DYNAMICS AND CONTROL OF MANIPULATING ROBOTS

by

PHILIP VARGHESE

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

APPROVED:

Hugh F. VanLandingham, Chairman

R.L.Moose

J.W.Roach

February, 1987

Blacksburg, Virginia

DYNAMICS AND CONTROL OF MANIPULATING ROBOTS

by

PHILIP VARGHESE

Hugh.F.VanLandingham,Chairman

Electrical Engineering

(ABSTRACT)

CSL 10/24/88

Dynamics and Control of manipulating robots currently is a major field of research in robotics. Various methods of efficiently formulating the non-linear dynamics have been proposed with an emphasis on reducing the computations neccesary to solve the dynamics.Several control concepts and algorithms have been suggested.

This thesis develops a dynamic simulator written in PL/1 for a general n-degree of freedom manipulator which can be used to evaluate the performance of different control algorithms. The dynamics is based on the Newton-Euler formulation. Three different control algorithms are simulated and their performance is compared with respect to tracking error,control input requirement and computational efficiency

The three algorithms studied are

- Control based on nominal and perturbed inputs with the perturbed input computed from a linearised model of the manipulator about its nominal trajectory.

- Control based on decoupling of nonlinear systems.

- Adaptive control based on the estimation of the perturbed model.

A computational analysis of the three algorithms is also provided. The manipulator arm used for the simulation has three links.

ACKNOWLEDGEMENTS

The author wishes to express his gratitude to Dr. R.L.Moose and Dr. J.W.Roach for serving as members of his commitee and for their helpful suggestions.

The author is deeply indebted to Dr.H.F.VanLandingham who as chairman of the commitee and major adviser has been a constant source of guidance and encouragement.

Finally,the author would like to express his deep gratitude to his parents for their love and support.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1.0 INTRODUCTION

Dynamics and control of manipulating robots has been a topic of significant research interest in recent years. Various algorithms for efficient computation of robot dynamics as well as control algorithms have been proposed. Due to the complexity of the dynamics of robot arms many of these control algorithms are suitable only for a restricted class of manipulators. From a control system point of view manipulators can be classed under time-varying,highly coupled non-linear systems. Therefore the analysis of the performance and suitability of a particular algorithm for a given system is usually done by means of a computer simulation. A simulation can give a great deal of information about a specific control algorithm which would otherwise be very difficult to obtain by means of analysis alone.

The basic problem with robot dynamics is its complexity even for relatively simple arms with two or three links. For more than three links it is impractical to represent the dynamics in a closed form. However,a point by point dynamic model can be constructed in a relatively simple and recursive manner which can be used to study the dynamics of a manipulator of arbitrary structure and complexity. This formulation of the dynamics recursively,can be done in several ways. The Newton-Euler method is a commonly used one and is

relatively simple to put into a recursive form for easy computer evaluation. This is the method that is used to develop the dynamics simulator in this thesis.

The function of a robot dynamics simulator is esentially two-fold. Given a desired motion to be performed by the manipulator what are the inputs to be given to the joint actuator motors and conversely to determine what the motion of the manipulator will be for a given time history of control inputs. Robot arms are systems of variable structure. During a given motion task,the various joint actuators experience a widely varying range of loads both due to the variable structure and due to the high degree of coupling between the various degrees of freedom. Therefore it is important to know whether a given actuator with some specifications can actually realise the desired motion for the manipulator. Thus a dynamics simulation not only helps in selecting a suitable actuator motor but also in determining the limits of the motion a given actuator can perform- for example the maximum load that can be moved or the maximum speed with which you can move an object. Any control algorithm prescribes a certain control law by which control inputs may be determined to acheive a particular motion of the arm. By performing a motion simulation of the control algorithm we can easily determine how well it is working. Robustness,tracking capability,noise sensitivity are a few

of the important features of the control algorithm that can be evaluated with respect to the given manipulator.

The dynamics simulator developed as part of this thesis is a computer program written in PL/1 that computes the forward and inverse dynamics of a manipulator of arbitrary construction. It computes the dynamic model on a point by point basis and a linearised model of the manipulator around a given point. The linearised model is used in some control algorithms. The point by point dynamic model is represented in the form of a system of second-order differential-equations whose co-effecients are computed by the program. With the rigid body assumption for the links, this formulation is an exact one and will give very accurate values for the accelerations and forces that are computed. To study a given manipulator the input to the program consists of a minimal amount of input data consisting of information regarding the geometric and mechanical details of the manipulator and the parameters of the actuator motors. The inverse kinematics of the manipulator is not derived within the program itself and the specific inverse kinematics relationships for each manipulator must be hand derived and supplied to the program in the form of formulae for direct evaluation. This limitation which is discussed in more detail in the kinematics section is not a serious one,however. Inverse kinematics are more easily and effeciently derived by hand rather than computed with a program. With the exception of

this relatively minor limitation this program is general enough to be applied to manipulator arms of arbitrary complexity.

In order to simulate a given control algorithm it is neccesary to add a subroutine,with the control laws, in the simulation program and to inter-face it with the main routine. The program listing given in Appendix D contains the control algorithm based on computed linearised models. Chapter 2 develops the general ideas of the kinematics and dynamics formulation used in this thesis and the details of kinematics and dynamics are described in Appendices A and B respectively. Chapter 3 discusses the simulation aspects and the complete state model development of the manipulator,with a brief description of trajectory planning and the actuator models. Chapters 4,5 and 6 describe the three diffrent algorithms simulated in the thesis. The simulation results of the three control algorithms is given in Chapter 7. The equations of motion for the manipulator that is simulated are derived in Appendix C.

The manipulator model used for the simulation and testing of the program is a three link arm with two rotational and one translational joint. The actuator motors are chosen to be dc permanent magnet motors with armature voltage control. The first of the three control algorithms considered is the one based on linearised models(Chapter 4). In this algorithm the basic principle consists of providing an input to the

actuator that consists of two parts- a nominal input computed directly by a forward dynamics evaluation and a perturbed input based on a linearised model of the manipulator about its current position that serves to minimise the error between the nominal or desired trajectory and the actual tarajectory. The former is commonly refered to as the computed torque technique and is not sufficient to ensure that the manipulator will track the desired trajectory due to errors in modelling,disturbance inputs etc.

The second algorithm is one that is based on nonlinear decoupling theory. The basic idea here is to provide appropriate non-linear state feed-back that allows you to control the individual actuator sub-systems as though they were uncoupled. Although this algorithm works very well as seen in the simulation results it is not easy to implement for manipulators of greater complexity than three links. This is because the non-linear state feed-back laws are derived directly from the equations of motion of the manipulator expressed in a closed form. The third algorithm is an adaptive one and is very similar to the one based on linearised models. Both algorithms make use of the concept of nominal and perturbed inputs but the difference lies in the way the perturbed input is computed. Instead of using a computed linearised model the model parameters(time-varying) are estimated using the well known recursive least squares(RLS) algorithm and the perturbed input is computed based on this

estimated model parameters. Although the RLS algorithm works best with constant parameter systems it can be used in this case with the assumption that the system parameters are 'slowly time-varying'. If we can obtain reasonably good parameter tracking which will in turn result in good path-tracking then this algorithm has a distinct advantage over the one based on computed model parameters. The first is that the amount of computation is reduced, secondlly the algorithm is made less sensitive to modelling errors and can be expected to perform better in the presence of random noise inputs.

2.0 CHAPTER 2-KINEMATICS AND DYNAMICS

2.1 KINEMATICS

A manipulating robot is characterised by a set of rigid links where adjacent links are connected by joints which usually have one degree of freedom. Such a combination of mechanical members is referred to as a kinematic-chain. It is an open chain if one end is rigidly fixed to the ground, the other end being free to move . A closed kinematic-chain is one in which both ends are fixed. Depending upon the application,a robot-arm may assume both these configurations.

Joints are of two types usually - rotational and translational or linear. Six degrees of freedom are required to arbitrarily position and orient the free-end or the gripper-end of the manipulator,within its work-space. However it is possible to have more or less than six degrees of freedom and its not uncommon to come across industrial robots with five degrees of freedom.The links and joints of the manipulator are usually numbered starting from the base end.

Since each link can move with one degree of freedom with respect to the previous link,the joint variable(an angle or a distance depending upon the type of joint) is defined as the position of one link with respect to its previous link measured from some reference position. The vector whose

components are the numbers corresponding to the joint variables defines the configuration of the manipulator at any instant. In order to keep track of the positions of all points on the manipulator it is neccesary to define a co-ordinate system that is fixed with respect to all links viz. a base co-ordinate system, as well as a separate co-ordinate system on each link whose position and orientation is fixed with respect to that particular link(link co-ordinate system). The origin of the link co-ordinate system may be located on any point on the link,but it is convenient to loacte it at either the COG(center of gravity) or at the center of the joint coresponding to that link. Due to certain simplifications that result in the formulation of the manipulator dynamics,the COG is chosen as the origin of the link co-ordinate system here.

As the manipulator is moved,the link co-ordinate system changes both in position as well as orientation with respect to the base co-odinate system. The orientation of each link co-ordinate system may be described by an orthogonal transformation which is a 3×3 matrix whose columns represent the unit-vectors along the axes of the link co-ordinate systems represented in the base co-ordinate system. The elements of the transformation matrix are functions of the joint variables. The origin of each link co-ordinate system may be located by adding up the vectors joining the COG to the center of the joint connecting each link to the previous link

and repeating the process till the origin of the base co-ordinates is reached. A vector may be represented either in the base co-ordinate system or in the link co-ordinate system and conversion from one to the other is performed by multiplying the vector with the appropriate transformation matrix.

When the manipulator is performing a given task the position and orientation of the gripper or end-effector which is fixed to the last link is to be maintained along a desired path. This path may be described by specifying the x,y,z co-ordinates of the end-effector for all points along the path. For a manipulator with several degrees of freedom there may be several sets of values for the joint position vector that allow the end effector to occupy the same position and orientation. A choice can be made on the basis that we would like the variation of the joint variables to be as smooth as possible. The transformation from the x,y,z co-ordinates(in the base co-ordinate system) to the corresponding joint variables is referred to as the backward kinematics solution. The reverse situation where the joint variables are given and the x,y,z co-ordinates of the end-effector is to be determined is referred to as the forward kinematic solution. The latter problem is a much simpler one since the solution is always unique. The backward kinematic solution is determined from the geometry of the given manipulator i.e. the lengths of the various links and the type of joints connecting them etc. Since it reduces to a problem of solving

a set of non-linear equations it is not a trivial task to obtain the solutions in a closed form by means of a computer program. However,given the geometric details of a given manipulator it is possible to write down the backward kinematic solution in the form of formulae or equations which can be used in the program. This is a convenient way to handle this problem,since the solution of the backward kinematics can be obtained quite easily by hand for any given manipulator.

Given the geometric details of a manipulator,the individual links are asembled one by one as described in Appendix A [1]. The process of assembly consists of determining the orthogonal transformation matrices or the A-matrices of each link,the vectors joining the link COG to the adjacent joints and the vectors representing the joint axes. These are first computed with all the joint variables set to zero. Then for a given set of values for the joint variables,the new vectors are determined by rotating or translating them about the appropriate joint axes. Once the vectors joining the COG of the links to the adjacent joints are known we can determine the x,y,z co-ordinates of the end-effector by simply adding the vectors(represented in the base co-ordinate system). This completes the process of the forward kinematics solution.

2.2 DYNAMICS FORMULATION

Each joint of the manipulator is provided with an actuator that controls the positon of that joint. The dynamic model of an n-link manipulator can be represented by n non-linear,time-varying and highly coupled second-order differential equations. In vector notation they be written as

$$H(\underline{q})\underline{q}'' + \underline{h}(\underline{q},\underline{q}') = \underline{\tau} \qquad (3.1)$$

where

- $\underline{q},\underline{q}',\underline{q}''$ = n×1 vectors of joint position,velocity and acceleration.
- $H(\underline{q})$= n×n inertia matrix whose elements are functions of joint positions only.
- $\underline{h}$ = n×1 vector of velocity-dependent and gravity forces.
- $\underline{\tau}$ = n×1 vector of joint torques or joint forces.

To express the differential-equations of motion explicitly with closed form expressions for the elements of H and $\underline{h}$ is an extremely tedious task even for the simplest manipulators with only two or three links. However the dynamics formu-

lation given in appendix B allows the co-effecients of the differential-equations to be evaluated in a point by point form [1]. This dynamic model is the starting point for all the control algorithms considered here. The method of computing this point by point model is given in detail in appendix B and will only be described very briefly here.

There is only one degree of freedom between adjacent links. Suppose the velocity and acceleration(both linear and angular) of the COG of link i-1 is known as well as the velocity and acceleration of joint i. We can then compute the velocity and acceleration of the COG of link i. Thus starting from link 0 or the base we can compute the velocity and acceleration of the COG of all the links in terms of the joint positions,velocities and accelerations. Now to compute the joint torques we start from the last link or link n and proceed as follows. Consider the free-body diagram of link n shown in Figure 1 on page 15. Subscript i refers to the inertial force and moment given by

$$\underline{F}_i = m\underline{a} \quad , \quad \underline{M}_i = I\underline{\omega} \tag{3.2}$$

where

- m = mass of link n.
- $\underline{a}$ = linear acceleration of link n COG

- $\underline{\omega}$ = angular accelleration of link n COG

- I = inertia tensor of link n

Subscript r refers to the reaction forces and moment at joint n and subscript e refers to the known external force and moment acting on link n. The reaction force and moment can be determined by considering the the dynamic equilibrium of the free-body. If joint n is a rotational joint then we can take the component of the reaction moment along the joint axis and this represents the torque to be applied at joint n. If it is a linear joint,the reaction force is projected along the joint axis and represents the joint force. In the expression for the joint torque or joint force the velocity dependent term and the inertia term are written separately.

Next we consider the free-body diagram of link n-1 and compute joint n-1 torque just as above. Following this procedure the torque or force for each joint is determined. From the expressions for the joint torques we can obtain expressions for the elements of the inertia matrix H and the velocity dependent force vector $\underline{h}$. The expressions are easily put in a recursive form for effecient computations of the various quantities of interest.

As in the case of kinematics we can look at the dynamics problem in two ways. Given a desired joint acceleration we would like to know what the applied forces or torques at the

actuators should be. This may be done by a direct evaluation of the left hand side of Equation 2.1 on page 11 and is reffered to as the computation of the forward dynamics. On the other hand, given a set of joint torques or forces that are applied to the joint actuators,the resulting acceleration in the joints is to be computed. This is called as the backward or inverse dynamics computation and may be done by expresssing Equation 2.1 on page 11 in a modified form (Equation 2.3).

$$\underline{q}'' = H^{-1}(\underline{q})[\underline{\tau} - \underline{h}(\underline{q},\underline{q}')] \qquad (3.3)$$

The inverse dynamics involves the computation of the inverse of the inertia matrix H.

The dynamics formulation that is used here is commonly referred to as the Newton-Euler method. Another way to solve the dynamics is to use Lagrange's equations. From the computational point of view the former method is more efficient[3]. Although one of the criteria for evaluating the different control algorithms is their computational effeciency,in this case it does not make any difference to the analysis since the same dynamics formulation is used for the simulation of all the algorithms.

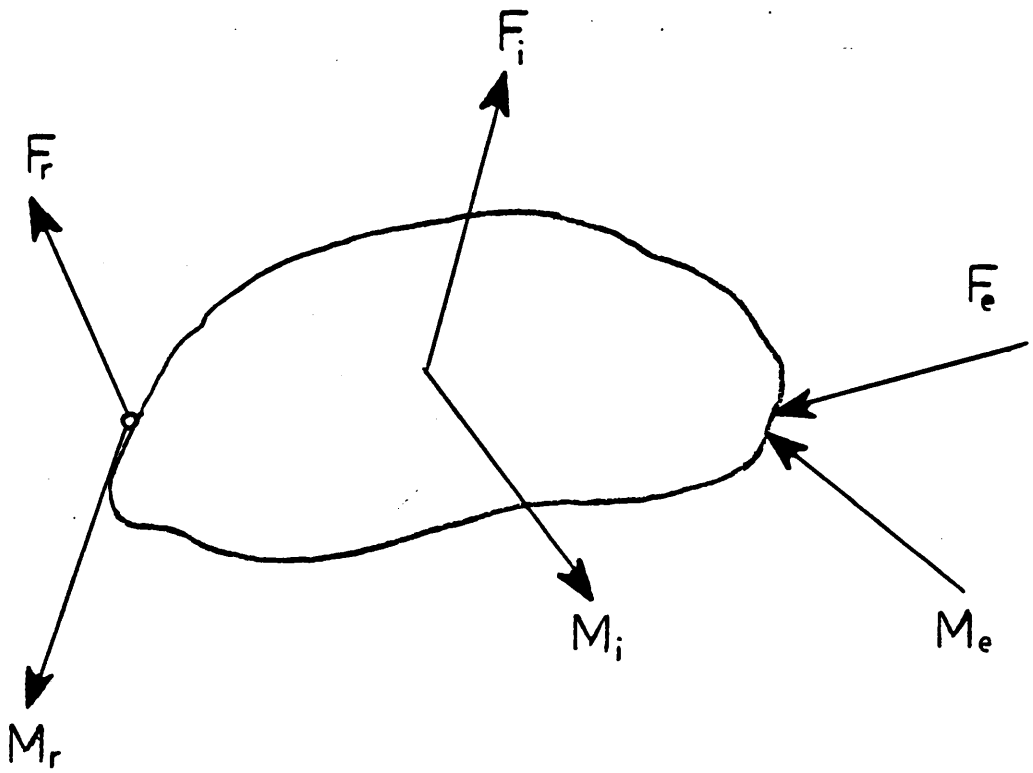


Figure 1. Link n: free-body diagram

3.0 CHAPTER 3-SIMULATION OF MANIPULATOR DYNAMICS

A typical task for a manipulator may be specified as the motion from a given starting point A to some other point B in a given amount of time T. The actual path of the manipulator end-effector during this task may or may not be specified. It may only be only be neccesary to maintain a certain orientation for the object that is moved,for instance. In such a situation the actual trajectory that is chosen may come indirectly from other considerations such as minimum-time control or minimum-energy control or the presence of other objects(obstacles) within the work-space of the manipulator. The problem of determining the actual path of the manipulator end-effector,given a starting point,final point and total time is called trajectory-planning or path-planning. Trajectory-planning,particularly in the presence of obstacles (collision-free path-planning) is a subfield by itself in robotics and will not be dealt with in great detail here.

3.1 CHOICE OF A MANIPULATOR MODEL

Since,one of the main objectives of this thesis is an evaluation of different control algorithms,it is important to chose a manipulator model to which all the control algo-

rithms may be applied for the sake of comparison of relative performance. This is not an easy task since some algorithms are suitable for application to only the simplest types of robot arms. The non-linear control algorithm with arbitrary pole-placement[4] is a case in point. To apply this algorithm it is necessary to derive the differential-equations of motion of the manipulator in an analytically explicit form. Due to the complexity of manipulator dynamics,this turns out to be an unwieldy and tedious task of algebraic manipulation even for a three or four link manipulator. In general,rotational-joints especially ones closer to the base add greatly to the complexity of the dynamics whereas translational-joints reduce the complexity. In view of this difficulty a three-link manipulator was chosen for this study. The Stanford manipulator is a 6-joint arm and its geometric and mechanical details are easily accesible[2]. The model used here(Figure 2 on page 23) may be called as the modified Stanford manipulator as it is the same manipulator with the last three links discarded. It has two rotational-joints and a linear-joint which have the same geometric dimensions and structure as the Stanford-manipulator.

3.2 TASK SPECIFICATION AND TRAJECTORY PLANNING

The task for the manipulator is assumed to be the motion between two points specified in base co-ordinates in a time

interval of 1 second ,which is chosen arbitrarily. Due to the difficulties mentioned earlier in obtaining the differental-equations of motion for the manipulator,the starting and final points are chosen in such a way that one of the joints(the first one) does not need to be moved. Thus for deriving the equations of motion the manipulator behaves essentially like a 2-link arm with considerable reduction in the complexity of the dynamics as shown in Appendix C. To further simplify matters it is assumed that the end-efffector is to move uniformly (in cartesian co-ordinates) along a straight line path between the two points in the given total time for the task. A uniform motion in cartesian co-ordinates does not necessarily translate into uniform motion for the joint variables due to the non-linear relationship between joint variables and x,y,z co-ordinates of the end-effector (as seen in the inverse-kinematic relationships in Appendix A). The desired path having thus been specified,the path is divided into a suitable number of steps which are chosen to be 100 steps all equal in length. Thus the desired position for the end-effector is obtained step by step by means of a linear interpolation between the starting and final points. At each point the inverse-kinematic relationship is applied and the path is thus obtained in joint co-ordinates. The desired joint velocities and accelerations are obtained by differentiating the expressions for forward kinematics as shown in Appendix A. At each step the desired

values and actual values(obtained by simulation) of the joint positions,velocities and accelerations are fed back to the control algorithm which computes the actuator torques and forces to be applied to minimise the errors.

3.3 SIMULATION

The general flow chart of the simulation of manipulator dynamics is shown in Figure 3 on page 24. The trajectory-planner specifies the desired positions,velocities and accelerations of the joints at each of the 100 points along the trajectory. As explained in Appendix B we can compute the inertia matrix H and the velocity dependent term $\underline{h}$ given the position and velocity vector.

It is assumed that dc permanent magnet motors are used for all the three joint actuators. The models for the dc motors of each joint can be represented by second-order differential equations which in state-variable form can be written as shown below.

$$\underline{x}_i' = A_i\underline{x}_i + \underline{b}_i u_i + \underline{f}_i p_i \qquad (4.4)$$

where

$$A_i \in R^{2\times 2} \qquad \underline{b}_i, \underline{f}_i \in R^2$$

are constant matrices and vectors easily obtainable from the motor constants and

$\underline{x}_i = [\ q_i \quad q_i'\ ]^T$, u_i=armature voltage, p_i=output torque

Now defining

$\underline{x} = [q_1\ q_1'\ q_2\ q_2' \ldots\ q_n\ q_n']^T$, $\underline{u} = [u_1\ u_2 \ldots u_n]^T$

$B=\text{diag}(\underline{b}_i)$, $F=\text{diag}(\underline{f}_i)$, $\underline{p}=[p_1\ p_2 .. p_n]^T$, $A=\text{diag}(A_i)$

the combined state model can be written as

$$\underline{x}' = A\underline{x} + B\underline{u} + F\underline{p} \tag{4.5}$$

The mechanical model of the manipulator is given by

$$H\underline{q}'' + \underline{h} = \underline{p} \tag{4.6}$$

Defining a matrix T such that $q'' = Tx'$ the two equations above can be combined to give the complete state model of the system as

$$\underline{x}' = A_c\underline{x} + B_c\underline{u} + \underline{p}_c \tag{4.7}$$

where

$A_c = SA$, $B_c = SB$, $\underline{p}_c = S\underline{h}$ and $S=[I-FHT]^{-1}$

Given a desired torque vector $\underline{p}$ the required actuator input voltages $\underline{u}$ can be calculated from

$$\underline{u} = [HTB]^{-1}[(I-HTF)\underline{p} - \underline{h} - HTA\underline{x}\ ] \qquad (4.8)$$

Using the sampling time of 0.01 sec the continuous time model of Equation 3.7 on page 20 can be converted to discrete time. Although the matrices A,B,p are time-varying it is assumed that they remain constant over each time-step.

$$\underline{x}(k+1)= A_d\underline{x}(k) + B_d\underline{u}(k) + \underline{p}_d(k) \qquad (4.9)$$

For a given input and state the next state is computed by a direct evaluation of the right hand side of Equation 3.9. This will represent the actual state of the manipulator and is compared with the desired state trajectory to compute a correction torque to minimise the tracking errors. It must be noted that the matrices A,B and p must be computed at each sampling time. The complete algorithm is shown schematically in Figure 3 on page 24. The desired joint trajectories are shown in Figure 4 on page 25, Figure 5 on page 26 and Figure 6 on page 27.

In this simulation it must be noted that the dynamics of the individual controllers for actuator motor are not taken into account for the sake of simplicity. This would result in some delays in the feed-back paths to the motor being ignored. Actuators commonly used in industrial robot arms are permanent magnet dc motors,stepper motors and hydraulic or pneumatic drives. These may be modeled by second-order or third-order linear differential-equations with constant coeffecients and can be inputed to the program in the form of the state-variable matrices.

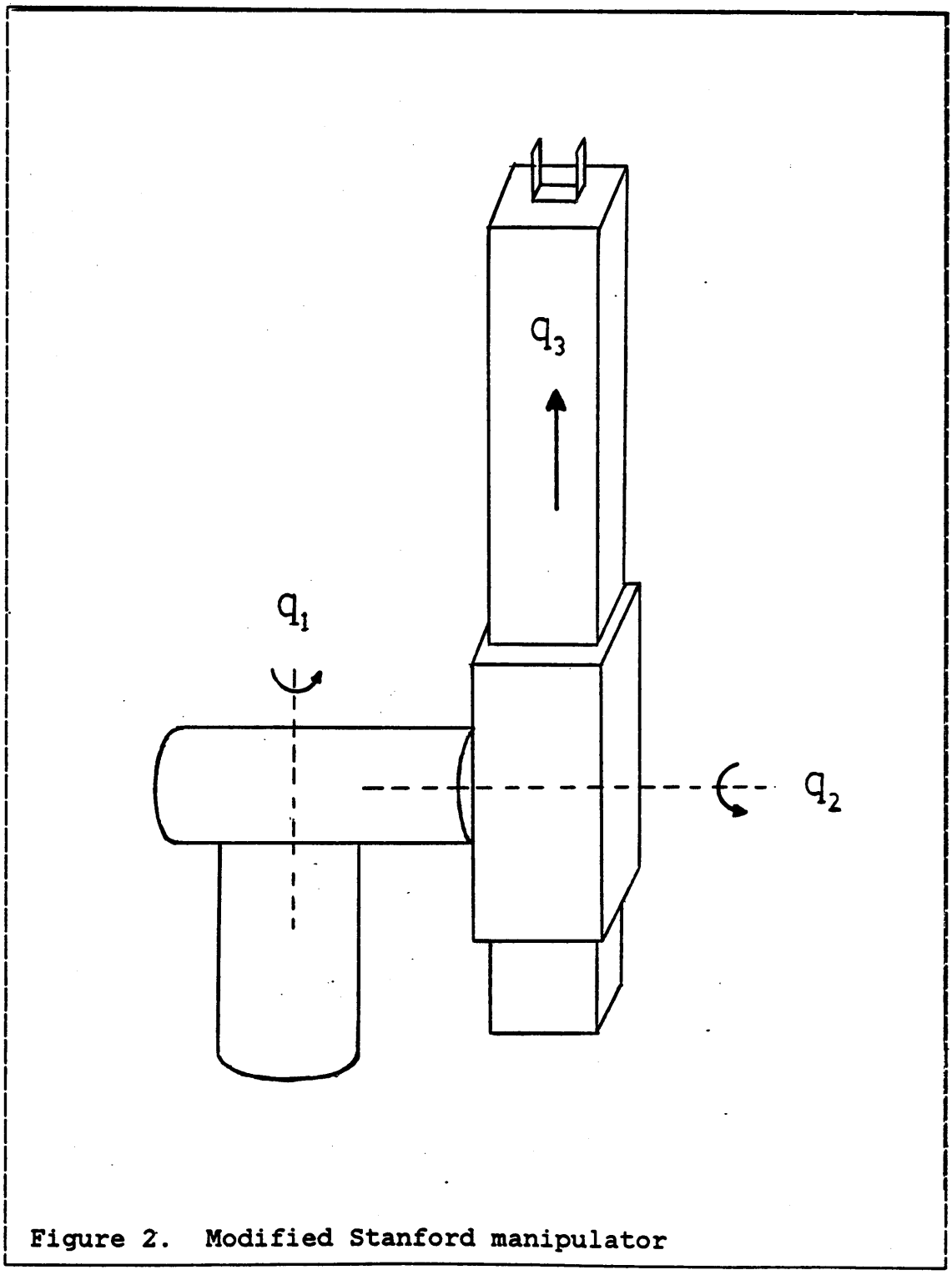


Figure 2. Modified Stanford manipulator

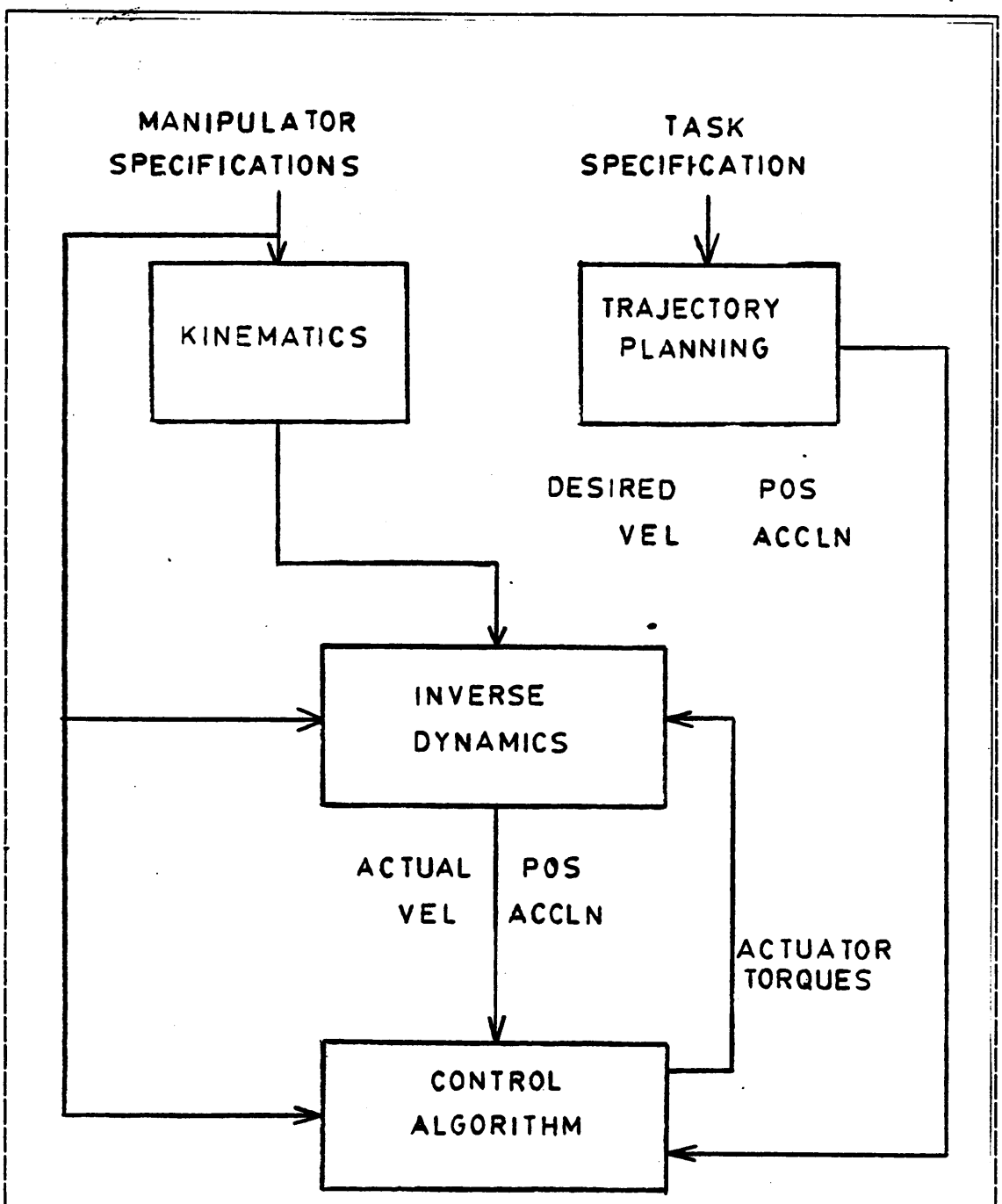


Figure 3. Manipulator dynamics: Simulation flow-chart.

