A Dynamic Simplex Approach for Optimization of Real-time Distributed Multi-body Simulations

Zhi Qian Ren

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 2005

Your file Votre référence
ISBN: 0-494-10272-1
Our file Notre référence
ISBN: 0-494-10272-1

ABSTRACT

A Dynamic Simplex Approach for Optimization of
Real-time Distributed Multi-body Simulations

Zhi Qian Ren

This thesis presents a new approach for dynamic optimization of real-time distributed multi-body simulations. It continuously optimizes network communication for a given model partitioning. The optimization problem proposed is to minimize a network performance cost function using the recently developed dynamic simplex method (DSM). This iterative minimization approach requires a small number of function evaluations, no derivatives, and it can track a moving optimum in a stable manner. Furthermore, evidence indicates that the approach converges close to the optimal solution provided it doesn't vary too rapidly. Experimental investigations are performed on a time division multiple access (TDMA) computational network to validate the new approach.

ACKNOWLEDGEMENTS

I am very grateful to my research advisor Professor B. W. Gordon of the Mechanical and Industrial Engineering Department at Concordia University. This work would have been impossible without his encouragement, support and guidance. Gilles Huard you deserve a great deal of gratitude for installing and testing equipment. I also give thanks to Lu Jiang, whose patience and explanation gave me lots of help on network simulation. Zhou Sheng Feng was a good partner and helped me get over many programming difficulties. Liu and Rum also deserve thanks for their advices and support. On a personal level, I would like to thank my family. I could not have finished my thesis without their motivation and emotional support.

Zhi Qian Ren

Concordia University

Montreal, Canada

Table of contents

List of figures

Nomenclature

m	Mass of body (kg)
$\boldsymbol{P}$	Linear position of mass in global coordinates (m)
$\boldsymbol{V}$	Linear velocity of mass in global coordinates (m/s)
$\boldsymbol{e}$	Angular orientation of mass in global coordinates
$\mathbf{R}$	Rotation matrix
$\boldsymbol{\omega}_{\mathrm{g}}$	Angular velocity of mass in global coordinates (radian/s)
$\mathbf{I}_{\mathrm{g}}$	Inertia tensor in global coordinates
F_{g}	Force acting on the body in global coordinates (N)
$\boldsymbol{F}_{\mathrm{ext}}$	External force (N)
$\boldsymbol{F}_{\mathrm{con}}$	Constraint force (N)
z	A vector of Lagrange multipliers
$\boldsymbol{Tq}_{\mathrm{g}}$	Torque acting on the body in global coordinates (N. m)
$\boldsymbol{g}$	Position constraint equation (m)
$\dot{\boldsymbol{g}}$	Velocity constraint equation (m/s)
$\ddot{\boldsymbol{g}}$	Acceleration constraint equation (m/s^2)
$\boldsymbol{w}$	Position constraint equation in sliding mode control (m)
$\dot{\boldsymbol{w}}$	Velocity constraint equation in sliding mode control (m/s)
$\ddot{\boldsymbol{w}}$	Acceleration constraint equation in sliding mode control (m/s^2)
$\mathbf{J}$	Constraint Jacobian matrix

ε	A certain amount of error
μ	Dynamic control parameter in sliding mode control
K	A parameter in sliding mode control
n_b	The number of rigid bodies
n_j	The number of joints
n_c	The number of nodes
m_c	Constraint dimension
Δt	Simulation time step size (s)
τ	Time delay (s)
T	The measurement period (s)
T s	The measurement period in server (s)
k_i	The number of bytes every transfer (bytes/transfer)
γ	A positive and very small value
m_r	The number of successive reflections
D_{smin}	The minimum simplex size
D_s	The simplex size
v	The maximum movement speed of the optimum
T_{TDMA}	TDMA period
Δt_c	Computational time (s)
T_{send}	Communication time (s)

1 Introduction

1.1 Motivation

Multi-body systems can range from relatively simple examples to very complex engineering systems. Multi-body systems can be defined as a group of rigid bodies interconnected by kinematic joints and/or force elements. Some or all of the bodies can move relative to one another in such problems. Simulation of complex multi-body systems can consume an extensive amount of CPU time. Distributed simulation can reduce the time required for simulation and also allow real-time computation. This approach has been used extensively in aerospace, automotive, and virtual prototype applications. However, currently there does not exist a systematic approach for optimization of distributed multi-body simulations.

In this thesis, dynamic optimization of real-time distributed multi-body simulations is investigated in a systematic manner. In order to enhance stability and performance of distributed systems, two phases of optimization are required. One is the optimal design of a single node's stabilization algorithm. The other is the optimal operation of distributed simulations. The first problem can be solved using an appropriate stabilization algorithm. For the second problem, because the optimal communication rate varies as the simulation changes, dynamic optimization is needed for efficiency. However, currently no approach exists for dynamic optimization of real-time distributed multi-body simulations. This thesis represents a step towards a general framework for dynamic optimization of distributed mechanical simulations. This is achieved by presenting a new dynamic simplex approach for optimization of communication in distributed multi-body simulations.

1.2 Literature review

The related literature can be divided into two areas described in the following sections.

1.2.1 Stabilization of multi-body systems

The first area is the work on stabilization of multi-body simulations for a single node. In order to implement multi-body simulations, much knowledge about modeling needs to be studied. Parviz [1] and Haug [13] introduce fundamental theories of computational mechanics, and Baraff [15] discusses physically based modeling. These theories and methods can be used to model these systems and develop computer programs. The constraint stabilization is an extension of feedback control theory applied to the dynamic analysis of multi-body systems. One of the goals in designing a feedback controller is to suppress the growth of error and achieve a stable response. The stability problem has been overcome by several stabilization methods, which include Baumgarte stabilization method [8], penalty formulations [9], projection methods [10], a differential algebraic approach [11], and coordinate partitioning methods [12]. A new approach based on SMC

(Sliding Mode Control) is introduced, and it is based on the work of Gordon [6][5] and Rum [5]. Gordon incorporates SMC with simulation of thermo-fluid systems, while Rum fits SMC into simulation of deformable objects that consist of rigid and flexible structural elements. SMC is a robust method, and its main advantage is in dealing with model uncertainty. Based on SMC method, the performance of single node simulation is optimized.

1.2.2 Optimal operation of distributed simulations

The second area of related work is the literature on the optimal operation, which solves the optimization problem of distributed simulations to obtain the optimal operating point "on the fly." The main goal is to design a controller to adjust the time delays of each node, so that the overall system performance is improved. That is, given multiple stabilized subsystems, what is the best way to allocate resources, so that optimality conditions are met. Hence, a cost function needs to be designed, and an optimization problem is constructed and solved using various algorithms. Furthermore, it can be considered as the design of networked control systems (NCS), which is based on a limited bandwidth and has enjoyed significant industrial interest recently because of its capability of boosting the stability of systems, and there are various analytical optimization methods [16]-[22]. However, those investigations mainly focus on LTI systems, and no researchers have studied optimization of distributed simulation of multi-body systems specially. Therefore, in this thesis, we will focus attention on numerical methods available for solving this problem, which can be classified into the following four types of algorithms stated as follows:

The first class of optimization methods includes Gradient Descent, Conjugate Gradient, and Newton's method. Each method has its strengths and weaknesses, but all of these methods require computation of gradients. Each method is very efficient when it works, but its shortage is that it may diverge or may converge to an unwanted solution, and then it becomes unstable.

Secondly, for online optimization, the Sequential Quadratic Programming (SQP) method is always used, and it is based on the idea of finding a search direction by linearizing the cost function and constraint functions and then taking a suitable step in this direction. The disadvantage of this approach is highly dependent on the availability of a good process model.

Thirdly, if the model of system is too sophisticated, one can use an empirical black-box model, and several methods have been developed, such as Response Surface Methodology (RSM) [23], Dynamic Response Surface Methodology (DRSM) [24], and a adaptive optimization technique with Recursive Least-squares (RLS) method [29]. RSM

[23] is particularly useful for optimizing a running system. In applying the RSM, we fit an empirical response surface into the response or dependent variables influenced by several independent variables, and we can search or hunt for an optimum with this fitted surface. However, the RSM technique can be successfully applied to the process with a static optimum but not to a moving optimum. For tracking the moving optimum, Edwards and Jutan [24] have extended the traditional RSM to DRSM, and the difference with RSM is that a quadratic surface is estimated as the process moves close to the true optimum. Although the DRSM is able to track the moving optimum, its parameters greatly influence its performance, and even a slight variation can cause significant deterioration from the true optimum. Also, importantly, a large number of measurements are required in each iteration. For systems with slow dynamics, Bamberger and Isermann [29] innovated an adaptive optimization technique, in which a Hammerstein model is applied to the system and the model parameters using RLS method are adjusted online. This approach greatly decreases the time for optimization because it doesn't wait for the process to come close to steady state for using the predicted steady state based on the Hammerstein model. Many researches [30]-[32] have focused attention on this approach and get relatively good results. However, by simulation study [33], the main disadvantage of this approach is that convergence rates predicted is always faster than those in practice.

Finally, compared with SQP and the empirical black-box model based optimization approaches, direct search methods are not concerned about process modeling to perform the optimization. The most popular direct search method is the Nelder–Mead simplex method [25], which has been extensively used in many fields because of its simplicity and efficiency. A chronological bibliography of the Nelder-Mead simplex method in reference [34] shows its great amount of applications and rapid growth. However, this technique cannot be applied to the process with a moving optimum. Xiong and Jutan [26] have extended the method to track the moving optimum. Compared with the Nelder-Mead Simplex method, DSM uses a fixed simplex size and allows tracking of the moving optimum. Xiong and Jutan applied it in chemical engineering, and the extension of this method to distributed simulation of multi-body systems is studied in this thesis.

1.3 Thesis outline and contributions

1.3.1 Thesis outline

The thesis is organized as follows. Chapter 2 introduces how to model multi-body systems, deduces the stabilization of multi-body systems, and incorporates SMC method with multi-body systems. Chapter 3 applies DSM to distributed multi-body systems. Chapter 4 realizes TDMA Networked Simulation Examples using DSM. In the last section, the conclusion and future study are given.

1.3.2 Thesis contributions

In this thesis, a new approach for dynamic optimization of real-time distributed multi-body simulations is investigated, and the main contributions of this thesis are summarized as follows:

- A new cost function is defined and combined with the dynamic simplex optimization algorithm. The optimization problem is to minimize the network performance cost function defined. And the new cost function is combined with a dynamic simplex optimization method, which is a very simple iterative minimization method and requires only several function evaluations (measurements), no derivatives, and it always converges and is fairly stable.
- A new approach is developed for continuously optimizing network communication for a given model partitioning. We will present the method for cutting the multi-body systems for a given example. Moreover, based on the partitioning method, a controller designed obtains data from distributed systems and continuously adjusts time delays of the distributed network, that is, optimizes network communication.
- Experimental support is provided that indicates the approach converges close to the optimal solution provided it doesn't vary too rapidly. Two different time-delay distributions, which are space distribution and time distribution, have been examined. It has been verified that the approach can catch the optimum, follow its movement continuously, enhance the stability of systems, and minimize simulation errors.

- A realistic experimental investigation is performed on a TDMA (time division multiple access) computational network. Compared with other protocols, TDMA allows us to change the time delays on line through the communication schedule. Moreover, we have implemented dynamic optimization for real-time distributed multi-body systems based on TDMA protocol, and experimental results are satisfactory.

2 Modelling of Multi-body systems

2.1 Newton-Euler equations

The translation and rotational equations of motion for an unconstrained rigid body, to which there is no kinematical joint attached to eliminate any of its degrees of freedom, are given from equations (2.1) and (2.2). These two equations are the so-called Newton-Euler equations of motion for an unconstrained body, and can be expressed in matrix form of equation (2.3).

$$\boldsymbol{Tq}_{\mathrm{gi}} = \mathbf{I}_{\mathrm{gi}}\dot{\boldsymbol{\omega}}_{\mathrm{gi}} + \boldsymbol{\omega}_{\mathrm{gi}} \times (\mathbf{I}_{\mathrm{gi}}\boldsymbol{\omega}_{\mathrm{gi}}) \tag{2.1}$$

$$\boldsymbol{F}_{\mathrm{gi}} = \mathrm{m}_{\mathrm{i}}\dot{\boldsymbol{V}}_{\mathrm{i}} \tag{2.2}$$

$$\begin{bmatrix} \mathbf{L}_{\mathrm{i}} & 0 \\ 0 & \mathbf{I}_{\mathrm{gi}} \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{V}}_{\mathrm{i}} \\ \dot{\boldsymbol{\omega}}_{\mathrm{gi}} \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{\omega}_{\mathrm{gi}} \times (\mathbf{I}_{\mathrm{gi}}\boldsymbol{\omega}_{\mathrm{gi}}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}_{\mathrm{gi}} \\ \boldsymbol{Tq}_{\mathrm{gi}} \end{bmatrix} \tag{2.3}$$

Where, $\mathbf{L}_{\mathrm{i}} = \mathrm{diag}[\mathrm{m}_{\mathrm{i}}, \mathrm{m}_{\mathrm{i}}, \mathrm{m}_{\mathrm{i}}]$. Equation (2.3) can also be written in the compact form

$$\mathbf{M}_{\mathrm{i}}\ddot{\boldsymbol{x}}_{i} + \boldsymbol{b}_{\mathrm{i}} = \boldsymbol{c}_{\mathrm{i}} \tag{2.4}$$

Given initial conditions, which are values at time zero for position and velocity, equation (2.4) can be easily solved using numerical methods. The extension and solution of equation (2.4) for a collection of n bodies is described in equation (2.5).

$$\begin{bmatrix} \mathbf{M}_1 & & & & \\ & \mathbf{M}_2 & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \mathbf{M}_n \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \cdot \\ \cdot \\ \ddot{x}_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ b_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \cdot \\ \cdot \\ c_n \end{bmatrix} \tag{2.5}$$

This can be expressed in the more compact notation as follows

$$\mathbf{M}\ddot{x} + b = c \tag{2.6}$$

2.2 Constraint dynamics

The constraint, which is said to be equality-constraint, can be written as an equality. Equality-constrained dynamics have been studied extensively, and jointed bodies are the most common example of dynamic simulation with equality constraints. While the constraint, which is said to be inequality-constraint, contains inequalities that are not integrable in closed form. In this thesis, only equality constraint is discussed. Position constraint equation is described by

$$g = g(x) = 0 \tag{2.7}$$

And the kinematical joints in the system can be represented as $\mathrm{m_c}$ independent constraints, which is the number of degrees of freedom that the constraint removes from the system. If the constraint function returns a zero vector, then the position state x satisfies the constraint.

When the position of a body is constrained, its velocity also needs to be constrained. The velocity constraint equation is obtained by taking the time derivative of the position constraint equation (2.7), that is

$$\dot{g} = g_x \dot{x} = \mathbf{J}\dot{x} = \mathbf{J}v = 0 \qquad \textbf{(2.8)}$$

Where, $\mathbf{J}$ is the constraint Jacobian matrix, which is partial derivative of the position constraint equation with respect to the coordinates. For a system that is composed of n bodies and has m constraints, our velocity constraint equation looks like this:

$$\dot{g} = \mathbf{J}v = \begin{bmatrix} \mathbf{J}_{1\_1} & \mathbf{J}_{1\_2} & \cdots & \cdots & \cdots & \mathbf{J}_{1\_n} \\ \mathbf{J}_{2\_1} & \mathbf{J}_{2\_2} & \cdots & \cdots & \cdots & \mathbf{J}_{2\_n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathbf{J}_{m\_1} & \mathbf{J}_{m\_2} & \cdots & \cdots & \cdots & \mathbf{J}_{m\_n} \end{bmatrix} \begin{bmatrix} V_1 \\ \omega_{g1} \\ V_2 \\ \omega_{g2} \\ \cdot \\ \cdot \\ V_n \\ \omega_{gn} \end{bmatrix} = 0 \qquad \textbf{(2.9)}$$

Where, $\mathbf{J}_{i\_j}$ is the constraint i, which is related with rigid body j. Similarly, the time derivative of the velocity equation (2.8) yields the acceleration constraint equation

$$\ddot{\mathbf{g}} = \mathbf{J}\ddot{x} + \dot{\mathbf{J}}\dot{x} = 0 \qquad \textbf{(2.10)}$$

Equation (2.10) can be written as:

$$\mathbf{J}\ddot{x} = \mathbf{J}_1\dot{V}_1 + \mathbf{\Omega}_1\dot{\omega}_{g1} + \mathbf{J}_2\dot{V}_2 + \mathbf{\Omega}_2\dot{\omega}_{g2} = -\dot{\mathbf{J}}\dot{x} \qquad \textbf{(2.11)}$$

Because,

$$\mathbf{J}\ddot{x} = \mathbf{J}\mathbf{M}^{-1}(F_{ext} + F_{con}) \qquad \textbf{(2.12)}$$

Where F_{ext} is the external force vector, and F_{con} is the constraint force vector between connected bodies. F_{con} is introduced by each kinematical joint, and it can be expressed in terms of the constraint Jacobian matrix and a vector of Lagrange multipliers that contains the magnitudes of the constraint forces.

$$F_{\text{con}} = \mathbf{J}^{\mathrm{T}} z \tag{2.13}$$

Substituting equation (2.13) in equation (2.12) obtains:

$$\begin{aligned} \mathbf{J}\ddot{x} &= \mathbf{JM}^{-1}(F_{\text{ext}} + F_{\text{con}}) \\ &= \mathbf{JM}^{-1}F_{\text{ext}} + \mathbf{JM}^{-1}\mathbf{J}^{\mathrm{T}} z \\ &= \mathbf{JM}^{-1}F_{\text{ext}} + \mathbf{JM}^{-1}\mathbf{J}^{\mathrm{T}} z \end{aligned} \tag{2.14}$$

Substituting equation (2.14) in equation (2.11) obtains:

$$\mathbf{JM}^{-1}\mathbf{J}^{\mathrm{T}} z = -\dot{\mathbf{J}}\dot{x} - \mathbf{JM}^{-1}F_{\text{ext}} \tag{2.15}$$

2.3 Constraint stabilization

The major approach to dealing with model uncertainty in this thesis is SMC. SMC realization is “a robust method that allows us to ignore various terms and make efficient simulations in a systematic manner” [4]. The main purpose of incorporating SMC is to make use of its robustness, locking and decoupling the original nonlinear system into linear subsystems. And this chapter, SMC method has been successfully fitted into the general case of constraint dynamics of multi-body systems.

2.3.1 Design SMC controller

Constraint dynamics allows models to be imprecise, which may come from the accuracy level of constraints, the simplified representation of the systems' dynamics, and the time delays of the network in distributed simulations. In most cases, the accuracy level required for constraints allows us to permit a certain amount of error ε. Therefore the position constraint equation can be described as an algebraic inequality

$$w = g(x) \le \varepsilon \tag{2.16}$$

And in order to fit SMC into the constraint dynamics, let $w = g(x)$ be the regulation error in the variable x, and let

$$\tilde{w} = [w, \dot{w} \ldots w^{(n-1)}]^{T} \tag{2.17}$$

be the regulation error vector. In order to have the regulation of $\tilde{w} = 0$, a time-varying surface $s(t)$ in the state-space $R(n)$ is defined, where

$$s(t) = (\mu \frac{d}{dt} + 1)^{n-1} \tilde{w} \tag{2.18}$$

And μ is a strictly positive constant and will be interpreted later. n is the index number, which is 3 according to the definition of index of a DAE [28]. Therefore, equation (2.18) can be expressed as equation (2.19).

$$s(t) = \mu^2 \ddot{w} + 2\mu \dot{w} + w \tag{2.19}$$

The time-varying surface $s(t)$ is the SMC controller, which forces the motion to the desired dynamics. In order to obtain the vector of Lagrange multipliers z, the derivative of equation (2.18) w.r.t. time is found as follows:

$$\dot{s}(t) = \mu^2 \dddot{w} + 2\mu \ddot{w} + \dot{w} \tag{2.20}$$

Where, the time derivative of acceleration constraint equation $\dddot{w}$ yields

$$\dddot{\mathbf{w}} = \dot{\mathbf{J}}\ddot{x} + J\dddot{x} + \ddot{J}\dot{x} + \dot{J}\ddot{x} = 2\dot{J}\ddot{x} + J\dddot{x} + \ddot{J}\dot{x} = 0 \tag{2.21}$$

According to equation (2.12) and (2.13),

$$\begin{aligned} \mathbf{J}\dddot{x} &= \mathbf{JM}^{-1}(\dot{F}_{\text{ext}} + \dot{F}_{\text{con}}) \\ &= \mathbf{JM}^{-1}\dot{F}_{\text{ext}} + \mathbf{JM}^{-1}(\dot{\mathbf{J}}^{\mathrm{T}} z + \mathbf{J}^{\mathrm{T}} \dot{z}) \\ &= \mathbf{JM}^{-1}\dot{F}_{\text{ext}} + \mathbf{JM}^{-1}\dot{\mathbf{J}}^{\mathrm{T}} z + \mathbf{JM}^{-1}\mathbf{J}^{\mathrm{T}} \dot{z} \end{aligned} \tag{2.22}$$

Substituting equation (2.22) into equation (2.21) obtains:

$$\dddot{w} = \dot{\mathbf{J}}\ddot{x} + \mathbf{J}\dddot{x} + \ddot{\mathbf{J}}\dot{x} + \dot{\mathbf{J}}\ddot{x} = \mathbf{JM}^{-1}\mathbf{J}^{\mathrm{T}}\dot{z} + \mathbf{JM}^{-1}\dot{F}_{\text{ext}} + \mathbf{JM}^{-1}\dot{\mathbf{J}}^{\mathrm{T}} z + 2\dot{\mathbf{J}}\ddot{x} + \ddot{\mathbf{J}}\dot{x} = 0 \tag{2.23}$$

Substituting equation (2.23) into equation (2.20) obtains:

$$\dot{s}(t) = \mu^2 \mathbf{JM}^{-1}\mathbf{J}^{\mathrm{T}}\dot{z} + \mu^2(\mathbf{JM}^{-1}\dot{F}_{\text{ext}} + \mathbf{JM}^{-1}\dot{\mathbf{J}}^{\mathrm{T}} z + 2\dot{\mathbf{J}}\ddot{x} + \ddot{\mathbf{J}}\dot{x}) + 2\mu\ddot{w} + \dot{w} \tag{2.24}$$

Now defining

$$\mathbf{J}_s = \mu^2 \mathbf{JM}^{-1}\mathbf{J}^{\mathrm{T}} \tag{2.25}$$

$$a = \mu^2(2\dot{\mathbf{J}}\ddot{x} + \mathbf{JM}\dot{F}_{\text{ext}} + \mathbf{JM}^{-1}\dot{\mathbf{J}}^{\mathrm{T}} z + \ddot{\mathbf{J}}\dot{x}) + 2\mu\ddot{w} + \dot{w} \tag{2.26}$$

Equation (2.24) can be written as:

$$\dot{s}(t) = \mathbf{J}_s \dot{z} + a \tag{2.27}$$

If the equation is solved for $\dot{z}$, that is, $\dot{s}(t) = 0$, the sliding motion can be steered into the desired boundary layer, and then the vector of Lagrange multipliers z is obtained by integrating $\dot{z}$. At this stage the idea from SMC is incorporated with the constraint dynamics of multi-body systems, and one can make use of its robustness properties.

