NAG1-756

CU-CSSC-90-26

CENTER FOR SPACE STRUCTURES AND CONTROLS

319217

P. 190

# CONSTRAINT TREATMENT TECHNIQUES AND PARALLEL ALGORITHMS FOR MULTIBODY DYNAMIC ANALYSIS

by

Jin-Chern Chiou

November 1990

COLLEGE OF ENGINEERING
UNIVERSITY OF COLORADO
CAMPUS BOX 429
BOULDER, COLORADO 80309

# CONSTRAINT TREATMENT TECHNIQUES AND PARALLEL ALGORITHMS FOR MULTIBODY DYNAMIC ANALYSIS

by

JIN-CHERN CHIOU

B. S., Feng-Chia University, Taiwan, R. O. C., 1981

M. S., University of Colorado at Boulder, 1986

A dissertation submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirement for the degree of

Doctor of Philosophy

Aerospace Engineering Sciences

1990

Chiou, Jin-Chern (Ph.D., Aerospace Engineering Sciences)
Constraint Treatment Techniques and Parallel Algorithms
for Multibody Dynamic Analysis
Dissertation directed by Professor K. C. Park

Computational procedures for kinematic and dynamic analysis of three-dimensional multibody dynamic (MBD) systems are developed from the differential-algebraic equations (DAEs) viewpoint. First, to minimize constraint violations during the time integration process and to obviate degraded constraint force solution involving ill-conditioned matrices, two robust and efficient constraint treatment techniques, *i. e.*, penalty constraint stabilization technique and natural partitioning scheme, are developed. The computational issues for enhancing accuracy, stability, and programming modularity of the techniques are also addressed for MBD analysis.

Second, to treat the governing equations of motion, a two-stage staggered explicit-implicit numerical algorithm, that takes advantage of a partitioned solution procedure and a robust and parallelizable integration algorithm, is developed. Mainly, this algorithm uses a two-stage staggered central difference algorithm to integrate the translational coordinates and the angular velocities. The angular orientations of bodies in MBD systems are then obtained by using an implicit algorithm via the kinematic relationship between Euler parameters and angular velocities. It is shown that the combination of the present solution procedures yields a computationally more accurate solution.

Third, to speed up the computational procedures, parallel implementation of the present constraint treatment techniques, two-stage staggered explicit-implicit numerical algorithm has been efficiently carried out. To this end, the DAEs and the constraint treatment techniques have been transformed into arrowhead matrices to which Schur complement form has been derived. By fully exploiting the sparse matrix structural analysis techniques, a parallel preconditioned conjugate gradient numerical algorithm is used to solve the systems equations written in Schur complement form.

To evaluate the computational procedures developed in the present work, a software testbed has been designed and implemented in both sequential and parallel computers. This testbed has been used to demonstrate the robustness and efficiency of the constraint treatment techniques, the accuracy of the two-stage staggered explicit-implicit numerical algorithm, and the speed up of the Schur-complement-based parallel preconditioned conjugate gradient algorithm on a parallel computer.

This dissertation for the Doctor of Philosophy degree by

Jin-Chern Chiou

has been approved for the

Department of

Aerospace Engineering Sciences

by

[illegible]

K. C. Park

[illegible]

C. A. Felippa

Date 11-14-90

To my mother and to my wife, Min-Hui,

and

in memory of my father.

## ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor K. C. Park, for his guidance, patience, and encouragement during the course of my thesis studies. Without his excellent research and outstanding knowledge of computational dynamics, the research and thesis would not have been possible.

I also wish to thank Professor C. Farhat for his invaluable support and suggestions concerning the practical aspects of running parallel computers. To Professor C. A. Felippa for the many long hours he has spent in critiquing the thesis writing. The knowledge gained as a result of their assistance is immeasurable. I am indebted to all these people.

I would also like to thank my committee members; Professor R. Su, Adjoint Professors J. Housner and E. Schmitz; for devoting their time to reading through my thesis and making valuable suggestions.

Finally, the financial support provided by Dr. J. Housner of the NASA/Langley Research Center through grant NAG-1-756 and Air Force Office of Scientific Research through grant F49620-87-C-0074 is gratefully acknowledged.

## CONTENTS

PRECEDING PAGE BLANK NOT FILMED

## Figures

Figure

# CHAPTER I

# INTRODUCTION

## 1.1 OVERVIEW

The kinematic and dynamic analysis of three dimensional multibody dynamic (MBD) systems has attracted the attention of many researchers over the past two decades. This is due to the fact that many mechanical systems of interest in industry can be effectively modeled by using systems of linked bodies. Moreover the rapid development of computer hardware and software has also played an important role in making the computer simulation of MBD systems more realistic if the number of bodies in the systems remain small. These research activities have primarily concentrated on improving either the design and verification of the control system, or the system design and dynamic analysis of multidisciplinary engineering problems. As a result, several stand-alone general-purpose computer programs [1-11] which are based on different approaches have been developed. These computer programs possess the capability to automatically generate and numerically integrate the equations of motion of multibody problems such as robot arm maneuvers, spacecraft dynamics, and ground vehicle dynamics.

However, when these systems become complex, computational efficiency becomes a dominant issue during the preliminary design stage that may require many analysis iterations. This has motivated several research groups to make effective use of parallel computational technology [12-14] in order to speed up the dynamics analysis of MBD systems, thus ultimately

achieving real-time simulation for large-scale problems. The key issues in exploiting the parallelism inherent in MBD systems include a versatile data structure for describing system topology, an automatic procedure to generate the system equations of motion, a streamlined treatment of system constraints, a robust time integration algorithm, and an easy interpretation of the simulation results.

### 1.1.1 Systematic Study of MBD Formulations

In general, the equations of motion for MBD systems can be derived and expressed in various forms depending upon the type of coordinates chosen to describe the configuration of the bodies. An important kinematic characteristic of these coordinates is how they treat the joints that are used to describe the kinematic relationships of the bodies in the systems. Thus if an arbitrary set of coordinates is chosen, the final system dynamic equations can be interpreted as results of two basic approaches: the augmentation approach and the elimination approach as shown in Fig. 1.1. The first approach gives a set of differential-algebraic equations whereas the second approach gives a set of second-order differential equations.

In order to understand the advantages and disadvantages of using different coordinates to derive the equations of motion, four choices of coordinates will be discussed. The first choice is to use a set of *independent coordinates*, which determine the position of bodies with the least possible number of state variables. A minimal set of second-order differential equations, which is given in terms of system independent variables, is obtained in which the constraint conditions are absent. However, the rapidly growing complexity in the derivation as the number of variables increases,

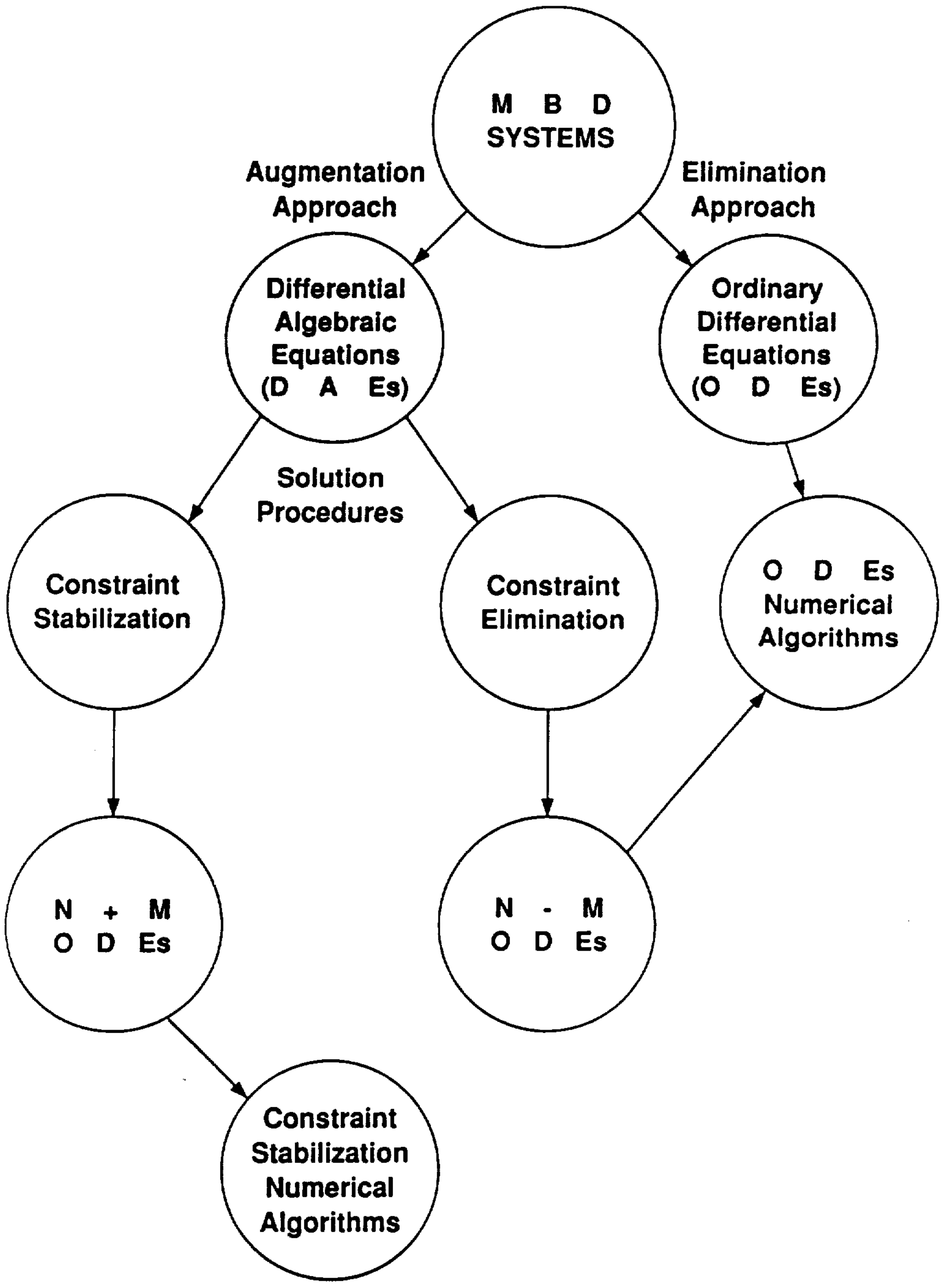


Fig. 1.1 MBD Equations of Motion and Their Corresponding Computational Procedures

and the high degree of nonlinearity of the equations of motion make these coordinates difficult to implement in a general-purpose computer program.

The second choice is that of *relative coordinates* [1-4,15,16], which define the orientation of each moving body with respect to either a non-moving body or another adjacent moving body. For an open tree structure, the number of relative coordinates is equal to the number of independent coordinates. For a closed-loop system, constraint equations are imposed via Lagrange multipliers, in which case the number of relative coordinates exceeds that of independent coordinates. Relative coordinates have the disadvantage in that they do not directly determine the position of each body in the system; thus postprocessing of the simulation results is needed. Furthermore, when the system consists of several closed loops, extensive preprocessing is needed to identify an appropriate set of independent variables.

The third choice is that of *natural coordinates* [17,18], which define a body using two or more moving coordinates rigidly attached to it. These moving coordinates are located preferably at the joints of the mechanism, and can be shared by adjacent bodies. The main advantage of natural coordinates is that they lead to a simple computer implementation and easy formulation in conjunction with quadratic or linear constraint equations. However, the presence of a fully populated mass matrix renders these coordinates less attractive in parallel computation. Another drawback of these coordinates is that during the process of numerical integration a position can be reached which causes the matrix that is used to identify the variable dependencies to become singular. Should this happen, a new linear combination matrix need to be constructed in order to continue the integration process.

The last choice is that of *Cartesian coordinates* [5-8], which define the position of each particle in each individual body in the system with respect to an inertial reference frame. The angular orientation of each body is defined by the body-fixed reference frame via Euler parameters or Euler angles. The main advantages of this choice is that the equations of motion are easy to derive, which facilitates the development of general-purpose computer programs. Since these coordinates yield a maximal set of equations, redundant coordinates and Lagrange multipliers have to be solved as part of the simulation process, which may lead to computational inefficiency unless special attention is paid to computational issues.

If independent coordinates are used, the equations of motion are generated in terms of system degrees of freedom expressed in differential equation form. Obviously this approach leads to a minimal set of equations of motion but suffers from the appearance of dense solution matrices and highly nonlinear kinematic descriptions.

When d'Alembert's principle of virtual work together with Lagrange multipliers are applied to the systems based on relative coordinates, natural coordinates or Cartesian coordinates, the resulting equations for MBD systems are given by a set of second-order differential equations augmented with algebraic constraint equations. These combined system of equations belong to the class of differential-algebraic equations (DAEs). In contrast to the independent coordinates approach, DAEs make use of a larger number of equations yet preserve the sparsity of the solution matrix as well as the simplicity of the kinematic relationships. Furthermore, the approach is amenable to implementation in modular, general-purpose MBD programs.

### 1.1.2 Evaluations of Existing Computational Procedures

A closed-form solution of the MBD equations is in general impossible except for highly simplified problems. Thus time integration algorithms must be used to obtain the numerical solution of the system governing equations. For the second-order differential equations produced by the elimination approach, both the modified explicit central difference formula and as well as stiffly-stable formulas in conjunction with the Newton-Raphson algorithm may yield reasonably stable and accurate solutions. As for differential algebraic equations, Gear [19,20] has investigated a special class of numerical algorithms for the solution of some restricted DAE problems. Orlandea *et al.* [6] have applied this solution technique together with a sparse matrix formulation but encountered numerical problems because the discrete system equations to be solved often become numerically stiff and ill-conditioned.

An alternative approach, advocated by Gear and Petzold [21,22], relies on augmenting the second-order governing equilibrium equations with twice time-differentiated constraint equations so that numerical ordinary differential equations solvers can be applied. However, numerical integration algorithms provide only an approximate solution. As a result, numerical errors will propagate and accumulate so that eventually the constraint conditions are no longer satisfied within the desired accuracy. One approach to stabilize the constraint violations was proposed by Baumgarte [23,24] who modified the original constraint equations to form a set of relaxation differential constraint equations. Park and Chiou [25,26] have shown that for some MBD problems Baumgarte's constraint stabilization technique suffers from ill-conditioning in the solution for Lagrange multipliers. Furthermore, for complicated MBD systems, the process of determining optimal relaxation parameters that are used to tailor the constraint violations to each specific problems may encounter computational difficulties.

An ultimately different approach to avoid constraint violations consists of eliminating the Lagrange multipliers from DAEs so that a set of second-order differential equations is obtained. This can be done by identifying system dependent and independent variables from the given constraint Jacobian matrix so that the null space of the constraint Jacobian matrix can be formed and consequently used to eliminate the Lagrange multipliers. In order to find a set of numerically superior independent variables, several numerical algorithms have been employed to decompose the constraint Jacobian matrix. These algorithms include: the generalized coordinate partitioning scheme [27], the singular value decomposition [28,29], the natural coordinates partitioning scheme [17,18], and the null space scheme [30-32]. Since these computational schemes for determining the set of independent coordinates can become computationally expensive, the chosen set of independent coordinates is maintained during the numerical simulation until the specified accuracy criteria are violated. When this occurs, it is necessary to choose a new set of independent coordinates by repeating the identification process.

Recently, methods based on $O(n)$ algorithms, where $n$ is the number of generalized coordinates, and several variations have been proposed [33-39]. These algorithms are primarily applicable to MBD systems consisting of tree topologies in which their equations of motion may be recursively solved in $O(n)$ operations. If the system topology embodies multiple closed loops, significant modifications are required in order to obtain numerical solutions.

Moreover, the presence of closed loops may cause $O(n)$ algorithms to loose the simplicity of open tree topology in parallel computations.

## 1.2 OBJECTIVES

Since the aforementioned solution procedures suffer various drawbacks in the computer implementation, these have motivated us to look for alternative solution procedures that overcome those difficulties. Alternative solution procedures that involve either constraint stabilization or constraint elimination overcome the following disadvantages: unacceptable constraint violation during the process of time integration; degraded constraint force solution involving ill-conditioned matrix; large computational expense in computing the null space of the constraint Jacobian matrix; inefficiency in using the implicit iterative algorithms; and difficulties in extending existing algorithms to parallel computations.

The objectives of this dissertation are to develop: first, robust and efficient treatments of constraints; second, explicit-implicit integration algorithms to solve DAEs efficiently and accurately; and third, a parallel implementation procedure for a general multibody dynamics simulation capacity.

With these objectives in mind, we will first review the spatial kinematics of linked bodies by employing two sets of coordinates that describe the configuration and velocity of bodies in the system. Inertial coordinates are adopted to locate the center of mass of each of the bodies. Body-fixed coordinates are rigidly attached to the center of mass of each body so that angular orientations of the bodies in the system can be obtained as soon as angular representations are determined. Note that the purpose of chosing inertial coordinates for the translational motions and body-fixed coordinates for the angular motions is to decouple the inertia matrix to obtain the translational and rotational equations separately. After the kinematics of a body in space has been determined, a formulation based on d'Alembert's principle of virtual work is adopted to derive the system governing equations by treating the bodies in the system as originally independent of each other. Nonlinear kinematic constraint conditions, which are imposed to describe the interconnectivity between the various bodies, are appended to the formulation using Lagrange multipliers. The final system equations of motion, which are characterized as DAEs, not only enhance programming modularity but can also be generated automatically. As mentioned previously, the use of existing numerical time integration solution procedures may encounter computational difficulties. In this regard, two newly developed schemes based on constraint stabilization (penalty constraint stabilization scheme) and constraint elimination (natural partitioning scheme), are introduced to correct for the constraint violations accurately and efficiently.

### 1.2.1 Robust and Efficient Treatment of Constraints

The penalty constraint stabilization scheme is based on the observation that time-differentiated equations of penalty form retain a parabolic characteristic in time. Thus, as time progresses constraint violations will decay according to intrinsic time constants. This penalty time-differentiated form, which is given by the time rate of the constraint forces, enables us to overcome the difficulties that have been encountered in Baumgarte's technique. Furthermore, this scheme offers the attractive feature that the system equations can be processed in two modules: the generalized coordinates module and the generalized constraint forces module. This separation fits nicely in the framework of the partitioned procedure [40-42] adopted for the time integration.

The natural partitioning scheme, which is quite different from the penalty constraint stabilization scheme, uses the existing physical coordinates by explicitly identifying their dependent and independent coordinates without relying heavily on the numerical algorithms that have been mentioned previously. The resulting null space of the constraint Jacobian matrix can be generated in parallel if the system topology consists of several open chains. Applying the null space of the constraint Jacobian matrix to the governing DAEs leads to the elimination of the Lagrange multipliers and yields a set of second-order differential equations that are expressed in terms of system independent variables.

### 1.2.2 Explicit-Implicit Integration of MBD Equations

In the present formulation, both angular velocity-dependent centripetal accelerations and the angular accelerations appear in the equations of motion that represent the rotational motions of linked bodies. Direct time integration of angular velocities, except for some simple kinematic configurations, does not directly yield the angular orientation of a body in space. Hence, a partitioned solution procedure [40-42], which has the capability to separately solve coupled systems of equations while treating interaction terms as external forces, is used to separately integrate translational and rotational equations. To obtain a robust and parallelizable integration algorithm, the explicit central difference time integration formula is thought to be the best candidate to treat the partitioned translational and rotational quantities. If the central difference formula is adopted, the approximation of the angular velocity for the evaluation of angular acceleration leads to numerical instability for the governing equations of motion. This has motivated us to exploit a two-stage explicit procedure which stabilizes the central difference algorithm and delivers an accurate solution. The Euler parameters are used in the present solution procedure to represent the angular orientations of bodies. An implicit mid-point time integration algorithm is employed to integrate the Euler parameters by exploiting the kinematic relationship with the associated angular velocities. The specific implicit algorithm presented has been chosen because it is unconditionally numerically stable while it can be analytically inverted during actual computer implementation because of its special four by four matrix form.

Combining these solution algorithms a two-stage staggered explicit-implicit solution procedure [43] has been developed. This solution procedure, which invokes either the penalty constraint stabilization scheme or the natural partitioning scheme to stabilize the constraint violations, has been implemented in a computer program to validate and demonstrate its robustness and accuracy.

The present solution procedure based on the penalty constraint stabilization scheme consists of two modularized solvers: the generalized coordinate solver and the constraint force solver. The solution procedure based on the natural partitioning scheme includes the generalized coordinate solver and the independent coordinate solver. The generalized coordinate solver combines an improved version of the explicit central difference algorithm for integration of the translational coordinates and angular velocity with an implicit algorithm to update the Euler parameter representation of angular orientations by exploiting the uncoupled inertia expression. The procedure has successfully been interfaced with the penalty staggered stabilization technique which solves the constraint forces as independent variables by implicitly integrating a stabilized companion differential equation for the constraint forces in time. The combination of the two algorithms can be invoked in a sequential manner on the rigid and flexible components of the multibody system resulting in an attractive, modular solution procedure. As for the independent variable solver, a procedure based on body-by-body constraint Jacobian matrices is developed to explicitly form the null space of the constraint Jacobian matrix and consequently obtain system independent variables.

### 1.2.3 Parallel Implementation Procedures for MBD Analyzer

Since an MBD system may consist of hundreds or even thousands of bodies, the numerical solution may consume a prohibitive amount of CPU time. To reduce the CPU time dramatically, it is advantageous to develop efficient parallel algorithms by incorporating existing parallel computers. In general, issues involving parallel computations of MBD systems include generation of the system equations of motion, incorporation or elimination of constraint forces, integration of generalized coordinates, and interpretation of the simulation results. A method for exploiting the parallelism of the present constraint stabilization, constraint elimination and two-stage staggered explicit-implicit solution procedure has been developed. The main thrust of this method uses a Schur-complement-based parallel preconditioned conjugate gradient numerical algorithm to decide either the generalized acceleration vector and constraint forces of the penalty constraint stabilization scheme or the generalized and independent acceleration vectors of the natural partitioning scheme. The present algorithm has been implemented and tested on existing parallel computers and has provided encouraging results in practical MBD problems.

## 1.3 DISSERTATION OUTLINE

The dissertation is organized as follows. Chapter 2 reviews the spatial kinematics of rigid bodies used in the present MBD formulation. Chapter 3 employs d'Alembert's principle of virtual work to derive the governing equations of motion that consist of a set of algebraic constraint equations coupled with the second-order differential equations of motion. Chapter 4 derives the mechanical properties of the joints that connect bodies in the MBD system. These joints are introduced in the dynamic formulation using a set of algebraic constraint equations that are adjoined to the equations of motion in constraint Jacobian matrix form. Chapter 5 deals with the DAEs by reviewing several existing numerical solution procedures. Two newly developed schemes based on the penalty constraint stabilization scheme and the natural partitioning scheme are employed to overcome computational difficulties associated with those solution procedures. A two-stage staggered explicit-implicit algorithm for updating the translational coordinates, angular orientations, and constraint forces is developed based on the proposed technique. Chapter 6 analyzes and exploits the parallelism inherent in the solution procedures. Chapter 7 gives numerical example problems in order to demonstrate the robustness and efficiency of utilizing the present solution procedures. Chapter 8 summarizes the accomplishments of the present investigation and discusses directions for further research in the field of multibody systems.

# CHAPTER II

# SPATIAL KINEMATICS

## 2.1 Introduction

Kinematics, which is the study of the motion of particles and bodies without the forces associated with these motions, has been used to analyze the position, velocity, and acceleration of bodies and determine the design geometry of the bodies in the mechanical systems. In this chapter, we begin with the derivation of different finite rotational representations and subsequently obtain the position, velocity, and acceleration of a particle in space. Finally, we will consider the particle as if it has been attached to a rigid body and thus ultimately complete the derivation of kinematics for the rigid body.

## 2.2 Reference Frames

In mechanics, a most fundamental technique is using vectorial quantities to locate the position of a particle in a given reference frame. When the position vector from the origin of that reference frame to the particle has been defined, we can resolve this position vector by one or more systems of coordinates for a particular use. In many dynamics problems, relations between the component of the vector in various reference frames prove extremely useful. To derive such relations, let us consider a position vector $\mathbf{r}_p$ (Fig. 2.1) expressed in terms of three components parallel to the three axes of a Cartesian frame:

$$\mathbf{r}_p = r_1^e \underline{e}_1 + r_2^e \underline{e}_2 + r_3^e \underline{e}_3 \qquad (2.2.1)$$

where $(r_1, r_2, r_3)^e$ are the coordinates of the particle in the inertial reference frame, and $(\underline{e}_1, \underline{e}_2, \underline{e}_3)$ are the basis vectors fixed in the inertial reference frame. Similarly, $\mathbf{r}_p$ can be expressed in another reference frame as

$$\mathbf{r}_p = r_1^b \underline{b}_1 + r_2^b \underline{b}_2 + r_3^b \underline{b}_3 \tag{2.2.2}$$

where $(r_1, r_2, r_3)^b$ are the coordinates of $\mathbf{r}_p$ expressed in the **b** reference frame, and $(\underline{b}_1, \underline{b}_2, \underline{b}_3)$ are the basis vectors of an arbitrary moving reference frame.

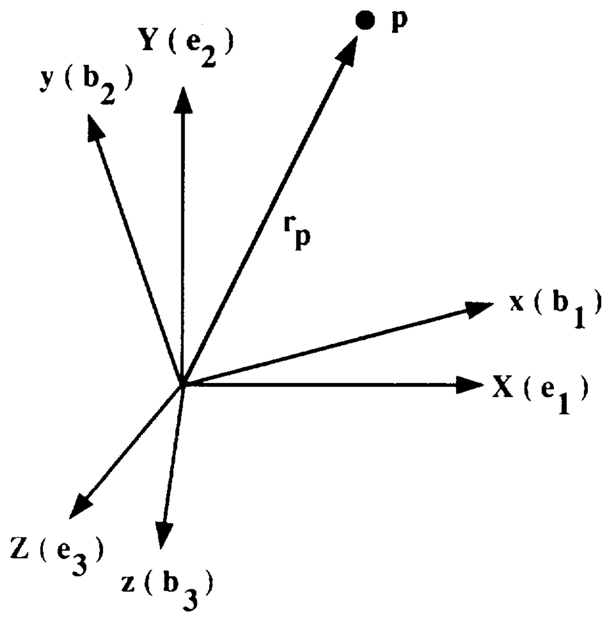


Fig. 2.1 Position Vector in Three-Dimensional Space

Since (2.2.1) and (2.2.2) describe the same vector, the components of the **e** reference frame must evidently be related to the **b** reference frame. The relation between the two bases can be established by writing the orthogonal projection of $\mathbf{r}_p$ with respect to the **e** basis vector so that

$$r_1^e = \mathbf{r}_p \cdot \underline{e}_1 = (\underline{e}_1 \cdot \underline{b}_1) r_1^b + (\underline{e}_1 \cdot \underline{b}_2) r_2^b + (\underline{e}_1 \cdot \underline{b}_3) r_3^b \tag{2.2.3}$$

$$r_2^e = \mathbf{r}_p \cdot \underline{e}_2 = (\underline{e}_2 \cdot \underline{b}_1)r_1^b + (\underline{e}_2 \cdot \underline{b}_2)r_2^b + (\underline{e}_2 \cdot \underline{b}_3)r_3^b \tag{2.2.4}$$

$$r_3^e = \mathbf{r}_p \cdot \underline{e}_3 = (\underline{e}_3 \cdot \underline{b}_1)r_1^b + (\underline{e}_3 \cdot \underline{b}_2)r_2^b + (\underline{e}_3 \cdot \underline{b}_3)r_3^b \tag{2.2.5}$$

These relations can be written in the following matrix form:

$$\mathbf{r}^e = \mathbf{R}^T \mathbf{r}^b \tag{2.2.6}$$

where

$$\mathbf{R}^T = \begin{bmatrix} \underline{e}_1 \cdot \underline{b}_1 & \underline{e}_1 \cdot \underline{b}_2 & \underline{e}_1 \cdot \underline{b}_3 \\ \underline{e}_2 \cdot \underline{b}_1 & \underline{e}_2 \cdot \underline{b}_2 & \underline{e}_2 \cdot \underline{b}_3 \\ \underline{e}_3 \cdot \underline{b}_1 & \underline{e}_3 \cdot \underline{b}_2 & \underline{e}_3 \cdot \underline{b}_3 \end{bmatrix} \tag{2.2.7}$$

$\mathbf{r}^e = [r_1^e, r_2^e, r_3^e]^T$, and $\mathbf{r}^b = [r_1^b, r_2^b, r_3^b]^T$. Equation (2.2.6) can be explicitly rewritten as the relation of the two basis vectors:

$$\mathbf{e} = \mathbf{R}^T \mathbf{b} \tag{2.2.8}$$

where matrix $\mathbf{R}$ is called the coordinate transformation matrix or direction cosine matrix between the two sets of axes.

Since $\mathbf{r}_p$ preserves the property of constant length regardless of the basis vectors selected,

$$\mathbf{r}_p^{b\,T} \mathbf{r}_p^b = \mathbf{r}_p^{e\,T} \mathbf{R}^T \mathbf{R} \mathbf{r}_p^e = \mathbf{r}_p^{e\,T} \mathbf{r}_p^e \tag{2.2.9}$$

which implies that

$$\mathbf{R}^T \mathbf{R} = \mathbf{I} \tag{2.2.10}$$

or

$$\mathbf{R}^{-1} = \mathbf{R}^T \tag{2.2.11}$$

A matrix satisfying relation (2.2.11) is called an orthogonal matrix. Pre-multiplying (2.2.8) by $\mathbf{R}$ and recalling (2.2.10) yields the relations of an arbitrary moving reference frame that is expressed in terms of the inertial reference frame as

$$\mathbf{b} = \mathbf{R}\mathbf{e} \tag{2.2.12}$$

## 2.3 Angular Orientations of a Particle

From the previous derivation, we conclude that at any specific time the position of a particle, which may be expressed in terms of suitable sets of reference frames, can be specified by a transformation matrix. As time passes, the position vector orientation changes and so does coordinate transformation matrix. This leads to the development of the *Euler* theorem which provides us with a foundation to develop various types of angular orientations so that the coordinate transformation matrix may be determined. *Euler's theorem states that two arbitrarily oriented dextral basis vectors* **b** *and* **e**, *with common origin can be made to coincide with one another by rotating one of them through a certain angle about an axis which passes through that origin.* In short, any rotation can be described by rotating a vector about a proper unit axis **n** through an angle $\phi$ as shown in Fig. 2.2. The rotational operator acting on the vector can be represented by

$$\mathbf{R}(\mathbf{n}, \phi) = \mathbf{n}\mathbf{n} + \cos\phi(\mathbf{I} - \mathbf{n}\mathbf{n}) + \sin\phi\mathbf{n} \times \mathbf{I} \tag{2.3.1}$$

Note that if the full coordinate transformation matrix is used, it means that we choose to parametrize **R** by nine parameters, the nine direction cosines themselves. However, the orthonormal property of the $3 \times 3$ coordinate transformation matrix **R** will lead to the consequence that it can only be represented by three independent parameters if the six orthonormal constraints are imposed. The choice of these parameters presents an important aspect in computing the angular orientation of bodies in MBD systems. Which kind of parameters one should adopt are a matter of judgment by individual researcher. The fewer the parameters, the fewer constraint equations need to be satisfied. However, sometimes it is necessary to compute the coordinate transformation matrix at every time integration step, which cancels some of the advantages by using a small number of parameters. Moreover, the use of three parameters always lead to a singular coordinate transformation matrix when certain angles are reached. To specify **R** with various parameters, several commonly used parameters will be listed along with some important properties so that the kinematic relationships of the parameters and their corresponding rotational operator are defined.

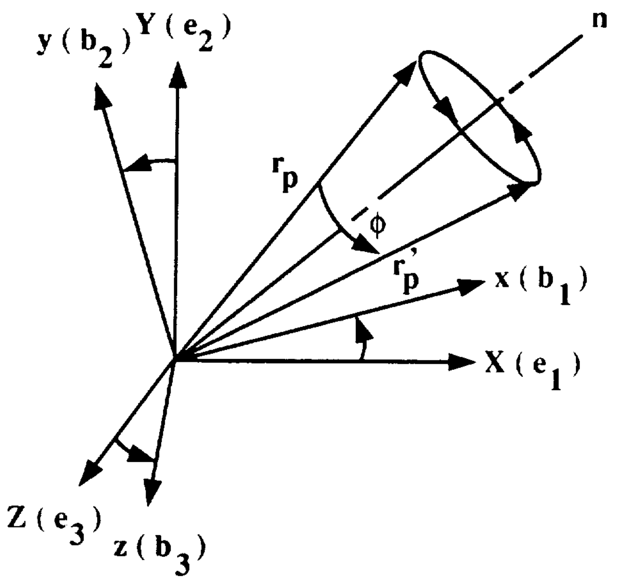


Fig. 2.2 System Rotational Coordinate

## 2.4 Euler Angles

The most common minimal set of parameters used to describe the angular orientation of a body in space are *Euler* angles. These angles provide a set of coordinates without involving any constraint equations. The *Euler* angles formalism consists of three successive rotations, started by rotating the $k$-axis through angle $\alpha$ of a specified orthogonal basis $[i, j, k]$. The resulting coordinates are labeled $[i', j', k']$. Next, rotate the $i'$-axis by an angle $\beta$ so that another set of coordinates $[i'', j'', k'']$ are obtained. Finally, rotate the $k''$-axis by an angle $\gamma$ to produce the desired system rotational axes. To express the effective rotational axis **n** through an angle $\phi$ in terms of these three successive rotations, the rotational operator can be written

$$\mathbf{R}(\mathbf{n}, \phi) = \mathbf{R}(k'', \gamma) \cdot \mathbf{R}(i', \beta) \cdot \mathbf{R}(k, \alpha) \tag{2.4.1}$$

or

$$\mathbf{R}(\mathbf{n}, \phi) = \begin{bmatrix} c\gamma c\alpha - s\gamma c\beta s\alpha & s\gamma c\alpha + c\gamma c\beta s\alpha & s\beta s\alpha \\ -c\gamma s\alpha - s\gamma c\beta c\alpha & -s\gamma s\alpha + c\gamma c\beta c\alpha & s\beta c\alpha \\ s\gamma s\beta & -c\gamma s\beta & c\beta \end{bmatrix} \tag{2.4.2}$$

The successive rotations in (2.4.1) are

$$\begin{aligned} \mathbf{b}'' &= \mathbf{R}(k'', \gamma)\mathbf{b}' \\ \mathbf{b}' &= \mathbf{R}(i', \beta)\mathbf{b} \\ \mathbf{b} &= \mathbf{R}(k, \alpha)\mathbf{e} \end{aligned} \tag{2.4.3}$$

with $(c \equiv \cos, s \equiv \sin)$,

$$\mathbf{R}(k'', \gamma) = \begin{bmatrix} c\gamma & s\gamma & 0 \\ -s\gamma & c\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}(i',\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\beta & s\beta \\ 0 & -s\beta & c\beta \end{bmatrix} \tag{2.4.4}$$

$$\mathbf{R}(k,\alpha) = \begin{bmatrix} c\alpha & s\alpha & 0 \\ -s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplying these together yields (2.4.2). The coordinate transformation matrix in terms of *Euler* angles presents some numerical difficulties: first, they involve trigonometric functions which are numerically expensive to compute; second, the coordinate transformation matrix becomes singular when $\beta = n\pi, n = 0, \pm1, \pm2, ...$, in which case both rotations along $k$ and $k''$ become collinear. This can be illustrated by setting $\beta = 0$ so that

$$\mathbf{R} = \begin{bmatrix} c(\alpha+\gamma) & s(\alpha+\gamma) & 0 \\ -s(\alpha+\gamma) & c(\alpha+\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which represents a single rotation $\alpha + \gamma$ about $k$-axis.

In mechanical analysis, sometimes it is necessary to calculate *Euler* angles by using a given coordinate transformation matrix where

$$\mathbf{R}(\alpha,\beta,\gamma) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.4.5}$$

To determine the corresponding *Euler* angles, we calculate first

$$\alpha = \tan^{-1}(\frac{r_{13}}{r_{23}}) \tag{2.4.6}$$

by recalling (2.4.2) so that $\beta$ and $\gamma$ can be evaluated without any ambiguity

$$s\beta = r_{13}s\alpha + r_{23}c\alpha \quad ; \quad c\beta = r_{33} \tag{2.4.7}$$

$$c\gamma = r_{11}c\alpha - r_{21}s\gamma \quad ; \quad s\gamma = r_{12}c\alpha - r_{22}s\gamma \tag{2.4.8}$$

## 2.5 Rodrigues Parameters

The *Rodrigues* parameters are defined as

$$\gamma = \mathbf{n} \tan \frac{\phi}{2} \tag{2.5.1}$$

where $\mathbf{n}$ and $\phi$ are the unit vectors along the rotational axes and the rotation angle. Obviously

$$\gamma^T \gamma = \tan^2 \frac{\phi}{2} \tag{2.5.2}$$

so that

$$\cos^2 \frac{\phi}{2} = \frac{1}{1 + \gamma^T \gamma} \tag{2.5.3}$$

Since

$$\cos \phi = 2 \cos^2 \frac{\phi}{2} - 1 \tag{2.5.4}$$

substituting (2.5.3) into (2.5.4) yields

$$\cos \phi = \frac{1 - \gamma^T \gamma}{1 + \gamma^T \gamma} \tag{2.5.5}$$

Again, $\mathbf{n} \sin \phi$ can be written in terms of $\gamma$ as

$$\mathbf{n} \sin \phi = 2\mathbf{n} \sin \frac{\phi}{2} \cos \frac{\phi}{2} = 2\gamma \cos^2 \frac{\phi}{2} = \frac{2\gamma}{1 + \gamma^T \gamma} \tag{2.5.6}$$

Replacement of $\mathbf{n} \sin \phi$ and $\cos \phi$ in (2.3.1) by (2.5.6) and (2.5.5) leads to

$$\mathbf{R} = \frac{1}{1 + \gamma^T \gamma} [(1 - \gamma^T \gamma)\mathbf{I} + 2(\gamma\gamma^T - \tilde{\gamma})] \tag{2.5.7}$$

where

$$\tilde{\gamma} = \begin{bmatrix} 0 & -\gamma_3 & \gamma_2 \\ \gamma_3 & 0 & -\gamma_1 \\ -\gamma_2 & \gamma_1 & 0 \end{bmatrix} \tag{2.5.8}$$

As in the case of *Euler* angles, the *Rodrigues* parameters use a minimal set of three variables. Unlike the *Euler* angles, the *Rodrigues* parameters expression do not involve trigonometric functions. This can be an advantage for actual numerical computation. However, they have the disadvantage of becoming infinite if the rotation angle $\phi$ becomes $\pm k\pi, k = 1, 2, \ldots\ldots$ Again it is sometimes desirable to compute the *Rodrigues* parameters if the coordinate transformation matrix is given. These relations are obtained by subtracting symmetrically from the off-diagonal terms of (2.5.7) which yield

$$4\gamma_i = [1 + \tan^2 \frac{\phi}{2}](R_{jk} - R_{kj}) \tag{2.5.9}$$

Since

$$1 + tr(\mathbf{R}) = \frac{4}{1 + \tan^2 \frac{\phi}{2}} \tag{2.5.10}$$

therefore by substituting (2.5.9) into (2.5.8), we obtain the expression of computing $\gamma_i$ that

$$\gamma_i = \frac{1}{1 + tr(\mathbf{R})}(R_{jk} - R_{kj}) \tag{2.5.11}$$

Note that if $1 + tr(\mathbf{R}) = 0$, then $\gamma_i$ approach infinity which occurs when $\phi = \pm k\pi$ as is concluded from previous definition.

## 2.6 Euler Parameters

To avoid degeneration of the coordinate transformation matrix for certain values of the parametrizing variables, one has to use more than the minimal set of three parameters. The choice of the parameters to represent the angular orientations of bodies in MBD systems needs to satisfy the following requirements:

(1) Singularity should not occur for any chosen parameters.

(2) To prevent expensive calculations of trigonometric functions, an algebraic description of finite rotations is preferred.

(3) To avoid redundancies in descriptions of parameters, a minimal set of parameters is preferred.

In the present research, *Euler* parameters have been chosen to represent the angular orientations of the bodies for the following reasons:

(1) *Euler* parameters satisfy the singularity free property that other sets of rotational parametrizations such as *Euler* angles can not provide.

(2) *Euler* parameters preserve the algebraic description of finite rotations of bodies in the systems.

(3) The use of *Euler* parameters may drastically simplify the mathematical formulation.

*Euler* parameters are defined as

$$\begin{aligned} q_0 &= \cos\frac{\phi}{2} \\ \mathbf{q}_n &= \mathbf{n}\sin\frac{\phi}{2} \end{aligned} \tag{2.6.1}$$

with the constraint equation

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \tag{2.6.2}$$

where $\mathbf{q}_n = [q_1, q_2, q_3]^T$. The time derivative of (2.6.2) is given by

$$q_0\dot{q}_0 + \mathbf{q}_n^T\dot{\mathbf{q}}_n = 0 \tag{2.6.3}$$

Introducing the standard trigonometric relationships

$$\begin{aligned} \cos\phi &= 2\cos^2\frac{\phi}{2} - 1 \\ \sin\phi &= 2\sin\frac{\phi}{2}\cos\frac{\phi}{2} \end{aligned} \tag{2.6.4}$$

and substituting (2.6.1) into (2.3.1), the rotational operator dyadic $\mathbf{R}$ becomes

$$\mathbf{R} = (2q_0^2 - 1)\mathbf{I} + 2(\mathbf{q}_n\mathbf{q}_n^T - q_0\tilde{\mathbf{q}}_n) \tag{2.6.5}$$

or

$$\mathbf{R} = 2\begin{bmatrix} q_0^2 + q_1^2 - \frac{1}{2} & q_1q_2 + q_0q_3 & q_1q_3 - q_0q_2 \\ q_1q_2 - q_0q_3 & q_0^2 + q_2^2 - \frac{1}{2} & q_2q_3 + q_0q_1 \\ q_1q_3 + q_0q_2 & q_2q_3 - q_0q_1 & q_0^2 + q_3^2 - \frac{1}{2} \end{bmatrix} \tag{2.6.6}$$

where

$$\tilde{\mathbf{q}}_n = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{2.6.7}$$

In contrast to *Euler* angles and *Rodrigues* parameters, the coordinate transformation matrix (2.6.6) can not become singular. Again, if **R** is given, the corresponding *Euler* parameters can be determined by taking the trace of **R** from (2.6.6) so that

$$q_0^2 = \frac{1 + tr(\mathbf{R})}{4} \tag{2.6.8}$$

Substituting (2.6.8) into the diagonal terms of (2.6.6) results in

$$q_i^2 = \frac{1 + 2R_{ii} - tr(\mathbf{R})}{4}, \quad i = 1, 2, 3 \tag{2.6.9}$$

Equations (2.6.8) and (2.6.9) determine the magnitudes of the *Euler* parameters. The off-diagonal terms of (2.6.6) can be used to decide the sign of the *Euler* parameters. Subtracting and adding symmetrically from the off-diagonal terms of (2.6.6) yields

$$\begin{aligned} q_1 &= \frac{R_{32} - R_{23}}{4q_0} \quad ; \quad q_0 = \frac{R_{32} - R_{23}}{4q_1} \\ q_2 &= \frac{R_{13} - R_{31}}{4q_0} \quad ; \quad q_0 = \frac{R_{13} - R_{31}}{4q_2} \\ q_3 &= \frac{R_{21} - R_{12}}{4q_0} \quad ; \quad q_0 = \frac{R_{21} - R_{12}}{4q_3} \end{aligned} \tag{2.6.10}$$

and

$$\begin{aligned} q_1q_2 &= \frac{R_{12} + R_{21}}{4} \\ q_2q_3 &= \frac{R_{23} + R_{32}}{4} \end{aligned} \tag{2.6.11}$$

$$q_3 q_1 = \frac{R_{31} + R_{13}}{4}$$

According to previous derivations, the following algorithm is used to determined the *Euler* parameters when **R** is given [44]:

```
det = max ( $\mathrm{tr}(\mathbf{R})$, $R_{11}, R_{22}, R_{33}$ )
if ( det = $\mathrm{tr}(\mathbf{R})$ ) then
    use (2.6.8) to find $q_0$
    use (2.6.10a) to find $q_1$, $q_2$, and $q_3$
elseif ( det = $R_{ii}$ ) then
    use (2.6.9) to find $q_i$
    Compute $q_0 = \frac{(R_{kj} - R_{jk})}{4q_i}$ from (2.6.10b)
    Compute $q_p = \frac{(R_{pi} + R_{ip})}{4q_i}$,  $p \neq i$ from (2.6.11)
endif
```

## 2.7 Angular Velocity

Consider the orientation of the **b** basis with respect to the **e** basis as given by (2.2.12). The time derivative of **b** is

$$\dot{\mathbf{b}} = \dot{\mathbf{R}}\mathbf{e} + \mathbf{R}\dot{\mathbf{e}} \tag{2.7.1}$$

Since **e** is a fixed basis vector, which implies $\dot{\mathbf{e}} = 0$, therefore

$$\dot{\mathbf{b}} = \dot{\mathbf{R}}\mathbf{e} = \dot{\mathbf{R}}\mathbf{R}^T\mathbf{b} \tag{2.7.2}$$

To relate $\dot{\mathbf{R}}$ and **R**, we differentiate the identity matrix (2.2.10) with respect to time:

$$\dot{\mathbf{R}}^T\mathbf{R} + \mathbf{R}^T\dot{\mathbf{R}} = 0 \tag{2.7.3}$$

By assuming that $\dot{\mathbf{R}} = \mathbf{S}\mathbf{R}$ and substituting into (2.7.3) we get

$$\mathbf{R}^T\mathbf{S}^T\mathbf{R} + \mathbf{R}^T\mathbf{S}\mathbf{R} = 0 \tag{2.7.4}$$

Premultiplying (2.7.4) by $\mathbf{R}$ and postmultiplying by $\mathbf{R}^T$ yields

$$\mathbf{S} + \mathbf{S}^T = 0 \tag{2.7.5}$$

which implies that $\mathbf{S}$ is a skew-symmetric matrix. Hence the matrix kinematic equation for the rotation can be defined as

$$\dot{\mathbf{R}} = \mathbf{SR} = \tilde{\omega}^T \mathbf{R} \tag{2.7.6}$$

or

$$\tilde{\omega}^T = \dot{\mathbf{R}}\mathbf{R}^T \tag{2.7.7}$$

where

$$\tilde{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.7.8}$$

The three components in (2.7.8) are the angular velocity components of the moving $\mathbf{b}$ basis relative to the inertial $\mathbf{e}$ basis that can be written into the following form by substituting (2.7.6) into (2.7.2)

$$\dot{\mathbf{b}} = \dot{\mathbf{R}}\mathbf{R}^T\mathbf{b} = \tilde{\omega}^T\mathbf{R}\mathbf{R}^T\mathbf{b} = \tilde{\omega}^T\mathbf{b} \tag{2.7.9}$$

where the angular velocity vector, $\omega$, can be written as

$$\omega = \omega_1\mathbf{b}_1 + \omega_2\mathbf{b}_2 + \omega_3\mathbf{b}_3 \tag{2.7.10}$$

From the present derivation, we conclude that the angular velocity $\omega$ is a function of the coordinate transformation matrix $\mathbf{R}$ and its time derivatives.

## 2.8 Time Derivatives of Euler Parameters

In this section, the relations between *Euler* parameters and angular velocities are derived. These relations are established by taking the time derivative of (2.6.5):

$$\dot{\mathbf{R}} = 4\dot{q}_0 q_0 \mathbf{I} + 2\dot{\mathbf{q}}_n\mathbf{q}_n^T + 2\mathbf{q}_n\dot{\mathbf{q}}_n^T - 2\dot{q}_0\tilde{\mathbf{q}}_n - 2q_0\dot{\tilde{\mathbf{q}}}_n \tag{2.8.1}$$

Substituting (2.6.5) and (2.8.1) into (2.7.7), the angular velocity, $\omega$, can be expressed in terms of *Euler* parameters and their time derivatives as

$$\omega = 2q_0\dot{\mathbf{q}}_n - 2\dot{q}_0\mathbf{q}_n - 2\tilde{\mathbf{q}}_n\dot{\mathbf{q}}_n = 2\mathbf{T}\dot{\mathbf{q}} \tag{2.8.2}$$

where

$$\mathbf{T} = \begin{bmatrix} -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \tag{2.8.3}$$

Differentiating (2.8.2) with respect to time yields

$$\dot{\omega} = 2\mathbf{T}\ddot{\mathbf{q}} + 2\dot{\mathbf{T}}\dot{\mathbf{q}} \tag{2.8.4}$$

Expansion of the product $\mathbf{Tq}$ shows that it vanishes, and so does $\dot{\mathbf{T}}\dot{\mathbf{q}}$. Hence $\dot{\omega} = 2\mathbf{T}\ddot{\mathbf{q}}$ and its inverse relation is

$$\ddot{\mathbf{q}} = \frac{1}{2}\mathbf{T}^T\dot{\omega} - \frac{1}{4}(\omega^T\omega)\mathbf{q} \tag{2.8.5}$$

Note that the scalar $\omega^T\omega = \omega^2$ can also be written as $4\dot{\mathbf{q}}^T\mathbf{q} = \omega^2$ if (2.8.2) is used. Appending the differential form of the constraint equation (2.6.3) to (2.8.2), the angular velocity can be written in terms of *Euler* parameters as

$$\begin{Bmatrix} 0 \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{Bmatrix} = 2\begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & q_3 & -q_2 \\ -q_2 & -q_3 & q_0 & q_1 \\ -q_3 & q_2 & -q_1 & q_0 \end{bmatrix}\begin{Bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{2.8.6}$$

The inverse of (2.8.6) is

$$\begin{Bmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix}\begin{Bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{Bmatrix} = A(\omega)\mathbf{q} \tag{2.8.7}$$

where $\mathbf{q} = [q_0, q_1, q_2, q_3]^T$. In chapter 5, these derivations will be used to formulate the computational sequences and obtain the angular orientations of bodies in the system.

