1988

A symbolic formulation for equations of motion of multibody systems

Alan G. Lynch
Iowa State University

Order Number 8909169

A symbolic formulation for equations of motion of multibody systems

Lynch, Alan G., Ph.D.

Iowa State University, 1988

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

A symbolic formulation for equations of motion of multibody systems

by

Alan G. Lynch

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa
1988

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my appreciation to my advisor and friend, Martin Vanderploeg, for his encouragement and assistance throughout my graduate program. I would also like to thank Professors Bernard, Huston, McConnell, and Pierson for their input while serving on my doctorate committee.

I wish to give special thanks to my fiancé, Mary, for her continued love and support in my work.

In addition, I thank the gentlemen in 0095E, Alan Hufnagel, Jay Shannan, Jeff Trom, Chaeyoun Oh, and Don Dusenberry for their inspiration and encouragement during my dissertation work.

DESCRIPTION OF NOMENCLATURE

The nomenclature presented in this thesis requires a brief explanation. The notation is necessary since complicated multibody systems are being analyzed. Three types of variables are discussed in this thesis: scalar variables, vectors, and matrices. Scalar variables will be defined by a lower-case letter (e.g., a). Vectors will be defined by a lower-case bold letter (e.g., $\mathbf{a}$). Matrices are defined by an upper-case bold letter (e.g., $\mathbf{A}$).

Superscripts are used to distinguish variables acting between bodies in a multibody system. As an example, the matrix $\mathbf{A}^{ij}$ is a transformation matrix from body j to body i and $\mathbf{d}^{ij}$ is a position vector from joint j to body i center of gravity. Subscripts are used to distinguish between elements in a vector or matrix. As an example, the term $\mathbf{A}_{ij}$ is the ith row and jth column of matrix $\mathbf{A}$.

LIST OF NOMENCLATURE

$\mathbf{A}^{ij}$	transformation matrix from body j to body i
a	design parameter for design sensitivity analysis
$\mathbf{B}$	partial velocity matrix
$\mathbf{B}_{ij}$	submatrix of the partial velocity matrix
$\dot{\mathbf{B}}$	time rate of change of the partial velocity matrix
$\mathbf{C}_l$	$ndf \times ndf$ linearized damping matrix
c^{ij}	damping coefficient applied to a TSDA or RSDA element
$\mathbf{DM}$	$nb \times nb$ distance matrix
$\mathbf{d}$	m_d vector of dependent coordinates
$\mathbf{d}^{ii}$	local position vector from joint i to body i center of gravity
$\mathbf{d}^{ij}$	position vector from joint j to body i center of gravity
e_i	Euler parameters, $i = 1, ..., 4$
F_j	holonomic generalized active force associated with the jth generalized coordinate
F_j^*	holonomic generalized inertia force associated with the jth generalized coordinate

$\mathbf{f}^i$	Cartesian force vector for body i
$\mathbf{f}$	system Cartesian force vector or *ndf* vector of conservative and nonconservative forces
$\hat{\mathbf{f}}$	system Cartesian force vector less the actuator torques and forces
$\mathbf{f}_R^{ij}$	RSDA torque associated with joint i
$\mathbf{f}_T^{ij}$	TSDA force associated with joint i
$\mathbf{g}$	general *ndf* vector of nonlinear equations
$\mathbf{h}^i$	Cartesian Coriolis and centrifugal force vector for body i
$\mathbf{h}$	system Cartesian Coriolis and centrifugal force vector
$\mathbf{h}^{RO}$	angular momentum vector of point O with respect to reference frame R
$\mathbf{I}_{nb}$	$nb \times nb$ identity matrix
$\mathbf{J}^i$	3×3 inertia matrix for body i
$\mathbf{K}_l$	$ndf \times ndf$ linearized stiffness matrix
k^{ij}	spring constant for a TSDA or RSDA element
$\mathbf{M}$	$ndf \times ndf$ generalized mass matrix
$\mathbf{M}_l$	$ndf \times ndf$ linearized mass matrix
$\mathbf{M}^i$	6×6 mass matrix for body i
md	number of constraints or dependent coordinates
n	number of generalized coordinates in a constrained system
nb	number of bodies in a system

ndf	number of degrees of freedom in a system
ni	number of independent coordinates of a constrained system
$\mathbf{p}^i$	4×1 vector of Euler parameters
$\mathbf{p}^{O\,i*}$	position vector from point O to i* defined in body i coordinates
p^i	attachment point for a TSDA element on body i
$\mathbf{Q}$	$n \times n$ orthogonal matrix used in QR decomposition
$\mathbf{Q}_1$	orthonormal basis to the constraint surface
$\mathbf{Q}_2$	orthonormal basis of the constraint tangent surface
$\mathbf{q}$	ndf vector of relative positions
$\dot{\mathbf{q}}$	ndf vector of relative velocities
$\ddot{\mathbf{q}}$	ndf vector of relative accelerations
$\mathbf{q}^*$	equilibrium point for linearization
q^i	ith generalized coordinate
$\dot{q}^i$	time rate of change of q^i or the ith generalized speed
$\mathbf{R}$	$n \times m_d$ upper triangular matrix used in QR decomposition
$RSDA$	rotational spring-damper-actuator
$\mathbf{r}^{RQ}$	vector position of point Q with respect to the reference frame R.

$\mathbf{r}^i$	resultant of all external forces for body i whose line of action passes through the mass center
$\mathbf{r}^{i*}$	Cartesian inertia force for body i
$\mathbf{s}^i$	local position vector from mass center to p^i
$\mathbf{s}^{iQ}$	local vector from body i mass center to point Q
$\mathbf{s}^{ij}$	local position vector from body j center of gravity to joint i
T	total trajectory time for simulation
$TSDA$	translational spring-damper-actuator
t	instantaneous time
$\mathbf{U}$	$n \times n_i$ eigenvector matrix defined in the q basis vectors
$\mathbf{u}^i$	unit vector used in definition of Euler parameters or unit vector about which the *ith* generalized coordinate rotates
$\mathbf{u}_i$	*ith* mode shape of system
$\mathbf{V}$	$n_i \times n_i$ eigenvector matrix defined in the z basis vectors
$\mathbf{v}^{RO}$	velocity of point O with respect to reference frame R
$\mathbf{v}^i$	linear velocity of the center of gravity of body i
$\dot{\mathbf{v}}^i$	linear acceleration of the center of gravity of body i
$\dot{\mathbf{y}}^{Ri}$	Cartesian velocity vector of body i with respect to reference frame R
$\dot{\mathbf{y}}^R$	system Cartesian velocity vector
$\ddot{\mathbf{y}}^R$	system Cartesian acceleration vector

$\mathbf{z}$	constant vector in generalized eigenvector problem or an ni vector of independent coordinates
α^i	initial undeformed length of a rotational spring
β^i	initial undeformed length of a translational spring
$\delta\mathbf{q}$	variation about $\mathbf{q}^*$ in linearization analysis
ϵ^i	determines the joint type for body i
$\boldsymbol{\lambda}$	nc vector of Lagrange multipliers
ν^{ij}	part of the linear velocity component for body i
π	$nb \times nb$ path matrix
ρ	$nb \times nb$ reference matrix
σ	constant scalar in generalized eigenvector problem
τ	Actuator torques and forces for inverse dynamic simulation
τ^i	sum of all external torques acting on body i
τ^{i*}	inertia torque for body i
τ^{ij}	external torque or linear actuator force applied at a TSDA or RSDA element
$\Phi_\mathbf{q}$	$ndf \times nc$ constraint Jacobian matrix
ϕ^i	rotation angle for Euler parameter definition
Φ_i	ith constraint equation for a closed-loop system
Ω^{ij}	part of angular velocity component for body i
ω^{Ri}	angular velocity of body i in reference frame R

ω^i angular velocity of body i with respect to body i coordinates

$\dot{\omega}^i$ angular acceleration of body i with respect to body i coordinates

$\tilde{\omega}^i$ 3×3 skew symmetric matrix of ω^i

ω *ith* natural frequency of system

1 INTRODUCTION

In recent years, mathematical models for multibody dynamic systems have become an important design and analysis tool. Multibody dynamic systems are represented by a series of rigid mechanical links containing a wide variety of joint types. The mathematical models are defined by differential equations which simulate large motions of the system. Several general-purpose computer programs exist which allow engineers to model large mechanical systems and to evaluate the dynamic characteristics of potential designs before the building of prototypes. This reduces cost and lead time on designs. These general-purpose simulation packages have several drawbacks. They are often difficult to use and are numerically inefficient. As a result, extensive research efforts have been directed at developing more efficient and user-friendly methods for modeling multibody dynamic systems.

Simulation of multibody mechanical systems, which are usually highly nonlinear, has only recently been made possible with the advent of the digital computer. As the power of digital computers developed, more sophisticated simulation techniques evolved. Originally, simulation involved the engineer generating equations of motion by hand and using the computer to integrate the equations of motion. These models were usually specific and only the system parameters could be varied such as lengths, masses, and inertias. The equations were dependent on the system topol-

ogy and had to be reformulated if the topology changed. With increasing computer power, a strategy evolved where the equations of motion are numerically generated at each time step during the numerical integration. Most of the general-purpose, multibody dynamic software packages in use today such as DRAM [1], ADAMS [2], IMP [3], and DADS [4,5] numerically generate equations of motion. The advantage of this method is that the computer can generate and solve any general class of multibody systems and the engineer is not required to form the equations of motion. However, the equations are obtained numerically and no symbolic equation is available. This can create a loss of insight into the problem, and without explicit equations, gradient information is difficult to obtain for control system design or optimal design of systems.

More recently, symbolic manipulation has been used to derive explicit equations. Symbolic computer programs are now available which can be used to generate equations of motion for a broad class of problems with greatly reduced effort. These equations are then available for numerical integration or other system analysis techniques.

One advantage of generating symbolic equations is that the efficiency of numerical integration can be increased. The equations are generated one time and can be integrated any number of times rather than being formed at each integration step as in the numerical methods. Another advantage is that explicit derivatives can be computed for linearization, control system design, or optimization. In addition, the symbolic equations often yield insight into system parameters. This insight, however, is often difficult for very large systems because of the extreme size and complexity of the equations.

This thesis presents a symbolic formulation for the equations of motion for multibody dynamic systems. Previously developed symbolic methods require a large amount of input from the user and are difficult to use. In contrast, the method presented in this thesis is easy to use because all system definition is in terms of Cartesian coordinates. The symbolic equations of motion are then derived in terms of relative coordinates. The usefulness of this approach has been demonstrated in numerical formulations [6] and the symbolic formulation developed in this thesis exhibits the same benefits

Chapter 2 reviews previous work in the area of symbolic generation of equations of motion. Chapter 3 presents the kinematics of multibody systems and introduces the partial velocity matrix, the key to this approach. Chapter 4 discusses the dynamics of multibody systems. Kane's dynamical equations are introduced and modified for symbolic formulation. Chapter 5 presents examples of symbolic equation generation for several dynamic systems. Chapter 6 extends the symbolic method to include linearization, constrained eigenvalue analysis, design sensitivity analysis, and substructuring. Finally, Chapter 7 provides conclusions and recommendations for future work.

2 LITERATURE REVIEW

Symbolic manipulation performs mathematical operations on defined symbols as well as numbers. It differs from numerical calculations since symbolic variables are retained through each mathematical operation. These symbols represent the numerical quantities in a general form.

Symbolic computing was started in the late 1960s at Massachusetts Institute of Technology [7]. At that time, the computer was confined to strictly arithmetic calculations. The goal was to enable the computer to do mathematical operations on defined symbols. Since then, three major software programs have evolved for symbolic computing. They are MACSYMA, from Symbolics Inc., SMP, from Inference Corp., and REDUCE, from Rand Corp.

The use of symbolic manipulation in generation of dynamic equations has only recently appeared in the literature. This approach has been presented in three major fields: machine dynamics, spacecraft dynamics and robotics.

In the field of machine dynamics, Dubowsky and Grant [8] were the first to present symbolic manipulation and apply it to a time domain analysis of a planar linkage system. They compared the symbolic versus the numeric method and found that it was feasible to apply symbolic methods with substantial savings in computation costs. Hussain and Noble [9] demonstrated the applications of sym-

bolic computing in the analysis of several linkage systems. Kortum and Schiehlen [10] introduced a general-purpose, multibody dynamic equation generator called NEWEUL. This code used the Newton-Euler method to formulate equations. It was developed mainly for applications in vehicle dynamics but could be applied to a wide variety of mechanisms [11,12]. NEWEUL requires, as input for each body in the system, the position vectors and translation vectors and also rotation and inertia tensors in symbolic form. Thus, for large systems, the input can become complex. Wittenburg and Wolz [13] developed a general-purpose code called MESA VERDE. The equations of motion were developed from the principle of virtual work. They also introduced the concept of system topology which was a mathematical way to describe the interconnection of the rigid bodies.

In the field of spacecraft dynamics, Levinson [14] applied Kane's equations to a seven degree of freedom satellite system. The symbolic manipulation language called FORMAC was used to generate explicit equations of motion. Rosenthal and Sherman [15] developed a multibody open-loop program called SD/EXACT. They demonstrated that their code provided a reduction in execution time of up to an order of magnitude for large problems when compared to numerical generation techniques.

Several publications dealing with symbolic equation generation have appeared in the robotics literature. Robot mechanisms often require a real-time controller which requires an efficient dynamic simulation. Literature on robotics [15-21] shows that most of the code developed via symbolic methods is based on the Lagrangian formulation and requires that the mechanism be a single chain, open-loop system. Leu and Hemati [16] derived the dynamic equations of motion for manipulators of

any number of degrees of freedom. They use the symbolic language MACSYMA to develop the open-loop equations consisting of rotational and translational joints.

2.1 Objectives and Scope

The main objective of this research is to formulate equations of motion for general, multibody systems using symbolic methods. The technique for generating the equations of motion should be general so that a wide range of mechanical systems can be modelled. Also, the input format should include an effective user interface which can be be set up for any class of problems.

Kim and Vanderploeg [6] presented a general-purpose method for formulating equations of motion which incorporates both an efficient user interface and an efficient solution process. They used a modified form of Kane's equations and numerically formulated the equations of motion. This thesis will extend this formulation to a symbolic form. Kim and Vanderploeg showed that this approach utilizes advantages from both Cartesian and relative coordinate schemes. The Cartesian coordinates will be used in defining the system, i.e., the joint definitions, transformation matrices, and inertia properties, while the relative coordinates will be used in defining the equations of motion resulting in a minimal set of differential equations.

3 KINEMATICS OF MULTIBODY SYSTEMS

This chapter presents the kinematic fundamentals for the formulation of dynamic equations for multibody systems. Kinematics is the study of the geometry of motion. Motion is described by the position, velocity, and acceleration of the components making up the system.

The first section of this chapter states the assumptions used in this study and the definition of rigid body motion. Section 2 discusses graph theory, which is important in understanding the interconnections in multibody mechanical systems. Sections 3 and 4 derive the position and velocity information for different types of kinematic joints. In Section 5, the partial velocity matrix is introduced. Finally, acceleration quantities are discussed in Section 6.

3.1 Rigid Body Motion

Rigid body motion is the major assumption that governs this analysis. By definition, a rigid body undergoes no deformations and the distance between any pair of points on the rigid body is assumed to be constant [22]. This assumption is valid if body deformations are negligible compared to the motion of the entire body.

A general rigid body can be represented in three dimensional space by six

coordinates. Three of the coordinates define the orientation of the body and the other three coordinates define the translational position. All six coordinates are defined with respect to an inertial reference frame.

The orientation coordinates are normally obtained using three successive rotational angles about an orthogonal axis. Two commonly used sets of orientation coordinates are Euler angles and Bryant angles. It has been demonstrated in the literature that three rotational coordinates cause singularity problems [5]. Another set of orientation coordinates, called Euler parameters, have no singularity problems and are more numerically efficient than three rotational angles. Euler parameters define the orientation of a body by an angle of rotation ϕ^i, about some unit vector $\mathbf{u}^i$. Thus, four coordinates define the general orientation of a rigid body defined as

$$\mathbf{p}^i = \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} cos(\phi^i/2) \\ \\ \mathbf{u}^i sin(\phi^i/2) \\ \\ \end{bmatrix} \tag{3.1}$$

where $\mathbf{p}^i$ is the 4×1 vector of Euler parameters. Since only three coordinates are needed to define the orientation of a body, Euler parameters are not independent. They are related by the constraint [5]

$$(\mathbf{p}^i)^T \mathbf{p}^i = e_0^2 + e_1^2 + e_2^2 + e_3^2 = 1 \tag{3.2}$$

This thesis will use Euler parameters for orientation coordinates.

Once the orientation coordinates have been defined, transformation matrices between bodies can be obtained. For a body i in a multibody chain, the transfor-

mation matrix from body i to the reference frame R can be defined as

$$\mathbf{A}^{Ri} = \mathbf{A}^{R1} \cdot \mathbf{A}^{12} \ldots \mathbf{A}^{i-1\,i} \tag{3.3}$$

where $\mathbf{A}^{i-1\,i}$ is the transformation matrix from body i to the preceding body, $i-1$, located in the multibody chain. Transformation matrices are needed to define global positions of points in space. Shown in Figure 3.1 is a point Q fixed in body i. Three position coordinates, which define the location of point Q, can be represented as

$$\mathbf{r}^{RQ} = \mathbf{r}^{RO} + \mathbf{A}^{Ri}\,\mathbf{s}^{iQ} \tag{3.4}$$

where $\mathbf{r}^{RQ}$ is the vector position of point Q with respect to the reference frame R, $\mathbf{r}^{RO}$ is the vector position of point O with respect to the reference frame R, and $\mathbf{s}^{iQ}$ is the local vector position from point O to Q defined in body i fixed coordinates.

The velocity of point Q is obtained by differentiating equation 3.4 with respect to time to yield

$$\dot{\mathbf{r}}^{RQ} = \dot{\mathbf{r}}^{RO} + \tilde{\boldsymbol{\omega}}^{Ri}\,(\mathbf{A}^{Ri}\,\mathbf{s}^{iQ}) \tag{3.5}$$

where $\dot{\mathbf{r}}^{RQ}$ is the vector velocity of point Q with respect to the reference frame R. The matrix product $\tilde{\boldsymbol{\omega}}\mathbf{p}$, where $\mathbf{p}$ is any vector, is equivalent to the cross product $\boldsymbol{\omega} \times \mathbf{p}$. The term $\tilde{\boldsymbol{\omega}}^{Ri}$ is defined as

$$\tilde{\boldsymbol{\omega}}^{Ri} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{3.6}$$

The components ω_x, ω_y, and ω_z represent the angular velocity components of body

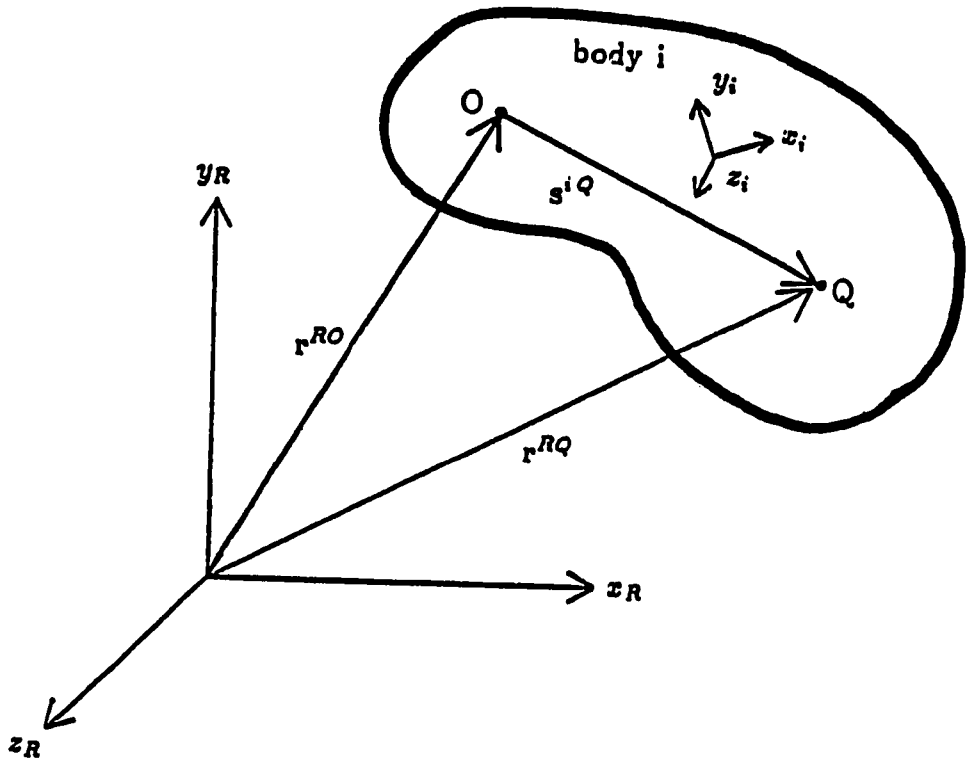


Figure 3.1: Definition of position vectors

i with respect to R. Equation 3.6 can now be written as

$$\mathbf{v}^{RQ} = \mathbf{v}^{RO} + \tilde{\boldsymbol{\omega}}^{Ri}\ {}_{\mathbf{A}}R^{i}\ {}_{\mathbf{s}}{}^{i}Q \tag{3.7}$$

where $\mathbf{v}^{RQ} = \frac{d}{dt}(\mathbf{r}^{RQ}) = \dot{\mathbf{r}}^{RQ}$.

The complete velocity components of the ith body can be represented by including the linear and angular velocity components in one Cartesian velocity vector defined as

$$\dot{\mathbf{y}}^{Ri} = \left[\ {}_{\mathbf{v}}R^{i T} \quad {}_{\boldsymbol{\omega}}R^{i T}\ \right]^{T}_{(6\times 1)} \tag{3.8}$$

where $\dot{\mathbf{y}}^{Ri}$ is the Cartesian velocity vector of body i with respect to reference frame R. The Cartesian velocity vector for a multibody system with nb bodies can then be assembled as

$$\dot{\mathbf{y}}^{R} = \left[\ \dot{\mathbf{y}}^{R1 T} \quad \dot{\mathbf{y}}^{R2 T} \quad \cdots \quad \dot{\mathbf{y}}^{R nb T}\ \right]^{T}_{(6nb \times 1)} \tag{3.9}$$

The Cartesian acceleration vectors are time differentiations of equations 3.8 and 3.9 defined as

$$\ddot{\mathbf{y}}^{Ri} = \left[\ \dot{\mathbf{v}}^{Ri T} \quad \dot{\boldsymbol{\omega}}^{Ri T}\ \right]^{T}_{(6\times 1)} \tag{3.10}$$

$$\ddot{\mathbf{y}}^{R} = \left[\ \ddot{\mathbf{y}}^{R1 T} \quad \ddot{\mathbf{y}}^{R2 T} \quad \cdots \quad \ddot{\mathbf{y}}^{R nb T}\ \right]^{T}_{(6nb \times 1)} \tag{3.11}$$

3.2 Graph Theory

Graph theory is an effective tool for generating equations of motion for large-scale multibody systems. This approach provides a mathematical representation of the connection of the rigid bodies. Wittenburg and others [1,3,6,23] have used graph theory to define the topological structure of mechanical systems.

The topology of a system defines the order and connection of bodies. The analysis starts by numbering bodies 1 through nb. A base body is then chosen as the main body of the system. For systems fixed to a nonmoving reference frame, the base body is defined as the ground. For systems which float, such as an automobile, the base body is a floating body and can be any body in the system.