Since computation of exact a can be potentially expensive, it is approximated by $\hat{a}$ that will be interpreted later. In order to satisfy the sliding condition, a term discontinuous across the surface $s(t)=0$ is added to $\dot{z}$.

$$\mathbf{J}_s\dot{z} = -\hat{a} - \mathrm{K}\,\mathrm{sign}(s) \tag{2.28}$$

$$\begin{aligned} \mathrm{sign}(s) &= +1 \quad \text{If } s>0 \\ \mathrm{sign}(s) &= -1 \quad \text{If } s<0 \end{aligned} \tag{2.29}$$

Substituting equation (2.28) in equation (2.27) obtains:

$$\dot{s}(t) = -\hat{a} + a - \mathrm{K}\,\mathrm{sign}(s) \tag{2.30}$$

Note that the control discontinuity K across the surface $s(t)=0$ increases with the extent of parametric uncertainty that leads in practice to control chattering. For the controller to perform properly, chattering must be eliminated, which can be achieved by smoothing out the control discontinuity in a thin boundary layer neighbouring the switching surface $s(t)$. In smoothed implementation:

$$\mathbf{J}_s\dot{z} = -\hat{a} - \mathrm{K}\,\mathrm{sat}(s/\varepsilon) \tag{2.31}$$

$$\begin{aligned} \mathrm{sat}(y) &= y \qquad \text{If } |y| \le 1 \\ \mathrm{sat}(y) &= \mathrm{sign}(y) \quad \text{Otherwise} \end{aligned} \tag{2.32}$$

The $\mathrm{sat}(y)$ is the saturation function, which is used to smooth the control and eliminate chattering phenomenon common to sliding mode control method. Substituting equation

(2.31) in equation (2.27) obtains:

$$\dot{s} = -\hat{\alpha} + \alpha - \mathrm{K}\mathrm{sat}(s/\varepsilon) \tag{2.33}$$

2.3.2 Choice of the parameters in SMC

2.3.2.1 Choice of $\hat{\alpha}$

$\hat{\alpha} = \alpha$ is the best choice for the precision of program, but the exact expression of α, which is shown in equation (2.26), is fairly complex and takes lots of time to compute. In order to decrease the computational overhead, the representation of the parameter α is simplified as $\hat{\alpha}$, and let

$$\hat{\alpha} = 2\mu\ddot{\mathbf{w}} + \dot{\mathbf{w}} \tag{2.34}$$

One can notice that $\ddot{w}$ and $\dot{w}$ have already been evaluated in the process of calculating s, so calculation of $\hat{\alpha}$ does not involve much computational overhead.

2.3.2.2 Choice of μ

The parameter μ has two kind of functionality. Firstly, it decides the response speed of the system, that is, the response speed of the system decreases as u decreases. Secondly, it affects bonds on the regulation error vector $\tilde{w}$ [7], which is translated from bounds on the switching surface s stated as follows:

$$\forall t \geq t_r,\ |s(t)| \leq \varepsilon \Rightarrow \forall t \geq t_r, |w| \leq \varepsilon,\ |\dot{w}(t)| \leq \frac{2\varepsilon}{\mu},\ |\ddot{w}(t)| \leq \frac{4\varepsilon}{\mu^2} \tag{2.35}$$

Where, t_r is the reaching time, after which SMC guarantees the surface $s(t)$ is reached within the desired accuracy bound, $|s(t)| \le \varepsilon$, and corresponding regulation error vector $\tilde{w}$ also is limited to some extent shown in equation (2.35). Note that initial conditions are not necessarily consistent in the above lemma.

2.3.2.3 Choice of K

In order to keep s at zero, the sliding condition of equation (Lyapunov function) (2.36) must be satisfied [4]:

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{dt}}s^2 \le -\eta|s| \tag{2.36}$$

Where, η is a strictly positive constant. By choosing K to be large enough, the sliding condition is guaranteed. According to equation (2.33), the equation (2.36) is

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{dt}}s^2 = \dot{s}s = (\alpha - \hat{\alpha} - \mathrm{K}\mathrm{sat}(s/\varepsilon))s = (\alpha - \hat{\alpha})s - \mathrm{K}\mathrm{sat}(s/\varepsilon)s \le -\eta|s| \tag{2.37}$$

So, letting

$$\mathrm{K} \ge |\alpha - \hat{\alpha}| + \eta \tag{2.38}$$

We get as desired, and the sufficient condition that from equation (2.38) needed to meet becomes

$$\mathrm{K} \ge |\alpha - \hat{\alpha}| + \eta \ge \mu^2 \mid (2\dot{\mathbf{J}}\ddot{x} + \mathbf{J}\mathbf{M}\dot{F}_{\mathrm{ext}} + \mathbf{J}\mathbf{M}^{-1}\dot{\mathbf{J}}^{\mathrm{T}}z + \ddot{\mathbf{J}}\dot{x}) \mid + \eta \tag{2.39}$$

All in all, $|\alpha - \hat{\alpha}| + \eta = \nabla$ is the approximate error. If a large enough K is chosen, parametric inaccuracies are compensated for, and we obtain $s \to 0$. From equation (2.39), one also can see that there is some kind of relationship between μ and K.

2.3.2.4 Choice of time step size Δt

In applying SMC method, in order to get the exact results of simulations, that is $w = 0$

and $\dot{w} = 0$, two conditions should be satisfied. One is that K is high enough, and the other is that time step size Δt is as small as possible. However, if time step size is too small, real-time simulations consume an extensive amount of CPU time. Compared with the exact method, a larger time step size is needed to get a higher efficiency. Therefore, the choice of time step size is a dilemma.

2.4 Simulation loop

The dynamic equations of a multi-body system can be expressed as equation (2.40) and (2.41), and Fig. 2-1 describes the steps for solving these equations.

$$\mathbf{M}\ddot{\boldsymbol{x}} + \boldsymbol{b} = \boldsymbol{c} + \mathbf{J}^{\mathrm{T}}\boldsymbol{z} \tag{2.40}$$

$$\mathbf{J}_{\mathrm{s}}\dot{\boldsymbol{z}} = -\hat{\boldsymbol{a}} - \mathrm{K}\,\mathrm{sat}(\boldsymbol{s}/\varepsilon) \tag{2.41}$$

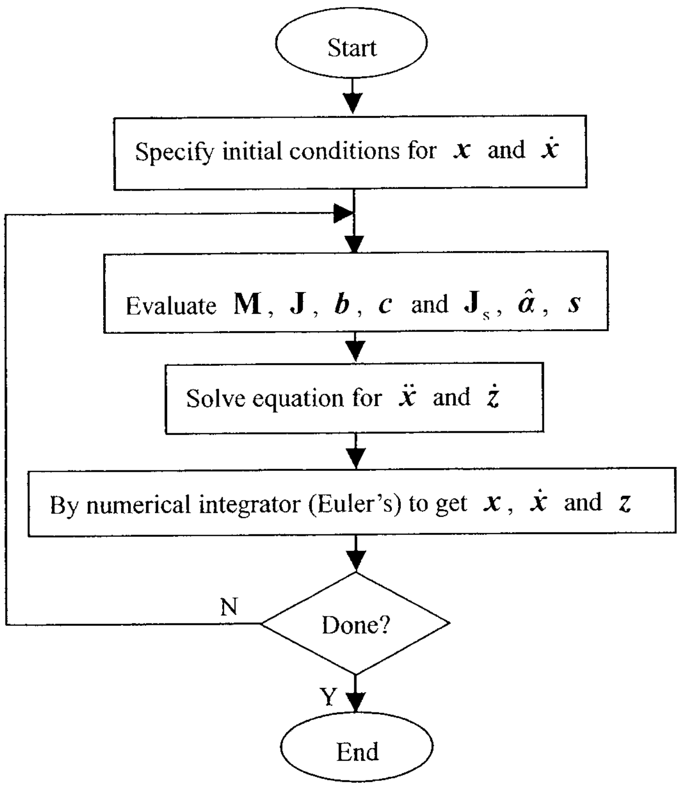

Start
Specify initial conditions for x and $\dot{x}$
Evaluate $\mathbf{M}$, $\mathbf{J}$, $\boldsymbol{b}$, c and $\mathbf{J}_s$, $\hat{a}$, s
Solve equation for $\ddot{x}$ and $\dot{z}$
By numerical integrator (Euler's) to get x, $\dot{x}$ and z
Done?
N
Y
End

Fig. 2-1 Simulation loop

Note that this method can work with any explicit integration algorithm (such as Runge-Kutta) in addition to Euler, which is used in our experiments.

3 A Dynamic Simplex Approach

Dynamic/online optimization is an effective method for performance improvement in distributed systems. Distributed simulation of multi-body systems is a time-dependent process, whose requirements must always be expressed in connection with time. Therefore, any time delay produced in the system can influence the simulation results and stability. So, the relationship between the allocated time delays of nodes and the performance of system is studied. Fig. 3-1 shows the process to conduct online optimization for a distributed system. Firstly, the data of position constraints is extracted from the distributed system, and then the controller deals with these data and obtains the optimal time delays, which are used for optimization to generate the optimal set points that will minimize the position constraints. The controller is based on the algorithm of dynamic simplex method (DSM) [26]. The main reasons we use the DSM algorithm for online optimization are stated as follows:

- Compared with gradient methods, it requires no computation of gradients but just function evaluations.
- Compared with SQP and empirical black-box model based optimization approaches, it does not require process modeling to perform the optimization.
- Compared with the Nelder-Mead Simplex method, it allows tracking of the moving optimum and has less function evaluations.

However, because DSM has been developed in recent years, weak theoretical basis and less actual engineering applications are its main disadvantages compared with other common methods.

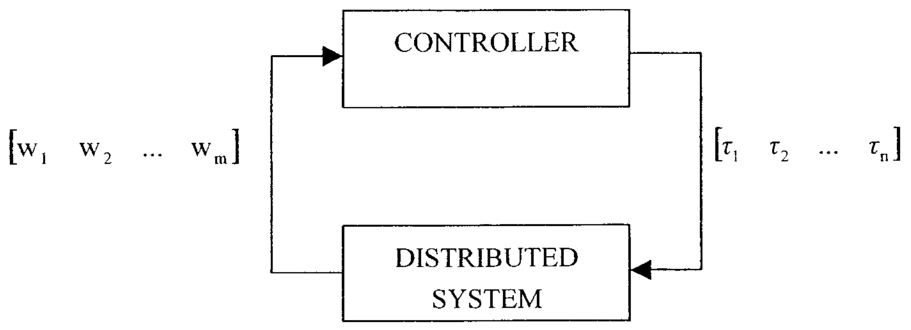

CONTROLLER
$[w_1 \ w_2 \ \dots \ w_m]$
$[\tau_1 \ \tau_2 \ \dots \ \tau_n]$
DISTRIBUTED SYSTEM

Fig. 3-1 Information flow diagram of dynamic optimization

3.1 Cost function

In this thesis, optimal allocation of bandwidth B is utilized to solve the problem of optimal cost function $C(\boldsymbol{\tau})$, and it is used to establish a relationship between the allocated time delays of nodes and the performance. Therefore, the overall system performance can be improved by adjusting the time delays of each node.

The network bandwidth B [14] should have a limited value and is defined as

$$\sum_{i=1}^{n} \frac{k_i}{\tau_i} = B \tag{3.1}$$

where τ_i is the communication time delay and k_i is the number of bytes transferred corresponding to each time delay.

For distributed real-time simulation of multi-body systems, stabilization of which is mainly dependent on the values of $\boldsymbol{w}$, which should be as small as possible, the cost function of the system is related to $\boldsymbol{w}$. The optimization problem is to minimize the network performance cost function, which is defined as the sum of the root mean squared (RMS) value of w_i over the measurement period T and is shown in equation (3.2).

$$C(\boldsymbol{\tau}) = \sum_{i=1}^{m} \sqrt{\frac{\int_{t}^{t+T} w_i^2(u,\boldsymbol{\tau})du}{T}} \qquad \textbf{(3.2)}$$

where T is the measurement period, $\boldsymbol{\tau}$ is a vector of time delays, and u is time.

The optimization problem can now be classified into the following two types:

Type 1: Unconstrained Problem (UP)

Find a vector of optimization variables, $\boldsymbol{\tau} = (\tau_1, \tau_2 \ldots, \tau_n)^T$, in order to minimize a cost function

$$C(\boldsymbol{\tau}) = \sum_{i=1}^{m} \sqrt{\frac{\int_{t}^{t+T} w_i^2(u,\boldsymbol{\tau})du}{T}} + b\sum_{i=1}^{n} \frac{k_i}{\tau_i} \qquad \textbf{(3.3)}$$

Where, b is a suitable weighting factor. The success of the method clearly depends on a clever choice of the weighting factor, and the optimization problem is to minimize the cost function $C(\boldsymbol{\tau})$.

Type 2: Constrained Problem (CP)

The optimization problem can also be expressed in the following form: find a vector of optimization variables, $\boldsymbol{\tau} = (\tau_1, \tau_2 \ldots, \tau_n)^T$, in order to minimize a cost function

$$C(\boldsymbol{\tau}) = \sum_{i=1}^{m} \sqrt{\frac{\int_{t}^{t+T} w_i^2(u, \boldsymbol{\tau}) du}{T}}, \text{ subject to } \sum_{i=1}^{n} \frac{k_i}{\tau_i} \leq B \quad . \tag{3.4}$$

In this situation, the bandwidth is less than or equal to a given value.

3.2 Controller design

The optimization algorithm forms the core of online optimization. The algorithm of the controller is based on dynamic simplex method (DSM) developed by Xiong and Jutan [26], the foundation of which is Nelder-Mead simplex method described in the following section.

3.2.1 Nelder-Mead simplex method

Nelder-Mead Simplex method is one of the most popular direct search methods, and it requires no derivatives but function evaluations, that is, it can be performed directly without the help of the system's model.

A simplex S is defined as a convex hull, which consists of N+1 points in N dimensions, so a simplex is a triangle in the two-dimensional case. In general, non-degenerated condition of the simplex should be satisfied, so that the volume of the simplex is nonzero.

The Nelder–Mead simplex method is an iterative algorithm with iteration k stated as follows:

1. The start simplex is set as $\boldsymbol{S}_0 = \{\boldsymbol{x}_j\}_{j=1}^{N+1}$ in iteration k and the cost function values on these vertices are $\{C_j\}_{j=1}^{N+1}$.

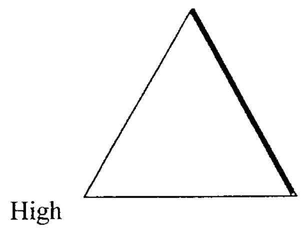
2. Sort the vertices of the simplex such that $C_1 \geq C_2 \geq ...... \geq C_{N+1}$. Because we are going to minimize the cost function C, $\boldsymbol{x}_{N+1}$ is referred as the best point, and $\boldsymbol{x}_1$ is referred as the worst point.

3. A series of operations are taken to reach the optimum, including reflection, expansion, contraction, or shrinkage (multiple contraction), and Fig. 3-2 gives a demonstration of a two-dimensional case. If the iteration finishes after reflection, it needs only one function evaluation; two function evaluations if termination after expansion or contraction; N+1 function evaluations if done after shrinkage. Therefore, a less number of measurements or function evaluations are required in each iteration compared with the other methods.

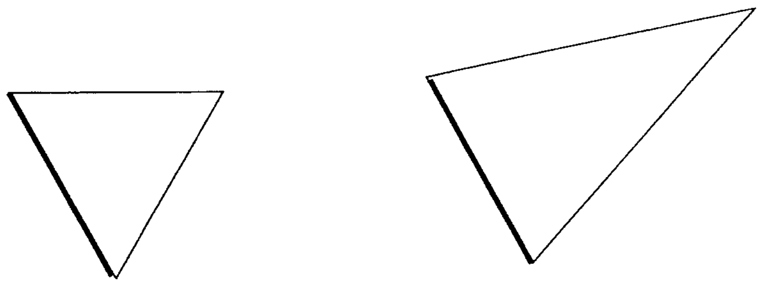
4. Test for convergence: If convergent conditions are satisfied, iteration is terminated.

5. Start the next iteration with the new simplex, which is chosen as the minimum function evaluation.

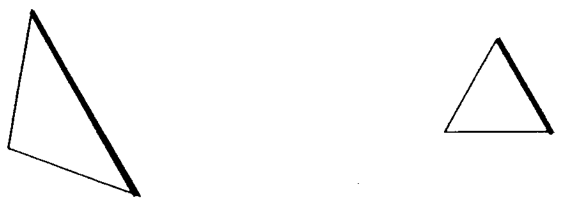
However, the Nelder-Mead simplex method cannot track the moving optimum, so its application is limited to the problem with a static optimum. Hence, Xiong and Jutan [26] have developed the dynamic simplex method (DSM) to track a moving optimum continuously.



(a) The beginning simplex



(b) Reflection (c) Reflection and expansion



(d) Contraction (e) Multiple contraction

Fig. 3-2 Possible operation for a step in the Nelder-Welder Simplex method

3.2.2 Dynamic simplex method

The dynamic simplex method (DSM) is based on Nelder-Mead simplex method. Compared with the Nelder-Mead Simplex method, DSM uses a fixed simplex size and allows tracking of moving optimum. DSM is an iterative algorithm with iteration k stated as following steps:

1. $\boldsymbol{S}_0 = \{\boldsymbol{x}_j\}_{j=1}^{N+1}$ is referred as the start simplex in iteration k and the cost function values on these vertices are $\{C_j\}_{j=1}^{N+1}$.

2. Make a queue of the vertices of the simplex such that $C_1 \geq C_2 \geq ...... \geq C_{N+1}$.

3.After m_r successive reflections , we have a series of new simplexes $\{\boldsymbol{S}_p\}_{p=1}^{m_r}$.

The reflection operation in a two-dimensional case is demonstrated in Fig. 3-3. After reflecting the worst (first) point x_1 of the start simplex S_0, one can get a new point x_4 and a simplex S_1. And then keep reflecting the first point of the newly generated simplex, so that S_2 and S_3 are obtained by the reflection operation of $x_2 \rightarrow x_5$ and $x_4 \rightarrow x_6$.

4. The start simplex of next iteration is selected.

Through equation (3.5), the average cost function value of these new simplexes $\{\boldsymbol{S}_p\}_{p=1}^{m_r}$ is computed.

$$\overline{C}_{S_p} = \frac{1}{N+1} \sum_{j=p+1}^{N+p+1} C_j \quad p = 1,2......,m_r \tag{3.5}$$

And then, the simplex $\boldsymbol{S}_q$ that satisfies $\overline{C}_{S_p} = \min(\overline{C}_{S_q})_{p=1}^{m_r}$ is chosen.