## 2.9 Velocity and Acceleration of a Particle

In the previous section, we have studied the rate change of a vector fixed in the moving reference frame. In the present section, we derive the expression for the time derivative of a vector whose components along the moving frame are varying with time. Such a vector can be expressed in terms of two different basis vectors as shown in (2.2.1) and (2.2.2) where

$$\mathbf{r}_p^e = \mathbf{r}_p^b = r_1^b b_1 + r_2^b b_2 + r_3^b b_3 \tag{2.9.1}$$

Differentiating (2.9.1) with respect to time and making the use of (2.7.9) yields

$$\dot{\mathbf{r}}_p^e = \dot{\mathbf{r}}_p^b + \omega \times \mathbf{r}_p^b \tag{2.9.2}$$

where $\dot{\mathbf{r}}_p^b$ denotes the time rate relative to **b** basis and $\omega \times \mathbf{r}_p^b$ denotes the time rate of $\mathbf{r}_p^b$ due to the rotational motion of the moving frame. Note that (2.9.2) represents the time derivative of the position vector $\mathbf{r}_p^e$ in an inertial reference frame whereas the vector $\mathbf{r}_p^b$ is expressed in terms of a moving reference frame that is valid for any vector in space. Thus, differentiating (2.9.2) with respect to time, we obtain an expression for the acceleration of point $p$:

$$\ddot{\mathbf{r}}_p^e = \ddot{\mathbf{r}}_p^b + 2\omega \times \dot{\mathbf{r}}_p^b + \dot{\omega} \times \mathbf{r}_p^b + \omega \times \omega \times \mathbf{r}_p^b \tag{2.9.3}$$

where $\ddot{\mathbf{r}}_p^e$ is the acceleration of $p$ in the **e** basis, $\ddot{\mathbf{r}}_p^b$ is the acceleration of $p$ in the **b** basis, $2\omega \times \dot{\mathbf{r}}_p^b$ is the Coriolis acceleration, $\dot{\omega} \times \mathbf{r}_p^b$ is the angular acceleration in the **b** basis, and $\omega \times \omega \times \mathbf{r}_p^b$ is the centripetal acceleration.

## 2.10 Velocity and Acceleration of a Rigid Body

Having derived the kinematics of a particle in space, it is appropriate to study the velocity and acceleration of a rigid body. Because an unconstrained rigid body possesses of six degrees of freedom, it is generally convenient to choose six coordinates that consist of three translations of a point within the body and three rotations about that point, to describe the motion of the body in space. To this end, consider a position vector $\mathbf{r}_p$ (Fig. 2.3) on the rigid body which can be decomposed to

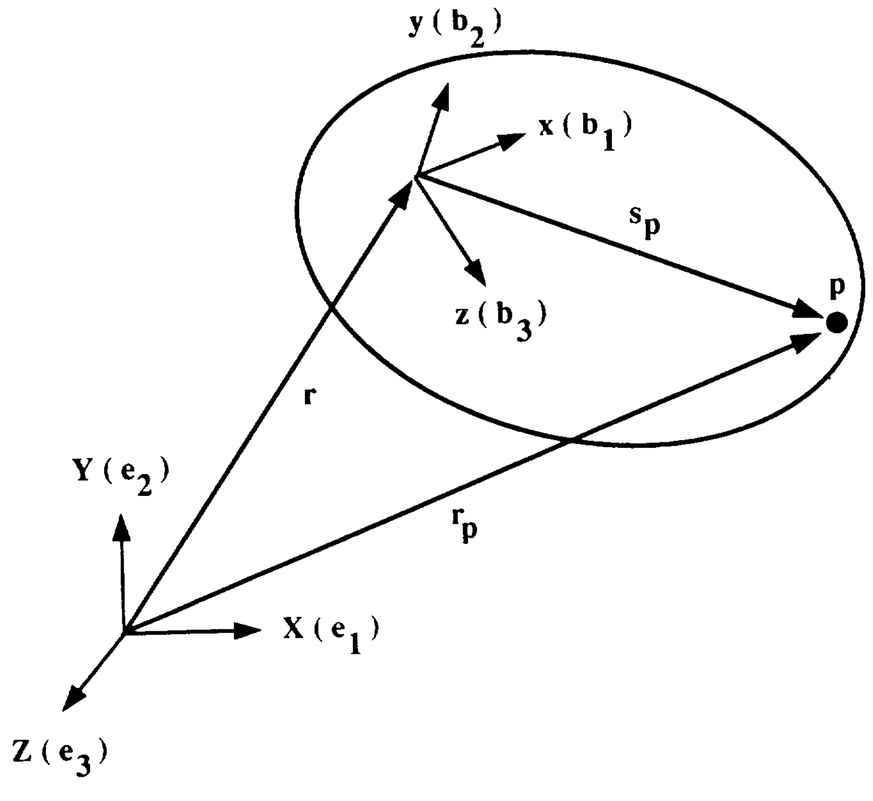


Fig. 2.3 Translation and Rotation of a Body in Three-Dimensional Space

$$\mathbf{r}_p = \mathbf{r}_o + \mathbf{s}_p = \mathbf{r}^T\mathbf{e} + \mathbf{l}^T\mathbf{b} \tag{2.10.1}$$

where $\mathbf{r}_o$ is the position vector from the origin of the inertial reference frame to the origin of the body-fixed reference frame, $\mathbf{s}_p$ is the position vector from the origin of the body-fixed reference frame to a point $p$ of the body, $\mathbf{e}$ is the basis vector of the inertial reference frame, and $\mathbf{b}$ is the basis vectors of the body-fixed reference frame that describes the orientation. Also $\mathbf{r}$ is the position vector of point $o$ in the inertial reference frame, and $\mathbf{l}$ is the position vector of point $p$ in the body-fixed reference frame. By adopting (2.9.2) and (2.9.3), the velocity and acceleration vectors of point $p$ can be expressed as

$$\dot{\mathbf{r}}_p = \dot{\mathbf{r}} + \dot{\mathbf{s}}_p + \boldsymbol{\omega} \times \mathbf{s}_p \tag{2.10.2}$$

and

$$\ddot{\mathbf{r}}_p = \ddot{\mathbf{r}} + \ddot{\mathbf{s}}_p + 2\boldsymbol{\omega} \times \dot{\mathbf{s}}_p + \dot{\boldsymbol{\omega}} \times \mathbf{s}_p + \boldsymbol{\omega} \times \boldsymbol{\omega} \times \mathbf{s}_p \tag{2.10.3}$$

Since there is no relative motion between the particle at point $p$ and $o$ for the rigid body, $\dot{\mathbf{s}}_p = 0$ and $\ddot{\mathbf{s}}_p = 0$. The final velocity and acceleration of $\mathbf{r}_p$ can be expressed as

$$\dot{\mathbf{r}}_p = \dot{\mathbf{r}}^T\mathbf{e} + \mathbf{l}^T\dot{\mathbf{b}} = \dot{\mathbf{r}}^T\mathbf{e} + \mathbf{l}^T\tilde{\boldsymbol{\omega}}^T\mathbf{b} \tag{2.10.4}$$

$$\ddot{\mathbf{r}}_p = \ddot{\mathbf{r}}^T\mathbf{e} + \mathbf{l}^T\dot{\tilde{\boldsymbol{\omega}}}^T\mathbf{b} + \mathbf{l}^T\tilde{\boldsymbol{\omega}}^T\tilde{\boldsymbol{\omega}}^T\mathbf{b} \tag{2.10.5}$$

Note that if points $o$ and $p$ coincide, which implies $\mathbf{l} = 0$, we can derive the equations of motion by separating the translational and rotational motion so that different numerical algorithms may be applied accordingly.

## 2.11 Concluding Remarks

Spatial kinematics relations needed to calculate quantities such as the position, velocity and acceleration vectors of particles and bodies have been reviewed. In discussing the motion of a particle as being attached to a rigid body, a frame of reference must be specified so that the dynamical equations of the body can be derived. In the present derivation, an inertial reference frame is used to locate the position of the center of mass of a body. A body-fixed reference frame is then employed to locate the position of a particle in the body. Such hybrid reference frames are chosen to decouple the translational and rotational equations so that an efficient numerical algorithm can be formulated as discussed in chapter 5.

Three representations of rotation have been studied. The advantages and disadvantages of these angular representations are discussed. *Euler* parameters have been chosen in the present derivation because they lead to simple algebraic equations that do not require the evaluation of trigonometric functions and they give a singularity-free representation of rotations. Other angular representations may require trigonometric functions and/or suffer from singularity for certain parameter values.

# CHAPTER III

# EQUATIONS OF MOTION FOR MBD SYSTEMS

## 3.1 Introduction

This chapter deals with the formulation of the equations of motion for MBD systems using variational methods. There are several advantages of employing variational methods in dynamics. First, the system of particles and rigid bodies is considered as a whole rather than being separated into individual components. Second, dynamic problems are formulated in terms of kinetic energy and work, both of which are scalar quantities. Third, constraint forces do no work. Fourth, the use of generalized coordinates makes the formulation versatile. In this regard, the reference frames and velocities reviewed in chapter 2 will be adopted to describe the configuration and motion of bodies in MBD systems.

As indicated in chapter 2, an inertial frame is used to described the translational motion whereas a body-fixed frame is used to described the rotational motion of bodies in the system. This frame decomposition causes the mass matrix to be decoupled into translational and rotational equations. This kinematic representation is introduced into *the principle of virtual work* to obtain dynamic relationships between the constraint forces and their kinematic constraint conditions and thus produce the governing equations of motion. When d'Alembert's principle is used in conjunction with *the principle of virtual work*, we extend *the principle of virtual work* to dynamic systems that are composed of an arbitrary number of rigid bodies with an arbitrary number of constraints that are used to restrict the motion of the bodies. The present formalism considers the motions of individual bodies as initially independent, and then applies restriction on those motion by the introduction of kinematic constraints. Such constraints are incorporated through the method of Lagrange multipliers. The resulting system of equations, which consists of second-order differential equations that introduce Lagrange multipliers as constraint forces as well as the algebraic constraint equations as constraint conditions, are known as differential-algebraic equations (DAEs). To cover further development in flexible MBD systems, the equations of motion that include elastic deformations, which have been derived by Downer [52], are given in time discrete form by taking the advantage of the previously chosen reference frames and formulation.

## 3.2 The Principle of Virtual Work

Since an MBD system involves a number of interconnected bodies, the study of its dynamics is simplified in many respects by considering the system as a whole rather than as a collection of components obeying Newton's laws of motion. This is accomplished, as noted previously, by basing the derivation on an overall scalar quantity: generalized work. Consequently, the principle of virtual work will be used to establish the system's equilibrium conditions. This principle may be stated as follows: *The work done by all the forces acting on a system in static equilibrium, during a virtual displacement compatible with the constraints of the system is equal to zero.* The mathematical expression is

$$\delta \mathbf{W} = \sum_{i=1}^{n} \mathbf{P}_i \cdot \delta \mathbf{r}_i = 0 \tag{3.2.1}$$

where $n$ denotes the total number of bodies, $\delta \mathbf{W}$ denotes the virtual work of the system, $\mathbf{P}_i$ denotes the resultant forces acting on each body, and $\delta\mathbf{r}_i$ denotes the virtual displacement of each body.

To interconnect and restrict the motion of bodies in the overall system, kinematic constraints are imposed. Two types of constraints must be distinguished:

(1) Holonomic: constraints that depend only on position. Such kinematic restrictions may be expressed as algebraic relations:

$$\mathbf{\Phi}_h(\mathbf{r}_p, t) = 0 \tag{3.2.2}$$

The variation of the holonomic constraints is given by

$$\delta\mathbf{\Phi}_h = \frac{\partial\mathbf{\Phi}_h}{\partial\mathbf{r}_p}\delta\mathbf{r}_p = \mathbf{B}_h\delta\mathbf{r}_p = 0 \tag{3.2.3}$$

(2) Nonholonomic: constraints that depend both on position and velocities. Such kinematic restrictions are expressed as in differential relations:

$$\dot{\mathbf{\Phi}}_{nh}(\mathbf{r}_p, \dot{\mathbf{r}}_p, t) = \mathbf{B}_{nh}\dot{\mathbf{r}}_p = 0 \tag{3.2.4}$$

The variation of the nonholonomic constraints is given by

$$\delta\mathbf{\Phi}_{nh} = \mathbf{B}_{nh}\delta\mathbf{r}_p = 0 \tag{3.2.5}$$

where $h$ and $nh$ refer to holonomic and nonholonomic constraints. Hence when systems are subjected to constraints, one may separate the resultant forces $\mathbf{P}_i$ into applied forces $\mathbf{F}_i^a$ and constraint forces $\mathbf{F}_i^c$, so that

$$\mathbf{P}_i = \mathbf{F}_i^a + \mathbf{F}_i^c \tag{3.2.6}$$

Substituting (3.2.6) into (3.2.1) yields

$$\delta\mathbf{W} = \sum_{i=1}^{n}\mathbf{F}_i^a \cdot \delta\mathbf{r}_i + \sum_{i=1}^{n}\mathbf{F}_i^c \cdot \delta\mathbf{r}_i = 0 \tag{3.2.7}$$

Since the variations $\delta\mathbf{r}_i$ do not violate the prescribed constraint forces, the work performed by the constraint forces in any virtual displacement is equal to zero, therefore we conclude that

$$\sum_{i=1}^{n}\mathbf{F}_i^a \cdot \delta\mathbf{r}_i = 0 \tag{3.2.8}$$

Note that for systems with constraints, the virtual displacements $\delta\mathbf{r}_i$ are not all independent. Thus we cannot interpret (3.2.8) as $\mathbf{F}_i^a = 0$. For an unconstrained system, the principle of virtual work can be used to calculate the equilibrium position of the system as

$$\delta\mathbf{W} = \sum_{i=1}^{n}\mathbf{F}_i^a \cdot \delta\mathbf{r}_i = \delta\mathbf{V} = \sum_{i=1}^{n}(\frac{\partial\mathbf{V}}{\partial\mathbf{r}_i} \cdot \delta\mathbf{r}_i) = 0 \tag{3.2.9}$$

where $\mathbf{V}$ is the potential energy. Since by hypothesis the virtual displacements $\delta\mathbf{r}_i$ are all independent the static equilibrium conditions can be obtained as expected:

$$\mathbf{F}_i^a = \frac{\partial\mathbf{V}}{\partial\mathbf{r}_i} = 0 \tag{3.2.10}$$

If a system is subjected to holonomic constraints

$$\mathbf{\Phi}(\mathbf{r}) = 0 \tag{3.2.11}$$

the method of Lagrange multipliers is used to augment the potential energy. According to this method, we multiply each of the constraints (3.2.11) by an undetermined multiplier $\boldsymbol{\lambda}_j$, and add all resulting expressions to the potential energy $\mathbf{V}$ to get

$$\mathbf{V}^a = \mathbf{V} + \sum_{j=1}^{m}(\boldsymbol{\lambda}_j \cdot \mathbf{\Phi}_j) \tag{3.2.12}$$

where $\mathbf{V}^a$ is the augmented potential energy and $m$ denotes the total number of the constraint equations. The variation of the augmented potential energy subjected to the constraint condition (3.2.11) can be written

$$\delta \mathbf{V}^a = \sum_{i=1}^{n} (\frac{\partial \mathbf{V}}{\partial \mathbf{r}_i} \cdot \delta \mathbf{r}_i) + \sum_{j=1}^{m} (\boldsymbol{\lambda}_j \cdot \delta \boldsymbol{\Phi}_j) = 0 \tag{3.2.13}$$

Substituting (3.2.3) into (3.2.13) yields

$$(\frac{\partial \mathbf{V}}{\partial \mathbf{r}_i} + \sum_{j=1}^{m} \boldsymbol{\lambda}_j \cdot \frac{\partial \boldsymbol{\Phi}_j}{\partial \mathbf{r}_i}) \cdot \delta \mathbf{r}_i = 0, \quad i = 1, ..., n \tag{3.2.14}$$

Note that the virtual displacements $\delta \mathbf{r}_i$ in (3.2.14) are still not independent, but the $m$ values of $\boldsymbol{\lambda}_j$ can be chosen so that

$$\frac{\partial \mathbf{V}}{\partial \mathbf{r}_i} + \sum_{j=1}^{m} \boldsymbol{\lambda}_j \cdot \frac{\partial \boldsymbol{\Phi}_j}{\partial \mathbf{r}_i} = 0, \quad i = n - m + 1, n - m + 2, ..., n \tag{3.2.15}$$

whereas the remaining $n - m$ virtual displacements $\delta \mathbf{r}_i (i = 1, ..., n - m)$ can be treated as independent variables so that

$$\frac{\partial \mathbf{V}}{\partial \mathbf{r}_i} + \sum_{j=1}^{m} \boldsymbol{\lambda}_j \cdot \frac{\partial \boldsymbol{\Phi}_j}{\partial \mathbf{r}_i} = 0, \quad i = 1, ..., n - m \tag{3.2.16}$$

From (3.2.15) and (3.2.16), we obtain the following equilibrium conditions

$$\frac{\partial \mathbf{V}}{\partial \mathbf{r}_i} + \sum_{j=1}^{m} \boldsymbol{\lambda}_j \cdot \frac{\partial \boldsymbol{\Phi}_j}{\partial \mathbf{r}_i} = 0, \quad i = 1, ..., n \tag{3.2.17}$$

This procedure enables us to treat all the virtual displacements as independent variables by expanding from $n - m$ unknowns to $n + m$ unknowns with $n$ values of $\mathbf{r}_i$ and $m$ values of $\boldsymbol{\lambda}_j$. Now, recalling (3.2.10) and comparing the expression of (3.2.7) and (3.2.17), we arrive at the conclusion that the system equilibrium conditions are enforced by the presence of the constraint forces

$$\mathbf{F}_i^c = \sum_{j=1}^{n} \boldsymbol{\lambda}_j \cdot \frac{\partial \boldsymbol{\Phi}}{\partial \mathbf{r}_i}, \quad i = 1, ..., n \tag{3.2.18}$$

This important result provides the relationship between the constraint forces and the kinematic conditions of the bodies in the system. The principle of virtual work was originally stated for a system in static equilibrium. Nevertheless, the principle can be applied to dynamic systems by simple recourse to d'Alembert's principle, which gives the dynamic equilibrium by including the inertial forces of the system with constraints.

## 3.3 D'Alembert's Principle

*D'Alembert's principle states that the law of state equilibrium applies to a dynamic system if the inertial forces as well as the external and constraint forces are considered as applied forces acting on the system.* Thus, for a body with density $\rho$, the dynamic equilibrium condition is given by

$$\mathbf{F}^i - \mathbf{F}^a - \mathbf{F}^c = 0 \tag{3.3.1}$$

where $\mathbf{F}^i = \rho\ddot{\mathbf{r}}_p$ are the inertia forces, and $\ddot{\mathbf{r}}_p$ are the acceleration vectors. If we apply d'Alembert's principle in conjunction with the principle of virtual work, the principle of virtual work is extended by writting the following equation:

$$\int_V \delta\mathbf{r}_p \cdot (\rho\ddot{\mathbf{r}}_p - \mathbf{F}^a - \mathbf{F}^c)dV = 0 \tag{3.3.2}$$

Where $\mathbf{F}^a$ may be considered to include many types of force acting on the body: viscous forces which resist velocities; spring forces which restore position equilibrium; and independent defined external forces. We shall refer to (3.3.2), which includes both the principle of virtual work and d'Alembert's principle, as *d'Alembert's principle of virtual work.* In present chapter, we use this formulation to derive the equations of motion for MBD systems.

## 3.4 Governing Equations of Motion

To derive the equations of motion for MBD systems, we start with an unconstrained rigid body by using (3.3.2) where $\mathbf{F}^c = 0$. There are many possibilities of choosing $\delta \mathbf{r}_p$ depending upon the coordinates one has employed. In present derivations, we adopt the velocity and acceleration vectors of point $p$ derived in (2.10.4), and (2.10.5). The virtual displacement $\delta \mathbf{r}_p$ can be obtained as

$$\delta \mathbf{r}_p = \delta \mathbf{r}^T \mathbf{e} + \mathbf{l}^T \delta \mathbf{b} = \delta \mathbf{r}^T \mathbf{e} + \mathbf{l}^T \delta \tilde{\alpha}^T \mathbf{b} \tag{3.4.1}$$

and the virtual rotational tensor $\delta \tilde{\alpha}$ is

$$\delta \tilde{\alpha} = -\delta \mathbf{R} \mathbf{R}^T \tag{3.4.2}$$

Substituting these two equations into (3.3.2) yields

$$\int_V (\delta \mathbf{r}^T \mathbf{e} + \delta \boldsymbol{\alpha}^T \tilde{l} \mathbf{b}) \cdot [\rho(\ddot{\mathbf{r}}^T \mathbf{e} + \mathbf{l}^T \dot{\tilde{\omega}}^T \mathbf{b} + \mathbf{l}^T \tilde{\omega}^T \tilde{\omega}^T \mathbf{b}) - \mathbf{f}^T \mathbf{b}] dV =$$

$$\delta \mathbf{r}^T [\mathbf{M}(\ddot{\mathbf{r}} + \mathbf{R}^T \tilde{\mathbf{r}}_c^T \dot{\omega} + \mathbf{R}^T \tilde{\omega} \tilde{\mathbf{r}}_c^T \omega) - \mathbf{F}] + \delta \boldsymbol{\alpha}^T [\mathbf{M} \tilde{\mathbf{r}}_c^T \mathbf{R} \ddot{\mathbf{r}} + \mathbf{J} \dot{\omega} + \tilde{\omega} \mathbf{J} \omega - \mathbf{M}_o] = 0 \tag{3.4.3}$$

where

$$\begin{aligned} &\mathbf{M} = \int_V \rho dV, \quad \mathbf{J} = \int_V \rho \tilde{\mathbf{l}} \tilde{\mathbf{l}}^T dV, \quad \mathbf{M} \tilde{\mathbf{r}}_c = \int_V \rho \tilde{\mathbf{l}} dV \\ &\mathbf{F} = \int_V \mathbf{R}^T \mathbf{f} dV, \quad \mathbf{M}_o = \int_V \tilde{\mathbf{l}} \mathbf{f} dV \end{aligned} \tag{3.4.4}$$

Performing the variation of $\delta \mathbf{r}$ and $\delta \boldsymbol{\alpha}$ independently, the equations of motion for a unconstrained rigid body can be written in the following forms

$$\mathbf{M}(\ddot{\mathbf{r}} + \mathbf{R}^T \tilde{\mathbf{r}}_c^T \dot{\omega} + \mathbf{R}^T \tilde{\omega} \tilde{\mathbf{r}}_c^T \omega) - \mathbf{F} = 0 \tag{3.4.5}$$

$$\mathbf{M} \tilde{\mathbf{r}}_c^T \mathbf{R} \ddot{\mathbf{r}} + \mathbf{J} \dot{\omega} + \tilde{\omega} J \omega - \mathbf{M}_o = 0 \tag{3.4.6}$$

A considerable simplification can be made in the equations of motion if the body-fixed coordinates are chosen such that the principle axes coincide with the center of mass. With this choice, all products of inertia vanish since

$$\tilde{\mathbf{r}}_c = \tilde{0} \tag{3.4.7}$$

and (3.4.5) and (3.4.6) reduce to

$$\mathbf{M}\ddot{\mathbf{r}} = \mathbf{F} \tag{3.4.8}$$

$$\mathbf{J}\dot{\omega} + \tilde{\omega}\mathbf{J}\omega = \mathbf{M}_o \tag{3.4.9}$$

Note that the translational and rotational mass matrices can be expressed as follows:

$$\mathbf{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \tag{3.4.10}$$

$$\mathbf{J} = \begin{bmatrix} J_{1,1} & J_{1,2} & J_{1,3} \\ J_{2,1} & J_{2,2} & J_{2,3} \\ J_{3,1} & J_{3,2} & J_{3,3} \end{bmatrix} \tag{3.4.11}$$

where **M** and **J** denote the mass and moment of inertia of the body. Equations (3.4.8) and (3.4.9) are known as Newton-Euler's equations of motion. Euler equations (3.4.9) are widely used in solving for the rotational motion of a rigid body. Note that, however, they are in general nonlinear and it may be difficult to solve analytically for angular velocity $\omega$ as a function of time. Furthermore, the time integral of $\omega$ does not correspond to any physical rotational representation that can be used to describe the orientation of the body. So if one wishes to find the angular orientation of a body, a set of parameters must be chosen in order to find the relation between the parameters that orient the bodies and their corresponding angular velocity.

For MBD systems, the constraint conditions are introduced into d'Alembert's principle of virtual work via Lagrange multipliers to restrict the motion of the bodies in the system. From a formulation point of view, there are several ways to impose the kinematic relationships between bodies during the motion. In the present derivation, we use the description of the unconstrained motion to describe each of the bodies separately. Therefore, the virtual displacement of (3.4.1) is not a kinematically admissible one for the constrained systems. The method of Lagrange multipliers must then be introduced to incorporate the constraint conditions into d'Alembert's principle of virtual work as has been indicated in the previous sections. To apply this method the constraints are multiplied by undetermined Lagrange multipliers $\boldsymbol{\lambda}$ and added to the virtual work of the unconstrained system:

$$\int_V [\delta\mathbf{r}_p \cdot (\rho\ddot{\mathbf{r}}_p - \mathbf{f}) + \delta\boldsymbol{\Phi} \cdot \boldsymbol{\lambda}] dV = 0 \tag{3.4.12}$$

where $\delta\mathbf{r}_p, \rho, \ddot{\mathbf{r}}_p, \mathbf{f}$, and $dV$ are defined in the previous derivations, $\boldsymbol{\lambda}$ are the Lagrange multipliers and $\delta\boldsymbol{\Phi}$ are the variations of the constraint equations. The augmented terms represent the work of the constraint forces, providing the reaction forces which are exerted on account of given kinematical constraints.

Substituting (3.2.3) and (3.2.5) into (3.4.12) yields

$$\begin{aligned} &\int_V [\delta\mathbf{r}_p \cdot (\rho\ddot{\mathbf{r}}_p - \mathbf{f}) + \delta\boldsymbol{\Phi}_h \cdot \boldsymbol{\lambda}_h + \delta\boldsymbol{\Phi}_{nh} \cdot \boldsymbol{\lambda}_{nh}] dV = \\ &\int_V \delta\mathbf{r}_p \cdot (\rho\ddot{\mathbf{r}}_p - \mathbf{f} + \mathbf{B}_h^T\boldsymbol{\lambda}_h + \mathbf{B}_{nh}^T\boldsymbol{\lambda}_{nh}) dV = 0 \end{aligned} \tag{3.4.13}$$

Performing the variation of $\delta\mathbf{r}$ and $\delta\boldsymbol{\alpha}$ independently, the equations of motion for MBD systems can be derived from (3.4.13) in the following matrix form

$$\begin{bmatrix} \mathbf{M} & 0 \\ 0 & \mathbf{J} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{r}} \\ \dot{\omega} \end{Bmatrix} + \mathbf{B}_h^T \boldsymbol{\lambda}_h + \mathbf{B}_{nh}^T \boldsymbol{\lambda}_{nh} = \begin{Bmatrix} \mathbf{F} \\ \mathbf{M}_o - \tilde{\omega}\mathbf{J}\omega \end{Bmatrix} \tag{3.4.14}$$

Augmenting (3.4.14) with the constraint equations (3.2.2) and (3.2.4), the differential-algebraic equations result:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{B}^T \boldsymbol{\lambda} = \mathbf{F} \tag{3.4.15}$$

that are subjected to satisfy holonomic constraints

$$\boldsymbol{\Phi}_h(\mathbf{u}, t) = 0 \tag{3.4.16}$$

and nonholonomic constraints

$$\dot{\boldsymbol{\Phi}}_{nh}(\dot{\mathbf{u}}, \mathbf{u}, t) = \mathbf{B}_{nh}\dot{\mathbf{u}} = 0 \tag{3.4.17}$$

where $\ddot{\mathbf{u}} = [\ddot{\mathbf{r}}, \dot{\omega}]^T$, $\mathbf{B}$ is the gradient of the holonomic and nonholonomic constraints (or constraint Jacobian matrix), $\boldsymbol{\lambda}$ is its corresponding constraint forces, $\mathbf{F}$ is the forces that include external forces and inertia forces due to centrifugal acceleration, and $\mathbf{u}$ is the generalized displacement vector. The mass matrix for j-th body is given by combining (3.4.10) and (3.4.11) as

$$M^j = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & J_{1,1} & J_{1,2} & J_{1,3} \\ 0 & 0 & 0 & J_{2,1} & J_{2,2} & J_{2,3} \\ 0 & 0 & 0 & J_{3,1} & J_{3,2} & J_{3,3} \end{bmatrix}^j \tag{3.4.18}$$

and the force vector for j-th body is

$$F^j = \begin{Bmatrix} f \\ M_o - \tilde{\omega}J\omega \end{Bmatrix}^j \tag{3.4.19}$$

In the present derivation we have replaced a system having $n - m$ unknowns by one with $n + m$ unknowns, considering the Lagrange multipliers $\lambda$ as additional variables where $n$ is the total number of degrees of freedom before imposing constraints and $m$ is the number of constraint equations. The advantages of the present derivation are: first, the mass matrix as shown in (3.4.18) can be partitioned into translational and rotational sets of equations, which later will lead to a convenient computational algorithm treats the rotational equations and the translational equations with different procedures. Second, the method of Lagrange multipliers preserves the symmetry of the resulting equations for all coordinates without distinguishing between dependent and independent variables. Third, the constraint Jacobian matrix that defines the kinematic relationships between interconnected bodies can be generated by using a set of stand-alone joint modules. Fourth, the presence of closed loops in the system topologies, require no special treatment so that preprocessing to identify independent variables can be avoided.

The velocity and acceleration equations for holonomic constraints are given by

$$\dot{\mathbf{\Phi}}_h = \mathbf{B}_h \dot{\mathbf{u}} + \mathbf{\Phi}_t \tag{3.4.20}$$

$$\ddot{\mathbf{\Phi}}_h = \mathbf{B}_h \ddot{\mathbf{u}} + \dot{\mathbf{B}}_h \dot{\mathbf{u}} + 2\mathbf{\Phi}_{ut} \dot{\mathbf{u}} + \mathbf{\Phi}_{tt} \tag{3.4.21}$$

The acceleration equation for nonholonomic constraints is given by

$$\ddot{\mathbf{\Phi}}_{nh} = \mathbf{B}_{nh} \ddot{\mathbf{u}} + \dot{\mathbf{B}}_{nh} \dot{\mathbf{u}} + 2\mathbf{\Phi}_{ut} \dot{\mathbf{u}} + \mathbf{\Phi}_{tt} \tag{3.4.22}$$

Regardless of the nature of the constraints, the equations of motion with the constraint acceleration equation can be augmented into the following matrix form:

$$\begin{bmatrix} \mathbf{M} & \mathbf{B}^T \\ \mathbf{B} & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} \mathbf{F} \\ \mathbf{c} \end{Bmatrix} \tag{3.4.23}$$

where $\mathbf{c} = -(\dot{\mathbf{B}}\dot{\mathbf{u}} + 2\mathbf{\Phi}_{ut}\dot{\mathbf{u}} + \mathbf{\Phi}_{tt})$. Since the left hand side of (3.4.23) is symmetric and sparse, several research groups have developed solution procedures tailored to solve these constraint-augmented equations. These solution procedures will be discussed in chapter 5.

## 3.5 Interaction Equations for Rigid and Flexible Bodies

Up to now, the bodies that comprise the MBD system have been rigid. This assumption does not hold when the bodies in the mechanical system are subject to elastic deformation that must be taken into account. The formulation presented in this section has been motivated by further developments in analysis and design of large-scale systems that consist of interconnected rigid and flexible bodies, all of which may undergo large angular rotations as well as deformation. As discussed in previous sections, the bodies (rigid or flexible) in MBD systems are treated initially independent of each other. Kinematic relationships between adjacent rigid or flexible bodies are specified through a set of nonlinear algebraic constraint equations that depend on the position and time.

The purpose of this section is to impose these kinematic relationships into rigid or flexible bodies of multibody systems so that their equations of motion, which can be written in time discrete form, can be obtained. Since that the formulation of flexible body dynamics is well documented, e.g., in Downer *et al.* [52], only the body interfacing requirements will be outlined. There are essentially two different connection cases to be considered in flexible MBD systems: first, two flexible bodies are connected by a specific joint; second, a flexible body connects a rigid body with a given kinematic relationship. These approaches are illustrated in the following sections as initial development for the two-stage staggered explicit-implicit algorithm, discussed in chapter 5, which is used to numerically integrate these sets of nonlinear equations.

### 3.5.1 The Equations of Motion: Interaction of Flexible Bodies

The discrete equations of motion for this approach can be expressed (Downer, Park, and Chiou [52]) as illustrated in Fig. 3.1 where

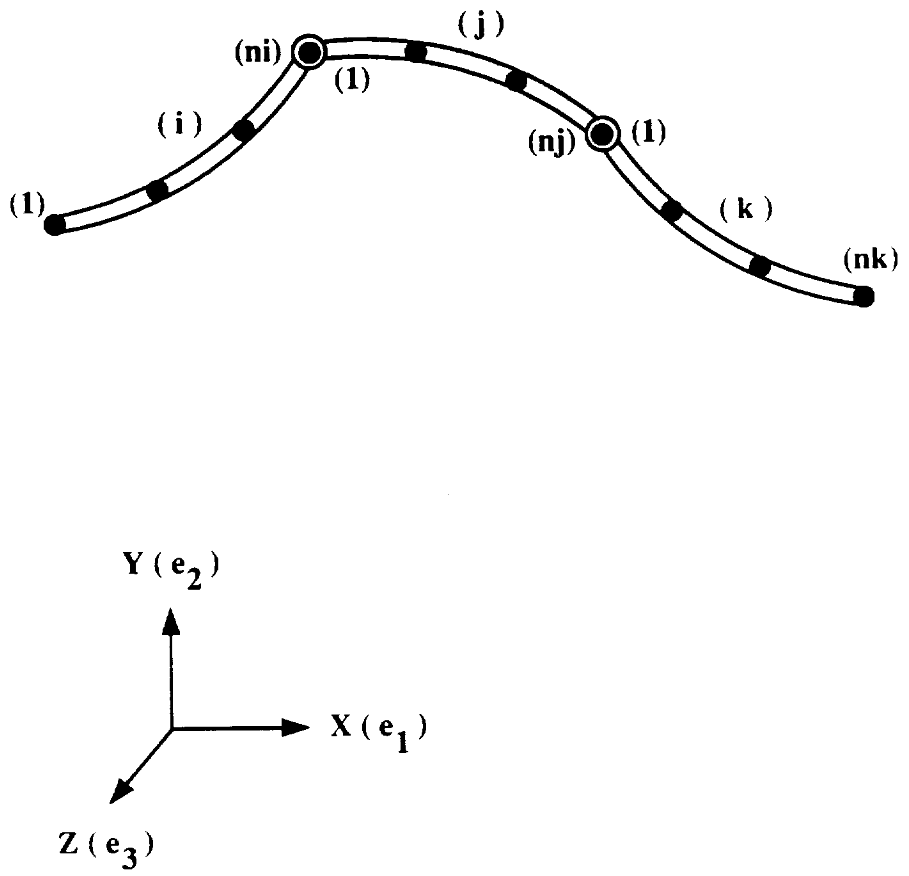


Fig. 3.1 Interaction of Flexible Bodies

$$\mathbf{M\ddot{u}} + \mathbf{D}(\dot{\mathbf{u}}) + \mathbf{S}(\mathbf{u}) + \mathbf{B}_h^T \boldsymbol{\lambda}_h + \mathbf{B}_{nh}^T \boldsymbol{\lambda}_{nh} = \mathbf{F} \tag{3.5.11}$$

or

$$\mathbf{M\ddot{u}} + \mathbf{B}_h^T \boldsymbol{\lambda}_h + \mathbf{B}_{nh}^T \boldsymbol{\lambda}_{nh} = \mathbf{Q} \tag{3.5.12}$$

subject to the constraint equations,

$$\boldsymbol{\Phi}_h(\mathbf{u}, t) = 0 \;\; ; \;\; \dot{\boldsymbol{\Phi}}_{nh}(\dot{\mathbf{u}}, \mathbf{u}, t) = \mathbf{B}_{nh}\dot{\mathbf{u}} = 0 \tag{3.5.13}$$

where

$$\mathbf{M} = \begin{bmatrix} M_i & 0 & 0 \\ 0 & M_j & 0 \\ 0 & 0 & M_k \end{bmatrix} \tag{3.5.14}$$

and $M_{nb} = diag[M_{(nb,1)}, ..., M_{(nb,nd)}]$

$$\ddot{\mathbf{u}} = [\ddot{u}_{(i,1)}, ..., \ddot{u}_{(i,ni)}, \ddot{u}_{(j,1)}, ..., \ddot{u}_{(j,nj)}, \ddot{u}_{(k,1)}, ..., \ddot{u}_{(k,nk)}]^T \tag{3.5.15}$$

$$\mathbf{B}_h = \begin{bmatrix} B_{(i,ni)} & B_{(j,1)} & 0 & 0 \\ 0 & 0 & B_{(j,nj)} & B_{(k,1)} \end{bmatrix} \tag{3.5.16}$$

$$\mathbf{Q} = \begin{Bmatrix} F_{(i,1)} - S_{(i,1)} - D_{(i,1)} \\ \cdots \\ F_{(i,ni)} - S_{(i,ni)} - D_{(i,ni)} \\ F_{(j,1)} - S_{(j,1)} - D_{(j,1)} \\ \cdots \\ F_{(j,nj)} - S_{(j,nj)} - D_{(j,nj)} \\ F_{(k,1)} - S_{(k,1)} - D_{(k,1)} \\ \cdots \\ F_{(k,nk)} - S_{(k,nk)} - D_{(k,nk)} \end{Bmatrix} \tag{3.5.17}$$

In the above equations $ni$, $nj$ and $nk$ are the total number of discrete nodal points, subscript $(nb, nd)$ denotes the $nd$-th node of the $nb$-th flexible body, $M$ is the mass matrix of $i$-th, $j$-th, and $k$-th bodies, $diag$ are the diagonal block matrices of each individual body, $\mathbf{D}(\cdot)$ is the generalized velocity-dependent force, $\mathbf{S}(\cdot)$ is the internal force operator due to member flexibility,

$\mathbf{B}_h$ is the gradient of the holonomic constraints that connect the nodes with prescribed joints, $\mathbf{B}_{nh}$ is the nonholonomic constraint Jacobian matrix, $\boldsymbol{\lambda}_h$ are the holonomic constraint forces, $\boldsymbol{\lambda}_{nh}$ are the nonholonomic constraint forces, $\mathbf{F}$ are external forces, and $\mathbf{u}$ is the generalized displacement vector.

In the present time discrete form, the flexible bodies are initially treated as independent of each other. Their kinematic relationships are then imposed by given specific constraint conditions at certain nodal points. Thus, the Lagrange multipliers will only be computed via the quantities of these constraint nodal points. We further address this issue in chapter 5 where the two-stage staggered explicit-implicit algorithm is developed.

### 3.5.2 The Equations of Motion: Interaction of Flexible and Rigid Bodies

The major difference introduced by the presence of rigid bodies pertains to the construction of the constraint Jacobian matrix, which significantly affects the computation of the constraint forces. The discrete equations of motion for this case can be expressed as shown in Fig. 3.2 where

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}(\dot{\mathbf{u}}) + \mathbf{S}(\mathbf{u}) + \mathbf{B}_h^T\boldsymbol{\lambda}_h + \mathbf{B}_{nh}^T\boldsymbol{\lambda}_{nh} = \mathbf{F} \qquad (3.5.21)$$

or

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{B}_h^T\boldsymbol{\lambda}_h + \mathbf{B}_{nh}^T\boldsymbol{\lambda}_{nh} = \mathbf{Q} \qquad (3.5.22)$$

subject to the constraint equations,

$$\boldsymbol{\Phi}_h(\mathbf{u},t) = 0 \;\; ; \;\; \dot{\boldsymbol{\Phi}}_{nh}(\dot{\mathbf{u}},\mathbf{u},t) = \mathbf{B}_{nh}\dot{\mathbf{u}} = 0 \qquad (3.5.23)$$

where

$$\mathbf{M} = \begin{bmatrix} M_i & 0 & 0 \\ 0 & M_j & 0 \\ 0 & 0 & M_k \end{bmatrix} \qquad (3.5.24)$$

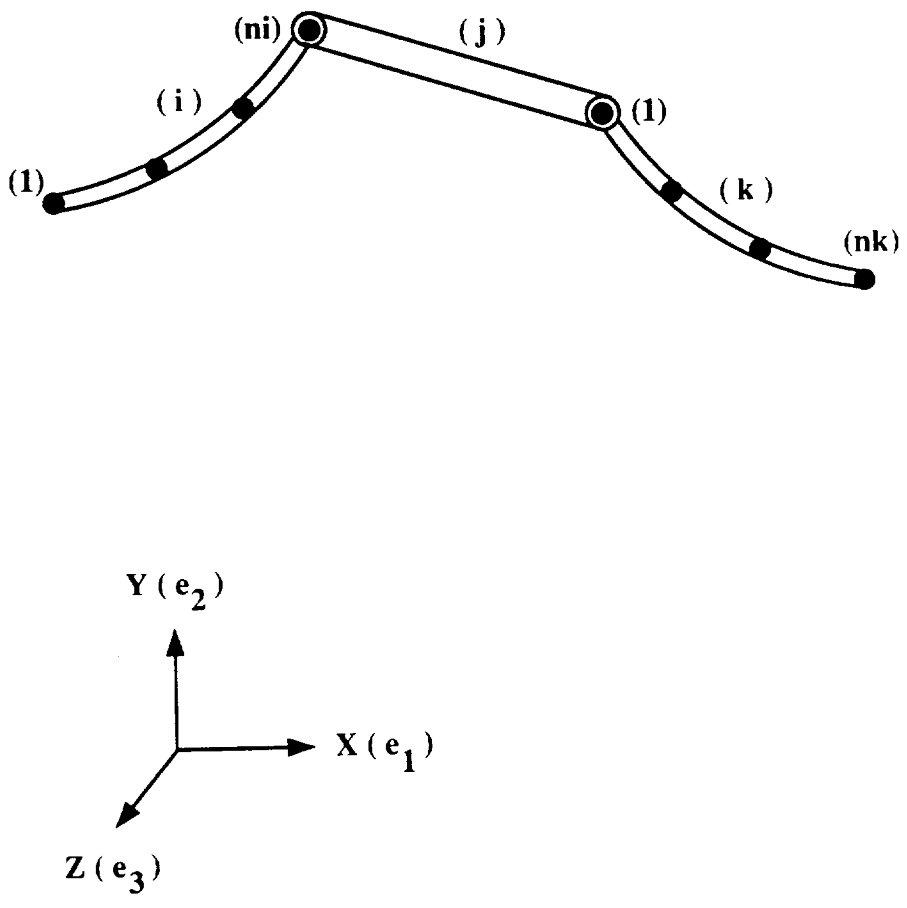


Fig. 3.2 Interaction of Flexible and Rigid Bodies

or

$$
\mathbf{M} = \begin{bmatrix}
M_{(i,1)} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \dots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & M_{(i,ni)} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & M_j & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & M_{(k,1)} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \dots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & M_{(k,1)}
\end{bmatrix} \qquad (3.5.25)
$$

$$\ddot{\mathbf{u}} = [\ddot{u}_{(i,1)}, ..., \ddot{u}_{(i,ni)}, \ddot{u}_j, \ddot{u}_{(k,1)}, ..., \ddot{u}_{(k,nk)}]^T \tag{3.5.26}$$

$$\mathbf{B} = \begin{bmatrix} B_{(i,ni)} & B_{(j,nl)} & 0 \\ 0 & B_{(j,nr)} & B_{(k,1)} \end{bmatrix} \tag{3.5.27}$$

$$\mathbf{Q} = \begin{Bmatrix} F_{(i,1)} - S_{(i,1)} - D_{(i,1)} \\ \cdots \\ F_{(i,ni)} - S_{(i,ni)} - D_{(i,ni)} \\ F_j \\ F_{(k,1)} - S_{(k,1)} - D_{(k,1)} \\ \cdots \\ F_{(k,nk)} - S_{(k,nk)} - D_{(k,nk)} \end{Bmatrix} \tag{3.5.28}$$

In these equations, $ni$ and $nk$ are the total number of discrete nodal points, subscript $(a, b)$ denotes the $b$-th node of the $a$-th flexible body, subscript $j$ denotes the j-th connected rigid body, $M$ consists of the mass matrix of flexible body $i$, $k$ and rigid body $j$, $\mathbf{D}(\cdot)$ is the generalized velocity-dependent force, $\mathbf{S}(\cdot)$ is the internal force operator due to member flexibility, $\mathbf{B}_h$ is the gradient of the holonomic constraints that connects the $ni$-th node of $i$-th flexible body to the left hand side of the $j$-th rigid body and the $nk$-th node of $k$-th flexible body to the right hand side of the $j$-th rigid body, $\mathbf{B}_{nh}$ is the nonholonomic constraint Jacobian matrix, $\lambda_h$ are the holonomic constraint forces, $\lambda_{nh}$ are the nonholonomic constraint forces, $F_{(a,b)}$ are the external forces, $F_k$ includes inertia forces due to centrifugal acceleration and external loads, and $\mathbf{u}$ is the generalized displacement vector.

## 3.6 Concluding Remarks

Two methodologies for deriving the equations of motion for systems with a number of bodies subject to kinematic constraint conditions may be distinguished. The first method makes explicit use of constraint conditions so that system dependent and independent variables can be identified and eliminated thus reducing the system equations to the number of independent variables. The second method introduces Lagrange multipliers in the equations of motion so that DAEs are obtained. The present research makes use of the latter method to generate the system dynamic equations in DAE form. Advantages gained by this choice are as follows. First as shown in (3.4.21), the symmetry of the DAEs for all coordinates is preserved which avoids having to distinguish dependent and independent system variables. Second, the constraint Jacobian matrix that establishes the kinematic relationships of contiguous bodies can be generated by using a set of stand-alone joint modules as presented in the next chapter. Third, the system topology whether open or closed-loop, require no special treatment, viz., preprocessing for a-priori identification of independent variables can be avoided.

The incorporation of flexible bodies, such as beams, in multibody systems is also discussed in this chapter. The computational issues regarding these discrete forms will be addressed during the development of the two-stage staggered explicit-implicit algorithm.

# CHAPTER IV

# KINEMATIC JOINTS AND FORCE ELEMENTS

## 4.1 Introduction

The equations of motion of MBD systems incorporating elastic deformations have been derived in the previous chapter. It is emphasized that the individual bodies are originally treated as independent of each other. Kinematic relations that link those bodies are established using constraint equations. In the previous chapter, however, the physical interpretation of the constraint Jacobian matrix has not been clearly defined. To complete the derivation of the equations of motion, the constraint Jacobian matrix pertaining to specific mechanical systems must be derived in detail to facilitate the development of a modular computer program.

A common feature in the construction of these constraint conditions is the use of *joints* to describe the interaction of contiguous bodies in MBD systems. *Joints* may range from rigid connectors, which allows no relative motion between two bodies, to devices that allows the relative separation of the bodies. Consequently, joint descriptions may involve from zero to six degrees of freedom. Two types of kinematic constraints, configuration-dependent (holonomic) constraints and velocity-dependent (nonholonomic) constraints, are used to describe joints. Spherical, universal, revolute, and translational joints provide examples of holonomic constraints. A rigid joint provides an example of nonholonomic constraint. In this chapter, the constraint equations pertaining to a spherical joint, universal joint, revolute joint, cylindrical joint, and a rigid joint are derived. As for other joints such as translational and skew joints, the kinematic principles discussed in this chapter can still be applied accordingly.

After the kinematic joints are derived, the force elements such as gravity, external forces, moments, actuators, dampers, and springs will be incorporated into DAEs in order to complete the assembly of a general-purpose computer program. Force elements are discussed after the treatment of mechanical joints because some of the constraint conditions used in kinematic joints can be applied to the force elements thus avoiding unnecessary derivations.

