NONLINEAR STRAIN-DISPLACEMENT RELATIONS IN THE DYNAMICS OF A TWO-LINK FLEXIBLE MANIPULATOR

by

Carlos Eduardo Padilla Santos

B.S. in Mechanical and Aerospace Engineering
Princeton University
(1986)

SUBMITTED TO THE DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May 1989

Signature of Author_____
Department of Aeronautics and Astronautics
May 12, 1989

Certified by_____
Prof. Andreas von Flotow
Assistant Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by_____
Prof. Harold Y. Wachman
Chairman, Departmental Graduate Committee

ABSTRACT

The equations of motion of a flexible, two-link, planar manipulator are derived via the Kane formalism, correctly linearized in small elastic deflections and speeds through the use of nonlinear strain-displacement relations. Ordinary differential equations are obtained as a result of modelling the links as Bernoulli-Euler beams and expanding the elastic deflections in terms of cantilever modes. A simulation is implemented that allows the determination of the manipulator's dynamic response to desired forcing functions in the form of joint torques. Slew maneuver simulation results are then compared for models with and without the properly modelled kinematics of deformation, in order to quantify the relative significance of certain nonlinear terms in the motion equations. It is found that rigid body motion limits exist below which elastic nonlinear terms are negligible. These are nonlinear terms that involve the generalized elastic coordinates and their time derivatives. These rigid body motion limits are set by the requirement that inconsistently linearized equations of motion (those derived using linear strain-displacement or linear kinematics of deformation) yield accurate results. Within these limits, it is found that equations derived ignoring all elastic nonlinear terms produce simulation results that are as good as the inconsistently and consistently linearized equations.

ACKNOWLEDGEMENTS

This research work was made possible by a National Science Foundation Minority Graduate Fellowship. The author wishes to thank the people at NSF who made the award possible.

Special thanks go to Prof. Andy von Flotow for his invaluable support and guidance. His accessibility, interest, and quick insight helped create an enjoyable and challenging environment.

The author also wishes to thank the many friends and relatives whose unfaltering support and caring made the work bearable; in particular Dave for his timely advice, and Javier for pushing to the limit.

Finally, I want to continue thanking my father, mother, sister, and brother for always being there.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Manipulators for space applications need to be lightweight for economical reasons. It is also desirable that they be faster and more autonomous than the SRMS (Space Shuttle Remote Manipulator System), which moves relatively slowly (0.1 deg/sec) in order to minimize elastic excitation, and is teleoperated. The first of these requirements results in manipulators with significant structural flexibility (the SRMS has a natural vibration frequency of 0.3 Hz unloaded and with extended and locked links); while the need to expand the performance/operational envelope requires that this flexibility be modelled accurately.

Much work has been done on the dynamic modelling of open chains of rigid and elastic bodies (see references in [25],[12]). The heightened interest in multibody simulation, particularly flexible multibody simulation, prompted a workshop on the subject sponsored by the Jet Propulsion Laboratory late last year [15]. As stated in the preface to the Proceedings of the Workshop, this interest is being driven by mission requirements put forth by NASA and the SDIO that call for high angular rate and acceleration articulation of complex arrangements of interconnected rigid and flexible bodies.

A review of this literature shows that it is common practice to assume small elastic deflections and beam-like links when modelling structural flexibility in appendages or robotic links. Small rigid body motions and velocities (translations and relative angular motion of the links) have sometimes been assumed to simplify the equations of motion. In particular, so-called "rate-linear" equations have been proposed to yield tractable motion equations when the rigid body rates are assumed small enough so that second order terms can be neglected [8]. It has been pointed out by Hollerbach [4], however, that gyroscopic terms (which are second order in rigid body rates) are as important as inertia terms in the motion equations for open chains of rigid links. For this reason, it would seem desirable to obtain motion equations that do not ignore these gyroscopic terms and are in consequence valid for rigid body motions. Some work has been done in this respect for the case of a manipulator, using finite element modelling for the small elastic deflections but allowing large rigid body motion and high velocities [29]. This results in a large number of equations that are reduced using Guyan model reduction.

In recent years a fundamental limitation of linear finite element formulations of flexible deformations in flexible multibody simulation programs such as TREETOPS[31], DISCOS[3], etc. has been pointed out [23,24,27]. This limitation could be characterized as a premature linearization of velocity expressions that is implicit in a linear finite element or modal formulation of the motion equations for flexible bodies [24]. Kane et al. [11] demonstrated this flaw of the traditional approach numerically by simulating a simple system consisting of a flexible beam attached to a rigid base spinning in the plane. This simulation yielded the surprising and intuitively wrong result of the beam diverging during a spin up maneuver [23]. Probably because of this simple example the "prematurely linearized" equations of motion are said to lack the "spin-stiffening effect."

The purpose of this thesis is to provide the equations of motion of a flexible, two-link, planar manipulator, correctly linearized in small elastic deflection and speeds, but fully nonlinear in rigid body motions. The links are modelled as Bernoulli-Euler beams and the elastic deflections are expanded in terms of cantilever (clamped-free) modes. This results in a small number of equations that can be derived analytically. The equations of motion are desired to provide the capability to determine the dynamic response of a two-link manipulator to desired forcing functions for control experiments purposes. It is further desired that the developed simulation allow for the use of a variable number of modes in order to test effects such as spillover when reduced-order controllers are tested.

In view of the limitation of certain linear finite element multibody programs, it is further desired to investigate the relative significance of "elastic" nonlinear terms in the equations of motion. These are nonlinear terms that involve the generalized elastic coordinates and their time derivatives. For this purpose, the simulation is developed with the capability of ignoring select elastic nonlinear terms, or all elastic nonlinear terms.

THESIS OVERVIEW

Chapter two presents the derivation of the linearized equations of motion for the two-link manipulator. Using Kane's formalism [9] the correctly linearized equations are obtained through the proper use of nonlinear strain-displacement relations.

In chapter three, a dynamic analysis of a simplified single beam example is used to investigate the relative significance of elastic nonlinear terms. Three levels of increasing simplification of the linearized motion equations are proposed for this investigation.

The results of the previous chapter are compared to the results of numerical simulation of the two-link manipulator in chapter four. Four different open loop trajectories are investigated and conclusions about the dynamic modelling and relative significance of nonlinear terms are drawn.

CHAPTER 2: DYNAMICS

INTRODUCTION

The equations of motion of a planar, flexible, two-link manipulator, linearized in small elastic coordinates and speeds, are developed in this chapter. After describing the system model and stating the assumptions made, the system kinematics are presented. In section 2.2, the kinematics of the system are derived without making any assumptions as to the size of generalized elastic coordinates. It is in section 2.3 that the kinematic expressions are linearized assuming small elastic coordinates and speeds. In the last section of this chapter the dynamic equations for the nonlinear system, correct to first order in elastic coordinates and speeds, are derived using Kane's formalism.

2.1 SYSTEM MODEL

The dynamics of a planar, two-link, flexible arm with revolute joints, composed of three rigid bodies connected by two slender uniform beams (see Fig. 2.1) are to be investigated. The equations of motion for this system are derived using Kane's formalism [9] together with second order strain-displacement relations. The equations are thus exact to first order in the beams' elastic generalized coordinates and speeds, but fully nonlinear in rigid body coordinates and rates. The normal to the plane of motion is parallel to the local acceleration of gravity, so gravity can be ignored. Independent axial extensions of the beams are ignored (i.e., the axial strain at the neutral axis is assumed zero for each beam). The elbow joint is actually modelled as two bodies: one attached to link 1 in a cantilevered way and the other, free to rotate with respect to the first, with link 2 attached to it in a cantilevered way also. Both elbow bodies are rigid and share the hinge point but their mass centers are allowed to be offset from the hinge point. One percent modal damping is added to the model to represent material damping. Actuation is assumed in the form of shoulder and elbow torques. The equations of motion for this system are available in Appendix C.

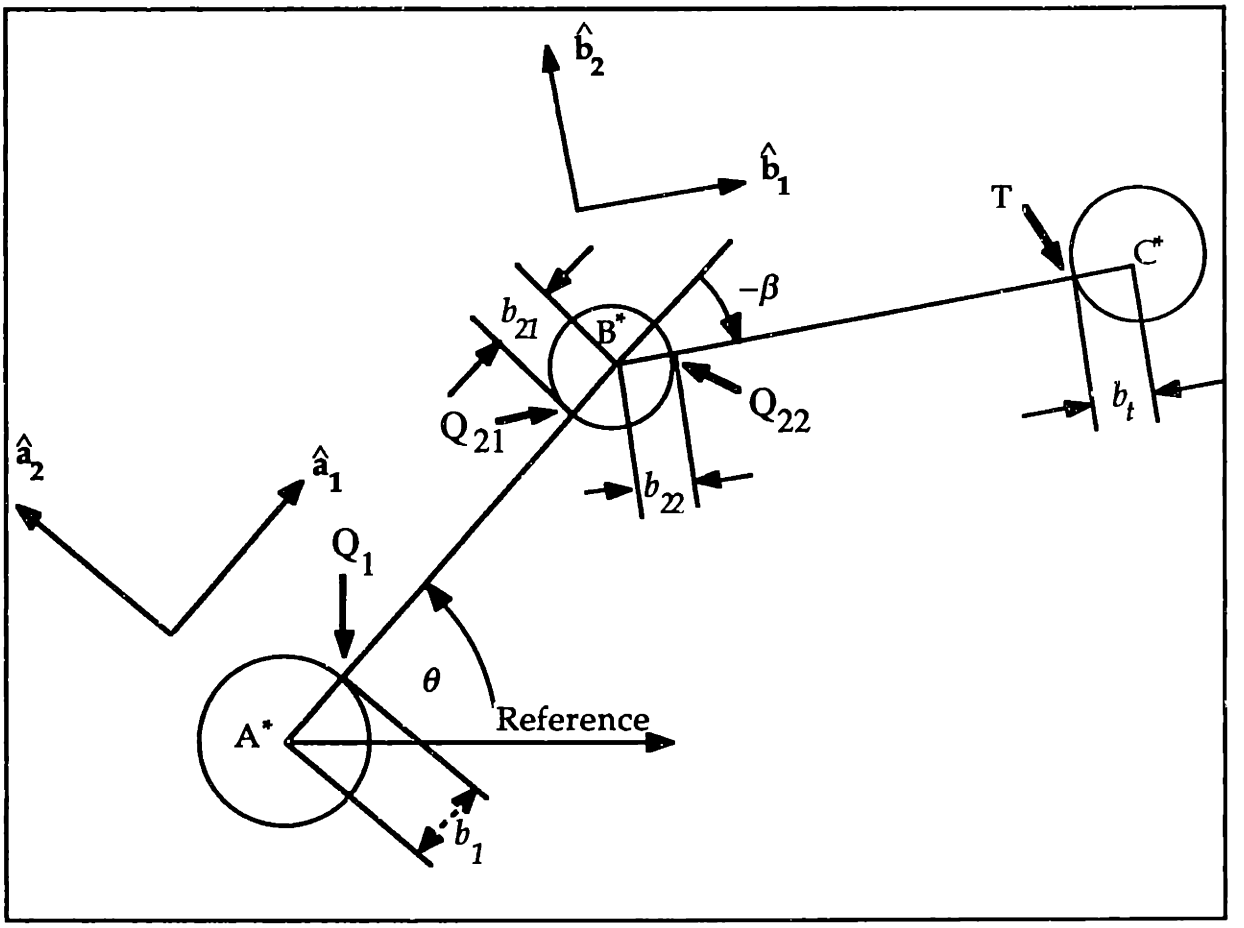


Figure 2.1: Schematic of the Two-link Flexible Manipulator

The rigid body generalized coordinates, the shoulder angle, θ, and the relative elbow angle, β, are defined in Figure 2.1. In Figure 2.2 it is anticipated that the transverse deflection of the inboard link, u_2, will be expanded in a series of assumed mode shapes. The time-dependent coefficients of these assumed modes, q_i (i=1,...,n), are the generalized elastic coordinates for the beam. An analogous definition for the outboard link results in generalized elastic coordinates p_j (j=1,...,m). This set of 2+n+m generalized coordinates uniquely specifies the configuration of the two-link manipulator.

2.1.1 List of Assumptions

The following is a list of the assumptions made in the derivation of the linearized equations of motion presented in the rest of this chapter.

1. The links are modelled as nominally straight, uniform beams, with symmetric cross-sections about axes $\mathbf{a}_2$ and $\mathbf{a}_3$ ($\mathbf{b}_2$ and $\mathbf{b}_3$) when undeformed.

2. The cross-sectional dimensions of the beams are much smaller than the lengths, so that rotary inertia and shear effects are negligible [18].

3. Torsion is ignored.

4. No relative motion is allowed between links and points of attachment to the rigid bodies.

5. Friction, damping, and flexibility have been assumed negligible at the joints.

6. Actuators are assumed to apply torques. Other actuator dynamics are not taken into consideration.

7. All rigid bodies are modelled as a lumped mass, situated at the mass center, and lumped moment of inertia, but the mass centers can be offset from the point of attachment of the links to account for planar dimensions of the same bodies.

8. The shoulder body rotates in inertial frame, but undergoes no translation.

9. The elbow joint is made up of two bodies that share the hinge point, but have mass centers that can be offset from the hinge point (see Fig. 2.3).

10. The relative elbow angle, β, is measured between the tangent to the neutral axis of link two (outboard) at its point of attachment to the elbow (y=0), and the tangent to the neutral axis of link one (inboard) at the end of link one attached to the elbow (i.e., $x=l_1$).

11. Elastic deformation coordinates and rates for both links are assumed small enough so that second order terms can be ignored.

2.2 SYSTEM KINEMATICS

2.2.1 Shoulder Body

Following Kane [9,10], an embedded reference frame is associated with each rigid body (see Fig. 2.1). Frame ($\mathbf{a}_1$,$\mathbf{a}_2$,$\mathbf{a}_3$) is associated with the shoulder body, denoted as body A, and its origin coincides with the body's center of mass. The $\mathbf{a}_1$ axis extends along the line $\mathbf{b}_1$ and thus along the neutral axis of the undeformed link 1. Axis $\mathbf{a}_2$ is perpendicular to this one in the plane of motion, and $\mathbf{a}_3$ is such that A is a right-handed, Cartesian coordinate system. Body A is free to rotate in the plane but pinned at its mass center A*, so that frame A can only rotate and undergoes no translation. The motion of A in the inertial frame N ($\mathbf{n}_1$,$\mathbf{n}_2$,$\mathbf{n}_3$) is pure rotation, where the angular velocity of A in N is given by

$$^N\underline{\omega}^A = \omega_3 \hat{\mathbf{a}}_3 \tag{1}$$

and ω_3 uniquely defines the motion of body A. This quantity is termed a generalized speed in Kane's formalism.

2.2.2 Inboard Link

Following [11], the motion of link 1 in the inertial reference frame N is described by introducing a rigid differential element P_1 of link 1, with centroid C_1 and elastic center E_1 located at a distance x from Q_1, when the link is undeformed (see Fig. 2.2). No relative motion is allowed between body A and link 1 at Q_1, i.e., link 1 is attached to A in a cantilevered way. The position of point E_1 (which coincides with C_1 under the stated assumptions) in the frame A after deformation is found by applying the translations

$$u_1 \hat{\mathbf{a}}_1 \quad \text{and} \quad u_2 \hat{\mathbf{a}}_2.$$

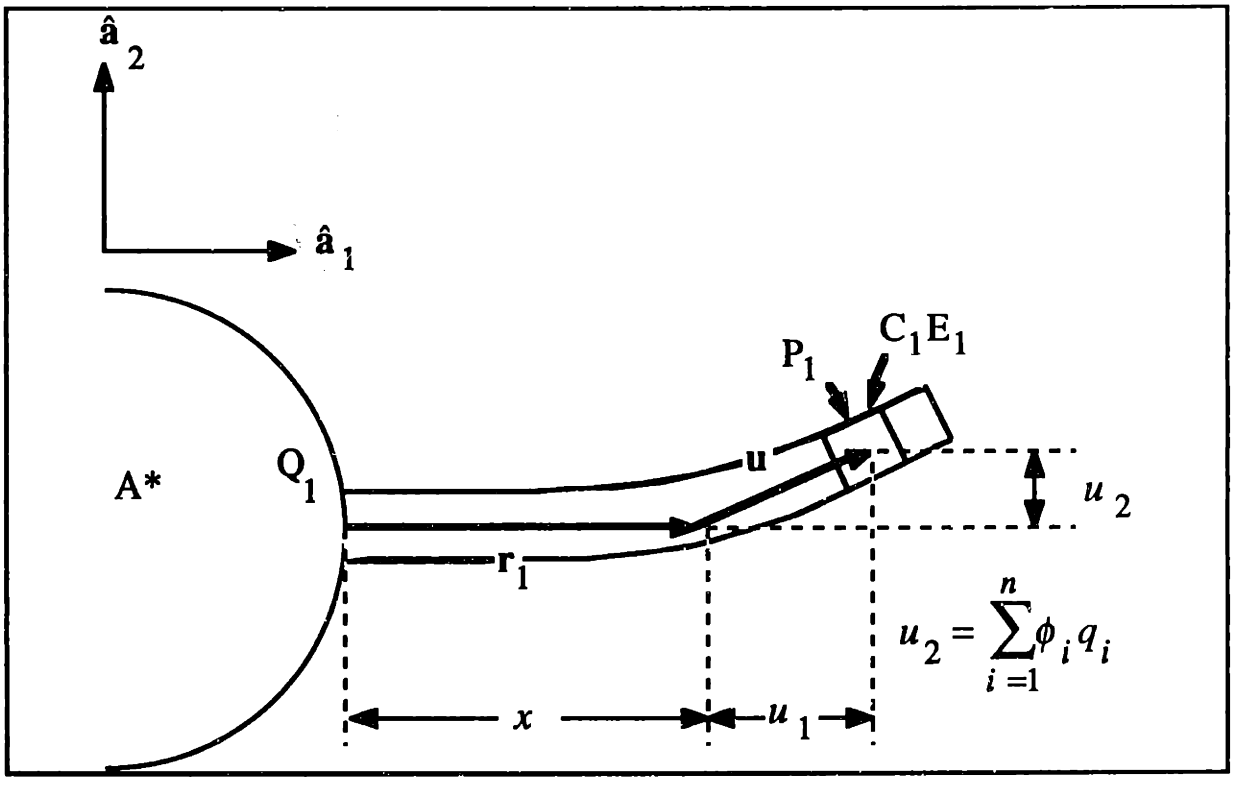


Figure 2.2: Definitions for Link Kinematics

Defining $\mathbf{r}_1$ and $\mathbf{u}$ as

$$\mathbf{r}_1 \equiv x\hat{\mathbf{a}}_1$$

$$\mathbf{u}(x,t) \equiv u_1(x,t)\hat{\mathbf{a}}_1 + u_2(x,t)\hat{\mathbf{a}}_2 \tag{1}$$

the position vector from Q_1 to E_1 (see Fig. 2.2) can be expressed as

$$\mathbf{p}^{Q_1E_1} = \mathbf{r}_1 + \mathbf{u}(x,t). \tag{2}$$

Further, since

$$\mathbf{p}^{E_1C_1} = 0,$$

$$\mathbf{p}^{Q_1C_1} = \mathbf{p}^{Q_1E_1} = \mathbf{r}_1 + \mathbf{u}(x,t) = (x + u_1)\hat{\mathbf{a}}_1 + u_2\hat{\mathbf{a}}_2. \tag{3}$$

Finally, the position vector for C_1 in frame A is obtained:

$$\mathbf{p}^{A^*C_1} = \mathbf{p}^{A^*Q_1} + \mathbf{p}^{Q_1C_1} = (b_1 + x + u_1)\hat{\mathbf{a}}_1 + u_2\hat{\mathbf{a}}_2. \tag{4}$$

Now it is possible to write down the velocities of Q_1 and P_1 in N:

$$\begin{aligned} {}^N\mathbf{v}^{Q_1} &= {}^N\mathbf{v}^{A^*} + {}^N\underline{\omega}^A \times \mathbf{p}^{A^*Q_1} + {}^A\mathbf{v}^{Q_1} \\ &= b_1\omega_3\hat{\mathbf{a}}_2 \\ {}^N\mathbf{v}^{C_1} &= {}^N\mathbf{v}^{A^*} + {}^N\underline{\omega}^A \times \mathbf{p}^{A^*C_1} + {}^A\mathbf{v}^{C_1} \\ &= (\dot{u}_1 - \omega_3 u_2)\hat{\mathbf{a}}_1 + [\dot{u}_2 + \omega_3(b_1 + x + u_1)]\hat{\mathbf{a}}_2 \end{aligned} \tag{5}$$

where

$${}^N\mathbf{v}^{A^*} = {}^A\mathbf{v}^{Q_1} = 0. \tag{6}$$

2.2.3 Elbow Joint

As mentioned in section 2.1, it is assumed that the elbow consists of two rigid bodies (see Fig. 2.3). Body B_1 is attached to link 1 at point Q_{21}, while body B_2 is connected to B_1 at the hinge point H. No relative motion is allowed between link 1 and body B_1 at Q_{21}. The mass center of B_1, B_1^*, can be offset from the hinge point. This

configuration of elbow bodies was chosen to allow the simulation maximum flexibility, as it is easier to align hardware hinge points than to place mass centers.

To characterize the motion of B_1 in N, it is convenient to first obtain the velocity of H in N:

$$ {}^N\mathbf{v}^H = {}^N\mathbf{v}^{A^*} + {}^N\underline{\omega}^A \times \mathbf{p}^{A^*H} + {}^A\mathbf{v}^H . \tag{1} $$

To be able to write down the distance from A* to H, note that the angle between the normal to the cross-section of the beam at x after deformation, and the neutral axis before deformation, is, for a Bernoulli-Euler beam (where distortion due to shear is ignored-see [18])

$$ \frac{\partial}{\partial x} u_2(x, t) $$

and defining

$$ \alpha_1 = \left. \frac{\partial u_2}{\partial x} \right|_{x = l_1}, \tag{2} $$

the position vector results:

$$ \mathbf{p}^{A^*H} = [b_1 + l_1 + u_1(l_1, t) + b_{21}\cos(\alpha_1)]\hat{\mathbf{a}}_1 + [u_2(l_1, t) + b_{21}\sin(\alpha_1)]\hat{\mathbf{a}}_2 . \tag{3} $$

This yields for the desired hinge point velocity vector

$$ \begin{aligned} {}^N\mathbf{v}^H = {} & \{\dot{u}_1(l_1, t) - b_{21}\dot{\alpha}_1 \sin \alpha_1 - \omega_3[u_2(l_1, t) + b_{21}\sin \alpha_1]\}\hat{\mathbf{a}}_1 \\ & + \{\dot{u}_2(l_1, t) + b_{21}\dot{\alpha}_1 \cos \alpha_1 + \omega_3[b_1 + l_1 + u_1(l_1, t) + b_{21}\cos \alpha_1]\}\hat{\mathbf{a}}_2 \end{aligned} \tag{4} $$

where the following convention has been used:

$$ \dot{\alpha}_1 = \frac{\partial}{\partial t}\left[\left. \frac{\partial u_2}{\partial x} \right|_{x = l_1} \right] . \tag{5} $$

The angular velocity of B_1 is then

$$ {}^N\underline{\omega}^{B_1} = (\dot{\alpha}_1 + \omega_3)\hat{\mathbf{a}}_3 \tag{6} $$

and it is possible to write the velocity of B_1^* in N as

$$
\begin{aligned}
{}^N\mathbf{v}^{B_1^*} &= {}^N\mathbf{v}^H + {}^N\underline{\omega}^{B_1} \times \mathbf{p}^{HB_1^*} + {}^{B_1}\mathbf{v}^{B_1^*} \\
&= {}^N\mathbf{v}^H - e_1(\dot{\alpha}_1 + \omega_3)\sin(\gamma_1 + \alpha_1)\hat{\mathbf{a}}_1 + e_1(\dot{\alpha}_1 + \omega_3)\cos(\gamma_1 + \alpha_1)\hat{\mathbf{a}}_2
\end{aligned} \tag{7}
$$

where the following expression has been used for the distance from the hinge point to the mass center of B_1

$$
\begin{aligned}
\mathbf{p}^{HB_1^*} &= e_1\cos(\gamma_1 + \alpha_1)\hat{\mathbf{a}}_1 + e_1\sin(\gamma_1 + \alpha_1)\hat{\mathbf{a}}_2 \\
{}^{B_1}\mathbf{v}^{B_1^*} &= 0.
\end{aligned} \tag{8}
$$

The velocity of point Q_{21} in body B_1 will be needed in the determination of the generalized active forces (see section 2.4.2), and so it is written down here for the sake of completeness:

$$
\begin{aligned}
{}^N\mathbf{v}^{Q_{21}} &= {}^N\mathbf{v}^{C_1}\Big|_{x=l_1} \\
&= [\dot{u}_1(l_1, t) - \omega_3 u_2(l_1, t)]\hat{\mathbf{a}}_1 + \{\dot{u}_2(l_1, t) + \omega_3[b_1 + l_1 + u_1(l_1, t)]\}\hat{\mathbf{a}}_2.
\end{aligned} \tag{9}
$$

The second elbow body, B_2, is free to rotate with respect to body B_1 about the hinge point H. Frame B ($\mathbf{b}_1$,$\mathbf{b}_2$,$\mathbf{b}_3$) is associated with body B_2, and the origin of the coordinate frame coincides with the hinge point. The axis $\mathbf{b}_1$ lies along line b_{22}, that is, also along the neutral axis of the undeformed link 2; $\mathbf{b}_2$ is perpendicular to $\mathbf{b}_1$ in the plane of motion; $\mathbf{b}_3$ completes a right-handed rectangular coordinate system. As in the case of B_1, B_2^* can be offset from the hinge point. The rotational motion of B_2 in N is characterized by

$$
\Omega_3 = \dot{\beta} \tag{10}
$$

where β is the relative angle between bodies B_1 and B_2, and the angular velocity of B_2 in the inertial frame is given by:

$$
{}^N\underline{\omega}^B = {}^N\underline{\omega}^{B_2} = (\omega_3 + \dot{\alpha}_1 + \Omega_3)\hat{\mathbf{a}}_3. \tag{11}
$$

The velocity of B_2^* in N can be written down immediately as

$$
\begin{aligned}
{}^{N}\mathbf{v}^{B_2^*} &= {}^{N}\mathbf{v}^{H} + {}^{N}\underline{\omega}^{B_2} \times \mathbf{p}^{HB_2^*} + {}^{B}\mathbf{v}^{B_2^*} \\
&= {}^{N}\mathbf{v}^{H} - e_2\, {}^{N}\omega^{B_2} \sin\gamma_2 \hat{\mathbf{b}}_1 + e_2\, {}^{N}\omega^{B_2} \cos\gamma_2 \hat{\mathbf{b}}_2
\end{aligned} \tag{12}
$$

where the following facts have been used:

$$
\begin{aligned}
\mathbf{p}^{HB_2^*} &= e_2 \cos\gamma_2 \hat{\mathbf{b}}_1 + e_2 \sin\gamma_2 \hat{\mathbf{b}}_2 \\
{}^{B}\mathbf{v}^{B_2^*} &= 0
\end{aligned} \tag{13}
$$

and the following definition has been made:

$$
{}^{N}\omega^{B_2} = \omega_3 + \dot{\alpha}_1 + \Omega_3. \tag{14}
$$

2.2.4 Outboard Link

Using a similar approach to that of section 2.2.2, the motion of link 2 in N is described by introducing a rigid differential element P_2 of link 2, with centroid C_2 and elastic center E_2, located at a distance y from point Q_{22} when the link is undeformed. Again relative motion is disallowed between body B_2 and link 2 at the point Q_{22} (see Fig. 2.1). Defining $\mathbf{r}_2$ and $\mathbf{v}$ as

$$
\begin{aligned}
\mathbf{r}_2 &\equiv y\, \hat{\mathbf{b}}_1 \\
\mathbf{v} &\equiv v_1(y,t)\hat{\mathbf{b}}_1 + v_2(y,t)\hat{\mathbf{b}}_2,
\end{aligned} \tag{1}
$$

and following the set of assumptions described in section 2.1, the position vector from H to C_2 results:

$$
\begin{aligned}
\mathbf{p}^{E_2C_2} &= 0 \\
\mathbf{p}^{Q_{22}C_2} &= \mathbf{p}^{Q_{22}E_2} = \mathbf{r}_2 + \mathbf{v}(y,t) = (y + v_1)\hat{\mathbf{b}}_1 + v_2\hat{\mathbf{b}}_2 \\
\mathbf{p}^{HC_2} &= \mathbf{p}^{HQ_{22}} + \mathbf{p}^{Q_{22}C_2} = (b_{22} + y + v_1)\hat{\mathbf{b}}_1 + v_2\hat{\mathbf{b}}_2.
\end{aligned} \tag{2}
$$

In equation (2), $\mathbf{v}$ is the vector that describes the change in position of E_2 due to the deformation.

It is now possible to write the velocities of Q_{22} and P_2 in the inertial reference frame:

$$
\begin{aligned}
{}^N\mathbf{v}^{Q_{22}} &= {}^N\mathbf{v}^H + {}^N\underline{\omega}^B \times \mathbf{p}^{HQ_{22}} + {}^B\mathbf{v}^{Q_{22}} \\
&= {}^N\mathbf{v}^H + {}^N\omega^{B_2} b_{22}\hat{\mathbf{b}}_2 \\
{}^N\mathbf{v}^{C_2} &= {}^N\mathbf{v}^H + {}^N\underline{\omega}^{B_2} \times \mathbf{p}^{HC_2} + {}^B\mathbf{v}^{C_2} \\
&= {}^N\mathbf{v}^H + (\dot{v}_1 - {}^N\omega^{B_2} v_2)\hat{\mathbf{b}}_1 + [\dot{v}_2 + {}^N\omega^{B_2}(b_{22} + y + v_1)]\hat{\mathbf{b}}_2
\end{aligned} \tag{3}
$$

where

$$
\begin{aligned}
\mathbf{p}^{HQ_{22}} &= b_{22}\hat{\mathbf{b}}_1 \\
{}^B\mathbf{v}^{C_2} &= \frac{d}{dt}\Big)_B \mathbf{p}^{HC_2}.
\end{aligned} \tag{4}
$$

2.2.5 Tip Mass

Finally, the kinematics of the "end effector," or tip body, are determined. Again it is assumed that the tip rigid body C is attached rigidly to link 2 at point T. As in the case of link 1, the angle between the normal to the cross-section of the beam at y after deformation and the neutral axis before deformation is given by:

$$\frac{\partial}{\partial y} v_2(y, t).$$

Note that as before, the above equation is true due to the assumption of Bernoulli-Euler beams. Defining

$$\alpha_2 = \left.\frac{\partial v_2}{\partial y}\right|_{y = l_2}, \tag{1}$$

the distance from H to C* is:

$$\mathbf{p}^{HC^*} = [b_{22} + l_2 + v_1(l_2, t) + b_t \cos(\alpha_2)]\hat{\mathbf{b}}_1 + [v_2(l_2, t) + b_t \sin(\alpha_2)]\hat{\mathbf{b}}_2. \tag{2}$$

Then the velocity in the inertial reference frame of the mass center C* of body C is given by:

$$
\begin{aligned}
{}^N\mathbf{v}^{C*} &= {}^N\mathbf{v}^H + {}^N\underline{\omega}^B \times \mathbf{p}^{HC*} + {}^B\mathbf{v}^{C*} \\
&= {}^N\mathbf{v}^H + \{\dot{v}_1(l_2, t) - b_t\dot{\alpha}_2 \sin\alpha_2 - {}^N\omega^{B2}[v_2(l_2,t) + b_t \sin\alpha_2]\}\hat{\mathbf{b}}_1 \\
&+ \{\dot{v}_2(l_2, t) + b_t\dot{\alpha}_2\cos\alpha_2 + {}^N\omega^{B2}[b_{22} + l_2 + v_1(l_2, t) + b_t\cos\alpha_2]\}\hat{\mathbf{b}}_2
\end{aligned} \tag{3}
$$

where

$$
{}^B\mathbf{v}^{C*} = \frac{d}{dt}\Big)_B \mathbf{p}^{HC*}. \tag{4}
$$

The angular velocity of C in the inertial frame is

$$
{}^N\underline{\omega}^C = (\omega_3 + \dot{\alpha}_1 + \Omega_3 + \dot{\alpha}_2)\hat{\mathbf{a}}_3. \tag{5}
$$

The velocity of point T in N is

$$
\begin{aligned}
{}^N\mathbf{v}^T &= {}^N\mathbf{v}^{C_2}\Big|_{y=l_2} \\
&= {}^N\mathbf{v}^H + [\dot{v}_1(l_2,t) - {}^N\omega^{B2}v_2(l_2, t)]\hat{\mathbf{b}}_1 \\
&+ \{\dot{v}_2(l_2, t) + {}^N\omega^{B2}[b_{22} + l_2 + v_1(l_2, t)]\}\hat{\mathbf{b}}_2.
\end{aligned} \tag{6}
$$

Together with the relation

$$
\omega_3 = \dot{\theta}, \tag{7}
$$

the above completes the kinematic description of the model. The transformation matrix to go from frame B ($\mathbf{b}_1$,$\mathbf{b}_2$,$\mathbf{b}_3$) to frame A ($\mathbf{a}_1$,$\mathbf{a}_2$,$\mathbf{a}_3$) is

$$
{}^A\mathbf{T}^B = \begin{bmatrix} \cos(\alpha_1 + \beta) & -\sin(\alpha_1 + \beta) & 0 \\ \sin(\alpha_1 + \beta) & \cos(\alpha_1 + \beta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}
$$

2.3 LINEARIZED DYNAMICS - ASSUMED-MODES METHOD

Following the development outlined in section A.1 (see Appendix A), a solution to the free vibration problem is assumed in the form of a series composed of a linear combination of admissible functions of the spatial coordinates, multiplied by time-dependent generalized coordinates [19]. This is equivalent to treating the continuous system as a discrete many-degrees-of-freedom system and will result in the motion equations being ordinary differential equations instead of partial differential equations. As pointed out in [19], the assumed-modes method is well suited to the case of systems with both distributed and discrete properties.

For the system model under consideration, and taking into account the assumptions stated at the beginning of the chapter, independent axial extensions of both links are ignored, as in the example of section A.2. The correct condition for this assumption is

$$\hat{\varepsilon}_{xx} = 0 \tag{1}$$

This results in the following expressions for the axial displacements of both link 1 and 2 in frames A and B respectively:

$$\begin{aligned} u_1(x,t) &= -\int_0^x \frac{1}{2}\left(\frac{\partial u_2}{\partial \sigma}\right)^2 d\sigma \\ v_1(y,t) &= -\int_0^y \frac{1}{2}\left(\frac{\partial v_2}{\partial \sigma}\right)^2 d\sigma \end{aligned} \tag{2}$$

which are correct to second order in the transverse displacement gradients.

Assuming now the following solutions for the transverse displacements of both links in frames A and B respectively

$$\begin{aligned} u_2(x,t) &= \sum_{i=1}^{n} \phi_i(x) q_i(t) \\ v_2(y,t) &= \sum_{i=1}^{m} \psi_i(y) p_i(t) \end{aligned} \tag{3}$$

expressions for u_1 and v_1 correct to second order in the generalized elastic coordinates q_i, p_i are obtained

$$u_1(x,t) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\int_0^x \phi_i'(\sigma)\phi_j'(\sigma)d\sigma\, q_i\, q_j$$

$$v_1(y,t) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\int_0^y \psi_i'(\sigma)\psi_j'(\sigma)d\sigma\, p_i\, p_j \tag{4}$$

Note that the hybrid continuous and discrete system has been transformed into a 2+n+m degrees of freedom system. The admissible functions $\phi(x)$ and $\psi(y)$ are chosen to be cantilevered (clamped-free) mode shapes for uniform beams of lengths l_1 and l_2 respectively. These are given explicitly as the following functions of spatial coordinates [2]:

$$\tilde{y}_i(s) = \cosh\left(\frac{\lambda_i s}{l_j}\right) - \cos\left(\frac{\lambda_i s}{l_j}\right) - \sigma_i\left(\sinh\left(\frac{\lambda_i s}{l_j}\right) - \sin\left(\frac{\lambda_i s}{l_j}\right)\right)$$

$$\tilde{y}_i = \begin{cases} \phi_i \\ \psi_i \end{cases}, \quad s = \begin{cases} x \\ y \end{cases}, \quad i = \begin{cases} 1,..,n & , j = 1 \\ 1,..,m & , j = 2 \end{cases} \tag{5}$$

where

$$\sigma_i = \frac{\sinh \lambda_i - \sin \lambda_i}{\cosh \lambda_i + \cos \lambda_i} \tag{6}$$

and λ_i are the solutions, arranged in increasing order, of the transcendental equation:

$$\cos \lambda \cosh \lambda + 1 = 0. \tag{7}$$

These mode shapes satisfy the following orthogonality conditions

$$\int_0^l \tilde{y}_i(s)\,\tilde{y}_j(s)ds = \int_0^l \tilde{y}''_i(s)\,\tilde{y}''_j(s)ds = \begin{cases} 0, & i \neq j \\ l, & i = j \end{cases} \tag{8}$$

where

$$\tilde{y}''_i(s) = \left(\frac{l}{\lambda_i}\right)^2 \frac{d^2\tilde{y}_i}{ds^2}$$

$$\frac{d^n}{d\left(\frac{\lambda_i s}{l}\right)^n}\tilde{y}_i = \frac{d^{n+4}}{d\left(\frac{\lambda_i s}{l}\right)^{n+4}}\tilde{y}_i. \tag{9}$$

Requiring now that the elastic generalized coordinates, q_i and p_j, and the elastic generalized speeds, dq_i/dt and dp_j/dt (i=1,...,n; j=1,...,m), be infinitesimally small, it should be possible to obtain equations of motion for the system model under consideration that are linear in these elastic coordinates and speeds. Heeding the lessons of Appendix A, the expressions for velocities and angular velocities obtained in section 2.2 are not linearized until <u>after</u> the formation of partial velocities and partial angular velocities but previous to the formation of the accelerations (see section 2.4). The velocities and angular velocities linearized in the elastic coordinates and speeds are presented below.

In what follows, a ~ represents a quantity that has been linearized with respect to the small elastic deflections and speeds. The following definitions will be used throughout the next two sub-sections to simplify the equations.

$$u_1(x,t) = \dot{u}_1(x,t) = 0 \qquad v_1(y,t) = \dot{v}_1(y,t) = 0$$

$$u_2(x,t) = \sum_{i=1}^{n}\phi_i(x)q_i(t) \qquad v_2(y,t) = \sum_{i=1}^{m}\psi_i(y)p_i(t)$$

$$\dot{u}_2(x,t) = \sum_{i=1}^{n}\phi_i(x)\dot{q}_i(t) \qquad \dot{v}_2(y,t) = \sum_{i=1}^{m}\psi_i(y)\dot{p}_i(t)$$

$$\alpha_1(t) = \sum_{i=1}^{n}\phi'_i(l_1)q_i(t) \qquad \alpha_2(t) = \sum_{i=1}^{m}\psi'_i(l_2)p_i(t)$$

$$\dot{\alpha}_1(t) = \sum_{i=1}^{n}\phi'_i(l_1)\dot{q}_i(t) \qquad \dot{\alpha}_2(t) = \sum_{i=1}^{m}\psi'_i(l_2)\dot{p}_i(t)$$

$$L_1 = b_1 + l_1 + b_{21} \qquad L_2 = b_{22} + l_2 + b_t$$

$$\tilde{c}_{\gamma_1} = \cos\gamma_1 - \sin\gamma_1 \cdot \alpha_1 \qquad \tilde{s}_{\gamma_1} = \sin\gamma_1 + \cos\gamma_1 \cdot \alpha_1$$

$$^{N}\vec{\omega}^{B_2} = \omega_3 + \Omega_3$$

$$\tilde{c}_{\gamma_2} = \cos(\gamma_2 + \beta) - \sin(\gamma_2 + \beta) \cdot \alpha_1 \qquad \tilde{s}_{\gamma_2} = \sin(\gamma_2 + \beta) + \cos(\gamma_2 + \beta) \cdot \alpha_1$$

$$\tilde{c}_{\beta} = \cos\beta - \sin\beta \cdot \alpha_1 \qquad \tilde{s}_{\beta} = \sin\beta + \cos\beta \cdot \alpha_1 \tag{10}$$

2.3.1 Linearized Velocities

$$^{N}\vec{v}^{Q} = b_1\omega_3\hat{a}_2$$

$$^{N}\vec{v}^{C_1} = -\omega_3 u_2\hat{a}_1 + [\dot{u}_2 + \omega_3(b_1 + x)]\hat{a}_2$$

$$^{N}\vec{v}^{H} = [-\omega_3(u_2(l_1, t) + b_{21}\alpha_1)]\hat{a}_1 + [\dot{u}_2(l_1) + b_{21}\dot{\alpha}_1 + \omega_3 L_1]\hat{a}_2 \tag{1}$$

$$^{N}\vec{v}^{B_1^*} = {}^{N}\vec{v}^{H} - [e_1\dot{\alpha}_1\sin\gamma_1 + e_1\omega_3\tilde{s}_{\gamma_1}]\hat{a}_1 + [e_1\dot{\alpha}_1\cos\gamma_1 + e_1\omega_3\tilde{c}_{\gamma_1}]\hat{a}_2$$

$$^{N}\vec{v}^{Q_{21}} = -\omega_3 u_2(l_1, t)\hat{a}_1 + [\dot{u}_2(l_1, t) + \omega_3(b_1 + l_1)]\hat{a}_2$$

$$^{N}\vec{v}^{B_2^*} = {}^{N}\vec{v}^{H} - \left[{}^{N}\vec{\omega}^{B_2} e_2\tilde{s}_{\gamma_2} + \dot{\alpha}_1 e_2\sin(\beta + \gamma_2)\right]\hat{a}_1$$

$$+ \left[{}^{N}\vec{\omega}^{B_2} e_2\tilde{c}_{\gamma_2} + \dot{\alpha}_1 e_2\cos(\beta + \gamma_2)\right]\hat{a}_2 \tag{2}$$

$$^{N}\vec{v}^{Q_{22}} = {}^{N}\vec{v}^{H} + b_{22}\left\{\left[-{}^{N}\vec{\omega}^{B_2}\tilde{s}_{\beta} - \dot{\alpha}_1\sin\beta\right]\hat{a}_1 + \left[{}^{N}\vec{\omega}^{B_2}\tilde{c}_{\beta} + \dot{\alpha}_1\cos\beta\right]\hat{a}_2\right\}$$

$$^{N}\vec{v}^{C_2} = {}^{N}\vec{v}^{H} - \left\{{}^{N}\vec{\omega}^{B_2}\left[v_2\cos\beta + (b_{22} + y)\tilde{s}_{\beta}\right] + \left[\dot{v}_2 + \dot{\alpha}_1(b_{22} + y)\right]\sin\beta\right\}\hat{a}_1$$

$$+ \left\{{}^{N}\vec{\omega}^{B_2}\left[-v_2\sin\beta + (b_{22} + y)\tilde{c}_{\beta}\right] + \left[\dot{v}_2 + \dot{\alpha}_1(b_{22} + y)\right]\cos\beta\right\}\hat{a}_2 \tag{3}$$

$$^{N}\vec{v}^{C^*} = {}^{N}\vec{v}^{H}$$

$$-\left\{{}^{N}\vec{\omega}^{B_2}\left[(v_2(l_2) + b_t\alpha_2)\cos\beta + L_2\tilde{s}_{\beta}\right] + \left[\dot{v}_2(l_2) + \dot{\alpha}_1 L_2 + \dot{\alpha}_2 b_t\right]\sin\beta\right\}\hat{a}_1$$

$$+\left\{{}^{N}\vec{\omega}^{B_2}\left[-(v_2(l_2) + b_t\alpha_2)\sin\beta + L_2\tilde{c}_{\beta}\right] + \left[\dot{v}_2(l_2) + \dot{\alpha}_1 L_2 + \dot{\alpha}_2 b_t\right]\cos\beta\right\}\hat{a}_2 \tag{4}$$

$$
\begin{aligned}
{}^{N}\tilde{\mathbf{v}}^{T} &= {}^{N}\tilde{\mathbf{v}}^{H} \\
&- \left\{ {}^{N}\tilde{\omega}^{B} \left[v_2(l_2)\cos\beta + (b_{22} + l_2)\tilde{s}_\beta \right] + \left[\dot{v}_2(l_2) + \dot{\alpha}_1(b_{22} + l_2) \right] \sin\beta \right\} \hat{\mathbf{a}}_1 \\
&+ \left\{ {}^{N}\tilde{\omega}^{B} \left[-v_2(l_2)\sin\beta + (b_{22} + l_2)\tilde{c}_\beta \right] + \left[\dot{v}_2(l_2) + \dot{\alpha}_1(b_{22} + l_2) \right] \cos\beta \right\} \hat{\mathbf{a}}_2
\end{aligned} \tag{5}
$$

2.3.2 Linearized Angular Velocities

$$
\begin{aligned}
{}^{N}\tilde{\Omega}^{A} &= \omega_3 \hat{\mathbf{a}}_3 \\
{}^{N}\tilde{\Omega}^{B_1} &= (\omega_3 + \dot{\alpha}_1)\hat{\mathbf{a}}_3 \\
{}^{N}\tilde{\Omega}^{B_2} &= (\omega_3 + \dot{\alpha}_1 + \Omega_3)\hat{\mathbf{a}}_3 \\
{}^{N}\tilde{\Omega}^{C} &= (\omega_3 + \dot{\alpha}_1 + \Omega_3 + \dot{\alpha}_2)\hat{\mathbf{a}}_3
\end{aligned} \tag{1}
$$

2.4 KANE'S DYNAMICAL EQUATIONS

In this section, Kane's dynamical equations are used to assemble the equations of motion for the system model, linearized in small generalized elastic coordinates and speeds.

According to the Kane formalism, the dynamic equations are given by

$$
\begin{aligned}
\tilde{F}_r^* + \tilde{F}_r &= 0 \\
r = \omega_3, \Omega_3, \dot{q}_i, \dot{p}_j \qquad & (i = 1,\ldots, n; j = 1,\ldots, m)
\end{aligned} \tag{1}
$$

where F_r^* is the generalized inertia force corresponding to the generalized speed r, while F_r is the associated generalized active force. Note that as in the previous section, a ~ signifies that the indicated quantity has been linearized with respect to the small elastic deflections and speeds.

For a collection of rigid bodies, the generalized inertia force is equal to the sum over all the bodies of the d'Alembert force dot multiplied by the partial velocity of each body's mass center with respect to a particular generalized speed, plus the inertial time derivative of the bodies' angular momenta, about their respective mass centers, dot multiplied by the corresponding partial angular velocity. The generalized active force is equal to the sum over all the forces and torques acting on the system of the forces and torques themselves

dot multiplied by the appropriate partial velocities and partial angular velocities, respectively, at the points or bodies of application. These definitions will become clearer when they are applied in the following sections.

The selected generalized speeds

$$\omega_3,\ \Omega_3,\ \dot{q_i},\ \dot{p_j} \qquad (i = 1,..,n; j = 1,..,m)$$

together with the kinematic relationships

$$\begin{aligned} \omega_3 &= \dot{\theta} & \Omega_3 &= \dot{\beta} \\ \dot{q_i} &= \frac{d}{dt} q_i & \dot{p_j} &= \frac{d}{dt} p_j \qquad (i = 1,..,n; j = 1,..,m) \end{aligned} \tag{2}$$

completely characterize the 2+n+m degree of freedom system.

Define m_X to be the mass of body X, and J_X to be the moment of inertia of body X about an axis passing through its mass center, X*, and parallel to $\mathbf{n}_3$ (X=A,B_1,B_2,C). ρ_i (i=1,2) is the mass per unit length of link i. ${}^N\tilde{\mathbf{a}}^P$ is the linearized acceleration of point P in the inertial reference frame, while ${}^N\underline{\alpha}^X$ is the angular acceleration of body X in the same frame. Further, following Kane, the partial velocity of point P, and the partial angular velocity of frame/body X, with respect to the generalized speed r are defined as

$$\begin{aligned} {}^N\tilde{\mathbf{v}}_r^P &= \frac{\partial}{\partial \dot{r}}\left[{}^N\tilde{\mathbf{v}}^P\right] \\ {}^N\underline{\omega}_r^X &= \frac{\partial}{\partial \dot{r}}\left[{}^N\underline{\omega}^X\right] \end{aligned} \tag{3}$$

respectively.