5. Measure again the response at point $\boldsymbol{x}_{N+1}$.

6. Start the next iteration with the new simplex $\boldsymbol{S}_q$

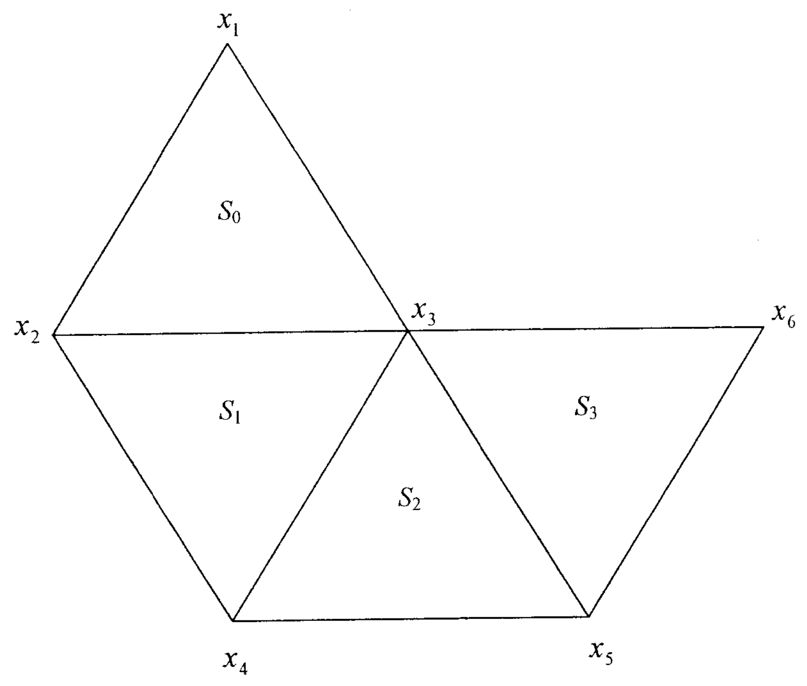


Fig. 3-3 Successive reflection in two-dimensional space

The difference between DSM and Nelder-Mead simplex method is stated as follows:

1. Unlike the Nelder-Mead simplex method, DSM has no step to test for convergence. There is a static point to converge to in the algorithm of Nelder-Mead simplex method, so this algorithm needs a step to test for convergence. However, the DSM algorithm is used to track a moving optimum continuously (no static point to converge to) and its cost function values are varying as time goes, so testing convergence is not necessary.

2. DSM has a fixed simplex size compared with the Nelder-Mead simplex method with a variable simplex size, so only reflection of the simplex is permitted. There are two reasons for choosing a fixed simplex size. One is keeping the simplex size large enough

to get the right value of the measurement contaminated by noise. The other is to conserve the volume of the simplex and maintain its non-degeneracy.

3. There is a difference in the choice of the new start simplex in the next iteration. In the Nelder-Mead simplex method, it is chosen as the minimum function evaluation. And in DSM, it is only determined by selecting the minimum average function value of the newly generated simplexes. This is because the DSM algorithm is trying to track the moving optimum, and sometimes it may have an increased function value.

4. At the end of each iteration, the Nelder-Mead method re-measures all the points for recalculating a new direction to keep better tracking of the moving optimum, which eliminates its advantage of few function evaluations although it is the best operation for finding the right direction; while DSM just re-measures the response at the old best point because it has the greatest influence on the search direction.

Although the Nelder-Mead simplex method is extraordinarily used in practice, it has very weak theoretical basis. In [39], the authors have proven convergence properties of the Nelder-Mead simplex method for dimension one and various limited convergence results for dimension two. Sufficient conditions of convergence for problems in n-dimensions using fixed simplex are presented in Torczon's paper [38]. Compared with the algorithm of Nelder-Mead simplex method, the DSM algorithm has no theoretical basis until now. But the DSM algorithm is based on engineer experience, and many simulated experiments are tested, yielding favourable results.

3.3 Selection of optimization parameters

There are several parameters in the proposed approach that significantly influence performance. These are described in detail in the following sections.

3.3.1 The measurement period T

For dealing with a moving optimum, a compromise has to be made with the value of T. T should remain small enough to reflect the changes in the system. But a very small value of T consumes an extensive amount of CPU time, so T should remain large enough to improve efficiency.

3.3.2 The number of successive reflections m_r

A large number of m_r need more function evaluations, and a small number of m_r is usually enough to trace the moving optimum.

3.3.3 The simplex size D_s

In order to keep up with a continuously moving optimum, the simplex size should remain large enough and at the same time ensure a minimum simplex size. The minimum simplex size is defined as $D_{smin} = (m_r + 1)vT$, where v is the maximum movement speed of the optimum. The choice of D_s is a dilemma. It should be chosen large enough so that the DSM algorithm can track the moving optimum, and it should not be too large so that the DSM algorithm can still achieve adequate accuracy. Therefore, a compromise has to be made.

3.4 Optimization testing

3.4.1 Model partitioning

In order to discuss how to cut the model of a multi-body system and arrange distribution equations, one example illustrated in Fig. 3-4 is introduced. The model is composed of 16 rigid bodies, which are connected by 24 rigid links. The dimension of each body is $(0.2 \times 0.2 \times 0.2)$ meters; the mass of that is 1 kg; each link is 1.8 meters long. Set $\varepsilon = 0.1$, $\mu = 0.05$, $K = 10$.

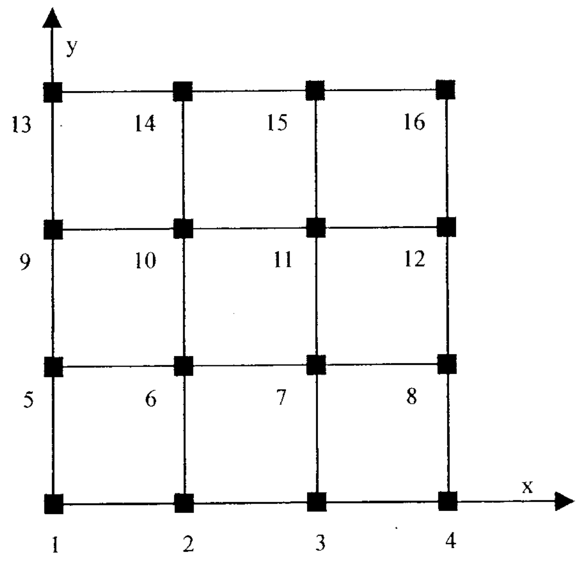


Fig. 3-4 Gridding model

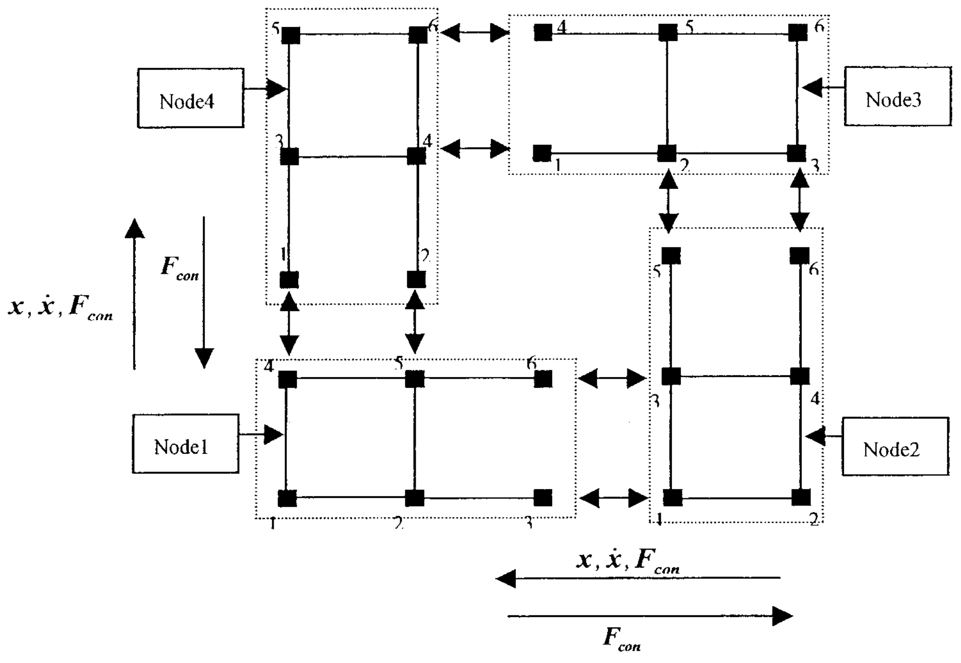


Fig. 3-5 Partitioning of gridding model

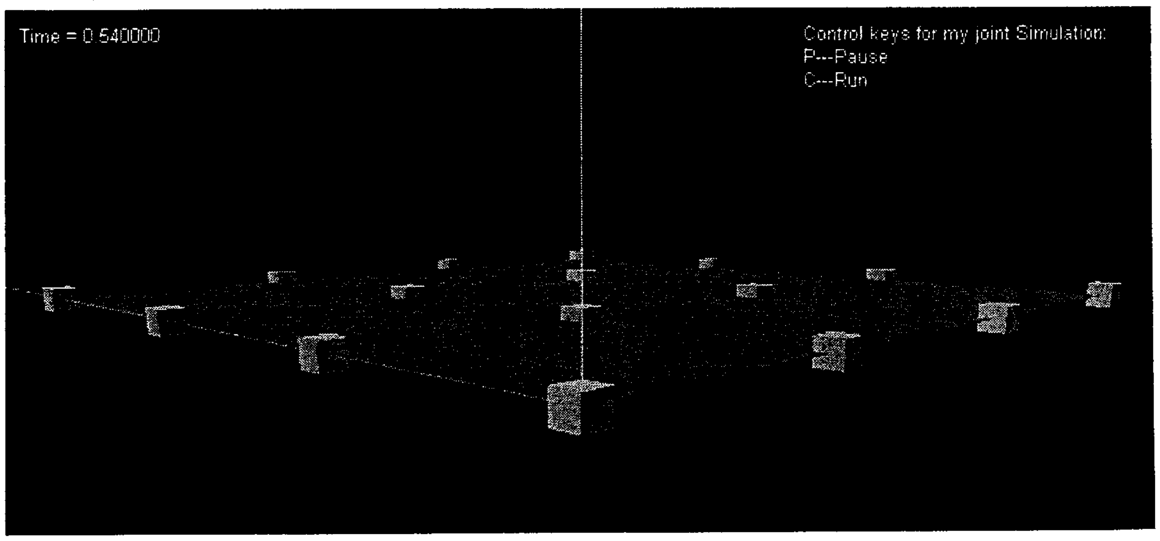


Fig. 3-6 Snapshot of simulation

Fig. 3-5 shows that the model is cut in four nodes, each of which is composed of six rigid bodies connected by six rigid links. In the partitioning method, the constraint located on the border cannot be repeated, for example, the constraint link between body 4 and 5 in node 1 is not shown in node 4 between body 1 and 2. ($\boldsymbol{x}$, $\dot{\boldsymbol{x}}$) of rigid bodies located on the border are transferred one way, while $\boldsymbol{F}_{\text{con}}$ of these is transferred bi-directionally.

The dynamic equations of node 1 can be expressed as follows:

$$\mathbf{M}\ddot{\boldsymbol{x}} + \boldsymbol{b} = \boldsymbol{c} + \mathbf{J}^{\mathrm{T}}\boldsymbol{z} \tag{3.6}$$

$$\mathbf{J}_{\mathrm{s}}\dot{\boldsymbol{z}} = -\hat{\boldsymbol{a}} - \mathrm{K}\mathrm{sat}(\boldsymbol{s}/\varepsilon) \tag{3.7}$$

Where, $\mathbf{M}$ is a (36×36) matrix shown in equation (3.8).

$$\mathbf{M} = \mathrm{diag}[\mathbf{M}_1 \quad \mathbf{M}_2 \quad . \quad . \quad \mathbf{M}_6] \tag{3.8}$$

Where, $\boldsymbol{x}$ is a (36×1) vector shown in equation (3.9).

$$\boldsymbol{x} = [\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad . \quad . \quad \boldsymbol{x}_6]^{\mathrm{T}} \tag{3.9}$$

In equations (3.10) and (3.11), where $\boldsymbol{x}_{\mathrm{i\_j}}$ and $\dot{\boldsymbol{x}}_{\mathrm{i\_j}}$ are the position and velocity vector of body j in node i respectively, that is, the linear position and velocity, the angular position and velocity of the rigid bodies located on the border in node 2 are transferred to those in node 1.

$$\boldsymbol{x}_3 = \boldsymbol{x}_{2\_1} \qquad \boldsymbol{x}_6 = \boldsymbol{x}_{2\_3} \tag{3.10}$$

$$\dot{\boldsymbol{x}}_3 = \dot{\boldsymbol{x}}_{2\_1} \qquad \dot{\boldsymbol{x}}_6 = \dot{\boldsymbol{x}}_{2\_3} \tag{3.11}$$

Where, $\mathbf{J}$ is a (6×36) matrix, and $\mathbf{J}^{\mathrm{T}}$ is a (36×6) matrix. $\mathbf{J}$ is shown in equation (3.12).

$$\mathbf{J}=\begin{bmatrix} \mathbf{J}_{1\_1} & \mathbf{J}_{1\_2} & \mathbf{J}_{1\_3} & \mathbf{J}_{1\_4} & \mathbf{J}_{1\_5} & \mathbf{J}_{1\_6} \\ \mathbf{J}_{2\_1} & \mathbf{J}_{2\_2} & \mathbf{J}_{2\_3} & \mathbf{J}_{2\_4} & \mathbf{J}_{2\_5} & \mathbf{J}_{2\_6} \\ \mathbf{J}_{3\_1} & \mathbf{J}_{3\_2} & \mathbf{J}_{3\_3} & \mathbf{J}_{3\_4} & \mathbf{J}_{3\_5} & \mathbf{J}_{3\_6} \\ \mathbf{J}_{4\_1} & \mathbf{J}_{4\_2} & \mathbf{J}_{4\_3} & \mathbf{J}_{4\_4} & \mathbf{J}_{4\_5} & \mathbf{J}_{4\_6} \\ \mathbf{J}_{5\_1} & \mathbf{J}_{5\_2} & \mathbf{J}_{5\_3} & \mathbf{J}_{5\_4} & \mathbf{J}_{5\_5} & \mathbf{J}_{5\_6} \\ \mathbf{J}_{6\_1} & \mathbf{J}_{6\_2} & \mathbf{J}_{6\_3} & \mathbf{J}_{6\_4} & \mathbf{J}_{6\_5} & \mathbf{J}_{6\_6} \end{bmatrix} \qquad \textbf{(3.12)}$$

Where z is a (6×1) vector; $\boldsymbol{b}$ and $\boldsymbol{c}$ are (36×1) vectors shown in equation (3.13), (3.14), (3.15), and (3.16). Where $F_{\mathrm{i\_con\_j}}$ and $\boldsymbol{Tq}_{\mathrm{i\_con\_j}}$ are the constraint force and torque acting on the body j in node i.

$$\boldsymbol{b}=[\boldsymbol{b}_1 \quad \boldsymbol{b}_2 \quad . \quad . \quad \boldsymbol{b}_6]^{\mathrm{T}} \qquad \textbf{(3.13)}$$

$$\boldsymbol{c}=[\boldsymbol{c}_1 \quad \boldsymbol{c}_2 \quad . \quad . \quad \boldsymbol{c}_6]^{\mathrm{T}} \qquad \textbf{(3.14)}$$

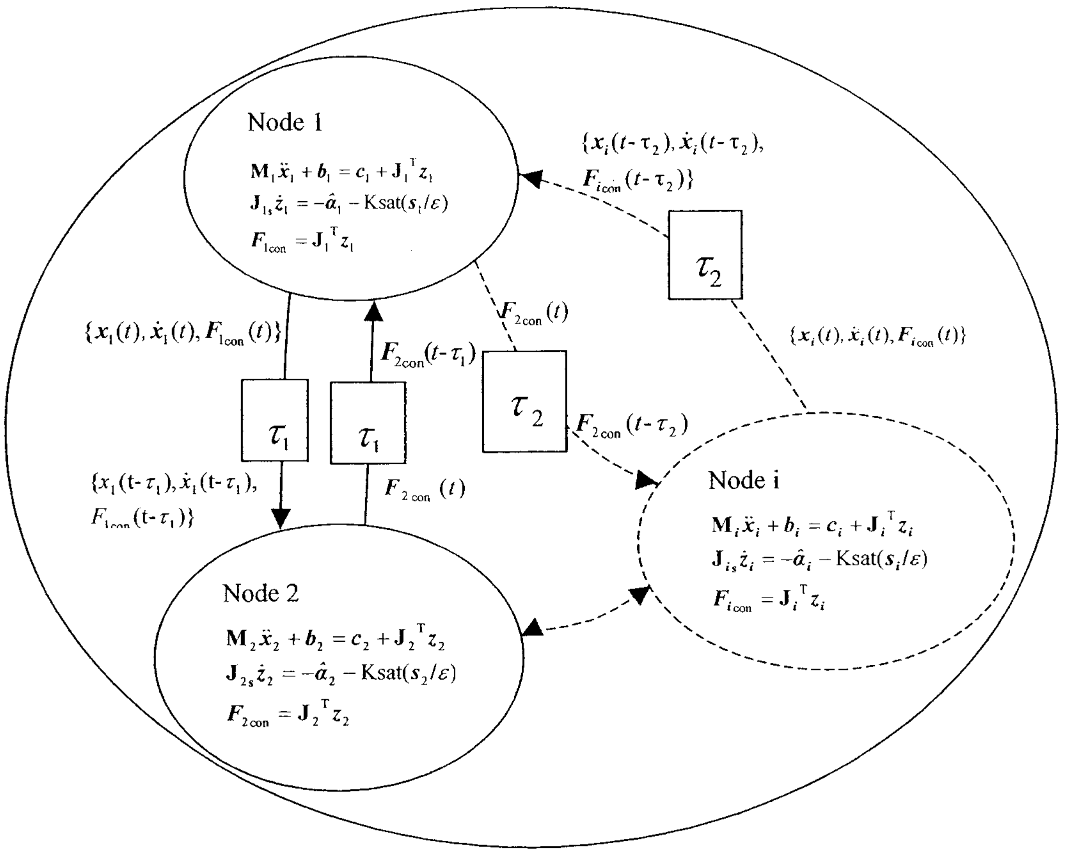
$$\begin{gathered} \boldsymbol{c}_3=\begin{bmatrix} \boldsymbol{F}_{\mathrm{g3}}+\boldsymbol{F}_{2\_\mathrm{con\_1}} \\ \boldsymbol{Tq}_{\mathrm{g3}}+\boldsymbol{Tq}_{2\_\mathrm{con\_1}} \end{bmatrix} \quad \boldsymbol{c}_4=\begin{bmatrix} \boldsymbol{F}_{\mathrm{g4}}+\boldsymbol{F}_{4\_\mathrm{con\_1}} \\ \boldsymbol{Tq}_{\mathrm{g4}}+\boldsymbol{Tq}_{4\_\mathrm{con\_1}} \end{bmatrix} \\ \boldsymbol{c}_5=\begin{bmatrix} \boldsymbol{F}_{\mathrm{g5}}+\boldsymbol{F}_{4\_\mathrm{con\_2}} \\ \boldsymbol{Tq}_{\mathrm{g5}}+\boldsymbol{Tq}_{4\_\mathrm{con\_2}} \end{bmatrix} \quad \boldsymbol{c}_6=\begin{bmatrix} \boldsymbol{F}_{\mathrm{g6}}+\boldsymbol{F}_{2\_\mathrm{con\_3}} \\ \boldsymbol{Tq}_{\mathrm{g6}}+\boldsymbol{Tq}_{2\_\mathrm{con\_3}} \end{bmatrix} \end{gathered} \qquad \textbf{(3.15)}$$

$$\boldsymbol{c}_{\mathrm{i}}=\begin{bmatrix} \boldsymbol{F}_{\mathrm{gi}} \\ \boldsymbol{Tq}_{\mathrm{gi}} \end{bmatrix} \quad \text{For } \mathrm{i}=1,2 \qquad \textbf{(3.16)}$$

This distribution system is examined in the TDMA network, which is based on the work of Lu [27] and will be interpreted later. Finally, experiments verify that the results of the distributed real-time TDMA networked simulation coincide with those of the single computer simulation. Therefore, the above model partitioning and the equations of the distributed multi-body system are adapted to the distribution simulation.

3.4.2 Time-delay distribution

Two different time-delay distributions are introduced. The first one is space distribution, that is, a certain area of the system has the same time delay. Fig. 3-7 gives a demonstration of space distribution of time delays, the area between node 1 and 2 is set as time delay 1, and the area between node 1 and i is set as time delay 2. The other is time distribution, that is, certain parameters transferred have the same time delay. Fig. 3-8 illustrates the time distribution of time delays, $\boldsymbol{F}_{con}$ is related with time delay 2, and $(\boldsymbol{x}, \dot{\boldsymbol{x}})$ are related with time delay 1.



