Real-time Distributed Simulation of Partial Differential Equations

Sheng Feng Zhou

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

September 2004

Your file Votre référence
ISBN: 0-612-94735-1
Our file Notre référence
ISBN: 0-612-94735-1

ABSTRACT

Real-time Distributed Simulation of Partial Differential Equations

Sheng Feng Zhou

Recent advances in high speed network technology has allowed clusters of computers to be linked together to form distributed computing networks capable of real-time simulation of large scale complex mechanical systems. Real-time distributed simulation can potentially be used to develop highly detailed virtual prototypes for testing and optimization in automotive, aerospace, and manufacturing industries. Industrial applications, however, have been limited until now due to lack of systematic methods for constructing real-time distributed simulations.

This thesis investigates new methods for real-time distributed simulation of partial differential equations (PDEs). This represents an important class of mechanical systems that in most cases requires multiple computers in order for simulations to proceed in real-time. The focus is on the wave equation since it has well known properties and it is representative of many types of PDE systems. The proposed approach uses an innovative time division multiple access (TDMA) real-time communication protocol based on gigabit Ethernet. Equation distribution and real-time simulation algorithms based on finite-difference approximations are developed for both explicit and implicit integration methods. Together, these results provide a more systematic approach for real-time distributed simulation of PDE systems.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my research advisor, Dr. Brandon W. Gordon, for his kind help, guidance, support and encouragement throughout my study.

I would like to thank Jiang Lu for his explanations of the network driver program. I thank Zhiqian Ren, Farshad Rum and Yunkun Yang for their helps: discussions give me inspirations. I thank Bin Liu for helping me in the network settings. I would like to express my thanks to Bo Zhang (Amilla) for her precious help in correcting this thesis.

Finally I thank my family. Without the support of my family, I could not have accomplished writing this thesis successfully.

Sheng Feng Zhou

Concordia University

Montreal, Canada

Table of Contents

List of Figures

List of Tables

Nomenclature

Δt	Simulation time step
T_c	Computational period
T_{com}	Communication period
T_{send}	Communication time
T_{TDMA}	TDMA period
Δt_c	Computational time
k	Spring stiffness coefficient
m	Mass
x	Position
$\dot{x}$	Derivative of position (speed)
v	Speed
$\dot{v}, \ddot{x}$	Derivative of speed (acceleration)
θ	Angular position
$\dot{\theta}$	Derivative of angular position (angular speed)
ω	Angular speed
$\dot{\omega}, \ddot{\theta}$	Derivative of angular speed (angular acceleration)
c	Wave speed
Lx	Width of wave surface
Ly	Height of wave surface
u	External force

b	Damping coefficient
h	Step size
z	DAE constrained state
J_s	Jacobi matrix
g	DAE constraint
Δt_s	Sliding implicit time step
g_r	Gravity
K	Sliding control parameter
ε	Standard singular perturbation parameter
μ	Dynamic control parameter in DAE
s	Sliding surface
I	Identity matrix

1 Introduction

1.1 Motivation

In the past, complex simulations were implemented by expensive supercomputers. But now, with the development of high-speed network technology, it is possible for standard computers to be linked together economically to form distributed computing networks to perform this task. Real-time distributed simulation has been used extensively in many application areas such as the defence industry, automotive and aerospace applications. However, there are currently few fundamental approaches to guide the development of distributed real-time simulation.

A mechanical system consists of complex interconnections of heterogeneous mechanical models that involve different types of equations with different methods of solution. Simulation of these problems generally requires the solution of large numbers of partial differential equations (PDEs) and ordinary differential equations (ODEs) with algebraic constraints.

This thesis investigates innovative methods for real-time distributed simulation of PDEs, including realistic visualization of distributed simulation results. The performance and scalability of the methods is also studied.

1.2 Literature review

This thesis mainly deals with the following areas: real-time distributed simulation methods, PDE, DAE, and PDAE systems, and real-time distributed simulation implementation. In the following literature review, the three areas indicated above are presented respectively.

1.2.1 Real-time distributed simulation methods

J. B. Roger and S. Robinson give a broad introduction of simulation and general concepts in their book [21]. Explicit Euler method is widely used for simulations because it is easy to implement. However, if the simulation time step becomes too large, the simulation becomes unstable. An improved large time step method known as "implicit Euler" method has also been investigated. D. Baraff and A. Witkin proposed an implicit Euler method in their cloth simulation. They applied Taylor series expansion and made the first order of approximation [2][3]. The simulation time step using this method is much larger than explicit Euler method, but the implementation of this method still has step size problems due to truncation errors, and the implementation of this method can be difficult for general nonlinear systems.

When implementing implicit simulations, we normally use Newton's method to find roots, which proceeds by iteration. When using Newton's method, the root finding procedure may take a short time, but sometimes it may take a long time until the results are obtained.

This form of non-determinism is acceptable for non-real-time simulation, but it cannot be used in real-time simulation because it is difficult to guarantee deterministic execution times required for real-time operation. Currently, no methods have been proposed for real-time implicit simulation.

1.2.2 PDE, DAE and PDAE systems

Many engineering models consist of PDEs. The wave equation is a typical PDE that can be used to represent many engineering systems (such as sound, fluid, vibrations, tissue, virtual reality, etc). It is similar to many PDE problems in engineering, so we use it as an example to investigate our simulation methods. To simulate the wave equation, first of all it must be discretized so that it can be solved numerically. There are several PDE references available such as H. Begehr and A. Jeffrey's [11] and M. Braun's [17].

A differential-algebraic equation (DAE) is a set of differential equations with a set of algebraic constraints [Appendix C: 11]. J.-J. E. Slotine and W. Li gave basic sliding control concepts and sliding surface design for non-linear control systems [7]. B. W. Gordon proposed a singularly perturbed sliding manifold (SPSM) approach [4]. F. Rum and B. W. Gordon applied the SPSM approach in deformable objects simulation (such as cloth simulation) using explicit and implicit Euler method [5].

PDAEs (Partial Differential-Algebraic Equation) are combinations of PDEs with algebraic constraints (DAEs). M. Günther used the PDAE concept in electrical circuit simulation [6]. Y.-I. L. Lim, et al. used a space–time conservation element and solution element method to solve a chemical engineering PDAE problem [20]. Currently there are no published approaches for real-time distributed simulation of PDAEs.

1.2.3 Real-time distributed simulation implementation

Real-time distributed simulation systems have broad applications in automotive, aerospace and industrial automation industries. To implement real-time simulation, several operating systems or software tools can be used, such as QNX [Appendix C: 13], VxWorks [Appendix C: 14] and TDMA protocol [12][Appendix C: 15]. TDMA (Time Division multiplexed Access [12]) being a protocol now is widely used in wireless communication area because of its fairness, simplicity, and it is deterministic. The TDMA real-time Ethernet protocol has several advantages [10]:

- Deterministic response time
- High data rate
- Various hardware available for use
- Fair protocol, no nodes can monopolize the network
- Low cost because the hardware is cheap
- Great potential to upgrade to new Ethernet hardware i.e. 10gigabit Ethernet

J. Lu designed an Ethernet based real-time distributed system and a TDMA protocol in his thesis [10]. Actually it is a set of real-time distributed simulation development tools. His thesis didn't deal with realistic real-time distributed simulation problems.

This thesis uses the TDMA protocol as real-time simulation tool. Several companies are developing real-time simulation products such as Opal-RT Technologies [Appendix C: 6] and Mechanical Simulation Corporation [Appendix C: 12]. Opal-RT Technologies' RT-LAB products provide software, hardware, and related solutions for real-time simulation applications. They use QNX and real-time Linux, Simulink and LabVIEW. Mechanical Simulation Corporation develops CarSim and TruckSim software package for simulating and analyzing the behavior of vehicles. No literatures have been found that use TDMA protocol and VenturCom RTX environment to implement the real-time distributed simulations.

1.3 Thesis outline and contributions

1.3.1 Thesis outline

This thesis consists of six chapters. Chapter 1 describes the motivation for the real-time distributed simulation and contributions of the thesis. Chapter 2 discusses the numerical simulation of PDEs, including ordinary explicit Euler method, implicit Euler method, wave equations and its simulation. Chapter 3 expounds the numerical simulation of PDAEs, including sliding implicit method and simulation of PDAEs (a combination of

DAE and PDE). Chapter 4 deduces the distribution of equations and real-time distributed simulation using TDMA protocol and proposes the distributing method (divide the equations into different computers). Chapter 5 describes simulation results optimization and analysis of the simulation system. Chapter 6 concludes this thesis. Appendix A describes distributed simulation with Winsock. Appendix B describes the visualization and animation of simulation results.

1.3.2 Thesis contributions

1. New method for real-time simulation: sliding implicit method

When implementing the implicit Euler method, we need to use Newton's method to find roots that proceeds by iteration. When solving non-linear algebraic equation problems, the root finding procedure may take a long time until the results are obtained. It is difficult to guarantee real-time operation. The new sliding implicit method proposed in this thesis has deterministic solving time and can guarantee real-time operation. This is a new result that has not been previous developed in current literature.

2. Implementation of TDMA based real-time distributed simulation of PDEs

First of all, this thesis gives one of the most important PDE (the wave equation) a clear numerical discretization solution so that it can easily be implemented in computer simulation. Secondly, the thesis gives an equation distribution algorithm. Finally, the thesis successfully implements the real-time distributed simulation of PDE and PDAE systems using a TDMA based protocol. This result is innovative and one of the first

investigations to implement real-time PDE simulation using this approach.

3. Implementation of real-time distributed simulation of PDEs with algebraic constraints (PDAEs)

It is difficult to evenly distribute the PDAE equation in each computer in real-time distributed simulation. Some computers may have more computational tasks; others may have less computational tasks. When implementing real-time simulations, measures must be taken to keep same time step for all of the computers, to make sure that the simulation results are correct and real-time. This thesis solves the problem successfully by adjust the appropriate simulation parameters.

4. Real-time distributed simulation analysis

Real-time distributed simulation analysis is an important part in real-time distributed simulations. This thesis uses two parameters to analysis the simulation performance: the CPU usage ratio and the network communication capacity. From the calculation of these two parameters, the thesis improves the performance by changing the computational tasks and the communication parameters. The CPU usage ratio can be reached as high as 0.97 for single computer, and 0.73 for four-computer distributed simulation, and the communication capacity is less than the maximum communication capacity. Finally the thesis gives the real-time distributed simulation scalability.

5. Real-time distributed software development

Currently no software package exists that is flexible enough to study these simulation research problems. The software developed in this thesis can be used to investigate new real-time equation distribution and communication protocols. The software is developed in modules (equation computation). When new equations are going to be simulated, they can be easily implemented by changing these modules.

2 Numerical Simulation of PDEs

Numerical solution of the wave equation using finite-difference approximation is investigated in this thesis since it has well known properties and it is representative of many types of PDE systems. In this chapter, wave equations, explicit Euler method, and implicit Euler method will be discussed.

2.1 Wave equation and its discretization

The wave equation is one of the most important PDE equations. The wave equation can be used to represent many engineering systems (such as sound, fluid, vibrations, tissue, virtual reality, etc).

1. 1D Case

The 1D wave equation with damping can be written in the following form:

$$\frac{\partial^2\theta}{\partial t^2} = c^2 \frac{\partial^2\theta}{\partial x^2} - b_1 \frac{\partial\theta}{\partial t} - b_2 \frac{\partial^3\theta}{\partial t \partial x^2} \tag{2.1}$$

where, c is the wave speed (m/s)

b_1 is external damping (1/s)

b_2 is internal damping (m^2/s)

θ is the wave displacement (m)

The domain is discretized using the one-dimensional grid shown in Figure 2-1.

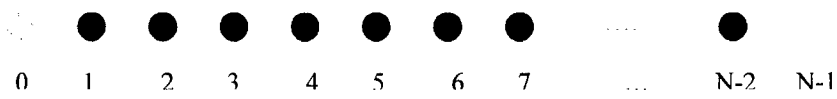

Figure 2-1 1D wave grid

where N is the total number of grid points. Points 0 and (N-1) represent the end point boundary conditions. Using a second order finite difference approximation:

$$\frac{\partial^2\theta_i}{\partial x^2} = \frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{\Delta x^2} \tag{2.2}$$

Apply equation (2.2) to equation (2.1),

$$\frac{d^2\theta_i}{dt^2} = c^2\left(\frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{\Delta x^2}\right) - b_1\frac{d\theta_i}{dt} - b_2\left(\frac{\dot{\theta}_{i+1} - 2\dot{\theta}_i + \dot{\theta}_{i-1}}{\Delta x^2}\right) = \psi_i \tag{2.3}$$

Let $\omega_i = \dot{\theta}_i$

Then

$$\frac{d\omega_i}{dt} = \frac{c^2}{\Delta x^2}\left(\theta_{i+1} - 2\theta_i + \theta_{i-1}\right) - b_1\omega_i - \frac{b_2}{\Delta x^2}\left(\omega_{i+1} - 2\omega_i + \omega_{i-1}\right)$$

$$= \frac{c^2}{\Delta x^2}\theta_{i+1} - \frac{2c^2}{\Delta x^2}\theta_i + \frac{c^2}{\Delta x^2}\theta_{i-1} - b_1\omega_i - \frac{b_2}{\Delta x^2}\omega_{i+1} + \frac{2b_2}{\Delta x^2}\omega_i - \frac{b_2}{\Delta x^2}\omega_{i-1}$$

$$= \frac{c^2}{\Delta x^2}\theta_{i+1} - \frac{2c^2}{\Delta x^2}\theta_i + \frac{c^2}{\Delta x^2}\theta_{i-1} - b_1\omega_i - \frac{b_2}{\Delta x^2}\omega_{i+1} + \frac{2b_2}{\Delta x^2}\omega_i - \frac{b_2}{\Delta x^2}\omega_{i-1}. \tag{2.4}$$

Let

$$a_1 = \frac{c^2}{\Delta x^2}, \quad a_2 = -\frac{b_2}{\Delta x^2}, \quad a_3 = -\frac{2c^2}{\Delta x^2},$$

$$a_4 = -2a_2 - b_1, \quad a_5 = a_1, \quad a_6 = a_2$$

We get:

$$\begin{cases} \dot{\theta}_i = \omega_i \\ \\ \dot{\omega}_i = a_1\theta_{i-1} + a_2\omega_{i-1} + a_3\theta_i + a_4\omega_i + a_5\theta_{i+1} + a_6\omega_{i+1}, i = 1,2,...N-1 \end{cases} \tag{2.5}$$

If an external input is applied to point 0 with boundary point N-1 fixed ($\theta_{N-1} = \omega_{N-1}=0$), then write equation (2.5) in matrix form:

$$\begin{pmatrix} \dot{\theta}_1 \\ \dot{\omega}_1 \\ \dot{\theta}_2 \\ \dot{\omega}_2 \\ \dot{\theta}_3 \\ \dot{\omega}_3 \\ \dot{\theta}_4 \\ \dot{\omega}_4 \\ \vdots \\ \dot{\theta}_{N-2} \\ \dot{\omega}_{N-2} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_3 & a_4 & a_5 & a_6 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & a_1 & a_2 & a_3 & a_4 & \cdots & 0 & 0 \\ & & & & \cdots & \cdots & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & a_3 & a_4 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \omega_1 \\ \theta_2 \\ \omega_2 \\ \theta_3 \\ \omega_3 \\ \theta_4 \\ \omega_4 \\ \vdots \\ \theta_{N-2} \\ \omega_{N-2} \end{pmatrix} + \begin{pmatrix} 0 \\ a_1\theta_0 + a_2\omega_0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

Written in vector form,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{u}, \quad \mathbf{x}, \mathbf{u} \in \Re_{2(N-2)\times 1}, \quad \mathbf{A} \in \Re_{2(N-2)\times 2(N-2)} \tag{2.6}$$

2. 2D Case

The 2D wave equation with damping can be written as follows:

$$\frac{\partial^2\theta}{\partial t^2} = c^2\left(\frac{\partial^2\theta}{\partial x^2} + \frac{\partial^2\theta}{\partial y^2}\right) - b_1\frac{\partial\theta}{\partial t} - b_2\left(\frac{\partial^3\theta}{\partial t\partial x^2} + \frac{\partial^3\theta}{\partial t\partial y^2}\right) \tag{2.7}$$

The domain is discretized using the two-dimensional grid shown in Figure 2-2.

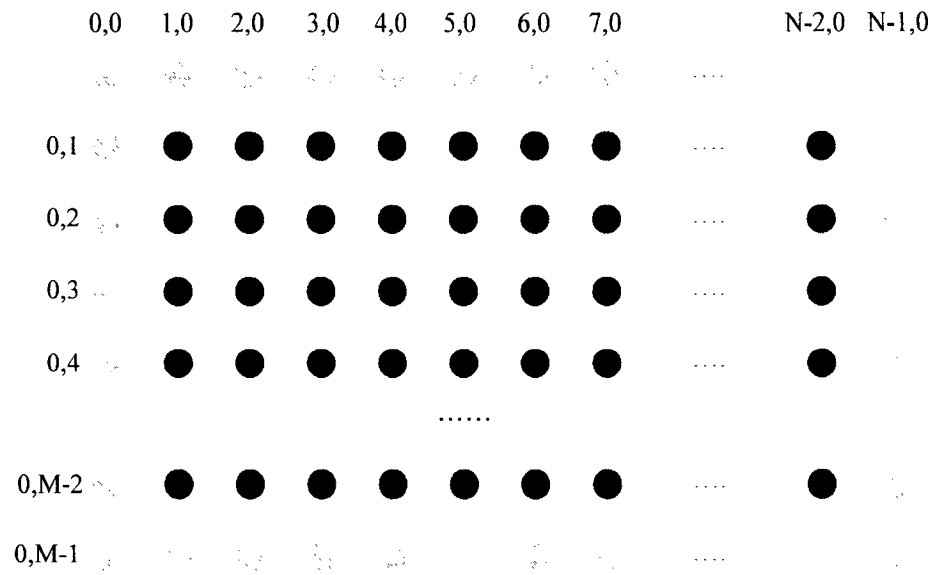

Figure 2-2 2D wave grid

Points (i, 0) (i = 1,2,...N-1) and (0, j) (j = 1,2,...M-1) represent the point boundary conditions. Use second order finite difference approximation:

$$\begin{cases} \dfrac{\partial^2\theta_{ij}}{\partial x^2} = \dfrac{\theta_{i+1,j} - 2\theta_{ij} + \theta_{i-1,j}}{\Delta x^2} \\ \dfrac{\partial^2\theta_{ij}}{\partial y^2} = \dfrac{\theta_{i,j+1} - 2\theta_{ij} + \theta_{i,j-1}}{\Delta y^2} \end{cases} \tag{2.8}$$

Apply (2.8) to (2.7),

$$\frac{d^2\theta_{ij}}{dt^2} = c^2\left(\frac{\theta_{i+1,j} - 2\theta_{ij} + \theta_{i-1,j}}{\Delta x^2} + \frac{\theta_{i,j+1} - 2\theta_{ij} + \theta_{i,j-1}}{\Delta y^2}\right) - b_1\frac{d\theta_{ij}}{dt}$$

$$- b_2\left(\frac{\dot\theta_{i+1,j} - 2\dot\theta_{ij} + \dot\theta_{i-1,j}}{\Delta x^2} + \frac{\dot\theta_{i,j+1} - 2\dot\theta_{ij} + \dot\theta_{i,j-1}}{\Delta y^2}\right) = \psi_{ij} \tag{2.9}$$