The connection of the rigid bodies is defined with respect to the base body. This representation is defined using two matrices, the path matrix and the reference matrix. The $nb \times nb$ path matrix is defined as

$$\pi_{ij} = \begin{cases} 1 & \text{if body } j \text{ is between the base body and the } ith \text{ body} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

The $nb \times nb$ reference matrix is defined as

$$\rho_{ij} = \begin{cases} 1 & \text{if } i = j \\ -1 & \text{if body } j \text{ is a reference body for body } i \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

The ith row of the path matrix determines the route or path from the ith body to the base body. The ith row of the reference matrix determines which body precedes the ith body. These two matrices have the interesting relationship that [6]

$$\rho\pi = \pi\rho = \mathbf{I}_{nb} \tag{3.14}$$

where $\mathbf{I}_{nb}$ is an $nb \times nb$ identity matrix.

Consider, as an example, the satellite system shown in Figure 3.2. The system contains a floating base body labelled body 1 and two arms containing rotational

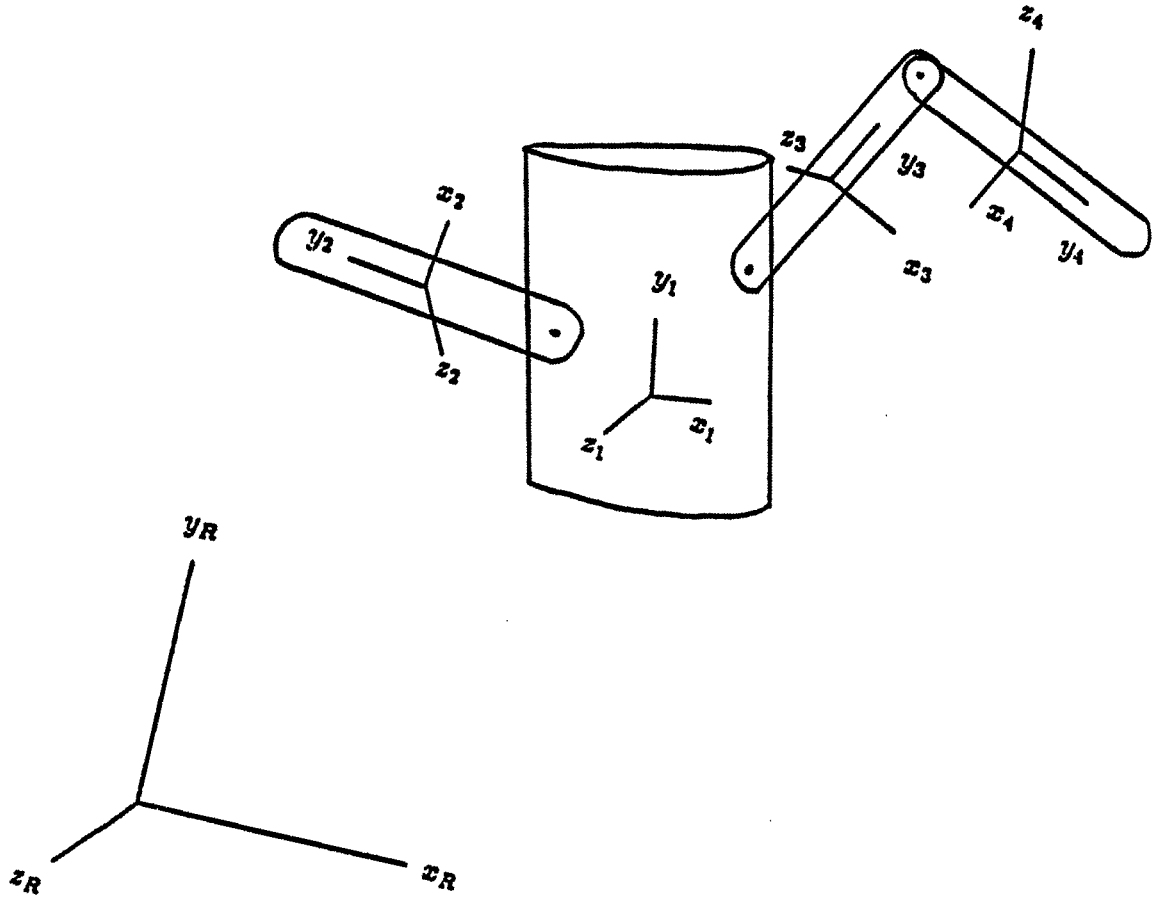


Figure 3.2: Multiple, rigid body satellite system

joints connected to body 1. The path matrix for this system is defined as

$$\pi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \tag{3.15}$$

and the reference matrix is defined as

$$\rho = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \tag{3.16}$$

Notice from the figure that the path from body 1 to body 4 must go through body 3. The fourth row of the path matrix shows mathematically this relationship. Also, from the fourth row of the reference matrix, body 3 precedes body 4.

Another important matrix in system topology is the distance matrix [6]. The distance matrix assigns a numerical value to a particular type of joint. This $nb \times nb$ matrix is defined as

$$\mathbf{DM}_{ij} = \begin{cases} 0.0 & \text{if body } i \text{ and } j \text{ are not connected} \\ 0.2 & \text{if body } i = j \text{ is a floating base body} \\ 1.0 & \text{if body } i \text{ and } j \text{ are connected by a revolute joint} \\ 1.1 & \text{if body } i \text{ and } j \text{ are connected by a translational joint} \\ 3.0 & \text{if body } i \text{ and } j \text{ are connected by a spherical joint} \end{cases} \tag{3.17}$$

For the example shown in Figure 3.1, the distance matrix is defined as

$$\mathbf{DM} = \begin{bmatrix} 0.2 & 1.0 & 1.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix} \tag{3.18}$$

The distance matrix will prove useful in formulation of the equations of motion and determining the number of generalized coordinates assigned to a joint.

3.3 Position Information for Multibody Systems

The formation of the position vectors is the first step in analyzing the kinematic equations for multibody systems. Consider a general multibody system shown in Figure 3.3. Body i is connected to body j through joint i and body j is connected to body k through joint j. The local position vector from joint j to body j center of gravity is defined as $\mathbf{d}^{jj}$. The local position vector from body j center of gravity to joint i is defined as $\mathbf{s}^{ij}$. All the position information for the kinematic analysis has been determined when $\mathbf{d}^{jj}$ and $\mathbf{s}^{ij}$ have been defined in each body of the system.

Quantities such as $\mathbf{d}^{ij}$, the position vector from joint j to body i center of gravity, are then calculated as

$$\mathbf{d}^{ij} = \mathbf{d}^{jj} + \mathbf{s}^{ij} + \mathbf{d}^{ii} \tag{3.19}$$

Figure 3.3 presents this vector graphically. The vector addition in equation 3.19 cannot be performed directly since $\mathbf{d}^{jj}$ and $\mathbf{s}^{ij}$ are defined with respect to body j coordinates, while $\mathbf{d}^{ii}$ is defined with respect to body i coordinates. Thus, $\mathbf{d}^{ij}$,

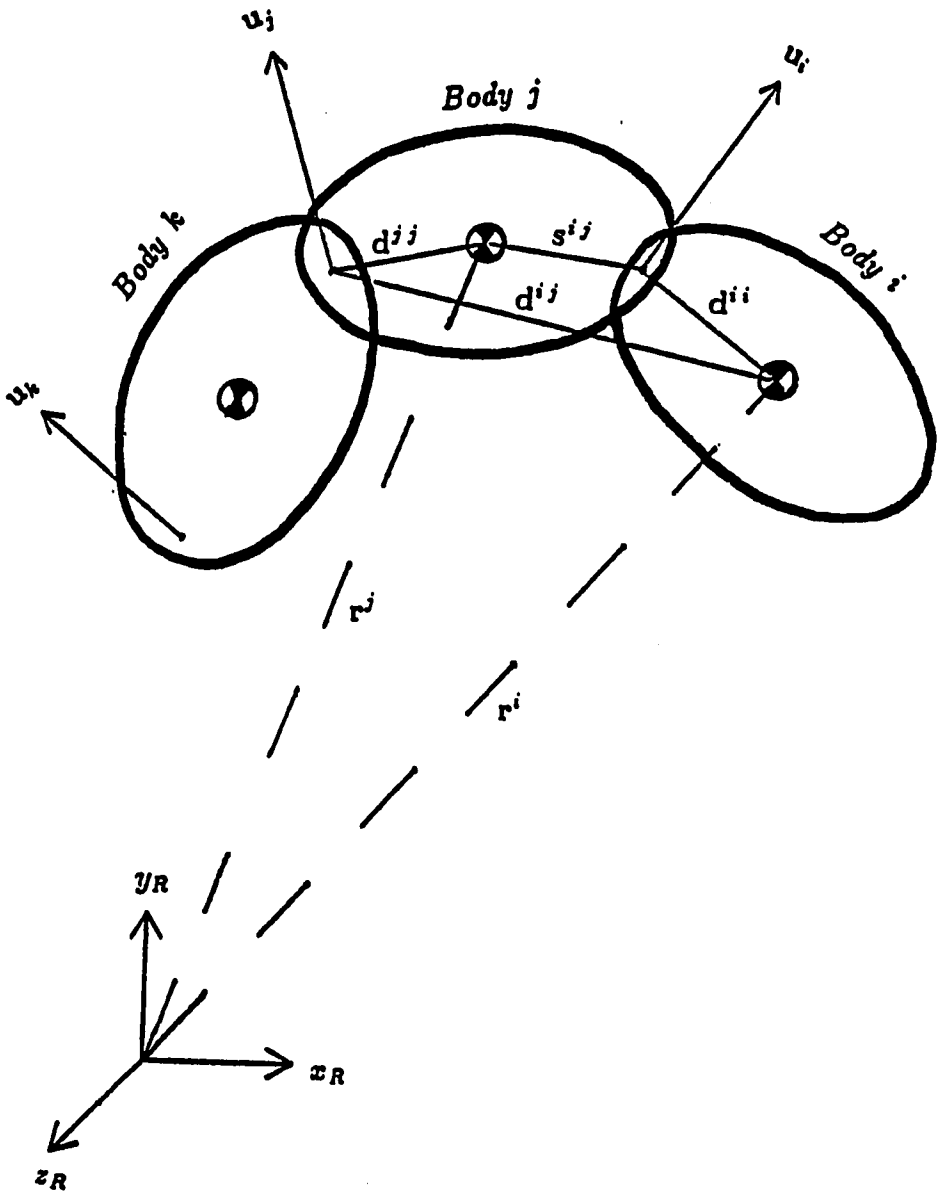


Figure 3.3: Distance vector definition

defined in body i coordinates, is given as

$$\mathbf{d}^{ij} = \mathbf{A}^{ij}(\mathbf{d}^{jj} + \mathbf{s}^{ij}) + \mathbf{d}^{ii} \tag{3.20}$$

where $\mathbf{A}^{ij}$ is the transformation matrix from body j to body i. The quantity $\mathbf{d}^{ij}$ will prove to be very useful in the formulation of the velocity and acceleration equations.

In order to generalize the calculation of $\mathbf{d}^{ij}$ for any body i or j in the system, the path and reference matrices are needed. For bodies adjacent to one another, the $\mathbf{d}^{ij}$ vector with respect to body i coordinates is defined as

$$\mathbf{d}^{ij} = \mathbf{A}^{ij}(\mathbf{d}^{jj} + \mathbf{s}^{ij}) + \mathbf{d}^{ii} \qquad \text{if } \rho_{ij} = -1 \tag{3.21}$$

where i and j range from 1 through nb. For bodies not connected together

$$\mathbf{d}^{ij} = \pi_{ij}\left[\ \mathbf{A}^{ik}(\mathbf{d}^{kj} + \mathbf{s}^{ik}) + \mathbf{d}^{ii}\ \right] \qquad \text{if } \rho_{ik} = -1 \text{ and } \rho_{ij} = 0 \tag{3.22}$$

where i, j, and k range from 1 through nb.

3.4 Formation of Cartesian Velocity Vector

The main quantities used in kinematics are the angular and linear velocity components. Angular velocity will be discussed first since it is needed for the solution of linear velocities.

Consider the multibody system shown in Figure 3.3. The angular velocity of body i is defined as

$$\omega^i = \omega^j + (1 - \epsilon^i)\dot{q}^i \mathbf{u}^i \tag{3.23}$$

where

$$\epsilon^i = \begin{cases} 0 & \text{if joint i is a revolute joint} \\ 1 & \text{if joint i is a translational joint} \end{cases} \tag{3.24}$$

and $\mathbf{u}^i$ is the unit vector about which the *ith* generalized coordinate q^i rotates. If the angular velocity of body j is substituted in the right hand side of equation 3.23 then $\boldsymbol{\omega}^i$ becomes

$$\boldsymbol{\omega}^i = \boldsymbol{\omega}^k + (1 - \epsilon^j)\dot{q}^j \mathbf{u}^j + (1 - \epsilon^i)\dot{q}^i \mathbf{u}^i \tag{3.25}$$

Performing this recursion until the base body is reached, the angular velocity can be formed in terms of the path matrix as [6]

$$\boldsymbol{\omega}^i = \sum_{j=1}^{nb} \pi_{ij}\, \boldsymbol{\Omega}^{ij} \tag{3.26}$$

where π_{ij} is the i, j component of the path matrix $\boldsymbol{\pi}$. The quantity $\boldsymbol{\Omega}^{ij}$ can be determined for any joint as

$$\boldsymbol{\Omega}^{ij} = \begin{cases} \boldsymbol{\omega}^j & \text{if } \mathbf{DM}_{ij} = 0.2, \text{ a floating base body} \\ \sum_k \dot{q}_k^j \mathbf{u}_k^j & \text{if } \mathbf{DM}_{ij} = 1.0, \text{ a revolute joint } (k = 1) \\ \mathbf{0} & \text{if } \mathbf{DM}_{ij} = 1.1, \text{ a translational joint} \end{cases} \tag{3.27}$$

Equation 3.27 can be easily extended to include universal or spherical joints by setting k equal to 2 or 3, respectively. Cylindrical joints can be formed using the combination of revolute and translational joints.

The linear velocity components will be formed by the same recursive process. Referring to Figure 3.3, the linear position of body i is found as

$$\mathbf{r}^i = \mathbf{r}^j + \epsilon^i q^i \mathbf{u}^i + \mathbf{s}^{ij} + \mathbf{d}^{ii} \tag{3.28}$$

the linear velocity of body i is found by differentiating equation 3.28 with respect to time and is given as

$$\mathbf{v}^i = \mathbf{v}^j + \epsilon^i \dot{q}^i \mathbf{u}^i + \tilde{\boldsymbol{\omega}}^i \epsilon^i q^i \mathbf{u}^i + \tilde{\boldsymbol{\omega}}^j \mathbf{s}^{ij} + \tilde{\boldsymbol{\omega}}^i \mathbf{d}^{ii} \tag{3.29}$$

where $\mathbf{v}^j$ is the linear velocity of the center of gravity of body j and ω^i and ω^j are the angular velocities of body i and j, respectively. The quantities $\mathbf{v}^j$ and ω^i can be substituted into the right hand side of equation 3.29 and $\mathbf{v}^i$ is then found as

$$\begin{aligned}\mathbf{v}^i &= \mathbf{v}^k + \epsilon^j \dot{q}^j \mathbf{u}^j + \epsilon^i \dot{q}^i \mathbf{u}^i + \tilde{\omega}^k \mathbf{s}^{jk} + \\ &\quad \tilde{\omega}^j \mathbf{d}^{ij} + (1-\epsilon^i)\dot{q}^i \tilde{\mathbf{u}}^i \mathbf{d}^{ii}\end{aligned} \tag{3.30}$$

Continue substituting $\mathbf{v}^k$ and ω^j until body k becomes the base body. The linear velocity of body i at the center of gravity is then found as

$$\mathbf{v}^i = \sum_{j=1}^{nb} \pi_{ij} \left(\Omega^{ij} \mathbf{d}^{ij} + \nu^{ij} \right) \tag{3.31}$$

where Ω^{ij} is given in equation 3.27 and $\mathbf{d}^{ij}$ is given in equations 3.21 and 3.22. The term ν^{ij} accounts for translating coordinates and floating base bodies and is given as

$$\nu^{ij} = \begin{cases} \mathbf{v}^j & \text{if } \mathbf{DM}_{ij} = 0.2, \text{ a floating base body} \\ \mathbf{0} & \text{if } \mathbf{DM}_{ij} = 1.0, \text{ a revolute joint} \\ \dot{q}^j \mathbf{u}^j & \text{if } \mathbf{DM}_{ij} = 1.1, \text{ a translational joint} \end{cases} \tag{3.32}$$

Once $\mathbf{v}^i$ and ω^i are formed in each body, the Cartesian velocity vectors can be defined as in equations 3.8 and 3.9.

3.5 Formation of the Partial Velocity Matrix

The Cartesian velocity vector represents the linear velocity at the center of gravity and the angular velocity of each body. The total derivative of the Cartesian position vector of a system with *ndf* generalized coordinates can be represented as

[24]

$$\dot{\mathbf{y}}^R = \frac{\partial \mathbf{y}^R}{\partial \mathbf{q}}\dot{\mathbf{q}} + \frac{\partial \mathbf{y}^R}{\partial t} \tag{3.33}$$

where $\dot{\mathbf{q}}$ is an *ndf* vector.

Since $\mathbf{y}$ is only a function of the generalized coordinates and not explicitly a function of time, then equation 3.33 becomes

$$\dot{\mathbf{y}}^R = \frac{\partial \mathbf{y}^R}{\partial \mathbf{q}}\dot{\mathbf{q}} = \mathbf{B}(\mathbf{q})\dot{\mathbf{q}} \tag{3.34}$$

where $\mathbf{B}(\mathbf{q})$ is the partial velocity matrix. The $\mathbf{B}(\mathbf{q})$ matrix has the dimensions $6nb \times ndf$ for a floating system system and $6(nb-1) \times ndf$ for a grounded system. This matrix will be used to convert coordinates specified in the Cartesian space into relative generalized coordinates.

Substituting the solutions for $\boldsymbol{\omega}^i$ and $\mathbf{v}^i$ from equations 3.26 and 3.31 respectively, into equation 3.34 results in [6]

$$\dot{\mathbf{y}}^i = \begin{bmatrix} \mathbf{v}^i \\ \boldsymbol{\omega}^i \end{bmatrix} = \sum_{j=1}^{nb} \mathbf{B}_{ij}\dot{\mathbf{q}}^j \tag{3.35}$$

where the joint between the ith and jth bodies constitute k generalized coordinates, $\mathbf{B}_{ij}$ is a $6 \times k$ submatrix and $\dot{\mathbf{q}}^j$ is a k generalized velocity vector. The submatrices, $\mathbf{B}_{ij}$, are a function of the type of joint and the system topology. Table 3.1 gives the $\mathbf{B}_{ij}$ submatrices in local body i coordinates for 3 commonly used types of joints. The path matrix, π, determines if the $\mathbf{B}_{ij}$ submatrices have nonzero entries [6].

As an example of the partial velocity matrix, consider the satellite system discussed previously and shown in Figure 3.2. The path and reference matrices are given by equations 3.15 and 3.16. The $\mathbf{B}$ matrix is a 24×9 matrix defined as in Table 3.2. Notice that the $\mathbf{B}$ matrix has been partitioned into a 4×4 matrix of

Table 3.1: $\mathbf{B}_{ij}$ Submatrices Defined in Body i Coordinates

Joint	$\mathbf{B}_{ij}$
Floating Base	$\begin{bmatrix} \mathbf{A}^{ij} & -\tilde{\mathbf{d}}^{ij}\mathbf{A}^{ij} \\ \mathbf{0} & \mathbf{A}^{ij} \end{bmatrix}^{(6\times 6)}$
Revolute	$\begin{bmatrix} \widetilde{\mathbf{A}^{ij}\mathbf{u}^j}\,\mathbf{d}^{ij} \\ \mathbf{A}^{ij}\mathbf{u}^j \end{bmatrix}^{(6\times 1)}$
Translational	$\begin{bmatrix} \mathbf{A}^{ij}\mathbf{u}^j \\ \mathbf{0} \end{bmatrix}^{(6\times 1)}$

submatrices $\mathbf{B}_{ij}$ and has the same dimension as the path matrix. Also, note that the nonzero entries in the **B** matrix are identical to the nonzero entries of the path matrix. As will be shown in the next chapter, the partial velocity matrix is an important calculation in formulation of the equations of motion.

Table 3.2: **B** matrix for satellite system

$$\mathbf{B} = \left[\begin{array}{cc|c|c|c} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{A}^{21} & -\tilde{\mathbf{d}}^{21}\mathbf{A}^{21} & \tilde{\mathbf{u}}^2\mathbf{d}^{22} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{21} & \mathbf{u}^2 & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{A}^{31} & -\tilde{\mathbf{d}}^{31}\mathbf{A}^{31} & \mathbf{0} & \tilde{\mathbf{u}}^3\mathbf{d}^{33} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{31} & \mathbf{0} & \mathbf{u}^3 & \mathbf{0} \\ \hline \mathbf{A}^{41} & -\tilde{\mathbf{d}}^{41}\mathbf{A}^{41} & \mathbf{0} & \widetilde{\mathbf{A}^{43}\mathbf{u}^3}\,\mathbf{d}^{43} & \tilde{\mathbf{u}}^4\mathbf{d}^{44} \\ \mathbf{0} & \mathbf{A}^{41} & \mathbf{0} & \mathbf{A}^{43}\mathbf{u}^3 & \mathbf{u}^4 \end{array}\right] \quad (24 \times 9)$$

3.6 Cartesian Accelerations

Cartesian accelerations are found by taking a time differentiation of equation 3.34

$$\ddot{\mathbf{y}}^R = \mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{B}}(\mathbf{q})\dot{\mathbf{q}} \tag{3.36}$$

where $\dot{\mathbf{B}}(\mathbf{q})$ is the time differentiation of $\mathbf{B}(\mathbf{q})$ and $\ddot{\mathbf{q}}$ is the *ndf* generalized acceleration vector. The $\dot{\mathbf{B}}(\mathbf{q})$ matrix has the same dimension and nonzero entries as the $\mathbf{B}(\mathbf{q})$ matrix. Table 3.3 gives the $\dot{\mathbf{B}}_{ij}$ submatrices for three typical joints. Note that $\dot{\mathbf{B}}_{ij}$ is not a strict time differentiation of $\mathbf{B}_{ij}$ since $\mathbf{B}_{ij}$ is expressed in terms of body i coordinates. These submatrices are differentiated to account for the moving i reference frame.

Table 3.3: $\dot{\mathbf{B}}_{ij}$ Submatrices Defined in Body i Coordinates

Joint	$\mathbf{B}_{ij}$
Floating Base	$\begin{bmatrix} \dot{\mathbf{A}}^{ij} + \tilde{\omega}^i \mathbf{A}^{ij} & -(\dot{\tilde{\mathbf{d}}}^{ij} \mathbf{A}^{ij} + \tilde{\mathbf{d}}^{ij} \dot{\mathbf{A}}^{ij} + \tilde{\omega}^i \tilde{\mathbf{d}}^{ij} \mathbf{A}^{ij}) \\ \mathbf{0} & \dot{\mathbf{A}}^{ij} + \tilde{\omega}^i \mathbf{A}^{ij} \end{bmatrix}^{(6 \times 6)}$
Revolute	$\begin{bmatrix} \dot{\widetilde{\mathbf{A}^{ij}\mathbf{u}^j}}\, \mathbf{d}^{ij} + \widetilde{\mathbf{A}^{ij}\mathbf{u}^j}\, \dot{\mathbf{d}}^{ij} + \tilde{\omega}^i \widetilde{\mathbf{A}^{ij}\mathbf{u}^j}\, \mathbf{d}^{ij} \\ \frac{d}{dt}(\mathbf{A}^{ij}\mathbf{u}^j) + \tilde{\omega}^i \mathbf{A}^{ij}\mathbf{u}^j \end{bmatrix}^{(6 \times 1)}$
Translational	$\begin{bmatrix} \frac{d}{dt}(\mathbf{A}^{ij}\mathbf{u}^j) + \tilde{\omega}^i \mathbf{A}^{ij}\mathbf{u}^j \\ \mathbf{0} \end{bmatrix}^{(6 \times 1)}$

4 DYNAMICS OF MULTIBODY SYSTEMS

This chapter presents the formulation for the dynamics of multibody systems. Dynamics is subdivided into kinematics and kinetics. Chapter 3 has already defined the kinematic equations needed to analyze multibody systems. Kinetics is defined as the study of motion and the forces causing motion.