Node 1
$\mathbf{M}_1\ddot{x}_1 + b_1 = c_1 + \mathbf{J}_1^{\mathrm{T}} z_1$
$\mathbf{J}_{1s}\dot{z}_1 = -\hat{a}_1 - \mathrm{Ksat}(s_1/\varepsilon)$
$F_{1\mathrm{con}} = \mathbf{J}_1^{\mathrm{T}} z_1$
Node 2
$\mathbf{M}_2\ddot{x}_2 + b_2 = c_2 + \mathbf{J}_2^{\mathrm{T}} z_2$
$\mathbf{J}_{2s}\dot{z}_2 = -\hat{a}_2 - \mathrm{Ksat}(s_2/\varepsilon)$
$F_{2\mathrm{con}} = \mathbf{J}_2^{\mathrm{T}} z_2$
Node i
$\mathbf{M}_i\ddot{x}_i + b_i = c_i + \mathbf{J}_i^{\mathrm{T}} z_i$
$\mathbf{J}_{is}\dot{z}_i = -\hat{a}_i - \mathrm{Ksat}(s_i/\varepsilon)$
$F_{i\mathrm{con}} = \mathbf{J}_i^{\mathrm{T}} z_i$
$\{x_1(t), \dot{x}_1(t), F_{1\mathrm{con}}(t)\}$
$\{x_1(t-\tau_1), \dot{x}_1(t-\tau_1), F_{1\mathrm{con}}(t-\tau_1)\}$
$F_{2\mathrm{con}}(t-\tau_1)$
$F_{2\mathrm{con}}(t)$
τ_1
τ_1
τ_2
τ_2
$F_{2\mathrm{con}}(t)$
$F_{2\mathrm{con}}(t-\tau_2)$
$\{x_i(t-\tau_2), \dot{x}_i(t-\tau_2), F_{i\mathrm{con}}(t-\tau_2)\}$
$\{x_i(t), \dot{x}_i(t), F_{i\mathrm{con}}(t)\}$

Fig. 3-7 Space distribution of time delays

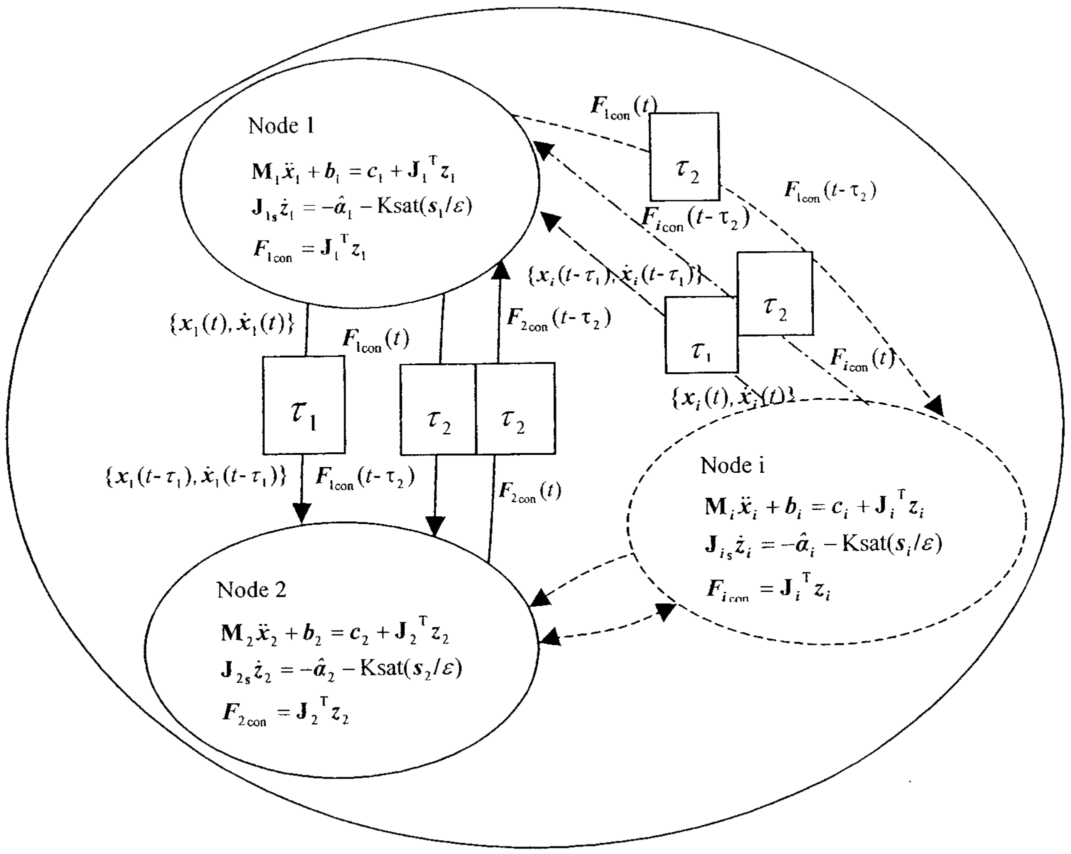

Node 1
$\mathbf{M}_1\ddot{x}_1 + b_1 = c_1 + \mathbf{J}_1^{\mathrm{T}} z_1$
$\mathbf{J}_{1s}\dot{z}_1 = -\hat{a}_1 - \mathrm{Ksat}(s_1/\varepsilon)$
$F_{1\mathrm{con}} = \mathbf{J}_1^{\mathrm{T}} z_1$
Node 2
$\mathbf{M}_2\ddot{x}_2 + b_2 = c_2 + \mathbf{J}_2^{\mathrm{T}} z_2$
$\mathbf{J}_{2s}\dot{z}_2 = -\hat{a}_2 - \mathrm{Ksat}(s_2/\varepsilon)$
$F_{2\mathrm{con}} = \mathbf{J}_2^{\mathrm{T}} z_2$
Node i
$\mathbf{M}_i\ddot{x}_i + b_i = c_i + \mathbf{J}_i^{\mathrm{T}} z_i$
$\mathbf{J}_{is}\dot{z}_i = -\hat{a}_i - \mathrm{Ksat}(s_i/\varepsilon)$
$F_{i\mathrm{con}} = \mathbf{J}_i^{\mathrm{T}} z_i$
$\{x_1(t), \dot{x}_1(t)\}$
$F_{1\mathrm{con}}(t)$
τ_1
τ_2
τ_2
$\{x_1(t-\tau_1), \dot{x}_1(t-\tau_1)\}$
$F_{1\mathrm{con}}(t-\tau_2)$
$F_{2\mathrm{con}}(t-\tau_2)$
$F_{2\mathrm{con}}(t)$
$F_{1\mathrm{con}}(t)$
τ_2
$F_{1\mathrm{con}}(t-\tau_2)$
$F_{i\mathrm{con}}(t-\tau_2)$
$\{x_i(t-\tau_1), \dot{x}_i(t-\tau_1)\}$
τ_1
τ_2
$F_{i\mathrm{con}}(t)$
$\{x_i(t), \dot{x}_i(t)\}$

Fig. 3-8 Time distribution of time delays

The model in Fig. 3-4 is examined, and the external forces acting on body 1 and 16 are set as $F = (f_x, f_y, f_z) = \begin{cases} (0,0,20\sin(2t)) \\ (0,0,0) \end{cases}$ N, which is described in Fig. 3-9. Set $\mathrm{k}_i = 1$; the measurement period T is chosen as 1 second; and time step size is equal to 0.001s; bandwidth is set as 400 Hz. The number of successive reflections $\mathrm{m_r}$ is chosen as 5; the maximum movement speed of the optimum $\mathrm{v} = 0.001$; and the minimum simplex size $\mathrm{D_{smin}} = (\mathrm{m_r} + 1)\mathrm{vT} = 0.006$, and $\mathrm{D_s} = 0.012$.

3.4.2.1 Space distribution of time delays

In the area between node 1 & 4, node 1 & 2, the time delay is same and is set as τ_1, so the transferring variables are $x(t-\tau_1)$, $\dot{x}(t-\tau_1)$ and $F_{con}(t-\tau_1)$. Similarly, the time delay is referred to τ_2 in the area between node 2 & 3, node 3 & 4. CP and UP are studied respectively.

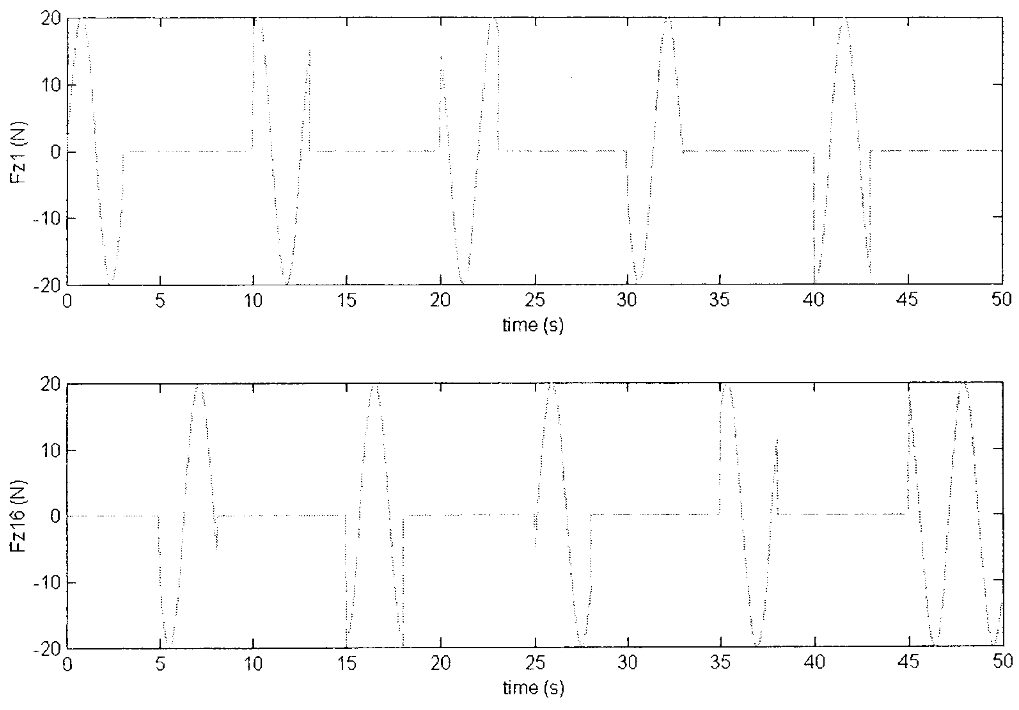


Fig. 3-9 Forces acting on the model

3.4.2.1.1 Constrained optimization problem

Bandwidth is constant in this testing. A two-variable problem with constraint is considered, that is, we control the variable τ_1 and set $\tau_2 = 1/(B - 1/\tau_1)$. Similarly, a multi-variable problem with constraint can be studied. For example, in a three-variable problem with constraint, we can control the variables τ_1 and τ_2, and set $\tau_3 = 1/(B - 1/\tau_1 - 1/\tau_2)$.

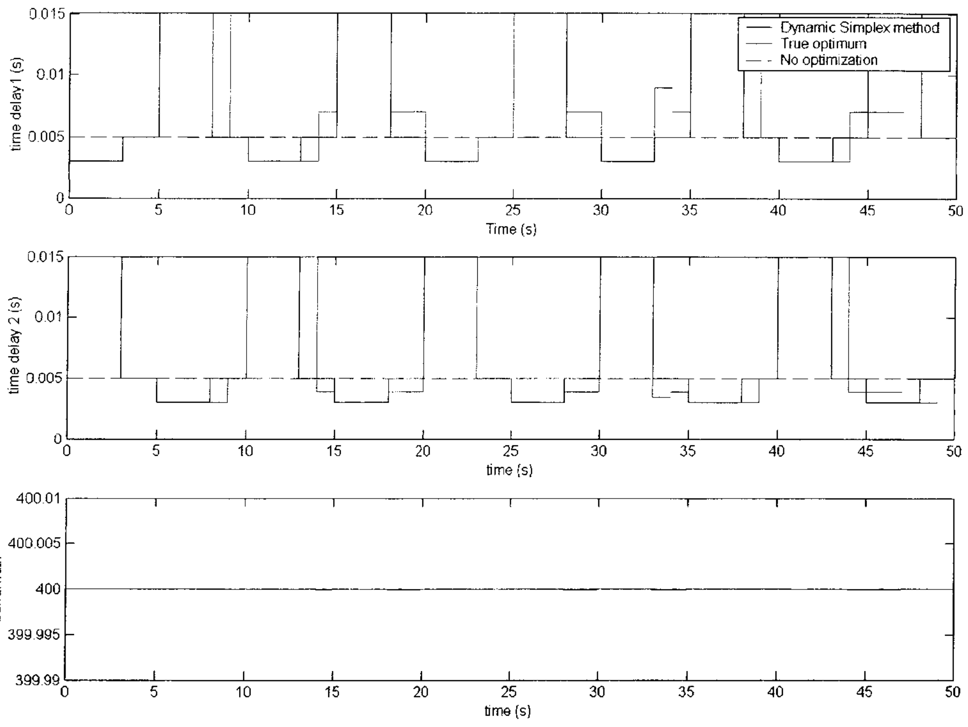


Fig. 3-10 Time delay 1, 2, and bandwidth versus time

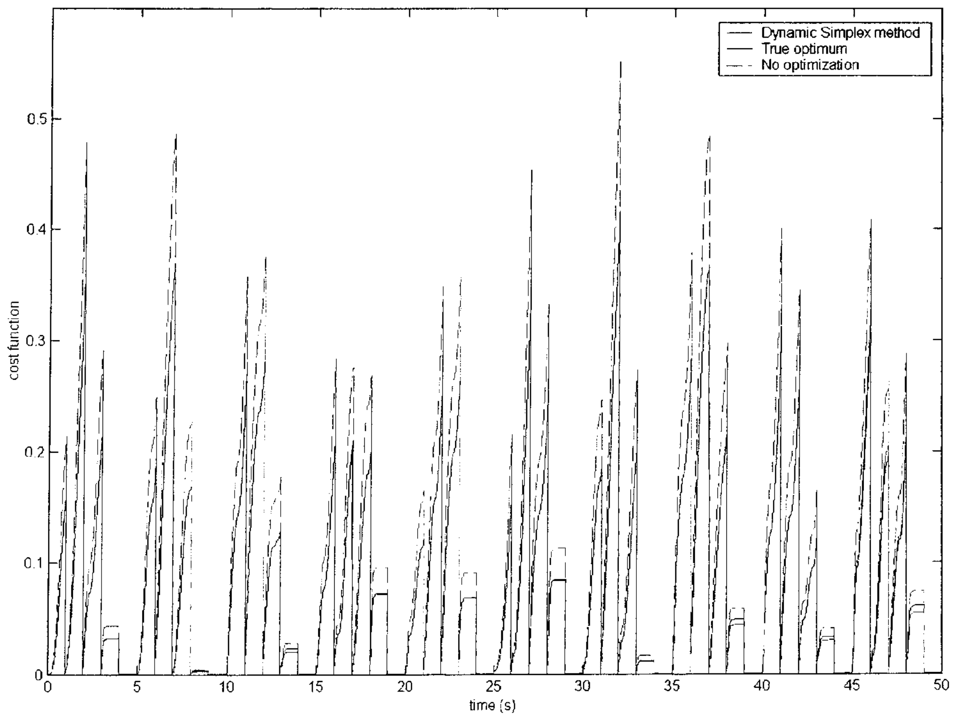


Fig. 3-11 Cost function value versus time

The experimental results of truce optimum and no optimization are used to make comparison with the DSM algorithm. The result of truce optimum is obtained by grid search method (see Appendix A), and no optimization means time delay 1 is equal to time delay 2. It can be seen from Fig. 3-11 that the result of DSM follows the true optimal value closely, and the result of no optimization does not. Moreover, the following conclusions can be obtained from Fig. 3-10 and Fig. 3-11.

- DSM can catch the optimum and follow its movement continuously
- As the value of the neighbouring force is bigger than that of the other farther force, the time delay of the nearby is smaller than that of the later.
- The results clearly show the cost function value is nearly zero as external forces acting are zero.

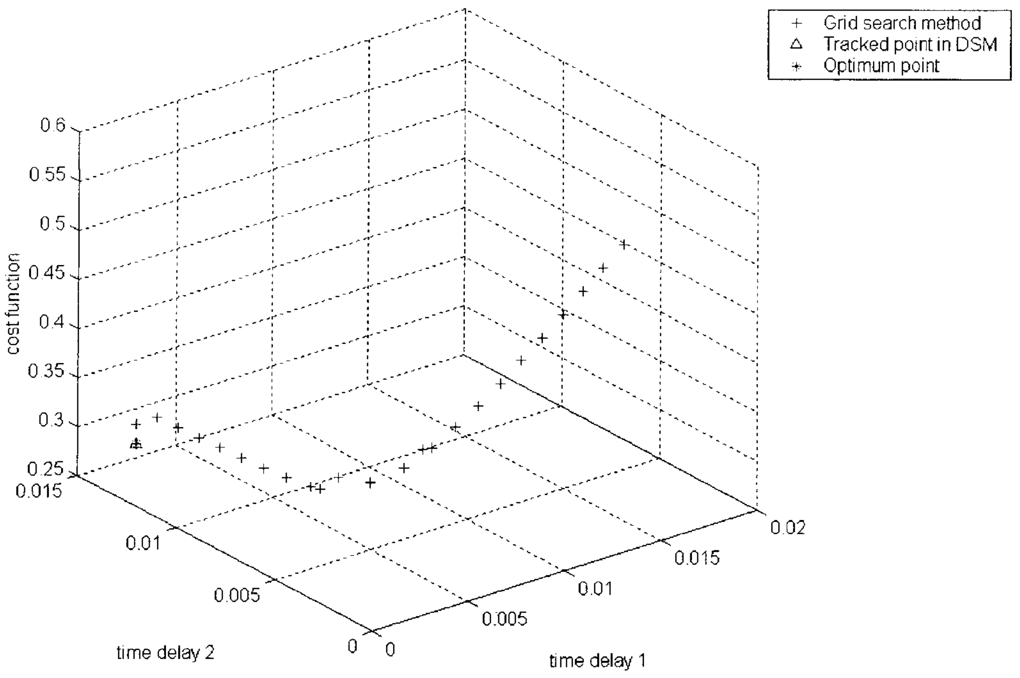


Fig. 3-12 Grid search method testing from 10s to 11s

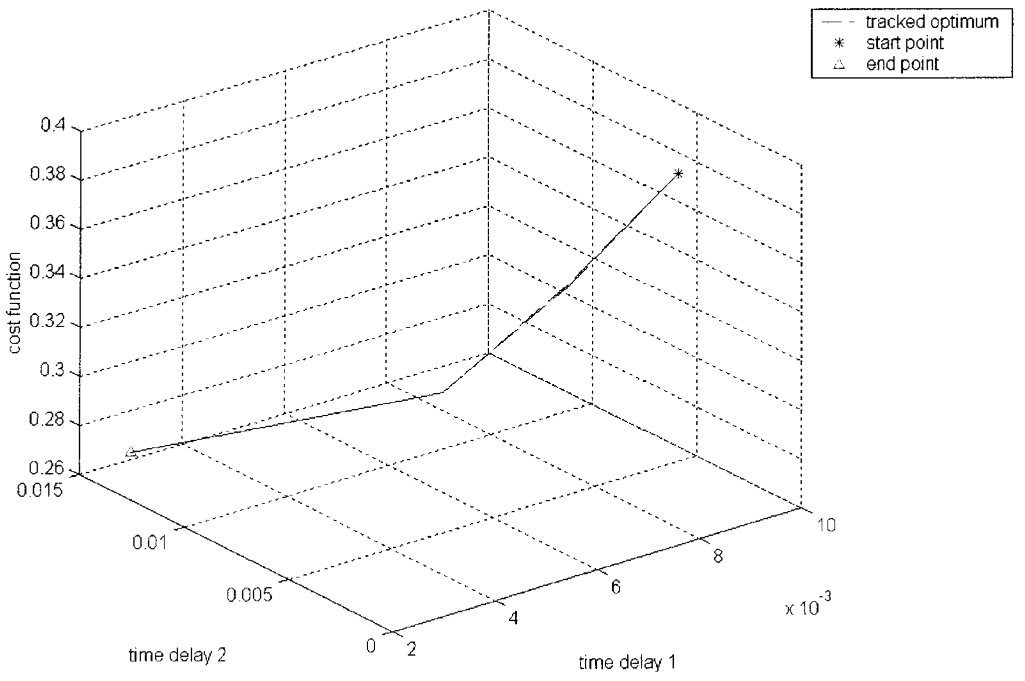


Fig. 3-13 Reflection operations from 10s to 11s

In applying grid search method, the optimum point (0.003, 0.015) from 10s to 11s is found and is consistent with the results shown in Fig. 3-10 and Fig. 3-11. Fig. 3-13 shows the reflection operations (from 10s to 11s) from start point to end point, and the end point is equal to the optimum point.