2.4.1 Generalized Inertia Forces

With the above definitions, the generalized inertia force for the system under consideration is given by

$$\begin{aligned} \tilde{F}_r^* = &- J_A\ {}^N\underline{\omega}_r^A \bullet {}^N\underline{\alpha}^A - \int_0^{l_1} {}^N\tilde{\mathbf{v}}_r^{C_1} \bullet {}^N\tilde{\mathbf{a}}^{C_1} \rho_1 dx - J_{B_1}\ {}^N\underline{\omega}_r^{B_1} \bullet {}^N\underline{\alpha}^{B_1} \\ &- J_{B_2}\ {}^N\underline{\omega}_r^{B_2} \bullet {}^N\underline{\alpha}^{B_2} - m_{B_1}\ {}^N\tilde{\mathbf{v}}_r^{B_1^*} \bullet {}^N\tilde{\mathbf{a}}^{B_1^*} - m_{B_2}\ {}^N\tilde{\mathbf{v}}_r^{B_2^*} \bullet {}^N\tilde{\mathbf{a}}^{B_2^*} \end{aligned}$$

$$-\int_0^{l_2} {}^N\tilde{\mathbf{v}}_r^{C_2} \bullet {}^N\tilde{\mathbf{a}}^{C_2} \rho_2 dy - J_C \; {}^N\boldsymbol{\omega}_r^C \bullet {}^N\boldsymbol{\alpha}^C - m_C \; {}^N\tilde{\mathbf{v}}_r^{C^*} \bullet {}^N\tilde{\mathbf{a}}^{C^*} \tag{1}$$

where as before

$$r = \omega_3, \Omega_3, q_i, p_j \qquad (i = 1,\ldots, n; j = 1,\ldots,m)$$

The partial velocities are obtained from the velocity expressions which are correct to second order in elastic generalized coordinates and speeds (see section 2.2), and are then linearized in these same coordinates and speeds. The linearized partial velocities are given in Table 2.1, and the partial angular velocities are presented in Table 2.2. The linearized accelerations are obtained by differentiating the linearized velocity expressions at the end of section 2.4, and ignoring terms of second order or higher in the elastic deflections and speeds.

r	${}^N\tilde{\mathbf{v}}_r^{O_1}$	${}^N\tilde{\mathbf{v}}_r^{C_1}$	${}^N\tilde{\mathbf{v}}_r^{O_{21}}$
ω_3	$b_1\hat{\mathbf{a}}_2$	$-\sum_{i=1}^{n}\phi_i q_i \hat{\mathbf{a}}_1 + (x + b_1)\hat{\mathbf{a}}_2$	$-\sum_{i=1}^{n}\phi_i(l_1) q_i \hat{\mathbf{a}}_1 + (l_1 + b_1)\hat{\mathbf{a}}_2$
$\dot{q}_j$	0	$-\sum_{i=1}^{n}\beta_{ij}(x) q_i \hat{\mathbf{a}}_1 + \phi_j(x)\hat{\mathbf{a}}_2$	$-\sum_{i=1}^{n}\beta_{ij}(l_1) q_i \hat{\mathbf{a}}_1 + \phi_j(l_1)\hat{\mathbf{a}}_2$

Table 2.1a: Linearized Partial Velocities

r	${}^N\tilde{\mathbf{v}}_r^{H}$	${}^N\tilde{\mathbf{v}}_r^{B_1^*}$
ω_3	$-\sum_{i=1}^{n}A_i(l_1) q_i \hat{\mathbf{a}}_1 + L_1\hat{\mathbf{a}}_2$	$\left[-\sum_{i=1}^{n}A_i(l_1) q_i - e_1\tilde{\xi}_{\gamma_1}\right]\hat{\mathbf{a}}_1 + \left[L_1 + e_1\tilde{c}_{\gamma_1}\right]\hat{\mathbf{a}}_2$
$\dot{q}_j$	$-\sum_{i=1}^{n}C_{ij}(l_1) q_i \hat{\mathbf{a}}_1 + A_j(l_1)\hat{\mathbf{a}}_2$	$\left[-\sum_{i=1}^{n}C_{ij}(l_1) q_i - \phi'_j(l_1) e_1\tilde{\xi}_{\gamma_1}\right]\hat{\mathbf{a}}_1$ $+\left[A_j(l_1) + \phi'_j(l_1) e_1\tilde{c}_{\gamma_1}\right]\hat{\mathbf{a}}_2$

Table 2.1b: Linearized Partial Velocities (cont.)

r	${}^N\tilde{\mathbf{v}}_r^{B_2^*}$	${}^N\tilde{\mathbf{v}}_r^{Q_{22}}$
ω_3	${}^N\tilde{\mathbf{v}}^H_{\omega_3} - e_2 \sin\gamma_2 \hat{\mathbf{b}}_1 + e_2 \cos\gamma_2 \hat{\mathbf{b}}_2$	${}^N\tilde{\mathbf{v}}^H_{\omega_3} + b_{22}\hat{\mathbf{b}}_2$
$\dot{q}_j$	${}^N\tilde{\mathbf{v}}^H_{\dot{q}_j} - \phi'_j(l_1)e_2 \sin\gamma_2 \hat{\mathbf{b}}_1 + \phi'_j(l_1)e_2 \cos\gamma_2 \hat{\mathbf{b}}_2$	${}^N\tilde{\mathbf{v}}^H_{\dot{q}_j} + \phi'_j(l_1)b_{22}\hat{\mathbf{b}}_2$
Ω_3	$- e_2 \sin\gamma_2 \hat{\mathbf{b}}_1 + e_2 \cos\gamma_2 \hat{\mathbf{b}}_2$	$b_{22}\hat{\mathbf{b}}_2$

Table 2.1c: Linearized Partial Velocities (cont.)

The symbols in Table 2.1 that have not been defined previously are the modal integrals and constants that depend on the mode shapes. These are defined in Appendix B, where they are also evaluated for the physical manipulator parameters chosen (see Table 4.1), and assuming cantilever mode shapes.

r	${}^N\tilde{\mathbf{v}}_r^{C_2}$	${}^N\tilde{\mathbf{v}}_r^{T}$
ω_3	${}^N\tilde{\mathbf{v}}^H_{\omega_3} - \sum_{i=1}^{m}\psi_i p_i \hat{\mathbf{b}}_1 + (y + b_{22})\hat{\mathbf{b}}_2$	${}^N\tilde{\mathbf{v}}^H_{\omega_3} - \sum_{i=1}^{m}\psi_i(l_2) p_i \hat{\mathbf{b}}_1 + (l_2 + b_{22})\hat{\mathbf{b}}_2$
$\dot{q}_j$	${}^N\tilde{\mathbf{v}}^H_{\dot{q}_j} - \phi'_j(l_1)\sum_{i=1}^{m}\psi_i p_i \hat{\mathbf{b}}_1 + \phi'_j(l_1)(y + b_{22})\hat{\mathbf{b}}_2$	${}^N\tilde{\mathbf{v}}^H_{\dot{q}_j} - \phi'_j(l_1)\sum_{i=1}^{m}\psi_i(l_2) p_i \hat{\mathbf{b}}_1 + \phi'_j(l_1)(l_2 + b_{22})\hat{\mathbf{b}}_2$
Ω_3	$- \sum_{i=1}^{m}\psi_i p_i \hat{\mathbf{b}}_1 + (y + b_{22})\hat{\mathbf{b}}_2$	$- \sum_{i=1}^{m}\psi_i(l_2) p_i \hat{\mathbf{b}}_1 + (l_2 + b_{22})\hat{\mathbf{b}}_2$
$\dot{p}_j$	$-\sum_{i=1}^{m}\beta^*_{ij}(y) p_i \hat{\mathbf{b}}_1 + \psi_j(y)\hat{\mathbf{b}}_2$	$-\sum_{i=1}^{m}\beta^*_{ij}(l_2) p_i \hat{\mathbf{b}}_1 + \psi_j(l_2)\hat{\mathbf{b}}_2$

Table 2.1d: Linearized Partial Velocities (cont.)

r	${}^N\bar{v}^{C^*}_r$
ω_3	${}^N\bar{v}^H_{\omega_3} - \sum_{i=1}^{m} B_i(l_2)\, p_i\, \hat{\mathbf{b}}_1 + L_2\hat{\mathbf{b}}_2$
$\dot{q}_j$	${}^N\bar{v}^H_{\dot{q}_j} - \phi'_j(l_1)\sum_{i=1}^{m} B_i(l_2)\, p_i\, \hat{\mathbf{b}}_1 + \phi'_j(l_1) L_2\hat{\mathbf{b}}_2$
Ω_3	$-\sum_{i=1}^{m} B_i(l_2)\, p_i\, \hat{\mathbf{b}}_1 + L_2\hat{\mathbf{b}}_2$
$\dot{p}_j$	$-\sum_{i=1}^{m} C^*_{ij}(l_2)\, p_i\, \hat{\mathbf{b}}_1 + B_j(l_2)\hat{\mathbf{b}}_2$

Table 2.1e: Linearized Partial Velocities (cont.)

r	${}^N\underline{\omega}^A_r$	${}^N\underline{\omega}^{B_1}_r$	${}^N\underline{\omega}^{B_2}_r$	${}^N\underline{\omega}^C_r$
ω_3	$\hat{\mathbf{a}}_3$	$\hat{\mathbf{a}}_3$	$\hat{\mathbf{a}}_3$	$\hat{\mathbf{a}}_3$
$\dot{q}_j$	0	$\phi'_j(l_1)\hat{\mathbf{a}}_3$	$\phi'_j(l_1)\hat{\mathbf{a}}_3$	$\phi'_j(l_1)\hat{\mathbf{a}}_3$
Ω_3	0	0	$\hat{\mathbf{a}}_3$	$\hat{\mathbf{a}}_3$
$\dot{p}_j$	0	0	0	$\psi'_j(l_2)\hat{\mathbf{a}}_3$

Table 2.2: Linearized Partial Angular Velocities

2.4.2 Generalized Active Forces

The generalized active force can be written as

$$F_r = (F_r)_I + (F_r)_C$$
$$r = \omega_3, \Omega_3, \dot{q}_i, \dot{p}_j \qquad (i = 1,\ldots, n; j = 1,\ldots,m) \tag{1}$$

where $(F_r)_I$ consists of those contributions to the generalized active forces due to internal forces; and $(F_r)_C$ are those contributions due to external or "control" forces. Taking into

account the definitions given at the beginning of this section, and referring to Fig. 2.3, the expression for the $(F_r)_I$ is

$$
\begin{aligned}
(\bar{F}_r)_I &= {}^N\bar{\omega}_r^A \bullet [M_1(0)\hat{\mathbf{a}}_3] + {}^N\bar{v}_r^{Q_1} \bullet [-V_1(0)\hat{\mathbf{a}}_2] + \int_0^{l_1} {}^N\bar{v}_r^{C_1} \bullet \left[-\frac{\partial V_1}{\partial x}\hat{\mathbf{a}}_2\right] dx \\
&+ {}^N\bar{v}_r^{Q_{21}} \bullet [V_1(l_1)\hat{\mathbf{a}}_2] + {}^N\bar{\omega}_r^{B_1} \bullet [-M_1(l_1)\hat{\mathbf{a}}_3] + {}^N\bar{\omega}_r^{B_2} \bullet [M_2(0)\hat{\mathbf{a}}_3] \\
&+ {}^N\bar{v}_r^{Q_{22}} \bullet [-V_2(0)\hat{\mathbf{b}}_2] + \int_0^{l_2} {}^N\bar{v}_r^{C_2} \bullet \left[-\frac{\partial V_2}{\partial y}\hat{\mathbf{b}}_2\right] dy + {}^N\bar{v}_r^{T} \bullet [V_2(l_2)\hat{\mathbf{b}}_2] \\
&+ {}^N\bar{\omega}_r^{C} \bullet [-M_2(l_2)\hat{\mathbf{a}}_3]
\end{aligned}
$$

$$
r = \omega_3, \Omega_3, q_i^{\cdot}, p_j^{\cdot} \qquad (i = 1,\ldots, n; j = 1,\ldots,m) \tag{2}
$$

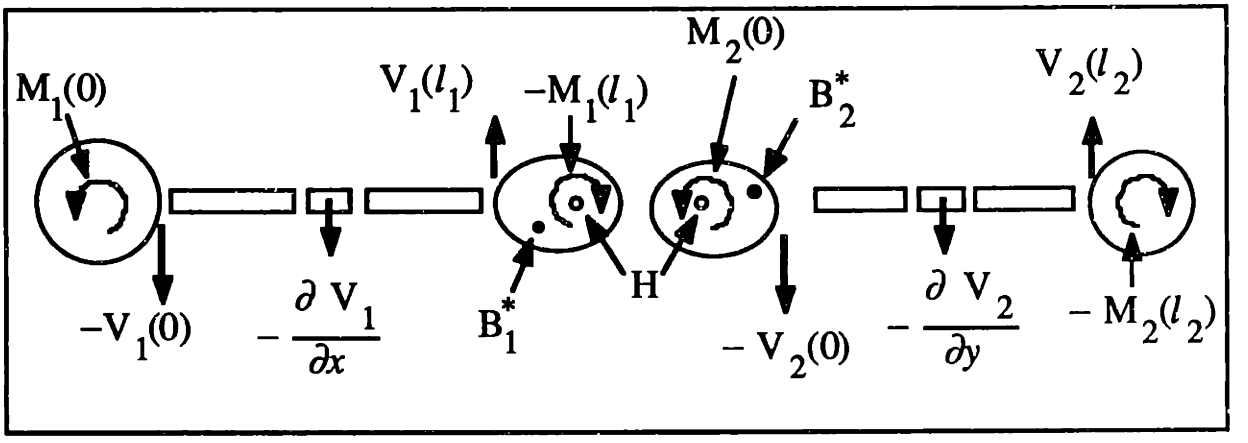


Figure 2.3: Schematic of Internal Forces and Elbow Joint Definitions

For a Bernoulli-Euler beam, under the governing assumptions presented in section 2.1,

$$
M_1(x) = (EI)_1 \frac{\partial^2 u_2}{\partial x^2} \qquad M_2(y) = (EI)_2 \frac{\partial^2 v_2}{\partial y^2}
$$

$$
V_1(x) = \frac{\partial}{\partial x} M_1(x) \qquad V_2(y) = \frac{\partial}{\partial y} M_2(y) \tag{3}
$$

where $(EI)_i$ (i=1,2) is the bending stiffness of link i. Taking into account the fact that link 1 is attached to body A in a cantilevered way, and link 2 is attached to body B_2 in a cantilevered way also, it is seen that

$$\phi_i(0) = \phi'_i(0) = 0 \qquad i = 1,..,n$$
$$\psi_j(0) = \psi'_j(0) = 0 \qquad j = 1,..., m \tag{4}$$

Using these boundary conditions, and executing the required operations, equations (2) reduce to:

$$\left(\tilde{F}_{\omega_3}\right)_I = 0$$
$$\left(\tilde{F}_{\Omega_3}\right)_I = 0$$

$$\left(\tilde{F}_{\dot{q}_j}\right)_I = -\sum_{i=1}^{n} H_{ij} q_i, \qquad j = 1,..., n$$
$$\left(\tilde{F}_{\dot{p}_j}\right)_I = -\sum_{i=1}^{m} H^*_{ij} p_i, \qquad j = 1,..., m, \tag{5}$$

where

$$H_{ij} = \int_0^{l_1} (EI)_1 \phi''_i(x)\phi''_j(x)dx$$
$$H^*_{ij} = \int_0^{l_2} (EI)_2 \psi''_i(y)\psi''_j(y)dy \tag{6}$$

and the orthogonality conditions governing these integrals have been presented in equation 2.3.8.

For the case at hand, the generalized active forces due to external forces have a particularly simple form

$$\left(\tilde{F}_r\right)_C = {}^N\boldsymbol{\omega}^A_r \bullet \mathbf{T}_S + {}^N\boldsymbol{\omega}^{B_1}_r \bullet (-\mathbf{T}_E) + {}^N\boldsymbol{\omega}^{B_2}_r \bullet \mathbf{T}_E$$
$$r = \omega_3, \Omega_3, \dot{q}_i, \dot{p}_j \qquad (i = 1,..., n; j = 1,..,m) \tag{7}$$

where the control forces are couples of torque $\mathbf{T}_S$ and $\mathbf{T}_E$ applied at the shoulder body, A, and the elbow body, B_2, respectively:

$$\mathbf{T}_S = T_1 \hat{\mathbf{a}}_3, \qquad \mathbf{T}_E = T_2 \hat{\mathbf{a}}_3$$

Again effecting the required dot products, and making use of Table 2.1, equations (7) simplify to

$$\left(\tilde{F}_{\omega_3}\right)_C = T_1$$

$$\left(\tilde{F}_{\Omega_3}\right)_C = T_2$$

$$\left(\tilde{F}_{\dot{q}_j}\right)_C = 0, \qquad j = 1,\ldots,n$$

$$\left(\tilde{F}_{\dot{p}_j}\right)_C = 0, \qquad j = 1,\ldots,m. \tag{8}$$

From equations (5) and (8) the linearized generalized active forces result

$$\tilde{F}_{\omega_3} = T_1$$

$$\tilde{F}_{\Omega_3} = T_2$$

$$\tilde{F}_{\dot{q}_j} = -\sum_{i=1}^{n} H_{ij} q_i, \qquad j = 1,\ldots,n$$

$$\tilde{F}_{\dot{p}_j} = -\sum_{i=1}^{m} H^*_{ij} p_i, \qquad j = 1,\ldots,m\,. \tag{9}$$

2.4.3 Equations of Motion

Finally, the 2+n+m ordinary differential equations of motion are obtained from

$$\tilde{F}^*_r + \tilde{F}_r = 0, \qquad r = \omega_3, \Omega_3, \dot{q}_i, \dot{p}_j \qquad (i = 1,\ldots, n; j = 1,\ldots,m) \tag{1}$$

Let **x** be a state vector such that

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_E \end{bmatrix} = [\theta, \beta, q_1, \ldots, q_n, p_1, \ldots, p_m]^T$$

and **T** be the control input vector

$$\mathbf{T} = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}$$

Then equation (1) can be written in the matrix form:

$$\begin{bmatrix} M_{RR}(\mathbf{x}) & M_{RE}(\mathbf{x}) \\ M_{ER}(\mathbf{x}) & M_{EE}(\mathbf{x}) \end{bmatrix} \ddot{\mathbf{x}} + \begin{bmatrix} 0 & 0 \\ 0 & K_{EE} \end{bmatrix} \mathbf{x} = \begin{bmatrix} I \\ 0 \end{bmatrix} \mathbf{T} + \mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}) \tag{2}$$

where F(x,dx/dt) is a 2+n+m vector of gyroscopic (centripetal and coriolis) forces;

$$(K_{EE})_{ij} = H_{ij}, \qquad i, j = 1, \ldots, n$$

$$(K_{EE})_{n+i,n+j} = H^*_{ij}, \quad i, j = 1, \ldots, m$$

and M(x) is the configuration dependent mass matrix.

Setting

$$\mathbf{y} = \begin{bmatrix} \mathbf{x} \\ \dot{\mathbf{x}} \end{bmatrix}$$

the matrix equation (2) can be written in the form

$$\dot{\mathbf{y}} = \begin{bmatrix} 0 & I \\ -M^{-1}K & 0 \end{bmatrix} \mathbf{y} + \begin{bmatrix} 0 \\ M^{-1} \end{bmatrix} \mathbf{T} + \begin{bmatrix} 0 \\ M^{-1}\mathbf{F} \end{bmatrix} \tag{3}$$

in which form it can be integrated straightforwardly using a Runge-Kutta fourth-order integration scheme (see chapter 4).

The full equations of motion that result from the above development are quite extensive. For this reason, they are presented in Appendix C in the matrix form of equation (2).

CHAPTER 3: DYNAMIC ANALYSIS

INTRODUCTION

In the previous chapter, the motion equations for the two link arm, correct to first order in small elastic deflections and speeds, and exact in rigid body motions, were obtained. It is now desired to investigate the relative significance of nonlinear terms which involve the elastic coordinates and speeds. For this purpose, this chapter begins with a brief general presentation of the equations of motion for open chains of elastic bodies, linearized in small elastic deflections and rates. After considering possible simplifications to these equations, three models are proposed for purposes of investigating the effects of these simplifications on the systems under consideration.

Using these three models as a guide, the simple one link example of section A.2 is analyzed. By virtue of its simplicity, this model serves to clearly identify the issues of importance, and to highlight the complexity introduced by inertial coupling between multiple links. Predictions using this example are made as to the relative significance of elastic nonlinear terms in the equations of motion of more complex systems. These predictions are tested in the next chapter via simulation of the two link arm using the equations developed in chapter two. The chapter ends with an eigenanalysis of the two link manipulator that will be useful when the simulation results are analyzed in chapter four.

3.1 FORM OF THE EQUATIONS OF MOTION FOR A CHAIN OF ELASTIC BODIES

The equations of motion of an open chain of elastic bodies can be expressed quite generally as [8]:

$$\begin{bmatrix} M_{RR}(x,q) & M_{RE}(x,q) \\ M_{ER}(x,q) & M_{EE}(x,q) \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} T_C \\ 0 \end{bmatrix} + \begin{bmatrix} T_{ext,R} \\ T_{ext,E} \end{bmatrix} + \begin{bmatrix} F_R(x, q, \dot{x}, \dot{q}) \\ F_E(x, q, \dot{x}, \dot{q}) \end{bmatrix}, \quad (1)$$

where x is a vector of rigid generalized coordinates; q is a vector of the elastic generalized coordinates; M_{RR}, M_{RE}, M_{EE} form the configuration-dependent mass matrix; T_C is a vector

of control forces and we assume only the rigid coordinates are directly actuated (as in joint-torque actuators in a manipulator); T_{ext} is a vector of other generalized external forces; K_{EE} is a constant stiffness matrix (see equation (2) below) and F is a vector of nonlinear inertial (including coriolis and centripetal) forces.

The important class of systems for which the elastic deformations remain small, so that terms of second order in q and dq/dt can be ignored, is often considered. Strictly speaking this requires that $\|q\|$ and $\|dq/dt\|$ be infinitesimally small. However it is known that if, for example, a flexible body is modelled as a beam, it is sufficient that the elastic deformations do not exceed one tenth of the length of the beam in order to use linear Bernoulli-Euler beam theory. At any rate, given this assumption of small elastic deflections, the previous equation could be expanded in order to show more explicitly the form of the nonlinear terms:

$$\begin{bmatrix} M_{RR}(x,q) & M_{RE}(x,q) \\ M_{ER}(x,q) & M_{EE}(x,q) \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} T_C \\ 0 \end{bmatrix} + \begin{bmatrix} T_{ext,R} \\ T_{ext,E} \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & K_{EE} \end{bmatrix} \begin{bmatrix} x \\ q \end{bmatrix}$$

$$+ \sum_{i=1}^{n} f_{1ii}(x)\dot{x}_i^2 + \frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq i}^{n} f_{1ij}(x)\dot{x}_i\dot{x}_j + \frac{1}{2}\sum_{i=1}^{n} f_{2ii}(x)q\dot{x}_i^2$$

$$+ \frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq i}^{n} f_{2ij}(x)q\dot{x}_i\dot{x}_j + \sum_{i=1}^{n} f_{3i}(x)\dot{q}\dot{x}_i - \begin{bmatrix} M^*_{RR}(x,q) \\ M^*_{ER}(x,q) \end{bmatrix} \ddot{x},$$

$$\left(M^*_{RR}(x,q)\right)_{ij} = m^T_{1ij}(x)q, \quad \left(M^*_{ER}(x,q)\right)_{ij} = m^T_{2ij}(x)q \tag{2}$$

where n above is the number of rigid body coordinates; f_{1ij} is a column matrix, but f_{2ij} and f_{3i} are n+m by m matrices, where m is the number of elastic coordinates.

In light of the discussion in Appendix A, some of the terms in equation (2) could be written as:

$$\begin{aligned} f_{2ij}(x) &= f^1_{2ij}(x) + f^2_{2ij}(x) \\ m_{1ij}(x) &= m^1_{1ij}(x) + m^2_{1ij}(x) \\ m_{2ij}(x) &= m^1_{2ij}(x) + m^2_{2ij}(x) \end{aligned} \tag{3}$$

where the superscript 2 terms can only be obtained through the use of displacement and velocity expressions, in the development of the equations of motion, that are accurate to

second order in the elastic generalized coordinates and speeds (q and u). This requires the use of nonlinear strain-displacement relations (as in App. A, [13]), nonlinear kinematic constraints [11, 24], or the use of a nonlinear "geometric stiffening" term appended to the incorrectly linearized equations of motion [27, 1].

The complexity of equations (2) and the difficulty involved in obtaining the nonlinear terms have historically prompted attempts at simplification. It is common [8] for example to assume small velocities and drop all terms nonlinear in rates. This results in rate-linear motion equations that greatly simplify the dynamicist's task. It has been pointed out [4], however, that in the case of n-link rigid manipulators in any configuration, the velocity and acceleration terms of the dynamic equations have the same relative significance at any speed of movement. The fact that the omission of these terms does not significantly affect simulation results is attributed to the fact that gravity and joint friction usually overpower inertial terms. These results have not been extended to chains of flexible bodies. One might argue that in some limit (e.g., vanishingly small q) the equations of motion of the flexible multibody system should reduce to those of the rigid multibody system. Then it seems that a good case could be made for the inclusion of at least nonlinear terms in the rigid body rates in the rate-linear equations (i.e., $f_{1ij}(x)$), particularly considering the fact that future, fast, space manipulators with low joint frictions are part of the class of systems under consideration.

Faced with this, there are two ways in which to proceed with respect to the equations of motion: we can be consistent, or we can be selective. To be consistent requires keeping all terms of order $\|q\|$ and $\|u\|$ in equation (2), i.e., no simplification. There is also the problem that the superscript ^{2} terms in equations (3) are not readily available in the general case since they depend on nonlinear elastic theory or nonlinear kinematics of deformation for their derivation. We could just make do with the superscript ^{1} terms in equations (3) (standard approach) but this would not be consistent nor justifiable since there is no *a priori* reason to guarantee $\|f_{2ij}{}^1\| >> \|f_{2ij}{}^2\|$, for example.

We are forced then to be selective, at least in the general case. Now we have to rely mostly on experience and simulation to determine which terms are important and which negligible under given conditions. Using a dynamic system very much like the one used in the simple example of section A.2, an empirical speed limit has been found beyond which the standard linear finite element or modal formulations of the model give erroneous results [11]. This limit is specified as follows: the magnitude of the spinning rate of the system has to be one order of magnitude less than the fundamental vibration frequency of the beam nominally stationary in Newtonian reference frame. This simulation result is used to claim

the following: unless the equations of motion exact to first order in elastic generalized coordinates and speeds are available, rigid body angular rates are limited to less than an order of magnitude lower than the lowest fundamental bending frequency of the system. In view of this, it might be possible to model the system accurately enough by just keeping the rate-linear equations together with terms f_{1ij}. In other words, it might not be detrimental to drop all nonlinear terms involving elastic coordinates and speeds. It can further be claimed that speed or acceleration limits also exist in translational rates or accelerations, arising both from those mass matrix terms that depend on q and cannot be obtained through the standard approaches and from such terms in f_{2ij}. In section 3.3 these claims are investigated analytically.

3.2 COMPARISON MODELS

In what follows, taking the above discussion into account, three types of models for a flexible multibody system under study shall be considered. The "consistent" model will be that which retains all terms to first order in elastic coordinates and speeds. This is the correctly, or consistently, linearized model. The equations of motion developed in chapter two for the two-link manipulator, and in Appendix A for the simple one link example, are representative of this type. This model implies no further simplification of the linearized equations and, as can be seen from Appendix C, even for the case of only two flexible links results in unwieldy expressions.

The second model to be considered is the "inconsistent" model. This is obtained through the use of linear kinematics of deformation in the determination of system velocities. It is clear from the discussion in Appendix A that this results in equations of motion that are incorrectly linearized. They lack the superscript ^{2} terms mentioned in the previous section. While this model results in equations that are as unwieldy as those of the consistent model, it yields erroneous results, as shall be seen. The advantage of this model lies in its use of linear elastic theory, and in the fact that, as it turns out, its results are valid within certain rigid body rate limits.

A "ruthlessly linearized", or simply "ruthless" model, shall be the last one considered. In this model all nonlinear terms which include elastic coordinates and speeds are ignored, including those terms in the mass matrix which depend on elastic coordinates. Referring to section 3.1, a ruthless model does not contain terms f_{2ij} and f_{3i}, or $m_{1ij}(x)$, $m_{2ij}(x)$. Unlike the first two models, the ruthless model yields equations of motion that are greatly simplified, and since terms nonlinear in rigid body coordinates and rates are kept,

this model degrades to the correct model of a rigid multibody system as flexible deflections approach zero.

In the next section, limits of validity for all three models shall be investigated.

3.3 SINGLE FLEXIBLE BODY EXAMPLE

Consider the simple, slender, uniform beam cantilevered to a rigid base, of section A.2 (see Fig. A.1). The equations of motion, exact to first-order in generalized elastic coordinates and speeds were presented in eq. (A.2.1.14) . All symbols have been defined in equations (A.2.1.15).

3.3.1 Pure Rotation

Specializing equation (A.2.1.14) to the prescribed motion of uniform rotation of the base,

$$v_1=v_2=0, \quad v_3=\Omega=\text{constant}$$

the following is obtained,

$$\sum_{i=1}^{n} G_{ij}\ddot{q}_i + \sum_{i=1}^{n}\left[H_{ij} + \Omega^2(b\mu_{ij} + \eta_{ij} - G_{ij})\right]q_i = 0$$

$$(j = 1,\ldots,n) \tag{1}$$

We recall that the terms μ_{ij} and η_{ij} cannot be obtained using linear kinematics of deformation but were obtained in section A.2.1 through the use of nonlinear strain-displacement relations. Equation (1) represents the consistent model. The term

$$\Omega^2(b\mu_{ij} + \eta_{ij})$$

is known as the geometric stiffness matrix for this specialized rigid body motion (rotation). Laskin et al. [13] point out that this matrix provides coupling among the transverse vibrational modes of the beam and this coupling leads to the phenomenon of geometric stiffening. They further mention that this stiffening mainly raises the value of the bending vibrational frequencies.

Indeed, noting that equation (2.3.8) is true for a variety of simple beam mode shapes, and using the relations of Appendix B, equation (1) becomes

$$\begin{bmatrix} \ddots & 0 & 0 \\ 0 & m_B & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \ddot{q}_i \\ \vdots \end{bmatrix} + \left\{ \begin{bmatrix} \ddots & 0 & 0 \\ 0 & m_B \omega_i^2 & 0 \\ 0 & 0 & \ddots \end{bmatrix} + \Omega^2 \begin{bmatrix} \ddots & \vdots & \iddots \\ \cdots & b\mu_{ij} + \eta_{ij} - m_B \delta_{ij} & \cdots \\ \iddots & \vdots & \ddots \end{bmatrix} \right\} \begin{bmatrix} \vdots \\ q_i \\ \vdots \end{bmatrix} = 0 \tag{2}$$

where the following equation has been used

$$\lambda_i^2 = \omega_i l^2 \sqrt{\frac{\rho}{EI}},$$

and the Kronecker delta is given by

$$\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}.$$

As it turns out in the case of cantilever mode shapes, the geometric stiffness matrix is positive definite (see section B.2.2), and dominates over the modal mass matrix G_{ij}. Consequently, equation (2) yields the correct spin-stiffening result as the angular rate Ω is increased.

The inconsistent model is obtained using linear kinematics of deformation, so that the geometric stiffness matrix is absent. In this case, equation (2) reduces to

$$\begin{bmatrix} \ddots & 0 & 0 \\ 0 & m_B & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \ddot{q}_i \\ \vdots \end{bmatrix} + m_B \begin{bmatrix} \ddots & 0 & 0 \\ 0 & \omega_i^2 - \Omega^2 & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ q_i \\ \vdots \end{bmatrix} = 0 \tag{3}$$

It is clear from the above equation that when Ω reaches the value of the first fundamental bending frequency of the beam, the stiffness matrix will be singular. This amounts to the model predicting divergence under large enough spin, which is incorrect. On the other hand, if Ω is much less than the first fundamental bending frequency, say about one order of magnitude less, then the discrepancy between equations (2) and (3) may be less noticeable and results using equation (3) might be qualitatively "correct."

From this it is apparent that a limit on the rigid body spin rate exists, below which the inconsistent model is "valid." This limit has been pointed out experimentally in the

literature in recent years [11,24,27]. Its analytical confirmation here serves as a check on the equations.

Ignoring all nonlinear terms that involve the elastic coordinates and speeds in equations (A.2.1.14), the ruthless model for uniform rotation of the base results in the equations

$$\begin{bmatrix} \ddots & 0 & 0 \\ 0 & m_B & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \tilde{q}_i \\ \vdots \end{bmatrix} + m_B \begin{bmatrix} \ddots & 0 & 0 \\ 0 & \omega_i^2 & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ q_i \\ \vdots \end{bmatrix} = 0 \tag{4}$$

While equation (4) does not exhibit the correct spin-stiffening of equation (2), it also does not show the incorrect destiffening of equation (3). Clearly, if the angular rate Ω is small enough, equation (4) will be as "valid" as equations (3) or (2), with the added advantage of yielding results that are more conservative than those of equations (3). In other words, even for large Ω, the ruthless model yields qualitatively similar results to those of the consistent one, while the inconsistent model fails.

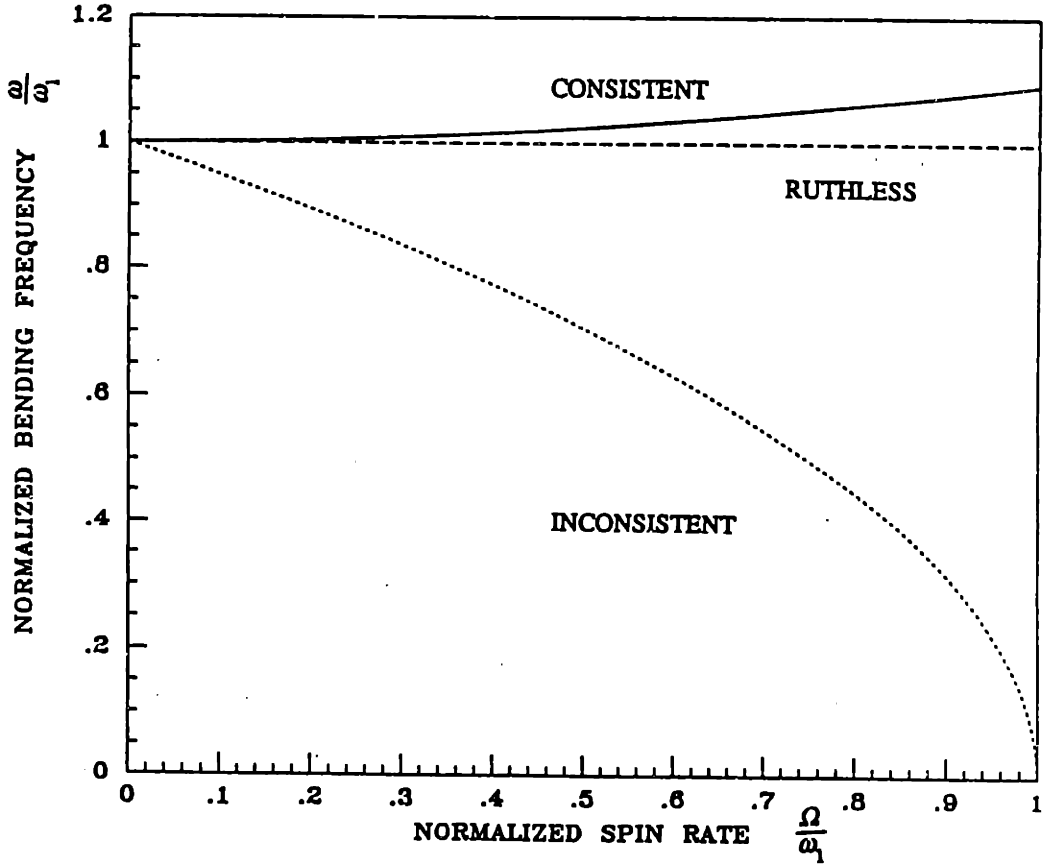


Fig. 3.1: Fundamental Bending Frequency of Spinning Beam v.s. Spin Rate

The above results are succinctly summarized in Fig. 3.1, where the first fundamental bending frequency of the spinning beam is plotted versus spin rate for all three models (for the case of only one assumed mode).

3.3.2 Pure Linear Acceleration

Specializing to the prescribed motion of constant translational acceleration along the neutral axis of the beam

$$\dot{v}_1 = g = \text{constant}, \qquad v_2 = v_3 = 0$$

equation (A.2.1.14) turns into

$$\sum_{i=1}^{n} G_{ij}\ddot{q}_i + \sum_{i=1}^{n} \left(H_{ij} - g\mu_{ij}\right) q_i = 0$$
$$(j = 1, \ldots, n) \tag{1}$$

which are the consistent model equations. Making use of relations (2.3.8), equation (1) becomes

$$\begin{bmatrix} \ddots & 0 & 0 \\ 0 & m_B & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \ddot{q}_i \\ \vdots \end{bmatrix} + \left\{ \begin{bmatrix} \ddots & 0 & 0 \\ 0 & m_B \omega_i^2 & 0 \\ 0 & 0 & \ddots \end{bmatrix} - g \begin{bmatrix} \ddots & \vdots & \iddots \\ \cdots & \mu_{ij} & \cdots \\ \iddots & \vdots & \ddots \end{bmatrix} \right\} \begin{bmatrix} \vdots \\ q_i \\ \vdots \end{bmatrix} = 0 \tag{2}$$

Since μ_{ij} is positive definite in this case, for large enough g destiffening is again predicted. For g large enough, the stiffness matrix becomes non-positive definite which implies that the beam buckles due to its own weight. The predicted buckling is correct, and is lost with the inconsistently or ruthlessly linearized models, as shall be seen. This suggests that a translational acceleration limit also exists beyond which our model is again grossly incorrect if we do not use equations exact to first order.

Reducing the model to a single assumed mode, it is easy to obtain a first approximation to this translational acceleration limit:

$$g = \frac{l\omega_1^2}{\xi_1} = \frac{EI\lambda_1^4}{m_B l^2 \xi_1} \tag{3}$$

ξ_1 and λ_1 are constants that depend on the mode shapes chosen (see section B.2.2). For the parameters used for the outboard link of the manipulator, this limit is approximately g~45g's.

The inconsistent and the ruthless models have the same equations for this case, mainly

$$\begin{bmatrix} \ddots & 0 & 0 \\ 0 & m_B & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ \ddot{q}_i \\ \vdots \end{bmatrix} + \begin{bmatrix} \ddots & 0 & 0 \\ 0 & m_B \omega_i^2 & 0 \\ 0 & 0 & \ddots \end{bmatrix} \begin{bmatrix} \vdots \\ q_i \\ \vdots \end{bmatrix} = 0 \tag{4}$$

Once again it might be expected that for small enough axial accelerations, g, equation (4) will yield results qualitatively similar to those of the consistent model, equation (2). It is significant that the ruthless model is as good as the inconsistent model.

3.3.3 General Stability Boundary of the Consistent Model

In general, from equations (A.2.1.14), an effective stiffness matrix can be written as

$$(K)_{ij} = H_{ij} + v_2 v_3 \mu_{ij} + v_3^2 [b\mu_{ij} + \eta_{ij} - G_{ij}] - \dot{v}_1 \mu_{ij} \tag{1}$$

whence it can be seen that besides being affected by axial acceleration and centripetal terms, the "dynamic" stiffness contains a coriolis term. Again using only one assumed mode, it is possible to obtain a three-dimensional stability boundary depending on the values, all functions of time, of dv_1/dt, v_2, and v_3. The boundary is given by the equation

$$y = -\left\{ \frac{\frac{1}{\xi_1} - z}{x} + \left[\frac{b}{l} + \frac{1}{\xi_1}(\gamma_1 - 1) \right] x \right\} \tag{2}$$

where x, y, and z are nondimensional variables given by

$$x = \frac{v_3}{\omega_1}, \qquad y = \frac{v_2}{l\omega_1}, \qquad z = \frac{\dot{v}_1}{l\omega_1^2},$$

and ξ_1 and γ_1 are constants that arise from the foreshortening matrices η_{ij} and μ_{ij}, and that depend on the mode shapes (see section B.2.2). Equation (2) is obtained by requiring the "dynamic" stiffness to equal zero.

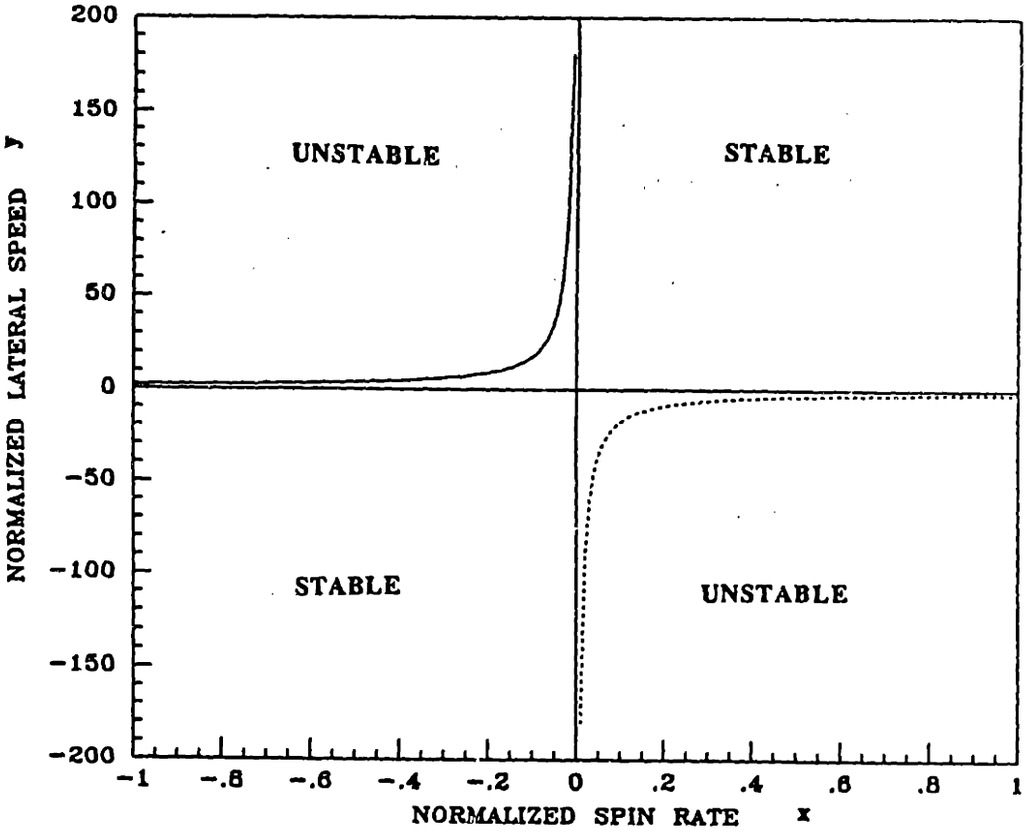


Fig. 3.2: Stability Boundary for Axial Acceleration Below Critical

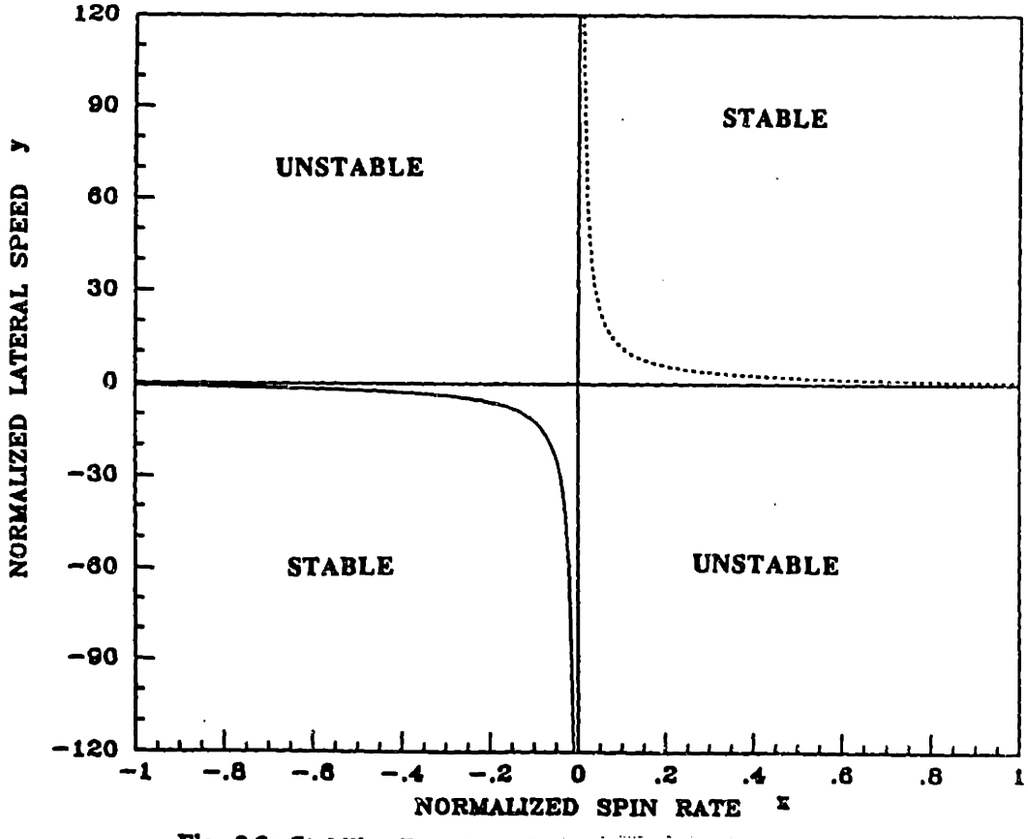


Fig. 3.3: Stability Boundary for Axial Acceleration Above Critical

Figs. 3.2 and 3.3 show two-dimensional projections onto the x-y plane for two non-dimensionalized values of axial acceleration. Not surprisingly, taking into consideration the previous results, the larger z is, the larger the region in the x-y plane where divergence is predicted. Also note there are two qualitatively different x-y boundaries, corresponding to values of z above and below the "critical" value which is the limit predicted in the previous section for the case of axial acceleration only.

This stability boundary is used in the simulation of the two-link manipulator (see section 4.3.3), to determine when (and if) divergence of the outboard link is predicted by the consistent equations for the tried maneuvers. This helped distinguish between correct results and numerical ill-behavior.

3.4 PREDICTIONS

From an examination of the analytical results presented above, the following predictions are ventured in what concerns the relative significance of nonlinear terms found in more complex systems.

For the values of the beam properties selected (see Table 4.1), and for maneuvers typical of a manipulator, it seems that the more restrictive limit on the validity of other than the correctly linearized equations is by far the one imposed on the rotational rigid body rates. Therefore, for rotational rigid body rates of less than an order of magnitude of the first fundamental bending frequency of a system of the type under consideration, no significant difference might be expected among the three models studied above. For rates larger than this limit, the inconsistent model will degrade, eventually predicting divergence; while the ruthless model will become increasingly more inaccurate with increasing rates, when compared to the consistent model.

3.5 EIGENANALYSIS OF THE TWO-LINK MANIPULATOR

An examination of the equations of motion for the two-link, flexible manipulator (see App. C) makes it clear that the complexity of the mass matrix and nonlinear inertial terms could be diminished immensely by dropping terms involving the generalized elastic coordinates and speeds (i.e., the "ruthless" case). It would be advantageous to know, then, if these terms have a significant effect on the dynamics of a chain of elastic bodies if we are limited to the low rotational speeds and translational accelerations encountered during slew maneuvers with joint torques limited by the requirement that flexible

deflections remain small. In order to investigate this, we propose to examine the three models presented in section 3.2, to note, consistent, inconsistent, and ruthless, as applied to the two-link arm. The complexity of the motion equations precludes an analytical study akin to the one presented in section 3.3. It is therefore necessary to rely on numerical simulation and compare the performance of the three "models" of the arm when it is subjected to several possibly typical maneuvers.