Several methods are available to formulate equations for multibody systems. The more popular methods include: Newton-Euler, Lagrange, and Kane's equations. The Newton-Euler method is based on a vector approach. The advantage of the Newton-Euler method is that an intuitive set of dependent Cartesian coordinates is normally used to define kinematic quantities and force terms. A large set of differential equations results from the use of the Cartesian coordinates. The disadvantage is that constraint equations must be introduced in order to solve the equations. This results in a large set of mixed differential-algebraic equations which are numerically difficult to solve. An independent set of coordinates can be obtained, but difficult substitutions must be performed to eliminate nonworking constraint forces acting between bodies.

Lagrange's method is based on an energy approach. This method usually uses an independent set of relative coordinates and results in the minimal number of differential equations. A dependent set of Cartesian coordinates can be used by

introducing Lagrange multiplier constraints. The disadvantage of this approach is that excessive differentiations must be performed and large intermediate terms are generated.

Kane's method combines the benefits of both the Newton-Euler and Lagrange methods and will be used to form the equations of motion in this chapter. Kane's method starts by defining the system with a dependent set of Cartesian coordinates and transforms them into equations in terms of an independent set of relative coordinates. This transformation is accomplished using the partial velocity matrix which is defined by system topology presented in Chapter 3. The advantage of this approach is that kinematic quantities and force terms can be represented using the intuitive set of Cartesian coordinates, however, the resulting equations of motion are in terms of a minimal, efficient set of generalized coordinates.

Section 1 of this chapter introduces Kane's equations. Section 2 derives a generalization of Kane's equations for mechanical systems. Section 3 derives the additional terms to include actuator forces in the dynamic model. Section 4 discusses symbolic computing of the equations of motion. Section 5 provides a brief algorithm for generation of symbolic equations of motion based on Kane's equations. Section 6 discusses numerical efficiency when using symbolic equations. Finally, Section 7 extends symbolic analysis in substructuring mechanical systems.

4.1 Kane's Equations for Dynamic Systems

If there exists a holonomic system with *nb* rigid bodies possessing *ndf* degrees of freedom in a Newtonian reference frame, then the equations that govern all motions

of the system are given by

$$F_j + F_j^* = 0 \qquad j = 1, \cdots, ndf \tag{4.1}$$

Equations 4.1 are known as Kane's dynamical equations [24]. The quantities F_j and F_j^* are defined as the holonomic generalized active force and holonomic generalized inertia force, respectively. F_j is given as

$$F_j = \sum_{i=1}^{nb} \{(\frac{\partial \omega^i}{\partial \dot{q}_j})^T \tau^i + (\frac{\partial \mathbf{v}^i}{\partial \dot{q}_j})^T \mathbf{r}^i\} \tag{4.2}$$

where $\dot{q}_j$ is defined as the generalized velocity corresponding to the generalized coordinate q_j. The terms $\partial \omega^i / \partial \dot{q}_j$ and $\partial \mathbf{v}^i / \partial \dot{q}_j$ are the partial angular velocities and partial linear velocities of body i with respect to $\dot{q}_j$, respectively. τ^i is a Cartesian vector which is the sum of all external torques acting on body i. $\mathbf{r}^i$ is a Cartesian vector which is the resultant of all external forces which act on body i.

The holonomic generalized inertia force, F_j^*, is given by

$$F_j^* = \sum_{i=1}^{nb} \{(\frac{\partial \omega^i}{\partial \dot{q}_j})^T \tau^{i*} + (\frac{\partial \mathbf{v}^i}{\partial \dot{q}_j})^T \mathbf{r}^{i*}\} \tag{4.3}$$

τ^{i*} is defined as body i inertia torque and $\mathbf{r}^{i*}$ is defined as body i inertia force.

The inertia torque can be represented using Cartesian coordinates as

$$\tau^{i*} = -\mathbf{J}^i \dot{\omega}^i - \tilde{\omega}^i \mathbf{J}^i \omega^i \tag{4.4}$$

where $\mathbf{J}^i$ is a 3×3 inertia matrix for body i and $\dot{\omega}^i$ is the angular acceleration. The Cartesian inertia force, $\mathbf{r}^{i*}$, is given as

$$\mathbf{r}^{i*} = -m_i \dot{\mathbf{v}}^i \tag{4.5}$$

where m_i is the mass of body i and $\dot{\mathbf{v}}^i$ is the linear acceleration at the mass center.

The acceleration terms are obtained as

$$\dot{\boldsymbol{\omega}}^i = \frac{d}{dt}(\boldsymbol{\omega}^i) = \frac{d}{dt}(\frac{\partial \boldsymbol{\omega}^i}{\partial \dot{\mathbf{q}}})\dot{\mathbf{q}} + \frac{\partial \boldsymbol{\omega}^i}{\partial \dot{\mathbf{q}}}\ddot{\mathbf{q}} \tag{4.6}$$

and

$$\dot{\mathbf{v}}^i = \frac{d}{dt}(\mathbf{v}^i) = \frac{d}{dt}(\frac{\partial \mathbf{v}^i}{\partial \dot{\mathbf{q}}})\dot{\mathbf{q}} + \frac{\partial \mathbf{v}^i}{\partial \dot{\mathbf{q}}}\ddot{\mathbf{q}} \tag{4.7}$$

substituting 4.4, 4.5, 4.6, and 4.7 into 4.3, the holonomic generalized inertia force becomes

$$\begin{aligned} F_j^* \;=\; & -\sum_{i=1}^{nb} \{\, [(\frac{\partial \boldsymbol{\omega}^i}{\partial \dot{q}_j})^T \mathbf{J}^i (\frac{\partial \boldsymbol{\omega}^i}{\partial \dot{q}_j}) + (\frac{\partial \mathbf{v}^i}{\partial \dot{q}_j})^T \mathbf{m}^i (\frac{\partial \mathbf{v}^i}{\partial \dot{q}_j})]\ddot{q}_j \\ & + [(\frac{\partial \boldsymbol{\omega}^i}{\partial \dot{q}_j})^T \mathbf{J}^i \frac{d}{dt}(\frac{\partial \boldsymbol{\omega}^i}{\partial \dot{q}_j}) + (\frac{\partial \mathbf{v}^i}{\partial \dot{q}_j})^T \mathbf{m}^i \frac{d}{dt}(\frac{\partial \mathbf{v}^i}{\partial \dot{q}_j})]\dot{q}_j \\ & + (\frac{\partial \boldsymbol{\omega}^i}{\partial \dot{q}_j})^T (\tilde{\boldsymbol{\omega}}^i \mathbf{J}^i \boldsymbol{\omega}^i) \,\} \end{aligned} \tag{4.8}$$

4.2 Generalization of Kane's Equations for Mechanical Systems

The equations for the holonomic generalized active forces and holonomic generalized inertia forces given by equations 4.2 and 4.8, respectively, can be substituted into equation 4.1 to arrive at the equations of motion. A simpler form of equations 4.2 and 4.8 can be obtained by substituting in the Cartesian velocity vectors, $\dot{\mathbf{y}}$, as in equation 3.8. Then, the holonomic generalized active forces and holonomic generalized inertia forces become

$$F_j = \sum_{i=1}^{nb} \{\, (\frac{\partial \dot{\mathbf{y}}^i}{\partial \dot{q}_j})^T \mathbf{f}^i \,\} \tag{4.9}$$

and

$$
\begin{aligned}
F_j^* &= -\sum_{i=1}^{nb}\{\,[(\frac{\partial \dot{\mathbf{y}}^i}{\partial \dot{q}_j})^T\,\mathbf{M}^i\,(\frac{\partial \dot{\mathbf{y}}^i}{\partial \dot{q}_j})]\ddot{q}_j \\
&+ [(\frac{\partial \dot{\mathbf{y}}^i}{\partial \dot{q}_j})^T\,\mathbf{M}^i\,\frac{d}{dt}(\frac{\partial \dot{\mathbf{y}}^i}{\partial \dot{q}_j})]\dot{q}_j \\
&+ (\frac{\partial \dot{\mathbf{y}}^i}{\partial \dot{q}_j})^T\,\mathbf{h}^i\,\}
\end{aligned}
\tag{4.10}
$$

The quantity, $\mathbf{f}^i$, is defined as the body i Cartesian force vector

$$
\mathbf{f}^i = \begin{bmatrix} \mathbf{r}^{iT} & \boldsymbol{\tau}^{iT} \end{bmatrix}^T_{(6\times 1)} \tag{4.11}
$$

The body i mass matrix, $\mathbf{M}^i$, is defined as

$$
\mathbf{M}^i = \left[\begin{array}{ccc|c} m^i & 0 & 0 & \\ 0 & m^i & 0 & \mathbf{0} \\ 0 & 0 & m^i & \\ \hline & \mathbf{0} & & \mathbf{J}^i \end{array}\right]_{(6\times 6)} \tag{4.12}
$$

and the quantity, $\mathbf{h}^i$, the Cartesian Coriolis and centrifugal force vector for body i, is defined as

$$
\mathbf{h}^i = \begin{bmatrix} \mathbf{0}^T & (\tilde{\boldsymbol{\omega}}^i \mathbf{J}^i \boldsymbol{\omega}^i)^T \end{bmatrix}^T_{(6\times 1)} \tag{4.13}
$$

Finally, equations 4.9 and 4.10 are added to form Kane's equations and put into matrix form as [6]

$$
[\mathbf{B}^T\mathbf{M}\mathbf{B}]\ddot{\mathbf{q}} = \mathbf{B}^T[\mathbf{f} - \mathbf{M}\dot{\mathbf{B}}\dot{\mathbf{q}} - \mathbf{h}] \tag{4.14}
$$

where $\mathbf{B}$ and $\dot{\mathbf{B}}$ are defined by equations 3.34 and 3.36. $\mathbf{M}$ is the system mass matrix given as

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}^1 & & & \mathbf{0} \\ & \mathbf{M}^2 & & \\ & & \ddots & \\ \mathbf{0} & & & \mathbf{M}^{nb} \end{bmatrix}_{(6nb \times 6nb)} \tag{4.15}$$

The quantity $\mathbf{f}$ is defined as the system Cartesian force vector

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}^{1T} & \mathbf{f}^{2T} & \cdots & \mathbf{f}^{nbT} \end{bmatrix}^T_{(6nb \times 1)} \tag{4.16}$$

and, $\mathbf{h}$, the system Cartesian Coriolis and centrifugal force vector, is given as

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}^{1T} & \mathbf{h}^{2T} & \cdots & \mathbf{h}^{nbT} \end{bmatrix}^T_{(6nb \times 1)} \tag{4.17}$$

Equation 4.14 represents an efficient representation of Kane's equations for multi-body dynamic systems.

This equation can be extended to include closed-loops by introducing constraint equations. Assume there are m_d independent constraint equations of the form

$$\Phi_i(\mathbf{q}, t) = 0 \qquad i = 1, \cdots, m_d \tag{4.18}$$

then equation 4.14 is given as [6]

$$[\mathbf{B}^T\mathbf{M}\mathbf{B}]\ddot{\mathbf{q}} \;=\; \mathbf{B}^T[\mathbf{f} - \mathbf{M}\dot{\mathbf{B}}\dot{\mathbf{q}} - \mathbf{h}] - \mathbf{\Phi}_\mathbf{q}^T\boldsymbol{\lambda} \tag{4.19}$$

where $\mathbf{\Phi}_\mathbf{q}^T$ is an $ndf \times m_d$ constraint Jacobian matrix defined as

$$\Phi_{\mathbf{q}\,ij} = \frac{\partial \Phi_i}{\partial q_j} \tag{4.20}$$

and $\boldsymbol{\lambda}$ is an m_d vector of Lagrange multipliers [6].

4.3 Spring-Damper-Actuator Forces

In order to model general mechanical systems, the formulation must include elements such as springs, dampers, and actuators between joints and connected rigid bodies. Using the formulation presented in this thesis, these forces are defined in the intuitive Cartesian space and input in the system force vector, **f**, in equation 4.16. For modeling convenience, these three elements are modelled as a combined force element for two joint movements: rotation and translation.

For rotation, a rotational spring-damper-actuator (RSDA) element is used to model the associated body torques. This element is typically associated with a revolute joint. Figure 4.1 shows an RSDA element acting between bodies i and j. The torque associated with joint i can be represented as [6]

$$f_R^{ij} = k^{ij}(q^i - \alpha^i) + c^{ij}\dot{q}^i + \tau^{ij} \,. \tag{4.21}$$

where k^{ij} is the spring constant, c^{ij} is the damping coefficient, and τ^{ij} is the external torque applied at joint i. The terms q^i and $\dot{q}^i$ are the relative joint angle and relative joint rate of body i, respectively, and α^i is the initial undeformed length of the spring. Equation 4.21 can be put into body i and j Cartesian force vectors as

$$\mathbf{f}^i = -f_R^{ij} \begin{bmatrix} \mathbf{0} \\ \mathbf{u}^i \end{bmatrix}_{(6\times 1)} \tag{4.22}$$

$$\mathbf{f}^j = f_R^{ij} \begin{bmatrix} \mathbf{0} \\ \mathbf{A}^{ji}\,\mathbf{u}^i \end{bmatrix}_{(6\times 1)} \tag{4.23}$$

where $\mathbf{u}^i$ is the unit vector from which the body i rotates and $\mathbf{A}^{ji}$ is the transformation matrix from body i to body j coordinates.

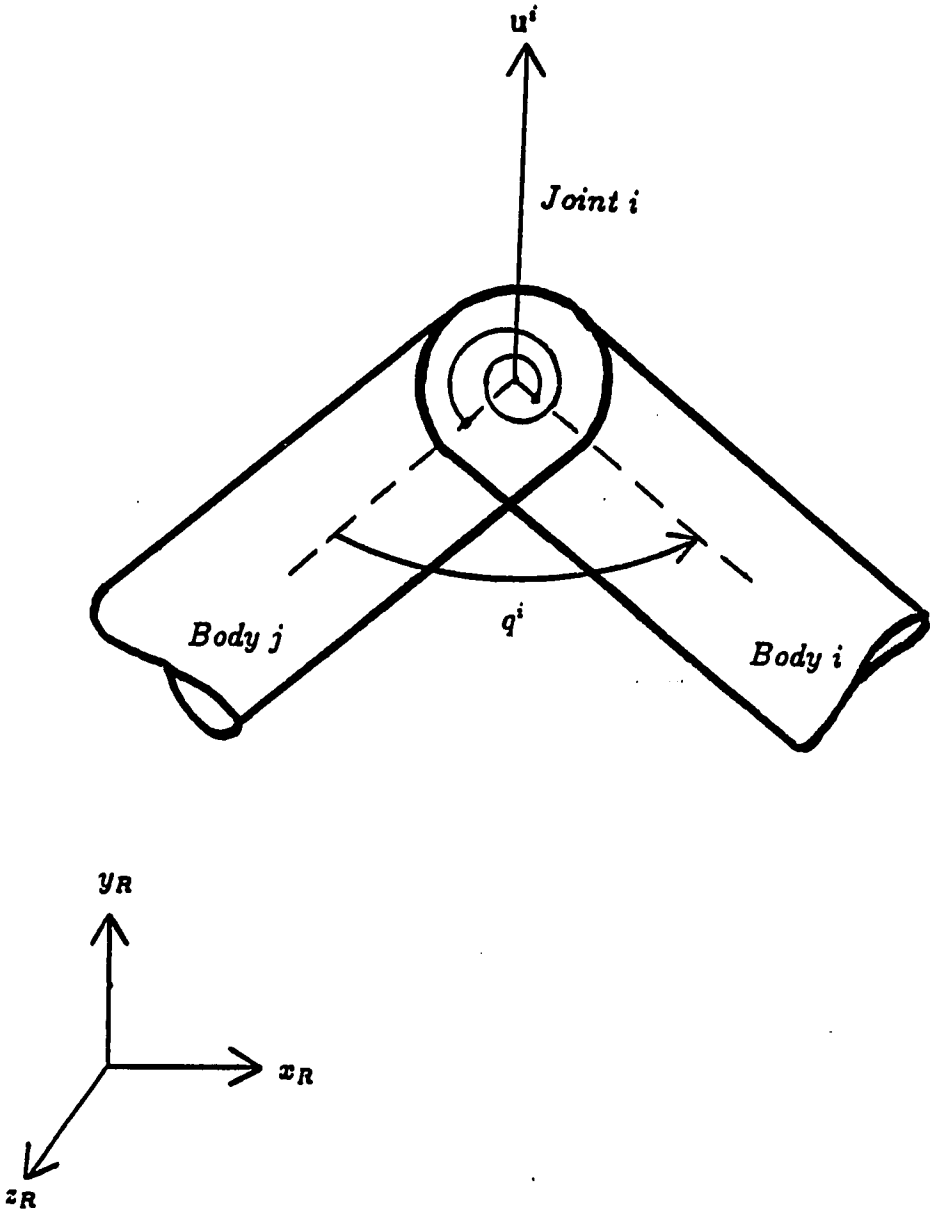


Figure 4.1: A rotational spring-damper-actuator

Figure 4.2 shows the translational spring-damper-actuator (TSDA) element. This force element acts between bodies i and j. The attachment points for the TSDA element is shown in Figure 4.2 at points p^i and p^j on bodies i and j, respectively. The force associated with this element can be represented as [6]

$$f_T^{ij} = k^{ij}(l^i - \beta^i) + c^{ij}\dot{l}^i + \tau^{ij} \tag{4.24}$$

where k^{ij}, c^{ij}, and τ^{ij} are the spring constant, damping coefficient, and linear actuator force, respectively. The terms l^i and $\dot{l}^i$ are the relative linear position and relative linear velocity associated with the translational movement and β^i is the initial undeformed length of the spring.

Equation 4.24 can be put into body i and j Cartesian force vectors as [6]

$$\mathbf{f}^i = -f_T^{ij} \begin{bmatrix} \mathbf{u}^i \\ \tilde{\mathbf{s}}^i \mathbf{u}^i \end{bmatrix}_{(6\times 1)} \tag{4.25}$$

$$\mathbf{f}^j = f_T^{ij} \begin{bmatrix} \mathbf{u}^i \\ \tilde{\mathbf{s}}^j \mathbf{A}^{ji}\, \mathbf{u}^i \end{bmatrix}_{(6\times 1)} \tag{4.26}$$

where $\mathbf{u}^i$ is the unit vector about which body i translates. The quantities $\mathbf{s}^i$ and $\mathbf{s}^j$ are local position vectors from the mass centers to p^i and p^j, respectively. The vectors $\mathbf{s}^i$ and $\mathbf{s}^j$ are shown graphically in Figure 4.2. The vectors $\tilde{\mathbf{s}}^i\mathbf{u}^i$ and $\tilde{\mathbf{s}}^j\mathbf{A}^{ji}\mathbf{u}^i$ account for the torque due to the TSDA force not passing through the mass centers of each body.

4.4 Equations of Motion via Symbolic Computing

As stated in Chapter 2, this research generates equations of motion for general, multibody systems using symbolic computing methods. Symbolic codes perform

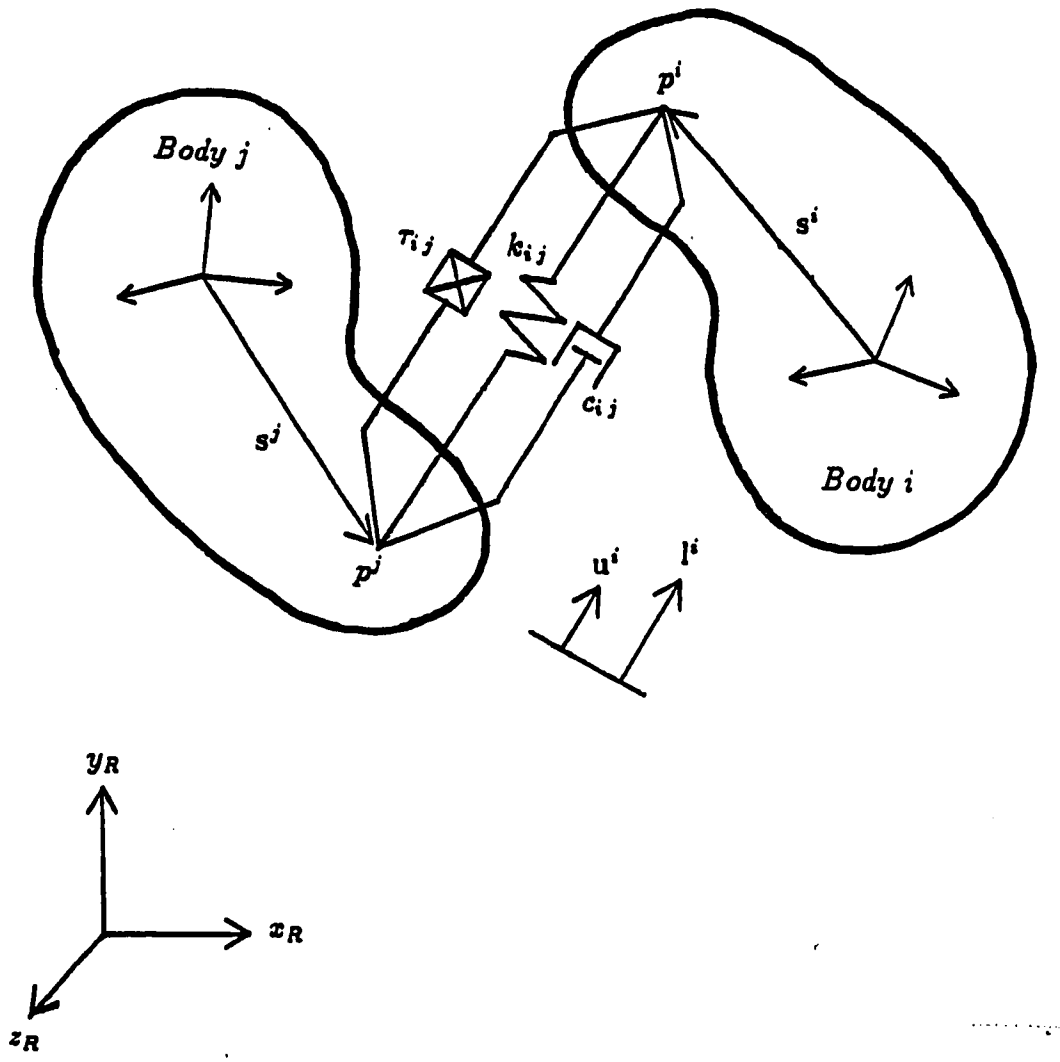


Figure 4.2: A translational spring-damper-actuator

mathematical operations on symbols rather than numbers. The results of these operations are explicit equations which are a function of the input symbols.

Symbolic computing of equations of motion offers many advantages over numerical or manual formulation. Algebra and differentiation errors typical of manual formulation are eliminated. In addition, symbolic equations are more efficient to integrate. This savings often offsets the time for symbolic equation generation after a small number of simulations. Linearization and sensitivity analysis are straightforward because the equations are known explicitly and can be directly differentiated.

Symbolic computing does, however, have some disadvantages. The main disadvantage is the size of systems that can be analyzed. Large systems, such as a 7 or 8 body mechanism, require a large amount of computer memory to carry out the symbolic computing. In this case, the size of the equations makes algebraic simplification of the resulting equations difficult. This is a major problem that is not addressed in the literature. For example, simplification of a three-dimensional, three link mechanism can take 3 times longer than forming the equations without any simplifications.