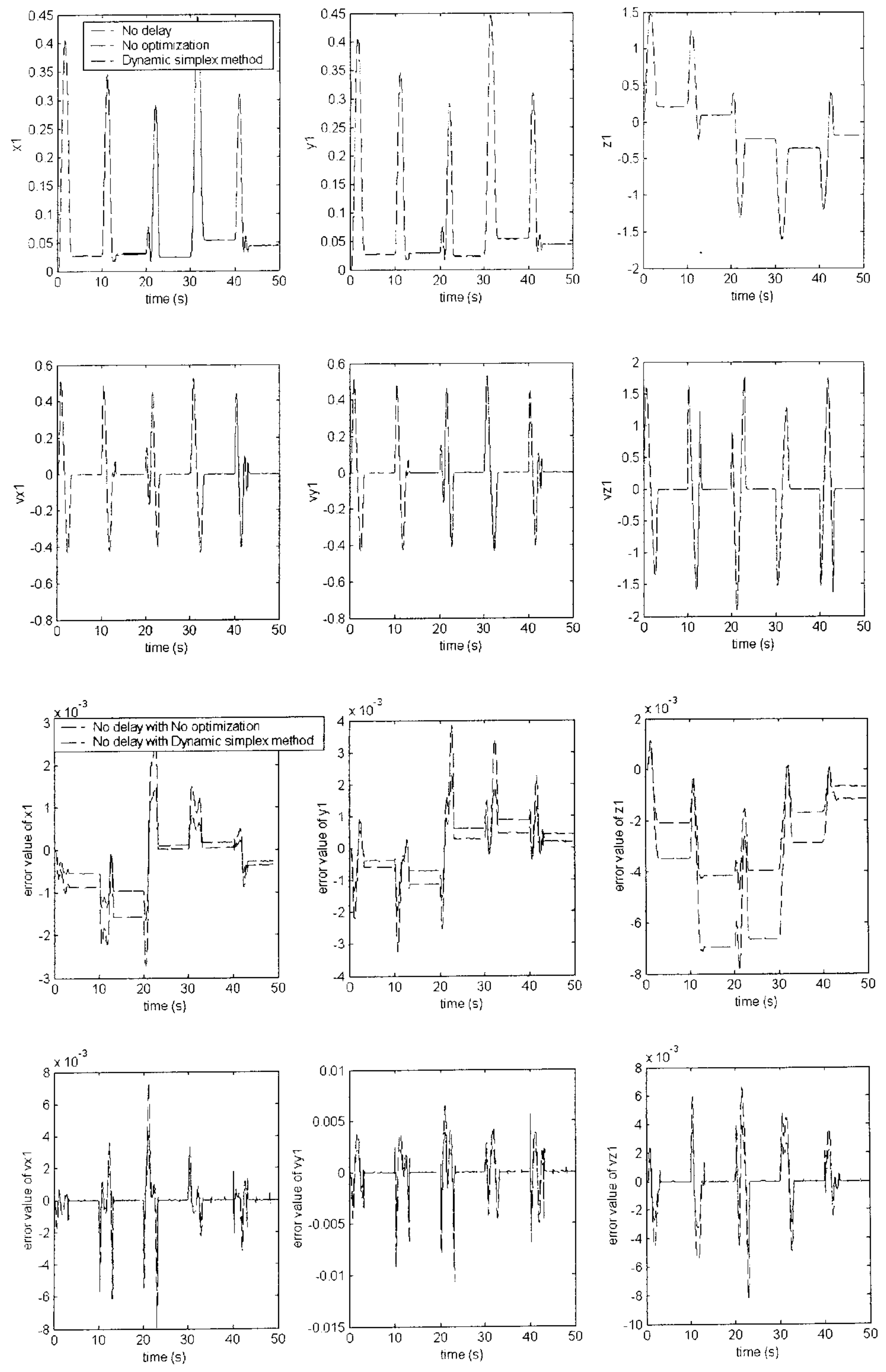


Fig. 3-14 Coordinates of body 1 versus time

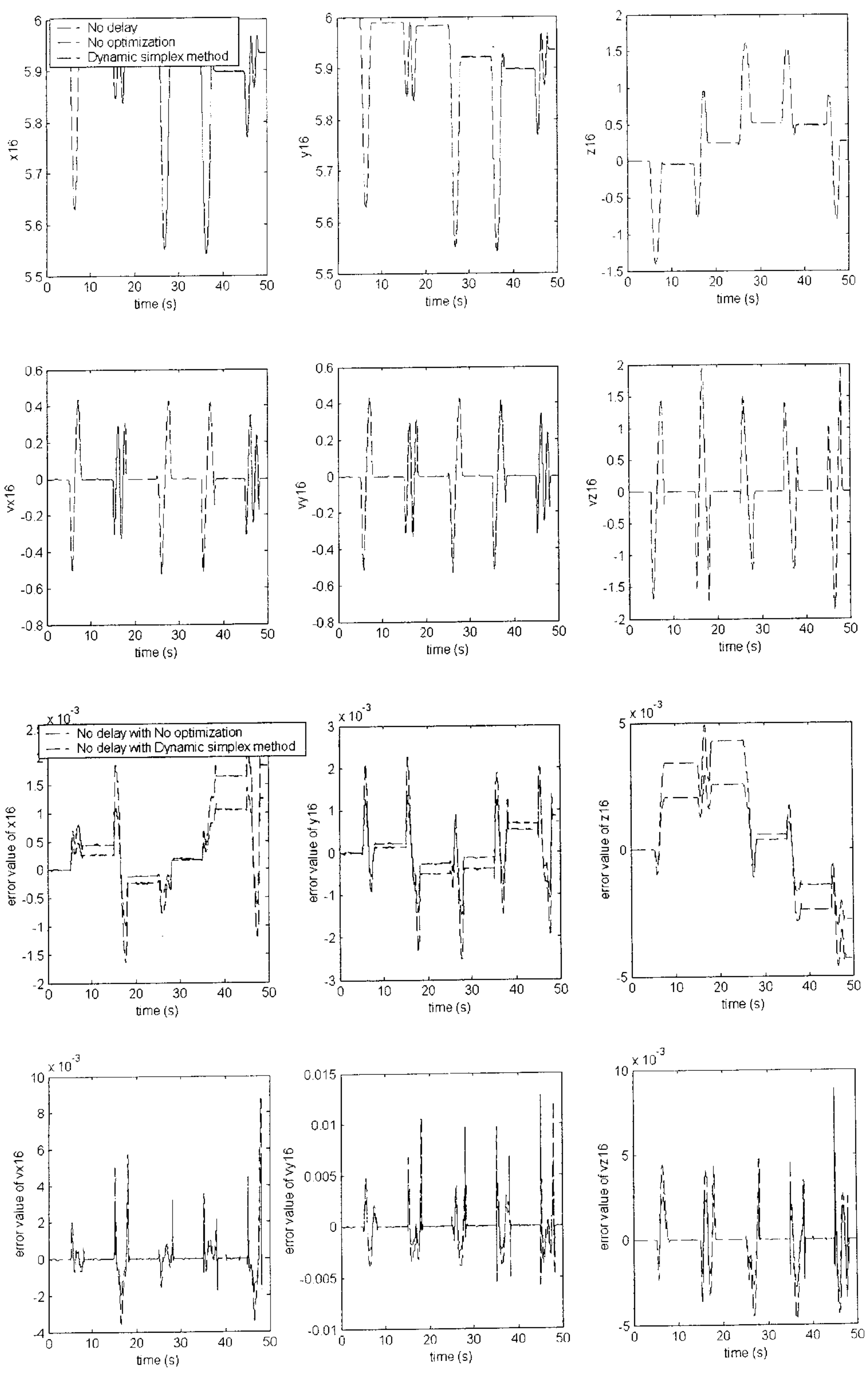
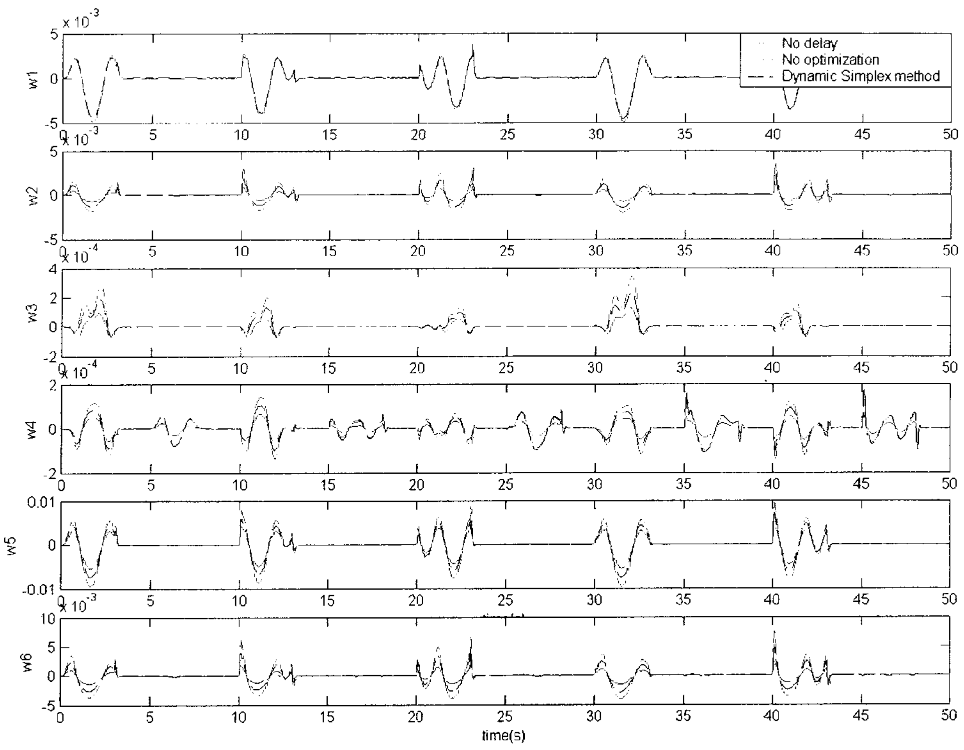


Fig. 3-15 Coordinates of body 16 versus time



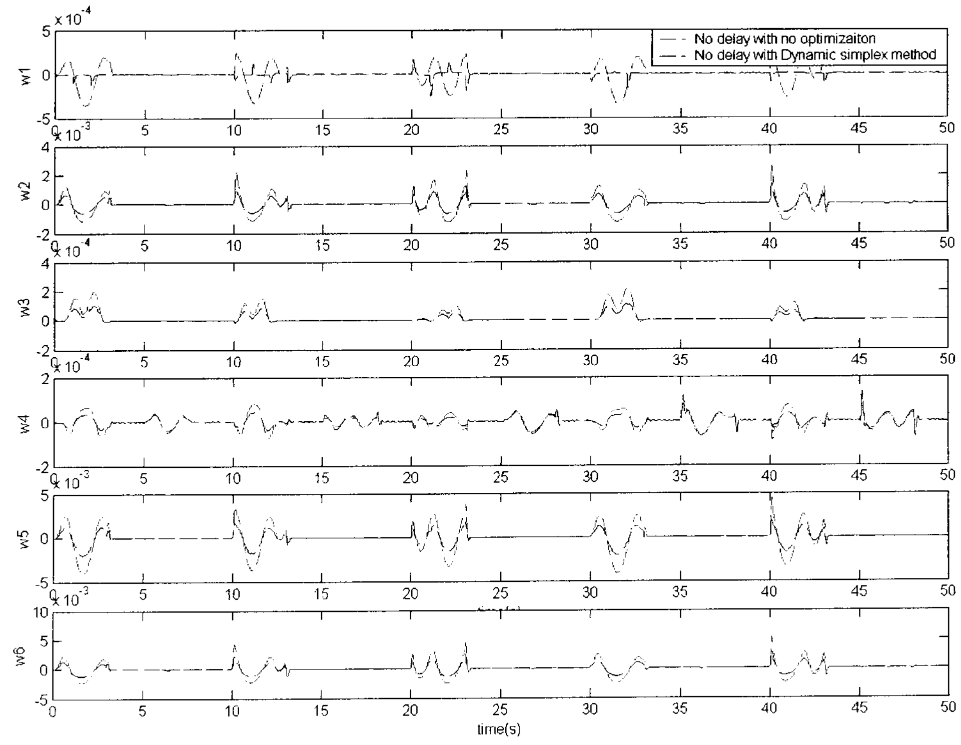


Fig. 3-16 Position constraint values versus time in node 1

Fig. 3-14 and Fig. 3-15 show position and velocity coordinates of body 1 and 16, and it can be seen that the simulation errors using the dynamic simplex method is less than those of no optimization. Therefore, as the position constraint functions are minimized , simulation errors (position, velocity) are also minimized simultaneously. The position constraint functions are optimized continuously as shown in Fig. 3-16.

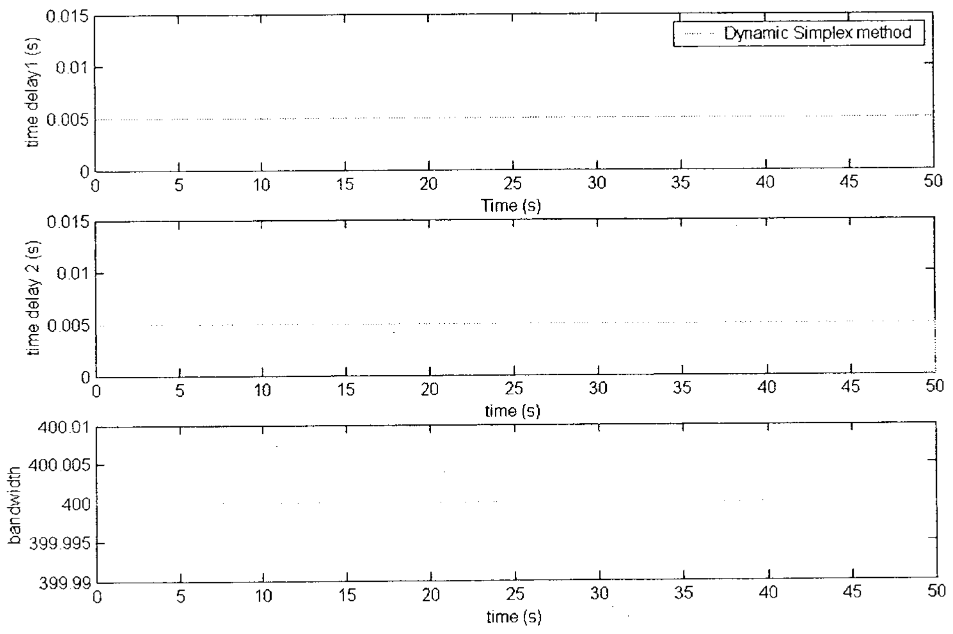


Fig. 3-17 Time delay 1, 2, and bandwidth as T = 10s

From the results shown in Fig. 3-17, we can see if T is chosen too large, the DSM algorithm cannot catch the optimum and follow its movement continuously.

3.4.2.1.2 Unconstrained optimization problem

Bandwidth is not constant, and it is less than 400 HZ. This is a two-variable problem without constraint that controls τ_1 and τ_2 simultaneously. Set $b = 5.0e-4$.