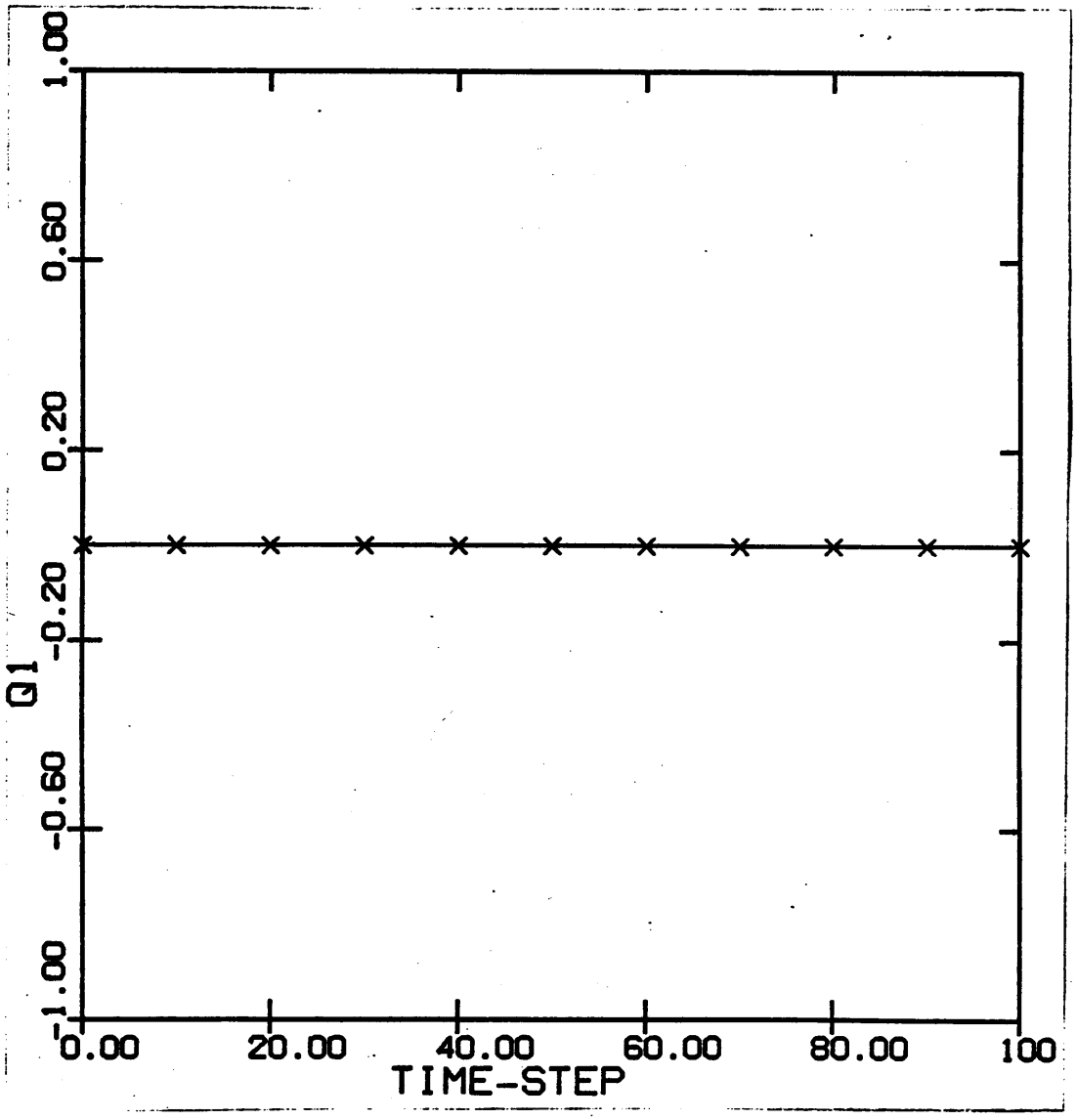


Figure 4. Joint Trajectory: Joint One

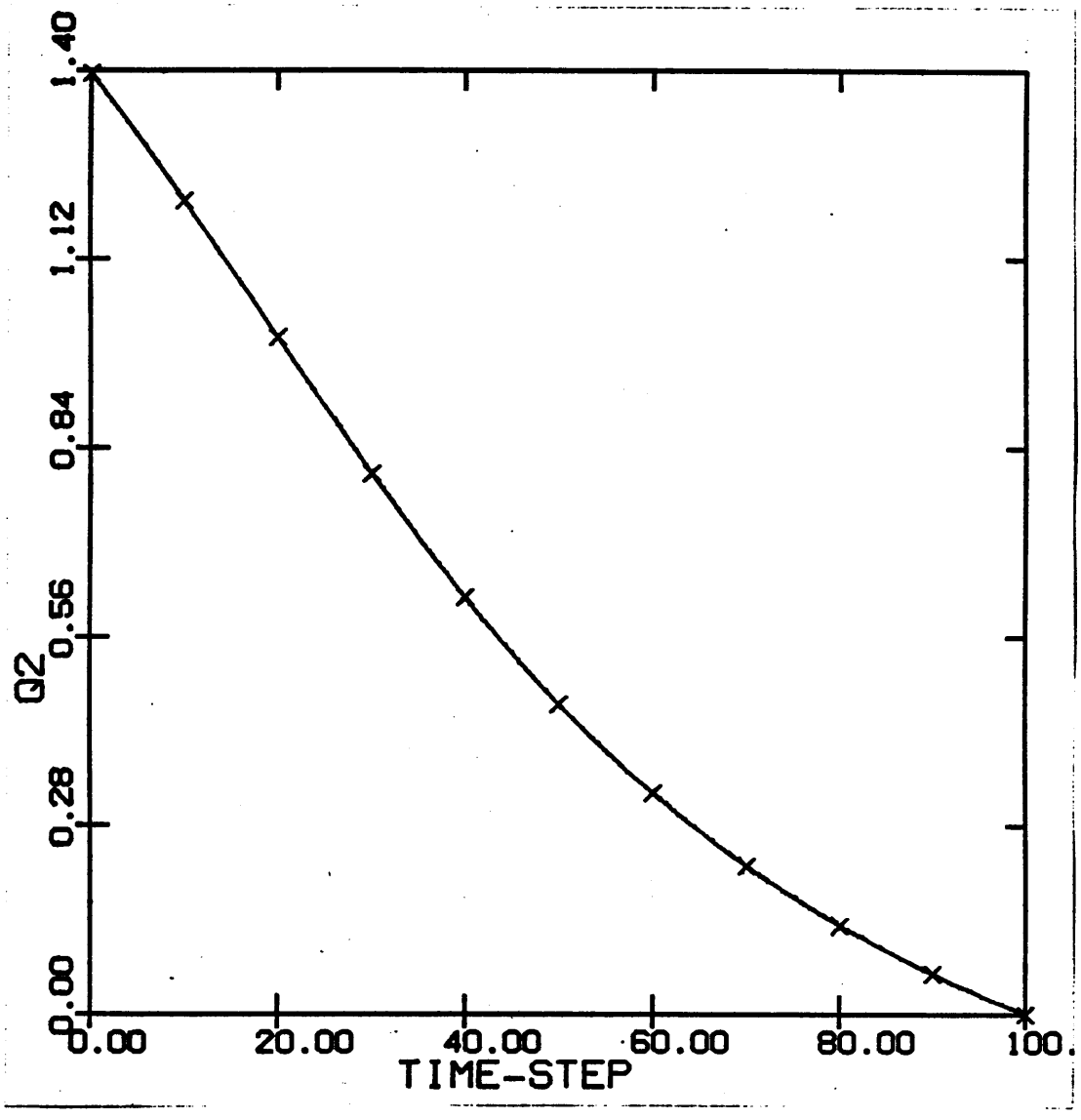


Figure 5. Joint Trajectory: Joint Two position

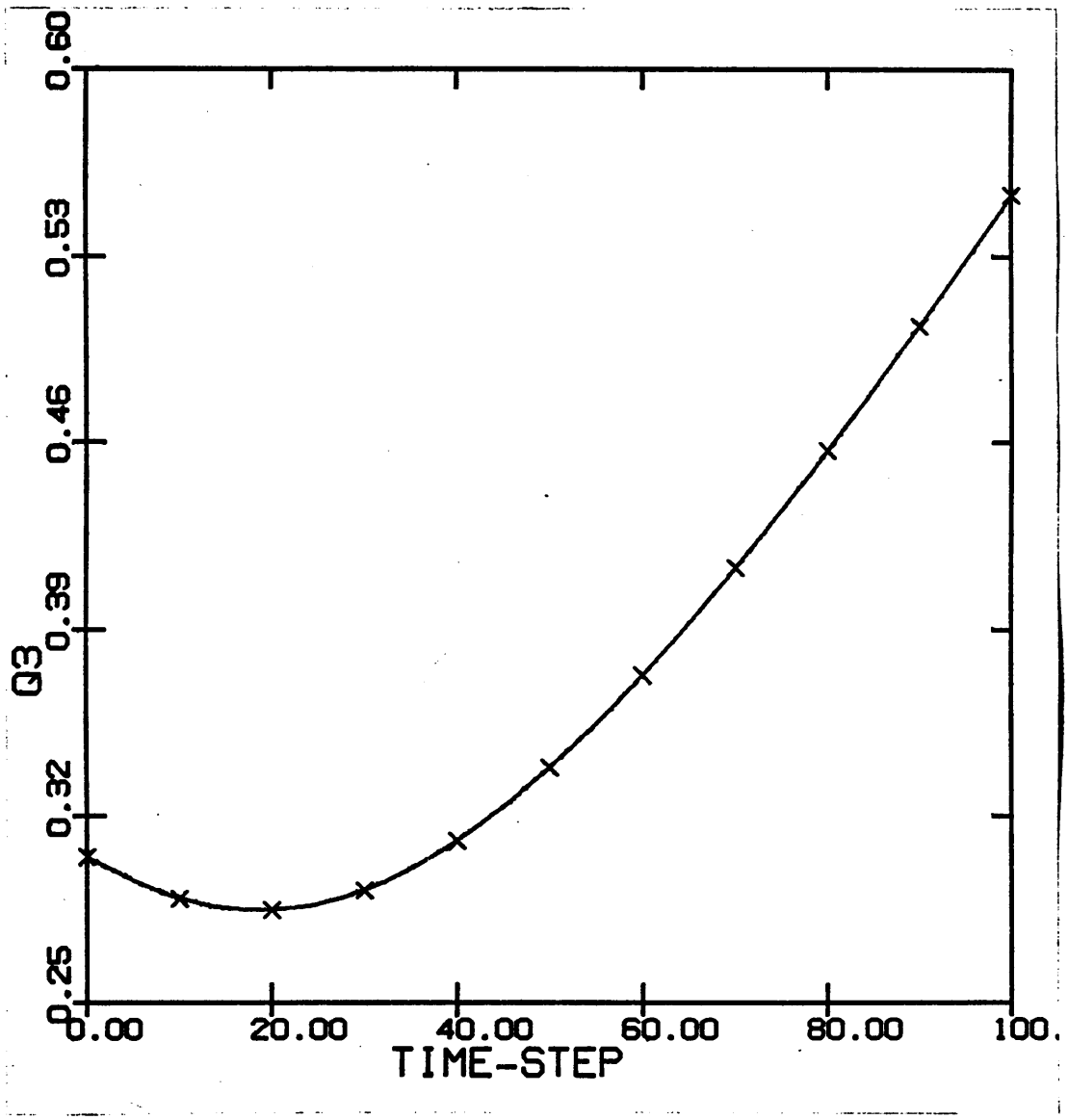


Figure 6. Joint Trajectory: Joint Three position

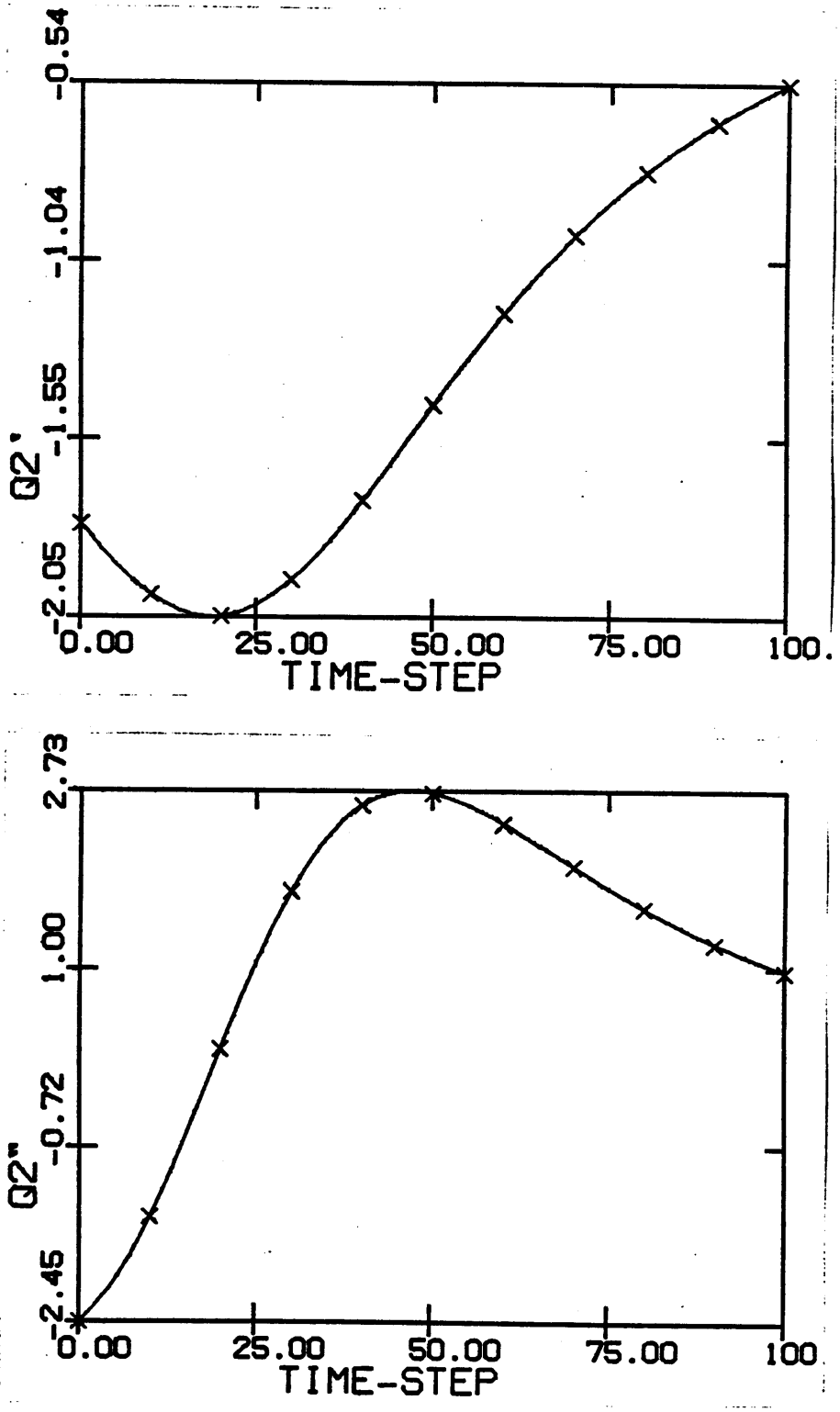


Figure 7. Joint Two: Desired velocity and acceleration

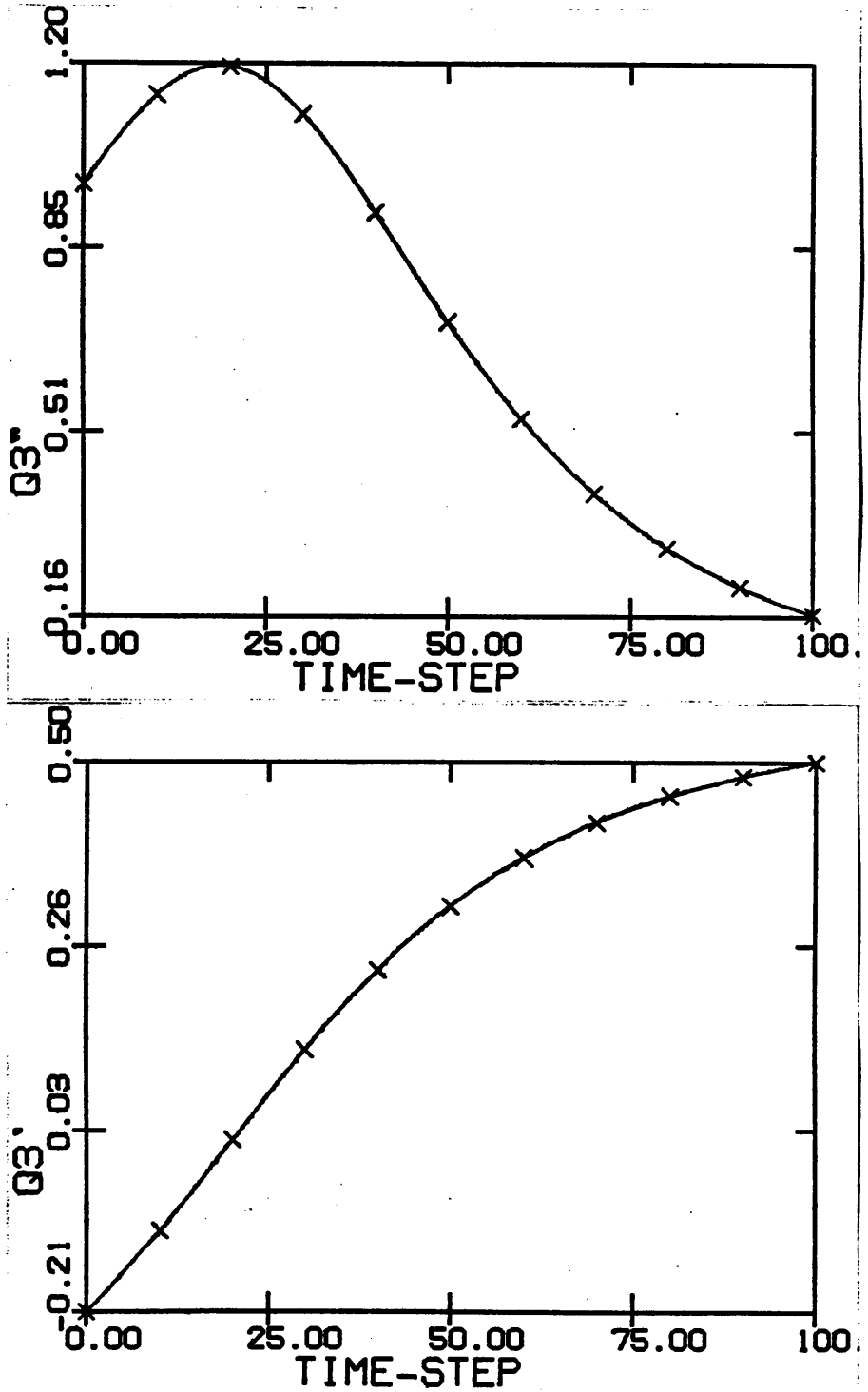


Figure 8. Joint Three: Desired velocity and acceleration

4.0 CHAPTER 4-CONTROL BASED ON LINEARISED MODELS

The two-fold control concept of nominal and perturbed control inputs was proposed by Vukobratovic[5]. In this method the control input to the joint actuators at any instant is composed of two components namely a nominal input computed by evaluation of the forward dynamics for the desired joint trajectory and a perturbed input to minimise the error between the desired positions and the measured or actual joint positions. This perturbed input is computed for a linearised model of the manipulator about its measured position and velocity on a point by point basis. The nominal input at each point is computed by evaluating the right hand side of Equation 4.10.

$$\underline{\tau}_n = H(\underline{q}_d)\underline{q}_d'' + \underline{h}(\underline{q}_d, \underline{q}_d') \qquad (5.10)$$

where

- $\underline{\tau}_n \in R^n$ is the nominal input vector
- $\underline{q}_d, \underline{q}_d', \underline{q}_d''$ = desired joint position, velocity and acceleration
- $H, \underline{h}$ are computed as shown in Appendix B

The linearised model derived in Appendix B can be written as

$$\underline{x}' = A\underline{x} + B\delta\underline{\tau} \qquad (5.11)$$

where

$$\underline{x} \in R^{2n} = \begin{bmatrix} \delta\underline{q} \\ \delta\underline{q}' \end{bmatrix} = \begin{bmatrix} \underline{q}_d - \underline{q}_a \\ \underline{q}_d' - \underline{q}_a' \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & I_n \\ -H^{-1}h_2 & -H^{-1}h_1 \end{bmatrix} \in R^{2n\times 2n}$$

$$B = \begin{bmatrix} 0 \\ H^{-1} \end{bmatrix} \in R^{2n\times n}$$

$$h_1 = \partial\underline{h}/\partial\underline{q}' \in R^{n\times n}$$

$$h_2 \in R^{n\times n} = \partial H/\partial\underline{q}.\underline{q}'' + \partial\underline{h}/\partial\underline{q}$$

and $\delta\underline{\tau} \in R^n$ is the perturbed input. With $\underline{u} \equiv \delta\underline{\tau}$ we can write Equation 4.11 on page 31 as

$$\underline{x}' = A\underline{x} + B\underline{u}$$

The discrete-time equivalent of this model is given by

$$\underline{x}(k+1) = \Phi(k)\underline{x}(k) + \Gamma(k)\underline{u}(k) \qquad (5.12)$$

where

$$\Phi(k) = e^{A(k)T}, \quad \Gamma(k) = \int_0^T e^{A(k)[t-\tau]}B(k)d\tau$$

Here T=.01 sec is the sampling interval.

Based on the one-step minimisation criterion

$$J(k) = 1/2\ [\ \underline{x}(k)^T Q\ \underline{x}(k) + \underline{u}(k)^T R\ \underline{u}(k)\ ] \qquad (5.13)$$

the optimum input is computed as

$$\underline{u}^*(k) = -\ [\ R + \Gamma(k)^T Q\ \Gamma(k)\ ]^{-1}\Gamma(k)^T Q\ \Phi(k)\underline{x}(k) \qquad (5.14)$$

This represents the perturbed input vector. The combined input vector $\underline{\tau}$ is given by

$$\underline{\tau}(k) = \underline{\tau}_n(k) + \underline{u}^*(k) \qquad (5.15)$$

$\Phi(k)$ and $\Gamma(k)$ are computed by a truncated series algorithm.

5.0 CHAPTER 5-NON-LINEAR CONTROL

The non-linear feedback control design applied to a manipulator was proposed by Freund[4]. To apply this design to robot arms it is necessary to obtain the differential-equations of motion in an explicit form. The feedback control laws obtained from this design decouple the various degrees of freedom at the same time enabling arbitrary pole-placement for the decoupled subsystems. The theory for the non-linear decoupling is briefly reviewed here.

5.1 NON-LINEAR DECOUPLING

Let the time-varying,non-linear system be represented by

$$\underline{x}'(t) = A(\underline{x},t) + B(\underline{x},t)\underline{u}(t) \qquad (6.16)$$

$$\underline{y}(t) = C(\underline{x},t) + D(\underline{x},t)\underline{u}(t) \qquad (6.17)$$

Let the input vector $\underline{u}(t)$ be of the form

$$\underline{u}(t) = F(\underline{x},t) + G(\underline{x},t)\underline{v}(t) \qquad (6.18)$$

where $\underline{v}(t)$ is the reference input vector. Then

$$\underline{x}' = (A+BF) + BG\underline{v} \tag{6.19}$$

$$\underline{y} = (C+DF) + DG\underline{v} \tag{6.20}$$

Here A,C are vectors of dimensions the same as the state-vector $\underline{x}$ and the input vector $\underline{u}$ respectively. Define the non-linear operator

$$N_A^k[.] \quad \text{for } k=1,2,... \quad \text{as}$$

$$N_A^k[C_i] = \partial/\partial t\,(N_A^{k-1}[C_i]) + \partial/\partial \underline{x}\,(N_A^{k-1}[C_i])A \tag{6.21}$$

and

$$N_A^0[C_i] = C_i \tag{6.22}$$

where C_i represents the i^{th} entry of C. Also define differential-orders d_i as

$$d_i = 0 \quad , \text{ if } D_i \neq \underline{0} \tag{6.23}$$

$$d_i = j \quad , \text{ if } D_i = 0 \tag{6.24}$$

where D_i is the i^{th} row of D and j is given by

$$\partial/\partial \underline{x} \; (N_A^{j-1}[C_i]) \neq \underline{0} \tag{6.25}$$

and

$$\partial/\partial \underline{x} \; (N_A^{p}[C_i]) = \underline{0} \quad \text{,for } 0 \leq p < j-1 \tag{6.26}$$

Then for $d_i \neq 0$, the following relationships hold.

$$N_{A+BF}^{k}[C_i] = N_A^{k}[C_i] \quad \text{,for} \quad k=0,1,\ldots d_i-1 \tag{6.27}$$

$$N_{A+BF}^{d_i}[C_i] = N_A^{d_i}[C_i] + [\partial/\partial \underline{x} \; (N^{d_i-1}[C_i])] \; BF \tag{6.28}$$

$$y_i = C_i \tag{6.29}$$

For $d_i=0$

$$N_A^{d_i}[C_i] = C_i \tag{6.30}$$

Now defining

$$y_i^* = d^{d_i}/dt^{d_i} \; (y_i) \tag{6.31}$$

it follows from Equation 5.23 on page 35 to Equation 5.31 that y^* can be written as

$$\underline{y}^* = C^* + D^*(F + G\underline{v}) \qquad (6.32)$$

where

$$C_i^* = N_A^{d_i}[C_i] \qquad (6.33)$$

and

$$D_i^* = D_i \quad , \quad \text{if } d_i = 0$$

$$D_i^* = \partial/\partial\underline{x}\,(N_A^{d_i-1}[C_i])\,B \quad , \quad \text{for } d_i \neq 0 \qquad (6.34)$$

Choosing

$$F = -D^{*-1}(M + C^*) \qquad (6.35)$$

and

$$G = -D^{*-1}A \qquad (6.36)$$

where

$$\Lambda = \text{diag}(\lambda_i) \tag{6.37}$$

$$M_i = 0 \quad \text{if} \quad d_i = 0$$

$$= \sum_{k=0}^{d_i-1} \alpha_{ki} y_i^{(k)} \quad \text{if} \quad d_i \neq 0 \tag{6.38}$$

co-oeffecients α_{ki} and λ_i are arbitrary. We get

$$\underline{y}^* = - M + \Lambda \underline{v} \tag{6.39}$$

If $d_i = 2$, then we can write

$$y_i^* = - \alpha_{0i} y_i - \alpha_{1i} y_i^{(1)} + \lambda_i v_i \tag{6.40}$$

or

$$d^2 y_i / dt^2 + \alpha_{1i} dy_i/dt + \alpha_{0i} y_i = \lambda_i v_i \tag{6.41}$$

which represents the i^{th} decoupled subsystem with co-efficients α_{1i}, α_{0i} and λ_i that can be chosen arbitrarily.