Let $\omega_{ij} = \dot\theta_{ij}$

$$\therefore \frac{d\omega_{ij}}{dt} = \frac{c^2}{\Delta x^2}\left(\theta_{i+1,j} - 2\theta_{ij} + \theta_{i-1,j}\right) + \frac{c^2}{\Delta y^2}\left(\theta_{i,j+1} - 2\theta_{ij} + \theta_{i,j-1}\right) - b_1\omega_{ij}$$

$$- \frac{b_2}{\Delta x^2}\left(\omega_{i+1,j} - 2\omega_{ij} + \omega_{i-1,j}\right) - \frac{b_2}{\Delta y^2}\left(\omega_{i,j+1} - 2\omega_{ij} + \omega_{i,j-1}\right)$$

$$= \frac{c^2}{\Delta x^2}\theta_{i-1,j} - \frac{b_2}{\Delta x^2}\omega_{i-1,j} + \frac{c^2}{\Delta y^2}\theta_{i,j-1} - \frac{b_2}{\Delta y^2}\omega_{i,j-1} - \left(\frac{2c^2}{\Delta x^2} + \frac{2c^2}{\Delta y^2}\right)\theta_{ij}$$

$$+ \left(\frac{2b_2}{\Delta x^2} + \frac{2b_2}{\Delta y^2} - b_1\right)\omega_{ij}$$

$$+ \frac{c^2}{\Delta x^2}\theta_{i+1,j} - \frac{b_2}{\Delta x^2}\omega_{i+1,j} + \frac{c^2}{\Delta y^2}\theta_{i,j+1} - \frac{b_2}{\Delta y^2}\omega_{i,j+1} \qquad (2.10)$$

Let:

$$a_1 = \frac{c^2}{\Delta x^2}, \qquad a_2 = -\frac{b_2}{\Delta x^2},$$

$$a_3 = \frac{c^2}{\Delta y^2}, \qquad a_4 = -\frac{b_2}{\Delta y^2},$$

$$a_5 = -2a_1 - 2a_3, \qquad a_6 = -2a_2 - 2a_4 - b_1,$$

$$a_7 = a_1, \qquad a_8 = a_2,$$

$$a_9 = a_3, \qquad a_{10} = a_4$$

We get:

$$\dot{\theta}_{ij} = \omega_{ij}$$

$$\dot{\omega}_{ij} = a_1\theta_{i-1,j} + a_2\omega_{i-1,j} + a_3\theta_{i,j-1} + a_4\omega_{i,j-1} + a_5\theta_{ij} + a_6\omega_{ij} + a_7\theta_{i+1,j} + a_8\omega_{i+1,j}$$

$$+ a_9\theta_{i,j+1} + a_{10}\omega_{i,j+1}, \qquad i = 1,2,...N-2,\ j = 1,2,...M-2 \qquad (2.11)$$

Converting 2D equations to 1D form and applying external input on point (0,1) with

boundary point fixed ($\theta_{i,0} = \theta_{i,M-1} = \theta_{0,j} = \theta_{N-1,j} = 0$), write equation (2.11) in matrix form,

$$\begin{pmatrix} \dot{\theta}_1 \\ \dot{\omega}_1 \\ \dot{\theta}_2 \\ \dot{\omega}_2 \\ \dot{\theta}_3 \\ \dot{\omega}_3 \\ \dot{\theta}_4 \\ \dot{\omega}_4 \\ \vdots \\ \dot{\theta}_n \\ \dot{\omega}_n \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_5 & a_6 & a_7 & a_8 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_1 & a_2 & a_5 & a_6 & a_7 & a_8 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_1 & a_2 & a_5 & a_6 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & a_1 & a_2 & a_5 & a_6 & \cdots & 0 & 0 \\ & & & & \cdots & \cdots & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & 0 & \cdots & a_3 & a_4 & \cdots & 0 & \cdots & a_5 & a_6 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \omega_1 \\ \theta_2 \\ \omega_2 \\ \theta_3 \\ \omega_3 \\ \theta_4 \\ \omega_4 \\ \vdots \\ \theta_n \\ \omega_n \end{pmatrix} + \begin{pmatrix} 0 \\ a_1\theta_{0,1} + a_2\omega_{0,1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \tag{2.12}$$

where n = (M-2)*(N-2).

Writing equation (2.12) in matrix form, we get:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{u}, \quad \mathbf{x}, \mathbf{u} \in \Re_{2n\times 1}, \quad \mathbf{A} \in \Re_{2n\times 2n} \tag{2.13}$$

2.2 Explicit Euler method

Explicit Euler's method is commonly used in solving finite different approximations of PDEs (ODEs) which have the form:

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}) \tag{2.14}$$

Assume $\mathbf{x}_{k+1} = \mathbf{x}(t+\Delta t)$, $\mathbf{x}_k = \mathbf{x}(t)$, Δt is the time increment step, and $\dot{\mathbf{x}}$ is the derivative of $\mathbf{x}$. Then use the first term of the Taylor series to approximate the new state as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{f}(t_k, \mathbf{x}_k)\,\Delta t \tag{2.15}$$

where $\mathbf{f}(t_k, \mathbf{x}_k)$ can be written in the following form:

$$\mathbf{f}(t_k, \mathbf{x}_k) = \dot{\mathbf{x}}_k = \mathbf{A}\mathbf{x}_k + \mathbf{u} \tag{2.16}$$

So,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + (\mathbf{A}\mathbf{x}_k + \mathbf{u})\Delta t$$

$$\mathbf{x}_{k+1} = (\mathbf{I} + \mathbf{A}\,\Delta t)\,\mathbf{x}_k + \mathbf{u}\,\Delta t \tag{2.17}$$

where **I** is an identity matrix.

2.3 Implicit Euler method

Although explicit Euler method is easy to use, it is unstable in solving PDEs for stiff systems. If the step size becomes bigger, the simulation results will become unstable. A widely used method called "implicit Euler' method is often used for improving numerical stability.

To solve the problem that the explicit method has the time increment equation (2.14) is as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{f}(t_{k+1}, \mathbf{x}_{k+1})\,\Delta t \tag{2.18}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + (\mathbf{A}\mathbf{x}_{k+1} + \mathbf{u})\,\Delta t$$

$$\mathbf{x}_{k+1} - \mathbf{A}\,\mathbf{x}_{k+1}\,\Delta t = \mathbf{x}_k + \mathbf{u}\Delta t$$

$$(\mathbf{I} - \mathbf{A}\Delta t)\,\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}\Delta t$$

So,

$$\mathbf{x}_{k+1} = (\mathbf{I} - \mathbf{A}\Delta t)^{-1}(\mathbf{x}_k + \mathbf{u}\Delta t) \tag{2.19}$$

This can be written in the following two forms:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{2.20}$$

or

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.21}$$

To solve this equation, Gauss-Jordan Elimination method, LU Decomposition method, SV Decomposition and Conjugate Gradient method can be used [1].

When using implicit Euler method for non-linear PDEs, equation (2.21) results which needs to be solved using Newton's method for $\mathbf{x}_{k+1}$. Using Newton's method to find roots proceeds by iteration. When solving non-linear algebraic equation problems, the numbers of iteration are unknown. It must keep finding solutions over and over again until the iterations finish. In some special cases, as illustrated in Figure 2-3, the root finding procedure may take a long time until the results are obtained, or may not converge at all [1].

Therefore, Newton's method cannot guarantee real-time execution. This is the main reason why Newton's method is not generally used in real-time simulation. To solve this problem, a new method called sliding implicit method will be proposed in chapter 3.

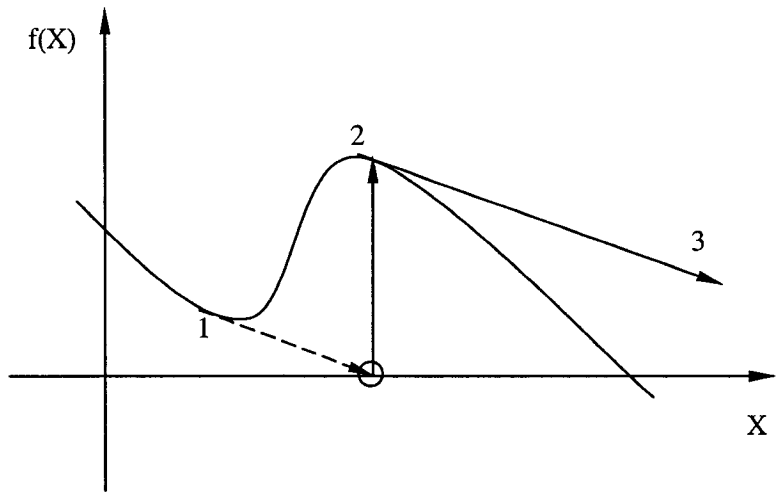

Figure 2-3 Newton's method encounters an extremum and shoots off

In this thesis, these methods are applied as numerical calculations. The comparison of these methods will be discussed in section 2.5; Explicit Euler method, implicit Euler method, and sliding implicit method are used respectively as simulation methods.

Here, take the one-dimensional wave equation as an example to compare the simulation results using explicit Euler and implicit Euler method. Parameters used are listed in Table 2-1.

Table 2-1 Simulation parameters (Implicit/Explicit Euler comparison)

Name	Δt	b_1	b_2	c	ω	N	L
Meaning	Time step	Internal damping	External damping	Wave speed	Angular speed of external input	Number of grids	Length
Value	0.001s	0.5 1/s	0.2 m^2/s	15 m/s	1.5 m/s	20	100m

Figure 2-4 shows the comparison results.

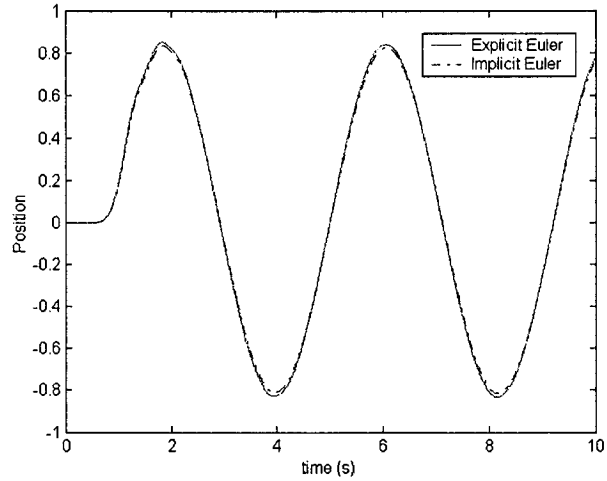

Figure 2-4 Comparison of explicit Euler and implicit Euler methods

The computation time of implicit equation (2.18) is longer than the explicit equation (2.15) for the same Δt. Figure 2-5 shows the computation time with number of grid points of these two equations.

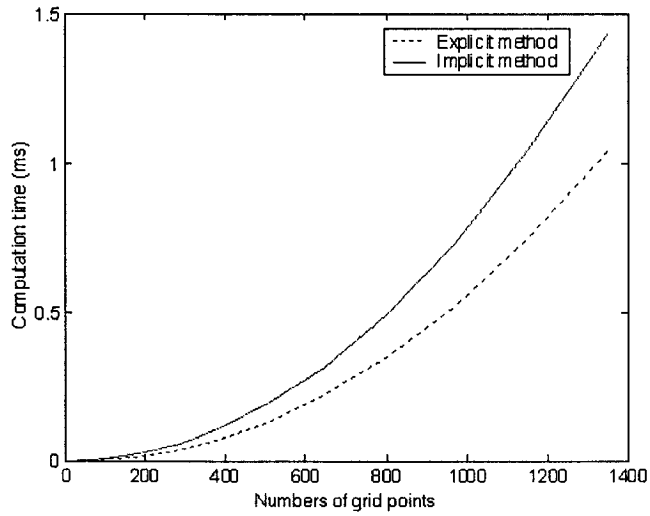

Figure 2-5 Computation time comparison of explicit and implicit method

2.4 Wave equation simulation results

From the equations and the simulation methods discussed above, the wave equation simulation using single computer and multiple computers (distributed) has been implemented. This chapter gives a single computer simulation. Distributed simulation will be discussed in chapter 4. Parameters used in this part are listed in Table 2-2.

Table 2-2 Simulation parameters (Single computer)

Name	Δt	b1	b2	M	N	ω
Meaning	Time step	Internal damping	External damping	Number of grids(Y)	Number of grids(X)	Angular speed of external input
Value	0.001s	0.2 1/s	0.2 m^2/s	22	22	2
Name	Lx	Ly	c			
Meaning	Width of wave surface	Height of wave surface	Wave speed			
Value	100m	100m	10 m/s			

Figure 2-6 shows the two-dimensional wave equation simulation results, the external input is in the upper left point; Figure 2-7 shows the simulation results when the input is in the middle.

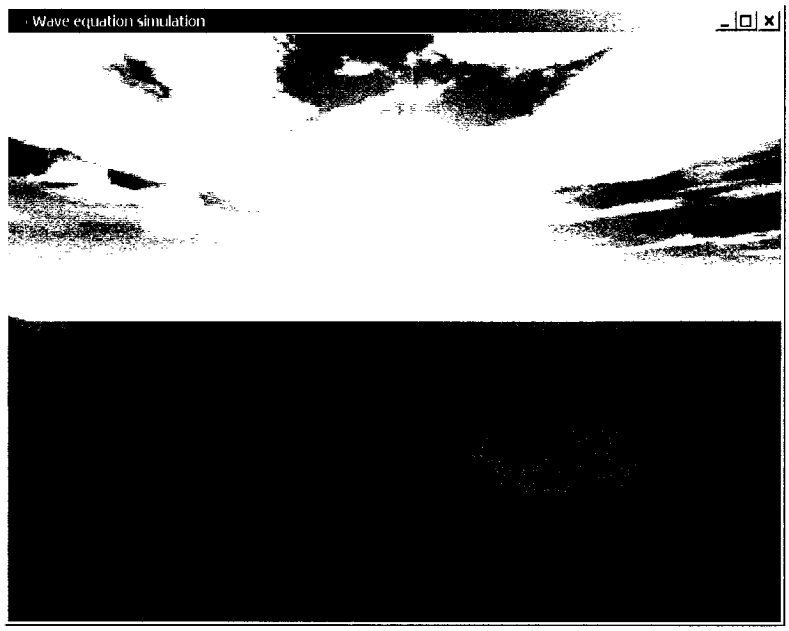

Figure 2-6 Wave equation simulation result (external input is in the upper left point)

Figure 2-7 Wave equation simulation result (external input is in the middle)

2.5 Solution of sparse linear systems

This section discusses the problem of solving linear sparse systems which occurs for implicit and sliding implicit approaches to the wave equation. The performance of this procedure is critical since it can take a significant amount of time for large matrix dimensions. As mentioned above, many simulation problems can be written in the form of **Ax=b** and solved using Gauss-Jordan Elimination, LU Decomposition, SV Decomposition or Conjugate Gradient methods [1].

Gauss-Jordan elimination is efficient in inverting a matrix. For solving sets of linear equations, Gauss-Jordan elimination produces both the solution of the equations for one or more right-hand side vectors b, and also the matrix inverse $\mathbf{A}^{-1}$.

LU decomposition method is convenient to solve matrix inverse $\mathbf{A}^{-1}$ and **Ax=b** equation; conjugate gradient method can be easily used in solving the **Ax=b** equation.

Different methods have different computation efficiency. Here, we take these methods to solve the one-dimensional wave equation as an example to calculate the matrix inverse $\mathbf{A}^{-1}$ using Gauss-Jordan elimination and LU decomposition, and solve the $\tilde{\mathbf{A}}\mathbf{x}=\mathbf{b}$ using LU decomposition and conjugate gradient methods. Figure 2-8 shows the computation time with matrix dimension (sparse matrix). Figure 2-9 shows the structure of matrix **A**.

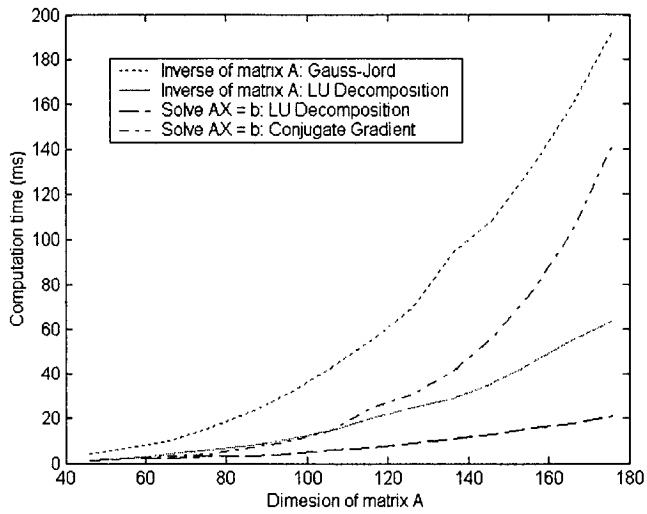

Figure 2-8 Computation time with matrix dimension (sparse matrix)

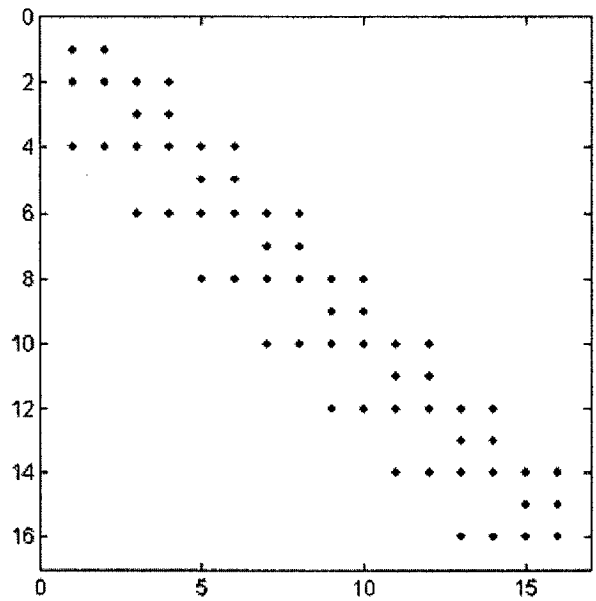

Figure 2-9 The structure of matrix A

Figure 2-10 shows a full number matrix computation time with the matrix dimension. The values of the matrix (**A**) in this example are set to:

A=0.2E+I

where, **E** is matrix of ones

I is identity matrix.