## 4.2 Spherical Joint

A spherical joint is characterized by imposing the equality of the positions of two connected bodies at a specific common location. This joint allows three relative rotational degrees of freedom during dynamic motion. Fig. 4.1 shows two adjacent bodies $i$ and $j$ connected by a spherical joint. The center of the spherical joint, called $p$, can be represented by the body-fixed coordinates $(b_1^i, b_2^i, b_3^i)$, and $(b_1^j, b_2^j, b_3^j)$ respectively. To restrict the relative motion of bodies $i$ and $j$, the algebraic constraint equations are expressed as

$$\mathbf{\Phi}_s = \mathbf{r}_i + \mathbf{s}_p^i - \mathbf{r}_j - \mathbf{s}_p^j = 0 \tag{4.2.1}$$

This relation can also be obtained by applying (2.10.1) as

$$\mathbf{e}^T \mathbf{\Phi}_s = \mathbf{r}_i^T \mathbf{e} + \mathbf{l}_{pi}^T \mathbf{b}_i - \mathbf{r}_j^T \mathbf{e} - \mathbf{l}_{pj}^T \mathbf{b}_j \tag{4.2.2}$$

Since $\mathbf{\Phi}_s$ is not function of time, if we differentiate $\mathbf{\Phi}_s$ once with respect to

time, the following relationship is established:

$$\dot{\mathbf{\Phi}}_s = \mathbf{B}_s \dot{\mathbf{u}} \tag{4.2.3}$$

or

$$\mathbf{e}^T \dot{\mathbf{\Phi}}_s = \mathbf{e}^T \mathbf{B}_s \dot{\mathbf{u}} \tag{4.2.4}$$

where $\mathbf{B}_s$ is the constraint Jacobian matrix and $\dot{\mathbf{u}}$ is the velocity vector containing the translational and rotational components of bodies $i$ and $j$, namely

$$\dot{\mathbf{u}} = [\ \dot{\mathbf{r}}_i\ ,\ \omega_i\ ,\ \dot{\mathbf{r}}_j\ ,\ \omega_j\ ]^T \tag{4.2.5}$$

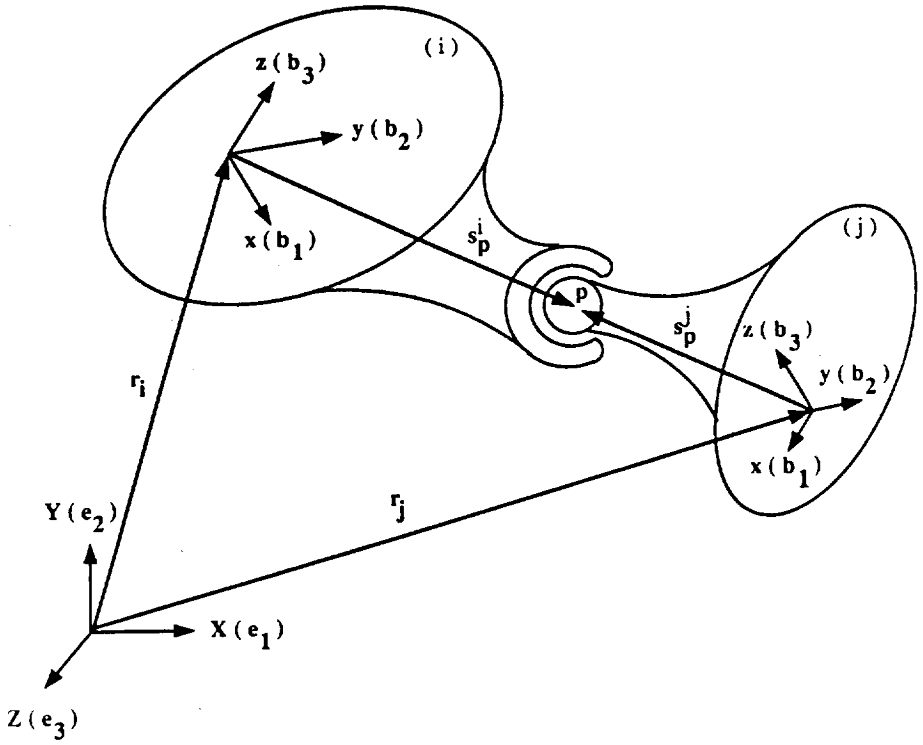


Fig. 4.1 A Spherical Joint

Equation (4.2.3) shows that if one wishes to obtain $\mathbf{B}_s$, we need to extract the velocity vector $\dot{\mathbf{u}}$ from the time derivative of the constraint equations and treat the remaining terms as the constraint Jacobian matrix. This can be demonstrated by the following algebraic calculations

$$\mathbf{e}^T\dot{\mathbf{\Phi}}_s = \dot{\mathbf{r}}_i^T\mathbf{e} + \mathbf{l}_{pi}^T\dot{\mathbf{b}}_i - \dot{\mathbf{r}}_j^T\mathbf{e} - \mathbf{l}_{pj}^T\dot{\mathbf{b}}_j \tag{4.2.6}$$

By substituting (2.7.2) into (4.2.6) and using the expression of (2.7.7), we obtain

$$\begin{aligned}\mathbf{e}^T\dot{\mathbf{\Phi}}_s &= (\dot{\mathbf{r}}_i^T + \mathbf{l}_{pi}^T\tilde{\omega}_i^T\mathbf{R}_i - \dot{\mathbf{r}}_j^T - \mathbf{l}_{pj}^T\tilde{\omega}_j^T\mathbf{R}_j)\mathbf{e} \\ &= \mathbf{e}^T(\dot{\mathbf{r}}_i + \mathbf{R}_i^T\tilde{\omega}_i\mathbf{l}_{pi} - \dot{\mathbf{r}}_j - \mathbf{R}_j^T\tilde{\omega}_j\mathbf{l}_{pj})\end{aligned} \tag{4.2.7}$$

The cross product of two vectors **a** and **b** is given by

$$\mathbf{a}\times\mathbf{b} = \tilde{\mathbf{a}}\mathbf{b} = -\tilde{\mathbf{b}}\mathbf{a} = -\mathbf{b}\times\mathbf{a} \tag{4.2.8}$$

Making the use of (4.2.8), (4.2.7) can be transformed to

$$\begin{aligned}\mathbf{e}^T\dot{\mathbf{\Phi}}_s &= \mathbf{e}^T(\dot{\mathbf{r}}_i + \mathbf{R}_i^T\tilde{\mathbf{l}}_{pi}^T\omega_i - \dot{\mathbf{r}}_j - \mathbf{R}_j^T\tilde{\mathbf{l}}_{pj}^T\omega_j) \\ &= \mathbf{e}^T[\mathbf{I}, (\tilde{\mathbf{l}}_{pi}^T\mathbf{R}_i)^T, -\mathbf{I}, (\tilde{\mathbf{l}}_{pj}^T\mathbf{R}_j)^T]\begin{bmatrix}\dot{\mathbf{r}}_i \\ \omega_i \\ \dot{\mathbf{r}}_j \\ \omega_j\end{bmatrix}\end{aligned} \tag{4.2.9}$$

Comparing the results of (4.2.4) and (4.2.9), we obtain the expression of the constraint Jacobian matrix $\mathbf{B}_s$ for a spherical joint where

$$\mathbf{B}_s = [\ \mathbf{I}\ ,\ (\tilde{\mathbf{l}}_{pi}\mathbf{R}_i)^T\ ,\ -\mathbf{I}\ ,\ -(\tilde{\mathbf{l}}_{pj}\mathbf{R}_j)^T\ ] \tag{4.2.10}$$

and **I** denotes the $(3\times 3)$ identity matrix. Similarly, the second time derivative of $\mathbf{\Phi}_s$ is given by

$$\begin{aligned}\ddot{\mathbf{\Phi}}_s &= \mathbf{B}_s\ddot{\mathbf{u}} + \dot{\mathbf{B}}_s\dot{\mathbf{u}} \\ &= \mathbf{B}_s\ddot{\mathbf{u}} + \dot{\mathbf{R}}_i^T\tilde{\mathbf{l}}_{pi}^T\omega_i - \dot{\mathbf{R}}_j^T\tilde{\mathbf{l}}_{pj}^T\omega_j \\ &= \mathbf{B}_s\ddot{\mathbf{u}} + \mathbf{R}_i^T\tilde{\omega}_i\tilde{\mathbf{l}}_{pi}^T\omega_i - \mathbf{R}_j^T\tilde{\omega}_j\tilde{\mathbf{l}}_{pj}^T\omega_j \\ &= \mathbf{B}_s\ddot{\mathbf{u}} + [0, \mathbf{R}_i^T\tilde{\omega}_i\tilde{\mathbf{l}}_{pi}^T, 0, \mathbf{R}_j^T\tilde{\omega}_j\tilde{\mathbf{l}}_{pj}^T]\begin{bmatrix}\dot{\mathbf{r}}_i \\ \omega_i \\ \dot{\mathbf{r}}_j \\ \omega_j\end{bmatrix}\end{aligned} \tag{4.2.11}$$

as

$$\dot{\mathbf{B}}_s = [\ 0\ ,\ -(\tilde{\mathbf{l}}_{pi}\tilde{\omega}_i\mathbf{R}_i)^T\ ,\ 0\ ,\ (\tilde{\mathbf{l}}_{pj}\tilde{\omega}_j\mathbf{R}_j)^T\ ] \tag{4.2.12}$$

and

$$\ddot{\mathbf{u}} = [\ \ddot{\mathbf{r}}_i\ ,\ \dot{\omega}_i\ ,\ \ddot{\mathbf{r}}_j\ ,\ \dot{\omega}_j\ ]^T \tag{4.2.13}$$

## 4.3 Universal Joint

The universal joint fixes two bodies at an arbitrary position in space and allows two relative rotational degrees of freedom during relative motion. Fig. 4.2 shows two bodies connected by a universal joint. The constraint equations for the universal joint can be expressed as if there were a spherical joint between connected bodies with two vectors $\mathbf{s}_i$ and $\mathbf{s}_j$ that are perpendicular. If two vectors remain perpendicular at all times their dot product vanishes. This kinematic relationship can be expressed as

$$\mathbf{\Phi}_{unv} = \left\{ \begin{array}{c} \mathbf{\Phi}_s \\ \mathbf{\Phi}_u = \mathbf{s}_i \cdot \mathbf{s}_j \end{array} \right\} = 0 \tag{4.3.1}$$

Since

$$\mathbf{s} = \underline{s}^T\mathbf{e} = \underline{l}^T\mathbf{b} = \underline{l}^T\mathbf{R}\mathbf{e} \tag{4.3.2}$$

we conclude that

$$\underline{s}^T = \underline{l}^T\mathbf{R} \quad ; \quad \underline{s} = \mathbf{R}^T\underline{l} \tag{4.3.3}$$

Replacement of $\underline{s}$ and $\underline{s}^T$ in (4.3.1.b) by (4.3.3) lead to

$$\mathbf{\Phi}_u = \underline{l}_i^T\mathbf{R}_i\mathbf{R}_j^T\underline{l}_j \tag{4.3.4}$$

Time differentiation of (4.3.4) yields

$$\dot{\mathbf{\Phi}}_u = \underline{l}_i^T\dot{\mathbf{R}}_i\mathbf{R}_j^T\underline{l}_j + \underline{l}_i^T\mathbf{R}_i\dot{\mathbf{R}}_j^T\underline{l}_j \tag{4.3.5}$$

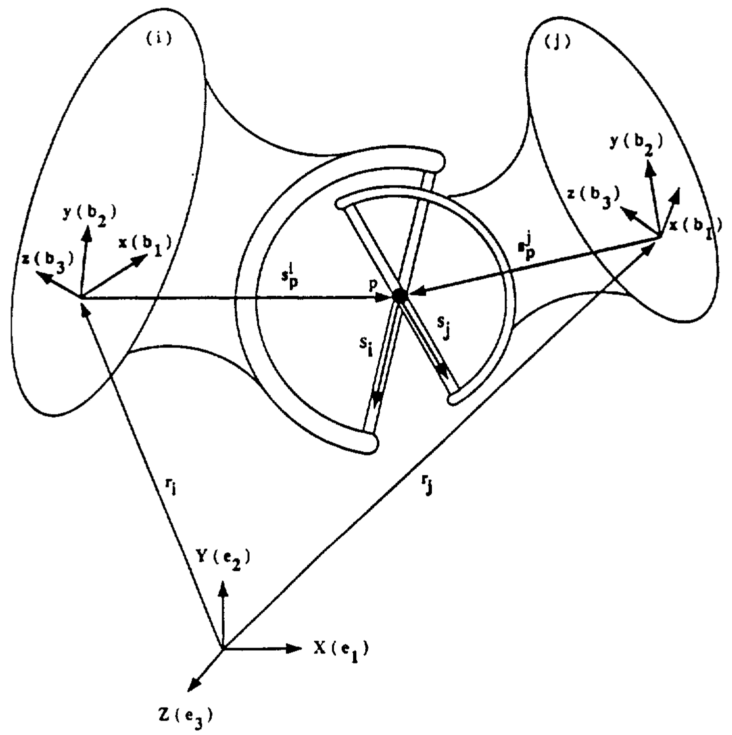


Fig. 4.2 An Universal Joint

If (2.7.6) is applied, (4.3.5) becomes

$$\dot{\Phi}_u = \underline{l}_i^T \tilde{\omega}_i^T \mathbf{R}_i \mathbf{R}_j^T \underline{l}_j + \underline{l}_i^T \mathbf{R}_i \mathbf{R}_j^T \tilde{\omega}_j \underline{l}_j \tag{4.3.6}$$

Since $\dot{\Phi}_u$ is a scalar, the transpose of the first term of (4.3.6) gives the same magnitude as

$$\dot{\Phi}_u = \underline{l}_j^T \mathbf{R}_j \mathbf{R}_i^T \tilde{\omega}_i \underline{l}_i + \underline{l}_i^T \mathbf{R}_i \mathbf{R}_j^T \tilde{\omega}_j \underline{l}_j \tag{4.3.7}$$

Applying (4.2.8), we obtain

$$\begin{aligned} \dot{\Phi}_u &= \underline{l}_j^T \mathbf{R}_j \mathbf{R}_i^T \tilde{\underline{l}}_i^T \omega_i + \underline{l}_i^T \mathbf{R}_i \mathbf{R}_j^T \tilde{\underline{l}}_j^T \omega_j \\ &= [\ 0\ ,\ \underline{l}_j^T \mathbf{R}_j \mathbf{R}_i^T \tilde{\underline{l}}_i^T\ ,\ 0\ ,\ \underline{l}_i^T \mathbf{R}_i \mathbf{R}_j^T \tilde{\underline{l}}_j^T\ ] \begin{bmatrix} \dot{\mathbf{r}}_i \\ \omega_i \\ \dot{\mathbf{r}}_j \\ \omega_j \end{bmatrix} \end{aligned} \tag{4.3.8}$$

and the constraint Jacobian matrix $\mathbf{B}_u$ can be expressed as

$$\mathbf{B}_u = [\, 0 \,,\, \underline{l}_j^T \mathbf{R}_j \mathbf{R}_i^T \tilde{\underline{l}}_i^T \,,\, 0 \,,\, \underline{l}_i^T \mathbf{R}_i \mathbf{R}_j^T \tilde{\underline{l}}_j^T \,] \tag{4.3.9}$$

Following the same procedure as in (4.2.11), the $\dot{\mathbf{B}}_u$ is found to be

$$\dot{\mathbf{B}}_u = [0,\, \underline{l}_j^T \tilde{\omega}_j^T \mathbf{R}_j \mathbf{R}_i^T \tilde{\underline{l}}_i^T + \underline{l}_j^T \mathbf{R}_j \mathbf{R}_i^T \tilde{\omega}_i \tilde{\underline{l}}_i^T \,,\, 0 \,,\, \underline{l}_i^T \tilde{\omega}_i^T \mathbf{R}_i \mathbf{R}_j^T \tilde{\underline{l}}_j^T + \underline{l}_i^T \mathbf{R}_i \mathbf{R}_j^T \tilde{\omega}_j \tilde{\underline{l}}_j^T] \tag{4.3.10}$$

Hence the gradient matrix of the constraint equations for the universal joint can be written as

$$\mathbf{B}_{unv} = \begin{bmatrix} \mathbf{B}_s \\ \mathbf{B}_u \end{bmatrix} \tag{4.3.11}$$

## 4.4 <u>Revolute Joint</u>

A revolute joint attaches two bodies in space and allow one rotational degree of freedom during actual motion. Fig. 4.3 shows two bodies connected by a revolute joint. The constraint equations for the revolute joint can be expressed as if there were a spherical joint connected two bodies with two vectors $\mathbf{s}_i$ and $\mathbf{s}_j$ that are always remained parallel to each other. Mathematically, their cross product is equal to zero. The constraint equations for the revolute joint can be expressed as

$$\mathbf{\Phi}_{rev} = \left\{ \begin{array}{c} \mathbf{\Phi}_s \\ \mathbf{\Phi}_{rv} = \mathbf{s}_i \times \mathbf{s}_j \end{array} \right\} = 0 \tag{4.4.1}$$

Equation (4.3.2) has concluded that $\underline{s} = \mathbf{R}^T \underline{l}$, time differentiation of $\underline{s}$ yields

$$\dot{\underline{s}} = \dot{\mathbf{R}}^T \underline{l} = \mathbf{R}^T \tilde{\omega} \underline{l} = -\mathbf{R}^T \tilde{\underline{l}} \omega \tag{4.4.2}$$

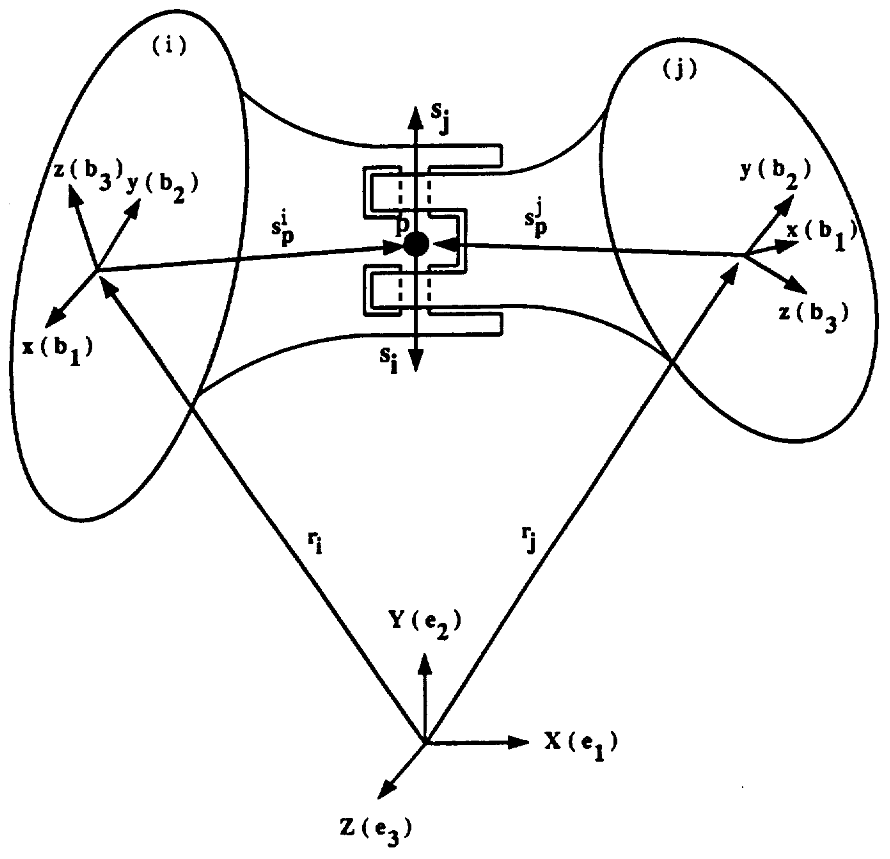


Fig. 4.3 A Revolute Joint

To obtain $\mathbf{B}_r$, the time differentiation of $\mathbf{\Phi}_{rv}$ is taken and given by

$$\dot{\mathbf{\Phi}}_{rv} = \dot{\mathbf{s}}_i \times \mathbf{s}_j + \mathbf{s}_i \times \dot{\mathbf{s}}_j = -\mathbf{s}_j \times \dot{\mathbf{s}}_i + \mathbf{s}_i \times \dot{\mathbf{s}}_j \tag{4.4.3}$$

Replacement of $\mathbf{s}$ and $\dot{\mathbf{s}}$ in (4.4.3) by (4.3.2) and (4.4.2) lead to

$$\begin{aligned} \dot{\mathbf{\Phi}}_{rv} &= \tilde{\mathbf{s}}_j \mathbf{R}_i^T \tilde{\underline{l}}_i \omega_i - \tilde{\mathbf{s}}_i \mathbf{R}_j^T \tilde{\underline{l}}_j \omega_j \\ &= [\ 0\ ,\ \tilde{\mathbf{s}}_j \mathbf{R}_i^T \tilde{\underline{l}}_i\ ,\ 0\ ,\ \tilde{\mathbf{s}}_i \mathbf{R}_j^T \tilde{\underline{l}}_j\ ] \begin{bmatrix} \dot{\mathbf{r}}_i \\ \omega_i \\ \dot{\mathbf{r}}_j \\ \omega_j \end{bmatrix} \end{aligned} \tag{4.4.4}$$

From (4.4.4), the constraint Jacobian matrix for $\mathbf{\Phi}_{rv}$ can be easily verified to be

$$\mathbf{B}_r = [\ 0\ ,\ \tilde{\mathbf{s}}_j \mathbf{R}_i^T \tilde{\underline{l}}_i\ ,\ 0\ ,\ -\tilde{\mathbf{s}}_i \mathbf{R}_j^T \tilde{\underline{l}}_j\ ] \tag{4.4.5}$$

The time derivative of (4.4.5) is found to be

$$\dot{\mathbf{B}}_r = [\ 0\ ,\ \dot{\tilde{\mathbf{s}}}_j \mathbf{R}_i^T \tilde{\underline{l}}_i + \tilde{\mathbf{s}}_j \mathbf{R}_i^T \tilde{\omega}_i \tilde{\underline{l}}_i\ ,\ 0\ ,\ -\dot{\tilde{\mathbf{s}}}_i \mathbf{R}_j^T \tilde{\underline{l}}_j - \tilde{\mathbf{s}}_i \mathbf{R}_j^T \tilde{\omega}_j \tilde{\underline{l}}_j\ ] \tag{4.4.6}$$

Hence the constraint Jacobian matrix for the revolute joint can be written as

$$\mathbf{B}_{rev} = \begin{bmatrix} \mathbf{B}_s \\ \mathbf{B}_r \end{bmatrix} \tag{4.4.7}$$

Notice that if two vectors are to remain parallel at all times, only two constraint equations are needed. Equation (4.4.5) yields three algebraic equations, of which only two equations are independent, i.e., one of the equations can be derived as linear combination of the remaining equations. A technique for selecting the proper set of equations from the overdetermined set is to compare the absolute values of each row equation of the gradient matrix of the constraint equations, and select the two equations that have the largest terms.

An alternative approach to modeling a revolute joint is to set up two proper vectors that are capable of representing their kinematical relationships as a revolute joint. This approach is followed below. Let $\mathbf{b}^i = [b_1, b_2, b_3]^i$ and $\mathbf{b}^j = [b_1, b_2, b_3]^j$ be two triad of orthogonal unit vectors attached to bodies $i$ and $j$ respectively (Fig. 4.4). The kinematic constraints for this revolute joint can be expressed as

$$\mathbf{\Phi}_r = \left\{ \begin{array}{c} \mathbf{\Phi}_s \\ \mathbf{\Phi}_{r1} = b_2^i \cdot b_1^j \\ \mathbf{\Phi}_{r2} = b_3^i \cdot b_1^j \end{array} \right\} = 0 \tag{4.4.8}$$

For any configuration, the constraint Jacobian matrix $\mathbf{B}_r$ of the revolute joint is derived and given by

$$\mathbf{B}_r = \begin{Bmatrix} c_2^i(\mathbf{R}^i)^T\mathbf{R}^j c_1^j \\ c_3^i(\mathbf{R}^i)^T\mathbf{R}^j c_1^j \end{Bmatrix} = 0 \tag{4.4.9}$$

where $c_1^j$, $c_2^i$, and $c_3^i$ are the components of $\mathbf{b}^i$ and $\mathbf{b}^j$.

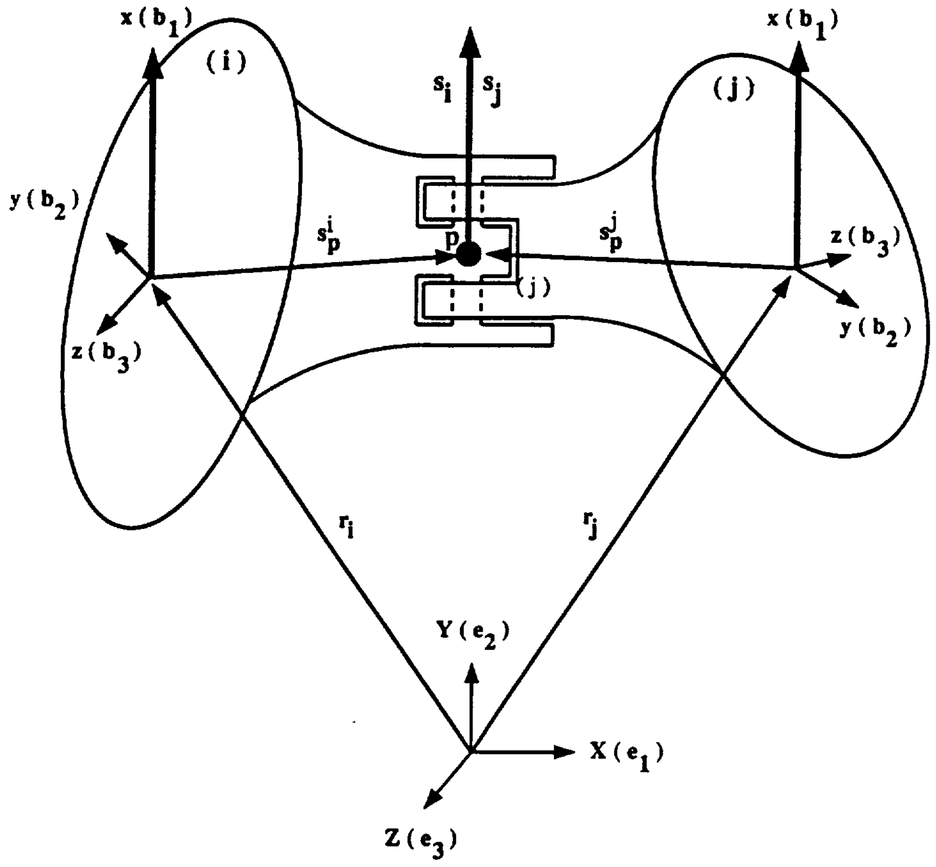


Fig. 4.4 A Modified Revolute Joint

## 4.5 Cylindrical Joint

A cylindrical joint provides one translational and rotational degree of freedom to two connected bodies. Fig. 4.5 shows the constraint conditions for a cylindrical joint. The constraint equations are derived from the condition that vector $\mathbf{u}_i$ must remain parallel to vectors $\mathbf{u}_j$ and $\mathbf{d}$:

$$\boldsymbol{\Phi}_{cyl} = \left\{ \begin{array}{l} \boldsymbol{\Phi}_{c1} = \mathbf{u_i} \times \mathbf{u_j} = \tilde{\mathbf{u}}_\mathbf{i}\mathbf{u_j} \\ \boldsymbol{\Phi}_{c2} = \mathbf{u_i} \times \mathbf{d} \ = \tilde{\mathbf{u}}_\mathbf{i}\mathbf{d} \end{array} \right\} = 0 \tag{4.5.1}$$

where $\mathbf{u}_i$ and $\mathbf{u}_j$ are given directional unit vectors based on their body-fixed frames so that the two bodies will slide according to that axis, and $\mathbf{d}$ is obtained from $\mathbf{d} = \mathbf{r_i} - \mathbf{r_j}$.

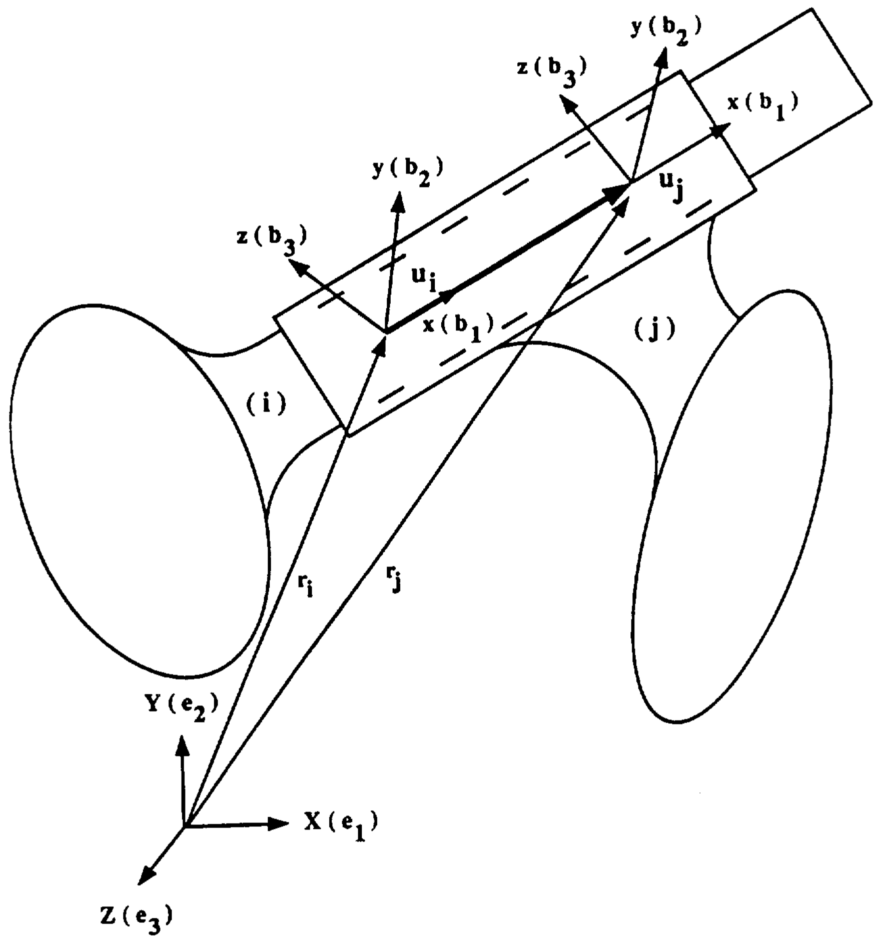


Fig. 4.5 A Cylindrical Joint

Since (4.5.1.a) has the same form as in (4.4.1.b), $\mathbf{B}_{c1}$ and $\dot{\mathbf{B}}_{c1}$ can be found in the same way as (4.4.5) and (4.4.6). If the first and second time derivatives of (4.5.1.b) are taken, $\mathbf{B}_{c2}$ and $\dot{\mathbf{B}}_{c2}$ can be obtained as

$$\mathbf{B}_{c2} = [\ \tilde{\mathbf{u}}_i\ ,\ \tilde{\mathbf{d}}\mathbf{R}_i^T \tilde{\underline{l}}_i\ ,\ -\tilde{\mathbf{u}}_i\ ,\ 0\ ] \tag{4.5.2}$$

$$\dot{\mathbf{B}}_{c2} = [\ \dot{\tilde{\mathbf{u}}}_i\ ,\ \dot{\tilde{\mathbf{d}}}\mathbf{R}_i^T \tilde{\underline{l}}_i + \tilde{\mathbf{d}}\mathbf{R}_i^T \tilde{\omega}_i \tilde{\underline{l}}_i\ ,\ -\dot{\tilde{\mathbf{u}}}_i\ ,\ 0\ ] \tag{4.5.3}$$

The final constraint equations for the cylindrical joint are:

$$\mathbf{B}_{cyl} = \begin{bmatrix} \mathbf{B}_{c1} \\ \mathbf{B}_{c2} \end{bmatrix} \tag{4.5.4}$$

## 4.6 Rigid Joint

A rigid joint by definition allows no relative motion between two bodies. Thus us a nonholonomic constraint that can be imposed as a spherical joint with no relative velocities among the connected bodies. The above statement can be expressed mathematically as following equations

$$\mathbf{\Phi}_{rigid} = \left\{ \begin{matrix} \mathbf{\Phi}_s \\ \mathbf{\Phi}_{rj} = \omega_i - \omega_j = \mathbf{B}_{rj}\dot{\mathbf{u}} \end{matrix} \right\} = 0 \tag{4.6.1}$$

The constraint Jacobian matrix of $\mathbf{\Phi}_{rj}$ is given by

$$\mathbf{B}_{rj} = [\ 0\ ,\ \mathbf{I}\ ,\ 0\ ,\ -\mathbf{I}\ ] \tag{4.6.2}$$

Hence the gradient matrix of the constraint equations for the rigid joint can be written as

$$\mathbf{B}_{rigid} = \begin{bmatrix} \mathbf{B}_s \\ \mathbf{B}_{rj} \end{bmatrix} \tag{4.6.3}$$

## 4.7 Force Elements in MBD Systems

In section 3.3, different types of forces acting on the bodies have been discussed. Forces that are commonly encountered in mechanical systems include gravitational forces, actuator forces, damping forces, spring forces, and external forces. In the present section, these forces will be formulated and incorporated into DAEs as

$$\mathbf{F} = \mathbf{F}_\omega + \mathbf{F}_g + \mathbf{F}_a + \mathbf{F}_d + \mathbf{F}_k + \mathbf{F}_f \tag{4.71}$$

where $\mathbf{F}_\omega$, $\mathbf{F}_g$, $\mathbf{F}_a$, $\mathbf{F}_d$, $\mathbf{F}_k$ and $\mathbf{F}_f$ denote centripetal, gravity, actuator, damping, spring and external forces, respectively. These force elements are analyzed in further detail below.

### 4.7.1 Gravitational Force

Since the acceleration of gravity is measured with respect to an inertial reference frame fixed in the earth, the gravitational force of a body with mass $m_g$ can be calculated by the equation

$$f_g = m_g g \tag{4.7.1.1}$$

where $f_g$ is the force created by the gravity and $g$ is the acceleration of gravity. If we choose a gravitational field acting on the negative $z$ direction of the inertial reference frame, the force $\mathbf{F}_g$ that contributes by this gravitational field on body $i$ is

$$F_g^{(i)} = [0, 0, -f_g^{(i)}, 0, 0, 0]^T \tag{4.7.1.2}$$

### 4.7.2 External Forces and Moments

Consider a force $f^{(i)}$ acting on body $i$ at point $p$ as shown in Fig. 4.6. The moment of $f^{(i)}$ about the origin of the body is

$$M_o^{(i)} = s^{(i)} \times f^{(i)} \tag{4.7.2.1}$$

where $s^{(i)} = l^T b^{(i)}$ is the position vector of point $p$ in the $i$-th body-fixed reference frame. The contribution of $f^{(i)}$ and $M_o^{(i)}$ to the force vector $\mathbf{F}_f$

on body $i$ is

$$\mathbf{F}_f^{(i)} = [f^{(i)}, M_o^{(i)}]^T \tag{4.7.2.2}$$

If a pure moment $\mathbf{m}_o^{(i)}$ acts on body $i$, then the force vector $\mathbf{F}_f$ on body $i$ becomes

$$\mathbf{F}_f^{(i)} = [0, \mathbf{m}_o^{(i)}]^T \tag{4.7.2.3}$$

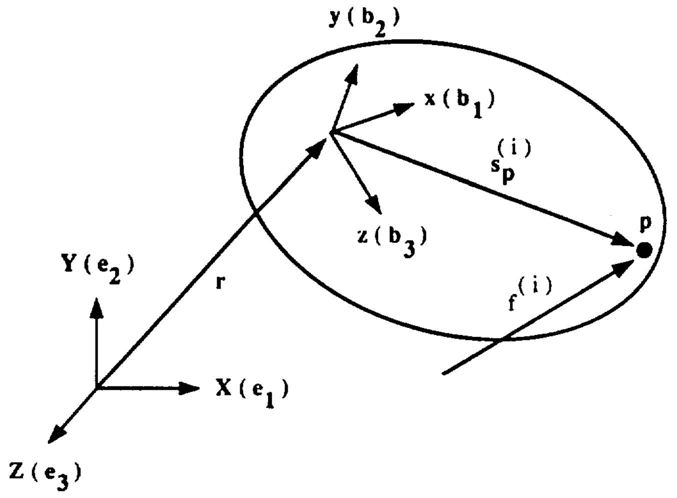


Fig. 4.6 A Point Force Acting on a Body

4.7.3 Actuator Forces

An actuator is a force element that provides a constant or a time-dependent pair of forces acting on two bodies in MBD systems. The direction of these forces is defined by the connecting points of bodies $i$ and $j$ (see Fig. 4.7) where the actuator is installed. A vector $l_{ij}$ connecting points $P_i$ and $P_j$ is defined as

$$l_{ij} = r_i^T e + l_i^T b_i - r_j^T e - l_j^T b_j \tag{4.7.3.1}$$

The actuator force $f_a$ acting on bodies $i$ and $j$ is given by

$$f_a^{(i)} = \pm f_a l_a \quad ; \quad f_a^{(j)} = \mp f_a l_a \tag{4.7.3.2}$$

where $l_a = \frac{l_{ij}}{|l_{ij}|}$. The $\pm$ sign constitutes a pull and a push forces that are given by the actuators. The contribution of $f_a^{(i)}$ and $f_a^{(j)}$ to the force vector $\mathbf{F}_a$ on bodies $i$ and $j$ are

$$\mathbf{F}_a^{(i,j)} = [f_a^{(i,j)}, s_{(i,j)}^T \times f_a^{(i,j)}]^T \tag{4.7.3.3}$$

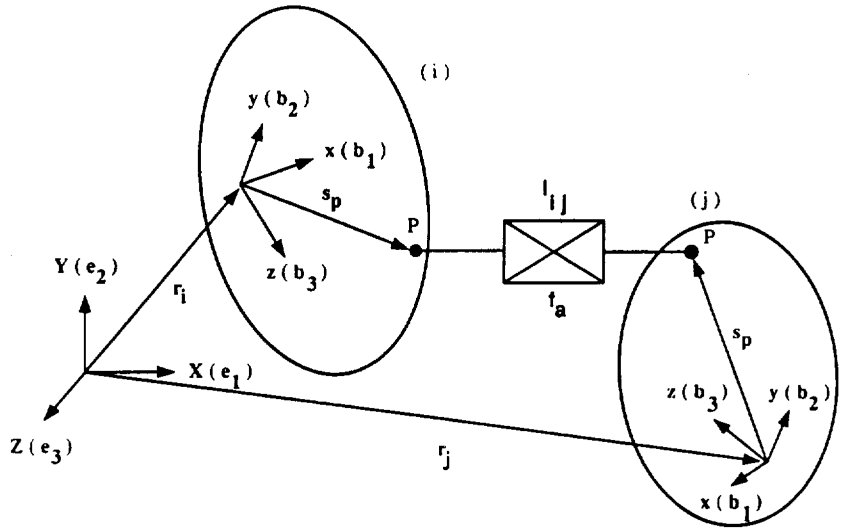


Fig. 4.7 An Actuator Acting on Two Bodies

### 4.7.4 Damping forces

Dampers dissipate relative body motions by converting mechanical energy to dissipated forms such as heat. The damping force between bodies $i$ and $j$ at point $Q_i$ and $Q_j$ (Fig. 4.8) is found to be

$$f_d = d\frac{l_{ij}^T \dot{l}_{ij}}{|l_{ij}^T l_{ij}|} \tag{4.7.4.1}$$

where $d$ is the damping coefficient and

$$\dot{l}_{ij} = \dot{r}_i^T e + l_i^T \dot{b}_i - \dot{r}_j^T e - l_j^T \dot{b}_j \tag{4.7.4.2}$$

The damping forces acting on bodies $i$ and $j$ are

$$f_d^{(i)} = \pm f_d l_d \quad ; \quad f_d^{(j)} = \mp f_d l_d \tag{4.7.4.3}$$

in which $l_d = \frac{l_{ij}}{|l_{ij}|}$. The contribution of $f_d^{(i)}$ and $f_d^{(j)}$ to the force vector $\mathbf{F}_d$ on bodies $i$ and $j$ can be found from (4.7.3.3).

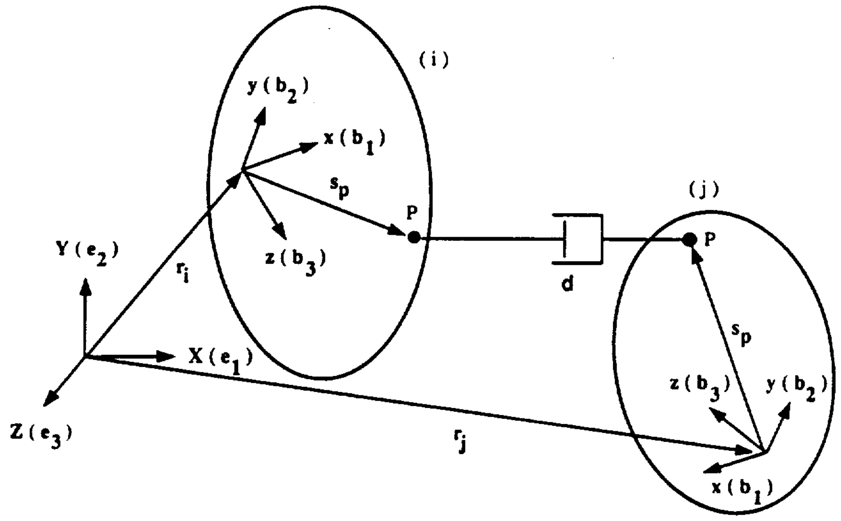


Fig. 4.8 A Damper Acting Between Two Bodies

## 4.7.5 Spring Forces

In mechanical systems, springs are often used to restore position equilibrium of two bodies. In Fig. 4.9, a spring is attached to two points $S_i$ and $S_j$ of bodies $i$ and $j$. The spring force is calculated by

$$f_s = k(l_d - l_0) \tag{4.7.5.1}$$

where $k$ is the spring coefficient, $l_d = |S_j - S_i|$ is the deformed length and $l_0$ is the undeformed length along the vector between two points $S_i$ and $S_j$. The spring forces acting on bodies $i$ and $j$ are

$$f_s^{(i)} = \pm f_s l_s \quad ; \quad f_s^{(j)} = \mp f_s l_s \tag{4.7.5.2}$$

where $l_s = \frac{l_{ij}}{|l_{ij}|}$. The contribution of $f_s^{(i)}$ and $f_s^{(j)}$ to the force vector $\mathbf{F}_s$ on bodies $i$ and $j$ can be found from (4.7.3.3).

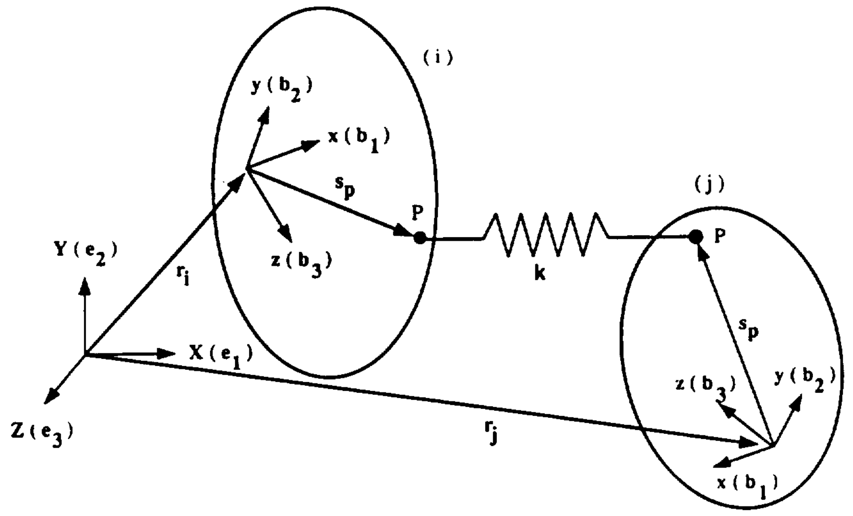


Fig. 4.9 A Spring Acting Between Two Bodies

## 4.8 Concluding Remarks

This chapter gives explicit mathematical expression for mechanical joints and forces pertaining to MBD systems. It is emphasized that the constraint Jacobian matrix is obtained by extracting velocity vectors from the time differentiation of the constraint equations. Under such circumstances, each joint, which is represented by a different constraint Jacobian matrix, can be written separately without risk of confusion. From a programming standpoint, this development enables MBD software to possess modularity so that the equations of motion for multidisciplinary engineering problems can be automatically generated.

The remaining issues regarding DAEs emphasize solution procedures. Chapter 5 reviews existing solution procedures, their advantages and disadvantages, and proposes two new constraint treatment schemes in conjunction with the two-stage staggered explicit-implicit algorithm to form a numerically robust solution procedure.

# CHAPTER V

# SOLUTION PROCEDURES FOR DAEs

## 5.1 Introduction

The equations of motion for MBD systems that are formulated in chapters 3 and 4 are characterized as DAEs. Since a closed form solution to DAEs cannot be found except for highly simplified problems, numerical methods must be applied in order to solve these highly nonlinear equations. Several existing numerical methods for solving DAEs are discussed in section 5.2. These numerical methods have primarily focused on the treatment of the constraint equations, which involved either constraint stabilization or constraint elimination. However, while these methods offer a varying degree of success, the lack of broadly applicable and robust numerical algorithms for solving DAEs remains as a challenging topic in the field of MBD systems. In this regard, two robust numerical methods that deal with both constraint stabilization and constraint elimination of DAEs are developed in sections 5.3 and 5.4. In section 5.5, a numerical algorithm called two-stage staggered explicit-implicit procedure is developed to integrate translational and rotational motions separately. The stability criteria of this algorithm will be derived by linearizing the Euler equations. It is shown that the present algorithm not only prevents the instability but also maintains the explicit nature of the algorithm.

## 5.2 Reviewing of Existing Solution Procedures

In reviewing existing DAEs solution procedures, a numerical method that is based on Gear's backward difference algorithm has been first developed to solve DAEs [20]. With little success by using this algorithm, Gear and Petzold [21,22] have solved DAEs by differentiating the constraint equations twice in time and augmenting these equations with the governing equations of motion to form a combined set of second-order differential equations. If the augmented constraint equations are numerically integrated, however, constraint violations will generally occur because of accumulated integration errors. Several solution procedures that are based on *the generalized coordinate partitioning technique* [27,53], *Baumgarte's constraint stabilization technique* [23,24], and *the null space method* [30-32,54] have been developed to overcome this key drawback. In the following sections we address these solution procedures in detail.

### 5.2.1 Stiffly-Stable Gear Method

The earliest numerical algorithm that was used to solve DAEs is the so-called stiffly-stable Gear algorithm [20] that has been applied to some restricted differential-algebraic equations. This algorithm was used to form a set of augmented equations of motion by considering the algebraic constraint equations as a special form of differential equations in which time derivatives of the variables do not appear. The equations of motion are then substituted into the backward difference formula and solved simultaneously with algebraic constraint equations that represent the kinematic joints of the system. The method starts with transforming DAEs into the following equations as

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{B}^T\boldsymbol{\lambda} = \mathbf{F} \tag{5.2.1.1}$$

$$\dot{\mathbf{u}} = \mathbf{v} \tag{5.2.1.2}$$

$$\mathbf{\Phi}(\mathbf{q},t) = 0 \tag{5.2.1.3}$$

which can be expressed into the following matrix form

$$\mathbf{f}(\mathbf{y},\dot{\mathbf{y}},t) = \begin{bmatrix} \mathbf{M} & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{v}} \\ \dot{\mathbf{u}} \\ \dot{\boldsymbol{\lambda}} \end{Bmatrix} + \begin{Bmatrix} \mathbf{F} - \mathbf{B}^T\boldsymbol{\lambda} \\ -\mathbf{v} \\ \mathbf{\Phi} \end{Bmatrix} = \mathbf{G}\dot{\mathbf{y}} + \mathbf{g} = 0 \tag{5.2.1.4}$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{M} & 0 & 0 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{g} = \begin{Bmatrix} \mathbf{F} - \mathbf{B}^T\boldsymbol{\lambda} \\ -\mathbf{v} \\ \mathbf{\Phi} \end{Bmatrix}$$

$$\mathbf{y} = [\mathbf{q},\mathbf{v},\boldsymbol{\lambda}]^T, \dot{\mathbf{y}} = [\dot{\mathbf{q}},\dot{\mathbf{v}},\dot{\boldsymbol{\lambda}}]^T$$

Solutions of (5.2.1.4) may contain both high and low frequency components depending upon the driving term $\mathbf{g}$ in (5.2.1.4) and the eigenvalues of the system, the present system may become numerically stiff. Numerically stiff systems are characterized by having solutions dominated by low-frequency components. However, due to the presence of high-frequency components, the time step of the explicit numerical algorithms must be kept relatively small. This means that a large number of time steps is required to obtain accurate solutions. Consequently, schemes that damp out errors associated with high-frequency components are desirable. The Gear algorithms, due to their stiffly-stable characteristics for high-frequency ranges, are thus well suited for solving stiff DAEs with parabolic characteristics.

The solution procedure starts with the Newton-Raphson algorithm, which is adopted to compute $\mathbf{y}$, applied to (5.2.1.4) as

$$\mathbf{f}_y^{(k)}\Delta\mathbf{y}^{(k)} + \mathbf{f}_{\dot{y}}^{(k)}\Delta\dot{\mathbf{y}}^{(k)} = -\mathbf{f}^{(k)} \tag{5.2.1.5}$$

where $k$ is the iteration cycle number. To obtain the relation between $\Delta \mathbf{y}^{(k)}$ and $\Delta \dot{\mathbf{y}}^{(k)}$, the $p^{th}$ order Gear algorithm for (5.2.1.5) is employed and written as

$$\mathbf{y}^{i+1} = \sum_{i=1}^{p-1} (a_j \mathbf{y}^{i-j}) + b\ h\ \mathbf{f}(\mathbf{y}^{i+1}, \dot{\mathbf{y}}^{i+1}) \qquad (5.2.1.6)$$

where $h$ is the time step, and $a_j$ and $b$ are the coefficients that need to be determined depending upon the order of the algorithms one wishes to apply. For the $k^{th}$ and $(k+1)^{th}$ iterations, (5.2.1.6) can be rewritten as

$$(\mathbf{y}^{i+1})^k = (\sum_{i=1}^{p-1} (a_j \mathbf{y}^{i-j}))^k + b\ h\ \mathbf{f}(\mathbf{y}^{i+1}, \dot{\mathbf{y}}^{i+1})^k \qquad (5.2.1.7)$$

$$(\mathbf{y}^{i+1})^{k+1} = (\sum_{i=1}^{p-1} (a_j \mathbf{y}^{i-j}))^{k+1} + b\ h\ \mathbf{f}(\mathbf{y}^{i+1}, \dot{\mathbf{y}}^{i+1})^{k+1} \qquad (5.2.1.8)$$

Since the summation term of (5.2.1.7) and (5.2.1.8) are only a function of the $i^{th}$ and previous time steps, they remain constant during the iteration. Subtracting (5.2.1.7) from (5.2.1.8) yields

$$\Delta \dot{\mathbf{y}}^{(k)} = \frac{1}{b\ h} \Delta \mathbf{y}^{(k)} \qquad (5.2.1.9)$$

which gives the relation of $\Delta \mathbf{y}^{(k)}$ and $\Delta \dot{\mathbf{y}}^{(k)}$. Substituting (5.2.1.9) back into (5.2.1.5), we obtain

$$(\mathbf{f}_y + \frac{1}{b\ h} \mathbf{f}_{\dot{y}})^{(k)} \Delta \mathbf{y}^{(k)} = (\mathbf{g}_y + \frac{1}{b\ h} \mathbf{G})^{(k)} \Delta \mathbf{y}^{(k)} = -\mathbf{f}^{(k)} \qquad (5.2.1.10)$$

Upon solving (5.2.1.10), the update value of $\mathbf{y}$ and $\dot{\mathbf{y}}$ can be obtained by

$$\mathbf{y}^{(k+1)} = \mathbf{y}^{(k)} + \Delta \mathbf{y}^{(k)} \qquad (5.2.1.11)$$

$$\dot{\mathbf{y}}^{(k+1)} = \dot{\mathbf{y}}^{(k)} + \frac{1}{b\ h} \Delta \mathbf{y}^{(k)} \qquad (5.2.1.12)$$

The drawbacks of this procedure are: first, it has expanded $n+m$ DAEs into $2n+m$ equations, thus for a large-scale system, it presents some inefficiency solving $\mathbf{y}$ and $\dot{\mathbf{y}}$ by using Newton-Raphson algorithm. Second, at starting time, there is no information available to determine the Lagrange multipliers $\boldsymbol{\lambda}$. An poor estimated $\boldsymbol{\lambda}$ may cause the system to diverge. Third, due to the requirement of many time steps, the time discretization may have compound effects on the constraint equations, and eventually lead to useless drifted solutions.

### 5.2.2 Direct Integration Method

In the previous section, we have observed the difficulty of choosing Lagrange multipliers. To overcome this difficulty, Gear and Petzold [21,22] purposed to convert the DAE system into a set of ordinary differential equations by appending the second time derivative of the constraint equations to the state equations. Combining (3.4.19) and (3.4.20) with the governing equations of motion, the resulting constraint-augmented equation can be written in the following matrix form:

$$\begin{bmatrix} \mathbf{M} & \mathbf{B}^T \\ \mathbf{B} & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{F} \\ \mathbf{c} \end{Bmatrix} \tag{5.2.2.1}$$

where $\mathbf{c} = -(\dot{\mathbf{B}}\dot{\mathbf{u}} + 2\boldsymbol{\Phi}_{ut}\dot{\mathbf{u}} + \boldsymbol{\Phi}_{tt})$. The Lagrange multipliers $\boldsymbol{\lambda}$ can be calculated by solving $\ddot{\mathbf{u}}$ of the first row of (5.2.2.1) and using the resulting expression substituted into the second row of (5.2.2.1) so that

$$\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T\boldsymbol{\lambda} = \mathbf{B}\mathbf{M}^{-1}\mathbf{F} + \mathbf{c} \tag{5.2.2.2}$$

If the $m \times m$ matrix $\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T$ is not singular, the acceleration vector $\ddot{\mathbf{u}}$ can be computed by substituting the result of (5.2.2.2) in to the first row of (5.2.2.1) to yield

$$\ddot{\mathbf{u}} = \mathbf{M}^{-1}[\mathbf{F} - \mathbf{B}^T(\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T)^{-1}(\mathbf{B}\mathbf{M}^{-1}\mathbf{F} + \mathbf{c})] \tag{5.2.2.3}$$

Numerical algorithms are then employed to compute the generalized velocity and position vectors at the next time step. Note that, upon using this approaches, several difficulties may occur: First, during the process of simulation, $\mathbf{BM}^{-1}\mathbf{B}^T$ may become ill-conditioned, which degrades the numerical accuracy of (5.2.2.2). Second, because numerical integration algorithms are used to compute $\ddot{\mathbf{u}}$, difficulty will occur for the reason that numerical time integration algorithms provide only an approximate solution of the equations. During the time integration, the numerical errors may start to accumulate to the point that the constraint conditions are no longer satisfied to the desired accuracies. Since there is no numerical mechanism to correct this defect, the solution may gradually diverge from the exact solution. This numerical difficulty is known as constraint violations. Third, the second time derivatives of the constraint equations do not necessarily represent the original algebraic constraint equations with fidelity in the case where nonlinear expressions are involved.