5.2 CONTROL DESIGN

Suppose the differential-equations for the manipulator are written in the form

$$\underline{q}'' = \underline{f} + B\underline{\tau} \tag{6.42}$$

where

$$\underline{f}(\underline{q},\underline{q}') \in R^n \quad \text{and} \quad B(\underline{q},\underline{q}') \in R^{n\times n}$$

Choosing $y_i = q_i$ and with $d_i=2$ for i=1,...n,Equation 5.32 on page 37 can be written as

$$\underline{q}'' = C^* + D^*\underline{\tau} \tag{6.43}$$

where

$$\underline{\tau} = F + G\underline{v} = -D^{*-1}(M + C^* - \Lambda\underline{v})$$

Comparing Equation 5.42 and Equation 5.43 we see that

$$C^* = \underline{f} \quad , \quad D^* = B \tag{6.44}$$

Thus the non-linear feedback law for decoupling is given by

$$\underline{\tau} = -B^{-1}[\ \underline{f} + M - \Lambda \underline{v}\ ] \qquad (6.45)$$

Since we are considering only the motion of links 2 and 3, the equations of motion are also derived only for these two links (Appendix C). With the C^* and D^* matrices as derived in Appendix C the control laws can be written as

$$\tau_1 = 0 \qquad (6.46)$$

$$\underline{\tau}_{23} = -D^{*-1}[C^* + M - \Lambda \underline{v}] \qquad (6.47)$$

where

$$\underline{\tau}_{23} = [\tau_2 \quad \tau_3]^T$$

With this feedback input link 2 and link 3 subsystems are given by

$$q_2'' + \alpha_{12} q_2' + \alpha_{02} q_2 = \lambda_2 v_2 \qquad (6.48)$$

$$q_3'' + \alpha_{13} q_3' + \alpha_{03} q_3 = \lambda_3 v_3 \qquad (6.49)$$

v_2 and v_3 being the reference inputs can be chosen to be the desired joint positions q_{2d} and q_{3d} respectively. The time

interval for each step is .01 sec. The rise time is therefore chosen to be .01 sec and critical damping is assumed. With these specifications the following values are obtained for the co-efficients.

- $\lambda_i = 10000$
- $\alpha_{0i} = 10000$
- $\alpha_{1i} = 200$, for i=2,3

Substituting these values in Equation 5.47 on page 40 completes the the design of the control laws. At time step k the reference input vector is simply the desired joint positions at the beginning of the k+1 time step. Summarising, the control laws are given by

$$\tau_1 = 0 \tag{6.50}$$

$$\tau_2 = - [c_1 + \alpha_{02}q_{2a} + \alpha_{12}q_{2a}' - \lambda_2 q_{2d}]/c_2 \tag{6.51}$$

$$\tau_3 = - [c_3 + \alpha_{03}q_{3a} + \alpha_{13}q_{3a}' - \lambda_3 q_{3d}]/c_4 \tag{6.52}$$

where $c_1,..c_4$ are given as in Appendix C. Subscript a refers to measured values of joint position and velocity at time instant k and subscript d represents the desired values of joint position at time instant k+1. Since the control laws have been derived on the assumption that link 1 is held constant at zero position, $\tau_1=0$. However due to coupling with links 2 and 3, joint 1 does move a little inspite of its actuation torque being zero, when the simulation is carried out.

6.0 CHAPTER 6-ADAPTIVE CONTROL BASED ON PARAMETER ESTIMATION

The concept of nominal and perturbed control can be modified by the way the perturbed input is computed. The algorithm described in Chapter 4 calculates the perturbed input based on a linearised model obtained by computation. Since this linearised model involves considerably more computation than the dynamic model we could consider estimating the model parameters by a suitable recursive estimation algorithm like RLS. In addition to reducing the amount of computation ,there is the advantage of obtaining a more accurate model based on actual measurements. Since the perturbed input is intended to reduce the deviation from the desired trajectory we could expect better performance,especially in the presence of noise or disturbance inputs than with a computed model.

6.1 ESTIMATION OF MODEL PARAMETERS

The discrete-time equivalent of the perturbed state equation is given by

$$\underline{x}(k+1) = F(k)\underline{x}(k) + G(k)\underline{u}(k) \qquad (7.53)$$

Define the measurement vector $\underline{\phi}(k)$ as

$$\underline{\phi}(k) = \begin{bmatrix} \underline{x}(k) \\ \\ \underline{u}(k) \end{bmatrix} \quad \varepsilon \quad R^{3n} \tag{7.54}$$

and the parameter matrix $\theta(k)$ by

$$\theta(k) = [F(k) \quad G(k)] \quad \varepsilon \quad R^{2n \times 3n} \tag{7.55}$$

The i^{th} row of the above matrix equation is

$$\underline{\theta}_i(k) = [\, f_{i1}\ f_{i2} \ldots . f_{i2n}\ g_{i1} \ldots . g_{in}]^T \quad \varepsilon \quad R^{3n} \tag{7.56}$$

In terms of $\underline{\theta}_i$ and $\underline{\phi}$ the i^{th} state equation can be written as

$$x_i(k+1) = \underline{\phi}(k)^T \underline{\theta}_i(k) \quad , \ i=1,2 \ldots . 2n \tag{7.57}$$

Let $\underline{\theta}_i^E(k)$ be the parameter estimate vector.

Applying RLS estimation the following recursive estimator can be derived

$$\underline{\theta}_i^E(k+1)=\underline{\theta}_i^E(k)+P(k)\underline{\phi}(k)[\underline{\phi}^T(k)P(k)\underline{\phi}(k)+\rho]^{-1}[\underline{x}_i(k+1)-\underline{\phi}^T(k)\underline{\theta}_i^E(k)] \tag{7.58}$$

$$P(k+1)=P(k)-P(k)\underline{\phi}(k)[\underline{\phi}(k)^TP(k)\underline{\phi}(k)+\rho]^{-1}\underline{\phi}(k)^TP(k) \tag{7.59}$$

where ρ is a positive number less than 1 and $P(0)=cI_{3n}$ where c is a large positive number. Let

$F^E(k)$ and $G^E(k)$ be the estimated parameter matrices at time k.

The optimum control law based on the one-step criterion can be used again to compute the perturbed input $\underline{u}(k)$. The nominal input is computed the same way as in Chapter 5.

The performance of this algorithm is strongly dependent on the exponential forgeting factor ρ. The sensitivity of the algorithm to noise is reduced by larger values of ρ and good tracking is achieved by relatively smaller values of ρ. The actual value to be used is determined in a trial and error manner for the particular system that is being estimated. For this application a value of $\rho=0.85$ was seen to give the best overall performance.

7.0 CHAPTER 7-SIMULATION RESULTS

The three control algorithms described in the previous chapters are simulated here using the program listed in Appendix d. The geometric details of the three-link arm including mass and moment of inertia for each link is shown in Table 1 on page 49.

The motion task specified is a simple straight line motion of the manipulator tip from one point to another at a constant velocity. In the base co-ordinate system these points are [0.30,0.1524,0.1530] and [0.0,0.1524,0.653]. The time for the task is chosen to be 1 second.

7.1 CONTROL BASED ON LINEARISED MODEL

In this algorithm the control input is computed in two parts- the nominal and the perturbed. The nominal input is based on the desired joint position,velocity, and acceleration and the perturbed input is computed based on a linearised model.

Figure 9 on page 50 and Figure 10 on page 51 show the desired and actual trajectories for the three joints. Figure 11 on page 52 and Figure 12 on page 53 show the nominal and perturbed control torques and forces. Figure 13 on

page 54 and Figure 14 on page 55 show the corresponding actuator motor voltages to be applied.

7.2 ADAPTIVE CONTROL ALGORITHM

This algorithm uses the same dynamic model as the previous algorithm to compute the nominal torque. The perturbed input is computed based on the estimated linearised model parameters using the RLS algorithm for slowly time-varying parameters. The performance of the estimation algorithm can be seen by comparing the estimated parameters with the computed parameters from the previous algorithm. Figure 15 on page 56 through Figure 22 on page 63 show the results of the simulation.

7.3 NON-LINEAR CONTROL ALGORITHM

As was described in the previous chapter the control inputs in this case can be evaluated from a formula that is derived based on the equations of motion of this manipulator The desired and actual positions and velocities are used along with other constants to arrive at the actuator torques and the corresponding control inputs. The results are plotted in Figure 23 on page 64 through Figure 28 on page 69.

In order to evaluate the comparitive performance of the three algorithms the errors in joint position are plotted in

Figure 29 on page 70,Figure 30 on page 71 and Figure 31 on page 72.

Table 1. Three-link arm mechanical data

		LINK 1	LINK 2	LINK 3
JOINT TYPE		ROTATIONAL	ROTATIONAL	LINEAR
LINK MASS (KG)		9.29	5.01	4.25
MOMENT OF INERTIA ($KG-M^2$)	IXX	0.276	0.108	2.510
	IYY	0.255	0.018	2.510
	IZZ	0.071	0.100	0.006
LOCATION OF COG WRT JOINT CENTER (M)	$\bar{X}$	0.0	0.0	0.0
	$\bar{Y}$	0.0175	-0.1054	0.0
	$\bar{Z}$	-0.1105	0.0	-0.6447
JOINT CLASSIFICATION (SEE APPENDIX A)		TYPE 1	TYPE 4	TYPE 4

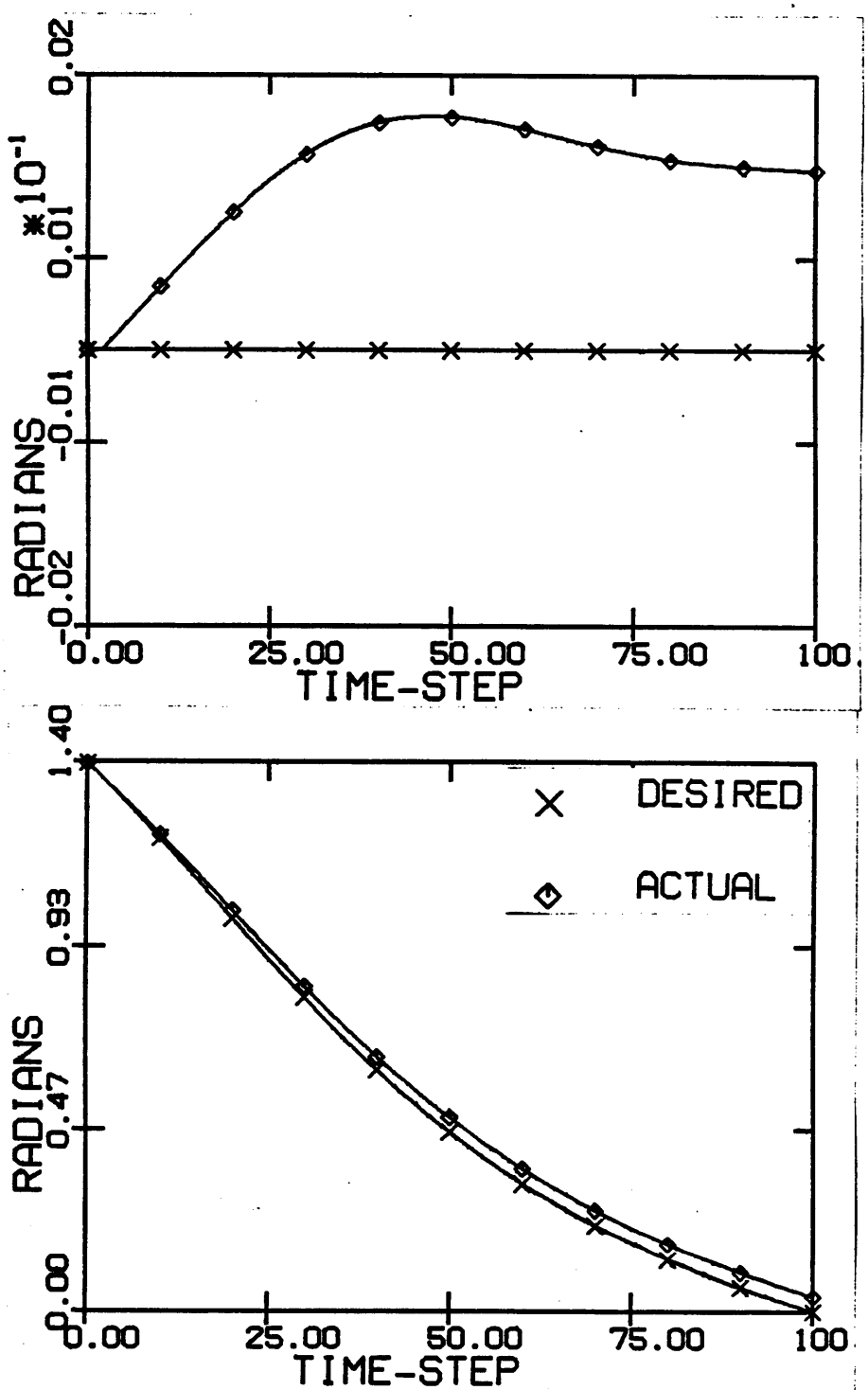


Figure 9. Linearised Model.: Joint 1,Joint 2 trajectories.

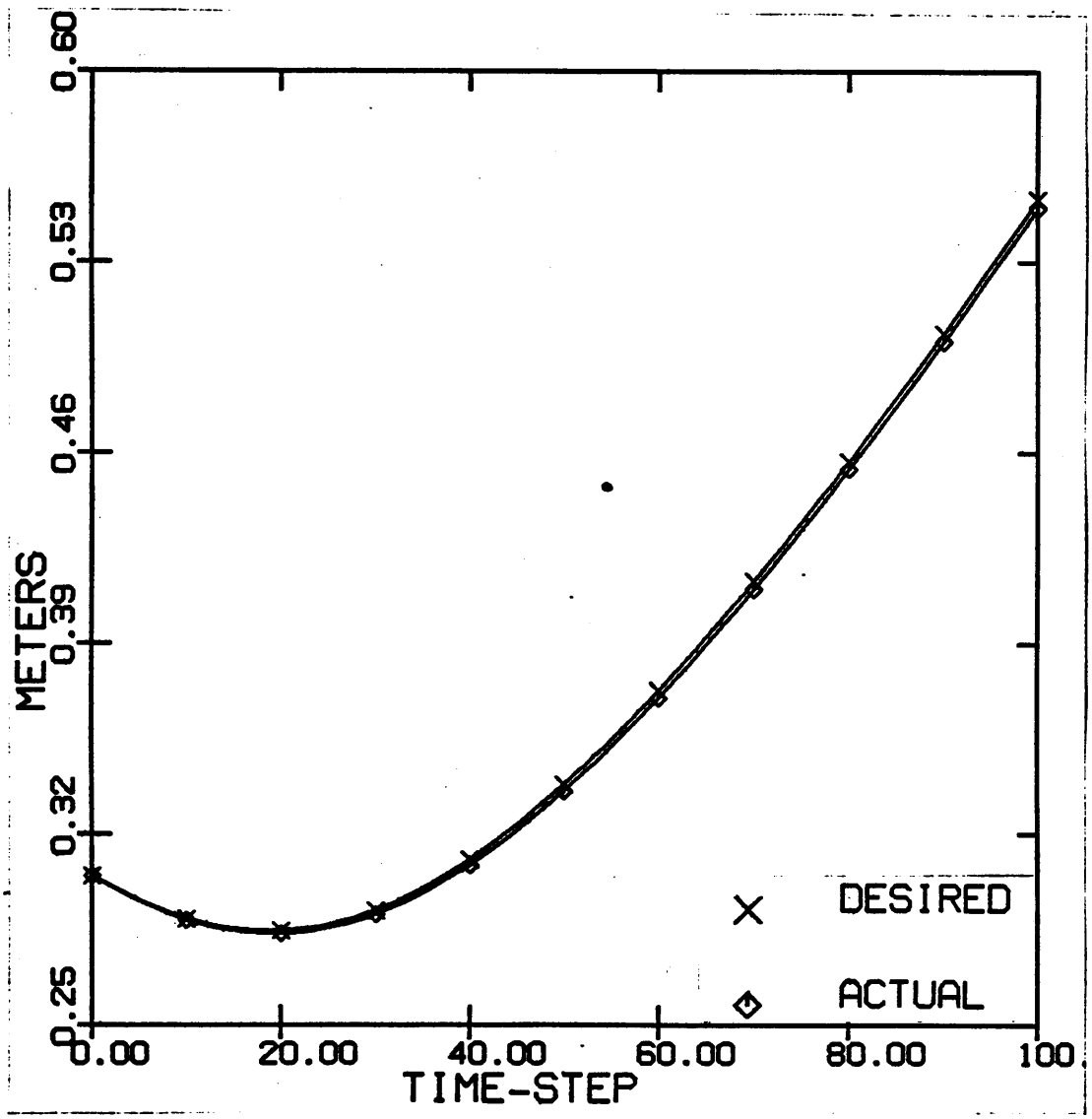


Figure 10. Linearised Model.: Joint 3 trajectory.

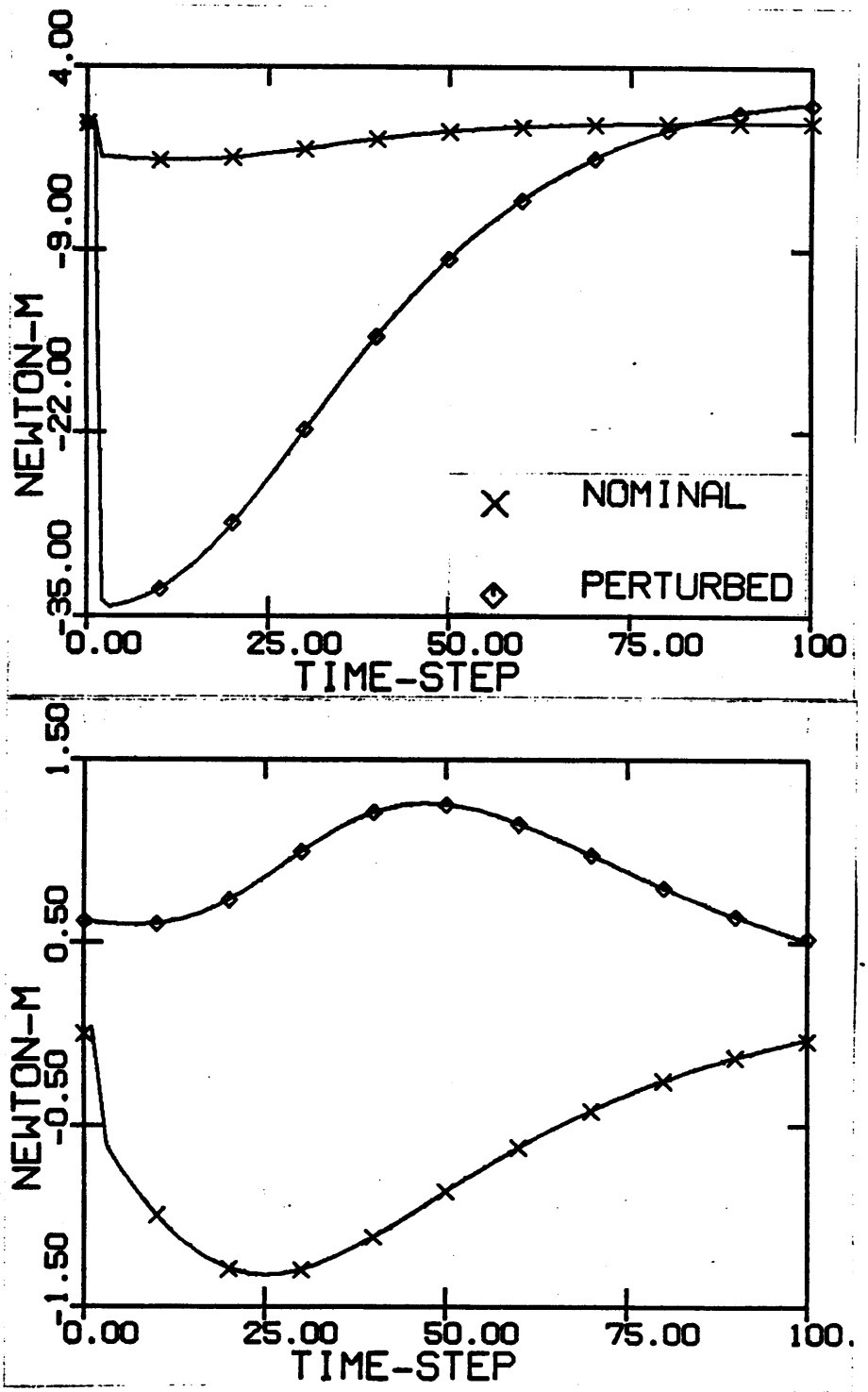


Figure 11. Linearised Model.: Joint 1,Joint 2 Nominal and Perturbed torques/forces.

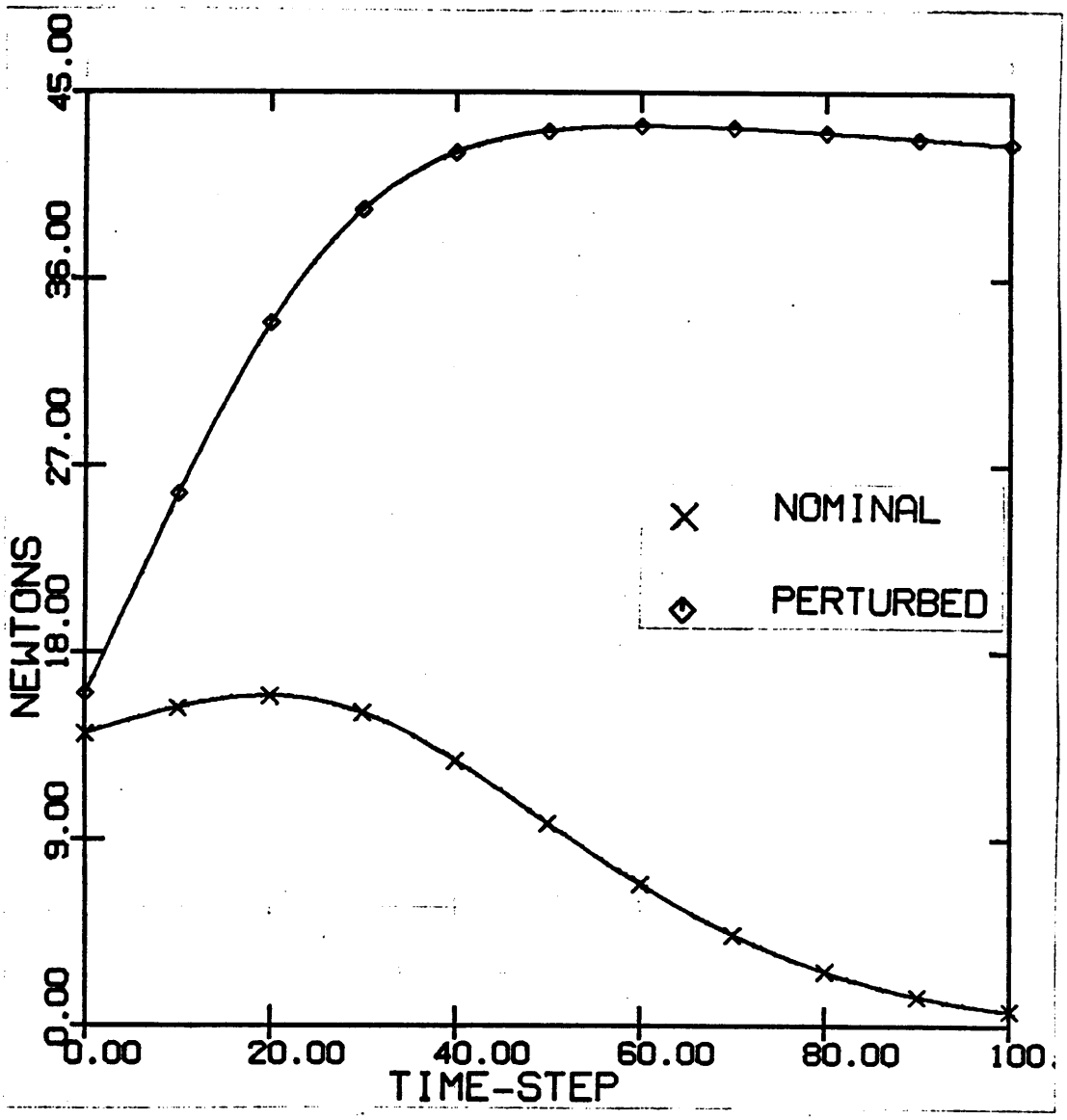


Figure 12. Linearised Model.: Joint 3 Nominal and Perturbed torques/forces.

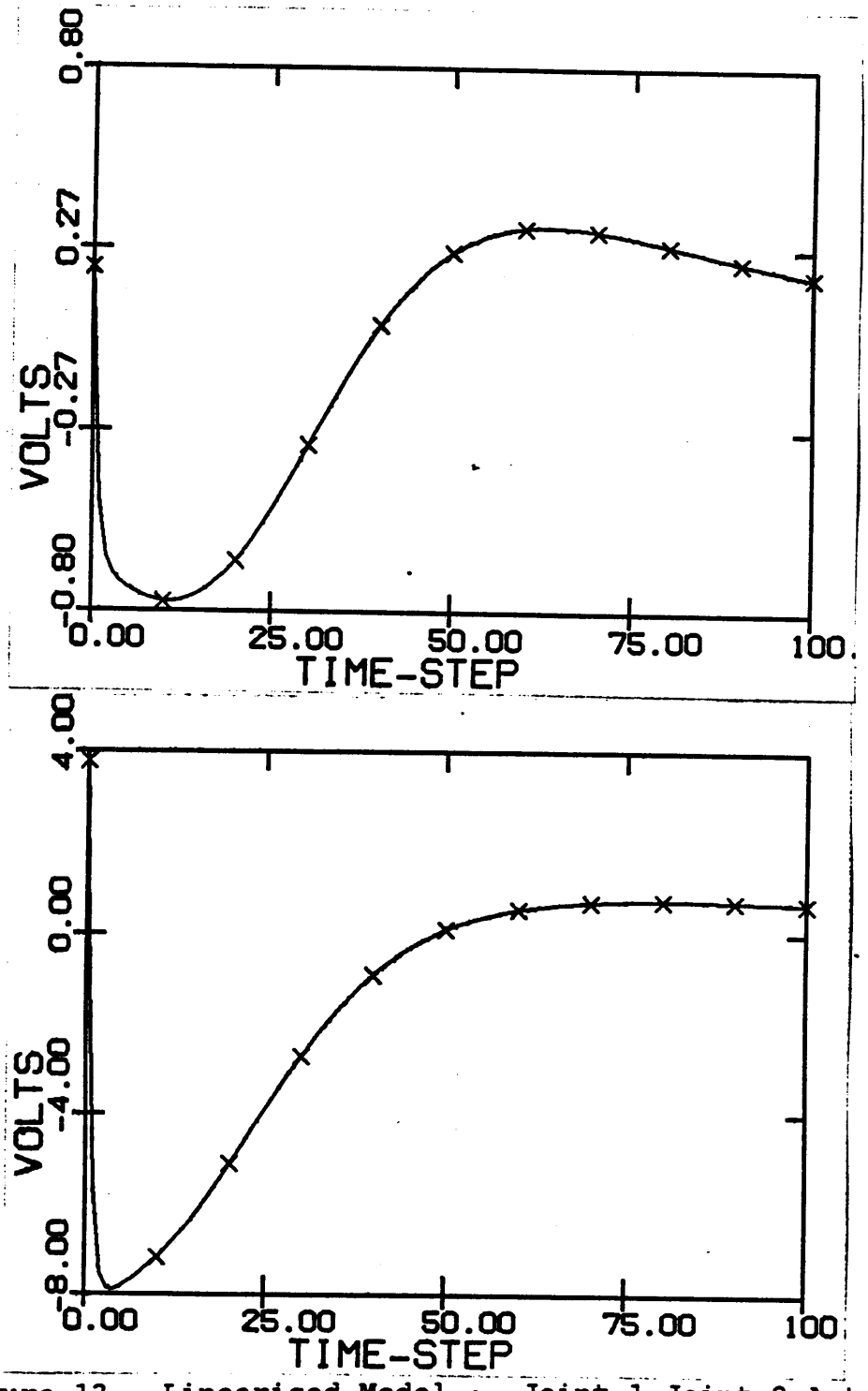


Figure 13. Linearised Model.: Joint 1,Joint 2 Actuator armature voltage.