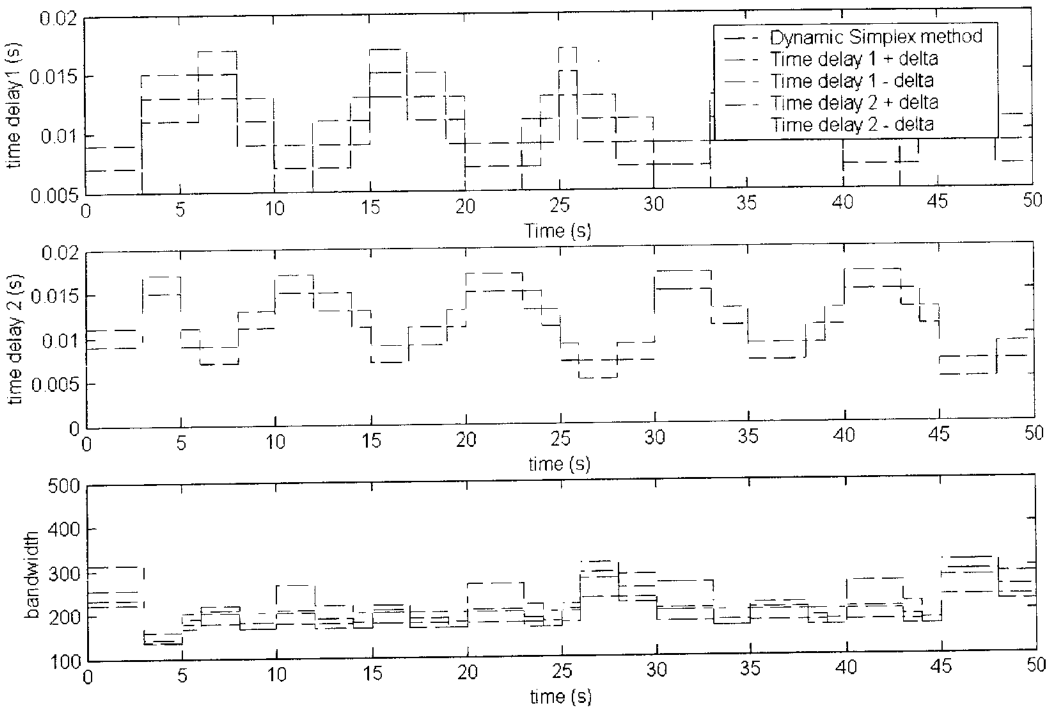


Fig. 3-18 Time delay 1, 2, and bandwidth versus time

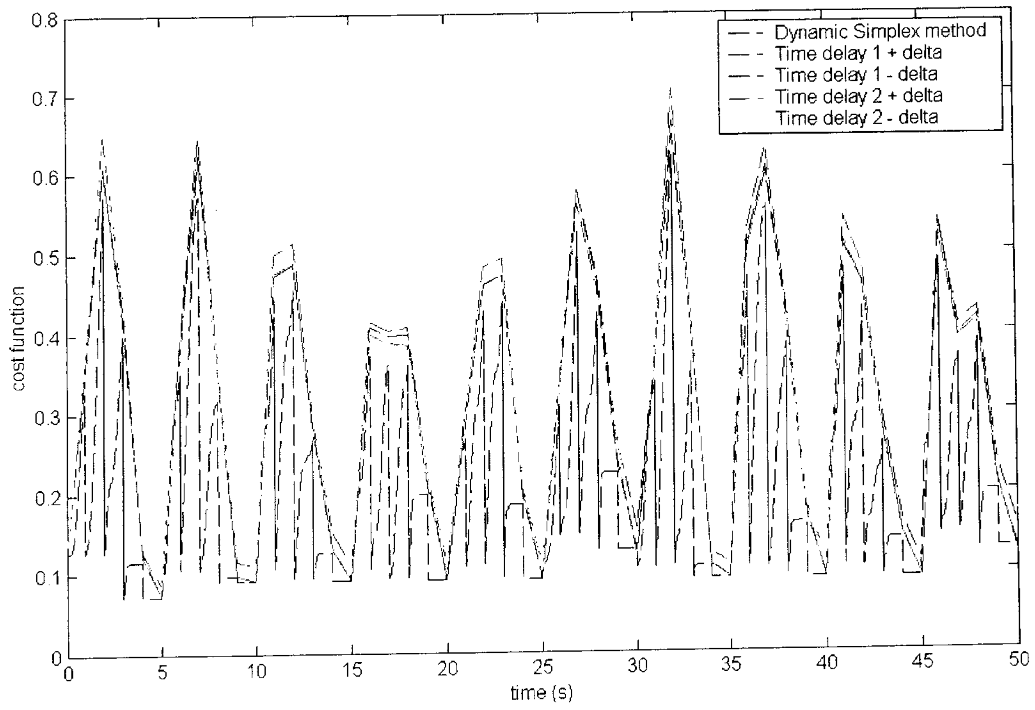


Fig. 3-19 Cost function value versus time

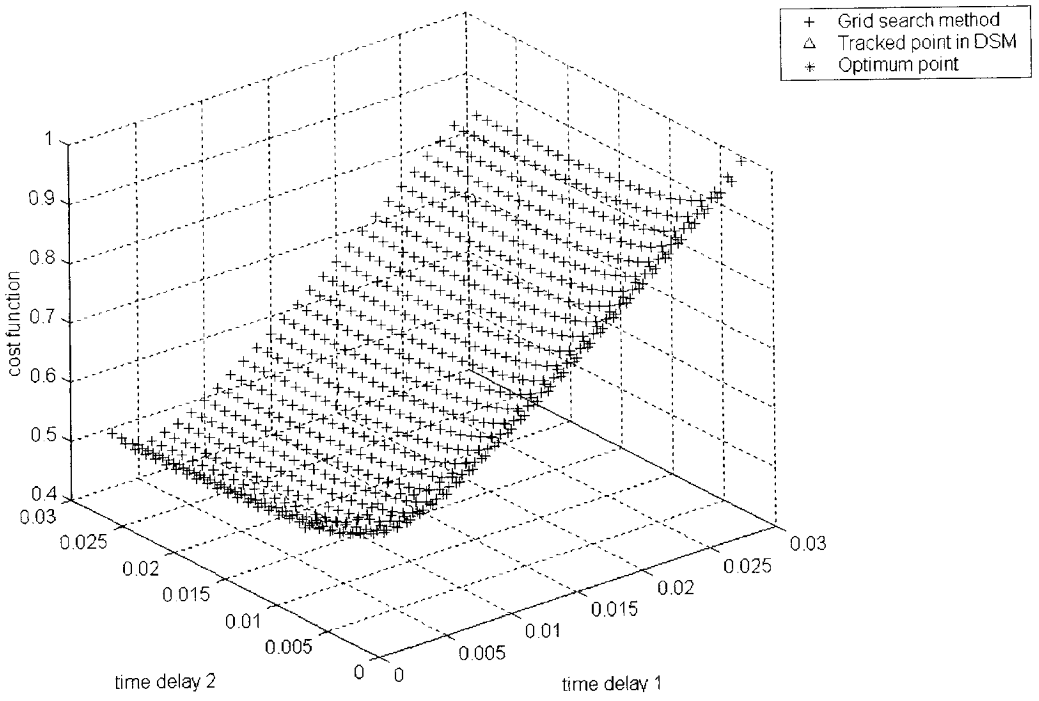


Fig. 3-20 Grid search method testing from 10s to 11s

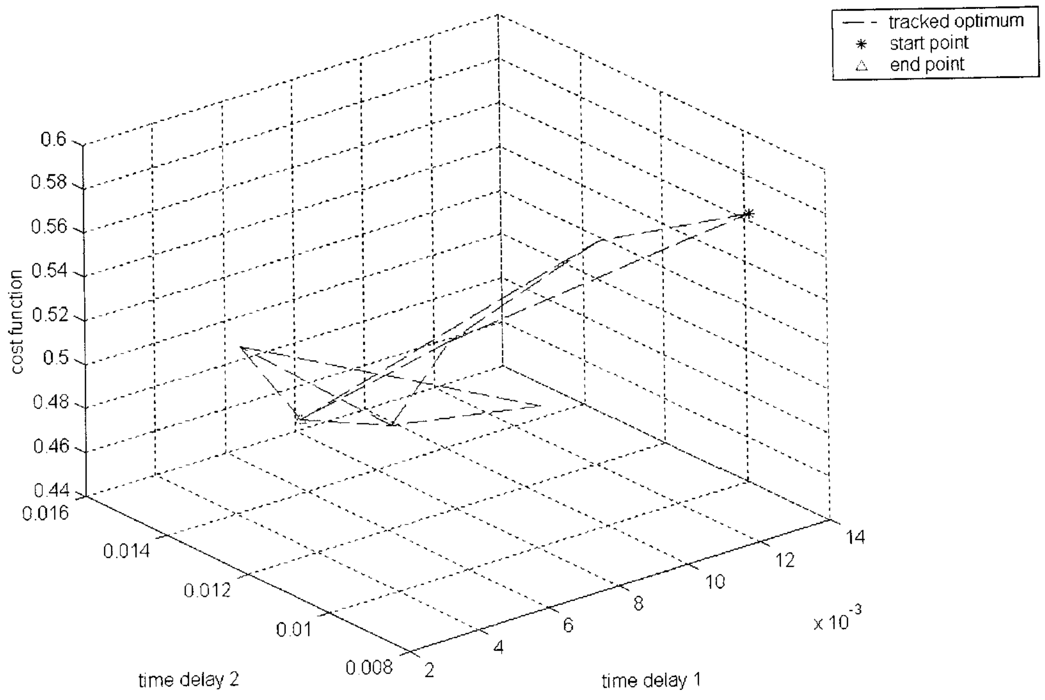


Fig. 3-21 Reflection operations from 10s to 11s

Four kinds of time-delay settings, which are $\{\tau_1 \pm \text{delta}, \tau_2\}$ and $\{\tau_1, \tau_2 \pm \text{delta}\}$, are used to make comparison with DSM, where, $\text{delta} = 0.002\text{s}$. Fig. 3-19 shows the DSM algorithm can catch the optimum compared with other time-delay settings. Fig. 3-21 shows the reflection operations (10s to 11s) from start point to end point, and the end point is equal to the optimum point. It can be seen from Fig. 3-20 that the optimum point from 10s to 11s is same as the tracked point in Fig. 3-18 and Fig. 3-19.

3.4.2.2 Time distribution of time delays

Time delay 1 is acting on the variables $x(t-\tau_1)$ and $\dot{x}(t-\tau_1)$, and time delay 2 is acting on the variable $F_{con}(t-\tau_2)$. Also, two types CP and UP are examined.

3.4.2.2.1 Constrained optimization problem

Compared with DSM, DSM + delta is the addition of the reflection to DSM; DSM - delta is the subtraction of the reflection from DSM. Compared with the result of DSM ± delta, the result of DSM is more optimal as shown in Fig. 3-23. Also we can see that time delay 2 is usually smaller than time delay 1. Thus, a smaller time delay of constraint forces can decrease the value of the cost function. The optimum point (0.007, 0.004) from 10s to 11s shown in Fig. 3-22 and Fig. 3-23 is verified by the grid search method shown in Fig. 3-24. Fig. 3-25 shows the reflection operations (10s to 11s) from the start point to the end point, and that the tracked point is same as the optimum point.

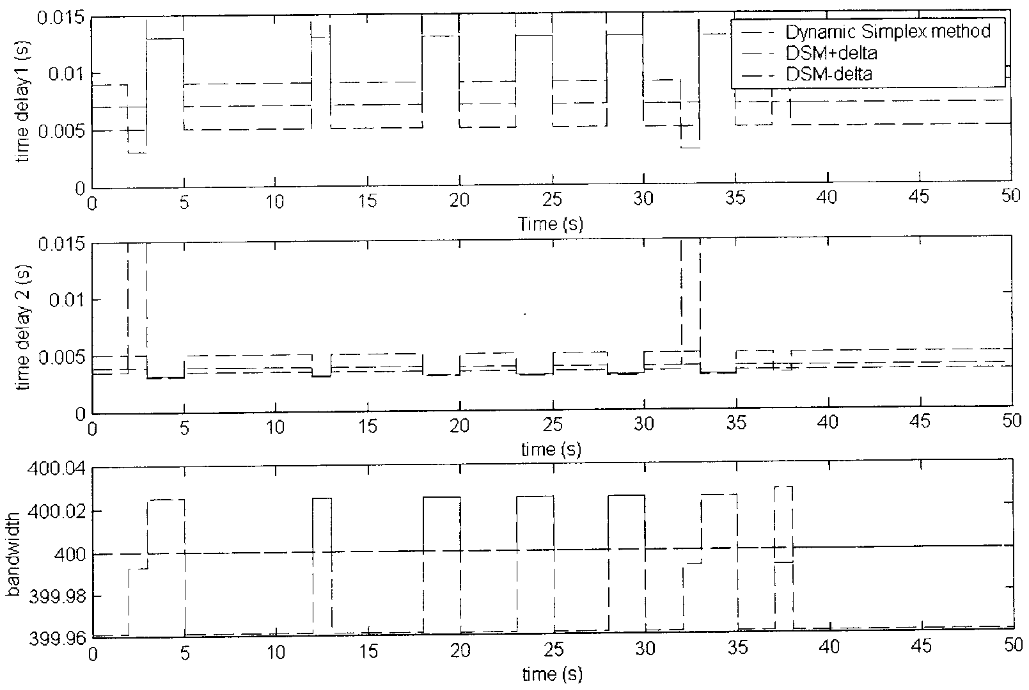


Fig. 3-22 Time delay 1, 2, and bandwidth versus time

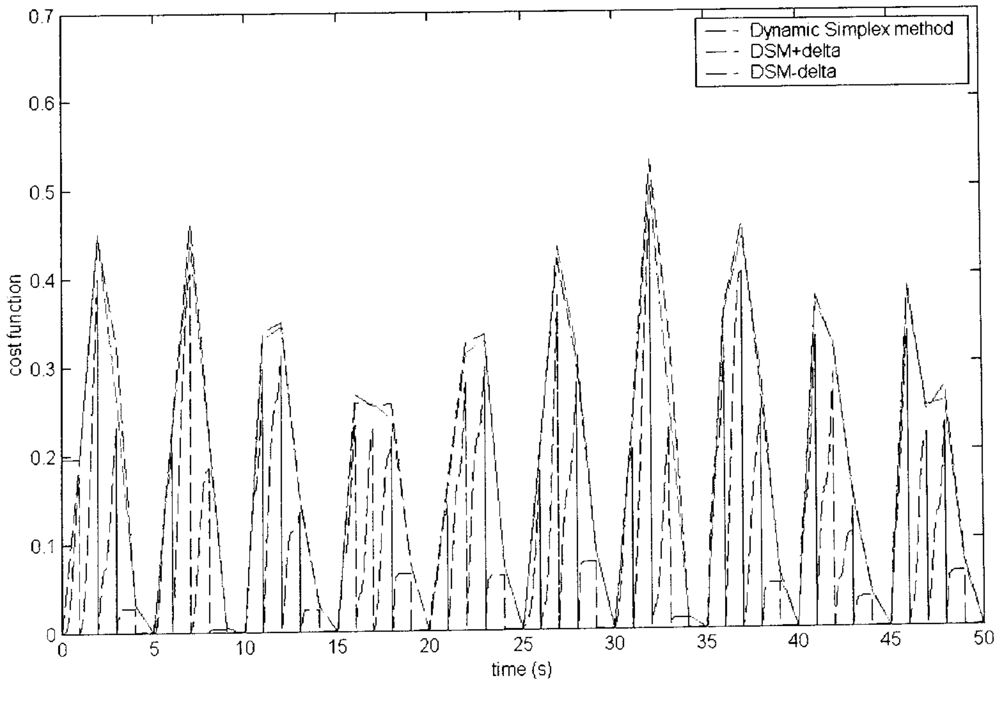


Fig. 3-23 Cost function value versus time

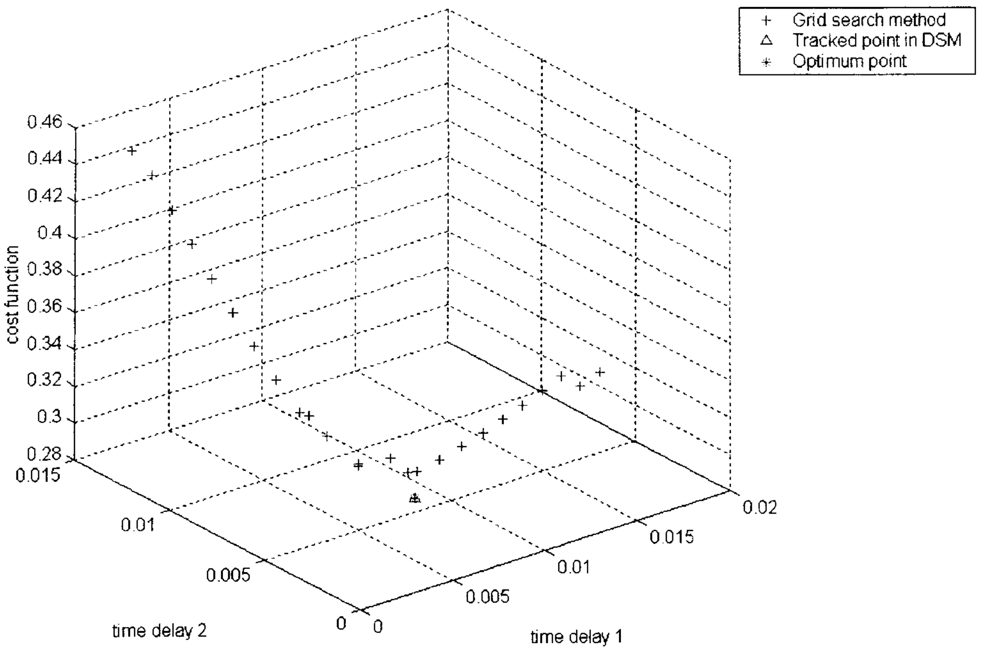


Fig. 3-24 Grid search method testing from 10s to 11s

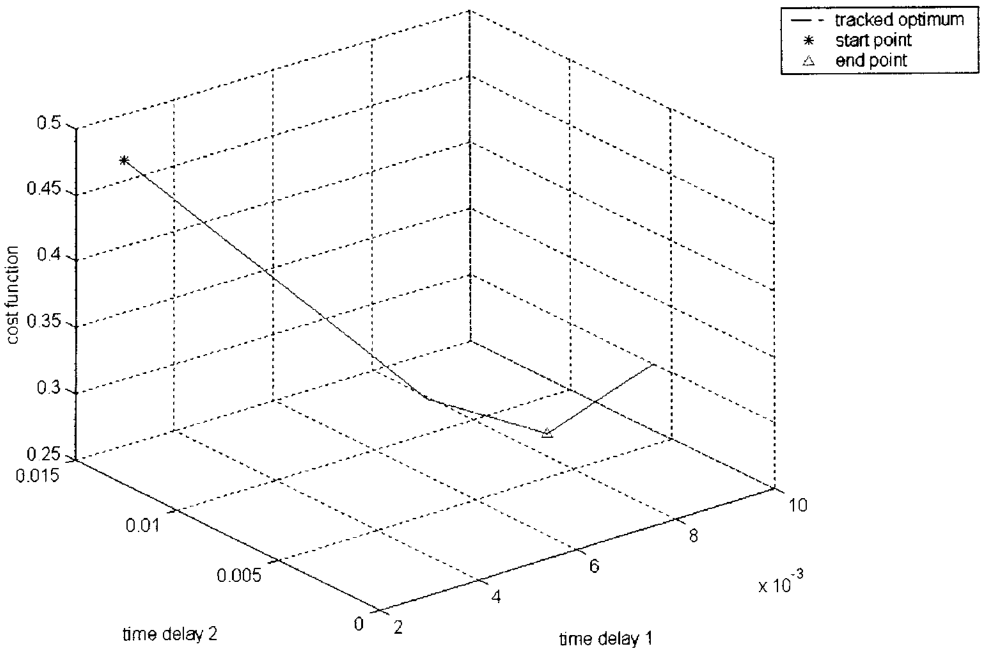


Fig. 3-25 Reflection operations from 10s to 11s

3.4.2.2.2 Unconstrained optimization problem

The bandwidth is not constant for this case, and it is less than 400 HZ. This is a two-variable problem without a constraint so that control τ_1 and τ_2 can be optimized simultaneously ($b = 5.0e-4$).