4.5 Algorithm for Formulation of Equations of Motion

The symbolic manipulation program MACSYMA [25] is used in this study to perform the symbolic manipulation necessary for generating equations of motion. The operations defined in MACSYMA require a standard input file typical for multibody simulation programs. Some of the standard input includes the system topology information, rotational and translational spring-damper-actuator information, relative joint definition information, and inertial properties. The required

input for this algorithm along with example input for different types of mechanisms is given in Appendix A. After this information has been input, the following steps are symbolically executed by MACSYMA.

1. From the system topology information, calculate the transformation matrices, $\mathbf{A}^{ij}$, and the unit vector about which the *ith* generalized coordinate rotates or translates, $\mathbf{u}^{i}$.

2. Calculate the $\mathbf{d}^{ij}$ vectors based on system topology and $\mathbf{A}^{ij}$ matrices as in equations 3.21 and 3.22.

3. Calculate the partial velocity matrix, $\mathbf{B}$, using $\mathbf{A}^{ij}$, $\mathbf{d}^{ij}$, and $\mathbf{u}^{i}$ terms.

4. Calculate $\dot{\mathbf{B}}$ from the $\mathbf{B}$ matrix.

5. Determine the Cartesian force vectors for each body such as gravity forces and spring-damper-actuator forces.

6. Form the symbolic equations of motion for an open-loop system using equation 4.14.

7. Transform equations of motion into FORTRAN output files.

The resulting equations are then output in FORTRAN subroutines for numerical integration. This algorithm is used to generate the equations of motion for several examples in Chapter 5.

A single pendulum will be used as an example to demonstrate the steps taken in the algorithm. Body 1 is the base body or ground. Body 2 is the pendulum which rotates about the global z axis and experiences gravity in the negative global y axis

direction. The center of gravity of the pendulum is located at a distance of dl22 from the pin joint. Moments of inertia are given with respect to the mass center of the pendulum and no products of inertia are defined. The symbolic algorithm for formulating equations of motion was executed and the MACSYMA results for each step are shown below.

1.

$$\mathbf{A}^{12} = \begin{bmatrix} \cos(q_2) & \sin(q_2) & 0 \\ -\sin(q_2) & \cos(q_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{u}^1 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

2.

$$\mathbf{d}^{11} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$$

$$\mathbf{d}^{21} = \begin{bmatrix} 0 & -dl22 & 0 \end{bmatrix}^T$$

$$\mathbf{d}^{22} = \begin{bmatrix} 0 & -dl22 & 0 \end{bmatrix}^T$$

3.

$$\mathbf{B} = \begin{bmatrix} dl22 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$$

4.

$$\dot{\mathbf{B}} = \begin{bmatrix} 0 & \dot{q}_2 dl22 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

5.

$$\mathbf{f} = \begin{bmatrix} -m_2 \sin(q_2) g & -m_2 \cos(q_2) g & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

6.

$$\ddot{q}_2(j2z + m_2 dl22^2) + m_2 \sin(q_2) dl22g = 0$$

7.
```
qtt(2)*(j2z + m2*dl22**2) + m2*sin(q(2))*dl22*g = 0.0
```

4.6 Numerical Efficiency using Symbolic Equations

In order to obtain increased efficiency when numerically integrating symbolic equations, it is necessary to take advantage of the recursive nature of equations resulting from multibody systems. This section will compare different approaches for using symbolic equations for numerical integration and compare the efficiency for each.

A pure symbolic approach will be defined when the complete symbolic equations are formed. Thus, there are no intermediate substitution of expressions which would simplify the final equations. As an example, assume that the generalized mass matrix, $\overline{\mathbf{M}} = \mathbf{B}^T\mathbf{M}\mathbf{B}$, is formed. The $\mathbf{B}$ and $\mathbf{M}$ matrices are symbolically calculated and then $\overline{\mathbf{M}}$ is symbolically formed. This results in a large number of redundant mathematical operations since components of $\mathbf{B}$ may appear more than once in $\overline{\mathbf{M}}$. The pure symbolic method does not take into account the recursive nature of multibody systems and is numerically inefficient for large mechanical systems.

A second method forms the symbolic matrices of the system such as $\mathbf{B}, \mathbf{M}, \dot{\mathbf{B}}$, etc., and then completes the rest of the formulation numerically. Therefore, at each

integration step, the numerical representations of the matrices are multiplied to form the equations of motion. This approach works well for larger systems since each component of **B** is formed only once during an integration step. The disadvantage of this approach is that it results in a large number of numerical zero multiplies due to the sparseness of the matrices. As an example, consider the calculation of $\overline{\mathbf{M}}$. The Cartesian mass matrix, **M**, is a positive definite, banded matrix, and the partial velocity matrix, **B**, contains only lower triangular components.

A third method forms the symbolic matrices of the system and numerically evaluates each matrix as in the previous method, but a second symbolic step is used to simplify expressions. Each nonzero entry in the matrices are given a symbolic name and are multiplied symbolically to remove the zero multiplies. Consider again the calculation of $\overline{\mathbf{M}}$. The **B** and **M** matrices are symbolically calculated. Then, temporary matrices $\hat{\mathbf{B}}$ and $\hat{\mathbf{M}}$ are set up which contain a symbolic constant for each of the nonzero components of **B** and **M**, respectively. Finally, $\overline{\mathbf{M}} = \hat{\mathbf{B}}^T\hat{\mathbf{M}}\hat{\mathbf{B}}$, is symbolically calculated before any integration is performed. This recursive method works well for large systems keeping equation size down by taking advantage of the recursive nature of these equations. Other recursive techniques are currently being investigated.

4.7 Substructuring of Mechanical System Models

Substructuring is another method to make use of recursive properties and to reduce the effort associated with the symbolic calculations. Substructuring can be used if the mechanical system has open chains wherein each chain can be analyzed separately and combined to form the system equations of motion. Substructuring

enables the symbolic calculations to be performed for one chain at a time, reducing the size of the equations that are generated and greatly reducing the simplification effort (simplification time increases by the square of the number of bodies in a system). In addition, if similar chains exist, the symbolic equations often need only be derived once and duplicated in the final system equations.

The path matrix is required in order to substructure a system. Partial velocity matrices and other system matrices are formed for each subsystem which are used to derive the subsystem equations of motion. The system path matrix is then used to reassemble these equations into the total system equations. An example is presented in Chapter 6 to demonstrate the details of this method.

5 EXAMPLES OF DYNAMIC SYSTEMS

This chapter will present several example problems using the the symbolic formulation of Kane's equations developed in Chapter 4. These examples demonstrate that this method results in efficient simulation as well as providing a symbolic set of dynamic equations for linearization.

Section 1 of this chapter presents the results of a three degree of freedom pendulum. Section 2 models the six degree of freedom Stanford robot arm. Finally, Section 3 presents results of an eight degree of freedom satellite system. Each section includes methods for validating the symbolic equations.

5.1 Three Degree of Freedom Pendulum

Figure 5.1 presents a three degree of freedom pendulum. Ground is the base body and all links are connected by a revolute joint. The three revolute joints contain rotational spring-damper-actuator (RSDA) elements. Each link rotates in a plane perpendicular to the rotation of the previous link. Appendix A contains the input data for the symbolic equation generator and parameters for the simulations to be presented.

Using the symbolic Kane's formulation, a set of second order, nonlinear, ordinary differential equations are generated. The same equations were generated

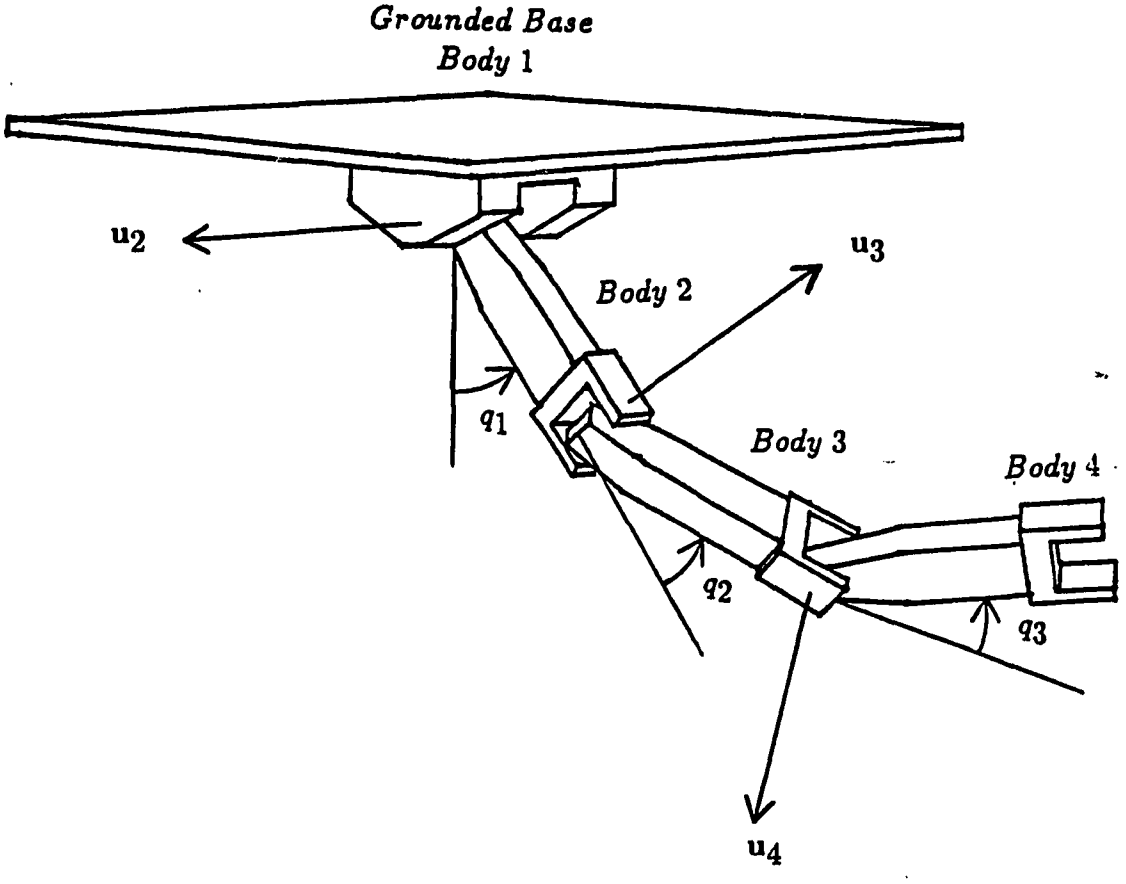


Figure 5.1: Three degree of freedom pendulum

manually and compared to verify the algorithm. Several days were necessary to form the equations manually while this symbolic algorithm generated the same results in less than 9 CPU minutes on a VAX 11/785.

Using the symbolic equations, a numerical integration was performed. The initial conditions of the relative joint coordinates are set at $q_i(0) = 30^\circ$, and $\dot{q}_i(0) = 0.0^\circ/\text{sec}$, where $i = 1, 2$, and 3. Figure 5.1 shows the initial configuration of the mechanism. The first simulation has no spring, damper, or actuator forces at the joints.

Figure 5.2 shows the three relative joint angles as a function of time. Figure 5.3 shows a plot of kinetic, potential and total energies during this integration. This plot provides a numerical verification of the dynamic system model since the total mechanical energy for a conservative system is constant. The total mechanical energy, as shown in Figure 5.3, deviates by less than the numerical integration error tolerance which indicates that both the underlying symbolic equations and the numerical integration results are correct.

This simulation was also duplicated using a numerical formulation based on Kane's equations [6]. The numerical formulation required 10.3 CPU seconds for 1 real time second of simulation. However, integration of the symbolic equations required 6.6 CPU seconds for 1 real time second using the same integration error tolerance. Also, pure symbolic equations were formed for this problem and thus, the least efficient method was used for numerical integration.

A second simulation was performed that included damping at the joints. Figure 5.4 shows the three relative coordinates which attain an equilibrium condition at 0.0°. Figure 5.5 presents the kinetic, potential, and total energy plots. Notice that

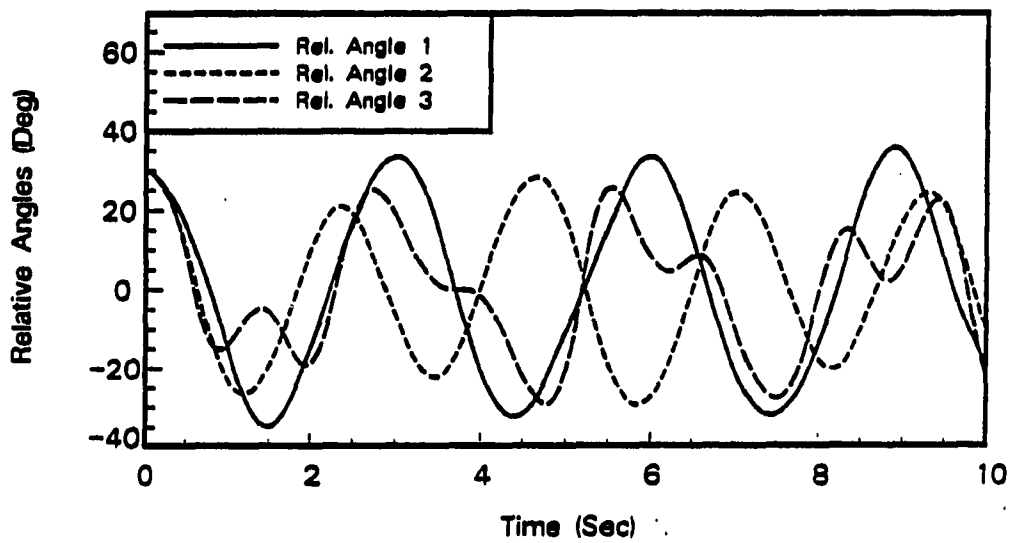


Figure 5.2: Rotational joint angles of three degree of freedom pendulum

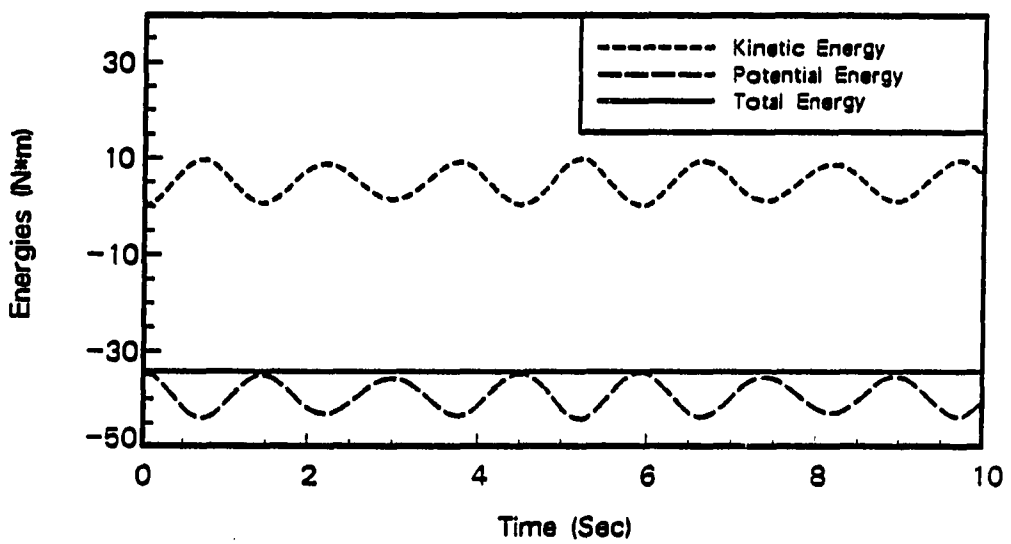


Figure 5.3: Energy components of three degree of freedom pendulum

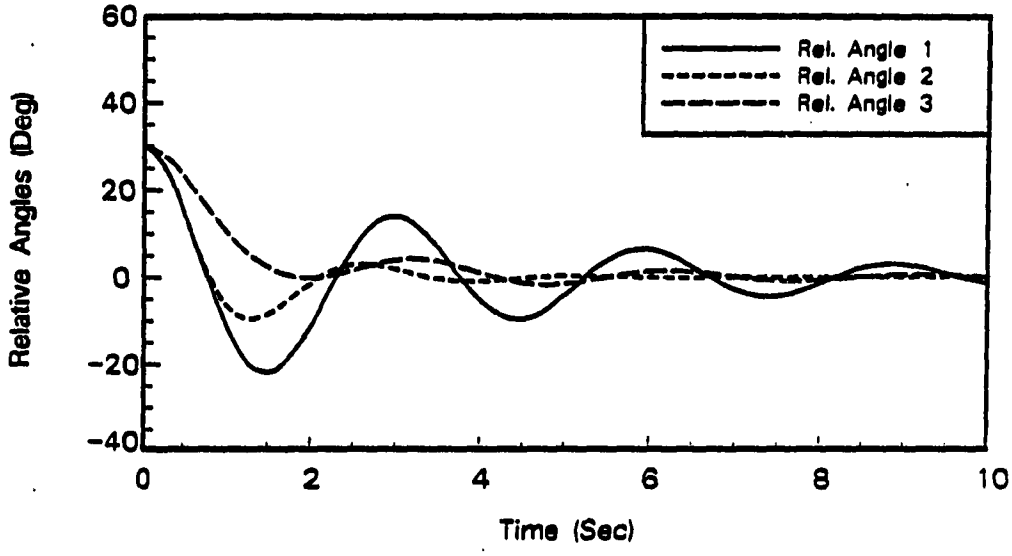


Figure 5.4: Rotational joint angles of damped three degree of freedom pendulum

the total energy no longer remains constant but decreases due to the dissipative effects of viscous damping.

Computer animation was also used to verify the simulations for a number of different initial conditions. Visualization of three dimensional motion is an effective tool for understanding the motion and detecting motion that does not appear correct. This verification method becomes important as multibody systems increase in complexity.

5.2. Six Degree of Freedom Stanford Arm

Figure 5.6 presents the six degree of freedom Stanford robot arm which is modelled with six bodies and six joints. Joints 1, 2, 4, 5, and 6 are revolute joints and joint 3 is a translational joint. Corresponding RSDA and TSDA actuator elements were included for each revolute and translational joint, respectively. Appendix A

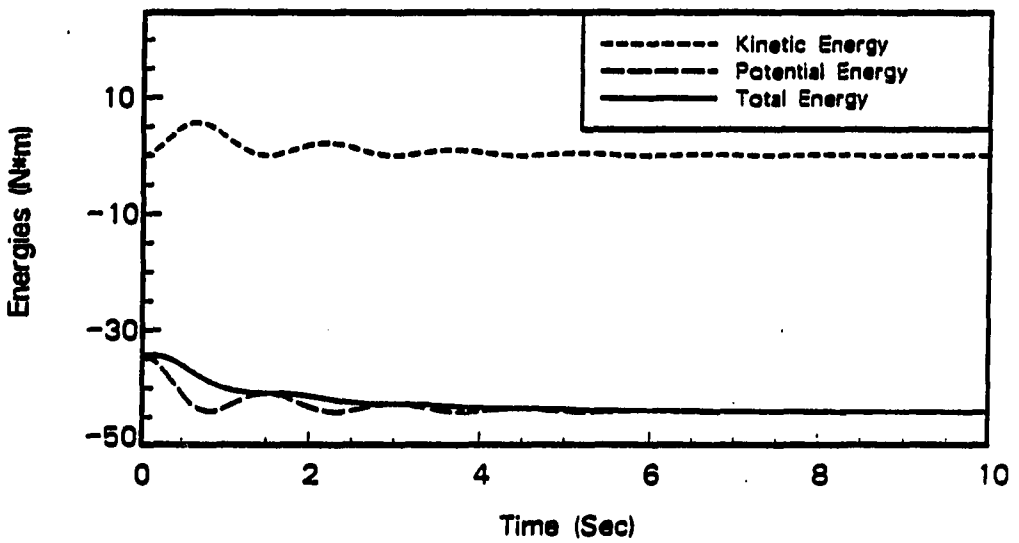


Figure 5.5: Energy components of damped three degree of freedom pendulum

contains the input data used to generate the symbolic equations. The robot arm link lengths and inertial properties were taken from Reference [26].

A simulation was performed with no actuator torques or damping and only gravity forces acting on each body. Figure 5.7 is a plot of kinetic, potential and total energy during this simulation. Again, the total energy varied by less than the integration error tolerance. The symbolic equations were formed using one level of intermediate expressions to increase integration efficiency and required 9.6 CPU seconds for 1 real time second of simulation. The numerical formulation method produced the same results in 11.4 CPU seconds for 1 real time second.

A second inverse dynamic simulation was performed for the Stanford Arm. This method solves for the corresponding actuator torques and forces given the position, velocity, and acceleration of each joint. Given the motion of the system,

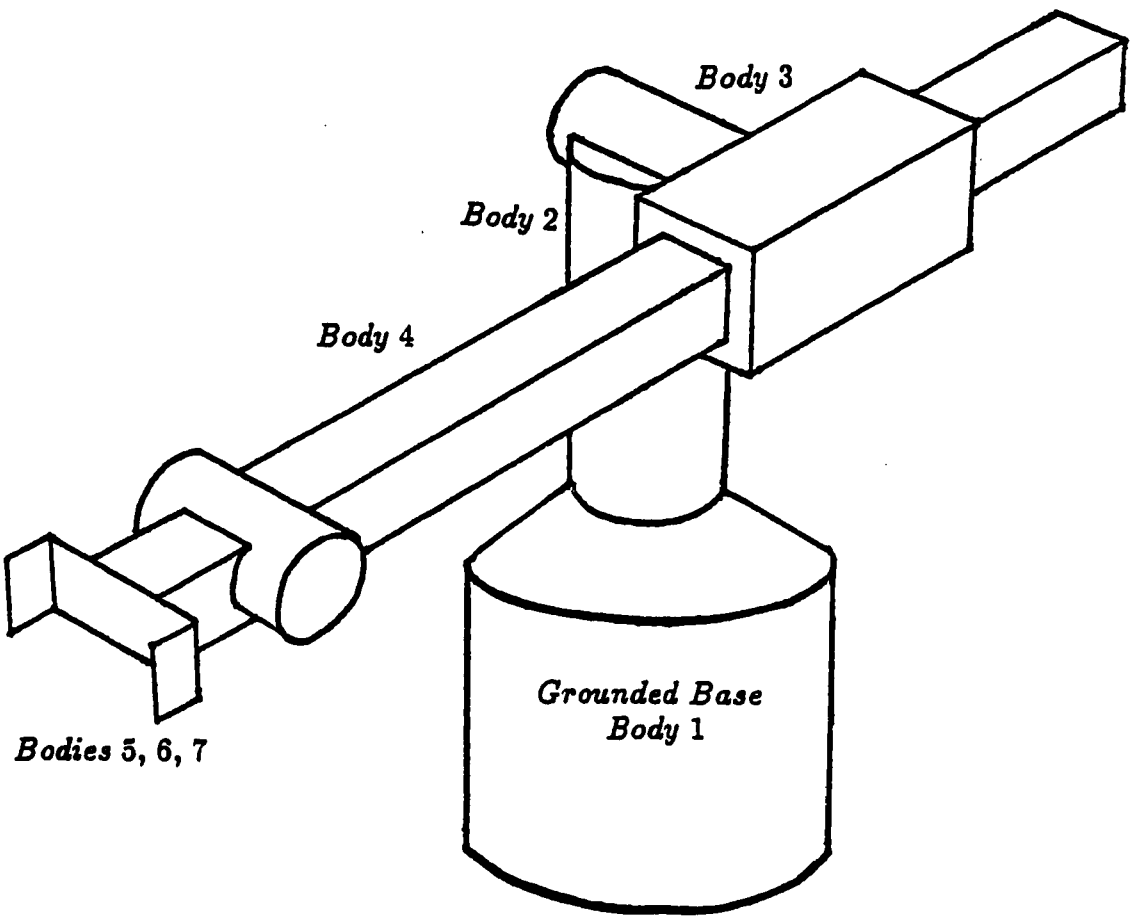


Figure 5.6: Six degree of freedom Stanford Arm

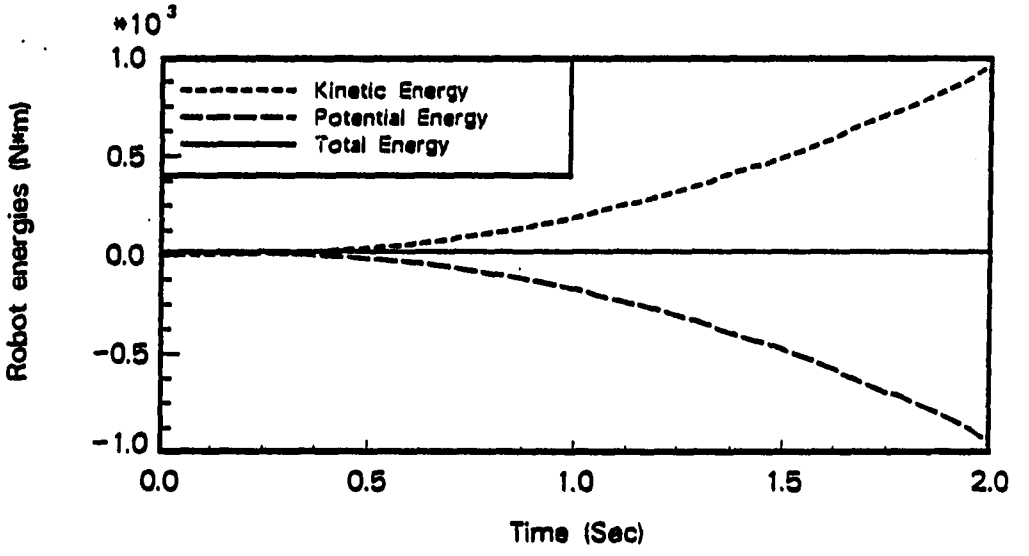


Figure 5.7: Energy components of Stanford Arm with no actuator elements

equation 4.14 is rearranged and solved for the actuator torques and forces

$$\tau = \mathbf{B}^T\mathbf{M}\mathbf{B}\ddot{\mathbf{q}} - \mathbf{B}^T(\hat{\mathbf{f}} - \mathbf{M}\dot{\mathbf{B}}\dot{\mathbf{q}} - \mathbf{h}) \tag{5.1}$$

Note that $\hat{\mathbf{f}}$ does not contain the actuator torques or forces.