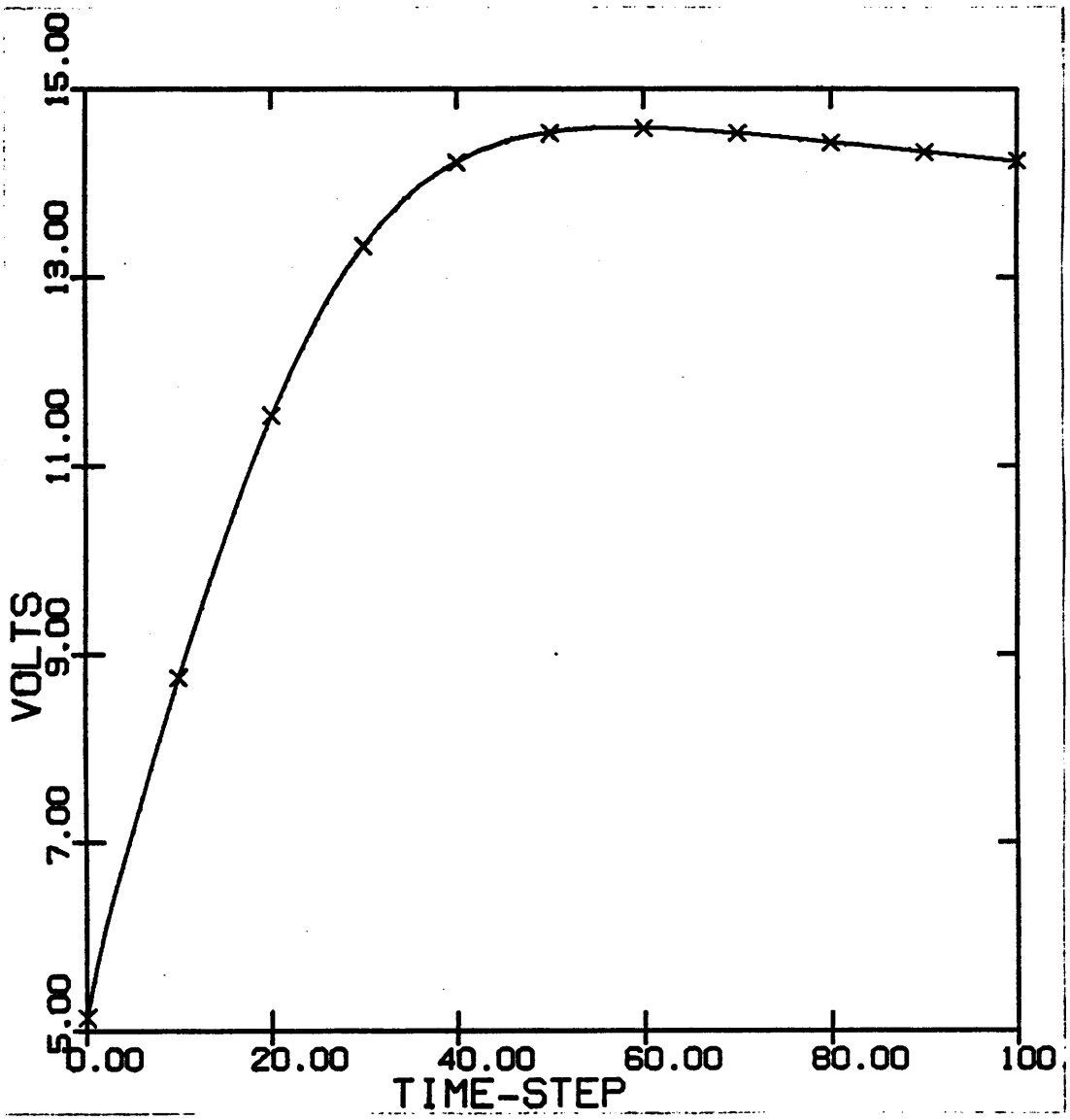


Figure 14. Linearised Model.: Joint 3 Actuator armature voltage.

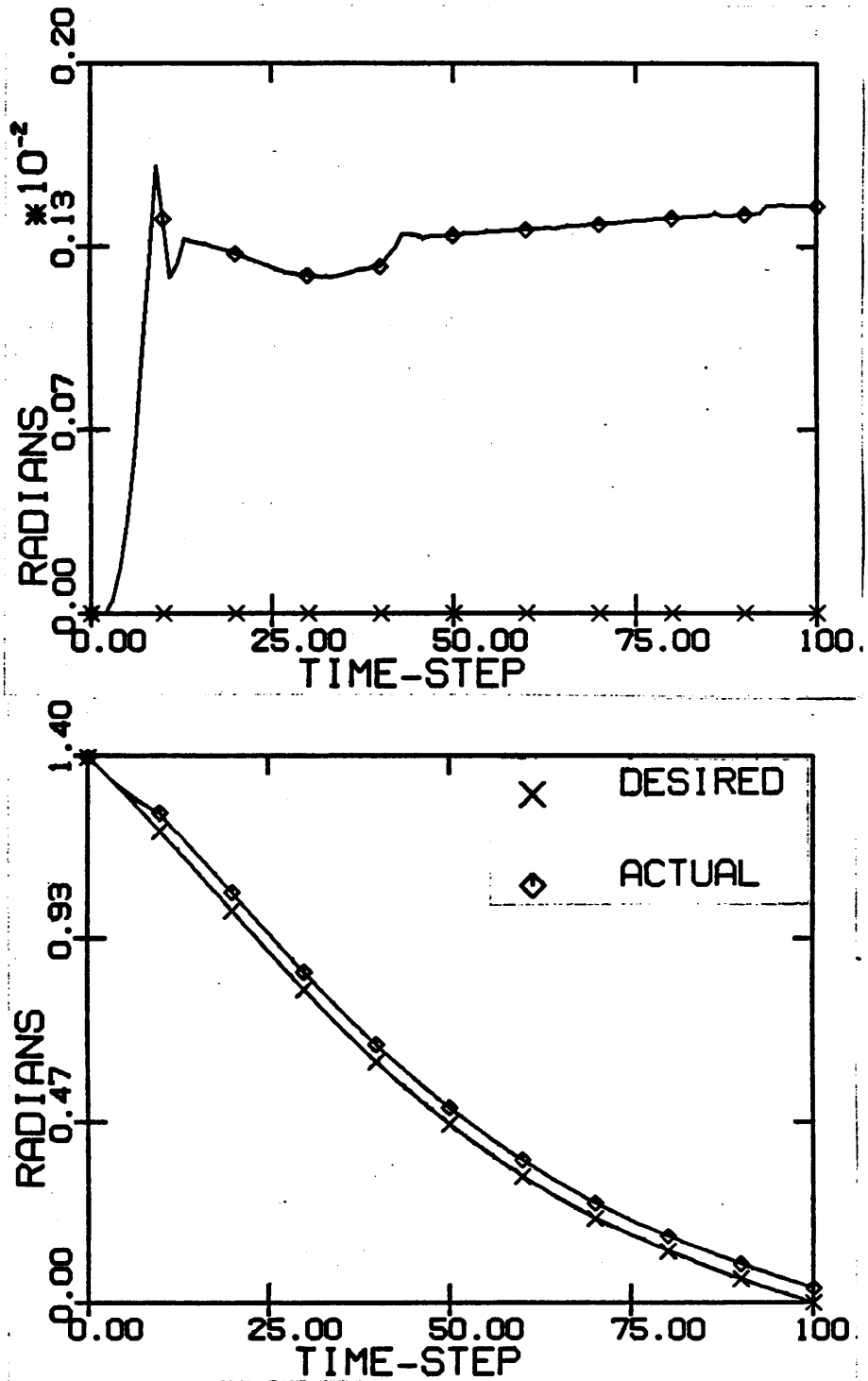


Figure 15. Adaptive control.: Joint 1,Joint 2 trajectories.

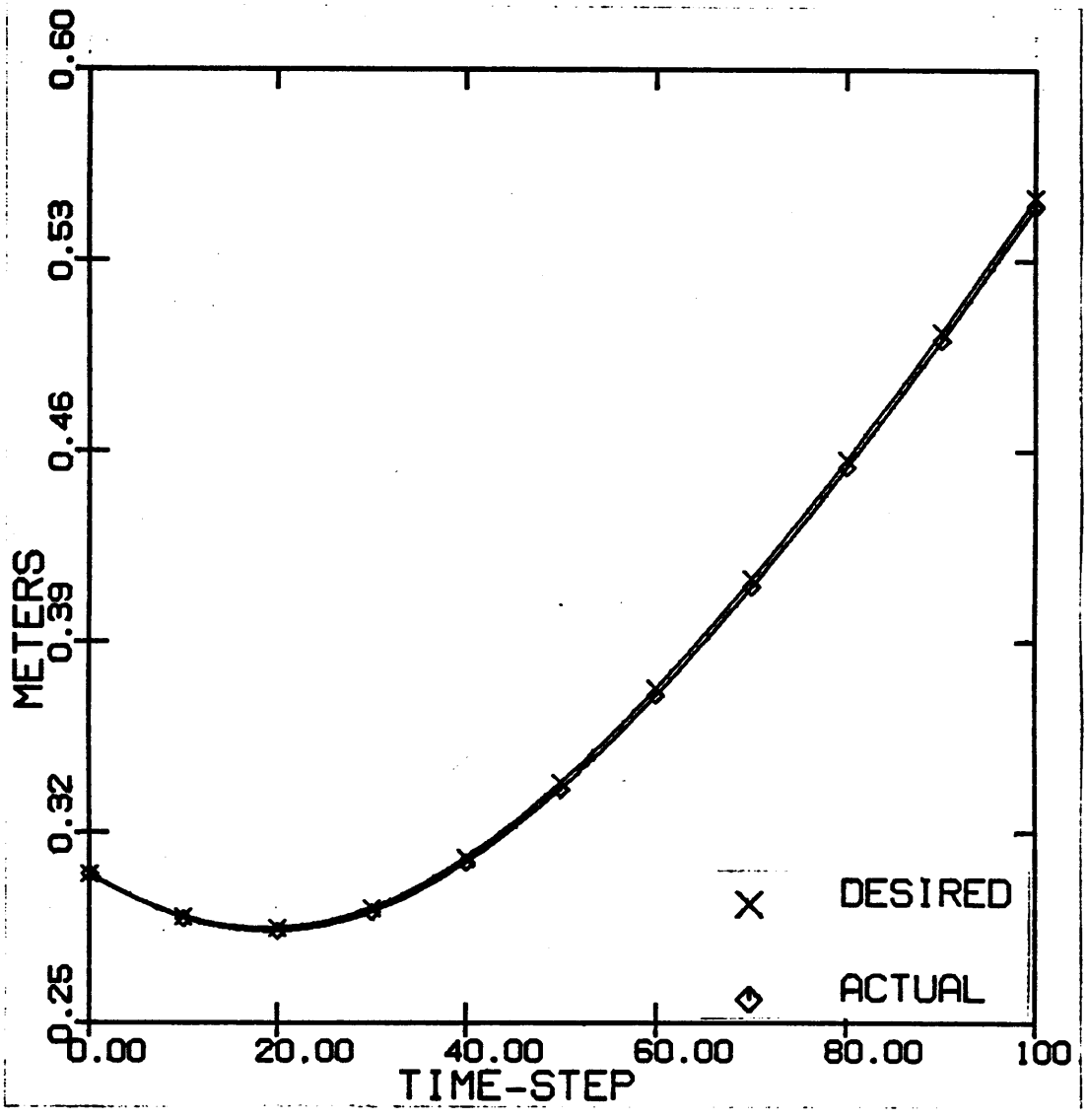


Figure 16. Adaptive control.: Joint 3 trajectory.

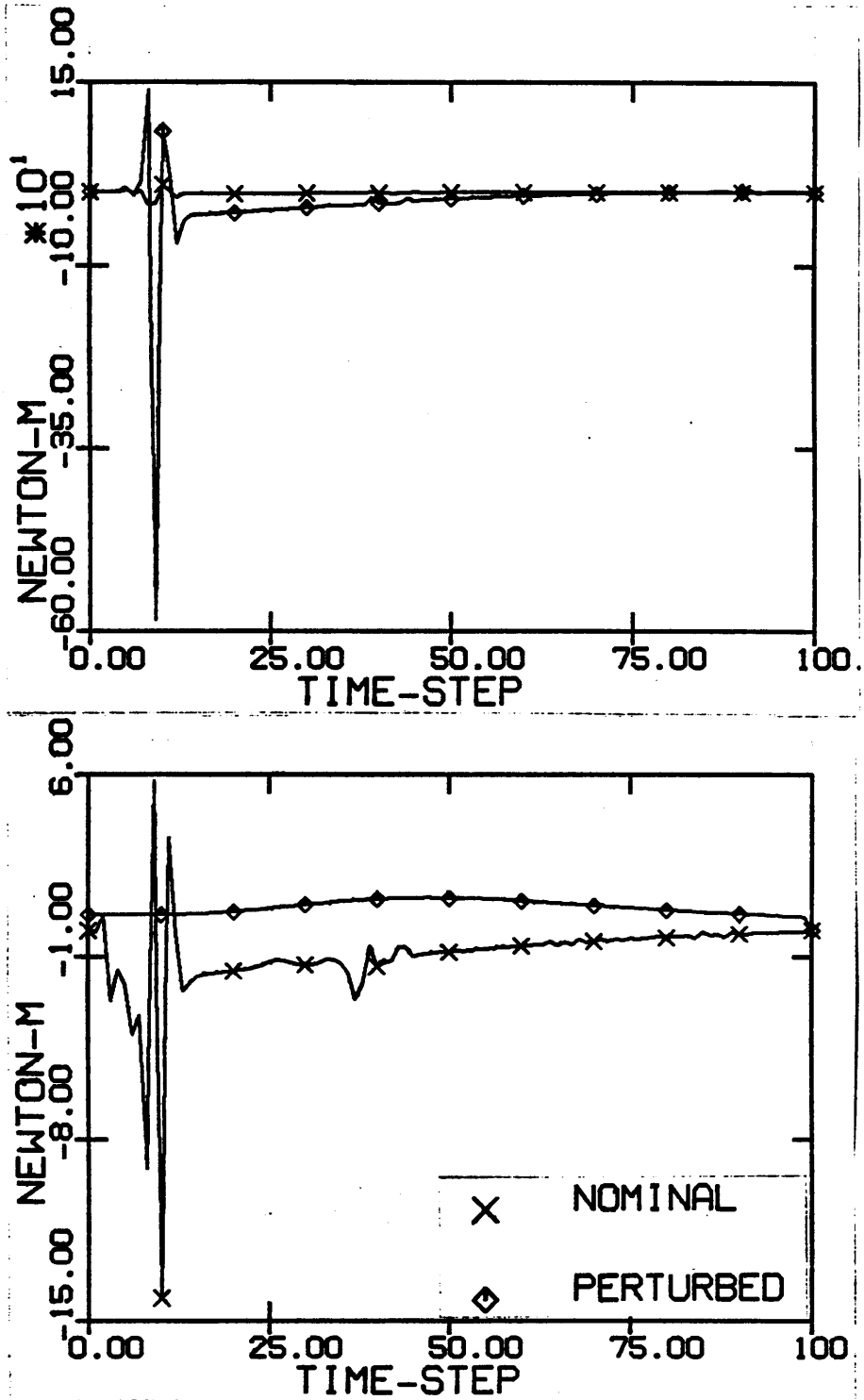


Figure 17. Adaptive control.: Joint 1,Joint 2 Nominal and Perturbed torques/forces.

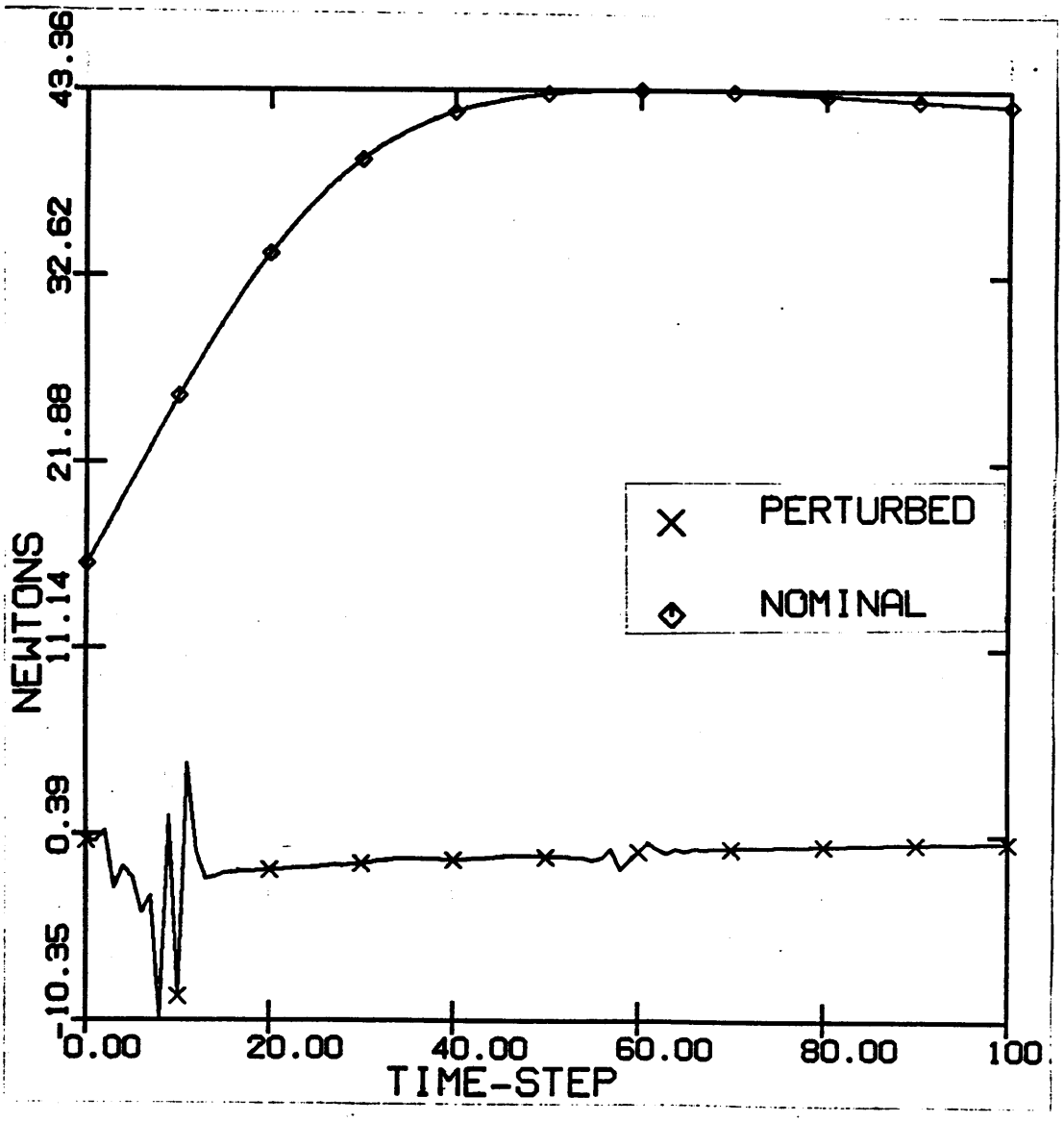
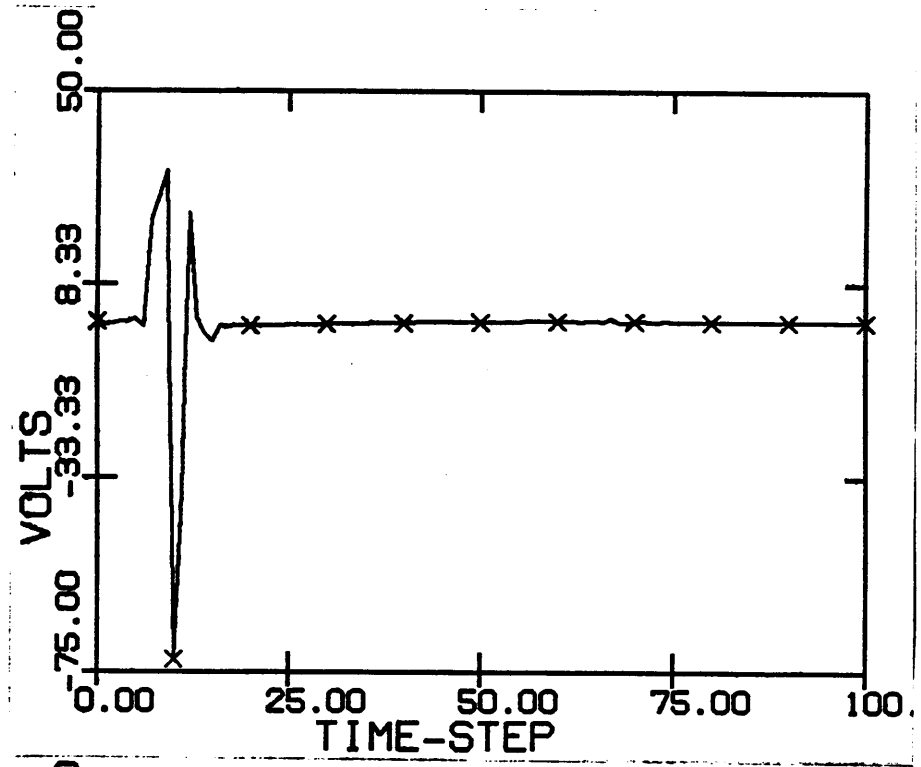


Figure 18. Adaptive control.: Joint 3 Nominal and Perturbed torques/forces.



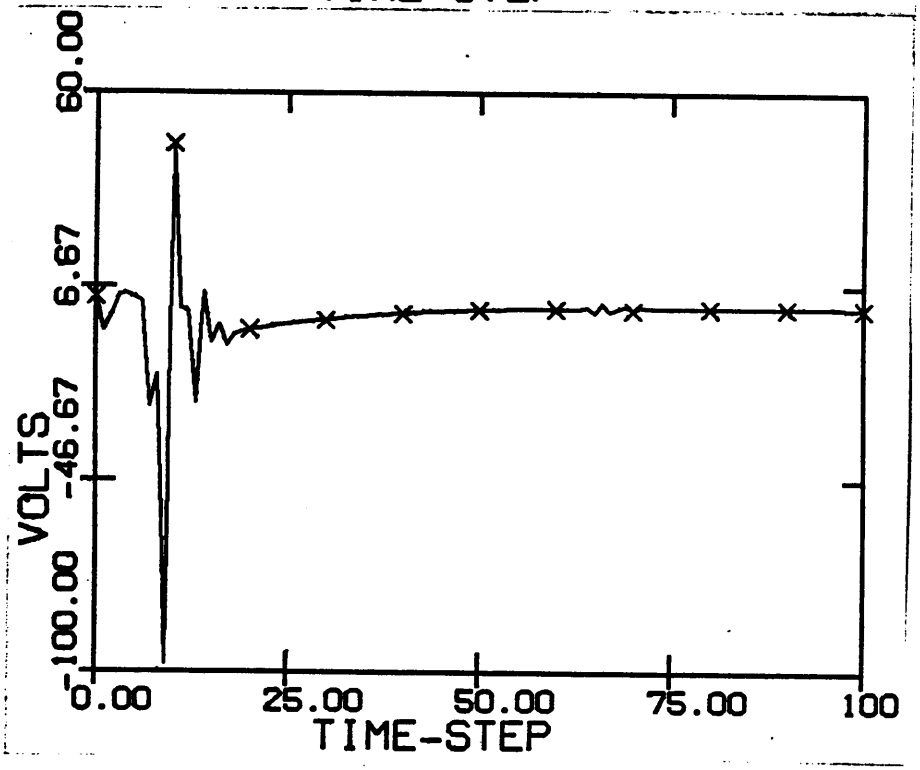


Figure 19. Adaptive control.: Joint 1,Joint 2 Actuator armature voltage.

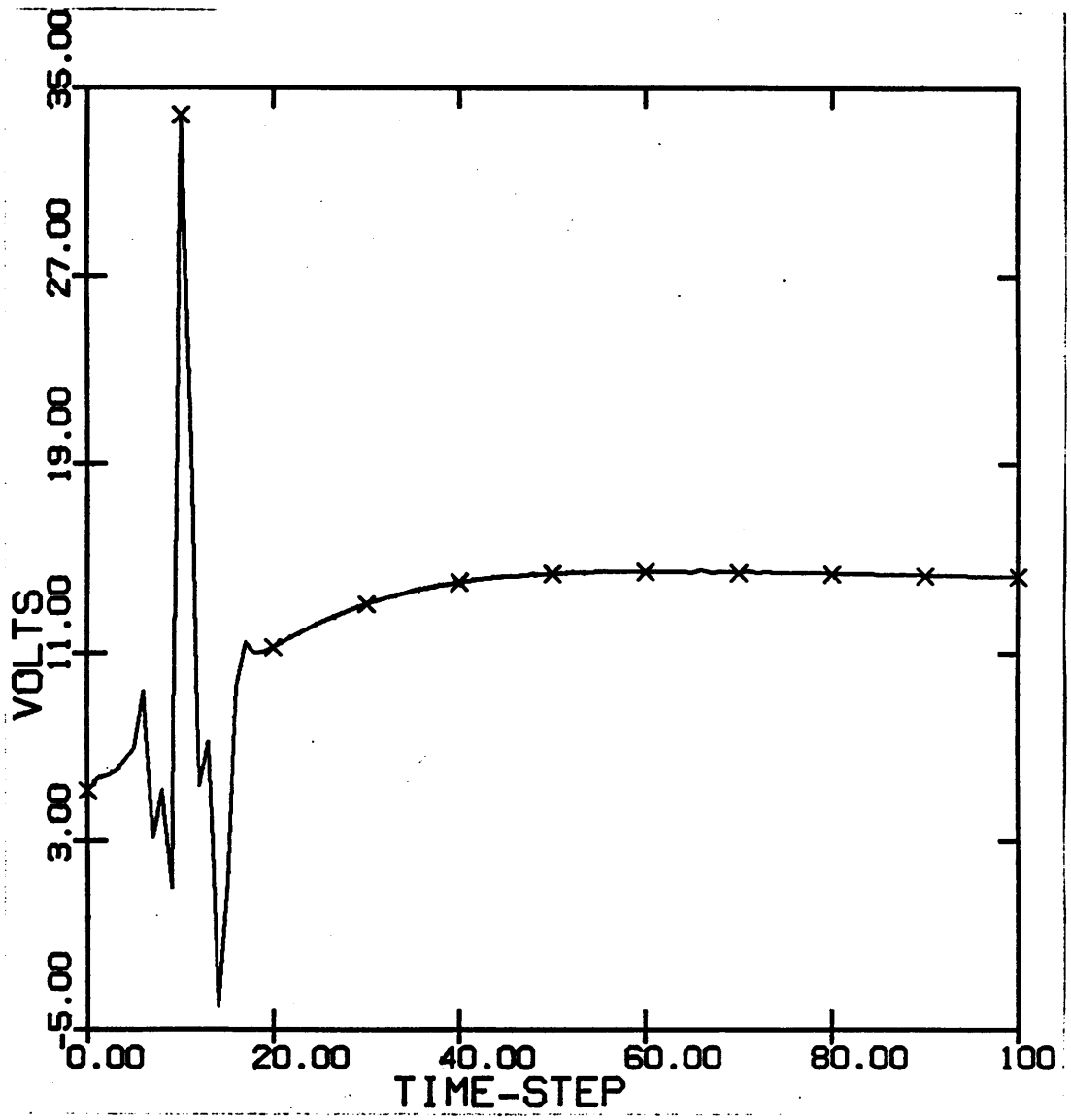


Figure 20. Adaptive control.: Joint 3 Actuator armature voltage.