To avoid aforementioned difficulties, several corrective methods have been proposed. These methods include the generalized coordinate partitioning, Baumgarte constraint stabilization, the penalty constraint, and null space, which are reviewed in the sequel.

### 5.2.3 Generalized Coordinate Partitioning Method

The generalized coordinate partitioning method (GCPM) was first developed by Calahan [53]. Wehage and Haug [27] use this idea to reduce the system equations and determine independent coordinates from the constraint equations. Their approaches are based on the fact that the $n$ generalized coordinates of DAEs are not all independent. If the $n$ coordinates can be partitioned into $m$ dependent and $n - m$ independent coordinates, then the velocity and acceleration vectors can be partitioned into $\dot{\mathbf{u}}^d$, $\dot{\mathbf{u}}^i$, $\ddot{\mathbf{u}}^d$, and $\ddot{\mathbf{u}}^i$ accordingly, where $d$ denotes the dependent generalized coordinates, and $i$ denotes the independent generalized coordinates. The constraint equations (3.4.14) and its time derivatives (3.4.18) and (3.4.19) can be rewritten into the following forms:

$$\mathbf{\Phi}(\mathbf{u}^d, \mathbf{u}^i) = 0 \qquad (5.2.3.1)$$

$$\mathbf{B}^d \dot{\mathbf{u}}^d = -\mathbf{B}^i \dot{\mathbf{u}}^i \qquad (5.2.3.2)$$

$$\mathbf{B}\ddot{\mathbf{u}} + \dot{\mathbf{B}}\dot{\mathbf{u}} = [\mathbf{B}^d | \mathbf{B}^i] \left\{ \begin{matrix} \ddot{\mathbf{u}}^d \\ \ddot{\mathbf{u}}^i \end{matrix} \right\} + \dot{\mathbf{B}}\dot{\mathbf{u}} = 0 \qquad (5.2.3.3)$$

Since $\mathbf{B}^d$ has $m$ full row rank, which indicates $|\mathbf{B}^d| \neq 0$, therefore the dependent acceleration vector is given by

$$\ddot{\mathbf{u}}^d = -\mathbf{B}^{d^{-1}}(\mathbf{B}^i \ddot{\mathbf{u}}^i + \dot{\mathbf{B}}\dot{\mathbf{u}}) \qquad (5.2.3.4)$$

The equations of motion (3.4.13) can be rewritten into the following partitioned form

$$\begin{bmatrix} \mathbf{M}^d & 0 \\ 0 & \mathbf{M}^i \end{bmatrix} \left\{ \begin{matrix} \ddot{\mathbf{u}}^d \\ \ddot{\mathbf{u}}^i \end{matrix} \right\} + \left\{ \begin{matrix} \mathbf{B}^{d^T} \\ \mathbf{B}^{i^T} \end{matrix} \right\} \lambda = \left\{ \begin{matrix} \mathbf{F}^d \\ \mathbf{F}^i \end{matrix} \right\} \qquad (5.2.3.5)$$

Premultiplying by $\mathbf{B}^{d^{-T}}$ on the first row of (5.2.3.5) yields

$$\mathbf{B}^{d^{-T}} \mathbf{M}^d \ddot{\mathbf{u}}^d + \lambda = \mathbf{B}^{d^{-T}} \mathbf{F}^d \qquad (5.2.3.6)$$

Substituting $\lambda$ of (5.2.3.6) into the second row of (5.2.3.5) yields

$$\mathbf{M}^i \ddot{\mathbf{u}}^i + \mathbf{B}^{i^T}(\mathbf{B}^{d^{-T}} \mathbf{F}^d - \mathbf{B}^{d^{-T}} \mathbf{M}^d \ddot{\mathbf{u}}^d) = \mathbf{F}^i \qquad (5.2.3.7)$$

Replacement of $\ddot{\mathbf{u}}^d$ in (5.2.3.7) by (5.2.3.4) leads to

$$(\mathbf{M}^i + \mathbf{B}^{*^T} \mathbf{M}^d \mathbf{B}^*)\ddot{\mathbf{u}}^i = \mathbf{F}^i - \mathbf{B}^{*^T} \mathbf{F}^d - \mathbf{B}^{*^T} \mathbf{M}^d \mathbf{B}^{d^{-1}} \dot{\mathbf{B}}\dot{\mathbf{u}} \qquad (5.2.3.8)$$

where $\mathbf{B}^* = \mathbf{B}^{d^{-1}}\mathbf{B}^i$. Equation (5.2.3.8) represents the reduced form of DAEs where a set of independent differential equations is given. Numerically, these independent differential equations can be solved without violating the constraint equations. From a computational point of view, (5.2.3.8) is used to integrate the independent variables, whereas the dependent variables are obtained by satisfying the constraint equations (5.2.3.1), (5.2.3.2), and (5.2.3.3) at each time step. During the process of solving independent acceleration vector, one may not under estimate the computational cost of factorizing the left hand side of the fully populated matrix (5.2.3.8). A GCPM algorithm is stated as follows:

(1) Given initial conditions $\mathbf{u}^0$, and $\dot{\mathbf{u}}^0$.

(2) Solve (5.2.2.1) for $\ddot{\mathbf{u}}^n$, and $\boldsymbol{\lambda}^n$.

(3) Specify (Compute) independent variable $\ddot{\mathbf{u}}^{i^n}$.

(4) Integrate $\ddot{\mathbf{u}}^{i^n}$ to obtain $\mathbf{u}^{i^{n+1}}$, and $\dot{\mathbf{u}}^{i^{n+1}}$.

(5) Solve (5.2.3.1) for $\mathbf{u}^{d^{n+1}}$ by using Newton-Raphson method ; $\mathbf{u}^{n+1}$ is obtained.

(6) Solve (5.2.3.2) for $\dot{\mathbf{u}}^{d^{n+1}}$ and $\dot{\mathbf{u}}^{n+1}$ is found.

(7) Go to step (2) until the required run time is reached.

In step (5), a good estimate of $\mathbf{u}^d$ is needed so that the Newton-Raphson iteration may converge within a few iterations. A reasonable approximation of $\mathbf{u}^d$ can be obtained by taking the Taylor series expansion up to the third terms as

$$\mathbf{u}^{d^{n+1}} = \mathbf{u}^{d^n} + h\dot{\mathbf{u}}^{d^n} + \frac{h^2}{2}\ddot{\mathbf{u}}^{d^n} \tag{5.2.3.9}$$

Wehage and Haug [27] have developed an algorithm to identify independent and dependent generalized coordinates by using L-U factorization to decompose the constraint Jacobian matrix $\mathbf{B}$. This algorithm occasionally leads to poorly conditioned matrices. When this occurs, it is necessary to choose a new set of independent generalized coordinates by repeating the L-U factorization process. This strategy not only increases the computing time but also propagates integration errors. In recent years, many research groups have developed numerical techniques such as the singular values decomposition [28,29] (SVD) and the QR decomposition [55] to factorize the constraint Jacobian matrix. These techniques provide some marginal advantages over L-U factorization, but the main idea remains basically the same.

### 5.2.4 Baumgarte's Constraint Violation Stabilization Method

To stabilize the constraint violations that occur in solving (5.2.2.1), Baumgarte [23,24] proposed a constraint violation stabilization method. This method modifies the original constraint equations to form a set of relaxation differential equations that has the capability to suppress the growth of error and achieve a stable response. In this method, Baumgarte replaces the second row of (5.2.2.1) by the following constraint equations:

$$\ddot{\mathbf{\Phi}} + 2\alpha\dot{\mathbf{\Phi}} + \beta^2\mathbf{\Phi} = 0 \tag{5.2.4.1}$$

for holonomic constraints, and

$$\dot{\mathbf{\Phi}} + \gamma\mathbf{\Phi} = 0 \tag{5.2.4.2}$$

for nonholonomic constraints where $\alpha$, $\beta^2$ and $\gamma$ are arbitrary positive constants for numerical stability. Obviously, $2\alpha\dot{\mathbf{\Phi}}$, $\beta^2\mathbf{\Phi}$, and $\gamma\mathbf{\Phi}$ are the terms used to stabilize the error committed by the violation of constraint equations and their time derivatives. To study the behavior of the method, the general solution of the first order ordinary differential equation for constant $\gamma$ (5.2.4.2) is expressed as

$$\mathbf{\Phi}_i = k_i e^{-\gamma t}, i = 1, ..., m \tag{5.2.4.3}$$

where the constant, $k_i$, are determined from initial conditions. Note that $\mathbf{\Phi}_i$ is decaying as time $t$ is progressing. For the second-order ordinary differential equation (5.2.4.1), the general solution for constant $\alpha$, $\beta$ is

$$\mathbf{\Phi}_i = k_{1i} e^{\mu_1 t} + k_{2i} e^{\mu_2 t}, i = 1, ..., m \tag{5.2.4.4}$$

in which $k_{1i}$ and $k_{2i}$ are integration constants that depend on the initial conditions, and

$$\mu_{1,2} = -\alpha \pm \sqrt{\alpha^2 - \beta^2} \tag{5.2.4.5}$$

In order to make the solution to the constraint equation decay optimally, the critically damped (5.2.4.4) requires that $\alpha = \beta$ and $\alpha > 0$ so that

$$\Phi_i = (k_{1i} + k_{2i} t) e^{-\alpha t} \tag{5.2.4.6}$$

Substituting (3.4.18) and (3.4.19) into (5.2.4.1) and replacing $\ddot{\mathrm{u}}$ from the equations of motion, the Lagrange multiplier for holonomic constraints yields the following expression

$$\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T\boldsymbol{\lambda} = \mathbf{B}\mathbf{M}^{-1}\mathbf{F} + \mathbf{c} + 2\alpha\mathbf{B}\dot{\mathbf{u}} + \beta^2\mathbf{\Phi} \tag{5.2.4.7}$$

When $\alpha$ and $\beta$ are equal to zero, equation (5.2.4.7) recovers the original second time derivatives of the constraint equations (5.2.4.3) where the numerical solution may diverge from the exact solution. For nonzero $\alpha$ and $\beta$, the numerical solution oscillates about the exact solution. The magnitude and frequency of these oscillations depend on the values of $\alpha$ and

$\beta$. Park and Chiou (1986, 1987) have shown that for some MBD problems Baumgarte's technique suffers from the following drawbacks:

(1) When $\mathbf{BM}^{-1}\mathbf{B}^T$ is ill-conditioned, the accuracy of Lagrange multipliers is considerably degraded.

(2) The errors committed in the constraint equations decay with one time constant regardless of the physical natural of dynamic problems.

(3) Large values of $\alpha$, $\beta$, and $\gamma$ will cause the damping terms to dominate the numerical solution of the equations of motion, and make the system numerically stiff.

Since $\alpha$ and $\beta$ play a key role in this procedure, an analysis of this method is undertaken to obtain relationships between the coefficients $\alpha$, $\beta$, and the time stepsize $h$. Mathematically, one can approximate the nonlinear constraint equations by using Taylor's series expansion. In such an expansion, the holonomic constraint equations at the $(t + h)$ time step are truncated after the second derivative terms:

$$\mathbf{\Phi}(t+h) = \mathbf{\Phi}(t) + h\dot{\mathbf{\Phi}}(t) + \frac{h^2}{2}\ddot{\mathbf{\Phi}}(t) \tag{5.2.4.8}$$

in which $t$ is the current time, and $h$ is the time stepsize used in the numerical integration process. Since the constraint conditions should be vanished at time step $(t + h)$, (5.2.4.8) becomes

$$\ddot{\mathbf{\Phi}} + \frac{2}{h}\dot{\mathbf{\Phi}} + \frac{2}{h^2}\mathbf{\Phi} = 0 \tag{5.2.4.9}$$

Comparing (5.2.4.5) with (5.2.4.1), $\alpha$ and $\beta$ can be expressed in terms of $h$ as

$$\begin{aligned} \alpha &= \frac{1}{h} \\ \beta &= \frac{\sqrt{2}}{h} \end{aligned} \tag{5.2.4.10}$$

Replacement of $\alpha$ and $\beta$ in (5.2.4.5) by (5.2.4.10) lead to

$$\mu_{1,2} = \frac{1}{h}(-1 \pm i), \quad i = \sqrt{-1} \tag{5.2.4.11}$$

It is noted that (5.2.4.10) does not reach the critical damped as one has shown in (5.2.4.6). But if a constant time step integration algorithm is used, this modified version of Baumgarte's technique indeed damps out the constraint violations faster than an arbitrarily assigned constant value of $\alpha$ and $\beta$. Recently, Bae and Yang [56] have developed a method to determine the optimal stabilization constants $\alpha$ and $\beta$ so that the constraint violations can be damped out efficiently. But the difficulties concerning the appearance of an ill-conditioned $\mathbf{BM}^{-1}\mathbf{B}^T$ remain unanswered.

### 5.2.5 Null Space Method

The null space method [30,54] and its variations [17,18,31,32] are alternative methods to deal with DAEs that adopt special numerical procedures to eliminate system constraint forces. Hemami and Weimer [54] introduced a matrix method by considering the $m$ dimensional subspace spanned by the rows of the constraint Jacobian matrix. Let $\mathbf{C}$ be the orthogonal complement of $\mathbf{B}$, and $\mathbf{A}^T$ be a $(n - m) \times n$ matrix whose rows span the subspace $\mathbf{C}$. By definition

$$\mathbf{A}^T\mathbf{B}^T = 0 \tag{5.2.5.1}$$

Premultiplying (3.4.13) by $\mathbf{A}^T$ and using the result given by (5.2.5.1), the governing second-order differential equations become

$$\mathbf{A}^T\mathbf{M}\ddot{\mathbf{u}} = \mathbf{A}^T\mathbf{F} \tag{5.2.5.2}$$

Since the choice of $\mathbf{A}^T$ is not unique, the reduced system equations are not uniquely determined. Recently, de Jalon *et al.* [17,18] have utilized this concept to obtain the null space of the constraint Jacobian matrix by using natural coordinates. Their procedure starts with the assumption that $n-m$ independent velocities $\dot{\mathbf{u}}^i$ can be selected as a projection of $\dot{\mathbf{u}}$ which can be expressed as

$$\dot{\mathbf{u}}^i = \mathbf{E}\dot{\mathbf{u}} \tag{5.2.5.3}$$

where $\mathbf{E}$ is the matrix which defines the linear combination. Combining (5.2.5.3) with (3.4.18) yields

$$\begin{bmatrix} \mathbf{B} \\ \mathbf{E} \end{bmatrix} \dot{\mathbf{u}} = \left\{ \begin{matrix} -\mathbf{\Phi}_t \\ \dot{\mathbf{u}}^i \end{matrix} \right\} \tag{5.2.5.4}$$

thus if $\begin{bmatrix} \mathbf{B} \\ \mathbf{E} \end{bmatrix}$ is not singular, then

$$\dot{\mathbf{u}} = \begin{bmatrix} \mathbf{B} \\ \mathbf{E} \end{bmatrix}^{-1} \left\{ \begin{matrix} -\mathbf{\Phi}_t \\ \dot{\mathbf{u}}^i \end{matrix} \right\} = [\mathbf{C}|\mathbf{A}] \left\{ \begin{matrix} -\mathbf{\Phi}_t \\ \dot{\mathbf{u}}^i \end{matrix} \right\} \tag{5.2.5.5}$$

If $\mathbf{\Phi}_t = 0$, we conclude that

$$\dot{\mathbf{u}} = \mathbf{A}\dot{\mathbf{u}}^i \tag{5.2.5.6}$$

where $\mathbf{A}$ is an $n \times (n-m)$ matrix whose column constitute a basis of the null space of $\mathbf{B}$. Replacement of $\dot{\mathbf{u}}$ in (3.4.18) by (5.2.5.6) leads to

$$\mathbf{B}\dot{\mathbf{u}} = \mathbf{B}\mathbf{A}\dot{\mathbf{u}}^i = 0 \tag{5.2.5.7}$$

But $\mathbf{u}^i \neq 0$, which implies

$$\mathbf{B}\mathbf{A} = 0 \tag{5.2.5.8}$$

Transposing of (5.2.5.8) gives

$$\mathbf{A}^T\mathbf{B}^T = 0 \tag{5.2.5.9}$$

Equation (5.2.5.9) has achieved the goal of constructing the null space of the constraint Jacobian matrix so that constraint forces can be eliminated. Augmenting (5.2.5.2) and (3.4.19), the final system of equations of the present procedure can be rewritten in the following matrix form

$$\begin{bmatrix} \mathbf{A}^T\mathbf{M} \\ \mathbf{B} \end{bmatrix} \ddot{\mathbf{u}} = \begin{Bmatrix} \mathbf{A}^T\mathbf{F} \\ -\dot{\mathbf{B}}\dot{\mathbf{u}} \end{Bmatrix} \tag{5.2.5.10}$$

Note that (5.2.5.10) not only destroys the symmetry of the matrix but also violates the constraint conditions when time integration algorithms are used. To avoid these drawbacks, one can either adopt the Baumgarte constraint stabilization technique or reduce and express (5.2.5.2) in terms of system independent variables by taking the time differential of (5.2.5.6) as

$$\ddot{\mathbf{u}} = \mathbf{A}\ddot{\mathbf{u}}^i + \dot{\mathbf{A}}\dot{\mathbf{u}}^i \tag{5.2.5.11}$$

Substituting (5.2.5.11) into (5.2.5.2) yields

$$\mathbf{A}^T\mathbf{M}\mathbf{A}\ddot{\mathbf{u}}^i = \mathbf{A}^T\mathbf{F} - \mathbf{A}^T\mathbf{M}\dot{\mathbf{A}}\dot{\mathbf{u}}^i \tag{5.2.5.12}$$

This expression eliminates the system constraint forces and achieves the goal of symmetrizing system equations without violating constraint equations. However, this approach suffers from two major drawbacks, the first one arising from the fact that if numerical integration algorithms are used to integrate the independent accelerations $\ddot{\mathbf{u}}^i$, the null space matrix $\mathbf{A}$ and its time derivative $\dot{\mathbf{A}}$ need to be evaluated in advance so that the independent acceleration can be determined. Since the null space matrix $\mathbf{A}$ and its time derivative $\dot{\mathbf{A}}$ are obtained by solving system dependent velocity and position, resolution of the constraint equations by using a costly Newton-Raphson iteration becomes unavoidable. The second drawback is that during the process of time integration some positions can be reached which can cause the matrix in (5.2.5.4) to become singular. The reason is that the chosen independent variables do not numerically represent uniquely the motion of all the possible mechanisms during the process of simulation. To be able to continue the integration process, a new matrix **E** must be chosen.

Since these solution procedures suffer to some degree from computational difficulties, we have motivated to look for alternative solution procedures that overcome such difficulties. These new solution procedures which involve either constraint stabilization (penalty constraint stabilized technique) or constraint elimination (natural partitioning scheme) will be developed in the following sections to emphasize their numerical robustness and efficiency.

## 5.3 Penalty Constraint Stabilization Technique

In Baumgarte's method, the objective is to minimize the error accumulated in the constraint equations. In the penalty technique, instead of trying to eliminate the constraint violations, the errors being committed in constraint equations will be controlled. In other words, by applying the concept of proportional control of the constraint equations, the Lagrange multipliers are determined from the violation of the constraint equations as (Lanczos, 1949 [51])

$$\boldsymbol{\lambda} = \frac{\boldsymbol{\Phi}}{\epsilon}, \quad 0 < \epsilon << 1 \tag{5.3.1}$$

where $\epsilon$ is the penalty coefficient. Substituting (5.3.1) into the equations of motion, the final equation becomes

$$\mathbf{M}\ddot{\mathbf{u}} + \frac{1}{\epsilon}\mathbf{B}^T\boldsymbol{\Phi} = \mathbf{F} \tag{5.3.2}$$

An important consideration in using penalty techniques is that we assume the constraint violations always exist. A reduction of the constraint violations may be accomplished by decreasing the penalty coefficient which provides large number for corresponding Lagrange multipliers. As a result, the system equation (5.3.2) will be greatly stiffened which makes the integration of the equations of motion become numerically difficult. On the other hand, if the penalty coefficient is increased, the errors propagate into the integration process and may lead to an unacceptable drifted solution. Furthermore, the penalty coefficient can not be a fixed constant. The reason is that during the process of integrating a constrained dynamic system, different constraint equations may require different penalty coefficients in order to stabilize different constraint violations. This motivates us to look for an alternative way to stabilize the constraint violations.

To overcome the difficulties that have occurred in the previous techniques, a penalty constraint violation stabilization procedure [19,20] has been successfully introduced to correct the errors accumulated in the constraint equations. This procedure is based on the observation that a time differential equation of penalty formula retains the characteristic of parabolic in time so that it is amenable to direct time integration and the constraint violations will decay according to the different time constant. To illustrate this procedure, the nonholonomic constraints will be treated first. The penalty technique for the nonholonomic constraints is expressed as

$$\epsilon \boldsymbol{\lambda} = \mathbf{B}\dot{\mathbf{u}} + \boldsymbol{\Phi}(t) \tag{5.3.3}$$

Time differentiation of (5.3.3) yields

$$\epsilon \dot{\boldsymbol{\lambda}} = \mathbf{B}\ddot{\mathbf{u}} + \dot{\mathbf{B}}\dot{\mathbf{u}} + \boldsymbol{\Phi}_t \tag{5.3.4}$$

Substituting $\ddot{\mathbf{u}}$ from the equations of motion into (5.3.4) leads to

$$\epsilon\dot{\boldsymbol{\lambda}} + \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T\boldsymbol{\lambda} = \dot{\mathbf{B}}\dot{\mathbf{u}} + \mathbf{B}\mathbf{M}^{-1}\mathbf{F} + \boldsymbol{\Phi}_t \tag{5.3.5}$$

If $\lambda = ye^{-\gamma t}$, the eigenvalues of homogeneous part of (5.3.5) are

$$(\gamma_k\mathbf{I} + \frac{\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T}{\epsilon})y = 0 \tag{5.3.6}$$

where $(\gamma_k, k = 1, ..., m)$ are the eigenvalues of (5.3.6). Since $\gamma_k$ dictates how the errors will decay with time, the different time constants will correct the errors committed in the constraint equations. This property overcomes the difficulty in Baumgarte's method that errors that have been accumulated in the constraint equations can only decay with a fixed given time constants.

For holonomic constraints, time derivative of (5.3.1) yields

$$\epsilon\dot{\boldsymbol{\lambda}} = \mathbf{B}\dot{\mathbf{u}} + \boldsymbol{\Phi}_t \tag{5.3.7}$$

Integrating the equations of motion once to obtain the velocity vector, we get

$$\dot{\mathbf{u}}^n = \mathbf{h}^n_{\dot{\mathbf{u}}} + \varrho\mathbf{M}^{-1}(\mathbf{F} - \mathbf{B}^T\boldsymbol{\lambda})^n \tag{5.3.8}$$

where $\varrho$ is half of the time stepsize $h$, and $\mathbf{h}^n_{\dot{\mathbf{u}}}$ is the historical velocity vector that depends upon the applied numerical algorithms. Substituting $\dot{\mathbf{u}}^n$ into (5.3.7) yields

$$\epsilon\dot{\boldsymbol{\lambda}}^n + \varrho\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T\boldsymbol{\lambda}^n = \mathbf{B}^n(\varrho\mathbf{M}^{-1}\mathbf{F} + \mathbf{h}^n_{\dot{\mathbf{u}}})^n + \boldsymbol{\Phi}_t \tag{5.3.9}$$

Regardless of the nature of the constraints as shown in (5.3.5) and (5.3.9), the computation of Lagrange multipliers will not cause any numerical difficulty even if $\mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T$ becomes ill-conditioned. Furthermore, this technique provides two sets of differential equations for solving generalized coordinates and constraint forces. Since these two differential equations can be solved separately, the software modularity of present procedure is enhanced. Several example problems have been examined and shown that the penalty constraint stabilization technique not only treat the constraints correctly but yields more robust solutions.

In the present section, we have developed a procedure to minimize the constraint violations without introducing any artificial damping. In chapter 7, several example problems will be used to demonstrate the robustness and efficiency of the present stabilized technique. An ultimately different approach to prevent constraint violations will be introduced in the next section by adopting the concept of the null space method.

## 5.4 Natural Partitioning Scheme

A partitioning scheme based on physical-coordinate variables is presented in this section to systematically eliminate system constraint forces and yield the equations of motion of multibody dynamics systems in terms of their independent coordinates. Key features of the present scheme include: First, an explicit determination of the independent coordinates. Second, a parallel methodology to construct the null space matrix of the constraint Jacobian matrix is addressed if system topologies consist of a number of tree structures. For a system that contains closed-loops, a cut-joint technique is used so that the present scheme can be applied. Third, an easy incorporation of the previously developed two-stage staggered explicit-implicit solution procedure.

### 5.4.1 A Single Open Chain MBD System

To demonstrate the present physical coordinates partitioning scheme for open loop systems, a three-dimensional triple-pendulum problem is cho-

C 2

sen (Fig. 5.1).

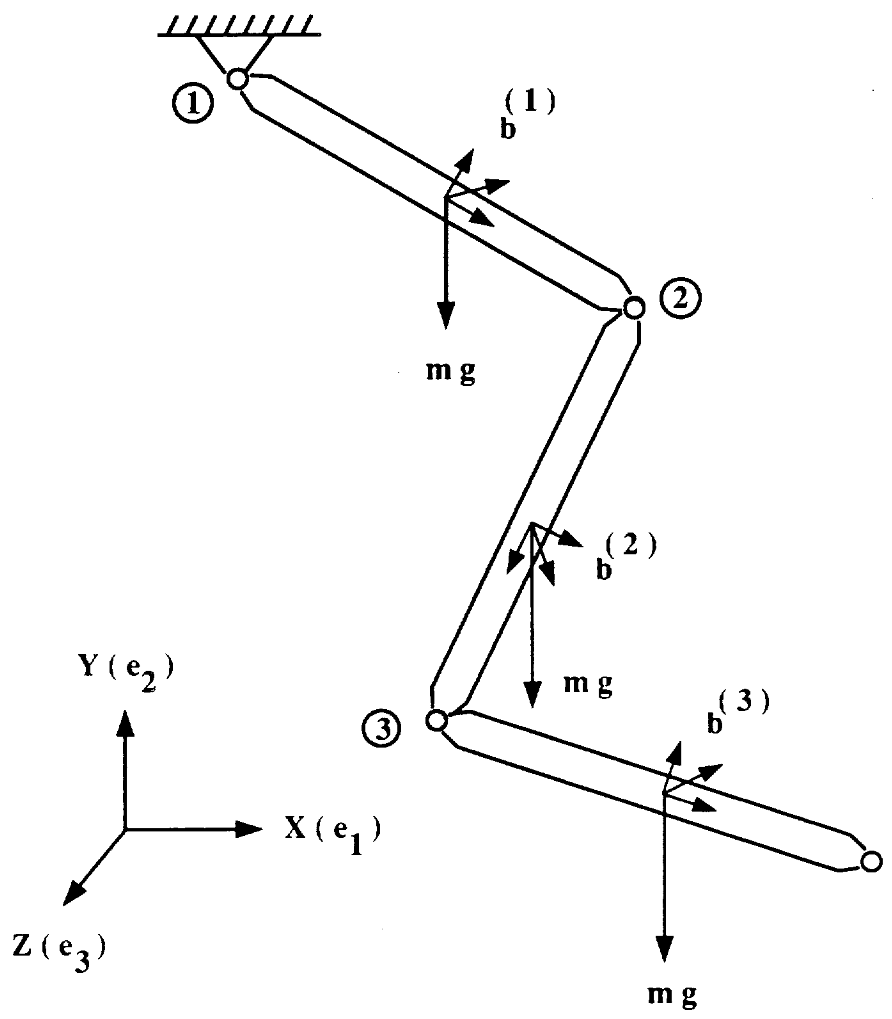


Fig. 5.1 Example of Open Chain a MBD System: Three-Dimensional Triple Pendulum

The constraint equations for this problem can be written as

$$\begin{bmatrix} [\mathbf{B}_{11}] & 0 & 0 \\ [\mathbf{B}_{21}] & [\mathbf{B}_{22}] & 0 \\ 0 & [\mathbf{B}_{32}] & [\mathbf{B}_{33}] \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \dot{\mathbf{u}}_2 \\ \dot{\mathbf{u}}_3 \end{Bmatrix} =$$

$$\begin{bmatrix} [-I] & [\mathbf{R}_{s11}] & 0 & 0 & 0 & 0 \\ [I] & [\mathbf{R}_{s21}] & [-\mathbf{I}] & [\mathbf{R}_{s22}] & 0 & 0 \\ 0 & 0 & [\mathbf{I}] & [\mathbf{R}_{s32}] & [-\mathbf{I}] & [\mathbf{R}_{s33}] \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \dot{\mathbf{u}}_2 \\ \dot{\mathbf{u}}_3 \end{Bmatrix} = 0 \qquad (5.4.1.1)$$

where the pendulum bodies are connected by three spherical joints and $\mathbf{R}_s$ are the function of rotational operators and position vectors from the center of mass of each body to the position of their connecting joints. To obtain the necessary projection matrix $\mathbf{A}$, we start with the first row of (5.4.1.1):

$$\mathbf{B}_{11}\dot{\mathbf{u}}_1 = [\ -\mathbf{I}\ ,\ \mathbf{R}_{s11}\ ]\ \dot{\mathbf{u}}_1 = 0 \qquad (5.4.1.2)$$

that can be partitioned into

$$[\mathbf{B}^d_{11}|\mathbf{B}^i_{11}] \begin{Bmatrix} \dot{\mathbf{u}}^d_1 \\ \dot{\mathbf{u}}^i_1 \end{Bmatrix} = 0 \qquad (5.4.1.3)$$

or

$$\mathbf{B}^d_{11}\dot{\mathbf{u}}^d_1 + \mathbf{B}^i_{11}\dot{\mathbf{u}}^i_1 = 0 \qquad (5.4.1.4)$$

where $\mathbf{B}^d_{11} = -\mathbf{I}$, $\mathbf{B}^i_{11} = \mathbf{R}_{s11}$, $d$ represents the dependent coordinates and $i$ represents the independent coordinates. Since $|\mathbf{B}^d_{11}| \neq 0$, the dependent velocity components of first body can be calculated as

$$\dot{\mathbf{u}}^d_1 = -{\mathbf{B}^d_{11}}^{-1}\mathbf{B}^i_{11}\dot{\mathbf{u}}^i_1 = \mathbf{P}_1\dot{\mathbf{u}}^i_1 \qquad (5.4.1.5)$$

where $\mathbf{P}_1 = -{\mathbf{B}^d_{11}}^{-1}\mathbf{B}^i_{11} = \mathbf{R}_{s11}$. The velocity vector of first body $\dot{\mathbf{u}}_1$ can be written in terms of independent velocities $\dot{\mathbf{u}}^i_1$ as

$$\dot{\mathbf{u}}_1 = \begin{Bmatrix} \dot{\mathbf{u}}^d_1 \\ \dot{\mathbf{u}}^i_1 \end{Bmatrix} = \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{I} \end{pmatrix} \dot{\mathbf{u}}^i_1 = \mathbf{Q}_1\dot{\mathbf{u}}^i_1 \qquad (5.4.1.6)$$

where $\mathbf{Q}_1 = \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{I} \end{pmatrix}$. Likewise, $\mathbf{B}_{22}$ of the second row of (5.4.1.1) can be partitioned into

$$\mathbf{B}_{21}\dot{\mathbf{u}}_1 + \mathbf{B}^d_{22}\dot{\mathbf{u}}^d_2 + \mathbf{B}^i_{22}\dot{\mathbf{u}}^i_2 = 0 \qquad (5.4.1.7)$$

or

$$\dot{\mathbf{u}}^d_2 = -{\mathbf{B}^d_{22}}^{-1}(\mathbf{B}_{21}\dot{\mathbf{u}}_1 + \mathbf{B}^i_{22}\dot{\mathbf{u}}^i_2) \qquad (5.4.1.8)$$

for $|\mathbf{B}^d_{22}| \neq 0$. Substituting (5.4.1.6) into (5.4.1.8) yields

$$\dot{\mathbf{u}}^d_2 = -\mathbf{B}^{d\ -1}_{22}(\mathbf{B}_{21}\mathbf{Q}_1\dot{\mathbf{u}}^i_1 + \mathbf{B}^i_{22}\dot{\mathbf{u}}^i_2) = \mathbf{R}_1\dot{\mathbf{u}}^i_1 + \mathbf{R}_2\dot{\mathbf{u}}^i_2 \qquad (5.4.1.9)$$

where $\mathbf{R}_1 = -\mathbf{B}^{d\ -1}_{22}\mathbf{B}_{21}\mathbf{Q}_1 = \mathbf{B}_{21}\mathbf{Q}_1$, and $\mathbf{R}_2 = -\mathbf{B}^{d\ -1}_{22}\mathbf{B}^i_{22} = \mathbf{B}^i_{22}$. The velocity vector of the second body, $\dot{\mathbf{u}}_2$, can be expressed in terms of the independent velocities, $\dot{\mathbf{u}}^i_1$ and $\dot{\mathbf{u}}^i_2$, as

$$\dot{\mathbf{u}}_2 = \begin{Bmatrix} \dot{\mathbf{u}}^d_2 \\ \dot{\mathbf{u}}^i_2 \end{Bmatrix} = \begin{bmatrix} \mathbf{R}_1 & \mathbf{R}_2 \\ 0 & \mathbf{I} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}^i_1 \\ \dot{\mathbf{u}}^i_2 \end{Bmatrix} = [\mathbf{S}_1|\mathbf{S}_2] \begin{Bmatrix} \dot{\mathbf{u}}^i_1 \\ \dot{\mathbf{u}}^i_2 \end{Bmatrix} \qquad (5.4.1.10)$$

where $\mathbf{S}_1 = \begin{pmatrix} \mathbf{R}_1 \\ 0 \end{pmatrix}$ and $\mathbf{S}_2 = \begin{pmatrix} \mathbf{R}_2 \\ \mathbf{I} \end{pmatrix}$. Applying the same procedure to the third row of (5.4.1.1), $\dot{\mathbf{u}}^d_3$ can be expressed as

$$\dot{\mathbf{u}}^d_3 = -\mathbf{B}^d_{33}(\mathbf{B}_{32}\dot{\mathbf{u}}_2 + \mathbf{B}^i_{33}\dot{\mathbf{u}}^i_3) = -\mathbf{B}^d_{33}[\mathbf{B}_{32}(\mathbf{S}_1\dot{\mathbf{u}}^i_1 + \mathbf{S}_2\dot{\mathbf{u}}^i_2) + \mathbf{B}^i_{33}\dot{\mathbf{u}}^i_3]$$

$$= \mathbf{V}_1\dot{\mathbf{u}}^i_1 + \mathbf{V}_2\dot{\mathbf{u}}^i_2 + \mathbf{V}_3\dot{\mathbf{u}}^i_3 \qquad (5.4.1.11)$$

where $\mathbf{V}_1 = -\mathbf{B}^{d\ -1}_{33}\mathbf{B}_{32}\mathbf{S}_1 = \mathbf{B}_{32}\mathbf{S}_1$, $\mathbf{V}_2 = -\mathbf{B}^{d\ -1}_{33}\mathbf{B}_{32}\mathbf{S}_2 = \mathbf{B}_{32}\mathbf{S}_2$, $\mathbf{V}_2 = -\mathbf{B}^{d\ -1}_{33}\mathbf{B}^i_{33} = \mathbf{B}^i_{33}$, and $\dot{\mathbf{u}}_3$ can be written in terms of $\dot{\mathbf{u}}^i_1$, $\dot{\mathbf{u}}^i_2$, and $\dot{\mathbf{u}}^i_3$ as

$$\dot{\mathbf{u}}_3 = \begin{Bmatrix} \dot{\mathbf{u}}^d_3 \\ \dot{\mathbf{u}}^i_3 \end{Bmatrix} = \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_2 & \mathbf{V}_3 \\ 0 & 0 & \mathbf{I} \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}^i_1 \\ \dot{\mathbf{u}}^i_2 \\ \dot{\mathbf{u}}^i_3 \end{Bmatrix} = [\mathbf{W}_1|\mathbf{W}_2|\mathbf{W}_3] \begin{Bmatrix} \dot{\mathbf{u}}^i_1 \\ \dot{\mathbf{u}}^i_2 \\ \dot{\mathbf{u}}^i_3 \end{Bmatrix} \qquad (5.4.1.12)$$

where $\mathbf{W}_1 = \begin{pmatrix} \mathbf{V}_1 \\ 0 \end{pmatrix}$, $\mathbf{W}_2 = \begin{pmatrix} \mathbf{V}_2 \\ 0 \end{pmatrix}$, and $\mathbf{W}_3 = \begin{pmatrix} \mathbf{V}_2 \\ \mathbf{I} \end{pmatrix}$. Combining (5.4.1.6), (5.4.1.10), and (5.4.1.12), we construct the physical velocities $\dot{\mathbf{u}}$ in terms of $\dot{\mathbf{u}}^i$ as

$$\begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \dot{\mathbf{u}}_2 \\ \dot{\mathbf{u}}_3 \end{Bmatrix} = \begin{bmatrix} \mathbf{Q}_1 & 0 & 0 \\ \mathbf{S}_1 & \mathbf{S}_2 & 0 \\ \mathbf{W}_1 & \mathbf{W}_2 & \mathbf{W}_3 \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}^i_1 \\ \dot{\mathbf{u}}^i_2 \\ \dot{\mathbf{u}}^i_3 \end{Bmatrix} \qquad (5.4.1.13)$$

or

$$\dot{\mathbf{u}} = \mathbf{A}\dot{\mathbf{u}}^i \qquad (5.4.1.14)$$

where **A** is the null space of the constraint Jacobian matrix that has been exploited in the previous section. Note that in the process of forming **A**, the inversion of the dependent matrices can be obtained analytically as opposed to the generalized coordinate partitioning scheme in which the inversion of the dependent matrices has to be carried out numerically. The scheme for constructing **A** provides a guideline to deal with MBD systems containing different topologies such as multiple open kinematic links and closed kinematic loops as discussed in the sequel.

### 5.4.2 A Multiple Open Chain System

If the MBD systems have more than one branch as shown in Fig. 5.2, the present scheme lends itself to multiprocessor computers. This property can be demonstrated by the following simple MBD system for which the constraint equations are given by

$$\begin{bmatrix} [\mathbf{B}_{11}] & 0 & 0 & 0 & 0 \\ [\mathbf{B}_{21}] & [\mathbf{B}_{22}] & 0 & 0 & 0 \\ 0 & [\mathbf{B}_{32}] & [\mathbf{B}_{33}] & 0 & 0 \\ [\mathbf{B}_{41}] & 0 & 0 & [\mathbf{B}_{44}] & 0 \\ 0 & 0 & 0 & [\mathbf{B}_{54}] & [\mathbf{B}_{55}] \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \dot{\mathbf{u}}_2 \\ \dot{\mathbf{u}}_3 \\ \dot{\mathbf{u}}_4 \\ \dot{\mathbf{u}}_5 \end{Bmatrix} = 0 \qquad (5.4.2.1)$$

Applying the scheme used in section 5.4.1 to both chains, the null space of the constraint Jacobian matrix **A** is decided as

$$\begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \dot{\mathbf{u}}_2 \\ \dot{\mathbf{u}}_3 \\ \dot{\mathbf{u}}_4 \\ \dot{\mathbf{u}}_5 \end{Bmatrix} = \begin{bmatrix} \mathbf{Q}_1 & 0 & 0 & 0 & 0 \\ \mathbf{S}_1 & \mathbf{S}_2 & 0 & 0 & 0 \\ \mathbf{W}_1 & \mathbf{W}_2 & \mathbf{W}_3 & 0 & 0 \\ \mathbf{Y}_1 & 0 & 0 & \mathbf{Y}_4 & 0 \\ \mathbf{Z}_1 & 0 & 0 & \mathbf{Z}_4 & \mathbf{Z}_5 \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}^i_1 \\ \dot{\mathbf{u}}^i_2 \\ \dot{\mathbf{u}}^i_3 \\ \dot{\mathbf{u}}^i_4 \\ \dot{\mathbf{u}}^i_5 \end{Bmatrix} \qquad (5.4.2.2)$$

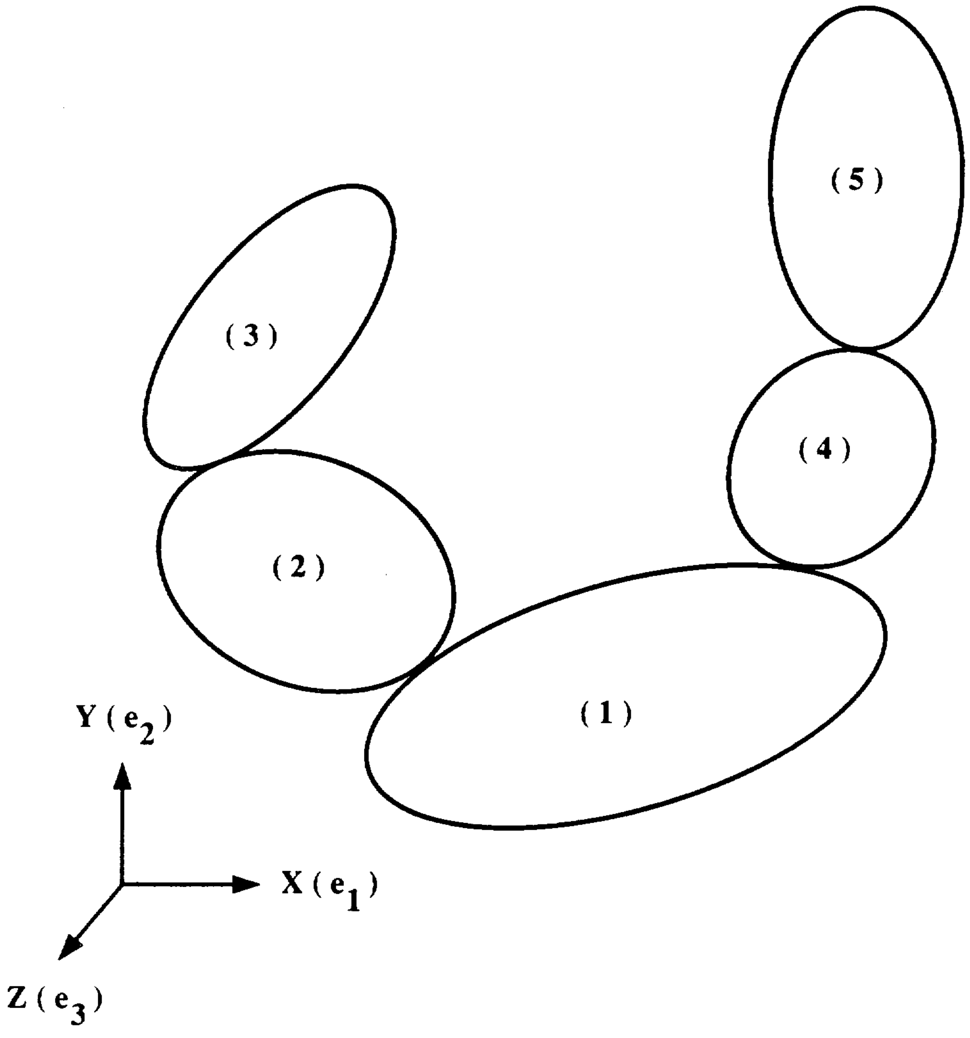


Fig. 5.2 Example of MBD Systems with Multiple Branches

Note that, in the physical coordinates partitioning scheme, once the first row of (5.4.2.2) is constructed, the second and fourth row of (5.4.2.2) can be constructed simultaneously according to the given $Q_1$. Again, if the first, second, and fourth rows of (5.4.2.2) are found, the third and fifth rows of (5.4.2.2) can be obtained according to their dependent branches respectly. Since MBD systems are the systems that include many kinematic loops, it is natural to utilize this development in a multiprocessor computer to compute the null space (at each branch) of the constraint Jacobian matrix.

### 5.4.3 A Closed-Loop MBD System

When the systems have one or more closed loops, difficulties arise in constructing the null space of the constraint Jacobian matrix as one can see by examining the following three body crank-slider problem (Fig. 5.3).

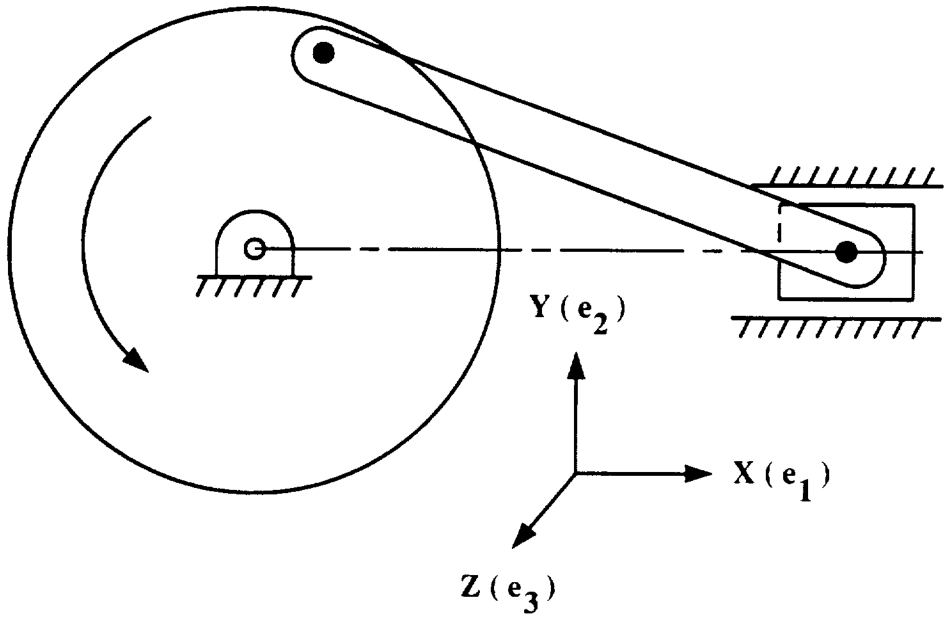


Fig. 5.3 Example of a Closed-Loop MBD System: The Crank-Slider Mechanism

The constraint equations for this problem are given by

$$\begin{bmatrix} [\mathbf{B}_{11}] & 0 & 0 \\ [\mathbf{B}_{21}] & [\mathbf{B}_{22}] & 0 \\ 0 & [\mathbf{B}_{32}] & [\mathbf{B}_{33}] \\ 0 & 0 & [\mathbf{B}_{43}] \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \dot{\mathbf{u}}_2 \\ \dot{\mathbf{u}}_3 \end{Bmatrix} = 0 \qquad (5.4.3.1)$$

It is obvious that joint 1 and 4 conflict in determining the null space of (5.4.3.1) according to the preceding scheme. Fortunely, there is an elegant way of overcoming that difficulty. The technique relys on "cut joints", which means cut the joints that are necessary to force the system topology to become open loop so that the existing solution procedure could be adopted. This technique is accomplished by partitioning (5.4.3.1) into the following form

$$\begin{bmatrix} [\mathbf{B}_{11}] & 0 & 0 \\ [\mathbf{B}_{21}] & [\mathbf{B}_{22}] & 0 \\ 0 & [\mathbf{B}_{32}] & [\mathbf{B}_{33}] \\ --- & --- & --- \\ 0 & 0 & [\mathbf{B}_{43}] \end{bmatrix} \begin{Bmatrix} \dot{\mathbf{u}}_1 \\ \dot{\mathbf{u}}_2 \\ \dot{\mathbf{u}}_3 \end{Bmatrix} = \begin{Bmatrix} \mathbf{B}_o \\ \mathbf{B}_c \end{Bmatrix} \dot{\mathbf{u}} = 0 \qquad (5.4.3.2)$$

or

$$\mathbf{B}_o\dot{\mathbf{u}} = 0, \quad \mathbf{B}_c\dot{\mathbf{u}} = 0 \qquad (5.4.3.3)$$

where $\mathbf{B}_o$ represents the open loop constraint Jacobian matrix, and $\mathbf{B}_c$ represents the remaining constraint Jacobian matrix after the joints have been cut. Performing the physical coordinates partitioning scheme to construct the null space of $\mathbf{B}_o$ as

$$\mathbf{B}_o\mathbf{A}_o = 0 \;\; ; \;\; \mathbf{A}_o^T\mathbf{B}_o^T = 0 \qquad (5.4.3.4)$$

Performing the algebraic calculations as in section 3.1 yields the equations of motion for a closed-loop MBD system as

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{B}_o^T\boldsymbol{\lambda}_o + \mathbf{B}_c^T\boldsymbol{\lambda}_c = \mathbf{F} \qquad (5.4.3.5)$$

Premultiplying $\mathbf{A}_o^T$ to the above equation yields

$$\mathbf{A}_o^T\mathbf{M}\ddot{\mathbf{u}} + \mathbf{A}_o^T\mathbf{B}_c^T\boldsymbol{\lambda}_c = \mathbf{A}_o^T\mathbf{F} \qquad (5.4.3.6)$$

or

$$\mathbf{M}_n\ddot{\mathbf{u}} + \mathbf{B}_n^T\boldsymbol{\lambda}_c = \mathbf{F}_n \qquad (5.4.3.7)$$

that are subjected to the constraints

$$\mathbf{B}_c\dot{\mathbf{u}} = 0 \tag{5.4.3.8}$$

where

$$\begin{aligned} \mathbf{M}_n &= \mathbf{A}_o^T\mathbf{M} \\ \mathbf{B}_n^T &= \mathbf{A}_o^T\mathbf{B}_c^T \\ \mathbf{F}_n &= \mathbf{A}_o^T\mathbf{F} \end{aligned} \tag{5.4.3.9}$$

Equations (5.4.3.7) and (5.4.3.8) can be solved either by employing the penalty constraint violation procedure that has previously developed or by constructing the null space for the new equations of motion.