The relative positions of the joints are specified as

$$q_i(t) = q_i(0) + \frac{q_i(T) - q_i(0)}{T}\{t - (T/2\pi)\sin(2\pi t/T)\} \qquad i = 1, \cdots, 6 \tag{5.2}$$

where $T = 10\ s$ is the total trajectory time. The initial conditions are defined as $q_1(0) = 0^\circ$, $q_2(0) = 90^\circ$, $q_3(0) = 0\ m$, $q_4(0) = 0^\circ$, $\cdots$, $q_6(0) = 0^\circ$, and the final conditions are defined as $q_i(T) = 60^\circ$ $(i = 1, 2, 4\ \cdots,\ 6)$ and $q_3(T) = 0.1m$.

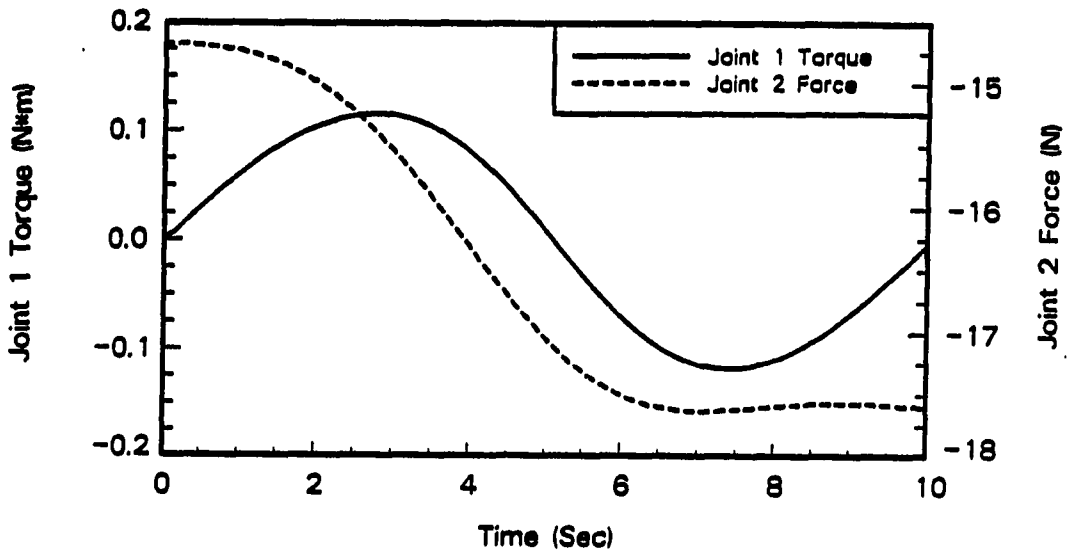


Figure 5.8: Inverse dynamic solution of joint torques 1 and 2 for Stanford Arm

The resulting torque/force histories of the inverse dynamic simulation are shown in Figures 5.8 through 5.10. As a check on this method, the actuator torques and forces are input to the forward dynamic model resulting in the joint positions given by equation 5.2. Figure 5.11 displays a series of frames from an animation of the responses.

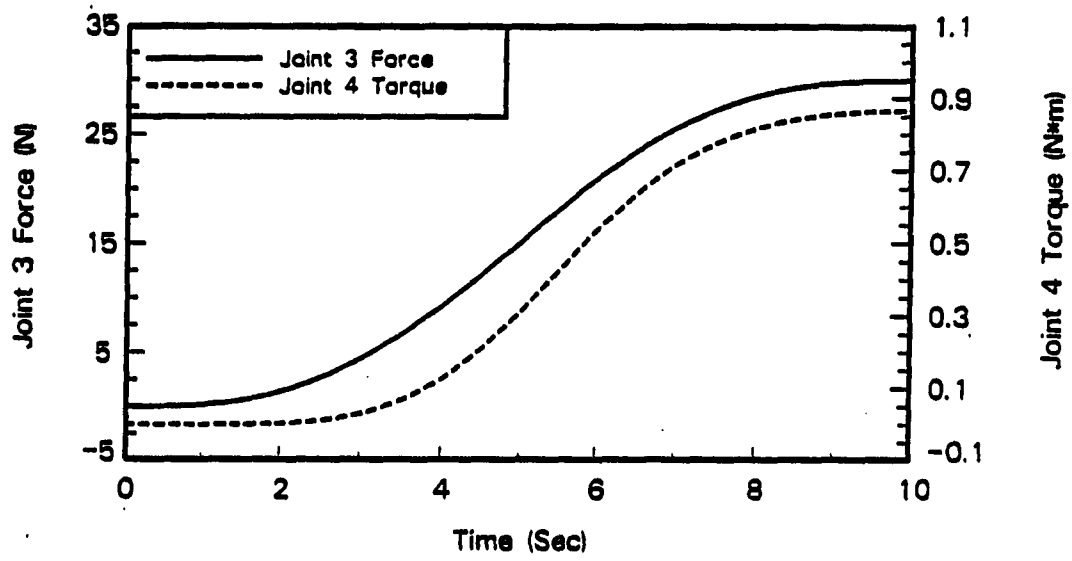


Figure 5.9: Inverse dynamic solution of joint force 3 and torque 4 for Stanford Arm

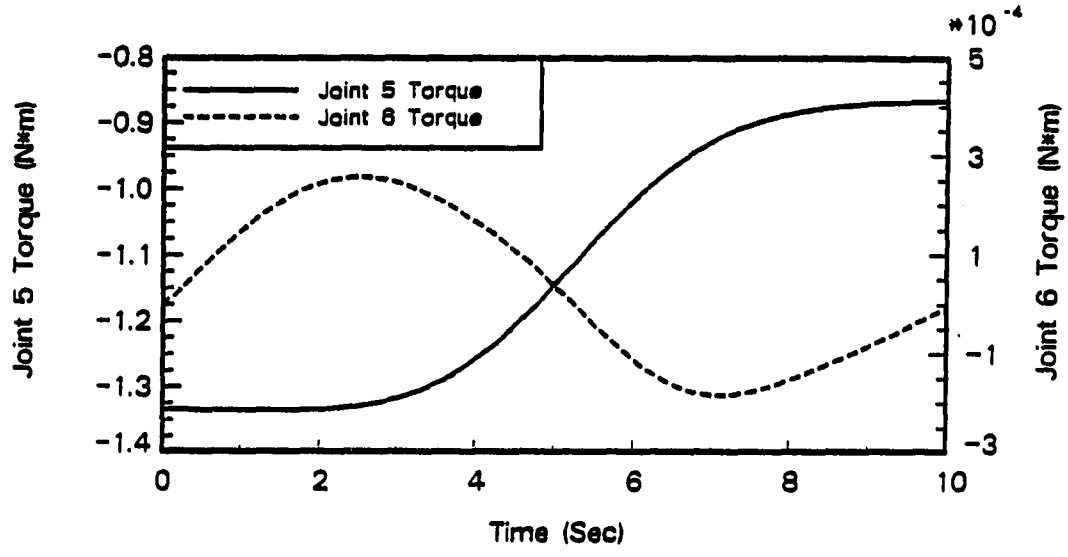
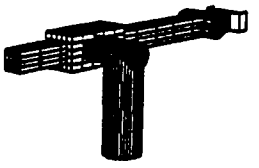
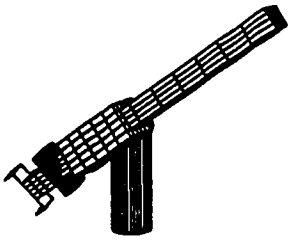


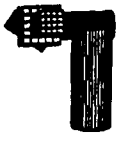
Figure 5.10: Inverse dynamic solution of joint torques 5 and 6 for Stanford Arm



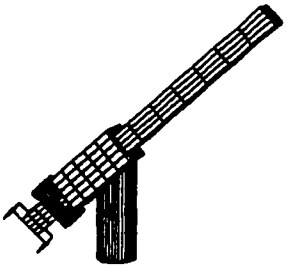
a.) t = 0.0



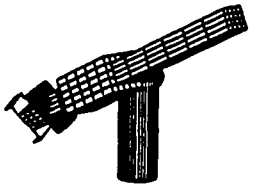
d.) t = 3.0



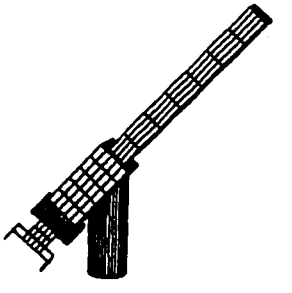
b.) t = 1.0



e.) t = 4.0



c.) t = 2.0



f.) t = 5.0

Figure 5.11: Animation of six degree of freedom robot

5.3 Eight Degree of Freedom Satellite System

Figure 5.12 presents an eight degree of freedom satellite. This system is composed of a floating base body and two identical links connected by revolute joints. The floating base body is modelled using seven generalized coordinates: three linear Cartesian coordinates and four Euler parameters. There are only six degrees of freedom associated with this body since the Euler parameters are not independent. Gravity is set to zero and two RSDA actuators are contained in the revolute joints.

Appendix A contains the input data for the symbolic generator and parameters for the simulations. As a means of verifying the model, the angular momentum was computed for the system. The angular momentum is given as

$$\mathbf{h}^{R\,O} = \sum_{i=1}^{nb} \mathbf{A}^{R\,i} \{ \mathbf{J}_i \omega^i + m_i \mathbf{p}^{O\,i*} \times \mathbf{v}^{O\,i*} \} \tag{5.3}$$

where $\mathbf{h}^{R\,O}$ is the angular momentum vector of point O, shown in Figure 5.12, with respect to reference frame R, $\mathbf{A}^{R\,i}$ is a 3×3 transformation matrix from body i to reference frame R, $\mathbf{p}^{O\,i*}$ is a Cartesian position vector defined in body i coordinates from point O to $i*$, the center of mass of body i, and $\mathbf{v}^{O\,i*}$ is the Cartesian velocity vector defined in body i coordinates from point O to $i*$. All the quantities contained in equation 5.3 can be extracted from the formulation of the symbolic equations of motion.

A simulation was performed to aid in verifying the formulation process. Initial relative conditions for bodies 2 and 3 were set at $-\pi/3$ and $+\pi/3$ radians with respect to the vertical y axis of body 1. Body 1 was given an initial angular velocity about the y axis as 2.0 rad/sec. Figure 5.13 shows the relative joint coordinates associated with the two revolute joints. This is intuitively pleasing since the relative

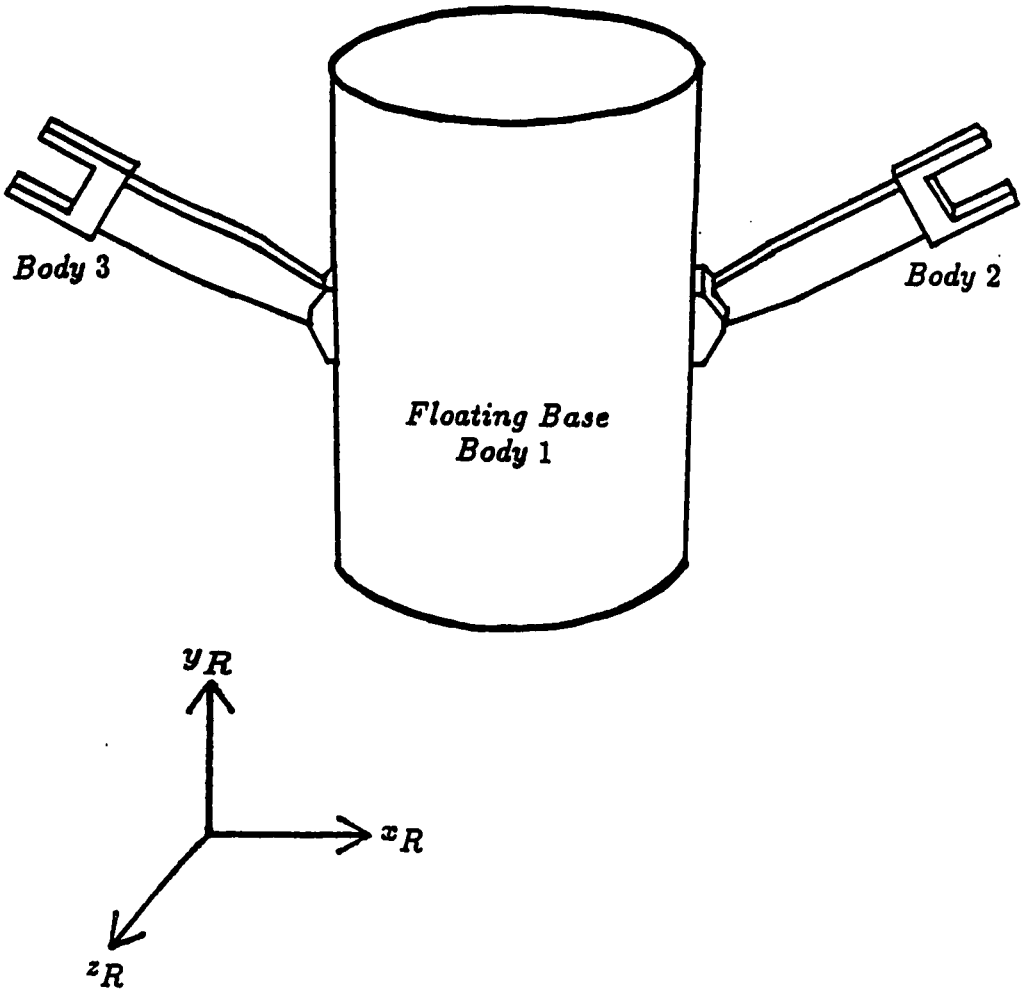


Figure 5.12: Eight degree of freedom satellite

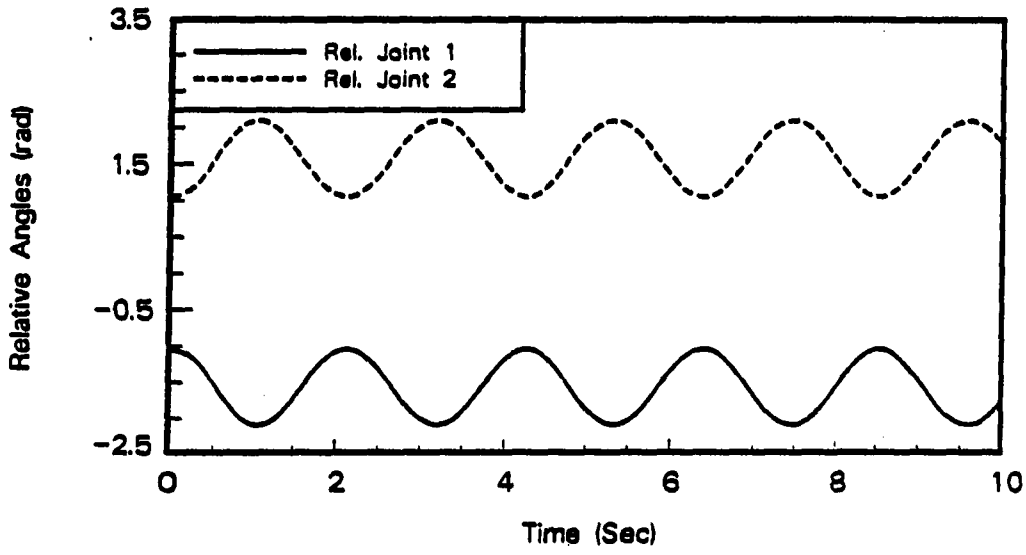


Figure 5.13: Satellite rotational joint angles 1 and 2

coordinates oscillate about $\pi/2$ radians due to the centrifugal force on each link. Figure 5.14 is a plot of the local angular velocities of body 1. Notice that the angular velocity about the y axis oscillates sinusoidally due to the relative motions of bodies 2 and 3. Figure 5.15 displays the simulation results of the global angular momentum components about the inertial reference frame. As expected, the angular momentum remains constant.

The same simulation conditions were used to compare the satellite system with the numerical formulation results. Both methods produced the same results, however, the numerical method required 13.4 CPU seconds while the symbolic method required 9.6 CPU seconds for 1 real time second of simulation.

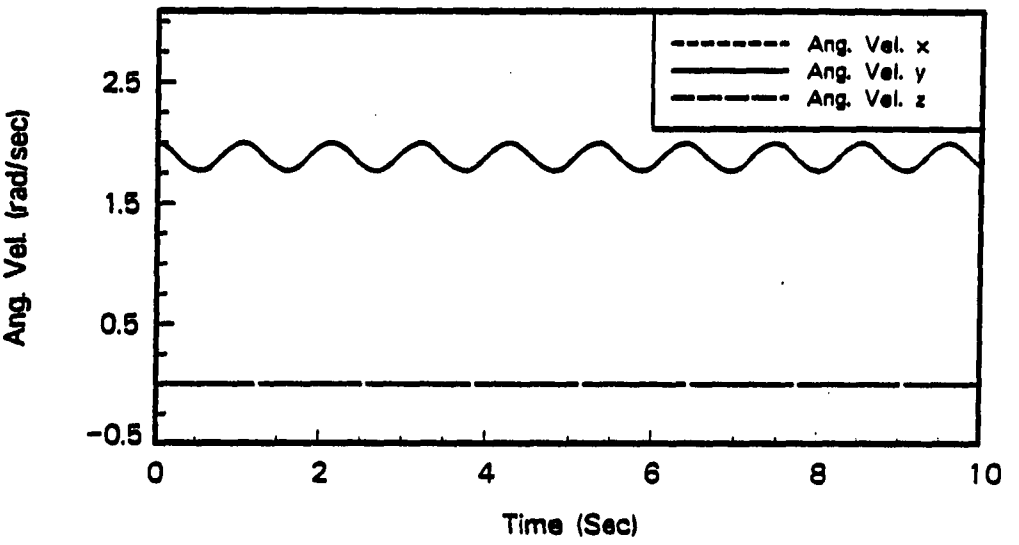


Figure 5.14: Local angular velocity components of satellite base body

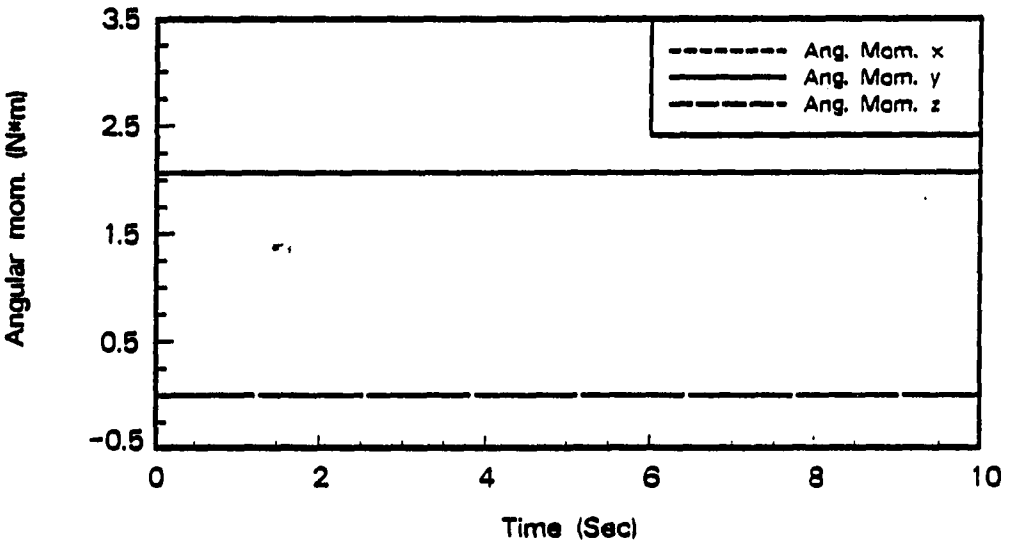


Figure 5.15: Angular momentum components of satellite

6 EXTENSIONS OF SYMBOLICS IN MECHANICAL SYSTEM ANALYSIS

A major benefit of symbolic over numerical formulation methods is that explicit, nonlinear equations are available. These equations can be used to form linearized equations about nominal operating points. The linear equations have many uses including control system analysis and sensitivity analysis.

Section 1 of this chapter presents the process for automatic linearization of nonlinear mechanical systems. An example problem is used to show the symbolic linearization procedure. Section 2 presents the eigenvalue solution of linearized constrained dynamic models. Two methods are used to produce the eigenvalue results of a spatial four bar mechanism. Section 3 uses the linearized equations for design sensitivity analysis. Finally, Section 4 presents a substructuring example.

6.1 Automatic Linearization of Nonlinear Mechanical Systems

A set of linearized equations can be formed by using a Taylor series approximation of the nonlinear equations about some equilibrium point. A symbolic formulation is ideal for obtaining explicit linearized equations because symbolic partial derivatives can be obtained by symbolically differentiating the nonlinear equations.