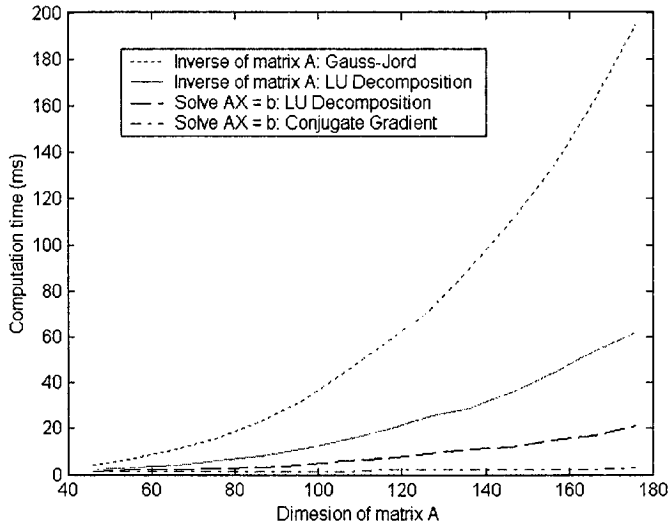

Figure 2-10 Computation time with matrix dimension (full number matrix)

3 Real-time Simulation of DAE and PDAEs

3.1 DAE and PDAE

A Differential-Algebraic Equation (DAE) is a set of differential equations combined with a set of algebraic constraints [Appendix C: 11].

Mechanical systems are often described by a mixed set of partial differential equations (PDEs) and differential algebraic equations (DAEs). These equations can be described in the general form [4]:

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{z}) \tag{3.1}$$

$$\mathbf{0} = \mathbf{g}(t, \mathbf{x}, \mathbf{z}) \tag{3.2}$$

where $\mathbf{x} \in \Re^n$, $\mathbf{z} \in \Re^m$, $\mathbf{f} : \rightarrow \Re^n$, and $\mathbf{g} : \Re \times \Re^n \times \Re^m \rightarrow \Re^m$.

In multi-body mechanical systems, **x** would represent the position and velocity described by momentum equations (3.1), **z** would represent the forces that are determined by kinematic constraints (3.2). In many cases, the constraints are identically singular with respect to **z**.

Real-time distributed simulation of DAEs is complicated by the constraint equations (3.2) which must be solved during each time increment of the simulation. This usually involves an iteration process to solve the nonlinear algebraic constraints. The

constraints will generally involve variables that are updated on many different processors. Therefore, to simultaneously solve the constraint equations communicated must occur between the computer nodes. Further more, the constraints imply instantaneous information transfer between computer nodes since changes in the state variables **x** imply immediate change in the constrained state **z**. Thus, the solution could potentially be very sensitive to communication delays that occur in distributed simulation. In the worst case, all computer nodes would have to be completely synchronized to the slowest node to guarantee integrity of the simulation. Furthermore, iteration cannot be tolerated in real-time simulation because they are not guaranteed to complete in a fixed time interval.

From Gordon's dissertation [4], a new approach using the concept of virtual control based on an important new analogy between DAE system and nonlinear control theory was proposed. The DAE modeling problem formulated as:

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{z}) \tag{3.3}$$

$$\dot{\mathbf{z}} = \mathbf{v} \tag{3.4}$$

$$\mathbf{w} = \mathbf{g}(t, \mathbf{x}, \mathbf{z}) \tag{3.5}$$

where **w** is an output equal to the violation of the constraints and **v** is the virtual control input.

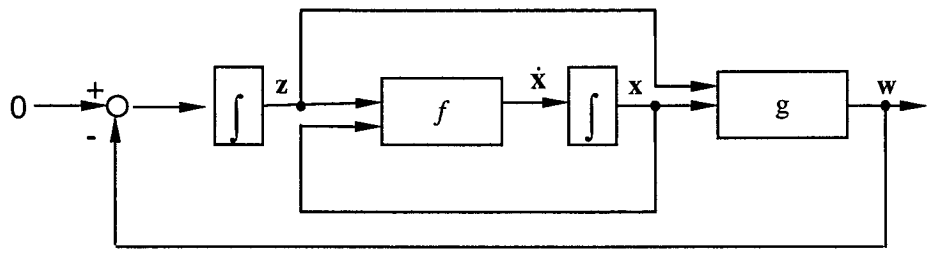

Figure 3-1 DAE realization approach diagram

By developing an appropriate virtual nonlinear controller, the output can be forced to zero that will result in a non-iterative ordinary differential equation model of DAE system, refer to Figure 3-2.

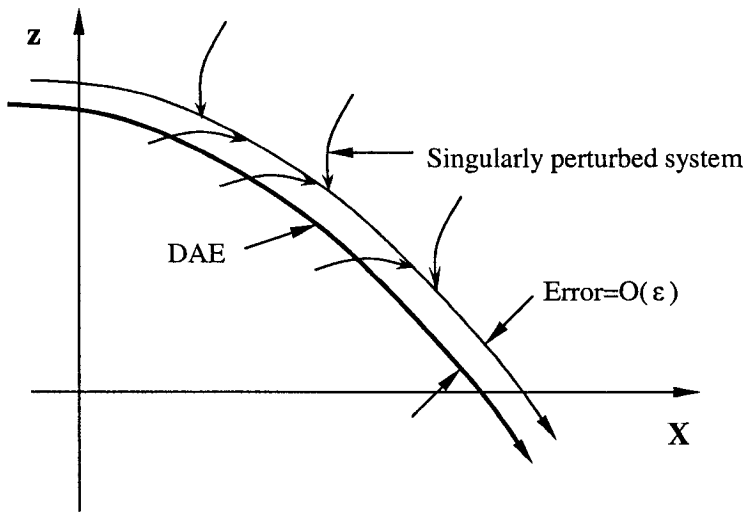

Figure 3-2 The singular perturbation approach

For the vector index to be defined **z** must be explicitly determined by

$$g_i(t,\mathbf{x}) = \mathbf{0} \;,\; \frac{dg_i}{dt}(t,\mathbf{x}) = \mathbf{0},\; \ldots \;,\; \frac{d^{r_i-1}g_i}{dt^{r_i-1}}(t,\mathbf{x},\mathbf{z}) = \mathbf{0} \qquad (3.6)$$

Using the new approach to model, the constraints are applied to a sliding manifold:

$$\mathbf{s}_i = \left[\mu\frac{d}{dt}+1\right]^{r_i-1} g_i, \quad \mu > 0 \tag{3.7}$$

When $\mathbf{s} = \mathbf{0}$, the constraints are asymptotically satisfied with stable values.

3.2 Sliding implicit method

As mentioned in Chapter 2, using Newton's method to solve implicit Euler time steps proceeds by iteration. When solving nonlinear algebraic equation problems, the numbers of iterations are unknown. One must keep finding solutions over and over again until the iterations converge. This cannot guarantee real-time simulation. The sliding implicit method has a deterministic solving time for a specific problem. It can be used to solve real-time problems. The method proceeds by replacing derivatives by finite difference approximations:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{f}(t_{k+1}, \mathbf{x}_{k+1}) \cdot \Delta t \tag{3.8}$$

$$\mathbf{x}_{k+1} - \mathbf{f}(t_{k+1}, \mathbf{x}_{k+1}) \cdot \Delta t = \mathbf{x}_k \tag{3.9}$$

where $\mathbf{x}_{k+1}$ is the new states data, and $\mathbf{x}_k$ is the last states data.

Let $\mathbf{z} = \mathbf{x}_{k+1}$, then

$$\mathbf{z} - \mathbf{f}(t_{k+1}, \mathbf{z}) \cdot \Delta t = \mathbf{x}_k \tag{3.10}$$

The constraints are:

$$\mathbf{g}(\mathbf{z}) = \mathbf{z} - \mathbf{f}(t_{k+1}, \mathbf{z}) \cdot \Delta t - \mathbf{x}_k \tag{3.11}$$

For the wave equation $\mathbf{f}(t_{k+1}, \mathbf{z})$ can be written in the following form:

$$\mathbf{f}(t_{k+1}, \mathbf{z}) = \dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{u} \tag{3.12}$$

Therefore,

$$\mathbf{g}(\mathbf{z}) = \mathbf{z} - (\mathbf{A}\mathbf{z} + \mathbf{u}) \cdot \Delta t - \mathbf{x}_k$$

$$\mathbf{g}(\mathbf{z}) = \mathbf{z} - \mathbf{A}\mathbf{z} \cdot \Delta t - \mathbf{u} \cdot \Delta t - \mathbf{x}_k$$

$$\dot{\mathbf{g}}(\mathbf{z}) = \dot{\mathbf{z}} - \mathbf{A}\dot{\mathbf{z}} \cdot \Delta t - \mathbf{A}\mathbf{z} - \mathbf{u} \tag{3.13}$$

Therefore we take index $r_i = 1$ for the sliding mode in equation (3-8). Thus,

$$\mathbf{s}(\mathbf{z}) = \mathbf{g}(\mathbf{z})$$

$$\dot{\mathbf{s}}(\mathbf{z}) = \dot{\mathbf{g}}(\mathbf{z}) = \dot{\mathbf{z}} - \mathbf{A}\dot{\mathbf{z}} \cdot \Delta t - \mathbf{A}\mathbf{z} - \mathbf{u}$$

So,

$$\dot{\mathbf{s}}(\mathbf{z}) = (\mathbf{I} - \mathbf{A} \cdot \Delta t)\dot{\mathbf{z}} - \mathbf{A}\mathbf{z} - \mathbf{u} \tag{3.14}$$

We get

$$\mathbf{J}_s = \frac{\partial \mathbf{s}}{\partial \mathbf{z}} = \mathbf{I} - \mathbf{A}\Delta t \tag{3.15}$$

and

$$\hat{\alpha} = -\mathbf{A}\mathbf{z} - \mathbf{u} \tag{3.16}$$

Using the singularly perturbed sliding manifold (SPSM) approach:

$$\varepsilon \cdot \dot{\mathbf{z}} = \mathbf{R} \cdot \mathbf{s} \tag{3.17}$$

where

$$\mathbf{R} = -\mathbf{J}_s^{-1}$$

So we get,

$$\mathbf{J}_s \dot{\mathbf{z}} = -\frac{\mathbf{s}}{\varepsilon} \tag{3.18}$$

Equation (3.18) can be written in the following form:

$$\mathbf{Ax=b}$$

where, $\mathbf{x}$ is $\dot{\mathbf{z}}$

$$\mathbf{A=J_s}$$

$$\mathbf{b} = -\frac{\mathbf{s}}{\varepsilon}$$

If $\dot{\mathbf{z}}$ is solved by using matrix inverse method, then equation (3.18) can be written in the following form:

$$\dot{\mathbf{z}} = -\frac{1}{\varepsilon}\mathbf{J}_s^{-1}\cdot\mathbf{s}$$

$$= -\frac{1}{\varepsilon}\left[(\mathbf{I}-\mathbf{A}\cdot\Delta t)^{-1}\cdot(\mathbf{z}-\mathbf{A}\cdot\mathbf{z}\cdot\Delta t-\mathbf{u}\cdot\Delta t-\mathbf{x}_k)\right]$$

$$= -\frac{1}{\varepsilon}\left\{(\mathbf{I}-\mathbf{A}\cdot\Delta t)^{-1}\cdot[(\mathbf{I}-\mathbf{A}\cdot\Delta t)\cdot\mathbf{z}-(\mathbf{u}\cdot\Delta t+\mathbf{x}_k)]\right\}$$

$$= -\frac{1}{\varepsilon}\left[\mathbf{z}-(\mathbf{I}-\mathbf{A}\cdot\Delta t)^{-1}(\mathbf{u}\cdot\Delta t+\mathbf{x}_k)\right]$$

$$= \frac{1}{\varepsilon}\left[(\mathbf{I}-\mathbf{A}\cdot\Delta t)^{-1}(\mathbf{u}\cdot\Delta t+\mathbf{x}_k)-\mathbf{z}\right] \tag{3.19}$$

If ε is very small, $\mathbf{z}$ can be solved using:

$$\mathbf{z} = \mathbf{z} + \dot{\mathbf{z}}\cdot\Delta t_s \tag{3.20}$$

Which converge to $\mathbf{x}_{k+1}$.

Figure 3-3 shows the time steps.

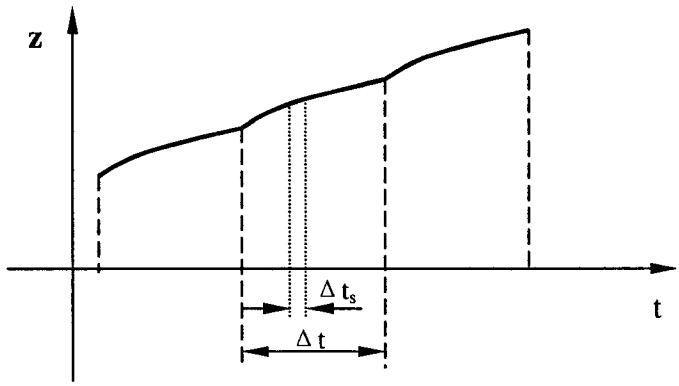

Figure 3-3 Simulation time steps

For singular perturbation approach (equation (3.17)), the ratio of $\frac{\Delta t}{\Delta t_s}$ can be selected as around 10. Take the mass-spring-damper system as an example, as shown in Figure 3-4.

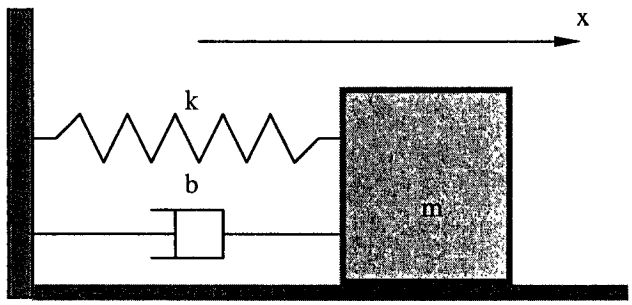

Figure 3-4 A mass-spring-damper system

From Newton's law,

$$ma = -kx - bv \tag{3.21}$$

$$m\ddot{x} = -kx - b\dot{x}$$

Let

$$\mathbf{x} = \begin{pmatrix} x \\ v \end{pmatrix}, \quad \dot{\mathbf{x}} = \begin{pmatrix} \dot{x} \\ \dot{v} \end{pmatrix}$$

Then,

$$\dot{\mathbf{x}} = A\mathbf{x} \tag{3.22}$$

where, $A = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{pmatrix}$

The parameters chosen are listed in Table 3-1.

Table 3-1 Sliding implicit method parameters (Single computer)

Name	m	k	c	t_f	ε	Δt
Meaning	mass	stiffness	Wave speed	Final time	ε parameter	Time step
Value	1 g	1 Ns/m	0.1 m/s	20s	0.0001	5 ms
Name	x_1	v_1				
Meaning	Initial position	Initial speed				
Value	1 m	0 m/s				

The results are shown in Figure 3-5, Figure 3-6, and Figure 3-7.

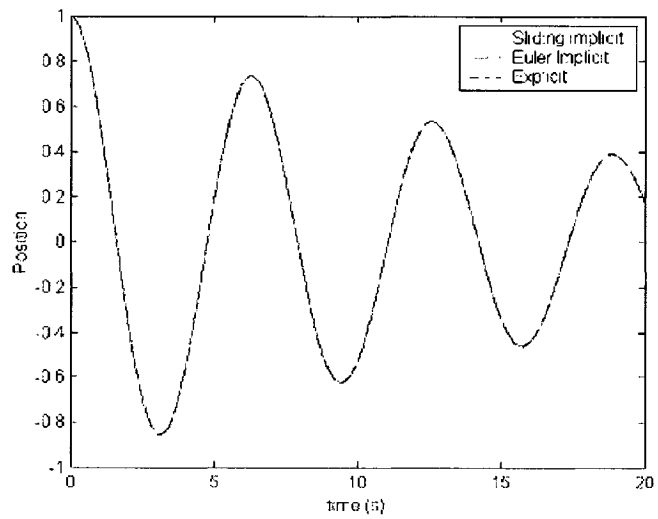

Figure 3-5 Comparison of sliding implicit and Euler method

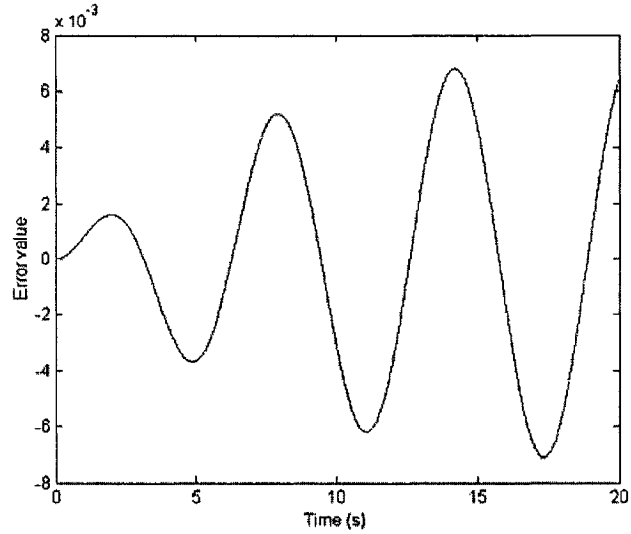

Figure 3-6 Sliding implicit and Euler implicit error

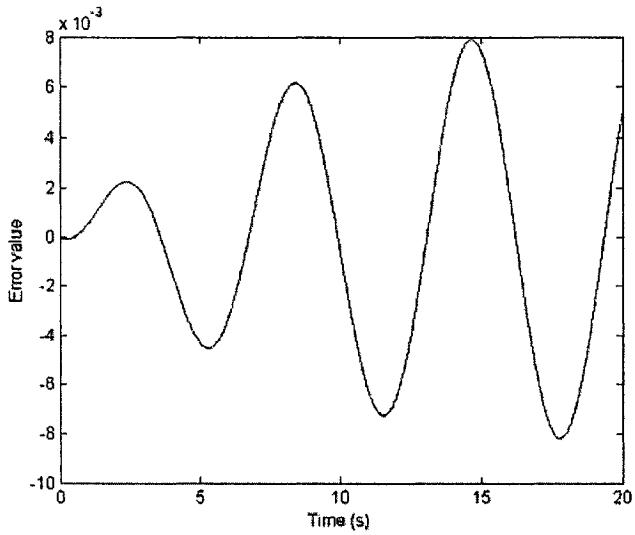

Figure 3-7 Sliding implicit and Euler explicit error

As a special case, if choose $\Delta t = 0.001$, $\varepsilon = 0.0001$, then zero error is obtained. It means that if correct parameters are chosen, the results are the same using sliding implicit and implicit Euler method. For explicit Euler method, when $\Delta t = 0.005$ seconds, It is

unstable, but for sliding implicit, when $\Delta t = 0.16$ seconds, it is still very stable, $\Delta ts = 0.16/10 = 0.016 > 0.005$, as shown in Figure 3-8 and Figure 3-9.