## 5.5 Explicit-Implicit Solution Procedures

In sections 5.3 and 5.4, two schemes have been developed to treat constraints efficiently. The remaining task for the numerical solution of the equations of motion of MBD systems is the computation of the generalized coordinates. A solution procedure called two-stage staggered explicit-implicit procedure [43] has been developed to integrate the governing equations of motion. This procedure based on the partitioned solution procedure has been used to partition the governing equations of motion into translational and rotational components. Two numerical algorithms are used to integrate the generalized coordinates and constraint forces of the penalty constraint stabilization technique, and the generalized coordinates and independent coordinates of the natural partitioning scheme. The following sections describes this procedure in more detail.

### 5.5.1 Partitioning the Governing Equations of Motion

By using the penalty constraint stabilization technique, the discrete equations of motion for MBD system derived in section 3.4 may be expressed as

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{B}^T\boldsymbol{\lambda} = \mathbf{F}$$
$$\epsilon\dot{\boldsymbol{\lambda}} = \mathbf{B}\dot{\mathbf{u}} + \boldsymbol{\Phi}_t \tag{5.5.1.1}$$

It is noted that the integration of angular velocity does not lead to angular orientations, which are obtained by integrating a separate set of kinematic equations involving angular velocities. This motivates us to partition the generalized coordinates $\dot{\mathbf{u}}$ into translational velocity vector, $\dot{\mathbf{r}}$ and angular velocity vector $\boldsymbol{\omega}$ into the following forms

$$\dot{\mathbf{u}} = \begin{pmatrix} \dot{\mathbf{r}} \\ \boldsymbol{\omega} \end{pmatrix} , \quad \ddot{\mathbf{u}} = \begin{pmatrix} \ddot{\mathbf{r}} \\ \dot{\boldsymbol{\omega}} \end{pmatrix} \tag{5.5.1.2}$$

so that the desired angular orientations can be obtained by solving a set of kinematic equations. Since the translation and the rotation components are decoupled in the mass matrix, the equations of motion can be further partitioned into

$$\begin{bmatrix} \mathbf{M}_r & 0 \\ 0 & \mathbf{M}_\omega \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{r}} \\ \dot{\omega} \end{Bmatrix} + \begin{Bmatrix} \mathbf{B}_r^T \\ \mathbf{B}_\omega^T \end{Bmatrix} \lambda = \begin{Bmatrix} \mathbf{F}_r \\ \mathbf{F}_\omega \end{Bmatrix} \tag{5.5.1.3}$$

where the subscripts $(r, \omega)$ refer to translational and rotational motion respectively. The translational and rotational parts of the constraint Jacobian matrix are given by $\mathbf{B}_r$ and $\mathbf{B}_\omega$. Note that $(\dot{\mathbf{r}}, \boldsymbol{\omega})$ can be partitioned into body-by-body degrees of freedom as

$$\dot{\mathbf{r}} = [\dot{\mathbf{r}}^1, \dot{\mathbf{r}}^2, \ldots\ldots, \dot{\mathbf{r}}^{nb}]^T$$
$$\boldsymbol{\omega} = [\boldsymbol{\omega}^1, \boldsymbol{\omega}^2, \ldots\ldots, \boldsymbol{\omega}^{nb}]^T \tag{5.5.1.4}$$

where $nb$ is the total number of bodies in the system and $\dot{\mathbf{r}}^i$ and $\omega^i$ are the translational and rotational velocity vectors for the i-th body where

$$\dot{\mathbf{r}}^i = [\dot{r}_1^i, \dot{r}_2^i, \dot{r}_3^i]^T$$
$$\boldsymbol{\omega}^i = [\omega_1^i, \omega_2^i, \omega_3^i]^T \tag{5.5.1.5}$$

### 5.5.2 Two-Stage Staggered Explicit-Implicit Procedure

Since DAEs are not differential equations, a special study is needed to select robust and stable algorithms that yields accurate solutions. In this regard, studies must be conducted in order to analyze different numerical algorithms with their stability and accuracy characteristics. In structural dynamics, the most widely used numerical formula of solving the discrete equations of motion is the central difference formula, the half-station version of which may be written:

$$\begin{aligned} \dot{\mathbf{u}}^{n+\frac{1}{2}} &= \dot{\mathbf{u}}^{n-\frac{1}{2}} + h\ddot{\mathbf{u}}^{n} \\ \mathbf{u}^{n+1} &= \mathbf{u}^{n} + h\dot{\mathbf{u}}^{n+\frac{1}{2}} \end{aligned} \tag{5.5.2.1}$$

where $h$ is the time stepsize. This numerical algorithm is an explicit integration algorithm with second order accuracy. If $\omega_{max}$ is the highest instantaneous frequency, the stability condition of central difference formula is $\omega_{max}h \leq 2$ which imposes the time stepsize limitation. This algorithm is attractive to parallel computations because of its robustness and explicit nature, therefore the application of present algorithm to MBD systems needs to be evaluated carefully. A dynamic torque-free top problem will be demonstrated here to investigate the drawback of the central difference algorithm if the equations of motion are a function of velocity. The torque-free top problem is governed by the equation

$$\mathbf{J}\dot{\omega} + \omega \times \mathbf{J}\omega = 0 \tag{5.5.2.2}$$

where $\mathbf{J}$ is the inertia tensor. If the central difference formula is used to integrate this equation of motion, the angular velocity $\omega$ is obtained at the half time step via the integration formula, while the angular acceleration $\dot{\omega}$ are obtained at the full time step via the governing equation. Since $\omega$ at the full time step are not available, the angular acceleration $\dot{\omega}$ at the full time step cannot be found. In order to continue the integration process, two approximations have been studied:

(1) $\omega^n \approx \omega^{n-\frac{1}{2}}$

(2) $\omega^n \approx \frac{1}{2}(\omega^{n-\frac{1}{2}} + \omega^{n+\frac{1}{2}})$

In order to assess the stability of these approximations, it is necessary to linearize the equtions of motion so that the stability criteria can be established. To linearize of (5.5.2.2), we recall the rotational operator (2.3.1) and rewrite it into the following matrix form

$$\mathbf{R}(\mathbf{n}, \phi) = \mathbf{n}^T\mathbf{n} + \cos\phi(\mathbf{I} - \mathbf{n}^T\mathbf{n}) - \sin\phi\tilde{\mathbf{n}} \qquad (5.5.2.3)$$

The series for trigonometric functions sine and cosine are

$$\sin\phi = \phi - \frac{\phi^3}{3!} + \frac{\phi^5}{5!} - + \cdots \qquad (5.5.2.4)$$

$$\cos\phi = 1 - \frac{\phi^2}{2!} + \frac{\phi^4}{4!} - + \cdots \qquad (5.5.2.5)$$

Replacement of $\sin\phi$ and $\cos\phi$ in (5.5.2.3) by (5.5.2.4) and (5.5.2.5) lead to

$$\mathbf{R}(\mathbf{n}, \phi) = \mathbf{I}_{3\times3} + \frac{\phi^2}{2!}(\mathbf{n}^T\mathbf{n} - \mathbf{I}) - \frac{\phi^3}{3!}\tilde{\mathbf{n}}^T - \frac{\phi^4}{4!}(\mathbf{n}^T\mathbf{n} - I) + + - - \cdots \qquad (5.5.2.6)$$

Next we observe the relations of the skew-symmetric matrix $\tilde{\mathbf{n}}^T$ that

$$(\tilde{\mathbf{n}}^T)^3 = -\tilde{\mathbf{n}}^T, \quad (\tilde{\mathbf{n}}^T)^4 = -(\tilde{\mathbf{n}}^T)^2 \qquad (5.5.2.7)$$

$$(\tilde{\mathbf{n}}^T)^5 = \tilde{\mathbf{n}}^T, \quad (\tilde{\mathbf{n}}^T)^6 = (\tilde{\mathbf{n}}^T)^2, \cdots \qquad (5.5.2.8)$$

and

$$\mathbf{n}^T\mathbf{n} - \mathbf{I} = (\tilde{\mathbf{n}}^T)^2 = -(\tilde{\mathbf{n}}^T)^4 = (\tilde{\mathbf{n}}^T)^6 = \cdots \qquad (5.5.2.9)$$

Making the use of (5.5.2.7-9), (5.5.2.6) becomes

$$\mathbf{R}(\mathbf{n},\phi) = \mathbf{I}_{3\times3} + \phi\tilde{\mathbf{n}}^T + \frac{\phi^2}{2!}(\tilde{\mathbf{n}}^T)^2 + \frac{\phi^3}{3!}(\tilde{\mathbf{n}}^T)^3 + \cdots \tag{5.5.2.10}$$

or

$$\mathbf{R}(\mathbf{n},\phi) = e^{\phi\tilde{\mathbf{n}}^T} = e^{\tilde{\boldsymbol{\theta}}^T} \tag{5.5.2.11}$$

where $\tilde{\boldsymbol{\theta}} = \phi\tilde{\mathbf{n}}$. Since we have linearized the coordinate transformation matrix, the relation between $\mathbf{b}^n$ and $\mathbf{b}^{n+1}$ with an rotational angle $\theta$ can be written into

$$\mathbf{b}^{n+1} = e^{\tilde{\boldsymbol{\theta}}^T}\mathbf{b}^n = e^{\tilde{\boldsymbol{\theta}}^T}\mathbf{R}^n\mathbf{e} = \mathbf{R}^{n+1}\mathbf{e} \tag{5.5.2.12}$$

or

$$\mathbf{R}^{n+1} = e^{\tilde{\boldsymbol{\theta}}^T}\mathbf{R}^n \tag{5.5.2.13}$$

To establish the relationships between the angular velocity and the linearized parameters $\boldsymbol{\theta}$, the angular velocity and the coordinate transformation matrix are related by

$$\tilde{\omega}^{n+1} = -\dot{\mathbf{R}}^{n+1}\mathbf{R}^{n+1\,T} = -\frac{d}{dt}(e^{\tilde{\boldsymbol{\theta}}^T}\mathbf{R}^n)(\mathbf{R}^{n\,T}e^{\tilde{\boldsymbol{\theta}}}) \tag{5.5.2.14}$$

Carrying out the time derivation of (5.5.2.14) leads to

$$\tilde{\omega}^{n+1} = \dot{\tilde{\boldsymbol{\theta}}} + e^{\tilde{\boldsymbol{\theta}}^T}\omega^n e^{\tilde{\boldsymbol{\theta}}} \tag{5.5.2.15}$$

Approximating $e^{\tilde{\boldsymbol{\theta}}^T}$ of (5.5.2.15) by the first two terms of (5.5.2.10) which is given by

$$e^{\tilde{\boldsymbol{\theta}}^T} = \mathbf{I} - \tilde{\boldsymbol{\theta}} \tag{5.5.2.16}$$

and substituting (5.5.2.16) back into (5.5.2.15) lead to

$$\tilde{\omega}^{n+1} \approx \dot{\tilde{\boldsymbol{\theta}}} + \tilde{\omega}^n + \tilde{\omega}^n\tilde{\boldsymbol{\theta}} + \tilde{\boldsymbol{\theta}}^T\tilde{\omega}^n \tag{5.5.2.17}$$

or

$$\tilde{\omega}^{n+1} \approx \dot{\tilde{\boldsymbol{\theta}}} + \tilde{\omega}^{n} + \widetilde{\tilde{\omega}^{n}\boldsymbol{\theta}} \qquad (5.5.2.18)$$

which can be expressed in vector form as

$$\boldsymbol{\omega}^{n+1} = \boldsymbol{\omega}^{n} + \dot{\boldsymbol{\theta}} + \boldsymbol{\omega}^{n} \times \boldsymbol{\theta} \qquad (5.5.2.19)$$

Taking the time differentiation of (5.5.2.19), the linearization of the angular acceleration $\dot{\boldsymbol{\omega}}^{n+1}$ is obtained as

$$\dot{\boldsymbol{\omega}}^{n+1} = \dot{\boldsymbol{\omega}}^{n} + \ddot{\boldsymbol{\theta}} + \dot{\boldsymbol{\omega}}^{n} \times \boldsymbol{\theta} + \boldsymbol{\omega}^{n} \times \dot{\boldsymbol{\theta}} \qquad (5.5.2.20)$$

Replacement of $\dot{\boldsymbol{\omega}}^{n+1}$ and $\boldsymbol{\omega}^{n+1}$ in (5.5.2.2) by (5.5.2.20) and (5.5.2.19) lead to

$$\mathbf{J}\dot{\boldsymbol{\omega}}^{n+1} + \boldsymbol{\omega}^{n+1} \times \mathbf{J} \cdot \boldsymbol{\omega}^{n+1} = \mathbf{J}\dot{\boldsymbol{\omega}}^{n} + \boldsymbol{\omega}^{n} \times \mathbf{J} \cdot \boldsymbol{\omega}^{n} +$$

$$\mathbf{J}\ddot{\boldsymbol{\theta}} + [\mathbf{J}\tilde{\omega}^{n} - (\widetilde{\mathbf{J}\boldsymbol{\omega}^{n}}) + \tilde{\omega}^{n}\mathbf{J}]\dot{\boldsymbol{\theta}} + [\mathbf{J}\dot{\tilde{\omega}}^{n} - (\widetilde{\mathbf{J}\boldsymbol{\omega}^{n}})\tilde{\omega}^{n} + \tilde{\omega}^{n}\mathbf{J}\tilde{\omega}^{n}]\boldsymbol{\theta} = 0 \qquad (5.5.2.21)$$

by neglecting any of the two linearized parameters product terms. The final linearized equations of motion can be expressed as

$$\mathbf{J}\ddot{\boldsymbol{\theta}} + \mathbf{D}\dot{\boldsymbol{\theta}} + \mathbf{K}\boldsymbol{\theta} = 0 \qquad (5.5.2.22)$$

where the gyroscopic damping $\mathbf{D}$ and the centrifugal force $\mathbf{K}$ are given by the following matrix forms:

$$\mathbf{D} = \mathbf{J}\tilde{\omega}^{n} - (\widetilde{\mathbf{J}\boldsymbol{\omega}^{n}}) + \tilde{\omega}^{n}\mathbf{J} \qquad (5.5.2.23)$$

$$\mathbf{K} = \mathbf{J}\dot{\tilde{\omega}}^{n} - (\widetilde{\mathbf{J}\boldsymbol{\omega}^{n}})\tilde{\omega}^{n} + \tilde{\omega}^{n}\mathbf{J}\tilde{\omega}^{n} \qquad (5.5.2.24)$$

The central difference formula in (5.5.2.1) can be algebraically transformed to

$$\ddot{\boldsymbol{\theta}}^{n} = \frac{\boldsymbol{\theta}^{n+1} - 2\boldsymbol{\theta}^{n} + \boldsymbol{\theta}^{n-1}}{h^{2}} \qquad (5.5.2.25)$$

with the first velocity approximation

$$\dot{\boldsymbol{\theta}}^{n} \approx \dot{\boldsymbol{\theta}}^{n-\frac{1}{2}} \qquad (5.5.2.26)$$

in which

$$\dot{\boldsymbol{\theta}}^{n-\frac{1}{2}} = \frac{\boldsymbol{\theta}^{n} - \boldsymbol{\theta}^{n-1}}{h} \qquad (5.5.2.27)$$

Substituting (5.5.2.25-27) into (5.5.2.22) yields

$$\mathbf{J}(\boldsymbol{\theta}^{n+1} - 2\boldsymbol{\theta}^{n} + \boldsymbol{\theta}^{n-1}) + h\mathbf{D}(\boldsymbol{\theta}^{n} - \boldsymbol{\theta}^{n-1}) + h^2\mathbf{K}\boldsymbol{\theta}^{n} = 0 \qquad (5.5.2.28)$$

The computational stability of this approach can be assessed by seeking a nontrivial solution of

$$\boldsymbol{\theta}^{n+1} = s\boldsymbol{\theta}^{n} = s^2\boldsymbol{\theta}^{n-1} \qquad (5.5.2.29)$$

such that

$$|s| \leq 1 \qquad (5.5.2.30)$$

for stability. Substituting (5.5.2.29) into (5.5.2.28) yields the following characteristic equation

$$|\mathbf{J}(s-1)^2 + h\mathbf{D}(s-1) + h^2\mathbf{K}s| = 0 \qquad (5.5.2.31)$$

In order to examine the stability requirement on the characteristic equation, one transforms $|s| \leq 1$ onto the entire left-hand plane of the $z$-domain via

$$s = \frac{1+z}{1-z} \qquad (5.5.2.32)$$

so that

$$|\mathbf{J}(\frac{2z}{1-z})^2 + h\mathbf{D}(\frac{2z}{1-z}) + h^2\mathbf{K}(\frac{1+z}{1-z})| = 0 \qquad (5.5.2.33)$$

It is noted that the stability criteria is often decided by the gyroscopic damping term rather that the centrifugal force term. To simplify the stability analysis, we set $J_1 = J_2 = J_3 = 1$ and replace them into (5.5.2.23) and (5.5.2.24) to obtain

$$\mathbf{D} = \tilde{\omega} \tag{5.5.2.34}$$

$$\mathbf{K} = \dot{\tilde{\omega}} \tag{5.5.2.35}$$

Further simplification can be made if we substitute $J_1 = J_2 = J_3 = 1$ back into the original governing equation (5.5.2.2) to obtain $\dot{\omega} = 0$ which implies $\mathbf{K} = 0$. Substituting this expression and (5.5.2.34) into (5.5.2.33), the determination of (5.5.2.32) becomes

$$|4z^2\mathbf{I} + 2hz(1 - z)\tilde{\omega}| = 0 \tag{5.5.2.36}$$

Expanding (5.5.2.36), the resulting polynominal equation expressed in terms of $z$ as

$$[4+h^2(\omega_1^2+\omega_2^2+\omega_3^2)]z^2-2h^2(\omega_1^2+\omega_2^2+\omega_3^2)z+h^2(\omega_1^2+\omega_2^2+\omega_3^2) = 0 \tag{5.5.2.37}$$

The Routh-Hurwitz criterion asserts that for stability all the coefficients in (5.5.2.37) must be positive. Since the coefficient of the second term of (5.5.2.37) is negative, we conclude that the present velocity approximation makes the central difference algorithm numerically unstable.

The second velocity approximation is

$$\dot{\theta}^n = \frac{1}{2}(\dot{\theta}^{n+\frac{1}{2}} + \dot{\theta}^{n-\frac{1}{2}}) \tag{5.5.2.38}$$

or

$$\dot{\theta}^n = \frac{1}{2h}(\theta^{n+1} - \theta^{n-1}) \tag{5.5.2.39}$$

By applying the same stability procedure, the z-polynominal equation is given by

$$4z^2 + h^2(\omega_1^2 + \omega_2^2 + \omega_3^3) = 0 \tag{5.5.2.40}$$

Again, by using Routh-Hurwitz criterion, this algorithm is proven to be unconditionally stable. Thus we conclude that once the present algorithm is employed, a stable and accurate solution is obtained. However, since $\omega^{n+\frac{1}{2}}$ is not available as part of integration process, the robustness and explicit nature of the algorithm have been lost. This has motivated us to develop the following two-stage staggered explicit algorithm which prevents the instability of the first velocity approximation while circumventing the unavailability of the second velocity approximation.

In the two-stage staggered algorithm, the computational sequences have been divided into the following steps if $\boldsymbol{\theta}^n$, $\dot{\boldsymbol{\theta}}^n$, and $\ddot{\boldsymbol{\theta}}^n$ are known:

(1) $\dot{\boldsymbol{\theta}}^{n+\frac{1}{2}} = \dot{\boldsymbol{\theta}}^{n-\frac{1}{2}} + h\ddot{\boldsymbol{\theta}}^n$

(2) $\boldsymbol{\theta}^{n+\frac{1}{2}} = \boldsymbol{\theta}^{n-\frac{1}{2}} + h\dot{\boldsymbol{\theta}}^n$

(3) $\ddot{\boldsymbol{\theta}}^{n+\frac{1}{2}}$ is obtained by using (5.5.2.21)

(4) $\dot{\boldsymbol{\theta}}^{n+1} = \dot{\boldsymbol{\theta}}^n + h\ddot{\boldsymbol{\theta}}^{n+\frac{1}{2}}$

(5) $\boldsymbol{\theta}^{n+1} = \boldsymbol{\theta}^n + h\dot{\boldsymbol{\theta}}^{n+\frac{1}{2}}$

(6) $\ddot{\boldsymbol{\theta}}^{n+1}$ is obtained by using (5.5.2.21)

In this regard, we compute $\dot{\boldsymbol{\theta}}^n$ by

$$\begin{aligned} \dot{\boldsymbol{\theta}}^n &= \dot{\boldsymbol{\theta}}^{n-1} + h\ddot{\boldsymbol{\theta}}^{n-\frac{1}{2}} \\ \boldsymbol{\theta}^{n+\frac{1}{2}} &= \boldsymbol{\theta}^{n-\frac{1}{2}} + h\dot{\boldsymbol{\theta}}^n \end{aligned} \tag{5.5.2.41}$$

Following a similar step the same as in the previous cases, the characteristic equation for the two-stage algorithm becomes

$$|(s^4 - 2s^2 + 1)\mathbf{I} + h\tilde{\omega}(s^3 - s)| = 0 \tag{5.5.2.42}$$

Expanding and transforming (5.5.2.42), the resulting z-polynomial equation is given by

$$h^2 \Delta\omega z^4 + 2(8 - h^2\Delta\omega)z^2 + h^2\Delta\omega = 0 \tag{5.5.2.43}$$

where $\Delta\omega = \omega_1^2 + \omega_2^2 + \omega_3^2$. Equation (5.5.2.43) implies that the two-stage algorithm is stable if the stepsize remains in the range of

$$h \leq \frac{2\sqrt{2}}{(\omega_1^2 + \omega_2^2 + \omega_3^2)^{\frac{1}{2}}} \tag{5.5.2.44}$$

The foregoing linearized stability analysis for the torque-free top problem confirms that the two-stage staggered algorithm not only provides a stable and accurate solution but maintains the explicit nature of the algorithm.

For MBD systems, several example problems have been studied. A procedure called two-stage explicit-implicit procedure, which implements the previous development, has been used to integrate the governing equations of motion. The two-stage explicit-implicit procedure uses the explicit central difference algorithm to integrate the translational and rotational velocities and the translational displacements. An implicit numerical algorithm is used to integrate the Euler parameters by imposing the kinematic relation between the angular velocity and the Euler parameters and their time derivatives. The following sections describe these procedures in more detail.

## 5.5.3 Update of Translational and Angular Velocities

Since (5.5.1.3) provides two sets of uncoupled differential equations, the translational and rotational acceleration vectors at $n$-step can be explicitly computed assuming $\mathbf{r}^n$, $\dot{\mathbf{u}}^n$, and $\boldsymbol{\lambda}^n$ are known. The translational and angular velocity are integrated by the central-difference formula, namely

$$\begin{aligned} \dot{\mathbf{r}}^{n+\frac{1}{2}} &= \dot{\mathbf{r}}^{n-\frac{1}{2}} + h\ddot{\mathbf{r}}^{n} \\ \omega^{n+\frac{1}{2}} &= \omega^{n-\frac{1}{2}} + h\dot{\omega}^{n} \end{aligned} \qquad (5.5.3.1)$$

where the translational displacement $\mathbf{r}^{n+\frac{1}{2}}$ is updated by using

$$\mathbf{r}^{n+\frac{1}{2}} = \mathbf{r}^{n-\frac{1}{2}} + h\dot{\mathbf{r}}^{n} \qquad (5.5.3.2)$$

Note that the updating of the next half step results by simply changing the index $n+\frac{1}{2}$ to $n+1$, $n$ to $n+\frac{1}{2}$, and $n-\frac{1}{2}$ to $n$ as the integration proceed.

## 5.5.4 Update of Euler Parameters

As indicated in section 2.8, the Euler parameters can be obtained by making the use of the kinematic relation (2.8.7). In the present algorithm, the Euler parameters are integrated by the following equation

$$\mathbf{q}^{n+1} = \mathbf{q}^{n} + h\dot{\mathbf{q}}^{n+\frac{1}{2}} \qquad (5.5.4.1)$$

Substituting (2.8.7) into the above equations yields

$$\mathbf{q}^{n+1} = \mathbf{q}^{n} + hA(\omega^{n+\frac{1}{2}})\mathbf{q}^{n+\frac{1}{2}} \qquad (5.5.4.2)$$

The updated $\mathbf{q}^{n+1}$ needs to satisfy the constraint equation (2.6.3). Among several possibilities the approximation

$$\mathbf{q}^{n+\frac{1}{2}} = \frac{1}{2}(\mathbf{q}^{n} + \mathbf{q}^{n+1}) \qquad (5.5.4.3)$$

has been found to give the most accurate result. Replacement of $\mathbf{q}^{n+\frac{1}{2}}$ in (5.5.4.2) by (5.5.4.3) yields

$$(\mathbf{I} - \frac{h}{2}A(\omega^{n+\frac{1}{2}}))\mathbf{q}^{n+1} = (\mathbf{I} + \frac{h}{2}A(\omega^{n+\frac{1}{2}}))\mathbf{q}^{n} \qquad (5.5.4.4)$$

The skew-symmetric matrices on the left hand side of (5.5.4.4) can be explicitly inverted via the formula

$$\begin{bmatrix} 1 & -a & -b & -c \\ a & 1 & c & -b \\ b & -c & 1 & a \\ c & b & -a & 1 \end{bmatrix}^{-1} = \frac{1}{1+a^2+b^2+c^2}\begin{bmatrix} 1 & a & b & c \\ -a & 1 & -c & b \\ -b & c & 1 & -a \\ -c & -b & a & 1 \end{bmatrix} \tag{5.5.4.5}$$

Hence, the solution of $\mathbf{q}^{n+1}$ can be obtained without solving a set of linear equations as

$$\mathbf{q}^{n+1} = \frac{1}{\Delta}[\mathbf{I} + \frac{h}{2}A(\omega^{n+\frac{1}{2}})][\mathbf{I} + \frac{h}{2}A(\omega^{n+\frac{1}{2}})]\mathbf{q}^{n} \tag{5.5.4.6}$$

in which $\Delta = \frac{h^2}{4}(1+\omega_1^2+\omega_2^2+\omega_3^2)$. Once $\mathbf{q}^{n+1}$ is known, the coordinate transformation matrix $\mathbf{R}$ may be updated via (2.6.6) which relates the body-fixed basis of each body to the inertia basis. Note that for the two-stage staggered algorithm, the Euler parameters at $n+\frac{1}{2}$ can be computed by shifting the index $n+1$ to $n+\frac{1}{2}$ and $n+\frac{1}{2}$ to $n$ as

$$\mathbf{q}^{n+\frac{1}{2}} = \frac{1}{\Delta}[\mathbf{I} + \frac{h}{2}A(\omega^{n})][\mathbf{I} + \frac{h}{2}A(\omega^{n})]\mathbf{q}^{n-\frac{1}{2}} \tag{5.5.4.7}$$

## 5.5.5 Update of Constraint Forces

To compute the constraint forces for the holonomic constraints (5.3.9), one integrates $\dot{\mathbf{u}}^{n+\frac{1}{2}}$ and $\boldsymbol{\lambda}^{n+\frac{1}{2}}$ with the following mid-point implicit numerical algorithm:

$$\begin{aligned} \dot{\mathbf{u}}^{n+\frac{1}{2}} &= \dot{\mathbf{u}}^{n} + \varrho\ddot{\mathbf{u}}^{n+\frac{1}{2}} \\ \boldsymbol{\lambda}^{n+\frac{1}{2}} &= \boldsymbol{\lambda}^{n} + \varrho\dot{\boldsymbol{\lambda}}^{n+\frac{1}{2}} \end{aligned} \tag{5.5.5.1}$$

where $\varrho = \frac{h}{2}$. When $\ddot{\mathbf{u}}^{n+\frac{1}{2}}$ is substituted by the equations of motion, the constraint equations for holonomic constraints becomes

$$(\epsilon \mathbf{I} + \varrho^2 \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T)^{n+\frac{1}{2}} \boldsymbol{\lambda}^{n+\frac{1}{2}} = \epsilon \boldsymbol{\lambda}^n + \varrho \mathbf{B}^{n+\frac{1}{2}} \dot{\mathbf{u}}^n + \varrho^2 (\mathbf{B}\mathbf{M}^{-1}\mathbf{F})^{n+\frac{1}{2}} + \varrho \boldsymbol{\Phi}_t \tag{5.5.5.2}$$

The present two-stage explicit-implicit staggered procedure given by (5.5.3.1-2), (5.5.4.7), and (5.5.5.2) constitutes a complete solution procedure for MBD systems that undergo large translations and rotations. The procedure of solving each quantity is given in the next section.

### 5.5.6 Penalty Constraint Stabilization Technique Implementation

[1] Initialize $\mathbf{r}, \dot{\mathbf{r}}, \omega$, and $\mathbf{q}$.

[2] Compute $\mathbf{F}$ at $n$ step.

[3] Compute initial $\boldsymbol{\lambda}$

[4] Compute $\ddot{\mathbf{u}} = [\ddot{\mathbf{r}}, \dot{\omega}]$ at $n$ step.

[5] Update translational and angular velocity $\dot{\mathbf{r}}, \omega$ at $n + \frac{1}{2}$ step by using (5.5.3.1).

[6] Compute $\mathbf{q}^{n+\frac{1}{2}}$ with (5.5.4.7).

[7] Compute $\lambda$ at $n + \frac{1}{2}$ step.

[8] Repeat [2] and [4] at $n + \frac{1}{2}$ step.

[9] Update $\mathbf{r}^{n+1}$ with (5.5.3.2).

[10] Repeat [5] and [6] at $n + 1$ step.

[11] Extrapolate $\boldsymbol{\lambda}$ to $n + 1$.

[12] Go to [2] for next time step.

### 5.5.7 Natural Partitioning Scheme Implementation

[1] Initialize $\mathbf{r}, \dot{\mathbf{r}}, \omega$, and $\mathbf{q}$.

[2] Integrate $\mathbf{r}^{n+\frac{1}{2}}$ by $\mathbf{r}^{n+\frac{1}{2}} = \mathbf{r}^{n-\frac{1}{2}} + h\dot{\mathbf{r}}^n$

[3] Form $\mathbf{A}$ at $n$ step.

[4] Compute $\mathbf{A}^T\mathbf{F} - \mathbf{A}^T\mathbf{M}\dot{\mathbf{A}}\dot{\mathbf{u}}^i$ at $n$ step.

[5] Solve for $\ddot{\mathbf{u}}^i$ at $n$ step.

[6] Compute $\ddot{\mathbf{u}}$ at $n$ step.

[7] Integrate $\dot{\mathbf{u}}$ by using

$$\left\{ \begin{matrix} \dot{\mathbf{r}} \\ \omega \end{matrix} \right\}^{n+\frac{1}{2}} = \left\{ \begin{matrix} \dot{\mathbf{r}} \\ \omega \end{matrix} \right\}^{n-\frac{1}{2}} + h \left\{ \begin{matrix} \ddot{\mathbf{r}} \\ \dot{\omega} \end{matrix} \right\}^{n}$$

[8] Compute $\mathbf{q}^{n+\frac{1}{2}}$ according to (5.3.4.7).

[9] Repeat [2] at $n+1$ step.

[10] Repeat [3]-[6] at $n+\frac{1}{2}$ step.

[11] Repeat [7] and [8] at $n+1$ to complete the step.

[12] Go to [2] for next time step.

## 5.6 Concluding Remarks

The existing solution procedures of DAEs can be characterized into three categories: (1) treatment of DAEs as a special type of differential equation; (2) stabilization of the constraint violations by adopting special techniques; (3) construction of the null space of the constraint Jacobian matrix by employing appropriate matrix algorithms. So far, the relative performance of these methods have been measured largely in terms of a sequential computational context. However, new parallel computers are expected to have significant influence on the algorithm development for the large-scale MBD systems and real-time simulation. To address this issue, two schemes, viz., constraint stabilization and constraint elimination, have been developed with parallel computation in mind. In the constraint stabilization scheme, the constraint violations that occur during the time integration process are stabilized by adopting the rate form of the penalty scheme.

On the other hand, the constraint elimination scheme uses a decomposition of the physical coordinates that manifests dependent and independent variables so that the null space of the constraint Jacobian matrix can be constructed explicitly. Because both schemes convert DAEs into special types of differential equations, an algorithm based on the explicit central difference formula and implicit mid-point rule was adopted to integrate the equations of motion and their constraint forces or independent variables. Several example problems are used later to demonstrate the robustness and efficiency of this algorithm.

After DAEs are solved, it is natural to search for a method that can dramatically reduce computer run-time. Again with parallel computers in mind, this goal can be achieved by adopting a Schur complement based parallel preconditioned conjugate gradient numerical algorithm that determines: (1) system acceleration vectors and constraint forces for the constraint stabilization scheme; (2) system acceleration vectors and independent acceleration vectors for the constraint elimination technique. These details are covered in the next chapter.

# CHAPTER VI

# PARALLEL IMPLEMENTATION OF THE GOVERNING EQUATIONS OF MOTION

## 6.1 Introduction

During the past two decades, multiprocessing computer architectures have undergone progressive development because of the increasing availability of low cost multiprocessors. New parallel computers consisting of tens, hundreds, and even thousands of processors, have motivated the design of parallel algorithms and promised to have a profound impact on numerical simulation capabilities in many engineering disciplines. Some computer programs that run well on conventional sequential computers do not necessarily transformed to programs that efficiently harness the capabilities of parallel computers. This is particular true for massively parallel architecture. Conversely, algorithms that are less efficient in sequential computers may reveal an inherent parallelism that makes them attractive for parallel computers.

Since an MBD system may consist of hundreds or even thousands of linked bodies, numerical solutions of such highly nonlinear systems may consume a large amount of CPU time. The ultimate purpose of real-time simulation has motivated the development of efficient parallel algorithms on existing parallel computers. The issues that directly pertain to parallel computations of MBD systems include: generation of the system equations of motion, incorporation or elimination of constraint forces, integration of generalized coordinates, and interpretation of the simulation results. As suggested in previous chapters, the governing DAEs can be generated concurrently without any particular difficulty. However, the solution process that involves constraint stabilization or constraint elimination may introduce difficulties in the parallel determination of system generalized coordinates. In the present chapter a parallel solution method is proposed that computes directly the system constraint forces and generalized accelerations or system independent and generalized accelerations at the elementary body-by-body level. Once these quantities are known, the physical coordinates of each body can be computed independently and in parallel.

This chapter is organized as follows: sections 6.2 and 6.3 review two MBD equations that were derived in previous chapters. By rewriting these equations into body-by-body level, the governing equations can be transformed into a so-called *Arrowhead* matrix. The advantage of this matrix structure is that the generalized coordinates of system can be integrated simultaneously by using a previously developed two-stage staggered explicit-implicit algorithm. Section 6.4 discusses a parallel solution algorithm based on the conjugate gradient (CG) numerical algorithm, which is then applied to these arrowhead system equations. Finally, in section 6.5 two CG preconditioners are introduced to improve the convergent rate of the conjugate gradient numerical algorithm.

## 6.2 Parallel Implementation of Penalty Constraint Stabilization Technique

In order to obtain the optimum performance of the solution procedure, a procedure is developed to utilize parallel processors for solving constraint forces and improving the computations involving $\mathbf{BM}^{-1}\mathbf{B}^T$. In the two-stage explicit-implicit procedure, the numerical solution of MBD systems is obtained by combining two modules: a generalized coordinates integrator and a Lagrange multiplier solver. The Lagrange multiplier solver integrates the constraint equations with the mid-point rule, namely

$$\begin{aligned} \dot{\mathbf{u}}^{n+\frac{1}{2}} &= \dot{\mathbf{u}}^{n} + \varrho \ddot{\mathbf{u}}^{n+\frac{1}{2}} \\ \boldsymbol{\lambda}^{n+\frac{1}{2}} &= \boldsymbol{\lambda}^{n} + \varrho \dot{\boldsymbol{\lambda}}^{n+\frac{1}{2}} \end{aligned} \tag{6.2.1}$$

where $\varrho$ is equal to the half of the time step $h$. When $\ddot{\mathbf{u}}^{n+\frac{1}{2}}$ is substituted from the equations of motion, the Lagrange multiplier solver is obtained. From the parallel implementation viewpoint, this solution scheme is not attractive on account of its complexity in computing $\mathbf{BM}^{-1}\mathbf{B}^{T}$. An alternative scheme is developed to compute the Lagrange multipliers more efficiently. The scheme uses previous derivations without substituting $\ddot{\mathbf{u}}^{n+\frac{1}{2}}$, which can be replaced by the governing equations of motion, into the Lagrange multiplier solver. This leads to the following equations

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{u}}^{n+\frac{1}{2}} + (\mathbf{B}^{T}\boldsymbol{\lambda})^{n+\frac{1}{2}} &= \mathbf{F}^{n+\frac{1}{2}} \\ (\mathbf{B}\ddot{\mathbf{u}})^{n+\frac{1}{2}} - \frac{\epsilon}{\varrho^{2}}\boldsymbol{\lambda}^{n+\frac{1}{2}} &= -\frac{\epsilon}{\varrho^{2}}\boldsymbol{\lambda}^{n} - \frac{1}{\varrho}\mathbf{B}^{n+\frac{1}{2}}\dot{\mathbf{u}}^{n} \end{aligned} \tag{6.2.2}$$

or in matrix form

$$\begin{bmatrix} \mathbf{M} & \mathbf{B}^{T} \\ \mathbf{B} & \mathbf{E} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{c} \\ \mathbf{d} \end{Bmatrix} \tag{6.2.3}$$

where $\mathbf{E} = -\frac{\epsilon}{\varrho^{2}}\mathbf{I}$, $\mathbf{c} = \mathbf{F}^{n+\frac{1}{2}}$, $\mathbf{d} = -\frac{\epsilon}{\varrho^{2}}\boldsymbol{\lambda}^{n} - \frac{1}{\varrho}\mathbf{B}^{n+\frac{1}{2}}\dot{\mathbf{u}}^{n}$. We partition $\mathbf{M}$, $\ddot{\mathbf{u}}$, and $\mathbf{B}$ as

$$\begin{bmatrix} M_{(1,1)} & 0 & 0 & \dots & B_{(1,1)} & \dots & B_{(1,nj)} \\ 0 & M_{(2,2)} & 0 & \dots & B_{(2,1)} & \dots & B_{(2,nj)} \\ 0 & 0 & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & M_{(nb,nb)} & B_{(nb,1)} & \dots & B_{(nb,nj)} \\ B_{(1,1)} & B_{(1,2)} & \dots & B_{(1,nb)} & E_{1} & 0 & 0 \\ \dots & \dots & \dots & \dots & 0 & \dots & 0 \\ B_{(nj,1)} & B_{(nj,2)} & \dots & B_{(nj,nb)} & 0 & 0 & E_{nj} \end{bmatrix} \begin{Bmatrix} \ddot{u}_{1} \\ \ddot{u}_{2} \\ \dots \\ \ddot{u}_{n} \\ \lambda_{1} \\ \dots \\ \lambda_{nj} \end{Bmatrix} = \begin{Bmatrix} c_{1} \\ c_{2} \\ \dots \\ c_{n} \\ d_{1} \\ \dots \\ d_{2} \end{Bmatrix} \tag{6.2.4}$$

where $nb$ and $nj$ denote the total number of bodies and joints in the system respectively. Note the arrowhead matrix structure that appears in the acceleration equations can be written as

$$\begin{aligned} M_j \ddot{u}_j + \sum_{i=1}^{nj} B_{(j,i)} \lambda_i = c_j, \quad j = 1, \ldots, nb \\ \sum_{j=1}^{nb} B_{(i,j)} \ddot{u}_j + E_i \lambda_i = d_i, \quad i = 1, \ldots, nj \end{aligned} \qquad (6.2.5)$$

where $i$ is joint index and $j$ is the body index. The solution of (6.2.5.a) is found to be

$$\ddot{u}_j = M_j^{-1}(c_j - \sum_{i=1}^{nj} B_{(j,i)} \lambda_i), \quad j = 1, \ldots, nb \qquad (6.2.6)$$

where each $M_j$ is diagonal matrix. Substituting (6.2.6) into (6.2.5.b) yields

$$(E_i - \sum_{j=1}^{nb} B_{(i,j)} M_j^{-1} B_{(j,i)}) \lambda_i = d_i - \sum_{j=1}^{nb} B_{(i,j)} M_j^{-1} c_j, \quad i = 1, \ldots, nj \qquad (6.2.7)$$

Replacing $\mathbf{E}$ back into (6.2.7) to recover the equations (4.7.9) where the constraint forces $\lambda$ can be solved at the individual joint by joint level as

$$(\epsilon I_i + \varrho^2 \sum_{j=1}^{nb} B_{(i,j)} M_j^{-1} B_{(j,i)}) \lambda_i = \varrho^2 \sum_{j=1}^{nb} B_{(i,j)} M_j^{-1} c_j - \varrho^2 d_i, i = 1, \ldots, nj \qquad (6.2.8)$$

For convenience we transform (6.2.8) into the following equations so that efficient numerical algorithms can be applied to obtain their corresponding solutions.

$$\mathbf{M}^* \mathbf{x} = \mathbf{b} \qquad (6.2.9)$$

where

$$\begin{aligned} \mathbf{M}^* &= \epsilon I_i + \varrho^2 \sum_{j=1}^{nb} B_{(i,j)} M_j^{-1} B_{(j,i)} \\ \mathbf{x} &= \lambda_i \\ \mathbf{b} &= \varrho^2 \sum_{j=1}^{nb} B_{(i,j)} M_j^{-1} c_j - \varrho^2 d_i \end{aligned} \tag{6.2.10}$$

The following aspects of the present procedure should be noted:

(1) The parallelism in the multibody system is exploited by mapping each processor onto a group of bodies so that independent computations such as the left hand side of (6.2.7) can be carried out concurrently.

(2) Since $M_j$ is a constant mass matrix, it needs to be factored only once.

(3) To solve for $\lambda_i$, a parallel sparse solver such as described in [57,58] may be utilized.

(4) Once $\boldsymbol{\lambda}$ is obtained, the evaluation of $\ddot{\mathbf{u}}$ from (6.2.6) is trivially parallelized.

## 6.3 Parallel Implementation of Natural Partitioning Scheme

In deriving the second-order differential equations for the null space of the constraint Jacobian matrix, one can augment (5.6.11) and (5.6.2) into the following form:

$$\begin{bmatrix} -\mathbf{M} & \mathbf{MA} \\ \mathbf{A}^T\mathbf{M} & 0 \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{u}} \\ \ddot{\mathbf{u}}^i \end{Bmatrix} = \begin{Bmatrix} -\mathbf{M}\dot{\mathbf{A}}\dot{\mathbf{u}}^i \\ \mathbf{A}^T\mathbf{F} \end{Bmatrix} \tag{6.3.1}$$

Applying the same procedure for this arrowhead matrix as in the previous section, the independent acceleration coordinates $\ddot{\mathbf{u}}^i$ are given by

$$\sum_{j=1}^{nb} D_{(k,j)} M_j^{-1} D_{(j,k)}) \ddot{\mathbf{u}}^i = \sum_{j=1}^{nb} D_{(k,j)} M_j^{-1} \mathbf{F}_j - \sum_{j=1}^{nb} D_{(k,j)} \dot{\mathbf{A}} \dot{\mathbf{u}}^i, \quad k = 1, \ldots, nj \tag{6.3.2}$$

where

$$\begin{aligned}&\sum_{j=1}^{nb} D_{(k,j)} = \sum_{j=1}^{nb} A_j^T M_j, \quad k = 1, ...nj\\&\sum_{k=1}^{nj} D_{(j,k)} = M_j A_j, \quad j = 1, ...nb\end{aligned} \tag{6.3.3}$$

## 6.4 A Parallel Conjugate Gradient Solution Method

It is known that current MBD programs, which have been developed over the last twenty years, have been tailored for sequential computers with core memory limitations. Limited core memory has motivated researchers to develop sparse matrix methods that will dramatically decrease computer storage. In selecting a solution scheme for multiprocessing computers, iterative solution methods are preferred over direct methods for two reasons: (1) they efficiently exploit the sparsity of the involved matrices and therefore requires less storage than direct algorithms; (2) they provide the solution with an accuracy control that direct algorithms cannot provide. Most studies of MBD algorithms often assume that the system equations have already been formed. As indicated in (6.2.3) and (6.3.1), the system equations can be generated independently and in parallel. It would be natural if the solution scheme can be processed at the body-by-body level without forming the system equations.

Among the iterative solution methods, the conjugate gradient method [57-62] appears to be a most promising candidate because of its rate of convergence and inherent parallelism. Convergence of the conjugate gradient method is usually expected within $N$ iterations, especially if a good preconditioner is used. As for its inherent parallelism, it will be evident in the following step by step sequences. The conjugate gradient method consists of iteratively minimizing the residual

$$r_k = b - \mathbf{M}^* x_k \tag{6.4.1}$$

at each step $k$ with some estimate of $x_k$ and searchs along a direction $p_k$. In each subsequent iteration, a new solution vector is updated by

$$x_{k+1} = x_k + \alpha p_k \tag{6.4.2}$$

where

$$\alpha = \frac{r_k^T r_k}{p_k^T \mathbf{M}^* p_k} \tag{6.4.3}$$

The residual $r_{k+1}$ is then updated by

$$r_{k+1} = r_k - \alpha \mathbf{M}^* p_k \tag{6.4.4}$$

A new search direction needs to be established from the updated solution so that the residual $r$ is reduced as the iteration proceeds. The new search direction $p_{k+1}$ is chosen so that it is conjugate to all previous search directions $p_1, p_2, ..., p_k$. This is accomplished by

$$p_{k+1} = r_{k+1} + \beta p_k \tag{6.4.5}$$

where

$$\beta = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \tag{6.4.6}$$

A preconditioned conjugate gradient scheme applicable to MBD system equations (6.2.9) is summarized in the following:

(1) Solve $\mathbf{M}^* \mathbf{x} = \mathbf{b}$ in parallel using all available processors

• Form the right hand side of the Schur complement:

For $j = 1$ to $N_p$ do concurrently

```
        Form $T_r(j) = M(j)^{-1}c(j)$
        Form $b(j) = d(j) - D(j)T_r(j)$
• Initialize:
        $x_0 = 0$
        $r_0 = b$
For $k = 1, \ldots\ldots, n$
        If $r_{k-1} < \epsilon$ then quit
        Else
• Compute the new conjugate search direction:
            Solve $\mathbf{P}z_{k-1} = r_{k-1}$ for $z_{k-1}$
            $\beta_k = z_{k-1}^T r_{k-1} / z_{k-2}^T r_{k-2} \quad (\beta_1 = 0)$
            $p_k = z_{k-1} + \beta_k p_{k-1} \quad (p_1 = z_0)$
• Form the left hand side of the Schur complement:
            For $j = 1$ to $N_p$ do concurrently
            Form $T_l(j) = D^T(j)p_k(j)$
            Form $T_l(j) = M(j)^{-1}T_l(j)$
            Form $\mathbf{M}(j)^* p_k(j) = -D(j)T_l(j)$
• Line search to update solution and residual:
            $\alpha_k = z_{k-1}^T r_{k-1} / p_k^T \mathbf{M}^* p_k$
            $x_k = x_{k-1} + \alpha_k p_k$
            $r_k = r_{k-1} - \alpha_k \mathbf{M}^* p_k$
        Endif
```

(2) Broadcast the part of $\mathbf{x}$ corresponding to the handled rows of $\mathbf{D}$ to neighboring processors and solve for $\ddot{\mathbf{u}}$ as in the following steps:

```
    For j = 1 to $N_p$ do concurrently
        • Receive $\mathbf{x}$
```

- Back substitute for $\ddot{\mathbf{u}}$
- Send $\ddot{\mathbf{u}}$ to host for output

As noted in (6.2.9), the preconditioned conjugate gradient numerical algorithm is used to obtain system constraint forces without forming the global constraint Jacobian matrix. This is because the major operation of the conjugate gradient is that involving the multiplication of a matrix by a vector. Thus, we can multiply $BM^{-1}B^T$ as

$$\begin{aligned} u_i^e &= BM^{-1}B^T p \\ &= \sum_{j=1}^{nb} B_{(i,j)} M_j^{-1} B_{(j,i)} p \\ &= \sum_{j=1}^{nb} B_{(i,j)} M_j^{-1} p^e \\ &= \sum_{j=1}^{nb} B_{(i,j)} u^e \end{aligned} \tag{6.4.7}$$

where $u_i^e = [u_i^{(1)}, u_i^{(2)}, ..., u_i^{(nj)}]$. This multiplication is performed in three steps, which add different contributions from prospective bodies to the entry of the resulting vector. The matrix-vector multiplications are performed directly at the body level and together produce the global vector $u_i^e$.

## 6.5 Preconditioners

To improve the convergence rate of CG, *preconditioning* [1,2,4-6] is wildly used to reduce the number of iterations required to convergence. This is achieved by solving the modified system

$$\mathbf{PM^*x} = \mathbf{Pb} \tag{6.5.1}$$

where $\mathbf{P}$ is the preconditioning matrix. Presumably, to obtain a computationally efficient algorithm, we want $\mathbf{P}$ to be an approximation to $\mathbf{M}^{*-1}$ in some sense, which is easy to calculate. Many choices of the preconditioning matrix have been proposed, ranging from using the diagonal entries of matrix $\mathbf{M}^*$ to some forms of the incomplete Cholesky factorization of $\mathbf{M}^*$. The selection of effective preconditioners remains a topic of much current research.

### 6.5.1 Diagonal Preconditioner

To complete the implementation of a preconditioned conjugate gradient algorithm, the preconditioning matrix $\mathbf{P}$ needs to be determined. The simplest choice consists of taking $\mathbf{P}$ to be a diagonal matrix, formed with the diagonal entries of the dense matrix $\mathbf{M}^*$:

$$\begin{aligned} \mathbf{P} = diag[\mathbf{M}^*] &= diag[\epsilon I_i + \varrho^2 \sum_{j=1}^{nb} B_{(i,j)} M_j^{-1} B_{(j,i)}] \\ &= diag[\epsilon I_i] + diag[\varrho^2 \sum_{j=1}^{nb} B_{(i,j)} M_j^{-1} B_{(j,i)}] \end{aligned} \tag{6.5.1.1}$$

where $diag$ denotes the diagonal entries of the corresponding matrices. Since $\mathbf{M}$ is a constant diagonal mass matrix, we can explicitly invert $\mathbf{M}$ as

$$M_j^{-1} = L_j L_j = L_j L_j^T, \quad j = 1, ..., nb \tag{6.5.1.2}$$

where $L_j = M_j^{-\frac{1}{2}}$. Making the use of (6.5.1.2), the preconditioning matrix $\mathbf{P}$ can be rewritten as

$$P_i = diag[\epsilon I_i] + \varrho^2 diag[\sum_{j=1}^{nb} G_{(i,j)} G_{(j,i)}], \quad i = 1, ..., nj \tag{6.5.1.3}$$

where $G_{(i,j)} = B_{(i,j)} L_j$. Note that all calculations pertaining to the second term of (6.5.1.3) can be carried out internally at the individual joint level. This development enables us to use multiprocessors in computing the preconditioning matrix $\mathbf{P}$.

