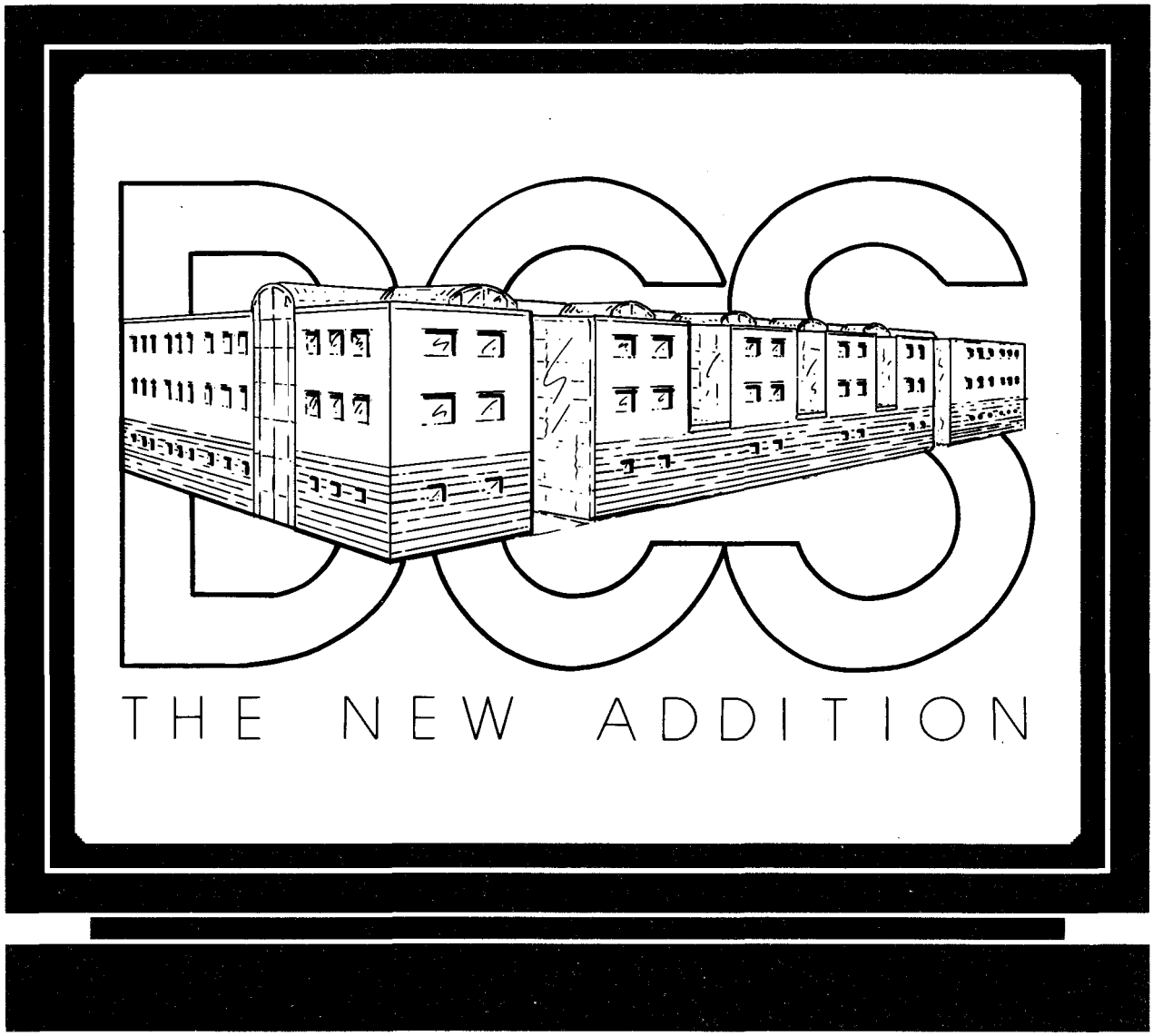
DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN



REPORT NO. UIUCDCS-R-90-1563

UILU-ENG-90-1701
DOE/ER/25026/34

WAVEFORM METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS

by

Fen-Lien Juang

January 1990

DOE/ER/25026--34

DE90 006593

REPORT NO. UIUCDCS-R-90-1563

WAVEFORM METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS

by

Fen-Lien Juang

January 1990

DEPARTMENT OF COMPUTER SCIENCE
1304 W. SPRINGFIELD AVENUE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, IL 61801

Supported in part by DOE Grant DEFG0287ER25026 and
submitted in partial fulfillment of the requirements of
the Graduate College for the degree of Doctor of Philosophy.

This work is lovingly dedicated to my parents

Kuei-Wei Juang and Yeh Hsieh,

my grandmother, Hsiao-Luan Hsieh

and my husband, Hsin-Fong Chen.

Acknowledgement

I would like to express my deep appreciation and gratitude to my thesis advisor, Professor C. William Gear, for his guidance and financial support during the years of my graduate study. I am also grateful to my advisor for his careful reading of the preliminary drafts and numerous comments during the preparation of this thesis. I also like to thank the other members on my committee, Professors Sameh, Saylor, Skeel, and Gallivan for their interest in my work and to all who have helped me throughout my academic life.

Thanks to the former and current occupants in 31 corridor, Ben, Dan, Jerry, Jim, John, Mike, Ren-Li, Ruth, Steve, Tam, Tom, Tony and Mr. Xu for their friendship. Special thanks to Sohail for his expert help in so many things.

I thank my parents and my grandmother for a lifetime of love and support. Finally, I am glad to thank my husband Hsin-Fong not only for his suggestion on the content and style of this thesis but also for his constant encouragement and support.

Contents

Chapter 1

Introduction

Large dynamical systems are likely to be described by variables that change at very different rates. The traditional approach to solving this kind of problem is to discretize all the variables with an identical mesh. This forces one to choose a mesh that is fine enough to accurately reflect the behavior of the most rapidly changing variable. Stiffness may force the use of implicit integration methods. The application of an implicit method involves the solution of a system of nonlinear algebraic equations. Due to the large dimension of a big system and the size of a fine mesh, an enormous processing time is inevitable. In addition to the cost consideration, real time and interactive applications require fast response. Approaches to reduce the processing time by taking advantage of the multirate property of a large system of ODEs have been suggested by Gear and Wells in [3, 4, 14]. These approaches are, first, to partition a big system into several subsystems, then to solve each subsystem *independently* (i.e. using different time steps or different methods for each subsystem). These approaches are called *multirate integration methods.*

One major problem of classical multirate integration (as discussed, for example, in [4, 14]) is the overhead of coordination between the subsystems. The coordination is needed at each step to pass current information between subsystems. The waveform relaxation method (see [7, 8, 15, 16]) is an approach which circumvents these problems by iterating the integrations using the old information from other subsystems. In this thesis, the waveform iteration will be studied under the assumption that the integrations are performed exactly.

Waveform relaxation has been shown to converge superlinearly on finite intervals by Nevan-

linna in [11]. In this thesis the order of accuracy of solutions generated by the waveform relaxation method is discussed. The order of accuracy of an iterate is characterized by the number of correct terms in its Taylor series. (A term in an iterate's Taylor series is correct if it matches the corresponding term in the true solution's Taylor series.) We will show that the accuracy at each step of the iteration is at least one order higher than the accuracy at previous step. Under some certain conditions the increase in order of accuracy after each iteration can be improved dramatically.

The waveform relaxation method is reviewed in Chapter 2 and two iterative approaches, **Waveform Gauss-Seidel** (WGS) and **Waveform Jacobi** (WJ) are discussed briefly. Several waveform relaxation methods based on different splittings are discussed. The waveform relaxation method is a generalized Picard method. Each approximate solution generated by the Picard method has one more correct term in its Taylor series than its previous one. In Chapter 3 we use some examples to show how the correct terms in the Taylor series of iterate generated by different waveform relaxation methods increase. Then we define the order of accuracy of an approximate solution and show that the order of accuracy of successive approximate solutions generated by the waveform relaxation method is increasing. In Chapter 4 we will show that the increase in the order of accuracy after each Waveform Gauss-Seidel iteration sweep is related to a system's partitioning and ordering. After a system is partitioned, the coupling relations among all subsystems can be indicated by a directed graph. Such a directed graph is called a system's dependency graph. We will prove that the average accuracy increase in waveform Gauss-Seidel is equal to the minimum ratio of C/d among all cycles in a system's dependency graph , where C is a cycle length and d is the number of times the numerical numbering of nodes in this cycle decreases. In Chapter 5 we will discuss some variant Gauss-Seidel approaches which can achieve better accuracy increase when being used to solve systems with special type of dependency graphs. In Chapter 6 we will discuss some implementation issues of the multirate integration method in waveform relaxation setting. We will present some numerical results from an experimental package for solving systems of differential equations by waveform Jacobi or waveform Gauss-Seidel. From the results we can see that the numerical results matches the theoretical results discussed in Chapter 4.

Chapter 2

Waveform Relaxation

Waveform methods were first proposed [7, 8] in the context of VLSI circuit simulation where they were used to solve differential-algebraic equations (DAEs). In this thesis we will examine their effectiveness for the solutions of ordinary differential equations (ODEs) which are special case of DAEs. The high cost of fabrication makes it important to verify the design of an integrated circuit by using simulation. One technique is to first construct a system of nonlinear ODEs that describe the given circuit, and then to solve the system with a numerical integration method. This is called *circuit simulation* [10].

The standard approach of solving ODE systems is based on three techniques [3, 10, 16]:

1. Using implicit integration methods to discretize the system of differential equations. (If the equations are stiff, stiffly stable methods must be used.)

2. Using a functional iteration or modified Newton method to solve the system of nonlinear algebraic equations obtained at each time point of the discretization. (If the equations are non stiff, only one functional iteration is needed, if stiff, an average of slightly more than one Newton iteration is needed.)

3. Using a direct method to solve the system of linear algebraic equations generated by Newton's method. (If the equations are non stiff, this last step is not necessary.)

As the size of ODE system grows, the standard approach can become inefficient. This is because the large systems usually contain variables that change at very different rates. The direct application of integration methods forces one to discretize all the variables identically

and the discretization must be fine enough to accurately reflect the behavior of the most rapidly changing variable. If each variable in the system could use the largest possible timestep that would accurately reflect its behavior, i.e. if we could use different stepsizes for different variables, then the efficiency of the simulation could be improved greatly. Approaches that allow different stepsizes for different components in solving systems of ordinary differential equations are called *Multirate Integration Methods* [3, 4, 14].

In contrast, waveform methods apply the iteration first to define, by a sequence of differential equations, a sequence of functions of time ("waveforms") which converge to the solution of the differential equations. The discretization of the resulting differential equations is done as a second step. Waveform methods can result in systems of ODEs which are mutually decoupled. This not only reduces communication requirements (in parallel processing) but also permits simple implementation of multirate integration.

Since the main computational bottleneck in solving stiff ordinary differential equations is the implicitness of the ODEs (numerical stability), it may be conceptually beneficial to apply iterative techniques in continuous time before discretization to handle implicitness [12]. *Waveform Relaxation* method is a class of continuous-time iterative methods, which was first used to speed up the simulation process of integrated circuit design [7] by allowing the individual variable of the systems to use different timesteps.

In this chapter we will review the basic idea of *Waveform Relaxation* method. Two popular approaches, **Waveform Gauss-Seidel** and **Waveform Jacobi**, will be presented in Section 2.1. General iteration formula of waveform relaxation will be given and several convergence properties will be discussed in Section 2.2.

2.1 Basic Idea

Waveform relaxation is a family of iterative methods that are applied to solve systems of ordinary differential equations. One of its basic ideas is to partition a big system into loosely coupled subsystems and to solve each subsystem independently over a part of the integration interval called a window. The coupling between subsystems is neglected in the sense that at each iteration sweep each subsystem is solved by using past values of other subsystems over

the window. The iterative process is continued until satisfactory convergence is obtained for each subsystem in a window. The same iterative process is performed in every window along the time axis until the entire integration interval has been considered. At each iteration sweep of waveform relaxation, every subsystem is discretized differently according to its behavior in the window. At the first iteration sweep, zero-th order (constant) extrapolations are usually used to approximate the values of variables in other subsystems; at later iteration sweeps, interpolations within the window are used to approximate the needed values. A *waveform* is a continuous representation of a solution component on a window.

Consider the following autonomous system of ordinary differential equations

$$\dot{\mathbf{u}} = F(\mathbf{u}), \qquad \mathbf{u}(0) = \mathbf{u}_0 \tag{2.1}$$

where $\mathbf{u} \in R^n$, and $F : R^n \to R^n$. Using waveform relaxation to solve (2.1), the system is first partitioned into m coupled subsystems

$$\begin{aligned} \dot{u}_1 &= f_1(u_1, u_2, \ldots, u_m), \qquad u_1(0) = u_{1,0} \\ &\vdots \\ \dot{u}_m &= f_m(u_1, u_2, \ldots, u_m), \qquad u_m(0) = u_{m,0} \end{aligned}$$

where $u_i \in R^{n_i}$, $\mathbf{u} = (u_1^T, u_2^T, \ldots, u_m^T)^T$, $f_i : R^n \to R^{n_i}$, $F = (f_1^T, f_2^T, \ldots, f_m^T)^T$, $1 \le i \le m$, and $\sum_{i=1}^m n_i = n$; then each subsystem

$$\dot{u}_i = f_i(u_1, \ldots, u_{i-1}, u_i, u_{i+1}, \ldots, u_m), \qquad u_i(0) = u_{i,0}$$

$1 \le i \le m$, is solved independently by using past values of $u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_m$.

For simplicity, in this section we only discuss systems which are partitioned into two subsystems, that is, the case $m = 2$.

$$\dot{u}_1 = f_1(u_1, u_2) \qquad u_1(0) = u_{1,0} \tag{2.2}$$

$$\dot{u}_2 = f_2(u_1, u_2) \qquad u_2(0) = u_{2,0} \tag{2.3}$$

The extension to arbitrary m is straightforward.

The idea of waveform relaxation will be illustrated using the two-time-scale system characterized by Eqns. (2.2) and (2.3). We assume $t \in [0, T]$, a finite interval.

Waveform Gauss-Seidel

In this approach, an initial guess to the solution of (2.2) or (2.3) is required to start the iterative process. Let us assume $u_2^{[0]}(t) \equiv u_{2,0}$, is the initial guess. We integrate Equation (2.2) with respect to u_1,

$$\dot{u}_1 = f_1(u_1, u_2^{[0]}), \qquad u_1(0) = u_{1,0}$$

to obtain the first approximation, $u_1^{[1]}$, to the solution. We then plug $u_1^{[1]}$ into Equation (2.3) and integrate (2.3) with respect to u_2,

$$\dot{u}_2 = f_2(u_1^{[1]}, u_2), \qquad u_2(0) = u_{2,0}.$$

We obtain $u_2^{[1]}$ as an approximation to the solution. Next $u_2^{[1]}$ is plugged back into Equation (2.2) and this equation is reintegrated to obtain $u_1^{[2]}$, which is a better approximation to the solution.

This iterative process can be written in the form

$$\left.\begin{aligned} \dot{u}_1^{[k]} &= f_1(u_1^{[k]}, u_2^{[k-1]}), & u_1^{[k]}(0) &= u_{1,0} \\ \dot{u}_2^{[k]} &= f_2(u_1^{[k]}, u_2^{[k]}), & u_2^{[k]}(0) &= u_{2,0} \end{aligned}\right\} \quad \text{for } k \geq 1,$$

$$u_2^{[0]}(t) \equiv u_{2,0}.$$

The process is terminated when the differences between the successive iterates are sufficiently small. To summarize, the waveform Gauss-Seidel method can be described by the following:

Algorithm 2.1: (Waveform Gauss-Seidel Algorithm for solving Eqn (2.1))

1. Partition system **u** into a number of subsystems, $u_1, \ldots, u_m$.
2. Set the iteration count, $k = 0$.
3. Guess initial waveforms for all variables, for example, $u_i^{[k]}(t) \equiv u_{i,0}\ \forall t \in [0, T]$.
4. **Repeat**

(a) Increase iteration count, $k = k + 1$.

(b) For each $(i \in \{1, \ldots, m\})$ { solve

$$\dot{u}_i^{[k]}(t) = f_i(u_1^{[k]}, \ldots, u_{i-1}^{[k]}, u_i^{[k]}, u_{i+1}^{[k-1]}, \ldots, u_m^{k-1]}) \qquad u_i^{[k]}(0) = u_{i,0}$$

for $(u_i^{[k]}(t);\ t \in [0, t])$ }

(c) **Until** $(\max_{1 \le i \le m} \max_{t \in [0,T]} \| u_i^{[k]}(t) - u_i^{[k-1]}(t) \| \le \epsilon$, a small positive number.)

Note that the differential equation in Algorithm (2.1) has only one unknown variable $u_i^{[k]}$. The variables $u_{i+1}^{[k-1]}, \ldots, u_m^{[k-1]}$ are known from the previous iteration and the variables $u_1^{[k]}, \ldots, u_{i-1}^{[k]}$ have already been computed.

Waveform Jacobi

In this approach, initial guesses $u_1^{[0]}$ and $u_2^{[0]}$ are both required. Assume $u_1^{[0]}(t) \equiv u_{1,0}$ and $u_2^{[0]}(t) \equiv u_{2,0}$ for $t \in [0, T]$. Then the following equations

$$\begin{aligned} \dot{u}_1 &= f_1(u_1, u_2^{[0]}), & u_1(0) &= u_{1,0} \\ \dot{u}_2 &= f_2(u_1^{[0]}, u_2); & u_2(0) &= u_{2,0} \end{aligned}$$

are integrated to produce better approximations $u_1^{[1]}$ and $u_2^{[1]}$. Then the initial guesses $u_1^{[0]}$ and $u_2^{[0]}$ are replaced by $u_1^{[1]}$ and $u_2^{[1]}$, and the process is repeated to obtain $u_1^{[2]}$ and $u_2^{[2]}$, the new approximations.

The iterative process can be written as follows:

$$\left.\begin{aligned} \dot{u}_1^{[k]} &= f_1(u_1^{[k]}, u_2^{[k-1]}), & u_1^{[k]}(0) &= u_{1,0} \\ \dot{u}_2^{[k]} &= f_2(u_1^{[k-1]}, u_2^{[k]}), & u_2^{[k]}(0) &= u_{2,0} \end{aligned}\right\} \quad \text{for } k \ge 1,$$

$$u_1^{[0]}(t) \equiv u_{1,0}, \qquad u_2^{[0]}(t) \equiv u_{2,0}.$$

The process is ended when both approximations converge. This method is suitable for implementation on multiprocessor computers since each subsystem can be handled by a different processor. The Waveform Jacobi method can be summarized by the following:

Algorithm 2.2: (Waveform Jacobi Algorithm for solving Eqn (2.1))

1. Partition system $\mathbf{u}$ into a number of subsystems, $u_1, \ldots u_m$.

2. Set the iteration count, $k = 0$.

3. Guess initial waveforms for all variables, for example, $u_i^{[k]}(t) \equiv u_{i,0}\ \forall t \in [0, T]$.

4. **Repeat**

   (a) Increase iteration count, $k = k + 1$.

   (b) For all $(i \in \{1, \ldots, m\})$ { solve

$$\dot{u}_i^{[k]}(t) = f_i(u_1^{[k-1]}, \ldots, u_{i-1}^{[k-1]}, u_i^{[k]}, u_{i+1}^{[k-1]}, \ldots, u_m^{k-1]}) \qquad u_i^{[k]}(0) = u_{i,0}$$

   for $(u_i^{[k]}(t);\ t \in [0,\ t])$ }

   (c) **Until** $(\max_{1 \le i \le m} \max_{t \in [0,T]} \| u_i^{[k]}(t) - u_i^{[k-1]}(t) \| \le \epsilon$, a small positive number.)

The obvious attraction of WJ method is that each subsystem can be integrated independently of the others in parallel. In the WGS method, the subsystems are integrated in sequence using the most recent values of the other subsystems. Both of these are particular examples of the general idea of *splitting* which will be discussed in next section.

2.2 General Iteration Formula

The idea of partitioning an ODE system can be generalized as splitting the right hand side of the ODE system in (2.1). Let $\mathcal{F}(\mathbf{u}, \mathbf{v})$ be chosen so that $\mathcal{F}(\mathbf{u}, \mathbf{u}) = F(\mathbf{u})$ and $\dot{\mathbf{u}} = \mathcal{F}(\mathbf{u}, \mathbf{v})$ is easy to solve for any given $\mathbf{v}$. Then the iteration formula is

$$\dot{\mathbf{u}}^{[k+1]} = \mathcal{F}(\mathbf{u}^{[k+1]}, \mathbf{u}^{[k]}), \qquad \mathbf{u}^{[k+1]}(0) = \mathbf{u}_0$$

with $\mathbf{u}^{[0]}(t) \equiv \mathbf{u}_0$. If we choose the splitting such that

$$\begin{aligned} \dot{\mathbf{u}} &= \mathcal{F}(\mathbf{u}, \mathbf{v}) \\ &= G(\mathbf{u}, \mathbf{v}) + F(\mathbf{v}) - G(\mathbf{v}, \mathbf{v}) \end{aligned}$$

or

$$\dot{\mathbf{u}} - G(\mathbf{u}, \mathbf{v}) = F(\mathbf{v}) - G(\mathbf{v}, \mathbf{v}), \tag{2.4}$$

the general iteration formula for waveform relaxation is then

$$\dot{\mathbf{u}}^{[k+1]} - G(\mathbf{u}^{[k+1]}, \mathbf{u}^{[k]}) = F(\mathbf{u}^{[k]}) - G(\mathbf{u}^{[k]}, \mathbf{u}^{[k]}), \quad \mathbf{u}^{[k+1]}(0) = \mathbf{u}_0 \tag{2.5}$$

where $\mathbf{u}^{[k]}$ is the k^{th} approximate solution generated by waveform relaxation. Suppose G is chosen so that the Jacobian matrix in Eqn (2.5) is block diagonal or block triangular, it is equivalent to the ODE system being decoupled into smaller subsystems and hence each subsystem can be solved independently or sequentially.

Different iterative methods can be derived from different splittings of the right hand side of the ODE system. Several iterations based on different splittings of the right hand side of Eqn (2.1) are listed below.

Example 1: If $G = (g_1, g_2, \ldots, g_m)^t$ and

$$g_i(\mathbf{u}^{[k+1]}, \mathbf{u}^{[k]}) = f_i(u_1^{[k]}, \ldots, u_{i-1}^{[k]}, u_i^{[k+1]}, u_{i+1}^{[k]}, \ldots, u_m^{[k]}), \qquad 1 \le i \le m$$

where $F = (f_1, f_2, \ldots, f_m)^t$, then, by the definition of G, $G(\mathbf{u}^{[k]}, \mathbf{u}^{[k]}) - F(\mathbf{u}^{[k]}) = 0$ in (2.5). So

$$\dot{\mathbf{u}}^{[k+1]} = G(\mathbf{u}^{[k+1]}, \mathbf{u}^{[k]}), \qquad \mathbf{u}^{[k+1]}(0) = \mathbf{u}_0,$$

is the iteration formula of waveform *Jacobi*. Under this splitting, the Jacobian matrix of the iteration formula is block diagonal, so each subsystem can be solved simultaneously. Hence waveform Jacobi is suitable for parallel computation.

□

Example 2: If $G = (g_1, g_2, \ldots, g_m)^t$ and

$$g_i(\mathbf{u}^{[k+1]}, \mathbf{u}^{[k]}) = f_i(u_1^{[k+1]}, \ldots, u_{i-1}^{[k+1]}, u_i^{[k+1]}, u_{i+1}^{[k]}, \ldots, u_m^{[k]}), \qquad 1 \le i \le m$$

where $F = (f_1, f_2, \ldots, f_m)^t$, then, by the definition of G, $G(\mathbf{u}^{[k]}, \mathbf{u}^{[k]}) - F(\mathbf{u}^{[k]}) = 0$ too in (2.5). So the iteration formula of waveform *Gauss-Seidel* is

$$\dot{\mathbf{u}}^{[k+1]} = G(\mathbf{u}^{[k+1]}, \mathbf{u}^{[k]}), \qquad \mathbf{u}^{[k+1]}(0) = \mathbf{u}_0.$$

Under this splitting, the system is partitioned into loosely coupled subsystems and the Jacobian matrix of the iteration formula is in lower block triangular form, so each subsystem will be solved in sequence.

□

Example 3: If G is chosen so that

$$G(\mathbf{u}, \mathbf{v}) = \frac{\partial F}{\partial \mathbf{u}}(\mathbf{v}) \cdot \mathbf{u},$$

where $\frac{\partial F}{\partial \mathbf{u}}(\mathbf{v})$ is the total derivative of F at $\mathbf{v}$ then we get the waveform *Newton* method:

$$\dot{\mathbf{u}}^{[k+1]} = F(\mathbf{u}^{[k]}) + \frac{\partial F}{\partial \mathbf{u}}(\mathbf{u}^{[k]}) \cdot (\mathbf{u}^{[k+1]} - \mathbf{u}^{[k]}) \qquad \mathbf{u}^{[k+1]}(0) = \mathbf{u}_0,$$

or

$$\dot{\mathbf{u}}^{[k+1]} - \frac{\partial F}{\partial \mathbf{u}}(\mathbf{u}^{[k]}) \cdot \mathbf{u}^{[k+1]} = F(\mathbf{u}^{[k]}) - \frac{\partial F}{\partial \mathbf{u}}(\mathbf{u}^{[k]}) \cdot \mathbf{u}^{[k]} \qquad \mathbf{u}^{[k+1]}(0) = \mathbf{u}_0.$$

In this iterative approach, the Jacobian matrix of the original system has to be computed at each iteration sweep, so if the system is very large it is impractical.

□

Example 4: If we choose the splitting $G(\mathbf{u}, \mathbf{v}) \equiv 0$, that is,

$$\dot{\mathbf{u}}^{[k+1]} = F(\mathbf{u}^{[k]}), \qquad \mathbf{u}^{[k+1]}(0) = \mathbf{u}_0$$

then we have the classical Picard method. As an iterative method, Picard is superlinearly convergent on any finite intervals [11].

□

We want to choose a splitting to accomplish several objectives: we want fast convergence, and for this, $G(\mathbf{u}, \mathbf{v})$ should, in some senses, be like $F(\mathbf{u})$; and we would also like the ODE (2.4) to be easy to integrate (by choosing a very simple function G). The Picard method yields the simplest integration: it is only a quadrature. However, its convergence is slow unless F is almost independent of $\mathbf{v}$. WJ and WGS require slightly more complex integrations, but they are simpler than the original problem because it has been reduced to a number of simpler subsystems. The important characteristic of these two splittings is that no communication

from other subsystems is needed during the integration of a single subsystem; it can happen prior to the integration. The goal of fast convergence is achieved by methods like waveform Newton in which we choose

$$G(\mathbf{u}, \mathbf{v}) = \frac{\partial F}{\partial \mathbf{u}}(\mathbf{v}) \cdot \mathbf{u}.$$

For this, $G(\mathbf{u}, \mathbf{v})$ "looks like" $F(\mathbf{u})$ in that their first derivatives are identical at $\mathbf{u} = \mathbf{v}$. Note that the error in successive iterates of a waveform method, $\epsilon^{[k]} = \mathbf{u}^{[k]} - \mathbf{u}$ satisfies

$$\dot{\epsilon}^{[k+1]} + G_{\mathbf{u}}\epsilon^{[k+1]} = (F_{\mathbf{v}} - G_{\mathbf{u}})\epsilon^{[k]} - O(\epsilon^{[k]} + \epsilon^{[k+1]})^2. \tag{2.6}$$

In waveform Newton, the first term on the right-hand side vanishes.

The fast convergence properties of waveform Newton are offset by the greater cost of each iteration. First there is the expensive computation of $\partial F/\partial \mathbf{v}$ at each step. Second, when the system of ODEs is very large and we try to integrate them on a parallel processor, there will be extensive communication between subsystems which destroys the potential advantages of parallel execution.

Finally, we summarize some convergence properties of waveform relaxation method. In order to guarantee that waveform relaxation applied to Eqn (2.1) will converge to the systems' solution, we first must guarantee that Eqn (2.1) has a solution. If we require that F is Lipschitz continuous with respect to $\mathbf{u}$, then a unique solution for the system exists [3]. In [16] it is shown that the waveform relaxation algorithm is a contraction mapping in an exponentially scaled norm and in [11] waveform relaxation is proved to converge superlinearly on any finite intervals. In the following chapter we will look at waveform relaxation from a different point of view. Instead of discussing the convergence property of waveform relaxation, we will discuss how the order of accuracy of successive approximate solutions increases.

Chapter 3

Accuracy Increase in Waveform Relaxation

In this chapter, we will look at waveform relaxation from a different point of view. Instead of discussing the convergence property of waveform relaxation, we will discuss how the order of accuracy of successive approximate solutions increases. We use Taylor expansions to demonstrate that the waveform relaxation method is Picard-like. That is, the waveform relaxation method can be considered as a generalization of the classical Picard method in the sense that the Taylor series expansions of successive approximate solutions generated by both methods have more and more terms coinciding with the Taylor series expansion of the exact solution. So when many iterations are performed the iterative solution will be a good approximant of the exact solution. In the Picard method each successive approximate solution gains exactly one more term in its Taylor series expansion, while in the waveform relaxation method the gains can be more than one. We will assume continuity of as many derivatives as necessary for our analysis.

In Section 1 we study some examples to see why waveform relaxation method can be considered as a generalized Picard method. From the examples given , we can see that the Taylor series expansion at each iteration matches a certain degree Taylor polynomial of the exact solution. As the iteration continues, the degree of the matching Taylor polynomial gets higher. This phenomenon motivates us to discuss the idea of *order of accuracy*.

In Section 2 we define the *order of accuracy* of an approximate solution and use it to prove Theorem 3.2 and Theorem 3.6, and a corollary that tell us how the order of accuracy increases

after each iteration sweep.

3.1 Taylor Series and Waveform Relaxation

Different splittings yield different waveform methods with different convergence properties. All waveform relaxation methods converge superlinearly on any finite interval so it is not possible to use a measure like rate of convergence to compare different splittings. In this section we consider, instead, the rate of increase in the order of successive approximations. In order to understand the motivation for discussing the *order of accuracy* of approximate solutions generated by iterative methods, in this section we will use some instructive examples to demonstrate how the Taylor expansions of the exact solution and of the approximate solutions generated by waveform relaxation method are related.

Consider Picard applied to the simple problem $y' = y$, $y(0) = 1$ starting from the approximation $y^{[0]}(t) = 1$. The k-th iterate is $y^{[k]}(t) = 1+t+t^2/2!+\ldots+t^k/k!$. Each successive iterate has one additional correct term in its power series. This is not peculiar to simple problems.

A Riccati equation, which will be solved by Picard and two forms of waveform relaxation is discussed below. (All the coefficients of the Taylor series and of the error terms in this section were computed with *Mathematica* [17].)

Consider the following Riccati equation:

$$\dot{u} = u - 2u^2, \quad u(0) = 3. \tag{3.1}$$

The exact solution to this equation is

$$u = \frac{1}{2 - \frac{5}{3}e^{-t}}.$$

It's Taylor series expansion up to the 32^{nd} power of t is listed below.

$$u(t) = 3 - 15t + \frac{165t^2}{2} - \frac{905t^3}{2} + \frac{19855t^4}{8} - \frac{108901t^5}{8} +$$

$$\frac{3583811t^6}{48} - \frac{137595781t^7}{336} + \frac{6037499171t^8}{2688} - \frac{298031091301t^9}{24192} +$$

$$\frac{16346453844611t^{10}}{241920} - \frac{986230018285381t^{11}}{2661120} + \frac{843006100707823t^{12}}{414720} -$$

$$\frac{4628356194666449701t^{13}}{415134720} + \frac{3230904376000550340 1t^{14}}{528353280} -$$

$$\frac{2923950581677828127898 1t^{15}}{87178291200} + \frac{2332702233725136390939 61t^{16}}{126804787200} -$$

$$\frac{1840430996148771309407877 7t^{17}}{1824038092800} + \frac{3067655953877759572869057 43t^{18}}{5543180697600} -$$

$$\frac{2461574341203724092547697936581t^{19}}{8109673360588800} + \frac{2454778134840881779927227640 1161t^{20}}{14744860655616000} -$$

$$\frac{3110184879504308408310255626625 4501t^{21}}{3406062811447296000} +$$

$$\frac{1795661949196018181937171693380257 9t^{22}}{358532927520768000} -$$

$$\frac{4734353491474502792085744422799540 18181t^{23}}{1723467782592331776000} +$$

$$\frac{6225866804705194943113785471528978897 1t^{24}}{41321904877338624000} -$$

$$\frac{8545468760110628823706819494174008019 180901t^{25}}{1034080669555399065600000} +$$

$$\frac{1107843804149974478203725729441121679 74909801t^{26}}{2444190673494579609600000} -$$

$$\frac{1804666522260917331728118944746989711 58885683781t^{27}}{7259246300278901440512000 00} +$$

$$\frac{2519557578871559600479694682858617505 769096999561t^{28}}{1847808149161902184857600000} -$$

$$\frac{3391045350328483652720253268779103672665296389571 77t^{29}}{45342369198665138228428800000} +$$

$$\frac{4958106581085616864210317612670118760817327647909 97t^{30}}{12087166088502668427264000000} -$$

$$\frac{1233342955852996538365486646783198484439286665838554933 81t^{31}}{54818924361186152118170419200000} +$$

$$\frac{1157588514217938630924161505047050110170488920874895606 33t^{32}}{93807785003099297742323712000000} -$$

$$O(t)^{33}$$

3.1.1 Picard Method

The traditional forms of Picard method are written as either

$$\mathbf{u}^{[k+1]}(t) = \mathbf{u}_0 + \int_0^t F(\mathbf{u}^{[k]}(\tau))d\tau$$

or

$$\dot{\mathbf{u}}^{[k+1]} = F(\mathbf{u}^{[k]}) \quad \mathbf{u}(0) = \mathbf{u}_0.$$

From the second equation we see that Picard is actually a special case of waveform relaxation by choosing

$$G(\mathbf{u}^{[k+1]}, \mathbf{u}^{[k]}) = G(\mathbf{u}^{[k]}, \mathbf{u}^{[k]}) = 0,$$

i.e. no splitting is used. To see how the approximate solutions, in their Taylor series expansions, generated by Picard are related to the exact solution, we now use Picard to solve the Riccati equation given in (3.1). The iteration scheme used is as follow:

$$\dot{u}^{[k+1]} = u^{[k]} - 2(u^{[k]})^2, \quad u^{[k+1]}(0) = 3.$$

The first five approximate solutions generated are listed below, where $u(t)$ is the exact solution:

$$u^{[1]}(t) \quad = \quad u(t) +$$

$$\frac{-165t^2}{2} + \frac{905t^3}{2} - \frac{19855t^4}{8} + \frac{108901t^5}{8} - \frac{3583811t^6}{48} +$$

$$\frac{137595781t^7}{336} - \frac{6037499171t^8}{2688} + \frac{298031091301t^9}{24192} - \frac{16346453844611t^{10}}{241920} +$$

$$O(t)^{11}$$

$$u^{[2]}(t) = u(t) +$$

$$\frac{605t^3}{2} - \frac{19855t^4}{8} + \frac{108901t^5}{8} - \frac{3583811t^6}{48} + \frac{137595781t^7}{336} -$$

$$\frac{6037499171t^8}{2688} + \frac{298031091301t^9}{24192} - \frac{16346453844611t^{10}}{241920} +$$

$$O(t)^{11}$$

$$u^{[3]}(t) = u(t) +$$

$$\frac{-6655t^4}{8} + \frac{72721t^5}{8} - \frac{3187811t^6}{48} + \frac{135435781t^7}{336} -$$

$$\frac{6037499171t^8}{2688} + \frac{298031091301t^9}{24192} - \frac{16346453844611t^{10}}{241920} +$$

$$O(t)^{11}$$

$$u^{[4]}(t) = u(t) +$$

$$\frac{14641t^5}{8} - \frac{1199231t^6}{48} + \frac{74422381t^7}{336} - \frac{4342500371t^8}{2688} +$$

$$\frac{253036822501t^9}{24192} - \frac{152117902926 11t^{10}}{241920} +$$

$$O(t)^{11}$$

$$u^{[5]}(t) = u(t) +$$

$$\frac{-161051t^6}{48} + \frac{18462301t^7}{336} - \frac{1525247471t^8}{2688} + \frac{114556106401t^9}{24192} -$$

$$\frac{84301514550 11t^{10}}{241920} + O(t)^{11}$$

From this example it is easy to see that after each Picard iteration exactly one additional correct term is picked up by the new approximation. And as the iteration continues, the number of correct terms in an approximation gets larger.

3.1.2 Waveform Relaxation

In the following we will see that same behavior occurs in waveform relaxation method. After each iteration one or more than one additional correct terms will be picked up, and the number of additional correct terms to be picked up after each iteration is related to the iteration scheme used.

First we use the following scheme

$$\dot{u}^{[k+1]} - u^{[k+1]} = -2(u^{[k]})^2, \quad u^{[k+1]}(0) = 3.$$

to solve the Riccati equation given in Equation (3.1). The splitting is $G(u^{[k+1]}, u^{[k]}) = u^{[k+1]}$. The first five approximate solutions with their error terms up to t^{11} are listed below.

$$u^{[1]}(t) = u(t) +$$

$$-90t^2 + 450t^3 - \frac{4965t^4}{2} + \frac{27225t^5}{2} - \frac{298651t^6}{4} + \frac{1638045t^7}{4} -$$

$$\frac{503124931t^8}{224} + \frac{8278641425t^9}{672} - \frac{1362204487051t^{10}}{20160} +$$

$$O(t)^{11}$$

$$u^{[2]}(t) = u(t) +$$

$$360t^3 - 2610t^4 + 13536t^5 - 74694t^6 + \frac{2866509t^7}{7} - \frac{125781381t^8}{56} +$$

$$\frac{1034830127t^9}{84} - \frac{28379260199t^{10}}{420} +$$

$$O(t)^{11}$$

$$u^{[3]}(t) = u(t) +$$

$$-1080t^4 + 10368t^5 - 71244t^6 + 412992t^7 - \frac{31481001t^8}{14} + \frac{86193864t^9}{7} -$$

$$\frac{9460505191t^{10}}{140} + O(t)^{11}$$

$$u^{[4]}(t) = u(t) +$$

$$2592t^5 - 31104t^6 + 257472t^7 - 1793664t^8 + \frac{78271722t^9}{7} - \frac{2287886688t^{10}}{35} +$$

$$O(t)^{11}$$

$$u^{[5]}(t) = u(t) +$$

$$-5184t^6 + \frac{523584t^7}{7} - \frac{5019408t^8}{7} + \frac{39830832t^9}{7} - \frac{201552948t^{10}}{5} +$$

$$O(t)^{11}$$

From above listing we can see that again exactly one additional correct term is picked up after each iteration and the leading error coefficient is different from that of the Picard method.

Next we use a different waveform relaxation scheme to solve the same Riccati equation by using the splitting, $G(u^{[k+1]}, u^{[k]}) = (1 - 4u^{[k]})u^{[k+1]}$. The scheme used is now

$$\dot{u}^{[k+1]} - (1 - 4u^{[k]})u^{[k+1]} = 2(u^{[k]})^2, \quad u^{[k+1]}(0) = 3.$$

In the following we list the first four approximate solutions with a few error terms.

$$u^{[1]}(t) = u(t) +$$

$$150t^3 - 1650t^4 + \frac{23565t^5}{2} - \frac{142615t^6}{2} + \frac{1616955t^7}{4} - \frac{8955375t^8}{4} +$$

$$\frac{24818061035t^9}{2016} - \frac{136200799141t^{10}}{2016} +$$

$$O(t)^{11}$$

$$u^{[2]}(t) = u(t) +$$

$$\frac{45000t^7}{7} - \frac{928125t^8}{7} + \frac{11167875t^9}{7} - \frac{207445425t^{10}}{14} +$$

$$\frac{1301077875t^{11}}{11} - \frac{5991092225t^{12}}{7} + \frac{9926905947025t^{13}}{1716} -$$

$$\frac{189613674132775t^{14}}{5096} + O(t)^{15}$$

$$u^{[3]}(t) = u(t) +$$

$$\frac{270000000t^{15}}{49} - \frac{10627031250t^{16}}{49} + \frac{3866146875000t^{17}}{833} -$$

$$\frac{59839216593750t^{18}}{833} + \frac{157062483419343750t^{19}}{174097} - \frac{1547180515307025 00t^{20}}{15827} +$$

$$O(t)^{21}$$

$$u^{[4]}(t) = u(t) +$$

$$\frac{1458000000000000000t^{31}}{74431} - \frac{1116865019531250000 0t^{32}}{74431} +$$

$$O(t)^{33}$$

In this scheme we see that more than one correct term is picked up after each iteration. Actually, the number of correct terms in the Taylor expansion of an approximation almost doubles after each iteration.

From these examples we see that the degree of the leading error term at each approximate solution generated by waveform relaxation method increases as the iteration continues. In next section we will show that the behavior we observe in these examples is that normally expected.

3.2 Order of Accuracy and Accuracy Increase

In this section we begin with the definition of the *order of accuracy.* Then we show that the increase in the order of accuracy after each iteration sweep will be at least one for different iterative schemes. By using the Fréchet derivative we also show that the increase can be in geometrical progression if the splitting is chosen carefully.

From the examples given in the previous section we see that exactly one additional correct term is picked up in each iteration. The reason is evident from a consideration of the error term, $\epsilon^{[k]}(t) = \mathbf{u}^{[k]}(t) - \mathbf{u}(t)$, which satisfies (2.6). If the partial derivatives in that equation are evaluated at suitable points near the solution, the higher order terms can be ignored in that

equation, so we find that

$$\epsilon^{[k+1]} = \int_0^t \hat{G}(\tau)\epsilon^{[k]}(\tau)d\tau$$

where $\hat{G}$ is the Greens function for the left hand side of (2.6). Clearly, if $\epsilon^{[k]}(t)$ is a power series starting with t^{k+1} then $\epsilon^{[k+1]}(t)$ will be a power series starting with t^{k+2}. This leads us to define the order of accuracy of an approximation as follows. Let $u_i(t)$ be the i^{th} component of the exact solution and $z_i(t)$ be the i^{th} component of an approximate solution to Equation (2.2).