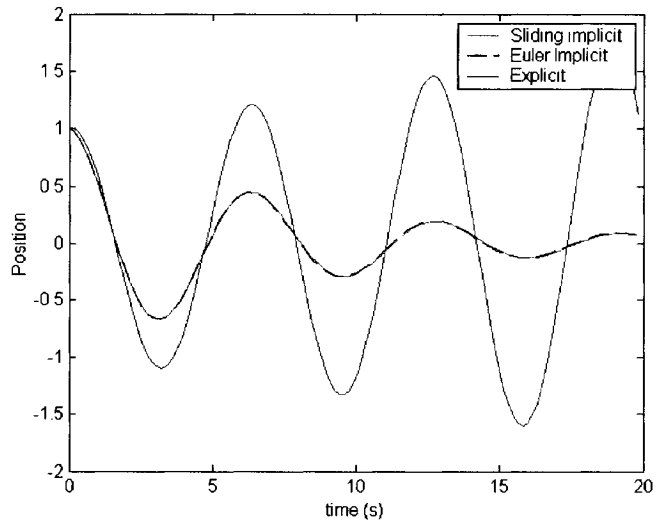

Figure 3-8 Comparison of sliding implicit and Euler method when we increase time step

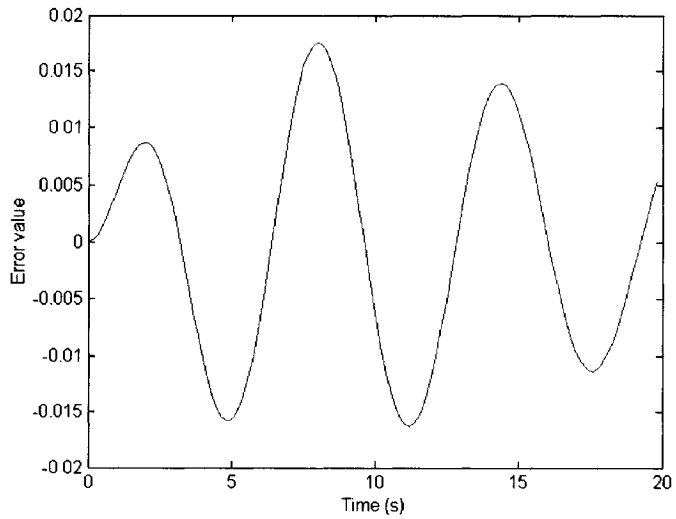

Figure 3-9 Sliding implicit and Euler implicit error

This means that the sliding implicit method is much more stable than the explicit Euler method.

3.3 Simulation of PDAE systems

The problem we study here is how to simulate the motion of an object with an external geometry touching the wave surface. The object can be at any shape. This problem can easily be solved using partial differential-algebraic equations (PDAEs).

1. 1D case

The object touches some of the wave grid points, as illustrated in Figure 3-10. The object can be any shape. Here we take the wedge as an example.

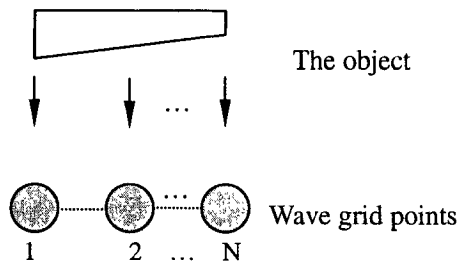

Figure 3-10 1D PDAE

The force diagram of the object is illustrated in Figure 3-11.

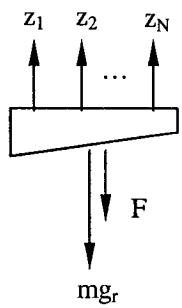

Figure 3-11 1D PDAE force analysis (object)

Using Newton's 2nd law,

$$m\ddot{y} = m \cdot g_r - \sum_{j=1}^{N} z_j + F \tag{3.23}$$

Write equation (3.23) in derivative form,

$$\begin{cases} \dot{v}_y = g_r - \frac{1}{m}\sum_{j=1}^{N} z_j + \frac{F}{m} \\ \dot{y} = v_y \end{cases} \tag{3.24}$$

Here an external force $F = A \cdot \sin(\omega \cdot t)$ is applied.

For the wave points, applying **z** to the wave equation (2.3) gives,

$$\frac{d^2\theta_i}{dt^2} = \psi_i + z_i \tag{3.25}$$

The constrain is as follows:

$$g_i = y - \theta_i \tag{3.27}$$

Derivatives of g:

$$\begin{aligned} \dot{g}_i &= \dot{y} - \dot{\theta}_i \\ &= v_y - \omega_i \end{aligned} \tag{3.28}$$

$$\begin{aligned} \ddot{g}_i &= \ddot{y} - \ddot{\theta}_i \\ &= \dot{v}_y - \dot{\omega}_i \\ &= g_r - \frac{1}{m}\sum_{j=1}^{N} z_j + \frac{F}{m} - (\psi_i + z_i) \end{aligned} \tag{3.29}$$

The index is r = 3:

$$\begin{aligned} \mathbf{s} &= \left(\mu\frac{d}{dt} + 1\right)^{3-1} \mathbf{g} \\ &= \mu^2 \ddot{\mathbf{g}} + 2\mu\dot{\mathbf{g}} + \mathbf{g} \end{aligned} \tag{3.30}$$

So,

$$\mathbf{J}_s = \frac{\partial \mathbf{s}}{\partial \mathbf{z}}$$

$$= -\mu^2 \cdot \mathbf{A} \tag{3.31}$$

Where, $\mathbf{A} = \begin{pmatrix} \frac{m+1}{m} & \frac{1}{m} & \cdots & \frac{1}{m} \\ & & \cdots & \\ \frac{1}{m} & \frac{m+1}{m} & \cdots & \frac{1}{m} \\ \vdots & \vdots & & \vdots \\ \frac{1}{m} & \frac{1}{m} & \cdots & \frac{m+1}{m} \end{pmatrix}$

$$\mathbf{s}, \mathbf{z} \in \Re_{N\times 1}, \quad \mathbf{A}, \mathbf{J}_s \in \Re_{N\times N}$$

Use SPSM approach, the equation is:

$$\dot{\mathbf{z}} = -\mathbf{J}_s^{-1} \cdot \frac{\mathbf{s}}{\varepsilon} \tag{3.32}$$

2. 2D case

2D case PDAE is shown in Figure 3-12 and Figure 3-14. Suppose the object touches IM and IN wave grids.

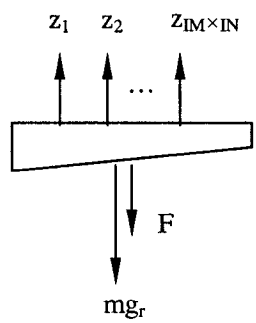

Figure 3-12 2D PDAE force analysis (object)

Use Newton's 2nd law,

$$m\ddot{y} = m \cdot g_r - \sum_{i=1}^{IN}\sum_{j=1}^{IM} z_{ij} + F \tag{3.33}$$

Write in derivative form,

$$\begin{cases} \dot{v}_y = g_r - \dfrac{1}{m}\sum_{i=1}^{IN}\sum_{j=1}^{IM} z_{ij} + \dfrac{F}{m} \\ \dot{y} = v_y \end{cases} \tag{3.34}$$

Here an external force $F = A \cdot \sin(\omega \cdot t)$ is applied.

For the wave points, applying **z** to the wave equation (2.9) gives,

$$\frac{d^2\theta_{ij}}{dt^2} = \psi_{ij} + z_{ij} \tag{3.35}$$

The constraint condition is as follows:

$$g_{ij} = y - u_{ij} \tag{3.36}$$

Derivatives of g:

$$\begin{aligned} \dot{g}_{ij} &= \dot{y} - \dot{u}_{ij} \\ &= v_y - w_{ij} \end{aligned} \tag{3.37}$$

$$\begin{aligned} \ddot{g}_{ij} &= \ddot{y} - \ddot{u}_{ij} \\ &= \dot{v}_y - \dot{w}_{ij} \\ &= g_r - \frac{1}{m}\sum_{i=1}^{IN}\sum_{j=1}^{IM} z_{ij} + \frac{F}{m} - \left(\psi_{ij} + z_{ij}\right) \end{aligned} \tag{3.38}$$

The index is r = 3:

$$s_{ij} = \left(\mu \frac{d}{dt} + 1\right)^{3-1} g_{ij}$$

$$= \mu^2 \ddot{g}_{ij} + 2\mu \dot{g}_{ij} + g_{ij} \tag{3.39}$$

Converting to one-dimensional form and written in vector form (k=(i-1)*IM+j):

So,

$$\mathbf{J}_s = \frac{\partial \mathbf{s}}{\partial \mathbf{z}}$$

$$= -\mu^2 \mathbf{A} \tag{3.40}$$

$$\text{where, } \mathbf{A} = \begin{pmatrix} \frac{m+1}{m} & \frac{1}{m} & \cdots & \frac{1}{m} \\ \cdots & & & \\ \frac{1}{m} & \frac{m+1}{m} & \cdots & \frac{1}{m} \\ \vdots & \vdots & & \vdots \\ \frac{1}{m} & \frac{1}{m} & \cdots & \frac{m+1}{m} \end{pmatrix}$$

$$\mathbf{s}, \mathbf{z} \in \Re_{(IM \times IN) \times 1}, \quad \mathbf{A}, \mathbf{J}_s \in \Re_{(IM \times IN) \times (IM \times IN)}$$

Using the SPSM approach, the equation is:

$$\dot{\mathbf{z}} = -\mathbf{J}_s^{-1} \cdot \frac{\mathbf{s}}{\varepsilon} \tag{3.41}$$

Using the following parameters (list in Table 3-2), the simulation results shows in Figure 3-13 (using wedge shape object).

Table 3-2 PDAE simulation parameters

Name	g_r	m	ε	μ	ω	N
Meaning	Gravity	Object mass	From equation	From equation	Angular speed of external input	Number of wave grids(X)
Value	9.8 m/s^2	0.003 g	0.1	0.2	2	22
Name	Lx	Ly	c	b1	b2	M
Meaning	Width of wave surface	Height of wave surface	Wave speed	Internal damping	External damping	Number of wave grids(Y)
Value	100m	100m	10 m/s	0.1 1/s	0.1 m^2/s	22

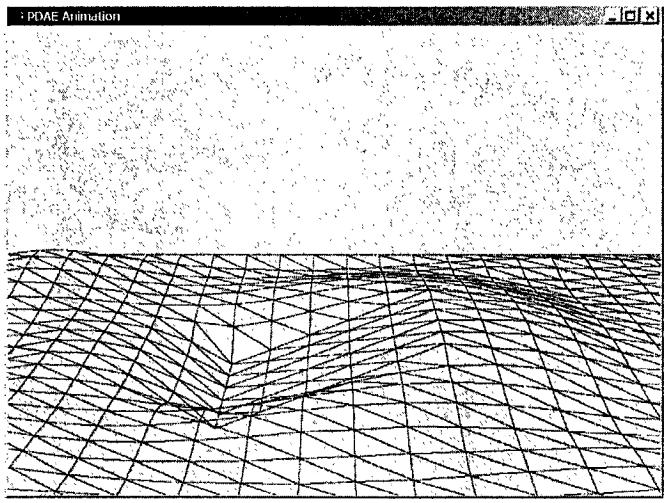

Figure 3-13 PDAE simulation result

4 Real-time Distributed Simulation of PDEs

4.1 Equation distribution algorithm

This section discusses the 2D wave equation distribution algorithm. After the equations are distributed to each computer in the network, each computer can only compute its own part.

4.1.1 One-dimensional systems

The one-dimensional equation distribution can be horizontal or vertical as illustrated in Figure 4-1 and Figure 4-2.

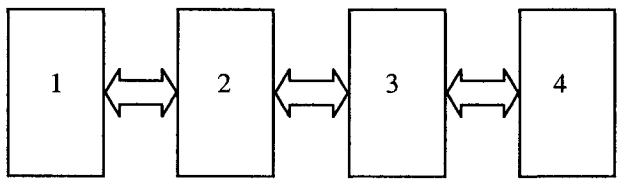

Figure 4-1 1D horizontal distribution

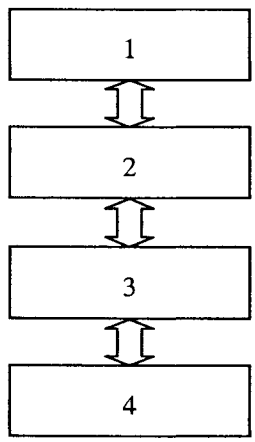

Figure 4-2 1D vertical distribution

The computation task is divided into N parts horizontally or vertically. Each part is calculated in one computer.

Figure 4-1 is a four parts horizontal distribution example. The arrows means that boundary values need to be sent between the parts. The boundaries are the left and right edge; Figure 4-2 is a vertical distribution. The boundaries are the upper and bottom edge.

Here, take vertical distribution to discuss. Part 3 in Figure 4-2 is a representative one. Its grid points are shown in Figure 4-3.

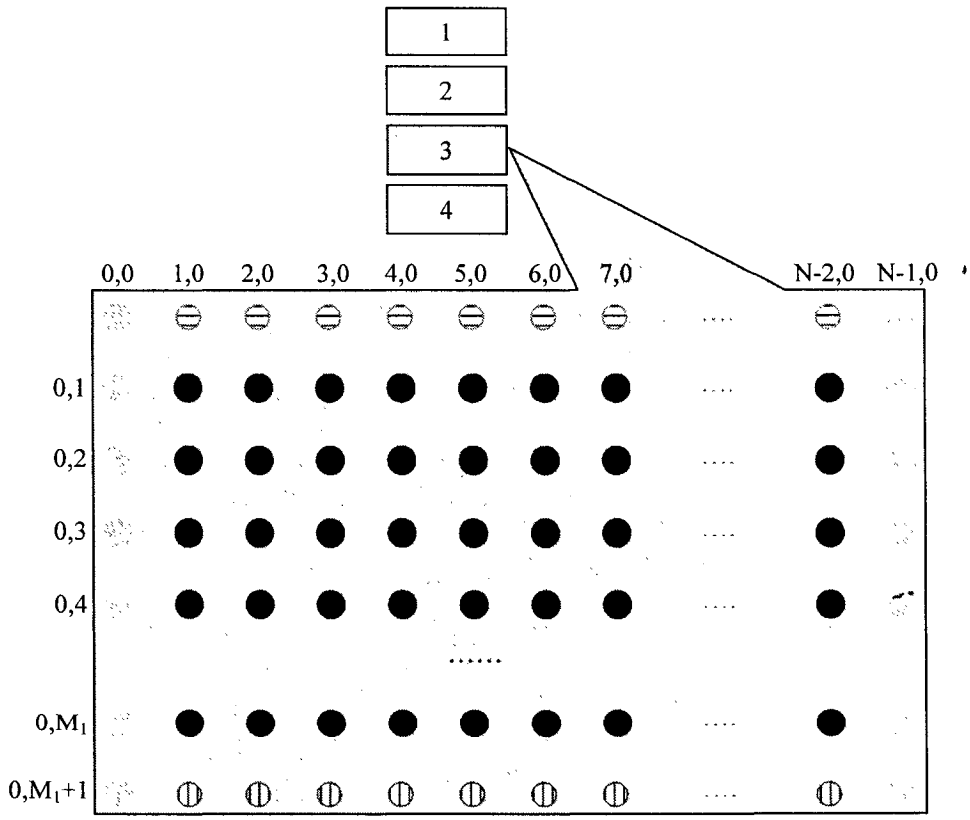

Figure 4-3 1D equation distribution with boundary points

where,

M is total vertical points

N is total horizontal

$M_1 = (M-2)/pn$ where, pn is the number of parts

In this part, upper boundary grids ( ) will be obtained from upper part (part 2 in this figure); bottom boundary grids ( ) will be obtained from bottom part (part 4 in this figure).

For top part (part 1 in this figure), upper boundary grids ( ) have:

$\theta_{i,0} = \omega_{i,0} = 0, i = 1...N-1$

For bottom part (part 4 in this figure), bottom boundary grids ( ) have:

$\theta_{i,M_1+1} = \omega_{i,M_1+1} = 0, i = 1...N-1$

4.1.2 Two-dimensional systems

In one-dimensional distribution, there is only one boundary or two boundary data need to communicate with their neighbours, while the two-dimensional case may have up to four boundary data to communicate with their neighbours. Figure 4-4 is a 9-part distribution case.

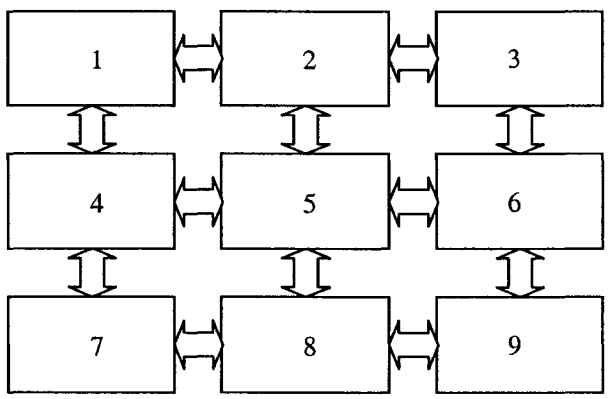

Figure 4-4 2D equation distribution

Here, part 5 in Figure 4-4 is a representative part. Its grid points are shown in Figure 4-5.

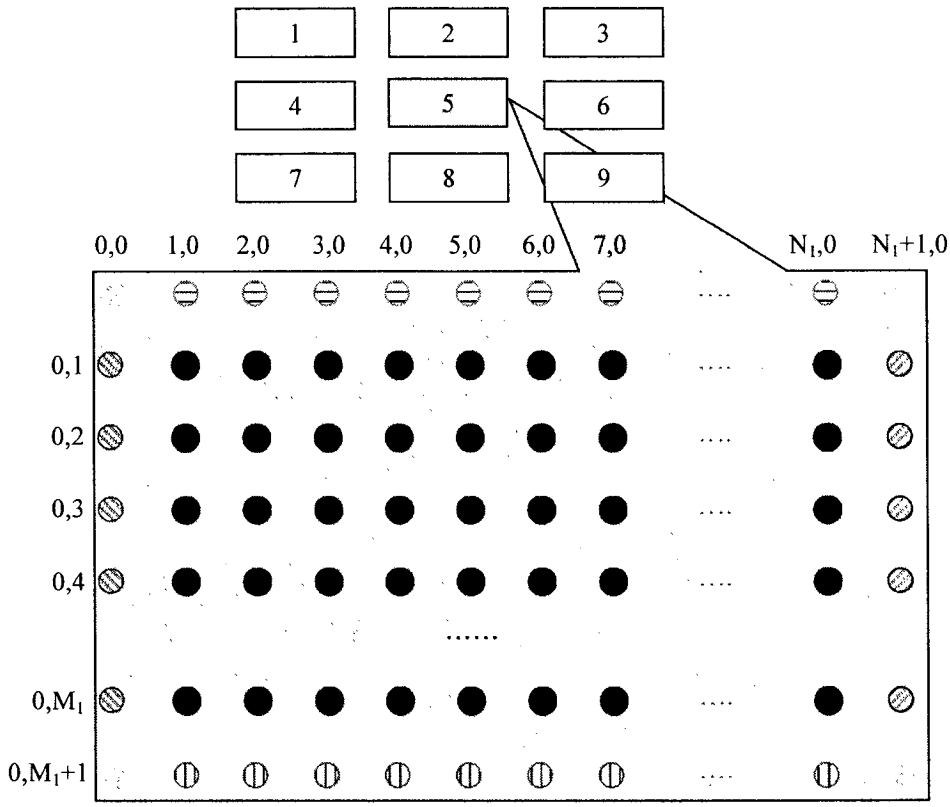

Figure 4-5 2D equation distribution with boundary points

where, $M_1 = (M-2)/Vpn$

$N_1 = (N-2)/Hpn$

where, Vpn is the number of vertical parts

Hpn is the number of horizontal parts

In this part, upper boundary grids ( ) will be obtained from upper part (part 2 in this figure); bottom boundary grids ( ) will be obtained from bottom part (part 8 in this figure); left boundary grids ( ) will be obtained from left part (part 4 in this figure);

right boundary grids ( ) will be obtained from right part (part 6 in this figure). For other parts, their boundary conditions are illustrated in Table 4-1.