In order to examine the above predictions on the manipulator, it is desirable to obtain a measure of the fundamental vibration frequency of the system, since this quantity dominates the first approximation of the limit on the validity of the simplified models. The two-link manipulator, however, is a nonlinear dynamical system with configuration dependent mass matrix, as presented in section 3.1 (see also App. C). In this case, the standard eigenanalysis does not strictly apply. To obtain a first order linear approximation of the system frequencies and mode shapes, a configuration dependent eigenanalysis is performed on the manipulator.

An examination of the equations for the system presented in Appendix C reveals that the mass matrix depends on the relative angle between the links (β), as well as on the elastic generalized coordinates of both links. Choosing the nominal configurations to be the undeformed links, all the generalized elastic coordinates are zero, and the system frequencies and mode shapes are determined for several values of β. For each configuration, the eigenanalysis is performed for both free and locked joints. The values of the parameters used in this analysis correspond to those presented in chapter four for the numerical simulation (see Table 4.1).

(deg)	Free joints (rad/sec)				Locked joints			
β	i= 1	2	3	4	i= 1	2	3	4
0	42.9	68.1	153.7	224.3	3.03	7.36	62.2	120.7
45	42.8	68.0	153.6	224.3	3.15	6.82	62.0	120.7
90	42.7	68.0	153.5	224.3	3.53	5.90	61.6	120.7
135	42.8	68.0	153.6	224.3	4.08	5.32	61.6	120.7
180	42.9	68.1	153.7	224.3	4.35	5.19	61.8	120.8

Table 3.1: Configuration Dependent Manipulator Frequencies

As might be expected, the free joints results are almost identical for all values of β considered (see Table 3.1). Fig. 3.5 shows these results for β equal to zero. For fixed joints, on the other hand, the lowest two frequencies move closer together on the $j\omega$ axis as β is increased from 0 to 180 degrees (see Fig. 3.4 and Table 3.1). The higher frequencies are less affected by changes in β. Figs. 3.6 to 3.10 show the first four fixed joints mode shapes for the different values of relative angle. Since only eight assumed modes (four per link) were used (as a maximum) in implementing the simulation, it is to be expected that only the first four frequencies of the system presented herein are accurate, while perhaps even fewer mode shapes.

(deg)		Link 2 rigid			(rad/sec)		Link 1 rigid		
β	i=1	2	3	4		i=1	2	3	4
0	3.52	29.2	320.5	907.0		4.67	63.6	202.0	479.2
90	4.04	23.9	318.6	905.6		-	-	-	-

Table 3.2: Frequencies for Each Link when Other is Rigid and Joints are Locked

For comparison purposes, and for reference, Table 3.2 lists the eigenfrequencies for each link, when the other is considered rigid and both joints are locked. In the following pages, the system mode shapes have been normalized so that the largest deflection of either link be approximately ten percent of the link lengths, which have been normalized to one.

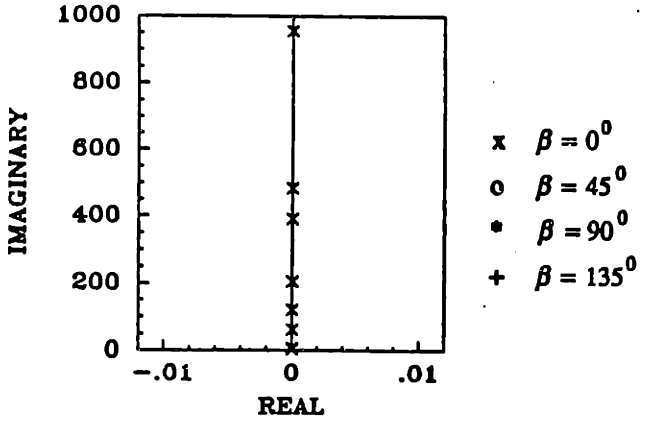


Fig. 3.4a: s-plane Plot of Manipulator Frequencies with Locked Joints for Elbow Angle equal to 0 deg.

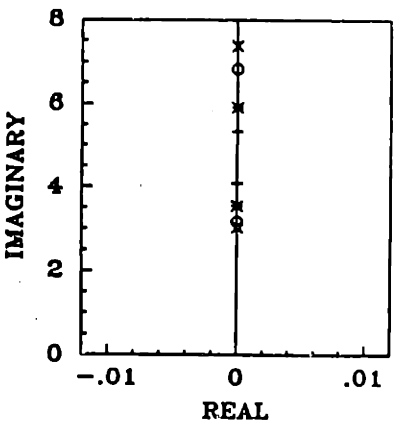


Fig. 3.4b: Close-up of Lower Frequency Range for Different Values of the Elbow Angle

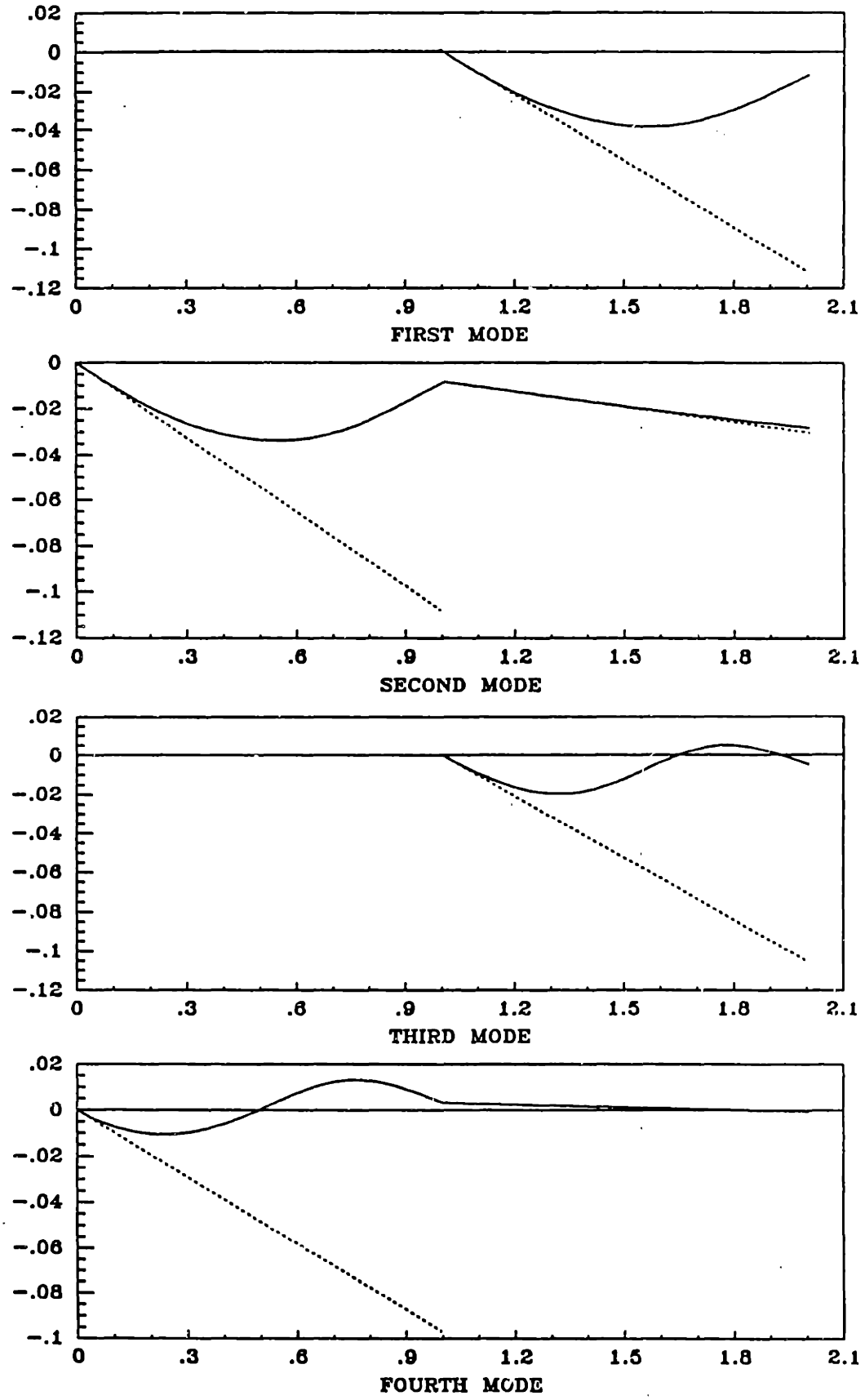


Fig. 3.5 First Four Manipulator Mode Shapes with Joints Free

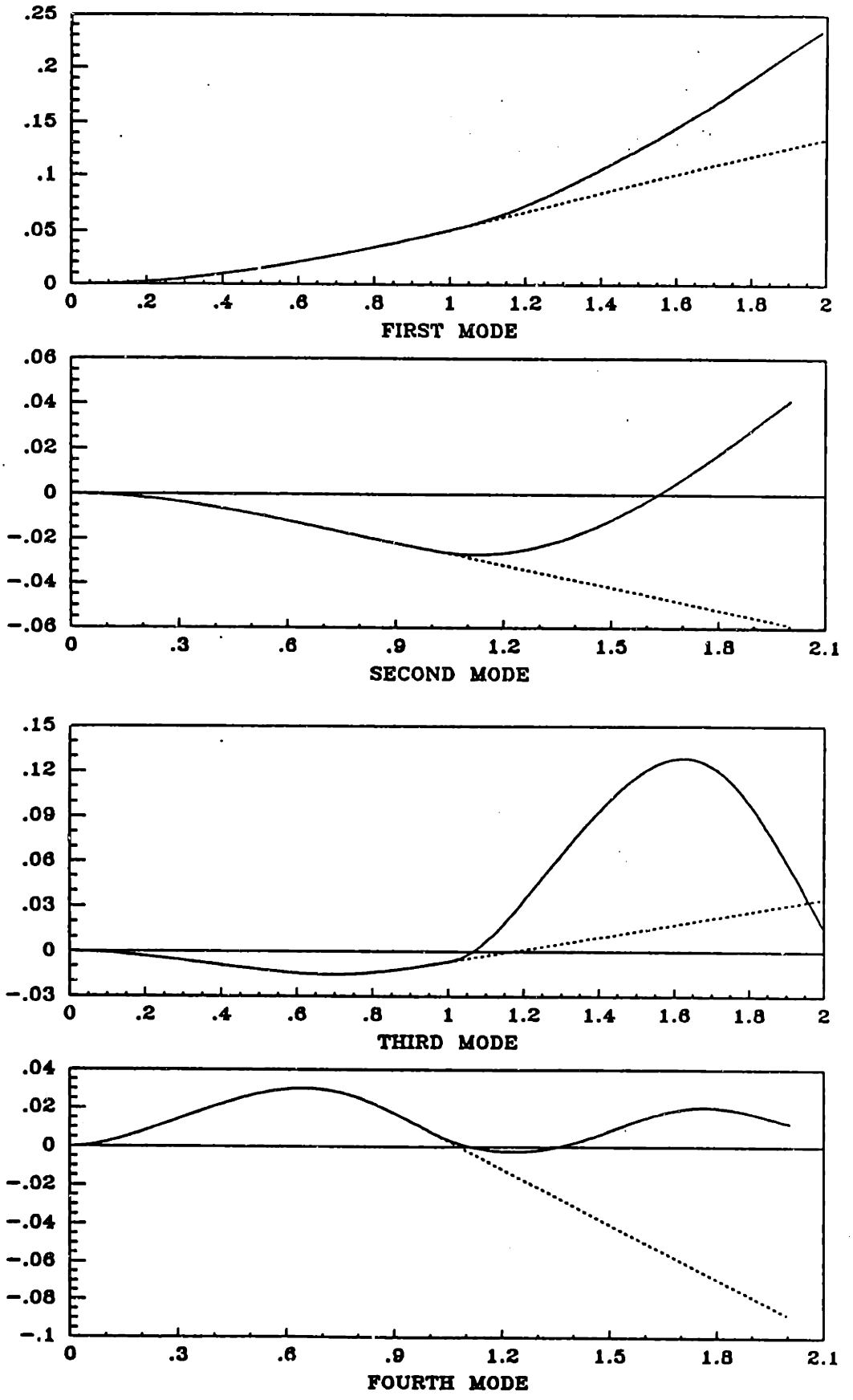
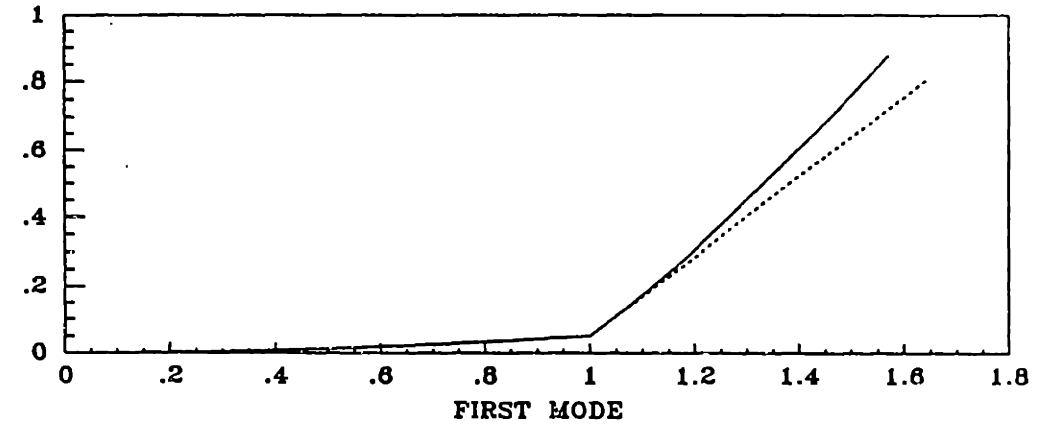
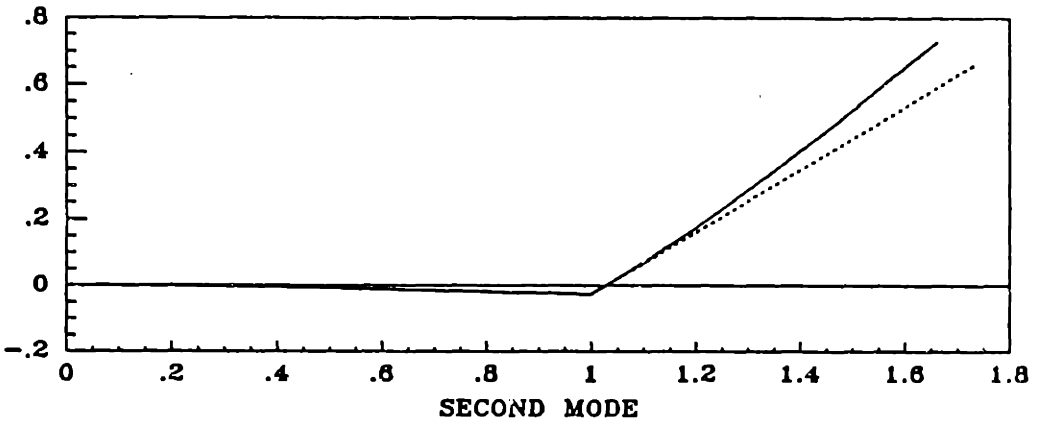
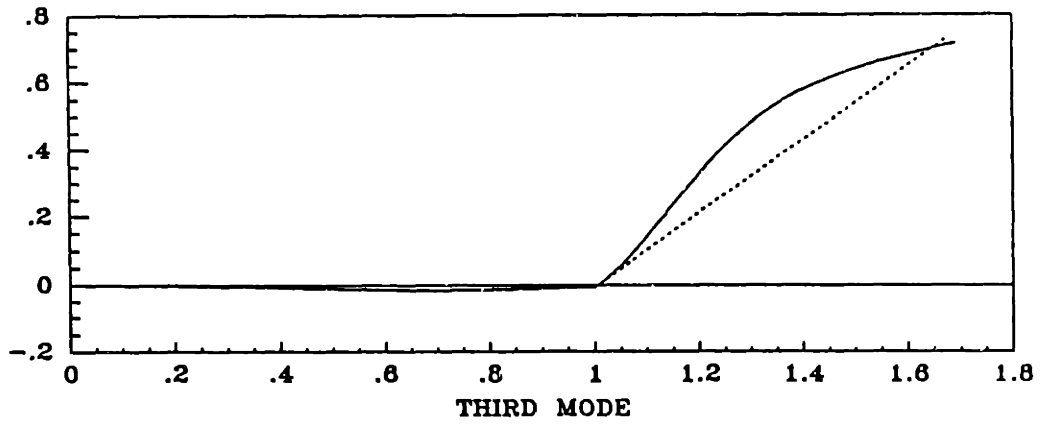


Fig. 3.6: First Four Manipulator Mode Shapes with Joints Locked (β=0 deg)







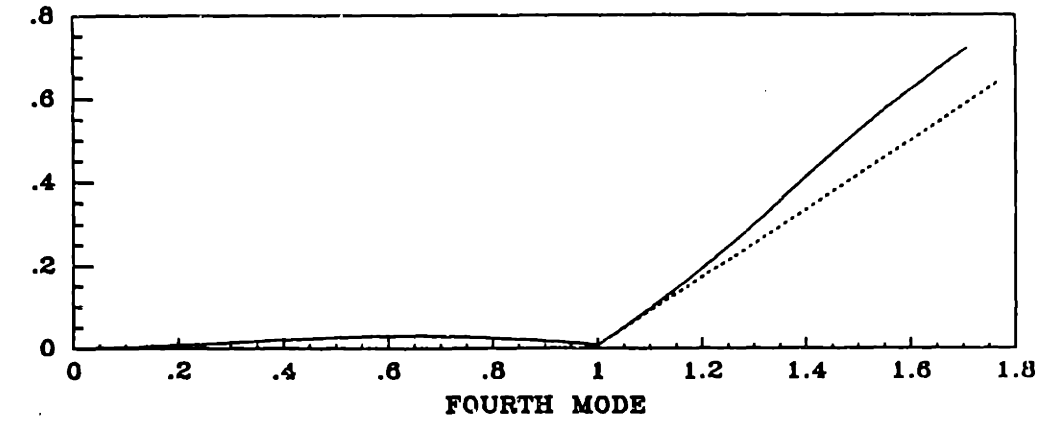


Fig. 3.7: First Four Manipulator Mode Shapes with Joints Locked (β=45 deg)

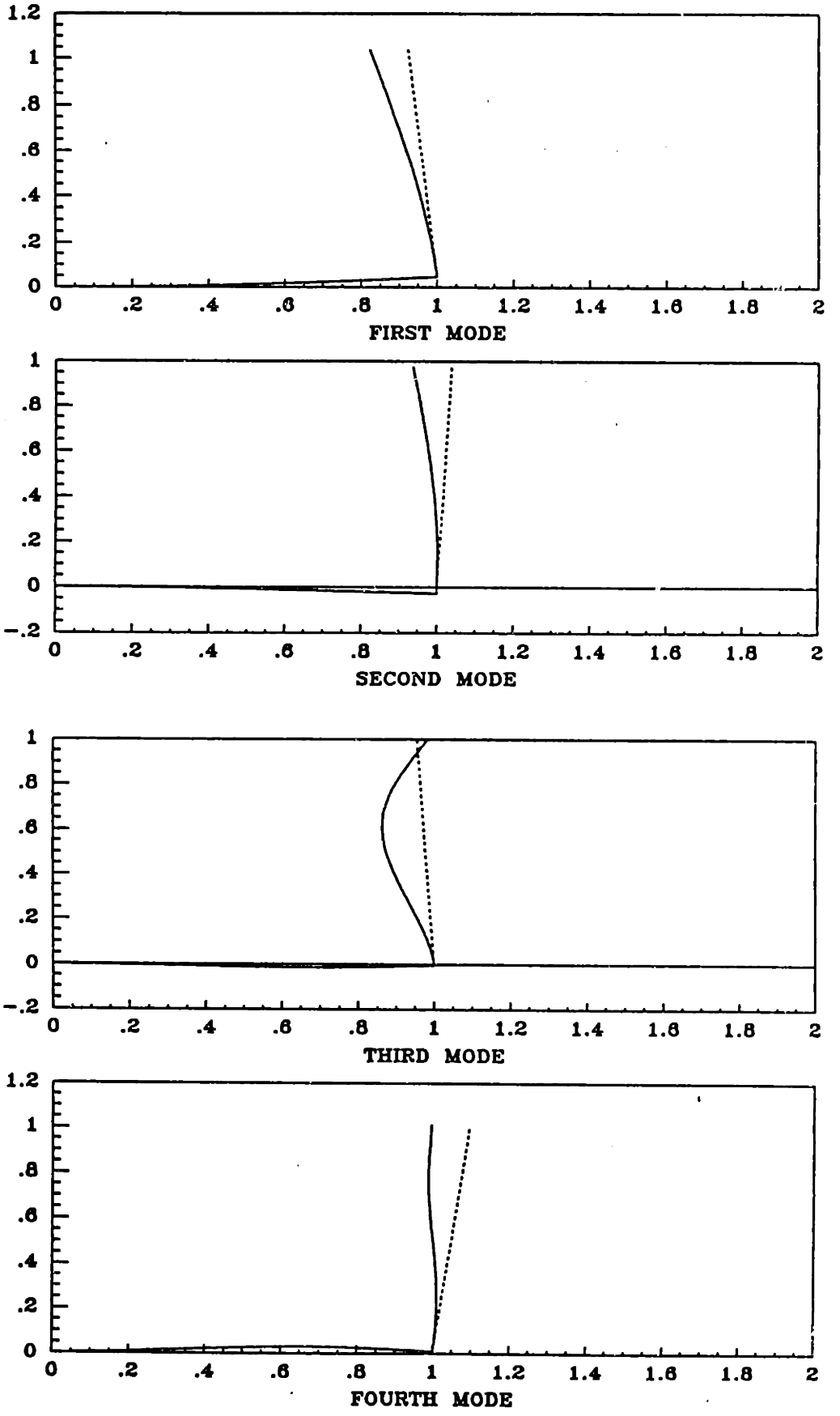


Fig. 3.8: First Four Manipulator Mode Shapes with Joints Locked (β=90 deg)

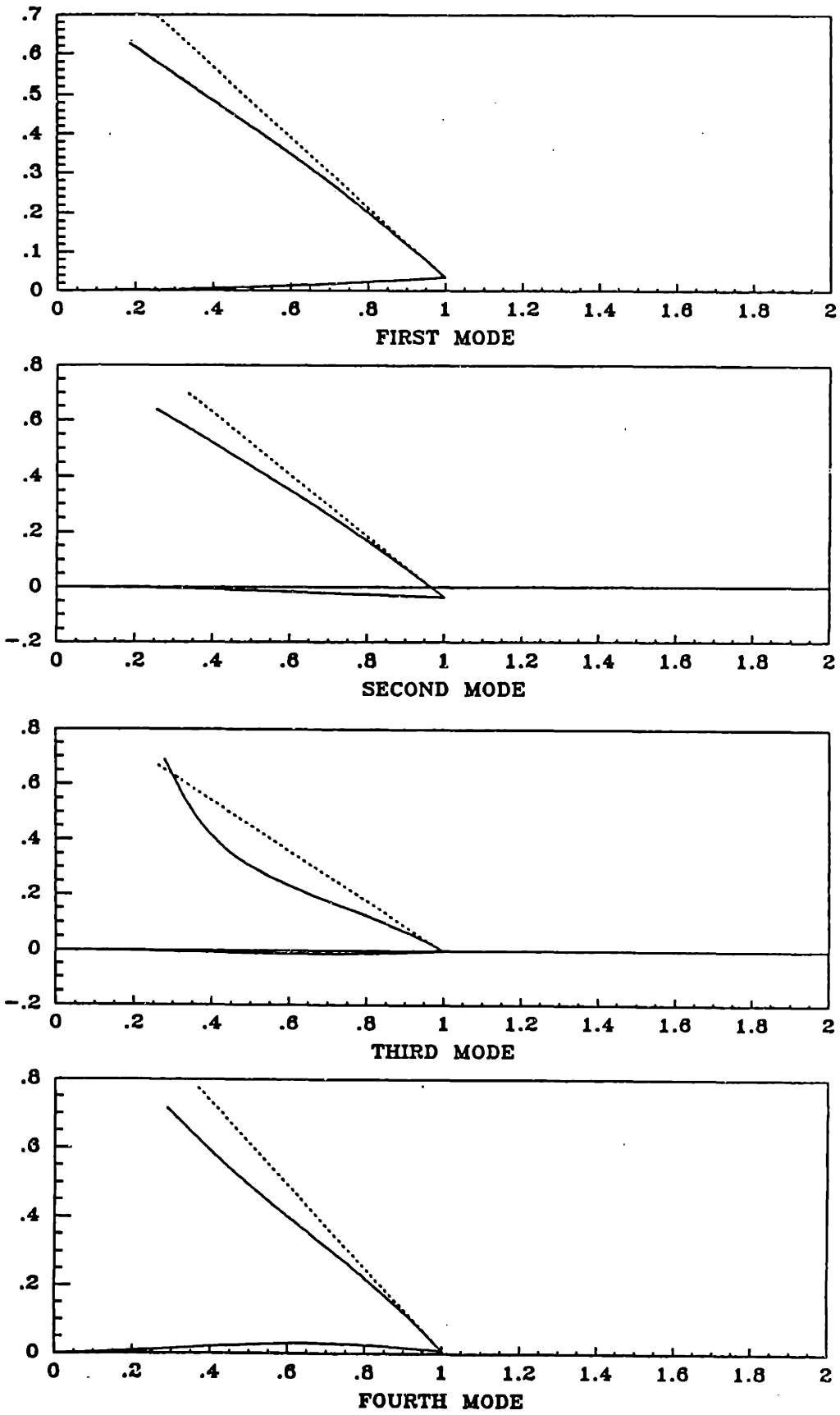


Fig. 3.9: First Four Manipulator Mode Shapes with Joints Locked (β=135 deg)

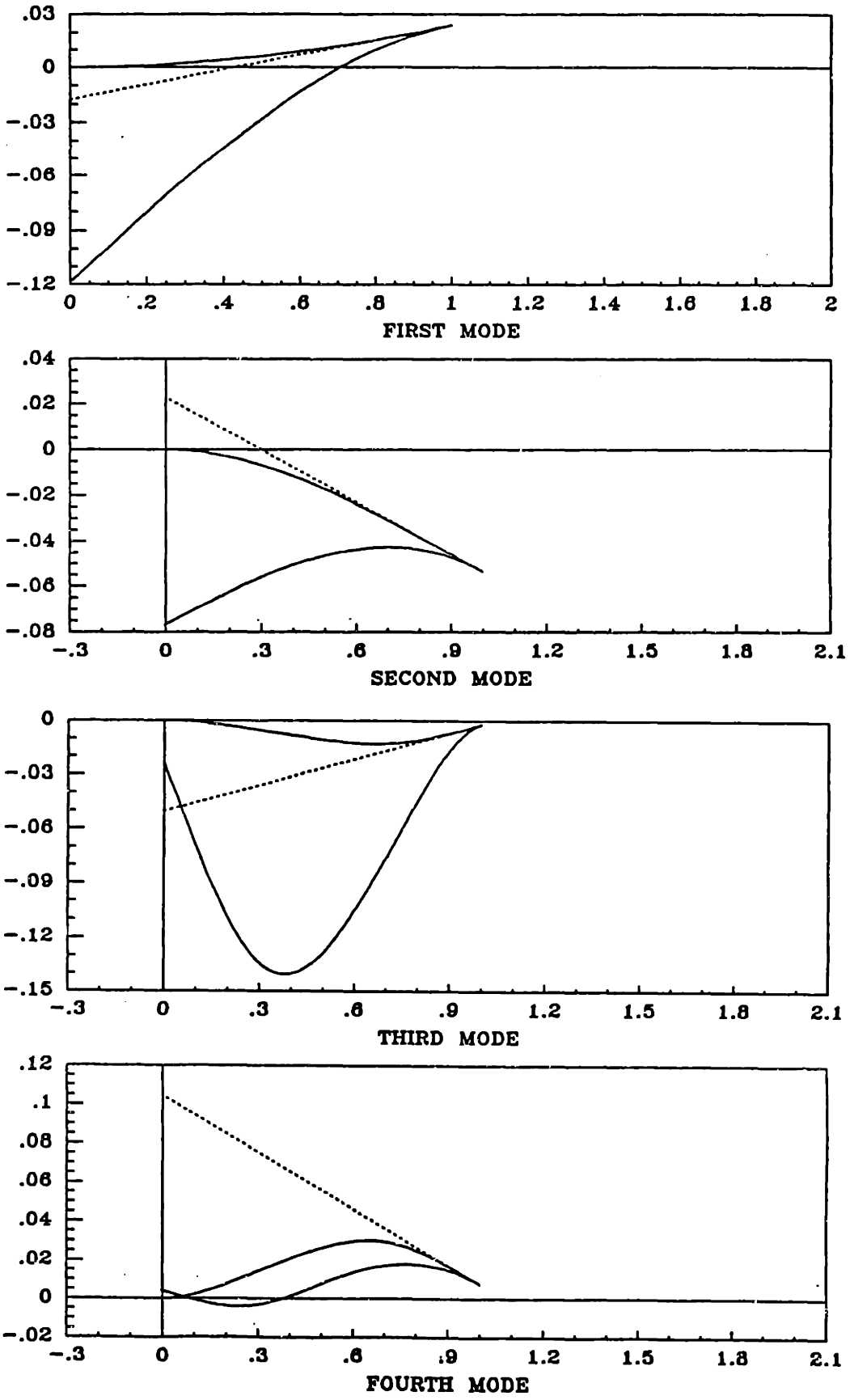


Fig. 3.10: First Four Mode Shapes with Joints Locked (β=180 deg)

CHAPTER 4: SIMULATION RESULTS

INTRODUCTION

In the previous chapter, three levels of increasing simplification of the correctly linearized equations of motion were proposed. Based on the results of the simple one beam example, predictions were made concerning the relative significance of elastic nonlinear terms in the equations of motion of chains of flexible bodies undergoing large rigid body motion, but small elastic deflections. Recall that elastic nonlinear terms are those nonlinear terms in the equations of motion which involve the generalized elastic coordinates and speeds.

In this chapter the same three models are used to test these predictions for the case of the manipulator with two flexible links. The chapter starts with a description of the numerical simulation capabilities. After a brief section on trajectory time-scaling, the simulation results of implementing four different trajectories for each model are presented.

4.1 NUMERICAL SIMULATION

The motion equations for the two-link, flexible, planar manipulator, consistently linearized in small elastic deflections and speeds, were programmed in FORTRAN and implemented in a VAXstation 2000. The code developed for this purpose allows the joint and tip bodies to have mass, moment of inertia, and planar dimensions, just as in the case of the equations developed in chapter two (see Fig. 2.1). A maximum of four cantilever modes per beam are implemented, although only two per link are used in most of the actual simulations. One percent damping ratio is assumed for each cantilever mode to approximately model material damping in the manipulator. The structure of the code is modular for ease of handling and to facilitate expansion. A short separate program handles user interface by creating the input files needed by the main program, which can then be run as time-sharing or in batch mode. This user interface has a short and a long version. The long version allows the user to input the physical manipulator parameters such as mass density, link lengths, modal damping, joint planar dimensions, etc. The short version lets

the user select initial and final times for the simulation, initial configuration of the manipulator, number of modes to be used for each link, initial time step to be tried, and the kind of model to be run, i.e., consistent, inconsistent, or ruthless (see section 3.2).

Actuation is assumed in the form of joint torques, as in chapter two. Torque time histories, which can be open or closed-loop, are created by a subroutine which is linked to the main program. This subroutine has access to the state vector and to the simulation time, to allow for feedback and time-varying forcing schemes. It can also generate computed torques time histories, given desired trajectories. It is in this last capacity that it was used for all of the cases presented below, with the notable exception of the time-optimal bang-bang controller (section 4.3.4).

As can be seen from Appendix C, the configuration dependent mass matrix is non-symmetric due to the linearization. For this reason, the mass matrix inversion needed to obtain the form of equation (2.4.3.3) for the equations of motion is executed using LU decomposition [6] instead of the computationally cheaper Cholesky decomposition. The latter requires symmetric, positive definite matrices. Once the equations are in the form of equation (2.4.3.3), they are integrated in time using a Runge-Kutta fourth-order scheme with adaptive time step [22].

The constants that depend on modal integrals are evaluated by subroutines that use the physical parameters' input file. These subroutines are mode shape dependent and need to be rewritten if other than cantilever mode shapes are desired (see App. B and D). Two routines, ENERGY and AMOMENTUM, provide energy and angular momentum checks. They determine the instantaneous energy and angular momentum of the system and compare it to the change in the corresponding quantity due to forcing and/or damping.

The output of the simulation is of two kinds. "Raw" data is output to the file DATFILE.DAT in a format which can be used by other FORTRAN programs. Data in MATRIXx [16] format is output to MATPLOT.DAT. The latter is used to obtain the plots presented in this chapter using MATRIXx. The "raw" data file is used by another FORTRAN program, POST_PROCESS, which processes it and puts it into the proper form to be used by ANIMATE_ARM, which does a graphics animation in a Vax Workstation screen. The FORTRAN code for the simulation, and ancillary programs and subroutines are included in Appendix D.

The values of the physical parameters utilized for the manipulator simulation were chosen to mimic an actual experimental testbed presently under construction at Martin

Marietta by Dr. Eric Schmitz. Table 4.1 shows the assumed properties for the manipulator (see also Fig. 2.1).

Physical Properties of Planar Manipulator with Two Flexible Links			
Mass of shoulder body (kg)	20.0	Length of link 1 (m)	0.9144
Mass density of link 1 (kg/m)	1.33937	Length of link 2 (m)	0.9144
Mass of elbow body (kg)	14.0		
Mass density of link 2 (kg/m)	0.669685	Other lengths (see Fig. 2.1):	
Mass of tip body (kg)	2.0	b_1 (m)	0.0762
		b_{21} (m)	0.0762
Moments of Inertia (about axis perpendicular to plane):		b_{22} (m)	0.0127
Shoulder body (kgm^2)	0.01	b_t (m)	0.0508
Elbow body (kgm^2)	0.03		
Tip body (kgm^2)	0.01		

Table 4.1: Physical Properties of the Two-link Manipulator

4.2 TIME SCALING OF TRAJECTORIES

In the following sections, reference is made to time-scaled trajectories. Time-scaling of nominal trajectories [7] is achieved by replacing time as the independent variable by the new variable

$$r = \alpha t, \qquad \alpha > 0$$

where α is a constant. When α is greater than one, the trajectory is sped up, while if α is less than one it is slowed down. Notice that with this substitution, the time derivatives of the trajectories become

$$\begin{aligned} \frac{d}{dt}\theta(r) &= \alpha\dot{\theta}(\alpha t) \\ \frac{d^2}{dt^2}\theta(r) &= \alpha^2\ddot{\theta}(\alpha t) \end{aligned} \tag{1}$$

From this it is apparent that rates scale like α, while accelerations, and thus torques, scale like the square of α. These relations are used in the following sections to select a scaling factor that yields a desired maximum value of angular rate, or a given maximum value of torque to obtain desired maximum link tip deflections.

4.3 RESULTS

All the trajectories presented below were run in open loop after the torques had been computed from the inverse dynamics problem (given the desired angular trajectories) assuming rigid links for the manipulator. The exception to this is the minimum-time trajectory of section 4.3.4. Although it was run open loop like the rest, maximum torque bang-bang control was implemented instead of computed torques. For all simulation runs, only two assumed modes per link were used, since this gave adequate results and was computationally much cheaper than running the full four modes per link. With two modes per link, the first two system vibration frequencies were obtained to within three percent of the value obtained using four modes per link.

4.3.1 Spinning of the Outboard Link

The first maneuver tried on the two-link manipulator is the smooth spin-up maneuver of the outboard link. This is done for purposes of comparison with recent published results [11,27], and thus as a form of validation of the correctly linearized equations of motion. For this reason, the same smooth spin-up as [27] is selected

$$\dot{\beta} = \frac{\Omega}{t_f}\left(t - \frac{2\pi}{t_f}\sin\left(\frac{2\pi}{t_f}t\right)\right) \tag{1}$$

where

$$t_f = 15 \text{ sec}$$
$$\Omega = 6 \text{ rad/sec}$$

and $d\beta/dt$ is the elbow angular rate.

The above trajectory results in a constant angular rate at the elbow of 6 rad/sec at the end of 15 seconds. To implement this on the two-link manipulator simulation, the shoulder angle (θ) was clamped, and the inboard link was assumed rigid. Note that the first fundamental bending frequency of the second (outboard) link, with the first (inboard) link rigid and the shoulder angle locked, is 4.67 rad/sec (see Table 3.2). Accordingly, from the discussion in chapter three, summarized in the predictions of section 3.4, it can be surmised that the inconsistent model will fail, while the ruthless model will yield results that are not correct, but more conservative. This is indeed the case as can be observed in the results presented in Fig. 4.1.

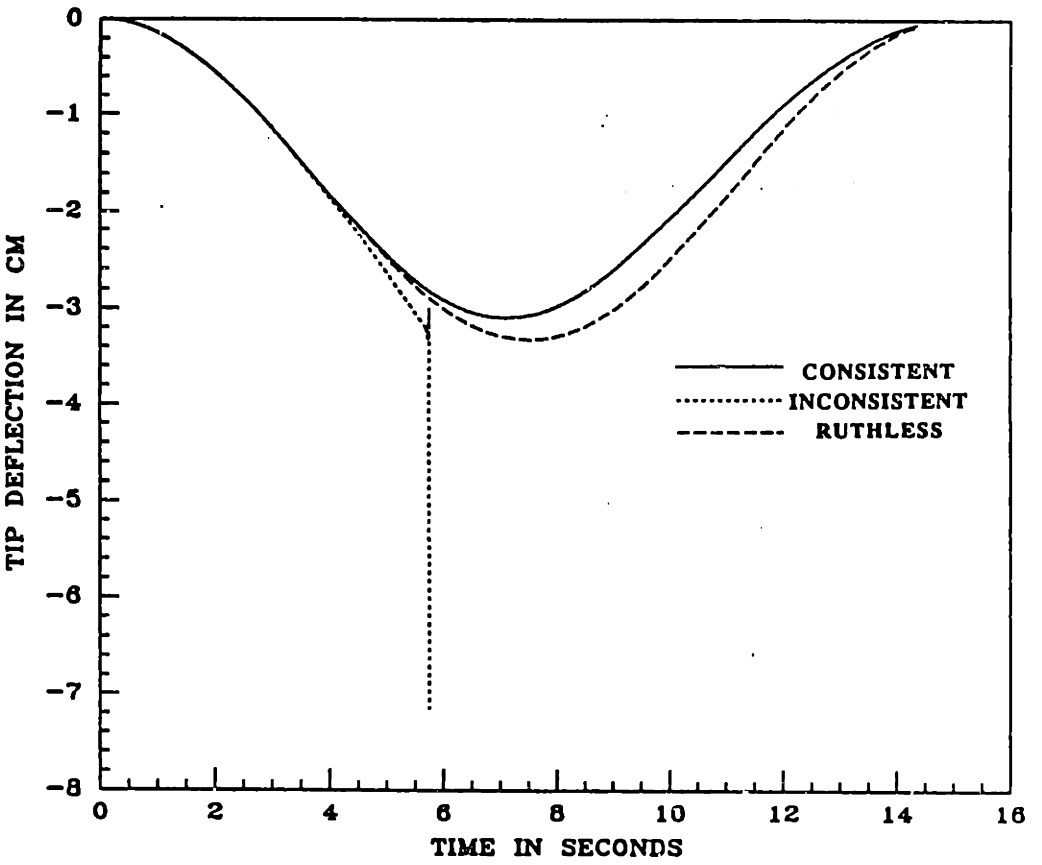


Fig. 4.1: Smooth Spin-up Maneuver for the Outboard Link

It can be seen from the figure that the consistent model yields results that are stiffer than those for the ruthless model. This agrees with the analysis presented in section 3.3.1. The two models yield otherwise qualitatively similar results. The inconsistent model, on the other hand, predicts divergence under the smooth spin-up.

4.3.2 First Smooth Trajectory

In section 3.4 it was predicted that the more severe limit on the validity of other than consistently linearized equations would be the limit on rigid body angular rates. In the case of chains of flexible bodies, as exemplified by the two-link manipulator in this simulation, the nonlinearity arising from dependence on configuration makes it difficult to select a "characteristic" angular rate in the general case. This suggests a case by case approach. With this in mind, the first trajectory tried on the full two-link manipulator is a smooth maneuver intended to mimic as much as possible a single slewing beam. This implies little motion of the relative elbow angle β, and thus small elbow angular rates. In this manner the shoulder angular rate dominates and becomes the "characteristic" angular rate of importance. This provides possibly the simplest way to test the rigid body angular rate limit in the two-link manipulator.

The smooth trajectory is obtained by assuming a form of the angular time histories that is quintic in time. This allows the specification of angle, angular rate and angular acceleration at the initial and final times of the trajectories [5]. Selecting

$$\theta_i = 0 \qquad \beta_i = 0.0873 \text{ rad}$$

$$\theta_f = \pi \text{ rad} \qquad \beta_f = 0$$

$$\dot{\theta}_i = \ddot{\theta}_i = \dot{\beta}_i = \ddot{\beta}_i = \dot{\theta}_f = \ddot{\theta}_f = \dot{\beta}_f = \ddot{\beta}_f = 0$$

$$t_f = 8.0 \text{ sec}$$

the coefficients for the quintic polynomial in time are obtained from the relations

$$a_1 = \dot{\theta}_i$$

$$a_2 = \frac{\ddot{\theta}_i}{2}$$

$$a_3 = \frac{20\theta_f - 20\theta_i - (8\dot{\theta}_f + 12\dot{\theta}_i)t_f - (3\ddot{\theta}_i - \ddot{\theta}_f)t_f^2}{2t_f^3}$$

$$a_4 = \frac{30\theta_f - 30\theta_i + (14\dot{\theta}_f + 16\dot{\theta}_i)t_f + (3\ddot{\theta}_i - 2\ddot{\theta}_f)t_f^2}{2t_f^4}$$

$$a_5 = \frac{12\theta_f - 12\theta_i - (6\dot{\theta}_f + 6\dot{\theta}_i)t_f - (\ddot{\theta}_i - \ddot{\theta}_f)t_f^2}{2t_f^5} \qquad (1)$$

with analogous relations for the relative elbow angle (β). The desired trajectory results

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

$$\beta(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5 \qquad (2)$$

Fig. 4.2 shows the computed torques for the nominal trajectory scaled by a factor of two (α=2.0).

Figs. 4.3 to 4.5 show the results of the three time-scaled trajectories tried. The maneuvers were scaled to vary the maximum value of the characteristic rigid body rate, i.e., the shoulder angular rate. The first fundamental bending frequency of the arm, in extended configuration and with joints locked (β=0) is approximately three radians per second (see section 3.5). The first case shown exhibits low shoulder angular rate relative

to the fundamental ($d\theta/dt$ about 0.2 rad/sec at maximum). As expected, the elbow rate is very small. For this case, all three models exhibit excellent agreement, as expected from the discussion in chapter three.

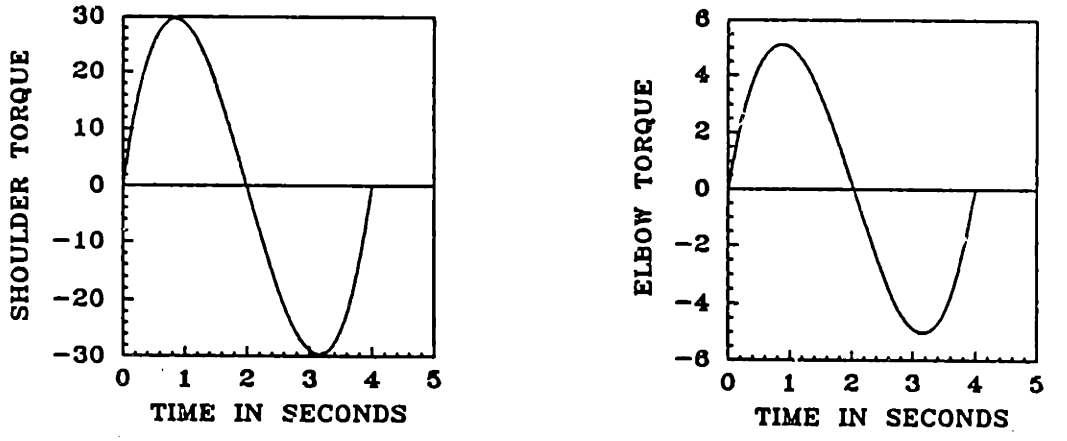


Fig. 4.2: Computed Torques for First Smooth Trajectory (Nm)

In the second case, the nominal trajectory was scaled down by a factor of 0.4297. In this case the shoulder angular rate is about 0.3 rad/sec at maximum, or ten percent of the first fundamental bending frequency of the system. This is the limit of accuracy of the inconsistent model, if the predictions of section 3.4 are correct. From Fig. 4.4 it is clear that while ruthless and consistent models still agree very well, the inconsistent model exhibits a high frequency "chattering." This was deemed a sort of onset of ill-conditioning, since the chattering seems to be numerical in nature. As shall be seen for the last case tried, for shoulder rates higher than this limit (ten percent of the fundamental), the inconsistent model fails altogether.

Figure 4.5 shows the last time-scaled trajectory, where the nominal trajectory was sped up (α=2.0). For this case, the shoulder angular rate is high, about half of the first fundamental vibration frequency of the manipulator. In this case, the plots show only the ruthless and consistent models, since the inconsistent model has failed. While good agreement is still evident between the two remaining models, a notable discrepancy is apparent in the elbow angle and elbow angular rates. This is to be expected since the "characteristic" rate is a significant fraction of the fundamental, and from the discussion in chapter three, it would be expected that the ruthless be valid accurately only for very low rigid angular rates compared to the fundamental. Note that the nominal trajectory cannot be sped up anymore without violating the small elastic deflection assumption, since for this case deflections of ten percent of the length of the second link are already in evidence.