Definition 3.1 *If $z_i(t) - u_i(t) = O(t)^{M_i+1}$ over a fixed, finite interval $[0, T]$, then the order of accuracy, $N(z_i)$, of $z_i(t)$ is M_i for $1 \leq i \leq n$. The order of accuracy of $\mathbf{z}(t)$, denoted by $N(\mathbf{z})$, is defined as $\min_{1\leq i\leq n} N(z_i)$*

The first basic theorem of accuracy increase in waveform relaxation is stated and proved below.

Theorem 3.2 *Suppose that the exact solution of (2.1) can be written as*

$$\mathbf{u}(t) = \sum_{i=0}^{M} \mathbf{a}_i t^i + O(t)^{M+1} \tag{3.2}$$

that is, $\mathbf{u}(t) \in C^M$. Given $\mathbf{z}(t)$, an approximate solution, which satisfies $\mathbf{z}(0) = \mathbf{u}_0$, define

$$\mathbf{R}(t) \doteq \mathbf{z}(t) - \mathbf{u}(t) = \sum_{i=N(\mathbf{z})+1}^{M} \mathbf{b}_i t^i + O(t)^{M+1}. \tag{3.3}$$

Let $\mathbf{y}(t)$ be the solution to the following system

$$\dot{\mathbf{y}} - G(\mathbf{y}, \mathbf{z}) = F(\mathbf{z}) - G(\mathbf{z}, \mathbf{z}), \tag{3.4}$$

where $G : R^{2n} \to R^n$, F and G are sufficiently smooth and define

$$\mathbf{E}(t) \doteq \mathbf{y}(t) - \mathbf{u}(t) = \sum_{i=N(\mathbf{y})+1}^{M} \mathbf{c}_i t^i + O(t)^{M+1}. \tag{3.5}$$

Then, if $F(\mathbf{z})$ and $G(\mathbf{y}, \mathbf{z})$ are sufficiently differentiable,

$$N(\mathbf{y}) \geq N(\mathbf{z}) + 1.$$

Proof: In Equation (3.3), it is easy to see that $\frac{d^k\mathbf{R}}{dt^k}(0) \equiv \mathbf{R}^{(k)}(0) = 0$ for $k = 1,\ 2, \ldots,$ $N(\mathbf{z})$. To prove the inequality $N(\mathbf{y}) \geq N(\mathbf{z}) + 1$, it suffices to show that $\mathbf{E}^{(k)}(0) = 0$ for $k = 1,\ 2, \ldots, N(\mathbf{z}) + 1$.

Consider the system

$$\dot{\mathbf{u}}(t) = F(\mathbf{u}(t)), \quad \mathbf{u}(0) = \mathbf{u}_0.$$

First subtract $G(\mathbf{u}(t), \mathbf{z}(t))$ from both sides of this system, we have

$$\dot{\mathbf{u}}(t) - G(\mathbf{u}(t), \mathbf{z}(t)) = F(\mathbf{u}(t)) - G(\mathbf{u}(t), \mathbf{z}(t)).$$

Next subtract this equation from (3.4), we get

$$\begin{aligned} (\dot{\mathbf{y}}(t) \ &- \ \dot{\mathbf{u}}(t)) - (G(\mathbf{y}(t), \mathbf{z}(t)) - G(\mathbf{u}(t), \mathbf{z}(t))) \\ &= \ (F(\mathbf{z}(t)) - G(\mathbf{z}(t), \mathbf{z}(t))) - (F(\mathbf{u}(t)) - G(\mathbf{u}(t), \mathbf{z}(t))). \end{aligned} \tag{3.6}$$

Before we continue the proof, for convenience we introduce the following notation

$$L_{\mathbf{w}}[p_1^*, p_2^*, \ldots, p_n^*] = \begin{bmatrix} \frac{\partial w_1}{\partial x_1}(p_1^*) & \frac{\partial w_1}{\partial x_2}(p_1^*) & \cdots & \frac{\partial w_1}{\partial x_n}(p_1^*) \\ \frac{\partial w_2}{\partial x_1}(p_2^*) & \frac{\partial w_2}{\partial x_2}(p_2^*) & \cdots & \frac{\partial w_2}{\partial x_n}(p_2^*) \\ \vdots & \vdots & & \vdots \\ \frac{\partial w_n}{\partial x_1}(p_n^*) & \frac{\partial w_n}{\partial x_2}(p_n^*) & \cdots & \frac{\partial w_n}{\partial x_n}(p_n^*) \end{bmatrix}$$

where

$$\mathbf{w}: \begin{cases} w_1 &= w_1(x_1, x_2, \ldots, x_n) \\ w_2 &= w_2(x_1, x_2, \ldots, x_n) \\ & \vdots \\ w_n &= w_n(x_1, x_2, \ldots, x_n) \end{cases}$$

is defined for all points $p = (x_1, \ldots, x_n)$ in an open set D and $p_1^*, \ldots, p_n^* \in D$.

Let

$$g_{\mathbf{z}(t)}(\mathbf{v}(t)) = G(\mathbf{v}(t), \mathbf{z}(t)) \tag{3.7}$$

and

$$Q_{\mathbf{z}(t)}(\mathbf{v}(t)) = F(\mathbf{v}(t)) - G(\mathbf{v}(t), \mathbf{z}(t)), \tag{3.8}$$

then apply Mean-Value Theorem, we have

$$g_{\mathbf{z}(t)}(\mathbf{y}(t)) - g_{\mathbf{z}(t)}(\mathbf{u}(t)) = (L_{g_{\mathbf{z}(t)}}[p_1^*, \ldots, p_n^*])(\mathbf{y}(t) - \mathbf{u}(t))$$

and

$$Q_{\mathbf{z}(t)}(\mathbf{z}(t)) - Q_{\mathbf{z}(t)}(\mathbf{u}(t)) = (L_{Q_{\mathbf{z}(t)}}[q_1^*, \ldots, q_n^*])(\mathbf{z}(t) - \mathbf{u}(t))$$

where $p_1^*, \ldots, p_n^*$ are points lying in the line segment which joins $\mathbf{y}(t)$ and $\mathbf{u}(t)$ and $q_1^*, \ldots, q_n^*$ are points lying in the line segment which joins $\mathbf{z}(t)$ and $\mathbf{u}(t)$. Note that $p_i^* = \alpha_i\, \mathbf{y}(t) + (1-\alpha_i)\, \mathbf{u}(t)$ and $q_i^* = \beta_i\, \mathbf{z}(t) + (1-\beta_i)\, \mathbf{u}(t)$ where $0 \le \alpha_i \le 1$, $0 \le \beta_i \le 1$ for $i = 1, \ldots, n$. Because of the smoothness of $\mathbf{u}(t)$, $\mathbf{z}(t)$ and $\mathbf{y}(t)$, $p_1^*, \ldots, p_n^*$ and $q_1^*, \ldots, q_n^*$ are sufficiently smooth as well.

Replacing $\mathbf{z}(t) - \mathbf{u}(t)$ by $\mathbf{R}(t)$ and $\mathbf{y}(t) - \mathbf{u}(t)$ by $\mathbf{E}(t)$ respectively in (3.6), this equation becomes

$$\dot{\mathbf{E}}(t) = L_{g_{\mathbf{z}(t)}}(t)\, \mathbf{E}(t) + L_{Q_{\mathbf{z}(t)}}(t)\, \mathbf{R}(t). \tag{3.9}$$

Differentiate Equation (3.9) k times with respect to t, we obtain the following general form

$$\mathbf{E}^{(k+1)}(t) = \sum_{j=0}^{k} \binom{k}{j} \frac{d^j}{dt^j} L_{g_{\mathbf{z}(t)}}(t) \mathbf{E}^{(k-j)}(t) + \sum_{j=0}^{k} \binom{k}{j} \frac{d^j}{dt^j} L_{Q_{\mathbf{z}(t)}}(t) \mathbf{R}^{(k-j)}(t) \tag{3.10}$$

Since $\mathbf{R}^{(k)}(0) = 0$ for $k = 0, 1, \ldots, N(z)$ and $\mathbf{E}(0) = 0$, by induction we can see that $\mathbf{E}^{(k)}(0) = 0$ for $k = 0, 1, \ldots, N(z)+1$. Thus the proof is complete. Note that $\mathbf{E}^{(N(z)+2)}(0)$ may not be zero since $\mathbf{R}^{(N(z)+1)}(0) \neq 0$.

Q. E. D.

This theorem tells us that the accuracy of current iteration is at least one order higher than that of the previous iteration. Here we emphasize the word "accuracy". In general, the increase of order of accuracy does not imply the convergence of the iterative scheme. But from this theorem we can see that as the iteration continues more and more terms of the Taylor expansion of each iterate coincide with the Taylor expansion of the exact solution.

Corollary 3.3 *Use the same notations as in Theorem 3.2. Let* $F(\mathbf{u}) = A\,\mathbf{u}$ *and* $G(\mathbf{w}, \mathbf{z}) = B\,\mathbf{w}$, *where both* A *and* B *are constant matrices. If* $\mathbf{R}^{(N(\mathbf{z}))}(0)$ *is not in the null space of* $A - B$, *then* $N(\mathbf{y}) = N(\mathbf{z}) + 1$

Proof: Equation (3.10) becomes

$$\mathbf{E}^{(k+1)}(t) = \sum_{j=0}^{k} \begin{pmatrix} k \\ j \end{pmatrix} \frac{d^j}{dt^j} B\, \mathbf{E}^{(k-j)}(t) + \sum_{j=0}^{k} \begin{pmatrix} k \\ j \end{pmatrix} \frac{d^j}{dt^j} (A - B)\, \mathbf{R}^{(k-j)}(t). \tag{3.11}$$

Since both A and B are constant matrices, the equation (3.11) is reduced to

$$\mathbf{E}^{(k+1)}(t) = B\, \mathbf{E}^{(k)}(t) + (A - B)\, \mathbf{R}^{(k)}(t). \tag{3.12}$$

Thus $\mathbf{E}^{(N(\mathbf{z})+2)}(0) = (A - B)\, \mathbf{R}^{(N(\mathbf{z})+1)}(0) \neq 0$, and the proof is complete.

Q. E. D.

Theorem 3.2 gives an inequality. Normally this is an equality unless there is cancellation, sparsity, or a special nature of the problem or splitting as described in Corollary 3.3. Cancellation is exploited in the waveform Newton method, while sparsity will be exploited in the WGS method which will be discussed in next chapter. Naturally we would like to know the exact accuracy increase for each iterative scheme. But this is a very hard problem to answer, even for the Picard method, the simplest WR method. We would expect the accuracy increase after one Picard iteration to be exactly one, but it is not always true. A problem with a special character is $\dot{y} = t^{n-1}y$. It increases in accuracy by n at each iteration starting from $t_0 = 0$, although that accuracy increase does not occur from other starting points.

Before we state and prove Theorem 3.6, we review the Fréchet derivative and the Taylor's Theorem for vector functions [2] which are needed for proving the aforementioned theorem.

Consider $\mathbf{w} : X \to Y$, where X and Y are normed linear spaces. Given $x \in X$, if a linear operator $d\mathbf{w}(x)$ exists which is continuous such that

$$\lim_{\|h\| \to 0} \frac{\| \mathbf{w}(x+h) - \mathbf{w}(x) - d\mathbf{w}(x)h \|_Y}{\| h \|_X} = 0$$

then $\mathbf{w}$ is said to be *Fréchet differentiable* at x, and $d\mathbf{w}(x)$ is said to be the Fréchet differential of $\mathbf{w}$ at x with increment h. $d\mathbf{w}(x) \in \mathcal{L}(X, Y)$, the space of all bounded linear operators from X to Y. If both X and Y are Banach spaces and the Fréchet derivative of $\mathbf{w}$ is a continuous linear operator, then $\mathcal{L}(X, Y)$ is again a Banach space, and hence we may consider the Fréchet differential of $d\mathbf{w}(\cdot) : X \to \mathcal{L}(X, Y)$.

If this differential exists we will denote it by $d^2\mathbf{w}(x)$ and it is clear that $d^2\mathbf{w}(x) \in \mathcal{L}(X, \mathcal{L}(X,Y))$. However it can be shown that $\mathcal{L}(X, \mathcal{L}(X,Y))$ is isometrically isomorphic to $\mathcal{L}(X \times X, Y)$ and $d^2\mathbf{w}(\cdot) : X \to \mathcal{L}(X \times X, Y)$. We may obviously continue this process so that

$$d^n\mathbf{w}(.) : X \to \mathcal{L}(X \times X \times \ldots \times X, Y).$$

Note that $d^n\mathbf{w}(x)$ is a bounded symmetric multilinear operator.

Theorem 3.4 *(***Taylor's Theorem***) Let E be an open subset in a Banach space X, and Y another Banach space. Then if $\mathbf{w} : E \to Y$ is n times differentiable at a point $x \in E$,*

$$\begin{aligned} \mathbf{w}(x+h) &= \mathbf{w}(x) + d\mathbf{w}(x)h + \frac{1}{2!}d^2\mathbf{w}(x)h^2 + \cdots + \frac{1}{n!}d^n\mathbf{w}(x)h^n + \\ & \quad r(x,h) \end{aligned} \tag{3.13}$$

where

$$\lim_{\|h\| \to 0} \frac{\| r(x,h) \|_Y}{\| h \|_X^n} = 0.$$

The terms $d^n\mathbf{w}(x)h^n$ need some explanation. $d^n\mathbf{w}(x)h^n$ is a map from $X \times X \times \ldots \times X$ (n times) into Y, so that its evaluation at the point $(h_1, h_2, \ldots, h_n)$ is

$$d^n\mathbf{w}(x)(h_1, h_2, \ldots, h_n).$$

The symbol $d^n\mathbf{w}(x)h^n$ is used to represent the above expression when $h_1 = h_2 = \ldots = h_n = h$.

Definition 3.5 *Use the same notation as in Theorem 3.4, we say that*

$$\mathbf{w}(x+h) - \mathbf{w}(x) = O(h)^K$$

if $d\mathbf{w}^k(x) = 0$ for $k = 1, 2, \ldots, K-1$.

Let us revisit the Riccati example with the waveform Newton method. We use the splitting $G(u^{[k+1]}, u^{[k]}) = (1 - 4u^{[k]})u^{[k+1]}$. The iteration is

$$\dot{u}^{[k+1]} - (1 - 4u^{[k]})u^{[k+1]} = 2(u^{[k]})^2, \qquad u^{[k+1]}(0) = 3.$$

The first three approximate solutions are

$$\begin{aligned} u^{[1]}(t) &= u(t) + 150t^3 + \mathrm{O}(t)^4 \\ u^{[2]}(t) &= u(t) + \frac{45000t^7}{7} + \mathrm{O}(t)^8 \\ u^{[3]}(t) &= u(t) + \frac{270000000t^{15}}{49} + \mathrm{O}(t)^{16} \end{aligned}$$

In this scheme we see that more than one correct term is picked up after each iteration. Actually, the number of correct terms in the Taylor expansion of an approximation almost doubles after each iteration because of the quadratic convergence of the Newton iteration.

This behavior is a particular case of Theorem 3.6 below which relates the order increase to the "closeness" of $G(\mathbf{v}, \mathbf{u})$ to $F(\mathbf{v})$. We define

$$Q_{\mathbf{u}}(\mathbf{v}) = G(\mathbf{v}, \mathbf{u}) - F(\mathbf{v}).$$

If Q is zero, then the "splitting" leads to the solution in one iteration. If $Q_{\mathbf{u}}(\mathbf{v})$ is relatively insensitive to changes in $\mathbf{v}$ near $\mathbf{v} = \mathbf{u}$, we get rapid convergence. If we add one more assumption to Theorem 3.2, we can obtain an iterative scheme which has *geometric accuracy increase.*

Theorem 3.6 *Same assumptions as given in Theorem 3.2, In addition, we assume that*

$$Q_{\mathbf{z}(t)}(\mathbf{z}(t)) - Q_{\mathbf{z}(t)}(\mathbf{u}(t)) = O(\mathbf{z} - \mathbf{u})^K.$$

Then

$$N(\mathbf{y}) \geq K\,(N(\mathbf{z}) + 1).$$

In particular, if $K = 2$,that is, $dF(\mathbf{u}(t)) = dG(\mathbf{v}(t), \mathbf{z}(t))$ w.r.t. $\mathbf{v}$ at $\mathbf{v} = \mathbf{u}$, this iteration scheme is the Waveform Newton method and it converges quadratically, i.e., we get orders of accuracy 0, 2, 6, 14, 30, ..., when starting with a constant zeroth iterate.

Before we prove this theorem, we discuss one possible method to solve systems of the form

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + \mathbf{f}(t), \quad \mathbf{x}(a) = \mathbf{c} \tag{3.14}$$

where we assume that all the entries of $A(t)$ and $\mathbf{f}(t)$ are smooth functions of t. We begin with the following definition.

Definition 3.7 *The transition matrix $\Phi(t,a)$ of the system (3.14) is the solution of the system*

$$\frac{d}{dt}\Phi(t,a) = A(t)\Phi(t,a), \quad \Phi(a,a) = I_{n\times n}.$$

Notes that $\Phi(t,a)$ is an $n \times n$ matrix.

The relationship between the transition matrix and the solution of the system (3.14) is given by the well known theorem:

Theorem 3.8 *The solution of the system (3.14) is given by*

$$\begin{aligned} \mathbf{x}(t) &= \Phi(t,a)\mathbf{c} + \int_a^t \Phi(t,s)\mathbf{f}(s)ds \\ &= \Phi(t,a)\mathbf{c} + \Phi(t,a)\int_a^t \Phi(a,s)\mathbf{f}(s)ds. \end{aligned} \tag{3.15}$$

Now we are ready to prove Theorem 3.6.

Proof: First we recall Equation (3.6)

$$\begin{aligned} (\dot{\mathbf{y}}(t) &- \dot{\mathbf{u}}(t)) - (G(\mathbf{y}(t),\mathbf{z}(t)) - G(\mathbf{u}(t),\mathbf{z}(t))) \\ &= (F(\mathbf{z}(t)) - G(\mathbf{z}(t),\mathbf{z}(t))) - (F(\mathbf{u}(t)) - G(\mathbf{u}(t),\mathbf{z}(t))). \end{aligned}$$

Let

$$g_{\mathbf{z}(t)}(\mathbf{v}(t)) = G(\mathbf{v}(t),\mathbf{z}(t)), \tag{3.16}$$

and

$$Q_{\mathbf{z}(t)}(\mathbf{v}(t)) = F(\mathbf{v}(t)) - G(\mathbf{v}(t),\mathbf{z}(t)) \tag{3.17}$$

then apply Mean-Value Theorem to Equation (3.16), we have

$$g_{\mathbf{z}(t)}(\mathbf{y}(t)) - g_{\mathbf{z}(t)}(\mathbf{u}(t)) = (L_{g_{\mathbf{z}(t)}}[p_1^*,\ldots,p_n^*])(\mathbf{y}(t) - \mathbf{u}(t)).$$

Equation (3.6) becomes

$$\begin{aligned} (\dot{\mathbf{y}}(t) - \dot{\mathbf{u}}(t)) &= L_{g_{\mathbf{z}(t)}}[p_1^*,\ldots,p_n^*]\,(\mathbf{y}(t) - \mathbf{u}(t)) + \\ &\quad (Q_{\mathbf{z}(t)}(\mathbf{z}(t)) - Q_{\mathbf{z}(t)}(\mathbf{u}(t))). \end{aligned} \tag{3.18}$$

We now prove the following statement: If

$$\mathbf{z} - \mathbf{u} = O(t)^{N_{\mathbf{z}}}$$

where $N_{\mathbf{z}} = N(\mathbf{z}) + 1$ and

$$Q_{\mathbf{z}(t)}(\mathbf{z}(t)) - Q_{\mathbf{z}(t)}(\mathbf{u}(t)) = O(\mathbf{z} - \mathbf{u})^K,$$

then

$$Q_{\mathbf{z}(t)}(\mathbf{z}(t)) - Q_{\mathbf{z}(t)}(\mathbf{u}(t)) = O(t)^{KN_{\mathbf{z}}}.$$

By definition,

$$Q_{\mathbf{z}(t)}(\mathbf{z}(t)) - Q_{\mathbf{z}(t)}(\mathbf{u}(t)) = \frac{1}{K!} d^K Q_{\mathbf{z}(t)}(\mathbf{u}(t))(\mathbf{z} - \mathbf{u})^K + r(\mathbf{u}, \mathbf{z} - \mathbf{u})$$

where

$$\lim_{\|\mathbf{z}-\mathbf{u}\| \to 0} \frac{\| r(\mathbf{u}, \mathbf{z} - \mathbf{u}) \|}{\| \mathbf{z} - \mathbf{u} \|^K} = 0.$$

Hence

$$\lim_{t \to 0} \| \frac{Q_{\mathbf{z}(t)}(\mathbf{z}(t)) - Q_{\mathbf{z}(t)}(\mathbf{u}(t))}{t^{N_{\mathbf{z}} K}} \|$$

$$= \lim_{t \to 0} \| \frac{d^K Q_{\mathbf{z}(t)}(\mathbf{u}(t))(\mathbf{z} - \mathbf{u})^K}{K!\, t^{N_{\mathbf{z}} K}} + \frac{r(\mathbf{u}, \mathbf{z} - \mathbf{u})}{t^{N_{\mathbf{z}} K}} \|$$

$$\leq \lim_{t \to 0} \| \frac{d^K Q_{\mathbf{z}(t)}(\mathbf{u}(t))(\frac{\mathbf{z}-\mathbf{u}}{t^{N_{\mathbf{z}}}}, \cdots, \frac{\mathbf{z}-\mathbf{u}}{t^{N_{\mathbf{z}}}})}{K!} \| + \lim_{t \to 0} \frac{\| r(\mathbf{u}, \mathbf{z} - \mathbf{u}) \|}{\| \mathbf{z} - \mathbf{u} \|^K} \| \frac{(\mathbf{z} - \mathbf{u})}{t^{N_{\mathbf{z}}}} \|^K$$

$$\leq \lim_{t \to 0} \frac{1}{K!} \| d^K Q_{\mathbf{z}(t)}(\mathbf{u}(t)) \| \| (\frac{\mathbf{z} - \mathbf{u}}{t^{N_{\mathbf{z}}}}, \frac{\mathbf{z} - \mathbf{u}}{t^{N_{\mathbf{z}}}}, \ldots, \frac{\mathbf{z} - \mathbf{u}}{t^{N_{\mathbf{z}}}}) \| +$$

$$\lim_{\|\mathbf{z}-\mathbf{u}\| \to 0} \frac{\| r(\mathbf{u}, \mathbf{z} - \mathbf{u}) \|}{\| \mathbf{z} - \mathbf{u} \|^K} \lim_{t \to 0} \| \frac{(\mathbf{z} - \mathbf{u})}{t^{N_{\mathbf{z}}}} \|^K$$

$$\leq \frac{1}{K!} \| d^K Q_{\mathbf{z}(t)}(\mathbf{u}(t)) \| \lim_{t \to 0} \| (\frac{\mathbf{z} - \mathbf{u}}{t^{N_{\mathbf{z}}}}, \frac{\mathbf{z} - \mathbf{u}}{t^{N_{\mathbf{z}}}}, \ldots, \frac{\mathbf{z} - \mathbf{u}}{t^{N_{\mathbf{z}}}}) \| .$$

The last line is bounded, since $\lim_{t \to 0} \| \frac{\mathbf{z}-\mathbf{u}}{t^{N_{\mathbf{z}}}} \|$ is bounded. Thus the proof of the statement is complete.

Replace the last part of system (3.18) by $\mathbf{f}(t)$ and note that $\mathbf{y}(0) - \mathbf{u}(0) = 0$. Let $\Phi(t, 0)$ be the transition matrix of system (3.18), by Theorem 3.8 we obtain

$$\mathbf{y}(t) - \mathbf{u}(t) = \Phi(t, 0) \int_0^t \Phi(0, s)\mathbf{f}(s)ds. \tag{3.19}$$

We have

$$\lim_{t \to 0} \frac{\int_0^t \Phi(0, s)\mathbf{f}(s)ds}{t^{N_\mathbf{z} K+1}}$$

$$= \lim_{t \to 0} \frac{\Phi(0, t)\mathbf{f}(t)}{(N_\mathbf{z}K + 1)t^{N_\mathbf{z}K}} \quad \text{(by L'Hospital's rule)}$$

$$= \Phi(0, 0) \lim_{t \to 0} \frac{\mathbf{f}(t)}{(N_\mathbf{z}K + 1)t^{N_\mathbf{z}K}}$$

$$= \lim_{t \to 0} \frac{\mathbf{f}(t)}{(N_\mathbf{z}K + 1)t^{N_\mathbf{z}K}}$$

which is bounded by the previous statement, that is

$$\int_0^t \Phi(0, s)\mathbf{f}(s)ds = O(t)^{N_\mathbf{z}K+1}.$$

Note that $\Phi(0, 0) = I_{n \times n}$. Therefore,

$$\mathbf{y} - \mathbf{u} = O(t)^{N_\mathbf{z}K+1},$$

and

$$N(\mathbf{y}) \geq N_\mathbf{z}K = (N(\mathbf{z}) + 1)K.$$

Q. E. D.

From Theorem 3.2 and Theorem 3.6, we see that the degree of the leading error term in the approximate solution generated by waveform relaxation method increases as iteration continues. And from the proofs we can see that the coefficient of the leading error term depends on derivatives of $F(\mathbf{u})$, the right hand side of the original equation, and $G(\mathbf{u}, \mathbf{z})$, the splitting function used. Suppose both F and G are sufficiently smooth, then their derivatives over any finite interval are bounded. If t is chosen sufficiently small, the error will approach 0 when

enough iterations are performed. Therefore, waveform relaxation methods converge over any small interval and converge superlinearly (see [11]), both can be seen from (3.5) and (3.10). This kind of convergence property in waveform relaxation also explains the phenomenon stated in [7] and [15] that the length of the convergent part is increasing after each iteration.

Note that $q \geq 1$ in Theorem 3.6 for any smooth F and G, but the case $q = 2$ is the only other "practical" one. As noted, for large systems even it is not practical because of the cost of computation of the Jacobian and the communication involved. For the WJ method, $q = 1$ so there is little that can be done to get faster order increase. In the next chapter we will examine WGS.

Chapter 4

Accuracy Increase in Waveform Gauss-Seidel

In previous chapter we defined the *order of accuracy* of approximate solutions generated by the waveform relaxation method and showed that the accuracy of whole system after one iteration sweep is at least one order higher than before the sweep starts. In this chapter we will discuss the accuracy increase property for a special approach, the Waveform Gauss-Seidel method. In the Gauss-Seidel approach a numerical ordering (numbering) of the subsystems is chosen to determine the order when a subsystem is to be solved within an iteration sweep. The accuracy increase in WGS is dependent on the numbering of the equations, so a "good" choice of numbering is very important in Gauss-Seidel approach.

In Section 4.1 a result, which is similar to the accuracy increase property for a whole system in Theorem 3.2, is obtained for each subsystem after its computation. One subsystem affects another if any of its variables appear on the right hand side of the differential equations describing the other and such a subsystem is called the incoming subsystem of the other. At each Gauss-Seidel sweep each subsystem is solved following the designated numerical ordering, (using the most recently computed values of other subsystems,) the accuracy increase in a subsystem after its computation can then be accumulated in the subsystem being computed next. The accuracy increase after one sweep of Waveform Gauss-Seidel is, therefore, usually greater than one.

The coupling relation between two subsystems is an oriented relationship and an adjacency matrix can be used to describe these coupling relations among all subsystems. A directed

graph that is built from the adjacency matrix is called the dependency graph of a system. If a system's dependency graph is acyclic we could get the exact solution with only one sweep of Waveform Gauss-Seidel when each subsystem is sequentially solved in a proper order; otherwise iteration sweeps are needed until a sufficiently accurate solution has been computed. Since each subsystem is to be solved as whole, it is denoted as a node in the dependency graph. In Section 4.2 we will introduce ascending chains in a cycle and discuss some accuracy increase properties in the Waveform Gauss-Seidel method under the assumption that the accuracy increase for each subsystem after its computation is exactly one over its incoming subsystems.

In Section 4.3 we will show that the average accuracy increase for the waveform Gauss-Seidel method is equal to the minimum value C/d among all cycles in the dependency graph, where C is the length of a cycle and d is the number of times the numbering of successive nodes around the cycle decreases. Note that the value C depends on the coupling relation after a system is partitioned and the value d depends on the numbering of nodes in the system's dependency graph that is imposed by the Gauss-Seidel approach. So after a system's partitioning, we should order the nodes to maximize this minimum.

4.1 Accuracy Increase for A Subsystem

In Waveform Gauss-Seidel we solve each subsystem sequentially and independently. When a subsystem is solved at one Gauss-Seidel sweep, the remaining subsystems are given approximants to the exact solutions. We then discuss how the accuracies of the remaining subsystems affect the accuracy of a subsystem after it is solved.

Theorem 4.1 *Consider the equation for the i^{th} component after partitioning,*

$$\dot{u}_i = f_i(u_1, \ldots, u_i, \ldots, u_m), \qquad u_i(0) = u_{i,0}. \tag{4.1}$$

The equation to be solved after applying the Gauss-Seidel scheme is

$$\dot{u}_i^{[k+1]} - f_i(u_1^{[k+1]}, \ldots, u_{i-1}^{[k+1]}, u_i^{[k+1]}, u_{i+1}^{[k]}, \ldots, u_m^{[k]}) = 0, \qquad u_i^{[k+1]}(0) = u_{i,0}. \tag{4.2}$$

Assume that

$$\left.\begin{array}{rcll} E_j^{[k+1]} \doteq u_j^{[k+1]} - u_j & = & O(t)^{N_j^{[k+1]}} & for \;\; j \le i, \\ E_j^{[k]} \doteq u_j^{[k]} - u_j & = & O(t)^{N_j^{[k]}} & for \;\; j > i. \end{array}\right\} \tag{4.3}$$

and all the $E_j^{[k]}$'s and $E_j^{[k+1]}$'s are sufficiently smooth. Then

$$N_i^{[k+1]} \ge min(N_1^{[k+1]}, \ldots, N_{i-1}^{[k+1]}, N_{i+1}^{[k]}, \ldots, N_m^{[k]}) + 1, with equality unless there is cancellation. \tag{4.4}$$

Proof: Let

$$M = min(N_1^{[k+1]}, \ldots, N_{i-1}^{[k+1]}, N_{i+1}^{[k]}, \ldots, N_m^{[k]}),$$

then for $r = 0, 1, 2, \ldots, M-1$

$$\left.\begin{array}{rcll} \frac{d^r}{dt^r} E_j^{[k+1]}(0) & = & 0 & for \;\; j = 1, 2, \ldots, i-1, \\ \frac{d^r}{dt^r} E_j^{[k]}(0) & = & 0 & for \;\; j = i+1, \ldots, m. \end{array}\right\} \tag{4.5}$$

From (4.1) and (4.2) we have

$$\begin{aligned} \dot{E}_i^{[k+1]} &= f_i(u_1^{[k+1]}, \ldots, u_i^{[k+1]}, u_{i+1}^{[k]}, \ldots, u_m^{[k]}) - f_i(u_1, \ldots, u_i, u_{i+1}, \ldots, u_m) \\ &= \sum_{j<i} f_{i,j} E_j^{[k+1]} + f_{i,i} E_i^{[k+1]} + \sum_{j>i} f_{i,j} E_j^{[k]}. \end{aligned} \tag{4.6}$$

By (4.2) and (4.5) we see that $\dot{E}_i^{[k+1]}(0) = 0$ from (4.6). Now differentiate (4.6) w.r.t. t to get

$$\begin{aligned} \ddot{E}_i^{[k+1]} &= \{\frac{d}{dt} f_{i,i} E_i^{[k+1]} + f_{i,i} \dot{E}_i^{[k+1]}\} + \\ &\sum_{j<i} \{\frac{d}{dt} f_{i,j} E_j^{[k+1]} + f_{i,i} \dot{E}_j^{[k+1]}\} + \sum_{j>i} \{\frac{d}{dt} f_{i,j} E_j^{[k]} + f_{i,i} \dot{E}_j^{[k]}\}. \end{aligned} \tag{4.7}$$

Since

$$\left.\begin{aligned} \dot{E}_j^{[k+1]}(0) &= E_j^{[k+1]}(0) = 0 \qquad for\ j < i, \\ \dot{E}_j^{[k]}(0) &= E_j^{[k]}(0) = 0 \qquad for\ j > i, \end{aligned}\right\}$$

from (4.3) and $\dot{E}_i^{[k+1]}(0) = 0$, we have $\ddot{E}_i^{[k+1]}(0) = 0$.

Differentiate Equation (4.6) r times, we get the following general form

$$\begin{aligned} \frac{d^{r+1}}{dt^{r+1}} E_i^{[k+1]}(t) &= \sum_{l=0}^{r} \binom{r}{l} \frac{d^l}{dt^l} f_{i,i}(t) \frac{d^{r-l}}{dt^{r-l}} E^{[k+1]}(t) + \\ & \sum_{j<i} \sum_{l=0}^{r} \binom{r}{l} \frac{d^l}{dt^l} f_{i,j}(t) \frac{d^{r-l}}{dt^{r-l}} E^{[k+1]}(t) + \\ & \sum_{j>i} \sum_{l=0}^{r} \binom{r}{l} \frac{d^l}{dt^l} f_{i,i}(t) \frac{d^{r-l}}{dt^{r-l}} E^{[k]}(t). \end{aligned} \tag{4.8}$$

By induction we get $\frac{d^M}{dt^M} E_i^{[k+1]}(0) = 0$. $\frac{d^{M+1}}{dt^{M+1}} E_i^{[k+1]}(0)$ will not be zero if there exists some $j \neq i$ such that $\frac{d^M}{dt^M} E_j^{[k+1]}(0) \neq 0$ for some $j < i$ or $\frac{d^M}{dt^M} E_j^{[k]}(0) \neq 0$ for some $j > i$ with the corresponding $f_{i,j} \neq 0$, unless there is numerical cancellation.

Q. E. D.

Note that only those $j's$ actually appearing on the right hand side of Equation (4.1) affect the derivations in the proof. Thus we have proved that after the computation of a subsystem, its accuracy is at least one order higher than the minimum accuracy of all its input subsystems. In particular , we consider a system of three equations,

$$\begin{aligned} \dot{u}_1^{[k+1]} &= f_1(u_1^{[k+1]}, u_2^{[k]}, u_3^{[k]}) \\ \dot{u}_2^{[k+1]} &= f_2(u_1^{[k+1]}, u_2^{[k+1]}, u_3^{[k]}) \\ \dot{u}_3^{[k+1]} &= f_3(u_1^{[k+1]}, u_2^{[k+1]}, u_3^{[k+1]}). \end{aligned}$$

If

$$\begin{aligned} u_1^{[k]} - u_1 &= O(t)^{N_1^{[k]}} \\ u_2^{[k]} - u_2 &= O(t)^{N_2^{[k]}} \\ u_3^{[k]} - u_3 &= O(t)^{N_3^{[k]}}, \end{aligned}$$

then from previous theorem we have

$$\begin{aligned} N_1^{[k+1]} &\geq min(N_2^{[k]}, N_3^{[k]}) + 1 \\ N_2^{[k+1]} &\geq min(N_1^{[k+1]}, N_3^{[k]}) + 1 \\ N_3^{[k+1]} &\geq min(N_1^{[k+1]}, N_2^{[k+1]}) + 1. \end{aligned}$$

This theorem assumes that all variables appear in all equations. If variable j appears in the equation for variable i only if $j \in I_i$ where I_i is a subset of $[1, \ldots, m]$, then (4.4) can be replaced by

$$N_i^{[k+1]} \geq \min_{j \in I_i}(N_j^{[k+H(i-j)]}) + 1, \tag{4.9}$$

where $H(i-j) = 1$ if $i > j$ and 0 otherwise. (A similar result holds for WJ with H identically zero.)

4.2 Accuracy Increase and Dependency Graphs

In the Gauss-Seidel scheme the numbering of the subsystems is important since it determines the order of their sequential solution. One subsystem affects another if any of its variables appear on the right hand side of the differential equations describing the other. This coupling is an oriented relationship and an adjacency matrix can be used to describe the coupling relations among all subsystems. A directed graph that is built from the adjacency matrix is called the dependency graph of a system. If a system's dependency graph is acyclic we could get the exact solution with only one waveform Gauss-Seidel iteration when each subsystem is sequentially integrated in a proper order; otherwise iterations are needed until a sufficiently accurate solution has been computed. (From now on a subsystem is referred to as a node in a dependency graph.)

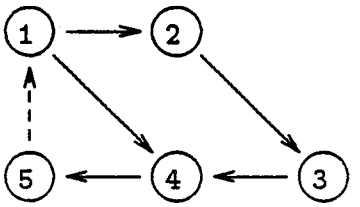
From Theorem 4.1, we know that the order of accuracy at one node after one waveform Gauss-Seidel iteration is at least one greater than the minimum order of its incoming nodes, and possibly more if there is fortuitous cancellation. But the fortuitous cancellation can only occur under very special conditions, so it will be ignored in general. Hence we assume equality in that theorem and investigate some examples to study the accuracy increase of the waveform

Gauss-Seidel method. From these examples we will see that the accuracy increase in the waveform Gauss-Seidel method is related to the coupling and the numbering on a given system's dependency graph.

4.2.1 Accuracy Increase and Ascending Chains

It is instructive to consider some simple examples. In these examples the notation $C/d = a/b$ means that there is a cycle of length C with d ascending chains in it (for detailed definition of ascending chains see def 4.2).

Example 1 : Consider a system with the following dependency graph after partitioning and ordering. This graph has two cycles and all the nodes inside each cycle are sequentially ordered, i.e. there is only one decrease in the numbering of all nodes around each cycle.



The two cycles are $\tilde{A}_1 = \{(1,2,3,4,5)\}$ of length $C_1 = 5$ and $\tilde{A}_2 = \{(1,4,5)\}$ of length $C_2 = 3$. $C_1/d_1 = 5/1$ and $C_2/d_2 = 3/1$. The sole numbering decrease is the branch $(5,1)$ shown as a dashed arrow. We list the order of accuracy and accuracy increase after each waveform Gauss-Seidel iteration in the following tables assuming that we start with $\mathbf{u}^{(0)}(t) = \mathbf{u}_0$.

Order of Accuracy												
Node No	Iteration No											
	0	1	2	3	4	5	6	7	8	9	10	···
1	0	1	4	7	10	13	16	19	22	25	28	···
2	0	2	5	8	11	14	17	20	23	26	29	···
3	0	3	6	9	12	15	18	21	24	27	30	···
4	0	2	5	8	11	14	17	20	23	26	29	···
5	0	3	6	9	12	15	18	21	24	27	30	···

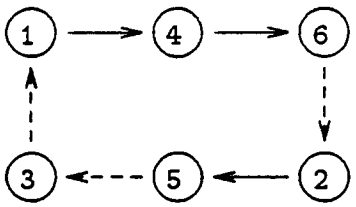
Accuracy Increase												
Node No	Iteration No											
	0	1	2	3	4	5	6	7	8	9	10	...
1		1	3	3	3	3	3	3	3	3	3	...
2		2	3	3	3	3	3	3	3	3	3	...
3		3	3	3	3	3	3	3	3	3	3	...
4		2	3	3	3	3	3	3	3	3	3	...
5		3	3	3	3	3	3	3	3	3	3	...

Form the second table we see that, after the waveform Gauss-Seidel iteration stabilizes, the accuracy increase after one iteration is equal to the minimum cycle length which is "3" in this example. The internal nodes of the cycle with minimum length in this example have been ordered sequentially around this cycle. In general, the internal nodes of a cycle may not be ordered sequentially; if this is the case, we can not achieve the accuracy increase equal to the cycle length in one waveform Gauss-Seidel iteration. However, we will show that, in the case of dependency graph with a single cycle, an accuracy increase equal to the length of the cycle will occur in some number of iterations.

□

Let us consider another example in which the nodes are not sequentially ordered around a cycle.

Example 2 : Consider a dependency graph which contains only one cycle and nodes inside the cycle are not sequentially ordered,



$$\tilde{A} = \{(1,4,6),(2,5),(3)\}; \qquad C/d = 6/3$$

where $(1,4,6),(2,5)$ and (3) are ascending chains of length 3, 2, and 1, respectively, in the given cycle. The tables of order of accuracy and accuracy increase are given below.

Order of Accuracy												
Node	Iteration No											
No	0	1	2	3	4	5	6	7	8	9	10	...
1	0	1	2	4	7	8	10	13	14	16	19	...
2	0	1	4	5	7	10	11	13	16	17	19	...
3	0	1	3	6	7	9	12	13	15	18	19	...
4	0	2	3	5	8	9	11	14	15	17	20	...
5	0	2	5	6	8	11	12	14	17	18	20	...
6	0	3	4	6	9	10	12	15	16	18	21	...

Accuracy Increase												
Node	Iteration No											
No	0	1	2	3	4	5	6	7	8	9	10	...
1		1	1	2	3	1	2	3	1	2	3	...
2		1	3	1	2	3	1	2	3	1	2	...
4		2	1	2	3	1	2	3	1	2	3	...
5		2	3	1	2	3	1	2	3	1	2	...
6		3	1	2	3	1	2	3	1	2	3	...

□

We see in Example 2 that the accuracy increase of "six" , the cycle length, is achieved in three iterations. That is, after the iteration stabilizes, we can pick up six more correct terms in the Taylor expansion of the approximate solution in every three iteration sweeps. We might notice that "three" is the number of times the numbering of successive nodes is "out of order" around the cycle. This is not a coincidence, and it will be seen as a general result, Theorem 4.3, for which we need to define the concept of an *ascending chain* in a cycle.

Suppose an ordering for the Gauss-Seidel method applied on a graph containing a cycle $\tilde{A}$ of length C has been chosen. Number the nodes of the graph according to the Gauss-Seidel ordering.

Definition 4.2 *An ascending chain of length l in a cycle $\tilde{A}$ is a sequence of nodes with numerical ordering $j_0, j_1, \ldots, j_{l-1}$, such that (1) $j_0 < j_1 < \cdots < j_{l-1}$, (2) there exists an edge from node j_i to node j_{i+1} for $i = 0, 1, \ldots, l-2$ in the cycle, and (3) no ascending chain in cycle $\tilde{A}$ contains $\{j_0, j_1, \ldots, j_{l-1}\}$ as a subsequence (in other words, it is as long as possible).*

It follows from the definition that any cycle can be decomposed into a mutually exclusive set of ascending chains and the number of ascending chains in a cycle equals the number of times

the numbering of successive nodes around the cycle decreases. By the equality assumption in Theorem 4.1, we know that after one waveform Gauss-Seidel iteration, each node, j_i, in an ascending chain can not have order of accuracy more than one greater than the order of accuracy of its predecessor node, $j_i - 1$, in the chain at this iteration, while the first node in an ascending chain can not have order of accuracy greater than one plus the order of accuracy, prior to the iteration, of its predecessor, the last node in the chain that precedes it. (If no other node except its predecessor in the cycle is connected to a node k, it will achieve exactly this order increase.)