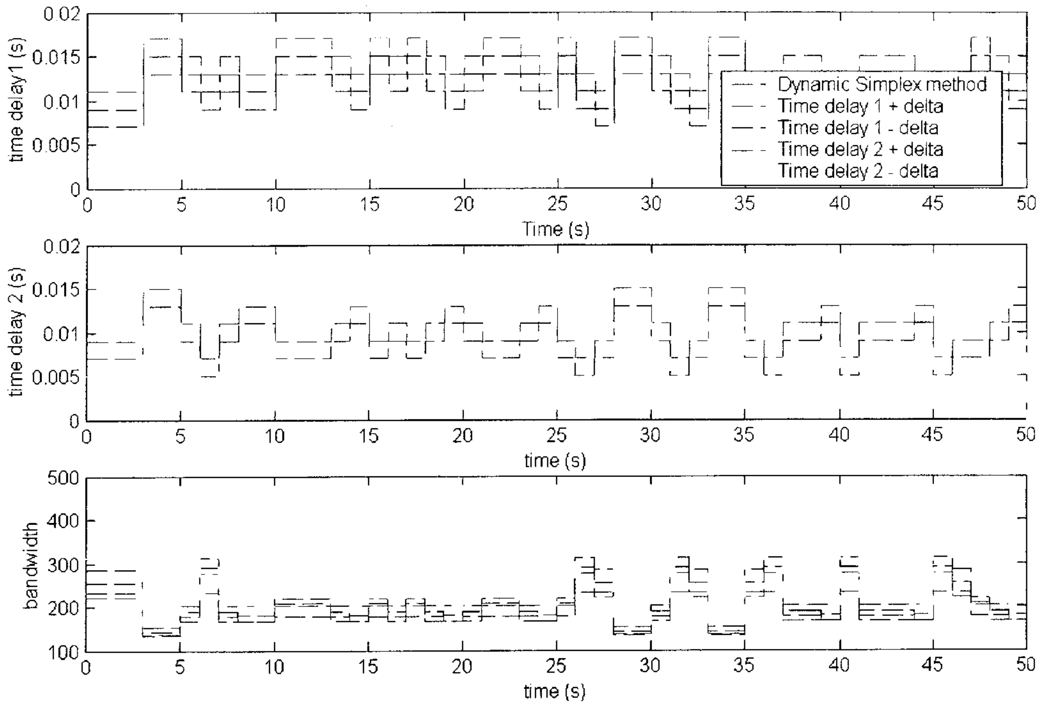


Fig. 3-26 Time delay 1, 2, and bandwidth versus time

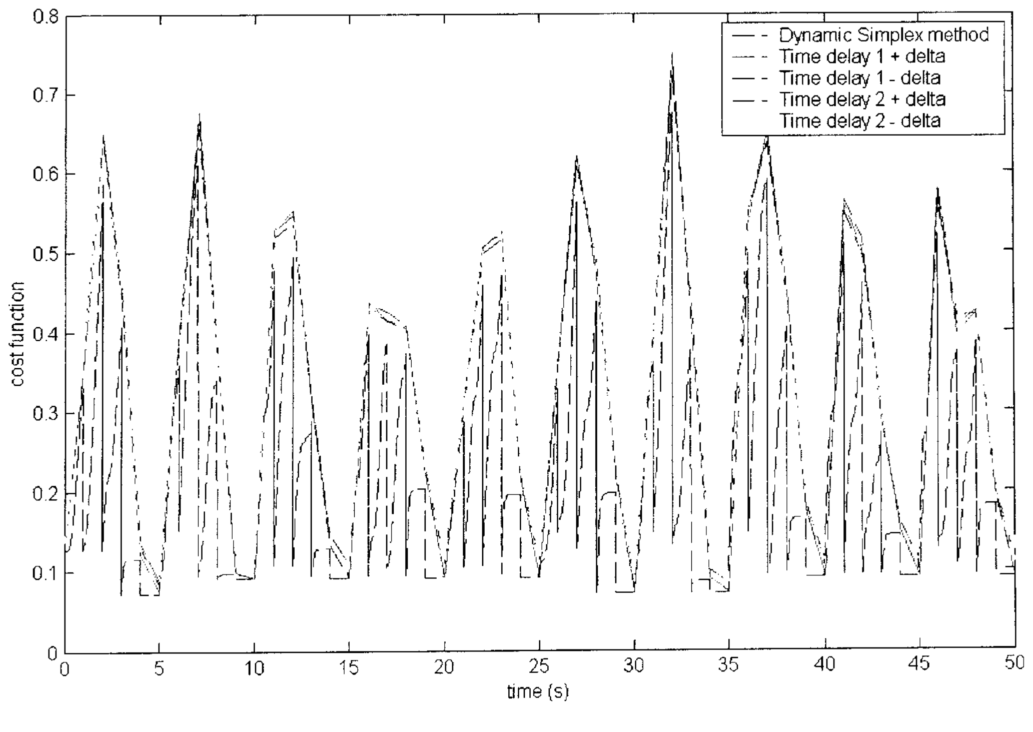


Fig. 3-27 Cost function value versus time

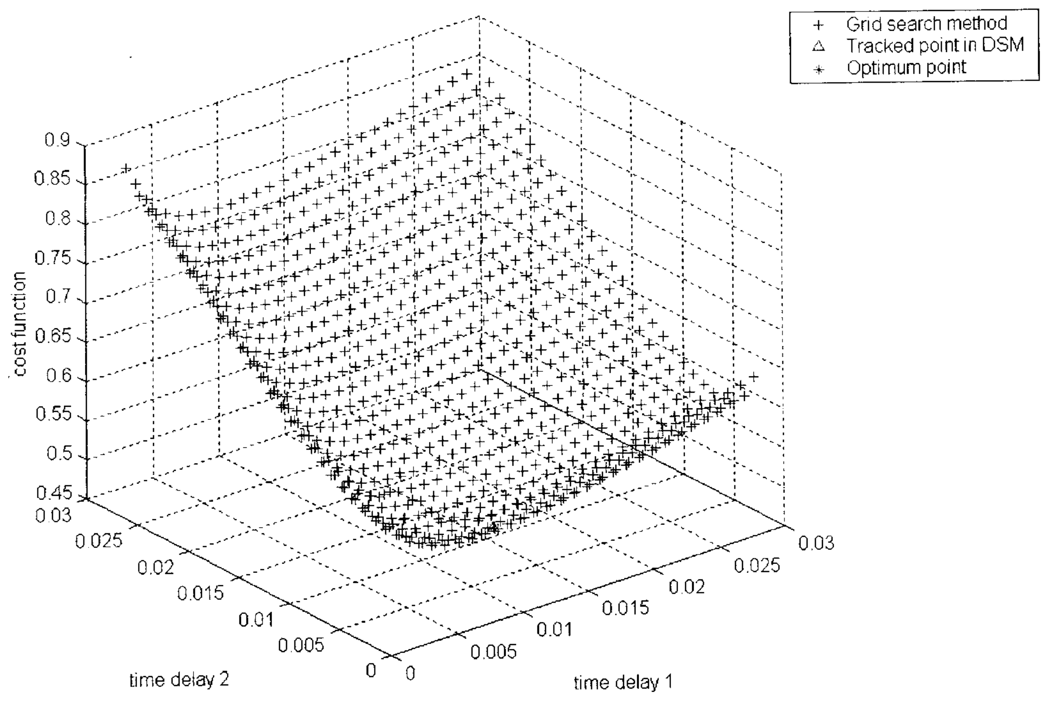


Fig. 3-28 Grid search method testing from 10s to 11s

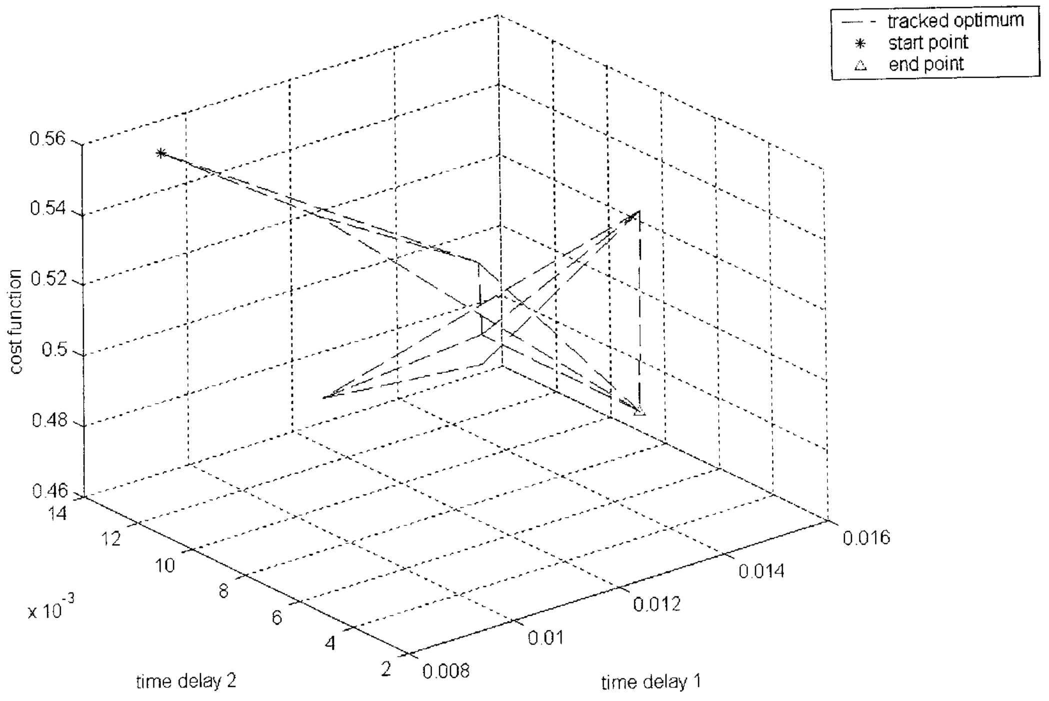


Fig. 3-29 Reflection operations from 10s to 11s

Where, delta is equal to 0.002s. Fig. 3-27 shows that the cost function value of DSM is less than other time-delay settings, and time delay 2 is always less than time delay 1 as shown in Fig. 3-26. It can be see from Fig. 3-28 that the optimum point from 10s to 11s is same as the tracked point in Fig. 3-26 and Fig. 3-29.

3.4.2.3 Conclusions

Therefore, the main conclusions of this section are summarized as follows:

- The DSM algorithm can catch the optimum and follow its movement continuously in space/time distribution of time delays. Considering the complexity of multi-body systems, a few percent marginal improvement can greatly enhance the stability of systems.
- If T is chosen too large, the DSM algorithm cannot catch the optimum and follow its movement continuously.
- The result clearly shows that the cost function value is nearly zero as external forces acting are zero.
- As the position constraints are minimized, the simulation errors (position, velocity) are minimized at the same time.
- Space distribution: As the neighbouring force is bigger than the farther force, the nearby time delay is smaller as compared with the later. That is, compared with smaller force's acting area the larger force's acting area needs faster response

(small time delay). It also means one can set the measurement sampling time T as large as possible until the external forces have been changed to other areas.

- Time distribution: Compared with other variables, higher frequency of constraint forces (smaller time delay) usually get more accurate results during simulation and decrease the value of cost function.

3.5 Assumptions and limitations

Although the DSM algorithm can greatly improve the performance of real-time distributed multi-body systems, several key assumptions and limitations of that should be considered. They are stated as follows.

- Optimal point cannot move too quickly
- The calculation speed of controller is faster enough to catch the moving optimum
- The network system has the ability of adjusting the time delays of the distributed system

4 TDMA Networked Simulation Examples

The primary goal of distributed simulation is to obtain a higher performance by parallel execution of multiprocessor or networked workstations. Real-time distributed simulation can be implemented based on several operating systems or software tools, including LynuxWorks [40], Vxworks [42], Tricomtek [41], QNX Software Systems [43], TDMA protocol [37][44], and so on. TDMA (time division multiplexed access) protocol is a fair, simple, and deterministic protocol and is widely used in wireless communication field. Lu [27] has utilized TDMA as a higher level protocol based on a real-time NIC driver to achieve an innovatory real-time distributed system based on Ethernet [35][36], and it has several advantages: "reducing layer of communication, controlling the timing precisely, and making Ethernet a deterministic network". Several companies are developing real-time distributed simulation products, such as Opal-RT Technologies [45], MPI/RT [46], Myricom Inc., [47], and so on. However, none of their implementations are based on TDMA protocol, VenturCom RTX environment, and Ethernet. In this thesis, we use TDMA network to realize dynamic optimization for real-time distributed multi-body simulations because it allows us to change the time delays on line. Compared with other protocols, TDMA gives us a more direct way of controlling the communication delays through the communication schedule.

4.1 Overview of TDMA networked communication

This section is based on Lu's thesis of "Design of Ethernet Based Real-time Distributed Systems" [27], in which a digital transmission technology TDMA has been used to allow several computer nodes to access a single channel without interference, that is, as one node on the network is sending out messages during its own unique time slot, there are no other nodes are sending out messages. Lu has verified that the clocks on different computers have drifts. Because clocks determine time slots, in order to avoid the interference on the time slot, a signal sent from master computer is used to synchronize the clocks of slave nodes to make one global clock, see Fig. 4-2. After the slave nodes receive the signal, the simulation begins.

Fig. 4-1 illustrates the structure and information flow of TDMA network between two nodes over the network. The connection solution of nodes is shown in Fig. 4-3. Hub/Switch 1 is used to connect gigabit Ethernet cards of nodes, which is mainly used in data communication with TDMA; while Hub/Switch 2 is used to connect fast Ethernet cards of nodes, which is used to control these computers by users, and Symantec pcAnywhere v10.5 is the control tool.

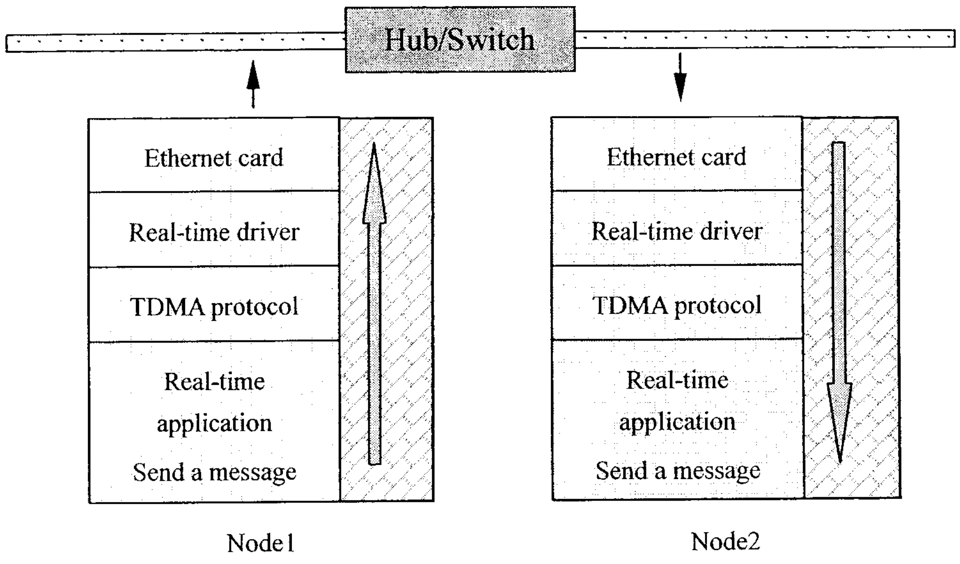


Fig. 4-1 Structure and data flow of TDMA network

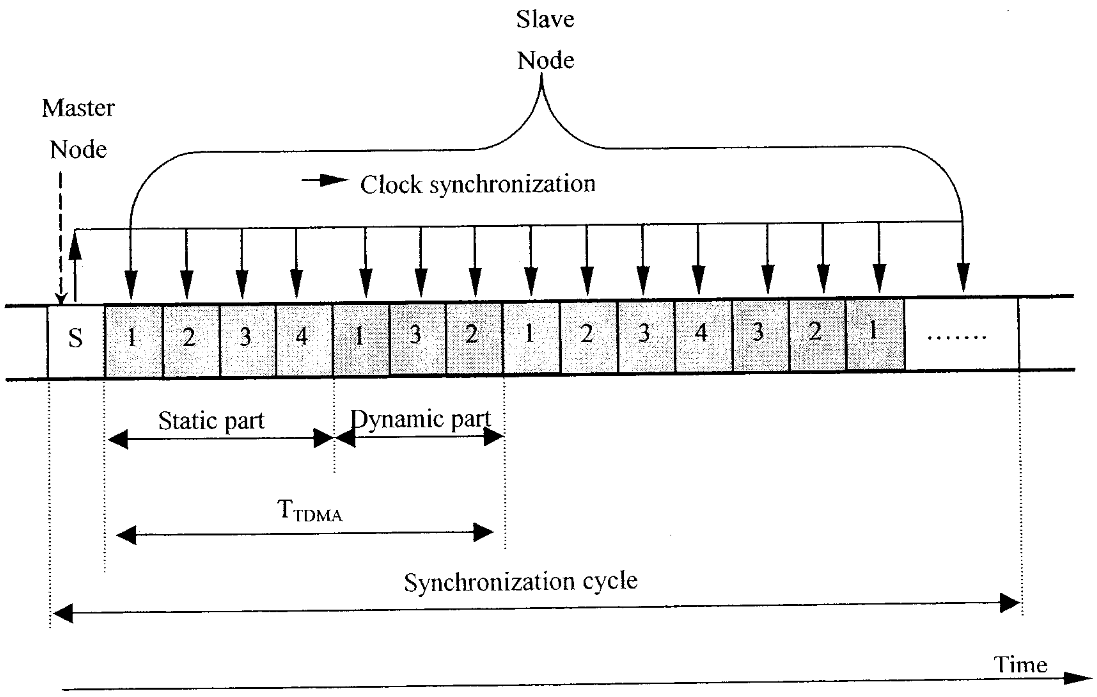


Fig. 4-2 Dynamic TDMA Ethernet

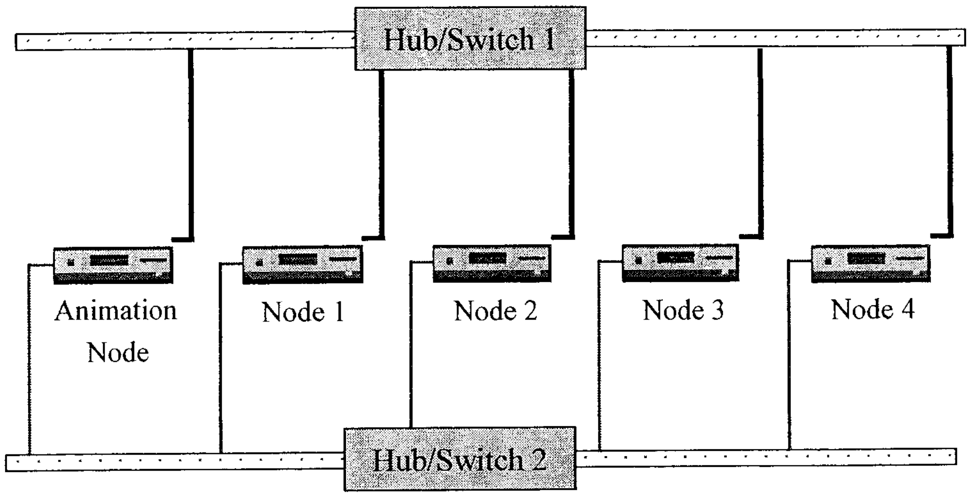


Fig. 4-3 Computer connection of TDMA network

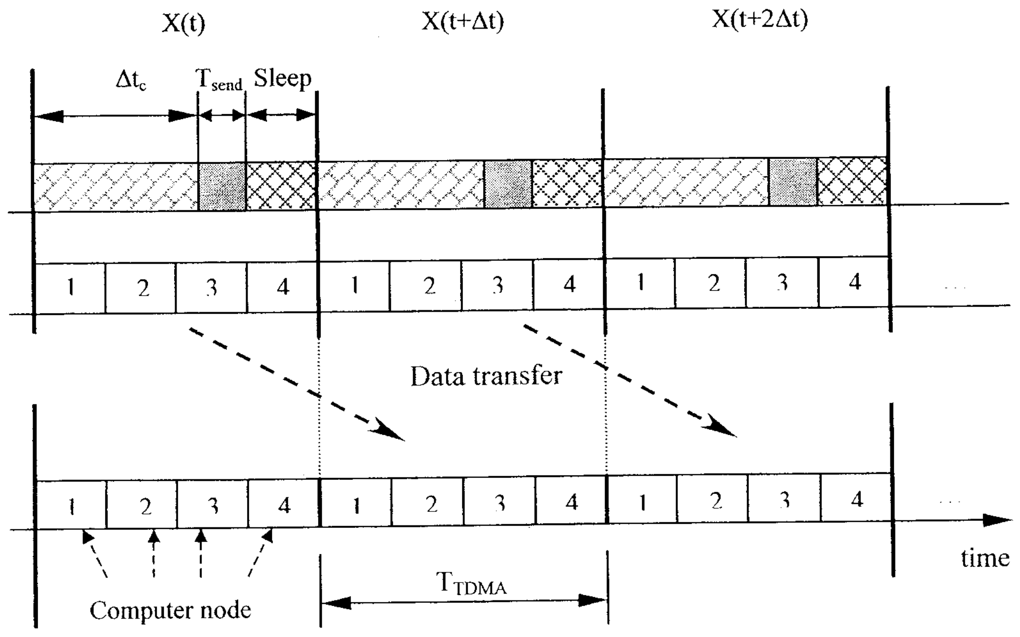


Fig. 4-4 Communication and computational diagram

Fig. 4-4 shows that the communication and computational diagram of distributed TDMA networked simulation, and every node receives data from last time step at each time step. If the computational time of one node is different from that of another node, then it is needed to set different waiting (sleep) time to synchronize them.

TDMA protocol can be classified as static and flexible TDMA protocols. Static TDMA protocol is shown in Fig. 4-4, in which size of time slot, size of synchronization slot, and sending sequence are unchanged as the programs run; compared with the static TDMA protocol, flexible TDMA protocol shown in Fig. 4-2 has higher efficiency and flexibility to meet the requirements of distributed real-time simulations, and can change them dynamically. A TDMA cycle in flexible TDMA protocol can be divided into a static part and a dynamic part. Every node in static part transfers the most urgent information, while that in dynamic part follows the slot list to send packets. Server can generate a new slot list with the guarantee of a list conformation mechanism to avoid collision, that is, only after all the nodes confirm the new list from server, the new list can be launched.

4.2 Implementation of distributed simulations on a TDMA network

Compared with static TDMA protocol, Flexible TDMA protocol has higher efficiency and flexibility to meet the requirements of distributed real-time simulations. In this section, we choose a flexible sending list with a fixed time slot size and a big fixed synchronization slot size to implement distributed simulation of multi-body systems.

Node 0, which maintains the TDMA slot list and sends synchronization signals, works as a master node (server) in the network. Fig. 4-5 describes the simulation process of distributed simulation of multi-body systems based on the dynamic TDMA Ethernet. The DSM algorithm is incorporated into the program of the server, which can obtain the optimal time delays and send corresponding list to the clients. Therefore, through adjusting the frequency of list shown in Fig. 4-2, the corresponding time delays of each node are set. The new paper of Lu has shown how to schedule the list to get the corresponding time delays of each node.

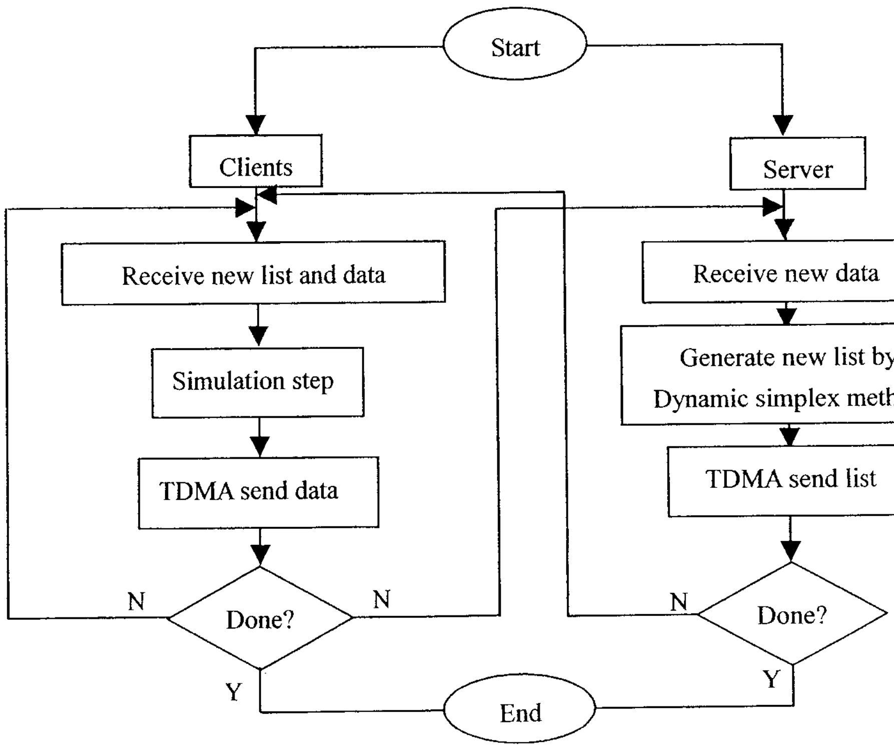


Fig. 4-5 Simulation process

As the server receives all the data transferred from clients, it generates a new list by a simulation program using the DSM algorithm. The simulation program in the server is a simulation program of the whole distributed system. In order to catch up with the simulation process of the whole distributed system, Ts (the measurement period in the server) should be chosen as small as possible compared with T. Because the server needs time to compute the new generated list and send it, there is a time lag $(Ts * m_r)$ in the distributed system.