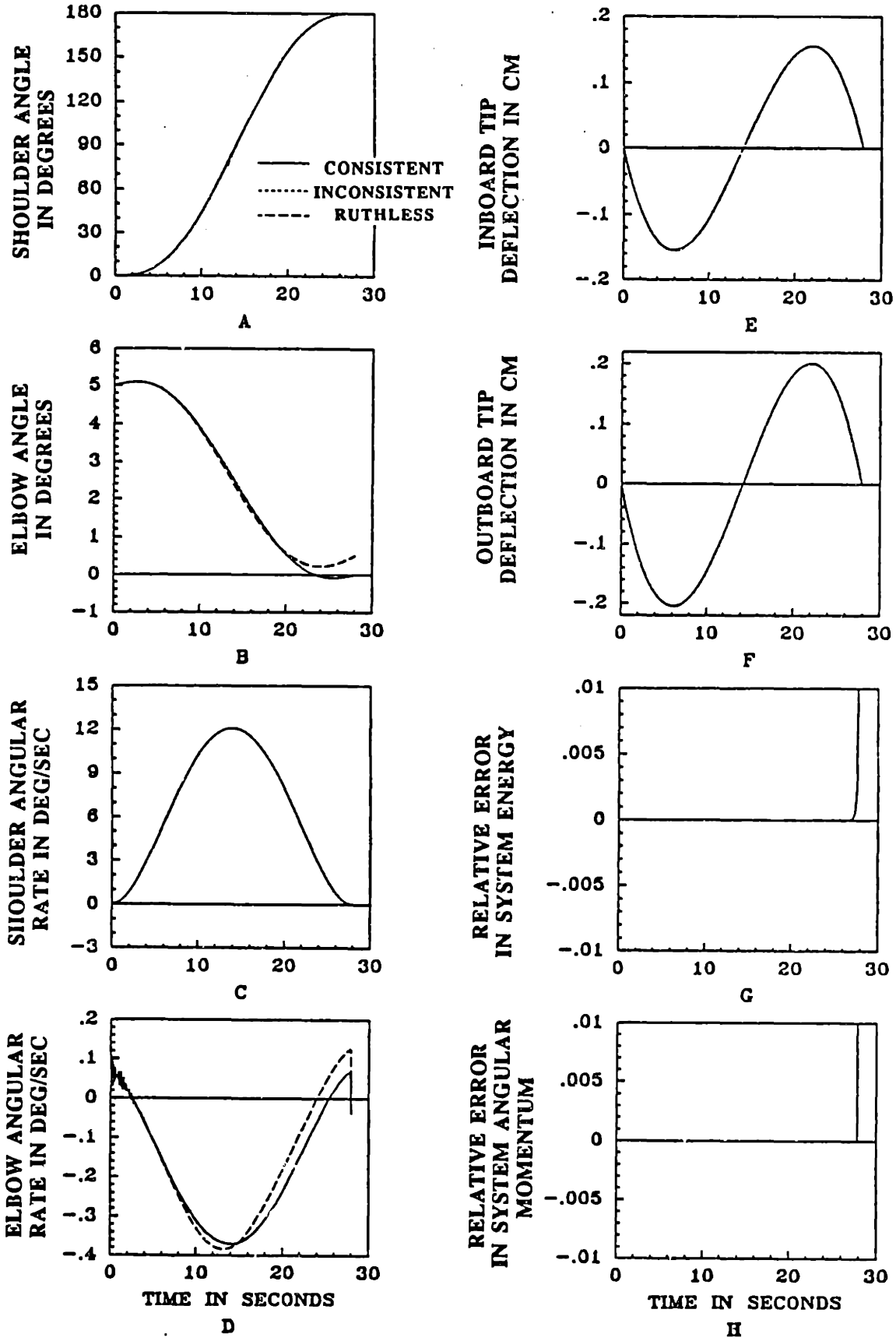


Fig. 4.3: First Smooth Trajectory with α=0.2865

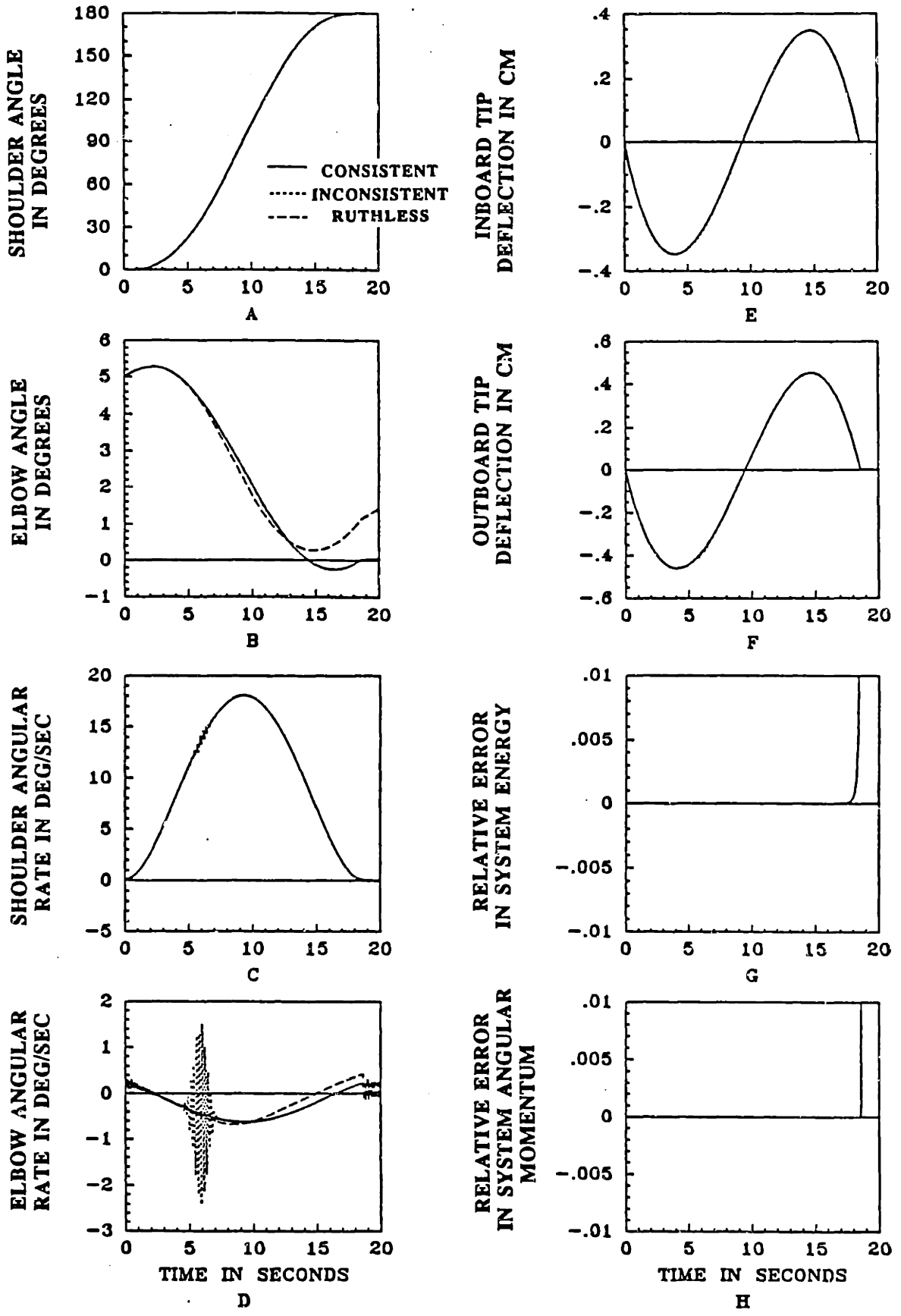


Fig. 4.4: First Smooth Trajectory with α=0.4197

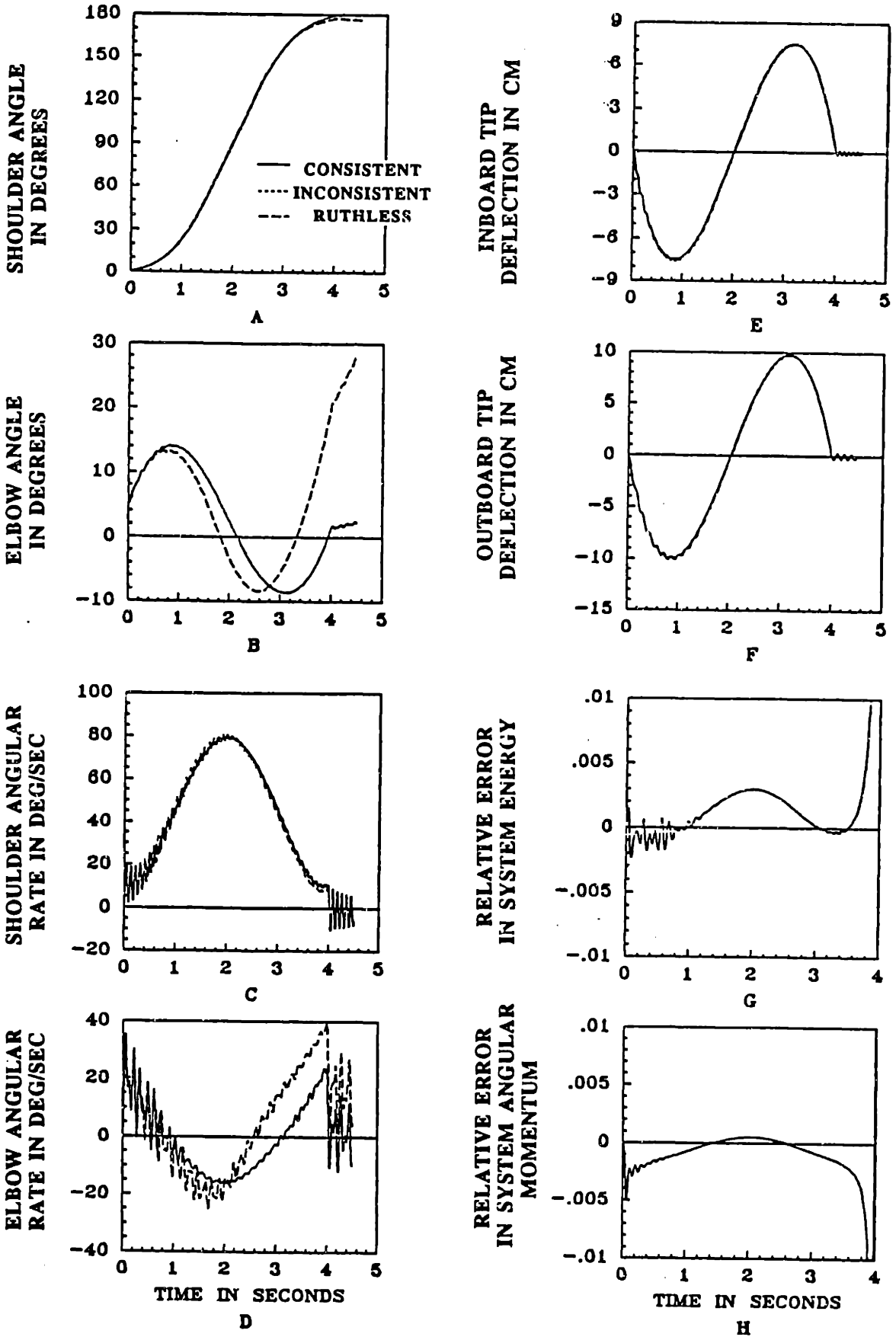


Fig. 4.5 First Smooth Trajectory with α=2.0

It is interesting to note that the inconsistent model fails whenever the shoulder angular rate is higher than ten percent of the fundamental, much earlier than predicted by the single beam example of chapter three. This could be due to the exacerbation of an ill-conditioning of the mass matrix due to the lack of the foreshortening terms. This situation could be the result of nonlinear interactions between the links that are absent in the one link example. For a more complete discussion of ill-conditioning in the simulation, see the discussion in section 4.4.

Finally, it is worth pointing out the excellent agreement in the tip deflections for both links, in all three cases. This is probably due to the fact that the equations of motion (see App. C) are elastically decoupled, and, while inertially coupled, the elastic degrees of freedom mass matrix (M_{EE} of section 3.1) does not depend on elastic nonlinear terms due to linearization. Also, the agreement in shoulder angle and angular rates is remarkable. This indicates that, depending on what state variable is of interest in a given trajectory, the ruthless model will be as good as the more cumbersome consistent model. In all cases, the ruthless is more conservative and better conditioned than the inconsistent model.

4.3.3 Second Smooth Trajectory

The second trajectory tried on the two-link manipulator is intended to examine the relative significance of the nonlinear elastic terms in the equations of motion of a chain of elastic bodies as a function of trajectory. In the previous section, the shoulder-rate dominant slew maneuver confirmed the predictions of section 3.4. It is now desired to see how these results are affected by a change in trajectory. For this purpose, a typical "deployment" maneuver of the two link manipulator is commanded.

As in the previous section, the smooth trajectory is obtained by assuming a form of the angular time histories that is quintic in time. Selecting

$$\theta_i = -1.570796 \qquad \beta_i = 2.96706 \text{ rad}$$

$$\theta_f = 0.2617994 \text{ rad} \qquad \beta_f = 0.8726646 \text{ rad}$$

$$\dot{\theta}_i = \ddot{\theta}_i = \dot{\beta}_i = \ddot{\beta}_i = \dot{\theta}_f = \ddot{\theta}_f = \dot{\beta}_f = \ddot{\beta}_f = 0$$

$$t_f = 4.0 \text{ sec}$$

and solving for the coefficients of the polynomial in time as in the previous section, the trajectory results in

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$
$$\beta(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3 + b_4 t^4 + b_5 t^5 \qquad (1)$$

Figure 4.6 shows the computed torques for the nominal trajectory.

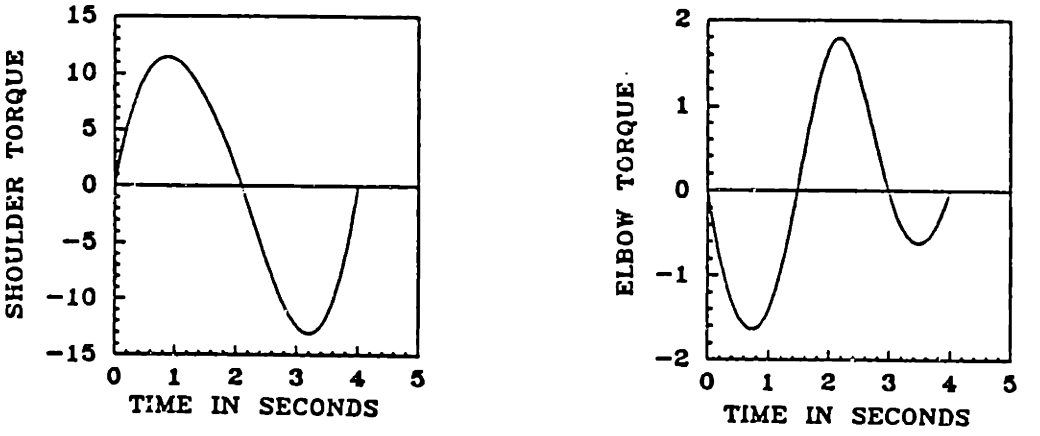


Fig. 4.6: Computed Torques for Second Smooth Trajectory (Nm)

Figure 4.7 shows the nominal trajectory. In this figure, plots for both the ruthless and the consistent models have been overlaid. The inconsistent model fails for this case. This is not surprising after examining the results of the previous section and noting that the angular rates for the nominal maneuver are as large as thirty percent of the "fundamental vibration frequency." From table 3.1 in section 3.5, the system frequencies for the manipulator with locked joints are seen to fluctuate from 3 rad/sec to 4 rad/sec for corresponding elbow relative angles of zero to 135 degrees. For this reason, in the case at hand reference is made to "one" fundamental frequency, and it is assumed that it lies in the range specified above and is about 3.5 rad/sec. For the two models shown, the agreement is again excellent for the shoulder angle and tip deflections, with the elbow angular position being off by only a maximum of ten percent relative error.

In the second case considered, the nominal maneuver is slowed down until the inconsistent model does not fail (α=0.1685). Figure 4.8 shows that for this case, all three models yield identical results. This confirms the predictions in section 3.4, and further suggests that within the limit of validity of the inconsistent model, the ruthless is as good as the inconsistent, and actually better since it is much easier to obtain. Note that the angular rates are well within the ten percent of the fundamental mark.

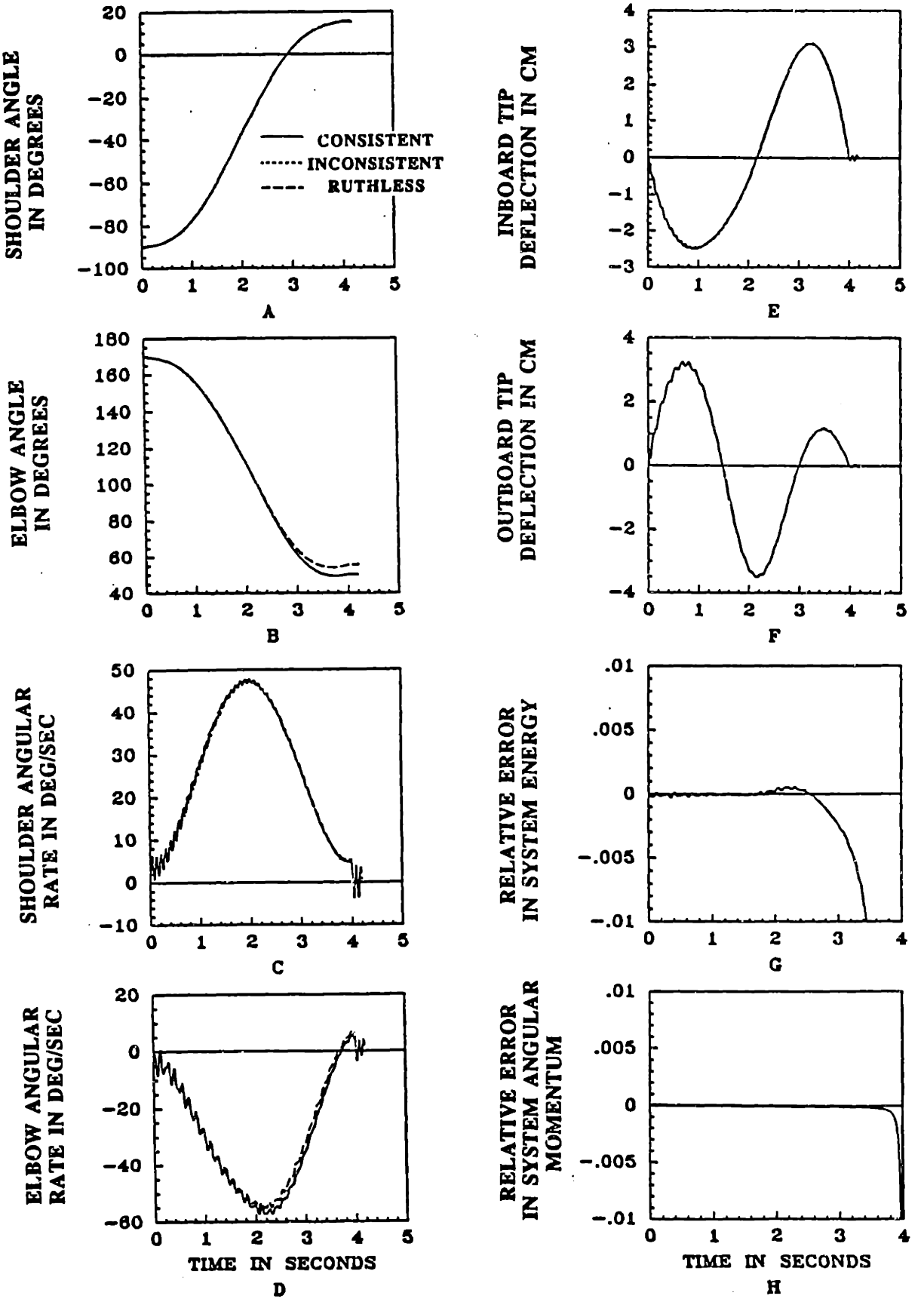


Fig. 4.7: Second Smooth Trajectory with α=1.0

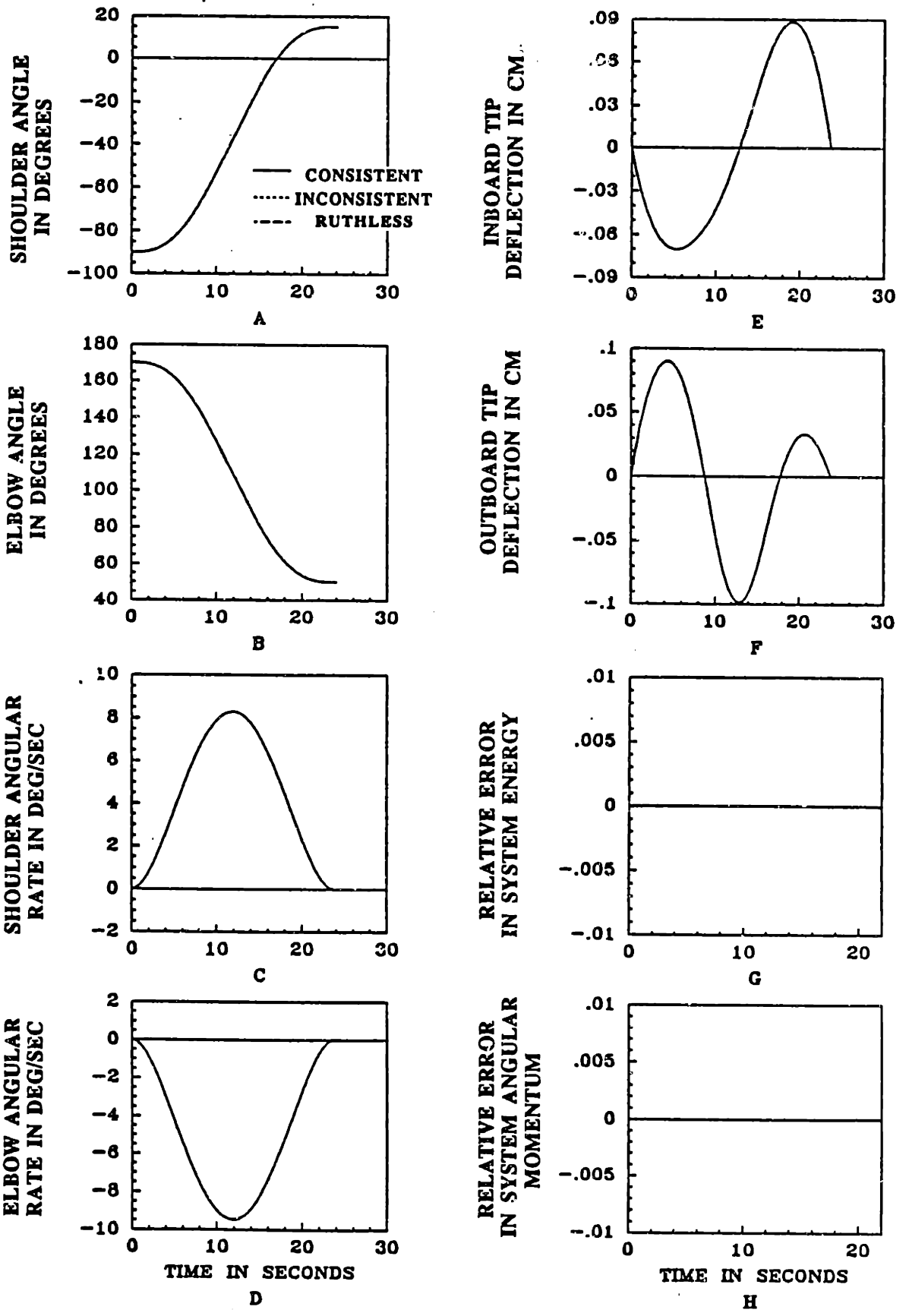


Fig. 4.8: Second Smooth Trajectory with α=0.1685

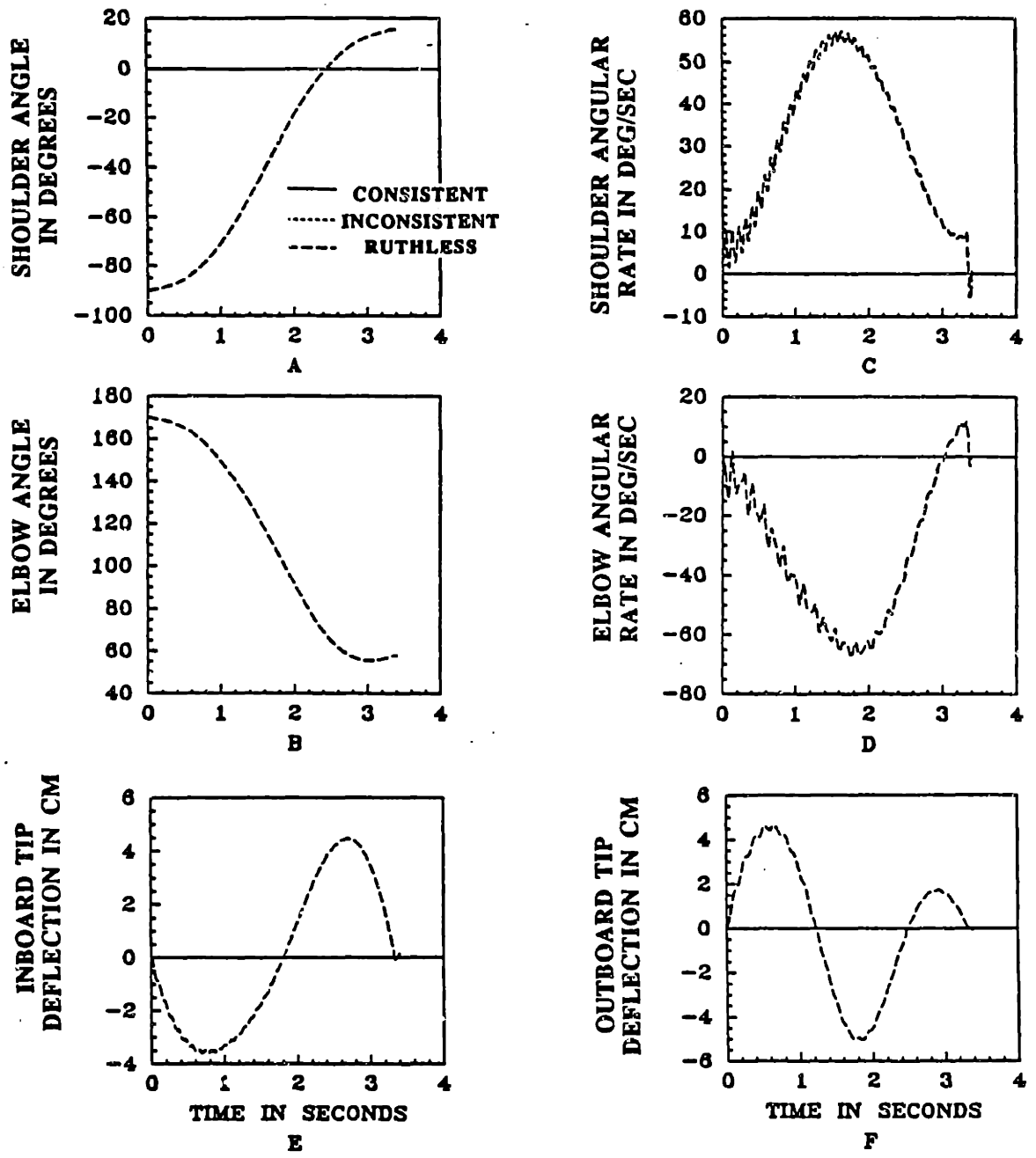


Fig. 4.9: Second Smooth Trajectory with α=1.2

The last case considered consists of the nominal trajectory scaled upwards in time (α=1.2). As in the previous section, it was desired to reach the "hard" simulation limit of link tip deflections of about ten percent of the link lengths. As Figure 4.9 shows, only the ruthless model did not fail under the given speed-up of the trajectory, even though tip deflections should only be about four percent of link lengths. To investigate these results, the stability boundary of section 3.3.3 was implemented in the simulation and checked at every time step to determine whether the simulation failure of the consistent model was due to the correct prediction of divergence, or due to some numerical or other phenomenon. For the case of two assumed modes per link, the lowest eigenvalue of the effective stiffness matrix (see section 3.3.3) was actually checked, with the conclusion that no divergence was predicted by the analytical consistent equations. Section 4.4 offers further comments on this problem.

In summary, the above results again show a strong correlation between the limit of validity of the inconsistent model and the maximum values of angular rates. Contrary to the case in section 4.3.2, however, for the present maneuver no "characteristic" rigid body rate is apparent. The limit at which the inconsistent model fails seems to be even before any of the two angular rates (shoulder or elbow) reach ten percent of the fundamental vibration frequency. A strong point can still be made, nevertheless, in that the ruthless model is as good as the inconsistent whenever the inconsistent is valid, and more conservative since the ruthless model does not fail. Even at high angular rates the ruthless model yields results that are quantitatively very close to the consistent model results.

4.3.4 Minimum-Time Trajectory

The last trajectory attempted on the two-link manipulator is a minimum time trajectory executed via bang-bang controls [17]. The idea behind implementing this maneuver is to get a feel for how the results of the previous two sections change when a "typical" fast maneuver is commanded with abrupt forcing. For comparison purposes, the torque bounds for the minimum-time problem are set to the maximum values of the torques obtained for the nominal first smooth trajectory (section 4.3.2) scaled by a factor of α=0.4297 (see Fig. 4.4). The actual forcing time history for the manipulator simulation is adapted from that presented by Meier and Bryson [17], to fit the chosen arm parameters and configuration.

The maneuver chosen was that referenced in [17] as one D/L=3.9, where L is the link length, assumed the same for both links, and D is the distance traveled by the manipulator end effector. The switching times for shoulder and elbow torques, and the final time are given as

$$t_{s1} = 1.13 \qquad t_{e1} = 1.46$$
$$t_{s2} = 1.44 \qquad t_f = 2.91$$
$$t_{s3} = 1.77$$

with initial angular values of

$$\theta_i = 0 \qquad \beta_i = -0.21 \text{ rad}$$

where it has been assumed that the normalized torque bounds are

$$\frac{T_e}{mL^2} = \frac{T_s}{mL^2} = \pm 1$$

where m is the link mass, also assumed the same for both links and the same for the tip load. Elbow mass and the moments of inertia of shoulder, elbow and tip effectors are ignored. For the case at hand , the switch times were scaled to obtain the proper trajectory taking into account elbow and tip masses, moments of inertia of all bodies, and different link masses. The resulting trajectory is shown in Fig. 4.10 for the case of both links rigid.

The results are presented in Fig. 4.11. As in the previous two sections, the inconsistent model fails for angular rates larger than about ten percent of the fundamental vibration frequency of the system (in this case about one third of the fundamental at maximum). For this reason, only the consistent and ruthless models are compared. As in section 4.3.2, the ruthless model yields results that are within ten percent relative error of the consistent results, and this for relatively large angular rates. The interesting thing to note is the high frequency oscillation induced in the flexible links by the bang-bang controller. Recall that all the maneuvers are run open loop, and that one percent modal damping was assumed for the cantilever mode shapes.

The large discrepancy shown in the energy and angular momentum balance plots is probably due to round-off error accumulation due to the relatively large tip deflections and high frequency excitation.

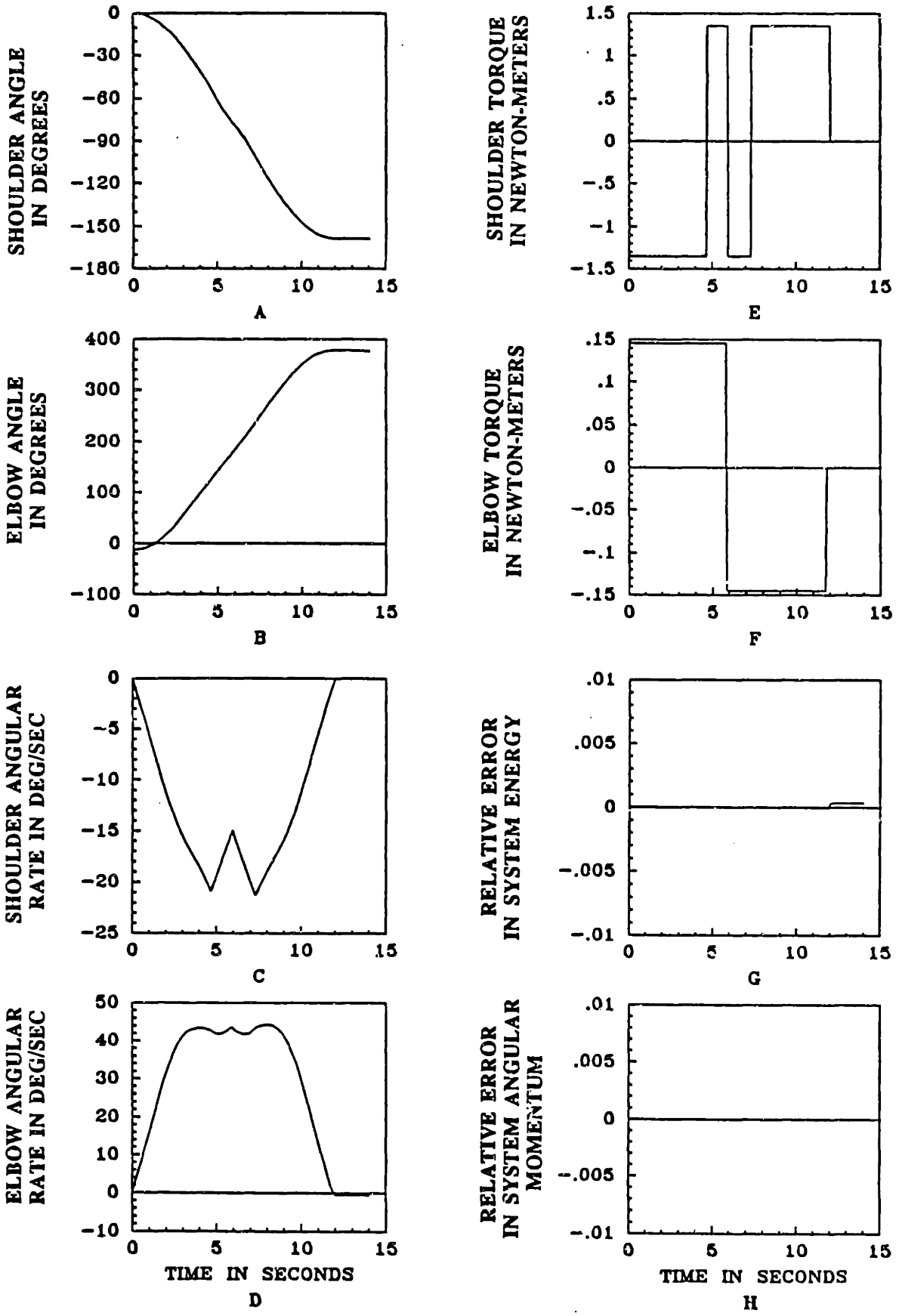


Fig. 4.10: Time-Optimal Trajectory for Two-link Rigid Manipulator

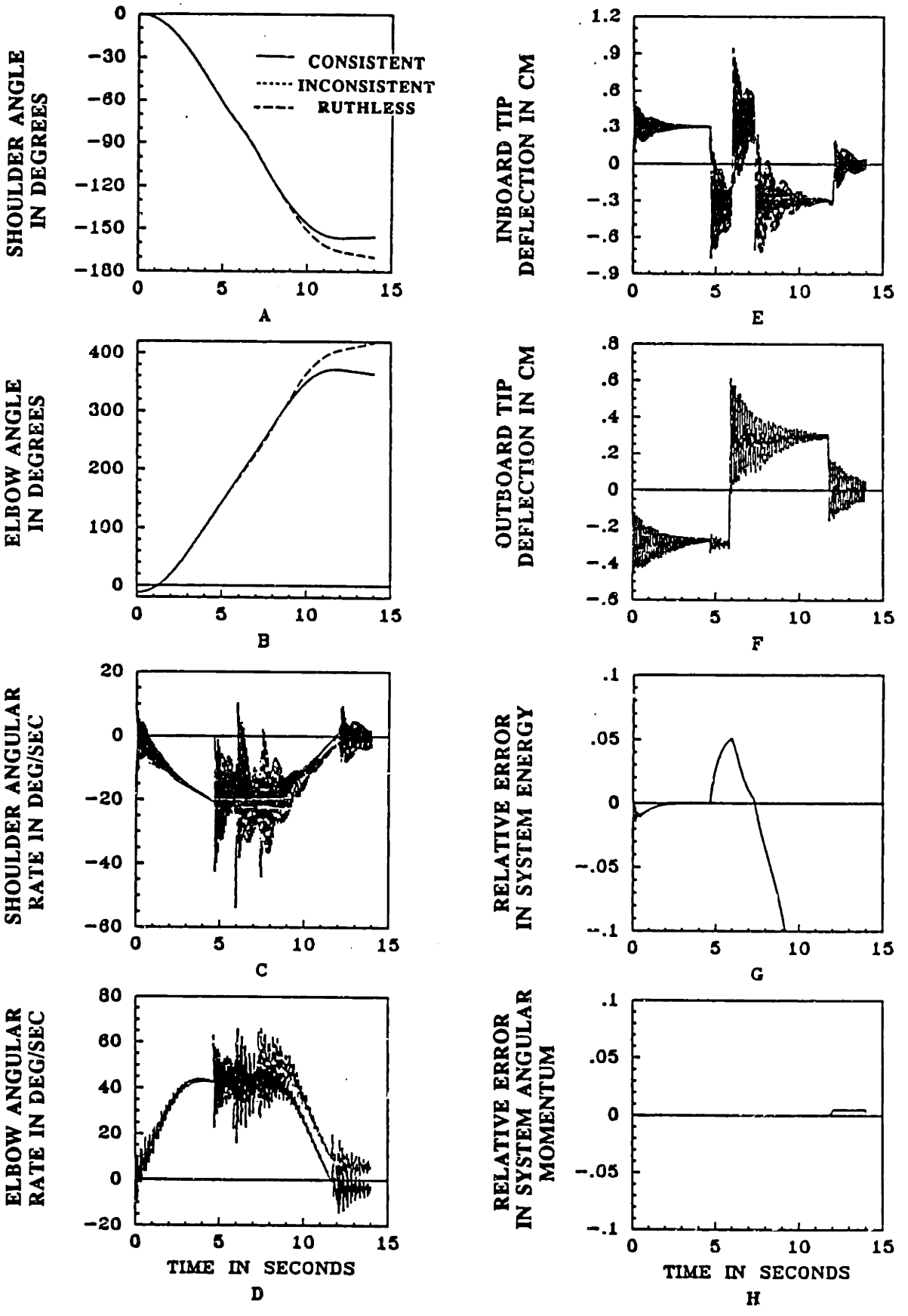


Fig. 4.11: Time-Optimal Trajectory for Two-link Flexible Manipulator

4.4 DISCUSSION

Having determined in section 4.3.3 that a numerical problem existed that might be responsible for some of the unexpected results in section 4.3, a more detailed examination of this problem is in order. The possibility of a programming error was ruled out to a large extent through thorough debugging and the use of energy and angular momentum checks. Numerical experiments determined that the problem was caused by ill-conditioning of the mass matrix brought about somehow by its dependence on the elastic coordinates. This suggests that this could be the reason behind the premature failing of the inconsistent model in the previous section. Numerical ill-conditioning of the mass matrix implies that at some configurations the configuration dependent mass matrix is numerically singular. A singular mass matrix indicates massless degrees of freedom being modelled. The problem therefore seems to come down to modelling error.

It appears that at least two ways of solving the problem are available. One way is to solve the problem numerically. This is most appealing at first glance since it entails keeping all the developed software. On the other hand, as shall be seen, it requires a much more computationally intensive algorithm to implement. Also, the second possible solution retains all of the derived equations of motion (and thus most of the software), while requiring changes to only a few mode shape dependent subroutines.

The first solution calls for determining at every time step, and thus for every configuration, if there are any massless degrees of freedom. This is achieved doing a singular value decomposition (SVD) of the mass matrix. If a near zero singular value is detected, a least squares projection is used to obtain the optimal (in a least squares sense) inversion of the mass matrix so that the acceleration vector thus obtained does not grow without bounds [22]. This is to some extent the equivalent of Guyan reduction performed on the mass matrix after it has been transformed into a space where the massless degrees of freedom are apparent. The problem with this solution is that it is very expensive computationally.

The other solution requires realizing that the selected assumed mode shapes are not appropriate for modelling the manipulator. In chapter two it was explained that cantilevered-free mode shapes were chosen because they simplified the equations of the two link arm. Indeed these mode shapes are easier to include in the development of the motion equations than the pinned, system or unconstrained mode shapes [25]. Unfortunately, the free end condition forces the cantilever mode shapes to have zero second

and third derivatives with respect to beam length evaluated at the tip of the link. In other words, these mode shapes do not support moment or shear at the "free" end of the link. In the manipulator, both links end in masses with moments of inertia, and are therefore not free. The problem here might be equivalent to the mode shapes selected for the system not being "complete" in the strain energy norm [28]. This then suggests that mode shapes for a cantilever beam with a tip body, with mass and moment of inertia, be used as the beam mode shapes for the determination of modal integrals in the equations of motion.

Both these possible solutions are being pursued at present and are offered as suggestions for further work.

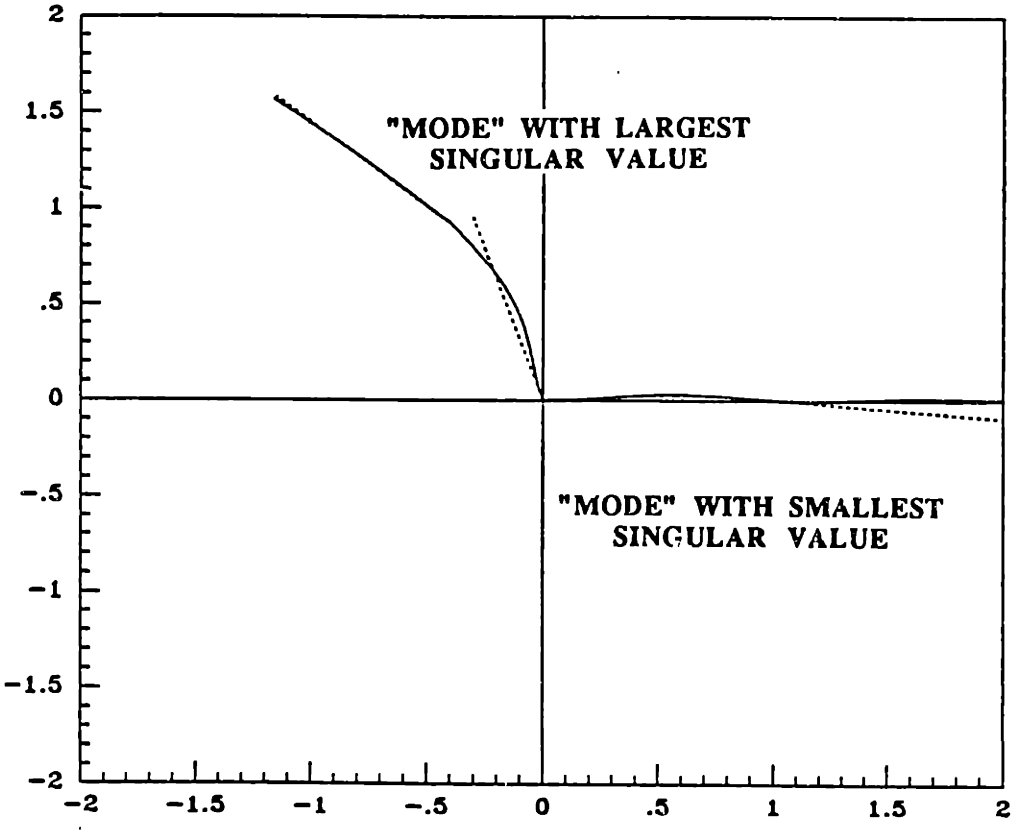


Fig. 4.12: Comparison of Right Singular Vectors for the Mass Matrix at Failure

It is possible that the "free play" allowed at the link tips by the cantilever mode shape assumption provides the connection between the two seemingly different approaches presented above. As a matter of fact, velocity "mode shapes" of the mass matrix at failure corresponding to near zero singular values suggest that it is the combination of positive angular rate with negative link tip velocity that yields a configuration that has little or no kinetic energy while the chosen state variable rates are nonzero. Fig. 4.12 shows these

velocity mode shapes (actually right singular vectors) for the largest and smallest singular values of the mass matrix at failure. The mode shapes have been normalized so that the largest tip deflection of either link be ten percent of the link lengths, which have been normalized to one. For the physical manipulator parameters chosen (see Table 4.1), the elbow is massive in comparison to link or tip masses. Therefore it is clear that for the mode with near zero velocity of the elbow joint the kinetic energy is near zero when compared to the kinetic energy for the largest singular value mode. Thus the mass matrix is numerically singular and this results in massless degrees of freedom being modelled.

CHAPTER 5: CONCLUSIONS

The equations of motion for a planar, two-link, flexible manipulator, fully nonlinear in rigid body motions and rates, but linearized in small elastic coordinates and speeds, were obtained. These motion equations, developed via the Kane formalism, were correctly linearized through appropriate use of nonlinear strain-displacement relations. The links were modelled as Bernoulli-Euler beams and the elastic deflections were expanded in terms of cantilever beam modes. The resulting extensive equations were coded in FORTRAN and implemented as a simulation that allows the determination of the manipulator dynamic response to desired forcing functions in the form of joint torques. The simulation allows for a variable number of modes to be used in the modelling of the link flexibility so that effects such as spillover can be tested in control experiments. Modularity in the software developed permits the mode shapes to be changed with changes to two ancillary subroutines that determine modal constants. Further, elastic nonlinear terms in the motion equations were "flagged" in the simulation. This was done to permit running of three different models, characterized by increasing simplification of the equations, in order to test the relative significance of these nonlinear terms.

Having looked into the general form of the linearized dynamics equations for chains of flexible bodies undergoing large rigid body motions, but small elastic deflections, it was concluded that some terms cannot be obtained through the use of linear strain-displacement relations. These terms were seen to be critical in the simple rotating beam example as they provide the geometric stiffness terms necessary to obtain physical results. The absence of these terms in inconsistently linearized equations limits their validity to angular rates lower than approximately ten percent of the first fundamental vibration frequency of the system. Other terms which are unavailable through the use of linear strain-displacement relations predicted (correctly) buckling under large axial accelerations. The fact that these terms are unobtainable for the general case of an arbitrary flexible body prompted considerations of possible simplifications to the general motion equations.

The two alternative models studied, the ruthlessly linearized model and the inconsistent model, are subject to several limits in applicability. While the consistent model requires that elastic coordinates and speeds be kept small, the two alternative models will

only be accurate if low angular rates are also maintained. There also exists some translational acceleration limit that needs to be considered, although for the cases studied this limit was of no consequence. Within the domain of validity of both simplified models, it appears the ruthless model yields results as accurate as the correct consistently linearized model. This coupled to the simplification of the dynamicist's task inherent in the adoption of ruthlessly linearized models makes this option an attractive alternative.

Simulation misbehavior at certain trajectories for relatively high rigid body angular rates was tracked down to numerical ill-conditioning of the configuration dependent mass matrix. This problem was attributed to modelling error inherent in choosing cantilever (clamped-free) modes to model the flexible deflections of the manipulator links. In chapter four it was suggested that this results in effectively modelling one (or more) massless degrees of freedom. Two possible solutions were outlined. A numerical solution entails condensing out these massless degrees of freedom through the use of a singular value decomposition approximate inversion of the mass matrix. This is expected to result in a simulation that is computationally very expensive. The use of mode shapes obtained from a cantilever beam with an end mass with moment of inertia is proposed as the analytical solution. Both these concepts are under study at present and are suggested as further work.

REFERENCES

1 Banerjee, A.K., Dickens, J.M., "Dynamics of an Arbitrary Flexible Body Undergoing Large Rotation and Translation with Small Vibration," Paper no. AIAA-89-1308, 30th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Mobile Alabama, April 3-5, 1989.

2 Blevins, R.D., *Formulas for Natural Frequency and Mode Shape*, Robert E. Krieger Publishing Company, Malabar, Florida, 1979.

3 Bodley, C. S., et al., "A Digital Computer Program for the Dynamic Interaction Simulation of Controls and Structure (DISCOS)," Vols. 1 & 2, NASA Technical Paper 1219, May 1978.

4 Brady, M., Hollerbach, J. M., et al., *Robot Motion: Planning and Control*, MIT Press, Cambridge, MA, 1982, pp 55-60.

5 Craig, J.J., *Introduction to Robotics Mechanics and Control*, Addison-Wesley Publishing Company, 1986, pp.191-219.

6 Golub, G.H., Van Loan, C.F., *Matrix Computations*, The Johns Hopkins University Press, Baltimore 1983, pp. 52-79.

7 Hollerbach, J.M., "Dynamic Scaling of Manipulator Trajectories," Journal of Dynamic Systems, Measurement, and Control, Vol. 106, March 1984, pp. 102-106.

8 Hughes, P. C., "Multibody Dynamics for Space Station Manipulators: Dynamics of a Chain of Elastic Bodies," DYNACON Report SS-2, DYNACON Enterprises Ltd., Feb. 1985.

9 Kane, T. R., Levinson, D. A., *Dynamics: Theory and Applications*, McGraw -Hill Book Company, 1985.

10 Kane, T. R., Likins, P. W., Levinson, D. A., *Spacecraft Dynamics*, McGraw-Hill Book Company, 1983, pp. 318-326.

11 Kane, T. R., Ryan, R. R., Banerjee, A. K., "Dynamics of a Cantilever Beam Attached to a Moving Base," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 2, March-April 1987, pp. 139-151.

12 Kelly, F., "On the Dynamics of Flexible Multibody Systems," Ph.D. Thesis, Dept. of Engineering Science, University of Cincinnati, Cincinnati, 1982.

13 Laskin, R. A., Likins, P. W., Longman, R. W., "Dynamical Equations of a Free-Free Beam Subject to Large Overall Motions," AAS/AIAA Astrodynamics Conference, Lake Tahoe, NV, August 3-5, 1981

14 Love, A. E. H., *A Treatise on the Mathematical Theory of Elasticity*, 4th ed., Dover Publications, New York, 1944.

15 Man, G., Laskin, R., Editors, Proceedings of the Workshop on Multibody Simulation, JPLD-5190, Jet Propulsion Laboratory, Pasadena, California, April 15, 1988.

16 MATRIXx User's Guide, Version 6.0, Integrated Systems Inc., Santa Clara, California, May 1986.

17 Meier, E.B., Bryson, Jr., A.E., "An Efficient Algorithm for Time-Optimal Control of a Two-link Manipulator," AIAA paper no. 87-2263.