Now we introduce some simple notations to express these ideas.

Let $\tilde{A}$ be a cycle of length C with d ascending chains. Let $l_1, l_2, \ldots, l_d$ be the lengths of the d ascending chains that follow the orientation of $\tilde{A}$ and W_{n,i,k_i} be the order of accuracy of the k_i^{th} node of the i^{th} ascending chain at the n^{th} waveform Gauss-Seidel iteration. For convenience, define $l_{i+d} = l_i$ and $W_{n,i+d,k} = W_{n,i,k}$ for all i. Then by assuming equality in Theorem 4.1, we have

$$\begin{aligned} W_{n,i,k_i+1} &\leq W_{n,i,k_i} + 1 \\ W_{n+1,i+1,0} &\leq W_{n,i,l_i-1} + 1 \end{aligned}$$

for $k_i = 0, 1, \ldots, l_i - 2$,and $n \geq 1$. For ease of derivation later, we define $W^T_{n,i} \equiv W_{n,i,0}$ and $W^H_{n,i} \equiv W_{n,i,l_i-1}$, i.e. $W^T_{n,i}$ ($W^H_{n,i}$) denotes the order of accuracy at the tail (head) node of the i^{th} ascending chain at the n^{th} waveform Gauss-Seidel iteration. And it is easy to see that

$$W_{n,i,k_i} \leq W^T_{n,i} + k_i \tag{4.10}$$

$$W^H_{n+1,i} \leq W^H_{n,i-1} + l_i \tag{4.11}$$

$$W^T_{n+1,i} \leq W^T_{n,i-1} + l_{i-1}. \tag{4.12}$$

Based on these relations, we then have the following result which says that the accuracy increase after *number-of-ascending-chains* iteration sweeps is bounded by the cycle length.

Theorem 4.3 *If a cycle $\tilde{A}$ of length C consists of d ascending chains, then, after the first iteration, the accuracy increase at the internal nodes of $\tilde{A}$ due to d waveform Gauss-Seidel iterations is bounded by C.*

Proof: By Eqn (4.12) for $n \geq 1$

$$\begin{aligned}
W^T_{n+d,i} &= W^T_{n+d,i+d} \\
&\leq W^T_{n+d-1,i+d-1} + l_{i+d-1} \\
&\leq W^T_{n+d-2,i+d-2} + l_{i+d-2} + l_{i+d-1} \\
&\vdots \\
&\leq W^T_{n+1,i+1} + l_{i+1} + \cdots + l_{i+d-1} \\
&\leq W^T_{n,i} + l_i + l_{i+1} + \cdots + l_{i+d-1} \\
&\leq W^T_{n,i} + C.
\end{aligned}$$

The proof for the remaining nodes in an ascending chain is similar:

$$\begin{aligned}
W_{n+d,i,k_i} &= W_{n+d,i+d,k_i} \\
&\leq W^T_{n+d,i+d} + k_i \\
&\leq W^H_{n+d-1,i+d-1} + 1 + k_i \\
&\leq W^H_{n+d-2,i+d-2} + l_{i+d-1} + 1 + k_i \\
&\leq W^H_{n+d-3,i+d-3} + l_{i+d-2} + l_{i+d-1} + 1 + k_i \\
&\vdots \\
&\leq W^H_{n+1,i+1} + l_{i+2} + \cdots + l_{i+d-2} + l_{i+d-1} + 1 + k_i \\
&\leq W^H_{n,i} + l_{i+1} + l_{i+2} + \cdots + l_{i+d-2} + l_{i+d-1} + 1 + k_i \\
&\leq W_{n,i,k_i} + l_i - 1 - k_1 + l_{i+1} + l_{i+2} + \cdots + l_{i+d-2} + l_{i+d-1} + 1 + k_i \\
&= W_{n,i,k_i} + \sum_{j=1}^{d} l_j \\
&= W_{n,i,k_i} + C.
\end{aligned}$$

Q. E. D.

In particular, when $d = 1$, that is all the internal nodes of a cycle are solved in cyclic order, the accuracy increase in one waveform Gauss-Seidel iteration is then bounded by the cycle length; which is the result we saw in Example 1.

If we assume that each node of a cycle has no other nodes connected to it except its predecessor in the cycle, then its order of accuracy after each waveform Gauss-Seidel iteration is exactly one over its predecessor's in the cycle, i.e.

$$W_{n,i,k_i+1} = W_{n,i,k_i} + 1 \tag{4.13}$$

$$W^T_{n+1,i+1} = W^H_{n,i} + 1 \tag{4.14}$$

for all i and k_i's. Thus

$$W_{n,i,k_i} = W^T_{n,i} + k_i \tag{4.15}$$

$$W^H_{n+1,i} = W^H_{n,i-1} + l_i \tag{4.16}$$

$$W^T_{n+1,i} = W^T_{n,i-1} + l_{i-1} \tag{4.17}$$

for $k_i = 0, 1, \ldots, l_i - 1$, $i = 1, 2, \ldots, d$ and $n \geq 1$. Hence for a single cycle the average accuracy increase in each iteration is C/d. Ignoring fortuitous cancellation, a cycle can not have a greater average accuracy increase, so it is clear that a bound on the average accuracy increase for a graph is given by $\min_i(C_i/d_i)$ where i indexes all the cycles in the graph. We will show that this bound is realized by all the graphs in section 4.3, so that we should order the nodes to maximize this minimum.

4.2.2 Accuracy Increase in a Single Cycle

If we examine the tables in Example 2 carefully we see that the accuracy increase at a given node at successive iteration followed a repetitive pattern after some initial irregularity. For some it was (1,2,3), for others it was (2,3,1), and for the remainder it was (3,1,2). The important property of these patterns is that they are *circular shifts* of a *partition* of the cycle length $C = 6$, where there are $d = 3$ members in the partition. In general we say that a set of d strictly positive integers $\{q_1, q_2, \ldots, q_d\}$ is an *integer partition* of C, if $\sum_{i=1}^{d} q_i = C$.

We now show that given a cycle of length C with d ascending chains of lengths $l_1, l_2, \ldots$, and l_d, and an integer partition, $\{q_1, q_2, \ldots, q_d\}$, of C, if the initial orders of accuracy at all nodes of the cycle are chosen carefully, then the accuracy increase at each node at every d successive waveform Gauss-Seidel iterations is $\{q_d, q_{d-1}, \ldots, q_2, q_1\}$ or its circular shifts.

Theorem 4.4 *Let $\tilde{A}$ be a cycle of length C with d ascending chains of lengths, $l_1, l_2, \ldots, l_d$, and let $\{q_1, q_2, \ldots, q_d\}$ be an integer partition of C. If the initial order of accuracies are chosen such that*

$$\begin{array}{rcll} W^T_{0,i} & = & \sum_{j=i}^{d}(q_j - l_j) & \text{for } i = 1, 2, \ldots, d \\ W_{0,i,k_i} & = & W^T_{0,i} + k_i & \text{for } k_i = 0, 1, \ldots, l_i - 1, \end{array}$$

then

$$W_{n,i,k_i} = W_{n-1,i,k_i} + q_{i-n} \tag{4.18}$$

for $k_i = 0, 1, 2, \ldots, l_i - 1$, $i = 1, 2, \ldots, d$, and $n \geq 1$, where $q_n = q_{n\%d}$ and $n\%d \equiv n \bmod (d)$ for any integer n.

Proof: By Eqn (4.15), $W_{n,i,k_i} - W_{n-1,i,k_i} = W^T_{n,i} - W^T_{n-1,i}$, i.e. all the nodes in an ascending chain have the same accuracy increase after each waveform Gauss-Seidel iteration, so it suffices to show that

$$W^T_{n,i} = W^T_{n-1,i} + q_{i-n} \tag{4.19}$$

for $i = 1, 2, \ldots, d$ and $n \geq 1$.

We prove this theorem by induction on n. When $n = 1$ and $i = 1, 2, \ldots, d$, from Eqn (4.17) and the choice of initial orders, we have

$$\begin{aligned} W^T_{1,i} &= W^T_{0,i-1} + l_{i-1} \\ &= \sum_{j=i-1}^{d}(q_j - l_j) + l_{i-1} \\ &= \sum_{j=i}^{d}(q_j - l_j) + q_{i-1} \\ &= W^T_{0,i} + q_{i-1}. \end{aligned}$$

Hence, (4.19) holds for $n = 1$. Assume that the statement is true for $n \leq m$ and $i = 1, 2, \ldots, d$, i.e.

$$W^T_{m,i} = W^T_{m-1,i} + q_{i-m}. \tag{4.20}$$

Then consider $n = m + 1$ and $i = 1, 2, \ldots, d$, from Eqns (4.16), (4.17), and (4.20) we have

$$\begin{aligned} W^T_{m+1,i} &= W^T_{m,i-1} + l_{i-1} \\ &= W^T_{m-1,i-1} + q_{(i-1)-m} + l_{i-1} \\ &= W^T_{m,i} + q_{(i-1)-m} \\ &= W^T_{m,i} + q_{i-(m+1)} \end{aligned}$$

Hence, by mathematical induction, Eqn (4.19) holds for all $n \geq 1$.

Q. E. D.

From Eqn (4.18) and the periodic behavior of q_j's we have

$$\begin{aligned} W_{n+d,i,k_i} &= W_{n,i,k_i} + \sum_{j=n+1}^{n+d} q_{i-j} \\ &= W_{n,i,k_i} + \sum_{j=1}^{d} q_j \\ &= W_{n,i,k_i} + C, \end{aligned}$$

which are exactly the results we saw in previous examples. Since all the nodes in an ascending chain have the same accuracy increase after each waveform Gauss-Seidel iteration, without loss of generality from now on we can use the head or tail node at each ascending chain to discuss the accuracy increase property. In Table 4.1 we list the accuracy increase at the tail node of each ascending chain after each waveform Gauss-Seidel iteration using the result in Theorem 4.4. From this table it is easy to see that the accuracy increase at any node in every d successive iterations is $\{q_d, q_{d-1}, \ldots, q_1\}$ or its circular shifts. To avoid the decreasing subscripts in q_j's as iteration proceeds, we let $p_j \doteq q_{d-j}$ for $j = 0, 1, \ldots, d-1$ and rewrite Table 4.1 to obtain Table 4.2.

Using this new table with some simple manipulations we have the following formula for the order of accuracy at the n^{th} waveform Gauss-Seidel iteration:

$$W^T_{n,i} = \sum_{j=i}^{d} (p_{d-j} - l_j) + \lfloor \frac{n}{d} \rfloor \times C + \sum_{j=1}^{n\%d} p_{d-i+j}, \tag{4.21}$$

for $i = 1, 2, \ldots, d$ and $n \geq 1$.

If we choose a specific integer partition of the cycle length C, we will not only have a nice formula for the order of accuracy at each node, but will have an accuracy increase pattern that can not be destroyed by other cycles in the same graph that do not have a smaller C/d.

	Iteration Index							
	0	1	2	$\cdots$	d	d+1	d+2	$\cdots$
l_1	$0 = \sum_{j=1}^{d}(q_j - l_j)$	$+q_d$	$+q_{d-1}$	$+\cdots$	$+q_1$	$+q_d$	$+q_{d-1}$	$+\cdots$
l_2	$l_1 - q_1 = \sum_{j=2}^{d}(q_j - l_j)$	$+q_1$	$+q_d$	$+\cdots$	$+q_2$	$+q_1$	$+q_d$	$+\cdots$
l_3	$l_1 - q_1 + l_2 - q_2 = \sum_{j=3}^{d}(q_j - l_j)$	$+q_2$	$+q_1$	$+\cdots$	$+q_3$	$+q_2$	$+q_1$	$+\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
l_i	$\sum_{1}^{i-1}(l_j - q_j) = \sum_{j=i}^{d}(q_j - l_j)$	$+q_{i-1}$	$+q_{i-2}$	$+\cdots$	$+q_i$	$+q_{i-1}$	$+q_{i-2}$	$+\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
l_{d-1}	$q_d - l_d + q_{d-1} - l_{d-1}$	$+q_{d-2}$	$+q_{d-3}$	$+\cdots$	$+q_{d-1}$	$+q_{d-2}$	$+q_{d-3}$	$+\cdots$
l_d	$q_d - l_d$	$+q_{d-1}$	$+q_{d-2}$	$+\cdots$	$+q_d$	$+q_{d-1}$	$+q_{d-2}$	$+\cdots$

Table 4.1: Accuracy increase at tail node of each ascending chain

	Iteration Index							
	0	1	2	$\cdots$	d	d+1	d+2	$\cdots$
l_1	$0 = \sum_{j=1}^{d}(p_{d-j} - l_j)$	$+p_0$	$+p_1$	$+\cdots$	$+p_{d-1}$	$+p_0$	$+p_1$	$+\cdots$
l_2	$\sum_{j=2}^{d}(p_{d-j} - l_j)$	$+p_{d-1}$	$+p_0$	$+\cdots$	$+p_{d-2}$	$+p_{d-1}$	$+p_0$	$+\cdots$
l_3	$\sum_{j=3}^{d}(p_{d-j} - l_j)$	$+p_{d-2}$	$+p_{d-1}$	$+\cdots$	$+p_{d-3}$	$+p_{d-2}$	$+p_{d-1}$	$+\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
l_i	$\sum_{j=i}^{d}(p_{d-j} - l_j)$	$+p_{d-i+1}$	$+p_{d-i+2}$	$+\cdots$	$+p_{d-i}$	$+p_{d-i+1}$	$+p_{d-i+2}$	$+\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
l_{d-1}	$p_0 - l_d + p_1 - l_{d-1}$	$+p_2$	$+p_3$	$+\cdots$	$+p_1$	$+p_2$	$+p_3$	$+\cdots$
l_d	$p_0 - l_d$	$+p_1$	$+p_2$	$+\cdots$	$+p_0$	$+p_1$	$+p_2$	$+\cdots$

Table 4.2: Accuracy increase at tail node of each ascending chain

Lemma 4.5 *Given a cycle of length C with d ascending chains of lengths $l_1, l_2, \ldots, l_d$, respectively. If*

$$p_j = \lfloor (j+1)\frac{C}{d} \rfloor - \lfloor j\frac{C}{d} \rfloor \qquad \text{for } j = 0, 1, \ldots, d-1, \tag{4.22}$$

then $\{p_0, p_1, \ldots, p_{d-1}\}$ is an integer partition of C. Further, if

$$W^T_{0,i} = C - \sum_{j=i}^{d} l_j + \lfloor (1-i)\frac{C}{d} \rfloor \tag{4.23}$$

then

$$W^T_{n,i} = C - \sum_{j=i}^{d} l_j + \lfloor (n+1-i)\frac{C}{d} \rfloor \tag{4.24}$$

for $i = 1, 2, \ldots, d$ and $n \geq 1$.

Proof: Since $\frac{C}{d} = \lfloor \frac{C}{d} \rfloor + \epsilon$, for some $0 \leq \epsilon < 1$, by assumption

$$\begin{aligned} p_j &= \lfloor (j+1)\frac{C}{d} \rfloor - \lfloor j\frac{C}{d} \rfloor \\ &= \lfloor j\frac{C}{d} + \lfloor \frac{C}{d} \rfloor + \epsilon \rfloor - \lfloor j\frac{C}{d} \rfloor \\ &= \lfloor \frac{C}{d} \rfloor + \lfloor j\frac{C}{d} + \epsilon \rfloor - \lfloor j\frac{C}{d} \rfloor \\ &\geq \lfloor \frac{C}{d} \rfloor \\ &\geq 1 \end{aligned}$$

and

$$\begin{aligned} \sum_{0}^{d-1} p_j &= \sum_{0}^{d-1} \lfloor (j+1)\frac{C}{d} \rfloor - \lfloor j\frac{C}{d} \rfloor \\ &= \lfloor d\frac{C}{d} \rfloor - \lfloor 0\frac{C}{d} \rfloor \\ &= C, \end{aligned}$$

therefore $\{p_0, p_1, \ldots, p_{d-1}\}$ is an integer partition of C. Moreover

$$\begin{aligned} \sum_{j=1}^{n\%d} p_{d-i+j} &= \lfloor (d-i+n\%d+1)\frac{C}{d} \rfloor - \lfloor (d-i+1)\frac{C}{d} \rfloor \\ &= \lfloor (n\%d+1-i)\frac{C}{d} \rfloor - \lfloor (1-i)\frac{C}{d} \rfloor. \end{aligned} \tag{4.25}$$

Substituting (4.25) into (4.21), we have

$$\begin{aligned}
W^T_{n,i} &= \sum_{j=i}^{d}(p_{d-j} - l_j) + \lfloor \frac{n}{d} \rfloor \times C + \lfloor (n\%d + 1 - i)\frac{C}{d} \rfloor - \lfloor (1-i)\frac{C}{d} \rfloor \\
&= \sum_{j=i}^{d}(p_{d-j} - l_j) + \lfloor \lfloor \frac{n}{d} \rfloor \times C + (n\%d + 1 - i)\frac{C}{d} \rfloor - \lfloor (1-i)\frac{C}{d} \rfloor \\
&= \sum_{j=i}^{d}(p_{d-j} - l_j) + \lfloor (\lfloor \frac{n}{d} \rfloor \times d + n\%d + 1 - i)\frac{C}{d} \rfloor - \lfloor (1-i)\frac{C}{d} \rfloor \\
&= \sum_{j=i}^{d}(p_{d-j} - l_j) + \lfloor (n + 1 - i)\frac{C}{d} \rfloor - \lfloor (1-i)\frac{C}{d} \rfloor \\
&= \sum_{j=i}^{d}(p_{d-j} - l_j) + \lfloor (n + 1 - i)\frac{C}{d} \rfloor + \sum_{j=1}^{i-1} p_{d-j} \\
&= \sum_{j=1}^{d} p_{d-j} - \sum_{j=i}^{d} l_j + \lfloor (n + 1 - i)\frac{C}{d} \rfloor \\
&= C - \sum_{j=i}^{d} l_j + \lfloor (n + 1 - i)\frac{C}{d} \rfloor
\end{aligned}$$

Q. E. D.

For a cycle of length C with d ascending chains, the set of integers, $\{p_0, p_1, \ldots, p_{d-1}\}$, defined in (4.22) is called the *natural partition* of C with respect to d. Let us use Example 2 again but choose the initial orders specified by (4.23), that is, based on the natural partition {2,2,2} of the cycle of length 6. The accuracy increase pattern is:

Order of Accuracy							
Node No	Iteration No						
	0	1	2	3	4	5	...
1	0	2	4	6	8	10	...
2	1	3	5	7	9	11	...
3	1	3	5	7	9	11	...
4	1	3	5	7	9	11	...
5	2	4	6	8	10	12	...
6	2	4	6	8	10	12	...

Accuracy Increase							
Node No	Iteration No						
	0	1	2	3	4	5	...
1		2	2	2	2	2	...
2		2	2	2	2	2	...
3		2	2	2	2	2	...
4		2	2	2	2	2	...
5		2	2	2	2	2	...
6		2	2	2	2	2	...

From the table we see that the phenomena described in Theorem 4.4 and Lemma 4.5 are satisfied. Next we examine how one cycle interacts with the remainder of a graph.

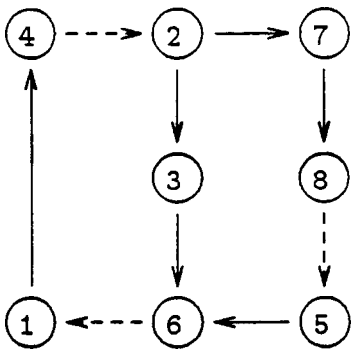
4.2.3 Accuracy Increase in General Graphs

We now discuss general graphs. Our analysis technique will be to analyze part of the graph and consider *driving terms* from other parts of the graph. These driving terms are the branches entering the part of the graph selected for analysis. The orders on the nodes at the start of these branches may, or may not, reduce the order of subsequent iterations of nodes in the selected part of the graph.

From Lemma 4.5 we see that, for a cycle of length C with d ascending chains, if the initial orders at all nodes of the cycle are chosen properly and if a driving term, if there is any, does not interfere with the order of accuracy in this cycle, then at the n^{th} waveform Gauss-Seidel iteration, the order of accuracy at each node of the cycle can be expressed as $\beta + \lfloor (n+\gamma)\frac{C}{d} \rfloor$ for some constant integers β and γ. If such a cycle is the only cycle in a system's dependency graph, then all the nodes in the dependency graph that are reachable from this cycle will have a similar pattern for their accuracy increase. (A node U is reachable from a cycle if there exists a directed path from any node in this cycle to U.)

Now let's look at an example first, which shows that result in Lemma 4.5 is satisfied not only by the nodes in the cycle with minimum C/d but also by any node that is reachable from that cycle.

Example 3: Consider a dependency graph that has two cycles and nodes are ordered as shown.



$\tilde{A}_1 = \{(1,4),(2,7,8),(5,6)\}; \quad C_1/d_1 = 7/3$
$\tilde{A}_2 = \{(1,4),(2,3,6)\}; \quad C_2/d_2 = 5/2$

Since cycle $\tilde{A}_1$ has smaller C/d ratio, we choose the initial orders inside $\tilde{A}_1$ according to (4.23) in Lemma 4.5. That is the *natural partition* {2,2,3} of 7, length of $\tilde{A}_1$, is considered. Below we list the order of accuracy and accuracy increase at each node.

Order of Accuracy												
Node No	Iteration No											
	0	1	2	3	4	5	6	7	8	9	10	...
1	0	3	5	7	10	12	14	17	19	21	24	...
2	0	2	5	7	9	12	14	16	19	21	23	...
3	4	3	6	8	10	13	15	17	20	22	24	...
4	1	4	6	8	11	13	15	18	20	22	25	...
5	1	3	5	8	10	12	15	17	19	22	24	...
6	2	4	6	9	11	13	16	18	20	23	25	...
7	1	3	6	8	10	13	15	17	20	22	24	...
8	2	4	7	9	11	14	16	18	21	23	25	...

Accuracy Increase												
Node No	Iteration No											
	0	1	2	3	4	5	6	7	8	9	10	...
1		3	2	2	3	2	2	3	2	2	3	...
2		2	3	2	2	3	2	2	3	2	2	...
3		-1	3	2	2	3	2	2	3	2	2	...
4		3	2	2	3	2	2	3	2	2	3	...
5		2	2	3	2	2	3	2	2	3	2	...
6		2	2	3	2	2	3	2	2	3	2	...
7		2	3	2	2	3	2	2	3	2	2	...
8		2	3	2	2	3	2	2	3	2	2	...

From the accuracy increase table, we see that the result in Lemma 4.5 is satisfied by all the other nodes, besides the nodes in cycle $\tilde{A}_1$, in the graph.

Next we want to see what will occur if the initial accuracies are not specified with the natural partition of 7. Let us use the partition {1,2,4} of 7 to specify the initial accuracies and list the order of accuracy and accuracy increase at all nodes.

Order of Accuracy												
Node No	Iteration No											
	0	1	2	3	4	5	6	7	8	9	10	...
1	0	4	5	7	10	12	14	17	19	21	24	...
2	1	2	6	7	9	12	14	16	19	21	23	...
3	4	3	7	8	10	13	15	17	20	22	24	...
4	1	5	6	8	11	13	15	18	20	22	25	...
5	2	4	5	9	10	12	15	17	19	22	24	...
6	3	4	6	9	11	13	16	18	20	23	25	...
7	2	3	7	8	10	13	15	17	20	22	24	...
8	3	4	8	9	11	14	16	18	21	23	25	...

Accuracy Increase												
Node No	Iteration No											
	0	1	2	3	4	5	6	7	8	9	10	...
1		4	1	2	3	2	2	3	2	2	3	...
2		1	4	1	2	3	2	2	3	2	2	...
3		-1	4	1	2	3	2	2	3	2	2	...
4		4	1	2	3	2	2	3	2	2	3	...
5		2	1	4	1	2	3	2	2	3	2	...
6		1	2	3	2	2	3	2	2	3	2	...
7		1	4	1	2	3	2	2	3	2	2	...
8		1	4	1	2	3	2	2	3	2	2	...

In this case we see that the partition {1,2,4} does not appear in the accuracy increase table, whereas the natural partition {2,2,3} of 7 does. Let us try another partition {1,1,5} of 7 to specify the initial orders and see how it affects the accuracy increase pattern.

Order of Accuracy												
Node No	Iteration No											
	0	1	2	3	4	5	6	7	8	9	10	...
1	0	5	5	7	10	12	14	17	19	21	24	...
2	1	2	7	7	9	12	14	16	19	21	23	...
3	4	3	8	8	10	13	15	17	20	22	24	...
4	1	6	6	8	11	13	15	18	20	22	25	...
5	3	4	5	10	10	12	15	17	19	22	24	...
6	4	4	6	9	11	13	16	18	20	23	25	...
7	2	3	8	8	10	13	15	17	20	22	24	...
8	3	4	9	9	11	14	16	18	21	23	25	...

Accuracy Increase												
Node	Iteration No											
No	0	1	2	3	4	5	6	7	8	9	10	...
1		5	0	2	3	2	2	3	2	2	3	...
2		1	5	0	2	3	2	2	3	2	2	...
3		-1	5	0	2	3	2	2	3	2	2	...
4		5	0	2	3	2	2	3	2	2	3	...
5		1	1	5	0	2	3	2	2	3	2	...
6		0	2	3	2	2	3	2	2	3	2	...
7		1	5	0	2	3	2	2	3	2	2	...
8		1	5	0	2	3	2	2	3	2	2	...

From the last table we see that this partition {1,1,5} of 7 is also not preserved in the accuracy increase pattern. From above discussions we may conclude that when a dependency graph contains more than one cycle, non-natural partitions of the length of the cycle with minimum C/d in the graph may not be preserved in the accuracy increase pattern when it is used to specify the initial orders.

□

Now we select for analysis any cycle with a minimum value of C/d. We will call this the *minimum cycle.* Let us initialize the orders in the graph such that the nodes in this cycle follow the pattern specified in (4.23) and all other nodes in the graph are infinitely accurate. (This is not possible in practice, but is used to show that the minimum cycle determines the average accuracy increase.) We know from Lemma 4.5 that the minimum cycle will maintain an average order increase of C/d unless a driving term restricts the order of some node in the cycle. Since all other nodes were initially set to order infinity, the only way for this to happen is for the orders of a chain of nodes starting from some point on the minimum cycle and ending on the minimum cycle (called the nodes on a *sidetrack path* of the minimum cycle) to be lowered by the minimum cycle so as to reduce the order of the minimum cycle. Suppose the chain is as shown in Figure 4.1 on the sidetrack path from node U to node V on cycle $\tilde{A}$, where cycle $\tilde{A}$ is a *minimum cycle.*

For the remaining part of this section, we want to show that the order in cycle $\tilde{A}$ will not be lowered by any sidetrack path.

Let $l_1, l_2, \ldots, l_d$ be the lengths of the d ascending chains of cycle $\tilde{A}$. Then $\sum_{j=1}^{d} l_j = C$.

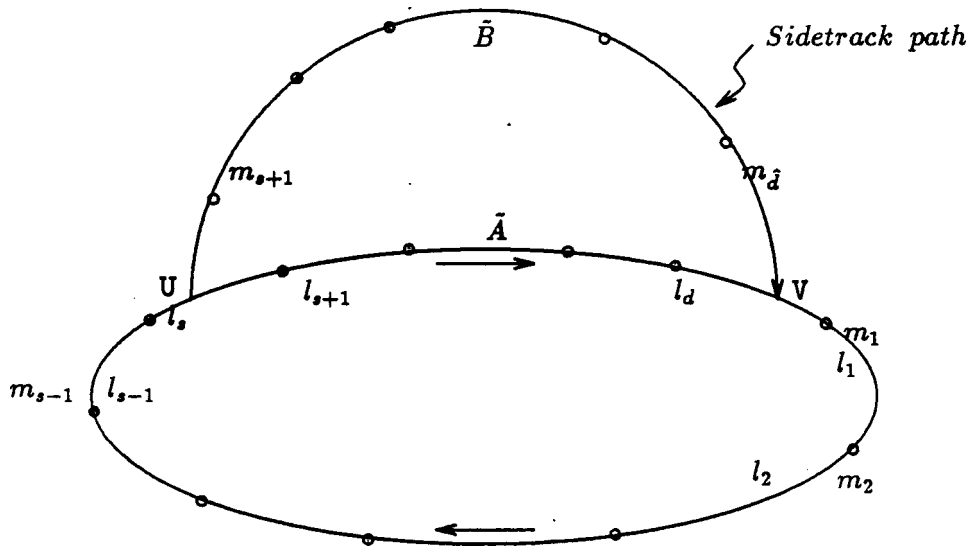


Figure 4.1: A minimum cycle with a sidetrack path

Let $\tilde{B}$ be the cycle consisting of any sidetrack path of $\tilde{A}$ from U to V and the path from V to U on $\tilde{A}$. Suppose $\tilde{B}$ has $\hat{d}$ ascending chains of length $m_1, m_2, \ldots, m_{\hat{d}}$ and $\sum_{j=1}^{\hat{d}} m_j = \hat{C}$. By assumption, $C/d \leq \hat{C}/\hat{d}$. We now number the ascending chains on both cycles such that node U is in the s^{th} chain of both cycles and node V is in the d^{th} chain of cycle $\tilde{A}$ and in the $\hat{d}^{th}$ chain of cycle $\tilde{B}$. This means $m_j = l_j$ for $j = 1, 2, \ldots, s-1$. Furthermore we assume node U is the k_s^{th} node in chain s and node V is the k_d^{th} node in chain d of cycle $\tilde{A}$. (Note the indexing of a node in a chain starts with 0.) So node V will be the $(m_{\hat{d}} - l_d + k_d)^{th}$ node in chain $\hat{d}$ of cycle $\tilde{B}$.

Now we show that the order coming into node V from the path on $\tilde{A}$ is no greater than the order coming from the sidetrack path.

Theorem 4.6 *Let W_{n,i,k_i} ($\hat{W}_{n,i,k_i}$) denote the order of the k_i^{th} node in chain i of cycle $\tilde{A}$ ($\tilde{B}$) at the n^{th} iteration. If*

$$W_{0,i,k_i} = C - \sum_{j=i}^{d} l_j + \lfloor (1-i)\frac{C}{d} \rfloor + k_i \qquad \text{for nodes on cycle } \tilde{A}$$

and

$$\hat{W}_{0,i,k_i} = W_{0,i,k_i} \qquad \text{for nodes on path V to U}$$
$$\hat{W}_{0,i,k_i} = \infty \qquad \text{for nodes on the sidetrack path,}$$

Then

$$W_{n,d,k_d-1} \leq \hat{W}_{n,\hat{d},(m_{\hat{d}}-l_d+k_d)-1}. \tag{4.26}$$

Note that the $(k_d-1)^{th}$ node on chain d of cycle $\tilde{A}$ and the $((m_{\hat{d}}-l_d+k_d)-1)^{th}$ node on chain $\hat{d}$ of cycle $\tilde{B}$ are predecessors of node V. (There may be others; they can be considered by the same mechanism.)

Proof: From Figure 4.1 we can see that the order of node U propagating through the sidetrack path will not affect node V until $\hat{d}-s+1$ iterations later. So the first possible lowering of order at node V by the sidetrack path will happen at the $(\hat{d}-s+1)^{th}$ iteration. Thus we first show that

$$W_{\hat{d}-s+1,d,k_d-1} \leq \hat{W}_{\hat{d}-s+1,\hat{d},(m_{\hat{d}}-l_d+k_d)-1}. \tag{4.27}$$

Since

$$\begin{aligned}
\hat{W}_{\hat{d}-s+1,\hat{d},(m_{\hat{d}}-l_d+k_d)-1} &= \hat{W}^T_{\hat{d}-s+1,\hat{d}} + (m_{\hat{d}}-l_d+k_d)-1 \\
&= \hat{W}^H_{\hat{d}-s,\hat{d}-1} + (m_{\hat{d}}-l_d+k_d) \\
&= \hat{W}^H_{1,s} + \sum_{j=s+1}^{\hat{d}-1} m_j + (m_{\hat{d}}-l_d+k_d) \\
&= \hat{W}^T_{1,s} + m_s - 1 + \sum_{j=s+1}^{\hat{d}-1} m_j + (m_{\hat{d}}-l_d+k_d) \\
&= W^T_{1,s} + \sum_{j=s}^{\hat{d}} m_j - l_d + k_d - 1 \\
&= C - \sum_{j=s}^{d} l_j + \lfloor (2-s)\frac{C}{d} \rfloor + \sum_{j=i}^{\hat{d}} m_j - l_d + k_d - 1 \\
&= \sum_{j=1}^{s-1} l_j + \lfloor (2-s)\frac{C}{d} \rfloor + \sum_{j=s}^{\hat{d}} m_j - l_d + k_d - 1 \\
&= \sum_{j=1}^{s-1} m_j + \lfloor (2-s)\frac{C}{d} \rfloor + \sum_{j=s}^{\hat{d}} m_j - l_d + k_d - 1 \\
&= \hat{C} + \lfloor (2-s)\frac{C}{d} \rfloor - l_d + k_d - 1
\end{aligned}$$

and

$$\begin{aligned} W_{\hat{d}-s+1,d,k_d-1} &= W^T_{\hat{d}-s+1,d} + k_d - 1 \\ &= C - l_d + \lfloor (\hat{d} - s + 1 + 1 - d)\frac{C}{d} \rfloor + k_d - 1 \\ &= C - l_d + \lfloor (\hat{d} - s + 2)\frac{C}{d} - C \rfloor + k_d - 1 \\ &= \lfloor (\hat{d} - s + 2)\frac{C}{d} \rfloor - l_d + k_d - 1, \end{aligned}$$

to prove (4.27) it suffices to show that

$$\lfloor (\hat{d} - s + 2)\frac{C}{d} \rfloor \leq \hat{C} + \lfloor (2 - s)\frac{C}{d} \rfloor.$$

This is easy, since $\frac{C}{d} \leq \frac{\hat{C}}{\hat{d}} \Rightarrow \hat{d}\frac{C}{d} \leq \hat{C}$, and hence

$$\lfloor (\hat{d} - s + 2)\frac{C}{d} \rfloor = \lfloor \hat{d}\frac{C}{d} + (2 - s)\frac{C}{d} \rfloor \leq \lfloor \hat{C} + (2 - s)\frac{C}{d} \rfloor = \hat{C} + \lfloor (2 - s)\frac{C}{d} \rfloor.$$

In subsequent iterations we have

$$\begin{aligned} \hat{W}_{n+\hat{d}-s+1,\hat{d},(m_{\hat{d}}-l_d+k_d)-1} &= W^T_{n+1,s} + \sum_{j=s}^{\hat{d}} m_j - l_d + k_d - 1 \\ &= \hat{C} + \lfloor (n + 2 - s)\frac{C}{d} \rfloor - l_d + k_d - 1 \end{aligned}$$

and

$$W_{n+\hat{d}-s+1,d,k_d-1} = \lfloor (n + \hat{d} - s + 2)\frac{C}{d} \rfloor - l_d + k_d - 1$$

from which (4.26) follows directly.

Q. E. D.

Theorems 4.3 and 4.6 are the key results used to prove Theorem 4.7 in the next section which is the main result of this chapter.

4.3 Average Accuracy Increase

We define the *average accuracy increase* of a node to be the limit of p_n/n as $n \to \infty$ where p_n is the order of accuracy of a node and n is the iteration number. The average accuracy increase of a numbered graph is the minimum average accuracy increase over all nodes. Theorem 4.3

shows that the average accuracy increase in a waveform Gauss-Seidel method can not exceed the C/d of the minimum cycle. Theorem 4.6 shows that if the minimum cycle is initialized to the natural accuracy (accuracy specified by the natural partition of its length) and the remaining nodes are infinitely accurate, no sidetrack paths destroy the natural accuracy. These results can now be combined to show that the average accuracy increase for the waveform Gauss-Seidel method is exactly the C/d of the minimum cycle in the dependency graph of a given system after partitioning and ordering.

Theorem 4.7 *Suppose a minimum cycle in the dependency graph of a given system is of length C and has d ascending chains. Then the waveform Gauss-Seidel method applied to this system has average accuracy increase C/d.*

Proof: Consider two identical dependency graphs, $\hat{G}_1$ and $\hat{G}_2$, with N nodes and identical numberings. If we start with all initial orders of accuracy on both graphs set to zero and run the waveform Gauss-Seidel iterations on both synchronously, the order of accuracy of corresponding nodes on the two graphs will be the same at all steps.

After M iterations, we will perturb the iteration on $\hat{G}_2$ in the following way: lower the order of accuracy at the nodes in the minimum cycle, $\tilde{A}_2$, of $\hat{G}_2$ following (4.23), i.e. using the natural partition of C. Then resume waveform Gauss-Seidel iterations on both graphs. The accuracy at any node in $\hat{G}_2$ will never surpass the accuracy of its corresponding node in $\hat{G}_1$. This is easy to see inductively: the new accuracy at a node being integrated is equal to one plus the minimum accuracy of its predecessors on a graph. If the accuracy of every node in $\hat{G}_1$ is at least as large as the accuracy of its corresponding node in $\hat{G}_2$, then the same condition holds after the integration and hence before the next integration. Therefore, the average accuracy increase of $\hat{G}_2$ is a lower bound for the average accuracy increase of the unperturbed problem on $\hat{G}_1$.

It remains to show that the average accuracy increase for $\hat{G}_2$ is C/d. Since M is fixed, the first M iterations can be ignored in computing the average. The important step is to choose M large enough that the orders of accuracy of the nodes not in the minimum cycle $\tilde{A}_2$ of $\hat{G}_2$ are effectively infinite at the perturbation, so that the result in Theorem 4.6 applies. Note that (i) after M iterations starting from accuracy 0, all nodes have accuracy $\geq M$, and (ii)

when a cycle is set to the accuracy specified by its natural partition as in (4.23), the accuracies assigned are $\leq N$. Hence the nodes not in $\tilde{A}_2$ will have an order of accuracy at least $M - N$ greater than those in $\tilde{A}_2$ after the perturbation.

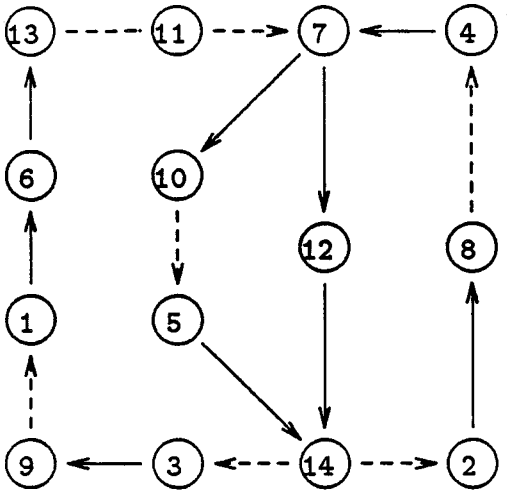
Now consider iterations on $\hat{G}_2$ after the perturbation. If the average accuracy increase is less than C/d, the graph $\hat{G}_2 - \tilde{A}_2$ must be lowering the order of $\tilde{A}_2$ (Theorem 4.6 implies that the propagation of an accuracy from $\tilde{A}_2$ into $\hat{G}_2 - \tilde{A}_2$ and back to $\tilde{A}_2$ can not be responsible for lowering the accuracy). Since the order of accuracy of $\hat{G}_2 - \tilde{A}_2$ can be made arbitrarily higher than that of $\tilde{A}_2$ at the perturbation ($M - N$ higher), the average accuracy of $\hat{G}_2 - \tilde{A}_2$ must be less than C/d. The argument can now be completed by induction on the size of the graph: It is certainly true for $N = 2$. Assume it is true for $2 < N \leq K - 1$. If $|\hat{G}_2| = K$ than either $\hat{G}_2$ has no cycles or $|\hat{G}_2 - \tilde{A}_2| \leq K - 2$ where $\tilde{A}_2$ is a minimum cycle. Hence the average accuracy increase of $\hat{G}_2 - \tilde{A}_2$ is at least C/d so it can not lower the order of $\tilde{A}_2$.

Q. E. D.

This tells us what the average is, but not how to number the graph to minimize that average. Finding such a numbering appears to be NP-hard, so a heuristic approach will almost certainly have to be used. The result above suggests using heuristics that attempt to maximize the length of ascending chains are appropriate, particularly those in short cycles.

Next we look at some examples to illustrate this result.

Example 4: Consider a dependency graph that has four cycles and the nodes are ordered as shown.



$\tilde{A}_1 = \{(1,6,13),(11),(7,10),(5,14),(3.9)\}$
$\tilde{A}_2 = \{(1,6,13),(11),(7,12,14),(3.9)\}$
$\tilde{A}_3 = \{(2,8),(4,7,10),(5,14))\}$
$\tilde{A}_4 = \{(2,8),(4,7,12,14)\}$

$C_1/d_1 = 10/5$
$C_2/d_2 = 9/4$
$C_3/d_3 = 7/3$
$C_4/d_4 = 6/2$

From the graph we see that cycle $\tilde{A}_1$ is the minimum cycle. We would like to see how the minimum cycle affects the remaining cycles in the graph. First we use the same initial orders for all nodes in the graph and list the order of accuracy and accuracy increase at all nodes.