Table 4-1 Boundary conditions

Part 1	Part 2	Part 3
$\theta_{i,0} = \omega_{i,0} = 0$ $\theta_{0,j} = \omega_{0,j} = 0$	$\theta_{i,0} = \omega_{i,0} = 0$	$\theta_{i,0} = \omega_{i,0} = 0$ $\theta_{N_1+1,j} = \omega_{N_1+1,j} = 0$
Part 4	Part 5	Part 6
$\theta_{0,j} = \omega_{0,j} = 0$		$\theta_{N_1+1,j} = \omega_{N_1+1,j} = 0$
Part 7	Part 8	Part 9
$\theta_{0,j} = \omega_{0,j} = 0$ $\theta_{i,M_1+1} = \omega_{i,M_1+1} = 0$	$\theta_{i,M_1+1} = \omega_{i,M_1+1} = 0$	$\theta_{N_1+1,j} = \omega_{N_1+1,j} = 0$ $\theta_{i,M_1+1} = \omega_{i,M_1+1} = 0$
$i=1...N_1, \quad j=1..M_1$		

4.2 Real-time simulation using TDMA communication

4.2.1 Real-time TDMA communication protocol

The TDMA (Time Division Multiplexed Access) protocol is a protocol based on time slots. This protocol cooperates with other nodes to guarantee that while its message is transmitting, there are no other nodes transmitting message on the network. By this method, Ethernet will send the message on the network without collision at anytime. Figure 4-6 shows the comparison of TCP/UDP/IP protocol and the TDMA protocol.

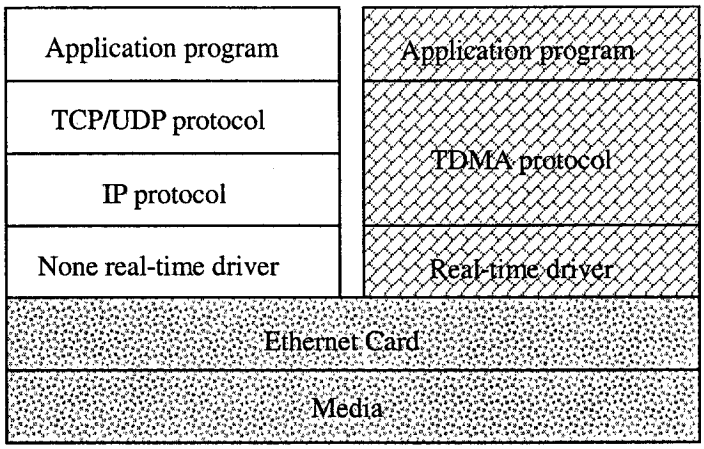

Figure 4-6 TCP/IP and TDMA protocol

The clocks on different computers have drift. All the computer nodes have to be synchronized to one global clock. A synchronization signal is used on the Ethernet Network. The master computer sends the synchronization signal. After the slaves receive the signal, the simulation begins. This TDMA protocol is implemented under Venturcom RTX environment.

Venturcom RTX (Real-Time Extension) operating system is a Windows NT, Windows 2000, and Windows XP extension. It adds real-time properties to the Windows operating systems, and provides an ability to use a single, low-cost platform to satisfy a full range of real-time and embedded application requirements. RTX adds a real-time subsystem RTSS to windows operating system. Figure 4-7 illustrates the message communication path between two computer nodes over the network. Figure 4-8 shows the computer connection diagram. Switch 1 connects fast Ethernet cards that used to control these computers by users.

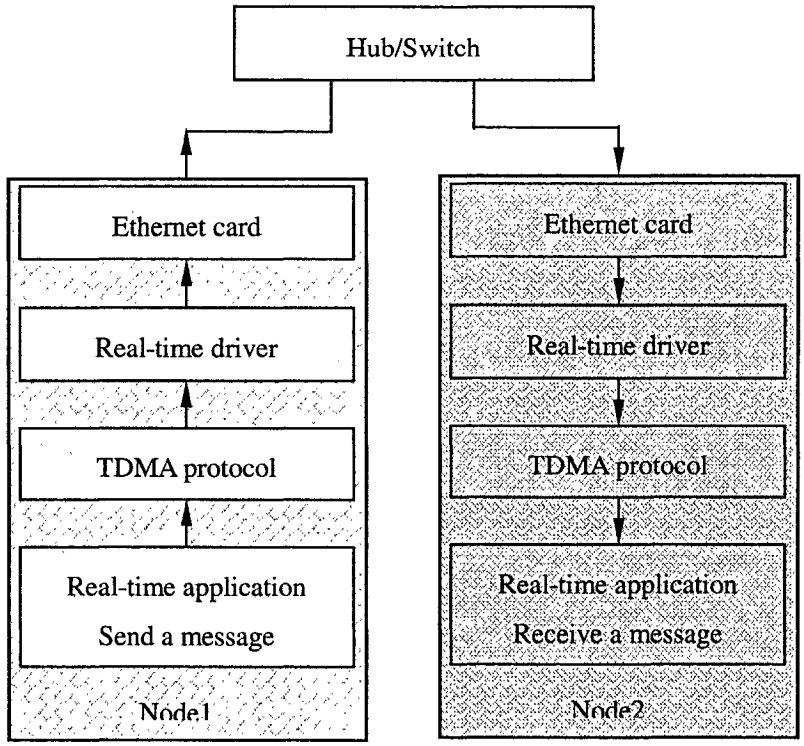

Figure 4-7 Information path

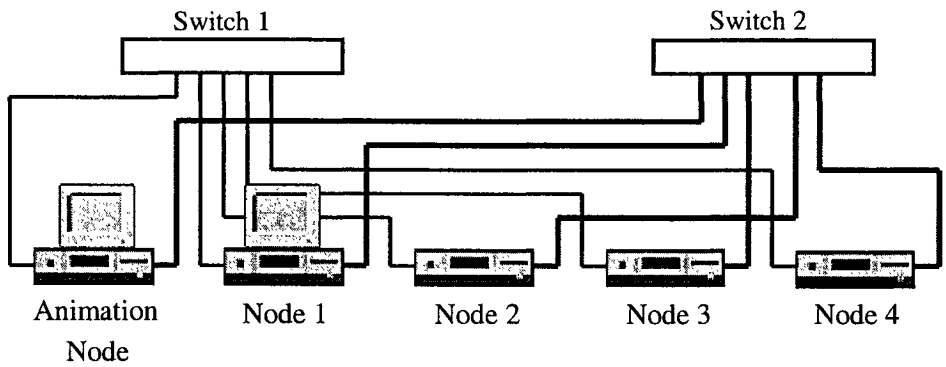

Figure 4-8 Computer connection using TDMA

At the beginning, all the slave computer nodes are waiting for synchronization signal. After the master computer node sends a synchronization signal, the network simulation begins. Then computers send and receive data by the clock time. When the slot time is arrived, the computer sends the packet. Just several microseconds before finish this

synchronization cycle, the computer starts to wait the synchronization signal again. Figure 4-9 shows TDMA protocol synchronized by signal.

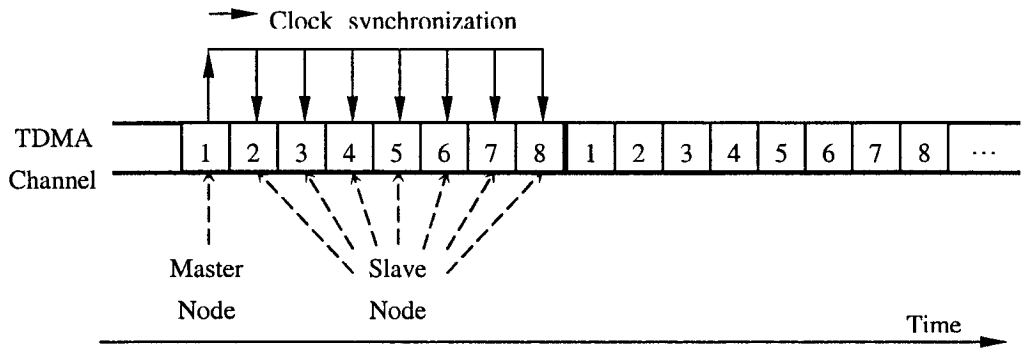

Figure 4-9 Synchronization for each computer

Figure 4-10 shows the computational diagram using TDMA communication. At each time step, computers get data from last time step. If the computational time is different from computer nodes, then set different waiting (sleep) time to synchronize them.

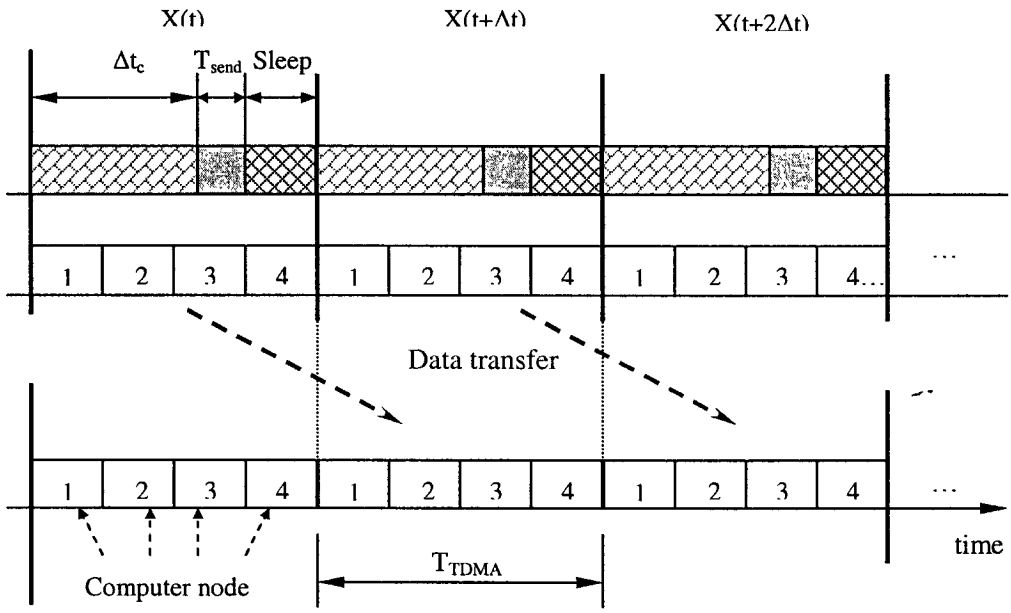

Figure 4-10 Simulation time

Figure 4-11 shows the computational diagram of two-computer nodes calculation time is different.

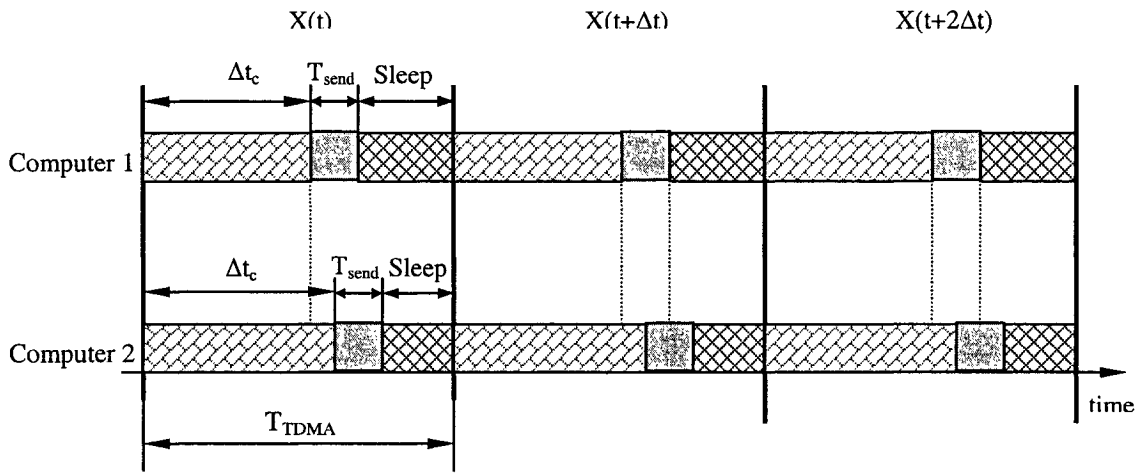

Figure 4-11 Computation time difference for each computer

4.2.2 Real-time distributed simulation

1. Wave equation with explicit Euler method

For the typical PDE wave equation, the computation can be evenly distributed to each of the computer nodes. The distributed algorithm used in the thesis is illustrated in Figure 4-12. Each of the four computer nodes needs to communicate boundary data with two neighbour computers.

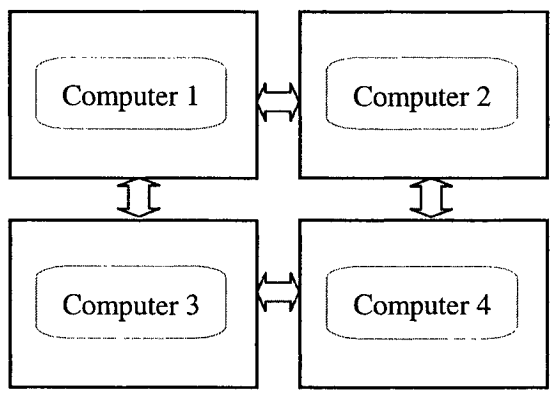

Figure 4-12 PDE distributed algorithm

In this Chapter, the simulation parameters are listed in Table 4-1.

Table 4-1 Simulation parameters

Name	Δt	T_{com}	T_c	N_{slot}	Slot Size	N
Meaning	Time step	Communication period	Computational period	Slot number		Number of grids(X)
Value	1 ms	4 ms	4.2 ms	8	0.5 ms	20
Name	Lx	Ly	c	b1	b2	M
Meaning	Width of wave surface	Height of wave surface	Wave speed	Internal damping	External damping	Number of grids(Y)
Value	100m	100m	10 m/s	0.2 1/s	0.2 m^2/s	20

Figure 4-13 and Figure 4-14 shows single computer and multiple computer (distributed) simulation comparison results (position) of point (3,2). The error is the results of communication delay.

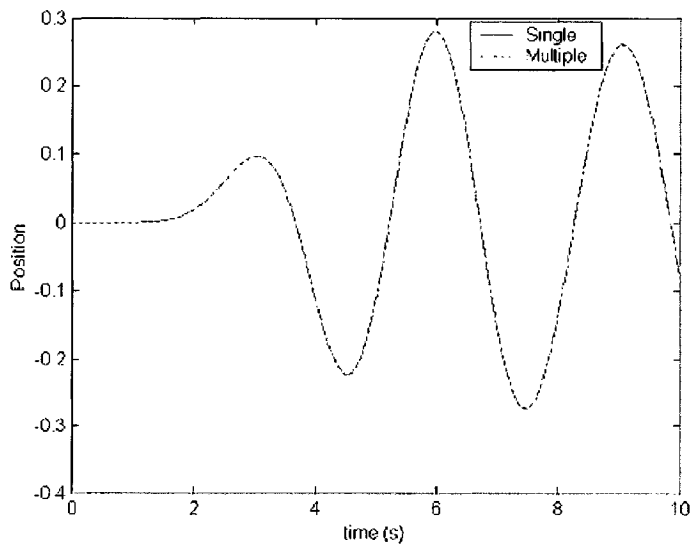

Figure 4-13 Comparison of single computer and distributed results: point (3,2)

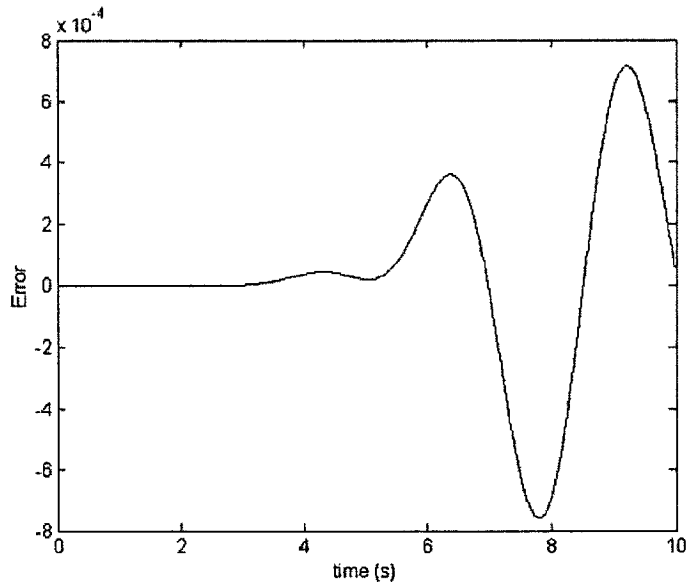

Figure 4-14 Single computer and distributed result error

Figure 4-15 and Figure 4-16 shows comparison results of point (3,4).

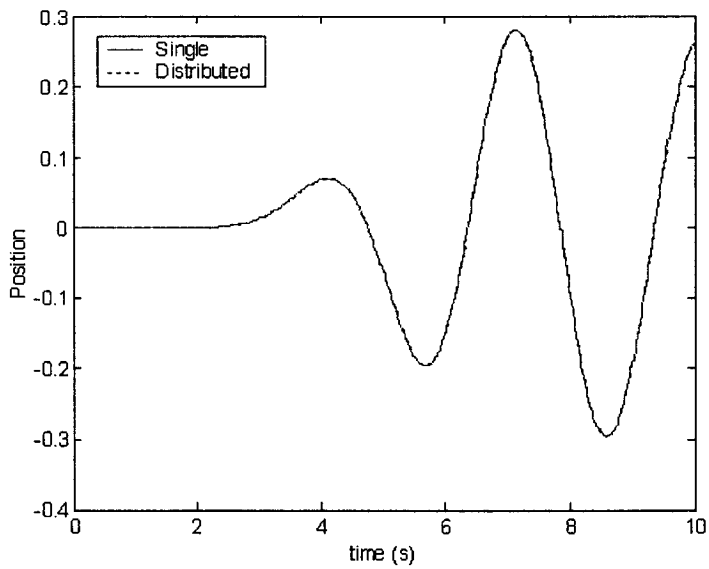

Figure 4-15 Comparison of single computer and distributed results: point (3,4)

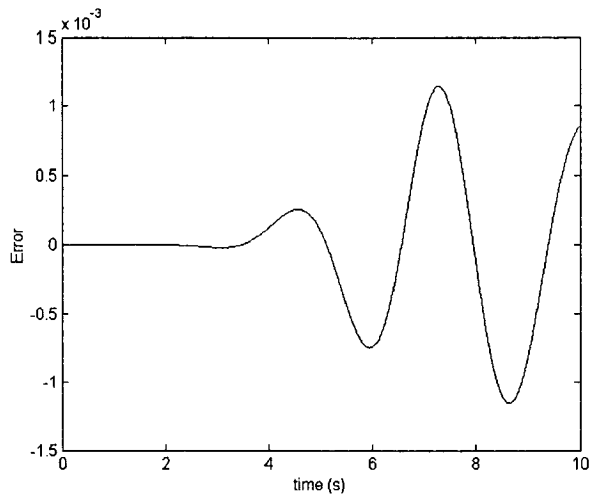

Figure 4-16 Single and distributed error

2. Wave equation with sliding implicit method

The sliding implicit method can guarantee deterministic execution, so it is a good choice in real-time simulation. The distributed algorithm is illustrated in Figure 4-12.