18 Meirovitch, L., *Analytical Methods in Vibrations*, Mcmillan Publishing Co., Inc., New York, 1967.

19 Meirovitch, L., *Elements of Vibration Analysis*, 2nd ed., McGraw-Hill Book Company, 1986.

20 Novozhilov, V. V., *Foundations of the Nonlinear Theory of Elasticity*, Graylock Press, Rochester, N.Y., 1953.

21 Pars, L. A., *A Treatise on Analytical Dynamics*, John Wiley & Sons, Inc., New York, N.Y., 1965.

22 Press, W.H., Flanner, B.P., Teukolsky, S.A., Vetterling, W.T., *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, London, 1986, pp. 550-559.

23 Ryan, R. R., "Flexible Multibody Dynamics: Problems and Solutions," JPL D-5190, Vol. I, Proceedings of the Workshop on Multibody Simulation, April 15, 1988, pp. 103-190.

24 Ryan, R. R., "Flexibility Modeling Methods in Multibody Dynamics," Paper no. AAS 87-431, AAS/AIAA Astrodynamics Specialist Conference, Kalispell, Montana, August 10-13, 1987.

25 Schmitz, E., "Experiments on the End-Point Position Control of a Very Flexible One-link Manipulator," Ph.D. Thesis, Dept. of Aeronautics and Astronautics, Stanford University, June 1985.

26 Simo, J. C., "Nonlinear Dynamics of Flexible Structures, A Geometrically Exact Approach," JPL D-5190, Vol. I, Proceedings of the Workshop on Multibody Simulation, April 15, 1988, pp. 235-297.

27 Spanos, J., Laskin, R. A., "Geometric Nonlinear Effects in Simple Rotating Systems," JPL D-5190, Vol. I, Proceedings of the Workshop on Multibody Simulation, April 15, 1988, pp. 191-218.

28 Storch, J., Strang, G., "Paradox Lost: Natural Boundary Conditions in the Ritz-Galerkin Method," Numerical Analysis Report 87-7, Dept. of Mathematics, M.I.T.

29 Sunada, W.H., Dubowsky, S., "On the Dynamic Analysis and Behavior of Industrial Robotic Manipulators with Elastic Members," ASME paper no. 82-DET-45, ASME Design and Production Engineering Technical Conference, Washington D.C., Sept. 12-15, 1982.

30 Timoshenko, S. P., Young, D. H., Weaver, W., *Vibration Problems in Engineering*, 4th ed., Wiley and Sons, 1974.

31 User's Manual for TREETOPS, "A Control System Simulation for Structures with a Tree Topology," Singh, Likins, et al., NASA contract no. NAS8-34588, Rev. B, November 1984.

32 von Flotow, A. H., Lecture Notes, MIT Aeronautics and Astronautics Dept., April 1988, pp. 850-858.

APPENDIX A: NONLINEAR STRAIN-DISPLACEMENT AND LINEARIZATION

This appendix first presents a general discussion on the proper linearization of the equations of motion of a flexible multibody system undergoing motion with large rigid body rates (not just angular velocities) and rigid body configuration changes, but with infinitesimal elastic deformations. Some of the terms in these motion equations cannot be obtained with linear kinematics of elastic deformation (i.e., the traditional linear finite element or modal formulation). It has been suggested [26] that this is not due to a fundamental flaw of the linear theories, but rather to the inability of these theories to account for nonlinear geometric effects. The role of nonlinear strain-displacement relations in the proper linearization is then investigated using the simple example of a Bernoulli-Euler beam cantilevered to a rigid base free to move in the plane.

A.1 GENERAL DISCUSSION

The problem that concerns us is that of obtaining the correctly linearized equations of motion for an important class of systems which exhibit large rigid body motions but small elastic deflections. Needless to say this class of systems is of vital interest in many fields, particularly in aerospace. By far the most common practice to date is to handle the flexibility through discretization of the desired continuous system. This is achieved by representing the solution as a finite series of time-dependent generalized elastic coordinates multiplied by space-dependent functions, as in an assumed modes approach, or in a finite element formulation [19]. Whichever formulation one uses, in light of the class of systems under study, the next step is to require that these elastic coordinates, together with the generalized elastic speeds, be infinitesimally small. In other words, we want these coordinates and speeds to be small enough so that only terms linear in them are kept in the equations of motion, as terms of second order or higher are negligible.

Now that our goal is clearly stated, it should be an easy matter to obtain the linearized equations of motion as long as we consistently drop all terms nonlinear in the elastic coordinates and the corresponding generalized elastic speeds. Of course, the word "consistently" is the catch. When do we linearize? That is to say: Does it matter at what step in our derivation of the equations of motion we start to linearize? To answer this question we have to consider the process by which we derive these equations.

Let us consider two of the more widely known methods to derive motion equations for complex systems: Lagrange's equations of motion [21] and Kane's dynamical equations [9]. Lagrange's equations for a holonomic system with n generalized coordinates q_k:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_k}\right) - \frac{\partial L}{\partial q_k} = Q_k, \qquad (k = 1, \ldots, n)$$
$$L = T - V \tag{1}$$

where the Lagrangian L is a function of the system kinetic and potential energies (T and V respectively). Q_k are n generalized non-potential forces. The important thing to note is that using this method to derive motion equations requires differentiating both the potential and kinetic energies of the system with respect to the generalized coordinates and speeds. If these q_i and u_i (=dq_i/dt) were to represent our generalized elastic coordinates, we see that the above differentiations imply that some terms linear in q_i and u_i in the energy expressions become terms of zeroth order in the elastic coordinates and speeds. More importantly, we see that terms of second order in the generalized coordinates and speeds in the energy expressions become terms of first order in the resulting equations of motion. Clearly then in order to obtain equations of motion correct to first order in q_i and u_i we need to have energy expressions for our system that are correct to second order in these same elastic generalized coordinates and speeds. More specifically the requirement demands in general that the expressions for displacements and velocities used in determining potential and kinetic energies be correct to second order in the elastic coordinates and speeds. Only by doing this can we ensure consistent linearization.

Kane's dynamical equations for a holonomic system of n particles with n generalized speeds u_i:

$$F_r + F_r^* = 0 \qquad (r = 1, \ldots, n)$$
$$F_r^* = \sum_{i=1}^{n} \mathbf{v}_r^{P_i} \bullet \mathbf{R}_i^* \quad , \quad \mathbf{R}_i^* = -m_i \mathbf{a}_i$$
$$\mathbf{v}^{P_i} = \sum_{i=1}^{n} \mathbf{v}_r^{P_i} u_i + \mathbf{v}_t \quad , \quad u_i = \dot{q}_i \tag{2}$$

where F_r is the generalized active force, F_r^* is the generalized inertia force, $\mathbf{R}_i^*$ is the inertia force for particle P_i in an inertial reference frame, and $\mathbf{v}^{P_i}$ is the velocity of this particle in the same frame. $\mathbf{v}_r^{P_i}$ is the r-th partial velocity of particle P_i in the inertial frame. Using a

similar argument as above, it is easy to see that since the partial velocity has to be correct to first order in the generalized coordinates and speeds, and again this term is obtained through differentiation of the velocity with respect to the generalized speeds, the velocity has to be correct to second order in q_i and u_i until we form the partial velocities. This is necessary if we want our equations of motion to be consistently linear in the generalized coordinates and speeds.

A.2 A SIMPLE EXAMPLE

We now present a simple yet very important example which will help to point out the practical consequences of the above discussion. This example could equally well represent either link of the manipulator which is the center piece of this thesis. Consider a simple uniform beam cantilevered to a solid body free to move in the plane with mass center at A* (see Fig. A.1). The frame N, defined by the unit vectors $\mathbf{n}_1$, $\mathbf{n}_2$, $\mathbf{n}_3$, is inertial, and we introduce the rotating frame A defined by the unit vectors $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, attached to body A and whose $\mathbf{a}_1$ axis lies along the undeformed neutral axis of the beam B initially. Let us derive the consistently linearized equations of motion using both Kane's and Lagrange's methods. We will ignore shear and rotary inertia effects (i.e., slender beam assumption). For simplicity, we assume no external forces act on the system.

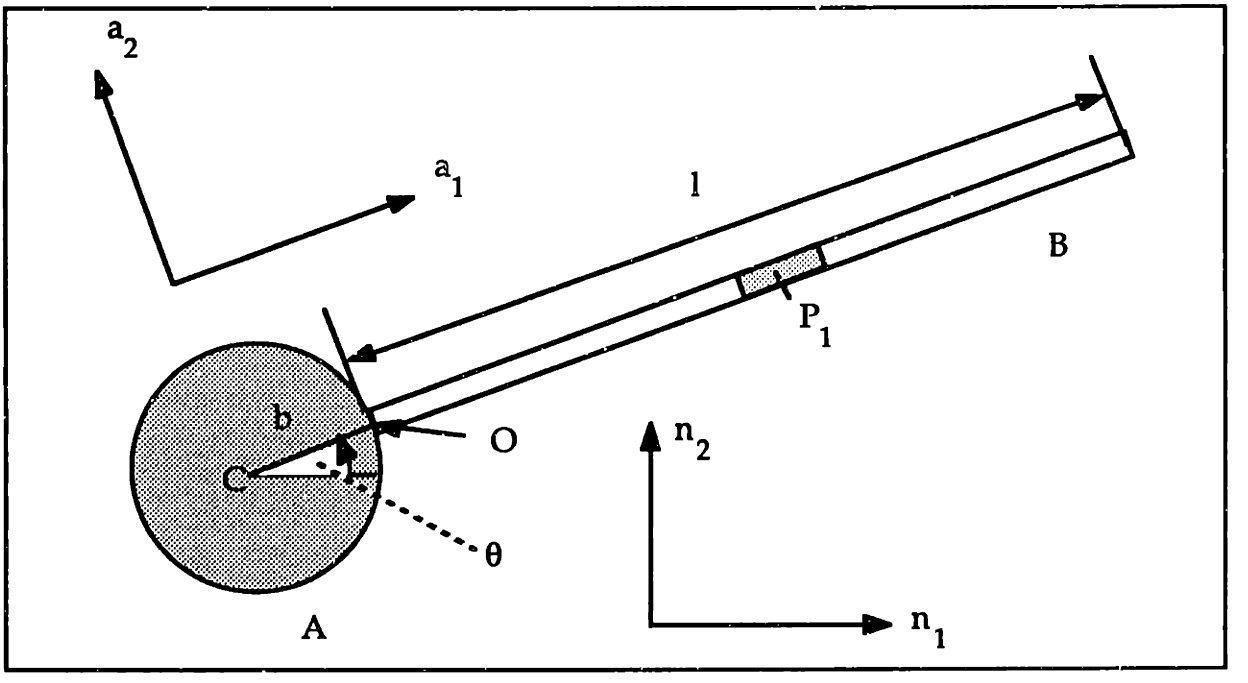


Figure A.1: Single Beam Attached to a Moving Base

A.2.1 Lagrange's Equations of Motion

First we form the kinetic and potential energies of our system:

$$T = \frac{1}{2} I_A \dot{\theta}^2 + \frac{1}{2} m_A \, {}^N\mathbf{v}^{A^*} \bullet {}^N\mathbf{v}^{A^*} + \frac{1}{2} \int_0^L \mathbf{v}^{P_1} \bullet \mathbf{v}^{P_1} \rho dx$$

$$V = \frac{1}{2} \int_0^L EI \left(\frac{\partial^2 u_2}{\partial x^2} \right)^2 dx \tag{1}$$

where m_A is the mass of the rigid body A; I_A is the moment of inertia of the rigid body A about its mass center; ${}^N\mathbf{v}^{A^*}$ is the velocity with respect to the inertial frame of A^*; $\mathbf{v}^{P1}$ is the velocity with respect to the inertial frame of the centroid of a rigid differential element P_1 of the beam; ρ and EI are the mass per unit length and bending stiffness of the beam respectively; and, $u_1(x,t)$ and $u_2(x,t)$ are respectively the axial and transverse displacements in frame A of the element P_1 a distance x along $\mathbf{a}_1$ from point O. Now we form $\mathbf{v}^{P1}$ from the kinematics of the system:

$$\mathbf{v}^{P_1} = \mathbf{v}^{A^*} + \underline{\omega}^A \times \mathbf{p}^{A^* P_1} + {}^A\mathbf{v}^{P_1}$$

$$\mathbf{p}^{A^* P_1} = (b + x + u_1)\hat{\mathbf{a}}_1 + u_2 \hat{\mathbf{a}}_2$$

$$\underline{\omega}^A = v_3 \hat{\mathbf{a}}_3 \quad , \quad v_3 = \dot{\theta}$$

$$\mathbf{v}^{A^*} = {}^N\mathbf{v}^{A^*} = v_1 \hat{\mathbf{a}}_1 + v_2 \hat{\mathbf{a}}_2$$

$${}^A\mathbf{v}^{P_1} = \dot{u}_1 \hat{\mathbf{a}}_1 + \dot{u}_2 \hat{\mathbf{a}}_2 \tag{2}$$

where $\underline{\omega}^A$ is the angular velocity of frame A ; $\mathbf{p}^{A^*P1}$ is the position vector between A^* and P_1 and ${}^A\mathbf{v}^{P1}$ is the velocity of P_1 in A. At this point it is customary to discretize the continuous variables $u_1(x,t)$ and $u_2(x,t)$. To this end we use an assumed modes approach and state

$$u_1(x,t) = \sum_{i=1}^{n} \phi_{1i}(x) p_i(t)$$

$$u_2(x,t) = \sum_{i=1}^{n} \phi_{2i}(x) q_i(t) \tag{3}$$

where we choose ϕ_{1i} and ϕ_{2i} to be cantilevered beam mode shapes [18], and n is the number of generalized elastic coordinates. Further recognizing that under the present assumptions the axial displacements are at least an order of magnitude smaller than the transverse displacements, it is tempting to neglect $u_1(x,t)$ entirely setting u_1=0.

Both these approaches, however, are only correct to first order in the elastic coordinates, as we will see. To obtain the correct relationships to second order, we need to turn to the nonlinear theory of elasticity as applied to beams, where we make use of nonlinear strain-displacement relations. For our case, a slender beam in pure bending, the strain-displacement relations become [20]:

$$\varepsilon_{xy} = \varepsilon_{yz} = \varepsilon_{yy} = \varepsilon_{zz} = \varepsilon_{xz} = 0$$

$$\varepsilon_{xx} = \hat{\varepsilon}_{xx} + y\chi_{xx} \quad \chi_{xx} = -\frac{d\theta}{dx}, \quad \frac{d\hat{u}_2}{dx} = \sin\ \theta \cong \theta$$

$$\hat{\varepsilon}_{xx} = \frac{d\hat{u}_1}{dx} + \frac{1}{2}\left[\left(\frac{d\hat{u}_1}{dx}\right)^2 + \left(\frac{d\hat{u}_2}{dx}\right)^2\right] \tag{4}$$

where the ^ indicates a quantity measured at the middle plane of the beam, i.e., the plane containing the neutral axis in this case. Note we are able to set $\sin(\theta)=\theta$ due to our assumption of small elastic deflections. Now we consider the simpler case in which we want to neglect independent axial displacements. The correct condition for this is to require

$$\hat{\varepsilon}_{xx} = 0 \tag{5}$$

and this yields the familiar expression for axial strain under pure bending:

$$\varepsilon_{xx} = -\ y\frac{d^2\hat{u}_2}{dx^2} \tag{6}$$

Given the above nonlinear strain-displacement relations, we see that equation (5) implies

$$\frac{d\hat{u}_1}{dx} + \frac{1}{2}\left[\left(\frac{d\hat{u}_1}{dx}\right)^2 + \left(\frac{d\hat{u}_2}{dx}\right)^2\right] = 0 \tag{7}$$

therefore to second order in the generalized elastic coordinates (q_i):

$$u_1(x,t) = \hat{u}_1(x,t) = -\int_0^x \frac{1}{2}\left(\frac{d\hat{u}_2}{d\sigma}\right)^2 d\sigma = -\frac{1}{2}\int_0^x \sum_{i=1}^{n}\sum_{j=1}^{n} \phi'_{2i}(\sigma)\phi'_{2j}(\sigma)d\sigma q_i q_j \tag{8}$$

where as above:

$$u_2(x,t) = \hat{u}_2(x,t) = \sum_{i=1}^{n} \phi_{2i}(x)q_i(t) \tag{9}$$

So we see that if we pick u_2(x,t) to be the independent transverse displacement, u_1(x,t) depends on u_2 to second order in the elastic coordinates. Now we can form the kinetic energy correct to second order. For completeness we note:

$$\dot{u}_1(x,t) = -\int_0^x \sum_{i=1}^{n}\sum_{j=1}^{n} \phi'_{2i}(\sigma)\phi'_{2j}(\sigma)d\sigma q_i \dot{q}_j \tag{10}$$

In view of the previous discussion we might wonder if our potential energy as given is correct to second order. To check this, we form the strain energy [14]:

$$S = \frac{1}{2}\int_V \tau_{xx}\varepsilon_{xx}dV = \frac{1}{2}\int_V E\varepsilon_{xx}^2 dV \tag{11}$$

where we ignore Poisson effects as customary for beam problems [30] and using equation (6):

$$S = \frac{1}{2}\int_V Ey^2\left(\frac{d^2u_2}{dx^2}\right)^2 dxdydz = \frac{1}{2}\int_0^L EI\left(\frac{d^2u_2}{dx^2}\right)^2 dx \tag{12}$$

Now we proceed to obtain the equations of motion correct to first order in q_i and u_i (=dq_i/dt):

$$\begin{aligned} &\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = 0, \\ &\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_r}\right) - \frac{\partial L}{\partial q_r} = 0, \qquad (r = 1,\ldots,n) \end{aligned} \tag{13}$$

This yields:

$$
\begin{bmatrix}
m_A + m_B & 0 & -\sum_{i=1}^{n} E_i q_i & \cdots & 0 & \cdots \\
0 & m_A + m_B & bm_B + e & \cdots & E_i & \cdots \\
-\sum_{i=1}^{n} E_i q_i & bm_B + e & b^2 m_B + 2eb + I_B + I_A & \cdots & bE_i + F_i & \cdots \\
\vdots & \vdots & \vdots & \ddots & \vdots & \iddots \\
-\sum_{i=1}^{n} \mu_{ij} q_i & E_j & bE_j + F_j & \cdots & G_{ij} & \cdots \\
\vdots & \vdots & \vdots & \iddots & \vdots & \ddots
\end{bmatrix}
\begin{bmatrix} \dot{v}_1 \\ \dot{v}_2 \\ \dot{v}_3 \\ \vdots \\ \ddot{q}_i \\ \vdots \end{bmatrix} =
$$

$$
\begin{bmatrix}
(m_A + m_B) v_2 v_3 + v_3^2 (bm_B + e) + 2v_3 \sum_{i=1}^{n} E_i \dot{q}_i \\
-(m_A + m_B) v_3 v_1 + v_3^2 \sum_{i=1}^{n} E_i q_i \\
-v_2 v_3 \sum_{i=1}^{n} E_i q_i - v_3 v_1 (bm_B + e) \\
\vdots \\
- v_3 v_1 E_j + v_3^2 \sum_{i=1}^{n} G_{ij} q_i - v_3^2 \sum_{i=1}^{n} (b\mu_{ij} + \eta_{ij}) q_i - v_2 v_3 \sum_{i=1}^{n} \mu_{ij} q_i \\
\vdots
\end{bmatrix} +
$$

$$
\begin{bmatrix}
\cdots & 0 & \cdots \\
\cdots & 0 & \cdots \\
\cdots & 0 & \cdots \\
\ddots & \vdots & \iddots \\
\cdots & -H_{ij} & \cdots \\
\iddots & \vdots & \ddots
\end{bmatrix}
\begin{bmatrix} \vdots \\ q_i \\ \vdots \end{bmatrix} \tag{14}
$$

where following Kane et al. [10] we have defined:

$$
m_B \equiv \int_0^L \rho dx, \quad e \equiv \int_0^L x \rho dx, \quad I_B \equiv \int_0^L x^2 \rho dx
$$

$$
H_{ij} \equiv \int_0^L EI \phi_{2i}''(x) \phi_{2j}''(x) dx, \quad E_i \equiv \int_0^L \phi_{2i}(x) \rho dx
$$

$$
F_i \equiv \int_0^L x \phi_{2i}(x) \rho dx, \quad G_{ij} \equiv \int_0^L \phi_{2i}(x) \phi_{2j}(x) \rho dx
$$

$$
(i, j = 1, \ldots, n) \tag{15}
$$

and we have further defined:

$$\mu_{ij} \equiv \int_0^L \rho \int_0^x \phi'_{2i}(\sigma)\phi'_{2j}(\sigma)d\sigma dx,$$
$$\eta_{ij} \equiv \int_0^L x\rho \int_0^x \phi'_{2i}(\sigma)\phi'_{2j}(\sigma)d\sigma dx \tag{16}$$

The above equations are equivalent to those derived in [11] through the use of geometric nonlinear constraints. The centripetal term in equation (14) exhibits the "geometric stiffening" terms (μ_{ij}, η_{ij}) which are not obtained through the standard finite element or assumed modes approach. We note that equation (14), correctly linearized in the small elastic generalized coordinates and speeds, has been obtained from a totally consistent approach to linearization. We mention in passing that the method presented above to obtain the proper second order (in q_i and u_i) expressions for the elastic deflections and speeds can be naturally extended to accommodate the case in which we are interested in independent axial displacements. In this case, we discretize the axial strain at the neutral axis and thus obtain the following expression for the axial displacement (at the neutral axis):

$$\hat{\varepsilon}_{xx} = \sum_{i=1}^{n} \phi_{\varepsilon i}(x)\, p_i$$
$$u_1(x,t) = \sum_{i=1}^{n} \int_0^x \phi_{\varepsilon i}(\sigma)d\sigma\, p_i - \frac{1}{2}\int_0^x \sum_{i=1}^{n}\sum_{j=1}^{n} \phi'_{2i}(\sigma)\phi'_{2j}(\sigma)d\sigma q_i\, q_j$$
$$\phi_{1i}(x) = \int_0^x \phi_{\varepsilon i}(\sigma)d\sigma \tag{17}$$

A.2.2 Kane's Dynamical Equations

In our case:

$$F_r^* = -I_B\, \omega_r^A \bullet \alpha^A - m_A\, \mathbf{v}_r^{A^*} \bullet \mathbf{a}^{A^*} - \int_0^l \rho\, \mathbf{v}_r^{P_1} \bullet \mathbf{a}^{P_1} dx,$$

$$(F_r)_{Ext} = 0, \qquad \alpha^A = \frac{d}{dt}(\omega^A), \qquad \mathbf{a}^{A*} = \frac{d}{dt}(^N\mathbf{v}^{A*})$$
$$\mathbf{a}^{P_1} = \frac{d}{dt}\left(\mathbf{v}^{P_1}\right), \qquad (r = \dot{\theta}, \dot{q}_i; i = 1, \ldots, n) \tag{1}$$

$(F_r)_{Ext}$ are generalized active forces due to external forces, and the time derivatives are with respect to the inertial frame. We need now to form the partial velocities and partial angular velocities correct to first order in q_i and u_i. This is simple enough using the second-order expression for $\mathbf{v}^{P1}$ derived in the previous example. The results are given in Table A.1:

r	ω_r^A	$^N\mathbf{v}_r^{A*}$	$\mathbf{v}_r^{P_1}$
$\dot{\theta}$	$\hat{\mathbf{a}}_3$	0	$-u_2\hat{\mathbf{a}}_1 + (b + x)\hat{\mathbf{a}}_2$
v_1	0	$\hat{\mathbf{a}}_1$	$\hat{\mathbf{a}}_1$
v_2	0	$\hat{\mathbf{a}}_2$	$\hat{\mathbf{a}}_2$
$\dot{q}_j$	0	0	$-\sum_{i=1}^{n}\int_0^x \phi'_{2i}(\sigma)\phi'_{2j}(\sigma)d\sigma\, q_i\hat{\mathbf{a}}_1 + \phi_{2j}(x)\hat{\mathbf{a}}_2$

Table A.1: Linearized Partial Velocities and Angular Velocities

We have again ignored independent axial extensions. Table 1 agrees with the linearized partial velocities obtained in [11]. And the equations of motion given by

$$F_r + F_r^* = 0, \qquad (r = \dot{\theta}, \dot{q}_1, \ldots, \dot{q}_n) \tag{2}$$

are the same as in equation A.2.1.14 as expected.

APPENDIX B: CALCULATION OF MODAL INTEGRALS AND CONSTANTS

In this Appendix, modal integrals and constants that appear in the equations of motion of Appendix C are defined and evaluated. For convenience, these will be divided into three groups: 1) constants that are independent of mode shapes; 2) mode shape dependent constants that can be obtained with linear strain-displacement relations (see discussion in Appendix A); and, 3) mode shape dependent constants that result from proper linearization (and would be missing from the motion equations otherwise). These constants are evaluated by three separate subroutines in the simulation code (see Appendix D): CONSTANT, CONMOD, and FOREST respectively.

The constants below are evaluated for the physical manipulator parameters presented in Table 4.1.

B.1 CONSTANTS INDEPENDENT OF MODE SHAPE

$$L_1 = b_1 + l_1 + b_{21} \qquad L_2 = b_{22} + l_2 + b_t$$

$$L_1 = 1.0668 \qquad L_2 = 0.9779$$

$$m_1 = \rho_1 l_1 \qquad m_2 = \rho_2 l_2$$

$$m_1 = 1.22472 \qquad m_2 = 0.61236$$

$$a_1 = \int_0^{l_1} x\rho_1 dx = \frac{1}{2} m_1 l_1 \qquad a_2 = \int_0^{l_2} y\rho_2 dy = \frac{1}{2} m_2 l_2$$

$$a_1 = 0.559942 \qquad a_2 = 0.279971$$

$$I_1 = \int_0^{l_1} x^2\rho_1 dx = \frac{1}{3} m_1 l_1^2 \qquad I_2 = \int_0^{l_2} y^2\rho_2 dy = \frac{1}{3} m_2 l_2^2$$

$$I_1 = 0.341341 \qquad I_2 = 0.170670$$

B.2 MODE SHAPE DEPENDENT CONSTANTS

Cantilever (clamped-free) mode shapes are assumed in the evaluation of the modal integrals presented below. The mode shapes are given by [2]

$$\tilde{y}_i(s) = \cosh\left(\frac{\lambda_i s}{l_j}\right) - \cos\left(\frac{\lambda_i s}{l_j}\right) - \sigma_i\left(\sinh\left(\frac{\lambda_i s}{l_j}\right) - \sin\left(\frac{\lambda_i s}{l_j}\right)\right)$$

$$\tilde{y}_i = \begin{cases} \phi_i \\ \psi_i \end{cases}, \quad s = \begin{cases} x \\ y \end{cases}, \quad i = \begin{cases} 1,..,n & , j = 1 \\ 1,..,m & , j = 2 \end{cases}$$

where

$$\sigma_i = \frac{\sinh \lambda_i - \sin \lambda_i}{\cosh \lambda_i + \cos \lambda_i}$$

and λ_i are the solutions in increasing order of the transcendental equation

$$\cos \lambda \cosh \lambda + 1 = 0.$$

$$\lambda_i = \begin{bmatrix} 1.87510407 \\ 4.69409113 \\ 7.85475744 \\ 10.99554073 \end{bmatrix} \qquad \sigma_i = \begin{bmatrix} 0.734095514 \\ 1.018467319 \\ 0.999224497 \\ 1.000033553 \end{bmatrix}$$

The following orthogonality relations hold:

$$\int_0^l \tilde{y}_i(s)\,\tilde{y}_j(s)ds = \int_0^l \tilde{y}''_i(s)\,\tilde{y}''_j(s)ds = \begin{cases} 0, & i \neq j \\ l, & i = j \end{cases}$$

with

$$\tilde{y}''_i(s) = \left(\frac{l}{\lambda_i}\right)^2 \frac{d^2 \tilde{y}_i}{ds^2}$$

$$\frac{d^n}{d\left(\frac{\lambda_i s}{l}\right)^n} \tilde{y}_i = \frac{d^{n+4}}{d\left(\frac{\lambda_i s}{l}\right)^{n+4}} \tilde{y}_i.$$

In all cases, the constants are evaluated for the first four mode shapes.

B.2.1 Constants obtained using linear strain-displacement relations

$$A_i(l_1) = \phi_i(l_1) + b_{21}\phi'_i(l_1) \qquad B_i(l_2) = \psi_i(l_2) + b_t\psi'_i(l_2)$$

$$A_i(l_1) = \begin{bmatrix} 2.229417 \\ -2.796795 \\ 3.308112 \\ -3.832654 \end{bmatrix} \qquad B_i(l_2) = \begin{bmatrix} 2.152944 \\ -2.531196 \\ 2.872074 \\ -3.221770 \end{bmatrix}$$

$$A_{ij}(l_1) = A_i(l_1)A_j(l_1)$$

$$A_{ij}(l_1) = \begin{bmatrix} 4.970302 & -6.235223 & 7.375162 & -8.544586 \\ & 7.822060 & -9.252109 & 10.719147 \\ & & 10.943603 & -12.678848 \\ \text{symmetric} & & & 14.689239 \end{bmatrix}$$

$$B_{ij}(l_2) = B_i(l_2)B_j(l_2)$$

$$B_{ij}(l_2) = \begin{bmatrix} 4.635172 & -5.449527 & 6.183418 & -6.936292 \\ & 6.406955 & -7.269785 & 8.154931 \\ & & 8.248812 & -9.253162 \\ \text{symmetric} & & & 10.379799 \end{bmatrix}$$

$$E_i = \int_0^{l_1} \phi_i(x)\rho_1 dx \qquad E_i^* = \int_0^{l_2} \psi_i(y)\rho_2 dy$$

$$E_i = \begin{bmatrix} 0.958946 \\ 0.531450 \\ 0.311600 \\ 0.222774 \end{bmatrix} \qquad E_i^* = \begin{bmatrix} 0.479473 \\ 0.265725 \\ 0.155800 \\ 0.111387 \end{bmatrix}$$

$$F_i = \int_0^{l_1} x\phi_i(x)\rho_1 dx \qquad F_i^* = \int_0^{l_2} y\psi_i(y)\rho_2 dy$$

$$F_i = \begin{bmatrix} 0.637019 \\ 0.101648 \\ 0.036303 \\ 0.018525 \end{bmatrix} \qquad F_i^* = \begin{bmatrix} 0.318509 \\ 0.050824 \\ 0.018151 \\ 0.009263 \end{bmatrix}$$

$$G_{ij} = \int_0^{l_1} \phi_i(x)\phi_j(x)\rho_1 dx$$

$$G_{ij} = \begin{bmatrix} 1.22472 & 0 & 0 & 0 \\ 0 & 1.22472 & 0 & 0 \\ 0 & 0 & 1.22472 & 0 \\ 0 & 0 & 0 & 1.22472 \end{bmatrix}$$

$$G_{ij}^* = \int_0^{l_2} \psi_i(y)\psi_j(y)\rho_2 dy$$

$$G_{ij}^* = \begin{bmatrix} 0.61236 & 0 & 0 & 0 \\ 0 & 0.61236 & 0 & 0 \\ 0 & 0 & 0.61236 & 0 \\ 0 & 0 & 0 & 0.61236 \end{bmatrix}$$

$$H_{ij} = \int_0^{l_1} (EI)_1 \phi''_i(x)\phi''_j(x)dx$$

$$H_{ij} = \begin{bmatrix} 1843.30 & 0 & 0 & 0 \\ 0 & 72393.95 & 0 & 0 \\ 0 & 0 & 567580.3 & 0 \\ 0 & 0 & 0 & 2179528.6 \end{bmatrix}$$

$$H^*_{ij} = \int_0^{l_2} (EI)_2 \psi''_i(y)\psi''_j(y)dy$$

$$H^*_{ij} = \begin{bmatrix} 226.37 & 0 & 0 & 0 \\ 0 & 8890.5 & 0 & 0 \\ 0 & 0 & 69702.8 & 0 \\ 0 & 0 & 0 & 267661.4 \end{bmatrix}$$

B.2.2 Constants obtained through proper linearization

$$\beta_{ij}(x) = \int_0^x \phi'_i(\sigma)\phi'_j(\sigma)d\sigma \qquad \beta^*_{ij}(y) = \int_0^y \psi'_i(\sigma)\psi'_j(\sigma)d\sigma$$

$$C_{ij}(l_1) = \beta_{ij}(l_1) + b_{21}\phi'_i(l_1)\phi'_j(l_1)$$

$$C_{ij}(l_1) = \begin{bmatrix} 5.773586 & -10.469668 & 8.248862 & -12.728235 \\ & 43.783916 & -38.123383 & 34.017360 \\ & & 106.991190 & -70.446206 \\ \text{symmetric} & & & 200.355635 \end{bmatrix}$$

$$C^*_{ij}(l_2) = \beta^*_{ij}(l_2) + b_t\psi'_i(l_2)\psi'_j(l_2)$$

$$C^*_{ij}(l_2) = \begin{bmatrix} 5.543348 & -9.670022 & 6.936073 & -10.889029 \\ & 41.006644 & -33.563899 & 27.629564 \\ & & 99.505825 & -59.959278 \\ \text{symmetric} & & & 185.663544 \end{bmatrix}$$

$$\mu_{ij} = \int_0^{l_1} \rho_1 \beta_{ij}(x)dx$$

$$\mu_{ij} = \begin{bmatrix} 0.742933 & -0.565643 & -1.435918 & -1.169454 \\ & 8.510827 & 2.532483 & -4.879840 \\ & & 28.808327 & 11.168055 \\ \text{symmetric} & & & 62.730250 \end{bmatrix}$$

$$\mu_{ij}^{*} = \int_0^{l_2} \rho_2 \beta_{ij}^{*}(y)\,dy$$

$$\mu_{ij}^{*} = \begin{bmatrix} 0.371467 & -0.282822 & -0.717959 & -0.584727 \\ & 4.255413 & 1.265741 & -2.439920 \\ & & 14.404163 & 5.584028 \\ \text{symmetric} & & & 31.365125 \end{bmatrix}$$

$$\eta_{ij} = \int_0^{l_1} \rho_1 x \beta_{ij}(x)\,dx$$

$$\eta_{ij} = \begin{bmatrix} 1.461503 & -0.839981 & -0.970443 & -0.669203 \\ & 7.934011 & 0.207477 & -3.566202 \\ & & 21.872910 & 4.010066 \\ \text{symmetric} & & & 44.157753 \end{bmatrix}$$

$$\eta_{ij}^{*} = \int_0^{l_2} \rho_2 y \beta_{ij}^{*}(y)\,dy$$

$$\eta_{ij}^{*} = \begin{bmatrix} 0.730751 & -0.419990 & -0.485221 & -0.334602 \\ & 3.967006 & 0.103739 & -1.783101 \\ & & 10.936455 & 2.005033 \\ \text{symmetric} & & & 22.078876 \end{bmatrix}$$

The following constants are used in sections 3.3.2 and 3.3.3:

$$\xi_1 = \frac{\mu_{11}}{\rho_1} = 0.554689$$

$$\gamma_1 = \frac{\eta_{11}}{m_1} = 1.193336.$$

APPENDIX C: EQUATIONS OF MOTION

The 2+n+m ordinary differential equations of motion for the two-link, planar, flexible manipulator are given below (refer to section 2.4.3). The motion equations are given in the form of equation (2.4.3.2):

$$\begin{bmatrix} M_{RR}(\mathbf{x}) & M_{RE}(\mathbf{x}) \\ M_{ER}(\mathbf{x}) & M_{EE}(\mathbf{x}) \end{bmatrix} \ddot{\mathbf{x}} + \begin{bmatrix} 0 & 0 \\ 0 & K_{EE} \end{bmatrix} \mathbf{x} = \begin{bmatrix} I \\ 0 \end{bmatrix} \mathbf{T} + \mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}) \tag{1}$$

where as in section 2.4.3 the following definitions have been made

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_E \end{bmatrix} = [\theta, \beta, q_1, \ldots, q_n, p_1, \ldots, p_m]^T$$

$$\mathbf{T} = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} \tag{2}$$

In the rest of this Appendix, the individual entries in the mass matrix, the stiffness matrix, and the vector of nonlinear forces are presented for the system defined in chapter 2. The definitions of Appendix B and equation (2.3.10) are used together with the following definitions to simplify the form of the entries.

$$I_{Tot} = J_A + I_1 + J_{B_1} + J_{B_2} + I_2 + J_C$$

$$I_{out} = J_{B_2} + I_2 + J_C$$

$$m_{Tot} = m_2 + m_{B_1} + m_{B_2} + m_C$$

$$c_\beta = \cos\beta, \qquad s_\beta = \sin\beta$$

$$co_1 = a_2 + b_{22} m_2 + m_{B_2} e_2 \cos\gamma_2 + m_C L_2$$

$$
\begin{aligned}
co_2 &= m_{B_2} e_2 \sin\ \gamma_2 \\
co_{3j} &= A_j(l_1) + L_1 \phi'_j(l_1) \\
co_{4j} &= E^*_j + m_C B_j(l_2) \\
co_{5ij} &= \phi'_j(l_1) A_i(l_1) + \phi'_i(l_1) A_j(l_1) \\
co_{6j} &= A_j(l_1) - L_1 \phi'_j(l_1)
\end{aligned}
\tag{3}
$$

C.1 Entries in the Stiffness Matrix

$$
\begin{aligned}
(K_{EE})_{ij} &= H_{ij}, & i, j &= 1, \ldots, n \\
(K_{EE})_{n+i,n+j} &= H^*_{ij}, & i, j &= 1, \ldots, m \\
(K_{EE})_{i,n+j} &= (K_{EE})_{n+j,i} = 0, & i &= 1, \ldots, n; \quad j = 1, \ldots, m
\end{aligned}
\tag{1}
$$

C.2 Entries in the Mass Matrix

$$
\begin{aligned}
&(M_{RR})_{1,1} = \\
&L_{Tot} + 2a_1 b_1 + m_1 b_1^2 + 2a_2 b_{22} + m_2 b_{22}^2 + m_{Tot} L_1^2 + m_{B_1}(e_1^2 + 2L_1 e_1 \cos\ \gamma_1) \\
&+ m_{B_2} e_2^2 + m_C L_2^2 + 2(co_1 s_\beta + co_2 c_\beta + m_{B_1} e_1 \sin\ \gamma_1) \sum_{i=1}^{n} A_i(l_1) q_i \\
&- 2m_{B_1} L_1 e_1 \sin\ \gamma_1 \alpha_1 - 2L_1 s_\beta \sum_{i=1}^{m} E^*_i p_i - 2m_C L_1 s_\beta \sum_{i=1}^{m} B_i(l_2) p_i \\
&+ 2L_1(co_1 \tilde{c}_\beta - co_2 \tilde{s}_\beta)
\end{aligned}
\tag{1}
$$

$$(M_{RR})_{2,1} = (M_{RR})_{1,2} =$$

$$I_{out} + 2a_2 b_{22} + m_2 b_{22}^2 + m_{B_2} e_2^2 + m_C L_2^2 + (co_1 s_\beta + co_2 c_\beta) \sum_{i=1}^{n} A_i(l_1) q_i$$

$$- L_1 s_\beta \sum_{i=1}^{m} E_i^* p_i - m_C L_1 s_\beta \sum_{i=1}^{m} B_i(l_2) p_i + co_1 \tilde{c}_\beta - co_2 \tilde{s}_\beta \tag{2}$$

$$(M_{RR})_{2,2} =$$

$$I_{out} + 2a_2 b_{22} + m_2 b_{22}^2 + m_{B_2} e_2^2 + m_C L_2^2 \tag{3}$$

$$(M_{ER})_{j,1} = \qquad (j = 1, \ldots, n)$$

$$F_j + b_1 E_j + \phi'_j(l_1)[J_{B_1} + I_{out} + 2a_2 b_{22} + m_2 b_{22}^2 + m_{B_1}(e_1^2 + L_1 e_1 \cos \gamma_1)$$

$$+ m_{B_2} e_2^2 + m_C L_2^2] + A_j(l_1)[m_{Tot} L_1 + m_{B_1} e_1 \cos \gamma_1] + \phi'_j(l_1)[co_1 s_\beta + co_2 c_\beta$$

$$+ m_{B_1} e_1 \sin \gamma_1] \sum_{i=1}^{n} A_i(l_1) q_i - co_{3j} m_{B_1} e_1 \sin \gamma_1 \alpha_1 - co_{3j} s_\beta \sum_{i=1}^{m} E_i^* p_i$$

$$- co_{3j} m_C s_\beta \sum_{i=1}^{m} B_i(l_2) p_i + [co_1 s_\beta + co_2 c_\beta + m_{B_1} e_1 \sin \gamma_1] \sum_{i=1}^{n} C_{ij}(l_1) q_i$$

$$+ co_{3j}(co_1 \tilde{c}_\beta - co_2 \tilde{s}_\beta) \tag{4}$$

$$(M_{ER})_{n+j,1} = \qquad (j = 1, \ldots, m)$$

$$F_j^* + b_{22} E_j^* + \psi_j(l_2) J_C + m_C B_j(l_2) L_2 + co_{4j} s_\beta \sum_{i=1}^{n} A_i(l_1) q_i - L_1 s_\beta \sum_{i=1}^{m} \mu_{ij}^* p_i$$

$$- m_C L_1 s_\beta \sum_{i=1}^{m} C_{ij}^*(l_2) p_i + co_{4j} L_1 \tilde{c}_\beta \tag{5}$$

$$(M_{ER})_{j,2} = \qquad (j = 1,\ldots,n)$$

$$\phi'_j(l_1)[I_{out} + 2a_2b_{22} + m_2b_{22}^2 + m_{B_2}e_2^2 + m_C L_2^2] - A_j(l_1)s_\beta \sum_{i=1}^{m} E_i^* p_i$$

$$- m_C A_j(l_1)s_\beta \sum_{i=1}^{m} B_i(l_2)\, p_i + (co_1 s_\beta + co_2 c_\beta)\sum_{i=1}^{n} C_{ij}(l_1)q_i$$

$$+ A_j(l_1)(co_1\tilde{c}_\beta - co_2\tilde{s}_\beta) \tag{6}$$

$$(M_{ER})_{n+j,2} = \qquad (j = 1,\ldots,m)$$

$$F_j^* + b_{22}E_j^* + \psi'_j(l_2)J_C + m_C B_j(l_2)L_2 \tag{7}$$

$$(M_{RE})_{1,j} = \qquad (j = 1,\ldots,n)$$

$$F_j + b_1E_j + \phi'_j(l_1)[J_{B_1} + I_{out} + 2a_2b_{22} + m_2b_{22}^2 + m_{B_1}(e_1^2 + L_1e_1\cos\gamma_1)$$

$$+ m_{B_2}e_2^2 + m_C L_2^2] + A_j(l_1)[m_{Tot}L_1 + m_{B_1}e_1\cos\gamma_1] + co_{3j}(co_1c_\beta - co_2s_\beta) \tag{8}$$

$$(M_{RE})_{1,n+j} = \qquad (j = 1,\ldots,m)$$

$$F_j^* + b_{22}E_j^* + \psi'_j(l_2)J_C + m_C B_j(l_2)L_2 + co_{4j}L_1c_\beta \tag{9}$$

$$(M_{RE})_{2,j} = \qquad (j = 1,\ldots,n)$$

$$\phi'_j(l_1)[I_{out} + 2a_2b_{22} + m_2b_{22}^2 + m_{B_2}e_2^2 + m_C L_2^2] + A_j(l_1)(co_1c_\beta - co_2s_\beta) \tag{10}$$

$$(M_{RE})_{2,n+j} = \qquad (j = 1,\ldots,m)$$

$$F_j^* + b_{22}E_j^* + \psi'_j(l_2)J_C + m_C B_j(l_2)L_2 \tag{11}$$

$$(M_{EE})_{i,j} = \qquad (i = 1,\ldots,n;\ j = 1,\ldots,n)$$

$$\phi'_j(l_1)\phi'_i(l_1)[J_{B_1} + I_{out} + 2a_2b_{22} + m_2b_{22}^2 + m_{B_1}e_1^2 + m_{B_2}e_2^2 + m_C L_2^2]$$
$$+ A_{ij}(l_1)m_{Tot} + G_{ij} + co_{5ij}m_{B_1}e_1\cos\gamma_1 + co_{5ij}(co_1c_\beta - co_2s_\beta) \tag{12}$$

$$(M_{EE})_{n+j,i} = (M_{EE})_{i,n+j} \qquad (i = 1,\ldots,n;\ j = 1,\ldots,m)$$

$$\phi'_i(l_1)[F_j^* + b_{22}E_j^* + \psi'_j(l_2)J_C + B_j(l_2)m_C L_2] + A_i(l_1)co_{4j}c_\beta \tag{13}$$

$$(M_{EE})_{n+i,n+j} = \qquad (i = 1,\ldots,m;\ j = 1,\ldots,m)$$

$$\psi'_j(l_2)\psi'_i(l_2)J_C + G_{ij}^* + m_C B_{ij}(l_2) \tag{14}$$

C.3 Entries in the Vector of Nonlinear Forces

$$F_1 =$$

$$[({}^N\omega^{B_2})^2 - \omega_3^2]\{(-co_1c_\beta + co_2s_\beta)\sum_{i=1}^{n}A_i(l_1)q_i + L_1c_\beta\sum_{i=1}^{m}E_i^*p_i$$
$$+ m_C L_1c_\beta\sum_{i=1}^{m}B_i(l_2)p_i + L_1(co_1\tilde{s}_\beta + co_2\tilde{c}_\beta)\}$$
$$- 2\omega_3\{(co_1s_\beta + co_2c_\beta + m_{B_1}e_1\sin\gamma_1)\sum_{i=1}^{n}A_i(l_1)\dot{q}_i - m_{B_1}e_1\sin\gamma_1 L_1\dot{\alpha}_1\}$$
$$+ 2\,{}^N\omega^{B_2}\{L_1s_\beta\sum_{i=1}^{m}E_i^*\dot{p}_i + m_C L_1s_\beta\sum_{i=1}^{m}B_i(l_2)\dot{p}_i\} \tag{1}$$

$$F_2 =$$

$$- \omega^3\{(-co_1c_\beta + co_2s_\beta)\sum_{i=1}^{n} A_i(l_1)q_i + L_1c_\beta\sum_{i=1}^{m} E_i^* p_i + m_C L_1 c_\beta \sum_{i=1}^{m} B_i(l_2)\, p_i$$

$$+ L_1(co_1\tilde{s}_\beta + co_2\tilde{c}_\beta)\}$$

$$- 2\omega_3(co_1s_\beta + co_2c_\beta)\sum_{i=1}^{n} A_i(l_1)\dot{q}_i \qquad (2)$$

$$F_{2+j} = \qquad (j = 1, \ldots, n)$$

$$\omega_3^2\{\sum_{i=1}^{n}[G_{ij} - (\eta_{ij} + b_1\mu_{ij})]q_i + m_{Tot}\sum_{i=1}^{n}[A_{ij}(l_1) - L_1C_{ij}(l_1)]q_i$$

$$+ \phi'_j(l_1)(co_1c_\beta - co_2s_\beta + m_{B_1}e_1\cos\gamma_1)\phi'_j(l_1)\sum_{i=1}^{n} A_i(l_1)q_i$$

$$+ co_{6j}\, m_{B_1}e_1\cos\gamma_1\alpha_1 - \phi'_j(l_1)L_1c_\beta\sum_{i=1}^{m} E_i^* p_i - \phi'_j(l_1)m_C L_1 c_\beta \sum_{i=1}^{m} B_i(l_2)\, p_i$$

$$- m_{B_1}e_1\cos\gamma_1\sum_{i=1}^{n} C_{ij}(l_1)q_i - \phi'_j(l_1)L_1(co_1\tilde{s}_\beta + co_2\tilde{c}_\beta) + co_{6j}\, m_{B_1}e_1\sin\gamma_1\}$$

$$+ ({}^N\omega^{B_2})^2\{A_j(l_1)c_\beta\sum_{i=1}^{m} E_i^* p_i + m_C A_j(l_1)c_\beta\sum_{i=1}^{m} B_i(l_2)\, p_i$$

$$+ (-co_1c_\beta + co_2s_\beta)\sum_{i=1}^{n} C_{ij}(l_1)q_i + A_j(l_1)(co_1\tilde{s}_\beta + co_2\tilde{c}_\beta)\}$$

$$- 2\omega_3\{\phi'_j(l_1)(co_1s_\beta + co_2c_\beta + m_{B_1}e_1\sin\gamma_1)\sum_{i=1}^{n} A_i(l_1)\dot{q}_i$$

$$- A_j(l_1)m_{B_1}e_1\sin\gamma_1\dot{\alpha}_1\}$$

$$+ 2({}^N\omega^{B_2})\{A_j(l_1)s_\beta\sum_{i=1}^{m} E_i^*\dot{p}_i + m_C A_j(l_1)s_\beta\sum_{i=1}^{m} B_i(l_2)\,\dot{p}_i\} \qquad (3)$$

$$F_{2+n+j} = \qquad (j = 1, \ldots, m)$$

$$- \omega_3^2 \{ - co_{4j} c_\beta \sum_{i=1}^{n} A_i(l_1) q_i + m_C L_1 c_\beta \sum_{i=1}^{m} C_{ij}^*(l_2)\, p_i + L_1 c_\beta \sum_{i=1}^{m} \mu_{ij}^*\, p_i$$

$$+ L_1 co_{4j} \tilde{s}_\beta \}$$

$$+ ({}^N\omega^B{}_2)^2 \{ \sum_{i=1}^{m} [G_{ij}^* - (\eta_{ij}^* + b_{22}\mu_{ij}^*)]\, p_i + m_C \sum_{i=1}^{m} [B_{ij}(l_2) - L_2 C_{ij}^*(l_2)]\, p_i \}$$

$$- 2\omega_3 \{ co_{4j} s_\beta \sum_{i=1}^{n} A_i(l_1) \dot{q}_i \} \qquad (4)$$