Order of Accuracy																	
Node	Iteration No																
No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	···
1	0	1	3	5	7	8	11	13	15	17	18	21	23	25	27	28	···
2	0	1	3	5	6	9	11	13	15	16	19	21	23	25	26	29	···
3	0	1	3	5	6	9	11	13	15	16	19	21	23	25	26	29	···
4	0	1	3	5	7	8	11	13	15	17	18	21	23	25	27	28	···
5	0	1	3	4	7	9	11	13	14	17	19	21	23	24	27	29	···
6	0	2	4	6	8	9	12	14	16	18	19	22	24	26	28	29	···
7	0	1	2	5	7	9	11	12	15	17	19	21	22	25	27	29	···
8	0	2	4	6	7	10	12	14	16	17	20	22	24	26	27	30	···
9	0	2	4	6	7	10	12	14	16	17	20	22	24	26	27	30	···
10	0	2	3	6	8	10	12	13	16	18	20	22	23	26	28	30	···
11	0	1	4	6	8	10	11	14	16	18	20	21	24	26	28	30	···
12	0	2	3	6	8	10	12	13	16	18	20	22	23	26	28	30	···
13	0	3	5	7	9	10	13	15	17	19	20	23	25	27	29	30	···
14	0	2	4	5	8	10	12	14	15	18	20	22	24	25	28	30	···

Accuracy Increase																	
Node	Iteration No																
No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	···
1		1	2	2	2	1	3	2	2	2	1	3	2	2	2	1	···
2		1	2	2	1	3	2	2	2	1	3	2	2	2	1	3	···
3		1	2	2	1	3	2	2	2	1	3	2	2	2	1	3	···
4		1	2	2	2	1	3	2	2	2	1	3	2	2	2	1	···
5		1	2	1	3	2	2	2	1	3	2	2	2	1	3	2	···
6		2	2	2	2	1	3	2	2	2	1	3	2	2	2	1	···
7		1	1	3	2	2	2	1	3	2	2	2	1	3	2	2	···
8		2	2	2	1	3	2	2	2	1	3	2	2	2	1	3	···
9		2	2	2	1	3	2	2	2	1	3	2	2	2	1	3	···
10		2	1	3	2	2	2	1	3	2	2	2	1	3	2	2	···
11		1	3	2	2	2	1	3	2	2	2	1	3	2	2	2	···
12		2	1	3	2	2	2	1	3	2	2	2	1	3	2	2	···
13		3	2	2	2	1	3	2	2	2	1	3	2	2	2	1	···
14		2	2	1	3	2	2	2	1	3	2	2	2	1	3	2	···

From the table we see that after every 5 iterations the accuracy at each node increases by 10, the cycle length of $\tilde{A}_1$. And the increase pattern at each node is {2,2,2,1,3}, or its circular shifts, which happens to be the ascending chain lengths of $\tilde{A}_1$ in reverse orientation. Next we choose the initial orders inside $\tilde{A}_1$ such that the natural partition {2,2,2,2,2} of C_1 is considered.

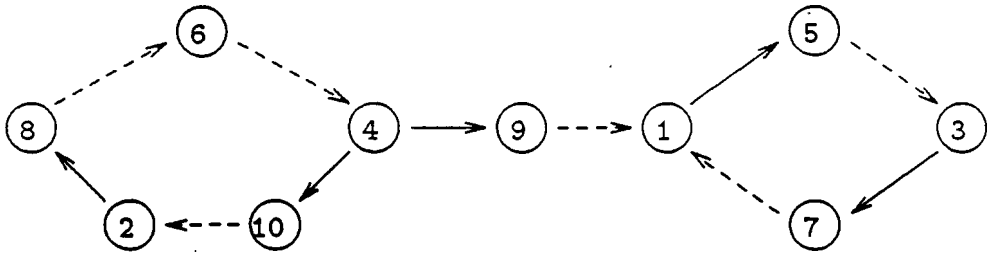
Order of Accuracy																	
Node	Iteration No																
No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	···
1	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	···
2	1	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	···
3	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	···
4	1	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	···
5	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	···
6	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	···
7	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	···
8	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	···
9	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	···
10	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	···
11	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	···
12	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	···
13	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	···
14	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	···

Accuracy Increase																	
Node	Iteration No																
No	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	···
1		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
2		1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
3		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
4		1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
5		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
6		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
7		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
8		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
9		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
10		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
11		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
12		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
13		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···
14		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	···

From the table we see that all the nodes in the graph have the same accuracy increase pattern, the natural partition of C_1.

□

Example 5 Consider a dependency graph that has two cycles and one of the cycles is reachable by the other, but not vice versa.



$\tilde{A}_1 = \{(2,8),(6),(4,10)\}; \quad C_1/d_1 = 5/3$
$\tilde{A}_2 = \{(1,5),(3,7)\}; \quad C_2/d_2 = 4/2$

Again we list the order of accuracy and accuracy increase at all nodes after each waveform Gauss-Seidel iteration. From the table we can see that the minimum cycle dominates the accuracy increase for the entire graph. Had the coupling from node 1 to node 4 via 9 been in the alter direction, cycle $\tilde{A}_1$ would have sustained an average increase of $5/3$ while cycle $\tilde{A}_2$ and node 9 an average increase of $4/2$.

Order of Accuracy												
Node No	Iteration No											
	0	1	2	3	4	5	6	7	8	9	10	...
2	0	1	3	4	6	8	9	11	13	14	16	...
3	0	1	3	5	6	8	10	11	13	15	16	...
4	0	1	2	4	6	7	9	11	12	14	16	...
5	0	2	4	5	7	9	10	12	14	15	17	...
6	0	1	3	5	6	8	10	11	13	15	16	...
7	0	2	4	6	7	9	11	12	14	16	17	...
8	0	2	4	5	7	9	10	12	14	15	17	...
9	0	2	3	5	7	8	10	12	13	15	17	...
10	0	2	3	5	7	8	10	12	13	15	17	...

Accuracy Increase												
Node No	Iteration No											
	0	1	2	3	4	5	6	7	8	9	10	...
1		1	2	1	2	2	1	2	2	1	2	...
2		1	2	1	2	2	1	2	2	1	2	...
3		1	2	2	1	2	2	1	2	2	1	...
4		1	1	2	2	1	2	2	1	2	2	...
5		2	2	1	2	2	1	2	2	1	2	...
6		1	2	2	1	2	2	1	2	2	1	...
7		2	2	2	1	2	2	1	2	2	1	...
8		2	2	1	2	2	1	2	2	1	2	...
9		2	1	2	2	1	2	2	1	2	2	...
10		2	1	2	2	1	2	2	1	2	2	...

□

4.4 Conclusion

We assumed that each node in a system's dependency graph belongs to at least one cycle in this chapter . The reason for such an assumption is that any nodes that do not belong to a cycle have no effect (except the source node at the first iteration) on the accuracy increase of the remaining nodes; hence they can be ignored when discussing the accuracy increase property for the whole system.

We have proved that the accuracy of a subsystem after each iteration sweep would be one order higher than the minimum accuracy of its incoming subsystems. We also showed that after a system is partitioned and the subsystems are ordered, the average accuracy increase

for the whole system is $\min C/d$, where C and d are determined by partitioning and ordering, respectively. Hence we may want to partition a system and order its subsystems so that the value $\min C/d$ is maximized. But such an optimization problem is probably NP hard so not computationally feasible. However, the result suggests heuristics that could be used to maximize the lengths of ascending chain in short cycles.

Although the waveform Gauss-Seidel is usually thought of as a serial method, there are several ways in which it can be used for parallel computation. In one approach, the integration of later nodes can be staggered in time. The first node is integrated over a small interval of time, $[t_0, t_1]$. Its output is then ready to be used for the integration of the second node over that interval while the first node is integrated over a second interval, $[t_1, t_2]$, and so on. In the m-th step, the i-th node is being integrated over the $(m+1-i)$-th time interval for $i = 1, \ldots, m$. Its inputs for that time interval from earlier variables with lower i values have already been computed. In the $(m+1)$-st step, the first node begins the second iteration of the first interval while each of the others advance.

In another approach, several nodes can be integrated in parallel. In this approach, for a p processor system, up to p nodes can be numbered with each number in the sequence so long as there are no branches between nodes with the same number. At each step of a single waveform Gauss-Seidel sweep, all nodes with the same number can be integrated in parallel. Since they are mutually independent, the order result of Theorem 4.1 applies, and hence all results in this chapter apply.

Chapter 5

Variant Gauss-Seidel Approaches

In this chapter we will discuss various approaches for solving systems with special type of dependency graphs; graphs whose nodes can be sequentially ordered within each cycle. That is, there is only one ascending chain in each cycle. By the results in Chapter 4, the accuracy increase in one sweep of Waveform Gauss-Seidel is bounded by the minimum cycle length of the corresponding dependency graph. In Section 5.1 we will examine some examples and see that more accuracy increase may be achieved by treating each cycle of minimum length in a given dependency graph as a new component so that the minimum cycle length of the reduced dependency graph is larger than before.

For systems with special type of coupling, it is possible to achieve more accuracy increase with a variant Gauss-Seidel method than the plain Gauss-Seidel does after each iteration sweep. This approach is to apply Gauss-Seidel iteration to shorter cycles more frequently than longer ones. That is, at each iteration sweep in addition to the sequential computation of all subsystems, more computations are performed, also sequentially, on components that form smaller cycles in the dependency graph. We call this approach *Hierarchical Gauss-Seidel* (HGS). From the same examples we will see that the accuracy increase after one sweep of Hierarchical Gauss-Seidel is bounded by the maximum cycle length of the given dependency graph. In Section 5.2 we use a measure to compare the efficiencies of Gauss-Seidel and hierarchical Gauss-Seidel. We introduce a method to determine when it is more efficient to use hierarchical Gauss-Seidel than regular Gauss-Seidel for problems with a special type of dependency graph.

5.1 Hierarchical Gauss-Seidel and Regrouping

In this section, we investigate four examples to study the accuracy increase property in Gauss-Seidel scheme with or without local relaxation. In these examples we also show that it is possible to improve accuracy increase after one sweep of Gauss-Seidel iteration by repartitioning. From these examples we see that the accuracy increase after one sweep of Gauss-Seidel like scheme are related to the cycle lengths in the dependency graph being used. This motivates us to choose a partition for a system such that the resulting dependency graph has a larger minimum cycle length. To simplify the computation and clarify the ideas, from now on we assume that the order of accuracy after the computation of each subsystem is exactly one above the minimum of its inputs.

Example 1:Suppose a system's adjacency matrix has the following structure.

$$\begin{bmatrix} \times & & & & \times \\ \times & \times & & & \\ & \times & \times & & \\ & & \times & \times & \\ \times & & & \times & \times \end{bmatrix}$$

Its corresponding dependency graph drawn below has two cycles, $\{1, 5\}$, and $\{1, 2, 3, 4, 5\}$.

$$\begin{array}{ccccc} (1) & \rightarrow & (2) & & \\ \updownarrow & & & \searrow & \\ (5) & \leftarrow & (4) & \leftarrow & (3). \end{array} \tag{5.1}$$

From the dependency graph we can easily compute the order of accuracy for each component by picking the minimum order of accuracy among all its incoming components and adding 1 to the minimum. Proceed with every component in the numerical ordering we then construct the accuracy table for the system such that the (i, j) entry of the accuracy table contains the order of accuracy of the i^{th} component after its computation at the j^{th} Gauss-Seidel iteration sweep. The accuracy table for the given system is:

Order of Accuracy		Iteration Index				
		0	1	2	3	4
	1	0	1	3	5	7
Node	2	0	2	4	6	8
	3	0	3	5	7	9
Index	4	0	4	6	8	10
	5	0	2	4	6	8

From the accuracy table we can see that after the iteration stabilizes the accuracy increase after each W GS iteration sweep is 2, which is the length of the shortest cycle in the dependency graph shown in (5.1).

We now solve the system by W GS with local relaxation on the shortest cycle {1,5}, i.e. besides solving each component once, more computations are performed on components 1 and 5 alternately until their accuracies fail to improve. We then have the following accuracy table.

Order of Accuracy		Iteration Index									
		0	1			2			3		
	1	0	1	3	5	6	8	10	11	13	15
Node	2	0	2			7			12		
	3	0	3			8			13		
Index	4	0	4			9			14		
	5	0	2	4	5	7	9	10	12	14	15

In this table we list the order of accuracy for each component whenever it is computed within an iteration sweep. So the accuracy for each component after one iteration sweep is indicated by the last nonzero entry at each row within an iteration index column. We can see from this table that after one sweep of W GS with local relaxation, the accuracy increase becomes 5, which is the largest cycle length of the dependency graph in (5.1).

Since {1, 5} forms the smaller cycle in the original dependency graph, we can group these two components together and consider the following dependence relation.

$$\begin{array}{ccccc} & & (2) & & \\ & \nearrow & & \searrow & \\ (1,5) & \leftarrow & (4) & \leftarrow & (3). \end{array} \tag{5.2}$$

Applying Gauss-Seidel to the new partitioning, we have the following accuracy table.

Order of Accuracy		Iteration Index				
		0	1	2	3	4
Compo-	1,5	0	1	5	9	13
nent	2	0	2	6	10	14
	3	0	3	7	11	15
Index	4	0	4	8	12	16

From the table, we see that the accuracy increase for W GS under this partition is 4, the length of the only cycle in graph (5.2).

Example 2: Consider a system with adjacency matrix:

$$\begin{bmatrix} \times & & & & \times \\ \times & \times & & & \\ & \times & \times & & \\ \times & & \times & \times & \\ & & & \times & \times \end{bmatrix}.$$

The corresponding dependency graph is:

$$\begin{array}{ccccc} (1) & \rightarrow & (2) & & \\ \uparrow & \searrow & & \searrow & \\ (5) & \leftarrow & (4) & \leftarrow & (3). \end{array} \tag{5.3}$$

It is easy to see that there are two cycles, $\{1, 4, 5\}$ and $\{1, 2, 3, 4, 5\}$, in the dependency graph. Follow the dependence relation we construct the following accuracy table of Waveform **Gauss-Seidel applied on the given system.**

Order of Accuracy		Iteration Index				
		0	1	2	3	4
	1	0	1	4	7	10
Node	2	0	2	5	8	11
	3	0	3	6	9	12
Index	4	0	2	5	8	11
	5	0	3	6	9	12

As we expect, after a few sweeps the accuracy increase after each sweep is 3, the length of the shortest cycle in the dependency graph (5.3). Now applying W GS with local relaxation on its

shortest cycle {1, 4, 5}, we have the following accuracy table.

Order of Accuracy		Iteration Index								
		0	1		2		3		4	
	1	0	1	4	6	9	11	14	16	19
Node	2	0	2		7		12		17	
	3	0	3		8		13		18	
Index	4	0	2	4	7	9	12	14	17	19
	5	0	3	5	8	10	13	15	18	20

From this table we see that the accuracy increase after each compound sweep is 5, the maximum cycle length of the dependency graph shown in (5.3).

Next we group the components 1, 4, and 5 together and consider the following partitioning.

$$\begin{array}{ccc} (2) & & \\ \uparrow & \searrow & \\ (1,4,5) & \leftarrow & (3) \end{array} \tag{5.4}$$

The accuracy table of W GS applied to the new dependency graph (5.4) is:

Order of Accuracy		Iteration Index				
		0	1	2	3	4
Compo-	1,4,5	0	1	4	7	10
nent	2	0	2	5	8	11
Index	3	0	3	6	9	12

The accuracy increase after each W GS sweep is now 3, the only cycle length on graph (5.4).

Example 3: Consider a system with the following adjacency matrix.

$$\begin{bmatrix} \times & & & & \times \\ \times & \times & & & \\ & \times & \times & & \\ & \times & \times & \times & \\ & & & \times & \times \end{bmatrix}$$

The corresponding dependency graph has two cycles, {1, 2, 4, 5} and {1, 2, 3, 4, 5} in it.

$$\begin{array}{ccccc} (1) & \rightarrow & (2) & & \\ \uparrow & & \downarrow & \searrow & \\ (5) & \leftarrow & (4) & \leftarrow & (3) \end{array} \tag{5.5}$$

Use the dependency graph we can construct the accuracy table of Gauss-Seidel easily. It is:

Order of Accuracy		Iteration Index				
		0	1	2	3	4
Node Index	1	0	1	5	9	13
	2	0	2	6	10	14
	3	0	3	7	11	15
	4	0	3	7	11	15
	5	0	4	8	12	16

From the table we see that the order of accuracy after each iteration sweep increased by 4, the minimum cycle length of the dependency graph shown in (5.5).

If solving the same system by W GS with local relaxation on its shortest cycle {1, 2, 4, 5} we have the following accuracy table.

Order of Accuracy		Iteration Index								
		0	1		2		3		4	
Node Index	1	0	1	5	6	10	11	15	16	20
	2	0	1	6	7	11	12	16	17	21
	3	0	3		8		13		18	
	4	0	3	4	8	9	13	14	18	19
	5	0	4	5	9	10	14	15	19	20

The table shows that accuracy increase after each sweep is 5, the maximum cycle length of the dependency graph shown in (5.5). If we group all four components, 1, 2, 4, 5 together, we have the dependence relation below.

$$(1, 2, 4, 5) \longleftrightarrow (3)$$

It is easy to conclude that the accuracy increase under this partition will be 2 after each W GS sweep.

Example 4: Consider a system whose adjacency matrix has the following structure.

$$\begin{bmatrix} \times & & & & \times & \\ \times & \times & \times & & & \\ & \times & \times & & & \\ & & \times & \times & & \times \\ & & & \times & \times & \\ & & & & \times & \times \end{bmatrix}$$

The corresponding dependency graph is drawn below,

$$\begin{array}{ccccc} (5) & \rightarrow & (1) & \rightarrow & (2) \\ \downarrow & \nwarrow & & & \updownarrow \\ (6) & \rightarrow & (4) & \leftarrow & (3) \end{array} \tag{5.6}$$

which has three cycles, {2, 3}, {4, 5, 6}, and {1, 2, 3, 4, 5}. Proceed as before, the order of accuracy at each W GS iteration is listed in the following table.

Order of Accuracy		Iteration Index						
		0	1	2	3	4	5	6
Node Index	1	0	1	3	7	9	11	13
	2	0	1	3	5	7	9	11
	3	0	2	4	6	8	10	12
	4	0	1	4	7	9	11	13
	5	0	2	5	8	10	12	14
	6	0	3	6	9	11	13	15

From this table we see that after iteration stabilizes the accuracy increase after each sweep is 2, the minimum cycle length in the dependency graph (5.6).

Next we use Gauss-Seidel with local relaxation on both the cycles {2, 3} and {4, 5, 6} in graph (5.6). In the following accuracy table we list the order of accuracy for each component whenever it is computed. Therefore the accuracy of one component after a complete iteration sweep is indicated by the last nonzero entry on its corresponding row within an iteration index column.

Order of Accuracy		Iteration Index													
		0	1				2				3				
	1	0	1				6				11				
Node	2	0	1	2	2		4	6	7		9	11	12		
	3	0	2	3	3		5	7	8		10	12	13		
	4	0			1	4			7	9			12	14	
Index	5	0			2	5			8	10			13	15	
	6	0			3	6			9	11			14	16	

From this table we see that the accuracy increase after each sweep is now 5, the maximum cycle length in the original dependency graph (5.6).

If we solve the internal components of the shortest cycle simultaneously, i.e. we group 2 and 3 together to form a bigger component, the new reduced dependency graph now becomes

$$\begin{array}{ccccc} (5) & \rightarrow & (1) & & \\ \downarrow & \nwarrow & & \searrow & \\ (6) & \rightarrow & (4) & \leftarrow & (2,3). \end{array} \tag{5.7}$$

Applying Gauss-Seidel to this new partitioning and the accuracy table is:

Order of Accuracy		Iteration Index					
		0	1	2	3	4	5
Compo-	1	0	1	3	6	9	12
nent	2,3	0	2	4	7	10	13
	4	0	1	4	7	10	13
	5	0	2	5	8	11	14
Index	6	0	3	6	9	12	15

The accuracy increase after each sweep is now 3, the minimum cycle length in the reduced dependency graph shown in (5.7). When Gauss-Seidel is used, this partitioning has better accuracy increase than the one shown in (5.6). If we now only group 4, 5, and 6, the internal components of another cycle in graph (5.6), together, and apply Gauss-Seidel to the following reduced dependency graph.

$$\begin{array}{ccc} (1) & \rightarrow & (2) \\ \uparrow & & \updownarrow \\ (4,5,6) & \leftarrow & (3) \end{array} \tag{5.8}$$

The new accuracy table is:

Order of Accuracy		Iteration Index						
		0	1	2	3	4	5	6
Compo-	1	0	1	4	6	8	10	12
nent	2	0	1	3	5	7	9	11
	3	0	2	4	6	8	10	12
Index	4,5,6	0	3	5	7	9	11	13

From the table we see that the accuracy increase after each W-GS sweep is 2, the minimum cycle length in the reduced dependency graph shown in (5.8). This approach does not improve the accuracy increase when compared with the first approach in this example.

5.2 Efficiency for Variant Gauss-Seidel Approaches

Different GS like approaches can achieve different accuracy increases by different amounts of work. We could compare different approaches with a measure of *efficiency*. Ideally, we define *efficiency* as *(accuracy increase per sweep) / (total work per sweep)*. If the cost of the integration of node i is w_i then the total work per sweep is $W = \sum_{i \in I} w_i$, where I is a multiset containing an entry for each integration of each node. In general, we do not have the relative size of w_i, but could proceed assuming all $w_i = 1$ to get $W = \| I \|$, the number of elements in W. Thus we have the following definition:

$$EF = Efficiency \doteq \frac{accuracy\ increase\ per\ sweep}{total\ units\ of\ work\ per\ sweep}.$$

Using this definition, we now measure the efficiencies for the Gauss-Seidel without and with local relaxation in first three examples. They are

Efficiency	GS	HGS
Example 1	2/5	5/9
Example 2	3/5	5/8
Example 3	4/5	5/9

From these data we see that it is not always more efficient to use Gauss-Seidel with local relaxation. And for systems with a special type of dependency graph, we have found a criterion to determine when it is more efficient to use Gauss-Seidel with local relaxation.

Theorem 5.1 *If there are only two cycles of different lengths in a system's dependency graph and these two cycles have at least one common node, then the Gauss-Seidel method without local relaxation is more efficient when the ratio of the length of the smaller cycle to the length of the larger cycle exceeds* $\frac{\sqrt{5}-1}{2}$*, the reciprocal of the golden ratio.*

Note: we have assumed that all internal nodes of each cycle have been solved sequentially in cyclic order.

Proof: Let the lengths of these two cycles be m and n, respectively, with $m < n$ and let $p = \lceil \frac{n}{m} \rceil$. Since $m < n$, we have $p \geq 2$. If the system is solved by Gauss-Seidel without local relaxation, it is easy to show that the accuracy increase in on sweep is exactly m, length of the smaller cycle. Since n units of work increases the order of accuracy by m, the efficiency for the Gauss-Seidel method is

$$EF(GS) = \frac{m}{n}.$$

If, besides solving each node once, we perform $p-1$ more Gauss-Seidel iterations on the smaller cycle, the accuracy increase at each node now becomes $min(n, \lceil \frac{n}{m} \rceil m) = n$. This shows that it is always possible to achieve accuracy increase by n after one sweep of Gauss-Seidel with local relaxation for this type of problems. And to achieve "n order increase" in one sweep, we need do $n + (\lceil \frac{n}{m} - 1 \rceil)m$ units of work, so the efficiency for the Gauss-Seidel method with local relaxation is

$$EF(HGS) = \frac{n}{n + (\lceil \frac{n}{m} \rceil - 1)m}.$$

The difference in efficiency of these two schemes is

$$\begin{aligned} & EF(HGS) - EF(GS) \\ = \; & \frac{n}{n+(p-1)m} - \frac{m}{n} \\ = \; & \frac{n^2 - (mn + (p-1)m^2)}{n(n+(p-1)m)} \end{aligned} \qquad (4.4a)$$

From the definition of p we have

$$
\begin{array}{rrcccl}
 & p-1 & < & \frac{n}{m} & \leq & p \\
\Leftrightarrow & \frac{n}{p} & \leq & m & < & \frac{n}{p-1} \qquad (4.4b) \\
\Leftrightarrow & \frac{1}{p}n^2 & \leq & mn & < & \frac{1}{(p-1)}n^2 \\
(4.4b) \Rightarrow & \frac{1}{p^2}n^2 & \leq & m^2 & < & \frac{1}{(p-1)^2}n^2 \\
\Rightarrow & \frac{p-1}{p^2}n^2 & \leq & (p-1)m^2 & < & \frac{1}{p-1}n^2 \\
\Rightarrow & \frac{2p-1}{p^2}n^2 & \leq & mn+(p-1)m^2 & < & \frac{2}{p-1}n^2 \qquad (4.4c)
\end{array}
$$

When $p \geq 3$, i.e. when $m < n/2$, from (4.4c) we have $mn + (p-1)m^2 < \frac{2}{2}n^2 = n^2$ and by (4.4a), this implies

$$EF(HGS) > EF(GS).$$

When $p = 2$, that is when $m \geq \frac{n}{2}$, by (4.4c) we have

$$\frac{3}{4}n^2 \leq mn + (p-1)m^2 < 2n^2.$$

This inequality does not help deciding which scheme is more efficient but it does tell us that for some $n/2 \leq m < n$, Gauss-Seidel with local relaxation is less efficient. In order for $EF(HGS) \geq EF(GS)$ to hold, from (4.4a) we need to solve

$$n^2 - mn - m^2 \geq 0,$$

which in turn requires that

$$m \leq \frac{\sqrt{5}-1}{2}n.$$

Therefore Gauss-Seidel with local relaxation is more efficient for the special type of problems when $\frac{m}{n} \leq \frac{\sqrt{5}-1}{2}$, the reciprocal of the golden ratio.

Q. E. D.

If we compare the efficiencies in the first three examples, we can see that they all follow this criterion.

Chapter 6

Numerical Experiments

In this final chapter, we analyze the numerical results obtained from an experimental package WRODE for solving system of ordinary differential equations using multirate techniques in waveform relaxation setting.

In section 6.1 we describe some implementation issues. In section 6.2 some experimental results are presented, the experiments in this section are designed to show how the selection of different Gauss-Seidel numberings and the Jacobi approach on waveform relaxation affects the efficiency. We also test the effects of window length on the overall performance of waveform relaxation. We present the statistical results of the waveform relaxation process both in table and graph formats. In the final section we examine the efficiency of the WGS method.

6.1 Implementation

WRODE is a code written in C to implement the waveform Gauss-Seidel and Jacobi iteration. It performs the direct method if no partitioning of a system is chosen. Users must provide the partitioning and numbering (ordering) of a given system and the coupling relations among all subsystems. WRODE will determine the window size automatically according to a user specified ratio of window length to maximum stepsize among all subsystems. At each waveform iteration sweep over a window, multirate integration method is used. A multirate integration method is one in which different equations are integrated using different time steps. The main objective of a multirate method is to reduce the integration time by using larger time stepsizes for those variables having slow behavior when compared to the fastest variables. This

approach obviously should use less computing time because the total computing time is roughly proportional to the number of integration steps taken for each equation, and in this approach we can reduce the number of steps taken by the slow components. Besides the reduction in the number of intergration steps, there are extra savings in matrix computations if implicit methods are used. Instead of solving a large system, several smaller subsystems are solved. The work required is roughly $O(\sum_{i=1}^{M} N_i^3)$ compared to $O((\sum_{i=1}^{M} N_i)^3)$. These savings are partially offset by the cost of interpolations used to compute approximations of variables integrated with large steps so that the derivatives of variables integrated with small steps can be computed. However,these additional costs are reduced if the system is sparse, because not all variables need to be interpolated or extrapolated to evaluate derivatives.

The basic approach of the implementation is

Repeat

1. Pick a window.
2. Perform waveform relaxation over the window until all waveforms converged.

Until whole integration interval is covered.

Note that if we treat each window as a time step and the waveform relaxation process as the predictor-corrector process in regular ODE integrator, then the waveform relaxation approach can be seen as a large scale ODE integrator.

For the rest of this section we discuss some issues that have been implemented in the code.

Data Representation

In WRODE the Nordsieck representation,

$$[y, hy', \frac{h^2 y''}{2!}, \ldots, \frac{h^k y^{(k)}}{k!}],$$

is used. Using this representation, no recomputation of coefficients is needed when the stepsize is changed, only scaling is required. At each step, a matrix multiplication by a Pascal triangle is performed for the predictor. The arithmetic work involved is proportional to both the size of the system and the square of the order. Therefore this representation is suitable for smaller systems and lower order methods. In multirate integration methods, we partition a big system into several smaller subsystems, then integrate each subsystem independently. Because of the small size of each subsystem, Nordsieck vector representation seems a good choice.

Mesh Point Synchronization

For further saving in computation time and reduction in error, it is desirable to reduce unnecessary interpolations by synchronizing the mesh points among all subsystems, that is, to force the mesh of slower components to be a subset of that of faster components. This can be achieved by letting the stepsize of a component be an integer multiple of the stepsize of the next faster component (if fixed step methods are used). To achieve as much synchronization as possible in variable step methods, we will limit stepsize changes to halving, doubling or powers of the same. A step may be halved at any time, but it may be doubled only when $(t - t_0)/h$ is even, where h is the current stepsize and t_0 is the initial time point of the current window. (This scheme guarantees that $(t - t_0)/h$ is an integer and tells us how far the integration has proceeded, measured in units of the current stepsize.)

Selection of Window Length

In general, after integrating the current time point, t_c, the integrator recommends a new stepsize, h_{new} and at the beginning of integration a scheme suggested by Shampine [13] is used to choose an initial stepsize, h_{init}. Based on the stepsizes suggested by the integrator for each subsystem, we have experimented with different choices of window length by choosing a two's-power multiple of the largest suggested stepsize among all subsystems at the start of a new window as the new window length. We maintain the window-length to stepsize ratio throughout entire integration interval.

System Partitioning and Numbering (Ordering)

The purpose of partitioning is to use as large a stepsize as possible for each subsystem to gain efficiency. Hence system partitioning plays a critical role in the overall performance of multirate methods. The order in which subsystems are integrated affects the rate of convergence in the Gauss-Seidel approach significantly. In the implementation we use static partitioning and ordering, i.e. we use the same partitioning and ordering throughout entire integration interval. But a lot of experiments are designed to see the effect of these two factors on the performance of WRODE and the numerical results all match the theoretical results discussed in Chapter 4.

Partial Waveform Convergence Exploitation

After each waveform has been computed, we need to check whether a waveform has converged over a window. If it has not, we first determine how far it has converged. Based on this information we can then locate the starting point of reintegration (over the same window) for each subsystem. We observe how much work can be saved by exploiting this property in the experiments.

Interpolation Methods

From the discussion in [3], we know that the order of discretization convergence of a multirate integration method depends partially on the order of the interpolation methods used. In the code the second order Hermitian interpolation method is used.

6.2 Numerical Results

In this section we compare the efficiency of different Gauss-Seidel numberings and the Jacobi approach on waveform relaxation. We also test the effects of window length on the overall performance of waveform relaxation. We present the statistical results of the waveform relaxation process both in table and graph formats.

At the upper left corner of each table is the label of the example. If the label is followed by '/' and a number, then this number indicates the number of subsystems being used. The partitioning and ordering used follows the example label. Each table contains the following fields (CPU time maybe missing in some tables) :

CPU time : Total time used (in seconds) for the waveform relaxation process. This does not include the time spent for reading inputs and setting up the initial waveforms.

total iterations : Total number of window iterations performed. If there are two numbers in this field separated by '/', the first one is the total iterations in convergent windows[1] and the second number denotes the total number of window iterations performed.

window numbers : Total number of convergent windows.

iteration/window : Average iterations needed to reach convergence in a window.

[1] A convergent window is a window over which WGS or WJ converges in less than or equal to five iterations.

steps performed : Total number of successful integration steps performed.

non-converged steps : Number of successful integration steps that are actually needed (convergent steps[2] are excluded) to be performed to reach convergence.

steps redundant (%) : The percentage of convergent steps that are reintegrated.

interpolations : Total number of interpolations performed. Note that interpolations may be needed when evaluating the derivatives at an intergration step as well as when comparing two successive waveforms to detect convergence.

function evaluations : Total number of function evaluations performed.

In the graphs we present the profiles of cpu time, integration steps, interpolations, non-converged steps and average iterations with respect to different ratio of window-length to stepsize. We also show the comparison of average iterations between Jacobi and Gauss-Seidel in different numbering or the comparison between different partitioning.

6.2.1 Linear Problems

In this subsection we consider linear systems of the following form :

$$\dot{\mathbf{y}} = A(\mathbf{y} - \phi(t)) + \dot{\phi}(t), \qquad \mathbf{y}(0) = \phi(0), \qquad 0 \leq t \leq 10, \tag{6.1}$$

It is easy to see that the exact solution to this type of problem is $\mathbf{y}(t) = \phi(t)$.

In the following five examples we consider five different matrices A's. In examples 6.1, 6.2 and 6.3, we examine matrices of size 4. Using 6.1 and 6.2, we study how WRODE performs on systems with different minimum cycle lengths. In example 6.3, we examine how the coupling factor affects the performance of WRODE. For the last two experiments, example 6.4 and 6.5, we consider matrices of size 6 and we show how the size of couplings between the fast and slow components affects the performance of WRODE. The Jacobian for these two examples has the 2 by 2 matrices on the diagonal

$$\begin{bmatrix} -50 & 49 \\ 49 & -50 \end{bmatrix}, \qquad \begin{bmatrix} -6 & 5 \\ 5 & -6 \end{bmatrix}, \qquad \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$

[2] A convergent step is an integration step that provides the same answer as the previous iteration (within desired error tolerance).

which have eigenvalues -99 and -1, -11 and -1, -1 and -1 or -2 and 0, respectively. Thus considered as systems by themselves, the slow class is stiff, the medium class is mildly stiff and the fast class is not stiff at all.

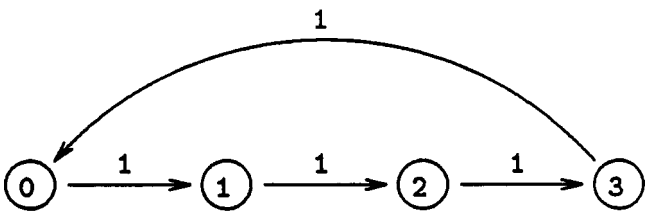


Figure 6.1: Dependency Graph of Example 6.1

Example 6.1 : In this example, we consider

$$A = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -5 & 0 & 0 \\ 0 & 1 & -10 & 0 \\ 0 & 0 & 1 & -20 \end{bmatrix}$$

and

$$\phi(t) = (cos(t), sin(t), cos(20t), sin(20t))^T.$$

We use Gauss-Seidel and Jacobi schemes in WRODE to solve this system and analyze their performance. Note that node 2 and node 3 are the fast components.

Partitioning the system into four subsystems each containing only one equation, labeled by (0), (1), (2), and (3), respectively. The dependency graph under this partitioning has a single cycle of length four. (See Fig 6.1) We compare the effects of different Gauss-Seidel numberings on the speed of convergence.

The label of each subsystem is fixed throughout different numberings. If a subsystem label, say (a), takes the precedence of another one, say (b), in a numbering, then subsystem (a) is integrated before subsystem (b). For example, the numbering (3102) means that subsystem (3) is integrated first, followed by subsystem (1), then subsystem (0) and subsystem (2) is integrated last. Under different numberings the cycle is partitioned into one, two, or three ascending chains, respectively. According to Theorem 4.7 the average order of accuracy increase after each Gauss-Seidel iteration is, respectively, 4/1, 4/2, and 4/3 for one, two and three ascending chains. Practically, we can not know exactly how many terms have been picked up after each iteration but from the numerical results obtained, among all Gauss-Seidel numberings, the numbering with one ascending chain has the best speed of convergence, which is expected,

while the numbering with three ascending chains has the worst speed of convergence. The Jacobi approach has only one order of accuracy increase after each iteration so its speed of convergence is worse than any of the Gauss-Seidel numbering. (See Graphs 6.1.1 - 6.1.3.)

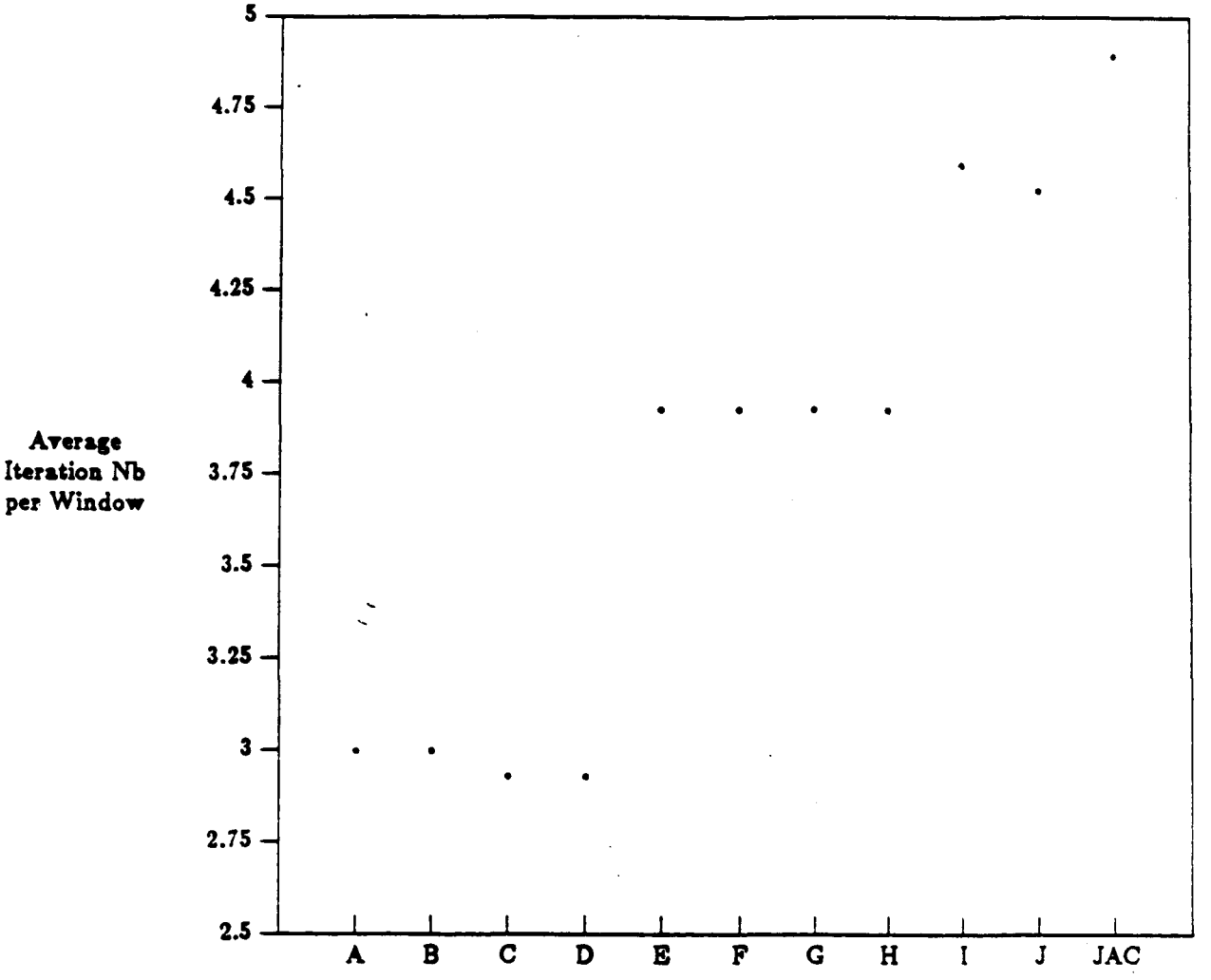
After comparing the efficiency among different numbering schemes, we then test the effects of window length on the overall performance of waveform relaxation. We list the statistics in Table 6.2.1. Since the computing time is roughly proportional to the total number of integration steps, from Table 6.2.1 we notice that for Jacobi scheme the most efficient ratio of window length to the maximum step size is 4, while 16 is the most efficient ratio for the Gauss-Seidel scheme under the numbering, (2 3 0 1). From the tables we also see that we can reduce the computing time from 16% to 39% by exploiting the partial waveform convergence .

Over all, we can see that the numbering (2 3 0 1) in Gauss-Seidel gives the best performance. This is not totally unexpected. The reason is that the constant extrapolation over a window is needed only when the fast component (2) is first integrated and we can expect small extrapolation error when the extrapolation work is performed on the slow component (1).

