexact ODE solver is given in Chapter 5. Since the conventional ODE solvers are inherently sequential, the inexact ODE solver is proposed by taking time points from only previous waveform iteration for time integration. As a result, this method is truly massively parallel, as the equation is completely unfolded both in system and in time.

1.6 Domain Decomposition

Overlapped partitioning technique has been applied to the waveform relaxation methods for solving large scale initial value problems that arising from VLSI circuit simulation [64, 76, 3]. The motivation is mainly to improve robustness as well as convergence. Particularly for tightly coupled systems such as bipolar circuits, the convergence of the naive waveform relaxation methods can be very poor. However, if we group the tightly coupled component together and the iterative process are only applied among the inter-subdomain, the convergence

will be improved [79]. Although the optimal partitioning for the criteria such as minimizing the spectral radius is believed to be NP-hard, some good heuristic algorithms have been reported [48, 3].

Another good reason for partitioning has to do with parallel or distributed processing on MIMD (*Multiple Instruction, Multiple Data*) computers or cluster workstations. A large circuit is divided into several subcircuits and each subproblem is distributed to a different processor. In Chapter 6, we argue that the overlapped partitioning technique can be applied to waveform Krylov subspace methods equally well as waveform relaxation methods. The idea is borrowed from the techniques in solving PDE problems, i.e., domain decomposition technique and preconditioning. The resulting algorithm is similar to the overlapped Schwarz method in solving the PDE problems.

1.7 Contributions of This Thesis

The main contributions of this thesis are:

- By using adjoint operator, we extend the bi-conjugate gradient method and the quasi minimal residual method in function space.
- The convergence behaviors of five of the previously untried waveform Krylov subspace methods, together with the waveform generalized minimal residual method, are investigated. Numerical results showed that convergence was achieved for some tightly coupled systems where the conventional waveform relaxation method would fail. Also, we observe that the convergence behaviors of these methods are similar to their non-functional counterparts.
- The parallel implementation of waveform Krylov subspace methods on a massively parallel computer is developed to demonstrate the effectiveness

of these methods in parallel environments. The use of inexact ODE solver is proposed which is shown to be well-suited for massively parallel machines.

- We propose using domain decomposition in order to integrate the overlapped partitioning techniques and waveform Krylov subspace methods.

1.8 Outlines of the Thesis

In this thesis, we will concentrate on the design and analysis of algorithms for linear ODE's. We will pay attention to the improvement in the system domain, such as using Krylov subspace methods and domain decomposition to accelerate the waveform relaxation methods, rather than in the time domain. Therefore, the development of advanced ODE methods for time domain, such as implicit Runge-Kutta methods and wavelet methods that have newly emerged will not be considered. The variables are real by default unless they are specified explicitly. The rest of this thesis is organized as follows:

Chapter 2 presents a review of waveform relaxation methods, including the corresponding convergence theory. The basic concepts in functional analysis and linear algebra that are used in this thesis will be introduced.

Chapter 3 starts with a review of Krylov subspace methods for linear algebraic systems. Their functional counterparts will then be introduced. In particular, we study the use of adjoint operator and the difficulty in using this operator will also be addressed. The convergence behaviors are investigated via numerical experiments.

Chapters 4 and 5 address the implementation issues of waveform Krylov subspace methods on a massively parallel computer, named DECmpp 12000/Sx

or MasPar, for solving an unstructured grid problem. We show that the use of inexact ODE solver is most suitable for this type of machines.

Chapter 6 presents some new ideas on how to integrate the overlapped partitioning technique and the waveform Krylov subspace methods by domain decomposition setting and preconditioning.

Chapter 7 concludes the thesis by presenting our main conclusions.

Appendix A contains pseudo codes of waveform Krylov subspace methods referred to in the text.

Appendix B contains an overview of recusive spectral bisection method that we have used in the experiment described in Chapter 6.

Chapter 2

Waveform Relaxation Methods

In this chapter, we review the waveform relaxation methods for solving a linear system of ordinary differential equations (ODE). The main purpose of this chapter is to provide the basic concepts of waveform relaxation and to introduce some notations. The basic convergence analysis of waveform relaxation is reviewed. The relation between waveform relaxation methods for ordinary differential equations and relaxation methods for linear operator equations in Banach space will also be described. This is the key to understanding the methods in the following chapters.

We exclude the discussion of the methods for non-linear systems. It is often that the non-linear equations are solved using the linearization by Newton's method (and this will be shown at the end of Chapter 3). Therefore it seems useful to study the waveform relaxation methods for linear systems in greater depth. For a complete introduction of the methods, the reader may consult the classic book of White and Sangiovanni-Vincentelli [79]. For more information of recent development on these methods, the reader may refer to the references at the end of this chapter.

2.1 Basic Idea

Consider the following linear time-varying system of ODE's with n unknown functions:

$$B(t)\dot{y}(t) + A(t)y(t) = f(t), \qquad y(t_0) = y_0, \qquad t \in [t_0, t_m] \tag{2.1}$$

where f and y are vector valued functions and t is a real variable. We refer t as *time* for convenience. $A(t)$ and $B(t)$ are square matrices and are usually large and sparse in our discussion. The cases of rectangular coefficients will not be considered. Moreover, $\dot{y}(t)$ denotes the derivative of $y(t)$ with respect to t. This is an initial value problem and the initial value y_0 is given at time t_0. The *Time-marching* approach is a standard technique for finding the numerical solution of Equation (2.1). In this approach, the system of ODE's is firstly discretized in the time domain by an implicit integration method such as Backward Euler method, and then the resulting linear algebraic equations are solved at each time-step directed from time zero. For example, if we approximate $\dot{y}(t_i)$ by $(y(t_i) - y(t_{i-1}))/\triangle t_i$ where $\triangle t_i \equiv t_i - t_{i-1}$ and $t_k > t_j$ for $k > j$, the standard Backward Euler method can be written as solving the following equation:

$$B(t_i)\left(\frac{y(t_i) - y(t_{i-1})}{\triangle t_i}\right) + A(t_i)y(t_i) = f(t_i)$$

for $i = 1, 2, \ldots, m$, where m is the number of discretized time points. The sparse Gaussian elimination is then used to solve the above linear algebraic equations at each time point. We refer the reader to the book of Brenan *et al.* [9] for further discussion on time-marching algorithms for solving initial-value problems.

However, it is well-known that Gaussian elimination is inefficient when the problem size is large. In the waveform relaxation methods, the system of equations are decomposed into individual ODE equations with single unknown by an iterative process. In matrix notation, waveform relaxation can be written as:

$$M_B(t)\dot{y}^{(k)}(t) + M_A(t)y^{(k)}(t) = N_B(t)\dot{y}^{(k-1)}(t) + N_A(t)y^{(k-1)}(t) + f(t), \quad (2.2)$$

where $A(t) = M_A(t) - N_A(t), B(t) = M_B(t) - N_B(t)$. The k-th iteration is denoted by $^{(k)}$. The choice of the splitting matrices is the same as that in classical relaxation methods. We refer the reader to the book of Hackbusch [32] for the general discussion of classical relaxation methods. For example in waveform Jacobi method, $M_A(t)$ and $M_B(t)$ are the diagonal parts of $A(t)$ and $B(t)$ respectively. Similarly in waveform Gauss-Seidel method, $M_A(t)$ and $M_B(t)$ are the lower triangular parts of $A(t)$ and $B(t)$ respectively. As in the classical relaxation methods for solving linear algebraic systems, $M_B(t)$, $M_A(t)$, $N_B(t)$ and $N_A(t)$ are not formed explicitly in actual implementation. They are used for illustrating and analyzing the algorithms only. Algorithm 2.1 and Algorithm 2.2 show the waveform Jacobi method (WJAC) and the waveform Gauss-Seidel method (WGS) respectively in actual implementation.

For example, consider the linear differential equation:

$$\begin{bmatrix} 1 & -2 & 0 \\ 0 & 5 & -6 \\ -7 & 0 & 9 \end{bmatrix} \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \\ \dot{y}_3 \end{bmatrix} + \begin{bmatrix} 10 & 0 & -3 \\ -4 & 11 & 0 \\ 0 & -8 & 12 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 13 \\ 14 \\ 15 \end{bmatrix}.$$

The waveform Jacobi method for this equation is written as:

$$\begin{aligned} \dot{y}_1^{(k)}(t) + 10y_1^{(k)}(t) &= 13 + 2\dot{y}_2^{(k-1)}(t) + 3y_3^{(k-1)}(t), \\ 5\dot{y}_2^{(k)}(t) + 11y_2^{(k)}(t) &= 14 + 6\dot{y}_3^{(k-1)}(t) + 4y_1^{(k-1)}(t), \\ 9\dot{y}_3^{(k)}(t) + 12y_3^{(k)}(t) &= 15 + 7\dot{y}_1^{(k-1)}(t) + 8y_2^{(k-1)}(t), \end{aligned}$$

and the waveform Gauss-Seidel method is written as:

$$\dot{y}_1^{(k)}(t) + 10y_1^{(k)}(t) = 13 + 2\dot{y}_2^{(k-1)}(t) + 3y_3^{(k-1)}(t),$$

Choose $y^{(0)}(t)$ for $t \in [t_0, t_m]$
for $k = 1, 2, \ldots,$ *max-iter*
  for $i = 1, 2, \ldots, n$
    solve

$$b_{i,i}\dot{y}_i^{(k)}(t) + a_{i,i}y_i^{(k)}(t) = f_i(t) - \sum_{j \neq i} b_{i,j}\dot{y}_j^{(k-1)}(t) - \sum_{j \neq i} a_{i,j}y_j^{(k-1)}(t)$$

    with $y_i^{(k)}(0) = y_{i,0}$
  end
  check convergence; continue if necessary
end

Algorithm 2.1: The Waveform Jacobi Method (WJAC)

Choose $y^{(0)}(t)$ for $t \in [t_0, t_m]$
for $k = 1, 2, \ldots,$ *max-iter*
  for $i = 1, 2, \ldots, n$
    $g_i(t) := \sum_{j<i} b_{i,j}\dot{y}_j^{(k)}(t) + \sum_{j>i} b_{i,j}\dot{y}_j^{(k-1)}(t)$
    $h_i(t) := \sum_{j<i} a_{i,j}y_j^{(k)}(t) + \sum_{j>i} a_{i,j}y_j^{(k-1)}(t)$
    solve

$$b_{i,i}\dot{y}_i^{(k)}(t) + a_{i,i}y_i^{(k)}(t) = f_i(t) - g_i(t) - h_i(t)$$

    with $y_i^{(k)}(0) = y_{i,0}$
  end
  check convergence; continue if necessary
end

Algorithm 2.2: The Waveform Gauss-Seidel Method (WGS)

$$
\begin{aligned}
5\dot{y}_2^{(k)}(t) + 11y_2^{(k)}(t) &= 14 + 6\dot{y}_3^{(k-1)}(t) + 4y_1^{(k)}(t), \\
9\dot{y}_3^{(k)}(t) + 12y_3^{(k)}(t) &= 15 + 7\dot{y}_1^{(k)}(t) + 8y_2^{(k)}(t).
\end{aligned}
$$

Each decomposed equation is solved individually. Waveform relaxation was originally proposed by Lelarasmee *et al.* [40] and have been widely studied in electrical and electronic engineering community for the past ten years.

As in the classical relaxation method, the individual equations can be solved in parallel fashion in the waveform Jacobi method and those in the waveform Gauss-Seidel method are often solved in serial fashion. The parallelization of the waveform Gauss-Seidel method can be done by multi-color technique. One of such implementation is presented in Chapter 4. Note that the convergent rate of the waveform Gauss-Seidel method depends also on the *ordering* of the equations. In circuit simulation, the ordering usually follows the signal flow of circuits in order to enhance the rate of convergence. For example MOS circuits often exhibit a kind of *unidirectional coupling.* The source-to-drain current of an MOS transistor is controlled by the gate voltage, but the gate voltage is almost independent of the drain and the source voltage. Therefore, if we update the gate voltage before the drain or source voltage, the convergent rate will be improved.

There is no doubt that waveform relaxation works in continuous-time, as it is supported by the theory of functional analysis on continuous functions (see also Section 2.4.1). However, practically, they are solved (approximated) by a numerical integration method, such as *Backward Euler* or *multi-step method* [79]. The method is then called *discretized waveform relaxation.* Sometimes we call discretized waveform relaxation simply waveform relaxation if it is understood in the context. Mostly only implicit integration methods are considered in discretized waveform relaxation. There is an exception that recently the use of *Runge-Kutta* method, which is an explicit method, was studied by Bellen and

Zennaro [7].

One of the advantages of waveform relaxation over the standard time marching algorithm is that multi-rate behavior and latency can be exploited [79], especially in simulation of digital circuits. In discretized waveform relaxation, the individual ODE can be integrated with their own time-steps and the alignments between each pair of discretized waveforms while updating can be done by interpolations. We refer to this as multi-rate integration. To maintain the same accuracy, the multi-rate integration needs less time-points than that of time marching algorithm and hence the total computation times are reduced (assume that interpolations are inexpensive compared with function evaluations).

2.2 Linear Operators between Banach Spaces

To study the waveform relaxation methods in continuous time, it is worthwhile to discuss first some notions concerning operators between *Banach spaces*. The contents of this section are covered by the books of Zeidler [80] and Piccinini et al. [55], or introductory functional analysis books. A Banach space generalizes the notion of $\mathbb{R}^n$ as a linear space with a length function. In our linear ODE problems, we can assume that all variables are in this space.

Let $\mathbb{X}$ and $\mathbb{Y}$ be two Banach spaces. A function $\mathcal{A}$: $\mathbb{X} \to \mathbb{Y}$ is called a linear operator if

$$\mathcal{A}(\alpha x_1 + \beta x_2) = \alpha \mathcal{A}(x_1) + \beta \mathcal{A}(x_2), \qquad \forall x_1, x_2 \in \mathbb{X},$$

where α and β are real scalars (or complex if the spaces are over the complex field). A *linear combination* of vectors $x_1, x_2, \ldots, x_m$ of $\mathbb{X}$ is an expression of the form $\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_m x_m$, where $\alpha_1, \alpha_2, \ldots, \alpha_m$ are scalars. The set of all linear combination of vectors $x_1, x_2, \ldots, x_m$ is called *span* and is denoted by

$$\text{span}\{x_1, x_2, \ldots, x_m\}.$$

This forms a linear subspace of $\mathbb{X}$. One important linear subspace named *Krylov subspace* will be discussed in Chapter 3.

A *norm* on $\mathbb{X}$, denoted by $\|\cdot\|$, is a function from $\mathbb{X}$ into $[0, \infty)$ that has the following properties:

1. $\|x\| = 0$ if and only if $x = 0$ for $x \in \mathbb{X}$;
2. $\|\alpha x\| = |\alpha| \cdot \|x\|$ for $\alpha \in \mathbb{R}$ and $x \in \mathbb{X}$;
3. $\|x + y\| \leq \|x\| + \|y\|$ for $x, y \in \mathbb{X}$.

We consider the linear operators that are continuous at every point x of $\mathbb{X}$. Recall that the function space $L(\mathbb{X}, \mathbb{Y})$ of continuous linear operators from $\mathbb{X}$ to $\mathbb{Y}$ is a Banach space under the operations

$$(\mathcal{A} + \mathcal{B})x = \mathcal{A}x + \mathcal{B}x$$
$$(\alpha\mathcal{A})x = \alpha(\mathcal{A}x)$$

and the norm

$$\|\mathcal{A}\| = \sup_{x \neq 0} \frac{\|\mathcal{A}x\|}{\|x\|} = \sup_{\|x\|=1} \|\mathcal{A}x\|.$$

$L(\mathbb{X}, \mathbb{X})$ are denoted by $L(\mathbb{X})$ for simplicity. The Banach fixed-point theorem is the fundamental convergence theorem for a broad class of iterative methods.

Theorem 2.2.1 *(The fixed-point theorem) Let $\mathbb{X}$ be a Banach space and let an operator $\mathcal{F}$: $\mathbb{X} \to \mathbb{X}$ be a contraction mapping such that*

$$\|\mathcal{F}u - \mathcal{F}v\| \leq \alpha\|u - v\| \qquad \forall u, v \in \mathbb{X}$$

and $0 \leq \alpha < 1$. The iteration $y^{(k+1)} = \mathcal{F}y^{(k)}$ has a unique fixed point y^* such that $y^* = \mathcal{F}y^*$.

A proof of this theorem can be found in [80, pp. 18–21]. Note that the operator $\mathcal{F}$ does not need to be linear.

The *spectrum* of a continuous linear operator $\mathcal{A}$, denoted as $\sigma(\mathcal{A})$, is the set of numbers $\lambda \in \mathbb{C}$ for which the *resolvent operator* $(\lambda I - \mathcal{A})^{-1}$ does not exist in $L(\mathbb{X})$. In case that $\mathcal{A}$ is a matrix in $\mathbb{R}^{n\times n}$, the elements of the spectrum are just the eigenvalues of that matrix. The following theorem is useful for the discussion of the convergence analysis of waveform relaxation:

Theorem 2.2.2 *(The spectral radius theorem). Let $\mathbb{X}$ be a Banach space and $\mathcal{A} \in L(\mathbb{X})$. If $\lambda \in \sigma(\mathcal{A})$, then $|\lambda| \leq \|\mathcal{A}\|$.*

A proof of this theorem can be found in [55, Theorem 1.13]. Define the spectral radius of the operator $\mathcal{A}$, denoted by $\rho(\mathcal{A})$, as $\sup\{|\lambda|, \lambda \in \sigma(\mathcal{A})\}$. In the following section, the relation between waveform relaxation methods and the linear operator theory just described will be presented.

2.3 Waveform Relaxation Operators for ODE's

Equation (2.1) can be viewed as a linear mapping from $y(t)$ to $f(t)$. Let $\mathcal{A}$ be a linear operator for such mapping. Therefore, we have:

$$(\mathcal{A}y)(t) = f(t), \tag{2.3}$$

where $\mathcal{A}$ is a differential operator and is equal to $(B(t)\frac{d}{dt} + A(t))$. Equation (2.3) is in the form highly similar to the linear algebraic system $Ax = b$. Therefore, it seems reasonable to apply methods of linear algebra to this equation, as the waveform relaxation does. Let $\mathcal{A} = \mathcal{M} - \mathcal{N}$. Assume that $\mathcal{M}$ is invertible. Let $\mathcal{K} = \mathcal{M}^{-1}\,\mathcal{N}$. From the fixed-point theorem, the iteration

$$y^{(k)}(t) = (\mathcal{K}y^{(k-1)})(t) + \psi(t)$$

where $\psi(t) = (\mathcal{M}^{-1}f)(t)$, has a unique fixed point $y^*(t)$ such that $(\mathcal{A}y^*)(t) = f(t)$ if it is a contraction. Note that $\mathcal{K}$ is a linear integral-differential operator in

the context of waveform relaxation, and is called *waveform relaxation operator.* The operators $\mathcal{M}$ and $\mathcal{N}$ for Equation (2.2) is equal to $(M_B(t)\frac{d}{dt} + M_A(t))$ and $(N_B(t)\frac{d}{dt} + N_A(t))$. In case of $B(t) = I$, Equation (2.1) becomes

$$\dot{y}(t) + A(t)y(t) = f(t), \qquad y(t_0) = y_0, \qquad t \in [t_0, t_m]. \tag{2.4}$$

Then $\mathcal{K}$ can be expressed explicitly as:

$$(\mathcal{K}u)(t) = \int_{t_0}^{t} \Phi(t,s)N_A(s)u(s)\, ds,$$

where $\Phi(t, t_0)$ is the *transition matrix* for $\dot{u}(t) = -M_A(t)u(t)$ [10, pp.38-40]. The transition matrix satisfies the following property:

$$\frac{d}{dt}\Phi(t,t_0) = -M_A(t)\Phi(t,t_0) \qquad \Phi(t_0,t_0) = I.$$

$\psi(t)$ can be expressed as:

$$\psi(t) = \Phi(t,t_0)y_0 + \int_{t_0}^{t} \Phi(t,s)f(s)\, ds.$$

Furthermore, if $A(t)$ is time invariant, Equation (2.4) can be rewritten as:

$$\dot{y}(t) + Ay(t) = f(t), \qquad y(t_0) = y_0, \qquad t \in [t_0, t_m] \tag{2.5}$$

the transition matrix can be further simplified as:

$$\Phi(t,s) = e^{-(t-s)M},$$

and the solution of Equation (2.2) can be rewritten as:

$$y^{(k)}(t) = e^{-tM}y_0 + \int_{t_0}^{t} e^{-(t-s)M}(Ny^{(k-1)}(s) + f(s))\, ds.$$

Returning to the general case, we classify waveform relaxation methods in terms of operators $\mathcal{M}$ and $\mathcal{N}$. Let $D_K(t)$, $-L_K(t)$ and $-U_K(t)$ represent the diagonal, strictly lower-triangular and strictly upper-triangular parts of matrix $K(t)$ respectively, three popular relaxation methods in function space are defined as:

Waveform Jacobi method (WJAC):

$$\mathcal{M} = D_B(t)\tfrac{d}{dt} + D_A(t) \text{ and}$$
$$\mathcal{N} = (L_B(t) + U_B(t))\tfrac{d}{dt} + (L_A(t) + U_A(t))$$

Waveform Gauss-Seidel method (WGS):

$$\mathcal{M} = (D_B(t) - L_B(t))\tfrac{d}{dt} + (D_A(t) - L_A(t)) \text{ and}$$
$$\mathcal{N} = U_B(t)\tfrac{d}{dt} + U_A(t)$$

Waveform Successive Overrelaxation method (WSOR):

$$\mathcal{M} = (D_B(t) - \omega L_B(t))\tfrac{d}{dt} + (D_A(t) - \omega L_A(t)) \text{ and}$$
$$\mathcal{N} = (\omega U_B(t) + (1-\omega)D_B(t))\tfrac{d}{dt} + (\omega U_A(t) + (1-\omega)D_A(t))$$

where ω is a scalar called *relaxation parameter*. As in the classical SOR method, the WSOR method defined above can be derived by applying the extrapolation to the waveform Gauss-Seidel method:

$$y^{(k)}(t) = y^{(k-1)}(t) + \omega(\bar{y}^{(k)}(t) - y^{(k-1)}(t))$$

where $\bar{y}$ denotes a Gauss-Seidel iterate. If $\omega = 1$, the waveform SOR method is simplified back to the waveform Gauss-Seidel method. Algorithm 2.3 illustrates the actual implementation of the WSOR method. Note that the definition of the waveform SOR method here is different from that in [46]. In [46], the over-relaxation is not applied to $B(t)$ since only the case of $B(t) = I$ was considered in that paper.

In [43], a different type of SOR was proposed for accelerating waveform relaxation. The extrapolation process is through a convolution type operator:

$$y^{(k)}(t) = y^{(k-1)}(t) + (\omega \star (\bar{y}^{(k)} - y^{(k-1)}))(t)$$

where $w(t)$ is a scalar function. The convolution product is given by

$$(w \star h)(t) = \int_{t_0}^{t} \omega(\tau)h(t-\tau)\,d\tau.$$

Choose $y^{(0)}(t)$ for $t \in [t_0, t_m]$
for $k = 1, 2, \ldots,$ *max-iter*
  for $i = 1, 2, \ldots, n$
    $g_i(t) := \sum_{j<i} b_{i,j}\dot{y}_j^{(k)}(t) + \sum_{j>i} b_{i,j}\dot{y}_j^{(k-1)}(t)$
    $h_i(t) := \sum_{j<i} a_{i,j}y_j^{(k)}(t) + \sum_{j>i} a_{i,j}y_j^{(k-1)}(t)$
    solve

$$b_{i,i}\dot{v}_i^{(k)}(t) + a_{i,i}v_i^{(k)}(t) = f_i(t) - g_i(t) - h_i(t)$$

    with $v_i^{(k)}(0) = y_{i,0}$
  end
  $y^{(k)}(t) := y^{(k-1)}(t) + \omega(v^{(k)}(t) - y^{(k-1)}(t))$
  check convergence; continue if necessary
end

Algorithm 2.3: The Waveform Successive Overrelaxation Method (WSOR)

It can be viewed as the relaxation parameter applied in frequency domain when we take the Laplace transform to both sides of the equation. As in the conventional SOR method, both waveform SOR methods suffers from the difficulty in determining the optimal value of the relaxation parameter.