APPENDIX D:

FORTRAN SIMULATION CODE

```
C*********************************************************************
C*********************************************************************
C          This is the main program for the two-link flex. arm
C       simulation developed by Carlos E. Padilla for his M.S. thesis.
C       DOUBLE PRECISION version.
C*********************************************************************
C*********************************************************************
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]DCONSTANTS.FOR'
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]DTORQUES.FOR'
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]DINVERSE.FOR'
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]DARMAIN.FOR'
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]DARMNEW.FOR'
C
C          In order to run the simulation, it is also necessary to
C       link the following files:
C         MATSAV.OBJ,DODEINT.OBJ,DRKQC.OBJ,DRK4.OBJ -for variable step
C         MATSAV.OBJ,DRKDUMB.OBJ,DRK4.OBJ           -for plain RK
C
        PROGRAM ARM_SIMULATION
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,NR,NOK,NBAD,KMAX,KOUNT,NSAV,MCOUNT,PMAX,NMAX
        INTEGER NSTEP
        PARAMETER (ZERO=0.0,NMAX=4,EPS=1.0D-6,TTLE=1.0D-3)
        CHARACTER*9 DAT,TIM
        REAL TINIT,TODE,TOUT
        COMMON /PROP/ N,M
        COMMON /INPU/ T1,T2,H1,PMAX,YSTART(8+4*NMAX)
        COMMON /PATH/ KMAX,KOUNT,DXSAV
        COMMON /DIAG/ HSAV,TSAV
        COMMON /DIAG/ NOK,NBAD,H1SAV,DAT,TIM,TINIT,TODE,TOUT
        COMMON /BUG/ HMIN,INTERVAL,FAVPT,TRA,TIM2,BETA,NSAV,CPS,THET
        COMMON /BUG/ MCOUNT
C
C          This is the main program of the two-link flexible arm
C       simulation.
C
        CALL DATE(DAT)
        CALL TIME(TIM)
C
C          Input the arm constants from the input files ARMDAT.DAT
C       and INIDAT.DAT (both in the THESIS sub-directory) created
C       by ARMINP.FOR.
C
        CALL INPUT
C
C          Using these constants, initialize the mass matrix and
C       the updating coefficients.
C
        TINIT=SECNDS(0.0)
        CALL INITIALIZE
        TINIT=SECNDS(TINIT)
C
C          Determine number of points to be saved (KMAX) and the
C       save interval (DXSAV).
C
        KMAX=PMAX
        DXSAV=(T2-T1)/PMAX
        INTERVAL=DXSAV
        FAVPT=SAVPTS
        TIM2=T2
```

```
      THET=YSTART(3+N+M)
      BETA=YSTART(4+N+M)
      NSAV=N
      CPS=EPS
      MCOUNT=0
C
C        Determine the trial initial time step if not provided.
C
      IF (H1 .EQ. ZERO) THEN
         CALL INITIME(H1)
      ENDIF
      H1SAV=H1
      HMIN=TTLE*H1
C
C        Solve the differential equations for the given time interval
C     with the given initial conditions.
C
      NR=2+N+M
C
C        Initial conditions for "integration checks."
C
      YSTART(1+2*NR)=ZERO
      YSTART(2+2*NR)=ZERO
      YSTART(3+2*NR)=ZERO
      YSTART(4+2*NR)=ZERO
      TODE=SECNDS(0.0)
      CALL ODEINT(YSTART,4+2*NR,T1,T2,EPS,H1,HMIN,NOK,NBAD)
C
C        The following lines implement the RKDUMB R.-K. integration.
C
c     NSTEP=INT((T2-T1)/H1*10.0)
c     PRINT *,'NSTEP=',NSTEP
c     CALL RKDUMB(YSTART,4+2*NR,T1,T2,NSTEP)
      TODE=SECNDS(TODE)
C
C        Output data from simulation to MATPLOT.DAT and DATFILE.DAT
C     and diagnostics to ARMDIAG.DAT.
C
      TOUT=SECNDS(0.0)
      CALL OUTPUT
      TOUT=SECNDS(TOUT)
      CALL OUT_DIAGNOSTICS
      END
```

```
C*********************************************************************
C*********************************************************************
C          This short program is used to interface with the user and
C       create appropriate input data files for the arm simulation.
C       It is written by Carlos E. Padilla as part of his M.S. thesis.
C       DOUBLE PRECISION version.
C*********************************************************************
C*********************************************************************
        PROGRAM CREATE_INPUT
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,FORE,I,CHOICE,BRANCH,PMAX,ANGMO,NMAX,RLINE,MDAMP
        INTEGER FOOL
        PARAMETER (ZERO=0.0,NMAX=4)
        DIMENSION BM(8),RL(4),LA(NMAX),EI(4),MM(4),JM(4),Y(8+4*NMAX)
        DIMENSION KSI(2*NMAX)
        OPEN (3,FILE='[USER.AVF.CARLOS.THESIS]INIDAT',STATUS='NEW',
     &       FORM='FORMATTED')
        PRINT *,'Short or long version of input? (0-short,1-long)'
        READ *,CHOICE
        PRINT *,'Enter initial and final times for simulation:'
        READ *,T1,T2
        PRINT *,'Enter # of pts. to be saved (0-defaults to 200; max. 500):'
        READ *,PMAX
        IF (PMAX .EQ. 0) PMAX=200
        PRINT *,'Ignore foreshortening? (0-No,1-Yes)'
        READ *,FORE
        PRINT *,'Include angular momentum and energy checks? (0-No,1-Yes)'
        READ *,ANGMO
        PRINT *,'Use ruthless equations? (0-No,1-Yes)'
        READ *,RLINE
        IF (RLINE.EQ.1) THEN
           PRINT *,'Use foolish linearization? (0-No,1-Yes)'
           READ *,FOOL
        ENDIF
        PRINT *,'Enter # of modes for links 1 and 2 (separated by spaces):'
        READ *,N,M
        PRINT *,'Enter displacement initial conditions for joint angles(rad):'
        READ *,Y(1),Y(2)
        PRINT *,'Enter velocity initial conditions for joint angles(rad/sec):'
        READ *,Y(3+N+M),Y(4+N+M)
        PRINT *,'Use default(all zero)initial cond. for links?(0-N,1-Y)'
        READ *,BRANCH
        IF (BRANCH .EQ. 1) THEN
           DO 50 I=1,N
              Y(2+I)=ZERO
              Y(4+N+M+I)=ZERO
50         CONTINUE
           DO 60 I=1,M
              Y(2+N+I)=ZERO
              Y(4+2*N+M+I)=ZERO
60         CONTINUE
        ELSE
           DO 100 I=1,N
              PRINT *,'Enter displ. initial cond. for link 1 mode',I,':'
              READ *,Y(2+I)
100        CONTINUE
           DO 200 I=1,M
              PRINT *,'Enter displ. initial cond. for link 2 mode',I,':'
              READ *,Y(2+N+I)
200        CONTINUE
```

```
            DO 300 I=1,N
               PRINT *,'Enter vel. initial cond. for link 1 mode',I,':'
               READ *,Y(4+N+M+I)
300         CONTINUE
            DO 400 I=1,M
               PRINT *,'Enter vel. initial cond. for link 2 mode',I,':'
               READ *,Y(4+2*N+M+I)
400         CONTINUE
         ENDIF
         PRINT *,'Enter initial time step (0.0 will yield default):'
         READ *,H1
         IF (CHOICE .NE. 1) GOTO 1000
         OPEN (1,FILE='[USER.AVF.CARLOS.THESIS]ARMDAT',STATUS='NEW',
     &          FORM='FORMATTED')
         PRINT *,'Enter lengths b1, b21, b22, bt, b, b':'
         READ *,BM(1),BM(2),BM(3),BM(4),BM(5),BM(6)
         PRINT *,'Enter angles beta12 and beta12':'
         READ *,BM(7),BM(8)
         PRINT *,'Enter rho1, rho2, l1, l2:'
         READ *,RL(1),RL(2),RL(3),RL(4)
         PRINT *,'Enter lambda(i):'
         READ *,LA(1),LA(2),LA(3),LA(4)
         PRINT *,'Enter E and I of link 1 and then of link 2:'
         READ *,EI(1),EI(2),EI(3),EI(4)
         PRINT *,'Include modal damping?(0-N,1-Y)'
         READ *,MDAMP
         IF (MDAMP.EQ.1) THEN
            PRINT *,'Same damping for all modes (link 1)?(0-N,1-Y)'
            READ *,CHOICE
            IF (CHOICE.NE.0) THEN
               PRINT *,'Enter damping coefficient (ksi):'
               READ *,TA
               DO 700 I=1,NMAX
                 KSI(I)=TA
700            CONTINUE
            ELSE
               DO 800 I=1,NMAX
                 PRINT *,'Enter damping coeff. (ksi) for link 1 mode',I,':'
                 READ *,KSI(1)
800            CONTINUE
            ENDIF
            PRINT *,'Same damping for all modes (link 2)?(0-N,1-Y)'
            READ *,CHOICE
            IF (CHOICE.NE.0) THEN
               PRINT *,'Enter damping coefficient (ksi):'
               READ *,TA
               DO 900 I=1,NMAX
                 KSI(NMAX+I)=TA
900            CONTINUE
            ELSE
               DO 1100 I=1,NMAX
                 PRINT *,'Enter damping coeff. (ksi) for link 2 mode',I,':'
                 READ *,KSI(NMAX+I)
1100           CONTINUE
            ENDIF
         ELSE
            DO 1200 I=1,2*NMAX
               KSI(I)=ZERO
1200        CONTINUE
         ENDIF
```

```
      PRINT *,'Enter masses ma, mb1, mb2, mc1:'
      READ *,MM(1),MM(2),MM(3),MM(4)
      PRINT *,'Enter inertias ja, jb1, jb2, jc1:'
      READ *,JM(1),JM(2),JM(3),JM(4)
      WRITE(1,*)BM(1),BM(2),BM(3),BM(4)
      WRITE(1,*)BM(5),BM(6),BM(7),BM(8)
      WRITE(1,*)RL(1),RL(2),RL(3),RL(4)
      WRITE(1,*)LA(1),LA(2),LA(3),LA(4)
      WRITE(1,*)EI(1),EI(2),EI(3),EI(4)
      WRITE(1,*)MM(1),MM(2),MM(3),MM(4)
      WRITE(1,*)JM(1),JM(2),JM(3),JM(4)
      WRITE(1,*) (KSI(I),I=1,NMAX)
      WRITE(1,*) (KSI(NMAX+I),I=1,NMAX)
      CLOSE (1)
1000  CONTINUE
      WRITE(3,*)T1,T2,H1,N,M
      WRITE(3,*)FORE,ANGMO,RLINE,FOOL
      WRITE(3,*)Y(1),Y(2),Y(3+N+M),Y(4+N+M),PMAX
      DO 500 I=1,N
         WRITE(3,*)Y(2+I),Y(4+N+M+I)
500   CONTINUE
      DO 600 I=1,M
         WRITE(3,*)Y(2+N+I),Y(4+2*N+M+I)
600   CONTINUE
      CLOSE (3)
      END
```

```
C*****************************************************************
C*****************************************************************
C          This file contains the subroutines that take care of the main
C       steps of the two-link flexible arm simulation.  All of the
C       following were written by Carlos E. Padilla as part of the M.S.
C       thesis work.
C       DOUBLE PRECISION version.
C*****************************************************************
C*****************************************************************
        SUBROUTINE INPUT
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,FORE,I,PMAX,ANGMO,NMAX,SING,RLINE,FOOL
        PARAMETER (NMAX=4)
        COMMON /CONTROL/ FORE,ANGMO,SING,RLINE,FOOL
        COMMON /PROP/ N,M,BM(8),RL(4),LA(NMAX),EI(4),MM(4),JM(4)
        COMMON /PROP/ KSI(2*NMAX)
        COMMON /INPU/ T1,T2,H1,PMAX,YSTART(8+4*NMAX)
        COMMON /BUG/ TRA,TRE,TRI,DX2
C
C          This subroutine is in charge of obtaining the constants
C       necessary for the arm simulation from the file ARMDAT.DAT.
C       The constants are in turn placed there in the appropriate order
C       by the user-interface program ARMINP.
C
        OPEN (1,FILE='[USER.AVF.CARLOS.THESIS]ARMDAT',STATUS='OLD',
     &       FORM='FORMATTED')
        OPEN (3,FILE='[USER.AVF.CARLOS.THESIS]INIDAT',STATUS='OLD',
     &       FORM='FORMATTED')
        READ(1,*)BM(1),BM(2),BM(3),BM(4)
        READ(1,*)BM(5),BM(6),BM(7),BM(8)
        READ(1,*)RL(1),RL(2),RL(3),RL(4)
        READ(1,*)LA(1),LA(2),LA(3),LA(4)
        READ(1,*)EI(1),EI(2),EI(3),EI(4)
        READ(1,*)MM(1),MM(2),MM(3),MM(4)
        READ(1,*)JM(1),JM(2),JM(3),JM(4)
        READ(3,*)T1,T2,H1,N,M
        READ(1,*) (KSI(I),I=1,NMAX)
        READ(1,*) (KSI(NMAX+I),I=1,NMAX)
        READ(3,*)FORE,ANGMO,RLINE,FOOL
        READ(3,*)YSTART(1),YSTART(2),YSTART(3+N+M),YSTART(4+N+M),PMAX
        DO 500 I=1,N
           READ(3,*)YSTART(2+I),YSTART(4+N+M+I)
 500    CONTINUE
        DO 600 I=1,M
           READ(3,*)YSTART(2+N+I),YSTART(4+2*N+M+I)
 600    CONTINUE
        DX2=SAVPTS
        CLOSE (1)
        CLOSE (3)
        RETURN
        END
C
C
C
        SUBROUTINE INITIALIZE
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,NR,I,J,N2,N3,M2,M3,FORE,ANGMO,NMAX
        PARAMETER (ZERO=0.0,TWO=2.0,NMAX=4)
        COMMON /CONTROL/ FORE,ANGMO
        COMMON /PROP/ N,M,BM(8),RL(4),LA(NMAX),EI(4),MM(4),JM(4)
```

```
      COMMON /PROP/ KSI(2*NMAX)
      COMMON /FIX/ MF(2+2*NMAX,2+2*NMAX),CO(3*NMAX+8),COM(3*NMAX,NMAX)
      COMMON /FIX/ E1(4*NMAX),G1(3*NMAX,NMAX),E2(4*NMAX),G2(3*NMAX,NMAX)
      COMMON /FIX/ BIJ1(3*NMAX,NMAX),BIJ2(3*NMAX,NMAX)
      COMMON /FIX/ ET1(NMAX,NMAX),ET2(NMAX,NMAX)
      COMMON /ANG1/ AN(3*NMAX+4)
C
C        This routine determines whether foreshortening constants are
C     to be evaluated or not, and then it evaluates all constants needed
C     for the simulation and initializes the mass matrix in MF and the
C     updating constants in CO and COM.
C
      DIMENSION CONL1(6),CONL2(6),PHI(NMAX,4),BI(NMAX),SI(NMAX)
      DIMENSION DAIJ(2+2*NMAX,2+2*NMAX)
      NR=2+N+M
      N2=2*N
      M2=2*M
      N3=3*N
      M3=3*M
C
C        Determine all integration constants from the input constants.
C
      CALL CONSTANT(CONL1,CONL2)
      CALL CONMOD(E1,G1,E2,G2)
      IF (FORE .EQ. 0) THEN
         CALL FOREST(BIJ1,BIJ2)
      ELSE
         CALL CMODES(N,LA,RL(3),RL(3),BI,PHI,SI)
         DO 100 I=1,N
            DO 200 J=1,N
               BIJ1(I,J)=ZERO
               BIJ1(N+I,J)=ZERO
               BIJ1(N2+I,J)=BM(2)*PHI(I,2)*PHI(J,2)
200         CONTINUE
100      CONTINUE
         CALL CMODES(M,LA,RL(4),RL(4),BI,PHI,SI)
         DO 300 I=1,M
            DO 400 J=1,M
               BIJ2(I,J)=ZERO
               BIJ2(M+I,J)=ZERO
               BIJ2(M2+I,J)=BM(4)*PHI(I,2)*PHI(J,2)
400         CONTINUE
300      CONTINUE
      ENDIF
C
C        Determine fixed parts of the mass matrix.
C
      SUM=JM(3)+JM(4)+CONL2(3)+TWO*CONL2(2)*BM(3)
      SUM=SUM+CONL2(1)*BM(3)*BM(3)+MM(3)*BM(6)*BM(6)
      MF(2,2)=SUM+MM(4)*CONL2(4)*CONL2(4)
      BASIC=MF(2,2)
      MF(2,1)=BASIC
      MF(1,2)=MF(2,1)
      SUM=BASIC+JM(1)+JM(2)+CONL1(3)+TWO*CONL1(2)*BM(1)
      SUM=SUM+CONL1(1)*BM(1)*BM(1)
      SUM=SUM+MM(2)*(BM(5)*BM(5)+TWO*CONL1(4)*CONL1(5))
      MASS=CONL2(1)+MM(2)+MM(3)+MM(4)
      MF(1,1)=SUM+MASS*CONL1(4)*CONL1(4)
      DO 500 J=1,N
         SUM1=BASIC+JM(2)+MM(2)*(BM(5)*BM(5)+CONL1(4)*CONL1(5))
```

```
            SUM1=E1(N3+J)*SUM1
            SUM1=SUM1+E1(N2+J)*(MASS*CONL1(4)+MM(2)*CONL1(5))
            MF(2+J,1)=SUM1+E1(N+J)+BM(1)*E1(J)
            MF(1,2+J)=MF(2+J,1)
            MF(2+J,2)=E1(N3+J)*BASIC
            MF(2,2+J)=MF(2+J,2)
            DO 600 I=1,J
               SUM2=BASIC+JM(2)+MM(2)*BM(5)*BM(5)
               SUM2=E1(N3+J)*E1(N3+I)*SUM2
               SUM2=SUM2+G1(N2+I,J)*MASS
               FAC=E1(N3+J)*E1(N2+I)+E1(N2+J)*E1(N3+I)
               SUM2=SUM2+MM(2)*CONL1(5)*FAC
               MF(2+J,2+I)=SUM2+G1(I,J)
               IF (I .NE. J) THEN
                  MF(2+I,2+J)=MF(2+J,2+I)
               ENDIF
  600       CONTINUE
  500    CONTINUE
         DO 700 J=1,M
            SUM=E2(M3+J)*JM(4)+E2(M+J)+BM(3)*E2(J)
            MF(2+N+J,1)=SUM+E2(M2+J)*MM(4)*CONL2(4)
            BASIC2=MF(2+N+J,1)
            MF(1,2+N+J)=MF(2+N+J,1)
            MF(2+N+J,2)=BASIC2
            MF(2,2+N+J)=MF(2+N+J,2)
            DO 800 I=1,N
               MF(2+N+J,2+I)=E1(N3+I)*BASIC2
               MF(2+I,2+N+J)=MF(2+N+J,2+I)
  800       CONTINUE
            DO 900 I=1,J
               SUM=E2(M3+J)*E2(M3+I)*JM(4)+G2(I,J)+MM(4)*G2(M2+I,J)
               MF(2+N+J,2+N+I)=SUM
               IF (I .NE. J) THEN
                  MF(2+N+I,2+N+J)=MF(2+N+J,2+N+I)
               ENDIF
  900       CONTINUE
  700    CONTINUE
C
C           Determine the coefficients for updating the mass matrix
C        and the offset vector b.
C
         CO(1)=CONL2(2)+BM(3)*CONL2(1)+MM(3)*CONL2(5)+MM(4)*CONL2(4)
         CO(2)=MM(3)*CONL2(6)
         CO(3)=CONL1(4)
         CO(4)=MM(4)
         CO(5)=MM(2)*CONL1(6)
         CO(6)=MASS
         CO(7)=MM(2)*CONL1(5)
         CO(8)=CONL2(4)
         DO 1000 J=1,N
            CO(8+J)=E1(N2+J)+CONL1(4)*E1(N3+J)
            CO(8+N+J)=E1(N2+J)-CONL1(4)*E1(N3+J)
            DO 1100 I=1,J
               COM(J,I)=E1(N3+I)*E1(N2+J)+E1(N3+J)*E1(N2+I)
               COM(N+J,I)=BIJ1(N+I,J)+BM(1)*BIJ1(I,J)
               IF (I .NE. J) THEN
                  COM(I,J)=COM(J,I)
                  COM(N+I,J)=COM(N+J,I)
               ENDIF
 1100       CONTINUE
```

```
 1000   CONTINUE
        DO 1200 J=1,M
           CO(8+N2+J)=E2(J)+MM(4)*E2(M2+J)
           DO 1300 I=1,J
              COM(N2+J,I)=BIJ2(M+I,J)+BM(3)*BIJ2(I,J)
              IF (I .NE. J) THEN
                 COM(N2+I,J)=COM(N2+J,I)
              ENDIF
 1300      CONTINUE
 1200   CONTINUE
C
C          Determine constant diagonal modal damping matrix.
C
        DO 1600 I=1,N
           ET1(I,I)=TWO*KSI(I)*SQRT(G1(N+I,I)*G1(I,I))
           DAIJ(2+I,2+I)=ET1(I,I)
           DO 1700 J=1,I-1
              ET1(I,J)=ZERO
              ET1(J,I)=ZERO
              DAIJ(2+I,2+J)=ET1(I,J)
              DAIJ(2+J,2+I)=ET1(J,I)
 1700      CONTINUE
 1600   CONTINUE
        DO 1800 I=1,M
           ET2(I,I)=TWO*KSI(NMAX+I)*SQRT(G2(M+I,I)*G2(I,I))
           DAIJ(2+N+I,2+N+I)=ET2(I,I)
           DO 1900 J=1,I-1
              ET2(I,J)=ZERO
              ET2(J,I)=ZERO
              DAIJ(2+N+I,2+N+J)=ET2(I,J)
              DAIJ(2+N+J,2+N+I)=ET2(J,I)
 1900      CONTINUE
 1800   CONTINUE
c       CALL WRMATRIX(NR,NR,DAIJ,'DAMP','CMATT0')
C
C          Determine the coefficients to update angular momentum
C       if needed.
C
        IF (ANGMO.EQ.1) THEN
           SUM=JM(1)+JM(2)+CONL1(3)+TWO*CONL1(2)*BM(1)
           SUM=SUM+CONL1(1)*BM(1)*BM(1)+CO(6)*CONL1(4)*CONL1(4)
           AN(1)=SUM+MM(2)*BM(5)*BM(5)+TWO*CO(7)*CONL1(4)
           SUM=JM(3)+JM(4)+CONL2(3)+TWO*CONL2(2)*BM(3)
           SUM=SUM+CONL2(1)*BM(3)*BM(3)+MM(3)*BM(6)*BM(6)
           AN(2)=SUM+MM(4)*CONL2(4)*CONL2(4)
           AN(3)=JM(2)+MM(2)*BM(5)*BM(5)+CONL1(4)*CO(7)
           AN(4)=JM(4)
           DO 1400 I=1,N
              AN(4+I)=E1(N+I)+BM(1)*E1(I)
 1400      CONTINUE
           DO 1500 I=1,M
              AN(4+N+I)=E2(M+I)+BM(3)*E2(I)
              AN(4+N+M+I)=E2(M3+I)
 1500      CONTINUE
        ENDIF
        RETURN
        END
C
C
C
```

```
      SUBROUTINE UPDATE(T,Y,TOR,MU,BU)
      IMPLICIT REAL*8 (A-Z)
      INTEGER N,M,I,J,N2,N3,M2,M3,MCOUNT,NMAX,NRE
      INTEGER FORE,ANGMO,SINGUL,RLINE,FNT,FOOL
      PARAMETER (ZERO=0.0,TWO=2.0,NMAX=4)
      COMMON /CONTROL/ FORE,ANGMO,SINGUL,RLINE,FOOL
      COMMON /PROP/ N,M
      COMMON /FIX/ MF(2+2*NMAX,2+2*NMAX),CO(3*NMAX+8),COM(3*NMAX,NMAX)
      COMMON /FIX/ E1(4*NMAX),G1(3*NMAX,NMAX),E2(4*NMAX),G2(3*NMAX,NMAX)
      COMMON /FIX/ BIJ1(3*NMAX,NMAX),BIJ2(3*NMAX,NMAX)
      COMMON /FIX/ ET1(NMAX,NMAX),ET2(NMAX,NMAX)
      COMMON /ANG2/ AQ,ADQ,PQ,PDQ,EP,EDP,BP,BDP,FNT
      COMMON /DAMP/ DDQ(NMAX),DDP(NMAX)
      COMMON /BUG/ TRA,TRE,TRI,TRO,TRU,CRA,NRE,CRI,CRO
      COMMON /BUG/ MCOUNT
      COMMON /OUS/ SAV(12,NMAX)
C
C        This routine is called by the DERIVS routine used by the
C     differential equation solver to determine the current value of
C     the derivatives to be integrated.  UPDATE returns in MU and BU
C     the current value of the mass matrix and offset vector b, given
C     the current "state" vector Y, and the current values
C     of the command joint torques TOR.
C
      DIMENSION MU(2+2*NMAX,2+2*NMAX)
      DIMENSION BU(2+2*NMAX),TOR(2),Y(3+4*NMAX)
      DIMENSION CQ(NMAX),GQ(NMAX),NBMQ(NMAX),AAQ(NMAX),HQ(NMAX)
      DIMENSION LCQ(NMAX),UP(NMAX),CP(NMAX),GP(NMAX),NBMP(NMAX)
      DIMENSION BBP(NMAX),HP(NMAX),LCP(NMAX)
C     DIMENSION DDQ(NMAX),DDP(NMAX)
      N2=2*N
      M2=2*M
      N3=3*N
      M3=3*M
C
C        Generate time dependent factors.  Link 1.
C
      CB=COS(Y(2))
      SB=SIN(Y(2))
      AQ=ZERO
      PQ=ZERO
      ADQ=ZERO
      PDQ=ZERO
C
C        If rate-linear eqs. flag set, make mass matrix indep. of
C     elastic deflections q and p, too.
C
      IF (RLINE.EQ.1) THEN
         DO 110 J=1,N
            CQ(J)=ZERO
            GQ(J)=ZERO
            NBMQ(J)=ZERO
            AAQ(J)=ZERO
            HQ(J)=ZERO
            DDQ(J)=ZERO
            DO 210 I=1,N
               HQ(J)=HQ(J)+G1(N+I,J)*Y(2+I)
               DDQ(J)=DDQ(J)+ET1(I,J)*Y(4+N+M+I)
 210        CONTINUE
            LCQ(J)=ZERO
```

```
110       CONTINUE
          ELSE
             DO 100 J=1,N
                AQ=AQ+E1(N2+J)*Y(2+J)
                PQ=PQ+E1(N3+J)*Y(2+J)
                ADQ=ADQ+E1(N2+J)*Y(4+N+M+J)
                PDQ=PDQ+E1(N3+J)*Y(4+N+M+J)
                CQ(J)=ZERO
                GQ(J)=ZERO
                NBMQ(J)=ZERO
                AAQ(J)=ZERO
                HQ(J)=ZERO
                DDQ(J)=ZERO
                DO 200 I=1,N
                   CQ(J)=CQ(J)+BIJ1(N2+I,J)*Y(2+I)
                   GQ(J)=GQ(J)+G1(I,J)*Y(2+I)
                   NBMQ(J)=NBMQ(J)+COM(N+I,J)*Y(2+I)
                   AAQ(J)=AAQ(J)+G1(N2+I,J)*Y(2+I)
                   HQ(J)=HQ(J)+G1(N+I,J)*Y(2+I)
                   DDQ(J)=DDQ(J)+ET1(I,J)*Y(4+N+M+I)
200             CONTINUE
                LCQ(J)=CO(3)*CQ(J)
100          CONTINUE
          ENDIF
C
C            Generate time dependent factors. Link 2.
C
C         CAB=CB-SB*PQ
C         SAB=CB*PQ+SB
          CAB=COS(Y(2)+PQ)
          SAB=SIN(Y(2)+PQ)
          EP=ZERO
          BP=ZERO
          EDP=ZERO
          BDP=ZERO
          IF (RLINE.EQ.1) THEN
             DO 310 J=1,M
                UP(J)=ZERO
                CP(J)=ZERO
                GP(J)=ZERO
                NBMP(J)=ZERO
                BBP(J)=ZERO
                HP(J)=ZERO
                DDP(J)=ZERO
                DO 410 I=1,M
                   HP(J)=HP(J)+G2(M+I,J)*Y(2+N+I)
                   DDP(J)=DDP(J)+ET2(I,J)*Y(4+N2+M+I)
410             CONTINUE
                LCP(J)=ZERO
310          CONTINUE
          ELSE
             DO 300 J=1,M
                EP=EP+E2(J)*Y(2+N+J)
                BP=BP+E2(M2+J)*Y(2+N+J)
                EDP=EDP+E2(J)*Y(4+N2+M+J)
                BDP=BDP+E2(M2+J)*Y(4+N2+M+J)
                UP(J)=ZERO
                CP(J)=ZERO
                GP(J)=ZERO
                NBMP(J)=ZERO
```

```
               BBP(J)=ZERO
               HP(J)=ZERO
               DDP(J)=ZERO
               DO 400 I=1,M
                  UP(J)=UP(J)+BIJ2(I,J)*Y(2+N+I)
                  CP(J)=CP(J)+BIJ2(M2+I,J)*Y(2+N+I)
                  GP(J)=GP(J)+G2(I,J)*Y(2+N+I)
                  NBMP(J)=NBMP(J)+COM(N2+I,J)*Y(2+N+I)
                  BBP(J)=BBP(J)+G2(M2+I,J)*Y(2+N+I)
                  HP(J)=HP(J)+G2(M+I,J)*Y(2+N+I)
                  DDP(J)=DDP(J)+ET2(I,J)*Y(4+N2+M+I)
 400           CONTINUE
               LCP(J)=CO(8)*CP(J)
 300        CONTINUE
         ENDIF
C
C           This branch statement allows the omission of mass
C        matrix updates when UPDATE is called from the FORCING
C        routine by setting T<0.0.
C
         IF (T.LT.ZERO) GOTO 2000
C
C           Update the mass matrix.
C
         MU(2,2)=MF(2,2)
         SUM=(CO(1)*SB+CO(2)*CB)*AQ-CO(3)*SB*EP
         SUM=SUM-CO(4)*CO(3)*SB*BP+CO(1)*CO(3)*CAB
         SPEC=SUM-CO(2)*CO(3)*SAB
         MU(2,1)=SPEC+MF(2,1)
         MU(1,2)=MU(2,1)
         MU(1,1)=TWO*(SPEC+CO(5)*AQ-CO(5)*CO(3)*PQ)+MF(1,1)
         DO 500 J=1,N
            SUM=(CO(1)*SB+CO(2)*CB+CO(5))*E1(N3+J)*AQ
            SUM=SUM-CO(5)*CO(8+J)*PQ-CO(8+J)*SB*EP
            SUM=SUM-CO(4)*CO(8+J)*SB*BP
            SAV(1,J)=(CO(1)*SB+CO(2)*CB+CO(5))*CQ(J)
            SUM=SUM+(CO(1)*SB+CO(2)*CB+CO(5))*CQ(J)
            MU(2+J,1)=SUM+(CO(1)*CAB-CO(2)*SAB)*CO(8+J)+MF(2+J,1)
            MU(1,2+J)=(CO(1)*CB-CO(2)*SB)*CO(8+J)+MF(1,2+J)
C           MU(1,2+J)=MU(2+J,1)
            SUM=-E1(N2+J)*SB*EP-CO(4)*E1(N2+J)*SB*BP
            SAV(2,J)=(CO(1)*SB+CO(2)*CB)*CQ(J)
            SUM=SUM+(CO(1)*SB+CO(2)*CB)*CQ(J)
            MU(2+J,2)=SUM+(CO(1)*CAB-CO(2)*SAB)*E1(N2+J)+MF(2+J,2)
            MU(2,2+J)=(CO(1)*CB-CO(2)*SB)*E1(N2+J)+MF(2,2+J)
C           MU(2,2+J)=MU(2+J,2)
            DO 600 I=1,J
c changed cab to cb, sab to sb
               MU(2+J,2+I)=COM(J,I)*(CO(1)*CB-CO(2)*SB)+MF(2+J,2+I)
               IF (I .NE. J) THEN
                  MU(2+I,2+J)=MU(2+J,2+I)
               ENDIF
 600        CONTINUE
 500     CONTINUE
         DO 700 J=1,M
            SAV(3,J)=CO(3)*SB*UP(J)
            SAV(4,J)=CO(4)*CO(3)*SB*CP(J)
            SUM=CO(8+N2+J)*SB*AQ-CO(3)*SB*UP(J)-CO(4)*CO(3)*SB*CP(J)
            MU(2+N+J,1)=SUM+CO(3)*CO(8+N2+J)*CAB+MF(2+N+J,1)
            MU(1,2+N+J)=CO(3)*CO(8+N2+J)*CB+MF(1,2+N+J)
```

```
C          MU(1,2+N+J)=MU(2+N+J,1)
           MU(2+N+J,2)=MF(2+N+J,2)
           MU(2,2+N+J)=MF(2,2+N+J)
           DO 800 I=1,N
C changed cb to cab. and back.
              MU(2+N+J,2+I)=E1(N2+I)*CO(8+N2+J)*CB+MF(2+N+J,2+I)
              MU(2+I,2+N+J)=MU(2+N+J,2+I)
 800       CONTINUE
           DO 900 I=1,J
              MU(2+N+J,2+N+I)=MF(2+N+J,2+N+I)
              IF (I .NE. J) THEN
                 MU(2+N+I,2+N+J)=MU(2+N+J,2+N+I)
              ENDIF
 900       CONTINUE
 700    CONTINUE
c       if ((t.gt.1.4684185).and.(t.lt.1.5)) then
c          print *,'cb',cb
c          print *,'y',y(1),y(2),y(3)
c          print *,'y2',y(4),y(5),y(6)
c       endif
C
C          Update the offset vector b=tor+f-kx.
C
 2000   CONTINUE
        OM3=Y(3+N+M)
        OOM3=Y(4+N+M)
        if ((abs(om3).gt.1.e15).or.(abs(oom3).gt.1.e15)) then
           print *,'y from UPDATE',y
           singul=13
           goto 5000
        endif
        LOMB=OM3+OOM3
        LOMB2=OM3*OM3+TWO*OM3*OOM3+OOM3*OOM3
        LIFOM=LOMB2-OM3*OM3
C
C          FNT is a control variable used to allow the mass matrix
C       to be evaluated w/o q terms, while the non-linear vector is
C       evaluated consistently.  FNT=1 does the above.
C
        FNT=0
C
        IF ((RLINE.EQ.1).AND.(FNT.EQ.1)) THEN
           DO 120 J=1,N
              AQ=AQ+E1(N2+J)*Y(2+J)
              PQ=PQ+E1(N3+J)*Y(2+J)
              ADQ=ADQ+E1(N2+J)*Y(4+N+M+J)
              PDQ=PDQ+E1(N3+J)*Y(4+N+M+J)
              CQ(J)=ZERO
              GQ(J)=ZERO
              NBMQ(J)=ZERO
              AAQ(J)=ZERO
              HQ(J)=ZERO
              DDQ(J)=ZERO
              DO 220 I=1,N
                 CQ(J)=CQ(J)+BIJ1(N2+I,J)*Y(2+I)
                 GQ(J)=GQ(J)+G1(I,J)*Y(2+I)
                 NBMQ(J)=NBMQ(J)+COM(N+I,J)*Y(2+I)
                 AAQ(J)=AAQ(J)+G1(N2+I,J)*Y(2+I)
                 HQ(J)=HQ(J)+G1(N+I,J)*Y(2+I)
                 DDQ(J)=DDQ(J)+ET1(I,J)*Y(4+N+M+I)
```

```
 220          CONTINUE
              LCQ(J)=CO(3)*CQ(J)
 120       CONTINUE
C          CAB=CB-SB*PQ
C          SAB=CB*PQ+SB
           CAB=COS(Y(2)+PQ)
           SAB=SIN(Y(2)+PQ)
           DO 320 J=1,M
              EP=EP+E2(J)*Y(2+N+J)
              BP=BP+E2(M2+J)*Y(2+N+J)
              EDP=EDP+E2(J)*Y(4+N2+M+J)
              BDP=BDP+E2(M2+J)*Y(4+N2+M+J)
              UP(J)=ZERO
              CP(J)=ZERO
              GP(J)=ZERO
              NBMP(J)=ZERO
              BBP(J)=ZERO
              HP(J)=ZERO
              DDP(J)=ZERO
              DO 420 I=1,M
                 UP(J)=UP(J)+BIJ2(I,J)*Y(2+N+I)
                 CP(J)=CP(J)+BIJ2(M2+I,J)*Y(2+N+I)
                 GP(J)=GP(J)+G2(I,J)*Y(2+N+I)
                 NBMP(J)=NBMP(J)+COM(N2+I,J)*Y(2+N+I)
                 BBP(J)=BBP(J)+G2(M2+I,J)*Y(2+N+I)
                 HP(J)=HP(J)+G2(M+I,J)*Y(2+N+I)
                 DDP(J)=DDP(J)+ET2(I,J)*Y(4+N2+M+I)
 420          CONTINUE
              LCP(J)=CO(8)*CP(J)
 320       CONTINUE
        ELSE
           IF (FOOL.EQ.1) THEN
              BU(1)=TOR(1)
              BU(2)=TOR(2)
              DO 1200 J=1,N
                 BU(2+J)=-HQ(J)-DDQ(J)
 1200         CONTINUE
              DO 1300 J=1,M
                 BU(2+N+J)=-HP(J)-DDP(J)
 1300         CONTINUE
              GOTO 5000
           ENDIF
        ENDIF
C
C          The above IF statement allows the simulation of rate-linear
C       equations.  1/20/89- Modified to simulate rate-linear in small
C       elastic rates, but non-linear in rigid body rates.
C
        OMB=OM3+OOM3+PDQ
        NOMB2=TWO*(OM3+OOM3)*PDQ
        OMB2=LOMB2+NOMB2
        DIFOM=LOMB2+NOMB2-OM3*OM3
        COEF=(-CO(1)*CB+CO(2)*SB)*AQ+CO(3)*CB*EP
        COEF=COEF+CO(4)*CO(3)*CB*BP+CO(3)*CO(1)*SAB
        COEF=COEF+CO(3)*CO(2)*CAB
        SUM=-TWO*OM3*((CO(1)*SB+CO(2)*CB+CO(5))*ADQ-CO(5)*CO(3)*PDQ)
        SUM=SUM+LIFOM*COEF+NOMB2*(CO(3)*CO(1)*SB+CO(3)*CO(2)*CB)
        SUM=SUM+TWO*LOMB*(CO(3)*SB*EDP+CO(4)*CO(3)*SB*BDP)
        BU(1)=SUM+TOR(1)
        SUM=-OM3*OM3*COEF-TWO*OM3*(CO(1)*SB+CO(2)*CB)*ADQ
```

```
      BU(2)=SUM+TOR(2)
      DO 1000 J=1,N
         SAV(5,J)=NBMQ(J)
         SAV(6,J)=CO(6)*LCQ(J)
         SUM1=GQ(J)-NBMQ(J)+CO(6)*AAQ(J)-CO(6)*LCQ(J)
         SUM1=SUM1+(CO(1)*CB-CO(2)*SB+CO(7))*E1(N3+J)*AQ
         SUM1=SUM1+CO(7)*CO(8+N+J)*PQ-E1(N3+J)*CO(3)*CB*EP
         SAV(7,J)=CO(7)*CQ(J)
         SUM1=SUM1-E1(N3+J)*CO(4)*CO(3)*CB*BP-CO(7)*CQ(J)
         SUM1=SUM1-E1(N3+J)*(CO(1)*SAB+CO(2)*CAB)*CO(3)
         SUM1=SUM1+CO(5)*CO(8+N+J)
         SUM2=CB*EP+CO(4)*CB*BP+CO(1)*SAB+CO(2)*CAB
         SAV(8,J)=(-CO(1)*CB+CO(2)*SB)*CQ(J)
         SUM2=E1(N2+J)*SUM2+(-CO(1)*CB+CO(2)*SB)*CQ(J)
         SUM3=E1(N3+J)*(CO(1)*SB+CO(2)*CB+CO(5))*ADQ
         SUM3=SUM3-CO(5)*E1(N2+J)*PDQ
         SUM=OM3*OM3*SUM1+LOMB2*SUM2-TWO*OM3*SUM3
         SUM=SUM+NOMB2*E1(N2+J)*(CO(1)*SB+CO(2)*CB)
         SUM=SUM+TWO*LOMB*E1(N2+J)*(SB*EDP+CO(4)*SB*BDP)
         BU(2+J)=SUM-HQ(J)-DDQ(J)
 1000 CONTINUE
      DO 1100 J=1,M
         SAV(9,J)=CO(4)*CO(3)*CB*CP(J)
         SAV(10,J)=CO(3)*CB*UP(J)
         SAV(11,J)=NBMP(J)
         SAV(12,J)=CO(4)*LCP(J)
         SUM1=-CO(8+N2+J)*CB*AQ+CO(4)*CO(3)*CB*CP(J)
         SUM1=SUM1+CO(3)*CB*UP(J)+CO(3)*CO(8+N2+J)*SAB
         SUM2=GP(J)-NBMP(J)+CO(4)*(BBP(J)-LCP(J))
         SUM=-OM3*OM3*SUM1+LOMB2*SUM2-TWO*OM3*CO(8+N2+J)*SB*ADQ
         BU(2+N+J)=SUM-HP(J)-DDP(J)
 1100 CONTINUE
 5000 CONTINUE
      RETURN
      END
C
C
C
      INCLUDE '[USER.AVF.CARLOS.FORTRAN]DARMSEC.FOR'
C
C        This last command includes the rest of the main subroutines
C     that are placed in ARMSEC.FOR for ease of handling.
C
```

```
C*************************************************************************
C*************************************************************************
C          These FORTRAN subroutines evaluate the constants to be used
C       by the INTIALIZE and UPDATE routines in the two-link flexible
C       arm simulation.  Written by Carlos E. Padilla for his M.S.
C       thesis on 7-9-88.
C       DOUBLE PRECISION version.
C*************************************************************************
C*************************************************************************
        REAL*8 FUNCTION SINH(X)
        REAL*8 X
        SINH=(EXP(X)-EXP(-X))/2.0
        RETURN
        END
C
        REAL*8 FUNCTION COSH(X)
        REAL*8 X
        COSH=(EXP(X)+EXP(-X))/2.0
        RETURN
        END
C
C          As of now, given # of m.s. N, lambdai, length L, argument X,
C       the subroutine returns the modeshapes and the first three
C       derivatives d/dx in PHI(NMAX,4). Also the betai=lambdai/L
C       are returned in array BI(NMAX).  Further, the sigmai are
C       returned in SI(NMAX).
C
        SUBROUTINE CMODES(N,LA,L,X,BI,PHI,SI)
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,I,J,NMAX
        PARAMETER (NMAX=4)
        DIMENSION LA(NMAX),PHI(NMAX,4),BI(NMAX),SI(NMAX)
C
        DO 100 I=1,N
            SI(I)=(SINH(LA(I))-SIN(LA(I)))/(COSH(LA(I))+COS(LA(I)))
100     CONTINUE
C
        DO 200 J=1,N
            BI(J)=LA(J)/L
            BX=BI(J)*X
            SX=SIN(BX)
            CX=COS(BX)
            SINHX=SINH(BX)
            COSHX=COSH(BX)
C
            PHI(J,1)=(COSHX-CX-SI(J)*(SINHX-SX))
            PHI(J,2)=BI(J)*(SINHX+SX-SI(J)*(COSHX-CX))
            PHI(J,3)=BI(J)*BI(J)*(COSHX+CX-SI(J)*(SINHX+SX))
            PHI(J,4)=BI(J)*BI(J)*BI(J)*(SINHX-SX-SI(J)*(COSHX+CX))
 200    CONTINUE
        RETURN
        END
C
C
C
        SUBROUTINE CONSTANT(CONL1,CONL2)
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M
        COMMON /PROP/ N,M,BM(8),RL(4)
C
```

```
C          This routine evaluates constants that do not depend on the
C       mode shapes.  CONL1 on exit contains the constants for link 1
C       while CONL2 contains the constants for link2.
C
        DIMENSION CONL1(6),CONL2(6)
C
C          Determine constants for link 1.
C
        CONL1(1)=RL(1)*RL(3)
        CONL1(2)=CONL1(1)*RL(3)/2.0
        CONL1(3)=CONL1(2)*RL(3)*2.0/3.0
        CONL1(4)=BM(1)+BM(2)+RL(3)
        CONL1(5)=BM(5)*COS(BM(7))
        CONL1(6)=BM(5)*SIN(BM(7))
C
C          Determine constants for link 2.
C
        CONL2(1)=RL(2)*RL(4)
        CONL2(2)=CONL2(1)*RL(4)/2.0
        CONL2(3)=CONL2(2)*RL(4)*2.0/3.0
        CONL2(4)=BM(3)+BM(4)+RL(4)
        CONL2(5)=BM(6)*COS(BM(8))
        CONL2(6)=BM(6)*SIN(BM(8))
        RETURN
        END
C
C
C
        SUBROUTINE CONMOD(E1,G1,E2,G2)
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,I,J,N2,N3,M2,M3,NMAX,NR
        PARAMETER (ZERO=0.0,TWO=2.0,NMAX=4,SCALE=1.0)
        COMMON /PROP/ N,M,BM(8),RL(4),LA(4),EI(4)
C
C          This routine evaluates constants that depend on the mode
C       shapes but that do not include foreshortening.  N is the #
C       of modes for link 1.  M is the # of modes for link 2.  On exit
C       E1 contains the single-indexed mode constants for link 1 and
C       G1 contains the double-indexed mode constants for link 1.
C       Likewise for link 2.
C
        DIMENSION E1(4*NMAX),G1(3*NMAX,NMAX),E2(4*NMAX),G2(3*NMAX,NMAX)
        DIMENSION BI(NMAX),PHI(NMAX,4),SI(NMAX),KSIF(2+2*NMAX,2+2*NMAX)
        DIMENSION MMASS(2+2*NMAX,2+2*NMAX)
        L1=RL(3)
        L2=RL(4)
        NR=2+N+M
        N2=2*N
        M2=2*M
        N3=3*N
        M3=3*M
        SCA2=SCALE*SCALE
C
C          Determine constants for link 1.
C
        CALL CMODES(N,LA,L1,L1,BI,PHI,SI)
        DO 100 I=1,N
           E1(I)=SCALE*TWO*RL(1)*SI(I)/BI(I)
           E1(N+I)=SCALE*TWO*RL(1)/(BI(I)*BI(I))
           E1(N2+I)=SCALE*(PHI(I,1)+BM(2)*PHI(I,2))
```

```
            E1(N3+I)=SCALE*PHI(I,2)
            G1(I,I)=SCA2*RL(1)*RL(3)
            G1(N+I,I)=SCA2*BI(I)*BI(I)*BI(I)*BI(I)*RL(3)*EI(1)*EI(2)
C
C           Note: KSIF is used to save the values of the constant flex.
C         coordinates stiffness matrix.  MMASS is used to save the values
C         of the constant modal mass matrix.
C
            MMASS(2+I,2+I)=G1(I,I)
            KSIF(2+I,2+I)=G1(N+I,I)
            G1(N2+I,I)=E1(N2+I)*E1(N2+I)
            DO 200 J=1,I-1
               G1(I,J)=SCA2*ZERO
               G1(J,I)=G1(I,J)
               G1(N+I,J)=SCA2*ZERO
               G1(N+J,I)=G1(N+I,J)
               MMASS(2+I,2+J)=G1(I,J)
               MMASS(2+J,2+I)=G1(J,I)
               KSIF(2+I,2+J)=G1(N+I,J)
               KSIF(2+J,2+I)=G1(N+J,I)
               G1(N2+I,J)=E1(N2+I)*E1(N2+J)
               G1(N2+J,I)=G1(N2+I,J)
 200        CONTINUE
 100     CONTINUE
C
C           Determine constants for link 2.
C
         CALL CMODES(M,LA,L2,L2,BI,PHI,SI)
C
C           Note: LA is the same as for the above case in our special
C         case of catilever modes for both links.
C
         DO 300 I=1,M
            E2(I)=SCALE*TWO*RL(2)*SI(I)/BI(I)
            E2(M+I)=SCALE*TWO*RL(2)/(BI(I)*BI(I))
            E2(M2+I)=SCALE*(PHI(I,1)+BM(4)*PHI(I,2))
            E2(M3+I)=SCALE*PHI(I,2)
            G2(I,I)=SCA2*RL(2)*RL(4)
            G2(M+I,I)=SCA2*BI(I)*BI(I)*BI(I)*BI(I)*RL(4)*EI(3)*EI(4)
            MMASS(2+N+I,2+N+I)=G2(I,I)
            KSIF(2+N+I,2+N+I)=G2(M+I,I)
            G2(M2+I,I)=E2(M2+I)*E2(M2+I)
            DO 400 J=1,I-1
               G2(I,J)=SCA2*ZERO
               G2(J,I)=G2(I,J)
               G2(M+I,J)=SCA2*ZERO
               G2(M+J,I)=G2(M+I,J)
               MMASS(2+N+I,2+N+J)=G2(I,J)
               MMASS(2+N+J,2+N+I)=G2(J,I)
               KSIF(2+N+I,2+N+J)=G2(M+I,J)
               KSIF(2+N+J,2+N+I)=G2(M+J,I)
               G2(M2+I,J)=E2(M2+I)*E2(M2+J)
               G2(M2+J,I)=G2(M2+I,J)
 400        CONTINUE
 300     CONTINUE
c        CALL WRMATRIX(NR,NR,MMASS,'MMAS','MMATT0')
c        CALL WRMATRIX(NR,NR,KSIF,'STIF','KWATT0')
         CALL MATMATRIX(NR,NR,KSIF,'STIF','KMATT0')
         RETURN
         END
```

```
C
C
C
      SUBROUTINE FOREST(BIJ1,BIJ2)
      IMPLICIT REAL*8 (A-Z)
      INTEGER N,M,I,J,N2,M2,NMAX,NR
      PARAMETER (ONE=1.0,TWO=2.0,NMAX=4,SCALE=1.0)
      COMMON /PROP/ N,M,BM(8),RL(4),LA(4)
C
C        This subroutine evaluates the modal constants that arise
C     due to foreshortening.  N is the # of modes for link 1, and
C     M is the # of modes for link 2.  On return BIJ1 will contain
C     the constants (all double indexed) pertaining to link 1.
C     Likewise, BIJ2 will contain those of link 2.
C
      DIMENSION BIJ1(3*NMAX,NMAX),BIJ2(3*NMAX,NMAX),BI(NMAX)
      DIMENSION PHI(NMAX,4),SI(NMAX),BI3(NMAX),BI4(NMAX)
      DIMENSION MUIJ(2+2*NMAX,2+2*NMAX),ETIJ(2+2*NMAX,2+2*NMAX)
      DIMENSION CIJ1(2+2*NMAX,2+2*NMAX)
      L1=RL(3)
      L2=RL(4)
      NR=2+N+M
      N2=2*N
      M2=2*M
      SCA2=SCALE*SCALE
C
C        Determine constants for link 1.
C
      CALL CMODES(N,LA,L1,L1,BI,PHI,SI)
      L13=L1*L1*L1
      L12=L1*L1/TWO
      DO 100 I=1,N
         BI3(I)=BI(I)*BI(I)*BI(I)
         BI4(I)=BI(I)*BI3(I)
         SUM=3.0/TWO-11.0/8.0*SI(I)*BI(I)*L1
         SUM=SUM+ONE/TWO*(SI(I)*BI(I)*L1)**2
         BIJ1(I,I)=SCA2*RL(1)*SUM
         MUIJ(2+I,2+I)=BIJ1(I,I)
         SUM=5.0/4.0*L1+L13/(12.0*BI4(I))*PHI(I,4)*PHI(I,4)
         SUM=SUM-ONE/6.0*L13*PHI(I,1)*PHI(I,3)
         SUM=SUM+L13/12.0*PHI(I,2)*PHI(I,2)
         BIJ1(N+I,I)=SCA2*RL(1)*(SUM-L12*SI(I)*BI(I))
         ETIJ(2+I,2+I)=BIJ1(N+I,I)
         SUM=SI(I)*BI(I)*(TWO+SI(I)*BI(I)*L1)
         BIJ1(N2+I,I)=SCA2*(SUM+BM(2)*PHI(I,2)*PHI(1,2))
         CIJ1(2+I,2+I)=BIJ1(N2+I,I)
         DO 200 J=1,I-1
            DIV=BI4(I)-BI4(J)
            COEF=RL(1)/DIV
            SUM=8.0*BI4(I)*(BI(I)*BI(J))**2/DIV
            SUM=SUM+8.0*BI4(J)*(BI(I)*BI(J))**2/DIV
            SUM=SUM-(-1.0)**(I+J)*16.0*BI4(I)*BI4(J)/DIV
            LAC1=4.0*BI(J)*BI(J)*BI3(I)*SI(I)
            LAC2=4.0*BI(I)*BI(I)*BI3(J)*SI(J)
            SUM=SUM-L1*LAC1+L1*LAC2
            BIJ1(I,J)=SCA2*COEF*SUM
            BIJ1(J,I)=BIJ1(I,J)
            MUIJ(2+I,2+J)=BIJ1(I,J)
            MUIJ(2+J,2+I)=BIJ1(J,I)
            SUM=-4.0*BI4(I)*BI4(J)/DIV*L1*PHI(I,1)*PHI(J,1)
```

```
            FAC1=L1*PHI(I,2)*PHI(J,4)
            FAC2=L1*PHI(I,4)*PHI(J,2)
            FAC=-FAC1+L1*PHI(I,3)*PHI(J,3)-FAC2
            SUM=SUM-TWO*(BI4(I)+BI4(J))/DIV*FAC
            SUM=SUM+PHI(I,1)*PHI(J,4)-PHI(I,4)*PHI(J,1)
            SUM=SUM-TWO*PHI(I,2)*PHI(J,3)+TWO*PHI(I,3)*PHI(J,2)
            SUM=SUM-FAC2+FAC1-L12*LAC1
            BIJ1(N+I,J)=SCA2*COEF*(SUM+L12*LAC2)
            BIJ1(N+J,I)=BIJ1(N+I,J)
            ETIJ(2+I,2+J)=BIJ1(N+I,J)
            ETIJ(2+J,2+I)=BIJ1(N+J,I)
            SUM1=SI(J)*BI4(I)*BI(J)-SI(I)*BI4(J)*BI(I)
            SUM1=4.0*(-ONE)**(I+J)*SUM1/DIV
            SUM=4.0*(BI(I)*BI(J))**2*(SI(I)*BI(I)-SI(J)*BI(J))/DIV
            SUM=SUM1-SUM
            BIJ1(N2+I,J)=SCA2*(SUM+BM(2)*PHI(I,2)*PHI(J,2))
            BIJ1(N2+J,I)=BIJ1(N2+I,J)
            CIJ1(2+I,2+J)=BIJ1(N2+I,J)
            CIJ1(2+J,2+I)=BIJ1(N2+I,J)
 200        CONTINUE
 100     CONTINUE
C
C           Determine constants for link 2.
C
         CALL CMODES(M,LA,L2,L2,BI,PHI,SI)
         L23=L2*L2*L2
         L22=L2*L2/TWO
         DO 300 I=1,M
            BI3(I)=BI(I)*BI(I)*BI(I)
            BI4(I)=BI(I)*BI3(I)
            SUM=3.0/TWO-11.0/8.0*SI(I)*BI(I)*L2
            SUM=SUM+ONE/TWO*(SI(I)*BI(I)*L2)**2
            BIJ2(I,I)=SCA2*RL(2)*SUM
            MUIJ(2+N+I,2+N+I)=BIJ2(I,I)
            SUM=5.0/4.0*L2+L23/(12.0*BI4(I))*PHI(I,4)*PHI(I,4)
            SUM=SUM-ONE/8.0*L23*PHI(I,1)*PHI(I,3)
            SUM=SUM+L23/12.0*PHI(I,2)*PHI(I,2)
            BIJ2(M+I,I)=SCA2*RL(2)*(SUM-L22*SI(I)*BI(I))
            ETIJ(2+N+I,2+N+I)=BIJ2(M+I,I)
            SUM=SI(I)*BI(I)*(TWO+SI(I)*BI(I)*L2)
            BIJ2(M2+I,I)=SCA2*(SUM+BM(4)*PHI(I,2)*PHI(I,2))
            CIJ1(2+N+I,2+N+I)=BIJ2(M2+I,I)
            DO 400 J=1,I-1
               DIV=BI4(I)-BI4(J)
               COEF=RL(2)/DIV
               SUM=8.0*BI4(I)*(BI(I)*BI(J))**2/DIV
               SUM=SUM+8.0*BI4(J)*(BI(I)*BI(J))**2/DIV
               SUM=SUM-(-1.0)**(I+J)*16.0*BI4(I)*BI4(J)/DIV
               LAC1=4.0*BI(J)*BI(J)*BI3(I)*SI(I)
               LAC2=4.0*BI(I)*BI(I)*BI3(J)*SI(J)
               SUM=SUM-L2*LAC1+L2*LAC2
               BIJ2(I,J)=SCA2*COEF*SUM
               BIJ2(J,I)=BIJ2(I,J)
               MUIJ(2+N+I,2+N+J)=BIJ2(I,J)
               MUIJ(2+N+J,2+N+I)=BIJ2(J,I)
               SUM=-4.0*BI4(I)*BI4(J)/DIV*L2*PHI(I,1)*PHI(J,1)
               FAC1=L2*PHI(I,2)*PHI(J,4)
               FAC2=L2*PHI(I,4)*PHI(J,2)
               FAC=-FAC1+L2*PHI(I,3)*PHI(J,3)-FAC2
               SUM=SUM-TWO*(BI4(I)+BI4(J))/DIV*FAC
```

```
                SUM=SUM+PHI(I,1)*PHI(J,4)-PHI(I,4)*PHI(J,1)
                SUM=SUM-TWO*PHI(I,2)*PHI(J,3)+TWO*PHI(I,3)*PHI(J,2)
                SUM=SUM-FAC2+FAC1-L22*LAC1
                BIJ2(M+I,J)=SCA2*COEF*(SUM+L22*LAC2)
                BIJ2(M+J,I)=BIJ2(M+I,J)
                ETIJ(2+N+I,2+N+J)=BIJ2(M+I,J)
                ETIJ(2+N+J,2+N+I)=BIJ2(M+J,I)
                SUM1=SI(J)*BI4(I)*BI(J)-SI(I)*BI4(J)*BI(I)
                SUM1=4.0*(-ONE)**(I+J)*SUM1/DIV
                SUM=4.0*(BI(I)*BI(J))**2*(SI(I)*BI(I)-SI(J)*BI(J))/DIV
                SUM=SUM1-SUM
                BIJ2(M2+I,J)=SCA2*(SUM+BM(4)*PHI(I,2)*PHI(J,2))
                BIJ2(M2+J,I)=BIJ2(M2+I,J)
                CIJ1(2+N+I,2+N+J)=BIJ2(M2+I,J)
                CIJ1(2+N+J,2+N+I)=BIJ2(M2+J,I)
 400          CONTINUE
 300       CONTINUE
c          CALL WRMATRIX(NR,NR,MUIJ,'MUIJ','MUIJT0')
c          CALL WRMATRIX(NR,NR,ETIJ,'ETIJ','METAT0')
C          CALL MATMATRIX(NR,NR,MUIJ,'MUIJ','MUIJT0')
C          CALL MATMATRIX(NR,NR,ETIJ,'ETIJ','METAT0')