□

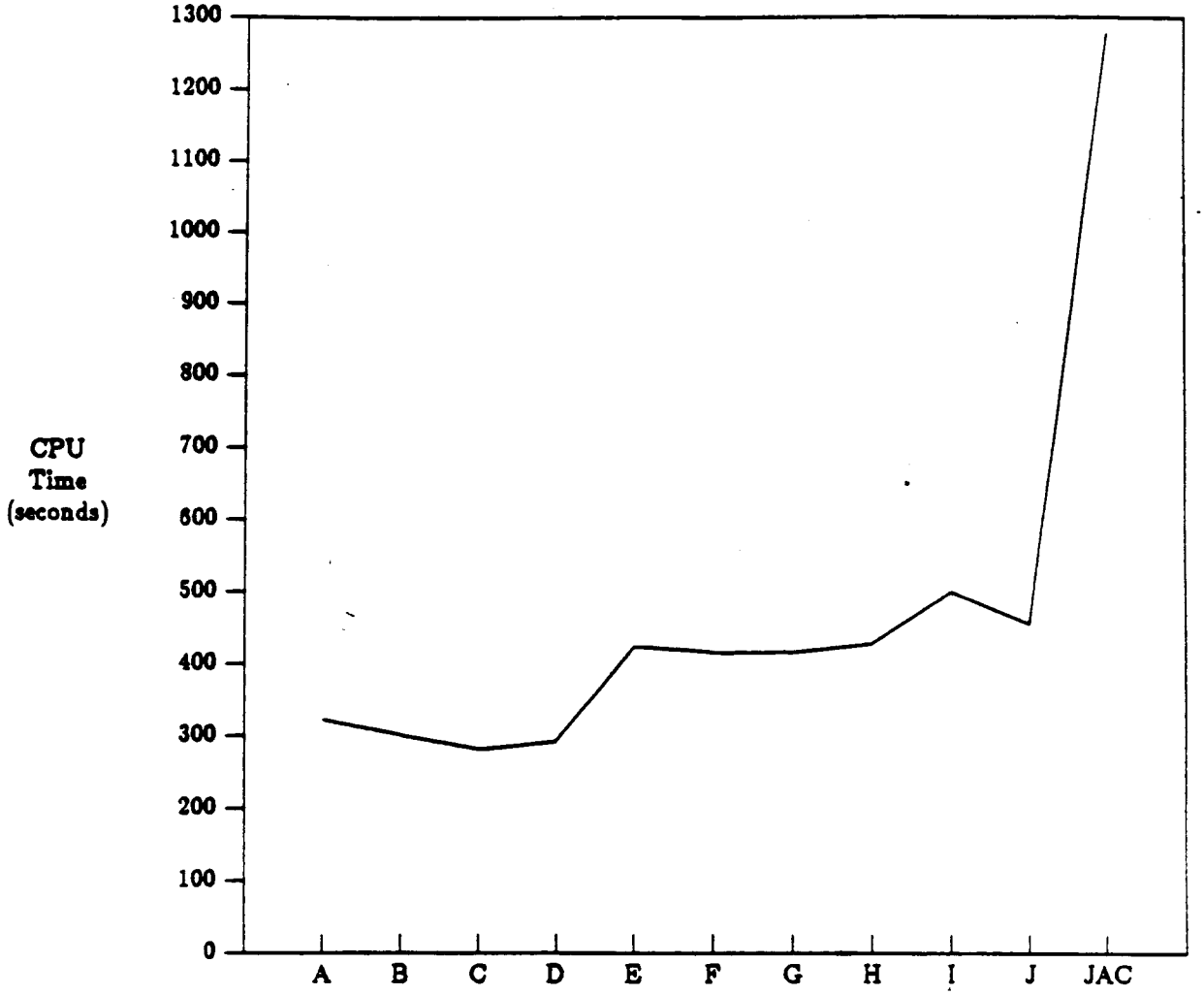
(6.1/4) (2)(3)(0)(1)	window length / max step size					
Gauss-Seidel	2	4	8	16	32	64
CPU time (second)	367.42	279.24	244.88	219.84	266.76	285.16
total iterations	348	101	44	24	16	9
window numbers	123	34	15	8	5	3
iteration/window	2.829	2.970	2.933	3	3.2	3
steps performed	17344	12936	10753	9033	10119	9690
non-converged steps	12367	8709	7297	6192	6154	7171
steps redundant (%)	28.69	29.74	32.14	31.45	39.18	25.99
interpolations	9841	8995	7820	5973	7947	7818
function evaluations	39084	28726	23699	19702	21941	21186
(6.1/4) (2)(3)(0)(1)	window length / max step size					
Jacobi	2	4	8	16		
CPU time (second)	638.42	599.22	1114.12	2321.74		
total iterations	491	169/179	142/237	139/354		
window numbers	123	35	29	28		
iteration/window	3.992	4.829	4.897	4.964		
steps performed	29298	26909	48999	99030		
non-converged steps	20250	19672	38959	82869		
steps redundant (%)	30.88	26.89	20.49	16.32		
interpolations	14914	17240	38958	65828		
function evaluations	67470	60357	108778	217996		

Table 6.1: Table 1 of Example 6.1



Example : 6.1
Gauss–Seidel (A – J) ¦ Jacobi (JAC)
A : 0 1 2 3 (4/1)
B : 3 0 1 2 (4/1)
C : 2 3 0 1 (4/1)
D : 1 2 3 0 (4/1)
E : 0 1 3 2 (4/2)
F : 0 2 1 3 (4/2)
G : 0 2 3 1 (4/2)
H : 0 3 1 2 (4/2)
I : 0 3 2 1 (4/3)
J : 3 2 1 0 (4/3)

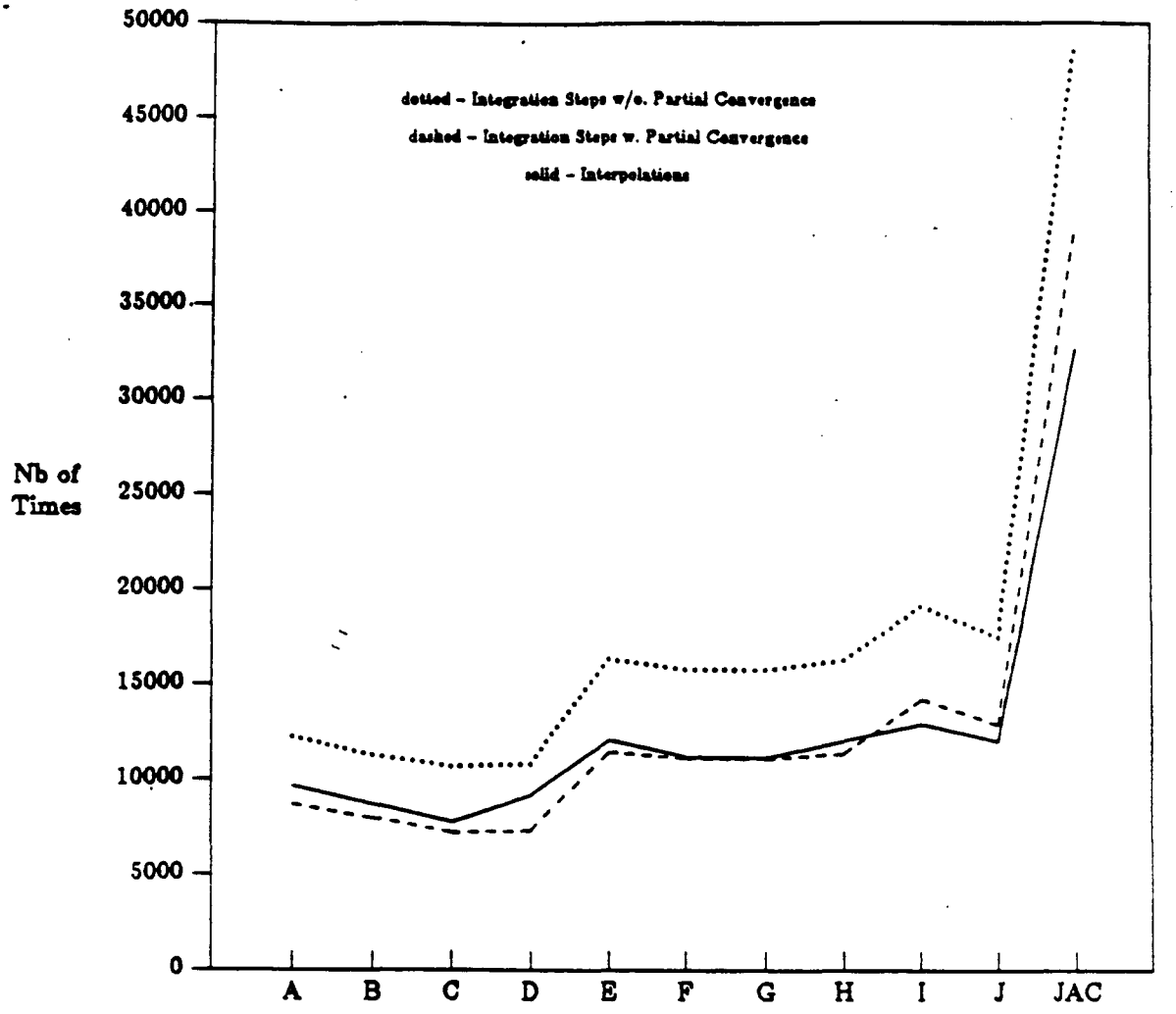
Figure 1 of Example 6.1



Window Length / Max Step Size = 8
Ordering Effects on Speed of Convergence

Example : 6.1
Gauss–Seidel (A – J) ¦ Jacobi (JAC)
A : 0 1 2 3 (4/1)
B : 3 0 1 2 (4/1)
C : 2 3 0 1 (4/1)
D : 1 2 3 0 (4/1)
E : 0 1 3 2 (4/2)
F : 0 2 1 3 (4/2)
G : 0 2 3 1 (4/2)
H : 0 3 1 2 (4/2)
I : 0 3 2 1 (4/3)
J : 3 2 1 0 (4/3)

Figure 2 of Example 6.1



Window Length / Max Step Size = 8
Ordering Effects on Speed of Convergence

Example : 4c
Gauss-Seidel (A – J) ¦ Jacobi (JAC)
A : 0 1 2 3 (4/1)
B : 3 0 1 2 (4/1)
C : 2 3 0 1 (4/1)
D : 1 2 3 0 (4/1)
E : 0 1 3 2 (4/2)
F : 0 2 1 3 (4/2)
G : 0 2 3 1 (4/2)
H : 0 3 1 2 (4/2)
I : 0 3 2 1 (4/3)
J : 3 2 1 0 (4/3)

Figure 3 of Example 6.1

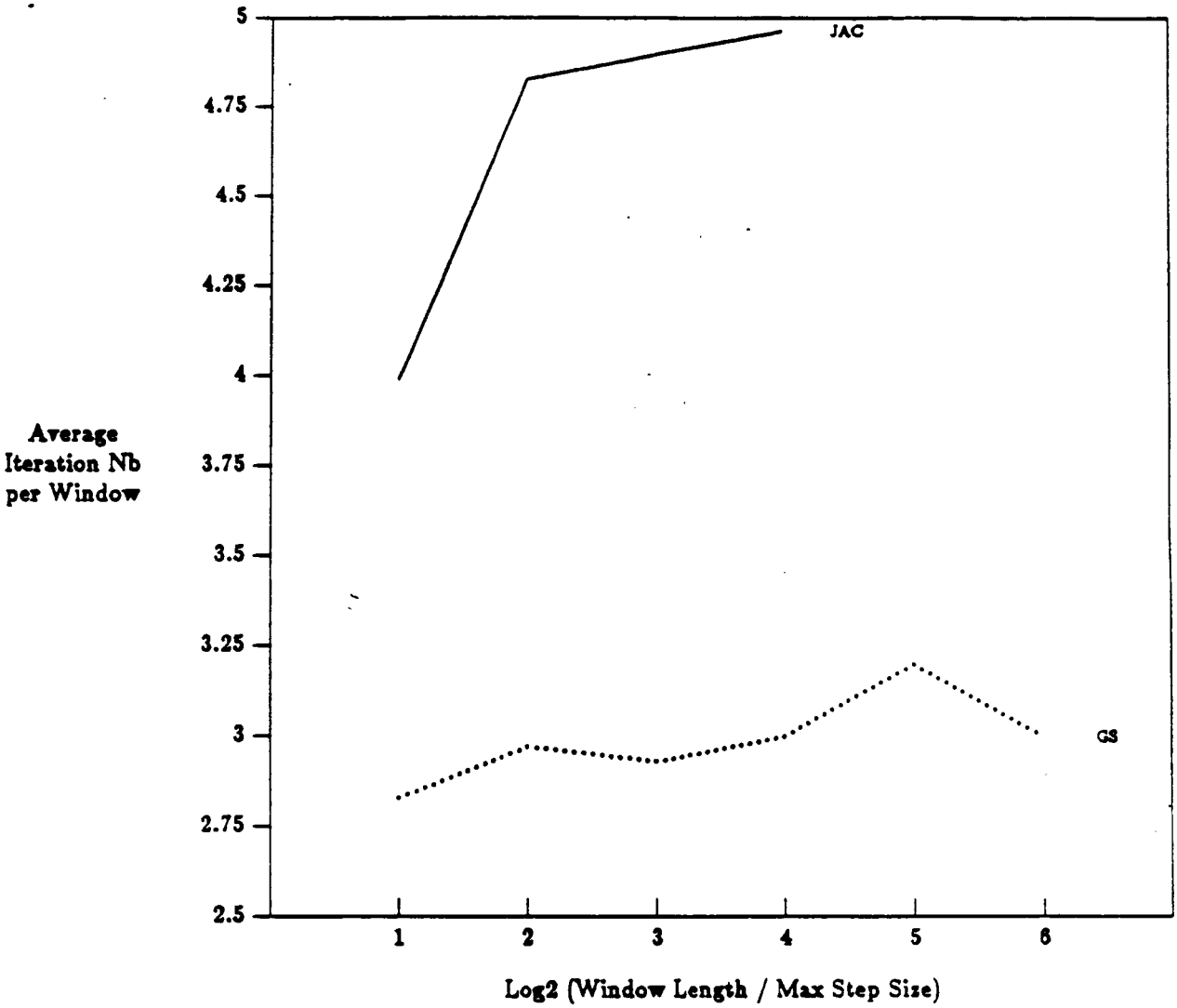


Figure 4 of Example 6.1

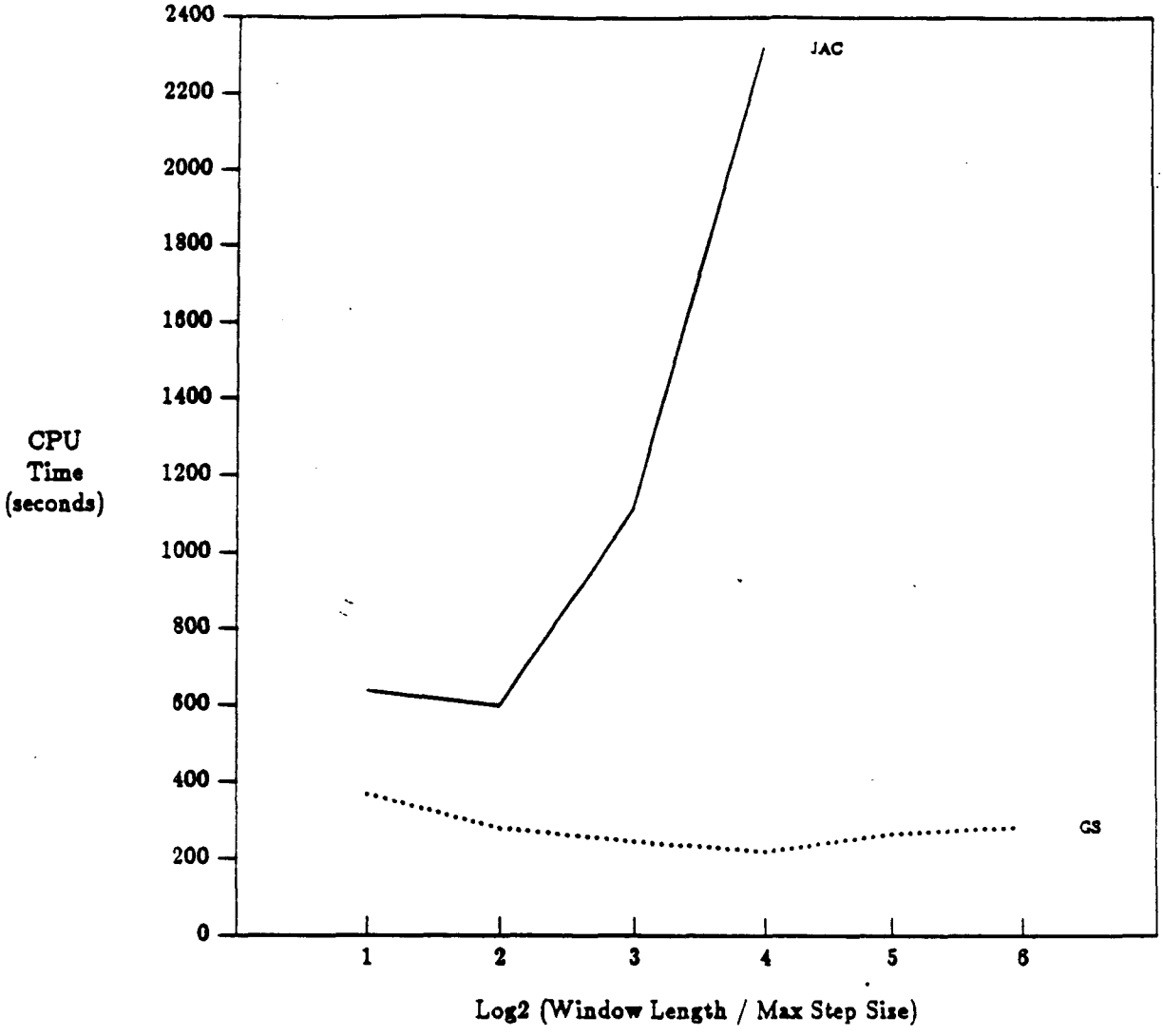


Figure 5 of Example 6.1

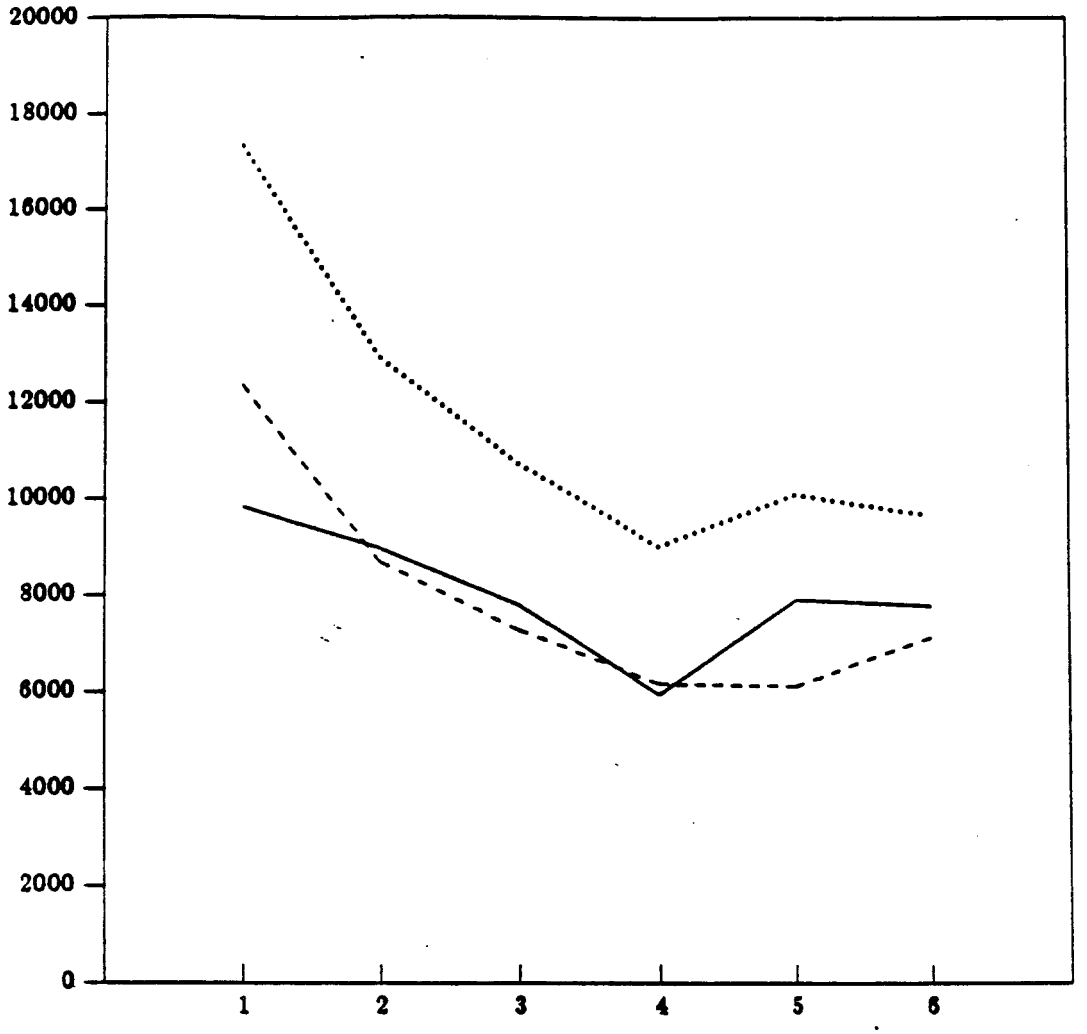
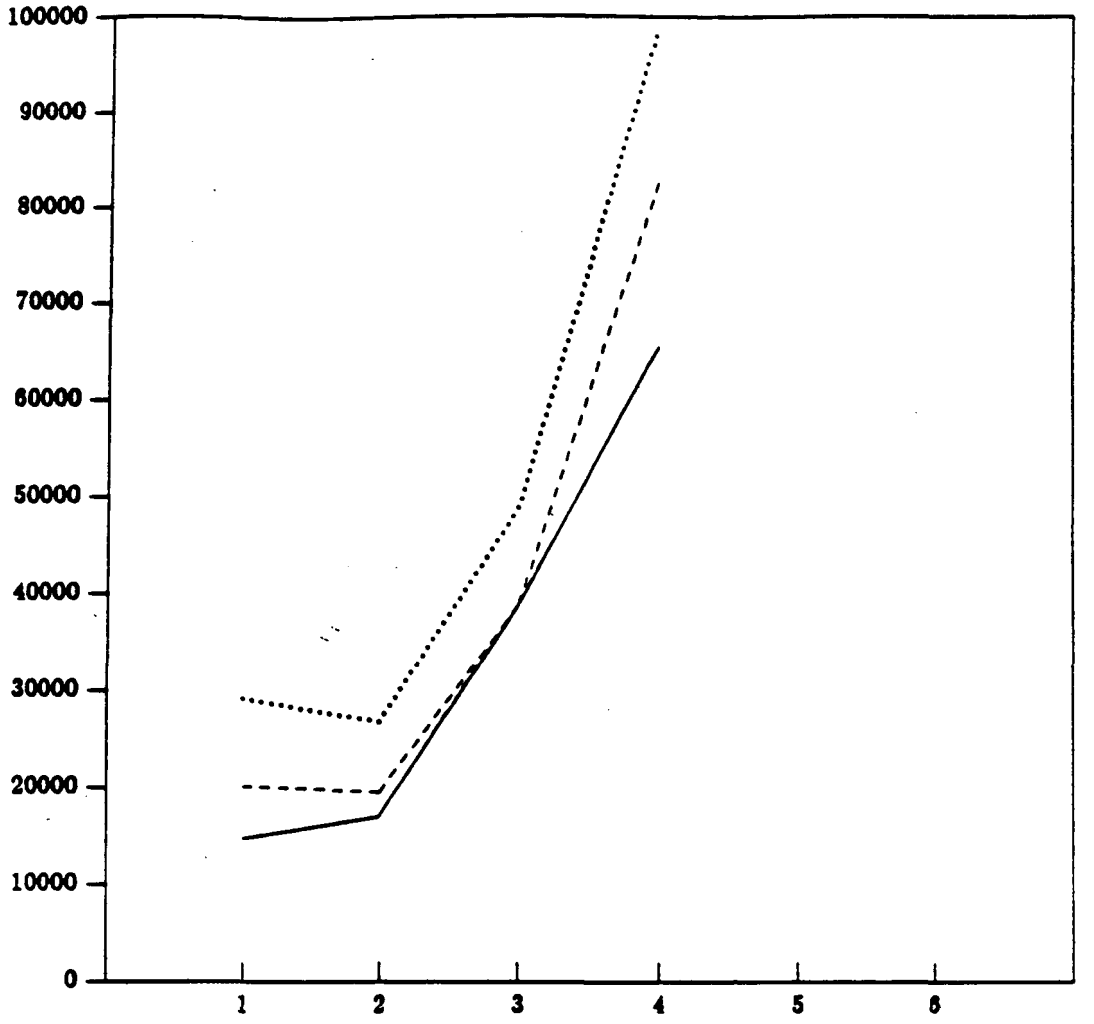


Figure 6 of Example 6.1



Example : 6.1

JAC : (0)(1)(2)(3)

dotted - # of Integration Steps w/o. Partial Convergence
dashed - # of Integration Steps w. Partial Convergence
solid - # of Interpolations

Figure 7 of Example 6.1

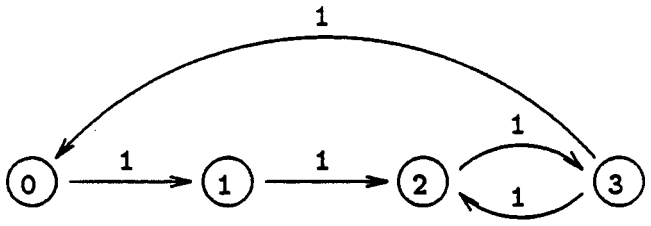


Figure 6.2: Dependency Graph of Example 6.2

Example 6.2 : In this example, we consider

$$A = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -5 & 0 & 0 \\ 0 & 1 & -10 & 1 \\ 0 & 0 & 1 & -20 \end{bmatrix}$$

and

$$\phi(t) = (cos(t), sin(t), cos(20t), sin(20t))^T.$$

This matrix differs in one position from that of Example 6.1. By changing the value at position (3,4) from 0 to 1, we construct a dependency graph with a completely different structure.

The dependency graph has two cycles of length four and two, respectively. (See Fig 6.2) In this example, we test the effects of partitioning on the speed of convergence, and observe that proper regrouping indeed improves the speed of convergence. If the system is partitioned into four subsystems then, by Theorem 4.7, the average accuracy increase is two; if the system is partitioned into three subsystems, i.e. equation 2 and equation 3 which are mutually coupled are grouped as a subsystem, then the average accuracy increase becomes three and hence the resulting Gauss-Seidel scheme converges faster. The comparisons are shown in the following graphs and tables. Theoretically, we expect that the second partitioning strategy provides a better performance because of the larger increase in order of accuracy. But from the statistics shown in Table 6.2, we can not be so sure about this in terms of processing time consumed. By simultaneously solving two variables differ only in phase shift, we may reduce the total number of interpolations, but these savings might be offset by the increase of the total number of integration steps and the number of function evaluations resulting from choosing smaller stepsizes.

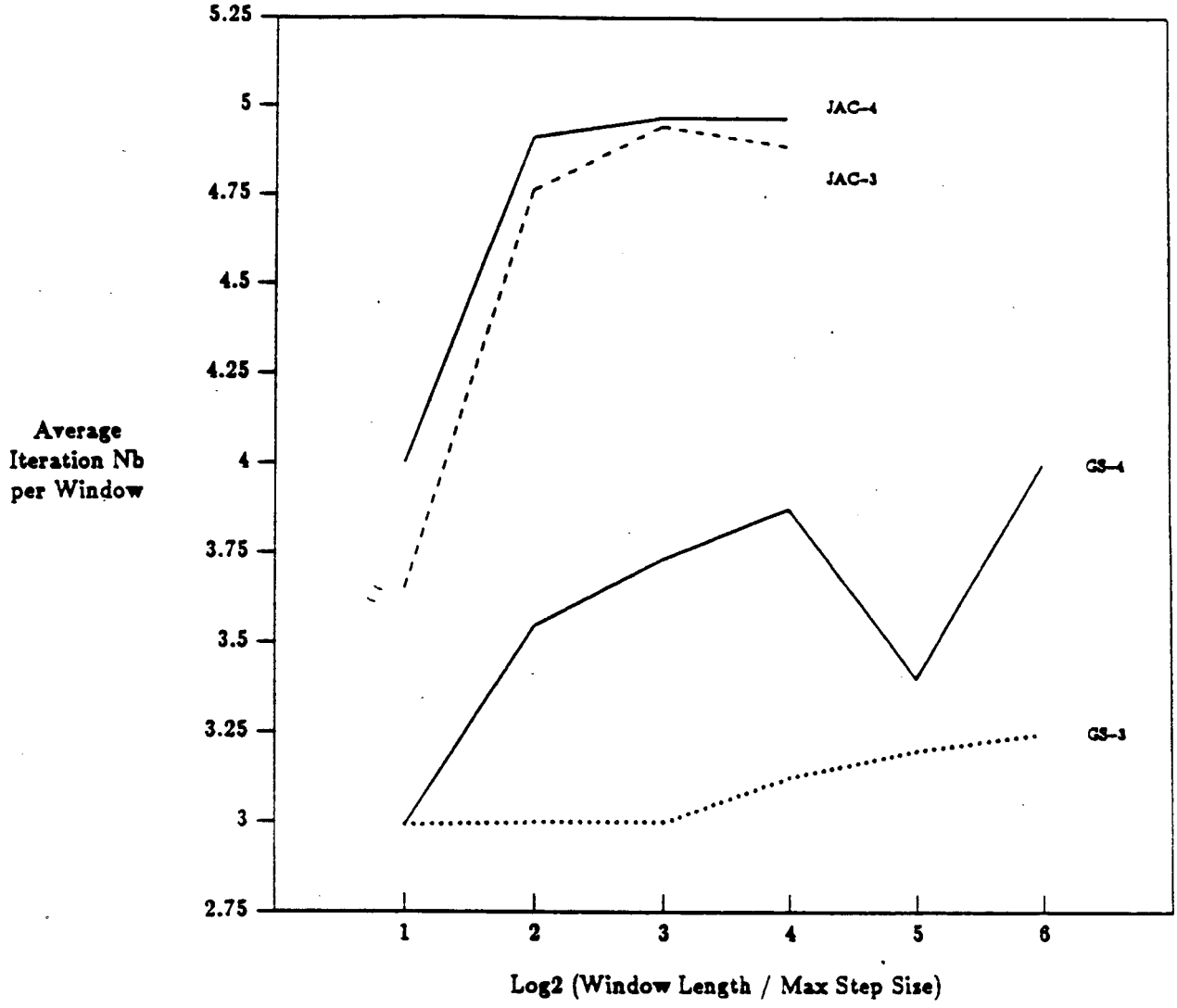
(6.2/3) (0)(1)(23)	window length / max step size					
Gauss-Seidel	2	4	8	16	32	64
total iterations	368	102	51	25	16	13
window numbers	123	34	17	8	5	4
iteration/window	2.992	3	3	3.125	3.2	3.25
steps performed	21504	16467	15354	15931	17190	17393
non-converged steps	15712	11673	10826	10481	10597	10725
steps redundant (%)	26.93	29.11	29.49	34.21	38.35	38.34
interpolations	9371	7778	7470	7737	8205	8200
function evaluations	51755	37964	34953	35953	38714	39165

(6.2/3) (0)(1)(23)	window length / max step size					
Jacobi	2	4	8	16		
total iterations	449	162	89	44/54		
window numbers	123	34	17	9		
iteration/window	3.650	4.765	4.944	4.889		
steps performed	26334	25758	24726	35021		
non-converged steps	19696	19088	18761	28799		
steps redundant (%)	25.20	25.89	24.12	17.77		
interpolations	8620	10139	9987	14304		
function evaluations	63295	59327	56331	79045		

(6.2/4) (0)(1)(2)(3)	window length / max step size					
Gauss-Seidel	2	4	8	16	32	64
total iterations	368	117	56	31	17	12
window numbers	123	33	15	8	5	3
iteration/window	2.992	3.545	3.733	3.875	3.4	4
steps performed	21724	17649	15196	13617	12215	14381
non-converged steps	16130	12325	10855	9876	9438	10799
steps redundant (%)	25.75	30.16	28.56	27.47	22.73	24.90
interpolations	16631	17762	15542	12024	13007	17369
function evaluations	50003	39538	33594	29725	26493	31484

(6.2/4) (0)(1)(2)(3)	window length / max step size					
Jacobi	2	4	8	16		
total iterations	492	172/182	159/304	154/409		
window numbers	123	35	32	31		
iteration/window	4	4.914	4.968	4.968		
steps performed	29399	27133	64824	112071		
non-converged steps	23170	21597	54791	98962		
steps redundant (%)	21.18	20.40	15.48	11.69		
interpolations	18869	24404	60220	104055		
function evaluations	67761	60931	143823	246389		

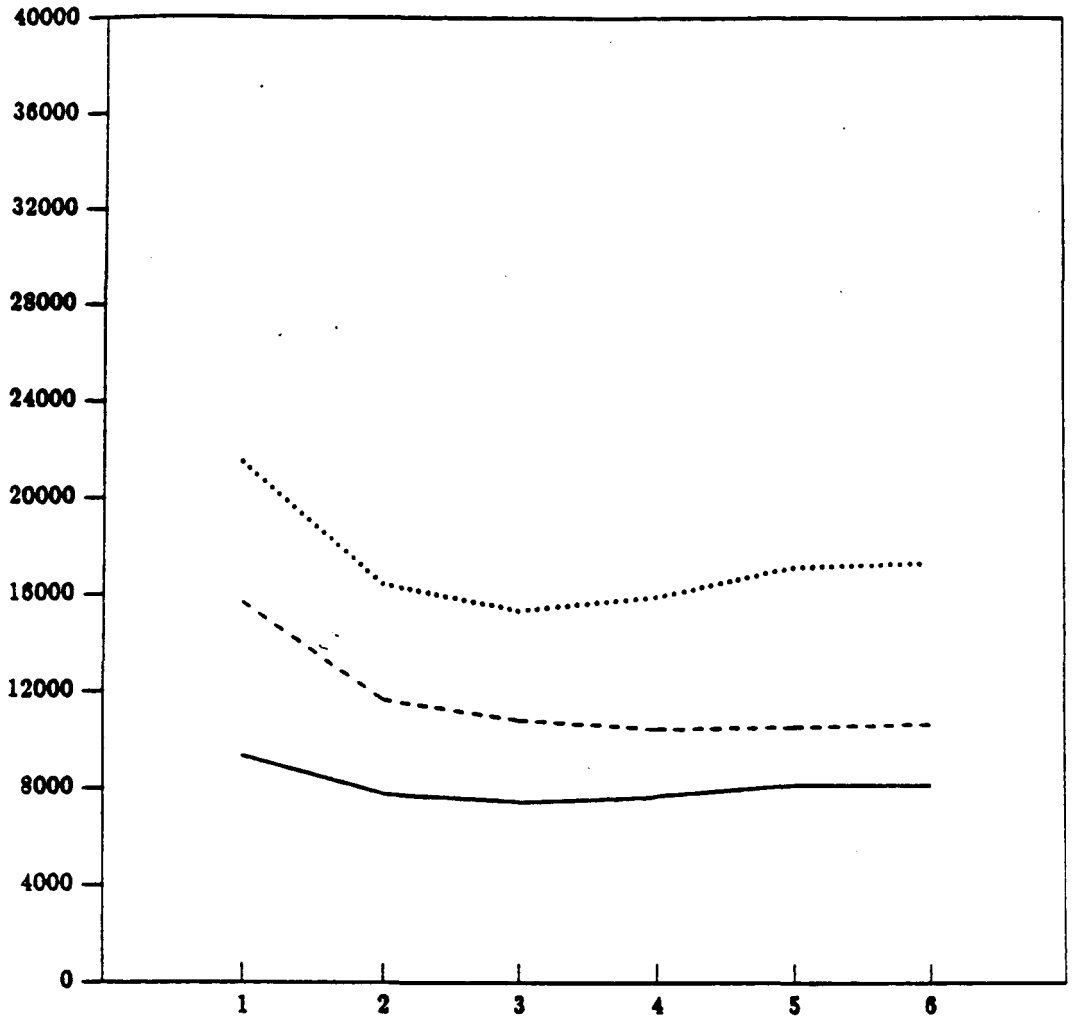
Table 6.2: Some Statistics of Example 6.2



Example : 6.2

GS–3, JAC–3 : (0)(1)(23)
GS–4, JAC–4 : (0)(1)(2)(3)

Figure 1 of Example 6.2

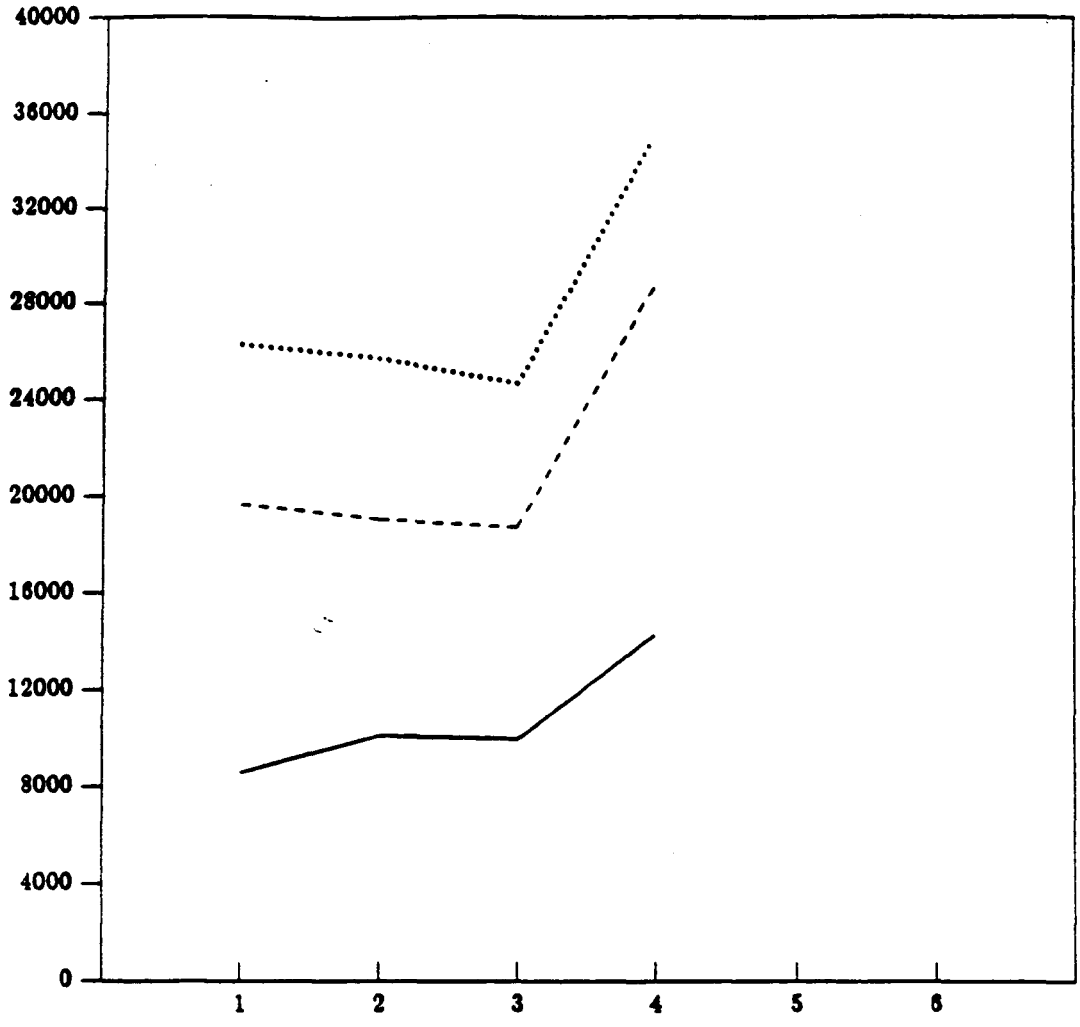


Example : 6.2

GS–3 : (0)(1)(23)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 2 of Example 6.2

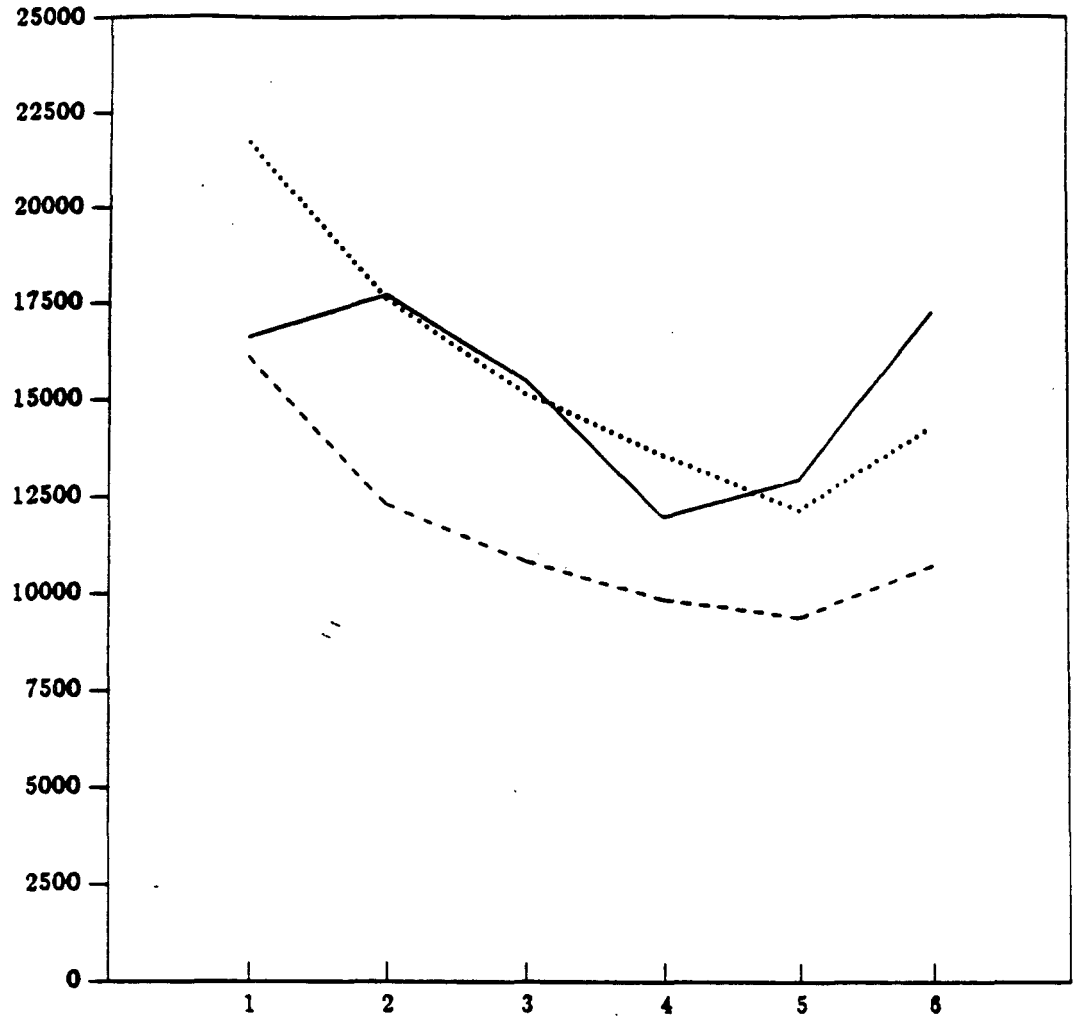


Example : 6.2

JAC-3 : (0)(1)(23)

dotted - # of Integration Steps w/o. Partial Convergence
dashed - # of Integration Steps w. Partial Convergence
solid - # of Interpolations

Figure 3 of Example 6.2

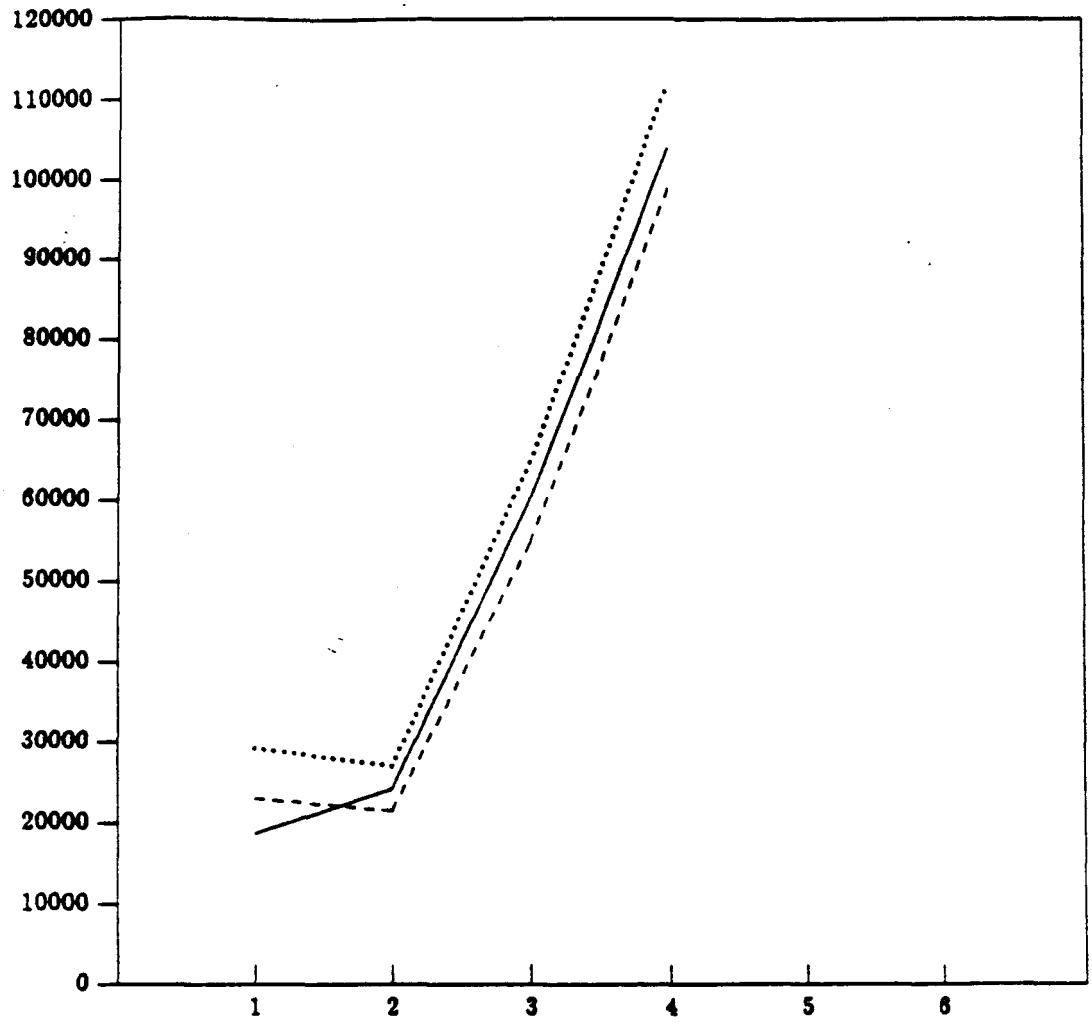


Example : 6.2

GS–4 : (0)(1)(2)(3)

dotted - # of Integration Steps w/o. Partial Convergence
dashed - # of Integration Steps w. Partial Convergence
solid - # of Interpolations

Figure 4 of Example 6.2



Example : 6.2

JAC–4 : (0)(1)(2)(3)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 5 of Example 6.2

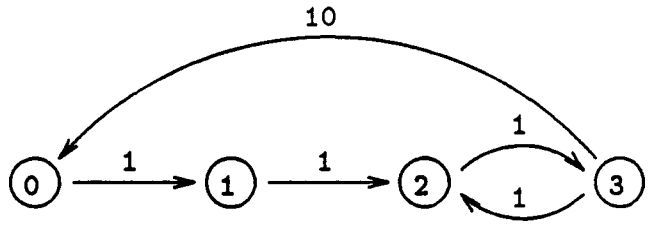


Figure 6.3: Dependency Graph of Example 6.3

Example 6.3 : In this example, we consider

$$A = \begin{bmatrix} -1 & 0 & 0 & 10 \\ 1 & -5 & 0 & 0 \\ 0 & 1 & -10 & 1 \\ 0 & 0 & 1 & -20 \end{bmatrix}$$

and

$$\phi(t) = (cos(t), sin(t), cos(20t), sin(20t))^T.$$

The system's Jacobian matrix has the same zero-nonzero structure as that of the Example 6.2 does, but the value at position (1,4) has been changed from 1 to 10.