For unconstrained mechanical systems, the nonlinear equations can be generalized as [27]

$$\mathbf{M}(\mathbf{q},t)\,\ddot{\mathbf{q}} + \mathbf{h}(\dot{\mathbf{q}},\mathbf{q},\mathbf{f},t) = \mathbf{0} \tag{6.1}$$

where $\mathbf{M}$ is an $ndf \times ndf$ (number of degrees of freedom) generalized mass matrix, $\ddot{\mathbf{q}}$, $\dot{\mathbf{q}}$, and $\mathbf{q}$ are ndf relative coordinate vectors of acceleration, velocity, and position, respectively, $\mathbf{h}$ is the ndf vector of coriolis, centrifugal, and gravity terms, $\mathbf{f}$ is the ndf vector of conservative and nonconservative forces, and t accounts for the time-varying properties of the mechanical system. These equations can be combined to form a more general set of nonlinear equations as [27]

$$\mathbf{g}(\ddot{\mathbf{q}},\dot{\mathbf{q}},\mathbf{q},\mathbf{f},t) = \mathbf{0} \tag{6.2}$$

Equation 6.2 can be linearized about an operating point $\mathbf{q}^* = (\ddot{\mathbf{q}}_0, \dot{\mathbf{q}}_0, \mathbf{q}_0, \mathbf{f}_0, t)$ as

$$\delta\mathbf{g} = \frac{\partial \mathbf{g}}{\partial \ddot{\mathbf{q}}}\Big|_{\mathbf{q}^*}\delta\ddot{\mathbf{q}} + \frac{\partial \mathbf{g}}{\partial \dot{\mathbf{q}}}\Big|_{\mathbf{q}^*}\delta\dot{\mathbf{q}} + \frac{\partial \mathbf{g}}{\partial \mathbf{q}}\Big|_{\mathbf{q}^*}\delta\mathbf{q} + \frac{\partial \mathbf{g}}{\partial \mathbf{f}}\Big|_{\mathbf{q}^*}\delta\mathbf{f} = 0 \tag{6.3}$$

where $\delta\ddot{\mathbf{q}}$, $\delta\dot{\mathbf{q}}$, $\delta\mathbf{q}$, and $\delta\mathbf{f}$ are the variations about $\ddot{\mathbf{q}}_0$, $\dot{\mathbf{q}}_0$, $\mathbf{q}_0$, $\mathbf{f}_0$, and t, respectively. Assuming the mechanical system is time invariant and that equation 6.3 is homogeneous by choosing $\delta\mathbf{f} = 0$. Equation 6.3 may then be represented as

$$\frac{\partial \mathbf{g}}{\partial \ddot{\mathbf{q}}}\Big|_{\mathbf{q}^*}\ddot{\mathbf{q}}_l + \frac{\partial \mathbf{g}}{\partial \dot{\mathbf{q}}}\Big|_{\mathbf{q}^*}\dot{\mathbf{q}}_l + \frac{\partial \mathbf{g}}{\partial \mathbf{q}}\Big|_{\mathbf{q}^*}\mathbf{q}_l = \mathbf{0} \tag{6.4}$$

where $\delta\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_l$, etc., or in the familiar form as

$$\mathbf{M}_l\,\ddot{\mathbf{q}}_l + \mathbf{C}_l\,\dot{\mathbf{q}}_l + \mathbf{K}_l\,\mathbf{q}_l = \mathbf{0} \tag{6.5}$$

where $\mathbf{M}_l$, $\mathbf{C}_l$, and $\mathbf{K}_l$ are the linearized mass, damping, and stiffness matrices, and $\ddot{\mathbf{q}}_l$, $\dot{\mathbf{q}}_l$, and $\mathbf{q}_l$ are the local relative coordinate vectors of acceleration, velocity, and position.

As an example, the nonlinear equations of the three degree of freedom pendulum discussed in Chapter 5 were linearized about equilibrium points $\ddot{q}_0 = \dot{q}_0 = q_0 = 0$, i.e., the minimum potential energy state. Appendix A contains the system parameters used to derive the nonlinear equations of motion. Appendix B presents the symbolic equations of motion for both the nonlinear and linear models. The symbolic $\mathbf{M}_l$, $\mathbf{C}_l$, and $\mathbf{K}_l$ matrices are now available for eigenvalue and sensitivity analysis for any operating point.

6.2 Eigenvalue Solution of Linearized Constrained Dynamic Models

Symbolic computing is useful in eigenvalue analysis since linear matrices are derived symbolically in terms of any general equilibrium points. The resulting eigenvalue solutions from local linearizations provide more general information than numerical integration [27]. Eigenvalue solution of unconstrained systems such as the triple pendulum are straightforward. Constrained closed-loop problems are more challenging. This section presents two eigenvalue solution methods for linearized constrained dynamic models. The QZ method of Moler and Stewart [28] solves a large generalized eigenvalue problem using the set of generalized coordinates and Lagrange multipliers. The second method, developed in this thesis, uses QR decomposition to reduce the generalized eigenvalue problem to an independent set of coordinates defined on the tangent surface to the constraint Jacobian. A closed-chain spatial four bar mechanism will be used to demonstrate these two methods.

For this study, the damping matrix is zero. If the damping matrix was included in the model, the equations would have to be arranged in first order state vector form. The second order form, constrained mass and stiffness matrices of dimension

$(n + m_d) \times (n + m_d)$ are defined as

$$\begin{bmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \ddot{\mathbf{q}} \\ \ddot{\boldsymbol{\lambda}} \end{Bmatrix} + \begin{bmatrix} \mathbf{K} & \boldsymbol{\Phi}_{\mathbf{q}}^T \\ \boldsymbol{\Phi}_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{q} \\ \boldsymbol{\lambda} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \tag{6.6}$$

where n is the number of generalized coordinates and m_d is the number of constraints or dependent coordinates. The $n \times n$ open-loop mass and stiffness matrices are represented as $\mathbf{M}$ and $\mathbf{K}$ and $\boldsymbol{\Phi}_{\mathbf{q}}$ is the $m_d \times n$ constraint Jacobian matrix. Finally, $\mathbf{q}$ is the n vector of generalized coordinates and $\boldsymbol{\lambda}$ is the m_d vector of Lagrange multipliers.

Equation 6.6 may be used to solve for the corresponding eigenvalues. Physically, the constraint equations have infinite stiffness [27] which lead to eigenvalues of infinite magnitude. The QZ method of Moler and Stewart [28] is able to solve this ill-posed problem. The QZ eigenvalue problem is an order $(n + m_d)^3$ calculation. An alternative method developed in this thesis uses the QR decomposition of the constraint Jacobian matrix to identify the null space of the constraint Jacobian. This eigenvalue problem is presented below and is of order $n_i^3 + n^2$ calculation where n_i is the number of independent coordinates or degrees of freedom in the constrained dynamic model.

If the constraint Jacobian matrix $\boldsymbol{\Phi}_{\mathbf{q}}$ has full row rank, then there is an orthogonal matrix $\mathbf{Q}(n \times n)$ and an upper triangular matrix $\mathbf{R}(n \times m_d)$ such that [6]

$$\boldsymbol{\Phi}_{\mathbf{q}}^T = \mathbf{Q}\,\mathbf{R} \tag{6.7}$$

This matrix transformation is known as QR decomposition [29]. The matrix $\mathbf{R}$ is

defined as

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix} \tag{6.8}$$

where $\mathbf{R}_1$ is an $m_d \times m_d$ upper triangular matrix. The matrix $\mathbf{Q}$ is partitioned as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1^{(n \times m_d)} & \mathbf{Q}_2^{(n \times n_i)} \end{bmatrix} \tag{6.9}$$

where $\mathbf{Q}_1$ forms an orthonormal basis normal to the constraint surface, i.e., the row space of $\Phi_{\mathbf{q}}^T$ and $\mathbf{Q}_2$ forms an orthonormal basis of the constraint tangent surface or the null space of $\Phi_{\mathbf{q}}^T$. Equations 6.8 and 6.9 can be substituted into 6.7 resulting in

$$\Phi_{\mathbf{q}}^T = \mathbf{Q}_1 \mathbf{R}_1 \tag{6.10}$$

Premultiplying 6.10 by $\mathbf{Q}_2^T$ gives

$$\mathbf{Q}_2^T \Phi_{\mathbf{q}}^T = \mathbf{Q}_2^T \mathbf{Q}_1 \mathbf{R}_1 = \mathbf{0} \tag{6.11}$$

Thus, we can premultiply the first row of equation 6.6 by $\mathbf{Q}_2^T$ eliminating the Lagrange multiplier ($\boldsymbol{\lambda}$)

$$\mathbf{Q}_2^T \mathbf{M} \ddot{\mathbf{q}} + \mathbf{Q}_2^T \mathbf{K} \mathbf{q} = \mathbf{0} \tag{6.12}$$

Also, the linearized coordinate basis vector, $\mathbf{q}$, can be partitioned as

$$\mathbf{q} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{Bmatrix} \mathbf{d} \\ \mathbf{z} \end{Bmatrix} \tag{6.13}$$

where $\mathbf{d}$ is an m_d vector of dependent coordinates and $\mathbf{z}$ is an n_i vector of independent coordinates. If the new generalized coordinate basis vector is chosen to lie on the tangent to the constraint surface, then $\mathbf{d} = 0$ and

$$\mathbf{q} = \mathbf{Q}_2 \mathbf{z} \tag{6.14}$$

Substituting 6.14 into 6.12 results in a condensed differential equation solution defined as

$$\mathbf{Q}_2^T \mathbf{M} \mathbf{Q}_2 \ddot{\mathbf{z}} + \mathbf{Q}_2^T \mathbf{K} \mathbf{Q}_2 \mathbf{z} = \mathbf{0} \tag{6.15}$$

which represents the equations of motion for a linearized constrained dynamic system. The classic eigenvalue solution leads to the generalized eigenvalue problem for a linearized, constrained dynamic model

$$\mathbf{Q}_2^T \mathbf{K} \mathbf{Q}_2 \mathbf{x} = \sigma^2 \mathbf{Q}_2^T \mathbf{M} \mathbf{Q}_2 \mathbf{x} \tag{6.16}$$

where $\mathbf{x}$ is a n_i constant vector and σ is a constant scalar. The eigenvector solution, however, is defined in terms of the $\mathbf{z}$ basis vectors. In order to obtain the eigenvectors in terms of the original $\mathbf{q}$ basis vectors, the following matrix transformation must be performed;

$$\mathbf{U} = \mathbf{Q}_2 \mathbf{V} \tag{6.17}$$

where $\mathbf{V}$ is the $n_i \times n_i$ eigenvector matrix defined in the $\mathbf{z}$ basis and $\mathbf{U}$ is the $n \times n_i$ eigenvector matrix defined in the $\mathbf{q}$ basis.

As an example, consider the closed-loop, spatial four bar mechanism presented in Figure 6.1. The four bar mechanism, shown in an equilibrium configuration, has two revolute joints attached to the ground body, a spherical joint and a universal joint. An open-loop dynamic system with holonomic constraints is modelled by cutting the spherical joint. This results in 4 generalized coordinates with 3 constraints ($n = 4$ and $m_d = 3$) and 1 degree of freedom ($n_i = 1$).

The symbolic data input to generate the symbolic equations along with the system parameters is presented in Appendix C. Both the QZ method and the condensed

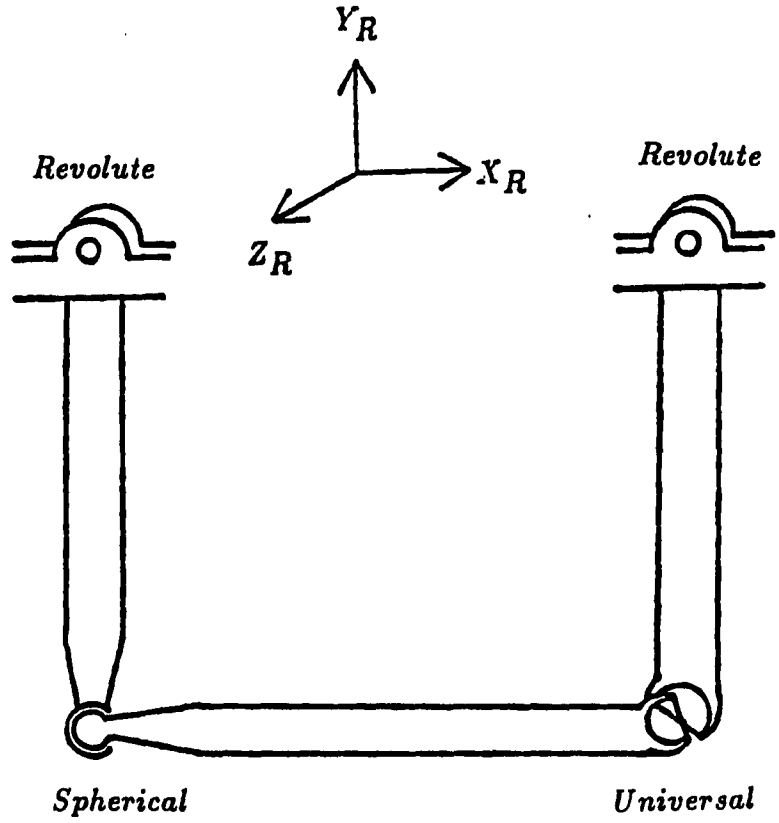


Figure 6.1: Closed-loop, spatial four bar mechanism

QR decomposition method resulted in the same eigenvalue of 12.887. However, the QZ method solved for the eigenvalues using equation 6.6 of matrix dimension 7×7, an order n^3 calculation, and resulted in 6 eigenvalues with infinite magnitude. The condensed QR decomposition method used equation 6.16 of dimension 1×1, a scalar. This eigenvalue solution is an order n_i^3 calculation, the QR decomposition is an order n^2 calculation [30], and the matrix multiplies that results in equation 6.16 is an order n^2 calculation. Thus, a more efficient solution is obtained using the QR decomposition. This savings increases as the size of the systems increases.

6.3 Design Sensitivity Analysis

Many researchers have used optimization in the time domain for systems with a repeatable motion such as a slider crank. Time domain analysis is not as useful for systems with varying equilibrium conditions such as a robot. In these cases, optimizing the eigenvalues of the system about some operating point can be very useful. This analysis only holds for the region in which the linearization is valid. Thus, many linearizations are often considered in order to understand the total operating region. This section will look at a scheme to compute eigenvalue derivatives with respect to the design variables.

Assume that the mechanical system is linear with no damping. The rate of change of the *ith* eigenvalue with respect to design variables is defined by [?]

$$\frac{\partial \lambda_i}{\partial a_j} = \frac{\mathbf{u}_i^T \left(\frac{\partial \mathbf{K}}{\partial a_j} - \frac{\partial \mathbf{M}}{\partial a_j}\lambda_i\right) \mathbf{u}_i}{\mathbf{u}_i^T \mathbf{M}\, \mathbf{u}_i} \tag{6.18}$$

where λ_i is the *ith* eigenvalue, a_j is the *jth* design variable of interest, $\mathbf{u}_i$ is the *ith* eigenvector, and $\mathbf{M}$ and $\mathbf{K}$ are the linearized mass and stiffness matrices. The partial

derivative defined in equation 6.18 indicates the sensitivity of the *ith* eigenvalue to the *jth* design variable. These sensitivities give the engineer information regarding possible modifications in the design variables in order to achieve a desired set of eigenvalues.

Symbolic formulation is ideal for computing design sensitivities. In equation 6.18, two partial derivatives of the mass and stiffness matrices are needed with respect to the design variables. These are easily formed by symbolically differentiating the **M** and **K** matrices. A numerical method would have to calculate a finite difference approximation of these matrices since **M** and **K** are not known explicitly. This can often lead to numerical errors. Also, for larger systems, finite difference approximations lead to increased eigenvalue solutions which are of order n^3 calculations compared to the order n^2 calculations for equation 6.18. Although this is not relevant for the following example, it leads to significant savings for large problems or for a large number of design variables.

The triple pendulum will be used as an example of design sensitivity analysis. It is desired to change the three eigenvalues of the system by changing the three pendulum lengths, i.e., the design parameters are the pendulum lengths. A 3×3 matrix of sensitivities is computed using equation 6.18. Given the three desired eigenvalues, the Newton Raphson method can be used to solve for the required changes in pendulum lengths.

The eigenvalue results for this example are presented in Tables 6.1 and 6.2. The system parameters used to obtain this result are given in Appendix A. Table 6.1 lists the first iteration of the Newton Raphson method using design sensitivity analysis. The second and third columns display the three desired eigenvalues and

Table 6.1: First iteration to desired eigenvalues using design sensitivity analysis

Eigenvalues	Desired Eigenvalue	Initial Eigenvalue	Eigenvalue from one sensitivity estimate
λ_1	20.0	46.0	28.4
λ_2	40.0	67.7	41.7
λ_3	130.0	198.6	156.7

three initial eigenvalues, respectively. The third column displays the results of the first iteration using the sensitivity information. Table 6.2 lists the second iteration to the desired eigenvalues. Notice that the eigenvalues have converged rapidly to the desired solution with only two iterations.

6.4 Substructuring Example

Substructuring was introduced in Chapter 4 as a means of separating a multi-chain mechanism into smaller components for equation formulation. Substructuring can be used to reduce memory requirements and increase the efficiency of the symbolic formulation. This section presents an example of substructuring to demonstrate the methodology.

As an example of a substructured system, consider the eight degree of freedom satellite system presented in Chapter 5. The two chain mechanism shown in Figure

Table 6.2: Second iteration to desired eigenvalues using design sensitivity analysis

Eigenvalues	Desired Eigenvalue	Initial Eigenvalue	Eigenvalue from one sensitivity estimate
λ_1	20.0	28.4	21.5
λ_2	40.0	41.7	40.0
λ_3	130.0	156.7	132.9

5.12 is redrawn in Figure 6.2 as two subsystems. Each arm can be modelled independently with the same floating base body. This is evident by the system path matrix

$$\pi = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{6.19}$$

Note that no coupling exists between bodies 2 and 3, i.e., $\pi_{23} = 0$. The partial velocity matrix for subsystem 1 is

$$\mathbf{B}_1 = \left[\begin{array}{cc|c} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \hline \mathbf{A}^{21} & -\tilde{\mathbf{d}}^{21}\mathbf{A}^{21} & \tilde{\mathbf{u}}^2\mathbf{d}^{22} \\ \mathbf{0} & \mathbf{A}^{21} & \mathbf{u}^2 \end{array}\right]^{(12\times 7)} \tag{6.20}$$

where $\mathbf{B}_1$ has been partitioned into a 2×2 matrix of submatrices $\mathbf{B}_{1_{ij}}$. The partial

velocity matrix for subsystem 2 is

$$\mathbf{B}_2 = \left[\begin{array}{ccc} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \hline \mathbf{A}^{31} & -\tilde{\mathbf{d}}^{31}\mathbf{A}^{31} & \tilde{\mathbf{u}}^3\mathbf{d}^{33} \\ \mathbf{0} & \mathbf{A}^{31} & \mathbf{u}^3 \end{array}\right]^{(12\times 7)} \tag{6.21}$$

The system partial velocity matrix is an 18×8 matrix and is constructed using $\mathbf{B}_1$, $\mathbf{B}_2$, and the system path matrix, π. Referring to the path matrix in equation 6.19, the elements π_{21} and π_{22} correspond to the elements of subsystem 1 partial velocity matrix $\mathbf{B}_{1_{21}}$, $\mathbf{B}_{1_{22}}$, respectively. Likewise, the elements π_{31} and π_{33} correspond to the elements of subsystem 2 partial velocity matrix $\mathbf{B}_{2_{21}}$, $\mathbf{B}_{2_{22}}$, respectively. Since both subsystems share the same floating base body, the element π_{11} can be formed by either $\mathbf{B}_{1_{11}}$ or $\mathbf{B}_{2_{11}}$. Finally, the system partial velocity matrix is given as

$$\mathbf{B} = \left[\begin{array}{cccc} \mathbf{I}_3 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{A}^{21} & -\tilde{\mathbf{d}}^{21}\mathbf{A}^{21} & \tilde{\mathbf{u}}^2\mathbf{d}^{22} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^{21} & \mathbf{u}^2 & \mathbf{0} \\ \hline \mathbf{A}^{31} & -\tilde{\mathbf{d}}^{31}\mathbf{A}^{31} & \mathbf{0} & \tilde{\mathbf{u}}^3\mathbf{d}^{33} \\ \mathbf{0} & \mathbf{A}^{31} & \mathbf{0} & \mathbf{u}^3 \end{array}\right]^{(18\times 8)} \tag{6.22}$$

The other matrices and vectors such as $\dot{\mathbf{B}}, \mathbf{M}, \mathbf{f}$, and $\mathbf{h}$ are formed in the same manner. The resulting equations of motion are identical to those obtained in Chapter 5.

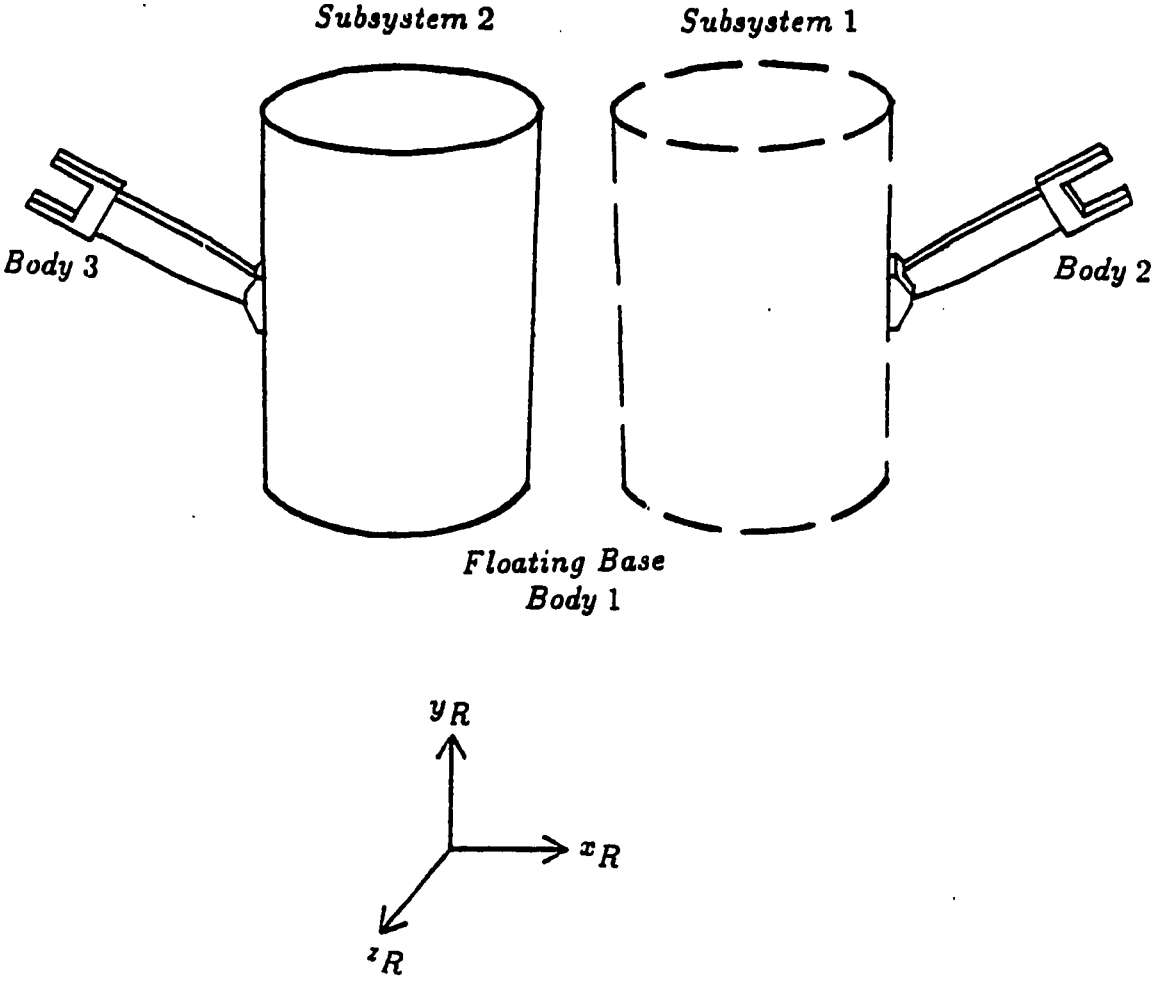


Figure 6.2: Substructuring of eight d.o.f. satellite system

7 CONCLUSIONS

This thesis presented a symbolic formulation for the equations of motion of multibody dynamic systems. The topology of the mechanical system was used to define an efficient representation of the interconnection of rigid bodies. A modified form of Kane's dynamical equations was used to derive the equations of motion. The partial velocity matrix transformed the equations in terms of Cartesian coordinates to equations in terms of an independent set of relative coordinates. The Cartesian coordinates allowed for easy system definition while the transformation to relative coordinates resulted in a minimal, efficient set of differential equations.