2.4 Convergence Analysis

The convergence analysis of waveform methods can be divided into continuous-time analysis and discrete-time analysis. Continuous-time analysis assumes that the individual ODE are solved exactly (or analytically). So it is independent of the integration methods and gives us a more general statements of the convergence. The discrete-time analysis is basically for discretized waveform relaxation algorithm and the numerical integration method is usually restricted to the linear multi-step method.

2.4.1 Continuous-time Convergence Analysis

The theoretical continuous-time analysis of waveform relaxation method is based on the contraction mapping and fixed point theorems in Banach space. Recall that $\mathcal{K}$ is the waveform relaxation operator mentioned in the previous section. From the fixed-point theorem, the waveform iteration

$$y^{(k)}(t) = (\mathcal{K}y^{(k-1)})(t) + \psi(t),$$

will converge to the unique solution y^* of Equation (2.3) with $y^{(0)}$ arbitrary if it is a contraction. According to the previous section, the necessary and sufficient condition for convergence is that the spectral radius of $\mathcal{K}$ is less than one. In particular, if $B(t) = I$, the spectral radius is equal to zero and $\mathcal{K}$ is always a contraction [46].

The error is given by the estimate

$$\|y^{(k+1)}(t) - y^*(t)\| \leq \|\mathcal{K}\| \, \|y^{(k)}(t) - y^*(t)\|.$$

Let $e_k(t) = y^*(t) - y^{(k)}(t)$. The reduction factors $\|e_{k+1}(t)\|/\|e_k(t)\|$ tend to the spectral radius of $\mathcal{K}$ [32, pp. 51]:

$$\lim_{k\to\infty} \frac{\|e_{k+1}\|}{\|e_k\|} = \rho(\mathcal{K}).$$

Therefore, the spectral radius can be used to represent the *asymptotic* convergent rate of the iterations. The following theorem gives a sufficient condition that the waveform Jacobi method and the waveform Gauss-Seidel method will converge.

Theorem 2.4.1 *([79, Theorem 4.1]) The waveform Jacobi method and the waveform Gauss-Seidel method converges if $B(t)$ is strictly diagonally dominant for all $t \in [t_0, t_m]$.*

Note that the convergence does not depend on $A(t)$. The matrix $B(t)$ is often a capacitance matrix for circuit simulation. It is known that if one node

of each capacitors is connected to the ground node, the resulting capacitance matrix is strictly diagonally dominant. Then the convergence of the waveform Jacobi method and the waveform Gauss-Seidel method is guaranteed for this type of circuits assuming that the step-size is small enough.

2.4.2 Discrete-time Convergence Analysis

In discrete-time analysis, we assume that each individual ODE equation is solved by a linear multi-step method. Here we only give the main results of the fixed-rate analysis. In the fixed-rate analysis, we assume that the time frames of all discretized waveforms are aligned. For the discussion of the multi-rate analysis, see [79]. Let the time interval be discretized at time points $t_0 < t_1 < t_2 < \cdots < t_m$ and let $\triangle t_j = t_j - t_{j-1}$. The standard discretized waveform iteration is formulated as:

$$\begin{aligned} &\frac{1}{\triangle t_j}\sum_{i=0}^{l}\alpha_i M_B(t_{j-i})y^{(k)}(t_{j-i}) + \sum_{i=0}^{l}\beta_i M_A(t_{j-i})\,y^{(k)}(t_{j-i}) = \\ &\frac{1}{\triangle t_j}\sum_{i=0}^{l}\alpha_i N_B(t_{j-i})y^{(k-1)}(t_{j-i}) + \sum_{i=0}^{l}\beta_i N_A(t_{j-i})\,y^{(k-1)}(t_{j-i}) + \sum_{i=0}^{l}\beta_i f(t_{j-i}), \end{aligned} \tag{2.6}$$

for $1 \leq j \leq m$.

Recall that the following three integration methods are frequently used:

- **Forward Euler method:** $\alpha = (1, -1)$ and $\beta = (0, 1)$,
- **Backward Euler method:** $\alpha = (1, -1)$ and $\beta = (1, 0)$,
- **Crank–Nicolson method:** $\alpha = (1, -1)$ and $\beta = (0.5, 0.5)$.

The equation above is totally unfolded and can be embedded in a huge sparse linear algebraic system. The convergence analysis is hence not different from the traditional one. The main theorem of convergence is given in [61]:

Theorem 2.4.2 *The waveform relaxation that corresponds to Equation (2.6) converges if and only if the following condition holds:*

$$\max_{1\le j\le m}\left(\rho((\frac{\alpha_0}{\beta_0}\frac{M_B(t_j)}{\triangle t_j}+M_A(t_j))^{-1}(\frac{\alpha_0}{\beta_0}\frac{N_B(t_j)}{\triangle t_j}+N_A(t_j)))\right)<1. \qquad (2.7)$$

Proof : We start the proof from Equation (2.6). Define the error $e^{(k)}(t_j)\equiv y^{(k)}(t_j)-y^{(k-1)}(t_j)$. It can easily be shown that

$$\frac{1}{\triangle t_j}\sum_{i=0}^{l}\alpha_i M_B(t_{j-i})e^{(k)}(t_{j-i})+\sum_{i=0}^{l}\beta_i M_A(t_{j-i})\,e^{(k)}(t_{j-i})=$$
$$\frac{1}{\triangle t_j}\sum_{i=0}^{l}\alpha_i N_B(t_{j-i})e^{(k-1)}(t_{j-i})+\sum_{i=0}^{l}\beta_i N_A(t_{j-i})\,e^{(k-1)}(t_{j-i})$$

for $1\le j\le m$. The above system of equations can be rewritten in matrix form as

$$\begin{bmatrix} G_{1,1} & & & \\ G_{2,1} & G_{2,2} & & \\ \vdots & \vdots & \ddots & \\ G_{m,1} & G_{m,2} & \cdots & G_{m,m}\end{bmatrix}\begin{bmatrix} e^{(k)}(t_1)\\ e^{(k)}(t_2)\\ \vdots\\ e^{(k)}(t_m)\end{bmatrix}=\begin{bmatrix} H_{1,1} & & & \\ H_{2,1} & H_{2,2} & & \\ \vdots & \vdots & \ddots & \\ H_{m,1} & H_{m,2} & \cdots & H_{m,m}\end{bmatrix}\begin{bmatrix} e^{(k-1)}(t_1)\\ e^{(k-1)}(t_2)\\ \vdots\\ e^{(k-1)}(t_m)\end{bmatrix}$$

or denoted as

$$G\,\tilde{e}^{(k)}=H\,\tilde{e}^{(k-1)}$$

where

$$\tilde{e}^{(k)} = (e^{(k)}(t_1)^T, e^{(k)}(t_2)^T,\ldots,e^{(k)}(t_m)^T)^T$$

$$G_{p,q} = \begin{cases} \frac{\alpha_0}{\triangle t_p}M_B(t_p)+\beta_0 M_A(t_p) & \text{if } p=q,\\ \frac{\alpha_{p-q}}{\triangle t_q}M_B(t_q)+\beta_{p-q}M_A(t_q) & \text{if } q<p\le q+l,\\ 0 & \text{otherwise.}\end{cases}$$

$$H_{p,q} = \begin{cases} \frac{\alpha_0}{\triangle t_p}N_B(t_p)+\beta_0 N_A(t_p) & \text{if } p=q,\\ \frac{\alpha_{p-q}}{\triangle t_q}N_B(t_q)+\beta_{p-q}N_A(t_q) & \text{if } q<p\le q+l,\\ 0 & \text{otherwise.}\end{cases}$$

Because G is a block lower triangular matrix, G^{-1} is also a block triangular matrix with diagonal submatrices $(G_{p,p}^{-1})$. Since H is a block lower triangular matrix, consequently $(G^{-1}H)$ is a block lower triangular matrix with diagonal submatrices $(G_{p,p}^{-1}H_{p,p})$. We know that the eigenvalues of a lower triangular matrix are determined by the eigenvalues of its diagonal submatrices. Therefore, we have

$$\begin{aligned}\rho(G^{-1}H) &= \max_{1\le p\le m} \rho(G_{p,p}^{-1}H_{p,p}) \\ &= \max_{1\le p\le m} \rho\left((\frac{\alpha_0}{\beta_0}\frac{M_B(t_p)}{\triangle t_p} + M_A(t_p))^{-1}(\frac{\alpha_0}{\beta_0}\frac{N_B(t_p)}{\triangle t_p} + N_A(t_p))\right).\end{aligned}$$

Since the equation converges if and only if $\rho(G^{-1}H) < 1$, this gives the theorem. □

Note that the spectral radius is independent of the time interval. The convergence characteristic is dominated by $B(t)$ if the time-step is small enough. Therefore, if $B(t)$ is strictly diagonally dominant, the waveform Jacobi method and the waveform Gauss-Seidel method will converge properly by using small time-steps. Waveform relaxation has been certified to be successfully applied to the simulation of MOS circuits, in which the Jacobian matrices are often diagonally dominant. However, for tightly coupled systems such as bipolar circuits, the convergence of the waveform relaxation methods can be extremely slow. This deficiency prohibits the use of waveform relaxation methods for general circuit simulation. In the next chapter, we will present a more sophisticated technique, namely Krylov subspace method, that potientially accelarates waveform relaxation.

2.5 Further references

Waveform relaxation have been successfully applied to MOS circuit simulation. Other applications presently being investigated include electric power system analysis [23, 21, 22], chemical process [62, 63], semiconductor device simulation [60, 42, 78] and transmission line simulation [36, 8, 57, 58, 2, 1, 15, 16, 20, 18, 17, 19, 45, 39]. Nevanlinna investigated quite extensively on the convergence behavior of waveform relaxation [46, 49, 50, 51, 52, 47]. Recently the spectra and the pseudo-spectra of waveform relaxation operators are studied in [44]. Vandewalle *et al.* worked on the use of waveform relaxation for parabolic partial differenetial equations with multigrid acceleration [75, 67, 71, 72, 70, 73, 68, 69, 74, 33, 34, 35]. The method can be viewed as the functional extension of classical multigrid relaxation in Banach spaces. In [43], waveform Krylov subspace methods were proposed that we will investigate deeply in the rest of this thesis.

Chapter 3

Waveform Krylov Subspace Methods

In the previous chapter, we introduced the basic idea of the waveform relaxation methods. In recent years, Krylov subspace methods have become popular over relaxation methods in solving large linear algebraic systems. It is natural to apply these techniques to accelerate the waveform relaxation methods. In this chapter, we firstly review the basic concepts of Krylov subspace methods and waveform Krylov subspace methods in §3.1 and §3.3 respectively. In addition, we generalize a range of Krylov subspace methods described in [5] using a similar technique by Lumsdaine *et al* [43]. The methods include the *waveform conjugate gradient* method (WCG), the *waveform conjugate gradient squared* method (WCGS), the *waveform bi-conjugate gradient stabilized* method (WBiCG-Stab) and the *waveform generalized minimal residual* method (WGMRES). In §3.4, we derive an adjoint operator that is used in the *waveform bi-conjugate gradient* method (WBiCG) and the *waveform quasi-minimal residual* method (WQMR). The difficulty for developing the adjoint operator will be addressed. Finally,

numerical results on a sequential machine are given in §3.5. The parallel implementation on a MasPar computer will be given later in Chapters 4 and 5.

3.1 Overview of Krylov Subspace Methods

We briefly describe the basic concepts of Krylov subspace methods for linear algebraic systems $Ax = b$, where A is an $n \times n$ matrix. This field of research is active and new methods are still emerging. In this section, we introduce the most popular ones that are described in Barrett *et al.* [5]. For further details, the reader may consult the references therein.

Consider the linear relaxation iteration

$$Mx^{(m+1)} = Nx^{(m)} + b \qquad m = 0, 1, \ldots, \tag{3.1}$$

where $A = M - N$. This iteration can be rewritten as

$$x^{(m+1)} = M^{-1}(Nx^{(m)} + b) = x^{(m)} + M^{-1}(b - Ax^{(m)}).$$

Let $r_0 = b - Ax^{(0)}$. It follows by induction that $x^{(m)}$ can be expressed as

$$x^{(m)} = x^{(0)} + \sum_{i=0}^{m-1} \alpha_i (M^{-1}A)^i M^{-1} r_0, \tag{3.2}$$

where α_i is a constant. Let us say $M = I$. This refers to the Richardson iteration to Equation (3.1). The equation above is then simplified as:

$$x^{(m)} = x^{(0)} + \sum_{i=0}^{m-1} \alpha_i A^i r_0. \tag{3.3}$$

Hence, $x^{(m)}$ is constructed by $x^{(0)}$ plus a vector from the space spanned by $A^i r_0$, $i = 0, 1, \ldots, m-1$. Denote the Krylov subspace

$$K_m(A; r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{m-1} r_0\}.$$

Equation (3.3) can be rewritten as

$$x^{(m)} = x^{(0)} + z, \qquad \text{with } z \in K_m(A; r_0).$$

In the classical relaxation iteration, the values of α_i are predetermined, independent of A. Krylov subspace methods differ from relaxation methods in that the values of α_i change at each iteration in a way that tries to meet two criteria. One is orthogonality. The residual $r^{(m)}$ ($\equiv b - Ax^{(m)}$) is either orthogonal to $K_m(A; r_0)$ or $K_m(A^T; r_0)$. The other one is the minimization of residuals. For example, in the generalized minimal residual method (GMRES), the orthonormal basis $(v_1, v_2, \ldots, v_m)$ for $K_m(A; r_0)$ is formed by a modified Gram-Schmidt procedure or Householder transformations. Let $v_1 = r_0/\|r_0\|_2$ and let $\langle x, y\rangle$ denote an inner product of a vector x and a vector y. The so-called *Arnoldi process* is given by:

1. $w_m := Av_m$

2. For $k = 1, 2, \ldots, m$,

   $h_{k,m} := \langle w_m, v_k\rangle$

   $w_m := w_m - h_{k,m}v_k$

   end

3. $h_{m+1,m} := \|w_m\|_2$

   $v_{m+1} := w_m/h_{m+1,m}$

Step 1 is used to expand the Krylov subspace $K_m(A; r_0)$. Each time when the subspace is expanded in one more ranking, a new unit vector v_{m+1} that is mutually orthogonal to all the previous ones is computed by the modified Gram-Schmidt procedure shown in Step 2 and Step 3. The GMRES iterates are constructed as

$$x^{(m)} := x^{(0)} + y_1v_1 + y_2v_2 + \cdots + y_mv_m$$

where y_i are chosen to minimize the residual norm $\|b - Ax^{(m)}\|_2$. The minimization is computed without $x^{(m)}$ having been formed as shown immediately below.

To find y_i, let H_m be the $(m+1) \times m$ Hessenberg matrix whose non-zero entries are $h_{i,j}$. Define $V_m = (v_1, v_2, \ldots, v_m)$, $y = (y_1, y_2, \ldots, y_m)^T$, $\beta = \|r_0\|_2$ and $e_1 = (1, 0, 0, \ldots, 0)^T$. Followed by the Arnoldi process, it can be shown that

$$AV_m = V_{m+1} H_m.$$

Let $z = V_m y$. Then we have

$$\begin{aligned}
\min \|b - Ax^{(m)}\|_2 &= \min \|b - A(x^{(0)} + z)\|_2 \\
&= \min \|r_0 - Az\|_2 \\
&= \min \|r_0 - AV_m y\|_2 \\
&= \min \|r_0 - V_{m+1} H_m y\|_2 \\
&= \min \|V_{m+1}(\beta e_1 - H_m y)\|_2 \\
&= \min \|\beta e_1 - H_m y\|_2.
\end{aligned}$$

The solution of y is given by solving $H_m y = \beta e_1$. Note that the factorization of the Hessenberg matrix H_m can be updated efficiently with the *Given Rotation* method in linear time. Since the set of orthogonal vectors v_i is less than n, the iteration process will terminate in at most n steps (assuming exact arithmetic). Note that all computed orthogonal vectors v_i have to be kept in the GMRES method. To reduced the storage cost, the restart versions of GMRES, denoted by GMRES(k), are used to limit the number of vectors to be k.

The GMRES method uses long recurrences to retain orthogonality and to minimize the norm of residual. In fact Faber and Manteuffel proved that one cannot minimize the norm of residual using short recurrences for general nonsymmetric matrix [29]. Note that if A is *symmetric positive definite*, the well-known

conjugate gradient method can minimizes $\|r_m\|_{A^{-1}}$ using two-term recurrence. Some other Krylov subspace methods, such as the bi-conjugate gradient method (BiCG) and the Quasi-Minimal Residual method (QMR) take another approach of generating the bi-orthogonal basis. In the BiCG method, two sequences of residuals that are mutually orthogonal (or bi-orthogonal) to each other are generated. One sequence of residuals is in $K_m(A; r_0)$ and the other is in $K_m(A^T; r_0)$. We call these A-sequence and A^T-sequence respectively. This method requires the matrix transpose A^T. The convergence may be irregular since there is no minimization process in the BiCG method. Even worse the method may break-down.

There are two types of break-down that may occur in the BiCG method. One is called *Lanczos break-down*, which can be partially cured via *Look-Ahead technique* (see Parlett, Taylor and Liu [53]). Another break-down occurs when the implicit LU decomposition of the reduced tridiagonal system does not exist. The QMR method solves the reduced tridiagonal system in a least square sense, similar to the approach in GMRES, to avoid this break-down. One can also observe that the convergence behavior of QMR is typically smoother than that of BiCG. The Conjugate Gradient Squared method (CGS) attempts to apply the updatings for the A-sequence and the A^T-sequence both to the same vectors. The resulting method is a variant of BiCG that ideally would double the convergence rate. In practice convergence may be much more irregular than that of BiCG. The Bi-conjugate gradient stabilized method (BiCG-Stab), like CGS, use different updates for the A^T-sequence. It can be interpreted as the product of BiCG and repeatedly applied GMRES(1). A residual vector is minimized locally, which leads to a smoother convergence behavior.

As in the above methods, as well as other Krylov subspace methods, inner products are used to retain the orthogonality while matrix-vector products (Av's) are used to expand the Krylov subspace. Note that we do not need an

explicit A as we access to it via the matrix-vector product.

3.2 Krylov Subspace methods in Hilbert Space

We have examined the waveform relaxation methods and their relations of operator equations in Banach space in Section 2.2. To extend the Krylov subspace methods to a function space, we need to look at a more specific space, namely the Hilbert space, where the concepts of inner product and orthogonality are introduced.

Let $\mathbb{X}$ be a Banach space over $\mathbb{R}$. A Hilbert space over $\mathbb{R}$ is a Banach space $\mathbb{X}$ together with an inner product defined as follows. An inner product on $\mathbb{X}$, denoted by $\langle x, y\rangle$, assigns to each pair of $x, y \in \mathbb{X}$ a number such that for all $x, y, z \in \mathbb{X}$ and $\alpha, \beta \in \mathbb{R}$ the following hold:

1. $\langle x, x\rangle \geq 0$, and $\langle x, x\rangle = 0$ if and only if $x = 0$.

2. $\langle x, \alpha y + \beta z\rangle = \alpha\langle x, y\rangle + \beta\langle x, z\rangle$

A vector x is said to be orthogonal to y if and only if $\langle x, y\rangle = 0$. Define a linear operator $\mathcal{A} : \mathbb{X} \to \mathbb{X}$ in a Hilbert space. The linear operator is called *positive definite* if and only if $\langle \mathcal{A}x, x\rangle > 0$ for all $x \neq 0$. The definition of the adjoint operator $\mathcal{A}^*$ is based on the condition:

$$\langle \mathcal{A}x, y\rangle = \langle x, \mathcal{A}^* y\rangle \qquad \text{for all } x, y \in \mathbb{X}.$$

The linear operator is called *self-adjoint* if and only if $\mathcal{A}$ is equal to $\mathcal{A}^*$. Note that if the linear operator is a matrix, its adjoint operator is the transpose of this matrix.

Krylov subspace can be defined similarly to the previous section by:

$$K_m(\mathcal{A}, r_0) = \text{span}\{r_0, \mathcal{A}r_0, \ldots, \mathcal{A}^{m-1}r_0\}.$$

Consider the linear ordinary differential equation given in Equation (2.1). We are now ready to apply the techniques of Krylov subspace method to solve this equation. The idea is the same as the arguments described in Section 2.2. Note that for the linear ordinary differential equations as in Equation (2.1), $\mathcal{A}$ is a pure differential operator. The mechanism of step-size control[1] will be lost. To solve this problem, we can apply the idea of preconditioning in the Krylov subspace methods. In the following section, we describe the approach of Lumsdaine *et al.* [43] in which the waveform Krylov subspace methods are applied to the preconditioned system.

3.3 Waveform Krylov Subspace Methods

As mentioned in Section 3.1, there are two essential operations that should be considered in the design of Krylov subspace methods, namely the matrix-vector product and the inner product. The inner product can be easily extended to a function space by:

$$\langle x(t), y(t) \rangle = \int_{t_0}^{t_m} x^T(s) y(s) ds, \tag{3.4}$$

and the norm of $y(t)$ is defined accordingly as:

$$\|y(t)\| = \sqrt{\langle y(t), y(t) \rangle}.$$

The matrix-vector product can be replaced by the operator-function product $(\mathcal{A}p)(t)$. However, recall that for the linear ordinary differential equation, $\mathcal{A} = (B(t)\frac{d}{dt} + A(t))$ is a pure differential operator. The step-size control mechanism is lost. In [43], the preconditioned system is used, i.e., $(\mathcal{M}^{-1}\mathcal{A}y)(t) = (\mathcal{M}^{-1}f)(t)$. Let $\psi(t) = (\mathcal{M}^{-1}f)(t)$. Then we have:

$$(\mathcal{M}^{-1}\mathcal{A}y)(t) \;=\; (\mathcal{M}^{-1}f)(t)$$

[1]Numercial integration software typically adjust the step-size $\triangle t$ during the course of the integration in order to reduce the truncation error.

$$\begin{aligned}(\mathcal{M}^{-1}(\mathcal{M}-\mathcal{N})y)(t) &= \psi(t)\\ ((\mathcal{I}-\mathcal{M}^{-1}\mathcal{N})y)(t) &= \psi(t)\\ ((\mathcal{I}-\mathcal{K})y)(t) &= \psi(t),\end{aligned}$$

where $\mathcal{I}$ is the identity operator. Recall that $\mathcal{K}$ is the waveform relaxation operator mentioned in Section 2.3. The operator-function product can be implemented as one step of a waveform relaxation iteration [43]. More specifically, the operator-function product $w(t) \equiv ((\mathcal{I}-\mathcal{K})p)(t)$ is given by the following procedure:

1. Solve the following equation for the intermediate variable $q(t)$:

$$\begin{aligned}&M_B(t)\dot{q}(t) + M_A(t)q(t) = N_B(t)\dot{p}(t) + N_A(t)p(t)\\ &\text{with} \quad q(t_0) = p_0 = 0\end{aligned}$$

2. Set $w(t) := p(t) - q(t)$.

In operator notation, the initial residual $r^{(0)}(t)$ is given by:

$$\begin{aligned}r^{(0)}(t) &= \psi(t) - ((\mathcal{I}-\mathcal{K})y^{(0)})(t)\\ &= ((\mathcal{K}y^{(0)})(t) + \psi(t)) - y^{(0)}(t).\end{aligned}$$

The correponding procedure for $r^{(0)}(t)$ is then given by:

1. Solve the following equation for the intermediate variable $q(t)$:

$$\begin{aligned}&M_B(t)\dot{q}(t) + M_A(t)q(t) = N_B(t)\dot{y}^{(0)}(t) + N_A(t)y^{(0)}(t) + f(t)\\ &\text{with} \quad q(t_0) = y_0\end{aligned}$$

2. Set $r^{(0)}(t) := q(t) - y^{(0)}(t)$.

Let $v_1(t) = r^{(0)}(t)/\|r^{(0)}(t)\|$. The Arnoldi process in a Hilbert space can be written as:

1. $w^{(m)}(t) := ((\mathcal{I} - \mathcal{K})v^{(m)})(t)$

2. For $k = 1, 2, \ldots, m$,

$$h_{k,m} := \langle w_m(t), v_k(t) \rangle$$
$$w_m(t) := w_m(t) - h_{k,m} v_k(t)$$

end

3. $h_{m+1,m} := \|w_m(t)\|$
$v_{m+1}(t) := w_m(t)/h_{m+1,m}$

Note that unlike the linear algebraic problem, the set of orthogonal vectors is infinite here. Hence, there is no guarantee that the WGMRES method converges in finite number of steps.