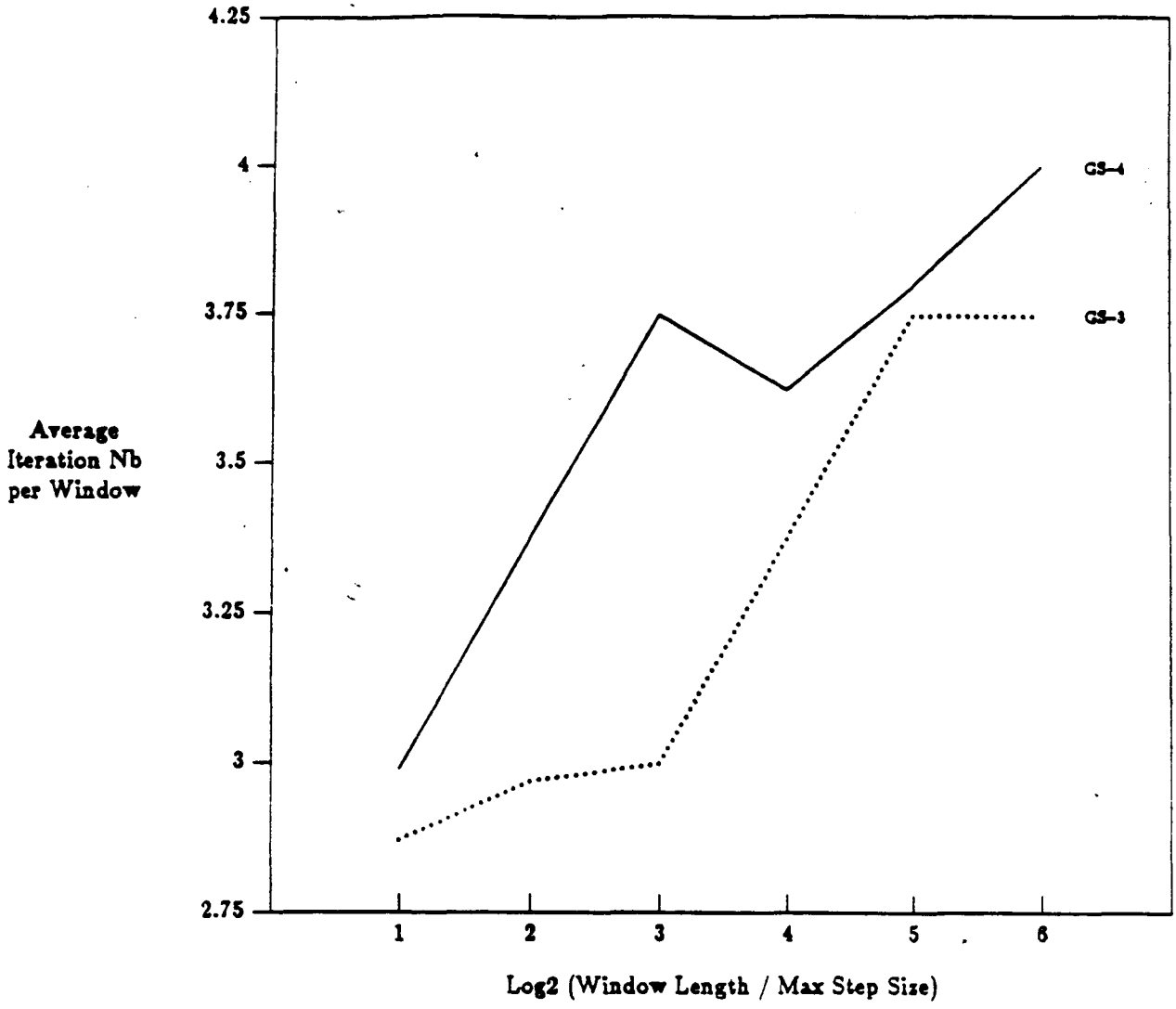
These two examples have the same dependency graphs but differ at one coupling factor. (See Fig 6.3) This example is designed to show how the size of coupling between the fast and slow components affects the performance of WRODE. From the statistical results, we see that the numbering of nodes is very sensitive to the coupling factor since some Gauss-Seidel numberings and the Jacobi scheme fail to work for this system (integration fails to converge). During the experiment, if node "0" was integrated before node "3" inside each Gauss-Seidel iteration sweep, the waveform relaxation broke down. This is because that the error at node "3" is amplified ten times after propagating to node "0". One possible solution for monitoring the propagation of interpolation (extrapolation) error from one component to another is that we may use a tighter error control at the component whose error will be amplified.

We list the statistics for the numbering (3)(0)(1)(2). From the table and the graph for CPU time, we notice that the CPU time may be a function of the sum of numbers of interpolation and function evaluation.

(6.3/3) (23)(0)(1)	window length / max step size					
Gauss-Seidel	2	4	8	16	32	64
CPU time (second)	334	258	260	297	404	421
total iterations	336	98	54	27	15	15
window numbers	117	33	18	8	4	4
iteration/window	2.872	2.970	3	3.375	3.75	3.75
steps performed	17094	14493	14303	15180	18174	18379
non-converged steps	12071	9750	9658	10289	12399	12403
steps redundant (%)	29.38	32.73	32.48	32.22	31.78	32.52
interpolations	5627	5161	5323	5824	7335	7078
function evaluations	40537	33245	32522	34290	40992	41448
(6.3/4) (3)(0)(1)(2)	window length / max step size					
Gauss-Seidel	2	4	8	16	32	64
CPU time (second)	470	392	381	352	339	368
total iterations	347	108	60	29	19	16
window numbers	116	32	16	8	5	4
iteration/window	2.991	3.375	3.75	3.625	3.8	4
steps performed	19884	16074	15348	13340	12346	12606
non-converged steps	14365	11386	10819	9846	9194	9504
steps redundant (%)	27.76	29.16	29.51	26.19	25.53	24.61
interpolations	17650	17457	16377	12933	13131	10094
function evaluations	45411	35832	33933	29088	26794	27427

Table 6.3: Some Statistics of Example 6.3

□



Example : 6.3

GS–3 : (23)(0)(1)
GS–4 : (3)(0)(1)(2)

Figure 1 of Example 6.3

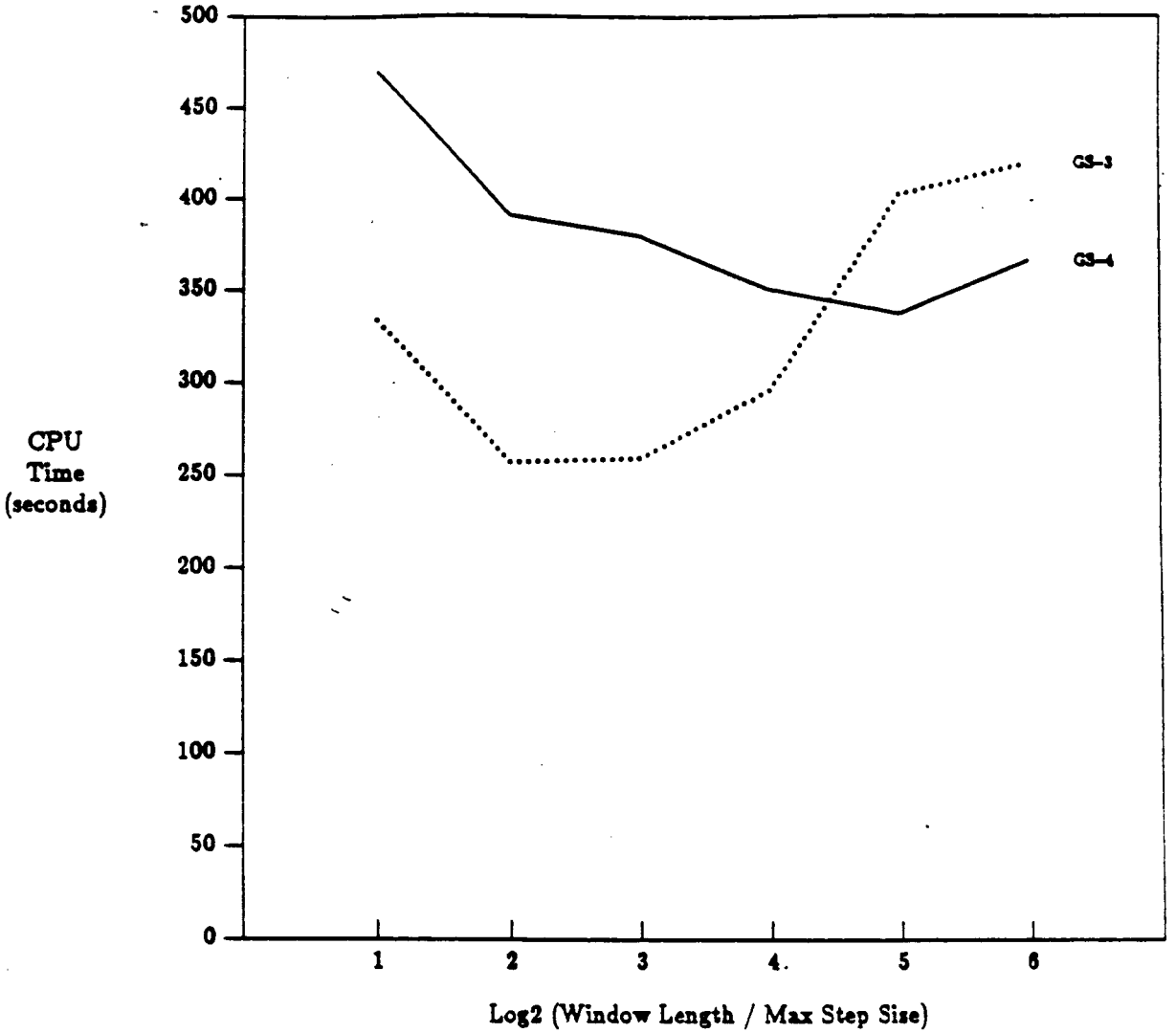
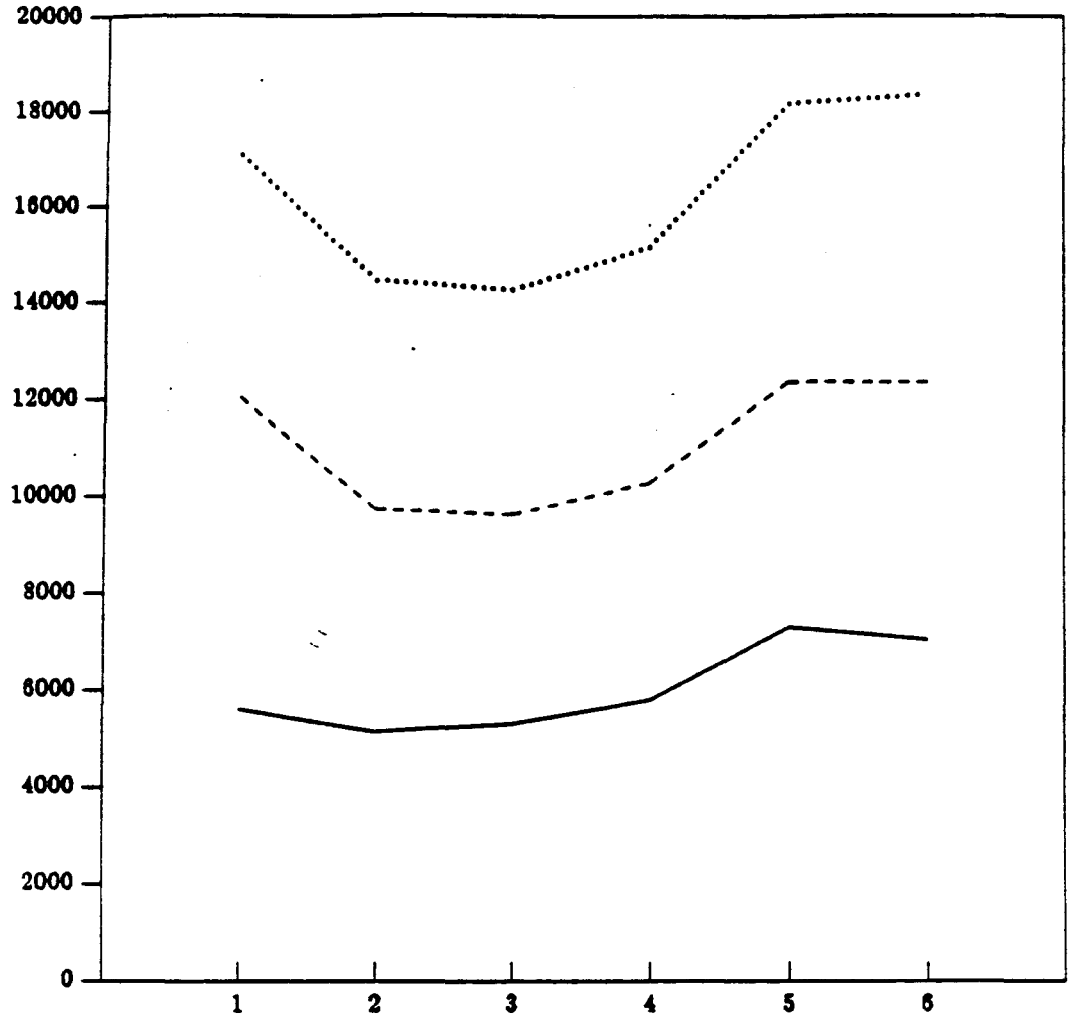


Figure 2 of Example 6.3

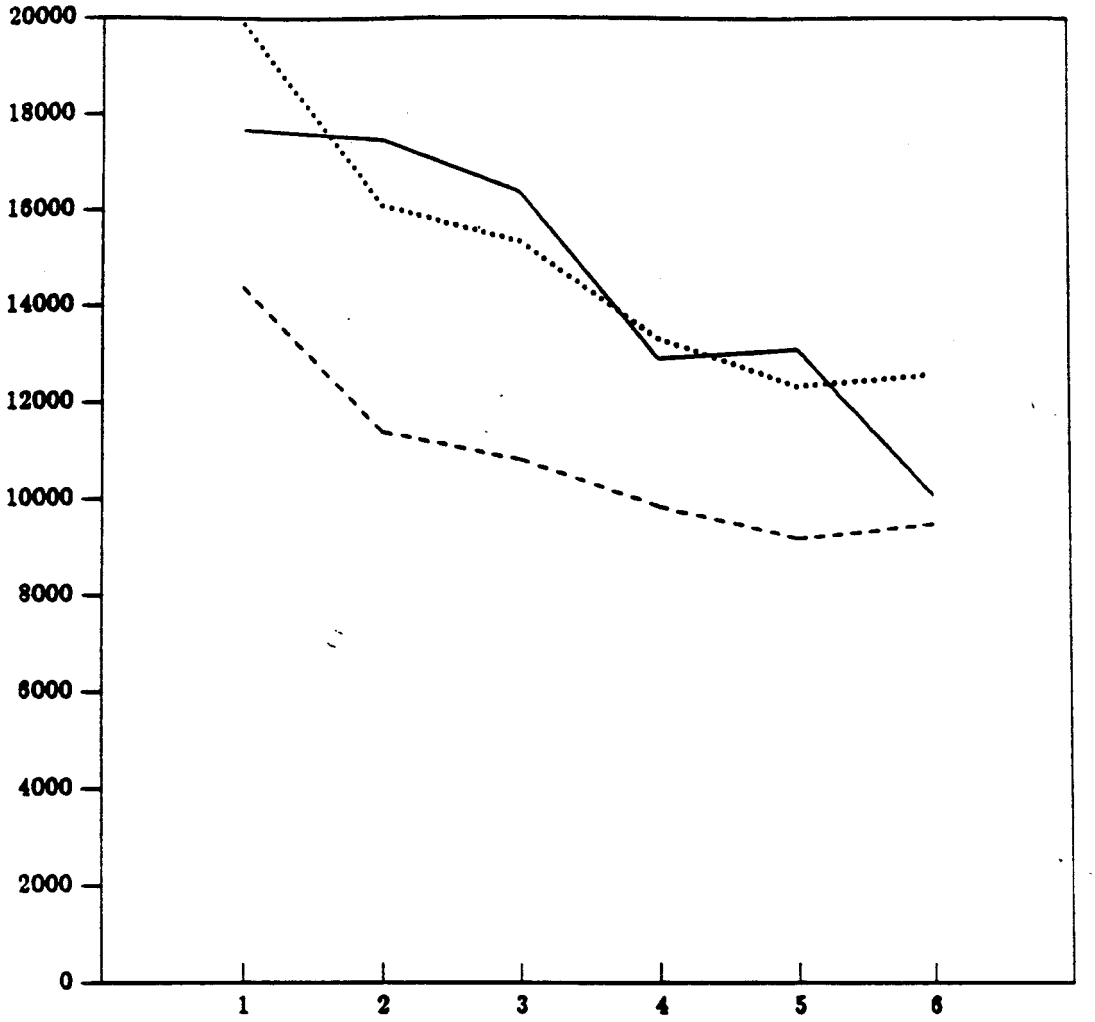


Example : 6.3

GS–3 : (23)(0)(1)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 3 of Example 6.3



Example : 6.3

GS–4 : (3)(0)(1)(2)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 4 of Example 6.3

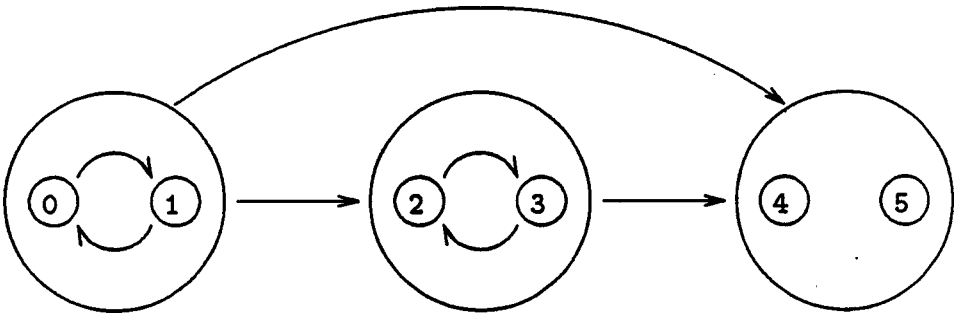


Figure 6.4: Dependency Graph of Example 6.4

Example 6.4 : In this example, we consider

$$A = \begin{bmatrix} 49 & -50 & 0 & 0 & 0 & 0 \\ -50 & 49 & 0 & 0 & 0 & 0 \\ 1 & 1 & -6 & 5 & 0 & 0 \\ 1 & 1 & 5 & -6 & 0 & 0 \\ 1 & 1 & 1 & 1 & -1 & 0 \\ 1 & 1 & 1 & 1 & 0 & -1 \end{bmatrix}.$$

and

$$\phi(t) = (cos(0.5t), sin(0.5t), cos(t), sin(t), cos(20t), sin(20t))^T.$$

Here we have a system whose Jacobian matrix is in lower block triangular form (1-way coupling, See Fig 6.4). This experiment is designed to demonstrate that if the Jacobian matrix is a lower block triangular matrix with M diagonal blocks, Gauss-Seidel scheme converges after one iteration if each subsystem is sequentially solved following the dependence relations, the second iteration is needed to detect waveform convergence and Jacobi converges within M iterations, the last iteration is used to detect waveform convergence. In this particular example two iterations are needed for Gauss-Seidel to reach convergence and less than four iterations are needed for Jacobi. This can be seen from the field *iteration / window* in Tables 6.4 and 6.5. Some statistical results are listed in Tables 6.4 and 6.5. In the tables, (6.3/3) and (6.3/4) respectively refers to partitioning into three and four subsystems.

From the tables we can see that the average iteration numbers needed for convergence at each window are the same for both partitionings. Notice that nodes '4' and '5' are uncoupled,

(6.4/3) (01)(23)(45)	window length / max step size					
Gauss-Seidel	2	4	8	16	32	64
CPU time (second)	395	251	247	271	275	302
total iterations	460	58	32	12	8	6
window numbers	230	29	16	6	4	3
iteration/window	2	2	2	2	2	2
steps performed	16588	9664	9200	8904	8344	9476
non-converged steps	16588	9664	9200	8904	8344	9476
steps redundant (%)	0	0	0	0	0	0
interpolations	24776	16652	16012	16148	15152	17412
function evaluations	42414	22662	21382	20350	18982	21462
(6.4/3) (01)(23)(45)	window length / max step size					
Jacobi	2	4	8	16		
CPU time (second)	570	455	439	466		
total iterations	653	113	63	23		
window numbers	230	29	16	6		
iteration/window	2.839	3.897	3.9375	3.833		
steps performed	25570	19366	18464	17734		
non-converged steps	19878	13396	12942	12814		
steps redundant (%)	22.26	30.83	29.91	27.74		
interpolations	21650	24740	23901	24105		
function evaluations	65594	45562	42946	40518		

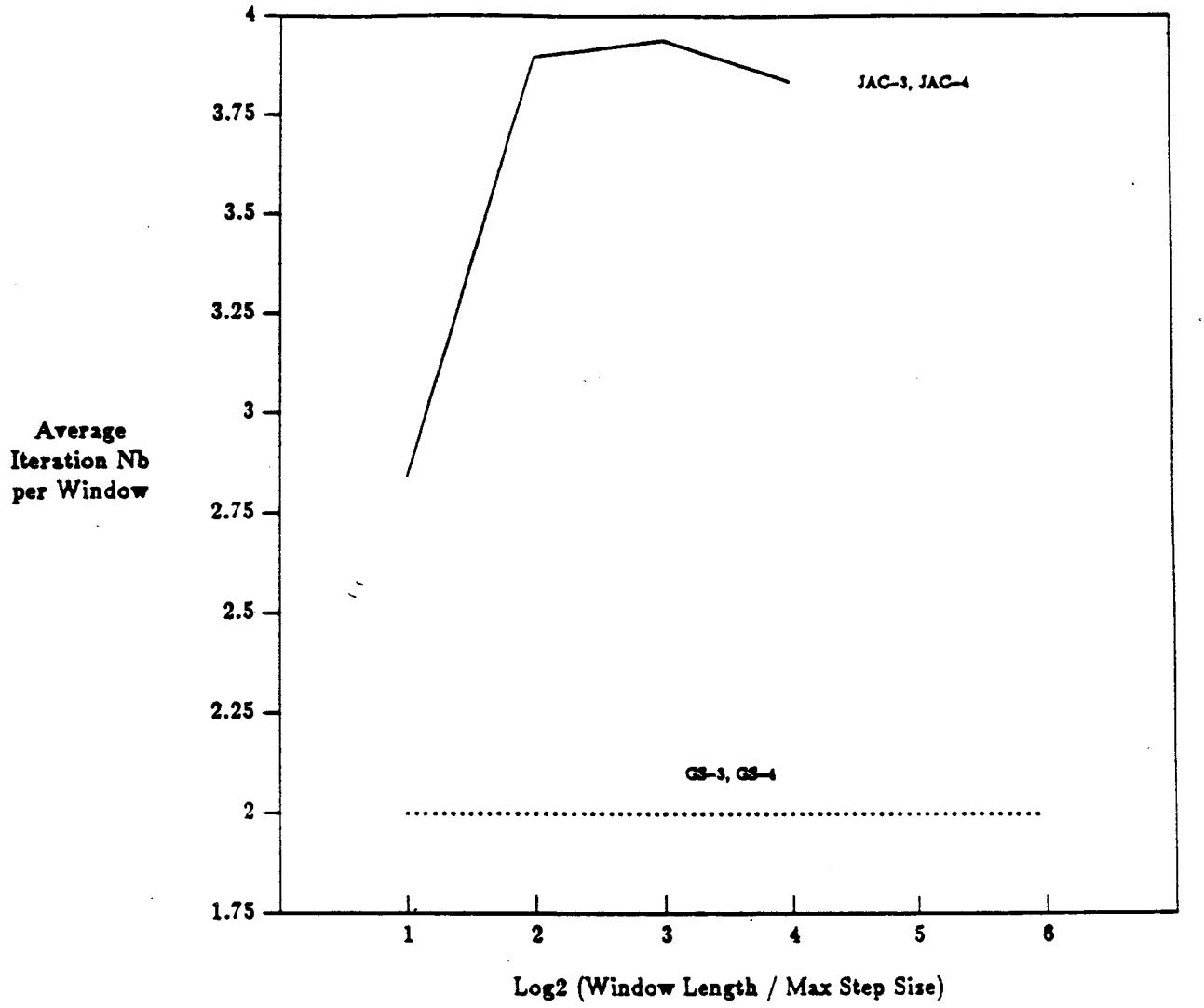
Table 6.4: Some Statistics of Example 6.4/3

they can be integrated independently at the cost of increasing the number of interpolations.

□

(6.4/4) (01)(23)(4)(5)	window length / max step size					
Gauss-Seidel	2	4	8	16	32	64
total iterations	460	58	32	12	8	6
window numbers	230	29	16	6	4	3
iteration/window	2	2	2	2	2	2
steps performed	17744	10084	9508	8858	8834	7836
non-converged steps	17744	10084	9508	8858	8834	7836
steps redundant (%)	0	0	0	0	0	0
interpolations	54760	34928	33412	32184	32440	28264
function evaluations	42892	22808	21382	19590	19494	17166
(6.4/4) (01)(23)(4)(5)	window length / max step size					
Jacobi	2	4	8	16		
total iterations	652	113	63	23		
window numbers	230	29	16	6		
iteration/window	2.834	3.897	3.9375	3.833		
steps performed	27173	20171	19030	16287		
non-converged steps	21161	13968	13354	11397		
steps redundant (%)d	22.12	30.75	29.83	30.02		
interpolations	48573	51797	49783	45738		
function evaluations	66185	45783	42844	36016		

Table 6.5: Some Statistics of Example 6.4/4



Example : 6.4

GS–3, JAC–3 : (0)(1)(23)
GS–4, JAC–4 : (0)(1)(2)(3)

Figure 1 of Example 6.4

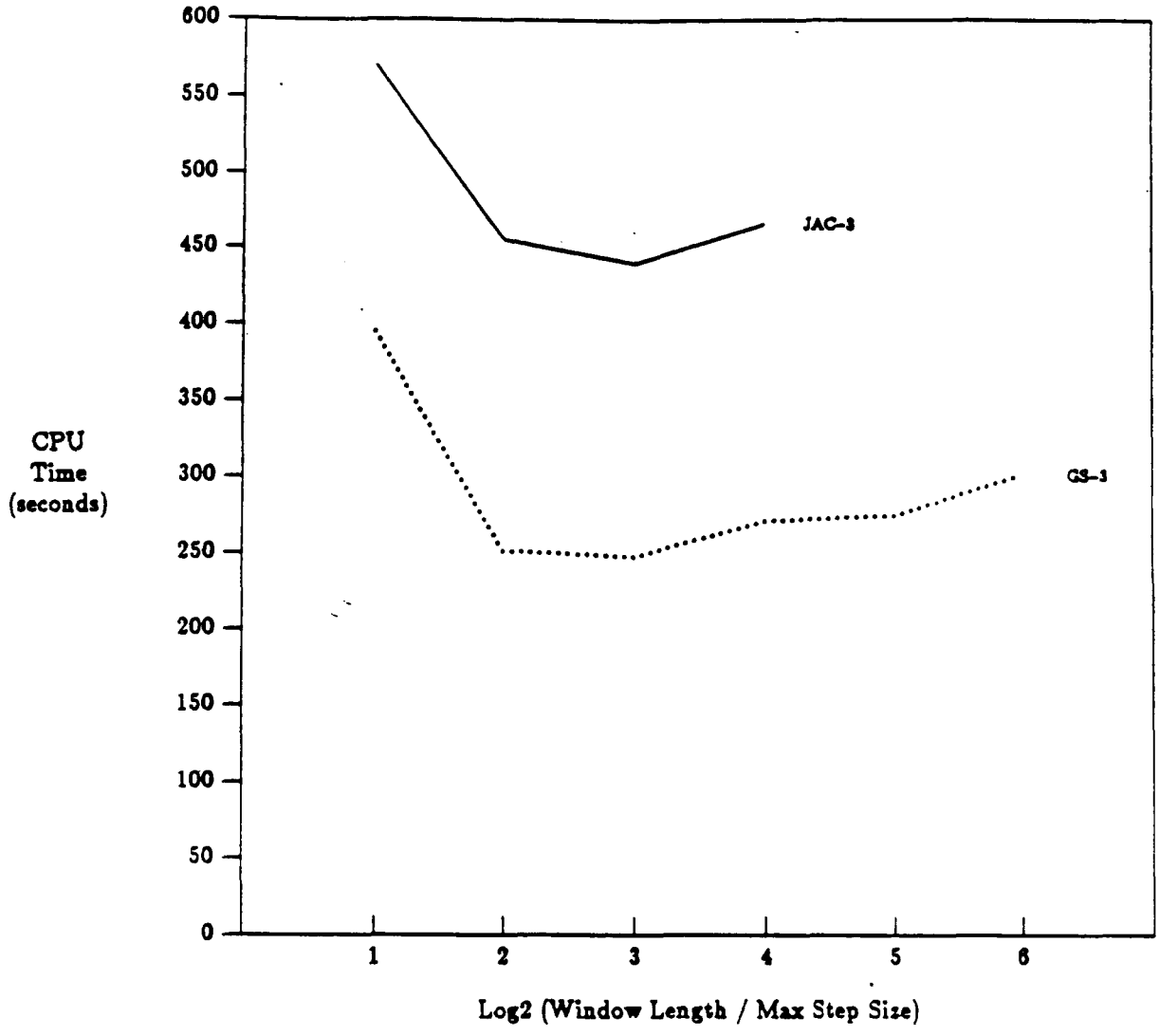
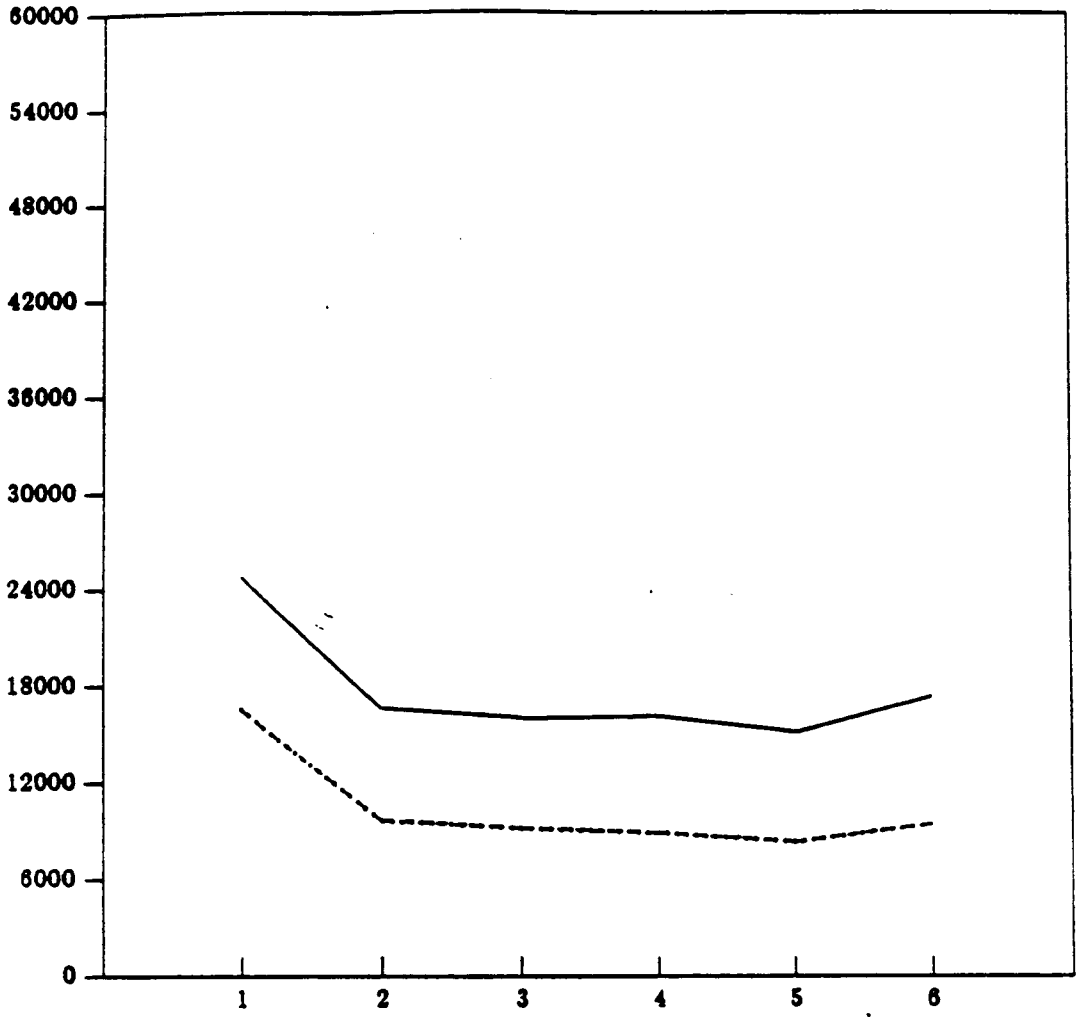


Figure 2 of Example 6.4



Example : 6.4

GS–3 : (01)(23)(45)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 3 of Example 6.4

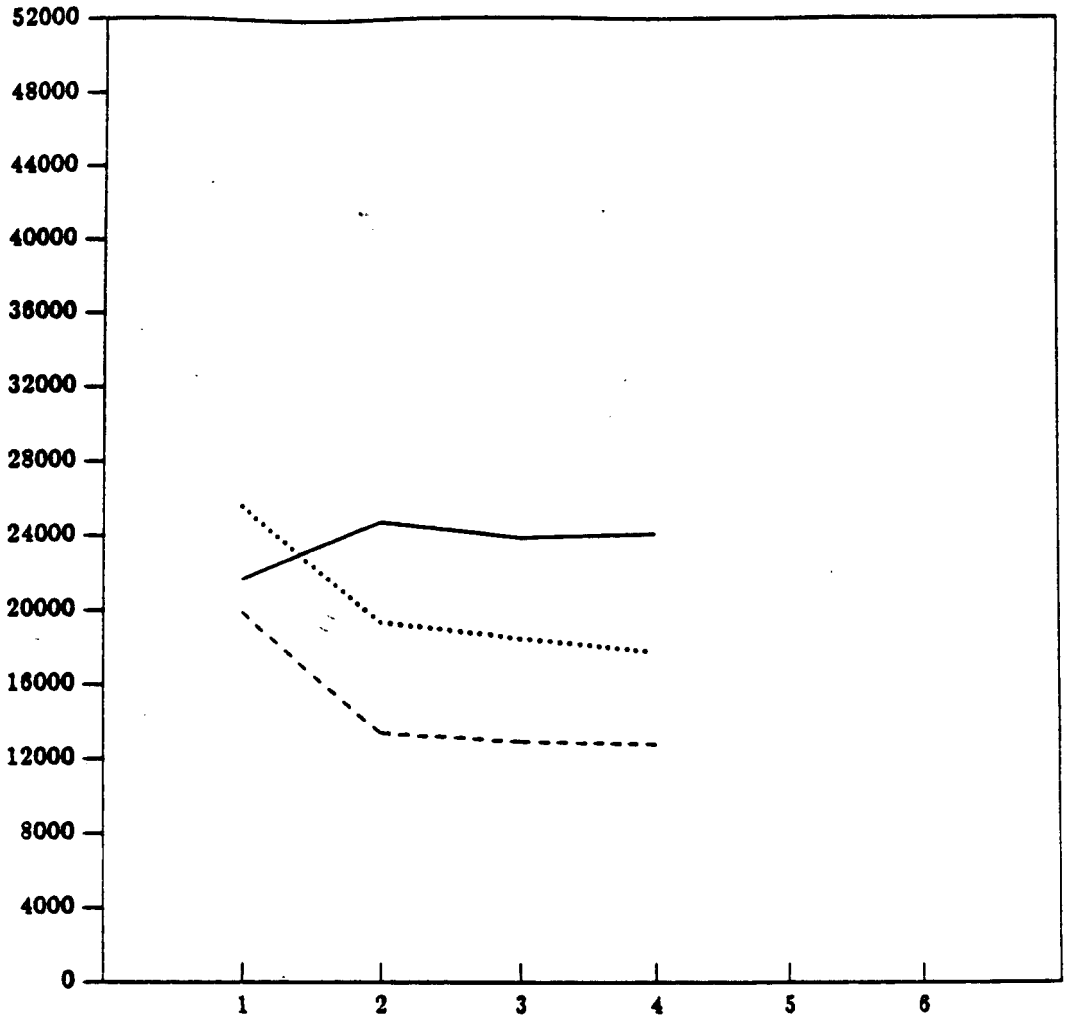
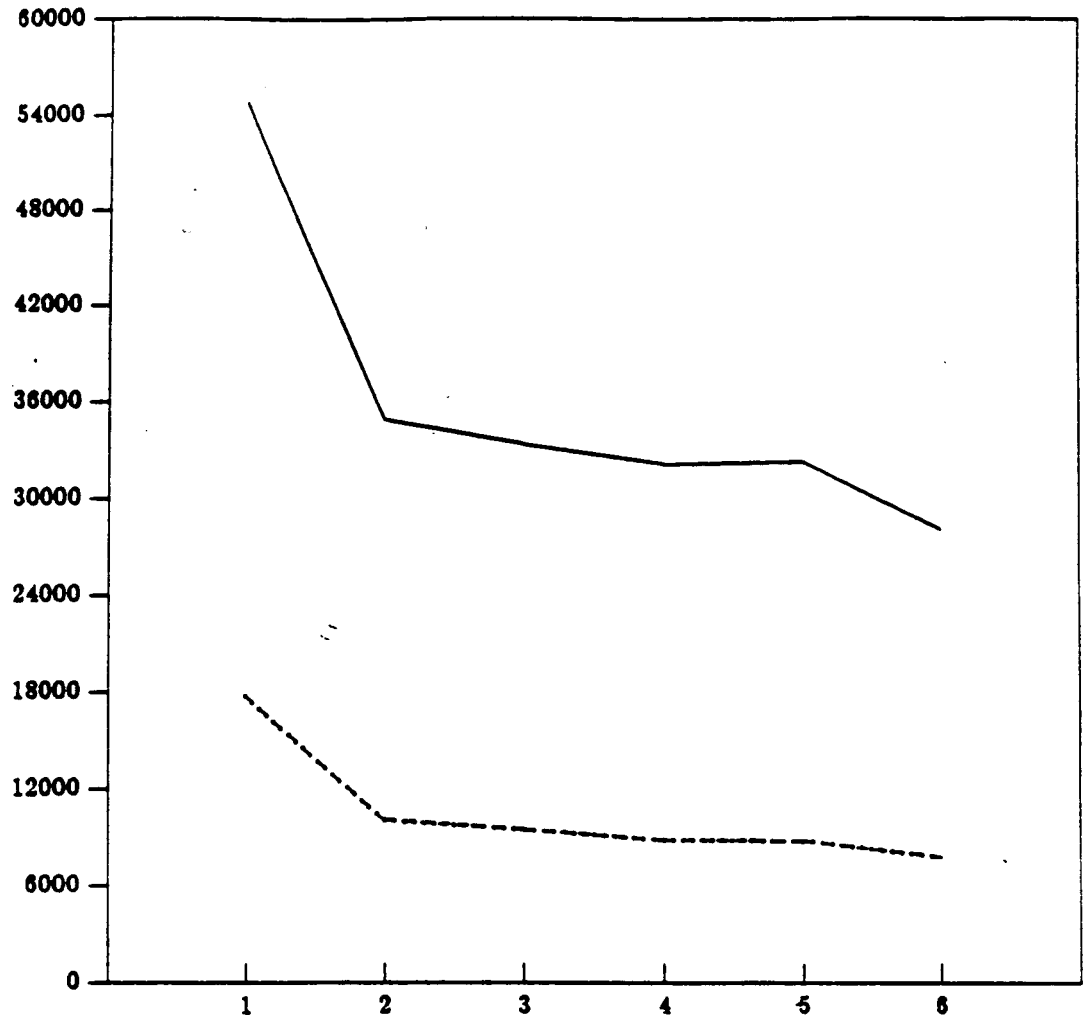