Several approaches were investigated for integrating the final symbolic equations. A large savings in computational efficiency was gained by taking advantage of the recursive nature of multibody dynamic systems. Examples in this thesis demonstrate the efficiency of integration of symbolically generated equations versus the numerical method.

Symbolically generated equations were extended in the areas of automatic linearization, eigenvalue solution of constrained mechanical systems, and design sensitivity analysis. Symbolic linearized equations allowed for linearization about any equilibrium point. Examples were presented in each area which demonstrated the utility of symbolic equations.

8 BIBLIOGRAPHY

[1] Chase, M. A. and Smith, D. C. "DAMN- A Digital Computer Program for the Dynamic Analysis of Generalized Mechanical Systems." SAE Transactions, paper 710244 (1971).

[2] Orlandea, N., Chase, M. A., and Calahn, D. A. "A Sparsity-Oriented Approach to Dynamic Analysis and Design of Mechanical Systems, Part I and II." Trans. ASME J. Engr. Indust. 99 (1977): 773-784.

[3] Sheth, P. N. and Uicker, J. J. "IMP (Integrated Mechanisms Program) A Computer-Aided Design System for Mechanisms and Linkages." Trans. ASME J. Engr. Indust. 94 (1972): 454-464.

[4] Wehage, R. A. and Haug, E. J. "Generalized Coordinate Partitioning of Dimension Reduction in Analysis of Constrained Dynamic Systems." Trans. ASME J. Mech. Design 104 (1982): 247-255.

[5] Nikravesh, P. E. and Chung, I. S. "Application of Euler Parameters to the Dynamic Reduction in Analyis of Constrained Dynamic Systems." Trans. ASME J. Mech. Design 104 (1982): 785-791.

[6] Kim, S. and Vanderploeg, M. "A General and Efficient Method for Dynamic Analysis of Mechanical Systems Using Velocity Transformations." J. of Mechanisms, Transmissions, and Automation in Design 93 (1985): 2-7.

[7] Genesereth, M. R. "A Brief Overview of MACSYMA." Mathlab Group, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1983.

[8] Dubowsky, S. and Grant, J. L. "Application of Symbolic Manipulation to Time Domain Analysis of Nonlinear Dynamic Systems." J. Dynamic Systems, Measurement, and Control, 97 (1975): 60-68.

[9] Hussain, M. A. and Noble, B. "Application of Symbolic Computation to the Analysis of Mechanical Systems, Including Robot Arms."

Computer Aided Analysis and Optimization of Mechanical System Dynamics. NATO ASI Series, Vol. F9. Berlin: Springer-Verlag, 1984.

[10] Kortum, W. and Schiehlen, W. "General Purpose Vehicle System Dynamics Software Based on Multibody Formalisms." Vehicle System Dynamics 14 (1985): 229-263.

[11] Schiehlen, W. O. "Computer Generation of Equations of Motion." Computer Aided Analysis and Optimization of Mechanical System Dynamics. NATO ASI Series, Vol. F9. Berlin: Springer-Verlag, 1984.

[12] Kreuzer, E. J. and Schiehlen, W. O. "Generation of Symbolic Equations of Motion for Complex Spacecraft." AAS/AIAA Astrodynamics Specialist Conference, Lake Placid, NY. Paper No. 83-302, 1983.

[13] Wittenburg, J. and Wolz, U. "MESA VERDE: A Symbolic Program for Nonlinear Articulated-Rigid-Body Dynamics." ASME Paper No. 85-DET-151, 1985.

[14] Levinson, D. A. "Equations of Motion for Multiple, Rigid Body Systems via Symbolic Manipulation." J. Spacecraft Dynamics and Control 14, No. 8 (1977): 479-487.

[15] Rosenthal, D. and Sherman, M. "Symbolic Multibody Equations via Kane's Method." AAS/AIAA Astrodynamics Specialist Conference, Lake Placid, NY. Paper No. 83-303, 1983.

[16] Leu, M. C. and Hemati, N. "Automated Symbolic Derivation of Dynamic Equations of Motion for Robotic Manipulators." J. Dynamic Systems, Measurement, and Control 108 (1986): 172-179.

[17] Faessler, H. "Computer-Assisted Generation of Dynamical Equations for Multibody Systems." The Int. J. Robotics Research 5, No. 3 (1986): 129-141.

[18] Murray, J. J. and Neuman, C. P. "ARM: An Algebraic Robot Dynamic Modeling Program." IEEE Intl. Conf. Robotics, Atlanta, Georgia, 1984.

[19] Cesareo, G. and Nicolo, F. "DYMIR: A Code for Generating Dynamic Models of Robots." IEEE Intl. Conf. Robotics, Atlanta, Georgia, 1984.

[20] Twu, J. and Krishnaswami, P. "Automated Generation of Robot Simulators Through Symbolic Computing." ASME Paper No. 86-DET-89, 1986.

[21] Stejskal, V. and Valasek, M. "Automated General Kinematic Analysis of Spatial Mechanisms Program OKAM2." ACTA Technica CSAV 31, No. 8 (1986): 137-148.

[22] Meirovitch, L. Methods of Analytical Dynamics. New York: McGraw-Hill, 1970.

[23] Wittenburg, J. Dynamics of Systems of Rigid Bodies. Stuttgart: B.G. Teubner, 1977.

[24] Kane, T.R. Dynamics: Theory and Applications. New York: McGraw-Hill, 1985.

[25] VAX UNIX MACSYMA Reference Manual, Version 11. Cambridge: Symbolics Inc., 1985.

[26] Kane, T.R. and Levinson, D.A. "The Use of Kane's Dynamical Equations in Robotics." The Int. J. Robotics Research 2, No. 3 (1983): 3-21.

[27] Sohoni, V. N. and Whitesell, J. "Automatic Linearization of Constrained Dynamic Models." J. Mechanisms, Transmissions and Automation in Design 108 (1986): 300-304.

[28] Moler, C. B. and Stewart, G. W. "An Algorithm for Generalized Matrix Eigenvalue Problems." SIAM J. of Numerical Analysis 10 (1973): 241-256.

[29] Golub, G. H. "Numerical Methods for Solving Linear Least Square Problems." Numerical Mathematics 7 (1965): 206-216.

[30] Atkinson, K. E. An Introduction to Numerical Analysis. New York: John Wiley & Sons, 1978.

9 APPENDIX A. INPUT FOR EXAMPLE MULTIBODY SYSTEMS

This appendix presents the symbolic/numerical data input for the example multibody systems discussed in Chapter 5. Section 1 presents the required symbolic input and comments for general systems. In Section 2, the symbolic input for the three degree of freedom pendulum is presented along with numerical data for the simulations discussed in Section 5.1. Section 3 displays the symbolic input for the six degree of freedom Stanford Arm. The numerical input was taken from Reference [26]. In Section 4, the symbolic/numerical input for the eight degree of freedom satellite system is presented.

9.1 Input for General Mechanical Systems

Input	Comments
1.) NBOD	Number of bodies in system (For grounded systems, include ground as a body)
2.) NGC	Number of generalized coordinates (For floating systems, NGC= NDOF + 1)
3.) **PIE**	NBOD × NBOD path matrix
4.) **REF**	NBOD × NBOD reference matrix
5.) **DM**	NBOD × NBOD distance matrix

6.) **RSDA**	NBOD × NBOD rotational spring-damper-actuator matrix $\mathbf{RSDA}_{ij} = \begin{cases} 1 & \text{If body i and j are connected by a revolute with an RSDA element} \\ 0 & \text{otherwise} \end{cases}$
7.) TSDAINFO 8.) GOTSDA	Translational spring-damper-actuator information $\text{GOTSDA} = \begin{cases} \text{START} & \text{TSDA information follows} \\ \text{STOP} & \text{No TSDA information} \end{cases}$
9.) NTSDA	Number of TSDA force elements
10.) **TSDA**	NTSDA × 2 connectivity matrix $\mathbf{TSDA}_{k1}$ = first body attached to TSDA $\mathbf{TSDA}_{k2}$ = second body attached to TSDA
11.) **PITSDA**	NTSDA × 3 matrix which contains the local x, y, z point of attachment for body $\mathbf{TSDA}_{k1}$
12.) **PJTSDA**	NTSDA × 3 matrix which contains the local x, y, z point of attachment for body $\mathbf{TSDA}_{k2}$
13.) **GRAVITY**	1 × 2 gravity matrix $\mathbf{GRAVITY}_{11}$ = Global x, y, or z gravitational field axis $\mathbf{GRAVITY}_{12}$ = −1.0, ..., +1.0 ratio of projection of gravitational field

	Revolute/translational information This sets up the transformation matrices between bodies i and j For the following: i, j= 1, ..., NBOD and $\mathbf{REF}_{ij} = -1$, $\mathbf{DM}_{ij} = 1.0$, or 1.1 (See Figure 9.1 for graphical representation of revolute/translational information)
14.) $\mathbf{pqi}_{ij}$	Revolute/translational joint axis unit vector w.r.t. body i
15.) $\mathbf{pqj}_{ij}$	Revolute/translational joint axis unit vector w.r.t. body j
16.) $\mathbf{rqi}_{ij}$	Revolute/translational unit vector perpendicular to $\mathbf{pqi}_{ij}$ w.r.t. body i
17.) $\mathbf{rqj}_{ij}$	Revolute/translational unit vector perpendicular to $\mathbf{pqj}_{ij}$ w.r.t. body j
18.) $\mathbf{dl}_{ii}$	Distance vectors from body i joint to body i center of gravity, i= 1, ..., NBOD (w.r.t. body i coordinates)
19.) $\mathbf{sl}_{ij}$	Distance vectors from body j center of gravity to body i joint (w.r.t. body j coordinates)

20.) **IBB**	1×2 base body array $\mathbf{IBB}_{11}$ = Number of base bodies $\mathbf{IBB}_{12} = \begin{cases} 0 & \text{if floating base body} \\ 1 & \text{if grounded base body} \end{cases}$
21.) $\mathbf{JIP}_i$	3×3 central inertia matrix for body i (i= 1, ..., NBOD)

9.2 Input for Three Degree of Freedom Pendulum

Symbolic Input

```
    NBOD: 4;
    NGC:  3;
    PIE: MATRIX(   [ 1, 0, 0, 0 ],
                   [ 1, 1, 0, 0 ],
                   [ 1, 1, 1, 0 ],
                   [ 1, 1, 1, 1 ]   );
    REF: MATRIX(   [ 1, 0, 0, 0 ],
                   [-1, 1, 0, 0 ],
                   [ 0,-1, 1, 0 ],
                   [ 0, 0,-1, 1 ]  );
    DM: MATRIX(   [0.0, 1.0, 0.0, 0.0],
                  [1.0, 0.0, 1.0, 0.0],
                  [0.0, 1.0, 0.0, 1.0],
                  [0.0, 0.0, 1.0, 0.0]  );
    RSDA: MATRIX(   [ 0, 0, 0, 0 ],
                    [ 1, 0, 0, 0 ],
                    [ 0, 1, 0, 0 ],
                    [ 0, 0, 1, 0 ]  );
      TSDAINFO;
      GOTSDA: STOP;

  GRAVITY: MATRIX( [ Y, -1 ] );

PQI[2,1]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
PQJ[2,1]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
RQI[2,1]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
RQJ[2,1]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
PQI[3,2]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
PQJ[3,2]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
RQI[3,2]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
RQJ[3,2]:   MATRIX( [ 0 ], [ 0 ], [-1 ] );
PQI[4,3]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
PQJ[4,3]:   MATRIX( [-1 ], [ 0 ], [ 0 ] );
RQI[4,3]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
```

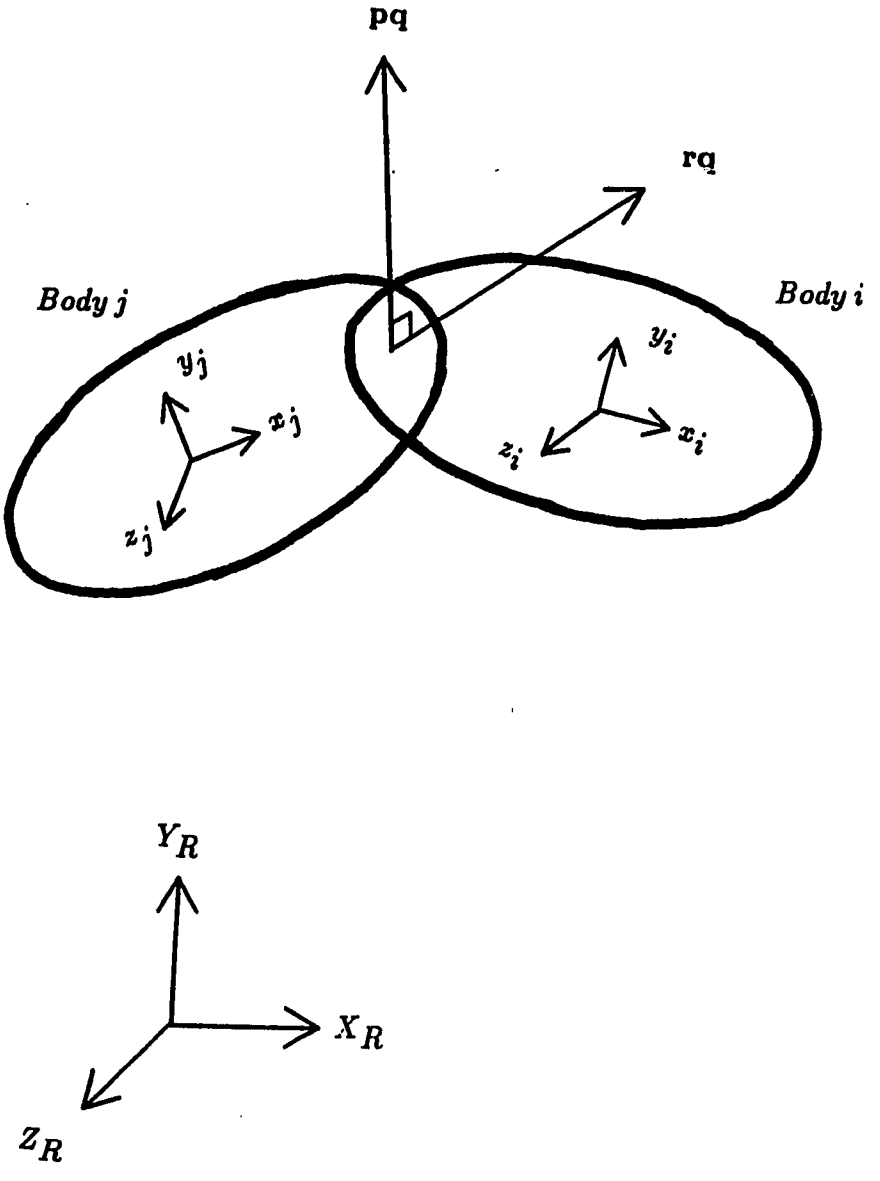


Figure 9.1: Graphical representation of revolute/translational joint information

```
RQJ[4,3]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
DL[1,1]:   MATRIX( [ 0 ], [ 0 ], [ 0 ] );
DL[2,2]:   MATRIX( [ 0 ], [-DL22], [ 0 ] );
DL[3,3]:   MATRIX( [ 0 ], [-DL33], [ 0 ] );
DL[4,4]:   MATRIX( [ 0 ], [-DL44], [ 0 ] );
SL[2,1]:   MATRIX( [ 0 ], [ 0 ], [ 0 ] );
SL[3,2]:   MATRIX( [ 0 ], [-SL32], [ 0 ] );
SL[4,3]:   MATRIX( [ 0 ], [-SL43], [ 0 ] );
IBB:   MATRIX( [ 1, 1 ] );

JIP[1]: MATRIX( [  0 ,  0 ,  0 ],
                [  0 ,  0 ,  0 ],
                [  0 ,  0 ,  0 ]  );

JIP[2]: MATRIX( [ J2X,  0 ,  0 ],
                [  0 , J2Y,  0 ],
                [  0 ,  0 , J2Z]  );

JIP[3]: MATRIX( [ J3X,  0 ,  0 ],
                [  0 , J3Y,  0 ],
                [  0 ,  0 , J3Z]  );

JIP[4]: MATRIX( [ J4X,  0 ,  0 ],
                [  0 , J4Y,  0 ],
                [  0 ,  0 , J4Z]  );
```

Numerical Input

Simulation 1

Link lengths: $DL22 = DL33 = DL44 = 0.5m$

$SL32 = SL43 = 0.5m$

Masses: $m_i = 1.0\,kg,\ i = 1, 2, 3$

Inertias: $\mathbf{JIP}_i = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.2 \end{bmatrix} kg\,m^2 \quad i = 1, 2, 3$

Simulation 2

Simulation 1 data and

Rotational damping: $c_i = 5.0\,Nms \quad i = 1, 2, 3$

9.3 Symbolic Input for Six Degree of Freedom Stanford Arm

```
NBOD:      7;
NGC:       6;
PIE: MATRIX(    [ 1, 0, 0, 0, 0, 0, 0 ],
                [ 1, 1, 0, 0, 0, 0, 0 ],
                [ 1, 1, 1, 0, 0, 0, 0 ],
                [ 1, 1, 1, 1, 0, 0, 0 ],
                [ 1, 1, 1, 1, 1, 0, 0 ],
                [ 1, 1, 1, 1, 1, 1, 0 ],
                [ 1, 1, 1, 1, 1, 1, 1 ]    );
REF: MATRIX(    [ 1, 0, 0, 0, 0, 0, 0 ],
                [-1, 1, 0, 0, 0, 0, 0 ],
                [ 0,-1, 1, 0, 0, 0, 0 ],
                [ 0, 0,-1, 1, 0, 0, 0 ],
                [ 0, 0, 0,-1, 1, 0, 0 ],
                [ 0, 0, 0, 0,-1, 1, 0 ],
                [ 0, 0, 0, 0, 0,-1, 1 ]    );
DM: MATRIX( [ 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0 ],
```

```
            [ 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0 ],
            [ 0.0, 1.0, 0.0, 1.1, 0.0, 0.0, 0.0 ],
            [ 0.0, 0.0, 1.1, 0.0, 1.0, 0.0, 0.0 ],
            [ 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0 ],
            [ 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0 ],
            [ 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0 ]  );
RSDA: MATRIX(  [ 0, 0, 0, 0, 0, 0, 0 ],
               [ 1, 0, 0, 0, 0, 0, 0 ],
               [ 0, 1, 0, 0, 0, 0, 0 ],
               [ 0, 0, 0, 0, 0, 0, 0 ],
               [ 0, 0, 0, 1, 0, 0, 0 ],
               [ 0, 0, 0, 0, 1, 0, 0 ],
               [ 0, 0, 0, 0, 0, 1, 0 ]  );
            TSDAINFO;
            GOTSDA: START;
            NTSDA :  1;
            TSDA  : MATRIX( [ 4, 3 ] );
            PITSDA: MATRIX( [ 0, 0, 0 ] );
            PJTSDA: MATRIX( [ 0, 0, 0 ] );
       GRAVITY: MATRIX( [ Y, -1 ] );
PQI[2,1]:  MATRIX( [0], [1], [0] );
PQJ[2,1]:  MATRIX( [0], [1], [0] );
RQI[2,1]:  MATRIX( [0], [0], [1] );
RQJ[2,1]:  MATRIX( [0], [0], [1] );
PQI[3,2]:  MATRIX( [1], [0], [0] );
PQJ[3,2]:  MATRIX( [1], [0], [0] );
RQI[3,2]:  MATRIX( [0], [0], [1] );
RQJ[3,2]:  MATRIX( [0], [0], [1] );
PQI[4,3]:  MATRIX( [0], [1], [0] );
PQJ[4,3]:  MATRIX( [0], [1], [0] );
RQI[4,3]:  MATRIX( [0], [0], [1] );
RQJ[4,3]:  MATRIX( [0], [0], [1] );
PQI[5,4]:  MATRIX( [0], [1], [0] );
PQJ[5,4]:  MATRIX( [0], [1], [0] );
RQI[5,4]:  MATRIX( [0], [0], [1] );
RQJ[5,4]:  MATRIX( [0], [0], [1] );
PQI[6,5]:  MATRIX( [1], [0], [0] );
PQJ[6,5]:  MATRIX( [1], [0], [0] );
RQI[6,5]:  MATRIX( [0], [0], [1] );
```

```
RQJ[6,5]:  MATRIX( [0], [0], [1] );
PQI[7,6]:  MATRIX( [0], [1], [0] );
PQJ[7,6]:  MATRIX( [0], [1], [0] );
RQI[7,6]:  MATRIX( [0], [0], [1] );
RQJ[7,6]:  MATRIX( [0], [0], [1] );
  DL[1,1]:      MATRIX( [ 0 ], [ 0 ], [ 0 ] );
  DL[2,2]:      MATRIX( [ 0 ], [ 0 ], [ 0 ] );
  DL[3,3]:      MATRIX( [DL33], [ 0 ], [ 0 ] );
  DL[4,4]:      MATRIX( [ 0 ], [PHI[4]], [ 0 ] );
  DL[5,5]:      MATRIX( [ 0 ], [ 0 ], [ 0 ] );
  DL[6,6]:      MATRIX( [ 0 ], [DL66], [ 0 ] );
  DL[7,7]:      MATRIX( [ 0 ], [ 0 ], [ 0 ] );
 SL[2,1]:      MATRIX( [ 0 ], [ 0 ], [ 0 ] );
 SL[3,2]:      MATRIX( [ 0 ], [SL32], [ 0 ] );
 SL[4,3]:      MATRIX( [ 0 ], [ 0 ], [ 0 ] );
 SL[5,4]:      MATRIX( [ 0 ], [SL54], [ 0 ] );
 SL[6,5]:      MATRIX( [ 0 ], [SL65], [ 0 ] );
 SL[7,6]:      MATRIX( [ 0 ], [SL76], [ 0 ] );
        IBB: MATRIX( [ 1, 1 ] );
 JIP[1]: MATRIX( [  0 ,  0 ,  0 ],
                 [  0 ,  0 ,  0 ],
                 [  0 ,  0 ,  0 ]  );
 JIP[2]: MATRIX( [ J2X,  0 ,  0 ],
                 [  0 , J2Y,  0 ],
                 [  0 ,  0 , J2Z]  );
 JIP[3]: MATRIX( [ J3X,  0 ,  0 ],
                 [  0 , J3Y,  0 ],
                 [  0 ,  0 , J3Z]  );
 JIP[4]: MATRIX( [ J4X,  0 ,  0 ],
                 [  0 , J4Y,  0 ],
                 [  0 ,  0 , J4Z]  );
 JIP[5]: MATRIX( [ J5X,  0 ,  0 ],
                 [  0 , J5Y,  0 ],
                 [  0 ,  0 , J5Z]  );
 JIP[6]: MATRIX( [ J6X,  0 ,  0 ],
                 [  0 , J6Y,  0 ],
                 [  0 ,  0 , J6Z]  );
```

```
JIP[7]: MATRIX( [ J7X,  0 ,  0 ],
                [  0 , J7Y,  0 ],
                [  0 ,  0 , J7Z]  );
```