By this process, the waveform conjugate gradient method (WCG), the waveform conjugate gradient squared method (WCGS), the waveform bi-conjugate gradient stabilized method (WBiCG-Stab) and the waveform generalized minimal residual method (WGMRES) can be derived (See Appendix A for pseudo-codes).

Note that if $(\mathcal{I}-\mathcal{K})$ is a *self-adjoint*[2] and *positive definite*[3] operator, the waveform conjugate gradient method will converge for any given initial iterates [54, pp.159]. Unfortunately, $\mathcal{K}$ is not self-adjoint for ODE problems in general.

3.4 Adjoint Operator for WBiCG and WQMR

Recall that the bi-conjugate gradient method (BiCG) and the quasi-minimal residual method (QMR) also need the matrix transpose, i.e. A^T. Analogously, to implement the WBiCG method and the WQMR method, we propose using an adjoint operator. Before calculating the adjoint operator for the waveform

[2]An operator is self-adjoint if and only if $\langle \mathcal{K}(x), y \rangle = \langle x, \mathcal{K}(y) \rangle$ for all x and y.
[3]An operator is positive definite if and only if $\langle \mathcal{K}(x), x \rangle > 0$ for all x.

relaxation operator, let us consider the adjoint operator for the differential operator $\frac{d}{dt}$ (c.f. [24, pp. 604]). Let $\mathbb{X}$ be the Hilbert space $L_2(t_0, t_m)$ whose norm and inner product are defined as those in the previous section. Let $\mathcal{T} = \frac{d}{dt}$ with

$$D(\mathcal{T}) = \{z \in \mathbb{X} \mid z(t_0) = 0 \text{ and } \frac{dz}{dt} \in L_2(t_0, t_m)\},$$

where $D(\mathcal{T})$ denotes the domain of $\mathcal{T}$. The inverse of $\mathcal{T}$ is given by

$$(\mathcal{T}^{-1}x)(t) = \int_{t_0}^{t} x(s)ds.$$

To calculate $(\mathcal{T}^{-1})^*$, we consider

$$\begin{aligned}
\langle \mathcal{T}^{-1}x, y\rangle &= \int_{t_0}^{t_m}\int_{t_0}^{t} x^T(s)y(t)ds\ dt \\
&= \int_{t_0}^{t_m}\int_{s}^{t_m} x^T(s)y(t)dt\ ds \quad \text{by change of variables} \\
&= \int_{t_0}^{t_m} x^T(s)\left(\int_{s}^{t_m} y(t)\ dt\right) ds \\
&= \langle x, (\mathcal{T}^{-1})^* y\rangle.
\end{aligned}$$

We have $[(\mathcal{T}^{-1})^* y](s) = \int_s^{t_m} y(t)\ dt$. This can be approximated numerically by reverse integration. It is easy to see that

$$\begin{aligned}
\mathcal{T}^* y &= -\frac{dy}{dt} \\
\text{with } D(\mathcal{T}^*) &= \{z \in \mathbb{X} \mid z(t_m) = 0 \text{ and } \frac{dz}{dt} \in L_2(t_0, t_m)\},
\end{aligned}$$

Therefore, we propose the adjoint operator-function product $(\mathcal{I} - \mathcal{K}^*)p(t)$ be given by the following procedure:

1. Solve the following equation in reverse time for the itermediate variable $q(t)$:

$$\begin{aligned}
-M_B^T(t)\dot{q}(t) + M_A^T(t)q(t) &= p(t) \\
\text{with} \quad q(t_m) &= p(t_m) = 0
\end{aligned}$$

2. Set $w(t) = p(t) - (N_B^T(t)\dot{q}(t) + N_A^T(t)q(t))$

The pseudo-codes of the WBiCG method and the WQMR method are given in Appendix A.

Note that in a discrete time system, the property $\langle \mathcal{K}(x), y \rangle = \langle x, \mathcal{K}^*(y) \rangle$ may not hold exactly. In the experiment as described below, Equation (3.5) acts as the inner product for the discrete system. It matches nicely with the Backward Euler method for fixed-rate implementation. However, if we choose the Backward Euler method as the ODE solver and the trapezoidal rule as the integrator in Equation (3.4), the two inner products may have some discrepancy. Some numerical experiments showed that this difference will cause the WBiCG method or the WQMR method not to converge. The situation may be even worse when the multi-rate integration technique is used. Therefore, the waveform Krylov subspace methods that requires the adjoint operator may need further investigation.

3.5 Numerical Experiments

In this section, experimental results will be given. Although we follow the works of Lumsdaine *et al.* [43] on the waveform Krylov subspace methods, some results here are new in this field of research. Firsltly, we give the first evidence that the waveform Krylov subspace methods are applicable to tightly coupled circuit systems. Secondly, the convergence behaviors of five waveform Krylov subspace methods are examined extensively by solving an unstructured grid problem. Particularly the effects on the length of the time-interval are investigated. The results are valuable in discussion on the convergence of waveform Krylov subspace methods.

All the computations were done in MATLAB on a Sparc workstation. The basic ODE solver was the Backward Euler method and the basic relaxation

method for the waveform Krylov subspace methods was the WJAC method. The restart version of WGMRES was used and the restart value k was 30. The inner product as defined in Equation (3.4) was replaced by the following formula in the discretized system:

$$\langle x(t), y(t) \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} \triangle t_j x_i(t_j) y_i(t_j) \tag{3.5}$$

where m is the number of points in the time domain.

3.5.1 Test Circuits

The linear circuit shown in Figure 3.1 is used as a test circuit. It was taken from [48]. The values of $C_1 = C_2 = C_3 = C_4 = 1$F. The values of $C_{b1} = C_{b2} = C_{b3} = C_{b4} = 0.1$F. The values of $g_1 = 1$mho, $g_2 = 2$mho, $g_3 = 4$mho and $g_4 = 8$mho. The values of $g_{m1} = g_{m2} = g_{m3} = g_{m4} = g_m$. The time interval is 10 sec. The time-step is 0.1 sec. The iteration stops whenever the residual is decreased relatively by 1×10^{-8} times its initial values. The results are shown in Table 3.1. Each entry of the table represents the total number of operator-function product used in the method, except the waveform Jacobi method where the entry represents the number of iterations. Assume that the operator-function product dominates the computation time of each iterations (which is a valid assumption in circuit simulation), the entries can represent the relative performance of the methods. Note that when g_m is small, the circuit is loosely coupled. The waveform Jacobi method converges (see Table 3.1). When g_m is large, the circuit becomes tightly coupled and the waveform Jacobi method did not converge in 1000 iterations, whereas all the Krylov subspace methods converged. In this experiment, the waveform conjugate gradient squared method (WCGS) converged fastest.

Another test circuit is a ring modulator, which was taken from [25]. This is

a non-linear circuit. The ODE is of the form

$$M\dot{y}(t) = F(t, y), \quad y(0) = y_0,$$

with

$$y \in \mathbb{R}^{15}, \quad t \in [0, 10^{-3}].$$

The function F is defined by [25]

$$F(t,y) = \begin{pmatrix} y_8 - 0.5y_{10} + 0.5y_{11} + y_{14} - R^{-1}y_1 \\ y_9 - 0.5y_{12} + 0.5y_{13} + y_{15} - R^{-1}y_2 \\ y_{10} - q(U_{D1}) + q(U_{D4}) \\ -y_{11} + q(U_{D2}) - q(U_{D3}) \\ y_{12} + q(U_{D1}) - q(U_{D3}) \\ -y_{13} - q(U_{D2}) + q(U_{D4}) \\ -R_p^{-1}y_7 + q(U_{D1}) + q(U_{D2}) - q(U_{D3}) - q(U_{D4}) \\ -y_1 \\ -y_2 \\ -(0.5y_1 - y_3 - R_{g2}y_{10}) \\ -(-0.5y_1 + y_4 - R_{g3}y_{11}) \\ -(0.5y_2 - y_5 - R_{g2}y_{12}) \\ -(-0.5y_2 + y_6 - R_{g3}y_{13}) \\ -(-y_1 + U_{in1} - (R_i + R_{g1})y_{14}) \\ -(-y_2 - (R_c + R_{g1})y_{15}) \end{pmatrix}, \tag{3.6}$$

and

$$M = \text{diag}[(C, C, C_s, C_s, C_s, C_s, C_p, L_h, L_h, L_{s2}, L_{s3}, L_{s2}, L_{s3}, L_{s1}, L_{s1})].$$

The auxiliary function U_{D1}, U_{D2}, U_{D3}, U_{D4}, q, U_{in1} and U_{in2} are:

$$U_{D1} = y_3 - y_5 - y_7 - U_{in2},$$

$$
\begin{aligned}
U_{D2} &= -y_4 - y_6 - y_7 - U_{in2}, \\
U_{D3} &= y_4 + y_5 + y_7 + U_{in2}, \\
U_{D4} &= -y_3 - y_6 + y_7 + U_{in2}, \\
q(U) &= \gamma(e^{\delta U} - 1), \\
U_{in1}(t) &= 0.5\sin(2000\pi t), \\
U_{in2}(t) &= 2\sin(2000\pi t).
\end{aligned}
$$

The values of the parameters C, C_s, C_p, R, R_p, L_h, L_{s1}, L_{s2}, L_{s3}, R_{g1}, R_{g2}, R_{g3}, R_i, R_c, γ and δ are given by:

$$
\begin{aligned}
C &= 1.6 \times 10^{-8}, \\
C_s &= 10^{-9}, \\
C_p &= 10^{-8}, \\
R &= 25000, \\
R_p &= 50, \\
L_h &= 4.45, \\
L_{s1} &= 0.002, \\
L_{s2} &= 5 \times 10^{-4}, \\
L_{s3} &= 5 \times 10^{-4}, \\
R_{g1} &= 36.3, \\
R_{g2} &= 17.3, \\
R_{g3} &= 17.3, \\
R_i &= 50, \\
R_c &= 600, \\
\gamma &= 40.67286402 \times 10^{-9}, \\
\delta &= 17.7493332.
\end{aligned}
$$

Finally, the inital vector y_0 is given by

$$y_0 = (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)^T.$$

We applied the Newton linearization to Equation (3.6) to yield the following iteration:

$$M\dot{y}^{(k+1)}(t) - J(t,y^{(k)})y^{(k+1)}(t) = F(t,y^{(k)}) - J(t,y^{(k)})y^{(k)}(t), \qquad (3.7)$$

where J is the Jacobian of F. Note that M tends to be singular as C_s tends to be zero. The system is ill-conditioned and it can be easily to shown that the spectral radius is greater or equal to one if M is singular. In this experiment, the differentiation of F was calculated analytically by hand and the formulas for J were hard-coded inside the program. The formula of Equation (3.7) at each Newton's iteration was solved by waveform Krylov subspace methods. We calculated the solution to 5 μ sec. with the time-step 0.1 μ sec. The results are shown in Table 3.2. Again the total number of operator-function products was counted for each method. The waveform Jacobi method failed to converge for this circuit. The waveform bi-conjugate gradient stabilized method converged fastest in this experiment.

As the results show, the waveform Krylov subspace methods can be used to accelerate the basic waveform relaxation methods for tightly coupled systems. The remaining question may be which waveform Krylov subspace method is most suitable for circuit simulation. As in the conventional Krylov subspace methods, there is no definite answer to this question. Particularly there is always a tradeoff between efficiency and robustness.

3.5.2 Unstructured Grid Problem

In this experiment, we intend to examine the convergence behavior of waveform Krylov subspace method. Consider a dimensionless heat conducting equation in

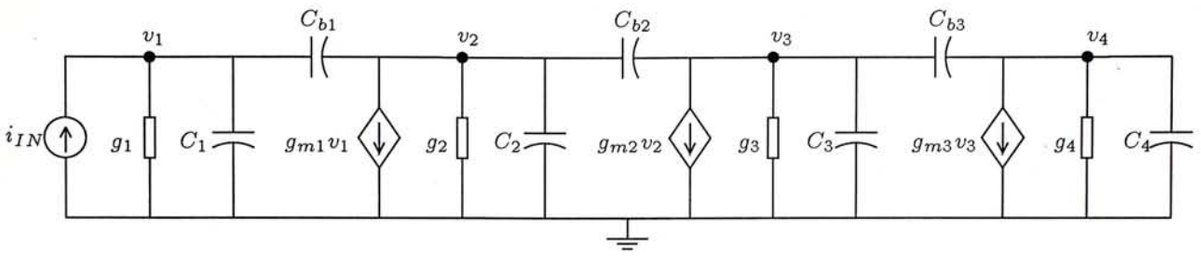

Figure 3.1: The linear test circuit (taken from [49])

Table 3.1: Comparison between the waveform methods applied to the linear circuit. WJAC is the classical waveform Jacobi method.

g_m	Classical	Waveform Krylov Subspace methods				
(mho)	WJAC	WCGS	WBiCG	WBiCG-Stab	WGMRES(30)	WQMR
10	74	44	68	42	30	68
20	139	66	88	72	60	88
30	>1000	90	104	96	120	104

Table 3.2: Comparison between the waveform methods applied to the ring modulator circuit.

The method	WCGS	WBiCG	WBiCG-Stab	WGMRES(30)	WQMR
No. of $\mathcal{K}v$'s/$\mathcal{K}^*v$'s	418	742	252	600	738

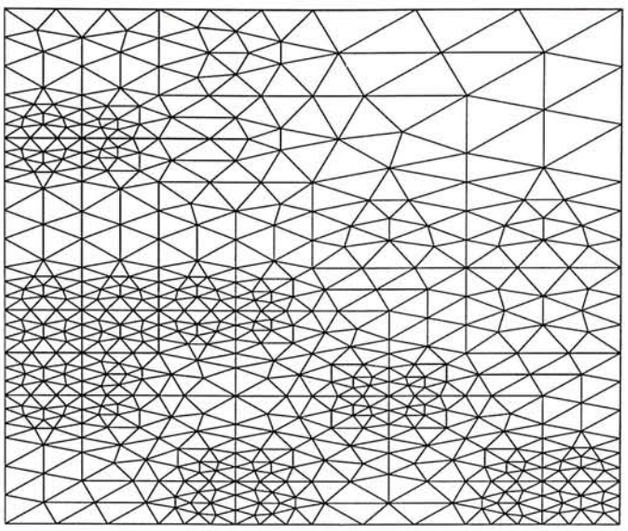

Figure 3.2: The "Eppstein" mesh

two space variables in an irregular domain Ω:

$$\begin{aligned} \frac{\partial u(x,y,t)}{\partial t} &= \frac{\partial^2 u(x,y,t)}{\partial x^2} + \frac{\partial^2 u(x,y,t)}{\partial y^2} + g(x,y,t), \quad t \in (0,\mathcal{T}) \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{3.8}$$

Using the finite element method for spatial discretization in an unstructured mesh, we transform the above equation to Equation (2.5). We consider the unstructured mesh "Eppstein" as shown in Figure 3.2 for our numerical experiment. It has 547 vertices, 1566 edges and 72 boundary vertices.

The time interval was set to be $(0, 60)$. The step size was fixed at 1.0 unit. The initial value was chosen to be the steady state solution of Equation (3.8) when $g = 1$. The transient solution was then computed by taking $g = 0$. This configuration simulated the situation when the loading was suddenly removed. A plot of the residual versus the number of iterations is shown in Figure 3.3.

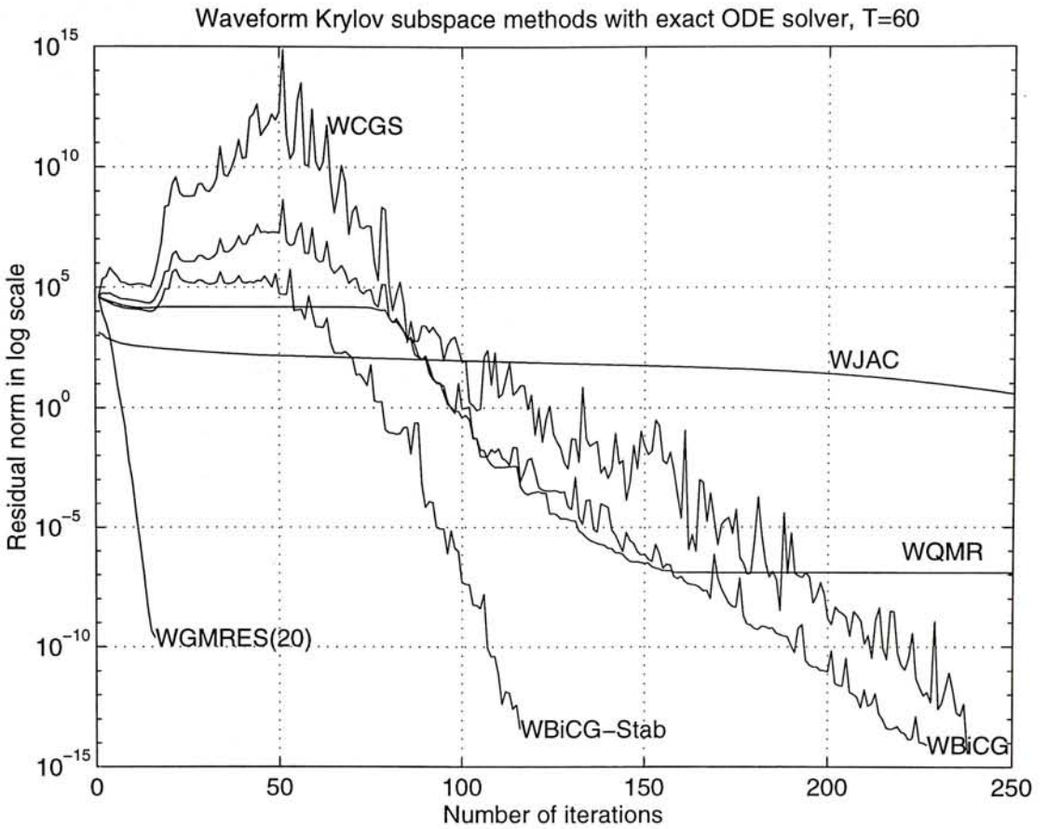

Figure 3.3: Log_{10} of the residual norm versus the number of iterations for the waveform Krylov subspace methods.

The result of the WCG method is not shown in this figure as it failed to converge as expected, because the operator is not self-adjoint. The WGMRES(20) method showed the most stable convergence behavior. This is because in WGMRES (or GMRES as well), the residual norm is minimized at each stage of iterations. Due to the absence of minimization process in the WBiCG method, the irregular convergence behavior was observed. The convergence behavior of the WQMR method was much smoother than that of the WBiCG method. However, the precision which the WQMR method can achieve seems to be low. As in the CGS method that always magnifies the irregular pattern in the BiCG method,

a highly irregular convergence behavior was observed in the WCGS method. On the contrast, a considerably smoother convergence behavior was observed in the WBiCG-Stab method. It shows that the convergence behaviors exhibited in this experiment are confirmed with the observations in their non-functional counterparts.

Next, we investigate the effect of the time interval. As shown in Figures 3.4–3.9, the longer the time interval, the larger number of iterations are required in all methods. In some cases, the waveform methods may even fail to converge if the time interval is too long. Practically, time-windowing technique can be applied, i.e., divide the time interval into several regions and solve each region step by step by the waveform methods. Note also that if the time-window size is too small, the advantage of the multi-rate integration is lost. See [79] for the discussion of the time-windowing technique.

In §5.3, we will return to this experiment by using a DECmpp 12000/Sx massively parallel computer.

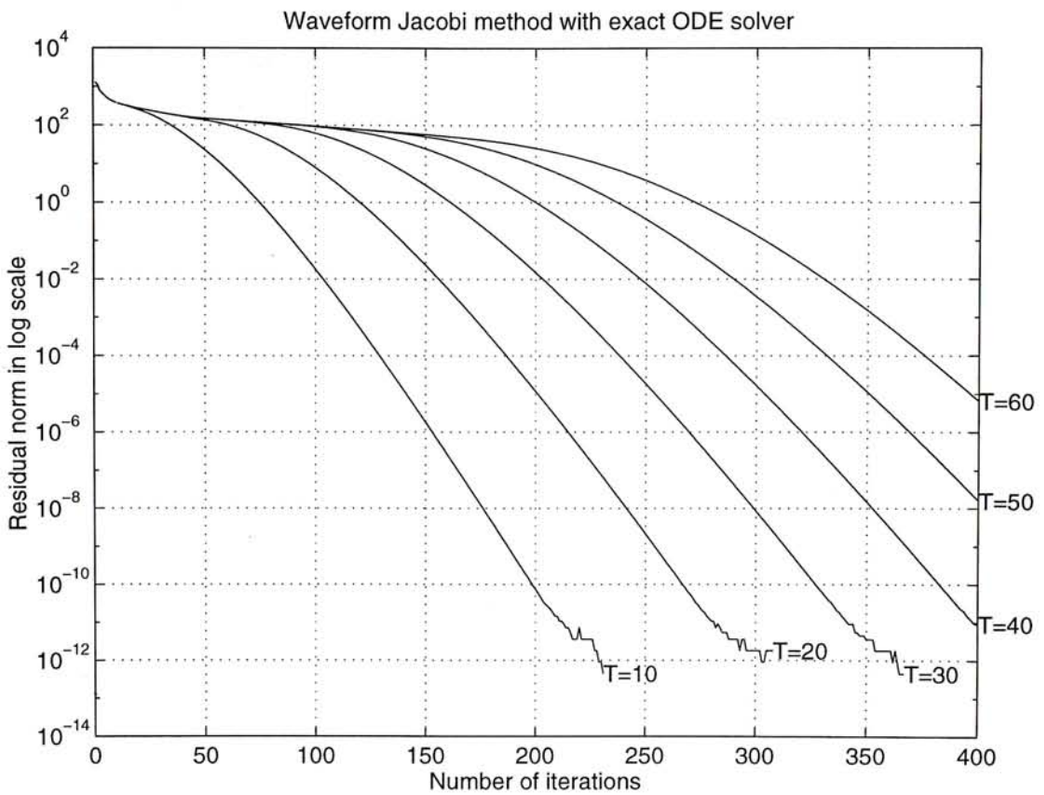

Figure 3.4: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform Jacobi method.

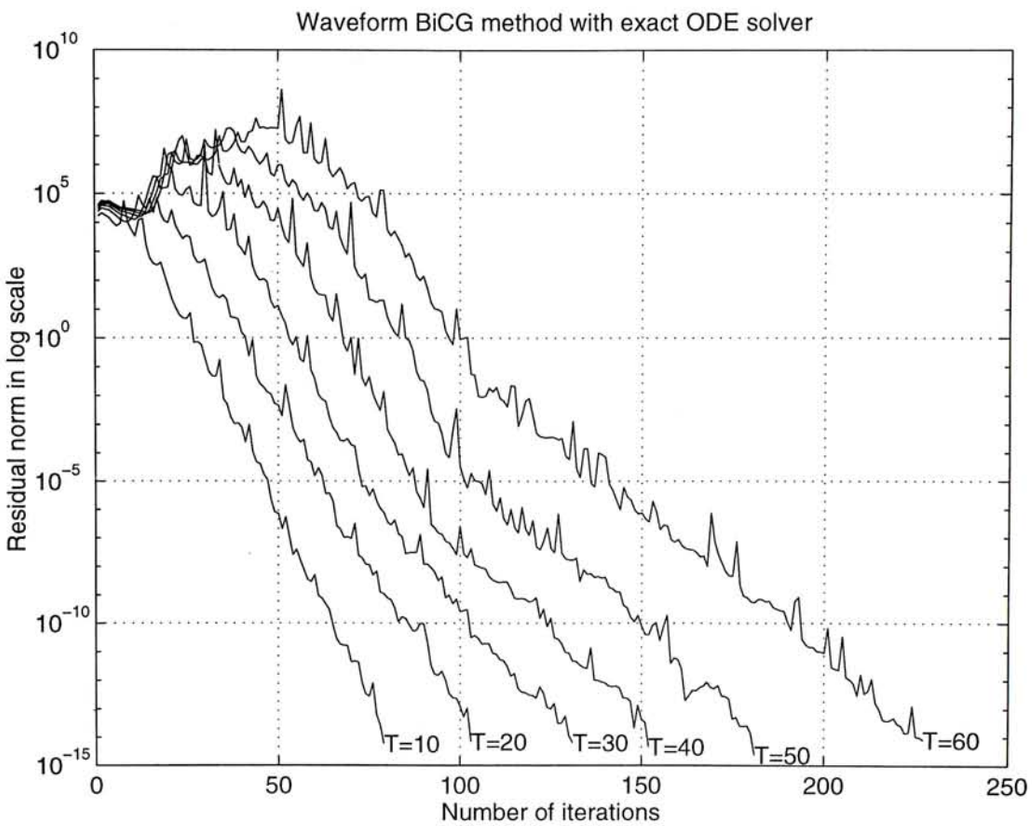

Figure 3.5: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform BiCG method.