           CALL WRMATRIX(NR,NR,CIJ1,'CIJ1','CIJ1T0')
           RETURN
           END
```

```
C*********************************************************************
C*********************************************************************
C          This file contains the rest of the main subroutines for the
C       two-link flex. arm simulation.  The following are all written
C       by Carlos E. Padilla.  This file is included by the FORTRAN
C       command at the end of the file ARMAIN.FOR.
C       DOUBLE PRECISION version.
C*********************************************************************
C*********************************************************************
        SUBROUTINE DERIVS(T,Y,DYDT)
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,I,J,NR,FORE,ANGMO,SINGUL,NMAX,COUNT
        PARAMETER (ZERO=0.0,NMAX=4)
        character dat*9,tim*9,AC(9)
        COMMON /CONTROL/ FORE,ANGMO,SINGUL
        COMMON /PROP/ N,M
        COMMON /DAMP/ DDQ(NMAX),DDP(NMAX)
        COMMON /ENER/ ME(2+2*NMAX,2+2*NMAX)
C
C          This is the user provided routine to determine the
C       time derivative of Y, DY/DT, used by the numerical recipes
C       ODEINT routine to solve the differential equations.
C
        DIMENSION Y(8+4*NMAX),DYDT(8+4*NMAX),MU(2+2*NMAX,2+2*NMAX)
        DIMENSION BU(2+2*NMAX),TOR(2)
        dimension be(2+2*nmax)
        NR=2+N+M
C
        DYDT(1)=Y(1+NR)
        DYDT(2)=Y(2+NR)
        DO 100 I=1,N
           DYDT(2+I)=Y(2+NR+I)
 100    CONTINUE
        DO 200 I=1,M
           DYDT(2+N+I)=Y(2+NR+N+I)
 200    CONTINUE
C
        CALL FORCING(T,Y,TOR)
        CALL UPDATE(T,Y,TOR,MU,BU)
c       if ((t.gt.0.5993199).and.(t.lt.0.6)) then
c          print *,t
c          print *,y(2)
c          do 110 i=1,nr
c             print *,(mu(i,j),j=1,nr)
c 110      continue
c       endif
        IF (SINGUL.NE.0) GOTO 3000
C
C          This is to output MU and BU for a small number of times.
C
c       DATA AC/'1','2','3','4','5','6','7','8','9'/
c       IF (T.GE.0.5993211) THEN
c          IF (COUNT.LE.8) THEN
c             PRINT *,T,y(2),Y(nr+2)
c             COUNT=COUNT+1
c             CALL MATMATRIX(NR,NR,MU,'MUSI','MUSIM'//AC(COUNT))
c             CALL MATMATRIX(NR,1,BU,'BUSI','BUSIM'//AC(COUNT))
c          ENDIF
c       ENDIF
C
```

```
C          Derivatives for "integration checks."  Angular momentum,
C       energy due to torques, and energy dissipation due to damping.
C
        DYDT(1+2*NR)=TOR(1)
        DYDT(2+2*NR)=TOR(1)*Y(1+NR)
        DYDT(3+2*NR)=TOR(2)*Y(2+NR)
        SUM=ZERO
        DO 800 I=1,N
           SUM=SUM+DDQ(I)*Y(2+NR+I)
 800    CONTINUE
        DO 900 I=1,M
           SUM=SUM+DDP(I)*Y(2+NR+N+I)
 900    CONTINUE
        DYDT(4+2*NR)=SUM
C
C          If checking of energy is on, save MU in ME.
C
c       if (t.eq.zero) then
        IF (ANGMO.EQ.1) THEN
           DO 400 I=1,NR
              be(I)=bu(I)
              DO 500 J=1,I
                 ME(I,J)=MU(I,J)
                 IF (I.NE.J) ME(J,I)=MU(J,I)
 500          CONTINUE
 400       CONTINUE
        ENDIF
c       else
c          do 600 i=1,nr
c             do 700 j=1,i
c                mu(i,j)=me(i,j)
c                if (i.ne.j) mu(j,i)=me(j,i)
c 700         continue
c 600      continue
c       endif
C
C          The following lines do row scaling of the system MU=BU
C       so as to improve the conditioning of the mass matrix by
C       setting the infinity norms of the rows to 1.
C
c       DO 1100 I=1,NR
c          SON=MU(I,1)
c          DO 1200 J=2,NR
c             IF (SON.LT.MU(I,J)) SON=MU(I,J)
c 1200     CONTINUE
c          DO 1300 J=1,NR
c             MU(I,J)=1.0/SON*MU(I,J)
c 1300     CONTINUE
c          BU(I)=1.0/SON*BU(I)
c 1100  CONTINUE
C
C          Invert the mass matrix to obtain the acceleration vector.
C
        if (t.eq.zero) call matmatrix(nr,nr,mu,'MUT0','MUMATX')
c       if (t.eq.zero) call wrmatrix(nr,nr,mu,'mut0','M0WRIT')
        CALL LUDEC(NR,NR,MU)
        if (singul.ne.0) then
           open (1,file='[user.avf.carlos.thesis]overflow',status='unknown',
     &          access='append',form='formatted')
           call date(dat)
```

```
            call time(tim)
            write(1,*)dat,' ',tim
            write(1,*)'t=',t
            write(1,*)'y=',y
            close (1)
c           if (angmo.eq.1) then
c              call matmatrix(nr,nr,me,'MUSI','MUMATX')
c              call wrmatrix(nr,nr,me,'musi','MUWRIT')
c           endif
         endif
         IF (SINGUL.NE.0) GOTO 1000
         CALL LUSOLVE(NR,NR,MU,BU)
         IF (SINGUL.NE.0) GOTO 2000
C
         DO 300 I=1,NR
            DYDT(NR+I)=BU(I)
 300     CONTINUE
C
C           The following two lines are to be used to freeze the
C        shoulder link for purposes of spinning of second link only.
C        They should be commented out during normal simulations.
C
c        dydt(1)=zero
c        dydt(1+nr)=zero
         GOTO 5000
 1000    PAUSE 'Mass matrix non-positve-definite. Aborting.'
         call matmatrix(nr,nr,me,'musi','musimx')
         GOTO 5000
 2000    PAUSE 'State vector growing without bound. Aborting.'
         call matmatrix(nr,nr,me,'musi','musimx')
         call matmatrix(nr,1,be,'besi','besimx')
         call matmatrix(nr,1,bu,'busi','busimx')
         GOTO 5000
 3000    PAUSE 'Rates too large in UPDATE. Aborting.'
         call matmatrix(nr,nr,mu,'musi','musimx')
 5000    CONTINUE
         RETURN
         END
C
C
C
         SUBROUTINE INITIME(T)
         IMPLICIT REAL*8 (A-Z)
         INTEGER N,M,NMAX
         PARAMETER (ONE=1.0,TWO=2.0,NMAX=4,API=3.141592654,SAFETY=0.9)
         COMMON /PROP/ N,M,BM(8),RL(4),LA(NMAX),EI(4)
C
C           This subroutine determines a trial initial time step 'H1'
C        which is returned in T.  It is based on the highest freq.
C        mode shape times a safety factor.
C
         L1=RL(3)
         L2=RL(4)
         F1=LA(N)**2/(TWO*API*L1*L1)*SQRT(EI(1)*EI(2)/RL(1))
         F2=LA(M)**2/(TWO*API*L2*L2)*SQRT(EI(3)*EI(4)/RL(2))
         IF (F1 .GT. F2) THEN
            F=F1
         ELSE
            F=F2
         ENDIF
```

```
      T=ONE/10.0*SAFETY*ONE/F
      RETURN
      END
C
C
C
      SUBROUTINE AMOMENTUM(T,Y,H,DELH)
      IMPLICIT REAL*8 (A-Z)
      INTEGER N,M,I,J,FORE,ANGMO,N2,NMAX,NR,SING,RLINE,FNT
      PARAMETER (ZERO=0.0,TWO=2.0,NMAX=4)
      COMMON /CONTROL/ FORE,ANGMO,SING,RLINE
      COMMON /PROP/ N,M
      COMMON /FIX/ MF(2+2*NMAX,2+2*NMAX),CO(3*NMAX+8),COM(3*NMAX,NMAX)
      COMMON /FIX/ E1(4*NMAX),G1(3*NMAX,NMAX),E2(4*NMAX),G2(3*NMAX,NMAX)
      COMMON /ANG1/ AN(3*NMAX+4)
      COMMON /ANG2/ AQ,ADQ,PQ,PDQ,EP,EDP,BP,BDP,FNT
      COMMON /ANG3/ HREF
C
C        This routine determines the angular momentum of the arm.
C     It is called by ODEINT if the control variable ANGMO is set
C     to 1 by the input program.  It uses the constants CO and
C     AN evaluated in the INITIALIZE routine and the time variables
C     evaluated in the UPDATE routine placed in the COMMON block
C     ANG2.
C
      DIMENSION Y(8+4*NMAX)
      N2=2*N
      NR=2+N+M
C
C        Determine time dependent factors not in COMMON block.
C
      CB=COS(Y(2))
      SB=SIN(Y(2))
      OM3=Y(3+N+M)
      OOM3=Y(4+N+M)
      LOMB=OM3+OOM3
C     IF ((RLINE.EQ.1).AND.(FNT.NE.1)) THEN
C        DO 300 J=1,N
C           AQ=AQ+E1(N2+J)*Y(2+J)
C           PQ=PQ+E1(N3+J)*Y(2+J)
C           ADQ=ADQ+E1(N2+J)*Y(4+N+M+J)
C           PDQ=PDQ+E1(N3+J)*Y(4+N+M+J)
C 300    CONTINUE
C        DO 400 J=1,M
C           EP=EP+E2(J)*Y(2+N+J)
C           BP=BP+E2(M2+J)*Y(2+N+J)
C           EDP=EDP+E2(J)*Y(4+N2+M+J)
C           BDP=BDP+E2(M2+J)*Y(4+N2+M+J)
C 400    CONTINUE
C     ENDIF
C     CAB=CB-SB*PQ
C     SAB=CB*PQ+SB
      CAB=COS(Y(2)+PQ)
      SAB=SIN(Y(2)+PQ)
      OMB=OM3+OOM3+PDQ
      FBEQ=ZERO
      FBEP=ZERO
      PDP=ZERO
      DO 100 I=1,N
         FBEQ=FBEQ+AN(4+I)*Y(4+N+M+I)
```

```
 100    CONTINUE
        DO 200 I=1,M
           PDP=PDP+AN(4+N+M+I)*Y(4+N2+M+I)
           FBEP=FBEP+AN(4+N+I)*Y(4+N2+M+I)
 200    CONTINUE
        SUM=OM3*(AN(1)+TWO*CO(5)*AQ-TWO*CO(3)*CO(5)*PQ)
        SUM=SUM+OMB*AN(2)
        FAC=CO(1)*CO(3)*CAB-CO(2)*CO(3)*SAB-CO(3)*SB*EP
        FAC=FAC+(CO(1)*SB+CO(2)*CB)*AQ-CO(4)*CO(3)*SB*BP
C
C          On 1/23/89 changed omb to lomb to linearize consistently.
C
        SUM=SUM+(OM3+LOMB)*FAC+AN(3)*PDQ
        SUM=SUM+PDQ*(CO(1)*CO(3)*CB-CO(2)*CO(3)*SB)
        SUM=SUM+(CO(6)*CO(3)+CO(7)+CO(1)*CB-CO(2)*SB)*ADQ
        SUM=SUM+CO(3)*CB*EDP+(CO(8)+CO(3)*CB)*CO(4)*BDP
        H=SUM+AN(4)*PDP+FBEQ+FBEP
        IF (T.EQ.ZERO) HREF=H
        DELH=H-(Y(1+2*NR)+HREF)
        RETURN
        END
C
C
C
        SUBROUTINE ENERGY(Y,TENER)
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,I,J,NR,NMAX
        PARAMETER (ZERO=0.0,TWO=2.0,NMAX=4)
        COMMON /PROP/ N,M
        COMMON /FIX/ MF(2+2*NMAX,2+2*NMAX),CO(3*NMAX+8),COM(3*NMAX,NMAX)
        COMMON /FIX/ E1(4*NMAX),G1(3*NMAX,NMAX),E2(4*NMAX),G2(3*NMAX,NMAX)
        COMMON /ENER/ MU(2+2*NMAX,2+2*NMAX)
C
C          This routine is determines the total energy of the arm.
C       It is called by ODEINT if the control variable ANGMO is set
C       to 1 by the input program.  It uses the constants G1 and G2
C       evaluated in the INITIALIZE routine and the variable MU
C       evaluated in the UPDATE routine placed in the COMMON block
C       ENER in the subroutine DERIVS.
C
        DIMENSION Y(8+4*NMAX)
        NR=2+N+M
C
C          Determine the arm Kinetic Energy.
C
        KE=ZERO
        DO 100 I=1,NR
           SUM=ZERO
           DO 200 J=1,NR
              SUM=SUM+MU(I,J)*Y(NR+J)
 200       CONTINUE
           KE=KE+SUM*Y(NR+I)
 100    CONTINUE
C
C          Determine the arm Potential Energy.
C
        PE1=ZERO
        PE2=ZERO
        DO 300 I=1,N
           SUM=ZERO
```

```
            DO 400 J=1,N
               SUM=SUM+G1(N+I,J)*Y(2+J)
 400        CONTINUE
            PE1=PE1+SUM*Y(2+I)
 300     CONTINUE
         DO 500 I=1,M
            SUM=ZERO
            DO 600 J=1,M
               SUM=SUM+G2(M+I,J)*Y(2+N+J)
 600        CONTINUE
            PE2=PE2+SUM*Y(2+N+I)
 500     CONTINUE
C
C           Return the total energy of the arm.
C
         TENER=1.0/TWO*(KE+PE1+PE2)
         RETURN
         END
C
C
C
         SUBROUTINE OUTPUT
         IMPLICIT REAL*8 (A-Z)
         INTEGER N,M,KMAX,KOUNT,NR,I,J,N2,NMAX
         PARAMETER (ZERO=0.0,NMAX=4,SCALE=1.0)
         COMMON /PROP/ N,M,BM(8),RL(4),LA(NMAX),EI(4),MM(4)
         COMMON /PATH/ KMAX,KOUNT,DXSAV,XP(500),YP(24,500),HP(500),EP(500)
         COMMON /PATH/ ME(500,15),DHP(500),TOR(2,500),SFOR(500),STIC(500)
C
C           This routine takes care of the output from the simulation.
C        It does any necessary calculations to the raw output from
C        ODEINT and stores it in MATRIXx format in MATPLOT.DAT.  It
C        will also be able to store output data in tabular form in
C        DATFILE.DAT.  Both files will be stored in the THESIS sub-
C        directory.
C
         DOUBLE PRECISION X(500,21),XM(500,16)
         DIMENSION Y1(500),Y2(500),DY1(500),DY2(500),PDQ(500)
         DIMENSION BI(NMAX),PHI(NMAX,4),SI(NMAX),PHI2(NMAX,4)
         OPEN (1,FILE='[USER.AVF.CARLOS.THESIS]MATPLOT',STATUS='NEW',
     &          FORM='FORMATTED')
         OPEN (3,FILE='[USER.AVF.CARLOS.THESIS]DATFILE',STATUS='NEW',
     &          FORM='FORMATTED')
         OPEN (4,FILE='[USER.AVF.CARLOS.THESIS]BUGFILE',STATUS='NEW',
     &          FORM='FORMATTED')
         NR=2+N+M
         N2=2*N
         L1=RL(3)
         L2=RL(4)
C
C           Determine small tip deflections of both links from the
C        output modal amplitudes and the mode shapes evaluated at the
C        tip.
C
         CALL CMODES(N,LA,L1,L1,BI,PHI,SI)
         CALL CMODES(M,LA,L2,L2,BI,PHI2,SI)
         DO 100 J=1,KOUNT
            Y1(J)=ZERO
            DY1(J)=ZERO
            PDQ(J)=ZERO
```

```
            Y2(J)=ZERO
            DY2(J)=ZERO
            DO 200 I=1,N
               Y1(J)=Y1(J)+SCALE*PHI(I,1)*YP(2+I,J)
               DY1(J)=DY1(J)+SCALE*PHI(I,1)*YP(4+N+M+I,J)
               PDQ(J)=PDQ(J)+SCALE*PHI(I,2)*YP(4+N+M+I,J)
 200        CONTINUE
            DO 300 I=1,M
               Y2(J)=Y2(J)+SCALE*PHI2(I,1)*YP(2+N+I,J)
               DY2(J)=DY2(J)+SCALE*PHI2(I,1)*YP(4+N2+M+I,J)
 300        CONTINUE
 100     CONTINUE
C
C           Copy values to MATRIXx-ready array, and store.
C
         DO 400 I=1,KOUNT
            X(I,1)=XP(I)
            X(I,2)=YP(1,I)
            X(I,3)=YP(2,I)
            X(I,4)=Y1(I)
            X(I,5)=Y2(I)
            X(I,6)=YP(3+N+M,I)
            X(I,7)=YP(4+N+M,I)
            X(I,8)=DY1(I)
            X(I,9)=DY2(I)
            X(I,10)=HP(I)
            X(I,11)=EP(I)
            X(I,12)=DHP(I)
            X(I,13)=TOR(1,I)
            X(I,14)=TOR(2,I)
            X(I,15)=YP(2*NR+1,I)
            X(I,16)=YP(2*NR+2,I)
            X(I,17)=YP(2*NR+3,I)
            X(I,18)=YP(2*NR+4,I)
            X(I,19)=PDQ(I)
            X(I,20)=SFOR(I)
            X(I,21)=STIC(I)
            XM(I,1)=X(I,1)
C           DO 800 J=1,15
C              XM(I,J+1)=ME(I,J)
C 800       CONTINUE
 400     CONTINUE
         CALL MATSAV(1,'DATAX',500,KOUNT,21,0,X,DUMMY,'(1P2E24.15)')
         CLOSE (1)
C        CALL MATSAV(4,'DATAX',500,KOUNT,16,0,XM,DUMMY,'(1P2E24.15)')
C        CLOSE (4)
C
C           Copy raw data to file.
C
         NR=2+N+M
         WRITE(3,*)N,M,KOUNT,RL(3),RL(4)
         WRITE(3,*)MM(1),MM(2),MM(3),MM(4)
         WRITE(3,*)LA(1),LA(2),LA(3),LA(4)
         DO 500 I=1,KOUNT
            WRITE(3,*)XP(I),YP(1,I),YP(2,I)
            DO 600 J=1,N
               WRITE(3,*)YP(2+J,I),YP(2+NR+J,I)
 600        CONTINUE
            DO 700 J=1,M
               WRITE(3,*)YP(2+N+J,I),YP(2+NR+N+J,I)
```

```
700       CONTINUE
500     CONTINUE
        CLOSE (3)
        RETURN
        END
C
C
C
        SUBROUTINE OUT_DIAGNOSTICS
        IMPLICIT REAL*8 (A-Z)
        INTEGER NOK,NBAD,NSAV,N,M,NR,MCOUNT,FORE,ANGMO,SINGUL
        INTEGER NCO
        PARAMETER (AMIN=60.0)
        CHARACTER*9 DAT,TIM
        REAL TINIT,TODE,TOUT
        COMMON /CONTROL/ FORE,ANGMO,SINGUL
        COMMON /PROP/ N,M
        COMMON /DIAG/ HSAV,TSAV
        COMMON /DIAG/ NOK,NBAD,H1SAV,DAT,TIM,TINIT,TODE,TOUT
        COMMON /DIA2/ NCO
        COMMON /BUG/ HMIN,DXSAV,SAVPTS,DX,TIM2,BETA,NSAV,EPS,THETA
        COMMON /BUG/ MCOUNT,Y(24)
C
C          This routine outputs diagnostics messages to the file
C       ARMDIAG.DAT in the subdirectory THESIS.
C
        OPEN (1,FILE='[USER.AVF.CARLOS.THESIS]ARMDIAG',STATUS='UNKNOWN',
     &       ACCESS='APPEND',FORM='FORMATTED')
        WRITE(1,*)'Date and time of start of simulation:',DAT,' ',TIM
        WRITE(1,*)'Number of good steps taken:', NOK
        WRITE(1,*)'Number of bad but retried steps taken:',NBAD
        WRITE(1,*)'Initial time step tried:',H1SAV
        WRITE(1,*)'Final time step used:',HSAV
        WRITE(1,*)'# of loops in RKQC for last time step:',NCO
        WRITE(1,*)'Time at end of simulation:',TSAV
        IF (TINIT .GE. AMIN) THEN
           TINIT=TINIT/AMIN
           WRITE(1,*)'Time to complete initialization (min.):',TINIT
        ELSE
           WRITE(1,*)'Time to complete initialization (sec.):',TINIT
        ENDIF
        IF (TODE .GE. AMIN) THEN
           TODE=TODE/AMIN
           WRITE(1,*)'Time to solve O.D.E. (min.):',TODE
        ELSE
           WRITE(1,*)'Time to solve O.D.E. (sec.):',TODE
        ENDIF
        IF (TOUT .GE. AMIN) THEN
           TOUT=TOUT/AMIN
           WRITE(1,*)'Time to complete output sequences (min.):',TOUT
        ELSE
           WRITE(1,*)'Time to complete output sequences (sec.):',TOUT
        ENDIF
        IF (SINGUL.NE.0) THEN
           WRITE(1,*)'****************************************'
           WRITE(1,*)'Simulation aborted.'
           WRITE(1,*)'See .LOG file for error message.'
           WRITE(1,*)'****************************************'
        ENDIF
C
```

```
C          Debugging diagnostics.
C
        WRITE(1,*)'DEBUGGING DIAGNOSTICS:'
        WRITE(1,*)'SINGUL= ',SINGUL
        WRITE(1,*)'HMIN= ',HMIN
        WRITE(1,*)'Number of HMIN exceeds: ',MCOUNT
        WRITE(1,*)'DXSAV= ',DXSAV
        WRITE(1,*)'T2= ',TIM2
        WRITE(1,*)'THETA DOT= ',THETA
        WRITE(1,*)'BETA DOT= ',BETA
        WRITE(1,*)'N= ',NSAV
        WRITE(1,*)'M= ',M
        WRITE(1,*)'EPS= ',EPS
        NR=2+N+M
        WRITE(1,*)'Y(I)      Y(NR+I)'
        DO 100 I=1,NR
           WRITE(1,*)Y(I),Y(NR+I)
 100    CONTINUE
        CALL DATE(DAT)
        CALL TIME(TIM)
        WRITE(1,*)'Date and time of end of simulation:',DAT,' ',TIM
        WRITE(1,*)' '
        CLOSE (1)
        RETURN
        END
```

```
C*************************************************************************
C*************************************************************************
C          This file contains subroutines written for the two-link
C       arm simulation from 4-29-89 on.  It is written by Carlos E.
C       Padilla as part of his M.S. thesis.
C       DOUBLE PRECISION version.
C*************************************************************************
C*************************************************************************
        SUBROUTINE STABICHK(Y,DYDT,OUT)
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,I,J,NR,N2,NMAX,FORE
        PARAMETER (ZERO=0.0,NMAX=4)
        COMMON /PROP/ N,M,BM(8)
        COMMON /CONTROL/ FORE
        COMMON /FIX/ MF(2+2*NMAX,2+2*NMAX),CO(3*NMAX+8),COM(3*NMAX,NMAX)
        COMMON /FIX/ E1(4*NMAX),G1(3*NMAX,NMAX),E2(4*NMAX),G2(3*NMAX,NMAX)
        COMMON /FIX/ BIJ1(3*NMAX,NMAX),BIJ2(3*NMAX,NMAX)
        COMMON /FIX/ ET1(NMAX,NMAX),ET2(NMAX,NMAX)
        COMMON /ANG2/ AQ,ADQ,PQ,PDQ
C
C          This subroutine checks the dynamic parameters of link
C       two of the two-link manipulator, to see if the stability
C       boundary for the consistent model predictions is exceeded.
C       If M=2, then the "exact" dynamic stiffness eigenvalues are
C       checked for non-positive definiteness.  A time trace of the
C       lowest eigenvalue is returned on output.
C       If M<>2, then a first order approx. is used using the stab.
C       boundary derived in Ch. 3 of my M.S. thesis.  The relative
C       distance
C       of u2 to the boundary (given u1dot and u3) is returned on
C       output.
C
        DIMENSION Y(8+4*NMAX),DYDT(8+4*NMAX)
        IF (FORE.EQ.1) GOTO 5000
        IF (M.EQ.0) GOTO 5000
        N2=2*N
        NR=2+N+M
        BETA=Y(2)
        DTHE=Y(1+NR)
        DDTHE=DYDT(1+NR)
        DBE=Y(2+NR)
        SB=SIN(BETA)
        CB=COS(BETA)
        SAB=PQ*CB+SB
        CAB=CB-PQ*SB
        L1=CO(3)
        B22=BM(3)
        DADQ=ZERO
        DO 100 I=1,N
           DADQ=DADQ+E1(N2+I)*DYDT(2+NR+I)
 100    CONTINUE
        U3=DTHE+DBE+PDQ
        U2=DTHE*AQ*SB+ADQ*CB+L1*DTHE*CAB
        U1DOT=-(DDTHE*AQ+2*DTHE*ADQ)*CB-DTHE*DTHE*L1*CAB
        U1DOT=U1DOT+DDTHE*L1*SAB+(DADQ-DTHE*DTHE*AQ)*SB
        IF (M.EQ.2) THEN
           COE1=U2*U3+U3*U3*B22-U1DOT
           A=G2(M+1,1)+COE1*BIJ2(1,1)+U3*U3*(BIJ2(M+1,1)-G2(1,1))
           B=G2(M+1,2)+COE1*BIJ2(1,2)+U3*U3*(BIJ2(M+1,2)-G2(1,2))
           D=G2(M+2,2)+COE1*BIJ2(2,2)+U3*U3*(BIJ2(M+2,2)-G2(2,2))
```

```
            S1=0.5*(A+D+SQRT((A-D)**2+4*B*B))
            S2=0.5*(A+D-SQRT((A-D)**2+4*B*B))
            IF (S1.LT.S2) THEN
               OUT=S1
            ELSE
               OUT=S2
            ENDIF
         ELSE
            H11=G2(M+1,1)
            MU11=BIJ2(1,1)
            ETA11=BIJ2(M+1,1)
            G11=G2(1,1)
            IF (U3.NE.ZERO) THEN
               CECK=-1/U3*(H11/MU11-U1DOT)
               CECK=CECK-U3*(B22+ETA11/MU11-G11/MU11)
               IF (CECK.NE.ZERO) THEN
                  OUT=(CECK-U2)/CECK
               ELSE
                  OUT=ZERO
               ENDIF
            ELSE
               OUT=0
            ENDIF
         ENDIF
5000     CONTINUE
         RETURN
         END
```

```
C*********************************************************************
C*********************************************************************
C          These subroutines were written by Carlos E. Padilla for
C       18.335 P.S. #6.  They are used here for his M.S. thesis  arm
C       simulation in order to invert the symmetric mass matrix
C       using Cholesky decomposition for maximun savings in execution
C       time.
C       DOUBLE PRECISION version.
C       On 9/9/88 routines were added to  allow PA=LU decomposition,
C       i.e., LU decomposition with partial pivoting, to handle mass
C       matrices that are asymmetric due to linearization.
C*********************************************************************
C*********************************************************************
C
C  Use Cholesky factorization.  Overwrites lower A with G - Cholesky
C    triangle.
C
        SUBROUTINE CHOLESKY(M,N,A)
        INTEGER NRA,NRE,NSING,NMAX
        REAL*8 ZERO
        PARAMETER (ZERO=0.0,NMAX=10)
        COMMON /CONTROL/ NRA,NRE,NSING
        INTEGER M,N,K,P,I
        REAL*8 A(NMAX,NMAX),S
C
        IF (M .NE. N) THEN
            PRINT *,'Matrix not square - Cholesky aborted'
            RETURN
        ENDIF
        DO 10 K=1,N
            S=0.0
            DO 15 P=1,K-1
                S=S+A(K,P)*A(K,P)
15          CONTINUE
            IF ((A(K,K)-S).LE.ZERO) GOTO 1000
            A(K,K)=SQRT(A(K,K)-S)
c           if (a(k,k).lt.3.4e-3) print *,'chol',k,a(k,k)
            DO 20 I=K+1,N
                S=0.0
                DO 25 P=1,K-1
                    S=S+A(I,P)*A(K,P)
25              CONTINUE
                A(I,K)=(A(I,K)-S)/A(K,K)
20          CONTINUE
10      CONTINUE
        GOTO 5000
 1000   NSING=1
 5000   CONTINUE
        RETURN
        END
C
C  Solve for y s.t. Ly=b, L n-by-n lower triangular. Overwrites b with y.
C
        SUBROUTINE FORWARD(NDIM,N,L,B)
        INTEGER NMAX
        PARAMETER (NMAX=10)
        INTEGER NDIM,N,I,J
        REAL*8 L(NMAX,NMAX),B(NMAX)
C
        DO 40 I=1,N
```

```
            DO 45 J=1,I-1
                B(I)=B(I)-L(I,J)*B(J)
45          CONTINUE
            B(I)=B(I)/L(I,I)
40      CONTINUE
        RETURN
        END
C
C  Solve for x s.t. Ux=y, U n-by-n upper triangular. Overwrites y with x.
C
        SUBROUTINE BACKWARD(NDIM,N,U,Y)
        INTEGER NMAX,NRA,NRE,NSING
        PARAMETER (NMAX=10,HUGE=1.0E35)
        COMMON /CONTROL/ NRA,NRE,NSING
        INTEGER NDIM,N,I,J
        REAL*8 U(NMAX,NMAX),Y(NMAX)
C
        DO 50 I=N,1,-1
            DO 55 J=I+1,N
                Y(I)=Y(I)-U(I,J)*Y(J)
55          CONTINUE
            Y(I)=Y(I)/U(I,I)
            IF (ABS(Y(I)).GT.HUGE) THEN
               NSING=2
               GOTO 5000
            ENDIF
50      CONTINUE
5000    CONTINUE
        RETURN
        END
C
C
C
        SUBROUTINE TRANSPOSE(M,N,A,B)
        INTEGER NMAX
        PARAMETER (NMAX=10)
        INTEGER M,N,I,J
        REAL*8 A(NMAX,NMAX),B(NMAX,NMAX)
C
        DO 30 I=1,M
            DO 35 J=1,N
                B(J,I)=A(I,J)
35          CONTINUE
30      CONTINUE
        RETURN
        END
C
C
C
        SUBROUTINE LUDEC(NDIM,N,A)
        IMPLICIT REAL*8 (A-Z)
        INTEGER NDIM,N,NMAX,I,J,K,P,IPVT
        INTEGER NRA,NRE,NSING
        PARAMETER (ZERO=0.0,ONE=1.0,NMAX=10,TINY=1.0E-12)
        COMMON /CONTROL/ NRA,NRE,NSING
        COMMON /LUBLOC/ IPVT(NMAX)
        DIMENSION A(NMAX,NMAX),W(NMAX)
C
C          This subroutine determines the LU decomposition
C       of matrix A using partial pivoting.  It is based on
```

```
C       algorithm from Golub and von Loan.  The subroutine returns
C       the permutation vector IPVT(N) (used in LUSOLVE), and
C       a unit lower triangular matrix in A(i>j) and an upper
C       triangular matrix in A(i<=j).
C
        IPVT(N)=1
        DO 100 K=1,N-1
           P=K
           DO 200 I=K+1,N
              IF (ABS(A(I,K)).GT.ABS(A(P,K))) P=I
 200       CONTINUE
           IPVT(K)=P
           IF (P.NE.K) IPVT(N)=-IPVT(N)
           IF (ABS(A(P,K)).LE.TINY) GOTO 1000
           DO 300 J=1,N
              T=A(P,J)
              A(P,J)=A(K,J)
              A(K,J)=T
              W(J)=T
 300       CONTINUE
           DO 400 I=K+1,N
              T=A(I,K)/A(K,K)
              A(I,K)=T
              DO 500 J=K+1,N
                 A(I,J)=A(I,J)-T*W(J)
 500          CONTINUE
 400       CONTINUE
 100    CONTINUE
        DET=IPVT(N)
        DO 600 I=1,N
           DET=DET*A(I,I)
 600    CONTINUE
        IF (ABS(DET).GT.TINY*TINY) GOTO 5000
 1000   NSING=1
 5000   CONTINUE
        RETURN
        END
C
C
C
        SUBROUTINE LUSOLVE(NDIM,N,A,B)
        IMPLICIT REAL*8 (A-Z)
        INTEGER NDIM,N,NMAX,I,J,K,P,IPVT
        PARAMETER (ZERO=0.0,ONE=1.0,NMAX=10)
        COMMON /LUBLOC/ IPVT(NMAX)
        DIMENSION A(NMAX,NMAX),B(NMAX),DIAG(NMAX)
C
C          This subroutine solves the matrix equation
C       PAx=Pb, where A is LU=PA from subroutine LUDEC.
C       It returns the value of x in B.
C
        DO 100 K=1,N-1
           P=IPVT(K)
           T=B(P)
           B(P)=B(K)
           B(K)=T
 100    CONTINUE
        DO 200 I=1,N
           DIAG(I)=A(I,I)
           A(I,I)=ONE
```

```
 200    CONTINUE
C
C          Ensure L=A(I>=J) is unit lower triangular, then
C       solve Ly=Pb.
C
        CALL FORWARD(NDIM,N,A,B)
        DO 300 I=1,N
           A(I,I)=DIAG(I)
 300    CONTINUE
C
C          Ensure U=A(I<=J) is upper triangular again, then
C       solve Ux=y.
C
        CALL BACKWARD(NDIM,N,A,B)
        RETURN
        END
C
C
C
        SUBROUTINE MATMATRIX(N,M,A,NAM,NFIL)
        IMPLICIT REAL*8 (A-Z)
        INTEGER NMAX
        PARAMETER (NMAX=10)
        INTEGER N,M,I,J
        DIMENSION A(NMAX,NMAX)
        DOUBLE PRECISION X(NMAX,NMAX)
        CHARACTER*4 NAM
        CHARACTER*6 NFIL
C
C          This subroutine writes the matrix A to the MATRIXx data
C       file FILE.DAT in the subdirectory THESIS.
C
        OPEN (1,FILE='[USER.AVF.CARLOS.THESIS]'//NFIL,STATUS='NEW',
     &        FORM='FORMATTED')
        DO 100 I=1,N
           DO 200 J=1,M
              X(I,J)=A(I,J)
 200       CONTINUE
 100    CONTINUE
        CALL MATSAV(1,NAM,NMAX,N,M,0,X,DUMMY,'(1P2E24.15)')
        CLOSE (1)
        RETURN
        END
C
C
C
        SUBROUTINE WRMATRIX(N,M,A,NAM,NFIL)
        IMPLICIT REAL*8 (A-Z)
        INTEGER NMAX
        PARAMETER (NMAX=10)
        INTEGER N,M,I,J
        DIMENSION A(NMAX,NMAX)
        CHARACTER*4 NAM
        CHARACTER*6 NFIL
C
C          This subroutine writes the matrix A to the ouput file
C       FILE.DAT in the subdirectory THESIS.
C
        OPEN (1,FILE='[USER.AVF.CARLOS.THESIS]'//NFIL,STATUS='NEW',
     &        FORM='FORMATTED')
```

```
      WRITE (1,1100) NAM,N,M
      DO 100 I=1,N
         WRITE (1,1000) I,(A(I,J),J=1,INT(M/2))
         WRITE (1,1000) I,(A(I,J),J=M-INT(M/2)+1,M)
100   CONTINUE
1000  FORMAT (I3,<M/2>(D18.10))
1100  FORMAT (' MATRIX  ',A4,'(',I2,' BY',I3,')')
      CLOSE (1)
      RETURN
      END
```

```
C*************************************************************************
C*************************************************************************
C          This file contains INCLUDE commands that ensures all the
C       necessary files are compiled in order to solve the ODE's for
C       the two-link flex. arm using Runge-Kutta fourth order with
C       variable step-size.  (See Numerica Recipes for reference.)
C       This was written as part of the simulation package for
C       Carlos E. Padilla's M.S. thesis.
C       DOUBLE PRECISION version.
C*************************************************************************
C*************************************************************************
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]DRK4.FOR'
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]DRKQC.FOR'
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]DODEINT.FOR'
```

```
      SUBROUTINE ODEINT(YSTART,NVAR,X1,X2,EPS,H1,HMIN,NOK,NBAD)
        IMPLICIT INTEGER (I-N)
        IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (MAXSTP=350000,NMAX=24,TWO=2.0,ZERO=0.0,TINY=1.D-30,
     &           API=3.141592654)
      COMMON /PATH/ KMAX,KOUNT,DXSAV,XP(500),YP(24,500),HP(500),EP(500)
      COMMON /PATH/ SM(500,15),DHP(500),DOR(2,500),SFOR(500),STIC(500)
      COMMON /ENER/ SME(10,10)
      COMMON /CONTROL/ NFORE,NANG,NSING
      COMMON /PROP/ N,M
      COMMON /DIAG/ HSAV,TSAV
      COMMON /BUG/ TRA,TRE,TRI,TRO,TRU,CRA,NRE,CRI,CRO
      COMMON /BUG/ MCOUNT,YW(NMAX)
      COMMON /OUS/ FORT(12,4)
      DIMENSION YSTART(NMAX),YSCAL(NMAX),Y(NMAX),DYDX(NMAX),TOR(2)
      X=X1
      H=SIGN(H1,X2-X1)
      XFRAC=0.0
      XSTE=(X2-X1)/10.0
      NCHECK=0
      MPREV=0
      NOK=0
      NBAD=0
      KOUNT=0
      NR=2+N+M
      NSING=0
      YMAX=ZERO
      DO 11 I=1,NVAR
        Y(I)=YSTART(I)
11    CONTINUE
      XSAV=X-DXSAV*TWO
      DO 16 NSTP=1,MAXSTP
        IF (X.GE.XFRAC) THEN
           PRINT *,'Current time:',x,'out of:',x2
           XFRAC=XFRAC+XSTE
        ENDIF
        IF (NSTP.GE.NCHECK) THEN
           PRINT *,'# of steps so far:',NSTP,' HMIN exceeds:',MCOUNT
           MDIF=MCOUNT-MPREV
           MPREV=MCOUNT
           NCHECK=NCHECK+1000
           IF (1000-MDIF.LE.300) THEN
              NSING=14
              PRINT *,'HMIN exceeds 70% of last 1000 steps taken.'
              print *,'yscal=',yscal
              print *,'y=',y
              GOTO 5000
           ENDIF
        ENDIF
        CALL DERIVS(X,Y,DYDX)
        IF(NSING.NE.0) GOTO 5000
c       DO 300 I=1,NVAR
c          IF ((I.GT.2).AND.(I.NE.NR+1).AND.(I.NE.NR+2)) THEN
c             IF ((ABS(Y(I))+ABS(H*DYDX(I))).GT.YMAX) THEN
c                YMAX=ABS(Y(I))+ABS(H*DYDX(I))+TINY
c             ENDIF
c          ENDIF
c 300   CONTINUE
        DO 12 I=1,NVAR
           IF ((I.GT.2).AND.(I.NE.NR+1).AND.(I.NE.NR+2)) THEN
```

```
c          IF (I.NE.(1+2*NR)) THEN
             YSCAL(I)=ABS(Y(I))+ABS(H*DYDX(I))+TINY
           ELSE
c            IF (ABS(Y(I)).GT.API) THEN
               YSCAL(I)=ABS(Y(I))+ABS(H*DYDX(I))+TINY
c            ELSE
c               YSCAL(I)=API
c            ENDIF
           ENDIF
C
12       CONTINUE
         IF(KMAX.GT.0)THEN
           IF(ABS(X-XSAV).GT.ABS(DXSAV)) THEN
             IF(KOUNT.LT.KMAX-1)THEN
               KOUNT=KOUNT+1
               XP(KOUNT)=X
               DO 13 I=1,NVAR
                 YP(I,KOUNT)=Y(I)
13             CONTINUE
               IF(NANG.EQ.1) THEN
                  SFOR(KOUNT)=FORT(2,1)
                  CALL AMOMENTUM(X,Y,AMOM,DAM)
                  CALL ENERGY(Y,ENER)
                  CALL FORCING(X,Y,TOR)
                  CALL STABICHK(Y,DYDX,STAB)
                  DOR(1,KOUNT)=TOR(1)
                  DOR(2,KOUNT)=TOR(2)
                  HP(KOUNT)=AMOM
                  DHP(KOUNT)=DAM
                  EP(KOUNT)=ENER
                  STIC(KOUNT)=STAB
C                 SM(KOUNT,1)=SME(1,1)
C                 SM(KOUNT,2)=SME(2,1)
C                 SM(KOUNT,3)=SME(3,1)
C                 SM(KOUNT,4)=SME(3,2)
C                 SM(KOUNT,5)=SME(3,3)
C                 SM(KOUNT,6)=SME(4,1)
C                 SM(KOUNT,7)=SME(4,2)
C                 SM(KOUNT,8)=SME(4,3)
C                 SM(KOUNT,9)=SME(4,4)
C                 SM(KOUNT,10)=SME(5,1)
C                 SM(KOUNT,11)=SME(5,3)
C                 SM(KOUNT,12)=SME(5,4)
C                 SM(KOUNT,13)=SME(6,1)
C                 SM(KOUNT,14)=SME(6,3)
C                 SM(KOUNT,15)=SME(6,4)
               ENDIF
               XSAV=X
             ENDIF
           ENDIF
         ENDIF
         IF((X+H-X2)*(X+H-X1).GT.ZERO) H=X2-X
         CALL RKQC(Y,DYDX,NVAR,X,H,EPS,YSCAL,HDID,HNEXT)
         IF(NSING.NE.0) THEN
            print *,'yscal',yscal
            GOTO 5000
         ENDIF
         IF(HDID.EQ.H)THEN
           NOK=NOK+1
         ELSE
```

```
          NBAD=NBAD+1
        ENDIF
        DO 100 I=1,NVAR
          YW(I)=Y(I)
 100    CONTINUE
        IF((X-X2)*(X2-X1).GE.ZERO)THEN
          DO 14 I=1,NVAR
            YSTART(I)=Y(I)
14        CONTINUE
          IF(KMAX.NE.0)THEN
            KOUNT=KOUNT+1
            XP(KOUNT)=X
            DO 15 I=1,NVAR
              YP(I,KOUNT)=Y(I)
15          CONTINUE
            IF(NANG.EQ.1) THEN
                SFOR(KOUNT)=FORT(2,1)
              CALL AMOMENTUM(X,Y,AMOM,DAM)
              CALL ENERGY(Y,ENER)
              CALL FORCING(X,Y,TOR)
              CALL STABICHK(Y,DYDX,STAB)
              DOR(1,KOUNT)=TOR(1)
              DOR(2,KOUNT)=TOR(2)
              HP(KOUNT)=AMOM
              DHP(KOUNT)=DAM
              EP(KOUNT)=ENER
              STIC(KOUNT)=STAB
C               SM(KOUNT,1)=SME(1,1)
C               SM(KOUNT,2)=SME(2,1)
C               SM(KOUNT,3)=SME(3,1)
C               SM(KOUNT,4)=SME(3,2)
C               SM(KOUNT,5)=SME(3,3)
C               SM(KOUNT,6)=SME(4,1)
C               SM(KOUNT,7)=SME(4,2)
C               SM(KOUNT,8)=SME(4,3)
C               SM(KOUNT,9)=SME(4,4)
C               SM(KOUNT,10)=SME(5,1)
C               SM(KOUNT,11)=SME(5,3)
C               SM(KOUNT,12)=SME(5,4)
C               SM(KOUNT,13)=SME(6,1)
C               SM(KOUNT,14)=SME(6,3)
C               SM(KOUNT,15)=SME(6,4)
            ENDIF
          ENDIF
          HSAV=H
          TSAV=X
          RETURN
        ENDIF
        IF (ABS(HNEXT).LT.HMIN) THEN
            HNEXT=HMIN
            MCOUNT=MCOUNT+1
        ENDIF
C        IF(ABS(HNEXT).LT.HMIN) GOTO 1000
        H=HNEXT
16    CONTINUE
      PAUSE 'Too many steps.'
      GOTO 5000
1000  PAUSE 'Stepsize smaller than minimum.'
5000  CONTINUE
      HSAV=HNEXT
```

```
      TSAV=X
        DO 200 I=1,NVAR
           YW(I)=Y(I)
200     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RKQC(Y,DYDX,N,X,HTRY,EPS,YSCAL,HDID,HNEXT)
        IMPLICIT INTEGER (I-N)
        IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NMAX=24,FCOR=.0666666667,PGROW=-0.20,PSHRNK=-0.25,
     *    ONE=1.,SAFETY=0.9,ERRCON=6.D-4)
      EXTERNAL DERIVS
      COMMON /CONTROL/ NRA,NRE,NSING
        COMMON /DIA2/ NCO
      DIMENSION Y(NMAX),DYDX(NMAX),YSCAL(NMAX)
      DIMENSION YTEMP(NMAX),YSAV(NMAX),DYSAV(NMAX)
C      PGROW=-0.20
C      PSHRNK=-0.25
      XSAV=X
      DO 11 I=1,N
        YSAV(I)=Y(I)
        DYSAV(I)=DYDX(I)
11    CONTINUE
      H=HTRY
        NCO=0
1     HH=0.5*H
      CALL RK4(YSAV,DYSAV,N,XSAV,HH,YTEMP,DERIVS)
        IF(NSING.NE.0) THEN
           NSING=NSING+3
           GOTO 5000
        ENDIF
      X=XSAV+HH
      CALL DERIVS(X,YTEMP,DYDX)
        IF(NSING.NE.0) THEN
           NSING=3
           GOTO 5000
        ENDIF
      CALL RK4(YTEMP,DYDX,N,X,HH,Y,DERIVS)
        IF(NSING.NE.0) THEN
           NSING=NSING+6
           GOTO 5000
        ENDIF
      X=XSAV+H
      IF(X.EQ.XSAV) GOTO 1000
      CALL RK4(YSAV,DYSAV,N,XSAV,H,YTEMP,DERIVS)
        IF(NSING.NE.0) THEN
           NSING=NSING+9
           GOTO 5000
        ENDIF
      ERRMAX=0.
      DO 12 I=1,N
        YTEMP(I)=Y(I)-YTEMP(I)
        ERRMAX=MAX(ERRMAX,ABS(YTEMP(I)/YSCAL(I)))
12    CONTINUE
      ERRMAX=ERRMAX/EPS
      IF(ERRMAX.GT.ONE) THEN
        H=SAFETY*H*(ERRMAX**PSHRNK)
        NCO=NCO+1
        GOTO 1
      ELSE
        HDID=H
        IF(ERRMAX.GT.ERRCON)THEN
          HNEXT=SAFETY*H*(ERRMAX**PGROW)
        ELSE
          HNEXT=4.*H
        ENDIF
```

```
      ENDIF
      DO 13 I=1,N
        Y(I)=Y(I)+YTEMP(I)*FCOR
13    CONTINUE
      GOTO 5000
1000  PAUSE 'Stepsize not signiglcant in RKQC.'
5000  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RK4(Y,DYDX,N,X,H,YOUT,DERIVS)
        IMPLICIT INTEGER (I-N)
        IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (NMAX=24)
      COMMON /CONTROL/ NRA,NRE,NSING
      DIMENSION Y(NMAX),DYDX(NMAX),YOUT(NMAX)
      DIMENSION YT(NMAX),DYT(NMAX),DYM(NMAX)
      HH=H*0.5
      H6=H/6.
      XH=X+HH
      DO 11 I=1,N
        YT(I)=Y(I)+HH*DYDX(I)
11    CONTINUE
      CALL DERIVS(XH,YT,DYT)
        IF(NSING.NE.0) THEN
           NSING=1
           GOTO 5000
        ENDIF
      DO 12 I=1,N
        YT(I)=Y(I)+HH*DYT(I)
12    CONTINUE
      CALL DERIVS(XH,YT,DYM)
        IF(NSING.NE.0) THEN
           NSING=2
           GOTO 5000
        ENDIF
      DO 13 I=1,N
        YT(I)=Y(I)+H*DYM(I)
        DYM(I)=DYT(I)+DYM(I)
13    CONTINUE
      CALL DERIVS(X+H,YT,DYT)
        IF(NSING.NE.0) THEN
           print *,'rk4 x and h:',x,h
           print *,'rk4 state vector:',yt
           NSING=3
           GOTO 5000
        ENDIF
      DO 14 I=1,N
        YOUT(I)=Y(I)+H6*(DYDX(I)+DYT(I)+2.*DYM(I))
14    CONTINUE
 5000   CONTINUE
      RETURN
      END
```

```
C*******************************************************************
C*******************************************************************
C          This file contains the INCLUDE commands that ensures that
C       the appropriate files are compiled for solving the ODE's for
C       the two-link flex. arm simulation using Runge-Kutta fourth order
C       with no variable step size.  This was written as part of the
C       simulation package for Carlos E. Padilla's M.S. thesis.
C       DOUBLE PRECISION version.
C*******************************************************************
C*******************************************************************
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]DRK4.FOR'
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]DRKDUMB.FOR'
```

```
      SUBROUTINE RKDUMB(VSTART,NVAR,X1,X2,NSTEP)
        IMPLICIT INTEGER (I-N)
        IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (ZERO=0.0,TWO=2.0,NMAX=24)
      COMMON /PROP/ N,M
      COMMON /PATH/ KMAX,KOUNT,DXSAV,XP(500),YP(24,500),HP(500),EP(500)
      COMMON /PATH/ SM(500,15),DHP(500),DOR(2,500),SFOR(500)
      COMMON /ENER/ SME(10,10)
      COMMON /CONTROL/ NFORE,NANG,NSING
      COMMON /DIAG/ HSAV,TSAV
      COMMON /BUG/ TRA,TRE,TRI,TRO,TRU,CRA,NRE,CRI,CRO
      COMMON /BUG/ MCOUNT,YW(NMAX)
      COMMON /OUS/ FORT(12,4)
      DIMENSION VSTART(NVAR),V(NMAX),DV(NMAX),TOR(2)
      KOUNT=0
      DO 10 I=1,NVAR
        V(I)=VSTART(I)
10    CONTINUE
      X=X1
      XSAV=X-DXSAV*TWO
      H=(X2-X1)/NSTEP
      XFRAC=0.0
      XSTE=(X2-X1)/10.0
      NCHECK=0
      NSING=0
      NR=2+N+M
      DO 13 K=1,NSTEP
        IF (X.GE.XFRAC) THEN
           PRINT *,'Current time:',x,'out of:',x2
           XFRAC=XFRAC+XSTE
        ENDIF
        IF (K.GE.NCHECK) THEN
           PRINT *,'# of steps so far:',K
           NCHECK=NCHECK+1000
        ENDIF
        CALL DERIVS(X,V,DV)
        IF (NSING.NE.0) GOTO 5000
        IF(KMAX.GT.0)THEN
          IF(ABS(X-XSAV).GT.ABS(DXSAV))THEN
            IF(KOUNT.LT.KMAX-1)THEN
              KOUNT=KOUNT+1
              XP(KOUNT)=X
              DO 11 I=1,NVAR
                YP(I,KOUNT)=V(I)
11            CONTINUE
              IF(NANG.EQ.1) THEN
                 SFOR(KOUNT)=FORT(2,1)
                 CALL AMOMENTUM(X,V,AMOM,DAM)
                 CALL ENERGY(V,ENER)
                 CALL FORCING(X,V,TOR)
                 DOR(1,KOUNT)=TOR(1)
                 DOR(2,KOUNT)=TOR(2)
                 HP(KOUNT)=AMOM
                 DHP(KOUNT)=DAM
                 EP(KOUNT)=ENER
                 SM(KOUNT,1)=SME(1,1)
                 SM(KOUNT,2)=SME(2,1)
                 SM(KOUNT,3)=SME(3,1)
                 SM(KOUNT,4)=SME(3,2)
                 SM(KOUNT,5)=SME(3,3)
```

```
                    SM(KOUNT,6)=SME(4,1)
                    SM(KOUNT,7)=SME(4,2)
                    SM(KOUNT,8)=SME(4,3)
                    SM(KOUNT,9)=SME(4,4)
                    SM(KOUNT,10)=SME(5,1)
                    SM(KOUNT,11)=SME(5,3)
                    SM(KOUNT,12)=SME(5,4)
                    SM(KOUNT,13)=SME(6,1)
                    SM(KOUNT,14)=SME(6,3)
                    SM(KOUNT,15)=SME(6,4)
                  ENDIF
                  XSAV=X
                ENDIF
              ENDIF
            ENDIF
            CALL RK4(V,DV,NVAR,X,H,V,DERIVS)
            IF (NSING.NE.0) GOTO 5000
            IF(X+H.EQ.X)PAUSE 'Stepsize not significant in RKDUMB.'
            TSAV=X
            IF (K.EQ.NSTEP)THEN
              KOUNT=KOUNT+1
              XP(KOUNT)=X
              DO 12 I=1,NVAR
                 YP(I,KOUNT)=V(I)
 12           CONTINUE
                IF(NANG.EQ.1) THEN
                    SFOR(KOUNT)=FORT(2,1)
                  CALL AMOMENTUM(X,V,AMOM,DAM)
                  CALL ENERGY(V,ENER)
                  CALL FORCING(X,V,TOR)
                  DOR(1,KOUNT)=TOR(1)
                  DOR(2,KOUNT)=TOR(2)
                  HP(KOUNT)=AMOM
                  DHP(KOUNT)=DAM
                  EP(KOUNT)=ENER
                   SM(KOUNT,1)=SME(1,1)
                   SM(KOUNT,2)=SME(2,1)
                   SM(KOUNT,3)=SME(3,1)
                   SM(KOUNT,4)=SME(3,2)
                   SM(KOUNT,5)=SME(3,3)
                   SM(KOUNT,6)=SME(4,1)
                   SM(KOUNT,7)=SME(4,2)
                   SM(KOUNT,8)=SME(4,3)
                   SM(KOUNT,9)=SME(4,4)
                   SM(KOUNT,10)=SME(5,1)
                   SM(KOUNT,11)=SME(5,3)
                   SM(KOUNT,12)=SME(5,4)
                   SM(KOUNT,13)=SME(6,1)
                   SM(KOUNT,14)=SME(6,3)
                   SM(KOUNT,15)=SME(6,4)
                ENDIF
            ENDIF
            X=X+H
 13       CONTINUE
 5000     CONTINUE
          TSAV=X
          DO 100 I=1,NVAR
            YW(I)=V(I)
 100      CONTINUE
          RETURN
```