9.4 Input for Eight Degree of Freedom Satellite

Symbolic Input

```
NBOD:  3;
NGC:   9;
PIE: MATRIX(  [ 1, 0, 0 ],
              [ 1, 1, 0 ],
              [ 1, 0, 1 ]  );
REF: MATRIX(  [ 1, 0, 0 ],
              [-1, 1, 0 ],
              [-1, 0, 1 ]  );
DM: MATRIX(   [ 0.2, 1.0, 1.0 ],
              [ 1.0, 0.0, 0.0 ],
              [ 1.0, 0.0, 0.0 ]  );
RSDA: MATRIX(  [ 0, 1, 1 ],
               [ 1, 0, 0 ],
               [ 1, 0, 0 ]  );
   TSDAINFO;
   GOTSDA: STOP;
  GRAVITY: MATRIX( [ Y, 0 ] );
PQI[2,1]: MATRIX( [ 0 ], [ 0 ], [ 1 ] );
PQJ[2,1]: MATRIX( [ 0 ], [ 0 ], [ 1 ] );
RQI[2,1]: MATRIX( [ 1 ], [ 0 ], [ 0 ] );
RQJ[2,1]: MATRIX( [ 1 ], [ 0 ], [ 0 ] );
PQI[3,1]: MATRIX( [ 0 ], [ 0 ], [ 1 ] );
PQJ[3,1]: MATRIX( [ 0 ], [ 0 ], [ 1 ] );
RQI[3,1]: MATRIX( [ 1 ], [ 0 ], [ 0 ] );
RQJ[3,1]: MATRIX( [ 1 ], [ 0 ], [ 0 ] );
  DL[1,1]: MATRIX( [ 0 ], [ 0 ], [ 0 ] );
  DL[2,2]: MATRIX( [ 0 ], [DL22], [ 0 ] );
  DL[3,3]: MATRIX( [ 0 ], [DL33], [ 0 ] );
```

```
SL[2,1]: MATRIX( [SL21], [ 0 ], [ 0 ] );
SL[3,1]: MATRIX( [SL31], [ 0 ], [ 0 ] );
IBB:   MATRIX( [ 1, 0 ] );
JIP[1] : MATRIX( [I1X,  0,  0],
                 [ 0,I1Y,  0],
                 [ 0,  0,I1Z] );
JIP[2] : MATRIX( [I2X,  0,  0],
                 [ 0,I2Y,  0],
                 [ 0,  0,I2Z] );
JIP[3] : MATRIX( [I3X,  0,  0],
                 [ 0,I3Y,  0],
                 [ 0,  0,I3Z] );
```

Numerical Input

Link lengths: $DL22 = DL33 = 0.1\,m$

$SL21 = 0.1\,m$, $SL31 = -0.1\,m$

Masses: $m_1 = 20\,kg$, and $m_2 = m_3 = 10\,kg$

Inertias:

$$\mathbf{JIP}_1 = \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0 & 0.25 \end{bmatrix} kg\,m^2$$

$$\mathbf{JIP}_2 = \mathbf{JIP}_3 = \begin{bmatrix} 0.04 & 0 & 0 \\ 0 & 0.02 & 0 \\ 0 & 0 & 0.04 \end{bmatrix} kg\,m^2$$

10 APPENDIX B. SYMBOLIC EQUATIONS OF MOTION FOR TRIPLE PENDULUM

```
C
C=====================================================================
C                      NONLINEAR TRIPLE PENDULUM
C                         EQUATIONS OF MOTION
C=====================================================================
C
C---Nonlinear mass matrix... LHS1
C
 lhs1(1,1) = ma(4)*c(6)**2*s143**2+(2*ma(4)*c(6)*s132+2*ma(4)*c(6)*
1    *2*c(7)*d144+2*ma(4)*c(6)**2*d133+2*ma(4)*c(6)*d122)*s143+(ma(4
2    )+ma(3))*s132**2+(2*ma(4)*c(6)*c(7)*d144+(2*ma(4)+2*ma(3))*c(6)
3    *d133+(2*ma(4)+2*ma(3))*d122)*s132+c(6)**2*j4z+(1-c(6)**2)*c(7)
4    **2*j4y+((c(6)**2-1)*c(7)**2-c(6)**2+1)*j4x+(1-c(6)**2)*j3y+c(6
5    )**2*j3x+j2z+((ma(4)*c(6)**2-ma(4))*c(7)**2+ma(4))*d144**2+(2*m
6    a(4)*c(6)**2*c(7)*d133+2*ma(4)*c(6)*c(7)*d122)*d144+(ma(4)+ma(3
7    ))*c(6)**2*d133**2+(2*ma(4)+2*ma(3))*c(6)*d122*d133+(ma(4)+ma(3
8    )+ma(2))*d122**2
 lhs1(1,2) = ma(4)*s(6)*s(7)*d144*s143-s(6)*c(7)*s(7)*j4y+s(6)*c(7)
1    *s(7)*j4x+ma(4)*s(6)*c(7)*s(7)*d144**2+ma(4)*s(6)*s(7)*d133*d14
2    4
 lhs1(1,3) = ma(4)*c(6)*c(7)*d144*s143+ma(4)*c(7)*d144*s132+c(6)*j4
1    z+ma(4)*c(6)*d144**2+(ma(4)*c(6)*c(7)*d133+ma(4)*c(7)*d122)*d14
2    4
 lhs1(2,1) = ma(4)*s(6)*s(7)*d144*s143-s(6)*c(7)*s(7)*j4y+s(6)*c(7)
1    *s(7)*j4x+ma(4)*s(6)*c(7)*s(7)*d144**2+ma(4)*s(6)*s(7)*d133*d14
2    4
 lhs1(2,2) = ma(4)*s143**2+(2*ma(4)*c(7)*d144+2*ma(4)*d133)*s143+(1
```

```
1    -c(7)**2)*j4y+c(7)**2*j4x+j3z+ma(4)*c(7)**2*dl44**2+2*ma(4)*c(7
2    )*dl33*dl44+(ma(4)+ma(3))*dl33**2
 lhs1(2,3) = 0
 lhs1(3,1) = ma(4)*c(6)*c(7)*dl44*sl43+ma(4)*c(7)*dl44*sl32+c(6)*j4
1    z+ma(4)*c(6)*dl44**2+(ma(4)*c(6)*c(7)*dl33+ma(4)*c(7)*dl22)*dl4
2    4
 lhs1(3,2) = 0
 lhs1(3,3) = j4z+ma(4)*dl44**2
C
C---Nonlinear force term... RHS1
C
 rhs1(1,1) = (2*y(1)*y(2)*ma(4)*c(6)*s(6)*s(7)**2+2*y(1)*y(2)*ma(4)
1    *c(6)*s(6)*c(7)**2)*sl43**2+((2*y(1)*y(2)*ma(4)*s(6)*s(7)**2+2*
2    y(1)*y(2)*ma(4)*s(6)*c(7)**2)*sl32+(-ma(4)*s(5)*c(6)*s(7)**2-ma
3    (4)*s(5)*c(6)*c(7)**2)*grav+(2*y(1)*y(2)*ma(4)*c(6)*s(6)*c(7)*s
4    (7)**2+(-y(1)**2*ma(4)*c(6)*s(6)**2-y(1)**2*ma(4)*c(6)**3-y(2)*
5    *2*ma(4)*c(6)+y(1)**2*ma(4)*c(6))*s(7)+2*y(1)*y(3)*ma(4)*c(6)**
6    2*s(7)+y(3)**2*ma(4)*c(6)*s(7)+2*y(1)*y(2)*ma(4)*c(6)*s(6)*c(7)
7    **3+2*y(1)*y(2)*ma(4)*c(6)*s(6)*c(7))*dl44+(4*y(1)*y(2)*ma(4)*c
8    (6)*s(6)*s(7)**2+4*y(1)*y(2)*ma(4)*c(6)*s(6)*c(7)**2)*dl33+(2*y
9    (1)*y(2)*ma(4)*s(6)*s(7)**2+2*y(1)*y(2)*ma(4)*s(6)*c(7)**2)*dl2
:    2)*sl43
 rhs1(1,1)= rhs1(1,1)
&           +((-ma(4)*s(5)*s(7)**2-ma(4)*s(5)*c(7)**2-ma(3)*s(5))*gr
;    av+(2*y(1)*y(2)*ma(4)*s(6)*c(7)*s(7)**2+(-y(1)**2*ma(4)*s(6)**2
<    -y(1)**2*ma(4)*c(6)**2+y(1)**2*ma(4))*s(7)+2*y(1)*y(3)*ma(4)*c(
=    6)*s(7)+y(3)**2*ma(4)*s(7)+2*y(1)*y(2)*ma(4)*s(6)*c(7)**3)*dl44
>    +(2*y(1)*y(2)*ma(4)*s(6)*s(7)**2+2*y(1)*y(2)*ma(4)*s(6)*c(7)**2
?    +2*y(1)*y(2)*ma(3)*s(6))*dl33)*sl32+(y(3)*(y(2)*s(6)*s(7)**2+y(
@    2)*s(6)*c(7)**2)+y(1)*y(2)*c(6)*s(6)*s(7)**2+y(1)*y(2)*c(6)*s(6
1    )*c(7)**2+y(1)*y(2)*c(6)*s(6))*j4z+(y(3)*(-y(2)*s(6)*s(7)**2+2*
2    y(1)*s(6)**2*c(7)*s(7)+y(2)*s(6)*c(7)**2)+y(2)**2*c(6)*c(7)*s(7
3    )-2*y(1)*y(2)*c(6)*s(6)*c(7)**2)*j4y+(y(3)*(y(2)*s(6)*s(7)**2-2
4    *y(1)*s(6)**2*c(7)*s(7)-y(2)*s(6)*c(7)**2)-2*y(1)*y(2)*c(6)*s(6
5    )*s(7)**2-y(2)**2*c(6)*c(7)*s(7))*j4x-2*y(1)*y(2)*c(6)*s(6)*j3y
6    +2*y(1)*y(2)*c(6)*s(6)*j3x
 rhs1(1,1)= rhs1(1,1)
&                              +(((-ma(4)*c(5)*s(6)**2-ma(4)*c(5)*c(
7    6)**2)*s(7)-ma(4)*s(5)*c(6)*c(7))*dl44+(-ma(4)*s(5)*c(6)*s(7)**
```

```
8    2-ma(4)*s(5)*c(6)*c(7)**2-ma(3)*s(5)*c(6))*dl33+(-ma(4)*s(5)*s(
9    7)**2-ma(4)*s(5)*c(7)**2+(-ma(3)-ma(2))*s(5))*dl22)*grav
 rhs1(1,1)= rhs1(1,1)
&                                                          +(y(3)*
:    (2*y(2)*ma(4)*s(6)*s(7)**2-2*y(1)*ma(4)*s(6)**2*c(7)*s(7))+(-y(
;    1)**2*ma(4)*c(6)*s(6)**2-y(1)**2*ma(4)*c(6)**3-y(2)**2*ma(4)*c(
<    6)+y(1)**2*ma(4)*c(6))*c(7)*s(7)+2*y(1)*y(2)*ma(4)*c(6)*s(6)*c(
=    7)**2)*dl44**2+((2*y(1)*y(2)*ma(4)*c(6)*s(6)*c(7)*s(7)**2+(-y(1
>    )**2*ma(4)*c(6)*s(6)**2-y(1)**2*ma(4)*c(6)**3-y(2)**2*ma(4)*c(6
?    )+y(1)**2*ma(4)*c(6))*s(7)+2*y(1)*y(3)*ma(4)*c(6)**2*s(7)+y(3)*
@    *2*ma(4)*c(6)*s(7)+2*y(1)*y(2)*ma(4)*c(6)*s(6)*c(7)**3+2*y(1)*y
1    (2)*ma(4)*c(6)*s(6)*c(7))*dl33+(2*y(1)*y(2)*ma(4)*s(6)*c(7)*s(7
2    )**2+(-y(1)**2*ma(4)*s(6)**2-y(1)**2*ma(4)*c(6)**2+y(1)**2*ma(4
3    ))*s(7)+2*y(1)*y(3)*ma(4)*c(6)*s(7)+y(3)**2*ma(4)*s(7)+2*y(1)*y
4    (2)*ma(4)*s(6)*c(7)**3)*dl22)*dl44
 rhs1(1,1)= rhs1(1,1)
&                                              +(2*y(1)*y(2)*ma(4)*c(6)*s(6)
5    *s(7)**2+2*y(1)*y(2)*ma(4)*c(6)*s(6)*c(7)**2+2*y(1)*y(2)*ma(3)*
6    c(6)*s(6))*dl33**2+(2*y(1)*y(2)*ma(4)*s(6)*s(7)**2+2*y(1)*y(2)*
7    ma(4)*s(6)*c(7)**2+2*y(1)*y(2)*ma(3)*s(6))*dl22*dl33+kr(2,1)*(p
8    hi0(2)-y(4))+tau(2,1)-y(1)*cr(2,1)
 rhs1(2,1) = -y(1)**2*ma(4)*c(6)*s(6)*sl43**2+(-y(1)**2*ma(4)*s(6)*
1    sl32-ma(4)*c(5)*s(6)*grav+(y(3)*(2*y(2)*ma(4)*s(7)-2*y(1)*ma(4)
2    *s(6)*c(7))-2*y(1)**2*ma(4)*c(6)*s(6)*c(7))*dl44-2*y(1)**2*ma(4
3    )*c(6)*s(6)*dl33-y(1)**2*ma(4)*s(6)*dl22)*sl43+(y(1)**2*(-ma(4)
4    -ma(3))*s(6)*dl33-y(1)**2*ma(4)*s(6)*c(7)*dl44)*sl32+(y(3)*(-y(
5    1)*s(6)*s(7)**2-y(1)*s(6)*c(7)**2)-y(1)**2*c(6)*s(6)*s(7)**2-y(
6    1)**2*c(6)*s(6)*c(7)**2)*j4z
 rhs1(2,1)= rhs1(2,1)
&                                          +(y(3)*(-y(1)*s(6)*s(7)**2-2*y(2)*c
7    (7)*s(7)+y(1)*s(6)*c(7)**2)+y(1)**2*c(6)*s(6)*c(7)**2)*j4y+(y(3
8    )*(y(1)*s(6)*s(7)**2+2*y(2)*c(7)*s(7)-y(1)*s(6)*c(7)**2)+y(1)**
9    2*c(6)*s(6)*s(7)**2)*j4x+y(1)**2*c(6)*s(6)*j3y-y(1)**2*c(6)*s(6
:    )*j3x+((-ma(4)-ma(3))*c(5)*s(6)*dl33-ma(4)*c(5)*s(6)*c(7)*dl44)
;    *grav+(y(3)*(2*y(2)*ma(4)*c(7)*s(7)-2*y(1)*ma(4)*s(6)*c(7)**2)-
<    y(1)**2*ma(4)*c(6)*s(6)*c(7)**2)*dl44**2+((y(3)*(2*y(2)*ma(4)*s
=    (7)-2*y(1)*ma(4)*s(6)*c(7))-2*y(1)**2*ma(4)*c(6)*s(6)*c(7))*dl3
>    3-y(1)**2*ma(4)*s(6)*c(7)*dl22)*dl44+y(1)**2*(-ma(4)-ma(3))*c(6
?    )*s(6)*dl33**2+y(1)**2*(-ma(4)-ma(3))*s(6)*dl22*dl33+kr(3,2)*(p
```

```
@    hi0(3)-y(5))+tau(3,2)-y(2)*cr(3,2)
 rhs1(3,1) = ((-y(1)**2*ma(4)*c(6)**2-y(2)**2*ma(4))*s(7)+2*y(1)*y(
1    2)*ma(4)*s(6)*c(7))*dl44*sl43-y(1)**2*ma(4)*c(6)*s(7)*dl44*sl32
2    +y(1)*y(2)*s(6)*j4z+(y(1)*y(2)*s(6)*s(7)**2+(y(2)**2-y(1)**2*s(
3    6)**2)*c(7)*s(7)-y(1)*y(2)*s(6)*c(7)**2)*j4y+(-y(1)*y(2)*s(6)*s
4    (7)**2+(y(1)**2*s(6)**2-y(2)**2)*c(7)*s(7)+y(1)*y(2)*s(6)*c(7)*
5    *2)*j4x+(-ma(4)*c(5)*c(6)*s(7)-ma(4)*s(5)*c(7))*dl44*grav+((-y(
6    1)**2*ma(4)*c(6)**2-y(2)**2*ma(4)+y(1)**2*ma(4))*c(7)*s(7)+2*y(
7    1)*y(2)*ma(4)*s(6)*c(7)**2)*dl44**2+(((-y(1)**2*ma(4)*c(6)**2-y
8    (2)**2*ma(4))*s(7)+2*y(1)*y(2)*ma(4)*s(6)*c(7))*dl33-y(1)**2*ma
9    (4)*c(6)*s(7)*dl22)*dl44+kr(4,3)*(phi0(4)-y(6))+tau(4,3)-y(3)*c
:    r(4,3)
C
C=====================================================================
C                        LINEAR TRIPLE PENDULUM
C                          EQUATIONS OF MOTION
C
C     ABOUT      Q0(i) = 0.0 rad
C          d(Q0(i))/dt = 0.0 rad/sec      (i = 1, ..., 3)
C
C=====================================================================
C
C---Linearized mass matrix
C
 mlin(1,1) = ma(4)*sl43**2+(2*ma(4)*sl32+2*ma(4)*dl44+2*ma(4)*dl33+
1    2*ma(4)*dl22)*sl43+(ma(4)+ma(3))*sl32**2+(2*ma(4)*dl44+(2*ma(4)
2    +2*ma(3))*dl33+(2*ma(4)+2*ma(3))*dl22)*sl32+j4z+j3x+j2z+ma(4)*d
3    l44**2+(2*ma(4)*dl33+2*ma(4)*dl22)*dl44+(ma(4)+ma(3))*dl33**2+(
4    2*ma(4)+2*ma(3))*dl22*dl33+(ma(4)+ma(3)+ma(2))*dl22**2
 mlin(1,2) = 0
 mlin(1,3) = ma(4)*dl44*sl43+ma(4)*dl44*sl32+j4z+ma(4)*dl44**2+(ma(
1    4)*dl33+ma(4)*dl22)*dl44
 mlin(2,1) = 0
 mlin(2,2) = ma(4)*sl43**2+(2*ma(4)*dl44+2*ma(4)*dl33)*sl43+j4x+j3z
1    +ma(4)*dl44**2+2*ma(4)*dl33*dl44+(ma(4)+ma(3))*dl33**2
 mlin(2,3) = 0
 mlin(3,1) = ma(4)*dl44*sl43+ma(4)*dl44*sl32+j4z+ma(4)*dl44**2+(ma(
1    4)*dl33+ma(4)*dl22)*dl44
 mlin(3,2) = 0
```

```
 mlin(3,3) = j4z+ma(4)*dl44**2
C
C---Linearized damping matrix
C
 clin(1,1) = -cr(2,1)
 clin(1,2) = 0
 clin(1,3) = 0
 clin(2,1) = 0
 clin(2,2) = -cr(3,2)
 clin(2,3) = 0
 clin(3,1) = 0
 clin(3,2) = 0
 clin(3,3) = -cr(4,3)
C
C---Linearized stiffness matrix
C
 klin(1,1) = -ma(4)*grav*sl43+(-ma(4)-ma(3))*grav*sl32+(-ma(4)*dl4
1   4+(-ma(4)-ma(3))*dl33+(-ma(4)-ma(3)-ma(2))*dl22)*grav-kr(2,1)
 klin(1,2) = 0
 klin(1,3) = -ma(4)*dl44*grav
 klin(2,1) = 0
 klin(2,2) = -ma(4)*grav*sl43+((-ma(4)-ma(3))*dl33-ma(4)*dl44)*gra
1   v-kr(3,2)
 klin(2,3) = 0
 klin(3,1) = -ma(4)*dl44*grav
 klin(3,2) = 0
 klin(3,3) = -ma(4)*dl44*grav-kr(4,3)
```

12 APPENDIX C. INPUT FOR CONSTRAINED FOUR BAR MECHANSIM

Symbolic Input

```
    NBOD: 5;
    NGC:  4;
    PIE: MATRIX(   [ 1, 0, 0, 0 , 0 ],
                   [ 1, 1, 0, 0 , 0 ],
                   [ 1, 0, 1, 0 , 0 ],
                   [ 1, 0, 1, 1 , 0 ],
                   [ 1, 0, 1, 1 , 1 ]  );
    REF: MATRIX(   [ 1, 0, 0, 0 , 0 ],
                   [-1, 1, 0, 0 , 0 ],
                   [-1, 0, 1, 0 , 0 ],
                   [ 0, 0,-1, 1 , 0 ],
                   [ 0, 0, 0,-1 , 1 ]  );
    DM: MATRIX(   [0.0, 1.0, 1.0, 0.0, 0.0],
                  [1.0, 0.0, 0.0, 0.0, 0.0],
                  [1.0, 0.0, 0.0, 1.0, 0.0],
                  [0.0, 0.0, 1.0, 0.0, 1.0],
                  [0.0, 0.0, 0.0, 1.0, 0.0]  );
    RSDA: MATRIX(   [ 0, 0, 0, 0, 0 ],
                    [ 0, 0, 0, 0, 0 ],
                    [ 0, 0, 0, 0, 0 ],
                    [ 0, 0, 0, 0, 0 ],
                    [ 0, 0, 0, 0, 0 ]  );
      TSDAINFO;
      GOTSDA: STOP;
  GRAVITY: MATRIX( [ Y, -1 ] );
PQI[2,1]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
```

```
PQJ[2,1]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
RQI[2,1]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
RQJ[2,1]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
PQI[3,1]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
PQJ[3,1]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
RQI[3,1]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
RQJ[3,1]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
PQI[4,3]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
PQJ[4,3]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
RQI[4,3]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
RQJ[4,3]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
PQI[5,4]:   MATRIX( [ 0 ], [ 0 ], [ 1 ] );
PQJ[5,4]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
RQI[5,4]:   MATRIX( [ 1 ], [ 0 ], [ 0 ] );
RQJ[5,4]:   MATRIX( [ 0 ], [ 0 ], [-1 ] );
DL[1,1]:   MATRIX( [ 0 ], [ 0 ], [ 0 ] );
DL[2,2]:   MATRIX( [ 0 ], [-DL22], [ 0 ] );
DL[3,3]:   MATRIX( [ 0 ], [-DL33], [ 0 ] );
DL[4,4]:   MATRIX( [ 0 ], [ 0 ], [ 0 ] );
DL[5,5]:   MATRIX( [ 0 ], [-DL55], [ 0 ] );
SL[2,1]:   MATRIX( [ 0 ], [ 0 ], [ 0 ] );
SL[3,1]:   MATRIX( [ 0 ], [ 0 ], [ 0 ] );
SL[4,3]:   MATRIX( [ 0 ], [-SL43], [ 0 ] );
SL[5,4]:   MATRIX( [ 0 ], [ 0 ], [ 0 ] );
IBB:   MATRIX( [ 1, 1 ] );
JIP[1]: MATRIX( [  0 ,  0 ,  0 ],
                [  0 ,  0 ,  0 ],
                [  0 ,  0 ,  0 ]  );
JIP[2]: MATRIX( [  0 ,  0 ,  0 ],
                [  0 ,  0 ,  0 ],
                [  0 ,  0 , J2Z]  );
JIP[3]: MATRIX( [  0 ,  0 ,  0 ],
                [  0 ,  0 ,  0 ],
                [  0 ,  0 , J3Z]  );
JIP[4]: MATRIX( [  0 ,  0 ,  0 ],
                [  0 ,  0 ,  0 ],
                [  0 ,  0 , J4Z]  );
```

```
JIP[5]: MATRIX( [ J5X,  0 ,  0 ],
                [  0 , J5Y,  0 ],
                [  0 ,  0 , J5Z] );
```

Numerical Input

Link lengths: $DL22 = DL33 = DL44 = 0.5\,m$

$SL31 = 0.0$

$SL43 = 0.5\,m$

Masses: $M_i = 10.0\,kg,\ i = 2, 3, 5$

$M_4 = 0.1\,kg$

Inertias: $JiZ = 0.1\,kg\,m^2\ \ i = 2, 3, 5$

$J5X = 0.1\,kg\,m^2$

$J5Y = 0.1\,kg\,m^2$

$J4Z = 0.001\,kg\,m^2$