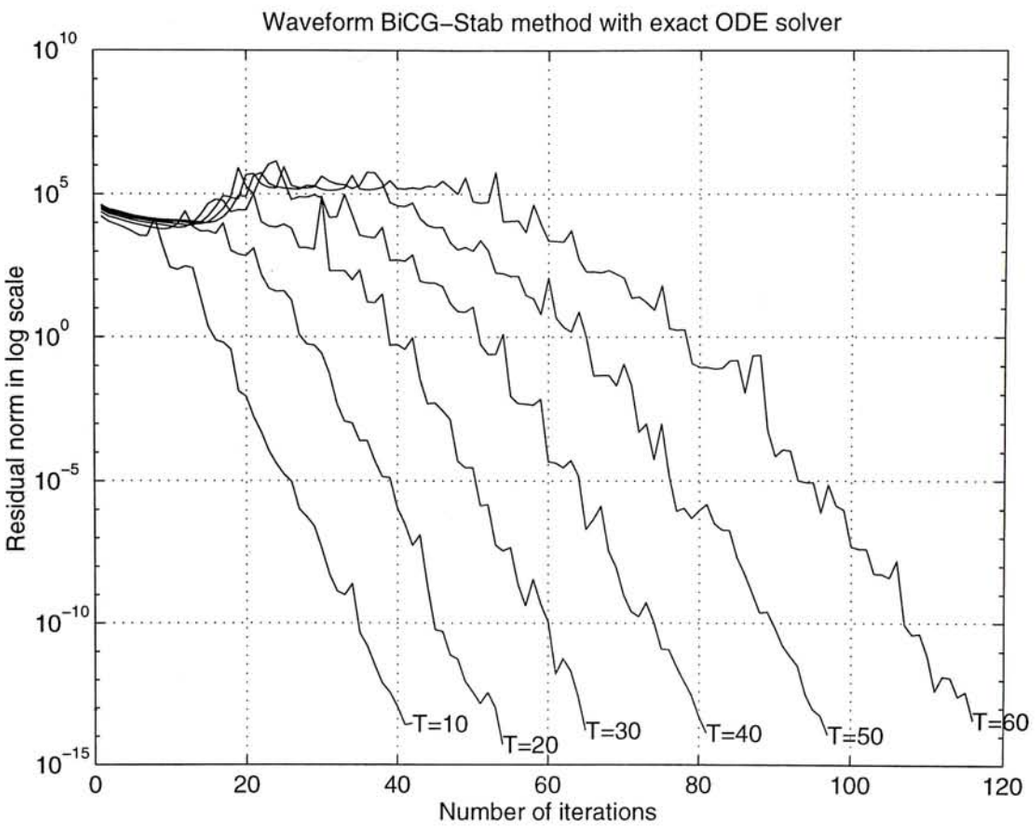

Figure 3.6: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform BiCG-STAB method.

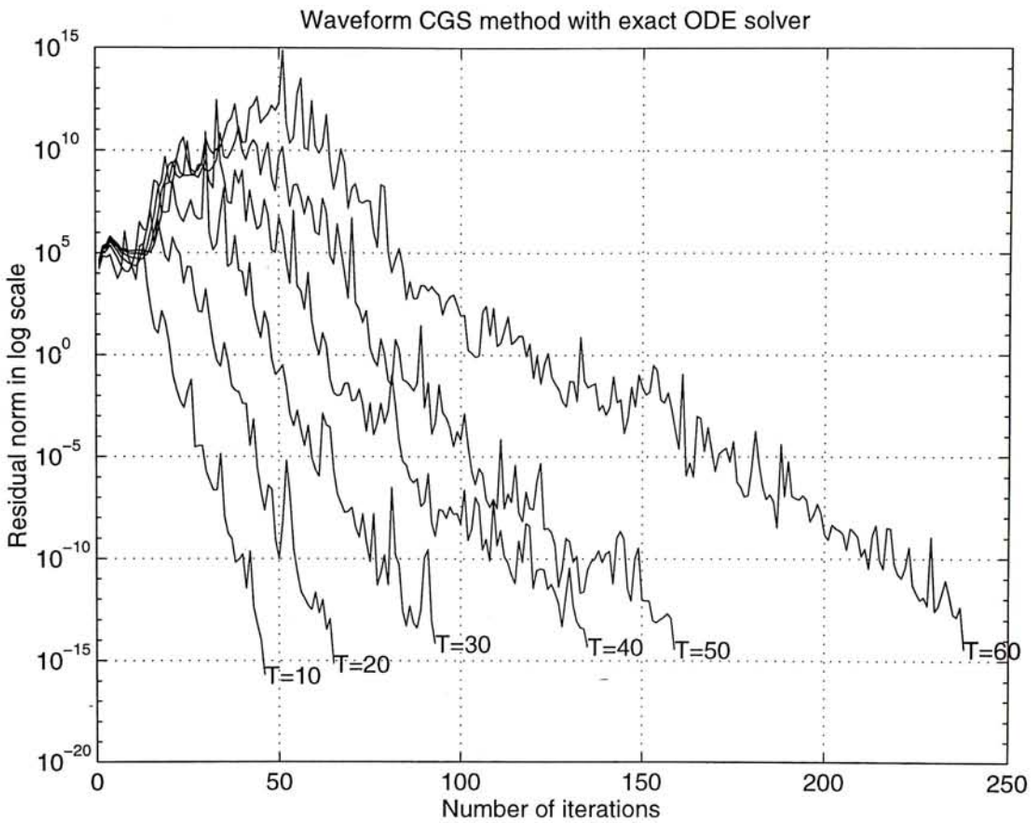

Figure 3.7: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform CGS method.

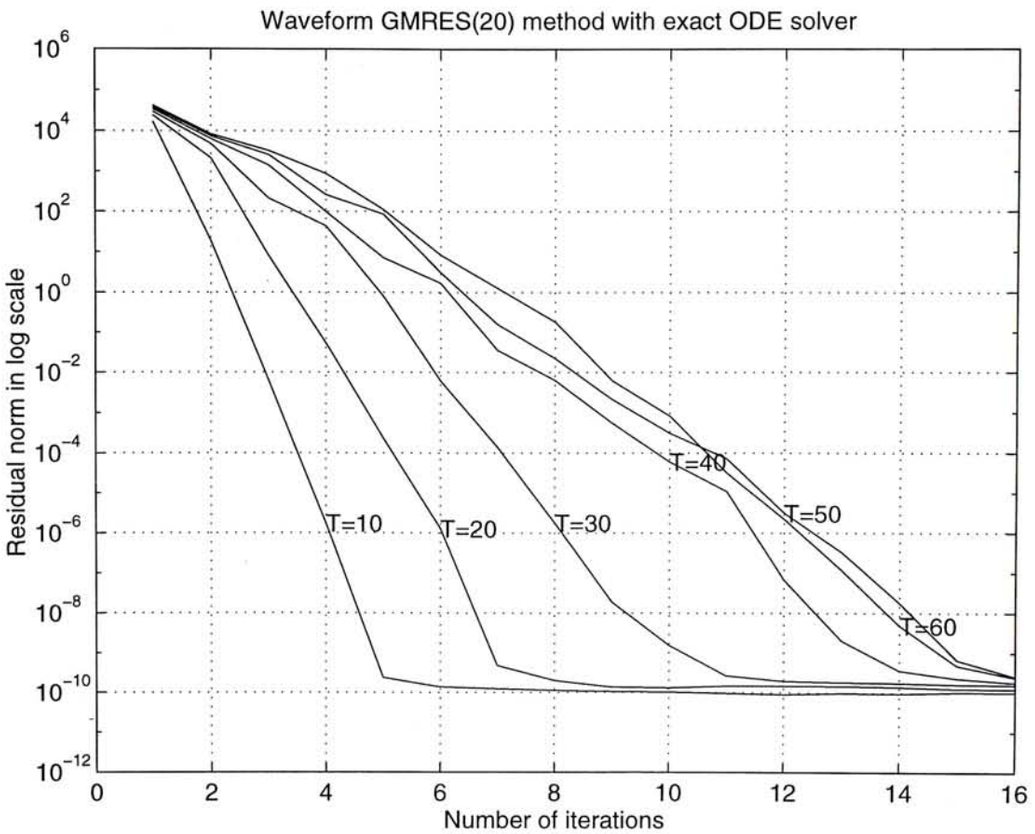

Figure 3.8: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform GMRES(20) method.

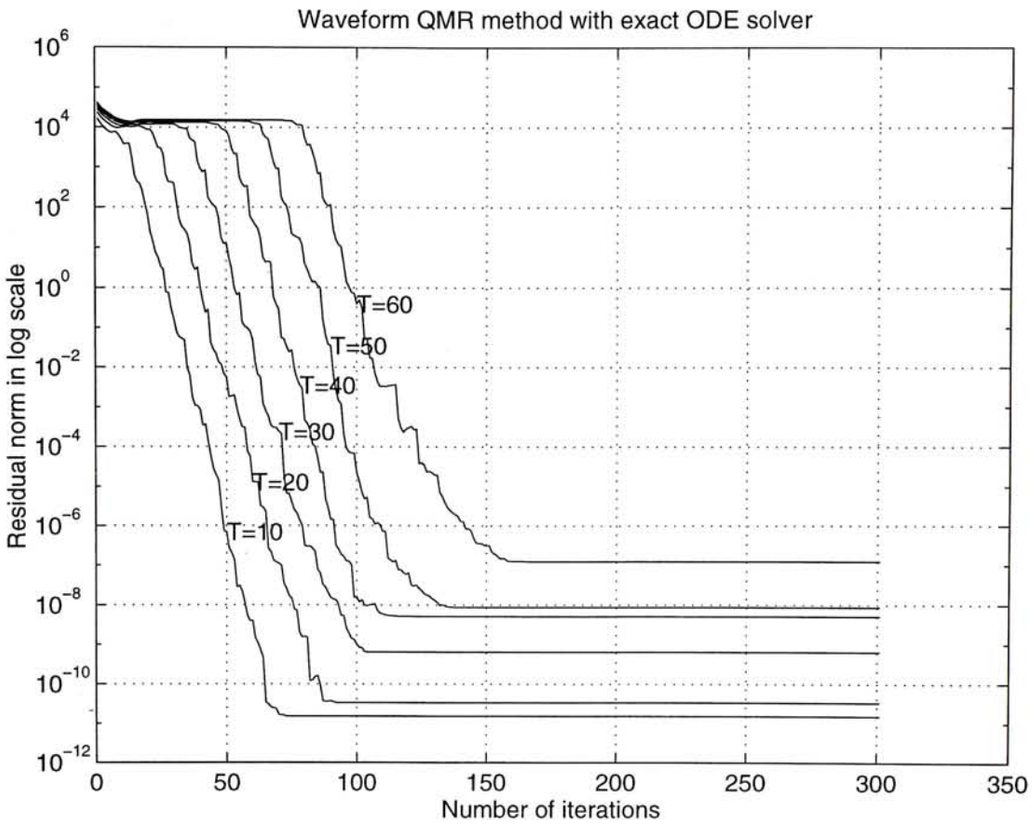

Figure 3.9: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform QMR method.

Chapter 4

Parallel Implementation Issues

In this chapter and the next chapter, we describe the parallel implementation of waveform methods on a DECmpp 12000/Sx massively parallel computer. This chapter provides the general techniques of the parallel implementation issues in case the reader wants to know the implementation details. The reader may skip to the next chapter that specifically describes the use of inexact ODE solver for waveform methods. In this chapter, we will firstly review the hardware of DECmpp 12000/Sx computer and the high performance Fortran in §4.1. Then a sparse matrix format for our implementation is described in §4.3. Section 4.4 contains the multi-coloring technique for the Gauss-Seidel implementation.

4.1 DECmpp 12000/Sx Computer and HPF

In this section, we introduce some basic hardware features of DECmpp 12000/Sx computer and how DECmpp 12000/Sx High Performance Fortran language (HPF) takes advantage of these features. The descriptions are however incomplete because of limited space. For more details, readers should read the user's guide and reference manual provided by Digital Equipment Corporation [26, 27].

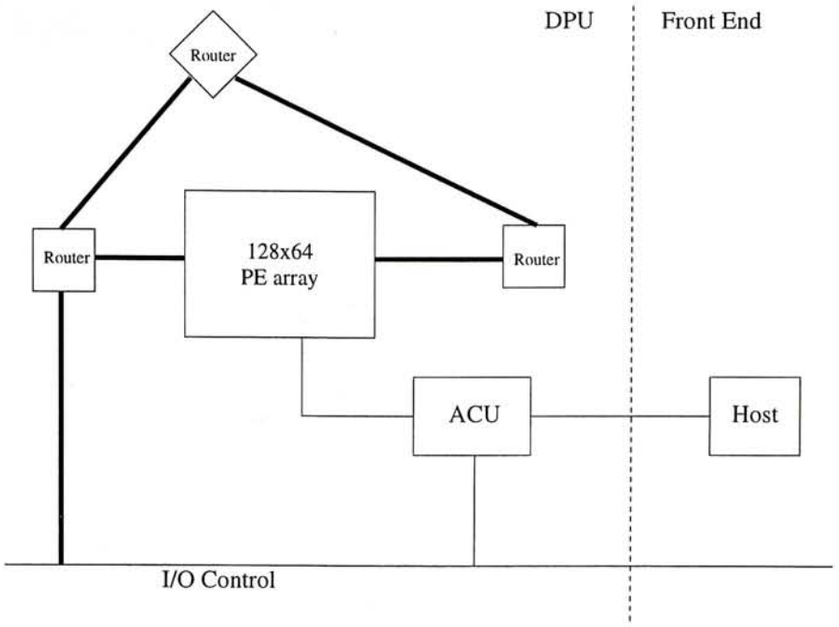

Figure 4.1: The block diagram of the MP-1 computer

Note that a detailed understanding of the DECmpp 12000/Sx architecture is not needed to program in HPF. However, it is important to understand some basic concepts of this machine in order to help you program more effectively.

DECmpp 12000/Sx computer is a *single-instruction multiple-data* (SIMD) type machine. It is also named as a MasPar computer. The model that we use is MP-1, which has 8,192 processor elements (PE) (maximum up to 16,384 PE's with the theoretical peak performance about 550Mflops/s. The more advanced model, MP-2, can achieve 2.4Gflops/s with 16,384 PE's. [66]) It mainly consists of a *Front-end*, which is a DECstation, and a *Data Parallel Unit* (DPU). All program developments can be done on the Front-end. The DPU has two components: the *Array Control Unit* (ACU) and the PE array. The ACU controls the executions of PE array and performs sequential program executions within DPU. The PE Array consists of 128×64 processors arranged in a rectangular shape. Each PE contains 64kbytes local memory. Figure 4.1 shows the block diagram of the MP-1 computer.

You can write Fortran 77 statements in HPF, for example,

```
do i=1,128
   do j=1,128
      C(i,j) = A(i,j) + B(i,j)
   end do
end do
```

When compiled on the DECmpp 12000/Sx, this fragment produces scalar code that executes on the Front-end. By using the array extensions of HPF, the previous operation can be recoded in HPF much more simply:

```
C(1:128,1:128) = A(1:128,1:128) + B(1:128,1:128)
```

The HPF compiler allocates this recoded segment on the DPU, where it executes in parallel fashion.

The compiler handles cases that the array size exceeds the machine size. However, by sizing the array to the machine size, better performance can be achieved. This is due to the fact that the HPF compiler now maps arrays that are less than or equal to the machine size into PE register instead of into PE memory. If the array is larger than the machine size, the whole array is stored in PE memory. To take advantage of this, it is better to write code so that the problem matches the machine size. Using a blocked algorithm can help you do this. The pay-off is that your program will be less comprehensive.

Each PE can identify itself by hard-coded local variables. In MPL (a C-like parallel language in DECmpp 12000/Sx), these variables are called *iproc*, *ixproc* and *iyproc* [28]. In HPF, there is no such similar variable. However, the

compiler internally use these variables to generate parallel code for the FORALL statement. For example:

```
forall (i=1:1000) A(i) = 1.0/i
```

Although each PE can only execute same instructions simultaneously, it can be set to be active or inactive. If a PE is in the inactive set, the instruction will not be carried out by this PE. In HPF, control of the active set is achieved by the WHERE statement. For example, execution of the statement

```
where (A.gt.4) A = -A
```

will negate every elements of matrix `A` that are greater than 4. *Masking* option in some intrinsic functions also makes use of this feature. For example:

```
error = sum(r*r, mask=id.eq.1)
```

There are two types of communication operations in HPF, namely Global router operations and XNet operations. Global router operations results from irregular communication patterns (for example, TRANSPOSE's). It makes use of the global router, which is a three-stage crossbar switch. This type of communication is efficient but can be sixteen times slower than XNet communication. Router code can also be generated by vector-valued subscripts, for example,

```
A(:) = B(I(:))
```

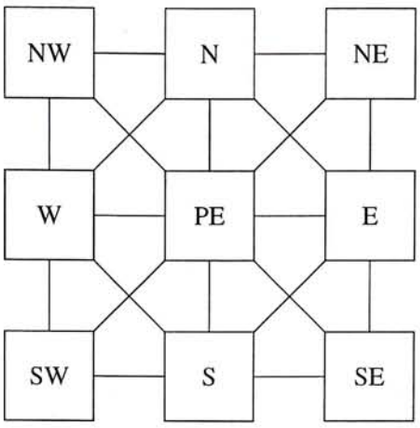

Figure 4.2: The block diagram of X-Net connections

that can be used in sparse matrix operations (see the next section). XNet operations result from regular communication patterns (for example, CSHIFT's and EOSHIFT's). Each PE connects to its eight nearest neighbors through XNet, which is the fastest type of communication preferable to router communication (see Figure 4.1). Also a SPREAD operation might use XNet communication to create the additional dimension.

The HPF compiler supports the *table-lookup feature* to take advantage of *local indirect addressing capability* of DECmpp 12000/Sx. For example, the following code fragment will utilize this feature without generating the router code:

```
      integer   B(1024)
      dimension TAB(10,1024)
      dimension A(1024)
cmpf map TAB(memory,allbits)
            ......
      forall (i=1:1024) A(i) = TAB(B(i),i)
```

One of the applications of this feature is *load balancing* [65].

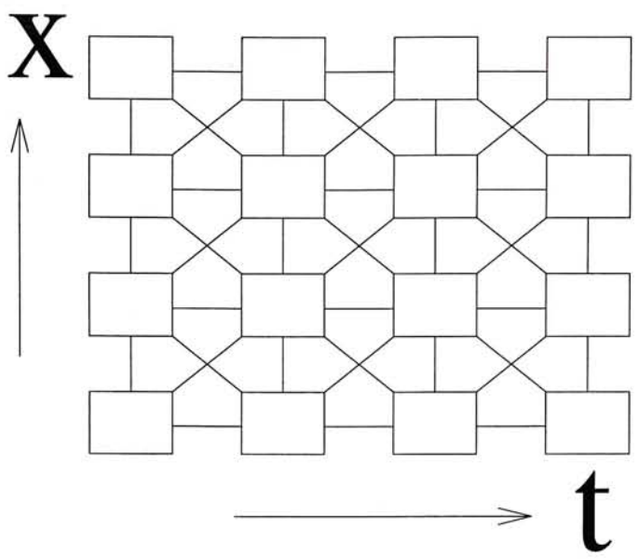

Figure 4.3: Data-mapping strategy

4.2 Data Mapping Strategy

In order to minimize the communication cost of our program, the following data–mapping strategy is used. Each processor contains a space–time variable. The x–direction of processors is referred to the space points and the y–direction of processors is referred to the time points. That is, one row of processors contains a whole discretized waveform of one space variable (See Figure 4.3).

Here, we assume that the number of variables does not exceed the number of processors since the parallel virtualization can be done automatically by the compiler [27]. The complier will internally wrap-around the data in the corresponding dimension.

4.3 Sparse Matrix Format

The sparse matrices are stored in a form that is similar to Purdue storage (see Barrett et al. [5, pp. 61]). For example, the following sparse matrix:

$$\begin{bmatrix} 8.9 & 0.0 & 0.0 & 1.2 & 0.0 \\ 0.0 & 5.6 & 3.4 & 7.8 & 0.0 \\ 0.0 & 0.0 & 8.9 & 0.0 & 3.4 \\ 8.9 & 0.0 & 0.0 & 7.8 & 0.0 \\ 2.3 & 6.7 & 0.0 & 0.0 & 5.6 \end{bmatrix}$$

will be stored as:

a(:,1)	a(:,2)	a(:,3)
8.9	1.2	0.0
5.6	3.4	7.8
8.9	3.4	0.0
7.8	8.9	0.0
5.6	6.7	2.3

together with an array of column indices:

col(:,1)	col(:,2)	col(:,3)
1	4	0
2	3	4
3	5	0
4	1	0
5	2	1

Note that for convenience, the diagonal elements of the sparse matrix are put in the first column of `a(:,:)`. Beside this, an additional one dimensional array is used to store the information about whether the nodes is a boundary or a non-boundary point specifically for unstructured grid problem. Figure 4.4 shows a code fragment that illustrates an implementation of the Jacobi method for

solving a system of linear algebraic equation. The global router communication is employed for the communication needed in the vector valued array indexing (i.e. `x(col(1:n,i))`).

```
c
c     Jacobi iteration
c
      ap(1:n) = 0.D0
      do i = 2, maxdegree
         where (bound(1:n) .eq. 1)
             ap(1:n) = ap(1:n) + x(col(1:n,i)) * a(1:n,i)
         endwhere
      enddo
      where (bound(1:n) .eq. 1)
         x(1:n) = b(1:n) - ap(1:n) / a(1:n,1)
      endwhere
c
c     end Jacobi iteration
c
```

Figure 4.4: The HPF code fragment for solving linear system by Jacobi method.

4.4 Graph Coloring for Unstructured Grid Problems

In order to parallelize the waveform Gauss–Seidel (WGS) code, a multi-coloring algorithm is employed. In this section, the graph coloring problem is reviewed and then the implementation issues are described. We adopt the notation and terminology of graph theory. Recall that a graph G representing a matrix consists of a set of vertices V and a set of edges E. A subset of V is called *independent* if there is no edge joining any of the vertices in the set. We denote an independent subset as I. To *color* the vertices of G means that we assign

Parallel coloring algorithm :
$V' := V$
while $V' \neq 0$
    Choose a maximal independent set I from V'
    Color I in parallel
    $V' := V' \setminus I$
end while

Algorithm 4.1: Outline of a parallel coloring algorithm

the vertices into several independent subsets. From the Gauss–Seidel implementation, the vertices in the same independent subset (color) can be updated in parallel. We call that G is *k–colorable* if k independent subsets are used. The smallest k is called the *chromatic number* and is denoted by $\chi(G)$.

It is well known that to determine the chromatic number of a graph is NP-hard. However, many heuristic algorithms can achieve good solution to the coloring problem practically. In this thesis, we implement a parallel graph coloring method that was proposed by Jones and Plassmann [37]. Recall that a maximal independent subset is an independent subset that no extra vertex can be inserted resulting in an independent subset. In other words, if a vertex is inserted to a maximal independent subset, it must be adjacent to one of the vertices of this subset. Note that there can be more than one maximal independent subset in a graph. The idea of the algorithm is to choose a maximal independent set from a graph and then assign all elements of the set in one color each time. The maximal independent set problem is solved by Ludy's alogrithm [41], which is proven to be in randomized NC. The method is outlined in Algorithm 4.1.