Figure 4 of Example 6.4

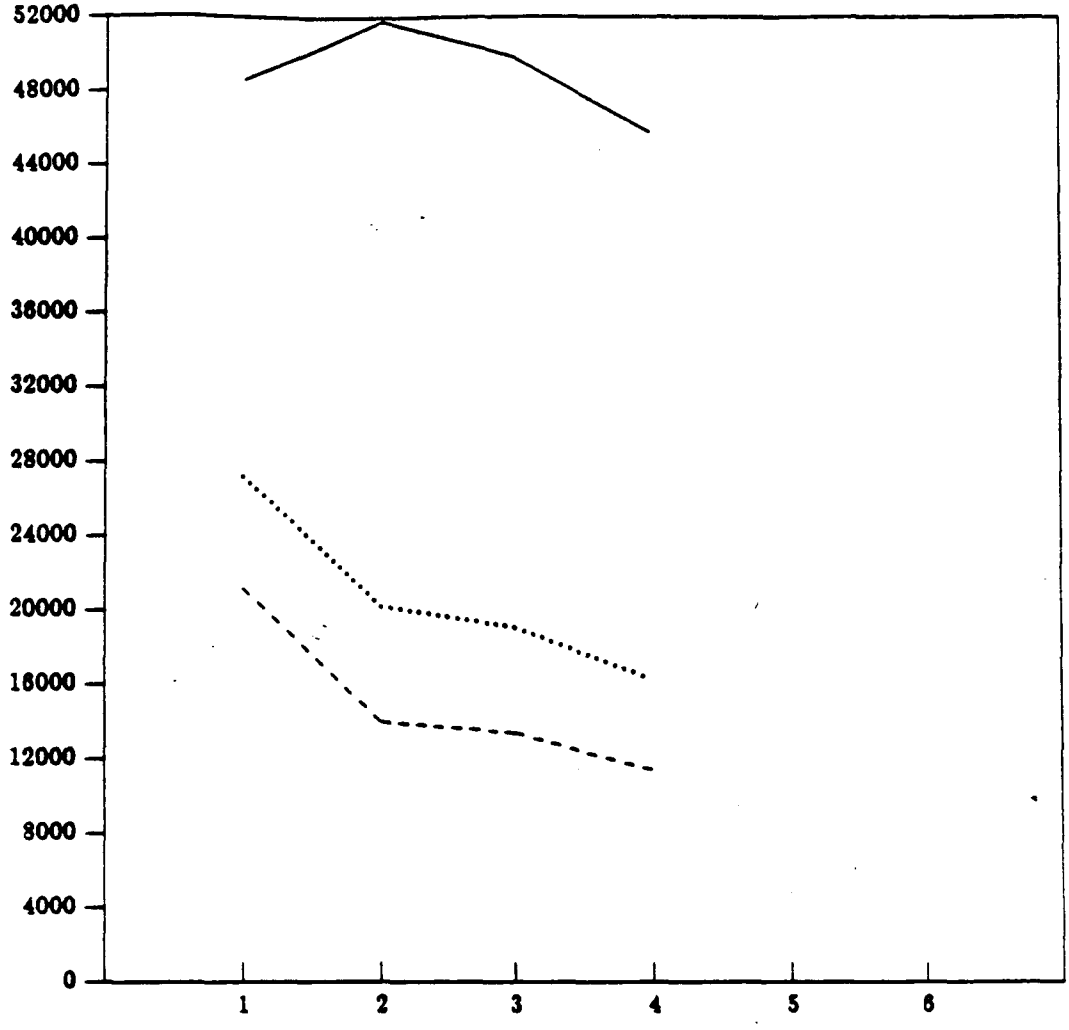


Example : 6.4

GS–4 : (01)(23)(4)(5)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 5 of Example 6.4



Example : 6.4

JAC–4 : (01)(23)(4)(5)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 6 of Example 6.4

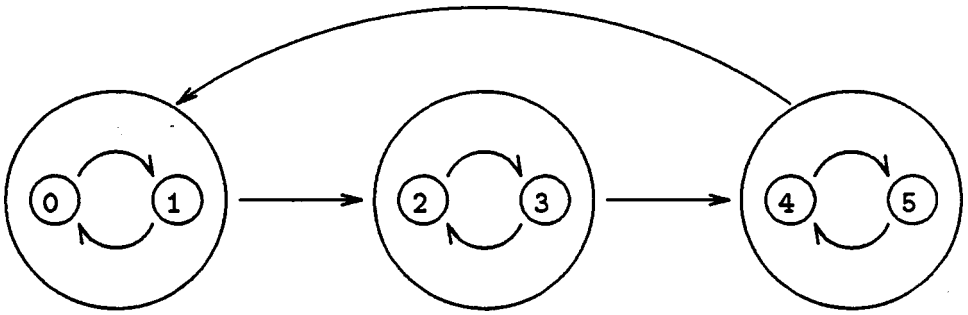


Figure 6.5: Dependency Graph of Example 6.5

Example 6.5 : In this example, we consider

$$A = \begin{bmatrix} 49 & -50 & 0 & 0 & -0.25 & 0 \\ -50 & 49 & 0 & 0 & 0 & -0.25 \\ 1 & 1 & -6 & 5 & 0 & 0 \\ 1 & 1 & 5 & -6 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & 1 \\ 0 & 0 & 1 & 1 & 1 & -1 \end{bmatrix}$$

and

$$\phi(t) = (cos(0.5t), sin(0.5t), cos(t), sin(t), cos(20t), sin(20t))^T.$$

In this example we partition the system into three subsystems and treat each subsystem as a node, thus the dependency graph has a cycle of length 3. In the experiments we consider three different numberings, (01)(23)(45), (45)(01)(23), and (45)(23)(01). Under these numberings the cycle is partitioned into one, one and two ascending chains. According to Theorem 4.7 we can expect better performance from the first two numberings. But it is difficult to compare the efficiencies of these two numberings. From the results we notice that the first numbering, (01)(23)(45), is more efficient when the ratio of window length to maximum stepsize is greater than 8 and the second numbering, (45)(01)(23), is more efficient when the ratio is less than 8.

We list the statistics in Table 6.7. From this table we also observe that the efficiency deteriorates very fast as the ratio of window length to maximum stepsize increases, especially in the Jacobi approach.

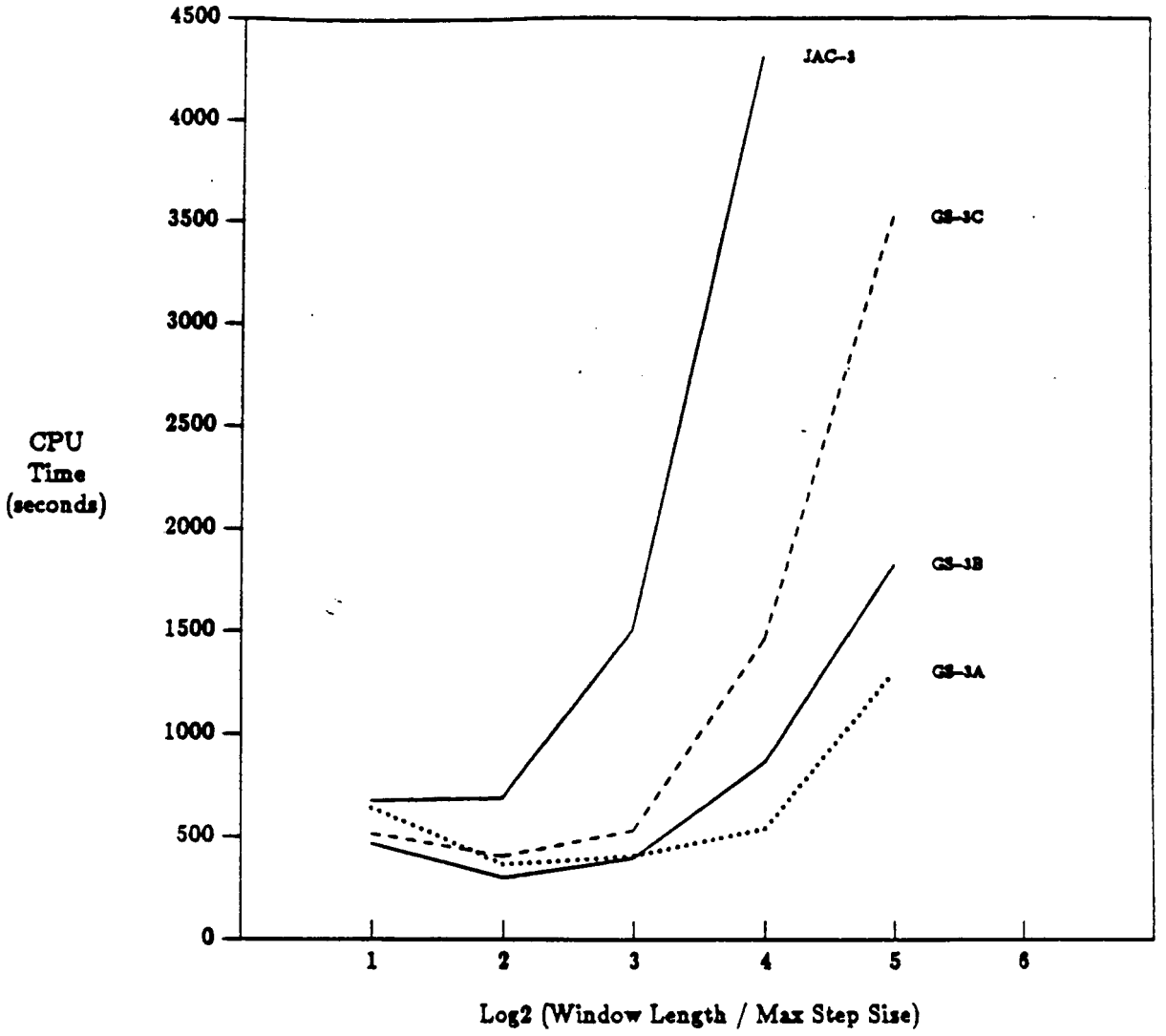
□

(6.5/3) (01)(23)(45)	window length / max step size					
Jacobi	2	4	8	16		
CPU time (second)	675.76	690.34	1514.4	4314.26		
total iterations	691	151/171	147/282	144/439		
window numbers	230	31	30	29		
iteration/window	3.004	4.871	4.9	4.966		
steps performed	32224	34386	74640	205272		
non-converged steps	26514	26612	61066	182898		
steps redundant (%)	17.72	22.61	18.18	10.90		
interpolations	13267	23507	51452	143366		
function evaluations	85162	81254	174538	474118		

Table 6.6: Some Statistics of Example 6.5, Jacobi

(6.5/3) (01)(23)(45)	window length / max step size					
Gauss-Seidel	2	4	8	16	32	64
CPU time (second)	642.02	366.62	410.84	544.8	1309.42	
total iterations	690	88	58	26	22/37	
window numbers	230	29	16	6	5	
iteration/window	3	3.034	3.625	4.333	4.4	
steps performed	30160	17694	19588	23676	54544	
non-converged steps	22324	12822	12154	14380	37608	
steps redundant (%)	25.98	27.53	37.95	39.26	31.05	
interpolations	20591	14158	16227	20578	42597	
function evaluations	79662	42226	45882	54274	124330	
(6.5/3) (45)(01)(23)	window length / max step size					
Gauss-Seidel	2	4	8	16	32	64
CPU time (second)	467.16	300.32	399.72	867.86	1833.64	
total iterations	639	88	61	35/50	32/67	
window numbers	230	29	16	8	7	
iteration/window	2.778	3.034	3.8125	4.375	4.571	
steps performed	23038	15054	17864	38374	78796	
non-converged steps	16910	10098	10904	24236	56212	
steps redundant (%)	29.72	32.92	38.96	36.84	28.66	
interpolations	12424	9996	12799	30067	62273	
function evaluations	58898	35250	41482	87694	178970	
(6.5/3) (45)(23)(01)	window length / max step size					
Backward Gauss-Seidel	2	4	8	16	32	64
CPU time (second)	515.8	405.74	535.06	1465.12	3534.04	
total iterations	649	115	80/90	77/152	78/228	
window numbers	230	29	17	16	16	
iteration/window	2.82	3.965	4.706	4.8125	4.875	
steps performed	25246	19982	25762	69100	161554	
non-converged steps	20084	14158	17332	53200	136462	
steps redundant (%)	20.45	29.14	32.72	23.01	15.53	
interpolations	11562	14085	19251	53182	126470	
function evaluations	65226	46978	59930	159022	368634	

Table 6.7: Some Statistics of Example 6.5, Gauss-Seidel

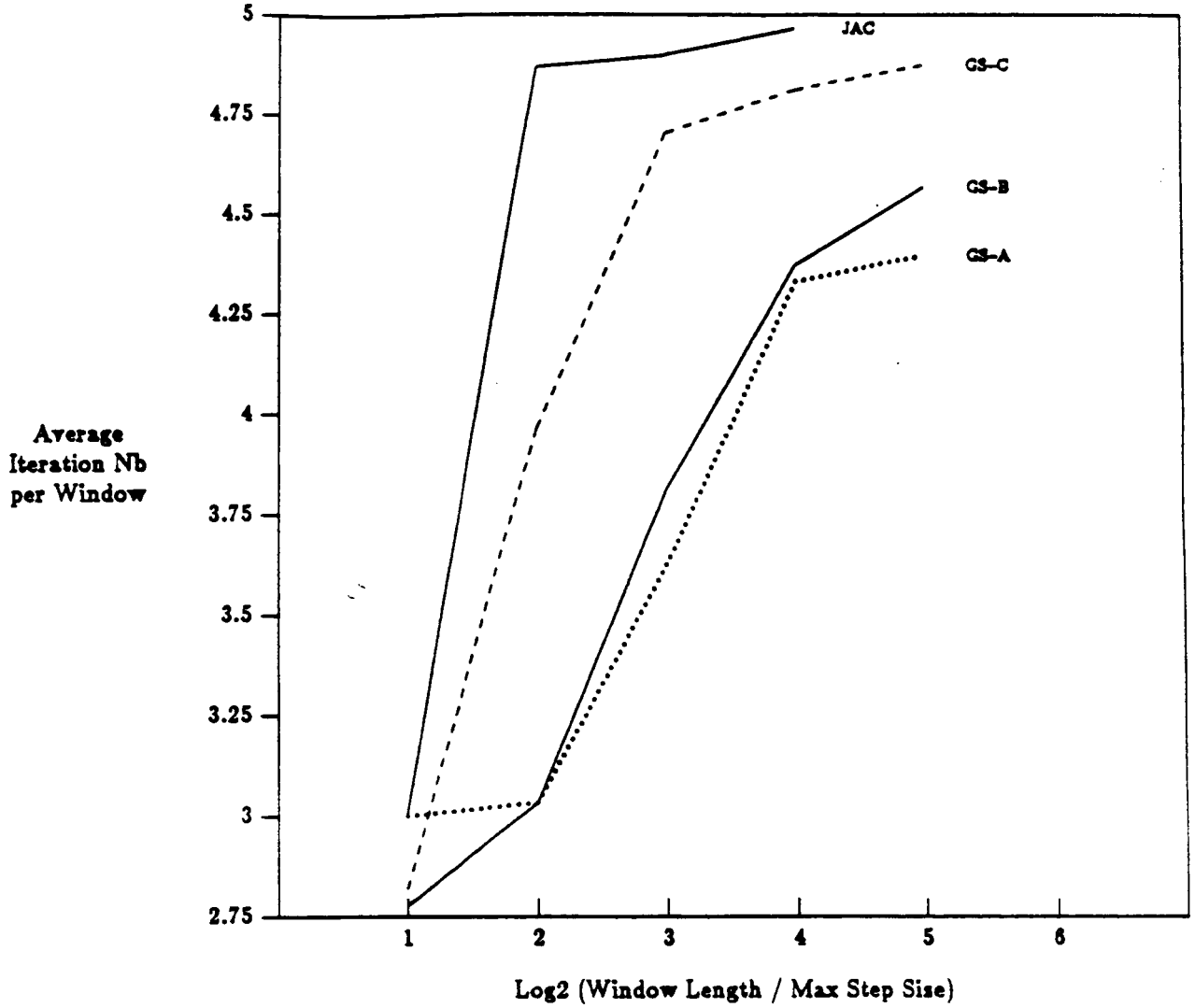


CPU Time for Direct Method : 180.24

Example : 6.5

GS-3A : (01)(23)(45)
GS-3B : (45)(01)(23)
GS-3C : (45)(23)(01)

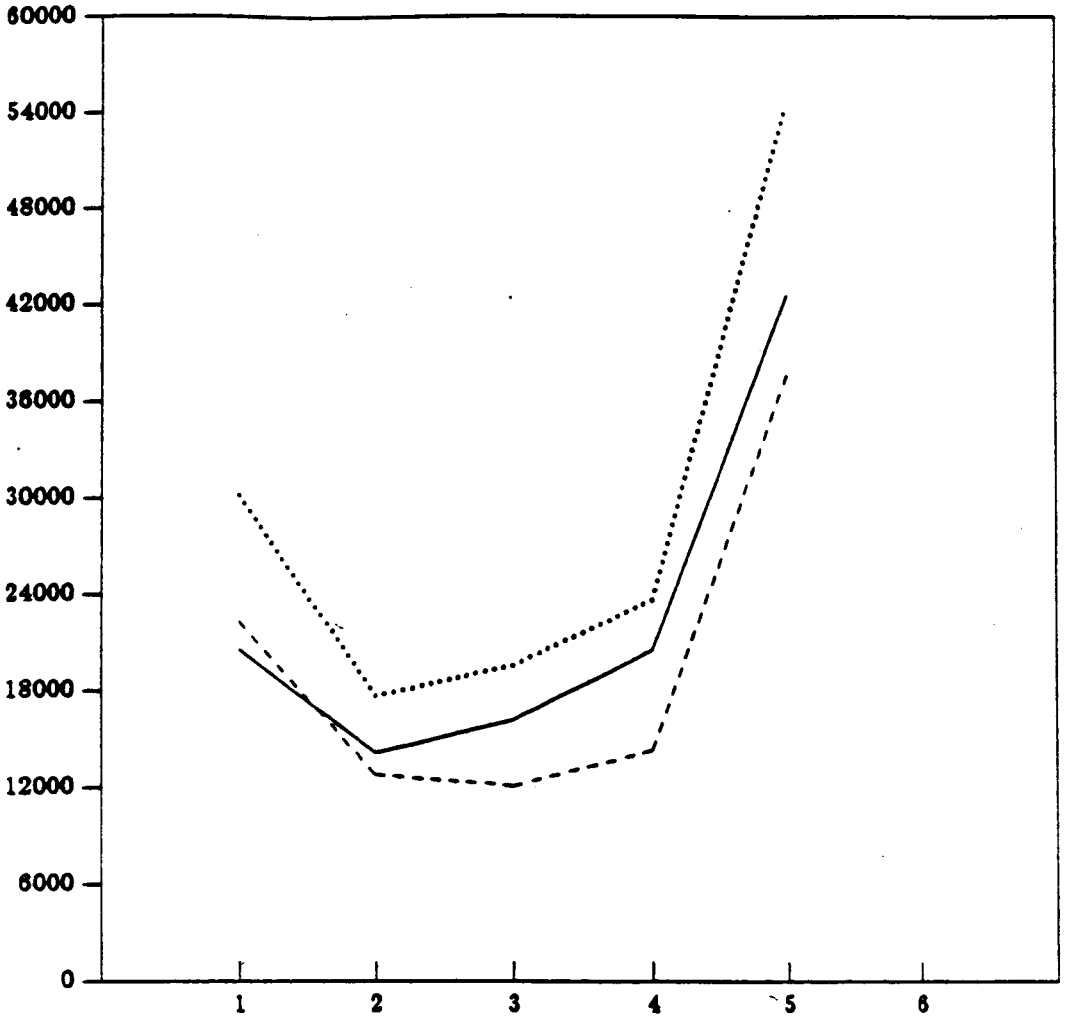
Figure 1 of Example 6.5



Example : 6.5

GS–A : (01)(23)(45)
GS–B : (45)(01)(23)
GS–C : (45)(23)(01)
JAC : (01)(23)(45)

Figure 2 of Example 6.5

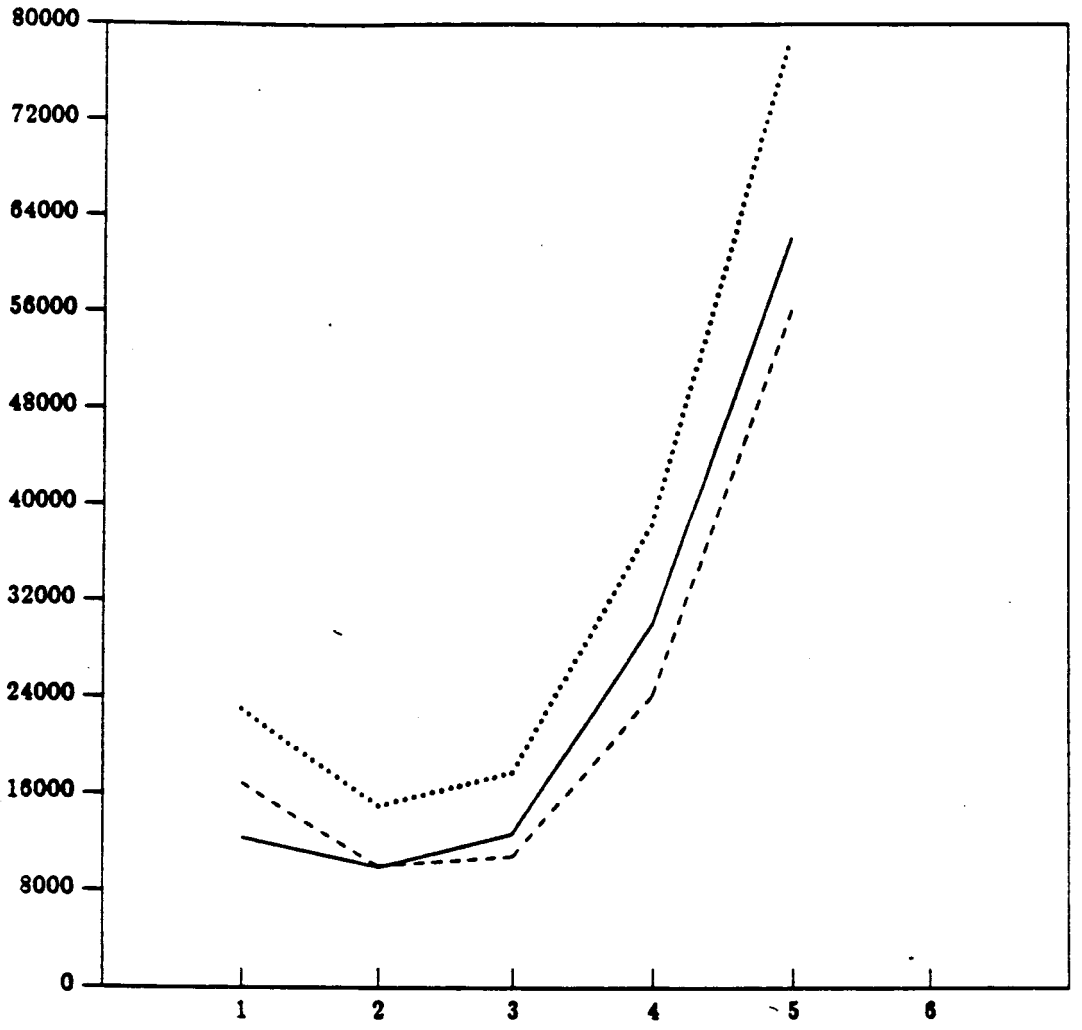


Example : 6.5

GS–3A : (01)(23)(45)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 3 of Example 6.5

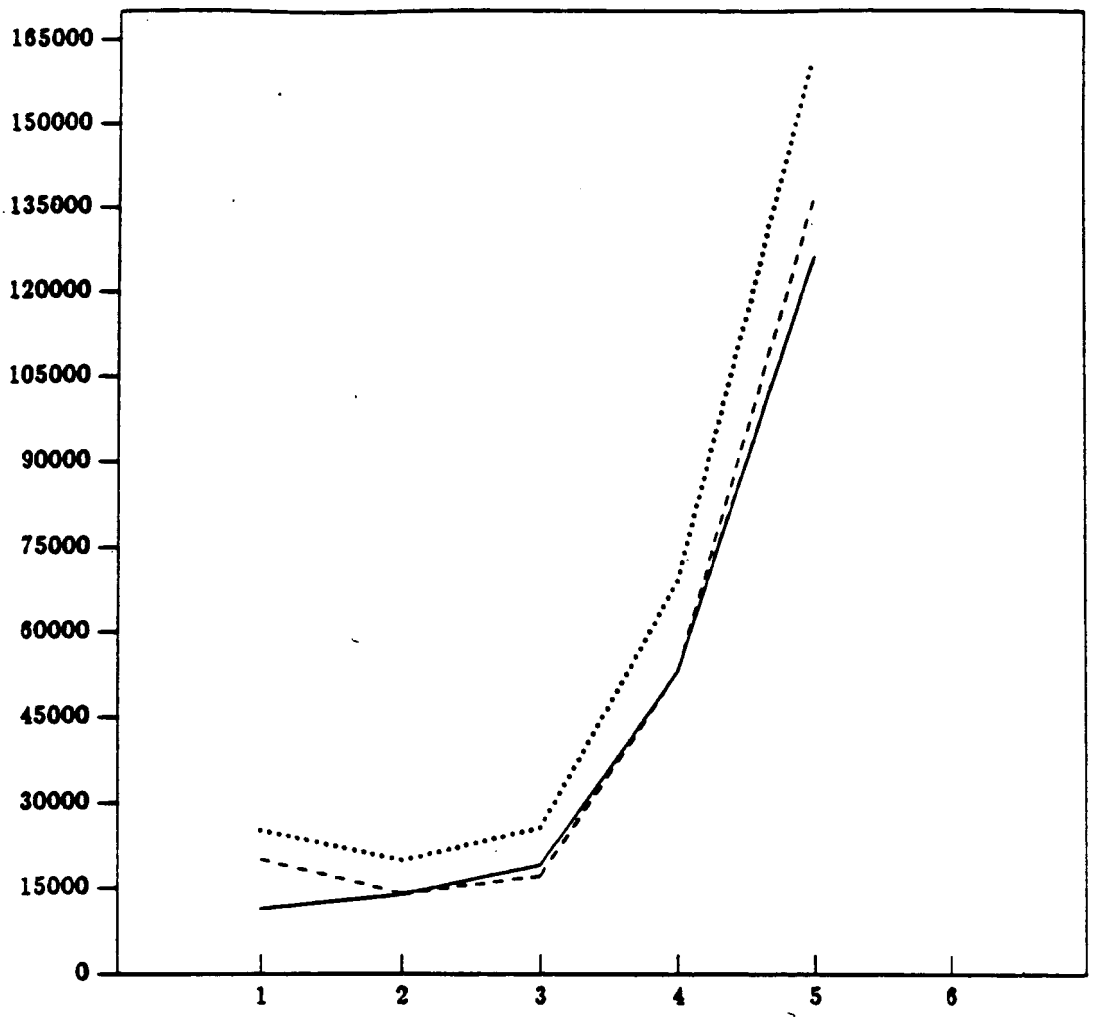


Example : 6.5

GS–3B : (45)(01)(23)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 4 of Example 6.5

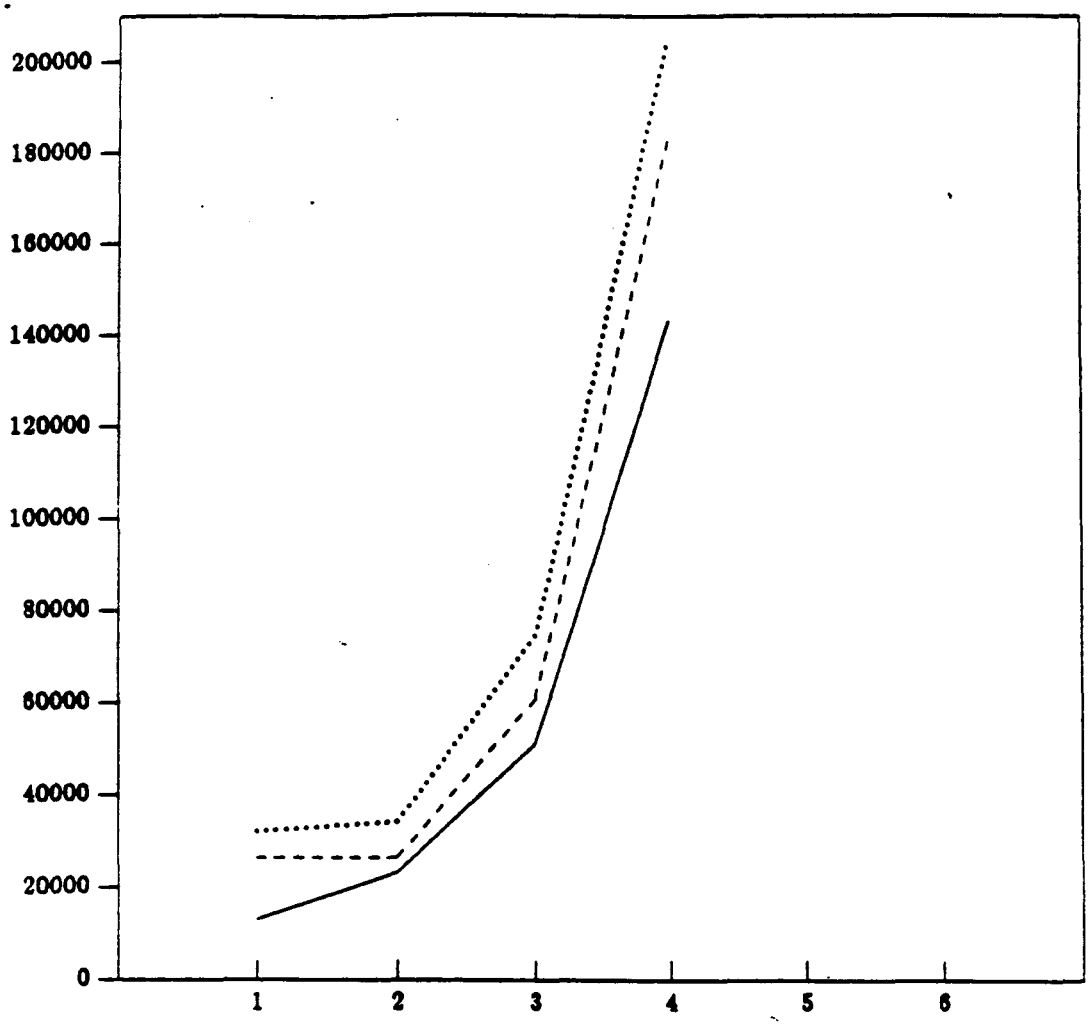


Example : 6.5

GS–3C : (45)(23)(01)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 5 of Example 6.5



Example : 6.5

JAC–3 : (01)(23)(45)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 6 of Example 6.5

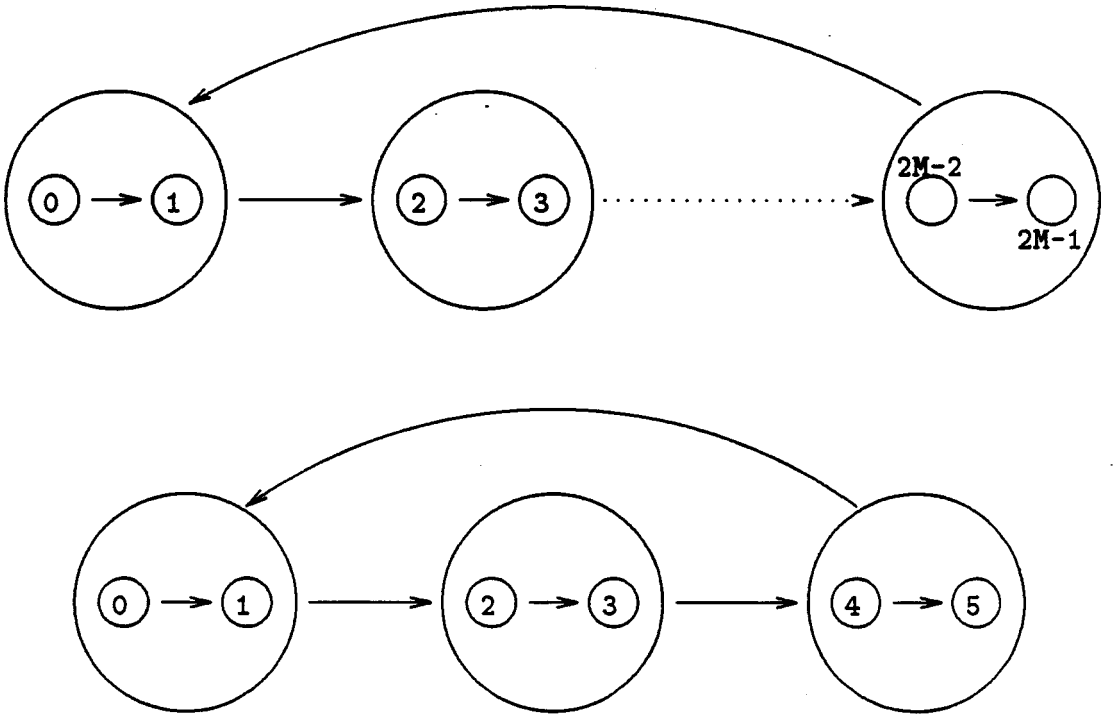


Figure 6.6: Dependency Graph of Example 6.6

6.2.2 Solution of a Wave-Like Equation

We analyzed the performance of WRODE on linear problems in previous subsection. In this subsection we will use the same approaches to solve a nonlinear problem, a "travelling wave" equation.

Example 6.6 :

$$\begin{aligned} \dot{x}_i &= y_{i-1} - x_i \\ \dot{y}_i &= -((y_i+1)e^{20(y_i+1)(x_i-0.5)} + (y_i-1)e^{20(y_i-1)(x_i+0.5)}) \end{aligned}$$

with $y_0 = y_M$ and $x_i(0) = -1^{(i-1)}$, $y_i(0) = -1^i$, for $i = 1, 2 \ldots, M$.

This is a highly nonlinear problem which was designed by Gear to simulate a series of capacitors and flip-flops and was used as an example of a multirate system. The capacitor x_i is slowly charged by its input y_{i-1}. When this capacitor reaches .5 the flip-flop y_i is triggered.

This allows the following capacitor x_{i+1} to slowly discharge and when its value falls below -.5, the next flip-flop y_{i+1} is triggered, etc. The behavior of the components is illustrated in figure 6 and 7.

Since the signal flows in one direction and flows back after some delay, a feedback loop is formed. (See Fig 6.6) Because of the circular dependence relations, the dependency graph of this system is a cycle of length equal to the number of subsystems. We want to test the efficiencies of the forward and backward Gauss-Seidel and Jacobi methods on this system.

To use waveform relaxation method to integrate this system over $[0, T]$, we partition the system into M subsystems, $(x_i, y_i), i = 1, 2, \ldots, M$, and integrate each subsystem independently or sequentially. In the forward (backward) Gauss-Seidel numbering we number each subsystem in the (reverse) direction of the signal flow which breaks the cycle into either one or $M - 1$ ascending chains. The statistics in Table 6.8 are the results for the case $M = 3$. From Table 6.8 and graphs, we see the expected result that forward Gauss-Seidel has the fastest speed of convergence while Jacobi has the slowest. If we break the original system into M subsystems, (y_i, x_{i+1}), i = 1, 1, ..., M-1, and (y_M, x_0), then the system becomes extremely stiff and the integrator breaks down very quickly.