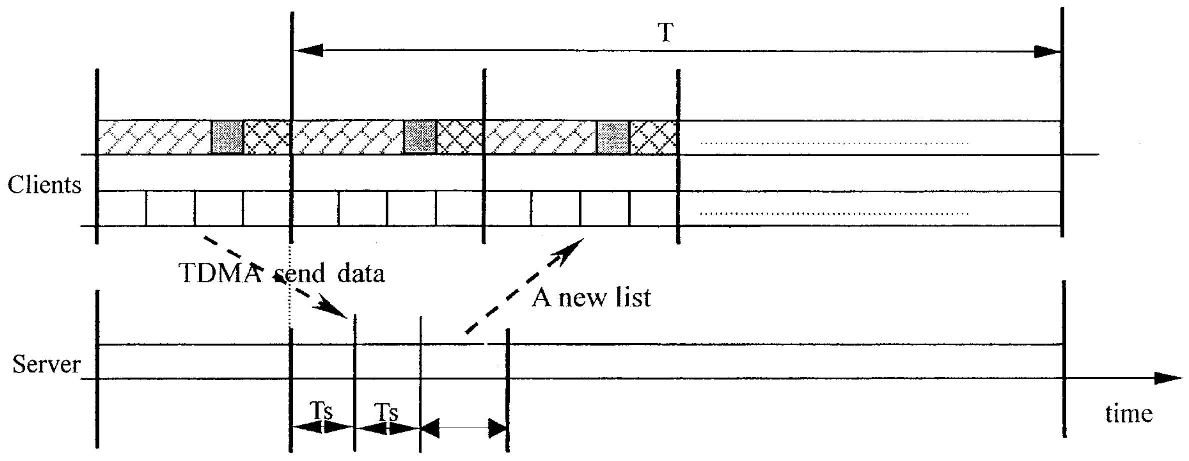


Fig. 4-6 Communication and computational diagram of server and clients

4.3 Experimental verification

The example shown in Fig. 3-4 has been examined, and we get similar results as the above sections. Moreover, the example shown Fig. 4-7 and Fig. 4-8 has been examined in a sixteen-node cluster. This is a constraint problem (CP) in space distribution of time delays, and similar parameters are set as section 3.4.2.1.1. And the simulation results shown in Fig. 4-9 and Fig. 4-10 have proven that the DSM algorithm can be applied to dynamic TDMA Ethernet.

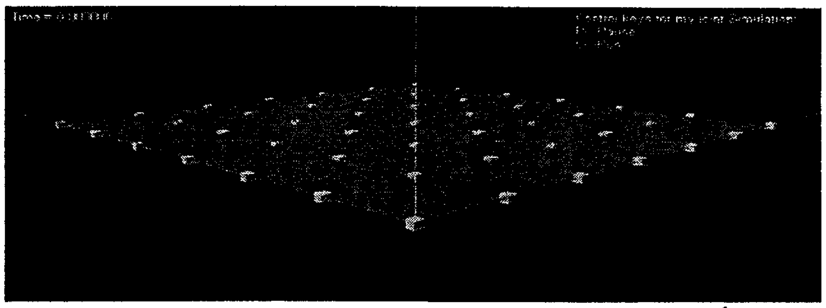


Fig. 4-7 Snapshot of simulation

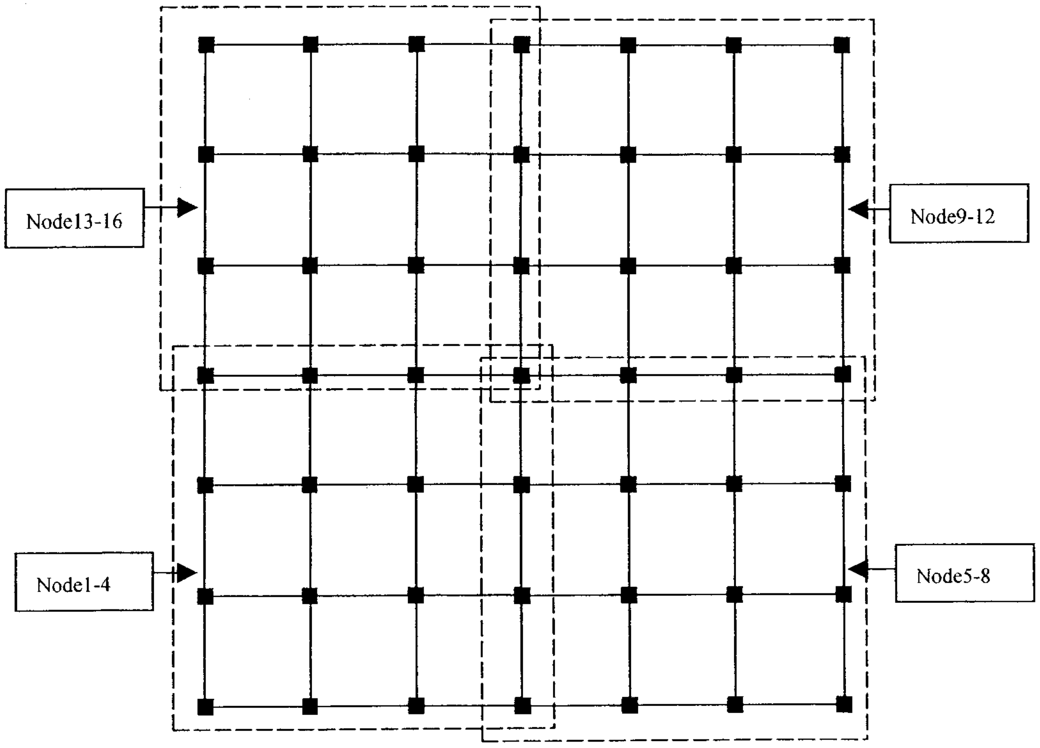


Fig. 4-8 Sixteen nodes

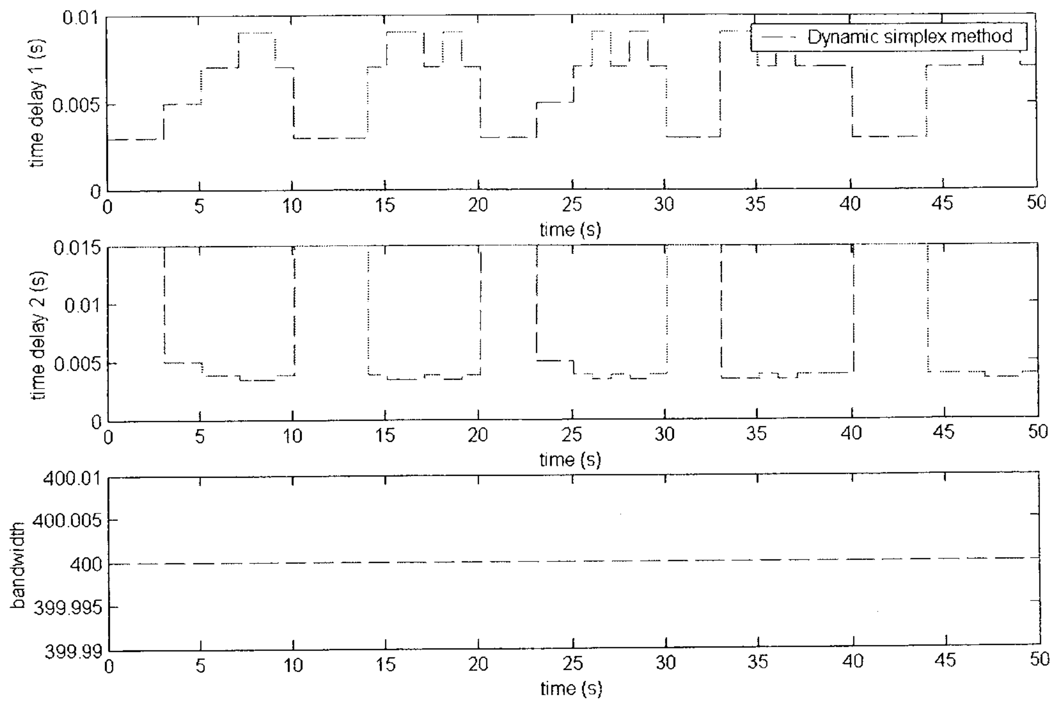


Fig. 4-9 Time delay 1, 2, and bandwidth versus time

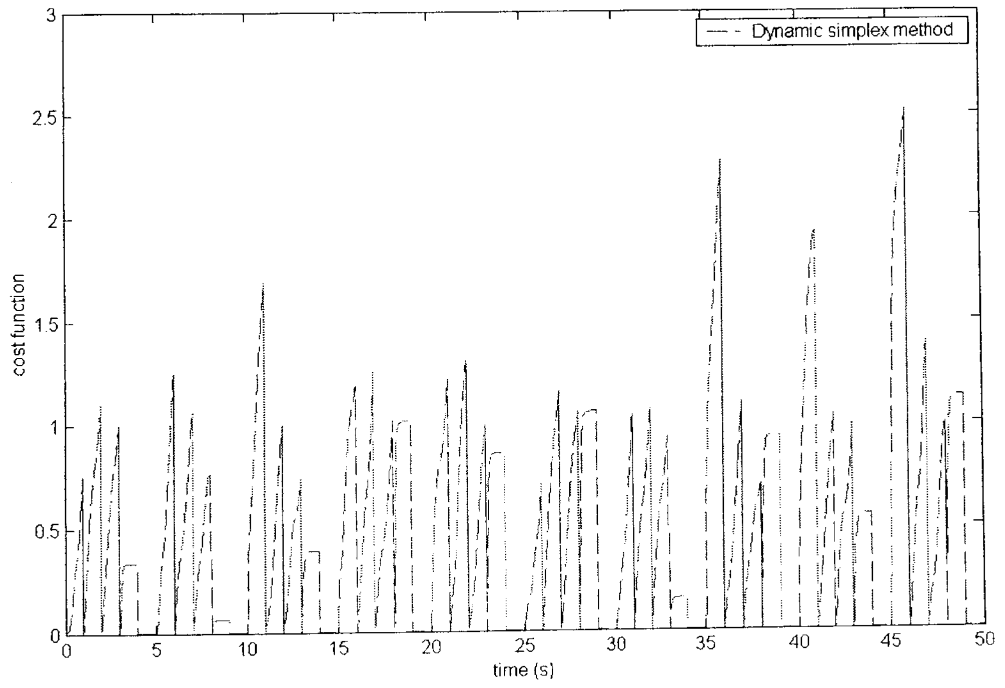


Fig. 4-10 Cost function value versus time

5 Conclusions and Future Work

5.1 Conclusions

In this thesis, a new approach for dynamic optimization of real-time distributed multi-body simulations is investigated. The main conclusions are summarized as follows:

1. A new cost function is defined and combined with the dynamic simplex optimization algorithm. The optimization problem is to minimize the network performance cost function, which is defined as the sum of the root mean squared (RMS) value of w_i over the measurement period T, and the effect of time delays in real-time distributed multi-body simulations is considered. The new cost function is combined with a dynamic simplex optimization method based on DSM, which is a very simple iterative minimization method and requires only several function evaluations (measurements), no derivatives, and it always converges and is fairly stable.

2. A new approach is developed for continuously optimizing network communication for a given model partitioning. We presented the method for cutting the multi-body system for a given gridding example shown before, which can be simulated in the distributed network. Moreover, based on the partitioning method, a controller designed obtains data from the distributed systems and continuously adjusts time delays of the distributed network, that is, optimizes

network communication.

3. Experimental support is provided that indicates the approach converges close to the optimal solution provided it doesn't vary too rapidly. Two different time-delay distributions, which are space distribution and time distribution, have been described. It has been verified that the approach can catch the optimum, follow its movement continuously, enhance the stability of systems, and minimize simulation errors.

4. A realistic experimental investigation is performed on a TDMA computational network. Compared with other protocols, TDMA allows us to change the time delays on line through the communication schedule. Moreover, we have implemented dynamic optimization for real-time distributed multi-body systems based on the TDMA protocol, and experimental results are satisfactory.

5.2 Future work

There are several directions that can be pursued. The most significant directions are:

1. Dynamic model partitioning optimization: In this thesis, we used a given partitioning method and a specific example. However, in actual practice, dynamic model partitioning optimization can greatly improve the performance of the distributed system.

2. Theoretical stability and performance analysis: Because DSM has been developed

in recent years, it has no theoretical basis. In the future, theoretical stability and performance analysis will be analyzed.

3. Improved optimization methods: More comparisons with other methods need to be considered, such as SQP and DRSM. Moreover, several extensions of DSM need to be studied, such as disturbance rejection and constraint handling.

4. Generalization to mechanical systems: We plan to apply the real-time distributed simulation approach of multi-body systems to more sophisticated mechanical systems.

References

[1] Parviz, E.N., *Computer-Aided Analysis of Mechanical Systems*, 1988 (Prentice-Hall, Inc).

[2] Baraff, D., "Linear-Time Dynamics using Lagrange Multipliers", *Computer Graphics Proceedings,* Annual Conference Series, SIGGRAPH 96, New Orleans, 1996.

[3] Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling W.T. *Numerical Recipes in C*, 1998 (Cambridge University Press).

[4] Slotine, J. –J.E. and Li, W., *Applied Nonlinear Control*, 1991 (Prentice Hall).

[5] Rum, F. and Gordon, B.W., "Simulation of Deformable Objects using Sliding Mode Control with Application to Cloth Animation", *International Conference on Computational Science 2004*, pp. 292-299, 2004.

[6] Gordon, B.W., "State Space Modelling of Differential-Algebraic Systems using Singularly Perturbed Sliding Manifolds", PhD Thesis, MIT, Mechanical Engineering Dept., August 1999.

[7] Slotine, J. -J.E., "Sliding Controller Design for Nonlinear Systems", *International Journal of Control*, Vol.40, No.2, pp. 421-434, 1984.

[8] Baumgarte, J., "Stabilization of constraints and integrals of motion in dynamical systems", *Computer methods in Applied Mechanical Engineering*, Vol. 1, pp. 1-16, 1972.

[9] Bayo, E. Garcia, J. J. and Serna, M.A., "A modified Lagrangian formulation for the dynamic analysis of constrained mechanical systems", *Computer methods in Applied Mechanical Engineering*, Vol. 71, pp. 183-195, 1988.

[10] Etch, E., "Convergence results for a coordinate projection method applied to mechanical systems with algebraic constraints", *SIAM Journal on Numerical Analysis*, Vol. 30, No.5, pp. 1467-1482, 1993.

[11] Gear, C.W. Leimkuhler B.J. and Gupta G.K., "Automatic integration of the Euler-Lagrange equations with constraints", *Computation and Applied Mathematics*, Vol. 12-13, pp. 77-90, 1985.

[12] Wehage, R.A. and Haug, E.J., "Generalized coordinate partitioning for dimensional reduction in analysis of constrained dynamical systems", *Mechanical Design*, Vol. 104, pp. 247-255, 1982.

[13] Haug, E.J., *Computer Aided Kinematics and Dynamics of Mechanical Systems, Vol. 1: basic methods*, 1989 (Allyn & Bacon, Inc).

[14] Shay, W.A. *Understanding data communications and networks*, 1999 (Pacific Grove, Calif).

[15] Baraff, D., http://www-2.cs.cmu.edu/~baraff/

[16] Rehbinder, H. and Sanfridson, M., "Scheduling of a limited communication channel for optimal control", *Proc. 39th IEEE Conf. On Decision and Control*, pp. 1011-1016, 2000.

[17] Hristu, D. and Morgansen, K., "Limited communication control", *Systems & Control Letters*, Volume 37, Issue 4, pp. 193-205, July 1999.

[18] Hristu, D., "Optimal control with limited communication", PhD thesis, The division of Engineering and Applied Science, Harvard University, Cambridge, USA, 1999.

[19] Nair, G.N. and Evans, R. J., "Stabilization with data-rate-limited feedback: tightest attainable bounds", *Systems and Control Letters*, Vol. 41, No. 1, , pp. 49-56, 2000.

[20] Nair, G.N. and Evans, R. J., "Communication-limited stabilization of linear systems", *Proc. 39th IEEE Conf. On Decision and Control*, pp. 1005-1010, 2000.

[21] Li, X. and Wong, W. S., "State estimation with communication constraints", *Systems and Control Letters*, Vol. 28, no. 1, pp. 49-54, 1996.

[22] Wong, W. S. and Brockett, R. W. "Systems with finite communication bandwidth constraints I: state estimation problems", *IEEE Trans. Automatic Control*, Vol. 42, No. 9, pp. 1294-1299, 1997.

[23] Box, G. E. P., "The exploration and exploitation of response surfaces: Some general considerations and examples", *Biometrics*, 10, pp. 16–60, 1954.

[24] Edwards, I. M., and Jutan, A., "Optimization and control using response surface methods", *Computers and Chemical Engineering*, pp. 441–453, 1997.

[25] Nelder, J. A., and Mead, R., "A simplex method for function minimization", *Computer Journal*, 7, pp. 308–313, 1965.

[26] Xiong, Q., and Jutan A. "Continuous optimization using a dynamic simplex method", *Chemical Engineering Science* 58, pp. 3817 – 3828, 2003.

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

[28] Brenan, K., Campbell, S. and Petzold L., *Numerical Solution of Initial Value Problems in Differential-Algebraic Equations,* 1989 (Amsterdam: North-Holland).

[29] Bamberger, W., and Isermann, R., "Adaptive on-line steady-state optimization of slow dynamic processes", *Automatica*, 14(2), pp. 223–230, 1978.

[30] Garcia, C. E., and Morari, M., "Optimal operation of integrated processing systems*", A.I.Ch.E.*, 27(6), 960, 1981.

[31] McFarlane, R. C., and Bacon, D. W., "Adaptive optimizing control of multivariable constrained chemical processes. 1. Theoretical development; 2. Application studies", *Industrial and Engineering Chemistry Research*, 28, pp. 1828–1834, 1989.

[32] Lee, K. S., and Lee, W. -K., "On-line optimizing control of a nonadiabatic 5xed bed reactor", *A.I.Ch.E*, 31(4), pp. 667–675, 1985.

[33] Hammer, J. W., and Richenberg, C. B., "On-line optimizing control of a packed-bed immobilized-cell reactor", *A.I.Ch.E.*, 34(4), pp. 626–632, 1988.

[34] Walters, F. H., Parker, L. R. Jr., Morgan, S. L., and Deming, S. N., " Sequential simplex optimization", 1991 (Boca Raton, FL: CRC Press).

[35] IEEE 802.3, IEEE Standard 802.3, 2000 Edition, 2000

[36] Held, G., *Ethernet Networks: Design, Implementation, Operation and Management*, John Wiley & Sons Canada, 2002.

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

[38] Torczon, V., "On the convergence of pattern search algorithms", *SIAM Journal on Optimization*, 7, pp. 1–25, 1997.

[39] Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E., "Convergence properties of the Nelder–Mead simplex method in low dimensions", *SIAM Journal of Optimization*, 9, pp. 112–147, 1998.

[40] LynuxWorks, http://www.lynuxworks.com/

[41] Tricomtek, http://www.tricomtek.com/

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

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

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

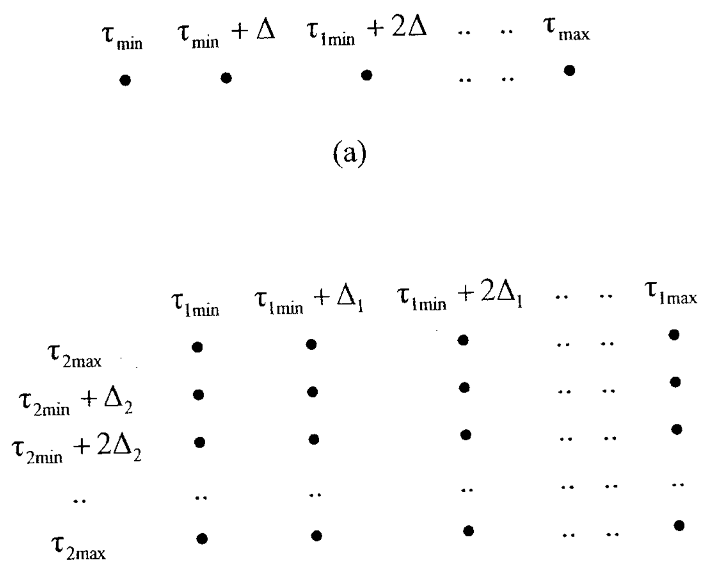
[45] Opal-RT Technologies, Inc: http://www.opal-rt.com/

[46] MPI/RT, http://www.mpirt.org/

[47] Myricom Inc., http://www.myri.com/

Appendix A Grid Search Method

In order to obtain the true optimum and make comparison with DSM, grid search method is introduced. The principle of grid search method is to try almost all possible values and combinations iteratively for the controlled variables at each measurement point, and then the corresponding cost function values of those are calculated and compared. Because we try to get the true optimum, the variables corresponding to the smallest cost function value is chosen.


τ_{min} $\tau_{min}+\Delta$ $\tau_{1min}+2\Delta$ .. .. τ_{max}
(a)
τ_{1min} $\tau_{1min}+\Delta_1$ $\tau_{1min}+2\Delta_1$ .. .. τ_{1max}
τ_{2max}
$\tau_{2min}+\Delta_2$
$\tau_{2min}+2\Delta_2$
..
τ_{2max}

(b)

Fig. A-1 Sketch of grid search method

Fig. A-1 (a) illustrates that as one variable (τ) is controlled, (τ_{min} - τ_{max}) different possible values are calculated and the corresponding cost function values are obtained at every estimate or measurement point, where, Δ is positive and very small. Fig. A-1 (b) describes a two-variable (τ_1, τ_2) controlled example, ($\tau_{1\min}$ - $\tau_{1\max}$)*($\tau_{2\max}$ - $\tau_{2\max}$) different possible combinations and corresponding results are obtained. Therefore, control of n variables needs a computational time equal to ($\tau_{\min}$ - $\tau_{\max}$) ^{n} . Therefore, the main disadvantage of grid search method is that many evaluations (measurements) are considered, so it consumes an extensive amount of CPU time.