Sometimes the optimal solution is not required. For example, if the number of processors is 100 and the number of vertices is 1000. Ten colors are then needed for parallelization even though the chromatic number may be six. On the contrary, the balancing of the color distribution is more important here, which means that each color set should contain about the same number of elements

and it cannot exceed the number of processors. More details of discussion of balanced graph coloring can be found in [31]. As [37] mentioned, Algorithm 4.1 generates the solution which is not well-balanced. In this thesis, we propose a simple post-processing algorithm to balance color usage of a given coloring assignment. The basic idea is as follows. Denote the number of elements of a set S as $|S|$. Given an initial solution of a coloring algorithm that constains m independent set $\{I_1, I_2, \ldots, I_m\}$. Choose the independent set that contains the largest number of nodes and denote it as I_j. Similarly choose the independent set that contains the smallest number of nodes and denote it as I_i. The idea is to equalize both set by re-assigning certain vertices from the larger set I_j to I_i. To make sure that the assignment will not conflict the independence, the vertices of I_j which belong to the neighbor of I_i (denoted by $N(I_i)$) should be removed first. The procedure is repeated until a balance allocation of colors has been achieved or no improvement can be further performed. The detail of the algorithm is shown in Figure 4.2.

We find that the algorithm can provide a well-balanced results and suits our test problems well. Since our primary goal is just to provide a coloring solution for the parallel implementation of waveform Gauss-Seidel method, detail performance analysis and comparsion with other balancing algorithms will not be presented in this thesis. In the next chapter, we will present the parallel implementation of waveform relaxation and waveform Krylov subspace methods on DECmpp 12000/Sx massively parallel computer. Numerical results will also be given.

```
Given a graph G and the initial solution {I_1, I_2, ..., I_m}.
balance := .false.
while .not. balance
    Choose I_j that contains the largest number of nodes
    Choose I_i that contains the smallest number of nodes
    a := ⌊(|I_j| - |I_i|)/2⌋
    if a > 0
        I_k := I_j \ N(I_i)
        if I_k = ∅
            balance := .true.
        else
            h := min(a, |I_k|)
            assign h vertices from I_k to I_i
        end
    else
        balance := .true.
    end
end
```

Algorithm 4.2: Outline of post-balancing heuristic for coloring

Chapter 5

The Use of Inexact ODE Solver in Waveform Methods

In the classification of parallel ODE methods for initial value problems (IVPs), waveform relaxation can be categorized as parallelism across system [30]. However, little discussion has been made on the parallelism across time, especially for large-scale parallelism. Bellen *et al.* [6] mentioned a low efficiency of iterative parallel ODE solvers because of the sequential nature of IVPs. Nevertheless, one may wish to parallelize both across system and across time if a massively parallel computer is available. In this chapter, we employ an inexact ODE solver which performs only one iteration of an iterative ODE method in each WR iteration. The idea of this method is also similar to "relaxing" both the space and time points simultaneously and to the one that was proposed by Bellen *et al.* [7]. The basic method is then accelerated by the Krylov subspace methods similar to that of Chapter 3.

In §5.1, we demonstrate the idea by considering a problem of solving the heat conducting equation in an irregular spatial domain that was described in §3.5.2. The formulation of waveform methods with an inexact ODE solver is derived.

Then we will examine the convergence properties of both the exact method and the inexact method in §5.1.1. In §5.3, numerical results are given.

5.1 Inexact ODE Solver for Waveform Relaxation

Consider the dimensionless heat conducting equation given in Equation (3.8) on an irregular domain Ω:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f(x,y,t), \quad t \in [0,T] \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

where $[0,T]$ is the time interval in interest. Using the finite element method for spatial discretization, we transform the above equation to a system of ODEs:

$$\dot{y}(t) + A\,y(t) = f(t), \quad \text{with } y(0) = y_0, \quad t \in [0,T],$$

where A is a general sparse matrix and y is the approximate solution of u on the grid point of the given domain. This discretization method is also known as the method of lines. The ODE equation above is just a simplified version of Equation 2.1 where $B(t) = I$ and A is time-invariant. A standard splitting technique can be applied to $A = M - N$:

$$\dot{y}(t) + M\,y(t) = N\,y(t) + f(t), \quad \text{with } y(0) = y_0, \quad t \in [0,T].$$

The k–th waveform iteration is then formulated as:

$$\dot{y}^{(k)}(t) + M\,y^{(k)}(t) = N\,y^{(k-1)}(t) + f(t), \text{ with } y(0) = y_0, \quad t \in [0,T].$$

We now consider the linear multi–step method with fixed time steps for time discretization. Let the time interval be discretized at time points $0 = t_0 < t_1 <$

$t_2 < \cdots < t_m = T$ and let $\triangle t_j = t_j - t_{j-1}$. The standard discretized waveform iteration is formulated as:

$$\begin{aligned} \sum_{i=0}^{l} \alpha_i y^{(k)}(t_{j-i}) \;+\; \triangle t_j \sum_{i=0}^{l} \beta_i M\, y^{(k)}(t_{j-i}) = \\ \triangle t_j \sum_{i=0}^{l} \beta_i N\, y^{(k-1)}(t_{j-i}) + \triangle t_j \sum_{i=0}^{l} \beta_i f(t_{j-i}), \end{aligned} \tag{5.1}$$

for $1 \leq j \leq m$. We will call this method the "exact" method. Let α be $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ and β be $(\beta_1, \beta_2, \ldots, \beta_k)$. In order to further "relax" the time points, we slightly modify Equation (5.1) such that the previous time points of integration can only be taken from the previous iteration:

$$\begin{aligned} \alpha_0 y^{(k)}(t_j) + \triangle t_j \beta_0 M\, y^{(k)}(t_j) \;=\; -\sum_{i=1}^{l} \alpha_i y^{(k-1)}(t_{j-i}) - \triangle t_j \sum_{i=1}^{l} \beta_i M\, y^{(k-1)}(t_{j-i}) \\ +\triangle t_j \sum_{i=0}^{l} \beta_i N\, y^{(k-1)}(t_{j-i}) + \triangle t_j \sum_{i=0}^{l} \beta_i f(t_{j-i}), \end{aligned}$$

or simplified as

$$\begin{aligned} (\alpha_0 I + \triangle t_j \beta_0 M)\, y^{(k)}(t_j) \;=\; -\sum_{i=1}^{l} (\alpha_i I + \triangle t_j \beta_i M)\, y^{(k-1)}(t_{j-i}) \\ +\triangle t_j \sum_{i=0}^{l} \beta_i N\, y^{(k-1)}(t_{j-i}) + \triangle t_j \sum_{i=0}^{l} \beta_i f(t_{j-i}). \end{aligned} \tag{5.2}$$

This will be called the "inexact" method.

5.1.1 Convergence Analysis

In this section, we present the fixed-rate discrete–time analysis of the "inexact" method. Let a spectral radius of matrix A, denoted by $\rho(A)$, be the largest absolute eigenvalue of A. A simplified version of Theorem 2.4.2 in Section 2.4.2 is given by:

Theorem 5.1.1 *The "exact" waveform relaxation that corresponds to Equation (5.1) converges if and only if the following condition holds:*

$$\max_{1\le j\le m} \rho((\alpha_0 I + \triangle t_j \beta_0 M)^{-1}(\triangle t_j \beta_0 N)) < 1. \tag{5.3}$$

Note that the spectral radius is independent of T. We perform the convergence analysis for our method similarly. Particularly, we have a theorem as follows.

Theorem 5.1.2 *The "inexact" waveform relaxation that corresponds to Equation (5.2) converges if and only if the following condition holds:*

$$\max_{1\le j\le m} \rho((\alpha_0 I + \triangle t_j \beta_0 M)^{-1}(\triangle t_j \beta_0 N)) < 1. \tag{5.4}$$

which is identical to inequality (5.3).

Proof : The technique of proving is similar to the proof of Theorem 2.4.2 in Section 2.4.2. Define the error $e^{(k)}(t_j) \equiv y^{(k)}(t_j) - y^{(k-1)}(t_j)$. It can easily be shown that

$$\begin{aligned}(\alpha_0 I + \triangle t_j \beta_0 M)\, e^{(k)}(t_j) \;&=\; -\sum_{i=1}^{l} (\alpha_i I + \triangle t_j \beta_i M)\, e^{(k-1)}(t_{j-i}) \\ &\quad + \triangle t_j \sum_{i=0}^{l} \beta_i N\, e^{(k-1)}(t_{j-i}).\end{aligned}$$

The above system of equations can be rewritten in matrix form as

$$\begin{bmatrix} G_1 & & & \\ & G_2 & & 0 \\ & & \ddots & \\ 0 & & & G_m \end{bmatrix} \begin{bmatrix} e^{(k)}(t_1) \\ e^{(k)}(t_2) \\ \vdots \\ e^{(k)}(t_m) \end{bmatrix} = \begin{bmatrix} H_{1,1} & & & \\ H_{2,1} & H_{2,2} & & 0 \\ \vdots & \vdots & \ddots & \\ H_{m,1} & H_{m,2} & \cdots & H_{m,m} \end{bmatrix} \begin{bmatrix} e^{(k-1)}(t_1) \\ e^{(k-1)}(t_2) \\ \vdots \\ e^{(k-1)}(t_m) \end{bmatrix}$$

or denoted as

$$G\,\tilde{e}^{(k)} = H\,\tilde{e}^{(k-1)}$$

where

$$
\begin{aligned}
\tilde{e}^{(k)} &= (e^{(k)}(t_1)^T, e^{(k)}(t_2)^T, \ldots, e^{(k)}(t_m)^T)^T \\
G_p &= \alpha_0 I + \triangle t_p \beta_0 M, \\
H_{p,q} &= \begin{cases} \triangle t_p \beta_0 N & \text{if } p = q, \\ -\alpha_{p-q} I + \triangle t_q \beta_{p-q} M + \triangle t_q \beta_{p-q} N & \text{if } q < p \leq q + k, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

Because G is a block diagonal matrix, G^{-1} is also block diagonal. Since H is a block lower triangular matrix, consequently $(G^{-1}H)$ is a block lower triangular matrix with diagonal submatrices $(G_p^{-1}H_{p,p})$. We know that the eigenvalues of a lower triangular matrix are determined by the eigenvalues of its diagonal submatrices. Therefore, we have

$$
\begin{aligned}
\rho(G^{-1}H) &= \max_{1 \leq p \leq m} \rho(G_p^{-1} H_{p,p}) \\
&= \max_{1 \leq p \leq m} \rho((\alpha_0 I + \triangle t_p \beta_0 M)^{-1} (\triangle t_p \beta_0 N)).
\end{aligned}
$$

□

The above theorem shows that the spectral radius of the iteration equation resulting from the "inexact" solver is the same as that from the standard method, and hence the new method is robust.

5.2 Inexact ODE Solver for Waveform Krylov Subspace Methods

Krylov subspace methods can be applied to Eqaution (5.2) by the similar techniques discussed in Chapter 3. The only problem is that we need a new adjoint operator for the "inexact" method. In this section, we firstly reformulate

Equation (5.1) and Equation (5.2) in matrix notations since they have been unfolded in space and time. Then the description of the waveform Krylov subspace methods and the adjoint operator will be further derived. Also, we employ the *Kronecker product notation* to simplify the mathematic matrix notations. Recall that the Kronecker product of two matrices $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{n\times n}$, denoted by $A \otimes B$, is a large matrix with the size of $(m\,n) \times (m\,n)$ formed by:

$$A \otimes B = \begin{bmatrix} a_{1,1}B & a_{1,2}B & \cdots & a_{1,m}B \\ a_{2,1}B & a_{2,2}B & \cdots & a_{2,m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1}B & a_{m,2}B & \cdots & a_{m,m}B \end{bmatrix},$$

where $a_{i,j}$ is the i-th row the j-th column element of A. It is easily shown that

$$\begin{aligned} (A+B)\otimes C &= (A\otimes C)+(B\otimes C) \\ \text{and} \quad (A\otimes B)^T &= A^T \otimes B^T \end{aligned}$$

for any matrices A, B and C. Let

$$\hat{\alpha} = \begin{bmatrix} \alpha_0 & & & & & \\ \alpha_1 & \alpha_0 & & & & \\ \cdot & \alpha_1 & \alpha_0 & & & \\ \alpha_l & \cdot & \cdot & \cdot & & \\ & \cdot & \cdot & \cdot & \cdot & \\ & & \alpha_l & \cdot & \alpha_1 & \alpha_0 \end{bmatrix} \quad \text{and} \quad \hat{\beta} = \begin{bmatrix} \beta_0 & & & & & \\ \beta_1 & \beta_0 & & & & \\ \cdot & \beta_1 & \beta_0 & & & \\ \beta_l & \cdot & \cdot & \cdot & & \\ & \cdot & \cdot & \cdot & \cdot & \\ & & \beta_l & \cdot & \beta_1 & \beta_0 \end{bmatrix}.$$

We denote a diagonal matrix with the diagonal elements $a_{1,1}$, $a_{2,2}$, ..., $a_{n,n}$ by diag[$(a_{1,1}, a_{2,2}, \ldots, a_{n,n})$] and let $\hat{\delta} = \text{diag}[(\triangle t_1, \triangle t_2, \ldots, \triangle t_m)]$. Let a discretized vector function of $w(t)$ be "stacked" in a column vector denoted by $\tilde{w}$. That is

$$\tilde{w} = (w(t_1)^T, w(t_2)^T, \ldots, w(t_m)^T)^T.$$

Denote the identity matrix of order k as I_k here. The "exact" method which corresponds to Eqaution (5.1) can be now rewritten as:

$$(\hat{\alpha} \otimes I_n)\tilde{y}^{(k)} + (\hat{\delta}\hat{\beta} \otimes M)\tilde{y}^{(k)} = (\hat{\delta}\hat{\beta} \otimes N)\tilde{y}^{(k-1)} + (\hat{\delta}\hat{\beta} \otimes I_n)\tilde{f},$$

or simplified as

$$[(\hat{\alpha} \otimes I_n) + (\hat{\delta}\hat{\beta} \otimes M)]\tilde{y}^{(k)} = (\hat{\delta}\hat{\beta} \otimes N)\tilde{y}^{(k-1)} + (\hat{\delta}\hat{\beta} \otimes I_n)\tilde{f}.$$

The "inexact" method which corresponds to Eqaution (5.2) can be rewritten as:

$$\begin{aligned}[(\alpha_0 I_m \otimes I_n) \quad + \quad & (\beta_0\hat{\delta} \otimes M)]\tilde{y}^{(k)} = [(\alpha_0 I_m - \hat{\alpha}) \otimes I_n]\tilde{y}^{(k-1)} \\ & +[\hat{\delta}(\beta_0 I_m - \hat{\beta}) \otimes M]\tilde{y}^{(k-1)} + (\hat{\delta}\hat{\beta} \otimes N)\tilde{y}^{(k-1)} + (\hat{\delta}\hat{\beta} \otimes I_n)\tilde{f},\end{aligned}$$

or simplified as

$$\begin{aligned}&[(\alpha_0 I_m \otimes I_n) + (\beta_0\hat{\delta} \otimes M)]\tilde{y}^{(k)} = \\ &\quad [(\alpha_0 I_m - \hat{\alpha}) \otimes I_n + \hat{\delta}(\beta_0 I_m - \hat{\beta}) \otimes M + (\hat{\delta}\hat{\beta} \otimes N)]\tilde{y}^{(k-1)} + (\hat{\delta}\hat{\beta} \otimes I_n)\tilde{f}.\end{aligned}$$

Define the matrices $\mathbf{M}, \mathbf{N}, \bar{\mathbf{M}}$ and $\bar{\mathbf{N}}$ as:

$$\begin{aligned}\mathbf{M} &\equiv (\hat{\alpha} \otimes I_n) + (\hat{\delta}\hat{\beta} \otimes M) \\ \mathbf{N} &\equiv (\hat{\delta}\hat{\beta} \otimes N) \\ \bar{\mathbf{M}} &\equiv \alpha_0 I_m + (\beta_0\hat{\delta} \otimes M) \\ \bar{\mathbf{N}} &\equiv (\alpha_0 I_m - \hat{\alpha}) \otimes I_n + \hat{\delta}(\beta_0 I_m - \hat{\beta}) \otimes M + (\hat{\delta}\hat{\beta} \otimes N).\end{aligned}$$

The discrete version of waveform relaxation operators defined in Chapter 2 (and hence are matrices here) for the "exact" and "inexact" methods are given by:

$$\mathbf{K} = \mathbf{M}^{-1}\mathbf{N} \quad \text{and} \quad \bar{\mathbf{K}} = \bar{\mathbf{M}}^{-1}\bar{\mathbf{N}}$$

respectively. The corresponding adjoint operators defined in Chapter 3 are given by:

$$\mathbf{K}^* = \mathbf{K}^T = \mathbf{N}^T\mathbf{M}^{-T} \quad \text{and} \quad \bar{\mathbf{K}}^* = \bar{\mathbf{K}}^T = \bar{\mathbf{N}}^T\bar{\mathbf{M}}^{-T}.$$

respectively. Note that

$$\begin{aligned}
\mathbf{M}^T &= (\hat{\alpha}^T \otimes I_n) + (\hat{\beta}^T \hat{\delta} \otimes M^T) \\
\mathbf{N}^T &= (\hat{\beta}^T \hat{\delta} \otimes N^T) \\
\bar{\mathbf{M}}^T &= \alpha_0 I_m + (\beta_0 \hat{\delta} \otimes M^T) \\
\bar{\mathbf{N}}^T &= (\alpha_0 I_m - \hat{\alpha}^T) \otimes I_n + (\beta_0 I_m - \hat{\beta}^T)\hat{\delta} \otimes M^T + (\hat{\beta}^T \hat{\delta} \otimes N^T).
\end{aligned}$$

Based on these formulas, the corresponding procedures can be setup for adjoint operator-function products.

5.3 Experimental Results

We consider again the unstructured mesh as shown in Figure 3.2 for our numerical experiments in DECmpp 12000/Sx computer. The data-mapping strategy and the sparse matrix format were described in the previous chapter.

First, we compare the two versions of waveform Jacobi method (WJAC) in two different time intervals, $(0, 63)$ and $(0, 127)$. The step size was fixed to be 1.0 unit. The initial value y_0 is chosen to be the steady state solution of Equation (3.8) when $f = 1$. The transient solution was then computed using the WR methods by taking $f = 0$. This configuration simulated the situation when the loading was suddenly removed. All the computations were in double precision. The iterations were stopped when the maximum absolute error $\max_{i,j} |y_i^{(k)}(t_j) - y_i^{(k-1)}(t_j)|$ is less than 1×10^{-6}. We plotted the absolute error versus the number of iterations as shown in Figure 5.1. The solid lines represent the results of $T = 63$ and the dotted lines represent the results of $T = 127$. We observed that the asymptotic behaviors of the four methods were similar. It supported the theoretical results mentioned in Theorem 5.1.2 that the spectral radii of both methods are identical.

Table 5.1: Comparison between the exact Backward Euler (BE) solver and the inexact BE solver for solving the equation (1) on the domain as shown in Figure 6 (Time step = 1.0) by WJAC

Time intervals	Inexact BE solver		Exact BE solver	
	DPU times	no. of iter.	DPU times	no. of iter.
0.0–10	20.0 sec.	162	73.8 sec.	154
0.0–20	31.3 sec.	234	224.2 sec.	215
0.0–30	41.8 sec.	298	420.3 sec.	270
0.0–40	49.7 sec.	359	840.3 sec.	321
0.0–50	57.8 sec.	418	1096.7 sec.	370
0.0–60	67.5 sec.	474	1365.3 sec.	416

In Table 5.1, we presented the comparison of the execution times and the numbers of iterations between the two WJAC methods. Only the execution times of the subroutines of the solvers were measured by the internal timer of the Data Processing Unit (DPU), excluding all the setup times. We observed that the number of iterations was increased linearly against the time interval in both methods, and the exact method always needed less iterations. However, because of the large–scale parallelism, the inexact method was much faster than the exact method.

In Table 5.2, we showed the measurement results of WJAC and WGS, while the inexact ODE solver was used. The configuration of the experiment was the same as the above experiment. Five colors were used in WGS, which means that each iteration consists of five sequential steps. A factor of two improvement was observed from WGS over WJAC after performing multi-coloring technique.

The results of waveform Krylov subspace methods are given in Table 5.3 and Table 5.4. The corresponding residual plots are given in Figures 5.3– 5.8.

We found that the WBiCG method, the WCGS method and the WQMR

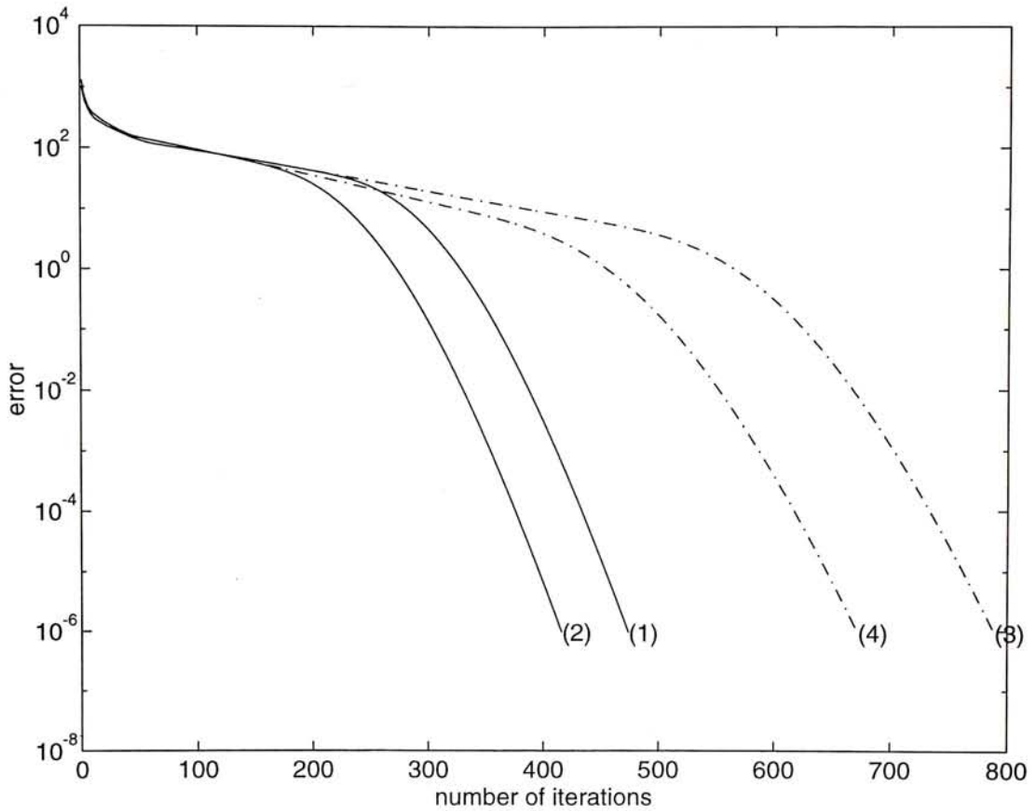

Figure 5.1: Log_{10} of the error versus the number of iterations. (1) The "inexact" method, time interval = (0,63). (2) The "exact" method, time interval = (0,63). (3) The "inexact" method, time interval = (0,127). (4) The "exact" method, time interval = (0,127).

Table 5.2: Comparison between WJAC and WGS on "Eppstein" mesh. (Time step = 1, Tolerance = 1e-6).

Time intervals	WJAC		WGS (5 colors)	
	DPU times	no. of iter.	DPU times	no. of iter.
0.0–10	20.0 sec.	162	12.1 sec.	89
0.0–20	31.3 sec.	234	22.0 sec.	131
0.0–30	41.8 sec.	298	27.8 sec.	168
0.0–40	49.7 sec.	359	33.7 sec.	204
0.0–50	57.8 sec.	418	39.3 sec.	238
0.0–60	67.5 sec.	474	45.0 sec.	272

Table 5.3: Comparison between the waveform Jacobi method (WJAC) and five waveform Krylov subspace methods in term of number of operator-function products. (Time step = 1, Tolerance = 1e-6).

Time interval	WJAC	WBiCG-Stab	WGMRES(20)	WBiCG	WCGS	WQMR
0.0–10	162	62	80	110	68	110
0.0–20	234	98	160	158	138	160
0.0–30	298	136	200	258	-	-
0.0–40	359	170	220	522	-	-
0.0–50	418	226	280	-	-	-
0.0–60	474	266	300	-	-	-

Table 5.4: Comparison between the waveform Jacobi method (WJAC) and five waveform Krylov subspace methods in term of execution times in second. (Time step = 1, Tolerance = 1e-6).