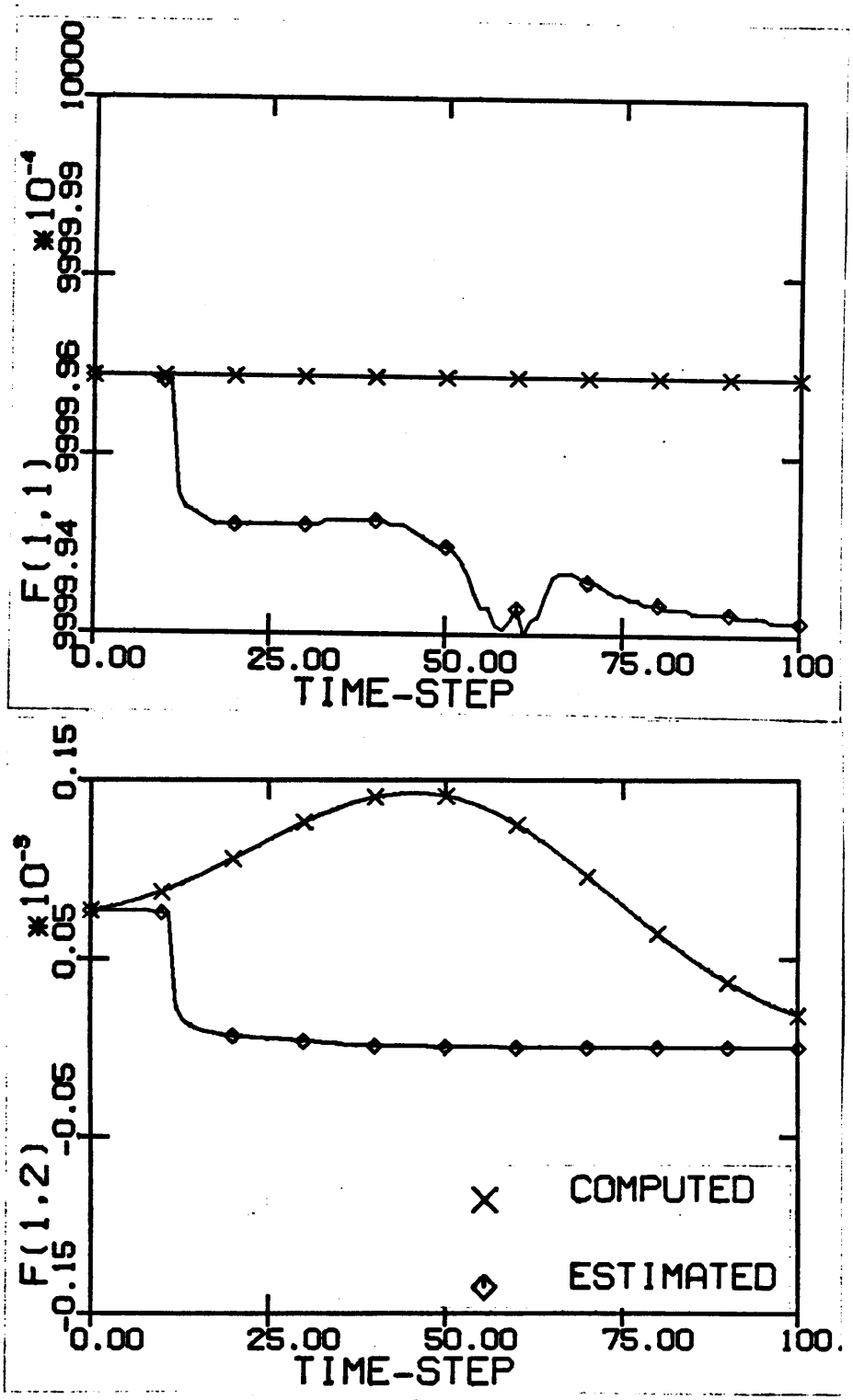


Figure 21. Adaptive control.: Estimated and computed linearised model parameters-F(1,1), F(1,2)

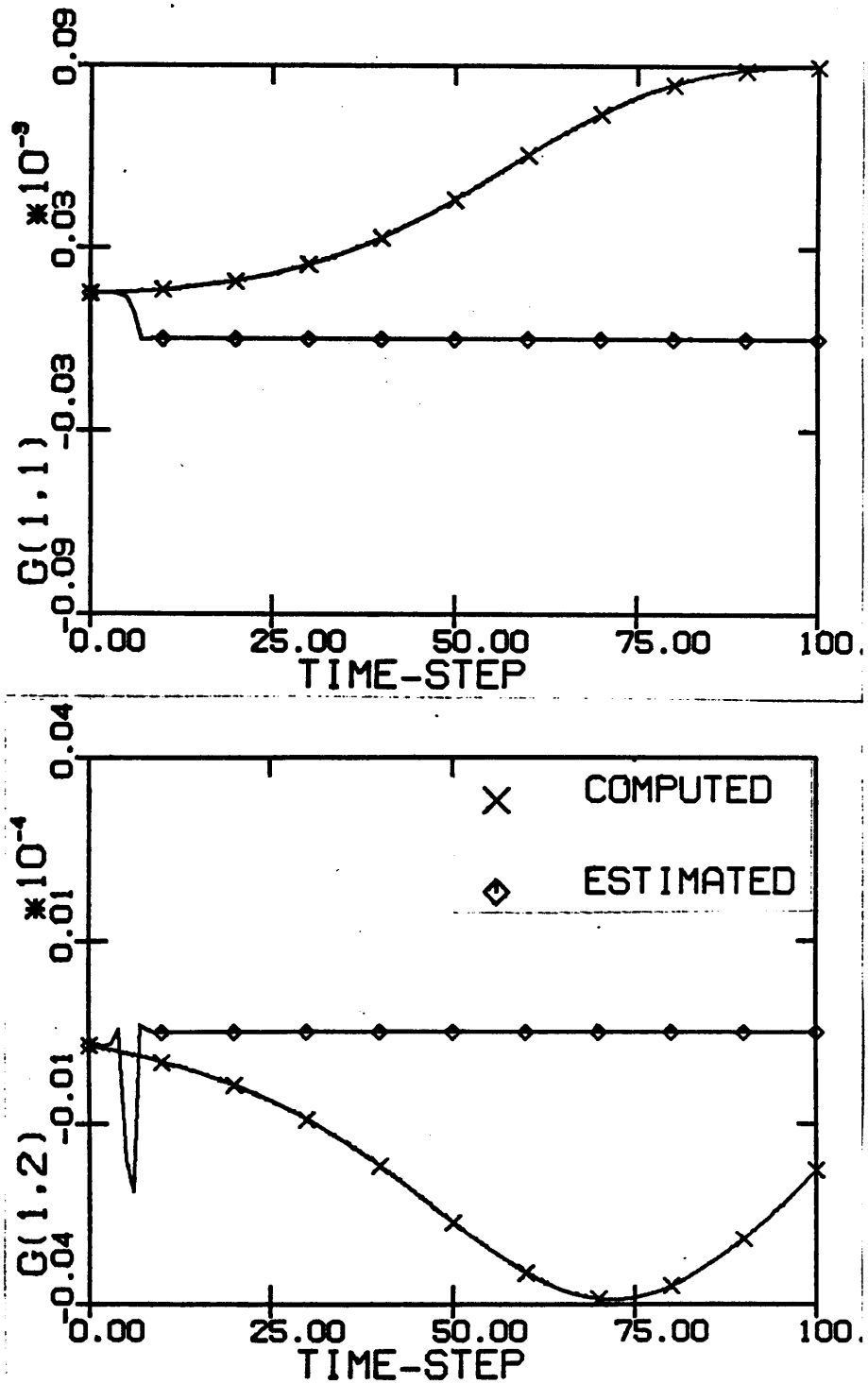


Figure 22. Adaptive control.: Estimated and computed linearised model parameters,G(1,1), G(1,2)

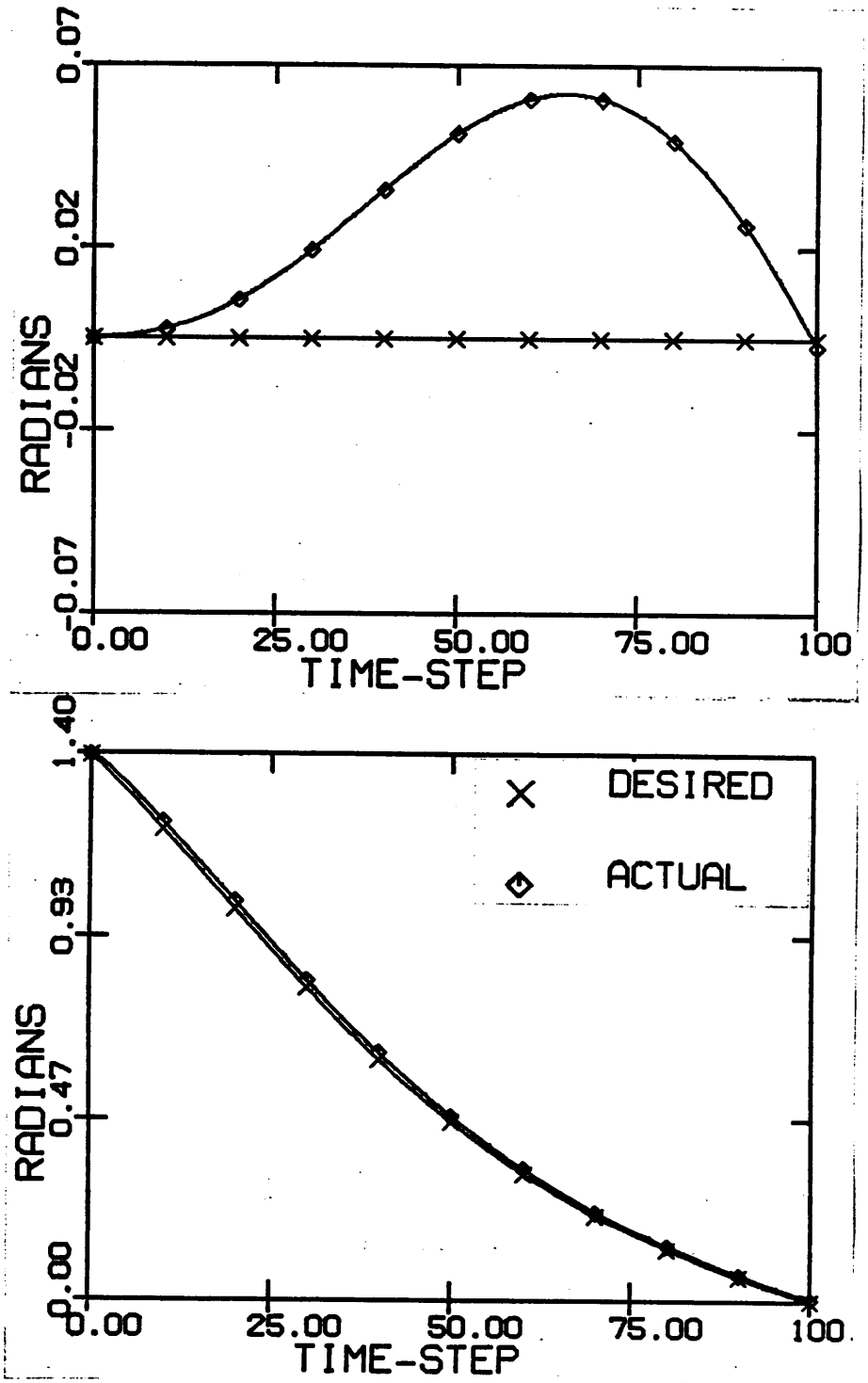


Figure 23. Non-linear control.: Joint 1,Joint 2 trajectories.

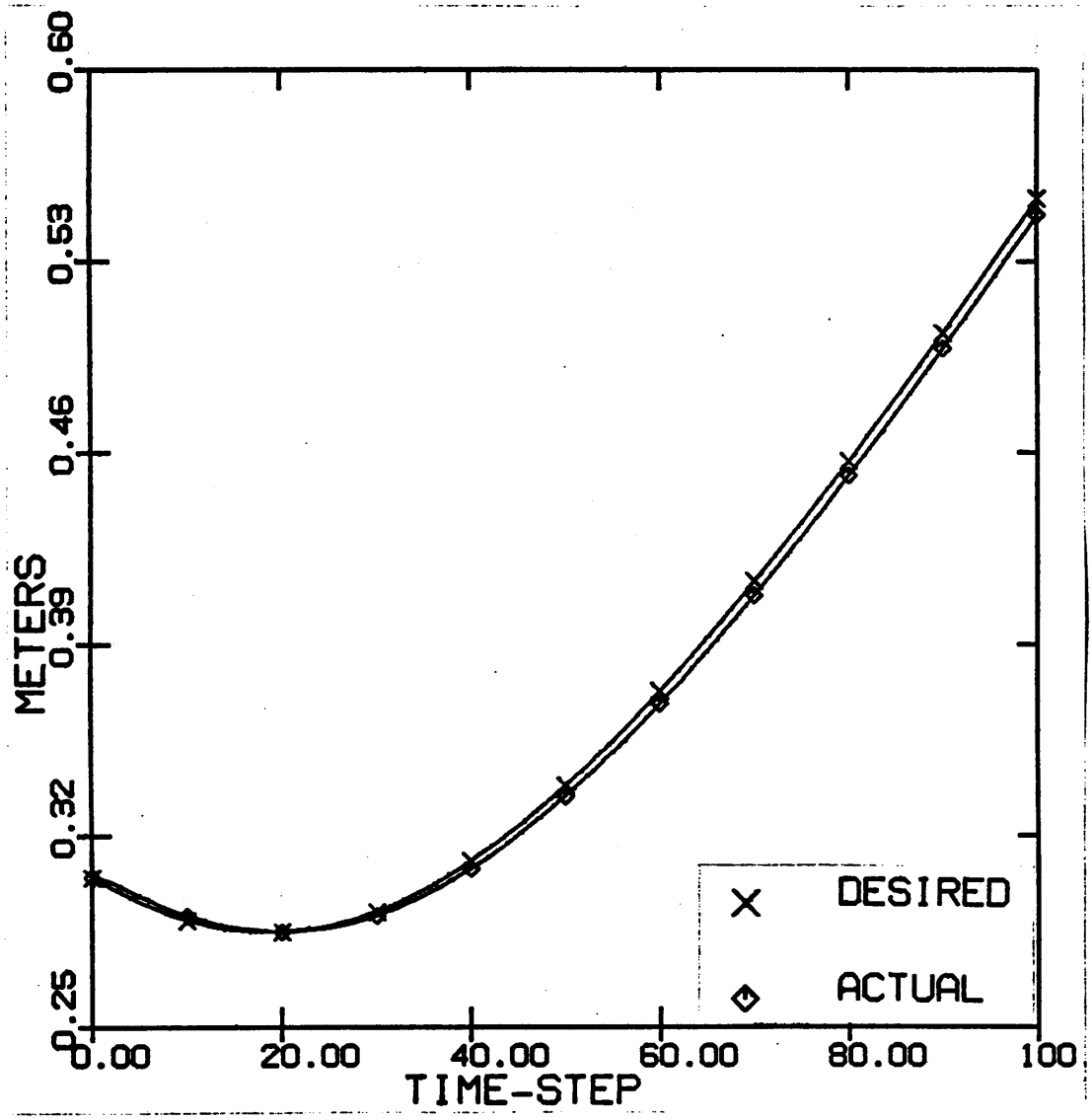


Figure 24. Non-linear control.: Joint 3 trajectory.

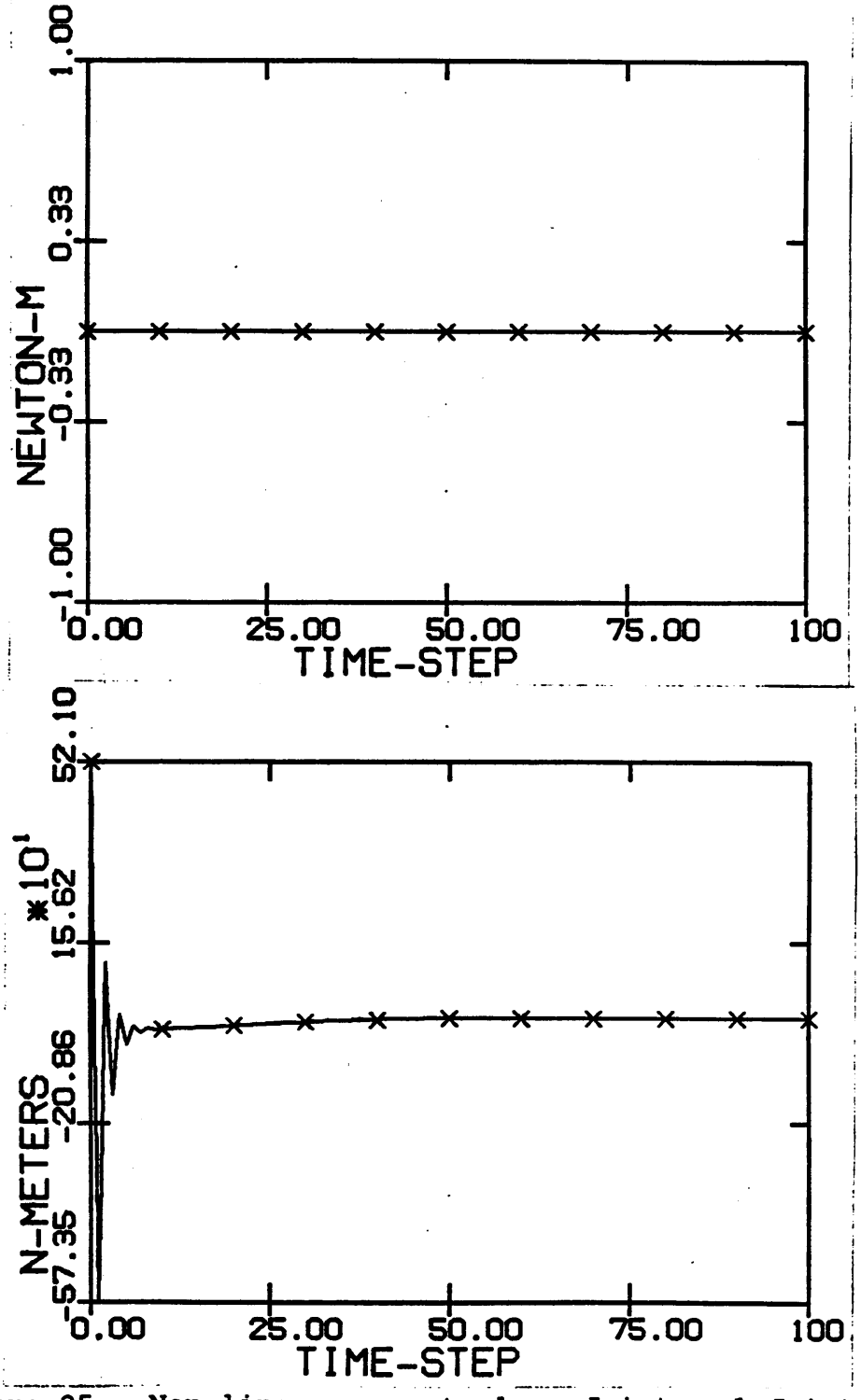


Figure 25. Non-linear control.: Joint 1,Joint 2 Torque/Force.

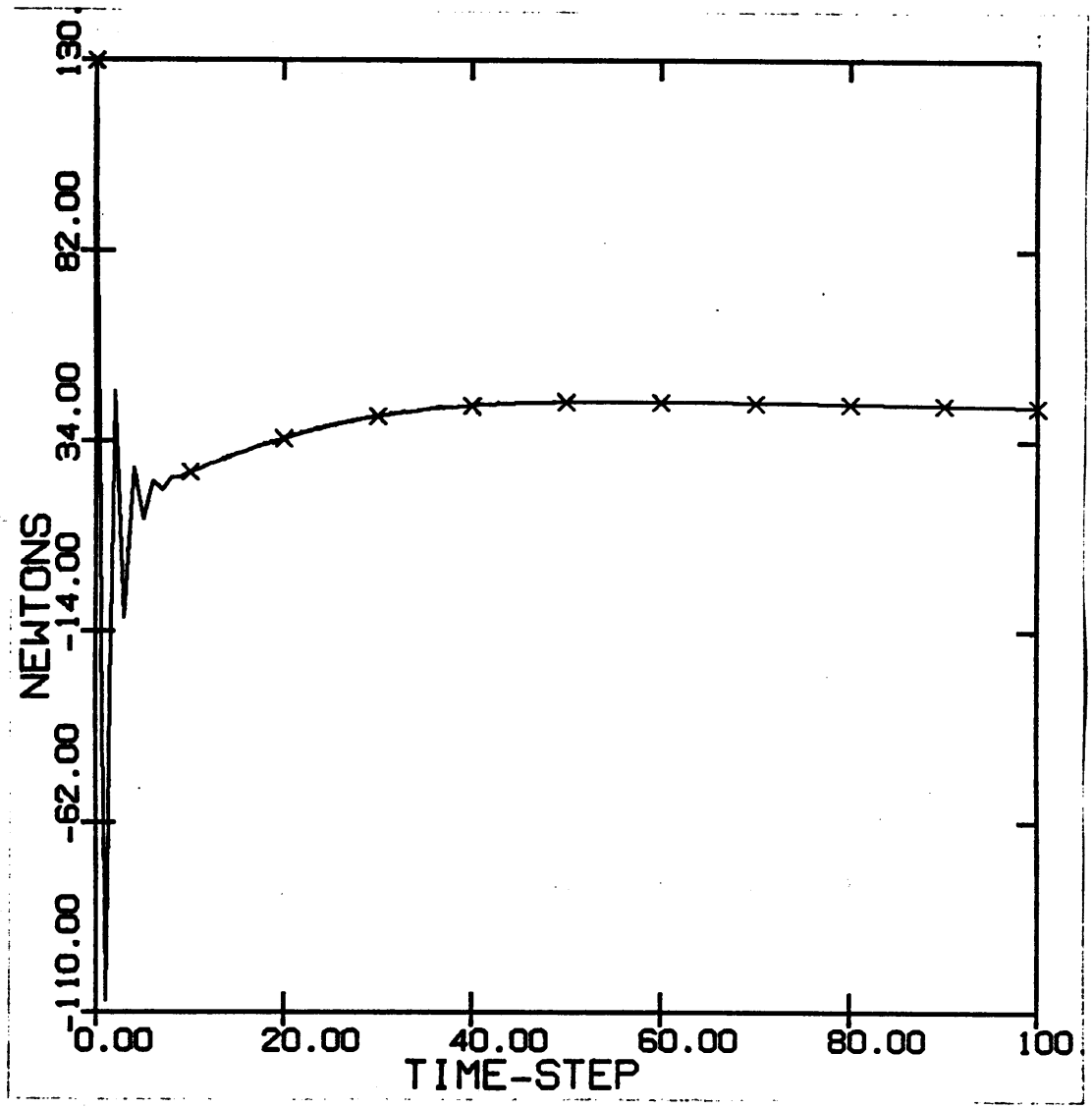


Figure 26. Non-linear control.: Joint 3 Torque/Force.

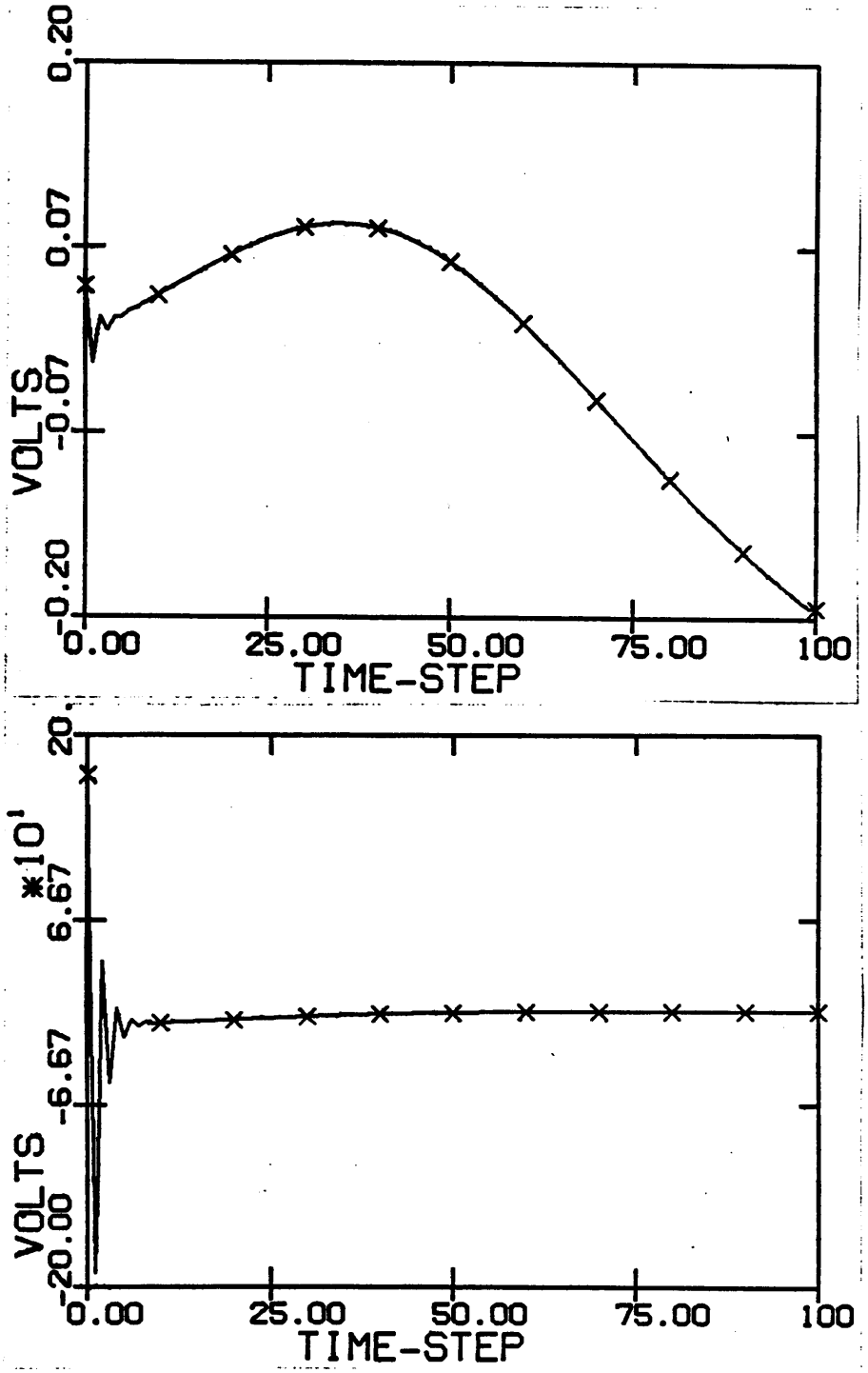


Figure 27. Non-linear control.: Joint 1,Joint 2 Actuator armature voltage.

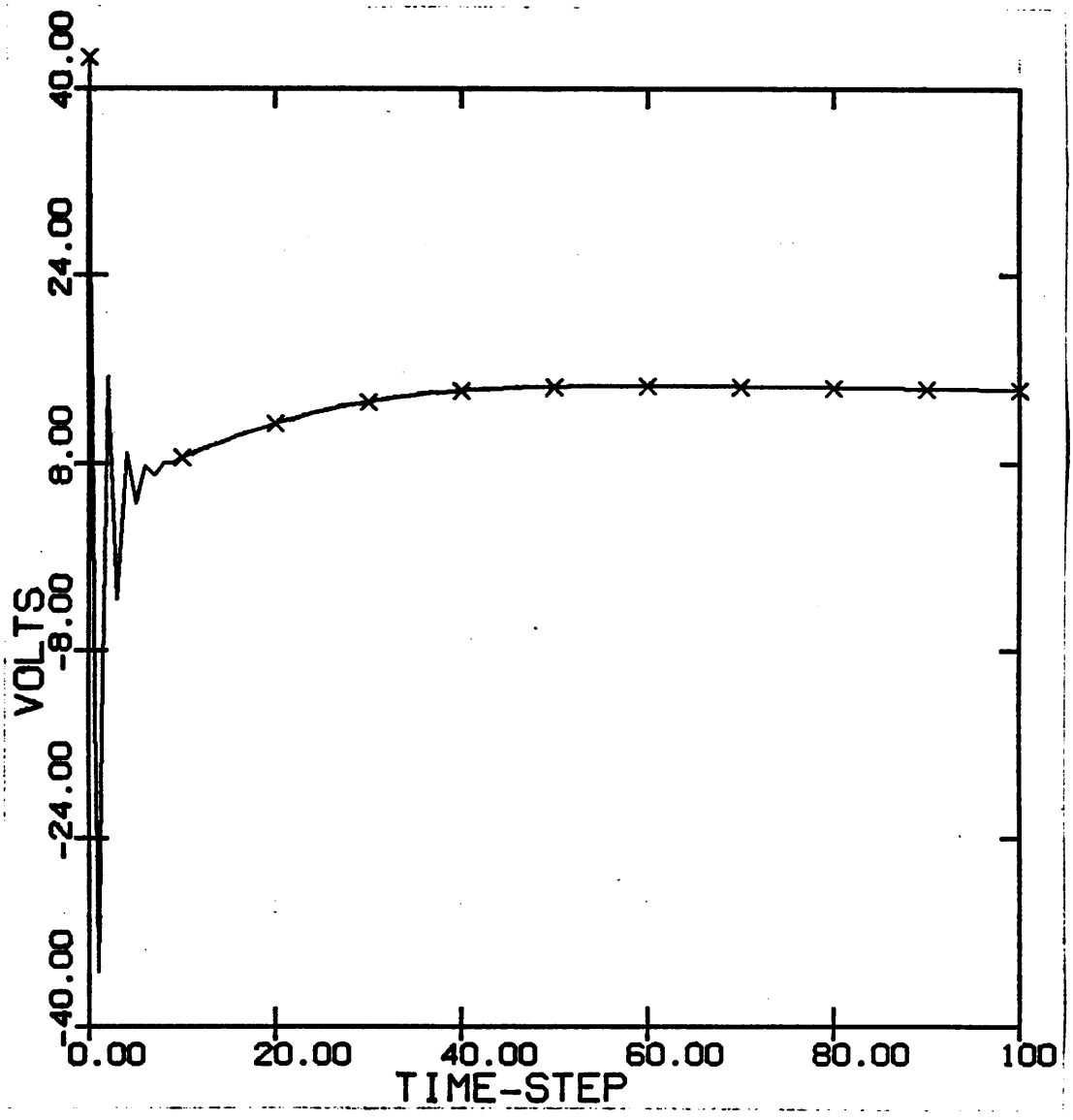


Figure 28. Non-linear control.: Joint 3 Actuator armature voltage.