### 6.5.2 Scaled Preconditioner

Another approach for choosing the preconditioning matrix consists of scaling $\mathbf{M}^*$ with its own diagonal entries as follows. Let

$$\mathbf{M}^* = \mathbf{D}(\mathbf{M}^*) + [\mathbf{M}^* - \mathbf{D}(\mathbf{M}^*)] \tag{6.5.2.1}$$

where $\mathbf{D}(\mathbf{M}^*)$ contains the diagonal entries of $\mathbf{M}^*$. Equation (6.5.2.1) can be algebraically transformed to the following relation:

$$\begin{aligned}\mathbf{M}^* &= \mathbf{D}(\mathbf{M}^*)^{\frac{1}{2}}\{\mathbf{I} + \mathbf{D}(\mathbf{M}^*)^{-\frac{1}{2}}[\mathbf{M}^* - \mathbf{D}(\mathbf{M}^*)]\mathbf{D}(\mathbf{M}^*)^{-\frac{1}{2}}\}\mathbf{D}(\mathbf{M}^*)^{\frac{1}{2}} \\ &= \mathbf{D}(\mathbf{M}^*)^{\frac{1}{2}}[\mathbf{I} + \mathbf{V}]\mathbf{D}(\mathbf{M}^*)^{\frac{1}{2}}\end{aligned} \tag{6.5.2.2}$$

where

$$\mathbf{V} = \mathbf{D}(\mathbf{M}^*)^{-\frac{1}{2}}[\mathbf{M}^* - \mathbf{D}(\mathbf{M}^*)]\mathbf{D}(\mathbf{M}^*)^{-\frac{1}{2}} \tag{6.5.2.3}$$

The preconditioning matrix $\mathbf{P}$ is taken to be

$$\mathbf{P} = \mathbf{M}^{*-1} = \mathbf{D}(\mathbf{M}^*)^{-\frac{1}{2}}[\mathbf{I} + \mathbf{V}]^{-1}\mathbf{D}(\mathbf{M}^*)^{-\frac{1}{2}} \tag{6.5.2.4}$$

If $||\mathbf{V}|| < 1$, the series expansion of $(\mathbf{I} + \mathbf{V})$ is

$$[\mathbf{I} + \mathbf{V}]^{-1} \approx \mathbf{I} - \mathbf{V} + \mathbf{V}^2 - \mathbf{V}^3 + \mathbf{V}^4 - \ldots \tag{6.5.2.5}$$

which is always valid since the scaling by $\mathbf{D}(\mathbf{M}^*)$ ensures that the eigenvalues of $\mathbf{V}$ are less than one. Substituting the first two terms of (6.5.2.5) into (6.5.2.4), we obtain

$$\mathbf{P} = \mathbf{D}(\mathbf{M}^*)^{-\frac{1}{2}}[\mathbf{I} - \mathbf{V}]\mathbf{D}(\mathbf{M}^*)^{-\frac{1}{2}} \tag{6.5.2.6}$$

This expansion can be calculated at the individual joint level as

$$P_i = D_i(\mathbf{M}^*)^{-\frac{1}{2}}[I_i - V_i]D_i(\mathbf{M}^*)^{-\frac{1}{2}}, \quad i = 1, \ldots, nj \tag{6.5.2.7}$$

Sections 6.3 and 6.4 have presented the analysis of the preconditioned conjugate gradient numerical algorithm applied to the arrowhead matrices. A prototype code for dynamics analysis of MBD systems on a shared-memory multiprocessor has been developed at the Center for Space Structures and Controls (CSSC). This code uses the software architecture and the numerical algorithm presented in sections 6.3 and 6.4.1. A test version called PMBS (Parallel Multi-Body System) has been implemented on the Alliant FX/8 by using the *Force* preprocessor [63] which is a macro-based extension to Fortran 77 for shared memory multiprocessors.

## 6.6 Concluding Remarks

In this chapter, we have reformulated the MBD equations and their corresponding stabilization techniques to take the advantage of the arrowhead coefficient matrices. A parallel numerical algorithm based on the preconditioned conjugate gradient scheme has been employed to obtain the solutions of systems involving these matrices. Since the use of a preconditioner may dramatically improve the convergence of the conjugate gradient scheme, two methods based on the diagonal entries of the solution matrices have been discussed.

In the next chapter, several example problems are solved. These problems illustrate the following aspects: correction of the constraint violations, obtaining accurate solution, preventing instability of employing existing explicit numerical algorithms, and solving system equations by using parallel numerical algorithms.

# CHAPTER VII

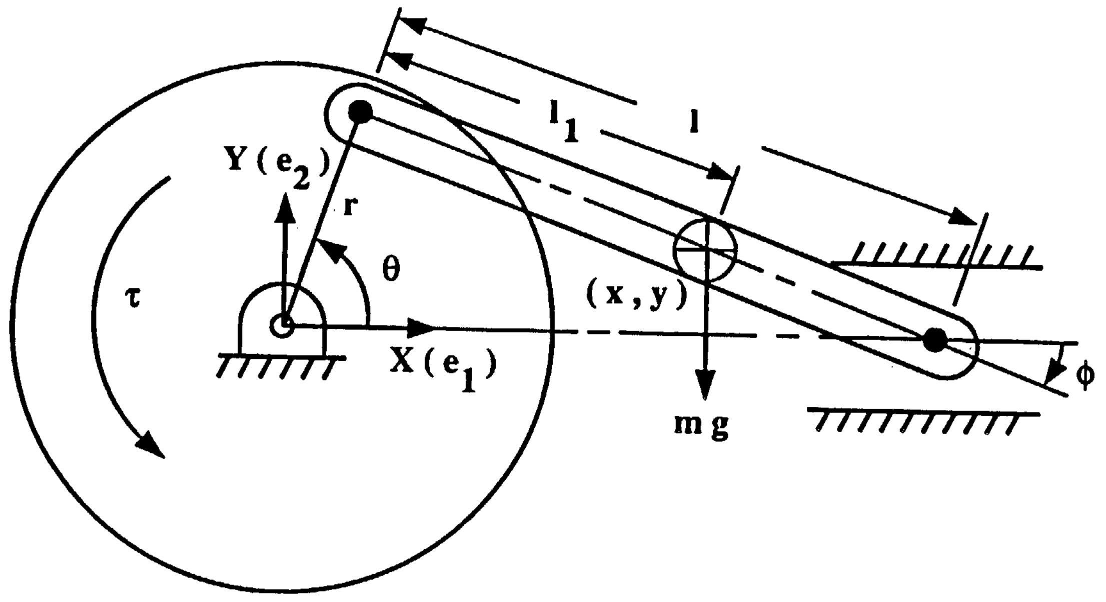
# NUMERICAL EXAMPLES

## 7.1 Introduction

In sections 5.8 and 5.9, two computational solution procedures have been developed to solve DAEs while maintaining constraint verification within an acceptable limit. In this chapter several example problems are carefully examined to demonstrate the robustness and effectiveness of these procedures as regards some important issues that affect the solution of DAEs. These issues include how to: (1) efficiently correct for constraint violations; (2) accurately obtain the solutions of the system equations; (3) elegantly overcome the ill-conditioned $\mathbf{BM^{-1}B^T}$ in solving Lagrange multipliers; (4) systematically handle systems with both holonomic and non-holonomic constraints; (5) analytically prevent instability of using explicit central difference formula by approximating the angular velocity for the evaluation of angular acceleration; (6) kinematically interact systems with flexible and rigid bodies easily; (7) systematically solve MBD systems with closed-loop system topology; (8) precisely deal with the systems contained specific time dependent constraints; (9) efficiently solve system dynamic equations by employing a parallel numerical algorithm. We begin the discussion of these issues by examining the following examples.

## 7.2 The Crank-Slider Mechanism

The first numerical example is a classical crank-slider mechanism (Fig. 7.2.1) whose governing equations of motion are characterized by the



7.2.1 The Crank-Slider Mechanism

following matrices:

$$\mathbf{M} = \begin{bmatrix} j_1 & 0 & 0 & 0 \\ 0 & j_2 & 0 & 0 \\ 0 & 0 & m & 0 \\ 0 & 0 & 0 & m \end{bmatrix} \tag{7.2.1}$$

$$\mathbf{u} = [\theta \;\; \phi \;\; x \;\; y]^T, \qquad \boldsymbol{\lambda} = [\lambda_1 \;\; \lambda_2 \;\; \lambda_3]^T \tag{7.2.2}$$

$$\mathbf{F} = [\tau \;\; 0 \;\; 0 \;\; -mg]^T \tag{7.2.3}$$

where **M**, **u**, $\boldsymbol{\lambda}$, and **F** denote the mass matrix, generalized coordinates, constraint forces and applied generalized force vector respectively. The kinematic constraint equations that define the revolute joint between the crank and connecting rod are expressed as follows with their corresponding constraint Jacobian matrix:

$$\boldsymbol{\Phi} = \begin{Bmatrix} r\cos\theta - (x - l_1\cos\phi) \\ r\sin\theta - (y - l_1\sin\phi) \\ (l - l_1)\sin\phi + y \end{Bmatrix} = 0 \tag{7.2.4}$$

$$\mathbf{B}^T = \begin{bmatrix} -r\sin\theta & r\cos\theta & 0 \\ l_1\sin\phi & l_1\cos\phi & (l - l_1)\cos\phi \\ -1 & 0 & 0 \\ 0 & -1 & 1 \end{bmatrix} \tag{7.2.5}$$

The connecting rod is originally placed in the horizontal position with a given torque that is applied at the crank. To carry out the numerical computation, the trapezoidal rule has been used to time-discretize the equations of motion. When the time increment $h = 0.01$ is used, it takes the time $T = 0.82$ second to complete one cycle of the mechanism.

Fig. 7.2.2 show the histories of the generalized coordinates in one cycle by using penalty constraint stabilization techniques. The penalty coefficient of the penalty constraint stabilization technique was chosen to be $\epsilon = 10^{-6}$. In order to compare the accuracy of the solutions to these dynamic equations, Baumgarte's technique [23,24] is selected to solve the same equations. Note that in order to obtain the same accuracy as in the penalty constraint stabilization technique, different combinations of $\alpha$ and $\beta$ have been tested. Figs 7.2.3a to 7.2.3b show that when $\alpha = \beta$ gradually increase from 70 to 275 and integration time step $h$ decrease from 0.01 to 0.0025, both stabilization techniques yield the same solutions.

In order to evaluate the performance of the two techniques from a different perspective, i.e., in terms of constraint violations, no iteration was performed at each integration time step. As time progresses, the three constraint conditions exhibited the same order of accuracy in each technique as shown in Fig. 7.2.4. Note that the error committed in this constraint condition for the penalty constraint stabilization technique remains about two digits lower than Baumgarte's technique over one cycle of run time.

Recently Haug and Yen [64] have proposed an implicit numerical integration algorithm via generalized coordinate partitioning technique to solve DAEs. Figs. 7.2.5 and 7.2.6 show the position error and velocity error of their solution procedure by solving present crank-slider mechanism. In order to compare these results, the two-stage staggered explicit-implicit algorithm is used to solve the same problem. Figs. 7.2.7 and 7.2.8 show the errors that are committed in computing positions and velocities of the mechanism are less than the algorithm proposed by Haug and Yen. Thus we conclude that the two-stage staggered explicit-implicit algorithm possesses the capability to improve the accuracy for the solution of DAEs.

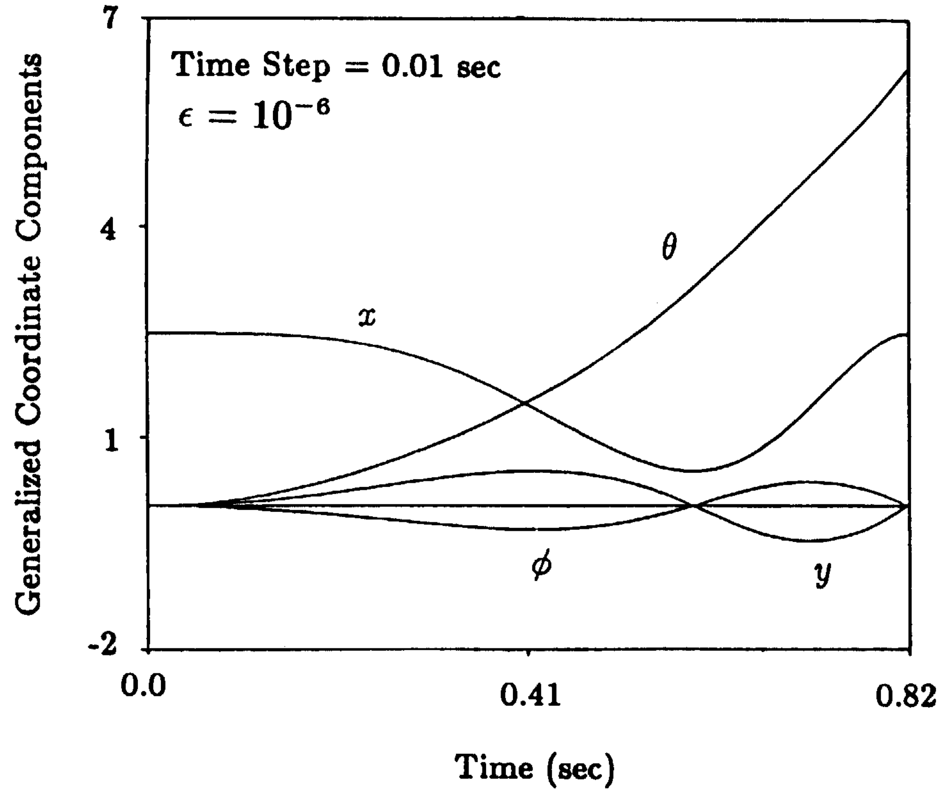
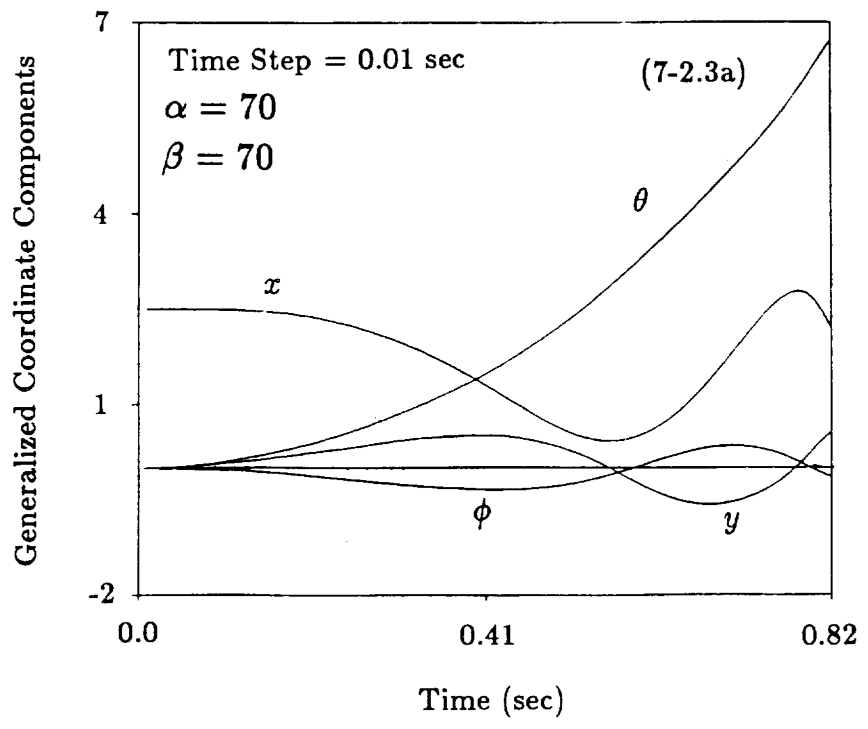


Fig. 7.2.2 Histories of the Generalized Coordinates: Penalty Constraint Stabilization Technique



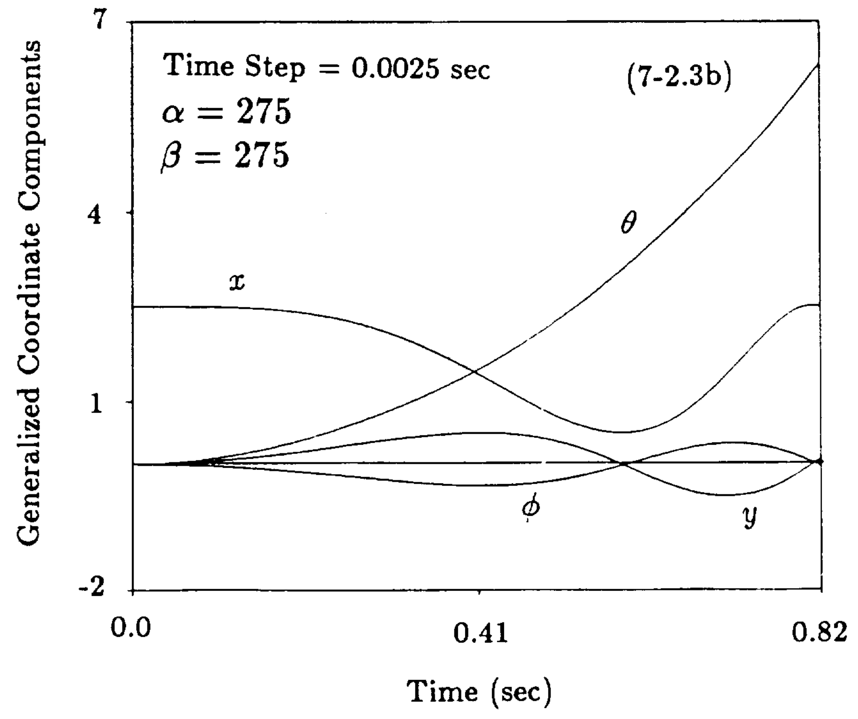
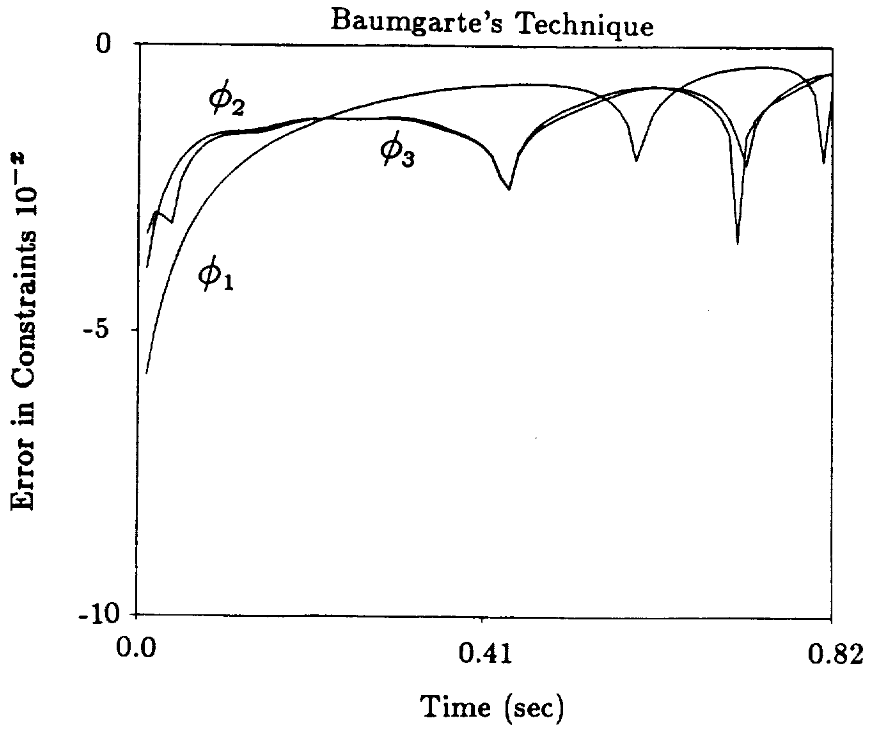


Fig. 7.2.3 Histories of the Generalized Coordinates: Baumgarte's Technique



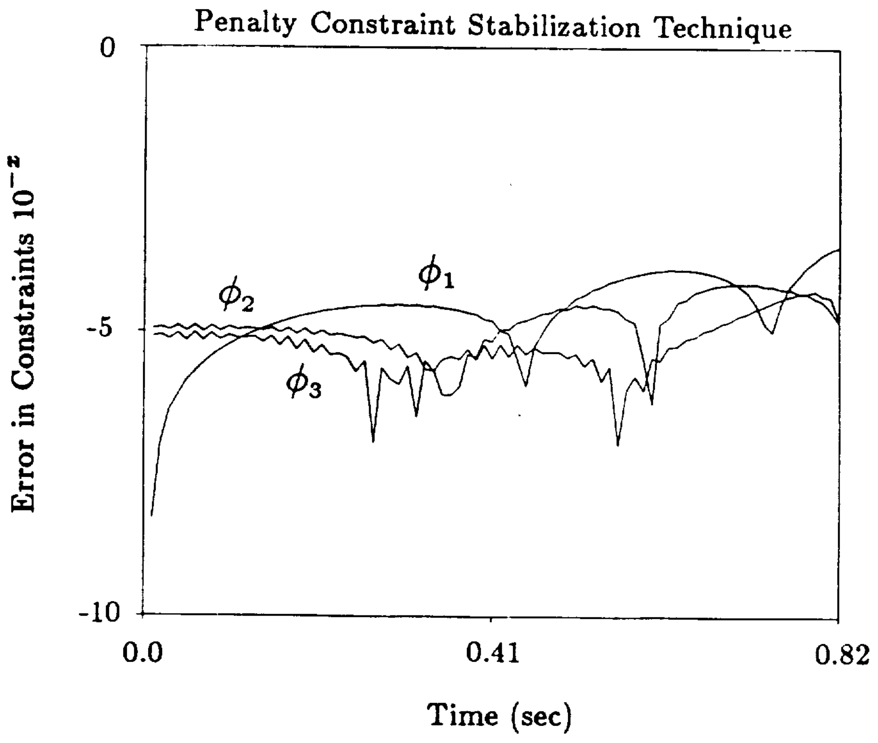


Fig. 7.2.4 Errors in Constraint Conditions, Performance of Two Techniques

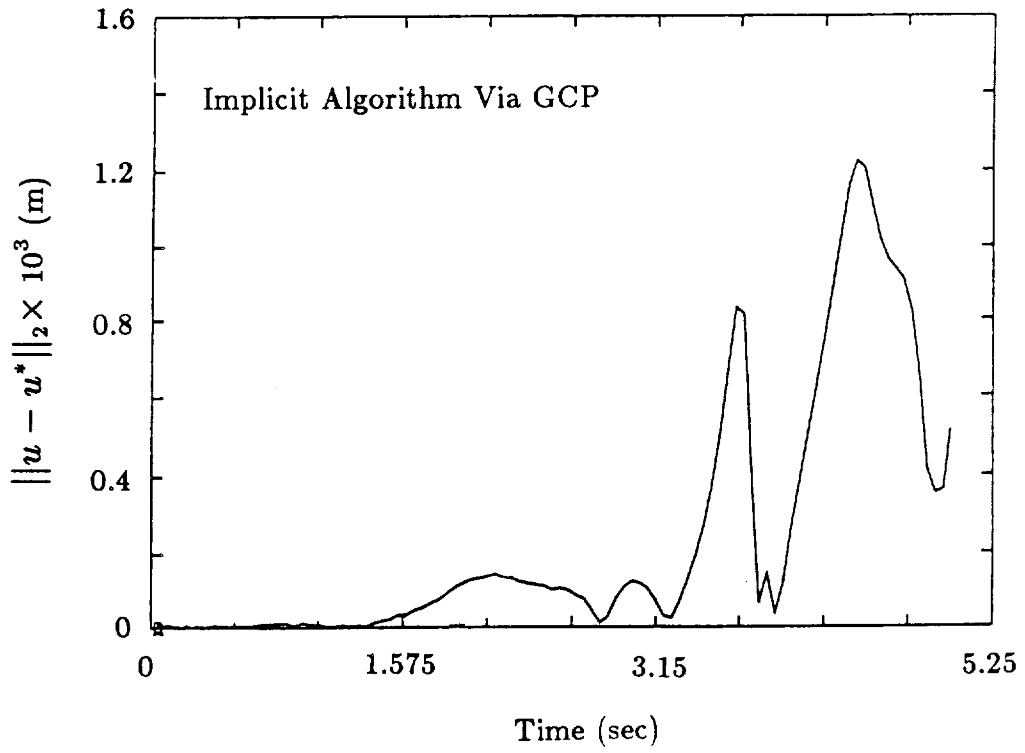


Fig. 7.2.5 Position Error [64], Time Step = 0.02 sec

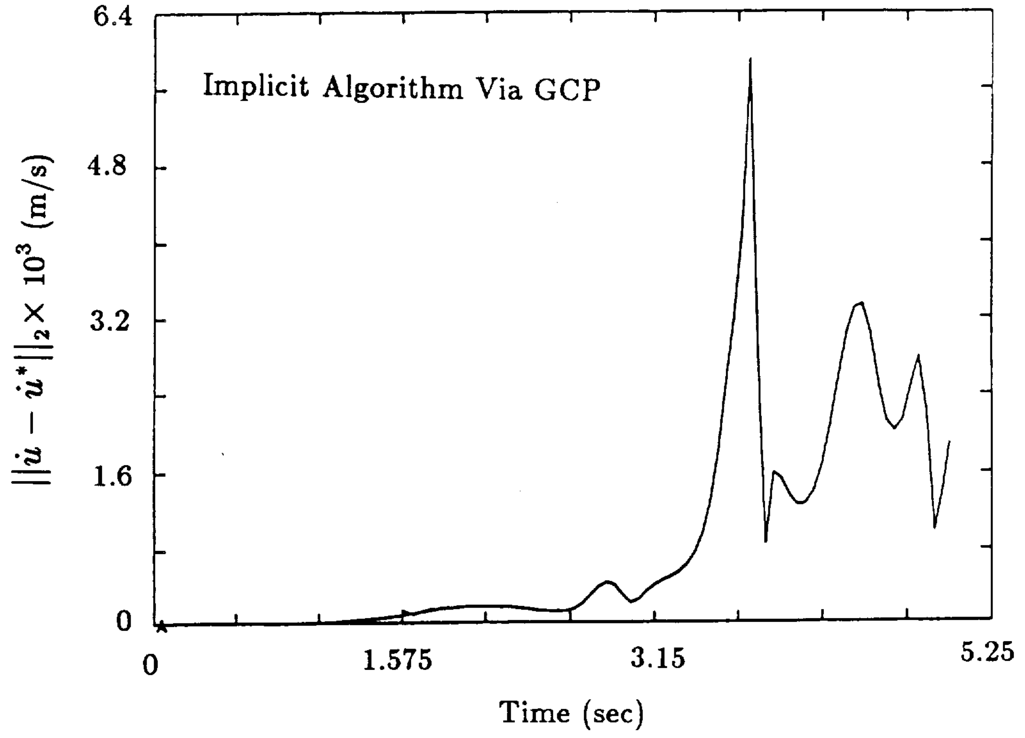


Fig. 7.2.6 Velocity Error [64], Time Step = 0.02 sec

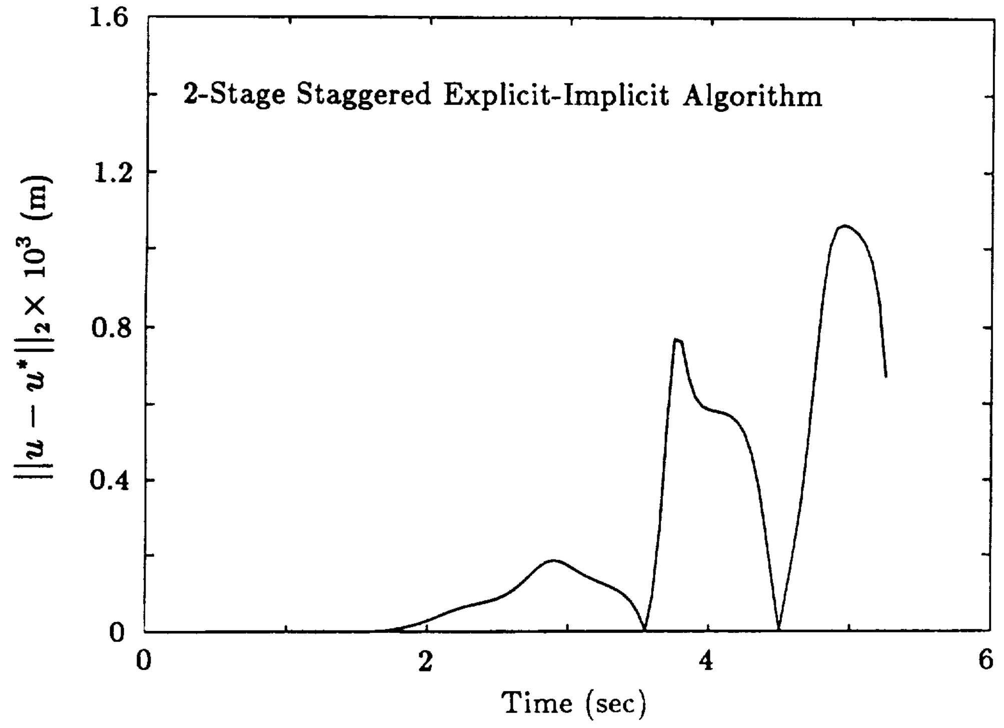


Fig. 7.2.7 Position Error, Time Step = 0.02 sec

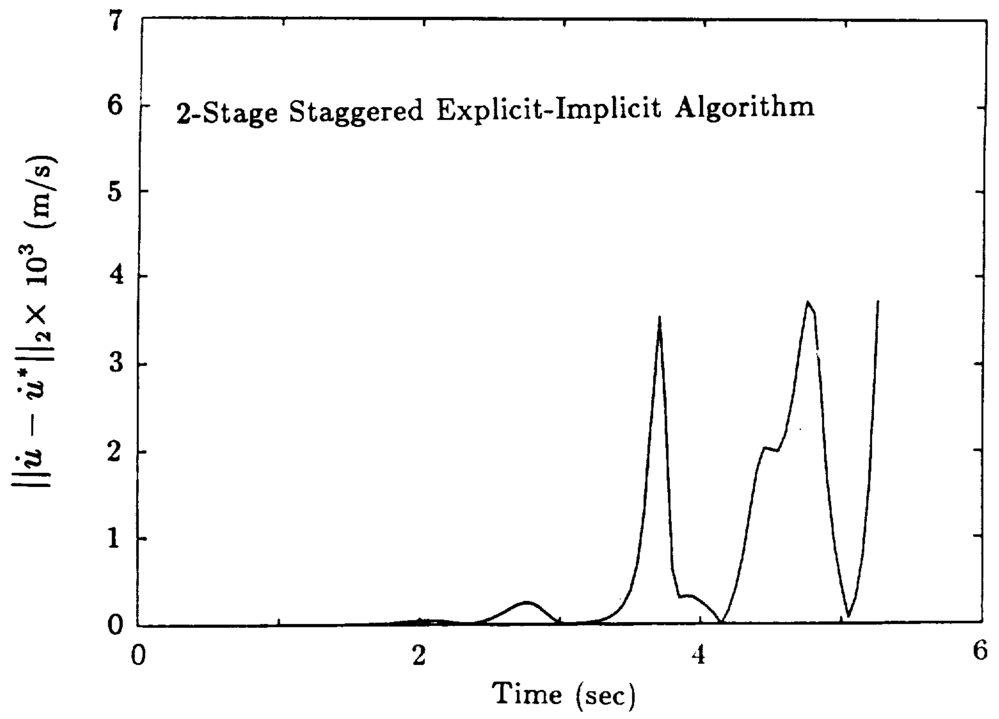


Fig. 7.2.8 Velocity Error, Time Step = 0.02 sec

## 7.3 Deployment of a Three-Link Manipulator

The second problem tested is a simplified version of the seven-link manipulator deployment problem. The three links are initially folded together with coil springs that are attached to each connecting joint. In order to make the link to deploy, a constant deploying force is then applied at the tip of the third link as shown in Fig. 7.3.1. The following quantities are obtained to characterized the corresponding equations of motion for the three-link manipulator:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} + \mathbf{B}^T\boldsymbol{\lambda} = \mathbf{F} \tag{7.3.1}$$

$$\boldsymbol{\Phi} = 0 \tag{7.3.2}$$

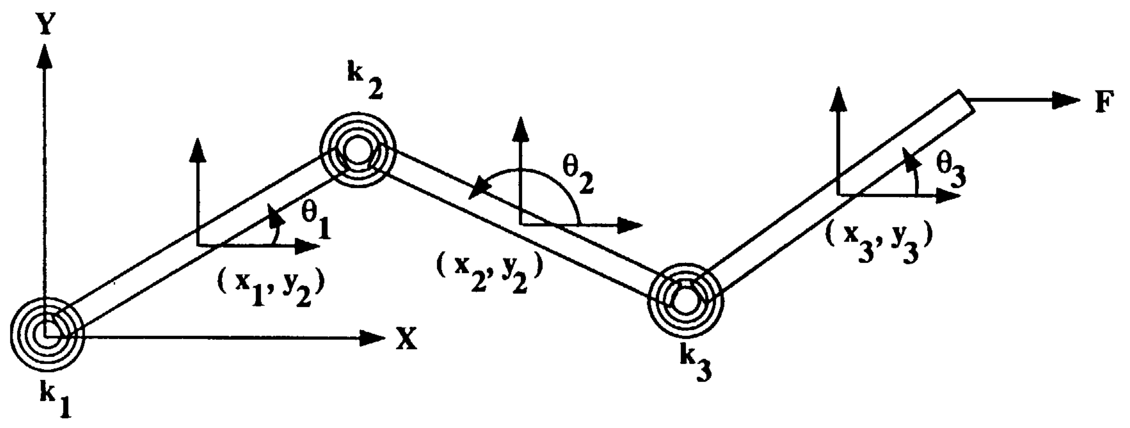
with

$$\mathbf{M} = \begin{bmatrix} \mathbf{j} & 0 & 0 \\ 0 & \mathbf{m}_x & 0 \\ 0 & 0 & \mathbf{m}_y \end{bmatrix} \tag{7.3.3}$$

$$\begin{aligned} \mathbf{j} &= diag[j_1 \quad j_2 \quad j_3] \\ \mathbf{m}_x &= diag[m_{x1} \quad m_{x2} \quad m_{x3}] \\ \mathbf{m}_y &= diag[m_{y1} \quad m_{y2} \quad m_{y3}] \end{aligned} \tag{7.3.4}$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{k}_\theta & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{7.3.5}$$

$$\mathbf{k}_\theta = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k_2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 \end{bmatrix} \tag{7.3.6}$$



7.3.1 Configuration of Three-Link Manipulator

$$\mathbf{u} = [\theta_1 \; \theta_2 \; \theta_3 \; x_1 \; x_2 \; x_3 \; y_1 \; y_2 \; y_3]^T \tag{7.3.7}$$

$$\boldsymbol{\lambda} = [\lambda_1 \; \lambda_2 \; \lambda_3 \; \lambda_4 \; \lambda_5 \; \lambda_6]^T \tag{7.3.8}$$

$$\mathbf{F} = [0 \; 0 \; 0 \; 0 \; 0 \; f \; 0 \; 0 \; 0]^T \tag{7.3.9}$$

and the constraint equations with their corresponding constraint Jacobian matrix:

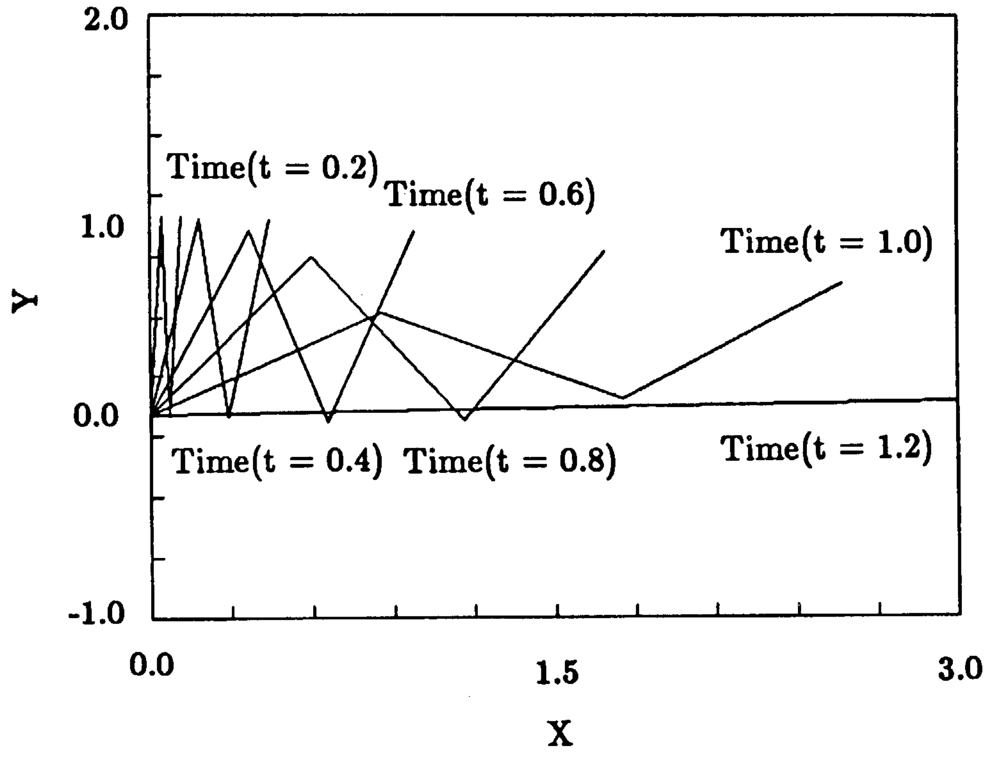
$$\boldsymbol{\Phi} = \left\{ \begin{array}{c} x_1 - \frac{l}{2}\cos\theta_1 \\ y_1 - \frac{l}{2}\sin\theta_1 \\ x_2 - x_1 - \frac{l}{2}\cos\theta_1 + \frac{l}{2}\cos\theta_2 \\ y_2 - y_1 - \frac{l}{2}\sin\theta_1 + \frac{l}{2}\sin\theta_2 \\ x_3 - x_2 + \frac{l}{2}\cos\theta_2 - \frac{l}{2}\cos\theta_3 \\ y_3 - y_2 + \frac{l}{2}\sin\theta_2 - \frac{l}{2}\sin\theta_3 \end{array} \right\} = 0 \tag{7.3.10}$$

$$\mathbf{B}^T = \begin{bmatrix} \frac{l}{2}\sin\theta_1 & -\frac{l}{2}\cos\theta_1 & \frac{l}{2}\sin\theta_1 & -\frac{l}{2}\cos\theta_1 & 0 & 0 \\ 0 & 0 & -\frac{l}{2}\sin\theta_2 & \frac{l}{2}\cos\theta_2 & -\frac{l}{2}\sin\theta_2 & \frac{l}{2}\cos\theta_2 \\ 0 & 0 & 0 & 0 & \frac{l}{2}\sin\theta_3 & -\frac{l}{2}\cos\theta_3 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.3.11}$$

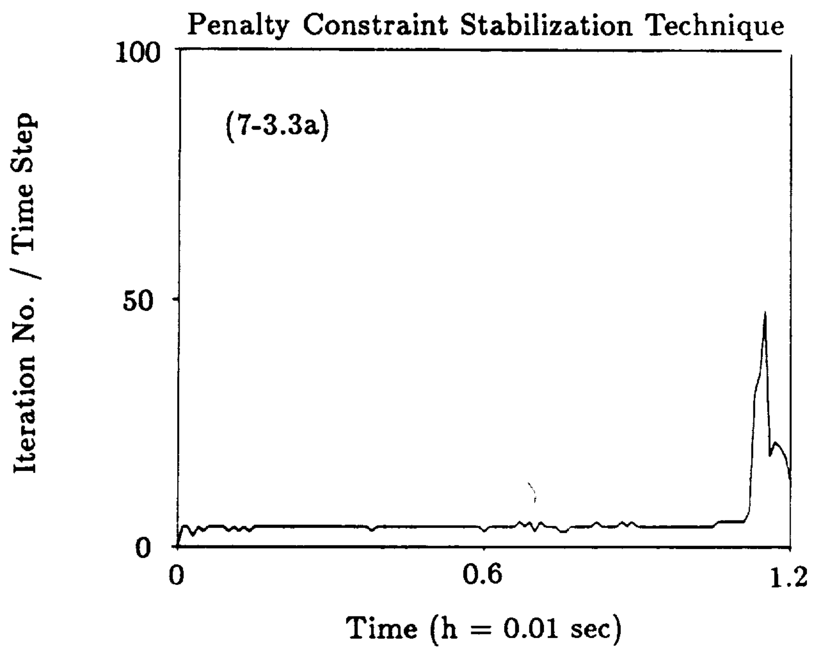
where *diag* denotes the diagonal terms of the representing matrices. A Newton-type iterative procedure with a specified accuracy criteria is employed to time-discretize both penalty constraint stabilization and Baumgarte's constraint stabilization techniques for the purpose that they can be assessed by the average number of iterations taken per time increment. The deployment sequence of the manipulator is illustrated in Fig. 7.3.2. With the accuracy of $10^{-6}$, the penalty constraint stabilization technique requires on the average about 4.5 iterations pre time increment (Fig. 7.3.3a), whereas Baumgarte's technique requires about 22 iterations per step (Fig. 7.3.3b).

An interesting aspect has been observed during the process of the links that are in straightening configuration (snap-through) where Baumgarte's technique fails to converge for time, $t \approx 1.1$, as manifested in Fig. 7.3.3b. This corroborates the prediction of the constraint forces where solution matrix $\mathbf{BM^{-1}B^T}$ for Baumgarte's technique becomes ill-conditioned. On the other hand, the penalty constraint stabilization technique still converges within 50 iterations (Fig. 7.3.3b) because of the existing $\dot{\lambda}$ in which overcomes the difficulty that has occurred in Baumgarte's technique. The property of overcoming the ill-conditioned $\mathbf{BM^{-1}B^T}$ has proven extremely useful. This is because during the dynamic simulation of any MBD system, an unknown position can be reached to cause the solutions of DAEs to numerically diverge.

From the example problems tested so far, we conclude that the penalty constraint stabilization technique yields both improved accuracy and computational robustness. In addition, the penalty constraint stabilization technique offers software modularity in that the solution of the constraint force $\boldsymbol{\lambda}$ can be carried out separately form that of the generalized coordinates $\mathbf{u}$. This can be accomplished by exchanging a set of vector plus the constraint Jacobian matrix for each solution module.



7.3.2 Deployment of Three-Link Manipulator



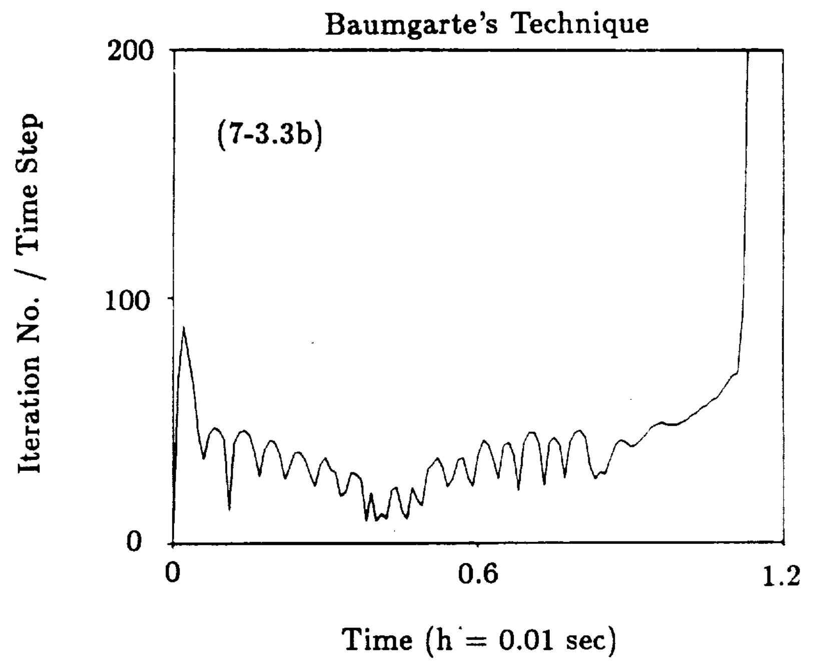


Fig. 7.3.3 Performance of Two Stabilization Techniques, Solution Accuracy = $10^{-6}$

## 7.4 Dynamics of a Bowling Ball

The study of the dynamics of a bowling ball was initiated by Houston *et al.* [65] whose equations of motion do not involve the constraint forces. In the present analysis, the equations of motion will adopt the form in (3.4.3) by incorporating both the holonomic and nonholonomic constraints as part of the system variables. Fig. 7.4.1 illustrates the geometry configuration of the bowling ball with a radius $a$ and an offset center $r_0$. The physical dimensions and initial conditions for the bowling ball are

$$m = 71.32\ N,\ a = 10.9\ cm,\ r_0 = 0\ or\ 0.15\ cm$$

$$J_1 = J_2 = J_3 = \frac{2}{5}ma^2,\quad \epsilon = 10^{-6}$$

$$x^0 = y^0 = 0,\quad q_0^0 = 1,\quad q_1^0 = q_2^0 = q_3^0 = 0$$

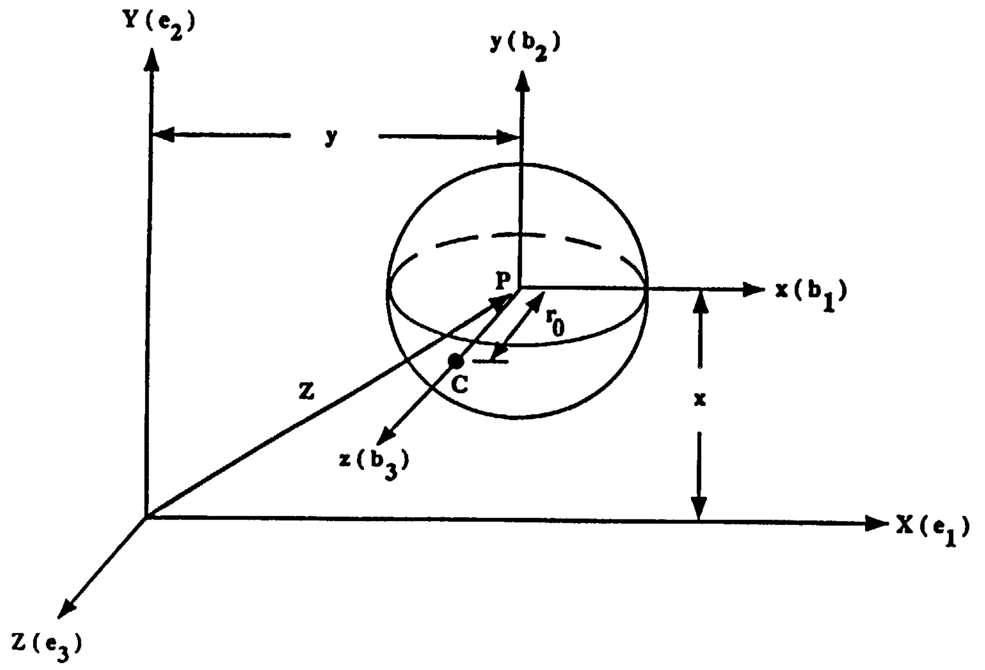
$$\omega_1 = -\omega_2 = -1,\quad \omega_3 = 0,\quad \dot{x}^0 = \dot{y}^0 = a\omega_1^0$$

The various matrices and vectors for the governing equations can be derived as

$$\mathbf{M} = \begin{bmatrix} m & 0 & -mr_0R_{12}^T & mr_0R_{11}^T & 0 \\ 0 & m & -mr_0R_{22}^T & mr_0R_{21}^T & 0 \\ -mr_0R_{12}^T & -mr_0R_{22}^T & J_1 & 0 & 0 \\ mr_0R_{11}^T & mr_0R_{21}^T & 0 & J_2 & 0 \\ 0 & 0 & 0 & 0 & J_3 \end{bmatrix} \tag{7.4.1}$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & -aR_{12} & -aR_{22} & -aR_{32} \\ 0 & 1 & aR_{11} & aR_{21} & aR_{31} \\ \cos x & -1 & 0 & 0 & \end{bmatrix} \tag{7.4.2}$$

$$\dot{\mathbf{u}} = [\dot{x}\ \ \dot{y}\ \ \omega_1\ \ \omega_2\ \ \omega_3]^T,\quad \lambda = [\lambda_1\ \ \lambda_2\ \ \lambda_3]^T \tag{7.4.3}$$



7.4.1 Solid Spherical Ball Rolling on a Flat Surface

$$\mathbf{F}_d = -mr_0 \begin{Bmatrix} \omega_1\omega_3 R^T_{11} + \omega_2\omega_3 R^T_{12} - (\omega_1^2 + \omega_2^2) R^T_{13} \\ \omega_1\omega_3 R^T_{21} + \omega_2\omega_3 R^T_{22} - (\omega_1^2 + \omega_2^2) R^T_{23} \end{Bmatrix} \tag{7.4.4}$$

$$\mathbf{F}_\omega = - \begin{Bmatrix} \omega_2\omega_3(J_2 - J_3) \\ \omega_3\omega_1(J_3 - J_1) \\ \omega_1\omega_2(J_1 - J_2) \end{Bmatrix} \tag{7.4.5}$$

$$\mathbf{f}_d = 0, \quad \mathbf{f}_\omega = mgr_0 \begin{Bmatrix} R^T_{32} \\ -R^T_{31} \\ 0 \end{Bmatrix} \tag{7.4.6}$$

$$F = \begin{Bmatrix} \mathbf{F}_d + f_d \\ \mathbf{F}_\omega + f_\omega \end{Bmatrix} \tag{7.4.7}$$

where the corotational basis vector **b** and the inertial basis vector **e** are related according to

$$\mathbf{b} = \mathbf{R}\mathbf{e} \tag{7.4.8}$$

The holonomic constraint condition requires the bowling ball to follow a sine curve,

$$\Phi = y - \sin x = 0 \tag{7.4.9}$$

The nonholonomic constraints are obtained by requiring the contact point of the bowling ball to maintain the no-slipping conditions where

$$\dot{x} = (\boldsymbol{\omega} \times r_0 e_3) \cdot e_1 \tag{7.4.10}$$

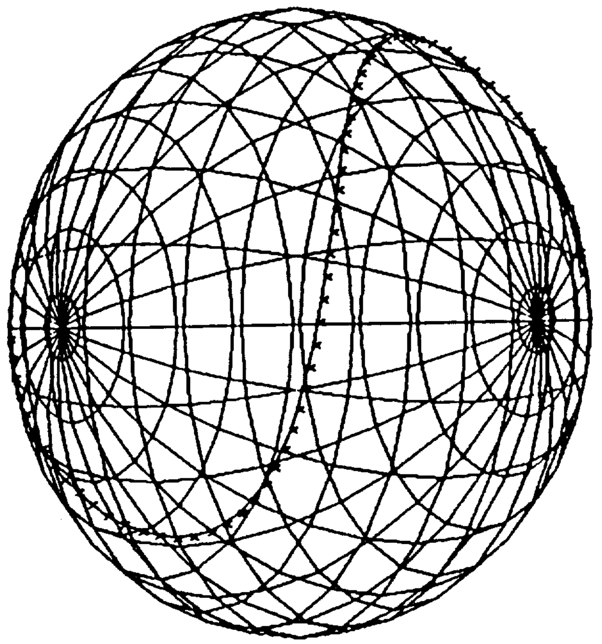
$$\dot{y} = (\boldsymbol{\omega} \times r_0 e_3) \cdot e_2 \tag{7.4.11}$$

The numerical algorithm that is based on the two-stage staggered explicit-implicit algorithm is used to integrate these nonlinear dynamic equations with a 40 second simulation time. The ball track that follows the sinusoidal curve is projected back on the ball itself as shown in Fig. 7.4.2. The simulations are tested for two cases: no offset (center of mass is located at the center of geometry) and offset (center of mass is not located at the center of geometry) of the bowling ball.