Time interval	WJAC	WBiCG-Stab	WGMRES(20)	WBiCG	WCGS	WQMR
0.0–10	20.0	8.0	13.2	14.5	8.8	14.7
0.0–20	31.3	14.3	28.9	23.9	20.5	48.6
0.0–30	41.8	23.1	38.6	40.6	-	-
0.0–40	49.7	26.7	41.6	84.0	-	-
0.0–50	57.8	35.4	52.5	-	-	-
0.0–60	67.5	41.7	55.8	-	-	-

method, which have less stable convergence behavior, failed to converge when the time intervals were long. In the previous section, we proved that the spectral radius of both exact and inexact WR method are the same. In spite of this, it can only be used to measure the asymptotic behavior. The results show here that some waveform Krylov subspace methods were less stable when the inexact ODE solver was used. However, the relatively stable WBiCG-Stab method and the WGMRES method still have good performance.

5.4 Concluding Remarks

We summarize the techniques that we described in the previous chapters and this chapter in an overall view. A methodology of design and analysis of waveform methods can be as follows. Consider a linear ordinary differential equation problem. We reformulate the problem in the form of $(\mathcal{A}y)(t) = f(t)$, where $\mathcal{A}$ is described by the giving equation. The techniques in linear algebra can then be applied, as if we are solving an ordinary matrix problem. The numerical results showed that the convergence behaviors of those methods are similar to

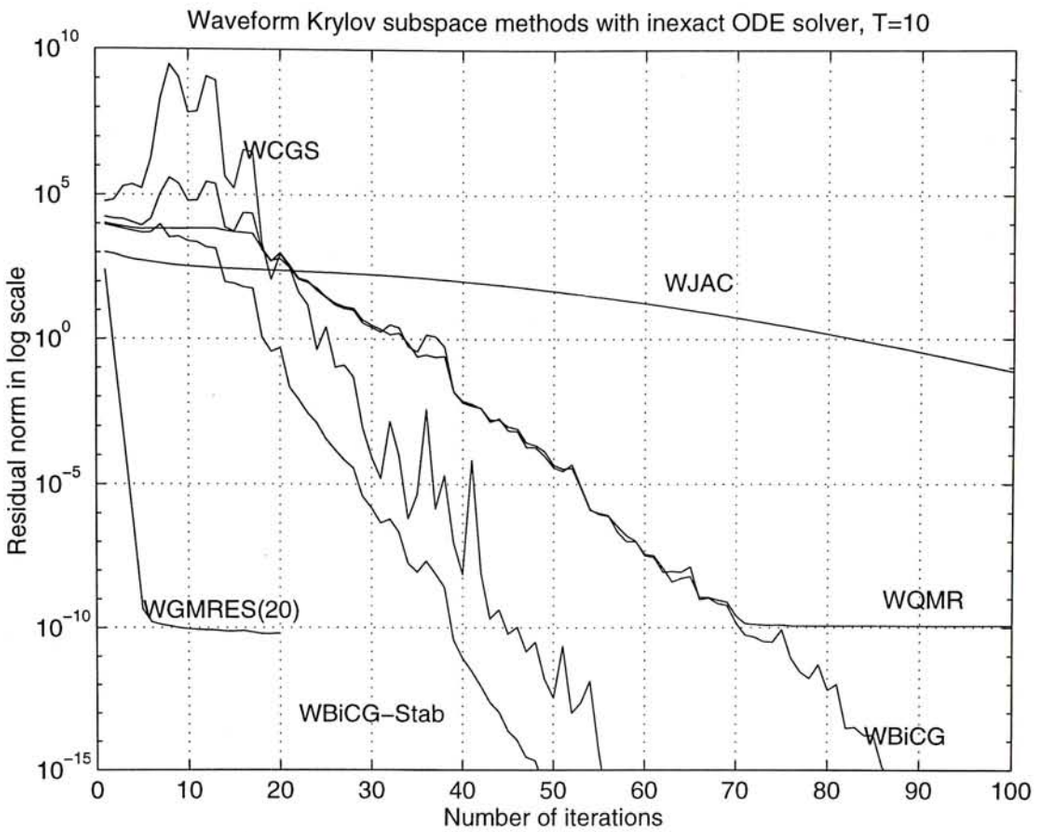

Figure 5.2: Log_{10} of the residual norm versus the number of iterations for the waveform Krylov subspace methods.

their non-functional counterparts. With this conceptual framework in mind, in the next chapter we will further consider the domain decomposition technique applied to the developed waveform Krylov subspace methods for solving large sparse ordinary differential equations. Domain decomposition is acted as a preconditioner in the waveform Krylov subspace methods. The resulting methods are similar to the overlapping Schwarz methods for solving elliptic partial differential equations.

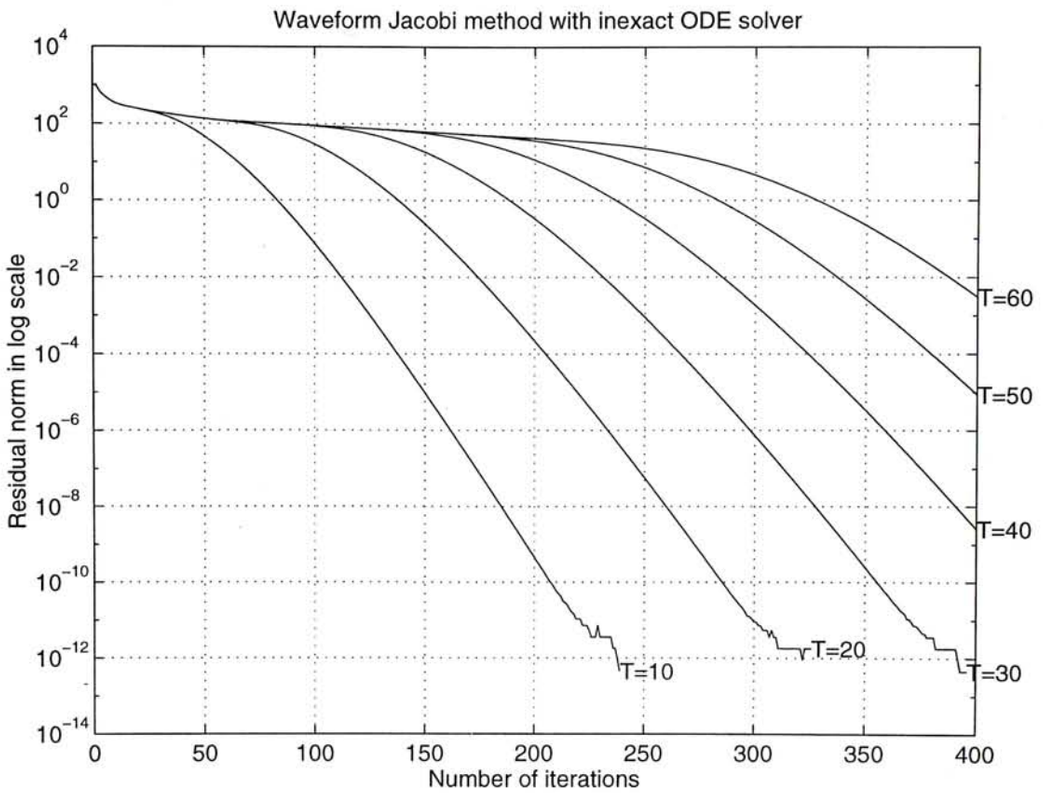

Figure 5.3: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform Jacobi method.

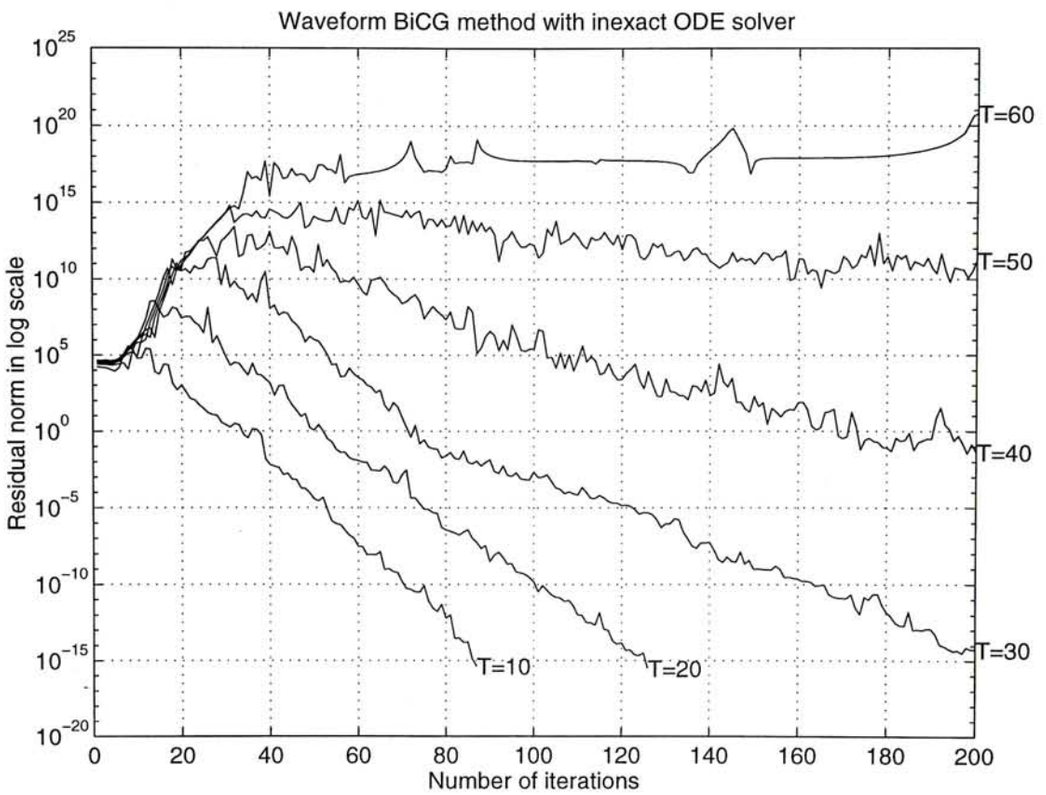

Figure 5.4: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform BiCG method.

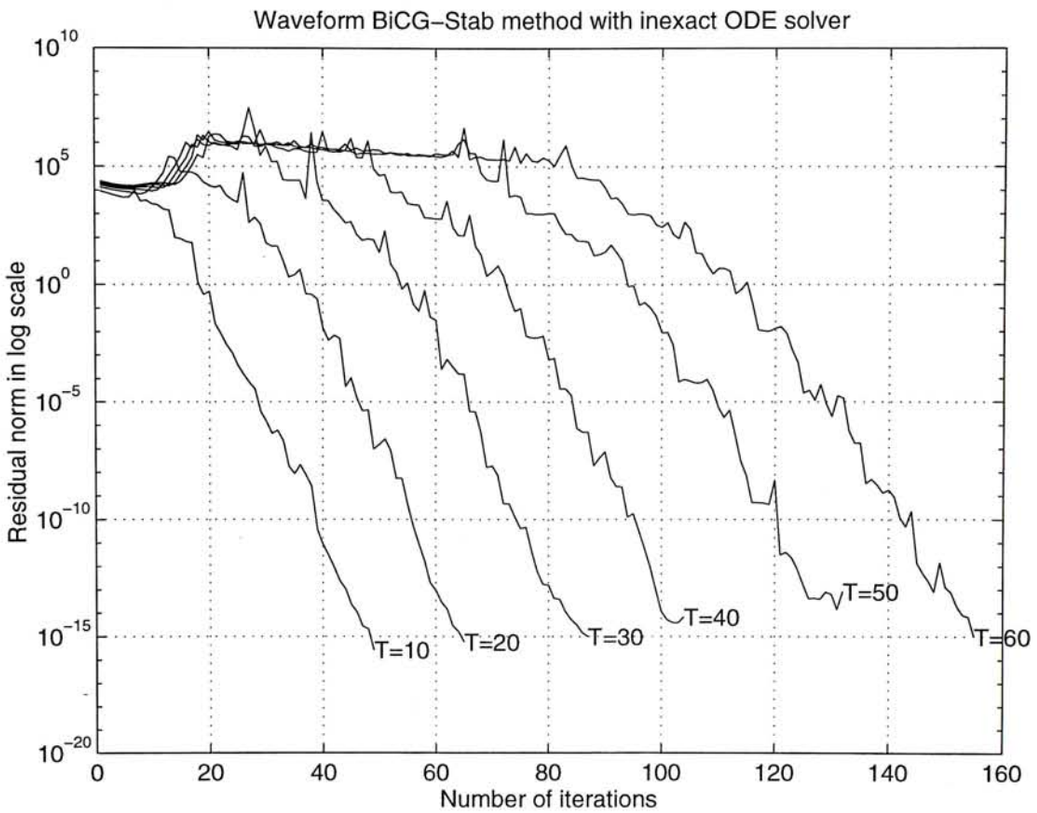

Figure 5.5: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform BiCG-STAB method.

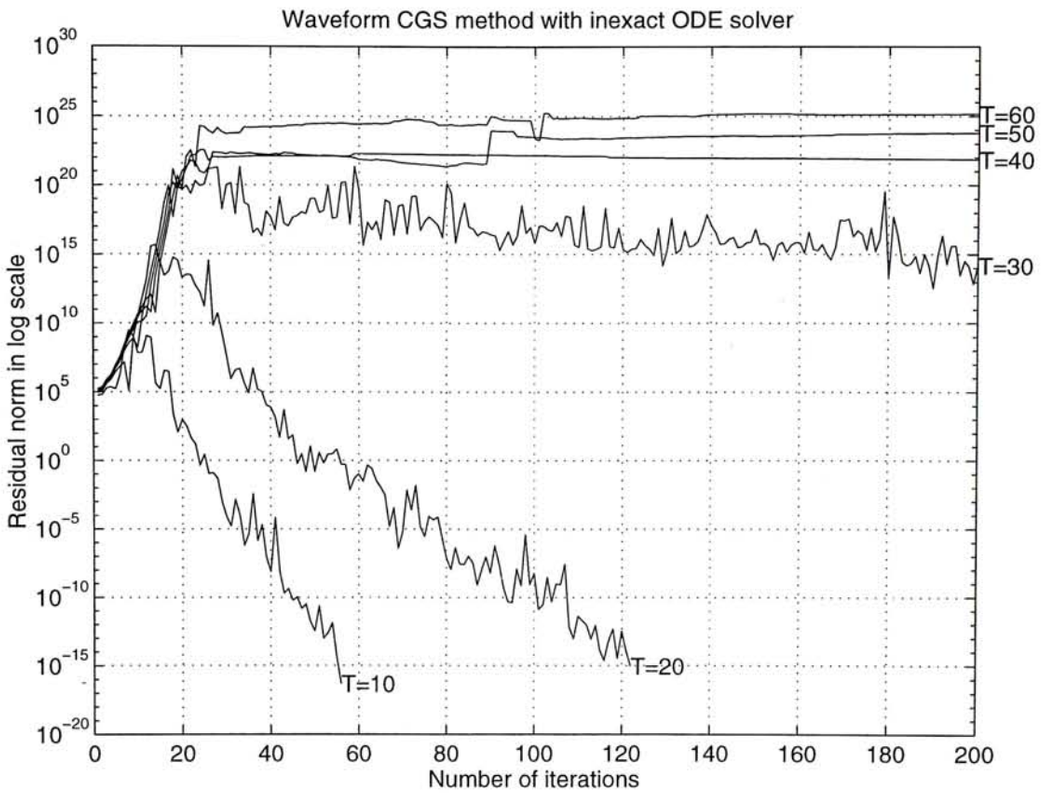

Figure 5.6: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform CGS method.

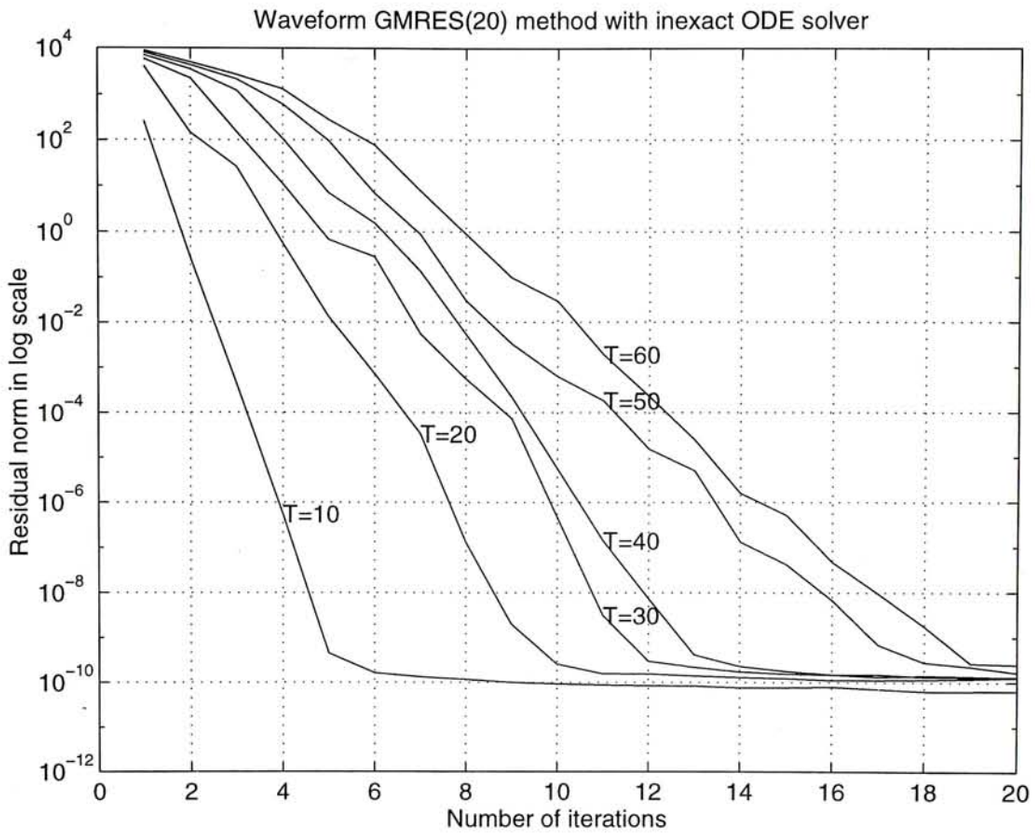

Figure 5.7: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform GMRES(20) method.

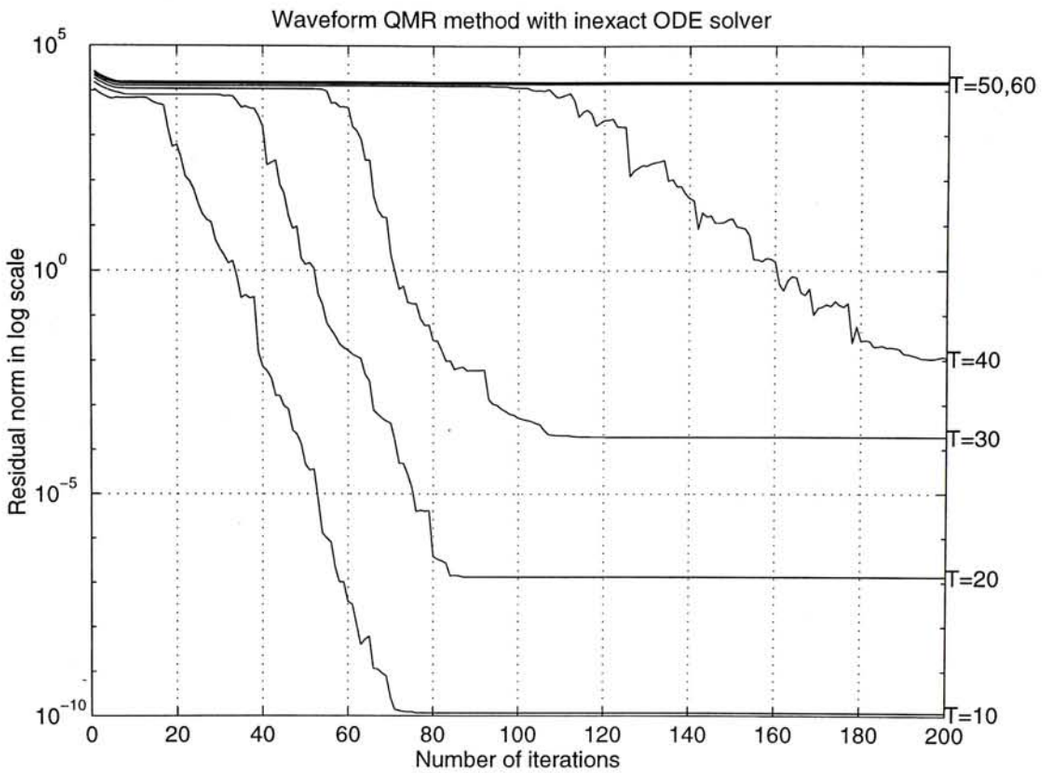

Figure 5.8: Log_{10} of the residual norm versus the number of iterations in different time intervals for the waveform QMR method.

Chapter 6

Domain Decomposition Technique

6.1 Introduction

Overlapped partitioning technique has been applied to waveform relaxation methods for solving large scale initial value problems that arise from VLSI circuit simulation [64, 76, 3]. The motivation is mainly to improve the robustness as well as convergence. Particularly for the tightly coupled systems such as bipolar circuits, the convergence of the naive waveform relaxation methods can be very poor. However, if we group the tightly coupled component together and the iterative process is only applied among the inter-subdomains, convergence will be improved. Although the optimal partitioning to meet certain criteria such as minimizing the spectral radius is believed to be NP-hard, some good heuristic algorithms have been reported [48, 3].

Another good reason for partitioning is concerned with parallel or distributed processing. A large circuit is divided into several sub-circuits and each sub-problem is distributed to a different processor. While minimizing the cut-set

for reducing the communication, we must maintain load-balancing. In this case, the problem of minimizing bisection cut-set was proven to be NP-complete. We refer the reader to the references [13] for the discussion of this issue.

In the following section, we will show how to integrate overlapped partitioning technique with waveform Krylov subspace methods by the domain decomposition setting. The methods are quite similar to the overlapping Schwarz methods for solving elliptic partial differential equations. Also, we propose the use of a differential operator $\mathcal{A}$ for generating a Krylov subspace sequence rather than the waveform relaxation operator $\mathcal{K}$. We conclude by a numerical experiment given in Section 6.3 to show that the differential operator is more appropriate in domain decomposition.

6.2 Overlapped Schwarz Methods

The domain decomposition algorithm described below works in the same way as one for linear algebraic equations [11, 13]. Let the domain Ω be divided into p overlapping subregions $\hat{\Omega}_1, \hat{\Omega}_2, \ldots, \hat{\Omega}_p$ such that $\bigcup_{i=1}^{p} \hat{\Omega}_i = \Omega$. The corresponding indices of the unknown vectors are denoted by $\hat{I}_1, \hat{I}_2, \ldots, \hat{I}_p$. Let $\hat{n}_i$ be the number of indices in $\hat{I}_i$. For each subdomain $\hat{\Omega}_i$, let R_i denote the $n \times \hat{n}_i$ restriction matrix that restricts a vector of $y(t)$ of length n to $R_i y(t)$ of length $\hat{n}_i$ by choosing the subvector having indices $\hat{I}_i$. The transpose of R_i is an extension matrix that extents the $y_i(t)$ back to $y(t)$. Thus

$$(R_i^T y_i(t))_k = \begin{cases} (y_i(t))_k & \text{if } k \in \hat{I}_i \\ 0 & \text{otherwise.} \end{cases}$$

Note that the matrices are not formed explicitly in the implementation. The additive Schwarz preconditioner is defined as:

$$\mathcal{M}^{-1} = \sum_{i=1}^{p} R_i^T \mathcal{A}_i^{-1} R_i.$$

where $(\mathcal{A}_i^{-1} R_i f)(t)$ represents the exact solution in subdomain $\hat{\Omega}_i$. In practice, we can use the inexact subdomain solver, such as incomplete LU factorization. The overlapping additive Schwarz method can be viewed as a generalized block Jacobi method. Similarly, the *overlapping multiplicative Schwarz method* represents a generalized block Gauss-Seidel method. That is, the solves on each subdomain are performed sequentially. In this thesis, we consider the additive method only since the convergence behaviors of the two methods are similar.