□

(6.6/3) (01)(23)(45)	window length / max step size					
Gauss-Seidel	2	4	8	16	32	64
CPU time (second)	75.68	73.70	67.88	79.98	82.5	80.5
total iterations	108	44	27	15	8	5
window numbers	48	19	11	6	3	2
iteration/window	2.25	2.316	2.455	2.5	2.667	2.5
steps performed	5254	5398	5052	5362	5498	4782
non-converged steps	4660	4248	4060	3642	3704	3032
steps redundant (%)	11.30	21.30	19.63	32.07	32.63	36.59
interpolations	1779	2255	2177	2400	2669	2193
function evaluations	13570	13310	12418	13006	13222	11550
(6.6/3) (45)(23)(01)	window length / max step size					
Backward Gauss-Seidel	2	4	8	16	32	64
CPU time (second)	95.78	85.16	89.98	101.48	85.26	208.94
total iterations	129	51	35	19	10	11/16
window numbers	48	19	11	6	3	3
iteration/window	2.6875	2.684	3.182	3.167	3.333	3.667
steps performed	6754	6282	6696	6672	5632	13032
non-converged steps	4440	4206	3768	2946	2596	5494
steps redundant (%)	34.26	33.04	43.72	55.84	53.90	57.84
interpolations	2591	2959	3084	3349	3122	6941
function evaluations	17394	15490	16474	16166	13518	31470
(6.6/3) (01)(23)(45)	window length / max step size					
Jacobi	2	4	8	16		
CPU time (second)	99.2	99.14	98.06	164.16		
total iterations	137	55	37	27/32		
window numbers	48	19	11	8		
iteration/window	2.854	2.895	3.364	3.375		
steps performed	7058	7182	7166	11032		
non-converged steps	4514	4264	3812	5032		
steps redundant (%)	36.04	40.62	46.8	54.38		
interpolations	2429	3142	3146	5387		
function evaluations	18202	17690	17594	26750		

Table 6.8: Some Statistics of Example 6.6

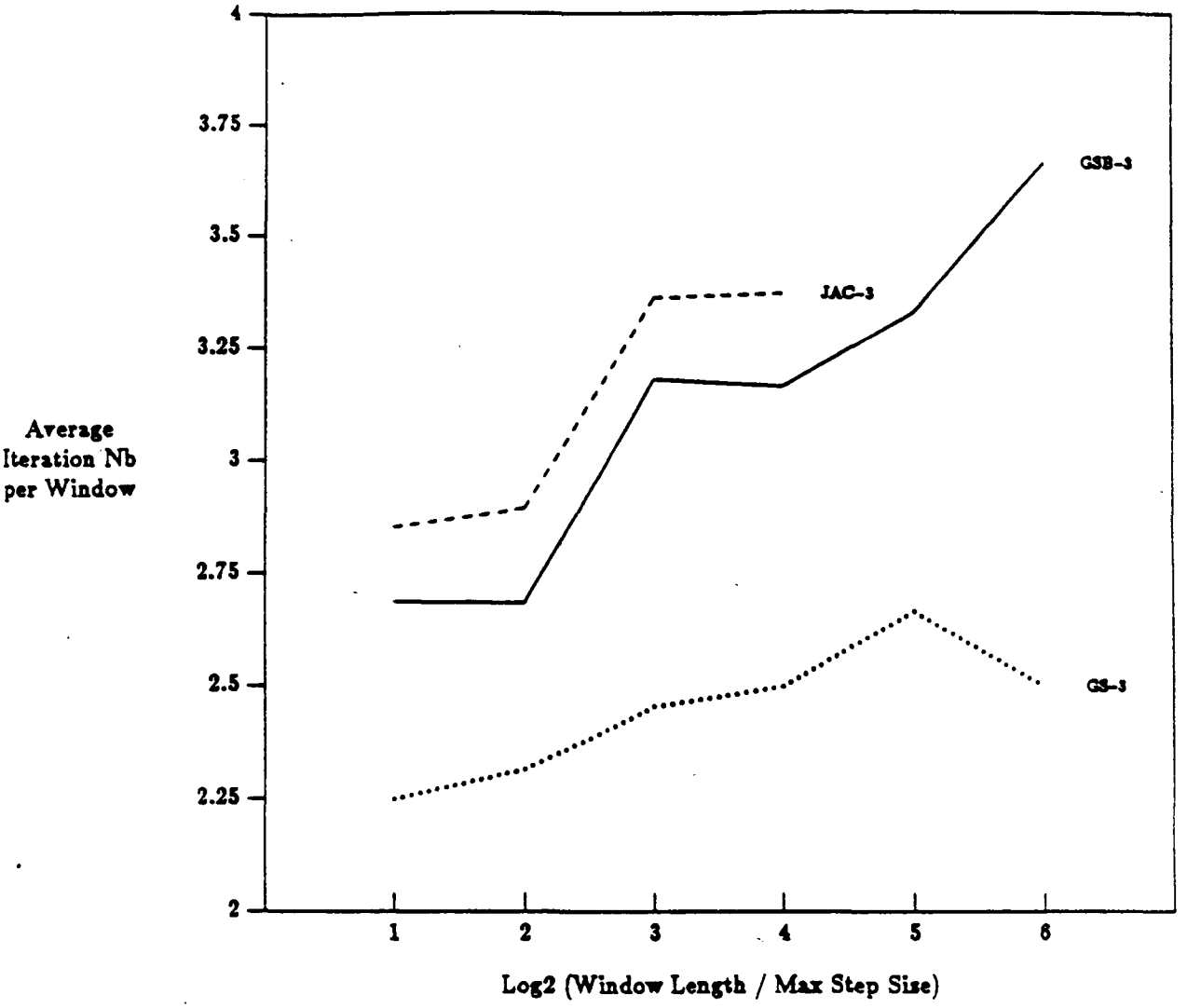


Figure 1 of Example 6.6

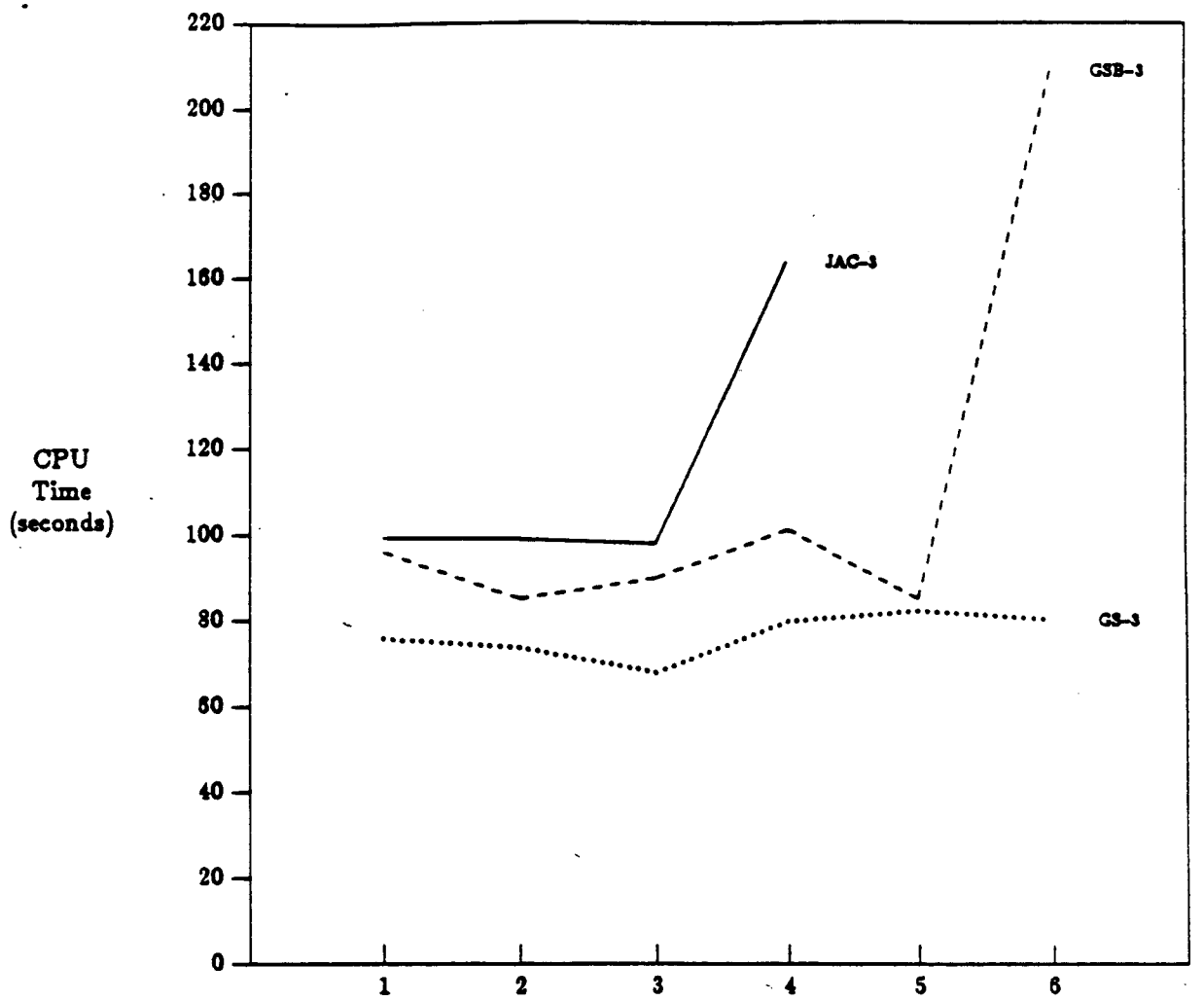
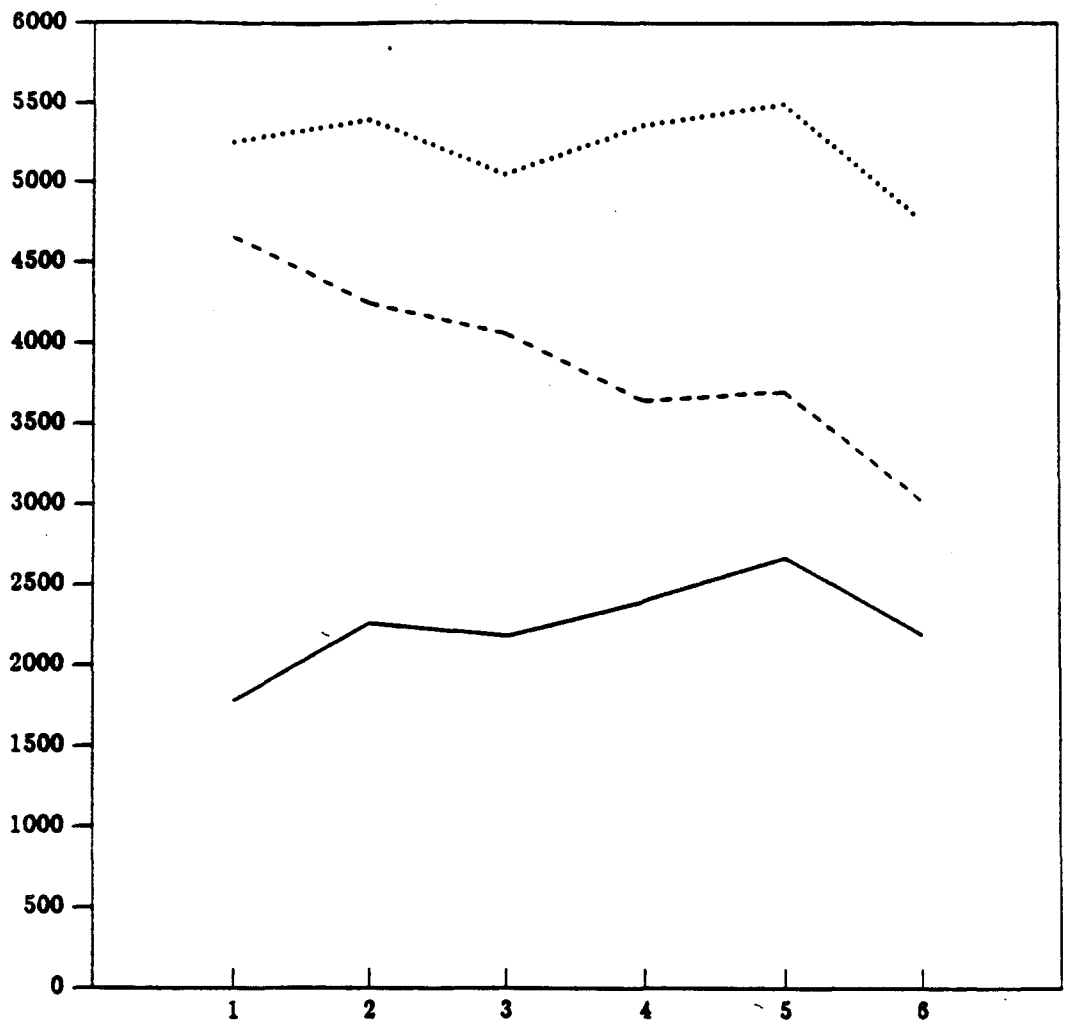


Figure 2 of Example 6.6

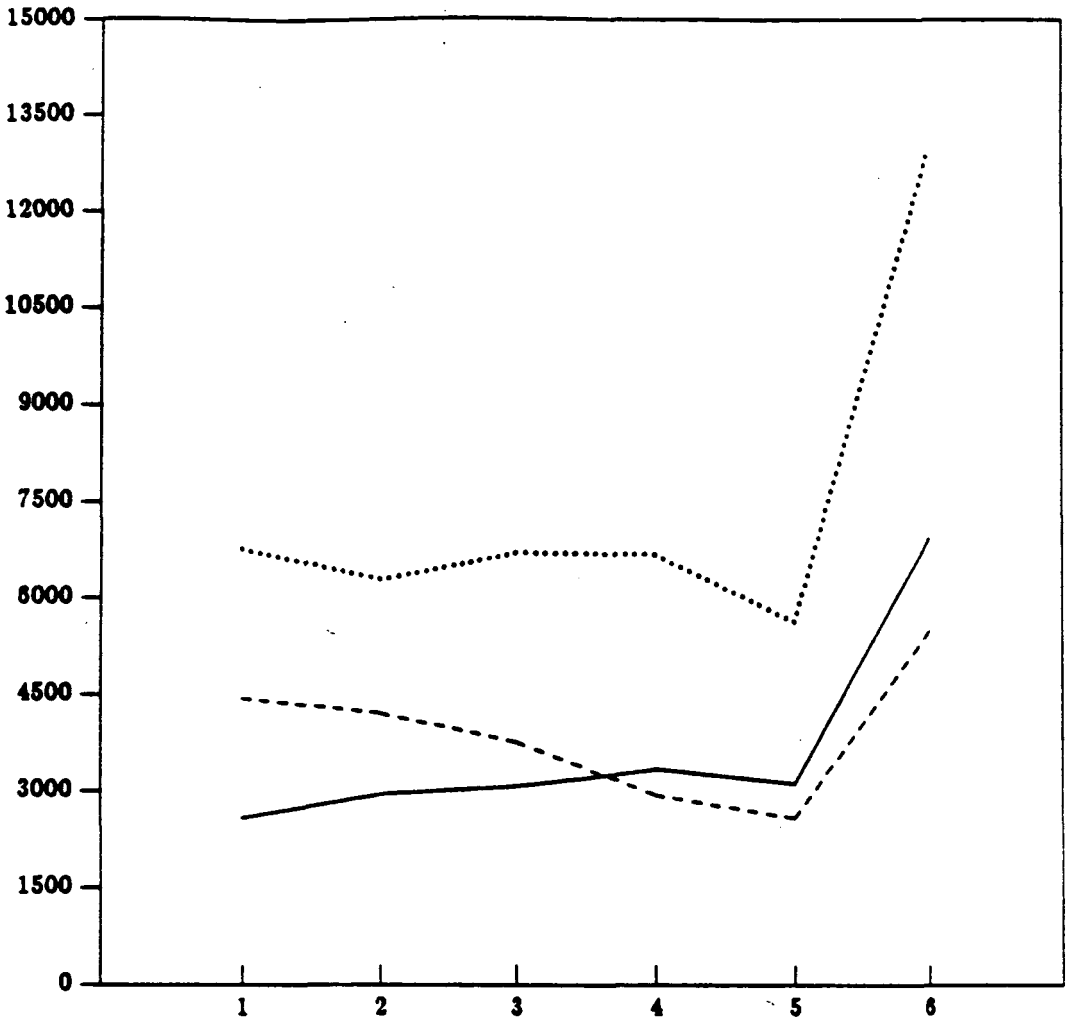


Example : 6.6

GS–3 : (01)(23)(45)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 3 of Example 6.6



Example : 6.6

GSB–3 : (45)(23)(01)

dotted – # of Integration Steps w/o. Partial Convergence
dashed – # of Integration Steps w. Partial Convergence
solid – # of Interpolations

Figure 4 of Example 6.6

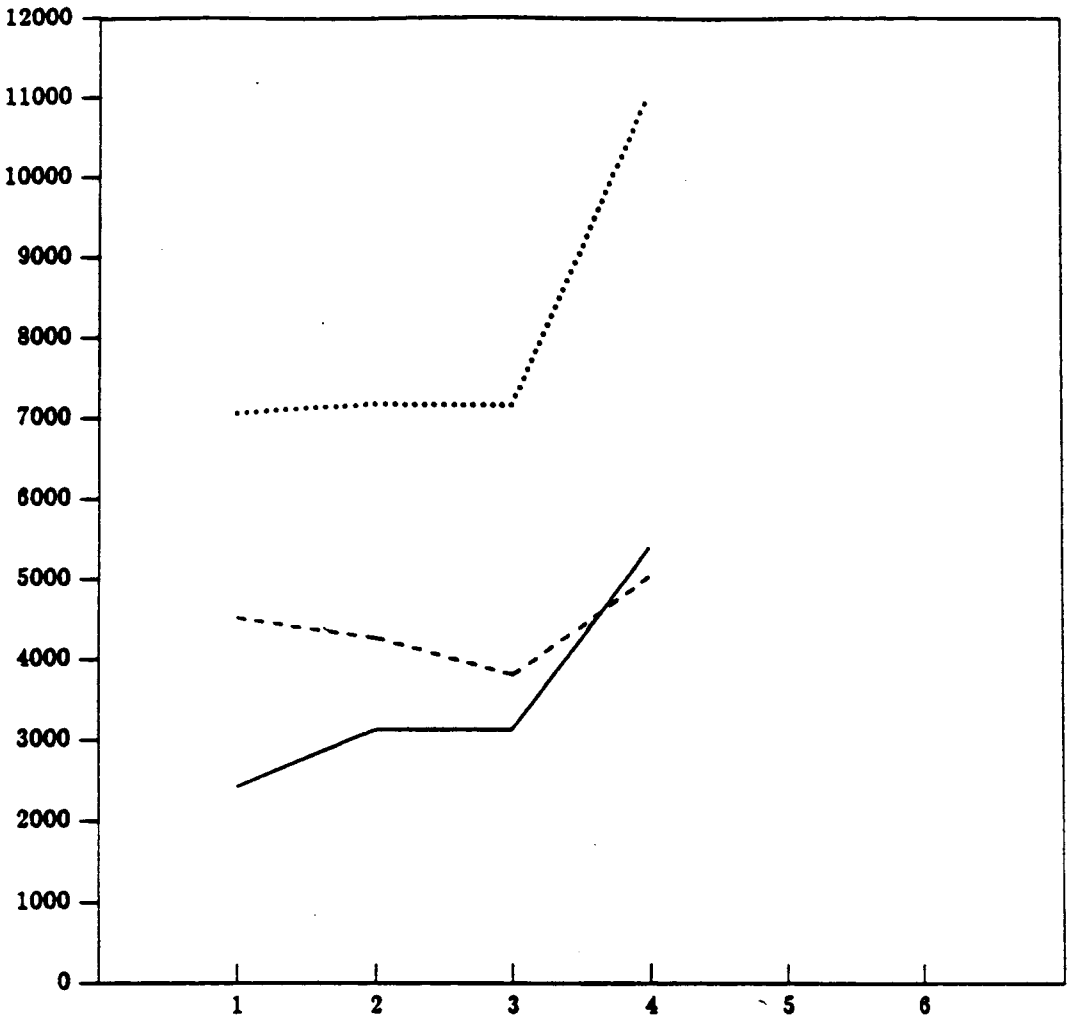


Figure 5 of Example 6.6

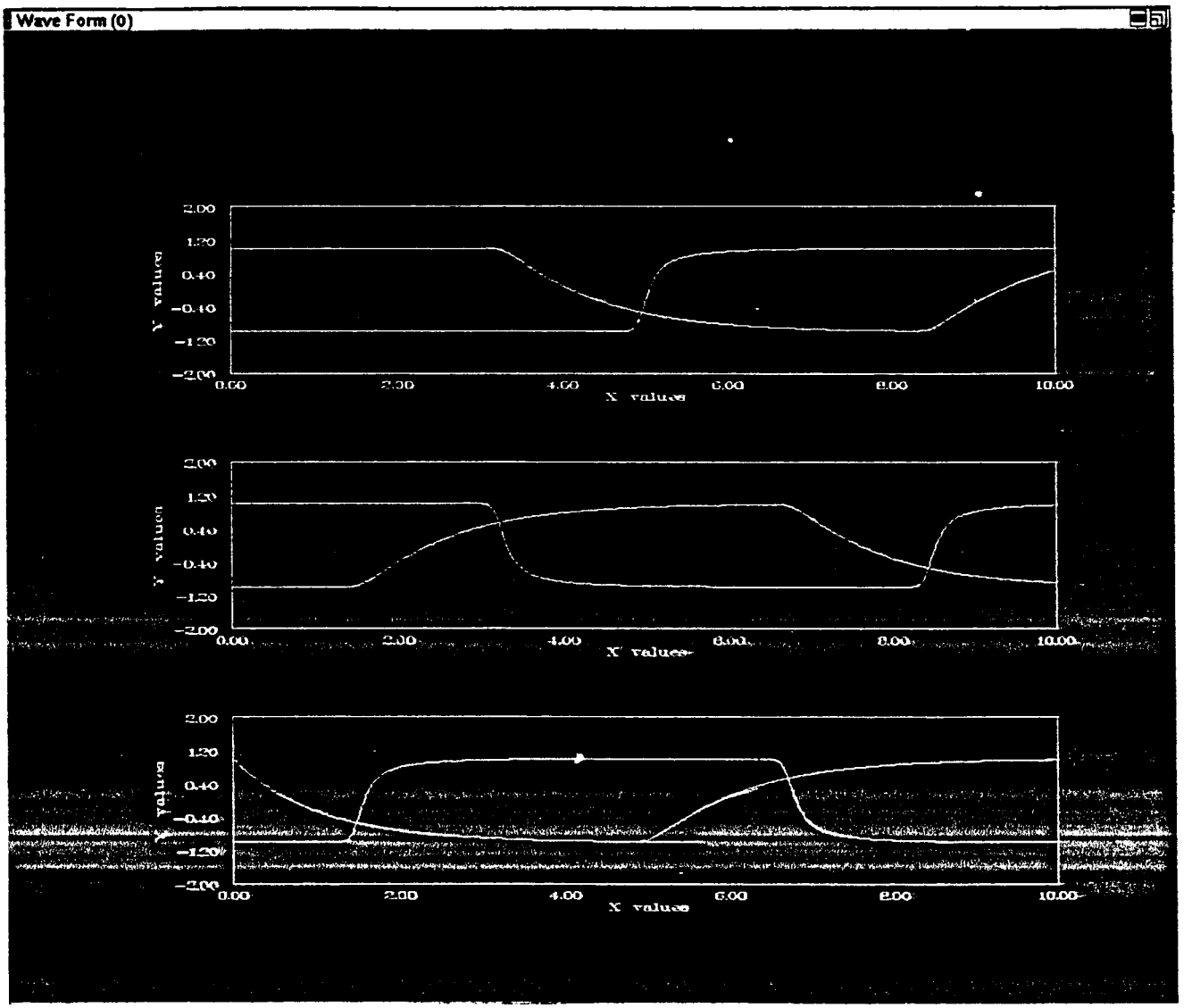


Figure 6 of Example 6.6

Travelling waves generated by Gauss-Seidel scheme

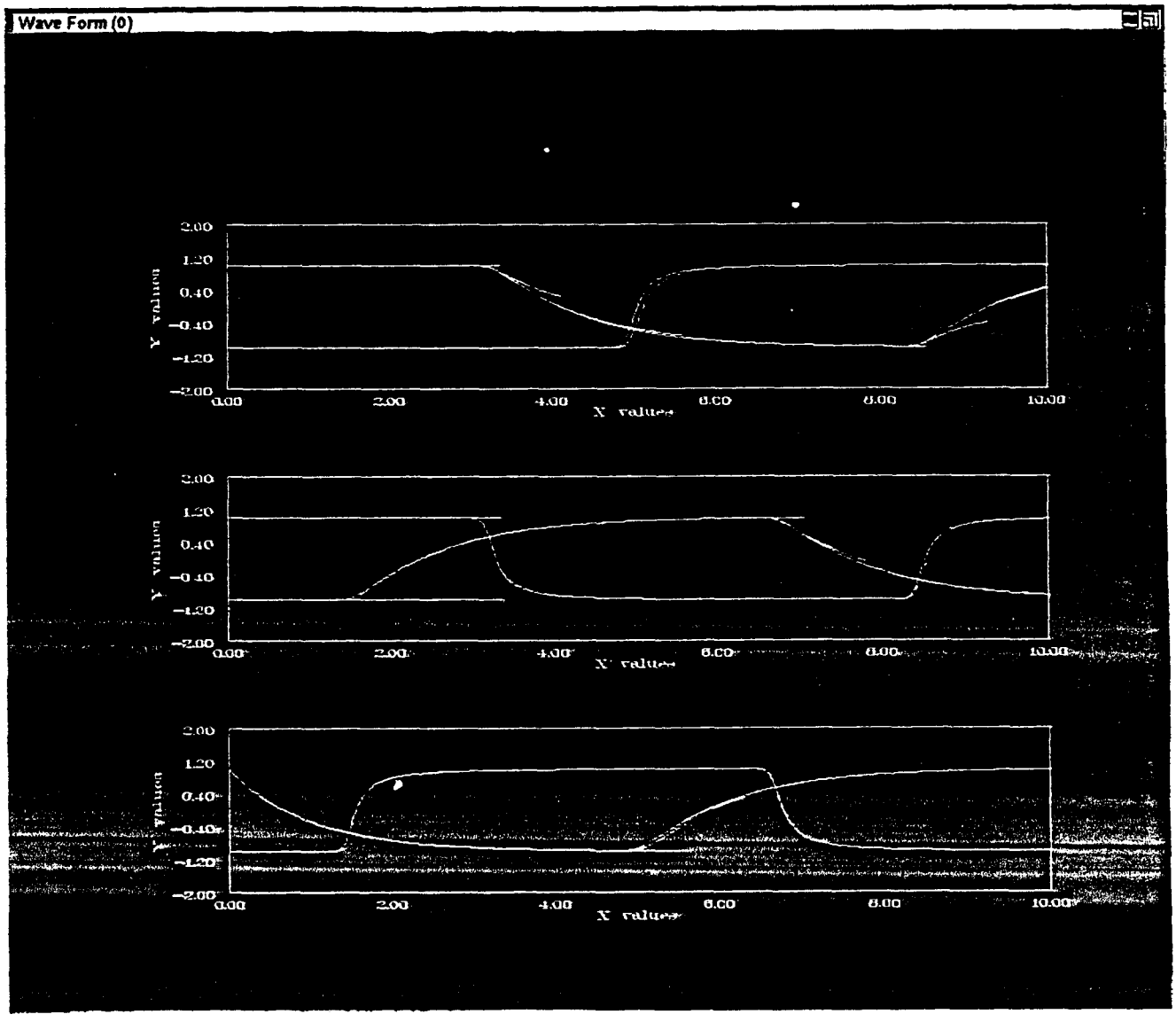


Figure 7 of Example 6.6

Travelling waves generated by Jacobi scheme

M	Tout	CPU Time							Best Ratio
		WGS						Direct	
		2	4	8	16	32	64		
3	10.0	19.24	16.48	15.48	17.36	18.42	15.68	5.44	2.846
	20.0	39.80	33.98	33.36	35.12	38.60	45.26	11.24	2.968
9	10.0	19.86	17.12	18.12	18.70	19.20	20.22	23.62	0.725
	20.0	47.18	41.24	58.88	52.24	51.56	49.00	50.50	0.817
	30.0	68.44	59.70	85.56	76.36	72.08	70.58	68.60	0.870
	40.0	120.60	83.94	110.62	131.14	118.78	120.36	101.32	0.828
15	10.0	23.38	21.68	19.86	20.26	19.62	23.70	58.98	0.333
	20.0	43.94	43.32	43.24	40.78	41.02	44.14	124.20	0.328
	30.0	108.08	71.50	79.68	75.92	94.06	89.14	168.56	0.424
	40.0	147.64	93.28	105.36	98.02	118.4		253.56	0.368

Table 6.9: Efficiency of WGS method

6.3 Efficiency of WGS

In this final section we look at two more examples to observe the efficiency of the WGS method when system size grows.

First example we use the same system of wave-like equations as described in Example 6.6. We experiment the CPU times used by the WGS method for $M = 3, 9, 15$, where $2 * M$ is the number of equations in the system.

We define the "Best Ratio" as follow:

$$Best\ Ratio = \frac{the\ least\ CPU\ time\ used\ by\ the\ WGS\ method}{CPU\ time\ used\ by\ the\ direct\ method}.$$

So when the best ratio is greater than one, this indicates that the direct method outperforms the WGS method. In Table 6.9 we compare the efficiency of WGS method with that of the direct method. From the table we notice that as the size of the system grows, the WGS method becomes more efficient.

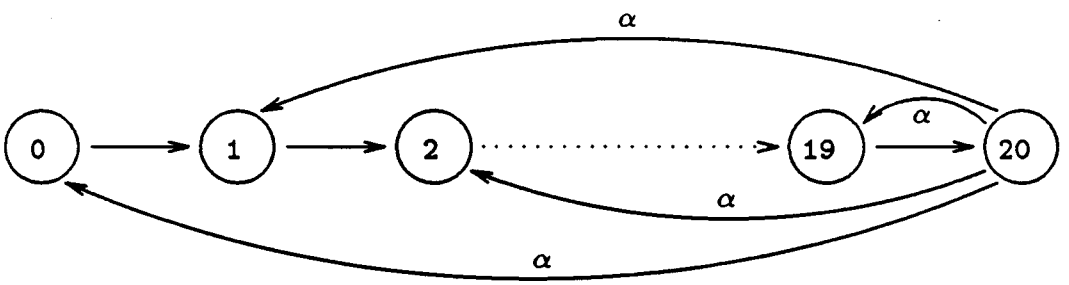


Figure 6.7: Dependency Graph of Example 6.7

Example 6.7 : In this example we consider the following system :

$$
\begin{aligned}
\dot{y_i} &= -10\ u_i + u_{i-1} + \alpha\ u_n + 0.1\ cos(0.1\,t) \qquad i = 1, \ldots, n-1 \\
\dot{y_n} &= -10\ u_n + u_{n-1} + 20\ cos(20\,t) \\
u_0 &= 0, \\
u_i &= y_i - sin(0.1\,t) \qquad i = 1, \ldots, n-1 \\
u_n &= y_n - sin(20\,t) \\
y_i(0) &= 0 \qquad i = 1, \ldots, n.
\end{aligned}
$$

The Jacobian matrix of this system has the following structure :

$$
\begin{bmatrix}
-10 & 0 & 0 & 0 & \cdots & 0 & \alpha \\
1 & -10 & 0 & 0 & \cdots & 0 & \alpha \\
0 & 1 & -10 & 0 & \cdots & 0 & \alpha \\
0 & 0 & 1 & -10 & \cdots & 0 & \alpha \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & 1 & -10
\end{bmatrix}
$$

and its dependency graph is shown in Figure 6.7.

We integrate this system from t = 0 to t = 4 with n = 21 and vary the size of α. The first 20 components form the slow class and the last component is the fast class. We consider the following three different partitionings : (a) the slow components, 0 to 19, forms one subsystem and the fast component, 20, itself forms the other subsystem; (b) the slow components, 0 to

9 and 10 to 19, form the first two subsystems and the fast component 20 forms the third subsystem; (c) the slow components, 0 to 4, 5 to 9, 10 to 14, and 15 to 19 form the first four subsystems and the fast component 20 forms the fifth subsystem. The subsystem to be integrated first is either the fast component or the slow subsystem containing component 0 and the order of integration follows the dependence relations.

The results of CPU time consumed by the WGS and the direct method with different coupling factors, α, are given in Table 6.10. The WGS method performs quite well when the coupling factor is small. From the table we also notice that when the coupling factor is large, the entire integration interval is divided into a very large number of windows so big page swapping overhead may occur.

In Table 6.11 we list the efficiencies of the WGS method under different partitionings. From the table we notice that when the size of a subsystem is large, even it contains all slow components, the subsystem is forced to use very small stepsize which limits the length of window to be chosen. Hence total number of windows increases in the entire integration interval and this causes poor performance on WRODE.

Since all the slow components depends on the single fast component in this system, no matter how we partition the system the reduced dependency graph always has a minimum cycle of length "2". The speed of convergence under these three partitionings are all the same (this can be observed from the table).

At the beginning of this chapter we mentioned that an advantage of using multirate method is the saving in matrix computation time at each integration step if an implicit method is used. The work required in solving a full linear system is roughly $O(N^3)$ where N is the dimension of the linear system. In this example under the three partitionings, (a), (b), and (c), the most matrix computation work required at each integration step is $O(20^3)$, $O(10^3)$, and $O(5^3)$, respectively. Total matrix computation work for a partitioning depends on total number of integration steps been performed under this partitioning. From the table we can see that under partitioning (c) less number of windows is needed to cover the entire integration interval, which indicates that larger stepsize is used and hence less number of integration steps is performed.

Sub-system #	α	CPU Time							Best Ratio
		WGS						Direct	
		2	4	8	16	32	64		
2	0.D0	22.06	21.30	17.66	17.80	14.94	13.50	45.22	0.298
		32.32	30.16	25.06	26.82	22.70	22.06		0.488
	1.D-3	65.56	43.22	48.28	37.02	37.06	24.36	45.00	0.541
		80.24	67.36	67.18	74.78	74.98	43.22		0.960
	1.D-2	97.60	105.36	97.38	85.00	77.36	67.62	44.90	1.506
		98.46	98.22	100.34	90.98	84.12	73.42		1.635
	1.D-1	295.34	220.50	168.18	127.02	108.56	96.74	44.68	2.165
		186.28	145.86	131.48	124.50	110.66	97.36		2.179

Sub-system #	α	Total Iter # / Window #					
		WGS					
		2	4	8	16	32	64
2	0.D0	28/14 = 2	18/9 = 2	10/5 = 2	8/4 = 2	4/2 = 2	4/2 = 2
		40/14 = 2.857	24/9 = 2.667	14/5 = 2.8	12/4 = 3	6/2 = 3	6/2 = 3
	1.D-3	122/61 = 2	40/20 = 2	31/15 = 2.067	14/7 = 2	8/4 = 2	4/2 = 2
		134/61 = 2.197	54/20 = 2.7	42/15 = 2.8	21/7 = 3	11/4 = 2.75	6/2 = 3
	1.D-2	217/108 = 2.009	105/39 = 2.692	79/28 2.821	44/15 = 2.933	24/8 = 3	12/4 = 3
		216/108 = 2	93/39 = 2.385	77/28 = 2.75	45/15 = 3	23/8 = 2.875	12/4 = 3
	1.D-1	501/218 = 2.298	221/77 = 2.870	102/34 = 3	45/15 = 3	24/8 = 3	12/4 = 3
		436/218 = 2	162/77 = 2.104	81/34 = 2.382	43/15 = 2.867	23/8 = 2.875	12/4 = 3

Table 6.10: Efficiency of WGS wrt Coupling Factors

Sub-system #	α	CPU Time							Best Ratio
		WGS						Direct	
		2	4	8	16	32	64		
2	1.D-1	295.34	220.50	168.18	127.02	108.56	96.74	44.68	2.165
		186.28	145.86	131.48	124.50	110.66	97.36		2.179
3	1.D-1	109.7	68.66	62.60	58.76	55.08	56.64	44.68	1.232
		62.24	64.16	59.44	58.60	59.26	62.06		1.312
5	1.D-1	50.06	45.04	45.82	36.36	41.86	37.54	44.68	0.814
		47.60	45.06	47.58	37.60	45.90	39.60		0.842

Sub-system #	α	Total Iter # / Window #					
		WGS					
		2	4	8	16	32	64
2	1.D-1	501/218 = 2.298	221/77 = 2.870	102/34 = 3	45/15 = 3	24/8 = 3	12/4 = 3
		436/218 = 2	162/77 = 2.104	81/34 = 2.382	43/15 = 2.867	23/8 = 2.875	12/4 = 3
3	1.D-1	219/75 = 2.92	57/19 = 3	30/10 = 3	18/6 = 3	9/3 = 3	9/3 = 3
		150/75 = 2	54/19 = 2.842	28/10 = 2.8	17/6 = 2.833	9/3 = 3	9/3 = 3
5	1.D-1	51/17 = 3	27/9 = 3	18/6 = 3	12/4 = 3	9/3 = 3	6/2 = 3
		47/17 = 2.765	24/9 = 2.667	17/6 = 2.833	11/4 = 2.75	9/3 = 3	6/2 = 3

Table 6.11: Efficiency of Different Partitioning

It is therefore clear that partitioning (c) has the best performance, that is, it consumes the least CPU time among the three partitionings, because of the least number of integration steps and the least matrix computation work required at each step.

In summary, the experiments in this chapter show that care must be used in selecting the numbering and partitioning when using WRODE. If use correctly WRODE can perform well even on small systems but is best suited for large system with a high percentage of slow components. The code will not perform well if the slow components are strongly dependent on the fast.

Chapter 7

Summary

From the results obtained in chapter 6, we see that the total work is basically proportional to the total number of integration steps been performed. (The profile of CPU time for different ratios of window length to stepsize is similar to the profile of the total number of integration steps.) Thus the ratio of window length to stepsize, which use the least number of integration steps, is the most efficient one.

Note that in shorter windows the advantage of multirate process is lost because of the restriction on the choice of stepsize, but the time grids for different subsystems are highly synchronized so it will reduce the number of interpolations significantly and it usually takes less iterations to reach convergence; in longer windows each subsystem can choose its stepsize as large as possible which will reduce the total number of integration steps, but it may need more iterations to reach convergence inside each window. When window length is short the total number of windows will increase and a lot of overhead occurs, for instance, the start up time for each waveform integration, the page swapping from subsystem to subsystem and from window to window. Observing the interaction among all the factors that affect the performance of waveform relaxation was the main goal of this experimental code.

From the tables we can see that the total number of function evaluations is roughly proportional to the total integration steps, so if we can reduce the number of integration steps we can save some computing time. To achieve this we should exploit the partial convergence behavior of successive waveforms and by doing this we can save from 7 % to 55 % of integration steps.

When a large ratio between window length and maximum step size was used, a lot of work

was wasted (especially in the Jacobi approach). This can be noticed from the huge number of integration steps and a large difference between the total number of window iterations been done and total number of iterations inside convergent windows.

From the execution profile we found that about 20 percent of the time was spent on number comparison and in subroutine *wpt_loc()*. This subroutine is called when interpolation is needed for computing approximations of input variables to evaluate the derivatives in a subsystem. It will locate the proper time point in the waveforms of input variables. Since we did not use a sophisticated strategy to locate a time point in a waveform (we always search the time point from the beginning of a waveform, a lot of floating point number comparisons will be performed when the target time point is near the end of a waveform.), we should be able to save at least 10 percent of the time if we can modify this routine. The result is shown in the following table.

CPU Time	WGS						Direct
	2	4	8	16	32	64	
w/o. FP coprocessor	75.68	73.70	67.88	79.88	82.50	80.50	28.44
w. FP coprocessor	21.46	18.54	17.52	21.20	22.18	23.56	5.48
efficient wpt_loc()	19.24	16.48	15.48	17.36	18.42	15.68	5.48
saving (%)	10.34	11.11	11.64	18.11	16.95	33.45	

When a subsystem is being integrated, it must reside in main memory, as do those variables which appear on the right hand sides of the equations defining the variables being integrated. The multirate approach has less page swapping compared to the traditional one when a large system is solved. In the traditional approach, at each time point, all variables have to reside in main memory, so a lot of page swapping is expected. In the multirate approach, at each time point of each subsystem, page swapping is far less frequent because of the smaller size of each subsystem; the large amount of page swapping occurs when integration moves from one subsystem to another inside a window and when integration moves from one window to another window. So the total page swapping will be proportional to the number of subsystems times the total number of windows being used. Using larger ratio of window length to maximum stepsize will reduce the number of windows to be used and hence reduce page swapping. How large the ratio should be chosen such that the speed of convergence will not deteriorate needs further study.

Bibliography

[1] J. A. Bondy and U. S. R. Murty, "Graph Theory with Applications", North Holland, 1976.

[2] R. F. Curtain and A. J. Pritchard, "Functional Analysis in Modern Applied Mathematics", Academic Press, 1977.

[3] C. W. Gear, "Automatic Multirate Methods for Ordinary Differential Equations", *Proc. IFIP* 1980, 717-722, North-Holland Publishing Company.

[4] C. W. Gear and D. R. Wells, "Multirate Linear Multistep Methods", *BIT* 24 (1984), 484-502.

[5] F. Juang, "Accuracy Increase in Waveform Relaxation", Report No. UIUCDCS-R-88-1466, Department of Computer Science, University of Illinois at Urbana-Champaign, 1988.

[6] F. Juang, C. W. Gear, "Accuracy Increase in Waveform Gauss-Seidel, Report No. UIUCDCS-R-89-1518, Department of Computer Science, University of Illinois at Urbana-Champaign, 1989.

[7] E. Lelarasmee, "The Waveform Relaxation Methods for The Time Domain Analysis of Large Scale Nonlinear Dynamical Systems", Ph.D. dissertation, University of California, Berkeley.

[8] E. Lelarasmee, A. E. 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.* Vol. 1, No. 3, pp. 131-145, July 1982.

[9] U. Miekkala and O. Nevanlinna, "Convergence of Dynamic Iteration Methods for Initial Value Problems", REPORT-MAT-A230, Helsinki University of Technology, Institute of Mathematics, Finland, 1985.

[10] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits", Electronic Research Laboratory Report No. ERL-M520, University of California, Berkeley, May 1975.

[11] O. Nevanlinna, "Remarks on Picard-Lindelöf Iteration", REPORT-MAT-A254, Helsinki University of Technology, Institute of Mathematics, Finland, December 1987.

[12] R. D. Skeel, "Waveform Iteration and The Shifted Picard Splitting", *SIAM J. Sci. Stat. Comput.* Vol. 10, No. 4, pp. 756-776, July 1989.

[13] L. F. Shampine "Starting an ODE Solver", Numerical Mathematics Division 5122, Sandia Laboratories, Albuquerque, NM 87115

[14] D. R. Wells, "Multirate Linear Multistep Methods for The Solution of Systems of Ordinary Differential Equations", Report No. UIUCDCS-R-82-1093, Department of Computer Science, University of Illinois at Urbana-Champaign, 1982.

[15] J. K. White, "The Multirate Integration Properties of Waveform Relaxation, with Applications to Circuit Simulation and Parallel Computation", Report No. ERL-85/90, University of California, Berkeley, 1985.

[16] J. White, F. Odeh, A.S. Sangiovanni-Vincentelli and A. Ruehli, "Waveform Relaxation: Theory and Practice", Memorandum No. UCB/ERL M85/65, 1985, Electronics Research Laboratory, College of Engineering, University of California, Berkeley.

[17] S. Wolfram, Mathematica, A System for Doing Mathematics by Computer, Addison-Sesley Publishing Company, Redwood City, CA, 1988. *Mathematica* is a trade mark of Wolfram Research, Inc.

DOE F 1332.16 (10-84)
(Formerly RA-427)

OMB Approval
No. 1910-1400

U. S. DEPARTMENT OF ENERGY

UNIVERSITY CONTRACTOR, GRANTEE, AND COOPERATIVE AGREEMENT RECOMMENDATIONS FOR ANNOUNCEMENT AND DISTRIBUTION OF DOCUMENTS

See Instructions on Reverse Side

1. DOE Report No.
DOE/ER/25026/34

2. DOE Contract No.
DEFG0287ER25026

3. Title
WAVEFORM METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS

4. Type of Document ("x" one)
- [x] a. Scientific and technical report
- [ ] b. Conference paper:
  - Title of conference _____
  - Date of conference _____
  - Exact location of conference _____
  - Sponsoring organization _____
- [x] c. Other (Specify) Ph.D. Thesis

5. Recommended Announcement and Distribution ("x" one)
- [x] a. Unrestricted unlimited distribution.
- [ ] b. Make available only within DOE and to DOE contractors and other U. S. Government agencies and their contractors.
- [ ] c. Other (Specify) _____

6. Reason for Recommended Restrictions

7. Patent and Copyright Information:
Does this information product disclose any new equipment, process, or material? [x] No [ ] Yes If so, identify page nos. _____
Has an invention disclosure been submitted to DOE covering any aspect of this information product? [x] No [ ] Yes
If so, identify the DOE (or other) disclosure number and to whom the disclosure was submitted.
Are there any patent-related objections to the release of this information product? [x] No [ ] Yes If so, state these objections.
Does this information product contain copyrighted material? [x] No [ ] Yes
If so, identify the page numbers _____ and attach the license or other authority for the government to reproduce.

8. Submitted by — Name and Position (Please print or type)
C. W. Gear, Professor and Principal Investigator

Organization
Department of Computer Science, University of Illinois at Urbana-Champaign

Signature	Phone	Date
C. W. Gear	217/333-0195	January 1990

FOR DOE OR OTHER AUTHORIZED USE ONLY

9. Patent Clearance ("x" one)
- [ ] a. DOE patent clearance has been granted by responsible DOE patent group.
- [ ] b. Report has been sent to responsible DOE patent group for clearance.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-90-1563	2.	3. Recipient's Accession No.

4. Title and Subtitle
WAVEFORM METHODS FOR
ORDINARY DIFFERENTIAL EQUATIONS

5. Report Date
January 1990

6.

7. Author(s)
Fen-Lien Juang

8. Performing Organization Rept. No.

9. Performing Organization Name and Address
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.
DOE DEFG0287ER25026

12. Sponsoring Organization Name and Address
Department of Energy
Chicago Operations Office
Argonne, Illinois 60439

13. Type of Report & Period Covered
Ph.D. Thesis

14.

15. Supplementary Notes

16. Abstracts

The traditional approach for solving large dynamical systems is time consuming. Waveform method, an iterative technique for solving systems of differential equations, can be used to reduce the processing time. Waveform method has been shown to converge superlinearly on finite intervals. In this thesis, a measure of speed of convergence is defined and is used to compare the value of different waveform methods. This measure is the rate of increase of order of accuracy.

The speed of the waveform Gauss-Seidel method depends on the numbering of the equations. The numbering of the equations corresponds to a numbering of the directed graph specifying the coupling relations among all equations. We show how to compute the rate of order increase from the structure of the numbered graph and hence the optimum numbering, that is, the one which maximizes the speed of convergence. Finally, in a variety of numerical experiments, conducted on a Sun 3/60, we demonstrate the different speeds of convergence correspond to different numberings and the effectiveness of the waveform Gauss-Seidel method for large sparse systems.

17. Key Words and Document Analysis. 17a. Descriptors

waveform relaxation
waveform Gauss-Seidel
waveform Jacobi
order of accuracy
accuracy increase

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
unlimited	UNCLASSIFIED	146
	20. Security Class (This Page) UNCLASSIFIED	22. Price