For the no offset case, Fig. 7.4.3 shows the angular velocities of the bowling ball during the 40 second run time. As expected, the angular velocities $\omega_1$ and $\omega_2$ show the periodic nature similar to a sine curve. Fig. 7.4.4 illustrates the histories of the three constraint forces that require the bowling ball to follow its course. The constraint forces $\lambda_1$ and $\lambda_2$ are used to show how the rolling contact conditions in the $x$ and $y$ directions are maintained. Whereas $\lambda_3$ provides the force that is needed to maintain the imposed sinusoidal trajectory. Hence, we conclude that the first two constraint forces are used to preserve the no-slipping conditions at the contact point and the third constraint force which corresponds to the steering force, is used to maneuver the ball.

For the offset case, Fig. 7.4.5 shows the angular velocities of the ball no longer exhibit the same periodic behavior as the no-offset case. However, the trend of the curves still follow the periodic nature of a sine curve. Likewise, the direction and steering forces in Fig. 7.4.6 become highly nonlinear with nonlinearly periodic behavior.

Convergence studies have been performed with increasing time step for the present two-stage staggered explicit-implicit algorithm. When the time step remains in the range of $h \leq 0.15$, the present algorithm maintains



7.4.2 Ball Track Projected on Three-Dimensional Sphere Surface

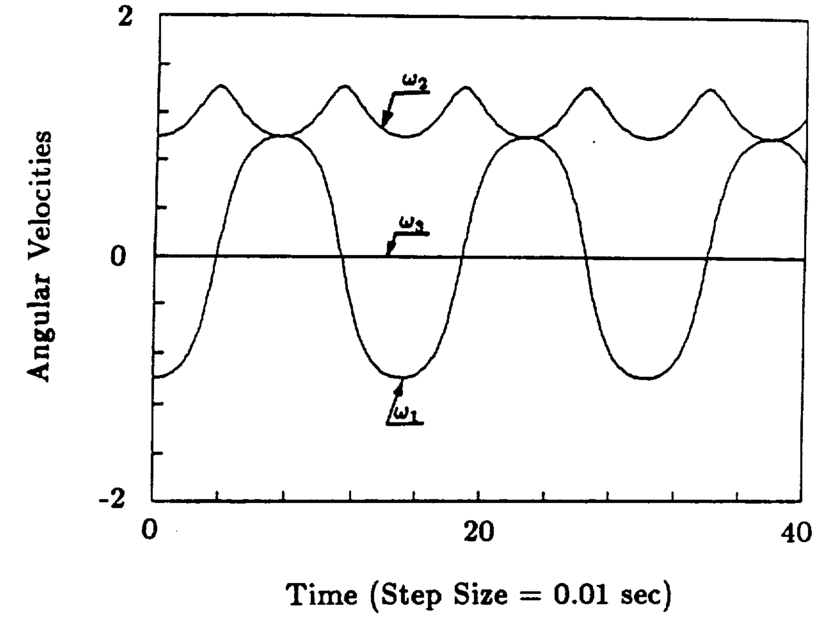


Fig. 7.4.3 Angular Velocities of the Sphere with No Offset

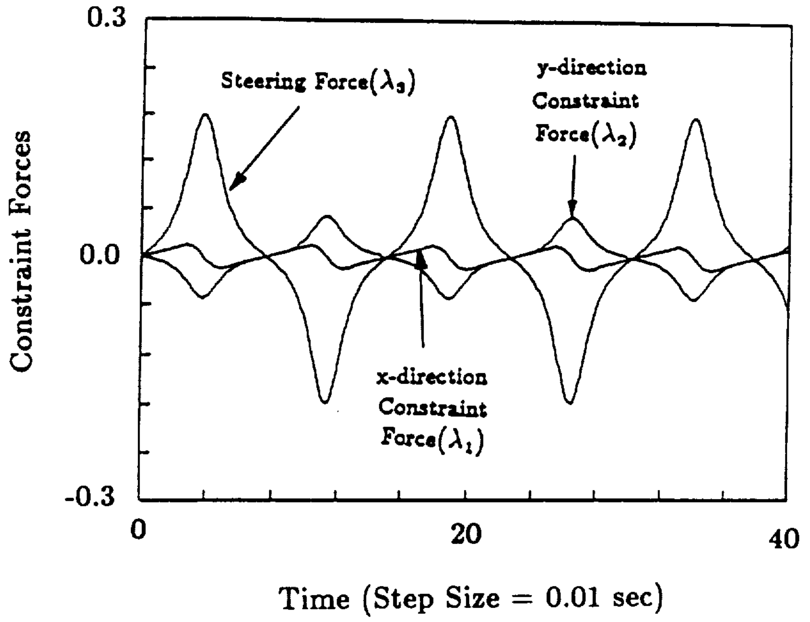


Fig. 7.4.4 Time Histories of Three Constraint Forces with No Offset

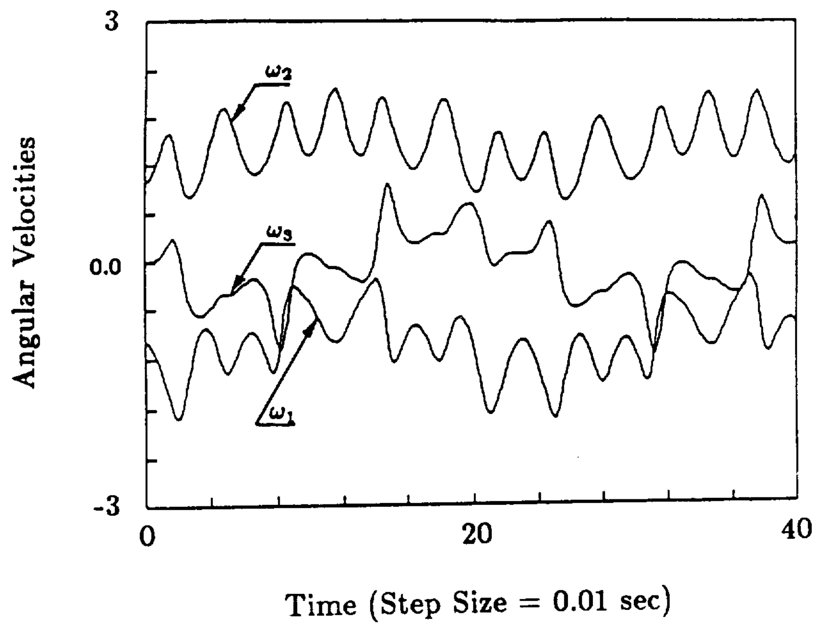


Fig. 7.4.5 Angular Velocities of the Sphere with Offset

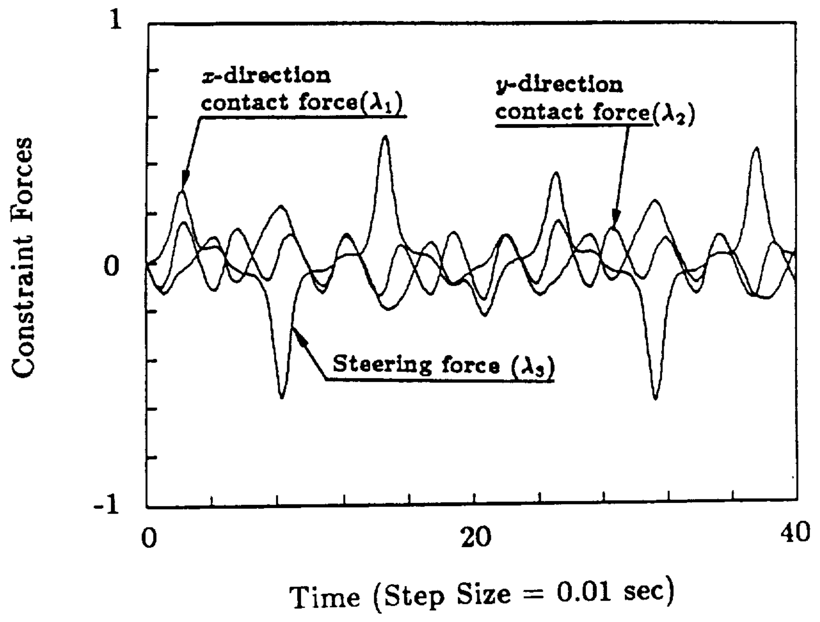


Fig. 7.4.6 Time Histories of Three Constraint Forces with Offset

both solution accuracy and stability.

To manifest the instability of the conventional approximation for the velocity dependent terms as alluded to in section 5.8.2, the following equation is used to integrate the equations of motion of the bowling ball:

$$\omega^{n+\frac{1}{2}} = \omega^{n-\frac{1}{2}} + h\dot{\omega}^n \tag{7.4.12}$$

where $\dot{\omega}^n$ is obtained by using

$$\dot{\omega}^n = \mathbf{J}^{-1}(\omega^n \times \mathbf{J}\omega^n + \mathbf{F}(\omega^n)) \approx \mathbf{J}^{-1}(\omega^{n-\frac{1}{2}} \times \mathbf{J}\omega^{n-\frac{1}{2}} + \mathbf{F}(\omega^{n-\frac{1}{2}})) \tag{7.4.13}$$

Fig. 7.4.7 illustrates angular velocity $\omega_2$ vs time for the converged solution, the present two-stage staggered explicit-implicit procedure with time step $h = 0.2$, and the conventional procedure with time step $h = 0.2$. Clearly, the conventional procedure begins to diverge after simulation time approaches 4 seconds, thus the instability of the conventional procedure is been confirmed. On the other hand, the two-stage staggered explicit-implicit procedure traces the converged solution faithfully during the 40 second simulation time.

Finally, the solution accuracy versus the time stepsize has been assessed for the offset center case with step sizes $h = 0.01, 0.2, 0.4$. The accuracy performance of the present procedure for different step sizes is given in Fig. 7.4.8 which provides the following guideline: in order to maintain a reasonable engineering accuracy, the step size should be confined to $h \leq 0.2$. The results given in the present section shows that the present computational procedure handles the large rotational and translational motions with robustness and efficiency.

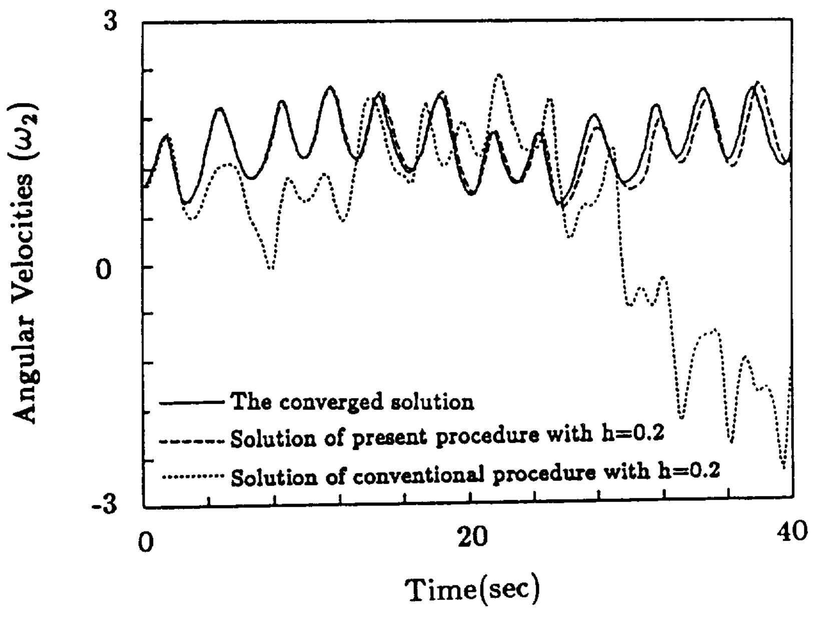


Fig. 7.4.7 Convergence Studies on Present and Conventional Procedure

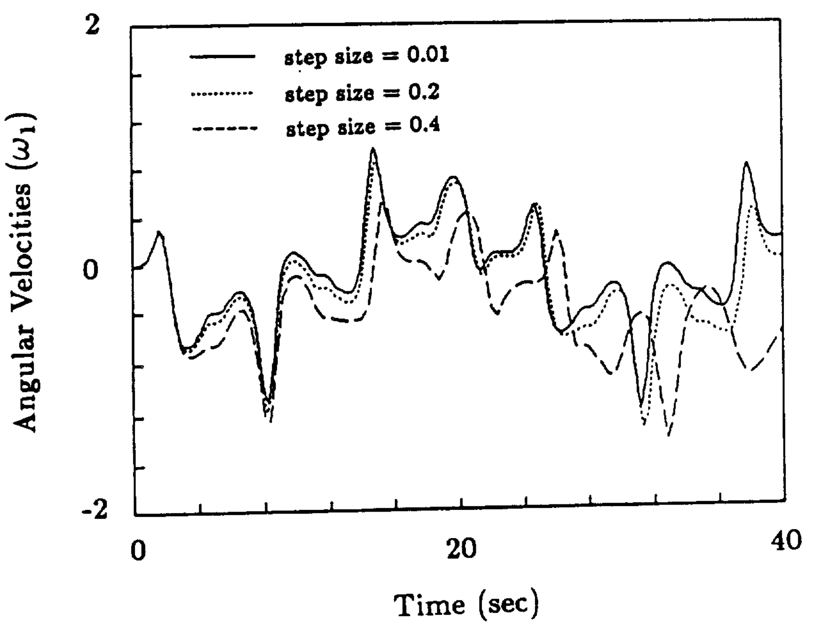


Fig. 7.4.8 Accuracy Comparison on Angular Velocity $\omega_1$ for Three Different Time Steps

## 7.5 Dynamic Simulation of a Closed Four-Bar Linkage

To examine different system topologies, a simple closed four-bar linkage (Fig. 7.5.1), composed of four individual bars connected with five spherical joints, is used to demonstrate the effectiveness of the proposed equations of motion. The governing equations of motion for this problem are identical with those of the previous problems, except that the constraint Jacobian matrix, **B**, that is given by

$$\mathbf{B} = \begin{bmatrix} B_r^{(1)} & 0 & 0 & 0 \\ B_l^{(1)} & B_r^{(2)} & 0 & 0 \\ 0 & B_l^{(2)} & B_r^{(3)} & 0 \\ 0 & 0 & B_l^{(3)} & B_r^{(4)} \\ 0 & 0 & 0 & B_l^{(4)} \end{bmatrix} \tag{7.5.1}$$

where $r$ and $l$ denote the left and right hand side of (4.2.10). Note that the present equations of motion can be directly integrated by using the penalty constraint stabilization technique, whereas the equations of motion that are derived by using relative coordinates require special methodology to identify system independent and dependent variables so that numerical algorithms can be applied to obtain the solutions.

In order to trigger large rotational motions, two vertical forces $F_y^{(1)} = F_y^{(4)} = 1$ are applied at the center of mass of the first and fourth bars. Fig. 7.5.2 shows the motion of each bar during the 8 seconds simulation time where the trajectories of each spherical joint can be seen explicitly. Due to the symmetry of the geometry and applied forces, the corresponding symmetries between angular velocities (Fig. 7.5.3) and the constraint forces (Fig. 7.5.4) of the first bar compared with those of the fourth bar, and so on are expected.

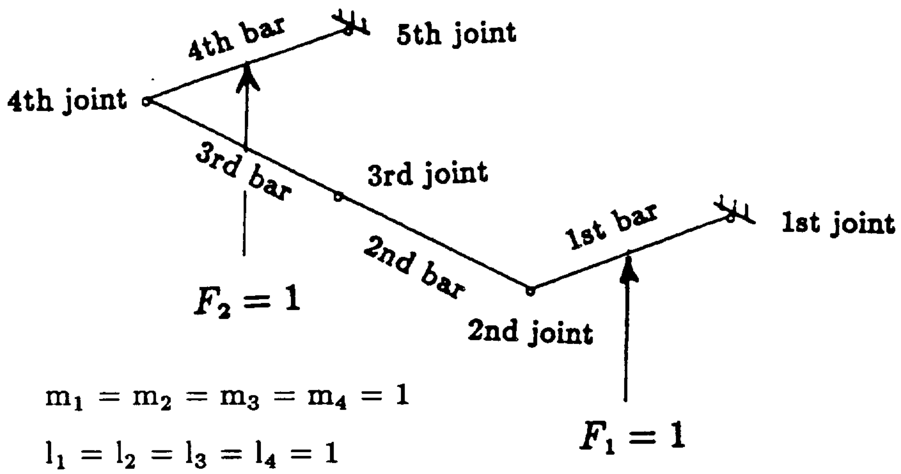


Fig. 7.5.1 Initial Configuration of a Four-Bar Linkage

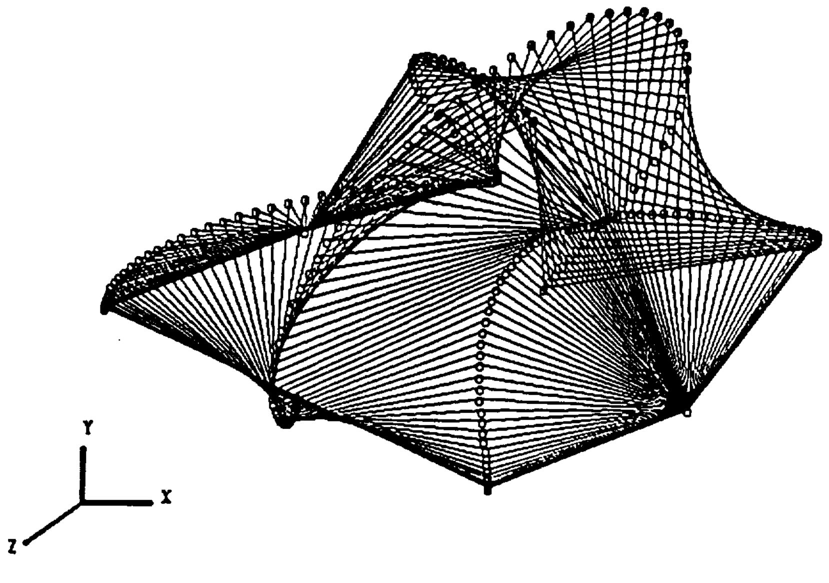
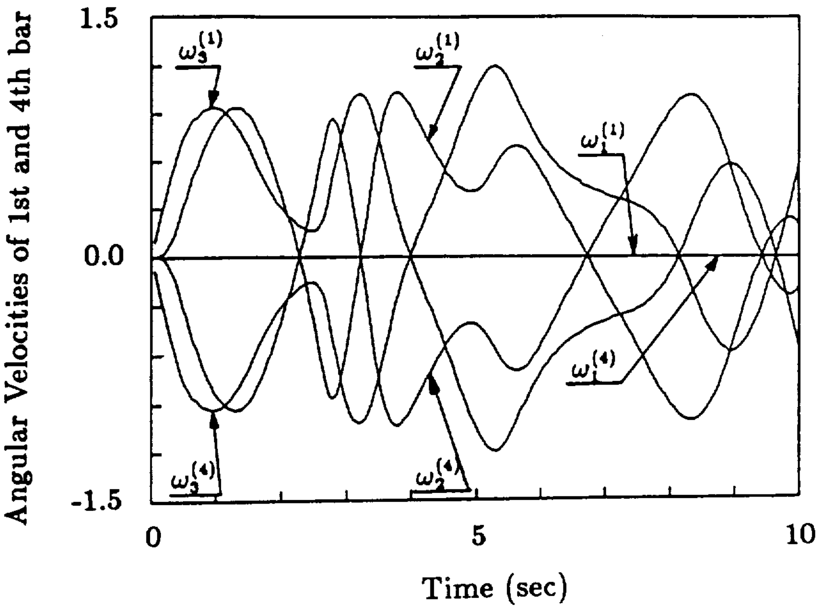


Fig. 7.5.2 Motion and Trajectories of the Four-Bar Linkage



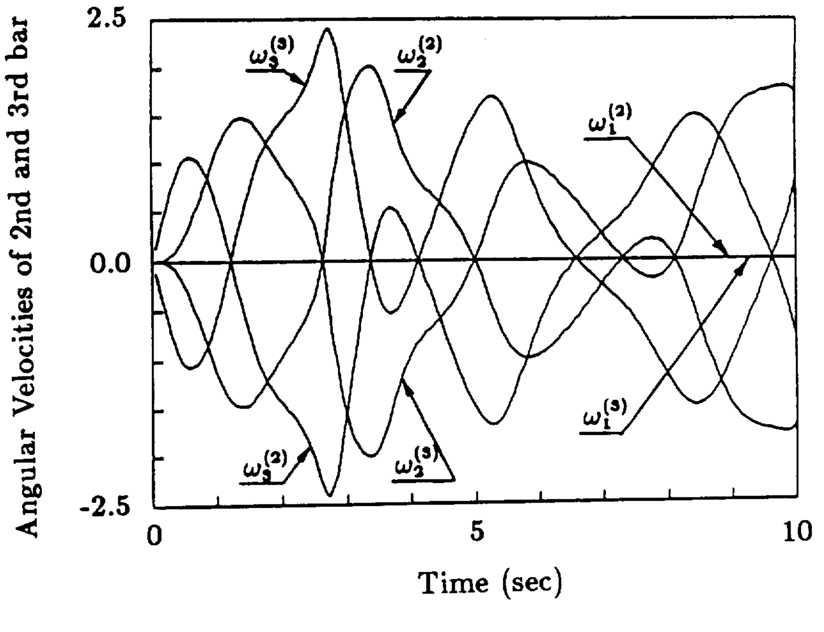
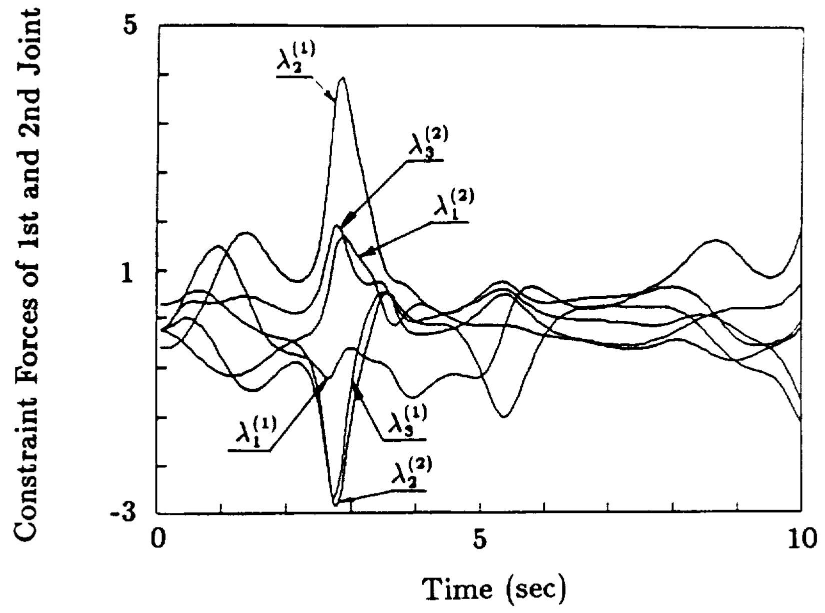


Fig. 7.5.3 Angular Velocities of the Four-Bar Linkage



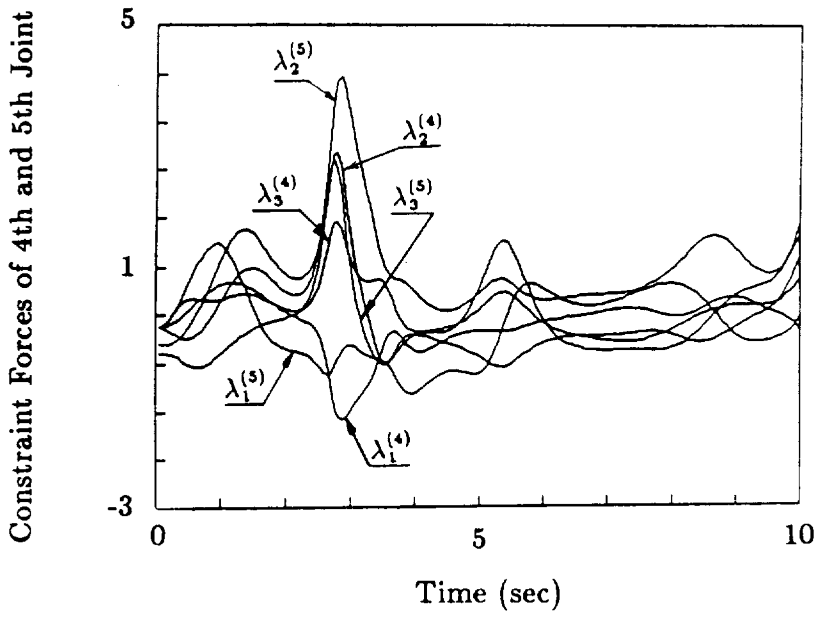


Fig. 7.5.4 Constraint Forces of the Four-Bar Linkage

## 7.6 Dynamic Simulation of a Space Crane

The dynamics of rigid space crane models and their inverse kinematics [66], vibration characteristics of selected crane configuration [67], and control of crane imperfections by adaptive elements [68] have been studied by several researchers. however, the transient dynamics of space crane including the flexible vibration effects has very little been reported.

To sufficiently model the flexibility of the space crane, a formulation based on a fully nonlinear continuum approach [52] has been developed and allowed large rotations and deformations. In this development, we model the space crane by using three-link flexible beams maneuvering under a specified nonholonomic tip velocity constraint. Three spherical joints are used to connect the links with the Lagrange multipliers that have been introduced to enforce the nonholonomic constraint at the third manipulator tip as well as the holonomic joint constraints. The trajectories of the rigid and flexible bodies and the tip velocity specification are given in Fig. 7.6.1 and Fig. 7.6.2. The corresponding joint torques for the rigid and flexible links are also given in Figs. 7.6.3 and 7.6.4. Note that there exists little difference in the two trajectories between the rigid and the flexible cases as shown in Fig. 7.6.1, however the significant differences in the joints torques will play an important role in the design of the vibration control.

In order to validate the feasibility, effectiveness, and accuracy of the present schemes, the three-link manipulator model has been applied to the three dimensional rigid body dynamic modeling of space crane for control design and analysis. The dynamic analysis of the space crane problem was initiated by Gawronski and Ih [66] who have provided the initial configuration and mass distribution of the space crane. In order to maneuver the

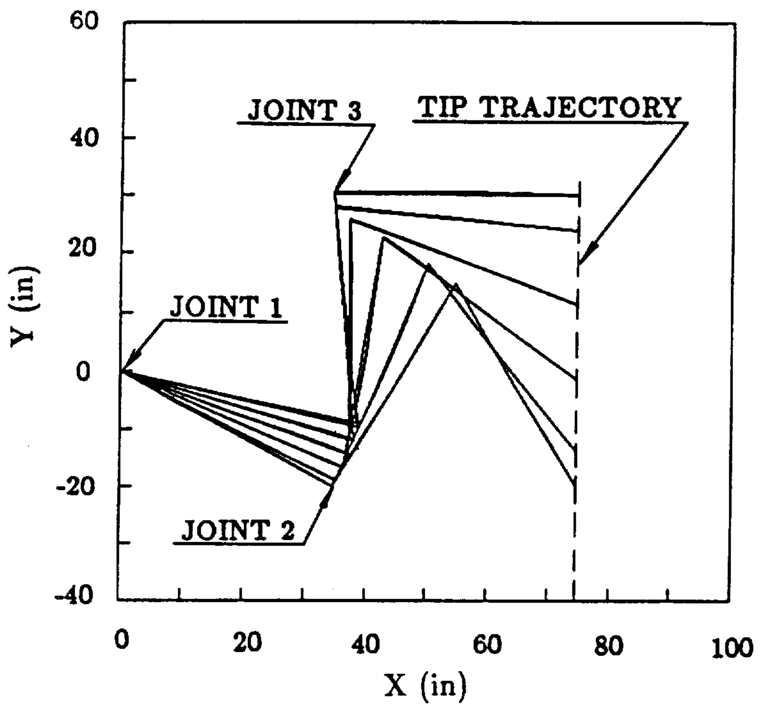


Fig. 7.6.1 Crane Tip Trajectory of Rigid and Flexible Members

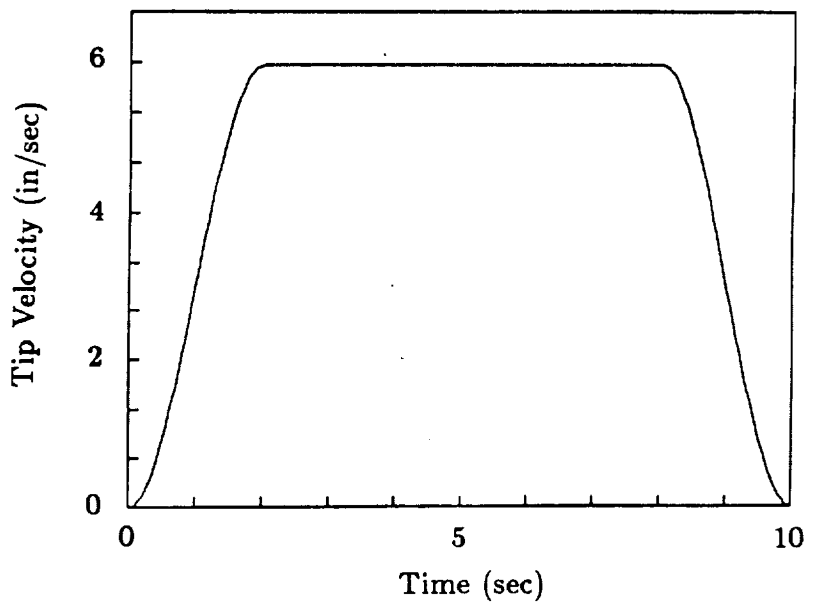


Fig. 7.6.2 Specified Crane Tip Velocity of Rigid and Flexible Members

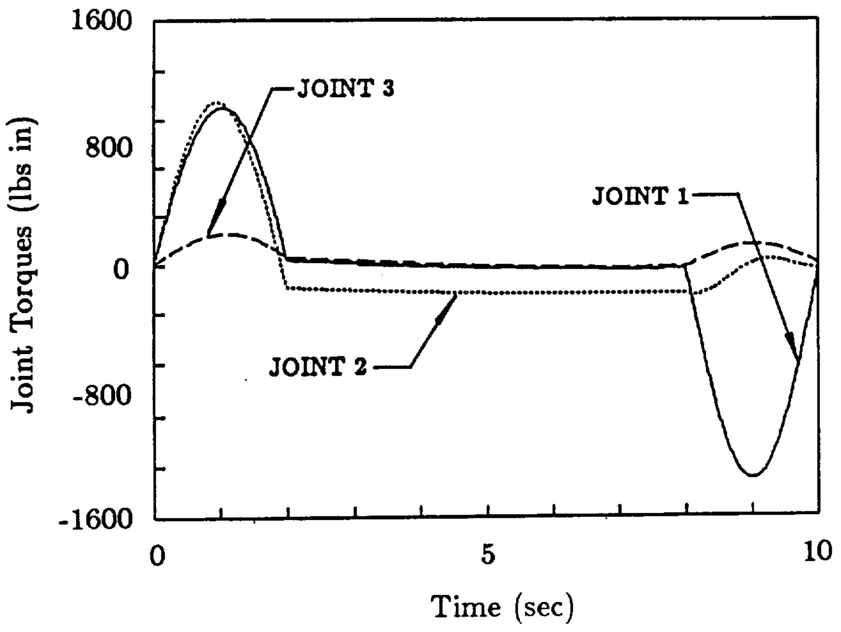


Fig. 7.6.3 Crane Joint Torques (Rigid Members) .vs. Time

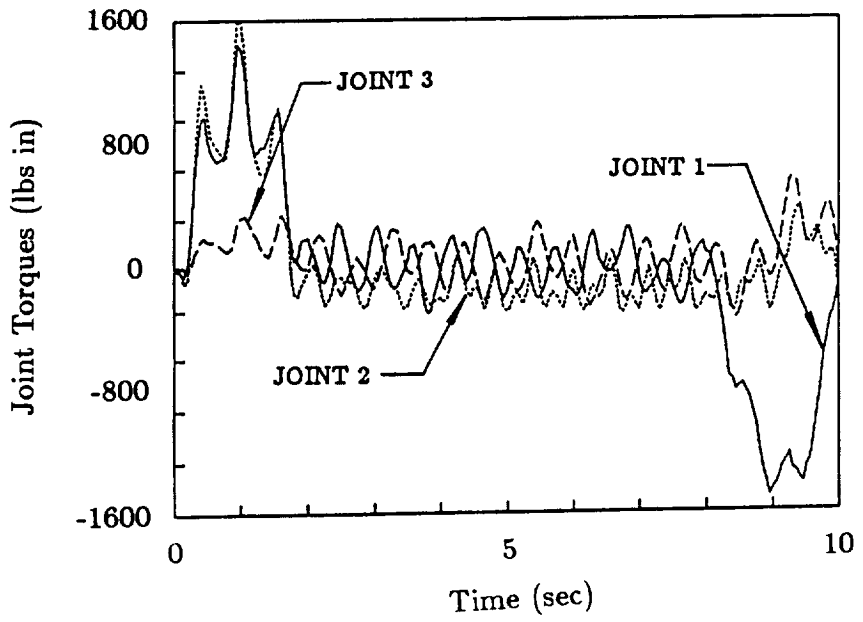


Fig. 7.6.4 Crane Joint Torques (Flexible Members) .vs. Time

space crane from one position to another position in space (Fig. 7.6.5), a holonomic constraint at the y-direction on the tip of the third link is imposed as follows:

$$y_t(t) = \left\{ \begin{array}{cc} 0.5[t - \omega^{-1}\cos(t_1)]v_0 & 0 \le t \le t_0 \\ y(t_0) + (t - t_0)v_0 & t_0 < t \le T - t_0 \\ y(T - t_0) + 0.5[t - T + t_0 - \omega^{-1}\cos(t_2)]v_0 & T - t_0 < t \le T \end{array} \right\} \tag{7.6.1}$$

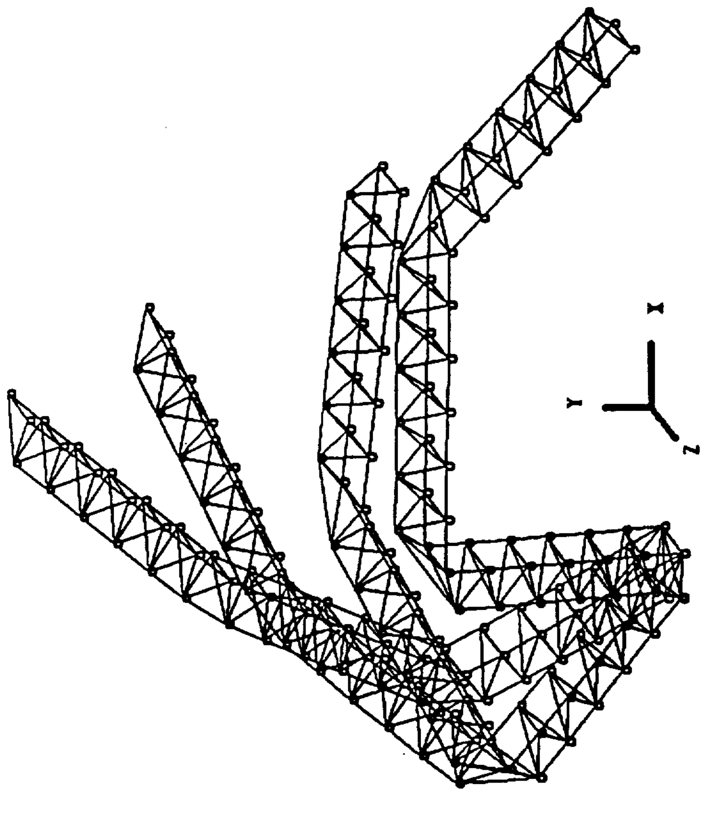
where $T$ is the total time of the tip movement, $t_0$ is the acceleration time, $\omega = \pi/t_0$, and $v_0$ is the maximal tip velocity, $\cos(t_1) = \cos(\omega t - 0.5\pi)$, and $\cos(t_2) = \cos(\omega t - 1.5\pi)$. The tip velocity, $v_y$, is obtained by taking time differentiation of (7.6.1) as

$$v_y(t) = \left\{ \begin{array}{cc} 0.5[1 + \sin(\omega t - 0.5\pi)]v_0 & 0 \le t \le t0 \\ v_0 & t_0 < t \le T - t_0 \\ 0.5[1 + \sin(\omega t - 1.5\pi)]v_0 & T - t_0 < t \le T \end{array} \right\} \tag{7.6.2}$$

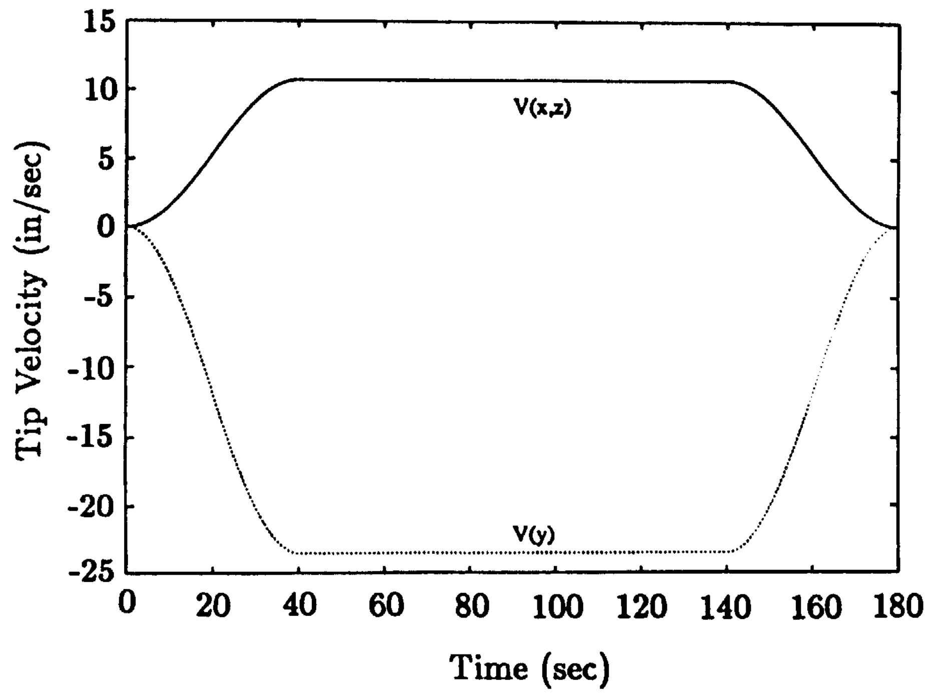
The final velocity constraints on $x$, $y$, $z$ (Fig. 7.6.6), and $\theta$ is obtained by

$$\begin{aligned} v_x(t) &= -0.454545 v_y(t) \\ v_z(t) &= -0.454545 v_y(t) \\ v_\theta(t) &= 0.000634665 v_y(t) \end{aligned} \tag{7.6.3}$$

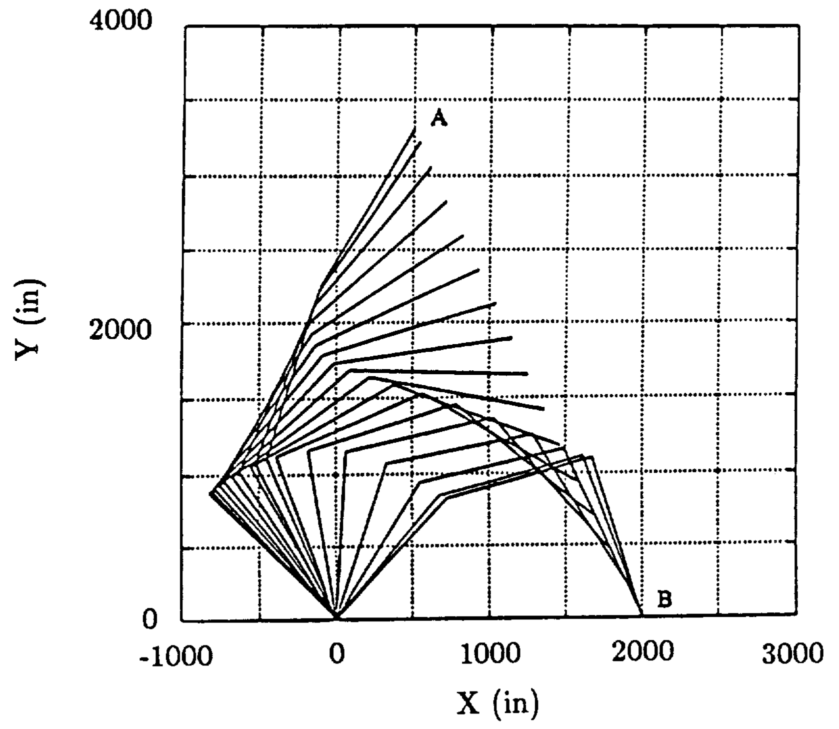
By adopting a previously developed three-link manipulator model, the space crane configurations that have been projected on the x-y plane and z-y plane during the 180 seconds simulation time are given in Figs. 7.6.7. Figs. 7.6.8 and 7.6.9 show the joint velocities, and joint torques of the space crane.



7.6.5 Crane Configuration and Subsequent Motion



7.6.6 Specified Crane Tip Velocities of Rigid and Flexible Members



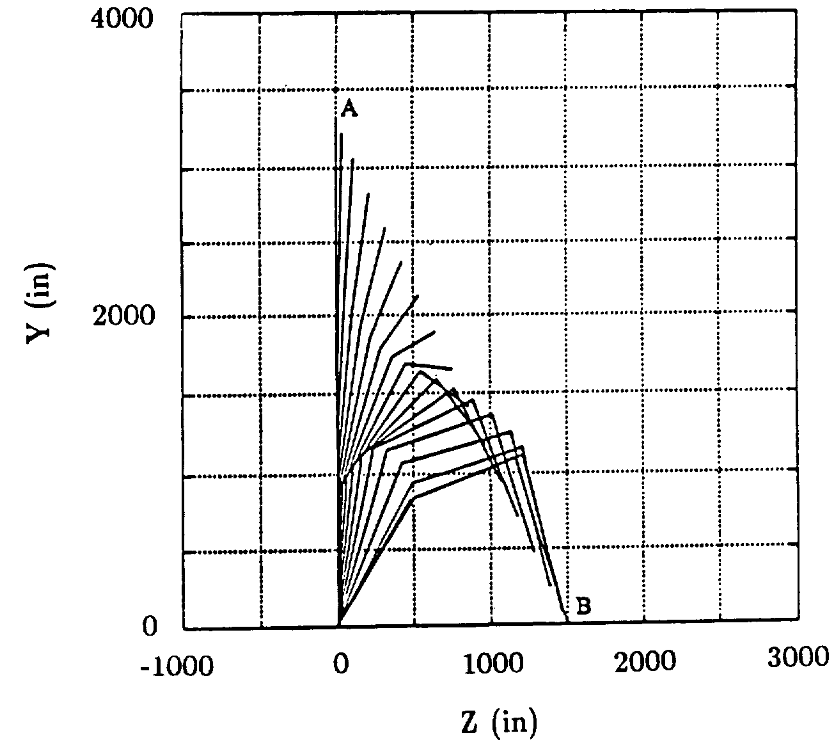


Fig. 7.6.7 Crane Trajectory: X - Y Plane and Z - Y Plane

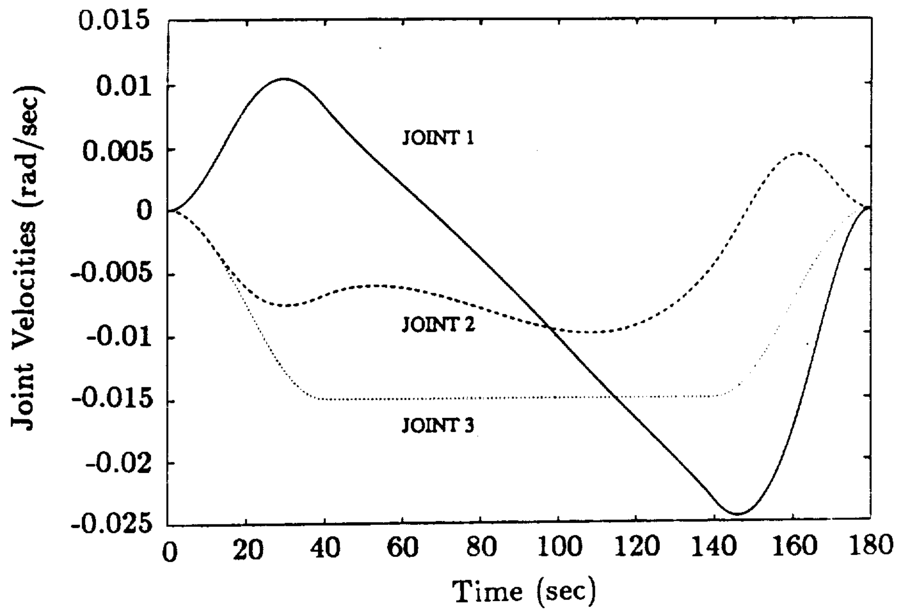


Fig. 7.6.8 Joint Angular Velocities (Rigid Members)

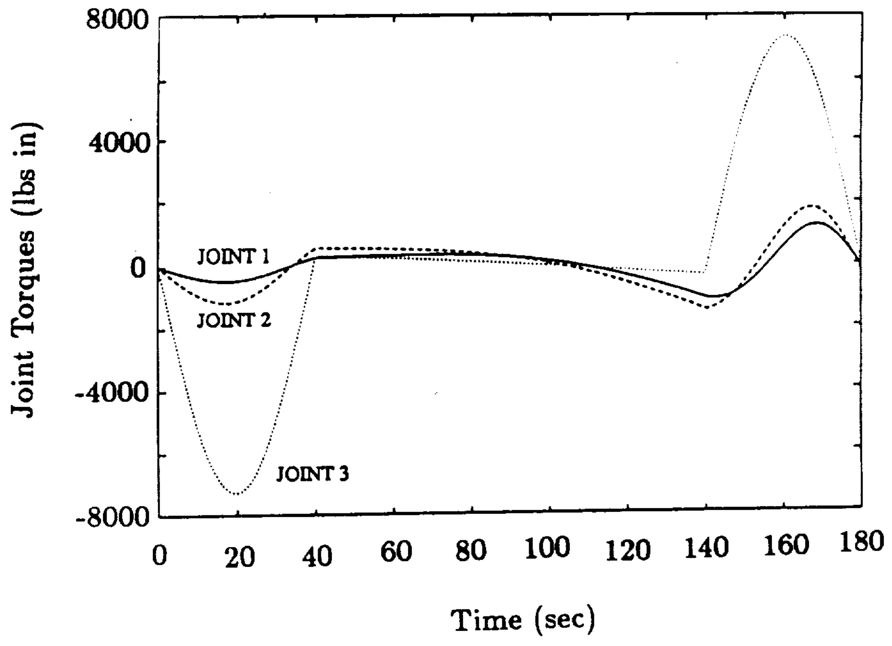


Fig. 7.6.9 Time History of Joint Torques (Rigid Members)

During the one on one comparison with the solutions given by Gawronski and Ih, the joint velocity and acceleration curves exhibit the same behaviors. Note that Gawronski and Ih's formulation are based on relative coordinates which are derived by Craig [69] whose formulation can only be applied to single open chain dynamic systems. Whereas in DAEs as previously derived, regardless system topologies and their given time dependent constraints, the solution procedure can equally be applied to different types of constraints in which the versatility of present general-purpose computer program to handle different MBD problem has been emphasized.

Finally, the flexible crane has been analyzed. Each arm is modeled as a spatial continuum beam whose material and equivalent geometrical quantities are chosen such that their fundamental frequencies match closely that analyzed by Sutter *et al.* [67] by the finite element truss models. The angular velocities and the joint torques are shown in Figs. 7.6.10 and 7.6.11. Note that the effect of flexibility is clearly manifested in the high oscillatory responses and the large stopping torques. Such large stopping torque requirements are in contrast to the zero torque at the end of the maneuvering in the case of rigid models.

The application of the developed software to the space crane problem indicates that, while rigid models can be analyzed with sufficient confidence and computational efficiency, the case of flexible models pose many unanswered difficulties. Specifically, it appears that no unique inverse dynamic analysis technique is available for the case of the flexible models. In addition, it is dangerous to use the maneuvering strategy developed based on the rigid models while flexible models may experience unwanted large stopping joint torques as shown in Fig. 7.6.11.