The convergence theory of domain decomposition methods for partial differential equations was studied quite maturely for the uniform grids. The discussion on the theory for unstructured grids can be found in a recent paper by Chan *et al.* [14]. For the circuit simulation, the convergence may rely on a good partitioning algorithm.

As is known for the case of linear algebraic problem, the convergence rate of this algorithm may deteriorate as the number of subdomain p increases and hence it may not be scalable. In the finite element context, this problem is successfully handled by inserting a coarse grid solver to the preconditioner, which acts as a mechanism for global communication of information. However, in the circuit simulation context, whether the "coarse grid" is meaningful is still in question. Here we only consider the overlapping Schwarz methods without coarse grid correction.

We notice that $\mathcal{M}^{-1}$ reasonablely approximates $\mathcal{A}^{-1}$. However, it may not well precondition the waveform relaxation operator $(\mathcal{I} - \mathcal{K})$. Hence, we attempt to use the differential operator $\mathcal{A}$ directly for generating the Krylov subspace sequence. The $(\mathcal{A}p)(t)$ product $w(t)$ corresponding to Equation 2.1 is given by:

$$w(t) := B(t)\dot{p}(t) + A(t)p(t).$$

Note that $\mathcal{A}$ is a pure differential operator and hence no integration method is applied. The step-size control mechanism then rely only on the subdomain

solvers.

In the following, we call the overlapping additive method for the waveform relaxation operator as Scheme 1 and the overlapping additive method for the differential operator as Scheme 2 respectively. In considering the multi-rate integration, Scheme 2 allows only each subdomain to have its own step-size. The interpolation occurs only at the interfaces between subdomains. In Scheme 1, more alignments are needed when updating. In the next section, we compare the convergence properties of these two schemes by numerical experiments.

6.3 Numerical Experiments

6.3.1 Delay Circuit

We consider the problem of solving the transient solution of the delay circuit shown in Figure 6.1, which is taken from [77, p. 618]. The values of the parameters are listed as follows: $C_1 = 0.0152$, $L_2 = 0.0451$, $C_3 = 0.0741$, $L_4 = 0.1016$, $C_5 = 0.1269$, $L_6 = 0.1499$, $C_7 = 0.1708$, $L_8 = 0.1916$, $C_9 = 0.2175$, $L_{10} = 0.2639$, $C_{11} = 0.3955$ and $R_L = 1.0$, which give an approximate one second delay time response. Then we cascade four parts of circuits, that each part is basically identical to the circuit shown in Figure 6.1, to form a four seconds delay circuit. Figure 6.2 shows the circuit diagram. The value of g_m is 1. The exact output waveforms are shown in Figure 6.3.

The circuit is naturally divided into four sub-domains. A small overlapping is introduced by including the load voltage (e.g. v_{11}) of the previous stage as the element of the current stage. The basic ODE solver is the Backward Euler method and the waveform relaxation operator for Scheme 1 is based on the waveform Jacobi method. The restart version of WGMRES was used and the restart value is 20.

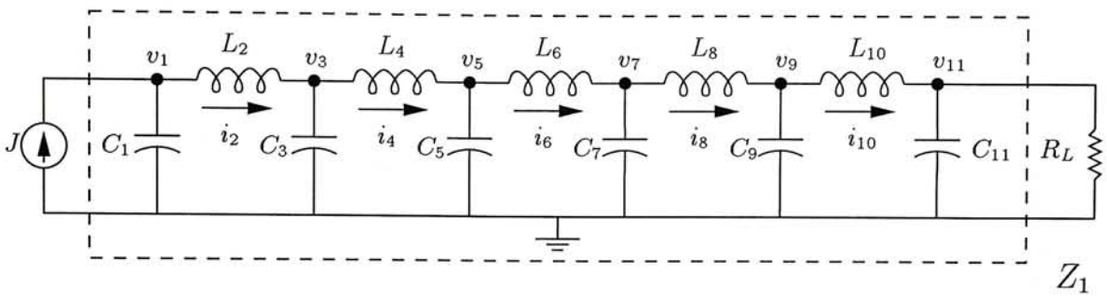

Figure 6.1: One second delay circuit

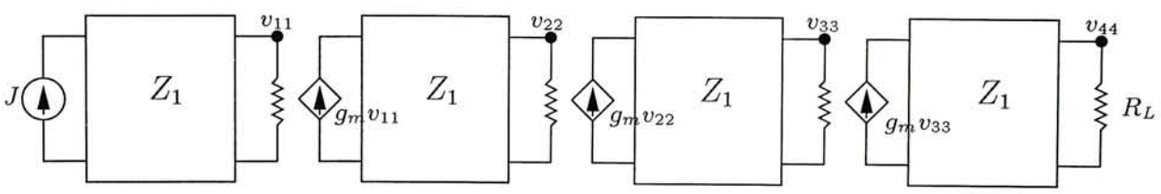

Figure 6.2: Four seconds delay circuit

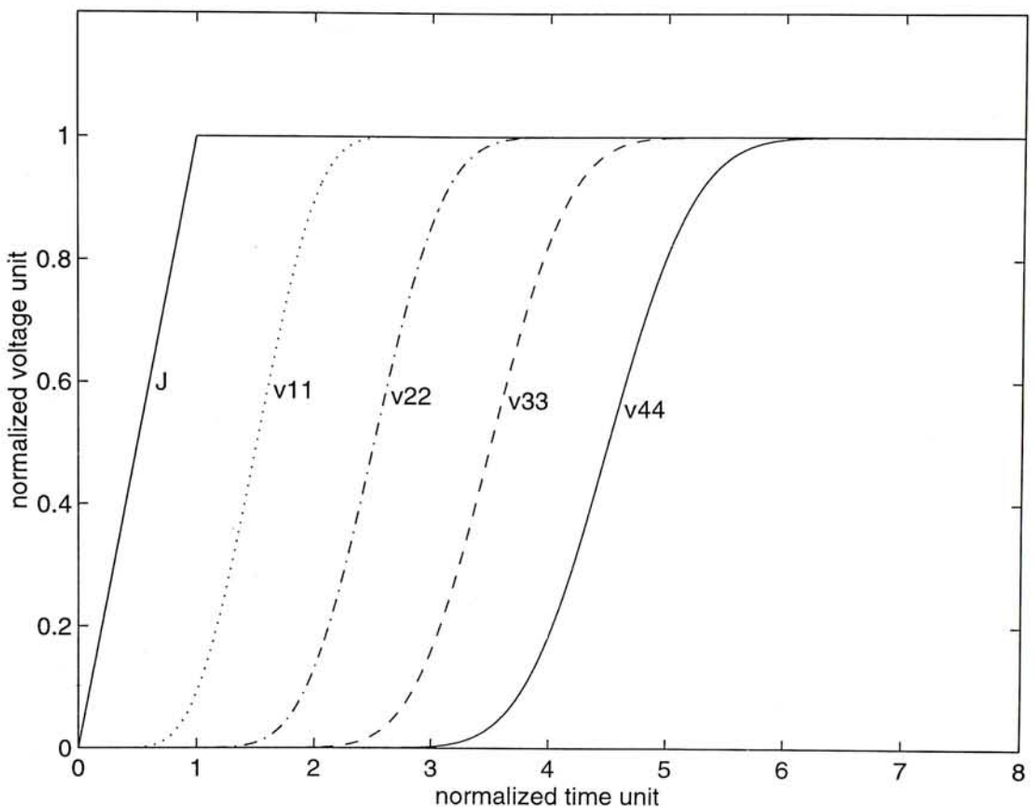

Figure 6.3: Output waveforms of the four seconds delay circuit

Table 6.1: The results of solving the transient solution of Delay circuit in term of number of operator-function products. (Time step = 0.01, Tolerance = 1e-8, T=2) The subdomain problems are solved by direct method.

methods	WBiCG	WBiCG-Stab	WCGS	WGMRES(20)	WQMR
no dd	230	breakdown	116	160	228
Scheme 1	fail	286	108	60	392
Scheme 2	12	6	4	20	12

The inner product as defined in Equation (3.4) was replaced by Equation (3.5) in the discretized system. The results are shown in Table 6.1.

As the result shown, the waveform Krylov subspace methods needed larger number of operator-function products in general when no domain decomposition was applied. Breakdown even occurred in the WBiCG-Stab method. When Scheme 1 was applied, however, the overall performance was poor. There are no great improvements in WCGS. The WQMR method needed more iterations to converge. Even worse, the WBiCG method failed to converge. Actually, this is the motivation that we investigate Scheme 2. In fact, there were only a few iterations needed in Scheme 2 for convergent. Of course in general the convergence depends on the spectrum of $\mathcal{M}^{-1}(\mathcal{I} - \mathcal{K})$ or $\mathcal{M}^{-1}\mathcal{A}$.

6.3.2 Unstructured Grid Problem

In our second experiment, we solve a dimensionless heat conducting equation in two space dimension on the unstructured mesh shown in Figure 6.4:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + g(x, y, t), \quad t \in (0, T) \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

The equation is discretized by a standard finite element method. The mesh is partitioned into eight nonoverlapping subdomains by *recursive spectral bisection* method (RSB) (see Appendix B for brief introduction to this method). The

Table 6.2: The results of solving the unstructured grid problem with Scheme 1 in term of number of operator-function products. (Time step = 1, Tolerance = 1e-8, 8 partitions). The subdomain problems are solved by direct method.

ovlp	WBiCG	WBiCG-Stab	WCGS	WGMRES(20)	WQMR
0	128	76	98	100	126
1	116	56	82	80	114
2	112	52	60	60	112
3	124	60	68	60	124
4	132	66	68	60	124

method was originally proposed by Pothen *et al.* [56]. To speed up the process of finding the *Fiedler vector* (see Appendix B for the definition of Fiedler vector), we employ the multilevel version of the RSB method [4]. Then the overlapping subdomains are obtained by extending one element from the boundary nodes. Figure 6.5 illustrates a subdomain with one-element extension. We can also apply the extension procedure again to obtain a larger overlapping subdomain. The subdomains were solved by direct method. The rest of the settings was the same as described in Chapter 3. Table 6.2 and Table 6.3 show the results of Schemes 1 and 2 respectively. In both tables, *ovlp*=0 means no overlapping and *ovlp*=k means that the extension procedure has been applied k times. Note that a considerable drop of the number of operator-function products was observed from *ovlp*=0 to *ovlp*=1. This characteristic is typical when domain decomposition is applied to elliptic partial differential equations. The reason for the considerable drop is that the information between subdomains can be communicated by sufficient small overlapping. However, the further enlarged overlapping may not significantly decrease the number of iterations, or even increase it. The optimal size of overlapping depends on the size of the meshes. In this experiment, Scheme 2 have only slightly better performance than Scheme 1.

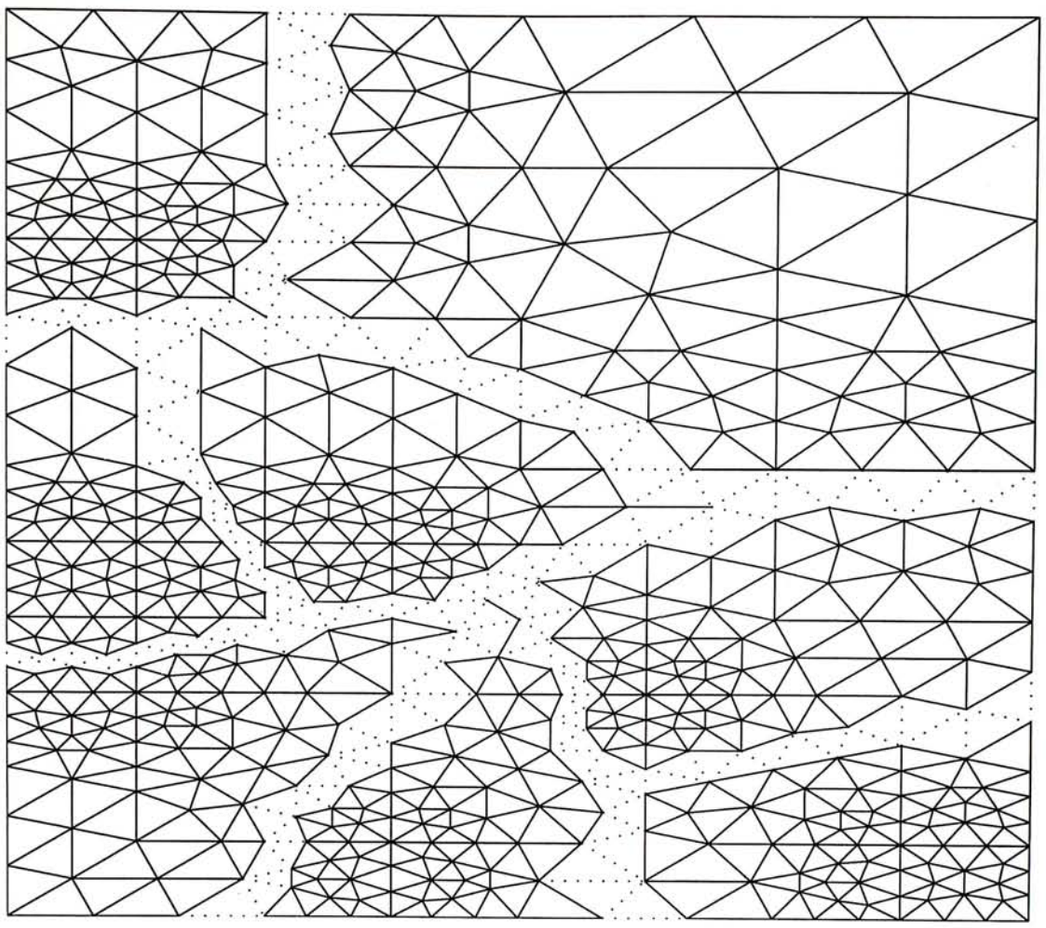

Figure 6.4: "Eppstein" grid partitioned into 8 subdomains

Table 6.3: The results of solving the unstructured grid problem with Scheme 2 in term of number of operator-function products. (Time step = 1, Tolerance = 1e-8, 8 partitions). The subdomain problems are solved by direct method.

ovlp	WBiCG	WBiCG-Stab	WCGS	WGMRES(20)	WQMR
0	130	82	92	100	130
1	86	50	56	60	84
2	82	42	52	60	82
3	74	40	40	40	74
4	76	40	44	40	76

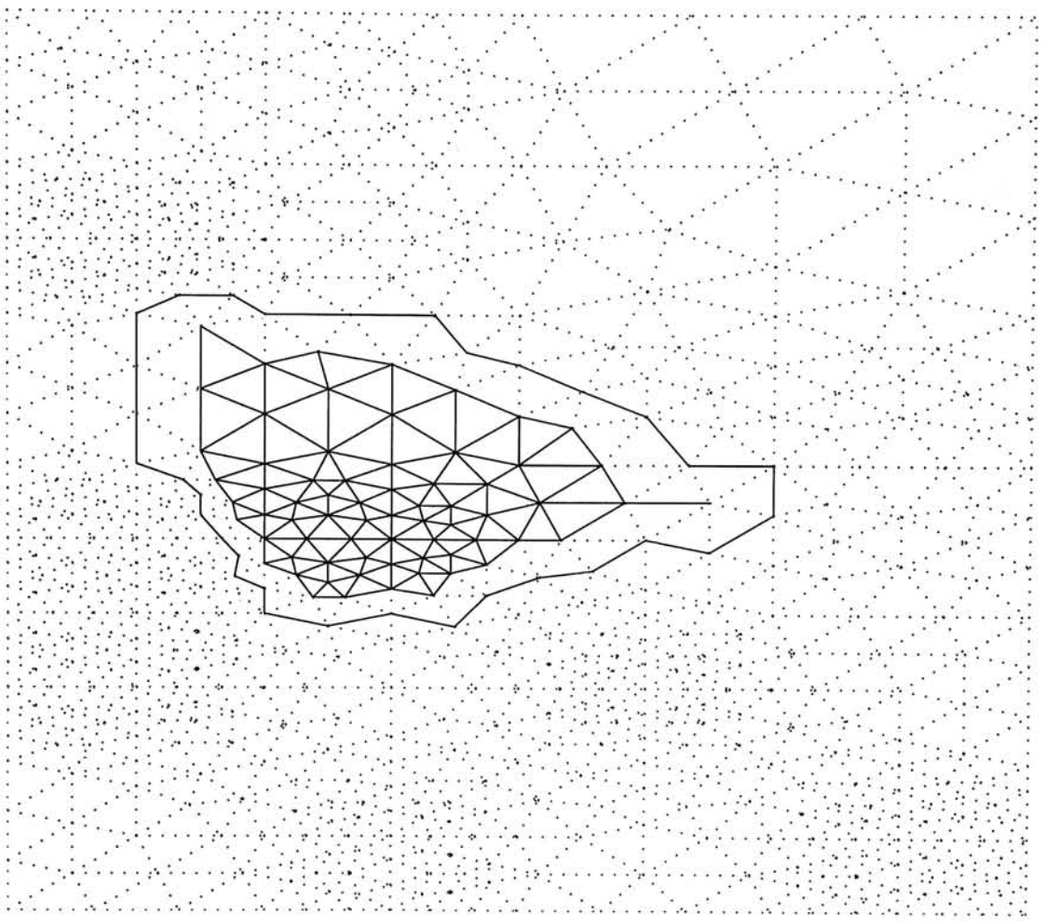

Figure 6.5: A subdomain with one-element extension

Chapter 7

Conclusions

7.1 Summary

Waveform methods are strongly supported by mathematical theories. They include functional analysis and operator theory. As you have seen in the previous chapters, the concepts and terminologies such as spectral radius and spectrum are throughout this thesis. Also as we treat the waveform relaxation methods in continuous time, the contraction mapping and fixed-pointed theorem in Banach space can be applied for convergence analysis. Note that not only the mathematical theories are useful for analysis, but also that they sometimes help us in design of algorithms. For example in this thesis, we have borrowed the idea of adjoint operator in Hilbert space for the design of the waveform bi-conjugate gradient method and the waveform quasi minimal residual method.

Another area that seems to be equally important in the design of waveform methods is Applied Linear Algebra. In this thesis, we have adopted the technique of Krylov subspace and the related techniques such as preconditioning and domain decomposition. The adoption is based on the generalization of the concept of matrices to linear operators in Hilbert space. Five previously untried

methods, together with the waveform generalized minimal residual method that was newly proposed by Lumsdaine *et al.* [43], have been investigated in Chapter 3. They include the waveform conjugate gradient method, the waveform bi-conjugate gradient method, the waveform conjugate gradient squared method, the waveform bi-conjugate gradient stabilized method and the waveform quasi minimal residual method.

These methods have been extensively examined via numerical experiment on both sequential and DECmpp 12000/Sx massively parallel computers. The test problems are taken from circuits and heat conducting equation on unstructured grids. One important evidence that we have observed from numerical experiments is that waveform Krylov subspace methods can be applied to tightly coupled systems. We also concluded that the convergence behaviors of those methods are similar to their non-functional counterparts. This is important because it makes the characteristics of the new methods more predictable as we have understood the traditional methods quite mutually well.

Although the new methods are the extension of the traditional methods, they do have their own properties that do not appear in the non-functional counterparts. In this thesis, we have pointed out that the waveform Krylov subspace methods which need adjoint operators may induce difficulty in actual implementation. The effect of the length of time interval has also been investigated.

In Chapter 5, we have developed data-parallel versions of waveform Krylov subspace methods on a DECmpp 12000/Sx computer. The programs were written in High Performance Fortran and hence can be portable to other platform. We have demonstrated the efficiency of the waveform Krylov subspace methods on parallel environments.

In Chapter 6, domain decomposition technique for waveform Krylov subspace methods was further investigated. This research was partially motivated by the fact that overlapped partitioning techniques for waveform relaxation were

studied in recent years. The techniques can be applied to the waveform Krylov subspace methods equally well by using domain decomposition and preconditioning. We have showed that domain decomposition can work well in function space. We also pointed out that the differential operator $\mathcal{A}$ is more suitable than the waveform relaxation operator $\mathcal{K}$ for generating the Krylov sequence if the overlapped Schwarz preconditioner has been used.

7.2 Future Works

In this thesis, we observed promising results of waveform Krylov subspace methods in solving large sparse ODE's. Therefore, in my opinion, these methods will be important in future research as well as industrial applications. Here, we identify some future works in this area.

Obviously, the practical implementation of waveform methods with the application of circuit simulation is worth being considered. This may be a cross-disciplinary project that invokes Computer Science, Electrical Engineering and Mathematics. Furthermore, the parallel implementation on distributed environments will be particular of interest, as the traditional methods may be inefficient on these environments. Beside the application of circuit simulation, waveform relaxation have been applied to other areas such as transmission line simulation and chemical processing. Hence, the more advanced techniques that we described in this thesis may be also suitable in these areas.

Beside the practical implementations, some theoretical researches are suggested. In this thesis, we mainly focused on the algorithms to handle the coupling in system domain. In theory, Krylov subspace methods can operate on any kinds of linear operators. Hence, intuitively the knowledge in time domain are not considered in Krylov subspace methods when the ODE equations are transformed into linear operators. On the other hand, some new methods such

as implicit Runge-Kutta method and wavelet method have been proposed recently for improvement in time domain. Nevertheless, at the same time the knowledge of the coupling has not been considered in these methods. Therefore, there is no conflict between these two approaches and the integration of these two approaches may further improve the overall performance.