END

```
C*********************************************************************
C*********************************************************************
C          This file contains the routine that determines the form of
C       the forcing functions for the two-link flex. arm simulation.
C       That is, it determines the joint torques.  This was written by
C       Carlos E. Padilla as part of his M.S. thesis work.
C       DOUBLE PRECISION version.
C*********************************************************************
C*********************************************************************
        SUBROUTINE FORCING(T,Y,TOR)
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,I,J,NMAX,NR,FORE,ANGMO,SING,RLINE,SRLINE
        PARAMETER (ZERO=0.0,NMAX=4,PI=3.141592654)
        COMMON /CONTROL/ FORE,ANGMO,SING,RLINE
        COMMON /PROP/ N,M
C
C          This subroutine determines the joint torques given the
C       time T and the state vector Y (for use in feedback control
C       laws).  The values of the torques are returned in TOR(2).
C
        COMMON /FORC/ A0,A1,A2,A3,A4,A5,B0,B1,B2,B3,B4,B5
C
C          The above common block is for use in this subroutine ex-
C       clusively to avoid recalculation of constants.
C
        DIMENSION Y(8+4*NMAX),YD(8+4*NMAX),TOR(2),DUM(2)
        DIMENSION MD(2+2*NMAX,2+2*NMAX),BD(2+2*NMAX)
C
        NR=2+N+M
C
C       X=-1.0
C
C          If x .lt. zero, the mass matrix is not updated.
C
        X=1.0
        DUM(1)=ZERO
        DUM(2)=ZERO
C
C          The following variable determines the time scaling of the
C       trajectory.
C
        AL=2.0
C
C          The following assumes angular rates and accelerations
C       are desired zero at t=0 and t=tf.
C
        IF (T.EQ.ZERO) THEN
           THEI=0.0
           THEF=PI
           BEI=0.0873
           BEF=0.0
           TF=8.0
           TF2=TF*TF
           TF3=TF*TF2
           TF4=TF*TF3
           TF5=TF*TF4
           A0=THEI
           A1=ZERO
           A2=ZERO
           A3=(20.*THEF-20.*THEI)/(2.*TF3)
```

```
      A4=(30.*THEI-30.*THEF)/(2.*TF4)
      A5=(12.*THEF-12.*THEI)/(2.*TF5)
      B0=BEI
      B1=ZERO
      B2=ZERO
      B3=(20.*BEF-20.*BEI)/(2.*TF3)
      B4=(30.*BEI-30.*BEF)/(2.*TF4)
      B5=(12.*BEF-12.*BEI)/(2.*TF5)
   ENDIF
   TA=AL*T
   TA2=TA*TA
   TA3=TA*TA2
   TA4=TA*TA3
   TA5=TA*TA4
   IF (TA.LE.TF) THEN
      YD(1)=A0+A1*TA+A2*TA2+A3*TA3+A4*TA4+A5*TA5
      YD(2)=B0+B1*TA+B2*TA2+B3*TA3+B4*TA4+B5*TA5
      YD(1+NR)=AL*(A1+2.*A2*TA+3.*A3*TA2+4.*A4*TA3+5.*A5*TA4)
      YD(2+NR)=AL*(B1+2.*B2*TA+3.*B3*TA2+4.*B4*TA3+5.*B5*TA4)
      AC1=AL*AL*(2.*A2+6.*A3*TA+12.*A4*TA2+20.*A5*TA3)
      AC2=AL*AL*(2.*B2+6.*B3*TA+12.*B4*TA2+20.*B5*TA3)
      THETA=YD(1)
      BETA=YD(2)
   ELSE
      YD(1)=THETA
      YD(2)=BETA
      YD(1+NR)=ZERO
      YD(2+NR)=ZERO
      AC1=ZERO
      AC2=ZERO
   ENDIF
   DO 100 I=1,N
      YD(2+I)=zero
      YD(2+NR+I)=zero
100  CONTINUE
   DO 200 J=1,M
      YD(2+N+J)=zero
      YD(2+NR+N+J)=zero
200  CONTINUE
C
   CALL UPDATE(X,YD,DUM,MD,BD)
   TOR(1)=MD(1,1)*AC1+MD(1,2)*AC2-BD(1)
   TOR(2)=MD(2,1)*AC1+MD(2,2)*AC2-BD(2)
C
   RETURN
   END
C
C
C
```

```
C*********************************************************************
C*********************************************************************
C          This file contains the routine that determines the form of
C       the forcing functions for the two-link flex. arm simulation.
C       That is, it determines the joint torques.  This was written by
C       Carlos E. Padilla as part of his M.S. thesis work.
C       DOUBLE PRECISION version.
C*********************************************************************
C*********************************************************************
        SUBROUTINE FORCING(T,Y,TOR)
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,I,J,NMAX,NR,FORE,ANGMO,SING,RLINE,SRLINE
        PARAMETER (ZERO=0.0,NMAX=4,PI=3.141592654)
        COMMON /CONTROL/ FORE,ANGMO,SING,RLINE
        COMMON /PROP/ N,M
C
C          This subroutine determines the joint torques given the
C       time T and the state vector Y (for use in feedback control
C       laws).  The values of the torques are returned in TOR(2).
C
        COMMON /FORC/ A0,A1,A2,A3,A4,A5,B0,B1,B2,B3,B4,B5
C
C          The above common block is for use in this subroutine ex-
C       clusively to avoid recalculation of constants.
C
        DIMENSION Y(8+4*NMAX),YD(8+4*NMAX),TOR(2),DUM(2)
        DIMENSION MD(2+2*NMAX,2+2*NMAX),BD(2+2*NMAX)
C
        NR=2+N+M
C
C       X=-1.0
C
C          If x .lt. zero, the mass matrix is not updated.
C
        X=1.0
        DUM(1)=ZERO
        DUM(2)=ZERO
C
C          The following variable determines the time scaling of the
C       trajectory.
C
        AL=2.0
C
C          The following assumes angular rates and accelerations
C       are desired zero at t=0 and t=tf.
C
        IF (T.EQ.ZERO) THEN
           THEI=-1.570796
           THEF=0.2617994
           BEI=2.967060
           BEF=0.8726646
           TF=4.0
           TF2=TF*TF
           TF3=TF*TF2
           TF4=TF*TF3
           TF5=TF*TF4
           A0=THEI
           A1=ZERO
           A2=ZERO
           A3=(20.*THEF-20.*THEI)/(2.*TF3)
```

```
            A4=(30.*THEI-30.*THEF)/(2.*TF4)
            A5=(12.*THEF-12.*THEI)/(2.*TF5)
            B0=BEI
            B1=ZERO
            B2=ZERO
            B3=(20.*BEF-20.*BEI)/(2.*TF3)
            B4=(30.*BEI-30.*BEF)/(2.*TF4)
            B5=(12.*BEF-12.*BEI)/(2.*TF5)
         ENDIF
         TA=AL*T
         TA2=TA*TA
         TA3=TA*TA2
         TA4=TA*TA3
         TA5=TA*TA4
         IF (TA.LE.TF) THEN
            YD(1)=A0+A1*TA+A2*TA2+A3*TA3+A4*TA4+A5*TA5
            YD(2)=B0+B1*TA+B2*TA2+B3*TA3+B4*TA4+B5*TA5
            YD(1+NR)=AL*(A1+2.*A2*TA+3.*A3*TA2+4.*A4*TA3+5.*A5*TA4)
            YD(2+NR)=AL*(B1+2.*B2*TA+3.*B3*TA2+4.*B4*TA3+5.*B5*TA4)
            AC1=AL*AL*(2.*A2+6.*A3*TA+12.*A4*TA2+20.*A5*TA3)
            AC2=AL*AL*(2.*B2+6.*B3*TA+12.*B4*TA2+20.*B5*TA3)
            THETA=YD(1)
            BETA=YD(2)
         ELSE
            YD(1)=THETA
            YD(2)=BETA
            YD(1+NR)=ZERO
            YD(2+NR)=ZERO
            AC1=ZERO
            AC2=ZERO
         ENDIF
         DO 100 I=1,N
            YD(2+I)=zero
            YD(2+NR+I)=zero
 100     CONTINUE
         DO 200 J=1,M
            YD(2+N+J)=zero
            YD(2+NR+N+J)=zero
 200     CONTINUE
C
         CALL UPDATE(X,YD,DUM,MD,BD)
         TOR(1)=MD(1,1)*AC1+MD(1,2)*AC2-BD(1)
         TOR(2)=MD(2,1)*AC1+MD(2,2)*AC2-BD(2)
C
         RETURN
         END
C
C
C
```

```
C*********************************************************************
C*********************************************************************
C          This file contains the routine that determines the form of
C       the forcing functions for the two-link flex. arm simulation.
C       That is, it determines the joint torques.  This was written by
C       Carlos E. Padilla as part of his M.S. thesis work.
C       DOUBLE PRECISION version.
C*********************************************************************
C*********************************************************************
        SUBROUTINE FORCING(T,Y,TOR)
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,I,J,NMAX,NR,FORE,ANGMO,SING,RLINE,SRLINE
        PARAMETER (ZERO=0.0,NMAX=4,PI=3.141592654)
        COMMON /CONTROL/ FORE,ANGMO,SING,RLINE
        COMMON /PROP/ N,M
C
C          This common block is for use in this subroutine exclusively,
C       in order to prevent repetition of constants from call to call.
C
c       COMMON /FSAVE/ BESAVE,OMSAVE,TSAVE,IB,MB,LB
C
C          This subroutine determines the joint torques given the
C       time T and the state vector Y (for use in feedback control
C       laws).  The values of the torques are returned in TOR(2).
C
        DIMENSION Y(8+4*NMAX),YD(8+4*NMAX),TOR(2),DUM(2)
        DIMENSION MD(2+2*NMAX,2+2*NMAX),BD(2+2*NMAX)
C
        NR=2+N+M
C
C       X=-1.0
C
C          If x .lt. zero, the mass matrix is not updated.
C
        X=1.0
        TORM1=1.5
        SCAL1=SQRT(1.0/(TORM1/15.962871))
        TORM2=(TORM1/(15.962871+1.9225763))*1.9225768
        SCAL2=SCAL1
        INC=1.2
C
C          Switch control of shoulder torque.
C
        TS1=1.13*SCAL1*INC
        TS2=1.44*SCAL1*INC
        TS3=1.77*SCAL1*INC
        TF1=2.91*SCAL1*INC
        IF (T.LE.TS1) THEN
           TOR(1)=-TORM1
        ELSE
           IF (T.LE.TS2) THEN
              TOR(1)=TORM1
           ELSE
              IF (T.LE.TS3) THEN
                 TOR(1)=-TORM1
              ELSE
                 IF (T.LE.TF1) THEN
                    TOR(1)=TORM1
                 ELSE
                    TOR(1)=ZERO
```

```
            ENDIF
          ENDIF
        ENDIF
      ENDIF
C
C       Switch control of elbow torque.
C
      TE1=1.42*SCAL2*INC
      TF2=2.84*SCAL2*INC
      IF (T.LE.TE1) THEN
        TOR(2)=TORM2
      ELSE
        IF (T.LE.TF2) THEN
          TOR(2)=-TORM2
        ELSE
          TOR(2)=ZERO
        ENDIF
      ENDIF
C     print *,t,tor(1),tor(2)
c     tor(1)=zero
c     tor(1)=zero
C
      RETURN
      END
C
C
C
```

```
C*********************************************************************
C*********************************************************************
C          This file contains the routine that determines the form of
C       the forcing functions for the two-link flex. arm simulation.
C       That is, it determines the joint torques.  This was written by
C       Carlos E. Padilla as part of his M.S. thesis work.
C       DOUBLE PRECISION version.
C*********************************************************************
C*********************************************************************
        SUBROUTINE FORCING(T,Y,TOR)
        IMPLICIT REAL*8 (A-Z)
        INTEGER N,M,I,J,NMAX,NR,FORE,ANGMO,SING,RLINE,SRLINE
        PARAMETER (ZERO=0.0,NMAX=4,PI=3.141592654)
        COMMON /CONTROL/ FORE,ANGMO,SING,RLINE
        COMMON /PROP/ N,M
C
C          This common block is for use in this subroutine exclusively,
C       in order to prevent repetition of constants from call to call.
C
c       COMMON /FSAVE/ BESAVE,OMSAVE,TSAVE,IB,MB,LB
C
C          This subroutine determines the joint torques given the
C       time T and the state vector Y (for use in feedback control
C       laws).  The values of the torques are returned in TOR(2).
C
        DIMENSION Y(8+4*NMAX),YD(8+4*NMAX),TOR(2),DUM(2)
        DIMENSION MD(2+2*NMAX,2+2*NMAX),BD(2+2*NMAX)
C
C
C          This forcing scheme is used to do smooth spin-up maneuver of elbow
C       link wiht "Kane's" formula.  Two statements in DARMSEC.FOR are
C       used to freeze the shoulder angle.
C
        NR=2+N+M
C
C       X=-1.0
C
C          If x .lt. zero, the mass matrix is not updated.
C
        DUM(1)=ZERO
        DUM(2)=ZERO
        X=1.0
        BEI=0.0
        TF=15.0
        OMF=6.0
        OM=2.0*PI/TF
        FAC=OMF/TF
        IF (T.LE.TF) THEN
           YD(2)=FAC*(T*T/2.+1./(OM*OM)*(COS(OM*T)-1.))+BEI
           YD(2+NR)=FAC*(T-1./OM*SIN(OM*T))
           AC2=FAC*(1.-COS(OM*T))
           BETA=YD(2)
           TB=T
        ELSE
           YD(2)=OMF*T+BETA-OMF*TB
           YD(2+NR)=OMF
           AC2=ZERO
        ENDIF
        YD(1)=ZERO
        YD(1+NR)=ZERO
```

```
      AC1=ZERO
      DO 100 I=1,N
         YD(2+I)=ZERO
         YD(2+NR+I)=ZERO
100   CONTINUE
      DO 200 I=1,M
         YD(2+N+I)=ZERO
         YD(2+NR+N+I)=ZERO
200   CONTINUE
      CALL UPDATE(X,YD,DUM,MD,BD)
      TOR(1)=MD(1,1)*AC1+MD(1,2)*AC2-BD(1)
      TOR(2)=MD(2,1)*AC1+MD(2,2)*AC2-BD(2)
C
      RETURN
      END
C
C
C
```

```
C*************************************************************************
C*************************************************************************
C          This FORTRAN program acts as a post-processor for the
C       tabulated data from the two-link flex. arm simulation.  It was
C       written by Carlos E. Padilla as part of his M.S. thesis work in
C       order to prepare data to be transferred to an IBM PC to be
C       used in a graphics demonstration of the arm motion.
C          This program takes raw data from the file DATFILE.DAT and
C       translates it into BASIC compatible data in the file ARMGAT.DAT
C       which is transferred to the PC.
C*************************************************************************
C*************************************************************************
        INCLUDE '[USER.AVF.CARLOS.FORTRAN]CMODES.FOR'
C
C          FAC establishes the scaling of the flex. deflections
C       for max. visibility in graphics animation.
C
        PROGRAM POST_PROCESS
        PARAMETER (ZERO=0.0,NMAX=4,FAC=1.0,SCAP=1.0)
        IMPLICIT REAL (A-Z)
        INTEGER N,M,KOUNT,I,J,K,NR
        DIMENSION T(500),YP(20,500),XR(20),YR(20),X2R(20),Y2R(20)
        DIMENSION LA(NMAX),BI(NMAX),PHI(NMAX,4),SI(NMAX)
        OPEN (1,FILE='[USER.AVF.CARLOS.THESIS]DATFILE',STATUS='OLD',
     &       FORM='FORMATTED')
        OPEN (3,FILE='[USER.AVF.CARLOS.GRAF]ARMGAT',STATUS='NEW',
     &       FORM='FORMATTED')
C
C          Read in data from DATFILE.DAT.
C
        READ(1,*)N,M,KOUNT,L1,L2
        READ(1,*)M1,M2A,M2B,M3
        READ(1,*)LA(1),LA(2),LA(3),LA(4)
        NR=2+N+M
        DO 100 I=1,KOUNT
           READ(1,*)T(I),YP(1,I),YP(2,I)
           DO 200 J=1,N
              READ(1,*)YP(2+J,I),YP(2+NR+J,I)
 200       CONTINUE
           DO 300 J=1,M
              READ(1,*)YP(2+N+J,I),YP(2+NR+N+J,I)
 300       CONTINUE
 100    CONTINUE
        CLOSE (1)
C
C          Manipulate data.
C
        XC=0.
        YC=0.
        LENG=1.5
        M2=M2A+M2B
        CH=M1
        IF (M2.GT.CH) CH=M2
        IF (M3.GT.CH) CH=M3
        IF (CH.EQ.ZERO) THEN
           R1=.025
           R2=.025
           R3=.025
        ELSE
c          Changed from .25 to .13 on Jan. 11,1989.
```

```
            MR=.13/CH
            R1=MR*M1
            IF (R1.EQ.ZERO) R1=.025
            R2=MR*M2
            R3=MR*M3
            IF (R3.LE.0.025) R3=.025
         ENDIF
         CH=L1
         IF (L2.GT.CH) CH=L2
         LR=1.0/CH
         WRITE(3,*) FAC
         WRITE(3,*) KOUNT,R1,R2,R3
         DO 400 I=1,KOUNT
            STH=SIN(YP(1,I))
            CTH=COS(YP(1,I))
            AL=ZERO
            TY1=ZERO
            CALL CMODES(N,LA,L1,L1,BI,PHI,SI)
            DO 800 K=1,N
               AL=AL+SCAP*PHI(K,2)*YP(2+K,I)
               TY1=TY1+SCAP*PHI(K,1)*YP(2+K,I)
800         CONTINUE
            TY1=FAC*TY1
            AL=FAC*AL
            XR(20)=LR*(L1*CTH-TY1*STH)
            YR(20)=LR*(L1*STH+TY1*CTH)
            BETA=YP(1,I)+AL+YP(2,I)
            SBE=SIN(BETA)
            CBE=COS(BETA)
            AL2=ZERO
            TY2=ZERO
            CALL CMODES(M,LA,L2,L2,BI,PHI,SI)
            DO 900 K=1,M
               AL2=AL2+SCAP*PHI(K,2)*YP(2+N+K,I)
               TY2=TY2+SCAP*PHI(K,1)*YP(2+N+K,I)
900         CONTINUE
            TY2=FAC*TY2
            AL2=FAC*AL2
            X2R(20)=LR*(L2*CBE-TY2*SBE)
            Y2R(20)=LR*(L2*SBE+TY2*CBE)
            X=XC+R1*CTH
            Y=YC+R1*STH
            DLX=LENG*CTH
            DLY=LENG*STH
            DX1=1./20.*L1
            TX1=DX1
            DX2=1./20.*L2
            TX2=DX2
            DO 500 J=1,19
               CALL CMODES(N,LA,L1,TX1,BI,PHI,SI)
               TY1=ZERO
               DO 600 K=1,N
                  TY1=TY1+SCAP*PHI(K,1)*YP(2+K,I)
600            CONTINUE
               TY1=FAC*TY1
               XR(J)=LR*(TX1*CTH-TY1*STH)
               YR(J)=LR*(TX1*STH+TY1*CTH)
               TX1=TX1+DX1
               CALL CMODES(M,LA,L2,TX2,BI,PHI,SI)
               TY2=ZERO
```

```
            DO 700 K=1,M
               TY2=TY2+SCAP*PHI(K,1)*YP(2+N+K,I)
700         CONTINUE
            TY2=FAC*TY2
            X2R(J)=LR*(TX2*CBE-TY2*SBE)
            Y2R(J)=LR*(TX2*SBE+TY2*CBE)
            TX2=TX2+DX2
500      CONTINUE
         XC2=X+XR(20)+R2*COS(YP(1,I)+AL)
         YC2=Y+YR(20)+R2*SIN(YP(1,I)+AL)
         X2=XC2+R2*CBE
         Y2=YC2+R2*SBE
         DLX2=LENG*CBE
         DLY2=LENG*SBE
         XC3=X2+X2R(20)+R3*COS(BETA+AL2)
         YC3=Y2+Y2R(20)+R3*SIN(BETA+AL2)
         WRITE(3,*)T(I)
         WRITE(3,*)X,Y,DLX,DLY
         WRITE(3,*)XC2,YC2,X2,Y2
         WRITE(3,*)DLX2,DLY2,XC3,YC3
         DO 1000 J=1,20
            WRITE(3,*)XR(J),YR(J),X2R(J),Y2R(J)
1000     CONTINUE
400   CONTINUE
      CLOSE (3)
      END
```

```
      PROGRAM ANIMATE_ARM
C
C        This animation program is written by Carlos E. Padilla as part
C      of his M.S. thesis to enact an animation of a two-link flexible
C      arm.  It is based on the example set by the CUBE.FOR program
C      in the VWSDEMO directory in the microvax.
C
      IMPLICIT INTEGER (A-Z)
      EXTERNAL ENABLE_WINDOW_RESIZE
      INTEGER UIS$CREATE_DISPLAY,UIS$CREATE_WINDOW
      REAL VP_WIDTH,VP_HEIGHT,WC_X1,WC_Y1,WC_X2,WC_Y2
      REAL RETX,RETY,xi
      REAL XC1,YC1,R1,R2,R3,S,XS,YS,XD,YD
      REAL X(500),Y(500),XR(20,500),YR(20,500),XC2(500),YC2(500)
      REAL X2(500),Y2(500),X2R(20,500),Y2R(20,500),DLX2(500)
      REAL XC3(500),YC3(500),T(500),DLX(500),DLY(500),DLY2(500)
      PARAMETER (TINY=0.02)
      CHARACTER*12 TIME(500),FAC
      COMMON WC_X1,WC_Y1,WC_X2,WC_Y2,VP_WIDTH,VP_HEIGHT
      COMMON NEW_ABS_X,NEW_ABS_Y,WD_ID,VD_ID
C
C
C
      VP_WIDTH=50.
      VP_HEIGHT=50.
C
      WC_X1=-3.
      WC_Y1=-3.
      WC_X2=3.
      WC_Y2=3.
C
      XC1=0.0
      YC1=0.0
C
C        Input data from ARMGAT.DAT.
C
      OPEN (1,FILE='[USER.AVF.CARLOS.GRAF]ARMGAT',STATUS='OLD',
     &      FORM='FORMATTED')
      READ(1,3000) FAC
      READ(1,*) KOUNT,R1,R2,R3
      DO 600 I=1,KOUNT
         READ(1,3000) TIME(I)
         READ(1,*) X(I),Y(I),DLX(I),DLY(I)
         READ(1,*) XC2(I),YC2(I),X2(I),Y2(I)
         READ(1,*) DLX2(I),DLY2(I),XC3(I),YC3(I)
         DO 700 J=1,20
            READ(1,*) XR(J,I),YR(J,I),X2R(J,I),Y2R(J,I)
 700     CONTINUE
 600  CONTINUE
      CLOSE(1)
 3000 FORMAT (A12)
C
C        Create display.
C
 6000 VD_ID=UIS$CREATE_DISPLAY(WC_X1,WC_Y1,WC_X2,WC_Y2,
     &                          VP_WIDTH,VP_HEIGHT)
      CALL UIS$DISABLE_DISPLAY_LIST(VD_ID)
      WD_ID=UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION',
     &                        'TWO-LINK FLEXIBLE ARM')
C
```

```
C
C
      CALL UIS$SET_RESIZE_AST(VD_ID,WD_ID,ENABLE_WINDOW_RESIZE,,
     &         NEW_ABS_X,NEW_ABS_Y,VP_WIDTH,VP_HEIGHT)
      CALL UIS$SET_WRITING_MODE(VD_ID,0,1,9)
      CALL UIS$SET_WRITING_MODE(VD_ID,0,2,3)
C
C        Write permanent text to window.
C
      CALL UIS$TEXT(VD_ID,2,'Time (sec):',-2.8,-2.8)
      CALL UIS$GET_ALIGNED_POSITION(VD_ID,1,RETX,RETY)
      CALL UIS$TEXT(VD_ID,2,'Flex. scale:'//FAC,1.6,-2.8)
C
C        Draw crosshairs.
C
         CALL UIS$PLOT(VD_ID,2,XC1-0.5,0.,XC1+0.5,0.)
         CALL UIS$PLOT(VD_ID,2,0.,YC1-0.5,0.,YC1+0.5)
C
C        Enter arm loop.
C
      PRINT *,'Leave trace? (0-No,1-Yes)'
      READ *,TRACE
      IF (TRACE.EQ.0) THEN
         PRINT *,'Plot guides? (0-No,1-Yes)'
         READ *,GUID
         STEP=75
      ELSE
         GUID=0
         STEP=1
      ENDIF
      CALL UIS$CIRCLE(VD_ID,2,XC1,YC1,R1)
      JM=1
      DO 2000 J=1,KOUNT
         CALL UIS$TEXT(VD_ID,2,TIME(J),RETX,RETY)
         XS=X(J)
         YS=Y(J)
         XD=X2(J)
         YD=Y2(J)
c        CALL UIS$CIRCLE(VD_ID,2,XC1,YC1,R1)
         IF (GUID.EQ.1) THEN
            call uis$plot(vd_id,2,xs,ys,xs+dlx(j),ys+dly(j))
         ENDIF
         CALL UIS$PLOT(VD_ID,2,XS,YS,XS+XR(1,J),YS+YR(1,J))
         DO 200 I=2,20
            CALL UIS$PLOT(VD_ID,2,XS+XR(I-1,J),YS+YR(I-1,J),
     &                    XS+XR(I,J),YS+YR(I,J))
            CALL UIS$PLOT(VD_ID,2,XS+XR(I-1,J),YS+YR(I-1,J))
 200     CONTINUE
         CALL UIS$CIRCLE(VD_ID,2,XC2(J),YC2(J),R2)
         IF (GUID.EQ.1) THEN
            call uis$plot(vd_id,2,xd,yd,xd+dlx2(j),yd+dly2(j))
         ENDIF
         CALL UIS$PLOT(VD_ID,2,XD,YD,XD+X2R(1,J),YD+Y2R(1,J))
         DO 300 I=2,20
            CALL UIS$PLOT(VD_ID,2,XD+X2R(I-1,J),YD+Y2R(I-1,J),
     &                    XD+X2R(I,J),YD+Y2R(I,J))
            CALL UIS$PLOT(VD_ID,2,XD+X2R(I-1,J),YD+Y2R(I-1,J))
 300     CONTINUE
         CALL UIS$CIRCLE(VD_ID,2,XC3(J),YC3(J),R3)
         IF (TRACE.EQ.1) GOTO 5000
```

```
C
C          The following three lines draw the tip trajectory.
C
           if (j.ne.1) then
              call uis$plot(vd_id,2,xc3(j-1),yc3(j-1),xc3(j),yc3(j))
           endif
           IF ((J.NE.1).AND.(J.NE.JM).AND.(J.NE.KOUNT)) THEN
c          CALL UIS$CIRCLE(VD_ID,2,XC1,YC1,R1)
           IF (GUID.EQ.1) THEN
              call uis$plot(vd_id,2,xs,ys,xs+dlx(j),ys+dly(j))
           ENDIF
           CALL UIS$PLOT(VD_ID,2,XS,YS,XS+XR(1,J),YS+YR(I,J))
           DO 400 I=2,20
              CALL UIS$PLOT(VD_ID,2,XS+XR(I-1,J),YS+YR(I-1,J),
     &                      XS+XR(I,J),YS+YR(I,J))
              CALL UIS$PLOT(VD_ID,2,XS+XR(I-1,J),YS+YR(I-1,J))
 400       CONTINUE
           CALL UIS$CIRCLE(VD_ID,2,XC2(J),YC2(J),R2)
           IF (GUID.EQ.1) THEN
              call uis$plot(vd_id,2,xd,yd,xd+dlx2(j),yd+dly2(j))
           ENDIF
           CALL UIS$PLOT(VD_ID,2,XD,YD,X2(J)+X2R(1,J),
     &                   YD+Y2R(1,J))
           DO 500 I=2,20
              CALL UIS$PLOT(VD_ID,2,XD+X2R(I-1,J),YD+Y2R(I-1,J),
     &                      XD+X2R(I,J),YD+Y2R(I,J))
              CALL UIS$PLOT(VD_ID,2,XD+X2R(I-1,J),YD+Y2R(I-1,J))
 500       CONTINUE
           CALL UIS$CIRCLE(VD_ID,2,XC3(J),YC3(J),R3)
           ELSE
              JM=JM+STEP
           ENDIF

 5000      CONTINUE
           CALL UIS$TEXT(VD_ID,2,TIME(J),RETX,RETY)
 2000   CONTINUE
        CALL UIS$TEXT(VD_ID,2,TIME(KOUNT),RETX,RETY)
c       CALL UIS$CIRCLE(VD_ID,2,XC1,YC1,R1)
        PRINT *,'Repeat? (0-No,1-Yes)'
        READ *,A
        IF (A.EQ.1) THEN
           CALL UIS$DELETE_DISPLAY(VD_ID)
           GOTO 6000
        ENDIF
        CALL UIS$SOUND_BELL('SYS$WORKSTATION')
        END
C
C
C
        SUBROUTINE enable_window_resize
C
C       This routine will allow the user to change the size of the window.
C       The user may not specify a window width or height of less than
C       3/10 of a centimeter.  This routine is called from the resize
C       ast routine.
C
        COMMON  wc_x1,wc_y1,wc_x2,wc_y2,vp_width,vp_height
        COMMON  new_abs_x,new_abs_y,wd_id,vd_id

        IF (vp_width  .LT. .30) vp_width  = .30
```

```
IF (vp_height .LT. .30) vp_height = .30
CALL UIS$RESIZE_WINDOW(vd_id,wd_id,new_obs_x,new_obs_y,
1       vp_width,vp_height,
2       wc_x1,wc_y1,wc_x2,wc_y2)

RETURN
END
```