Figure 4-17 and Figure 4-18 shows single computer and distributed simulation comparison results of point (3,2).

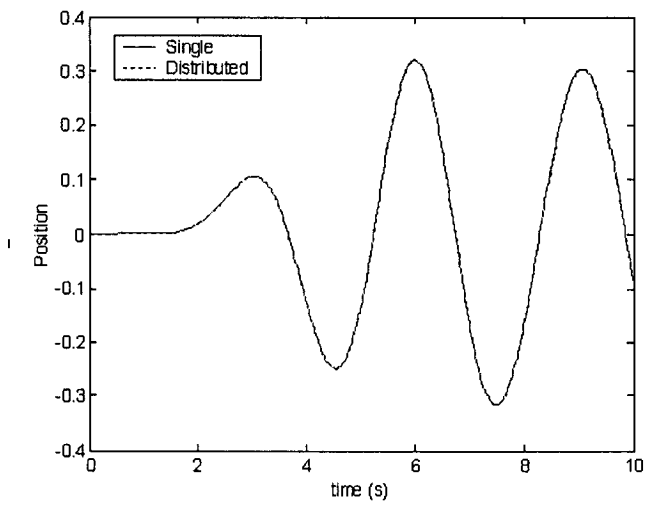

Figure 4-17 Comparison of single computer and distributed results: point (3,2)

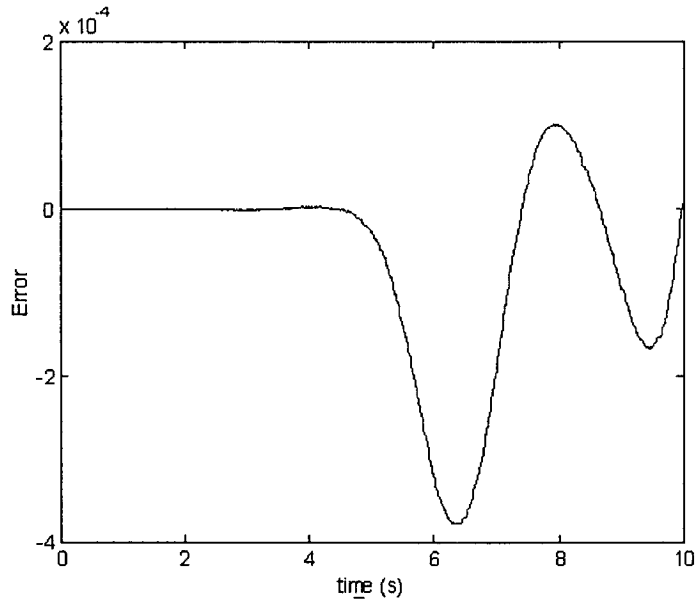

Figure 4-18 Single and distributed error

Figure 4-19 and Figure 4-20 shows comparison results of point (3,4).

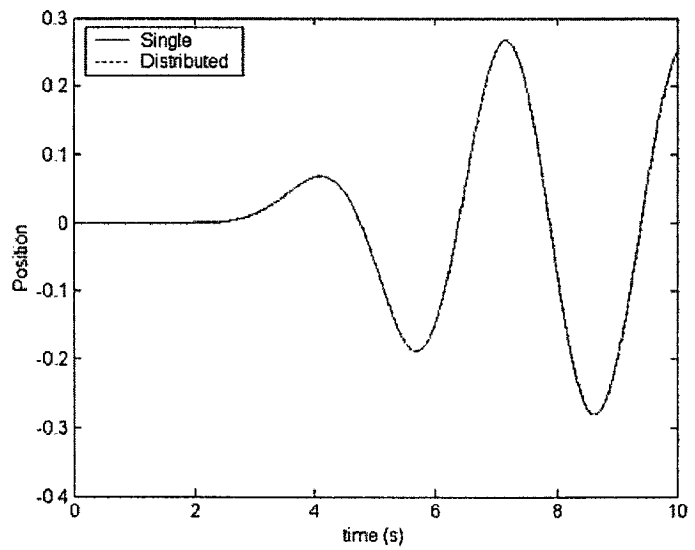

Figure 4-19 Comparison of single computer and distributed results: point (3,4)

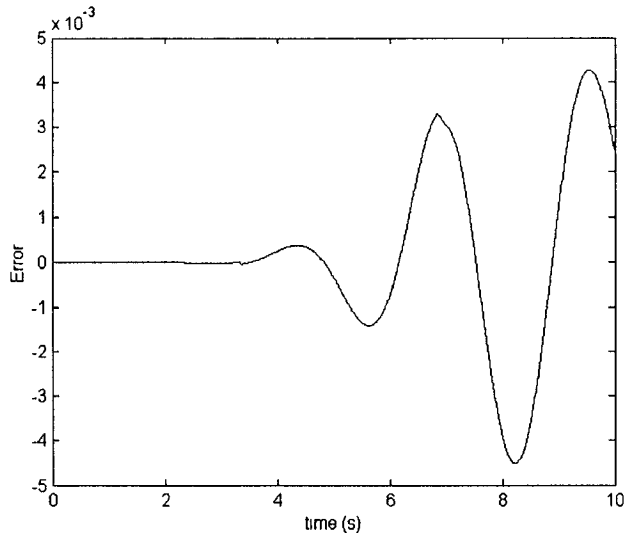

Figure 4-20 Single and distributed error

3. Real-time distributed implementation of the PDAE system

The PDAE consists of a DAE part and a PDE part. This system has been investigated and analysed in section 3.3. This section will describe the distributed simulation implementation of the PDAE system. The computation cannot be evenly distributed to each of the computer nodes. The distributed algorithm used in the thesis is illustrated in Figure 4-21. One computer calculates the DAE part; other three computers calculate the PDE part.

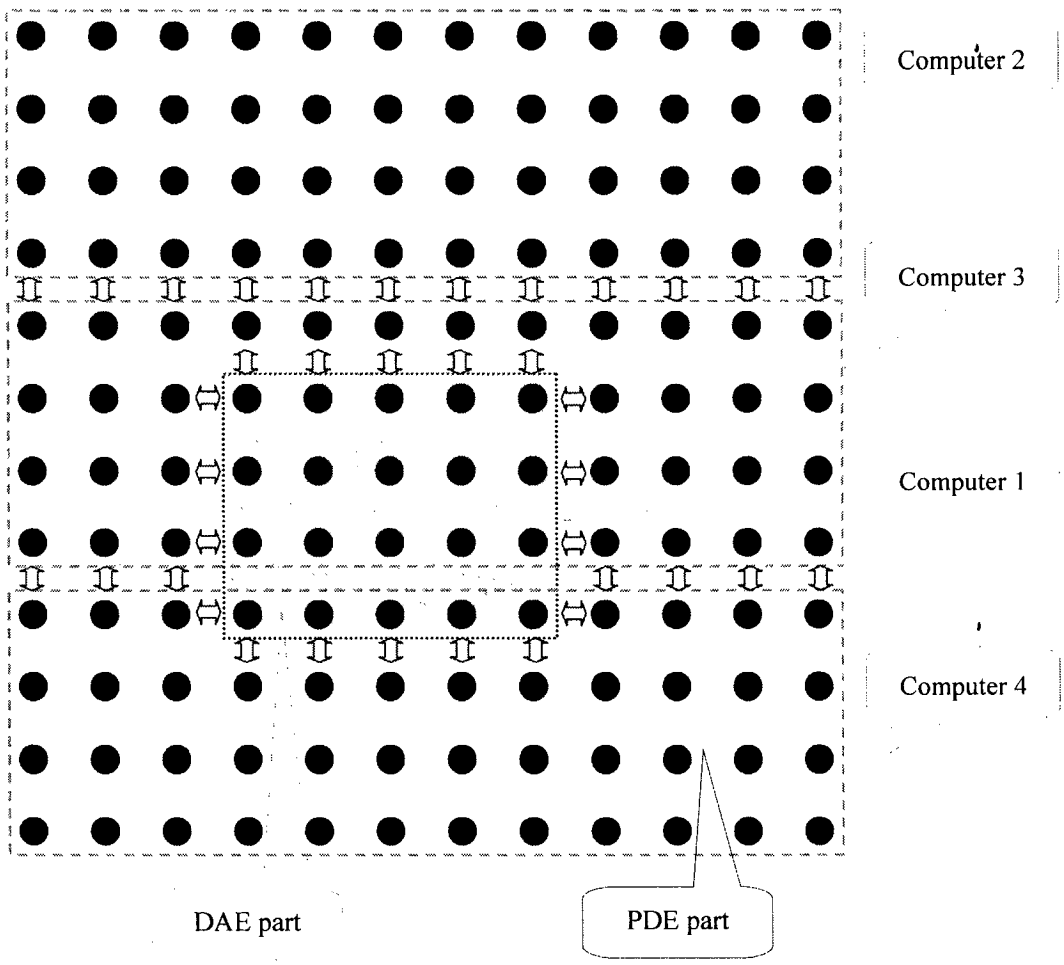

Figure 4-21 PDAE distribution simulation algorithm

Figure 4-22 shows single computer and distributed simulation comparison for point (8,6).

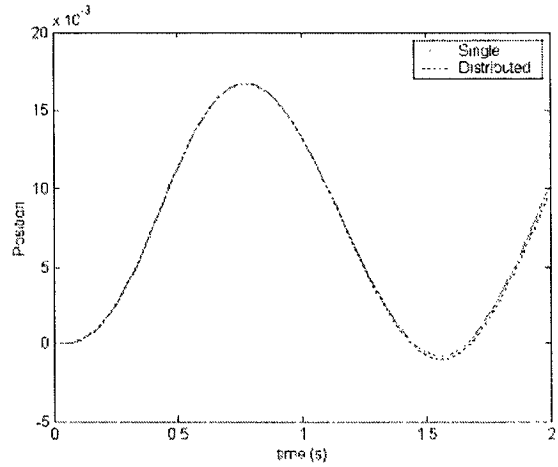

Figure 4-22 Comparison of single computer and distributed results: point (8,6)

Figure 4-23 shows comparison results of point (9,7).

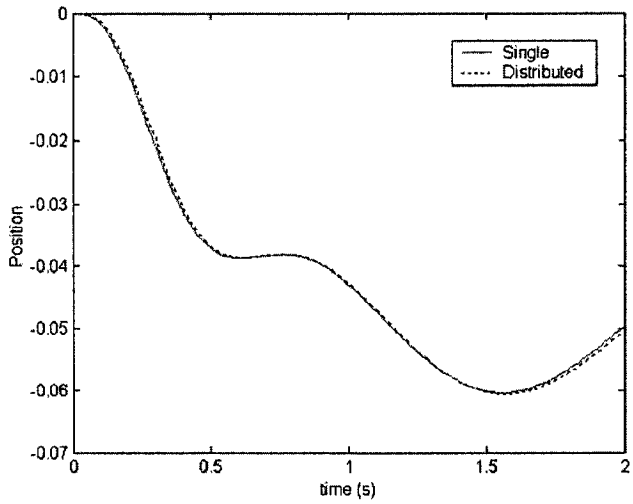

Figure 4-23 Comparison of single computer and distributed results: point (9,7)

5 Real-time Distributed Simulation Analysis

This chapter will optimize and analyse the real-time distributed simulation performance. The most important parameters used to evaluate the network performance are the CPU usage ratio and network communication rate.

5.1 CPU usage ratio

The CPU usage ratio is used to evaluate the computer's computational load. The bigger the CPU usage ratio, the heavier the computational load. In order to guarantee real-time property, this parameter should be less than 1 (100%). CPU usage ratio can be calculated using Equation (5.1).

$$R = \frac{\Delta t_c}{T_c} \tag{5.1}$$

where, R is CPU usage ratio

Δt_c is computational time

T_c is computational period

Figure 5-1 shows the relationship between the two parameters.

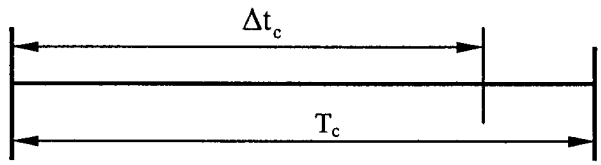

Figure 5-1 Computational time and computational period

In order to efficiently use the computational resources of the cluster we should have:

$$\begin{cases} \max(R_i) \\ \qquad i = 1...n \ \text{(where n is number of computers)} \\ R_i < 1 \end{cases} \tag{5.2}$$

The CPU usage ratio R is between 0 and 1. If the value is equal or greater than 1, the computation is over load, the simulation becomes non-real-time. However, we want it to be as large as possible by putting as many grid points on a computer as we can. In practice the maximum value of R may be less than one due to overhead associated with the network communication, so we need to experimentally determine the maximum R for a given application.

For a good network design, to fully utilize the CPU efficiently, the CPU usage ratio R should be as close as possible to 1. The CPU usage ratio will increase when the computation time increases. For the wave equation problem, the computational time for a simulation time step increases rapidly if the grid points increase, as illustrated in Figure 5-2.

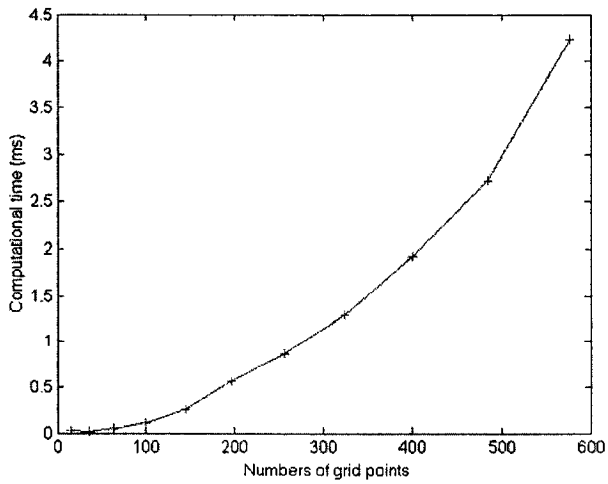

Figure 5-2 Computational time with grid points

Figure 5-3 shows the maximum CPU usage ratio on a single computer. The simulation parameters are listed in Table 5-1. They are chosen according to thesis [10].

Table 5-1 Simulation parameters (Single, implicit Euler method)

Name	Δt	T_{com}	T_c	N_{slot}	Slot Size	N
Meaning	Time step	Communication period	Computational period	Number of slots		Number of grids(X)
Value	6.8 ms	6.4 ms	6.8 ms	8	0.8 ms	16
Name	Lx	Ly	c	b1	b2	M
Meaning	Width of wave surface	Height of wave surface	Wave speed	Internal damping	External damping	Number of grids(Y)
Value	100m	100m	10 m/s	0.2 1/s	0.2 m^2/s	16

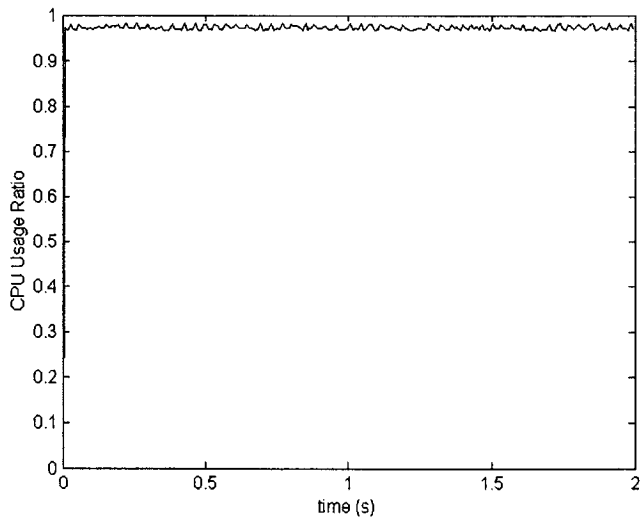

Figure 5-3 Maximum CPU usage ratio (Single computer, implicit Euler method)

Figure 5-4 shows the maximum CPU usage ratios for each of four computers in distributed simulation. The simulation parameters are listed in Table 5-2.

Table 5-2 Simulation parameters (distributed, implicit Euler method)

Name	Δt	T_{com}	T_c	N_{slot}	Slot Size	N
Meaning	Time step	Communication period	Computational period	Number of slots		Number of grids(X)
Value	4.9 ms	4.8 ms	4.9 ms	8	0.6 ms	24
Name	Lx	Ly	c	b1	b2	M
Meaning	Width of wave surface	Height of wave surface	Wave speed	Internal damping	External damping	Number of grids(Y)
Value	100m	100m	10 m/s	0.2 1/s	0.2 m^2/s	24

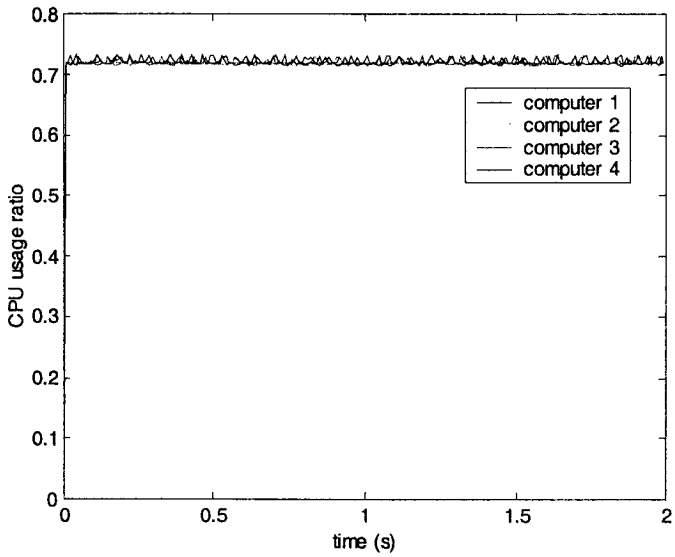

Figure 5-4 Maximum CPU usage ratios (distributed, implicit Euler method)

Another factor that affects the CPU usage ratio is the sleep time. In order to get bigger CPU usage ratio, the smallest sleep time should be used. In this thesis, the sleep time is chosen to be 1 ms (The smallest sleep time in the TDMA protocol).