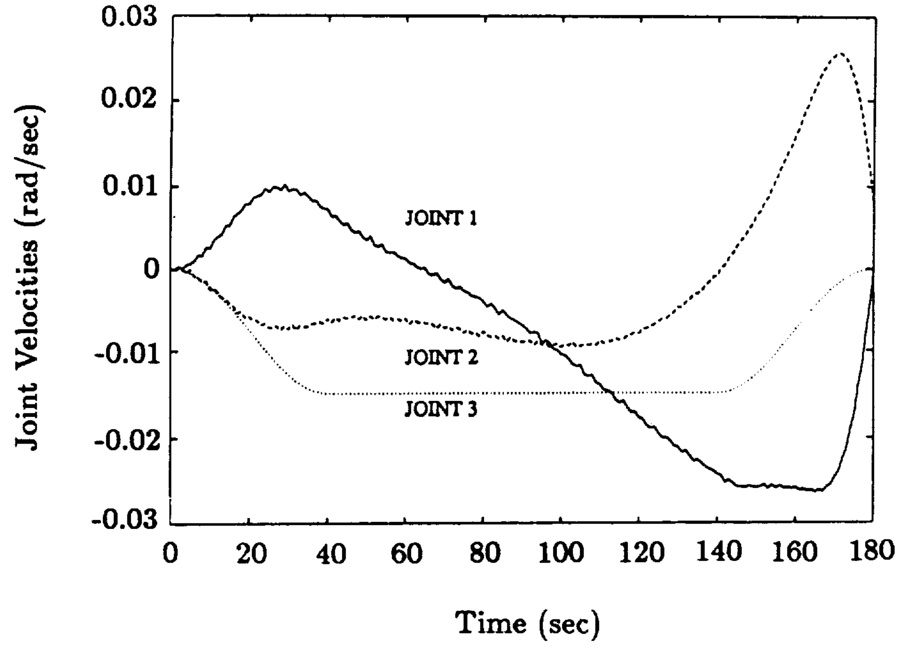


Fig. 7.6.10 Joint Angular Velocities (Flexible Members: 12 Elements)

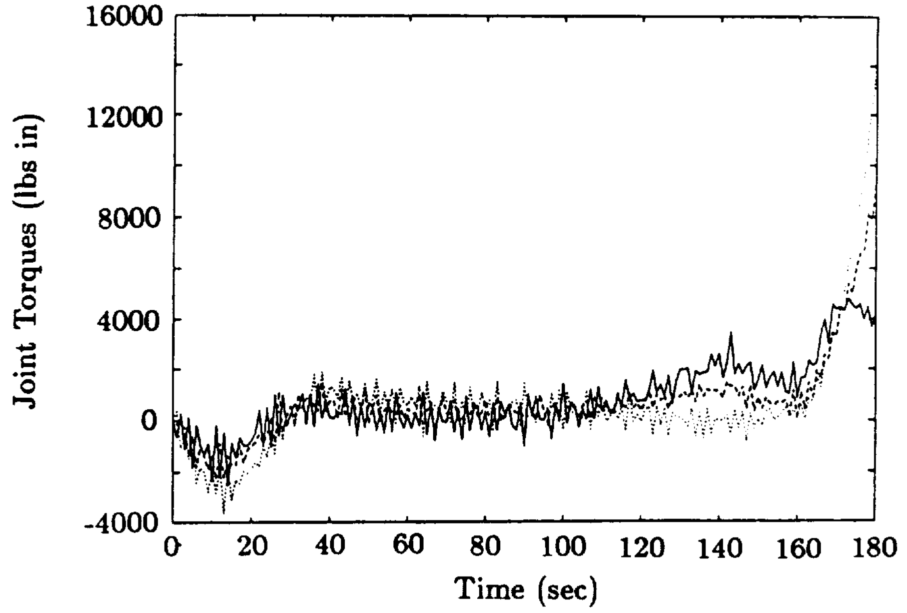


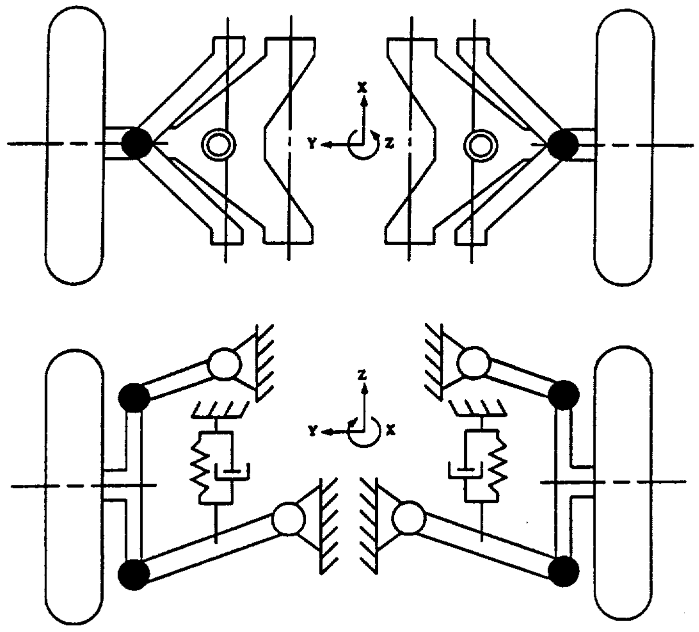
Fig. 7.6.11 Joint Torques (Flexible Members: 12 Elements)

## 7.7 Dynamic Simulation of Automobile Suspension Systems

To explore the parallelism of the present solution procedure, we select a vehicle model with multiple suspension systems. The configurations of the bodies and input data describing their initial conditions were provided by Professor P. Nikravesh of the University of Arizona, as shown in Fig. 7.7.1. According to the natural partitioned scheme used in section 5.9, the vehicle can be conventional partitioned into four subsystems (Fig. 7.7.2) where four independent processors can be assigned to each of the subsystem so that the null space of the constraint Jacobian matrix can be constructed in parallel. Note that the suspension systems possess four sets of springs and dampers with given locations, spring and damping coefficients. The tires of the vehicle are modeled by using unilateral spring elements. Initially, the vehicle is positioned in a height of one meter from the ground with initial velocities equal to zero. When the vehicle is been released, gravity acts as the external loads that force the vehicle to fall.

Fig. 7.7.3 illustrates one of the spring that reacts to the given external load during one second simulation run time. The displacements of each body, which simulate the behavior of the bodies in this system, are given in Figs. 7.7.4 to 7.7.8. The interesting features of this simulation are the CPU time consumption and the speed-up of using different processors in Alliant FX/8. Fig. 7.7.9 shows the dramatic reduction of the computer run time by employing one to four processors. Fig. 7.7.10 shows the speed up of using different number of processors which is calculated by dividing the total executing time on a sequential computer by the total executing time on a parallel computer. As expected, due to the overhead of the computations, the optimal speed up that can be achieved is less than the maximal number

(a)



(b)

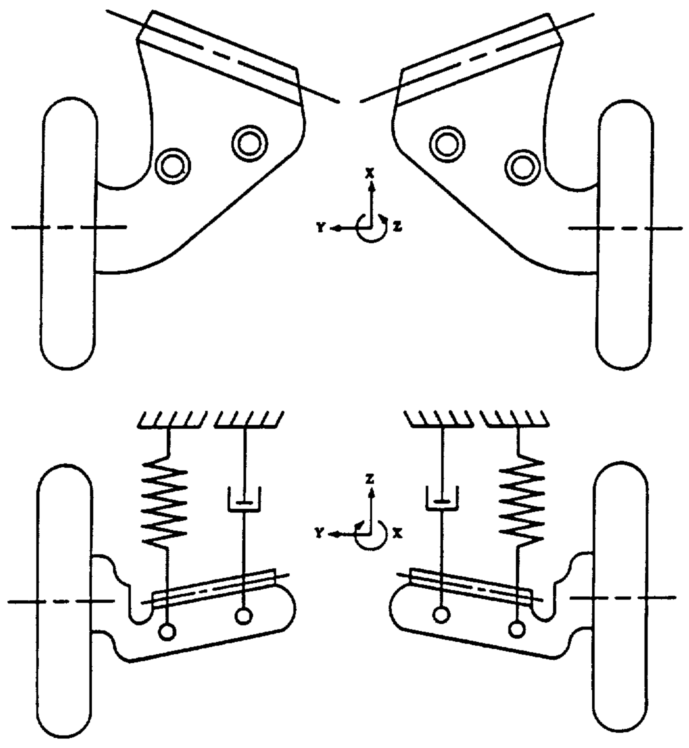


Fig. 7.7.1 Automobile Suspension Systems

(a) Front Suspension Systems: Top and Rear Views

(b) Rear Suspension Systems: Top and Rear Views

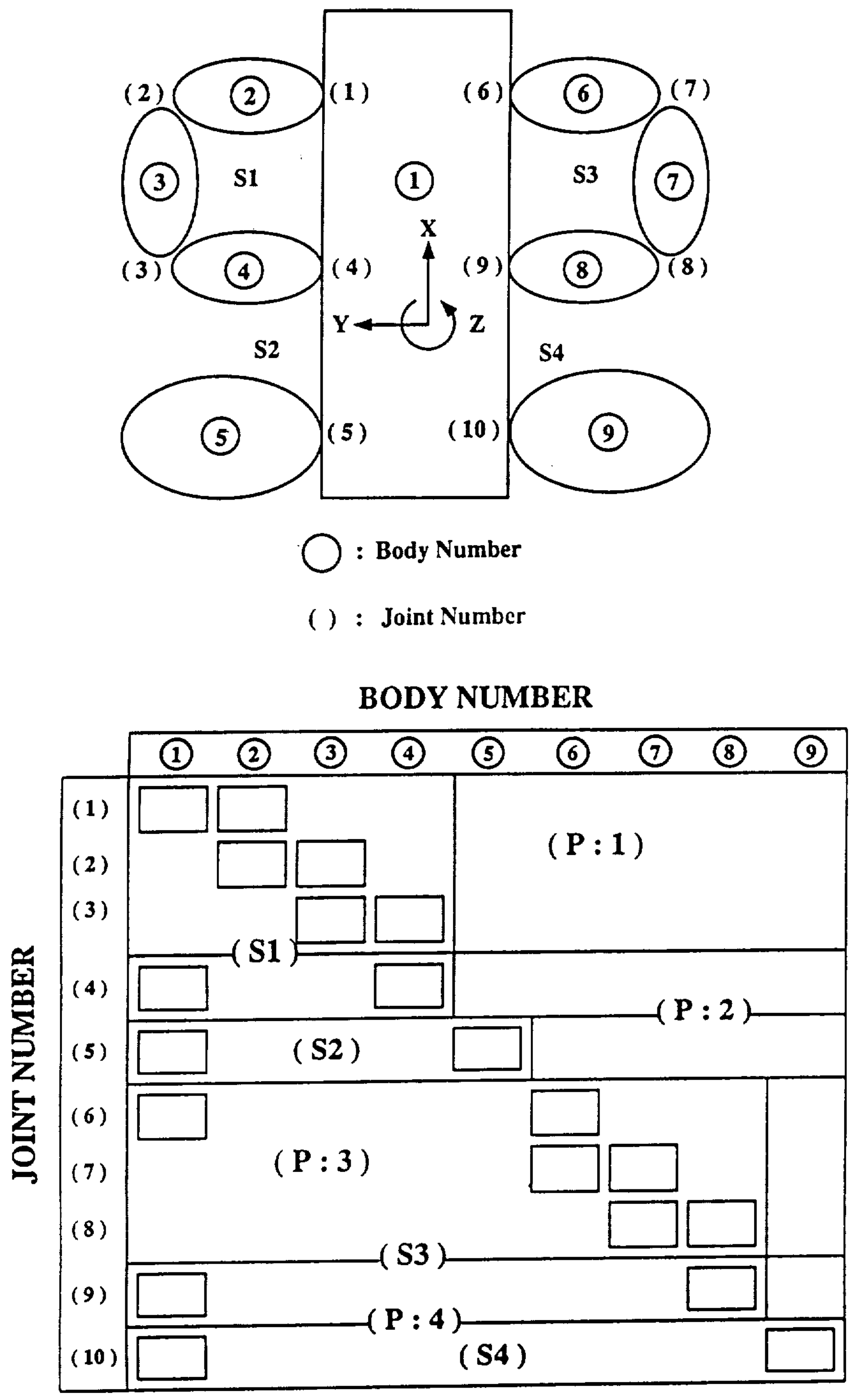


Fig. 7.7.2 Four Partitioned Subsystems of the Automobile Suspension Systems

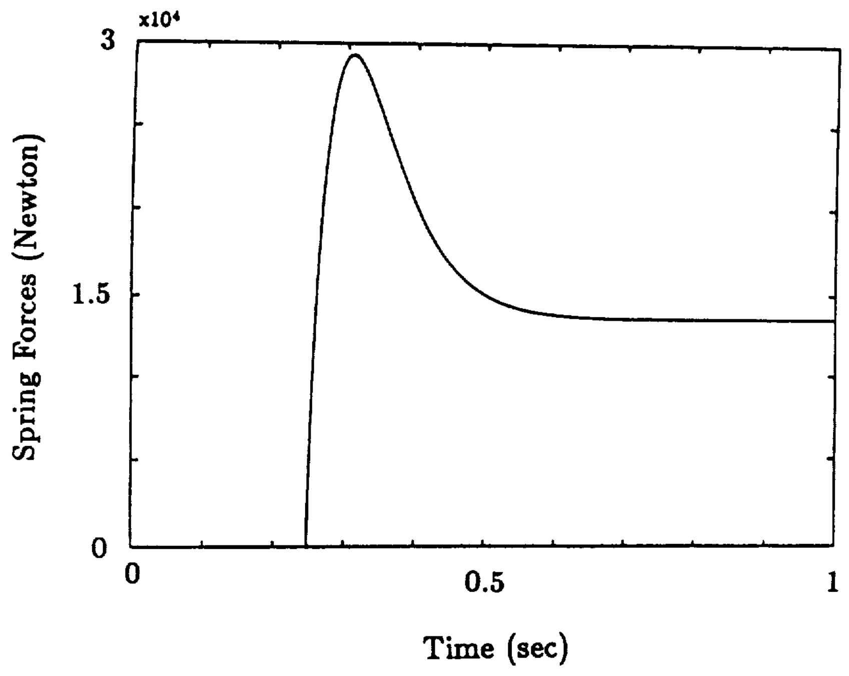


Fig. 7.7.3 Force Storage in Front Springs

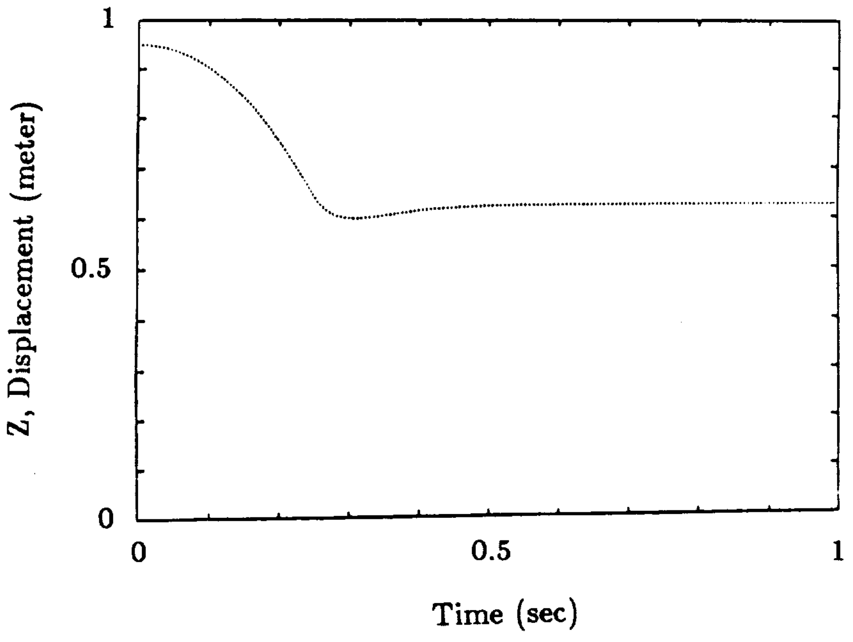


Fig. 7.7.4 Displacement History of Body 1

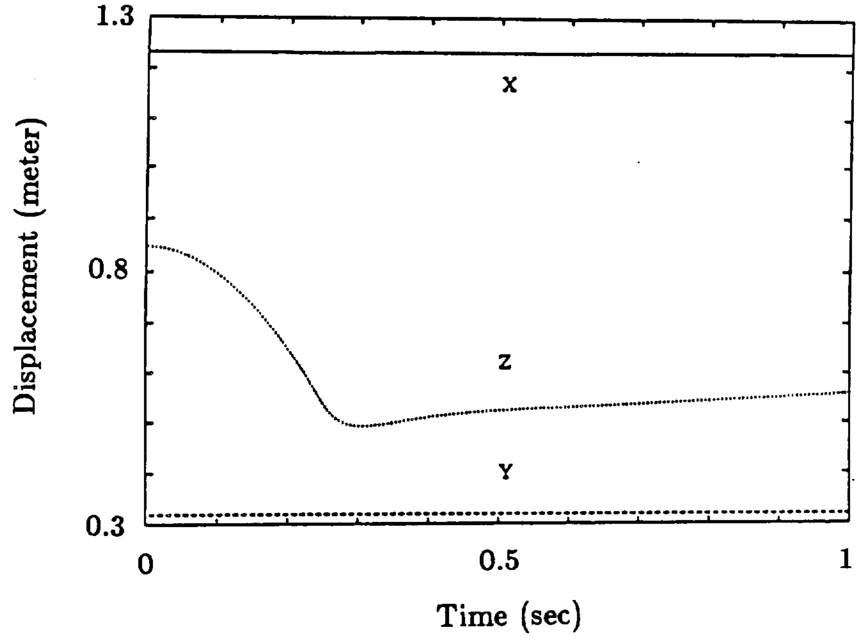


Fig. 7.7.5 Displacement Histories of Body 2

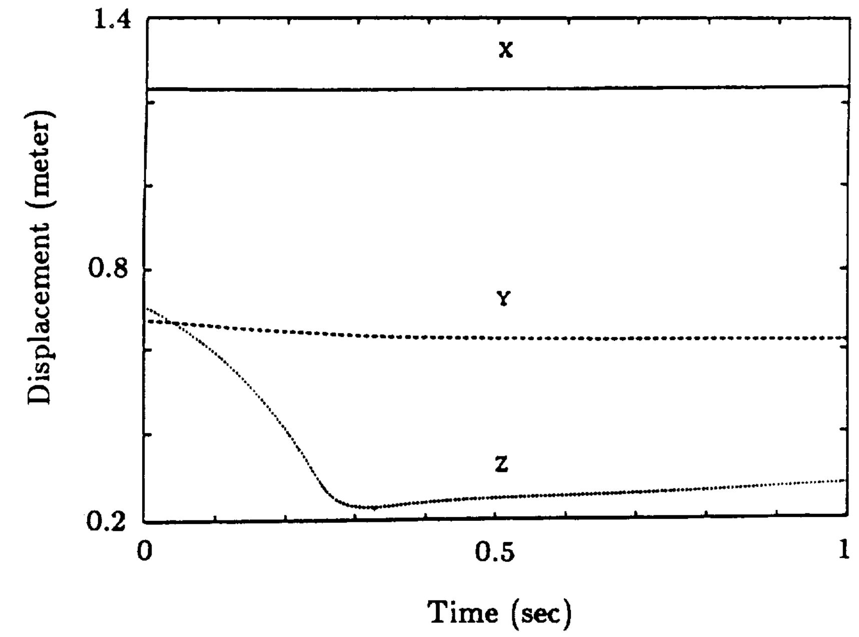


Fig. 7.7.6 Displacement Histories of Body 3

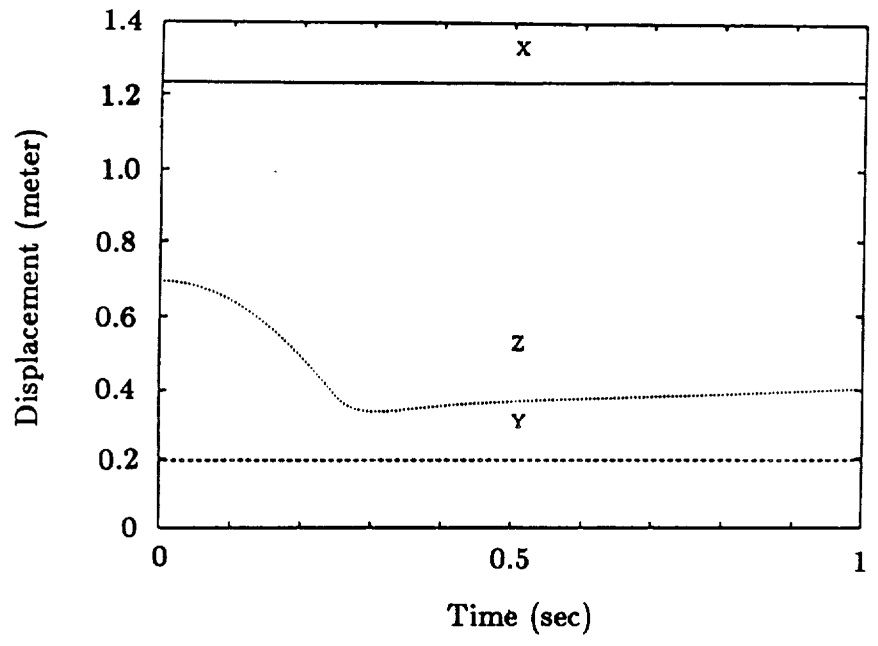


Fig. 7.7.7 Displacement Histories of Body 4

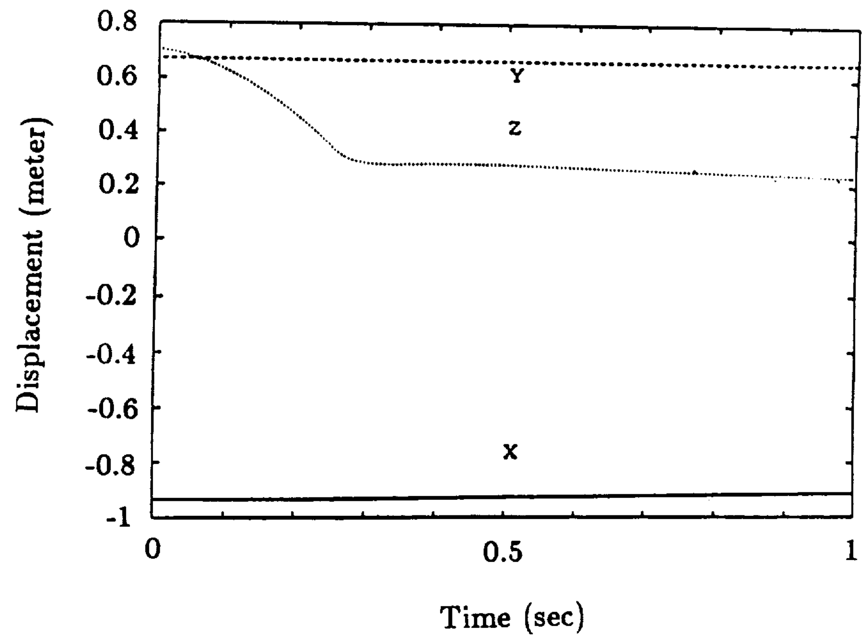


Fig. 7.7.8 Displacement Histories of Body 5

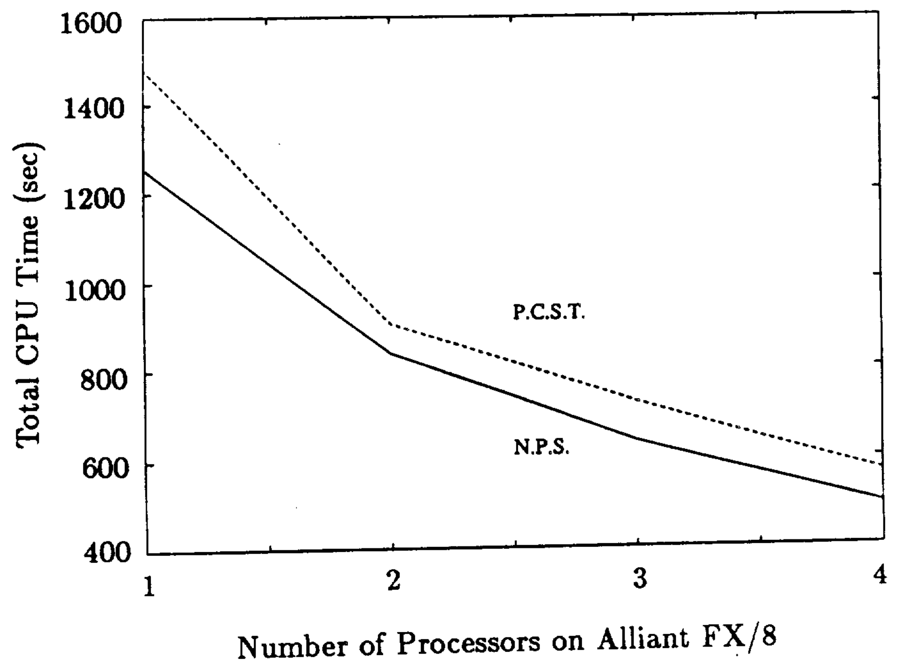


Fig. 7.7.9 Total Computer Run Time on Alliant FX/8: P.C.S.T. .vs. N.P.S.

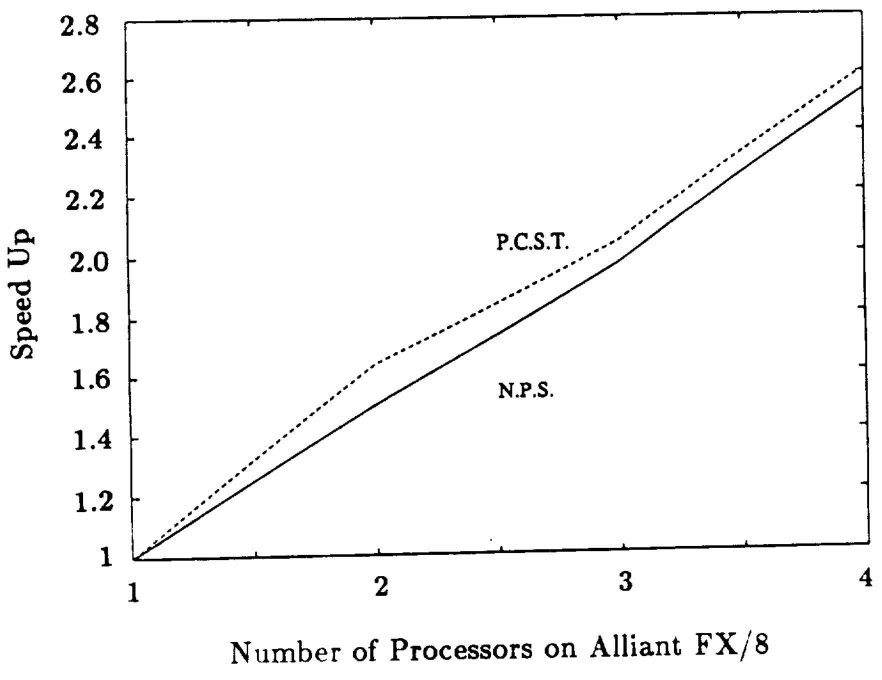


Fig. 7.7.10 Speed Up on Alliant FX/8: P.C.S.T. .vs. N.P.S.

of processors one has employed. The efficiency of using different processors is also calculated in dividing the speed up by the corresponding number of processors as shown in Fig. 7.7.11. Note that the solution procedure that use the penalty constraint stabilization technique(P.C.S.T) has also been adopted to solve this problem so that comparison can be made with present natural partitioned scheme (N.P.S.). The executing time, speed up, and efficiency of using P.C.S.T. are obtained as shown in Figs. 7.7.9, 7.7.10, and 7.7.11.

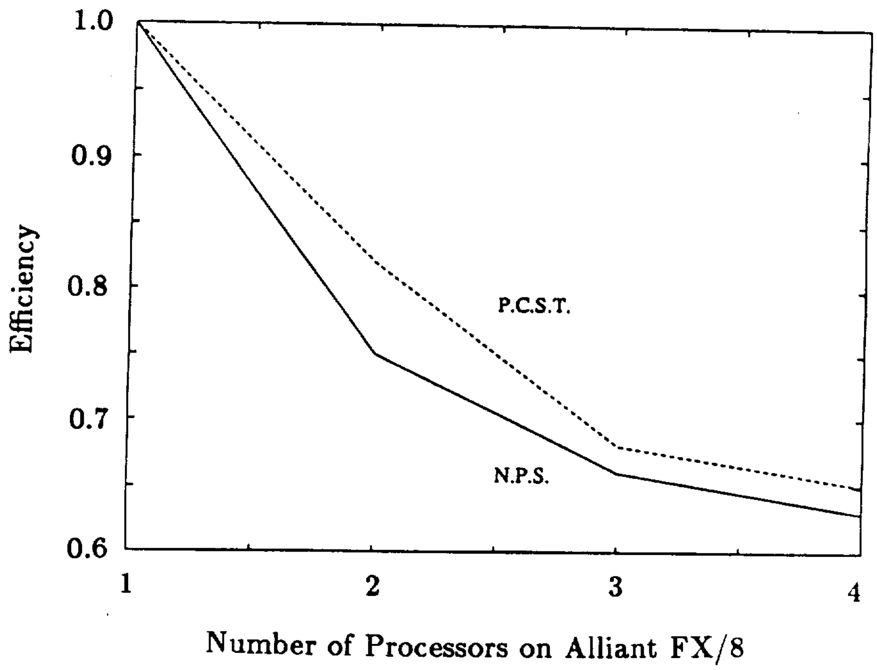


Fig. 7.7.11 Efficiency on Alliant FX/8: P.C.S.T. .vs. N.P.S.

## 7.8 Concluding Remarks

Present chapter has examined different MBD problems for the purpose that some important MBD issues regarding the solution of the DAEs can be addressed. The classical crank-slider mechanism problem has addressed the solution accuracy of proposed numerical schemes by comparing the results that are using the Baumgarte's technique and the solutions that are given by Haug and Yen. The three-link manipulator problem has exploited the robustness of the penalty constraint stabilization technique in solving the constraint forces where coefficient matrix $\mathbf{BM^{-1}B^{T}}$ becomes ill-conditioned whereas by comparing Baumgarte's technique. The dynamic of the bowling ball has provided the detail of dealing system consists of holonomic and nonholonomic constraints. On top of it, the robustness of the two-stage staggered explicit-implicit algorithm has been emphasized by comparing the conventional approach to calculate the angular velocity. A four-bar linkage problem has been examined to prove the feasibility of present DAEs formulation regarding system topologies. A problem involving maneuvering of a space crane along a specific time dependent trajectory has been solved to emphasized the versatility of the equations of motion and their corresponding solution procedures. The final numerical example problem has employed the nine bodies automobile suspension system to show the efficiency of using a parallel computer by using both proposed solution procedures. The results have encouraged us to further exploit a more efficient algorithm so that if the MBD systems consist of hundred of bodies, the speed up of the solution procedure can be constantly increased as the bodies in the systems increased.

# CHAPTER VIII

# CONCLUSIONS

## 8.1 Summary of Work

This dissertation has addressed two computationally oriented issues in multibody dynamic (MBD) research: constraint stabilization and constraint elimination. In constraint stabilization, a penalty constraint stabilization technique has been developed to efficiently control constraint violations that occur during the process of integrating DAEs. In constraint elimination, while maintaining stability, a new natural partitioning scheme has been developed to efficiently eliminate Lagrange multipliers from DAEs by explicitly identifying the independent coordinates at the joint level. When the null space of the constraint Jacobian matrix is constructed with this scheme, a second order differential equation system is obtained and expressed in terms of system independent variables.

The increasing dimensionality of MBD problems has motivated us to search for more robust and efficient numerical algorithms. In this regard, a two-stage staggered explicit-implicit procedure has been developed by exploiting the explicitness of the numerical algorithms so that they can be effectively converted to parallel computation. A Schur-complement-based parallel preconditioned conjugate gradient numerical algorithm has been used in the solution procedures in order to speed up these parallel computational schemes. Several simulation results have been verified by highly modular software developed and implemented as part of the dissertation.

The present multibody formulation is based on d'Alembert's principle of virtual work in which two different coordinate systems have been employed to describe the configuration of bodies in a multibody system. Inertial coordinates are used to locate the position of the center of mass of each individual body, whereas body-fixed coordinates which are rigidly attached to the center of mass are used to express the position of a particle on the body. By adopting this coordinate pair, one obtain a constant inertia matrix that can be partitioned into translational and rotational quantities to which numerical algorithms can be applied separately. Kinematic relationships of bodies in the systems are established by using constraints to enhance the modularity of the computer implementation. Constraints are incorporated into d'Alembert's principle of virtual work through the method of Lagrange multipliers. The resulting equations of motion, which are characterized as differential-algebraic equations (DAEs), consist of a set of second-order differential equations in conjunction with a set of algebraic equations that represent the constraint conditions.

During the process of integrating the equations of motion, time-discretization errors may accumulate in the constraint equations thus causing computed solutions to diverge. Several numerical techniques have been proposed to integrate DAEs and correct their constraint violations simultaneously.

The development of the penalty constraint violation technique has been motivated by the desire of obtaining a broadly applicable robust numerical algorithm for integration of DAEs. By converting the algebraic constraint equations into penalized first-order differential equations, the resulting equations retain parabolic-in-time characteristics. Such equations are well suited to direct time integration while constraint violations are forced to decay. From the numerical examples in Chapter 7, we conclude that the penalty constraint stabilization technique not only corrects the constraint violations stably and efficiently but also overcomes the difficulty of solving for the possibly ill-conditioned coefficient matrix $\mathbf{BM^{-1}B^T}$.

The natural partitioning scheme adopted here is motivated by the fact that an MBD system is governed by a set of second-order differential equations. For the purpose of automatically generating the system dynamic equations, we have deliberately maintained the equations of motion in DAEs form which represents a system having $n-m$ independent unknowns by one with $n+m$ unknowns, in which the $m$ Lagrange multipliers $\lambda$ are additional variables. By identifying the system dependent and independent variables, which are used to construct the null space of the constraint Jacobian matrix, we can transform the original DAEs into a set of second-order differential equations that are written in terms of independent variables. The natural partitioning scheme has been developed to explicitly determine the independent variables and consequently extract the null space of the constraint Jacobian matrix while avoiding the expensive numerical algorithms that have been proposed by other research groups.

A partitioned procedure for simulating the MBD systems has been developed to produce a more robust and efficient integration scheme. This divide-and-conquer computational strategy allows the dynamic analysis of MBD systems to be performed by assembling several modular software packages. Additional advantage of this modular organization is the simple interface with flexible beams module and that they can be adopted to integrate the equations of motion more efficiently. This procedure, which can be combined with the constraint force solver or the independent variable solver, has been characterized as a two-stage staggered explicit-implicit solution procedure. This procedure contains an efficient construction of solution matrices for both explicit and implicit time integration algorithms, a robust and stable treatment of constraint equations, and the possibility of parallel computations of constraint forces, independent variables, inertia forces and internal forces.

A highly modular software system has been designed and implemented for evaluating and validating the computational solution procedures for dynamic analysis of MBD systems. This software has been applied to several interesting MBD problems. The results confirm the effectiveness of the present computational schemes in regard to constraint stabilization and constraint elimination, the numerical accuracy of the two-stage staggered explicit-implicit algorithm, and the versatility of treating system with holonomic and/or nonholonomic constraints.

A Schur-complement-based parallel preconditioned conjugate gradient numerical algorithm has been developed and implemented on a parallel computer by assigning group of bodies to separate processors. It is shown that the present algorithm has provided a significant speed up in the numerical simulation of MBD problems such as automobile suspension systems.

In conclusion, the major contributions of this work can be summarized as follows:

(1) A treatment of holonomic and nonholonomic constraints as regards constraint stabilization and constraint elimination have been accurately and efficiently carried out.

(2) A two-stage staggered explicit-implicit numerical algorithm has been developed for the solution of MBD systems, which greatly enhances

the capability of simulating large-scale MBD systems.

(3) The modularity of the software implementation developed to validate and test these methods has facilitated further interdisciplinary efforts such as the incorporation of flexible beam dynamics.

(4) The effectiveness of using a Schur-complement-based parallel preconditioned conjugate gradient numerical algorithm has been verified to be highly effective in parallel MBD computations.

## 8.2 Directions for Further Research

Computer simulation nowadays plays an increasingly important role in the dynamic analysis and system design of MBD systems. The following areas of research are deemed important in extending these capabilities:

(1) The inclusion of friction effects in the joint kinematics. Those effects could have important influence on the local and global response of many MBD systems.

(2) The incorporation of contact-impact algorithms into MBD systems. Those algorithms would extend the capability of the present software to dynamic problems such as space shuttle docking and vehicle tire-ground interactions.

(3) The interaction with active control devices. This is important in applications that involve precision maneuvering and positioning. Such a development raises the issue of controlling DAEs, which is far more difficult than controlling ODEs.

(4) The validation of results obtained from the present software and the experimental testing of MBD systems. This is necessary to cross check both modeling and analysis capabilities incorporated in the present simulation.

## References

1. Bodley, C. S., Devers, A. D., Park, A. C., and Frish, H. P., "A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structures (DISCO)," NASA Technical Paper 1219, May 1978.

2. Chace, M. A., and Smith, D. A., "DAM-Digital Computer Program for the Dynamic Analysis of Generalized Mechanical Systems," SAE Paper No. 710244, Jan. 1971

3. Sheth, P. N., and Uicker, J. J., "IMP(Integrated Mechanism Program): A Computer-Aided Design Analysis System for Mechanisms and Linkages," ASME Journal of Engineering for Industry, Vol. 94, 1972, pp. 454-464.

4. Paul, B., "Analytical Dynamics of Mechanisms-A Computer Oriented Overview," Mechanism and Machine Theory, Vol. 10, No. 6, 1975, pp. 481-507.

5. Schiehlen, W. O., "Dynamics of Complex Multibody Systems," SM Archives, Vol. 9, 1984, pp. 159-195.

6. Orlandea, N., M. A. Chace, and D. A. Calahan, "A Sparsity-Oriented Approach to Dynamic Analysis and Design of Mechanical Systems - Part 1 and 2," ASME J. Engr. for Industry, Vol. 99, Aug. 1977, pp. 773-784.

7. Wehage, R. A., and E. J. Haug, "Generalized Coordinate Partitioning of Dimension Reduction in Analysis of Constrained Dynamic Systems," ASME J. Mech Design, Vol. 104, Jan. 1982, pp. 247-255.

8. Nikravesh, P. E., and O. K. Kwon, "Euler Parameters in Computational Kinematics and Dynamics, Part I amd II," ASME J. Mech. Tran. and Auto. in Design, Vol. 107, No. 3, Sep. 1985, pp. 358-369.

9. Huston, R. L., and C. E. Passerello, and M. W. Harlow, "Dynamics of Multirigid Body Systems," Journal Applied Mechanics, Vol. 45, No. 4, 1978, pp. 889-894.

10. Nikravesh P. E., Computer-Aided Analysis of Mechanical Systems, Prentice Hall, 1988.

11. Wittenburg, J., Dynamics of System of Rigid Bodies, B. G. Teubner, Stuttgart, 1977.

12. Bae, D. S., Haug, E. J. and Kuhl, J., "A Recursive Formulation for Constrained Mechanical System, Part III - Parallel Processor Implementation, " Mechanical Structures and Machines, Vol. 16, No. 2, 1988.

13. Zheng, Y-F. and Hemami, H., "Computation of Multibody System Dynamics by a Multiprocessor Scheme, " IEEE Tran. on Systems, Man, and Cybernetics, Vol. SCM-16, No. 1, 1986, 00. 102-110.

14. Yoon, Sugjoon, Real-Time Simulation of Constrained Dynamic Systems, Ph. D. Thesis, The University of Michigan, Ann Arbor, Michigan, 1990.

15. Bae, D. S. and Haug, E. J., "A Recursive Formulation for Constrained Mechanical System, Part I - Open Loop Systems, " Mechanical Structures and Machines, Vol. 15, No. 3, 1987, pp. 359-382.

16. Bae, D. S. and Haug, E. J., "A Recursive Formulation for Constrained Mechanical System, Part I - Closed Loop Systems, " Mechanical Structures and Machines, Vol. 15, No. 4, 1987, pp. 481-506.

17. Garcia de Jalon, J., Unda, J., Avello, A., and Jimenez, J. M., "Dynamic Analysis of Three-dimensional Mechanisms in 'Natural' Coordinates," ASME J. Mech. Tran. and Auto. in Design, Vol. 109, Dec. 1987, pp. 460-465.

18. Unda, J., Garcia de Jalon, J., Losantos, F., and Enparantza, R., "A Comparative Study on Some Different Formulations of the Dynamic Equations of Constrained Mechanical Systems," Transactions of the ASME, Vol. 109, Dec. 1987, pp. 466-474.

19. Gear, C. W., "The Simultaneous Numerical Solution of Differential-Algebraic Equations", IEEE Trans. Circuit Theory, TC-18, 1971, pp. 89-95

20. Gear, C. W., "The Numerical Solution of Problems Which May Have High Frequency Components ", Computer Aided Analysis and Optimization of Mechanical System Dynamics, ed., Haug. E. J., NATO ASI Series F, Vol. 9 Springer-Verlag, Heidelberg, 1984, pp. 335-349.

21. Gear, C. W. and Petzold, L. R., "ODE Methods for the Solution of Differential/Algebraic Systems ", SIAM Journal of Numerical Analysis, Vol. 21, No. 4, 1984, pp. 716-728.

22. Petzold, L. R., "Differential/Algebraic Equations Are Not ODE's ", SIAM Journal on Scientific and Statical Computing, Vol. 3, 1982, pp. 367-384.

23. Baumgarte, J. W., "Stabilization of Constraints and Integrals of Motion

in Dynamical System," Comp. Meth. Appl. Mech. Engr., 1, 1972, pp. 1-16.

24. Baumgarte, J. W., "A New Method of Stabilization for Holonomic Constraints," Journal of Applied Mechanics, 50, 1983, pp. 869-870.

25. Park, K. C., and Chiou, J. C., "Evaluation of Constraint Stabilization Procedures for Multibody Dynamical Systems," Proc. the 28th Structures, Structural Dynamics and Materials Conf., 1987, Part 2A, Monterey, CA, AIAA Paper No. 87-0927, pp. 769-773.

26. Park, K. C., and Chiou, J. C., "Stabilization of Computational Procedures for Constrained Dynamical System," Journal of Guidance, Control and Dynamical System, 1988, Vol. 11, No. 4, 3 pp. 65-370.

27. Wehage, R. A., and E. J. Haug, "Generalized Coordinate Partitioning of Dimension Reduction in Analysis of Constrained Dynamic Systems," ASME J. Mech Design, Vol. 104, Jan. 1982, pp. 247-255.

28. Walton, W. C., and Steeves, E. C., "A New Matrix Theorem and Its Application for Establishing Independent Coordinates for Complex Dynamical Systems with Constraints," NASA TR-R326, 1969.

29. Singh, R. P., and Likins, P. W., "Singular Values Decomposition for Constrained Dynamical Systems," Journal of Applied Mechanics, Vol. 52 Dec. 1985, pp. 943-948.

30. Liang, C. G., and Lance, G. M., "A Differentiable Null Space Method for Constrained Dynamic Analysis," J. Mech. Tran. and Auto. in Design, Vol. 109, Sep. 1987, pp. 405-411.

31. Ider, S. K., and Amirouche, F. M. L., "Coordinate Reduction in the Dynamics of Constrained Multibody Systems-A New Approach," Journal of Applied Mechanics, Vol. 55 Dec. 1988, pp. 899-904.

32. Amirouche, F. M. L., and T. Jia., "Pseudouptriangular Decomposition Method for Constrained Multibody Systems Using Kane's Equations," Journal of Guidance and Control, Vol. 11 No. 1 Jan.-Feb. 1988, pp. 39-46.

33. Armstrong, W. W., "Recursive Solution to the Equations of Motion of an N-Link Manipulator," Proc. 5th Would Congress Theory Mach. Mechanisms, Vol. 2, Montreal, ASME, 1979, pp. 1343-1346.

34. Orin, D. E., McGhee, R. B., Vukobratovic, M., and Hartoch, G., "Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods," Mathematics of Bioscience, Vol. 43, 1979, pp. 106-130.

35. Luh, J. Y. S., Walker, M. W., and Paul, R. P. C., "On-Line Computational Scheme for Mechanical Manipulators," ASME, Journal of Dynamic Systems, Measurement, and Control, June, 1980, Vol. 102 pp. 69-76.

36. Hollerbach, J. M., "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-10, No. 11, Nov. 1980, pp. 730-736.

37. Silver, W., "On the Equivalence of Lagrangian and Newton-Euler Dynamics for Manipulators," International Journal of Robotics Research, Vol. 1, No. 2, 1982, pp. 60-70.

38. Featherstone, W. R., "The Calculation of Robot Dynamics using Articulated-Body Inertias, " International Journal of Robotics Research, Vol. 2, No. 1, 1983, pp. 13-30.

39. Wehage, R. A., "Application of Matrix Partitioning and Recursive Projection to $O(n)$ Solution of Constrained Equations of Motion, " in Trends and Development in Mechanisms, Machines and Robotics, Vol. 2, ASME, 1988, pp. 221-230.

40. Park, K. C., Felippa, C. A., and DeRuntz, J. A., "Stabilization of Staggered Solution Procedures for Fluid-Structure Interaction Analysis," in Comp. Methods for Fluid-Structure Interaction Problems, Belytschko, T. and Geers, T. L., (editors), ASME, AMD Vol. 26, New York, N. Y., 1977, pp. 95-124.

41. Park, K. C., "Partitioned Analysis Procedures for Coupled-Field Systems: Stability Analysis," Journal of Applied Mechanics, 47, 1980, pp. 370-378.

42. Park, K. C., and Felippa, C. A., "Partitioned Analysis of Coupled Systems," in Computational Methods for Transient Analysis, Belytschko, T. and Hughes, T. J. R. (editors), Elsevier Pub. Co., 1983, pp. 157-219.

43. Park, K. C., Chiou, J. C., and Downer, J. D., "Explicit-Implicit Staggered Procedure for Multibody Dynamics Analysis," Journal of Guidance, Control, and Dynamics, Vol. 13, No. 3, May-June, 1990, pp. 562-570.

44. Spurrier, R. A., "Comment on Singular-Free Extraction of a Quaternion from a Direction-Cosine Matrix," Journal of Spacecraft and Rockets, Vol. 15, 1978, pp 255.

45. Greenwood, D. T., Principles of Dynamics, Prentice Hall, 1965.

46. Meirovitch, L, Methods of Analytical Dynamics, McGraw-Hill, 1970.

47. Spring, K. W., "Euler Parameters and the Use of Quaternion Algebraic in the Manipulation of Finite Rotations: A Review", Mechanism and Machine Theory, Vol. 21, No. 5, 1986, pp. 365-373.

48. Stuelpnagel, J., "On the Parametrization of the Three-Dimensional Rotational Group", SIAM Review, Vol. 6, No. 4, Oct. 1964, pp. 422-430.

49. Cheng, H. and Gupta, K. C., "An Historical Note on Finite Rotations", Journal of Applied Mechanics, March, 1989, Vol. 56, pp. 139-145.

50. Park, K. C., Class Notes: Mathematical Methods in Dynamics, Fall, 1988.

51. Lanczos, C., The Variational Principle of Mechanics, Dover, 1970.

52. Downer, J. D., Park, K. C. and Chiou, J. C., "A Computational Procedure for Multibody Systems Including Flexible Beam Dynamics ", Report No. CU-CSSC-90-08, Center for Space Structures and Controls, University of Colorado.

53. Calahan, D., "Numerical Considerations in the Transient Analysis and Optimal Design of Nonlinear Circuits,", Dig. Rec. Joint Conf. Mathematical and Computer Aids to Design, ACM/SIAM/IEEE. 1969.

54. Hemami, H. and Weimer, F. C., "Modeling of Nonholonomic Dynamic Systems With Applications ", Journal of Applied Mechanics, March 1981, Vol. 48, pp. 177-182.

55. Kim, S. S., and Vanderploeg, M. J., "QR Decomposition for State Space Representation of Constrained Mechanical Dynamic Systems, " ASME J. Mech. Tran. and Auto. in Design, Vol. 108, No. 2, June, 1986, pp. 183-188.

56. Bae, D. S., Yang, S. M., "A Stabilization Method for Kinematic and Kinetic Constraint Equations," 1989 NATO Advance Research Workshop, Real-Time Integration Methods for Mechanical System Simulation, Snowbird, Utah, Aug. 7-11, 1989.

57. Farhat, C., and Wilson, E., "A New Finite Element Concurrent Computer Program Architecture," International Journal for Numerical Methods in Engineering, Vol. 24, 1987, pp. 1771-1792.

58. Farhat, C., Crivelli, L., "A General Approach to Nonlinear FE Computations on Shared-Memory Multiprocessors," Comp. Meth. in App. Mech. and Engr. 72, 1989, pp. 153-171.

59. Hughes, T. J. R., Ferencz, R. M. and Hallquist, J. O., "Large-Scale Vectorized Implicit Calculations in Solid Mechanics on A Cray X-MP/48 Utilizing EBE Preconditioned Conjugate Gradients," Comp. Meth. in App. Mech. and Engr. 61, 1987, pp. 215-248.

60. Smith, I. M., Wong, S. W., Gladwell, I., and Gilvary B., "PCG Methods in Transient FE Analysis. Part I: First Order Problems," Intern. Journal for Numerical Methods in Eng, Vol. 28, 1989, pp. 1557-1566.

61. Smith, I. M., Wong, S. W., and Gladwell, I., "PCG Methods in Transient FE Analysis. Part II: Second Order Problems," Intern. Journal for Numerical Methods in Eng, Vol. 28, 1989, pp. 1567-1576.

62. Carter, W. T. Jr., Sham, T-.L., and Law, K. H., "A Parallel Finite Element Method and Its Prototype Implementation on a Hypercube," Computers and structures, Vol. 31, N0. 6, 1989, pp. 921-934.

63. Jordan, H., Benton, M. and Arenstorf, N., *Force User's Manual*, Department of Electrical and Computer Engineering, University of Colorado, Boulder, CO, 1987.

64. Haug, E. J., and Yen, J., "Implicit Numerical Integration of Constrained Equations of Motion Via Generalized Coordinate Partitioning," 1989 NATO Advance Research Workshop, Real-Time Integration Methods for Mechanical System Simulation, Snowbird, Utah, Aug. 7-11, 1989.

65. Houston, R. L., Passerello, C. Winget, J. M., and Sears, J., "On the Dynamics of a Weighted Bowling Ball," Journal of Applied Mechanics, Vol. 46, Dec. 1979, pp. 937-943.

66. Gawronski, W., Ih, C-H, C.,"3D Rigid Body Dynamic Modeling of Space Crane for Control Design and Analysis," JPL D-6878, Nov. 17, 1989.

67. Sutter, T. R., Bush, H. G., and Wallsom, R. E., "An Articulated-Truss Space Crane Concept," 31st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Material Conference, Long Beach, CA, April, 1990, AIAA Paper 90-0994.

68. Ramesh, A. V., Utku, S., Wada, B. K., and Chen, G. S. "Effect of Imperfection on Static Control of Adoptive Structures As a Space Crane," Adoptive Structures, The Winter Annual Meeting of the ASME, San Francisco, CA, Dec. 10-15, 1989, pp. 95-101.

69. Craig, J. J., *Robotics, Mechanics and Control*, Addison-Wesley, 1986.