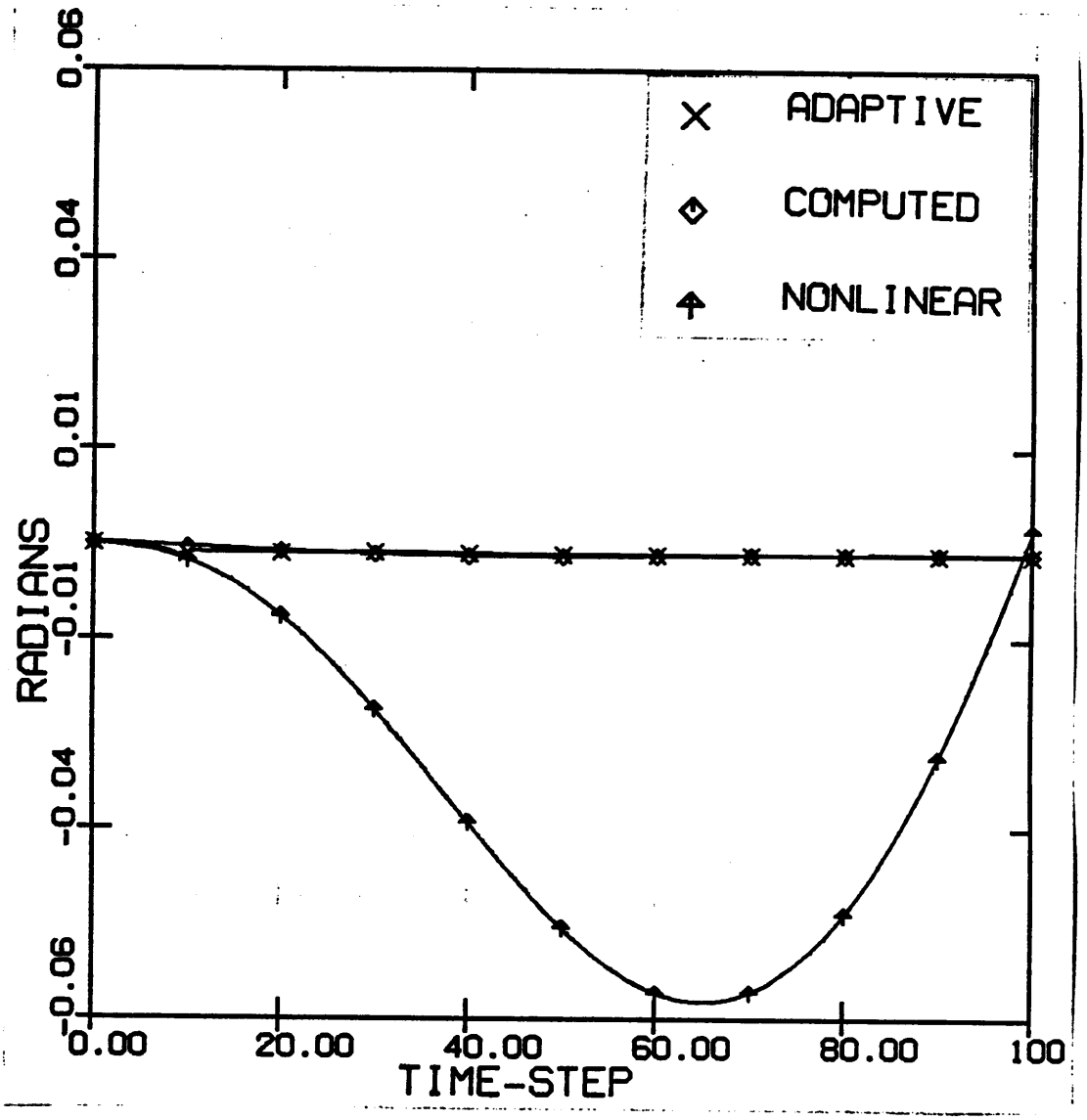


Figure 29. Tracking Errors.: Joint 1.

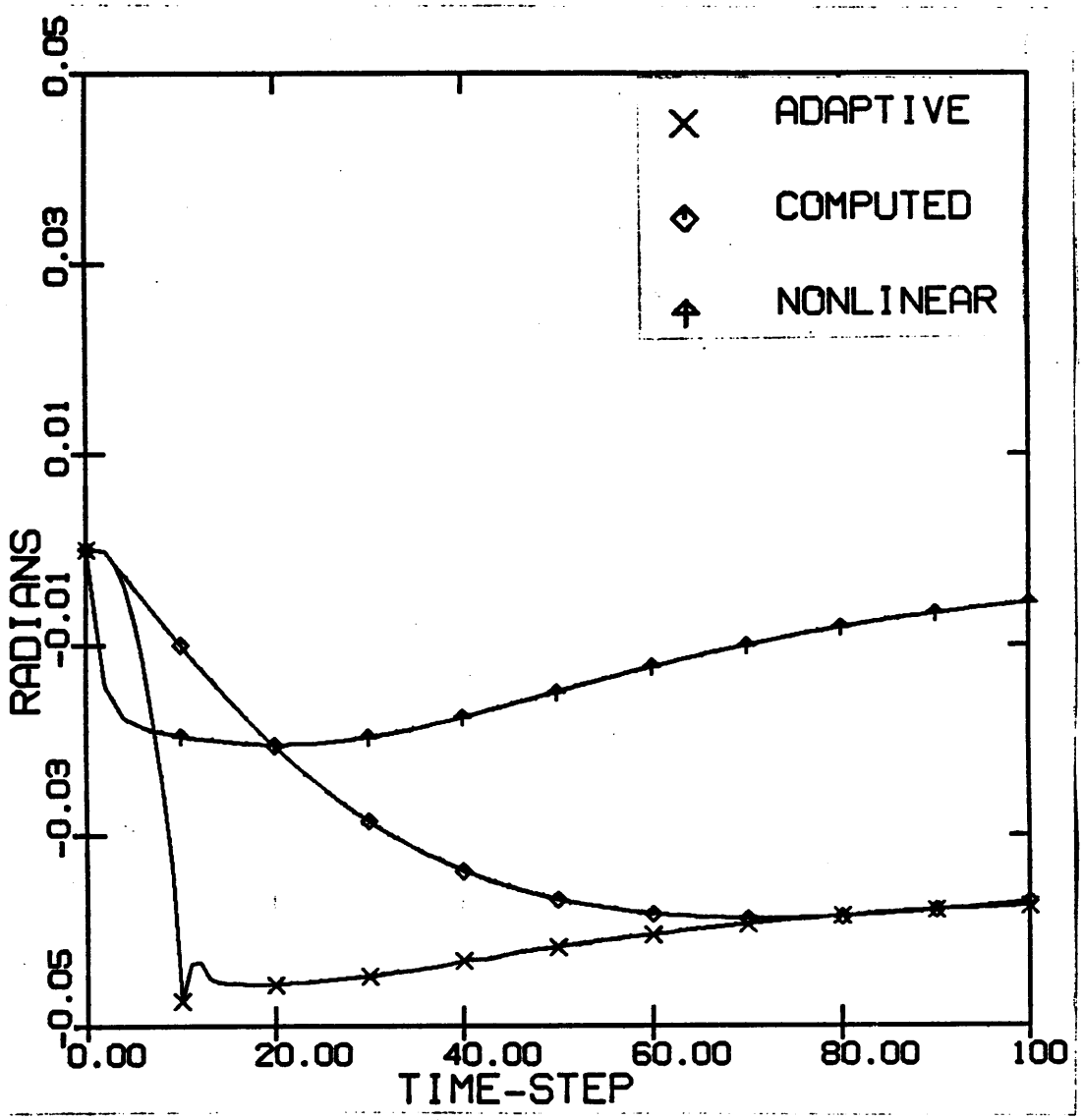


Figure 30. Tracking Errors.: Joint 2.

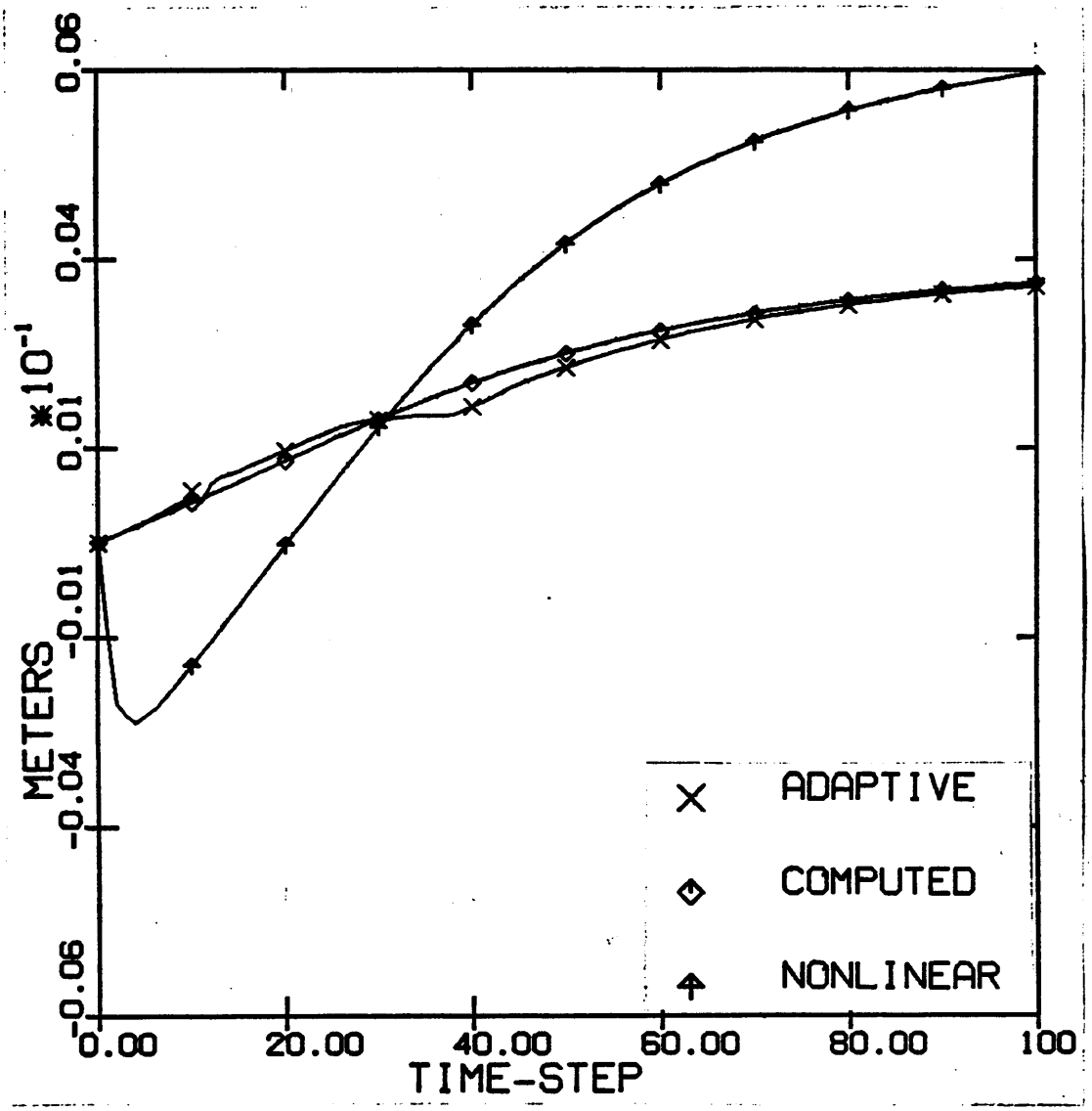


Figure 31. Tracking Errors.: Joint 3.

8.0 CHAPTER 8-CONCLUSION

The three control algorithms that were studied in this theses have very different computational requirements. It is important to know the amount of these computations neccesary to evaluate the control input in each step of the motion task in order to determine the suitability of a particular algorithm for a given manipulator application as well as a measure of comparison between the different algorithms.

8.1 COMPUTATIONAL ANALYSIS

The number of arithmetic operations are computed for an n-degree of freedom manipulator and the corresponding number for n=3 is also listed alongside each expression. In the case of the non-linear control algorithm,due to its very nature the number of arithmetic operations does not directly depend on the number of links alone but rather on the specific geometry of the manipulator. This is because the control laws are derived directly from the equations of motion of the manipulator,given explicitly in algebraic form. Therefore the number of operations cannot be given in the form of a general expression- the numbers indicated in this case represent the number of operations for the three-link arm studied in this thesis. M indicates the number of

multiplications/divisions and A represents the number of additions/subtractions.

8.1.1 CONTROL BASED ON LINEARISED MODEL

The computations for this algorithm can be split up into five parts as follows

- Computing nominal torques

$$M = (97n^2 + 185n)/2, \quad 714$$

$$A = (61n^2 + 105n)/2, \quad 432$$

- Computing linearised model

$$M = (56n^4 + 1005n^3 + 2749n^2 + 360n)/6, \quad 9582$$

$$A = (28n^4 + 393n^3 + 956n^2 + 156n)/3, \quad 7317$$

- Computing discrete time model

$$M = 28n^2 + 8n + 3 \ , \ 831$$

$$A = 28n^3 - 6n^2 + 4n \ , \ 714$$

- Computing optimum control input

$$M = (n^4 + 21n^3 + 20n^2)/3 \ , \ 276$$

$$A = (2n^4 + 39n^3 + 25n^2 - 24n)/6, \ 228$$

- Add nominal and perturbed inputs

$$M = 0$$

$$A = n \ , \ 3$$

The total number of operations to be carried out at each time-step is given by

$$M = (58n^4 + 1215n^3 + 3128n^2 + 915n)/6 \ , \ 11400$$

$$A = (58n^4 + 993n^3 + 2084n^2 + 633n)/6, \quad 8694$$

8.1.2 ADAPTIVE CONTROL

For this algorithm the computations can be broken up into four parts as follows

- Computing nominal torques.

$$M = (97n^2 + 185n)/2, \quad 714$$

$$A = (61n^2 + 105n)/2, \quad 432$$

- Estimation of linearised model parameters

$$M = (39n^2 + 25n)/2, \quad 213$$

$$A = (51n^2 + 3n)/2, \quad 234$$

- Computing optimum control input

$$M = (n^4 + 21n^3 + 20n^2)/3 \ , \ 276$$

$$A = (2n^4 + 39n^3 + 25n^2 - 24n)/6, \quad 228$$

- Add nominal and perturbed inputs

$$M = 0$$

$$A = \quad n \ , \ 3$$

The total number of operations to be carried out at each time-step is given by

$$M = (n^4 + 21n^3 + 224n^2 + 315n)/3 \ , \ 1203$$

$$A = (2n^4 + 39n^3 + 361n^2 + 306n)/6, \quad 897$$

8.1.3 NON-LINEAR CONTROL ALGORITHM

The number of operations in this case can be computed easily from the control laws given in Chapter6. Note that in this case the control laws are specified only for the last two links because the first link was assumed to be at rest to keep the equations of motion tractable. Therefore,the number of operations is slightly under-estimated.

$$M = 23$$

$$A = 16$$

As can be seen from the numbers given for the case n=3, there is order of magnitude difference bettween the three algorithms. The non-linear control algorithm is seen to have the least amount of computations once the control laws are formulated.

8.2 EVALUATION OF RELATIVE PERFORMANCE

The simulation results of the last chapter give a fairly detailed pictureof the performance of the three control algorithms. Path tracking is an important consideration for manipulator operation. The plots of the tracking errors for

the three algorithms show that the best overall tracking is achieved by the linearised model algorithm and the worst tracking is obtained by the non-linear algorithm. There is not much difference in the performance of the adaptive and the linearised model algorithms. For the case of joint 1 the non-linear algorithm shows the largest tracking error. This is not due to the inadequacy of the algorithm. The control torque was assumed to be zero for link 1 - therefore there was no correction torque to keep the link from moving due to the coupling forces between the links. However, in the case of joint2 the non-linear algorithm gives the least tracking error.

The input torques are quite smoothly varying for the linearised model algorithm and the non-linear one. The perturbations from the nominal torque are rather large for the adaptive algorithm,especially in the beginning. This is a serious drawback for the adaptive algorithm because in reality,since the actuator motors cannot exceed their rated torques,we can only apply the maximum rated torque when the required correction torque may be much larger in magnitude. This can lead to further increase in the tracking error making matters worse. Therefore the relatively good performance of the adaptive algorithm must be viewed with some degree of skepticism. It is quite possible that the algorithm may not work in an actual implementation due this limitation. As the plots of the parameter estimates indicate it is possible to

achieve good overall tracking even if the parameter estimates may not be very accurate.

It is clear from the computational analysis in the beginning of this chapter that there is a considerable difference between the the three algorithms. The non-linear algorithm appears to be deceptively simple compared to the other two. However,as was pointed out earlier,it is very difficult to implement this algorithm for all but the very simplest types of manipulators with less than three or four links. Since most industrial robots have at least five to six degrees of freedom,this algorithm is not very useful from a practical point of view. The adaptive and linearised model algorithms afford a better comparison since their performance is very similar. Both have $o(n^4)$ computational requirement but the linearised model algorithm has about 10 times more operations for the three-link case. The total number of computations is prohibitively large for the linearised model algorithm to be of any practical use. Therefore,from an implementation point of view,the adaptive algorithm remains the most feasible. If the large perturbations in the input torque can be reduced by a better estimation algorithm than the RLS algorithm used in this case, it is fair to claim that this is the best algorithm from a practical viewpoint.

8.3 CONCLUSION

It was the objective of this thesis to develop a complete dynamics-simulator for the general n-degree of freedom manipulator. That objective was achieved and it was possible to study in detail the performance of some control algorithms. The three control algorithms studied were compared based on simulation results and a computational analysis. The dynamics simulator was found to be a good tool for evaluating control algorithms for the non-linear dynamics of manipulators. It can also be used as an effective tool for the design of manipulators.

BIBLIOGRAPHY

[1]Vukobratovic M and Stokic D;"Control of manipulating robots Theory and applications",Berlin NewYork Springer Verlag,1982.

[2]Paul R P;"Robot Manipulators-Mathematics,Programming and Control", Cambridge Mass MIT Press,1982.

[3]Vukobratovic M and Potkonjak V;"Dynamics of manipulating robots Theory and applications",Berlin NewYork Springer Verlag,1982.

[4]Freund E;"Fast Non-linear control with arbitrary pole-placement for industrial robots and manipulators",The Int. J. of Robotics Research, Vol. 1,no. 1,Spring 1982.

[5]Vukobratovic M and Stokic D;"One engineering concept of dynamic control of manipulators",Trans. ASME,J of Dyn. Meas.Contr, 103(2):108-118,1981b.

[6]Lee C S G and Lee B H;"Resolved motion adaptive control for mechanical manipulators",Trans. ASME,J. of Dyn. Meas. Contr, Vol. 106,No. 2,June 1984.

[7]Ljung L and Soderstrom T;"Theory and practice of recursive identification",Cambridge Mass MIT Press,1983.

APPENDIX A. KINEMATICS

A.1 NOTATION

- $\underline{x}_i, \underline{y}_i, \underline{z}_i$ = Unit-vectors along the x,y,z axes of the i^{th} link fixed co-ordinate system, i=0,1,....n. i=0 refers to the base co-ordinate system itself.

- n= Total number of links (joints).

- $\underline{r}_{ij}$ = Position vector from the center of joint j to the COG of link i. The origin of the link fixed co-ordinate system is assumed to be located at the COG of the link.

- ${}^{i}\underline{r}_{ij}$ = Same vector as above represented in the co-ordinate system of link i.

- $\underline{e}_i$ = Unit-vector along the axis of joint i.

- ${}^{i}\underline{e}_{ii}$ = Unit-vector along axis of joint i, expressed in the co-ordinate system of link i.

- ${}^{i}\underline{e}_{i,i+1}$ = Unit-vector along axis of joint i+1, expressed in the co-ordinate system of link i.

A.2 DEFINITION OF JOINT POSITIONS

A.2.1 JOINT POSITIONS FOR ROTATIONAL JOINTS

A rotational joint is represented in Figure 32 on page 94. Joint i position is defined as the angle between the projection of the vectors $\underline{r}_{ii}, -\underline{r}_{i,i-1}$ onto the plane perpendicular to the axis vector $\underline{e}_i$. When one of the vectors $\underline{r}_{i-1,i}$ or $\underline{r}_{ii}$ are collinear with the axis vector $\underline{e}_i$, then measuring the joint position by this method fails. Accordingly we classify joints into the following four categories.

- Type 1 ... Both position vectors are in a plane perpendicular to the axis vector $\underline{e}_i$.
- Type 2 ... $\underline{r}_{i-1,i}$ is collinear with the axis vector $\underline{e}_i$.
- Type 3 ... $\underline{r}_{ii}$ is collinear with the axis vector $\underline{e}_i$.
- Type 4 ... Both position vectors are collinear with the axis vector $\underline{e}_i$.

Figure 32 on page 94 represents a type 1 joint. Figure 33 on page 95 show types 2,3 and 4. In the case of types 2,3 and 4 it necessary to define additional vectors for the purpose of measuring the joint angle. As far as the measurement of

the joint angle is concerned the lengths of the vectors $\underline{r}^*$ are not important but their directions must be chosen such that $-\underline{r}^*_{i-1,i}$ and $\underline{r}^*_{ii}$ coincide when the joint angle is zero.

A.2.2 JOINT POSITION FOR A LINEAR JOINT

Figure 34 on page 96 represents the joint position measurement for a linear joint. In this case the length of the vector $\underline{r}_{ii}$ depends on the joint position unlike a rotational joint where the length is constant.

A.3 ASSEMBLY OF LINK PAIRS

Suppose a co-ordinate system rigidly fixed to a link at some point is chosen and the vectors ${}^i\underline{r}_{ii}$, ${}^i\underline{r}_{i,i+1}$, ${}^i\underline{e}_{ii}$, ${}^i\underline{e}_{i,i+1}$ defining the position vectors from the center of the joints to the origin and the axes of rotation or translation of the joints at both ends respectively are known with the link co-ordinate system as shown in Figure 35 on page 97. The objective of the link assembly is to determine the orientation of the body fixed co-ordinate system with respect to the base co-ordinate system. That is we wish to determine direction vectors $\underline{x}_i$, $\underline{y}_i$ and $\underline{z}_i$ in the base co-ordinate system after assembling the link with the previous link so that the other vectors on the current link may be converted to base co-ordinates. Let

$$A_i = [\ \underline{x}_i \quad \underline{y}_i \quad \underline{z}_i\ ] \tag{A.60}$$

Then

$$\underline{a} = A_i\, {}^i\underline{a} \tag{A.61}$$

where

- ${}^i\underline{a}$ = Any vector represented in link i co-ordinate system.

- $\underline{a}$ = The same vector represented in base co-ordinates.

In the first step of the assembly the joint positions are all assumed to be zero. The position of the origin and the orientation of the link fixed co-ordinate system may be chosen arbitrarily. To simplify the dynamics we chose the origin at the COG and the orientation such that the co-ordinate axes coincide with the principal axes of inertia of the link. Suppose link i-1 is already assembelled,that is all the vectors defined on it are available in base co-ordinates. When link i is assembled ${}^i\underline{e}_{ii}$ must coincide with $\underline{e}_i$ and $-\underline{r}_{i-1,i}$ must coincide with ${}^i\underline{r}_{ii}$ (since q_i ,the joint variable is zero during link assembly). Define

$$\underline{r} = -\frac{\underline{e}_i \times (\underline{r}_{i-1,i} \times \underline{e}_i)}{|\underline{e}_i \times (\underline{r}_{i-1,i} \times \underline{e}_i)|} \quad , \quad {}^i\underline{r} = \frac{{}^i\underline{e}_{ii} \times ({}^i\underline{r}_{ii} \times {}^i\underline{e}_{ii})}{|{}^i\underline{e}_{ii} \times ({}^i\underline{r}_{ii} \times {}^i\underline{e}_{ii})|} \qquad (A.62)$$

noting that all quantities on the RHS are vectors and are not under-scored for clarity. $\underline{r}$ is a unit-vector perpendicular to $\underline{e}_i$ and opposite in direction to $\underline{r}_{i-1,i}$. When $\underline{r}_{i-1,i}$ is collinear with $\underline{e}_i$ then we use the vector $\underline{r}^*_{i-1,i}$ instead. Also define

$$\underline{a} = \frac{\underline{e} \times \underline{r}}{|\underline{e} \times \underline{r}|} \quad , \quad {}^i\underline{a} = \frac{{}^i\underline{e}_{ii} \times {}^i\underline{r}_{ii}}{|{}^i\underline{e}_{ii} \times {}^i\underline{r}_{ii}|} \qquad (A.63)$$

After assembly we have

$$\underline{e}_i = A_i\, {}^i\underline{e}_{ii} \quad , \quad \underline{r} = A_i\, {}^i\underline{r} \quad , \quad \underline{a} = A_i\, {}^i\underline{a} \qquad (A.64)$$

or

$$[\underline{e}_i \ \underline{r} \ \underline{a}] = A_i[{}^i\underline{e}_{ii} \ {}^i\underline{r} \ {}^i\underline{a}] = A_i B \qquad (A.65)$$

Thus

$$A_i = B^{-1}[\underline{e}_i \ \underline{r} \ \underline{a}] = B^T[\underline{e}_i \ \underline{r} \ \underline{a}] \qquad (A.66)$$

since B is an ortho-normal matrix. Then

$$\underline{e}_{i+1} = A_i \, {}^{i}\underline{e}_{i,i+1} \,, \qquad \underline{r}_{i,i+1} = A_i \, {}^{i}\underline{r}_{i,i+1} \tag{A.67}$$

Knowing $A_i, \underline{e}_{i+1}$ and $\underline{r}_{i,i+1}$ we proceed to assemble link i+1 in the same manner. This recursive process is summarised in the flow-chart given in Figure 37 on page 99

During the process of link assembly we assumed all the joint variables to be zero. To determine any vector in base co-ordinates we only need to multiply by the corresponding link transformation matrix A_i, the same vector expressed in the link fixed co-ordinate system. Thus we only need to keep track of the link transformation matrix A_i to determine the position of any point on the link.

The next step in the assembly process is to determine the A_i matrices for an arbitrary value of the joint position vector. Each joint is rotated or translated about its axis successively by the amount specified in the joint position vector, starting from the first joint. In the case of a rotational joint i, all vectors beyond joint i will be rotated by the angle q_i about the axis vector $\underline{e}_i$. For such a rotation of a vector about a fixed axis, the new vector (after rotation) may be determined from the simple formula

$$\underline{r}' = \underline{r} \cos(q_i) + [1 - \cos(q_i)](\underline{e}_i \cdot \underline{r}) + (\underline{e}_i \times \underline{r}) \sin(q_i) \tag{A.68}$$

where

- $\underline{r}$ = the vector before rotation
- $\underline{r}'$= the vector after rotation
- $\underline{e}_i$= the axis vector for rotation
- q_i = amount of rotation.

The process explained above may be conveniently put into the flow-chart form as shown in Figure 38 on page 100. Due to the recursive nature of all the computations they may be easily put into a computer program (Appendix E) so that the kinematics problem (forward) for an arbitrary manipulator can be solved.

A.4 INVERSE KINEMATICS

As was mentioned in Chapter 2 it is almost impossible to obtain a general algorithm,suitable for programming,to solve the inverse kinematics problem for an arbitrary manipulator. The reason for this being simply the fact that we have to deal with non-linear algebraic equations for which there is no systematic method of solution applicable to a wide variety of equations. However,given the geometric details of a manipulator one can easily derive the inverse kinematic solution and express it in the form of formulae for easy

evaluation. The manipulator model with the relevant geometric details is shown in Figure 39 on page 101, including the location of the base co-ordinate system,which may be chosen arbitrarily. The x,y,z co-ordinates of the point E,representing the center of the end-effector may be given in terms of the joint variables q_1, q_2 and q_3 as follows.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = d \begin{bmatrix} -\sin(q_1) \\ \cos(q_1) \\ 0 \end{bmatrix} + q_3 \begin{bmatrix} \cos(q_1)\sin(q_2) \\ \sin(q_1)\sin(q_2) \\ \cos(q_2) \end{bmatrix} \quad \text{(A.69)}$$

Since the trajectory chosen has $q_1 = 0$ at all points

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} q_3\sin(q_2) \\ d \\ q_3\cos(q_2) \end{bmatrix} \quad \text{(A.70)}$$

$$q_1 = 0 \quad , \quad q_3 = \sqrt{x^2 + z^2} \quad , \quad q_2 = \text{Atan}(x,z) \quad \text{(A.71)}$$

If the origin of the base co-ordinate system is located at some point other than the center of joint 1,then let

$$\underline{r} = [\, r_x \;\; r_y \;\; r_z \,]^T$$

be the vector from the center of joint 1 to the origin of the base co-ordinate sytem. Then the inverse kinematic equations are given by

$$q_1 = 0$$

$$q_2 = \text{Atan}[\, (x+r_x) \;,\; (z+r_z) \,] \qquad \text{(A.72)}$$

$$q_3 = \sqrt{(x + r_x)^2 + (z + r_z)^2}$$

The relationship between cartesian and joint velocities and accelerations may be easily obtained by direct differentiation of Equation A.70 and solving for the joint velocities and accelerations in terms of the cartesian velocities and accelerations.

$$q_1' = 0$$

$$q_2' = [x'\cos(q_2) - z'\sin(q_2)]/q_3$$

$$q_3' = x'\sin(q_2) + z'\cos(q_2)$$

Since we have assumed uniform motion between the end-points, x"=y"=z"=0 and the expressions for accelerations have the form

$$q_1'' = 0$$

$$q_2'' = -(2q_2'q_3')/q_3$$

$$q_3'' = q_3'(q_2')^2$$

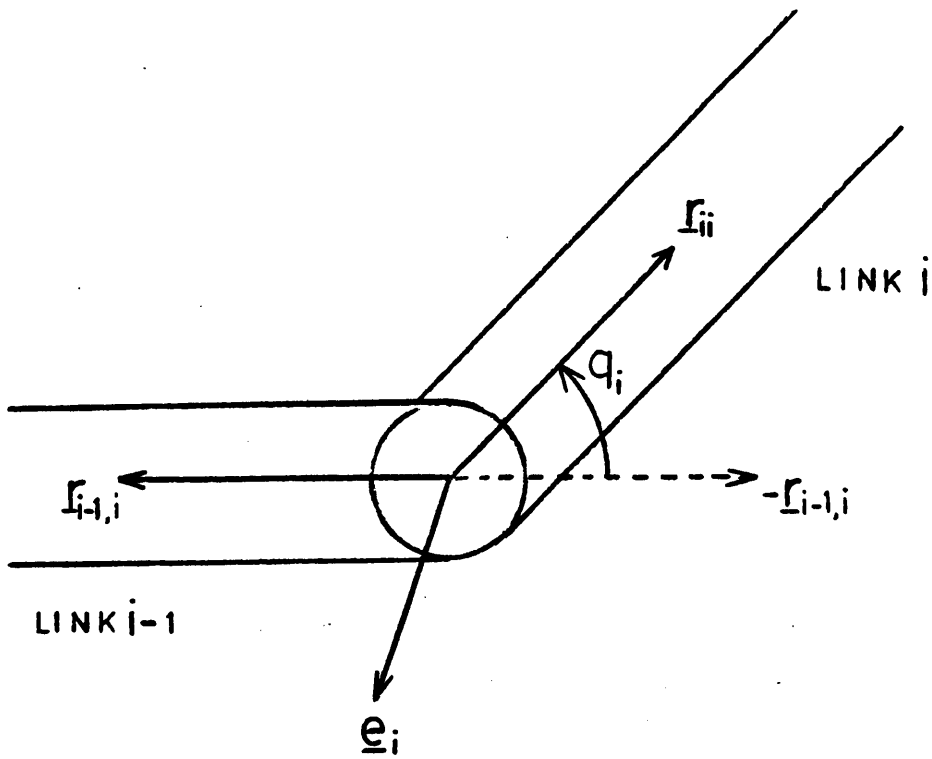


Figure 32. Rotational Joint: Definition of joint position .