5.2 Network communication rate

The network communication rate is another important criteria to evaluate the network performance. It is used to evaluate the network's capacity. The bigger the network communication rate is, the heavier the load on the network will be. The network communication rate must be less than the network capacity. The maximum data rate can be calculated as follows:

$$C_{max} = \frac{N_c N_p N_b}{T_{TDMA}} \tag{5.3}$$

where,

C_{max} is maximum communication rate (bps)

N_C is the number of computers

N_P is bytes per packet

N_b is bits/byte, $N_b = 8$

T_{TDMA} is TDMA period

If four computers are used in the network, then $N_c = 4$; for this network driver program, bytes per packet N_p=1500, TDMA period T_{TDMA}= 4.8 ms. From equation (5.3), the maximum communication rate is as follows:

$$C_{max} = \frac{4 \times 1500 \times 8}{0.0048} = 10000000(\text{bits/s}) = 9.54\text{Mbps}$$

The network communication rate C_i for each computer can be calculated using the

following equation:

$$C_i = n_{var}\, s_{var}\, \eta \quad \text{(bytes/second)}$$

$$= 8\, n_{var}\, s_{var}\, \eta \quad \text{(bits/second, bps)} \tag{5.4}$$

where,

C_i is network communication rate for computer i

η is frames per seconds

n_{var} is number of variables

s_{var} is size of each variables (bytes)

The frames per second η can be calculated by the following equation:

$$\eta = \frac{1}{T_{com}} \tag{5.5}$$

where T_{com} is the communication period.

The network communication rate C_i should have:

$$\begin{cases} \min (C_i) \\ \qquad\qquad i = 1...n \text{ (number of computers)} \\ C_i < C_{max} \end{cases} \tag{5.6}$$

For 26x26 grids, evenly distributed to four computers, and each grid uses two variables (θ and ω), so the total number of variables for each computer are:

$$n_{var} = 26 \times 26 \times 2 / 4 = 338$$

For float data type, the size of each variable $s_{var} = 4$, the communication period is set to 4 ms. From equation (5.6) the network communication rate for each computer is as follows:

$$C_i = 334 \times 4 \times \frac{1}{0.004} \times 8 \text{ (bps)}$$

$$= 2704000 \text{ bps} = 2.57 \text{ Mbps} \ll C_{max} \qquad i=1,2,3,4$$

5.3 Simulation scalability

This section discusses the relationship between number of grid points and number of computers in the network. It means how many grid points (computational tasks) we can reach if we increase the number of computers. Ideally, the relationship should be linear. But actually, communication capacity and CPU usage ratio will affect the results.

In order to make it comparable, one-dimensional equation distribution is used (as shown in Figure 4-2). The following steps are used to choose the simulation parameters:

1. Choose the number of computers (maximum slot number)
2. Choose the sleep time
3. Choose the slot time
4. Calculate the TDMA cycle (slot time * maximum number of slots)

Theoretically, the number of grid points can be calculated in a network is the number of grid points that a single computer can multipled by the number of computers. First of all, we test the maximum number of grid points for a single computer can compute, as listed in the second column of Table 5-3, then we use Equation (5.7) to calculate the maximum number of grid points multiple computer theoretically can compute.

$$\begin{cases} N_{max} = 24000\, N_{cpt} & \text{Case1} \\ N_{max} = 256\, N_{cpt} & \text{Case2} \\ N_{max} = 48\, N_{cpt} & \text{Case3} \end{cases} \tag{5.7}$$

where, N_{max} is maximum numbers of grid points can be simulated in the network

N_{cpt} is computer numbers in network

The calculation results are listed in Table 5-3 (column 3-9). Figure 5-5 shows the relationship between the two parameters in graphics mode.

Table 5-3 Maximum theoretical number of grid points and number of computers

Matrix A	1	2	3	4	5	6	7	8
Case1 Constant, Explicit Euler method	2400	4800	7200	9600	12000	14400	16800	19200
Case 2 Constant, Sliding implicit method	256	512	768	1024	1280	1536	1792	2048
Case 3 Varying, Sliding implicit method	48	96	144	192	240	288	336	384

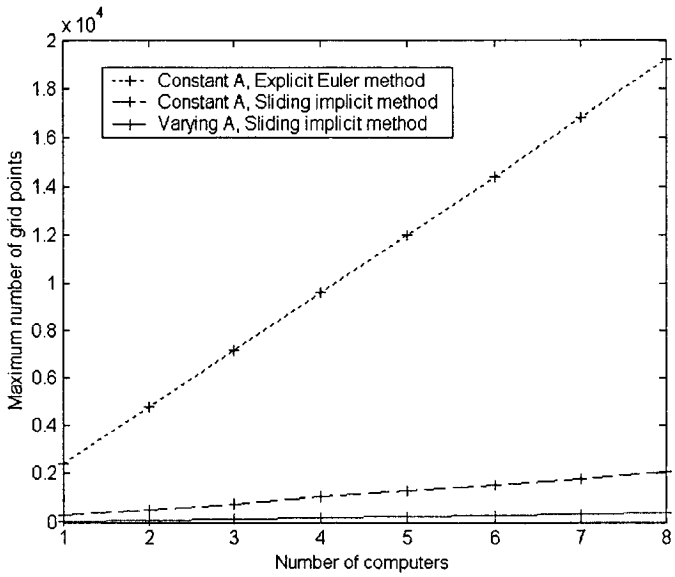

Figure 5-5 Maximum theoretical number of grid points and number of computers

Actually, the experimental maximum number of grid points are less because communication between computers takes CPU time. Table 5-4 shows the maximum experimental number of grid points with the number of computers. Figure 5-6 shows the relationship between the two parameters graphically.

Table 5-4 Maximum experimental number of grid points and number of computers

Matrix A	1	2	3	4	5	6	7	8
Case1 Constant, Explicit Euler method	2400	3456	5184	6912	8640	10368	12096	13824
Case 2 Constant, Sliding implicit method	256	364	546	728	910	1092	1274	1456
Case 3 Varying, Sliding implicit method	48	72	108	144	180	216	252	288

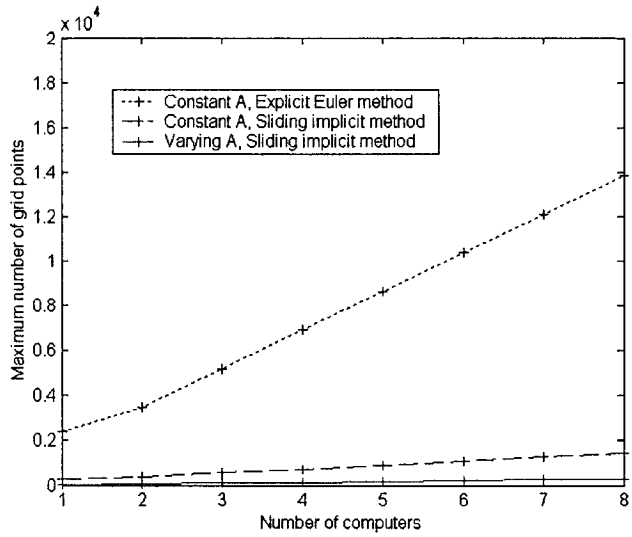

Figure 5-6 Maximum experimental number of grid points and number of computers

From Table 5-4 and Figure 5-6, we have the following equations:

$$\begin{cases} N_{max} = 1728\, N_{cpt} & \text{Case 1, } N_{cpt} > 1 \\ N_{max} = 182\, N_{cpt} & \text{Case 2, } N_{cpt} > 1 \\ N_{max} = 36\, N_{cpt} & \text{Case 3, } N_{cpt} > 1 \end{cases} \tag{5.8}$$

where, N_{max} is the maximum numbers of grid points can be simulated in the network

N_{cpt} is the computer numbers in network

Combining Figure 5-5 and Figure 5-6, Figure 5-7, Figure 5-8 and Figure 5-9 illustrate the comparison of maximum experimental and theoretical number of grid points with number of computers.

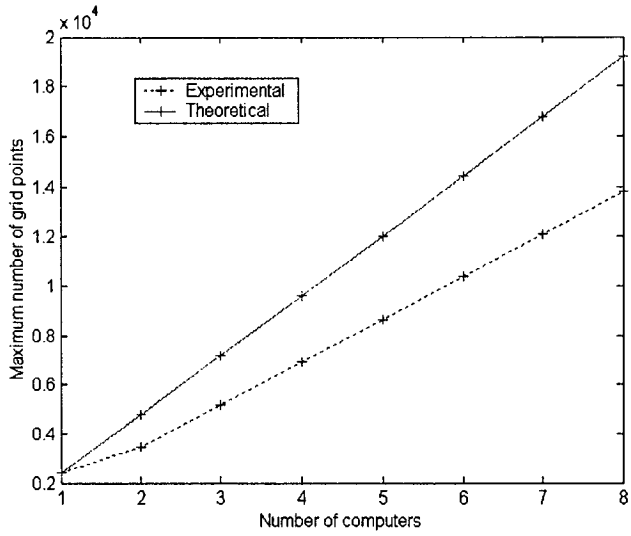

Figure 5-7 Comparison of maximum experimental and theoretical number of grid points with number of computers (case 1)

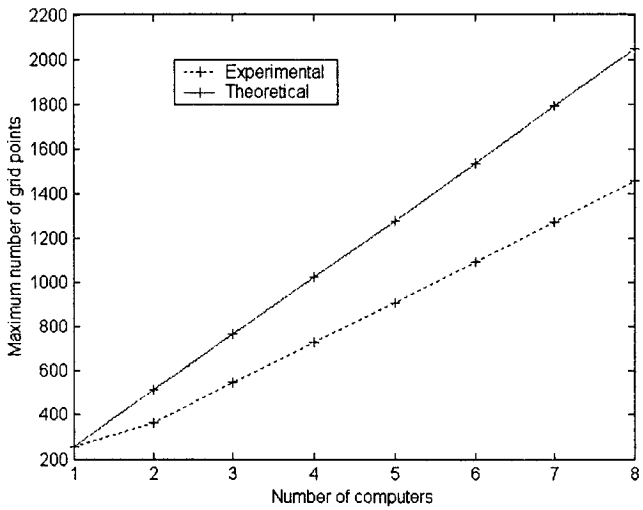

Figure 5-8 Comparison of maximum experimental and theoretical number of grid points with number of computers (case 2)

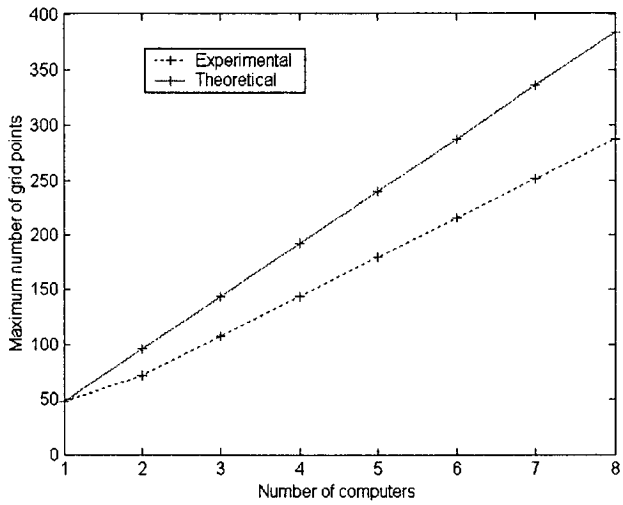

Figure 5-9 Comparison of maximum experimental and theoretical number of grid points with number of computers (case 3)

From Equation (5.7) and Equation (5.8), we can calculate that the ratio of experimental maximum number of grid points to theoretical maximum number of grid points is around 0.73. This means that the communication takes about 27% (1-0.73) of CPU time.

6 Conclusions and Future Work

6.1 Conclusions

In this thesis new approaches for real-time distributed simulation are implemented and investigated in detail. The main results and contributions of this thesis are summarized as follows:

1. Sliding implicit method

This thesis discusses and implements explicit and implicit Euler method in non-real-time distributed simulation. However, the implicit Euler method cannot guarantee real-time execution due to the difficulty in solving nonlinear algebraic equation problems. A new sliding implicit method is proposed for real-time implicit simulation.

2. New method for real-time simulation of PDAEs

By combining the PDEs with DAEs, a new approach for PDAEs (Partial Differential-Algebraic Equations) is proposed and implemented in distributed real-time simulation.

3. Real-time distributed simulation of PDEs

Real-time distributed simulation of PDEs using TDMA protocol communication are developed and implemented. The selection of simulation parameters and experimental verification are performed using the wave equation.

4. Optimization and analysis of real-time distributed simulation

The CPU usage ratio and network communication rate are investigated in real-time distributed simulations. The scalability is analysed for the simulation methods proposed in the thesis and experimentally verified.

6.2 Future work

In the future, the following problems will be investigated:

1. More sophisticated PDE and PDAE systems

The distributed simulation approach will be expanded to a larger class of PDE and PDAE systems with varying structure and boundary conditions. This will have great applications for virtual reality problems using PDE models.

2. New real-time equation distribution and communication protocols

Except for one-dimensional and two-dimensional equation distribution method, other complex distribution method will be investigated; new communication protocols other than TDMA will be investigated.

3. New applications

The real-time distributed simulation approach will be applied to many new industrial applications, such as high fidelity automotive and flight simulation problems that use PDE models. It will also be applied to enhance virtual reality and video game systems.

References

[1] William, H., Saul, A., William, T. V., and P. F. Brian, "Numerical Recipes in C", Cambridge University Press

[2] Baraff, D., and A. Witkin. "Large steps in cloth simulation. Computer Graphics", Proceedings SIGGRAPH, 1998

[3] Baraff, D., "Physically Based Modeling Implicit Methods for Differential Equations". Physically Based Modeling Class Notes – SIGGRAPH, 2002

[4] Gordon, B.W., and H. Asada, "Modeling, Realization, and Simulation of Thermo-fluid Systems Using Singularly Perturbed Sliding Manifolds". ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 122, No. 4, 2000, pp. 699-707.

[5] Rum, F., and B. W. Gordon, "Simulation of Deformable Objects using Sliding Mode Control with Application to Cloth Animation", conference ICCS

[6] Günther, M., "A PDAE Model for Interconnected Linear RLC Networks", Mathematical and Computer Modelling of Dynamical Systems 2001, Vol.7, No.2, pp. 189-203

[7] Slotine, J.-J. E., and W. Li, "Applied Nonlinear Control", Prentice Hall, 1991

[8] Schneider, J., and R. Westermann, "Towards Real-Time Visual Simulation of Water Surfaces", Proceedings Conference on Vision, Modelling and Visualization (VM) 2001, Stuttgart, Germany

[9] Loviscach, J. A., "Convolution-Based Algorithm for Animated Water Waves", EUROGRAPHICS 2002

[10] Lu, J., "Design of Ethernet based Real-time Distributed Systems", Master's thesis, Concordia University, Mechanical and Industrial Engineering Department, Jan, 2004

[11] Begehr, H., and A. Jeffrey, "Partial differential equations with complex analysis", Longman Scientific & Technical, Essex, England, 1992

[12] Koopman, P. J., and B. P. Upender, "Time Division Multiple Access Without a Bus Master", United Technologies Research Center Technical Report RR-9500470, June 30, 1995.

[13] Martinson, W. S., and P. I. Barton, "Index And Characteristic Analysis Of Linear PDAE Systems", SIAM J. SCI, Vol. 24, No. 3, pp. 905–923

[14] Li, S., Hyman, J. M., and L. Petzold, "Solution Adapted Mesh Refinement and Sensitivity Analysis for Parabolic Partial Differential Equation Systems", special volume of Lecture Notes in Computational Science and Engineering, Springer, 2001.

[15] Li, S., "Software and Algorithm for Sensitivity Analysis of Large-Scale Differential Algebraic Systems", J. Comput. and Appl. Math. 125 (2000), 131-145

[16] Brenan, K., Campbell, S., and L. Petzold, "Numerical Solutions of Initial-Value Problems in Differential-Algebraic Equations". New York: Elsevier, 1989.

[17] Braun, M., "Differential Equations & Their Applications", New York, Herdelberg Berlin

[18] Hamilton, J. A., David, J., and A. Nash, "Distributed Simulation", CRC Press, New York, 1992

[19] Ayres, F. J., "Theory and problems of Differential Equations", McGraw-Hill Book Company.

[20] Lim, Y.-I. L., Chang, S.-C., and S. B. Jorgensen, "A novel partial differential algebraic equation (PDAE) solver: iterative space–time conservation element/solution element (CE/SE) method", Computers & Chemical Engineering, Volume 28, Issue 8, 15 July 2004, Pages 1309-1324

[21] Brooks, R. J., and S. Robinson, "Simulation", Colin Lewis, 2001

Appendix A: Distributed algorithm using WinSock

Winsock is now widely used in industry. It is a networking API for Windows. This API provides functions for sending and receiving data in a network via a protocol, such as TCP/IP. A socket is a connection between two computers in a TCP/IP network. Sockets are used to exchange information such as data, files, mails and websites, etc.

There are two types of sockets: TCP (Transmission Control Protocol) sockets and UDP (User Datagram Protocol) sockets. These two types are called protocols. A protocol is a set of rules used to determine the way the information is received and sent in a network. To connect a socket to another computer, one need to know its IP-address and which port is going to connect to.

Winsock using client/server architecture to communicate data between connected computers. One is served as "the client", the other is used as "the server". The server can send data to and receive data from the other side - the client; the client, on the other hand, can receive data from and sent data to the server. Each client can send data to several servers; several clients can send data to one server in one time processing.

In this thesis, five computers are used to form a network: four of them are used to do computation and simulation; another one is used to display simulation results. The five computers connected using a hub or switch, as illustrated in Figure A-1.

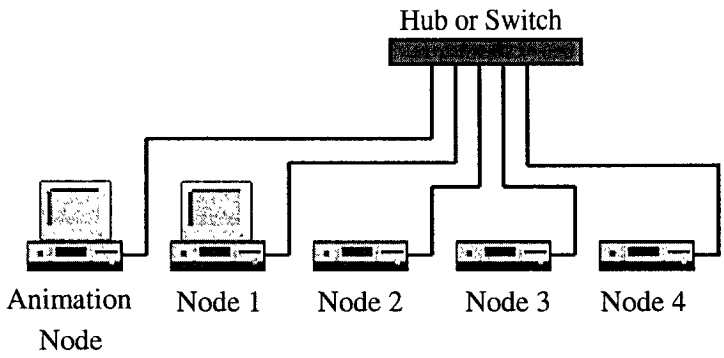

Figure A-1 Computer connection using Winsock

A.1 Distributed algorithm

A.1.1 One-dimensional distribution

Figure 4-1 and Figure 4-2 shows the equation distribution. Assume computer 1 calculate part 1, computer 2 calculate part 2, and so on. In the communication process, the following assumption is made:

At any time the computer that calculates the right part is used as the server (bottom part for vertical distributed); the one that calculates the left part (upper part for vertical distributed) is used as client. For example, computer 2 and computer 3 will calculate part 2 and part 3. When the two computers communicate each other, computer 3 is the server, computer 2 is the client.

When the computation starts, the computer that calculates the far right part (or far bottom for vertical distribution, for the case above, node 4) runs first, then computer 3 runs

communicating with computer 4, then computer 2 runs communicating with computer 3, and so on. As for the animation computer, it is always served as a server.