Appendix A

Pseudo Codes for Waveform Krylov Subspace Methods

Compute $r(t)$ for some initial guess $y(t)$
Choose $\tilde{r}(t)$ (for example, $\tilde{r}(t) = r(t)$)
for $i = 1, 2, \ldots$
    $z(t) := (\mathcal{M}^{-1}r)(t)$
    $\tilde{z}(t) := (\mathcal{M}^{*-1}\tilde{r})(t)$
    $\rho := \langle z(t), \tilde{z}(t) \rangle$
    if $\rho = 0$ method fails
    if $i = 1$,
        $p(t) := z(t)$
        $\tilde{p}(t) := \tilde{z}(t)$
    else
        $\beta := \rho / \rho_1$
        $p(t) := z(t) + \beta p(t)$
        $\tilde{p}(t) := \tilde{z}(t) + \beta \tilde{p}(t)$
    end
    $q(t) := (\mathcal{A}p)(t)$
    $\tilde{q}(t) := (\mathcal{A}^*\tilde{p})(t)$
    $\alpha_i := \rho_{i-1} / \langle \tilde{p}(t), q(t) \rangle$
    $y(t) := y(t) + \alpha p(t)$
    $r(t) := r(t) - \alpha q(t)$
    $\tilde{r}(t) := \tilde{r}(t) - \alpha \tilde{q}(t)$
    $\rho_1 := \rho$
    check convergence; continue if necessary
end

Algorithm A.1: The Waveform Bi-Conjugate Gradient Method (WBiCG)

Compute $r(t)$ for some initial guess $y(t)$
Choose $\tilde{r}(t)$ (for example, $\tilde{r}(t) = r(t)$)
for $i = 1, 2, \ldots$
    $\rho := \langle \tilde{r}(t), r(t) \rangle$
    if $\rho = 0$ method fails
    if $i = 1$,
        $p(t) := r(t)$;
    else
        $\beta := (\rho/\rho_1)(\alpha/\omega)$
        $p(t) := r(t) + \beta(p(t) - \omega v(t))$
    end
    $\hat{p}(t) := (\mathcal{M}^{-1}p)(t)$
    $v(t) := (\mathcal{A}\hat{p})(t)$
    $\alpha := \rho/\langle \tilde{r}(t), v(t) \rangle$
    $s(t) := r(t) - \alpha v(t)$
    check $\|s(t)\|_2$; if small enough: set $y(t) := y(t) + \alpha\hat{p}(t)$ and stop
    $\hat{s}(t) := (\mathcal{M}^{-1}s)(t)$
    $d(t) := (\mathcal{A}\hat{s})(t)$
    $\omega := \langle d(t), s(t) \rangle / \langle d(t), d(t) \rangle$
    $y(t) := y(t) + \alpha\hat{p}(t) + \omega\hat{s}(t)$
    $r(t) := s(t) - \omega d(t)$
    $\rho_1 := \rho$
    check convergence; continue if necessary
    for continuation it is necessary that $\omega \neq 0$
end

Algorithm A.2: The Waveform Bi-Conjugate Gradient Stabilized Method (WBiCG-Stab)

Compute $r(t)$ for some initial guess $y(t)$
for $i = 1, 2, \ldots$
    $z(t) := (\mathcal{M}^{-1} r)(t)$
    $\rho := \langle r(t), z(t) \rangle$
    if $i = 1$,
        $p(t) := z(t)$
    else
        $\beta := \rho / \rho_1$
        $p(t) := z(t) + \beta p(t)$
    end
    $q(t) := (\mathcal{A} p)(t)$
    $\alpha_i := \rho / \langle p(t), q(t) \rangle$
    $y(t) := y(t) + \alpha p(t)$
    $r(t) := r(t) - \alpha q(t)$
    $\rho_1 := \rho$
    check convergence; continue if necessary
end

Algorithm A.3: The Waveform Conjugate Gradient Method (WCG)

Compute $r(t)$ for some initial guess $y(t)$
Choose $\tilde{r}(t)$ (for example, $\tilde{r}(t) = r(t)$)
for $i = 1, 2, \ldots$
    $\rho := \langle \tilde{r}(t), r(t) \rangle$
    if $\rho = 0$ method fails
    if $i = 1$,
        $u(t) := r(t)$
        $p(t) := u(t)$
    else
        $\beta := \rho / \rho_1$
        $u(t) := r(t) + \beta q(t)$
        $p(t) := u(t) + \beta(q(t) + \beta p(t))$
    end
    $\hat{p}(t) := (\mathcal{M}^{-1} p)(t)$
    $\hat{v}(t) := (\mathcal{A}\hat{p})(t)$
    $\alpha := \rho / \langle \tilde{r}(t), \hat{v}(t) \rangle$
    $q(t) := u(t) - \alpha \hat{v}(t)$
    $\hat{u}(t) := (\mathcal{M}^{-1}(u + q))(t)$
    $y(t) := y(t) + \alpha \hat{u}(t)$
    $\hat{q}(t) := (\mathcal{A}\hat{u})(t)$
    $r(t) := r(t) - \alpha \hat{q}(t)$
    $\rho_1 := \rho$
    check convergence; continue if necessary
end

Algorithm A.4: The Waveform Conjugate Gradient Squared Method (WCGS)

$y^{(0)}(t)$ is an initial waveform
for $j = 1, 2, \ldots$
    $r(t) := \mathcal{M}^{-1}(f(t) - (\mathcal{A}y^{(0)})(t))$
    $v^{(1)}(t) := r(t)/\|r(t)\|_2$
    $s := \|r(t)\|_2 e_1$
    for $i = 1, 2, \ldots, m$,
        $w(t) := (\mathcal{M}^{-1}\mathcal{A}v^{(i)})(t)$
        $h_{k,i} := \langle w(t), v^{(k)}(t) \rangle$
        $w(t) := w(t) - h_{k,i}v^{(k)}(t)$
    end
    $h_{i+1,i} := \|w(t)\|_2$
    $v^{(i+1)}(t) := w(t)/h_{i+1,i}$
    Solve $H_k y = s$
    $y(t) := y^{(0)}(t) + V_k y$;
    check convergence; continue if necessary
end

Algorithm A.5: The Waveform GMRES(m) Method (WGMRES(m))

Compute $r(t)$ for some initial guess $x(t)$
$\tilde{v}(t) := r(t)$; $\tilde{y}(t) := (\mathcal{M}_1^{-1}\tilde{v})(t)$; $\rho := \|\tilde{y}(t)\|_2$
Choose $\tilde{w}(t)$ (for example, $\tilde{w}(t) = r(t)$)
$z(t) := (\mathcal{M}_2^{-1}\tilde{w})(t)$; $\xi := \|z(t)\|_2$
$\gamma := 1$; $\eta := -1$; $\theta := 0$
for $i = 1, 2, \ldots$
    if $\rho = 0$ or $\xi = 0$ method fails
    $v(t) := \tilde{v}(t)/\rho$; $\tilde{y}(t) := \tilde{y}(t)/\rho$
    $w(t) := \tilde{w}(t)/\xi$; $z(t) := z(t)/\xi$
    $\delta := \langle z(t), \tilde{y}(t)\rangle$; if $\delta = 0$ method fails
    $\tilde{y}(t) := (\mathcal{M}_2^{-1}y)(t)$; $\tilde{z}(t) := (\mathcal{M}_1^{*-1}z)(t)$
    if $i = 1$
        $p(t) := \tilde{y}(t)$; $q(t) := \tilde{z}(t)$
    else
        $p(t) := \tilde{y}(t) - (\xi\delta/\epsilon)p(t)$
        $q(t) := \tilde{z}(t) - (\rho\delta/\epsilon)q(t)$
    end
    $\tilde{p}(t) := (\mathcal{A}p)(t)$
    $\epsilon := \langle q(t), \tilde{p}(t)\rangle$; if $\epsilon = 0$ method fails
    $\beta := \epsilon/\delta$; if $\beta = 0$ method fails
    $\tilde{v}(t) := \tilde{p}(t) - \beta v(t)$
    $\tilde{y}(t) := (\mathcal{M}_1^{-1}\tilde{v})(t)$
    $\rho_1 := \rho$; $\rho := \|\tilde{y}(t)\|_2$
    $\tilde{w}(t) := (\mathcal{A}^*q)(t) - \beta w(t)$
    $z(t) = (\mathcal{M}_2^{*-1}\tilde{w})(t)$
    $\xi := \|z(t)\|_2$
    $\gamma_1 := \gamma$; $\theta_1 := \theta$
    $\theta := \rho/(\gamma_1|\beta|)$; $\gamma := 1/\sqrt{1+\theta^2}$; if $\gamma = 0$ method fails
    $\eta := -\eta\rho_1\gamma^2/(\beta\gamma_1^2)$
    if $i = 1$
        $d(t) := \eta p(t)$; $s(t) := \eta\tilde{p}(t)$
    else
        $d(t) := \eta p(t) + (\theta_1\gamma)^2 d(t)$
        $s(t) := \eta\tilde{p}(t) + (\theta_1\gamma)^2 s(t)$
    end
    $x(t) := x(t) + d(t)$; $r(t) := r(t) - s(t)$
    check convergence; continue if necessary
end

Algorithm A.6: The Waveform Quasi Minimal Residual Method (WQMR) without Look-ahead

Appendix B

Overview of Recursive Spectral Bisection Method

For solving unstructured grid problems on MIMD (*Multiple Instruction-streams Mulitple Data-streams*) computers, it is commonly that a mesh is firstly partitioned into pieces and then each piece is assigned to each processor. The partitioning problem here is to partition the mesh in the way that the communication overhead is to be minimized, while the load balancing should be maintained. The corresponding idealized mathematical graph problem can be formulated as finding a minimum cut bisection of an undirected graph. In this case, it is well-known that the problem is NP-complete. One of the most successful heuristic algorithms is the recursive spectral bisection method (RSB) [56]. In this appendix, we describe the method below.

Given an undirected, connected graph $G = (V, E)$. Assume that $|V| = n$ is even. Let L be the set of lattices vectors with components equals 1 or -1:

$$L = \{l \in \mathbb{R}^n \,|\, l_i \in \{-1, 1\}\}.$$

Let B be the set of *load balanced* vectors defined as:

$$B = \{b \in \mathbb{R}^n \mid \sum_{i=1}^{n} b_i = 0\}.$$

The set of bisection vectors, denoted by P, is defined as $L \cap B$. We associate each node of the graph with a variable x_i, whose value is 1 or -1 corresponding the two sides of the cut. The size of the cut set corresponding to a bisection vector x can be expressed as

$$|C|_{x \in P} = \frac{1}{4} \sum_{(v,w) \in E} (x_v - x_w)^2.$$

An $n \times n$ *Laplacian matrix* $Q = (q_{i,j})$ of G is defined as:

$$q_{i,j} = \begin{cases} -1 & \text{if } (v_i, v_j) \in E \\ \deg(v_i) & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Then it can be easily shown that

$$x^T Q x = \sum_{(v,w) \in E} (x_v - x_w)^2.$$

The Laplacian matrix Q has the property of positive semi-definite, i.e., $z^T Q z \geq 0$ for all $z \in \mathbb{R}^n$. The smallest eigenvalue of Q is equal to zero with the associated eigenvector $(1, 1, \ldots, 1)^T$. The eigenvector associated with the smallest nonzero eigenvalue λ_2 is called the *Fiedler vector*. The trick of the RSB method is to relax the constraint of $x \in P$ to the continuous space B. We have the following inequality:

$$\min_{x \in P} |C| = \frac{1}{4} \min_{x \in P} x^T Q x \geq \frac{1}{4} \min_{x \in B, \|x\|_2^2} x^T Q x = \frac{1}{4} x_2^T Q x_2 = \frac{n}{2} \lambda_2,$$

where x_2 is the Fielder vector normalized as $\|x_2\|_2^2 = n$. Since x_2 is orthogonal to $(1, 1, \ldots, 1)^T$, it is automatically in B. The minimum cut problem is then

relaxed to finding the eigenvector with the second smallest eigenvalue of the Laplacian matrix. After finding x_2, we need to map x_2 to the "nearby" vector $p \in P$ to get the solution. The heuristic is that when x_2 minimizes $x^T Q x$, it is reasonably believed that p is also to be a good solution. The "nearby" vector is commonly given by the *median cut* method, i.e., find the median value of the components of x_2 and map the components above that median value to +1 and map the components below that median value to -1. We denote the median cut vector as p_m. Recently, Chan *et al.* proved that p_m is in fact the nearest bisection vector of x_2 in any l_s-norm [12].

Bibliography

[1] V. G. Bandi and H. Asai. Acceleration techniques for waveform relaxation analysis of RLCG transmission lines driven by bipolar logic gates. *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, E76-A(9):1527–1534, Sept. 1993.

[2] V. G. Bandi and H. Asai. A waveform relaxation method applicable to the simulation of ECL circuits with gate level partitioning. *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, E76-A(4):657–660, April 1993.

[3] V. G. Bandi and H. Asai. Dynamically overlapped partitioning technique to implement waveform relaxation simulation of bipolar circuits. *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, E77-A(6):1080–1084, June 1994.

[4] S. T. Barnard and H. D. Simon. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Currency: Practice and Experience*, 6(2):101–117, April 1994.

[5] Richard Barrett et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, 1994.

[6] A. Bellen, R. Vermiglio, and Zennaro. Parallel ODE–solvers with stepsize control. *Journal of Computational and Applied Mathematics*, 31:277–293, 1990.

[7] A. Bellen and M. Zennaro. The use of Runge-Kutta formulae in waveform relaxation methods. *Appl. Numer. Math.*, 11(1-3):95–114, Jan. 1993.

[8] W. Beyene. Bi-level waveform relaxation analysis of package interconnects using scattering parameters. In *Proceedings of the 1992 International Symposium on Microelectronics*, pages 156–161, San Francisco, CA, USA, 1992.

[9] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-algebraic Equations.* SIAM, 1996.

[10] R. W. Brockett. *Finite Dimensional Linear Systems.* John Wiley and Sons, 1970.

[11] Xiao-Chuan Cai and Youcef Saad. Overlapping domain decomposition algorithms for general sparse matrices. Technical report, University of Minnesota, March 1993.

[12] T. F. Chan, P. Ciarlet, Jr., and W.-K Szeto. On the optimality of the median cut spectral bisection graph partitioning method. *SIAM J. Sci. Comput.*, to appear.

[13] Tony F. Chan and Tarek P. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.

[14] Tony F. Chan, Barry F. Smith, and Jun Zou. Overlapping Schwarz methods on unstructured meshes usnig non-matching coarse grids. *Numerische Mathematik*, to appear.

[15] F.-Y. Chang. The generalized method of characteristics for waveform relaxation analysis of lossy coupled transmission lines. *IEEE Trans. Microwave Theor. Tech.*, 37(12):2028–2038, Dec. 1989.

[16] F.-Y. Chang. The generalized method of characteristics for waveform relaxation analysis of lossy coupled transmission lines. In *IEEE 1989 MTT-S Int. Microwave Symp. Digest*, Long Beach, CA, USA, 1989.

[17] F.-Y. Chang. Waveform relaxation analysis of RLCG transmission lines. *IEEE Trans. Circuits Syst.*, 37(11):1394–1415, Nov. 1990.

[18] F.-Y. Chang. Waveform relaxation analysis of nonuniform lossy transmission lines characterized with frequency-dependent parameters. *IEEE Trans. Circuits Syst.*, 38(12):1484–1500, Dec. 1991.

[19] F.-Y. Chang. Transient simulation of nonuniform coupled lossy transmission lines characterized with frequency-dependent parameters. I. waveform relaxation analysis. *IEEE Trans. Circuits Syst. 1, Fundam. theory appl.*, 39(8):585–603, Aug. 1992.

[20] F.-Y. Chang, R. Wang, and O. Wing. Bi-level waveform relaxation simulation of transients in high-speed electronic packaging designs. In *Proc. 1991 Inter. Electron. Packaging Conf.*, San Diego, CA, USA, 1991.

[21] M. L. Crow and M. Ilic. The waveform relaxation algorithm for systems of differential/algebraic equations with power system applications. In *Proceedings of the 1989 American Control Conference*, Pittsburgh, PA, USA, 1989.

[22] M. L. Crow and M. Ilic. The parallel implementation of the waveform relaxation method for the simulation of structure-preserved power systems. In *Proc. IEEE Int. Symp. Circuits Syst.*, New Orleans, LA, USA, 1990.

[23] M. L. Crow, M. D. Ilic, and J. K. White. Convergence properties of the waveform relaxation method as applied to electric power systems. In *Proc. IEEE Int. Symp. Circuits Syst.*, pages 1863–1866, Portland, OR, USA, 1989.

[24] Ruth F. Curtain and Hans Zwart. *An introduction to infinite-dimensional linear systems theory.* Springer Verlag, 1995.

[25] J.J.B. de Swart, W.A. van der Veen, and B.P. Sommeijer. Test set for IVPODE and IVPDAE solvers. Avalible via Ftp (`ftp.cwi.nl:/pub/IVPtestset`), Jan. 1995.

[26] Digital Equipment Corporation. *DECmpp 12000 Sx - System Overview Manual*, 1993.

[27] Digital Equipment Corporation. *DECmpp Sx - High Performance Fortran User's Guide*, 1993.

[28] Digital Equipment Corporation. *DECmpp Sx Programming Language (ANSI) User's Guide*, 1993.

[29] V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21:315–319, 1984.

[30] C. W. Gear. Parallel methods for ordinary differential equations. Technical Report UIUCDCS–R–87–1369, Comp. Sci. Dept., University of Illinois, Urbana, 1987.

[31] Robert K. Gjertsen. Parallel graph coloring heuristics. Master's thesis, the Graduate College of the University of Illinois, 1994.

[32] W. Hackbusch. *Iterative solution of large sparse systems of equations.* Springer-Verlag, 1994.

[33] J. Janssen and S. Vandewalle. Multigrid waveform relaxation on spatial finite element meshes. In *Contributions to Multigrid*, volume 103 of *CWI Tracts*, pages 75–86, Amsterdam, 1994. CWI.

[34] Jan Janssen and Stefan Vandewalle. Multigrid waveform relaxation on spatial finite element meshes: The discrete-time case. *SIAM J. Numer. Anal.*, 17(1):to appear, Jan. 1996.

[35] Jan Janssen and Stefan Vandewalle. Multigrid waveform relaxation on spatial finite element meshes: The continuous-time case. *SIAM J. Numer. Anal.*, 33(2):456–474, April 1996.

[36] W. John and W. Rissiek. Application of the waveform relaxation method to determine reflection and crosstalk noise on signal lines. In *Proceedings. VLSI and Computer Peripherals. VLSI and Microelectronic Applications in Intelligent Peripherals and their Interconnection Networks*, Hamburg, West Germany, 1989. IEEE Comput. Soc. Press.

[37] M. T. Jones and P. E. Plassmann. A parallel graph coloring heuristic. *SIAM J. on Sci. Comput.*, 14(3):654–669, May 1993.

[38] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations.* Frontiers in Applied Mathematics. SIAM, 1995.

[39] F.-C. M. Lau and E. M. Deeley. Waveform relaxation analysis of cascaded transmission lines. *International Journal of Electronics*, 76(3):483–495, March 1994.

[40] E. Lelarasmee, A. Ruehli, and A. L. Sangiovanni-Vincentelli. The waveform relaxation method for time-domain analysis of large scale integrated circuits. *IEEE Trans. on CAD of IC and Sys.*, 1(3):131–145, July 1982.

[41] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, pages 1036–1053, 1986.

[42] A. Lumsdaine, M. Reichelt, and J. White. Conjugate direction waveform methods for transient two-dimensional simulation of MOS devices. In *Proc. IEEE Int. Conf. Comput. Aided Des.*, pages 116–119, Santa Clara, CA, USA, 1991.

[43] A. Lumsdaine, J. M. Squyres, and M. W. Reichelt. Waveform iterative methods for parallel solution of initial value problems. In *Scalable Parallel Libraries Conference*, Mississippi State, MS, Oct. 1994.

[44] Andrew Lumsdaine and Deyun Wu. Spectra and pseudospectra of waveform relaxation operators. *SIAM J. Sci. Comput.*, page to appear, 1996.

[45] Jun-Fa Mao and Zheng-Fan Li. Time-domain response analysis of nonuniform multiconductor transmission lines by the waveform relaxation method. *Acta Electronica Sinica*, 21(9):64–69, Sept. 1993.

[46] U. Miekkala and O. Nevanlinna. Convergence of dynamic iteration methods for initial value problems. *SIAM J. Sci. Stat. Comput.*, 8(4), July 1987.

[47] U. Miekkala and O. Nevanlinna. Quasinilpotency of the operators in Picard-Lindelöf iteration. *Numer. Funct. Anal. Optim.*, 13(1-2):203–221, 1992.

[48] M. E. Mokari-Bolhassan and T. N. Trick. A new iterative algorithm for the solution of large systems. In *Proceedings of the 28th Midwest Symposium on Circuits and Systems*, Kentucky, Aug. 1985.

[49] O. Nevanlinna. Remarks on Picard-Lindelöf iteration I. *BIT*, 29(2):328–346, 1989.

[50] O. Nevanlinna. Remarks on Picard-Lindelöf iteration II. *BIT*, 29(3):535–562, 1989.

[51] O. Nevanlinna. Linear acceleration of Picard-Lindelöf iteration. *Numer. Math.*, 57(2):147–156, 1990.

[52] O. Nevanlinna. Power bounded prolongations and Picard-Lindelöf iteration. *Numer. Math.*, 58(5):479–501, 1990.

[53] B. N. Parlett, D. R. Taylor, and Z. A. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Mathematics of Computation*, 44:105–124, 1985.

[54] W. M. Patterson. *Iterative Methods for the Solution of a Linear Operator Equation in Hilbert Space - A Survey.* Number 394 in Lecture Notes in Mathematics. Springer-Verlag, 1974.

[55] L. C. Piccinini, G. Stampacchia, and G. Vidossich. *Ordinary Differential Equations in R^n.* Springer-Verlag, 1984.

[56] A. Pothen, H. Simon, and K. P. Liou. Partitioning sparse matrices with eigenvector of graph. *SIAM J. Matrix Anal. Appl.*, 11(3):430–452, 1990.

[57] S. Qi and Q. Yang. Coupling fast Walsh transform with waveform relaxation technique to analyze lossy coupled transmission lines. In *1992 IEEE MTT-S International Microwave Symposium Digest*, volume 2, pages 897–898, Albuquerque, NM, USA, 1992.

[58] S. Qi and Q. Yang. Coupling fast Walsh transform with waveform relaxation technique to analyze lossy coupled transmission lines. In *Proceedings of the*

35th Midwest Symposium on Circuits and Systems, volume 1, pages 605–607, Washington, DC, USA, 1992.

[59] S. Raman and L. M. Patnaik. HIRECS: hypercube implementation of relaxation-based circuit simulation. *International Journal of High Speed Computing*, 1(3):399–432, Sept. 1989.

[60] M. Reichelt, J. White, et al. Waveform relaxation applied to transient device simulation. In *Proc. IEEE Int. Symp. Circuits Syst.*, Espoo, Finland, 1988.

[61] P. Saviz and O. Wing. Dynamic circuit restructuring for hierarchical waveform relaxation. In *Proc. IEEE Int. Symp. Circuits Syst.*, New Orleans, LA, USA, 1990.

[62] A. R. Secchi, M. Morari, and Jr. Biscaia, E. C. The waveform relaxation method in the concurrent dynamic process simulation. *Computers & Chemical Engineering*, 17(suppl. issue):S453–456, 1993.

[63] A. R. Secchi, M. Morari, and Jr. Biscaia, E. C. The waveform relaxation method in the concurrent dynamic process simulation. *Computers & Chemical Engineering*, 17(7):683–704, July 1993.

[64] T. Shima and Y. Kamatani. A circuit simulator based on the waveform-relaxation method using selective overlapped partition and classified latencies. In *Proc. IEEE Int. Symp. Circuits Syst.*, page 2915, Espoo, Finland, 1988.

[65] S. Tomboulian and M. Pappas. Indirect addressing and load balancing for faster solution to mandelbrot set on simd architectures. In *The Third Symposium on the Frontiers of Massively Parallel Computation*, College Park, MD, Oct. 1990.

[66] A. J. van der Steen. Overview of recent supercomputers. Technical report, Academic Computing Centre Utrecht, 1994. Available via NETLIB.

[67] S. Vandewalle. Waveform relaxation methods for solving parabolic partial differential equations. In *Proceedings of the Fifth Distributed Memory Computing Conference*, pages 575–584, Los Alamitos, CA, 1990. IEEE.

[68] S. Vandewalle. *The Parallel Solution of Parabolic Partial Differential Equations by Multigrid Waveform Relaxation Methods.* PhD thesis, Katholieke Universiteit Leuven, Belgium, 1992.

[69] S. Vandewalle. *Parallel Multigrid Waveform Relaxation for Parabolic Problems.* B. G. Teubner Verlag, Stuttgart, 1993.

[70] S. Vandewalle and R. Piessens. Multigrid waveform relaxation for solving parabolic partial differential equations. In W. Hackbusch and U. Trottenberg, editors, *Multigrid Methods III*, volume 98 of *International Series of Numerical Mathematics*, pages 377–388, Basel, 1991. Birkhäuser.

[71] S. Vandewalle and R. Piessens. Numerical experiments with nonlinear multigrid waveform relaxation on a parallel processor. *Appl. Numer. Math.*, 8:149–161, 1991.

[72] S. Vandewalle and R. Piessens. On dynamic iteration methods for solving time–periodic differential equations. In T. A. Manteuffel and S. F. McCormick, editors, *Preliminary Proceedings of the Fifth Copper Mountain Conference on Multigrid Methods*, volume 2, pages 357–375, Denver, 1991. University of Colorado.

[73] S. Vandewalle and R. Piessens. Efficient parallel algorithms for solving initial–boundary value and time–periodic parabolic partial differential equations. *SIAM J. Sci. Stat. Comput.*, 13(6):1330–1346, November 1992.

[74] S. Vandewalle and R. Piessens. On dynamic iteration methods for solving time-periodic differential equations. *SIAM J. Numer. Anal.*, 30:286–303, 1993.

[75] S. Vandewalle and D. Roose. The parallel waveform relaxation multigrid method. In G. Rodrigue, editor, *Parallel Processing for Scientific Computing*, pages 152–156. SIAM, Philadelphia, 1989.

[76] R. Wang and O. Wing. Waveform relaxation on tightly coupled systems. In *Proc. IEEE Int. Symp. Circuits Syst.*, Singapore, 1991.

[77] L. Weinberg. *Network Analysis and Synthesis.* McGraw-Hill, 1962.

[78] J. White and F. Odeh. Connecting waveform relaxation convergence properties to the A-stability of multirate integration methods. *Int. J. Comput. Math. Electr. Electron. Eng.*, 10(4):497–508, Dec. 1991.

[79] J. K. White and A. L. Sangiovanni-Vincentelli. *Relaxation techniques for the simulation of VLSI circuits.* Kluwer Academic Publishers, 1986.

[80] E. Zeidler. *Applied functional analysis : applications to mathematical physics.* Springer-Verlag, 1994.