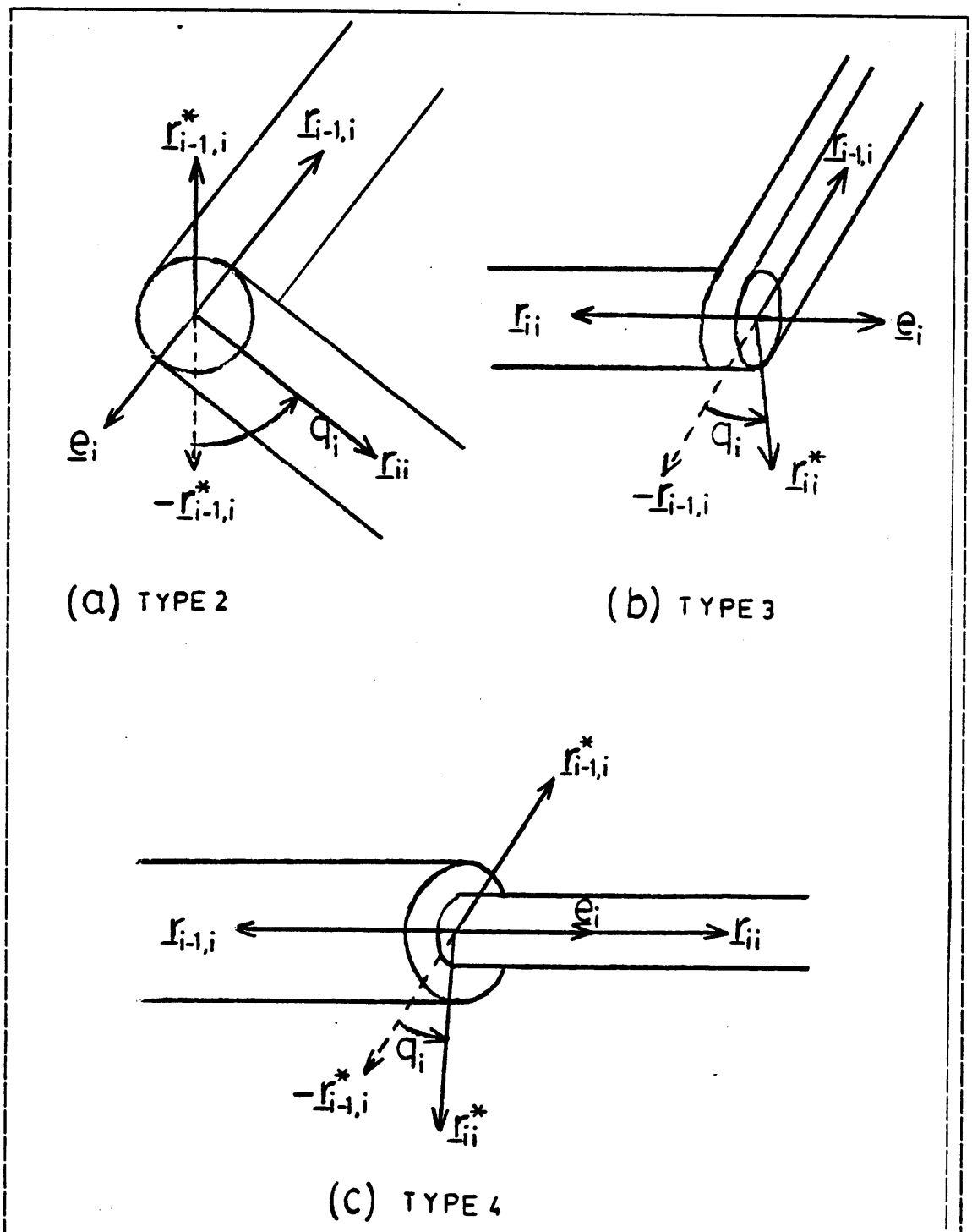
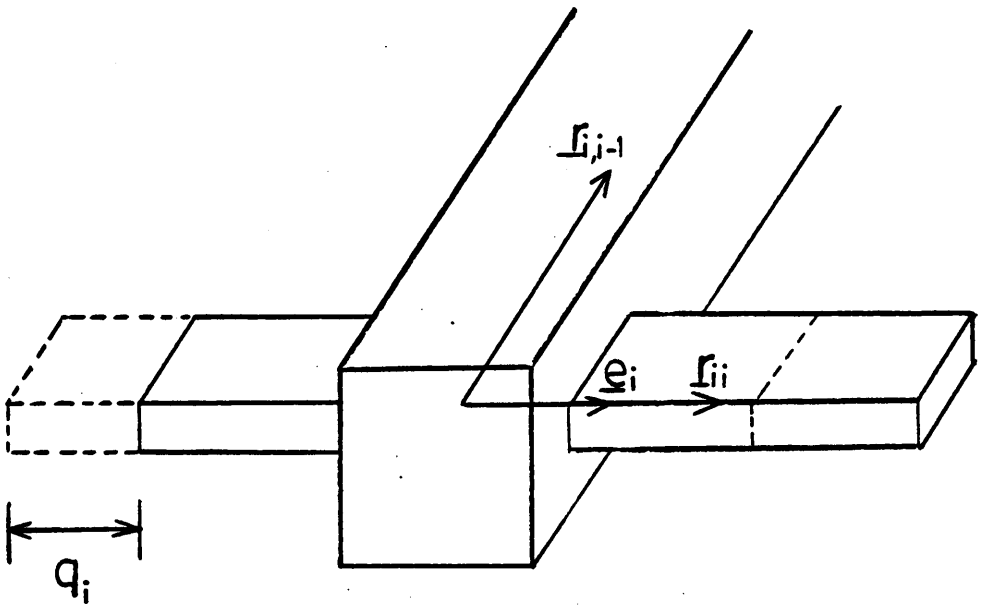


Figure 33. Rotational Joints: Classification of joint types.



$$\underline{r}_{ii} = \underline{r}_{ii}^{\circ} + q_i \underline{e}_i$$

Figure 34. Linear Joints: Definition of joint position

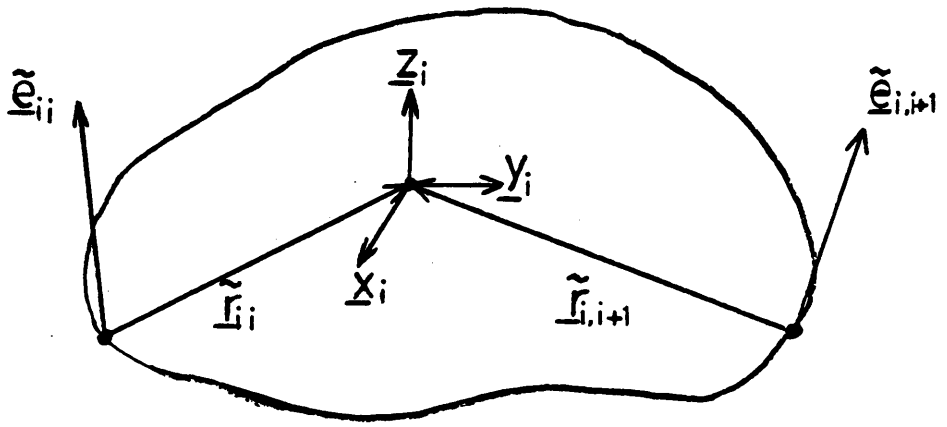


Figure 35. Link i: Body fixed co-ordinate system and vectors

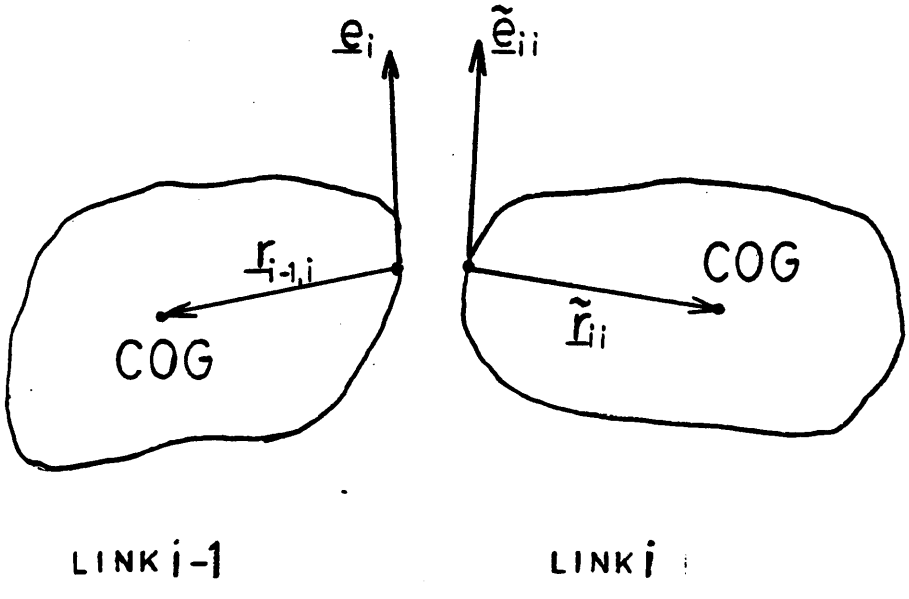


Figure 36. Link i: Assembly

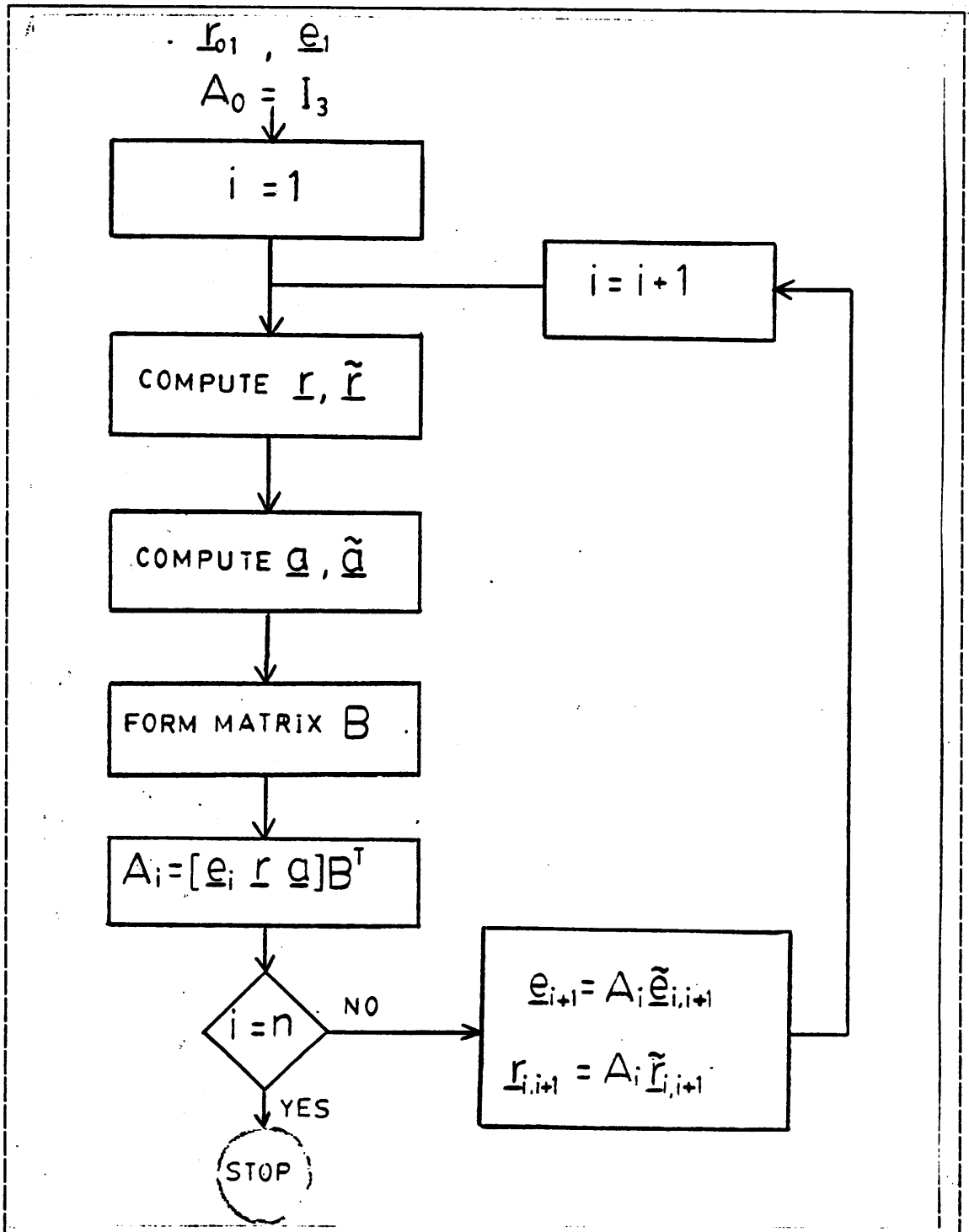


Figure 37. Link assembly: Computing transformation matrices

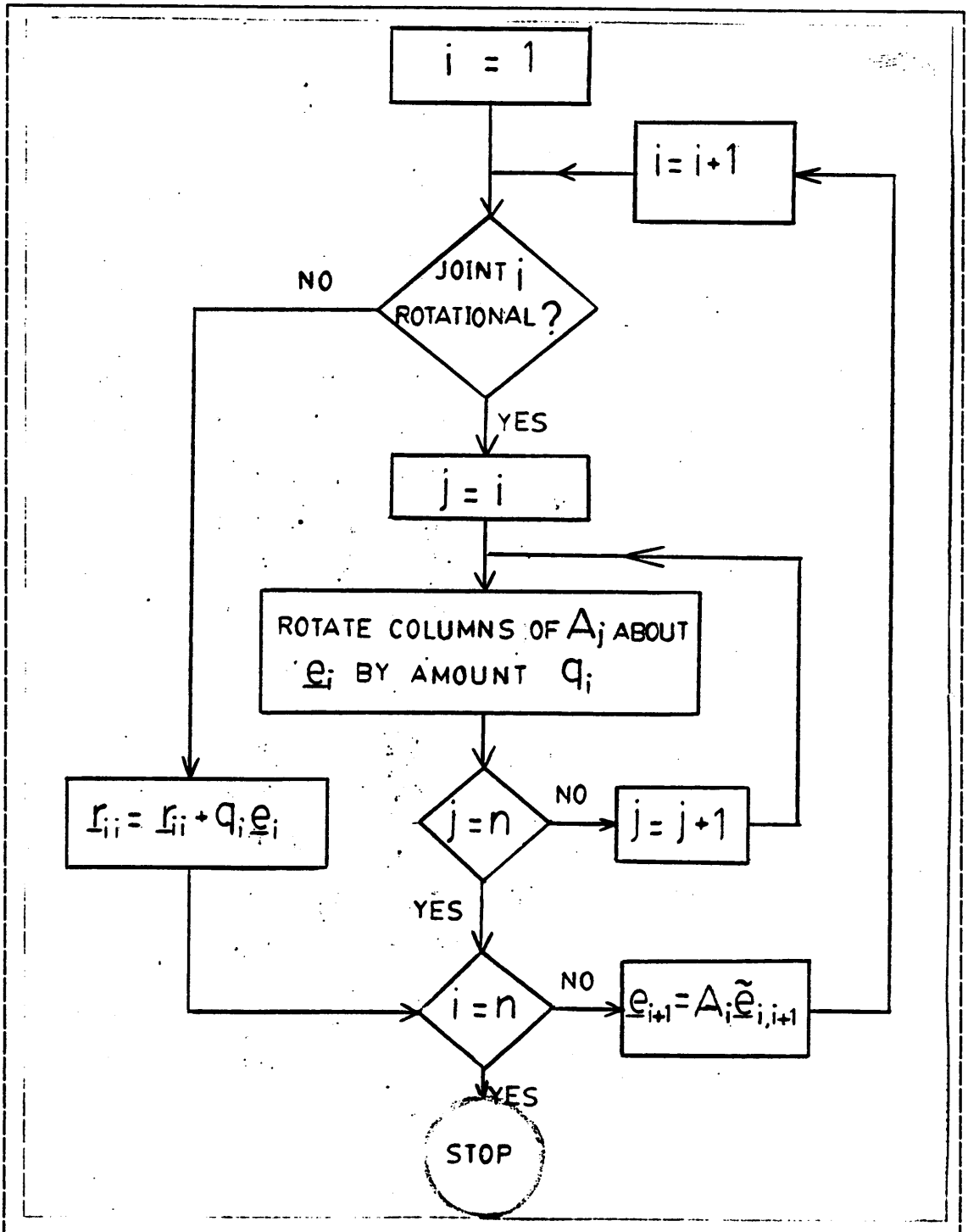


Figure 38. Link assembly: Rotation of transformation matrices

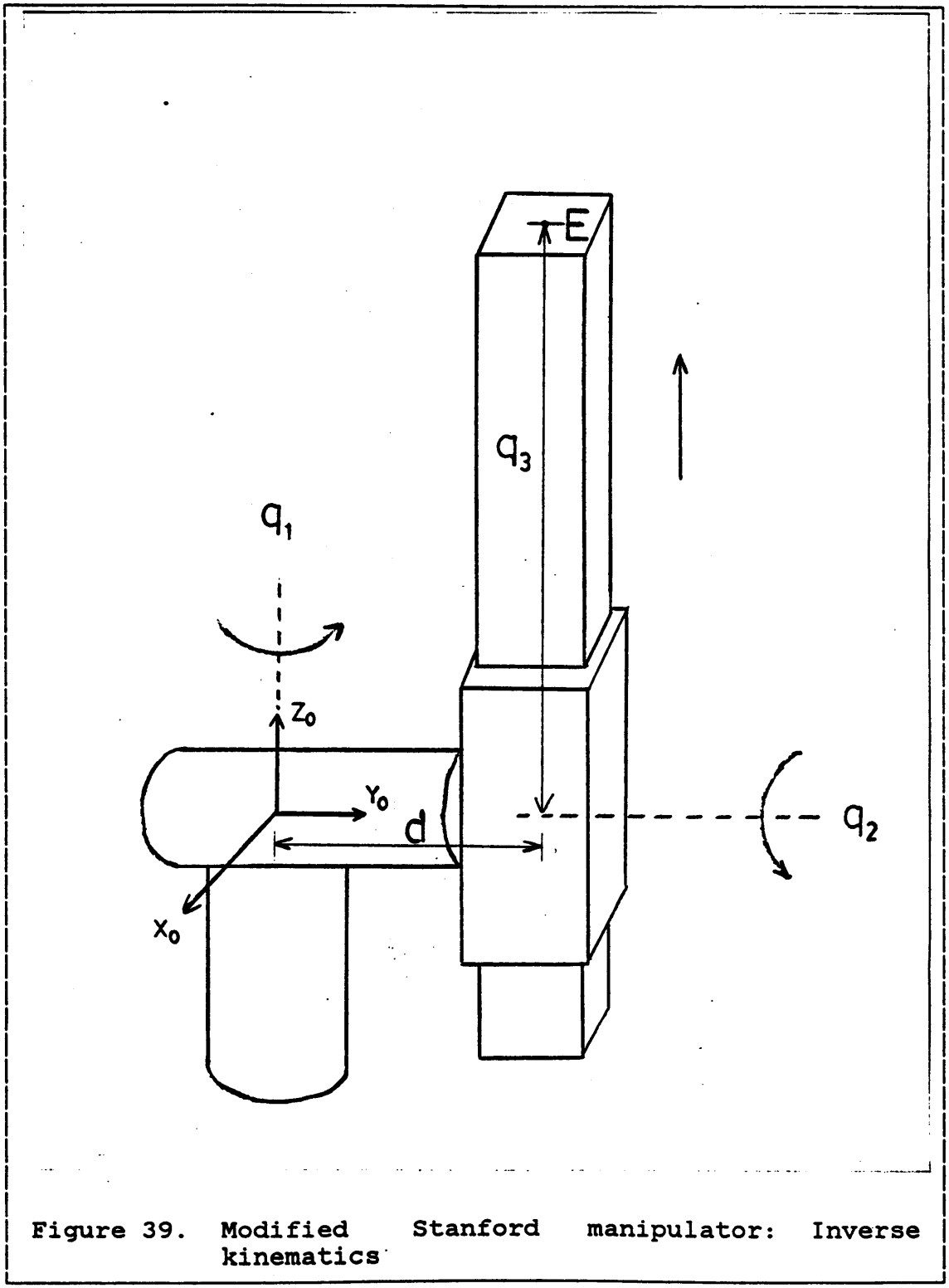


Figure 39. Modified Stanford manipulator: Inverse kinematics

APPENDIX B. DYNAMICS

B.1 RECURSIVE RELATIONSHIPS FOR VELOCITY AND ACCELERATION

Consider the rotational joint shown in Figure 40 on page 116. Let

- $\underline{v}_i, \underline{a}_i$ represent the linear velocity and acceleration of link i COG.
- $\underline{\omega}_i, \underline{\alpha}_i$ represent the angular velocity and acceleration of link i.
- q_i, q_i', q_i'' represent joint i position, velocity and acceleration respectively.

Then

$$\underline{\omega}_i = \underline{\omega}_{i-1} + q_i' \, \underline{e}_i \tag{B.73}$$

$$\underline{v}_i = \underline{v}_{i-1} - \underline{\omega}_{i-1} \times \underline{r}_{i-1,i} + \underline{\omega}_i \times \underline{r}_{ii} \tag{B.74}$$

$$\underline{\alpha}_i = \underline{\alpha}_{i-1} + q_i'' \underline{e}_i + q_i' (\underline{\omega}_{i-1} \times \underline{e}_i) \tag{B.75}$$

$$\underline{a}_i = \underline{a}_{i-1} - \underline{\alpha}_{i-1}\times\underline{r}_{i-1,i} - \underline{\omega}_{i-1}\times(\underline{\omega}_{i-1}\times\underline{r}_{i-1,i}) + \underline{\alpha}_i\times\underline{r}_{ii} + \underline{\omega}_i\times(\underline{\omega}_i\times\underline{r}_{ii}) \quad (B.76)$$

Similarly for the linear joint shown in Figure 41 on page 117 we can easily derive the recursive equations

$$\underline{\omega}_i = \underline{\omega}_{i-1} \quad (B.77)$$

$$\underline{\alpha}_i = \underline{\alpha}_{i-1} \quad (B.78)$$

$$\underline{v}_i = \underline{v}_{i-1} - \underline{\omega}_{i-1}\times\underline{r}_{i-1,i} + \underline{\omega}_i\times\underline{r}_{ii} + q_i'\underline{e}_i \quad (B.79)$$

$$\underline{a}_i = \underline{a}_{i-1} - \underline{\alpha}_{i-1}\times\underline{r}_{i-1,i} - \underline{\omega}_{i-1}\times(\underline{\omega}_{i-1}\times\underline{r}_{i-1,i}) + \underline{\alpha}_i\times\underline{r}_{ii} + \underline{\omega}_i\times(\underline{\omega}_i\times\underline{r}_{ii}) + 2\underline{\omega}_{i-1}\times q_i'\underline{e}_i + q_i''\underline{e}_i \quad (B.80)$$

Thus starting with i=0 (base) we can compute the linear and angular accelerations of the COG of each link in terms of the joint positions, velocities and accelerations.

Writing the equations for acceleration in a different form

$$\underline{\omega}_i = \sum_{j=1}^{i} \underline{\alpha}_{ij} q_j'' + \underline{\theta}_i + \underline{\alpha}_0 \quad (B.81)$$

$$\underline{a}_i = \sum_{j=1}^{i} \underline{\beta}_{ij} q_j'' + \underline{\eta}_i + \underline{a}_0 \qquad \text{(B.82)}$$

Comparing Equation B.81 with Equation B.75 on page 102 and Equation B.78 the following relationships can be derived.

For joint i rotational

$$\underline{\alpha}_{ij} = \underline{\alpha}_{i-1,j} \quad \text{for } j \neq i \qquad \text{(B.83)}$$

$$\underline{\alpha}_{ii} = \underline{e}_i \qquad \text{(B.84)}$$

$$\underline{\theta}_i = \underline{\theta}_{i-1} + q_i'(\underline{\omega}_{i-1} \times \underline{e}_i) \qquad \text{(B.85)}$$

For joint i linear

$$\underline{\alpha}_{ij} = \underline{\alpha}_{i-1,j} \quad \text{for } j \neq i \qquad \text{(B.86)}$$

$$\underline{\alpha}_{ii} = \underline{0} \qquad \text{(B.87)}$$

$$\underline{\theta}_i = \underline{\theta}_{i-1} \qquad \text{(B.88)}$$

Similarly comparing Equation B.82 with Equation B.76 on page 102 and Equation B.80 we obtain

For joint i rotational

$$\underline{\beta}_{ij} = \underline{\beta}_{i-1,j} + \underline{\alpha}_{i-1,j} \times (\underline{r}_{ii} - \underline{r}_{i-1,i}) \qquad \text{for } j \neq i \tag{B.89}$$

$$\underline{\beta}_{ii} = \underline{e}_i \tag{B.90}$$

$$\underline{\eta}_i = \underline{\eta}_{i-1} + \underline{\theta}_{i-1} \times (\underline{r}_{ii} - \underline{r}_{i-1,i}) + q_i' (\underline{\omega}_{i-1} \times \underline{e}_i) \times \underline{r}_{i-1,i} + \underline{\omega}_i \times (\underline{\omega}_i \times \underline{r}_{ii}) - \underline{\omega}_{i-1} \times (\underline{\omega}_{i-1} \times \underline{r}_{i-1,i}) \tag{B.91}$$

For joint i linear

$$\underline{\beta}_{ij} = \underline{\beta}_{i-1,j} + \underline{\alpha}_{i-1,j} \times (\underline{r}_{ii} - \underline{r}_{i-1,i}) \qquad \text{for } j \neq i \tag{B.92}$$

$$\underline{\beta}_{ii} = \underline{e}_i \tag{B.93}$$

$$\underline{\eta}_i = \underline{\eta}_{i-1} + \underline{\theta}_{i-1} \times (\underline{r}_{ii} - \underline{r}_{i-1,i}) + 2q_i' (\underline{\omega}_{i-1} \times \underline{e}_i) + \underline{\omega}_i \times (\underline{\omega}_i \times \underline{r}_{ii}) - \underline{\omega}_{i-1} \times (\underline{\omega}_{i-1} \times \underline{r}_{i-1,i}) \tag{B.94}$$

Let

- $\underline{F}_i$ = inertial force on link i

- $\underline{M}_i$ = inertial moment on link i

Then

$$\underline{F}_i = - m_i \underline{a}_i$$

$$= - m_i \left[\sum_{j=1}^{i} \underline{\beta}_{ij} q_j'' + \underline{\eta}_i + \underline{a}_0 \right] \qquad (B.95)$$

Or

$$\underline{F}_i = \sum_{j=1}^{i} \underline{a}_{ij} q_j'' + \underline{a}_i^0 - m_i \underline{a}_0 \qquad (B.96)$$

where

$$\underline{a}_{ij} = -m_i \underline{\beta}_{ij} \quad \text{and} \quad \underline{a}_i^0 = -m_i \underline{\eta}_i$$

Let J_{i1}, J_{i2}, J_{i3} be the moments of inertia of link i about its principal axes. [As mentioned in the kinematics section, the link co-ordinate system is chosen such that the principal axes coincide with the directions of x_i, y_i and z_i.] Let

$$q_{ik}^j = - A_i \,]_{jk}$$

Then

$$\underline{x}_i = \begin{bmatrix} q_{i1}^1 \\ q_{i1}^2 \\ q_{i1}^3 \end{bmatrix}, \quad \underline{y}_i = \begin{bmatrix} q_{i2}^1 \\ q_{i2}^2 \\ q_{i2}^3 \end{bmatrix}, \quad \underline{z}_i = \begin{bmatrix} q_{i3}^1 \\ q_{i3}^2 \\ q_{i3}^3 \end{bmatrix} \tag{B.97}$$

Now

$$\underline{M}_i = -d/dt(I\underline{\omega}) \tag{B.98}$$

where I is the 3×3 inertia tensor expressed in base coordinates.

$$I = A_i^T J_i A_i, \qquad J_i = \text{diag}(J_{i1}, J_{i2}, J_{i3}) \tag{B.99}$$

Taking the vector derivative in Equation B.98 on page 107 and using Equation B.99 on page 107 we can write

$$\underline{M}_i = T_i \underline{\alpha}_i + \underline{\lambda}_i \tag{B.100}$$

where

$$\underline{\lambda}_i = A_i \begin{bmatrix} (J_{i2}-J_{i3})(\underline{\omega}_i \cdot \underline{q}_{i2})(\underline{\omega}_i \cdot \underline{q}_{i3}) \\ (J_{i3}-J_{i1})(\underline{\omega}_i \cdot \underline{q}_{i3})(\underline{\omega}_i \cdot \underline{q}_{i1}) \\ (J_{i1}-J_{i2})(\underline{\omega}_i \cdot \underline{q}_{i1})(\underline{\omega}_i \cdot \underline{q}_{i3}) \end{bmatrix} \qquad \text{(B.101)}$$

and

$$[\ T_i\ ]_{jk} = \sum_{n=1}^{3} q_{in}^{j}\ J_{in}\ q_{in}^{k}$$

Substituting for $\underline{\alpha}_i$ in Equation B.100

$$\underline{M}_i = -T_i[\ \sum_{j=1}^{i} \underline{\alpha}_{ij} q_j{}''\ +\ \underline{\theta}_i\ +\ \underline{\alpha}_0\ ]\ +\ \underline{\lambda}_i \qquad \text{(B.102)}$$

Or

$$\underline{M}_i = \sum_{j=1}^{i} \underline{b}_{ij} q_j{}''\ +\ \underline{b}_i^0\ -\ T_i \underline{\alpha}_0 \qquad \text{(B.103)}$$

where

$$\underline{b}_{ij} = -T_i \underline{\alpha}_{ij} \quad , \quad \underline{b}_i^0 = -T_i \underline{\theta}_i + \underline{\lambda}_i$$

B.2 OBTAINING THE DIFFERENTIAL EQUATIONS OF MOTION

Suppose that the kinematic chain is broken at the i^{th} joint and the equilibrium of the chain from link i to link n is considered. At the broken joint i the reaction-force, $\underline{F}_{iR}$ and the reaction moment, $\underline{M}_{iR}$ replace the effect of links 1 through i-1 as shown in Figure 42 on page 118. Now let $\underline{\Gamma}_i$ and $\underline{L}_i$ represent the net force and moment respectively acting on the i^{th} link. The net force and moment are computed as the sum of the inertial and external forces acting on each link, as shown below.

$$\underline{\Gamma}_j = \underline{F}_j + \underline{F}_{jE} \quad , \quad \underline{L}_j = \underline{M}_j + \underline{M}_{jE} \tag{B.104}$$

The subscript E stands for external force or moment. Applying the dynamic equilibrium condition

$$\underline{F}_{iR} = - \sum_{j=i}^{n} \underline{\Gamma}_j \tag{B.105}$$

$$\underline{M}_{iR} = - \sum_{j=i}^{n} (\underline{L}_j + \underline{r}_{ji} \times \underline{\Gamma}_j) \tag{B.106}$$

Substituting Equation B.104 on page 109 in Equation B.105 on page 109 and Equation B.106 on page 109 and using the expressions for the inertial force and moment given by

Equation B.96 on page 106 and Equation B.103 we can rewrite Equation B.105 on page 109 and Equation B.106 on page 109 as

$$\underline{F}_{iR} = -\sum_{j=i}^{n}\left[\sum_{k=1}^{j}\underline{a}_{jk}q_k'' + \underline{a}_j^0 - m_j\underline{a}_0 + \underline{F}_{jE}\right] \quad \text{(B.107)}$$

$$\underline{M}_{iR} = -\sum_{j=i}^{n}\left[\sum_{k=1}^{j}(\underline{b}_{jk}q_k'') + \underline{b}_j^0 - T_j\underline{\alpha}_0 + \underline{M}_{jE} + \sum_{k=1}^{j}(\underline{r}_{ji}\times\underline{a}_{jk}q_k'') + \underline{r}_{ji}\times\underline{a}_j^0 - m_j\underline{r}_{ji}\times\underline{a}_0 + \underline{r}_{ji}\times\underline{F}_{jE}\right] \quad \text{(B.108)}$$

Here $\underline{a}_0, \underline{\alpha}_0$ represent the linear and angular accelerations of the base link.