This thesis uses the following functions to initialize server and client respectively:

```
int InitClntSock() {// initialize the client
    WSADATA Data;                    /* Structure for WinSock setup communication */
    int sock;
    WSAStartup(2, &Data);
    memset(&ClAddr, 0, sizeof(ClAddr));          /* Zero out structure */
    ClAddr.sin_family = AF_INET;                 /* Internet address family */
    ClAddr.sin_addr.s_addr = htonl(INADDR_ANY); /* Any incoming interface */
    ClAddr.sin_port = htons(7);                  /* Local port */
    sock = socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);     /* Create socket   */
    bind(sock, (struct sockaddr *) &ClAddr, sizeof(ClAddr));
    return sock;
}

int InitServSock(struct sockaddr_in *SvAddr, char *svIP) {// initialize the server
    WSADATA Data;                    /* Structure for WinSock setup communication */
    WSAStartup(2, &Data);
    memset(SvAddr, 0, sizeof(struct sockaddr_in *));          /* Zero out structure */
    SvAddr->sin_family = AF_INET;                /* Internet address family */
    SvAddr->sin_port     = htons(7);             /* Server port */
    SvAddr->sin_addr.s_addr = inet_addr(svIP);   /* Server IP address */
    return socket(PF_INET, SOCK_DGRAM, IPPROTO_UDP);
}
```

A.1.2 Two-dimensional distribution

This distribution mode is illustrated in Figure 4-4, an inclined line that drawn from left-bottom to upper-right is used to determine which computer is served as the client, and which one is used as the server. In one communication step, if the part is below the line, then the computer that calculate this part is the Server; if the part above the line, then the node that calculates this part is the client. Every three neighbor parts that in the "⌊" position form a group. For example, part 6, 8 and 9 form a group, part 5, 7 and 8 form a group, and so on. Take part 6, 8 and 9 as an example. These three parts form a group. Computer 9 will be the server; computer 6 and computer 8 will be the clients. See Figure A-2.

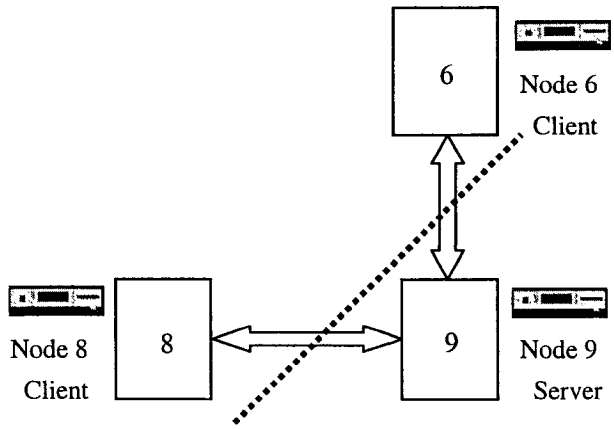

Figure A-2 Server and client

In this group, computer 9 calculates part 9 and runs first, waiting to receive data, then computer 6 calculates part 6 and computer 8 calculates part 8 and runs to send boundary data to computer 9 and receive value from the computer 9 (the server). The animation computer is served as a server.

A.2 Implementation and results

In this part, 2D wave equations are took as examples to illustrate the simulation results.

A.2.1 One-dimensional distribution results

Here we choose the following parameters:

$L_X = 100$ m; $L_Y = 100$ m; $c = 10$ m/s; $b1 = 0.2$ 1/s; $b2 = 0.2$ m^2/s; $\omega = 2$ m/s

Figure A-3 shows single computer simulation and distributed simulation comparison results of point (3,2) in 2D wave equation.

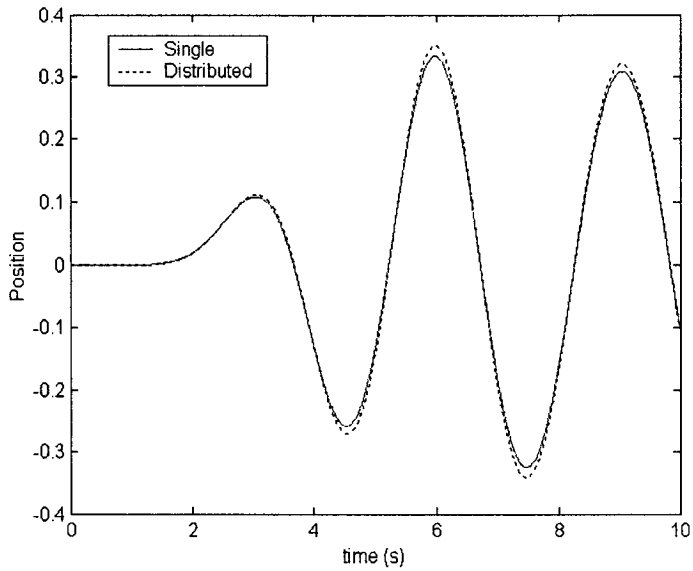

Figure A-3 Comparison of single computer and distributed results: point (3,2)

Figure A-4 shows the results comparison of point (4.3).

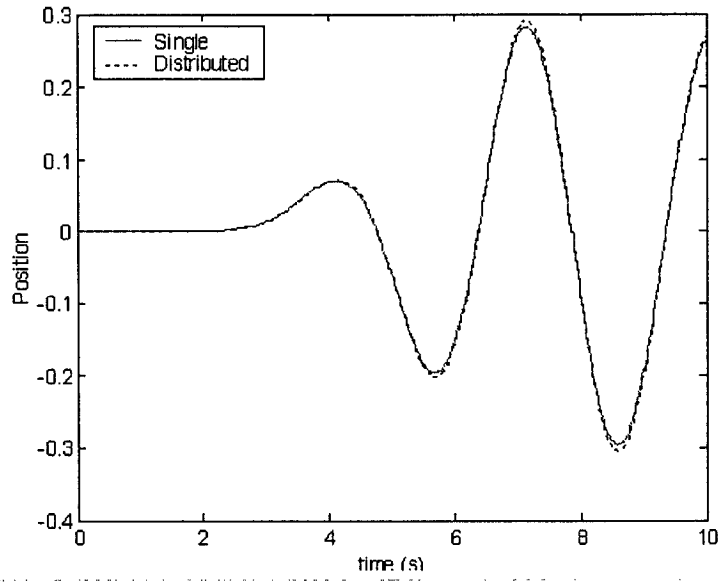

Figure A-4 Comparison of single computer and distributed results: point (3,4)

A.2.2 Two-dimensional partition results

Figure A-5 and Figure A-6 shows single computer simulation and distributed simulation comparison results of point (3,2) in 2D wave equation.

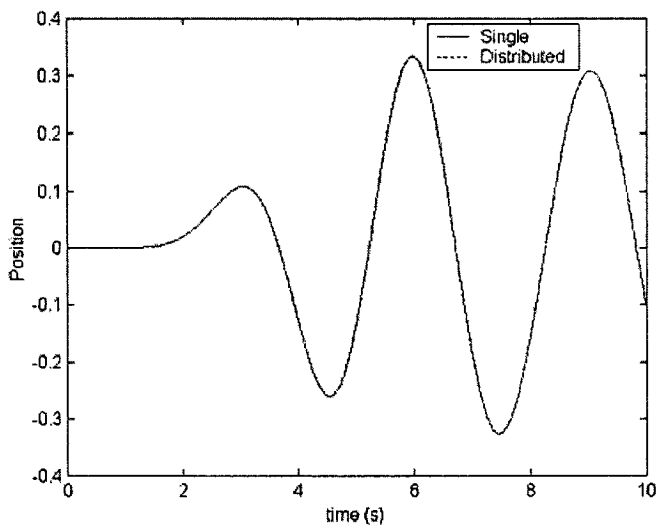

Figure A-5 Comparison of single computer and distributed results: point (3,2)

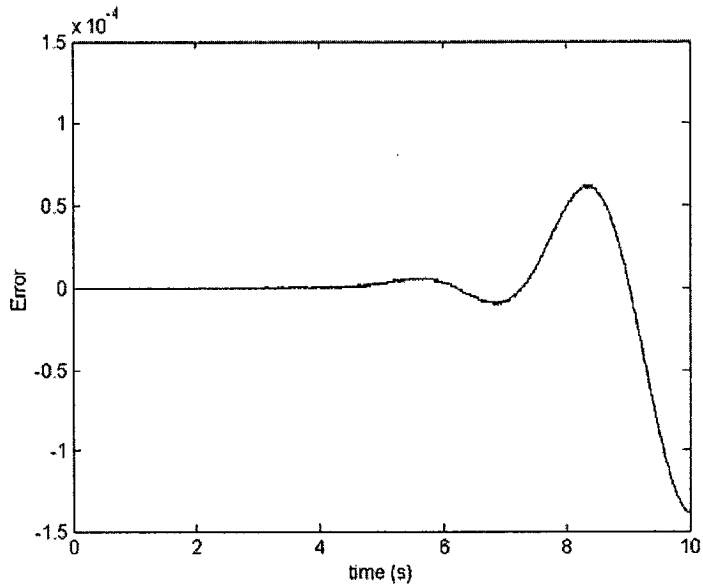

Figure A-6 Single and distributed error

Figure A-7 and Figure A-8 shows the results comparison of point (4, 3).

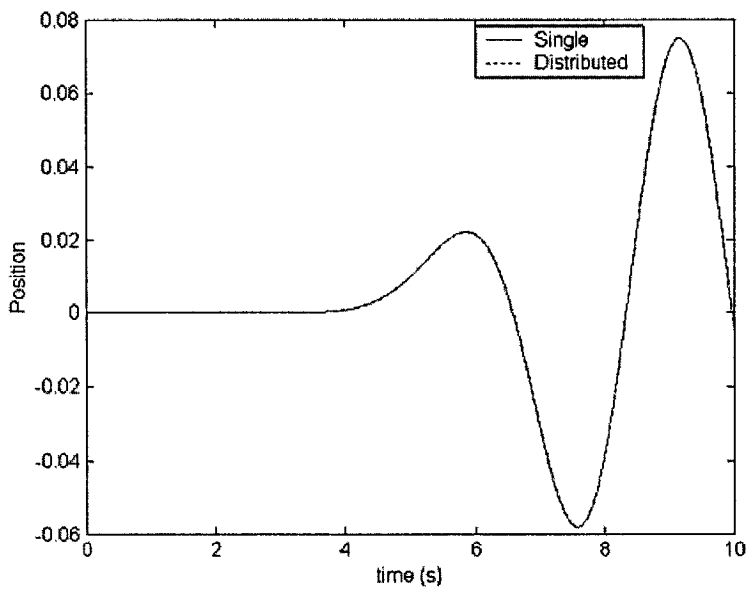

Figure A-7 Comparison of single computer and distributed results: point (3,4)

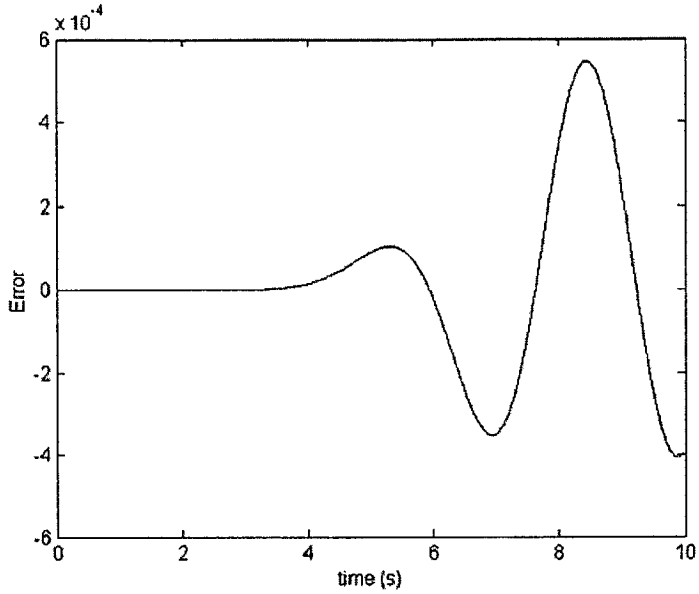

Figure A-8 Single and distributed error

Appendix B: Visualization/Animation of Simulation Results

To animation and view the simulation results on the screen make it more realistic. Nowadays, with the development of computer technology, computer graphics, multimedia are very common. It has good user interface than just plain text and numbers. There are many methods to view animation results. In this thesis, Microsoft DirectX 9.0 is used.

Microsoft DirectX is a set of low-level application programming interfaces (APIs) for creating graphics and other high-performance multimedia applications. It includes support for two-dimensional (2-D) and three-dimensional (3-D) graphics, sound effects and music, input devices, and networked applications. It includes: Microsoft Direct3D, Microsoft DirectInput, Microsoft DirectMusic, Microsoft DirectSound, and Microsoft DirectPlay.

In this thesis, animation uses Direct 3D. It is used to perform graphics functions at a lower level Windows GDI;

The animation of the simulation results is implemented in Windows 2000 environment, using Direct3D 9.0 and Visual C++ 6.0.

The process is as follows:

- Store calculated at previous frame in Buffer1, at current frame in Buffer2
- Use Buffer2 to render
- Exchange Buffer1 and Buffer2

Repeat the process continuously, the animation is implemented.

When rendering each scene, rectangle texture is used. In order to improve the rendering speed, making it much faster, an index buffer is used.

An index buffer is a memory buffer that holds indices that "point" to vertices in the vertex buffer. When a scene is rendered, DirectX performs certain calculations on each vertex such as lighting and transformations. What we want to do is minimise the amount of calculations that DirectX has to do, therefore, we need to minimise the number of vertices. An index buffer can be used to do this.

For example, if we want to draw a square, it will be made up from two triangles, which have six vertices (using a triangle list). But we only really need four vertices to define a square (one for each corner). We have to use six vertices because two of them are shared (have the same value). Because we have shared vertices, it's a good idea to use an index buffer. By defining the four corners of our square as vertices in the vertex buffer, then define six indices in the index buffer. Each of which "points" to a vertex in the vertex buffer, then render the triangles from the indices in the index buffer and so, only use four vertices. Figure B-1 below shows this example.

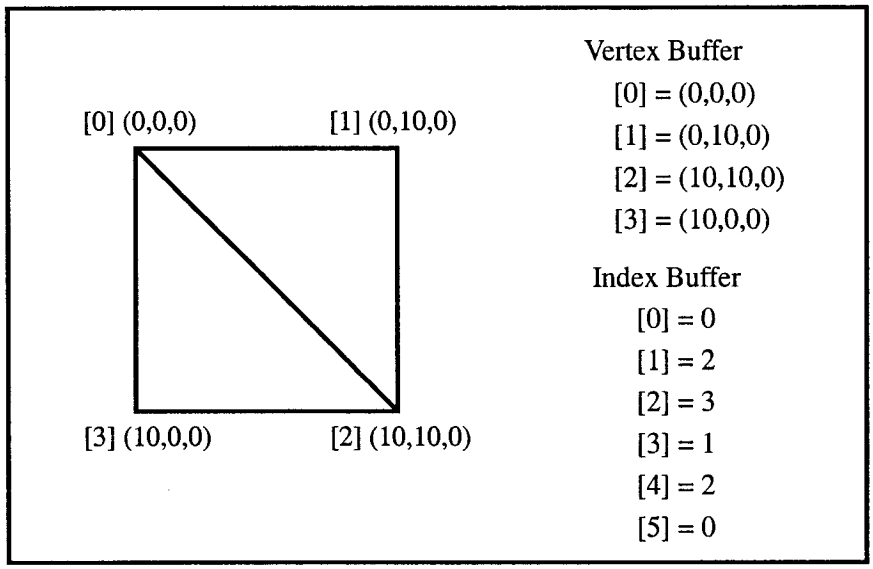

Figure B-1 Index buffer

The index buffer used in this animation is showed in Figure B-2.

The following source code is used for creating the index buffer:

```
for(z = 0; z < M; z++)
    {
     for(x = 0; x < N; x++)
     {
          vtx = x + z * (M + 1);
          pBuf[i + 0] = vtx;
          pBuf[i + 1] = vtx + N + 2;
          pBuf[i + 2] = vtx + N + 1;
          pBuf[i + 3] = vtx + 1;
          pBuf[i + 4] = vtx + N + 2;
          pBuf[i + 5] = vtx;
```

```
            i += 6;
        }
    }
```

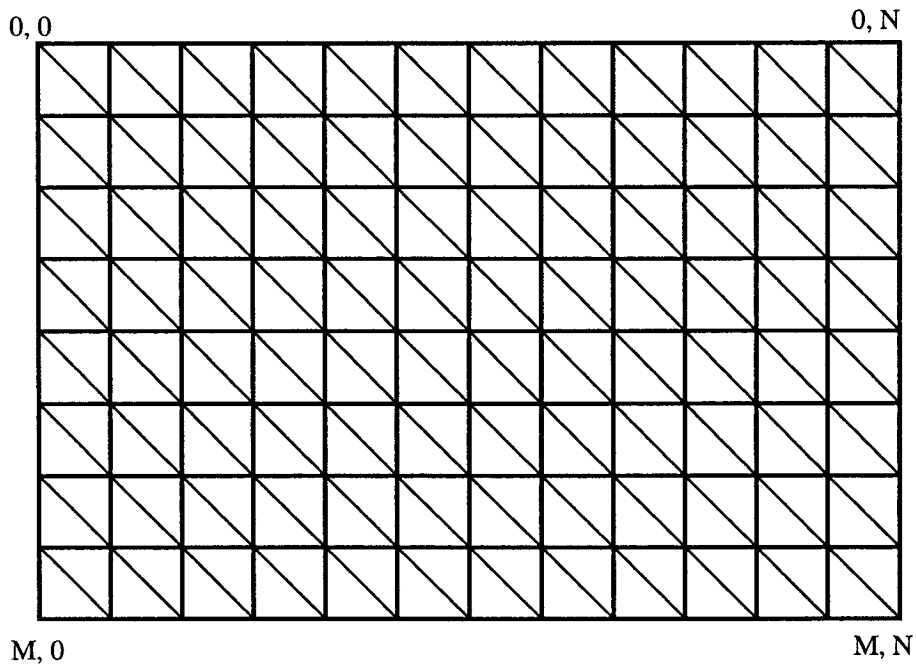

Figure B-2 Index buffer used in the thesis

The animation results are shown in Figure 2-6 and Figure 2-7.

Appendix C: Website References

[1] Microsoft Corporation, http://www.microsoft.com/windows/directx/

[2] VenturCom Inc., "RTX Reference guide", http://www.vci.com

[3] Game develop Forum, http://www.gamedev.net/community/forums/

[4] Intel developer service home. http://www.intel.com/cd/ids/developer/Asmo -na/eng/index.htm

[5] VTP, Virtual Terrain Project website: http://www.vterrain.org/index.html

[6] Opal-RT Technologies, Inc: http://www.opal-rt.com/

[7] Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/ Partial_differential_equation

[8] CAE electronics: http://www.cae.com/

[9] MPL/RT: http://www.mpirt.org/

[10] Mercury computer systems inc. http://www.mc.com/news/news_detail.cfm? press_id=2000_04_25_021347_466106pr.cfm

[11] Math World. http://mathworld.wolfram.com/Differential-Algebraic Equation.html

[12] Mechanical Simulation Corporation, http://www.carsim.com/index/

[13] QNX software systems: http://www.qnx.com/

[14] VxWorks Operating system: http://www.windriver.com/announces/vxworks/

[15] TDMA protocol: http://www.webopedia.com/TERM/T/TDMA.html

[16] Chrisitan, W., Fischer, S., et al., "The Wave Equation and Other PDEs", http://webphysics.davidson.edu/Faculty/wc/WaveHTML/node4.html

[17] Scirus databse: http://www.scirus.com/srsapp/advanced/index.jsp