If joint i is linear the joint force to be applied τ_i is given by

$$\tau_i = \underline{F}_{iR}\cdot\underline{e}_i \quad \text{(B.109)}$$

If joint i is rotational then the applied torque τ_i is given by

$$\tau_i = \underline{M}_{iR}\cdot\underline{e}_i \quad \text{(B.110)}$$

Thus in the differential equation of motion expressed as

$$\underline{\tau} = H\underline{q}'' + \underline{h} \quad \text{(B.111)}$$

we can compute the elements of the matrix H and the vector $\underline{h}$ by comparing Equation B.111 on page 110 with Equation B.107 on page 110 , Equation B.109 on page 110 or with Equation B.108 on page 110 , Equation B.110 on page 110 . Thus we obtain the following expressions for the components of the coeffecient matrices in the differential equation of motion.

For joint i linear

$$H_{ik} = (-\sum_{j=i}^{n} \underline{a}_{jk}) \cdot \underline{e}_i \qquad \text{(B.112)}$$

$$h_i = (-\sum_{j=i}^{n} [\underline{a}_j^0 - m_j \underline{a}_0 + \underline{F}_{jE}]) \cdot \underline{e}_i \qquad \text{(B.113)}$$

If joint i is rotational

$$H_{ik} = (-\sum_{j=i}^{n} [\underline{b}_{jk} + \underline{r}_{ji} \times \underline{a}_{jk}]) \cdot \underline{e}_i \qquad \text{(B.114)}$$

$$h_i = (-\sum_{j=i}^{n} [\underline{b}_j^0 - T_j \underline{\alpha}_0 + \underline{M}_{jE} + \underline{r}_{ji} \times (\underline{a}_j^0 - m_j \underline{a}_0 + \underline{F}_{jE})]) \cdot \underline{e}_i \qquad \text{(B.115)}$$

Formulation of the dynamic model is summarised in the flow chart in Figure 43 on page 119.

B.3 LINEARISED MODEL CONSTRUCTION

The dynamic model is given by

$$\underline{\tau} = H(\underline{q})\underline{q}'' + \underline{h}(\underline{q},\underline{q}')$$

For small perturbations about a nominal point the non-linear model may be replaced by its first-order approximation

$$\delta\underline{\tau} = H(\underline{q}_0)\delta\underline{q}'' + \partial/\partial\underline{q}[H(\underline{q}_0)]\underline{q}_0''\delta\underline{q} + \partial/\partial\underline{q}'[\underline{h}(\underline{q}_0,\underline{q}_0')]\delta\underline{q}' + \partial/\partial\underline{q}[\underline{h}(\underline{q}_0,\underline{q}_0')]\delta\underline{q} \qquad (B.116)$$

or

$$\delta\underline{\tau} = H(\underline{q}_0)\delta\underline{q}'' + H_1\delta\underline{q}' + H_2\delta\underline{q} \qquad (B.117)$$

where

$\underline{q}_0, \underline{q}_0', \underline{q}_0''$ = position, velocity and acceleration about which linearisation is done

$$H_1 = \partial/\partial\underline{q}'[\underline{h}(\underline{q}_0,\underline{q}_0')]$$

$$H_2 = \partial/\partial\underline{q}[H(\underline{q}_0)]\underline{q}_0'' + \partial/\partial\underline{q}[\underline{h}(\underline{q}_0,\underline{q}_0')]$$

Defining the state vector

$$\underline{\xi} = \begin{bmatrix} \delta\underline{q} \\ \\ \delta\underline{q}' \end{bmatrix}$$

Equation B.116 may be written in state-variable form as

$$\underline{\xi}' = \begin{bmatrix} 0 & I_n \\ \\ -H^{-1}H_2 & -H^{-1}H_1 \end{bmatrix} \underline{\xi} + \begin{bmatrix} 0 \\ \\ H^{-1} \end{bmatrix} \delta\underline{\tau} \qquad \text{(B.118)}$$

To construct this model it is neccesary to evaluate three derivatives $\partial/\partial\underline{q}[H]$ $(n\times n\times n)$, H_1 $(n\times n)$ and H_2 $(n\times n)$.

Let

$$\Delta_1\underline{a} = \partial/\partial q_1[\underline{a}] \quad , \quad \Delta_1{}^{\bullet}\underline{a} = \partial/\partial q_1'[\underline{a}]$$

The formula for rotating a vector $\underline{a}(q)$ about an axis $\underline{e}$ can be used to derive the expression for $\Delta_1\underline{a}$.

$$\underline{a}(q+\Delta q) = \underline{a}(q)\cos\Delta q + (1-\cos\Delta q)\underline{e}\bullet\underline{a}(q) + \underline{e}\times\underline{a}(q)\sin\Delta q$$

$$\partial/\partial q[\underline{a}] = \underline{e}\times\underline{a} \quad \text{or} \quad \Delta_1\underline{a} = \underline{e}_1\times\underline{a} \qquad \text{(B.119)}$$

Now if the 1^{th} joint is translational and $\underline{a}_i$ is a vector associated with link i then

$$\Delta_1\underline{a}_i = 0. \quad \text{Also} \quad \Delta_1\underline{a}_i = 0 \text{ if } i<1$$

Define

$$\eta_{1i} = 1 \quad i\geq 1$$
$$= 0 \quad \text{otherwise}$$

and

$$\xi_1 = 1 \text{ if joint 1 linear}$$
$$= 0 \text{ if joint 1 rotational}$$

The complete expression for the derivative may be written as

$$\Delta_1\underline{a}_i = (\underline{e}_1\times\underline{a}_i)(1-\xi_1)\eta_{1i} \qquad \text{(B.120)}$$

In the case of a composite vector i.e. a vector that may have components on more than one link,a similar expression can be obtained as shown below.

$$\Delta_1 \underline{r}_{ij} = [(\underline{e}_1 \times \underline{r}_{ij})\eta_{1,j-1} + (e_1 \times r_{i1})(\eta_{1i} - \eta_{1,j-1})](1-\xi_1) + \underline{e}_1(\eta_{1i}-\eta_{1,j-1})\xi_1 \qquad \text{(B.121)}$$

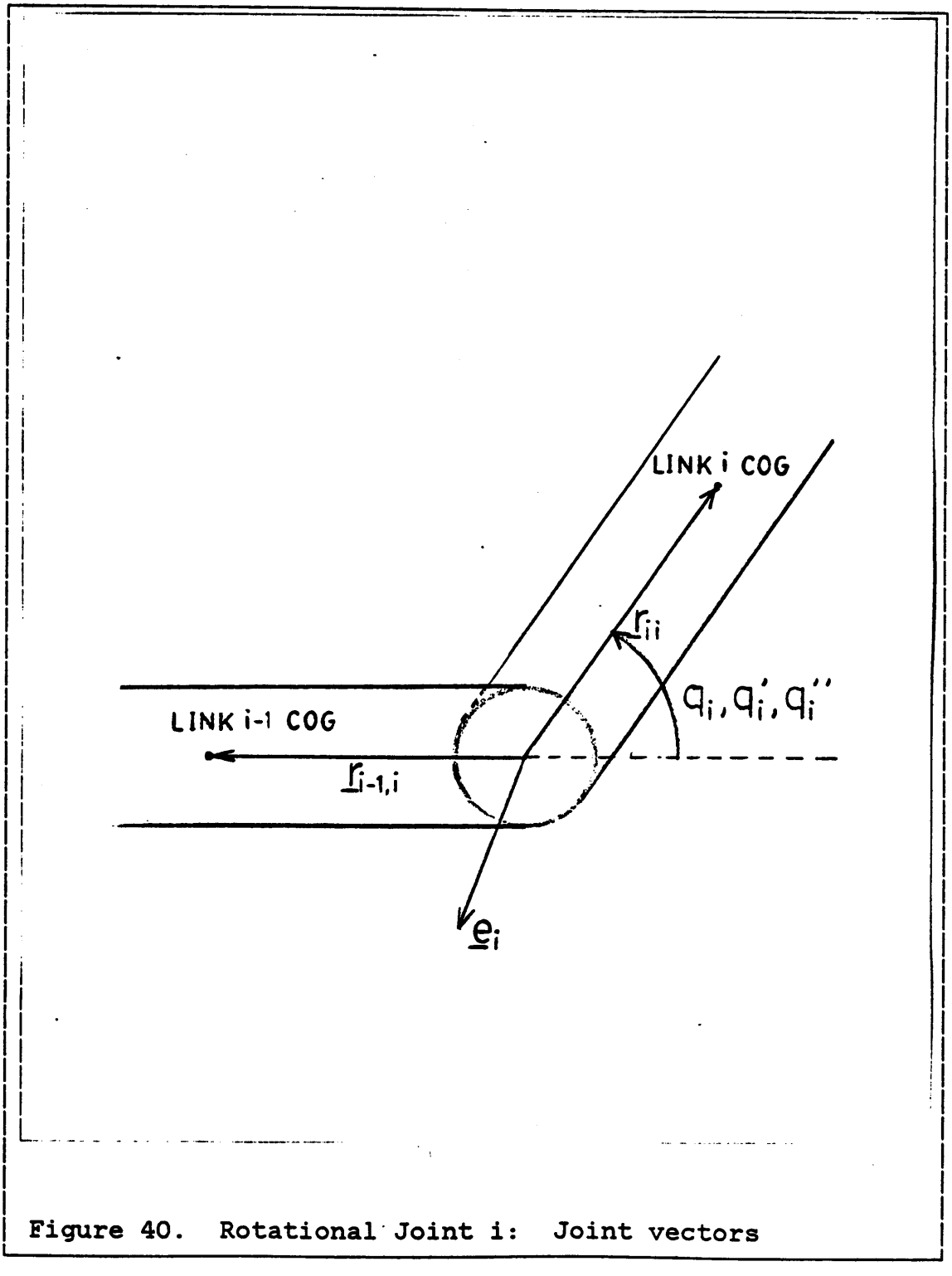


Figure 40. Rotational Joint i: Joint vectors

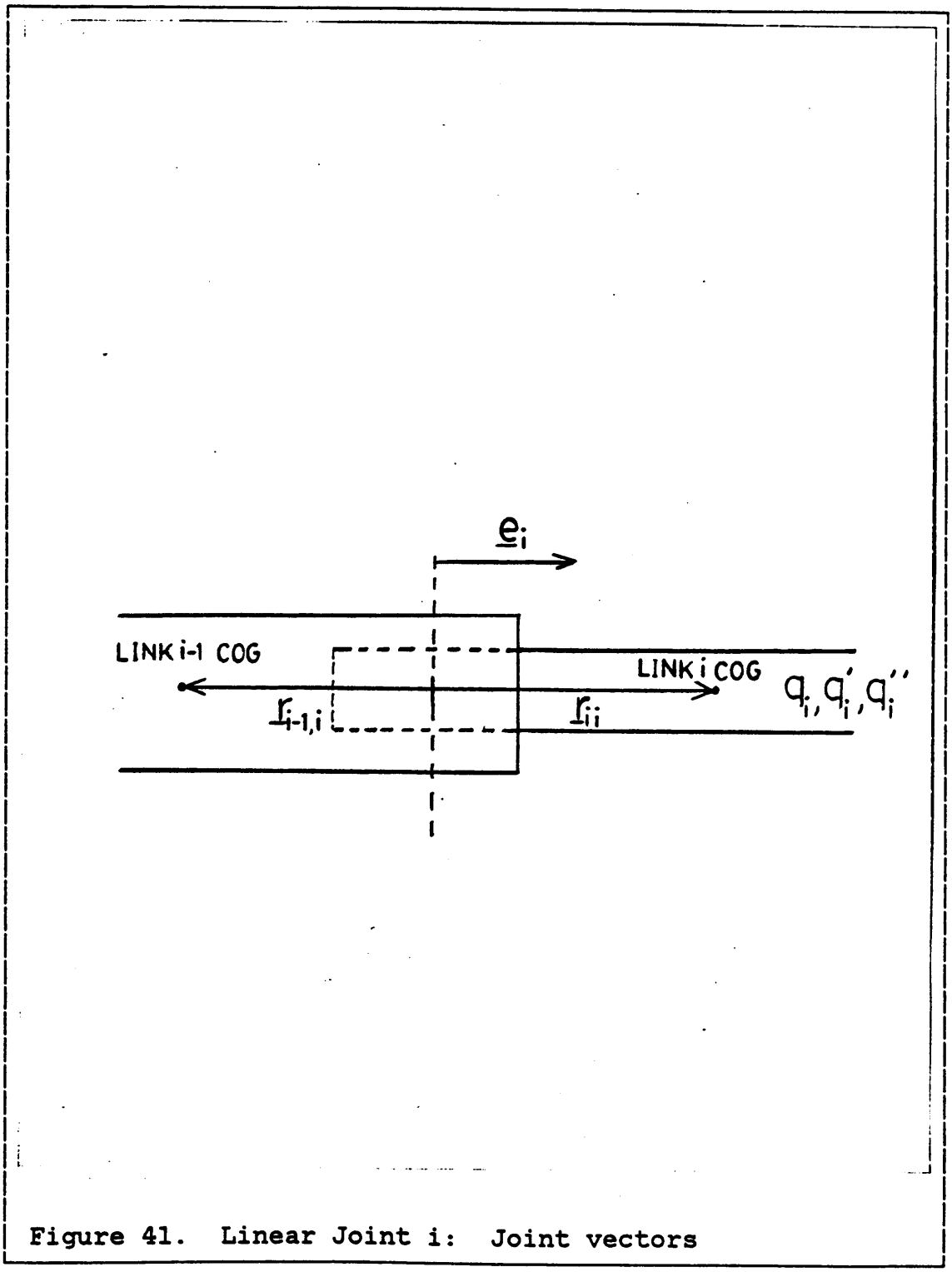


Figure 41. Linear Joint i: Joint vectors

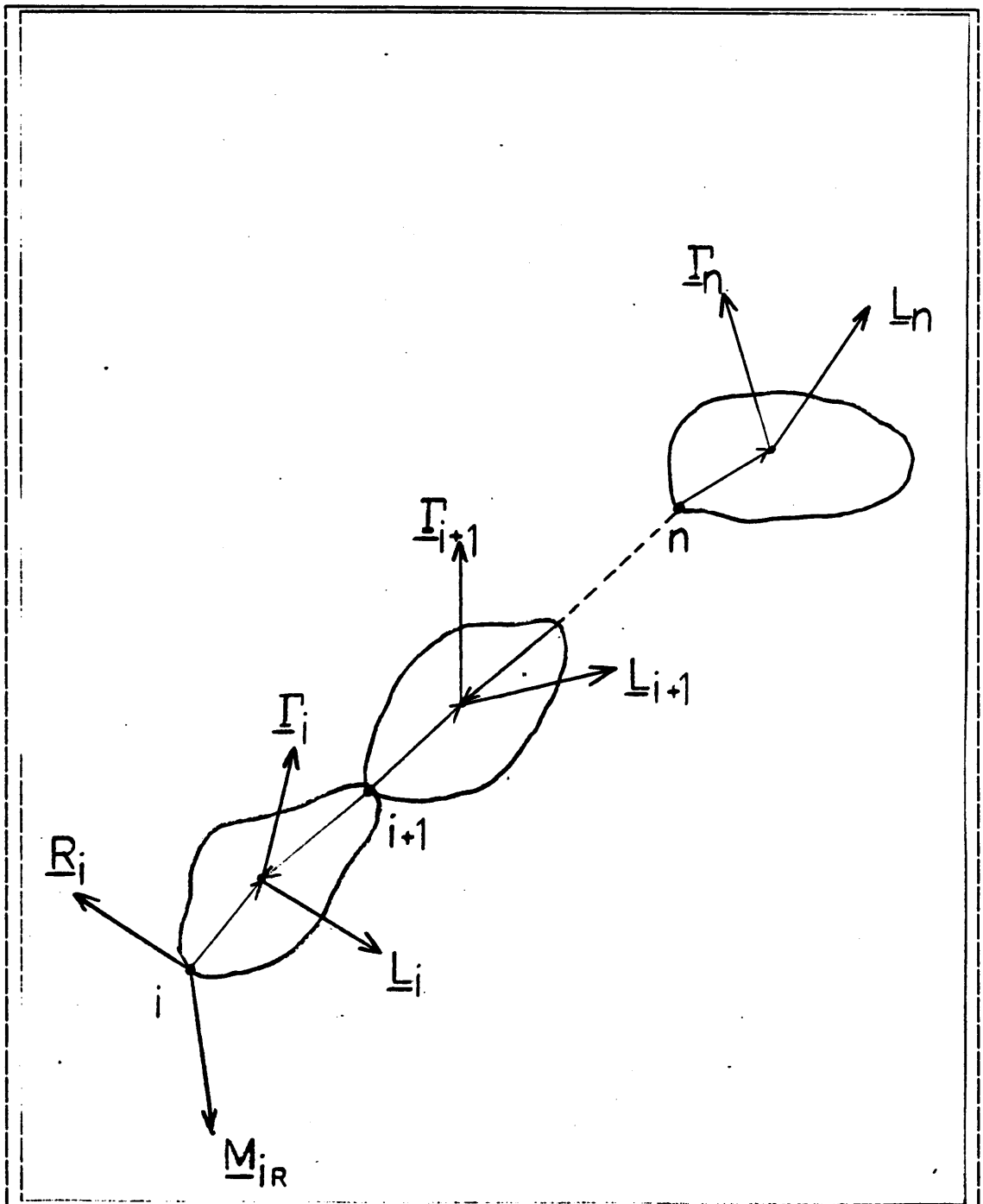


Figure 42. Joint i: Forces acting on links i to n

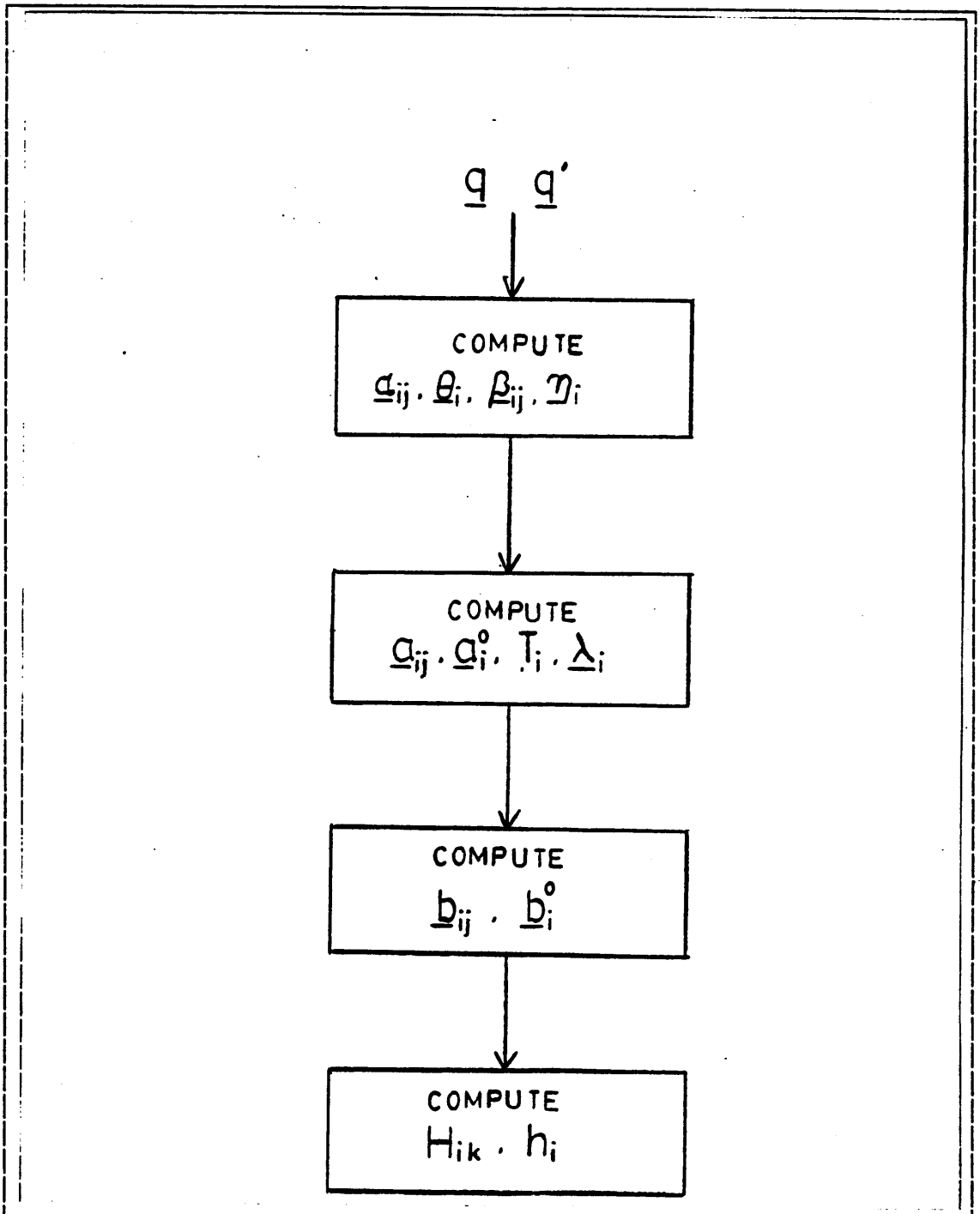


Figure 43. Dynamic model: Computing model parameters

APPENDIX C. EQUATIONS OF MOTION FOR 3-LINK ARM

The differential equations of motion can be derived from the Lagrange equations(Equation C.123 on page 120). The lagrangian L given by Equation C.122 on page 120 must first be derived.

$$L = T - V \tag{C.122}$$

where

- T = total kinetic energy of the manipulator
- V = total potential energy of the manipulator

The equations of motion are then obtained from

$$d/dt[\partial/\partial q_i'(L)] - \partial/\partial q_i(L) = \tau_i \quad , i=1,...n \tag{C.123}$$

Due to the complexity of the lagrangian for a 3-link arm,it is assumed here that the first link is fixed to the base and the resulting equations of motion are valid for the motion of the last two links,thus reducing the problem to that of a 2-link arm. The kinetic energy T_i for link i consists of two

components, rotational K.E., T_{iR} and translational K.E., T_{iT}. They are computed as follows

$$T_{iR}=1/2[\underline{\omega}_i^T J_i \underline{\omega}_i] \quad \text{(C.124)}$$

$$T_{iT}=1/2[m_i |\underline{v}_i|^2] \quad \text{(C.125)}$$

$$J_i = A_i I_i A_i^T \quad \text{(C.126)}$$

where

- $\underline{\omega}_i$ = angular velocity of link i wrt base co-ordinates.
- J_i = inertia tensor of link i wrt the base co-ordinates
- $\underline{v}_i$ = linear velocity of link i COG wrt base co-ordinates.
- m_i = mass of link i.
- A_i = Link i transformation matrix
- I_i = inertia tensor of link i wrt link co-ordinates

The relative orientation of the co-ordinate systems are shown in Figure 44 on page 125.

Co-ordinate frame 2 is obtained simply by a rotation q_2 about the y-axis of the base frame.

$$A_2 = \mathrm{Rot}(y,q_2) = \begin{bmatrix} C2 & 0 & S2 \\ 0 & 1 & 0 \\ -S2 & 0 & C2 \end{bmatrix} \qquad (C.127)$$

$A_3 = A_2$ since joint 3 is a linear joint. Let

$$I_2=\mathrm{diag}(i_{2x},i_{2y},i_{2z}) \ , \ I_3=\mathrm{diag}(i_{3x},i_{3y},i_{3z})$$

Now

$$\underline{\omega}_2 = \underline{\omega}_3 = [0 \ ,q_2',0]^T$$

Define

$$I_{3p}= \mathrm{diag}(i^p_{3x},i^p_{3y},i_{3z})$$

where

$$i^p_{3x}= i_{3x} + (d-q_3)^2 m_3 \ , \ i^p_{3y}= i_{3y} + (d-q_3)^2 m_3$$

In using Equation C.126, I_{3p} is to be used instead of I_3 since A_3 only describes the orientation of the link 3 co-ordinate system. Thus

$$T = \sum_{i=2}^{3} 1/2[\underline{\omega}_i^T J_i \underline{\omega}_i + m_i \underline{v}_i^T \underline{v}_i]$$

The potential energy only changes for link 3. We can take

$$V = V_3 = m_3 g[(q_3-d)\cos q_2 + d]$$

Substituting in Equation C.122 on page 120 we get

$$L=1/2\Big[(k_1+k_4)q_2^{2'} + k_2[(k_3-q_3)q_2']^2 + k_5 q_3^{2'} - k_6[(q_3-k_3)\cos q_2 + k_3]\Big]$$

where

$k_1=i_{2y}$, $k_2=m_3$, $k_3=d$, $k_4=i_{3y}$, $k_5=m_3$, $k=m_3 g$

Applying Equation C.123 on page 120 the equations of motion can be obtained easily

$$q_2'' = \frac{(q_3-k_3)(2q_2'q_3'k_2+k_6\sin q_2)}{k_1+k_4+k_2(q_3-k_3)^2} + \frac{1}{k_1+k_4+k_2(q_3-k_3)^2}\tau_2 \quad \text{(C.128)}$$

$$q_3'' = \frac{-(q_3-k_3)k_2q_2^{2'}-k_6\cos q_2}{k_5} + \frac{1}{k_5}\tau_3 \quad \text{(C.129)}$$

Writing in matrix form

$$\underline{q}''_{23} = C^* + D^*\underline{\tau}_{23} \quad \text{(C.130)}$$

where

$\underline{q}_{23}=[q_2\ q_3]^T$, $\underline{\tau}_{23}=[\tau_2\ \tau_3]^T$, $C^*=[c_1\ c_2]^T$, $D^*=\mathrm{diag}(d_1,d_2)$

The elements of matrices C^* and D^* are easily identified from Equation C.128 on page 124 and Equation C.129 on page 124 .

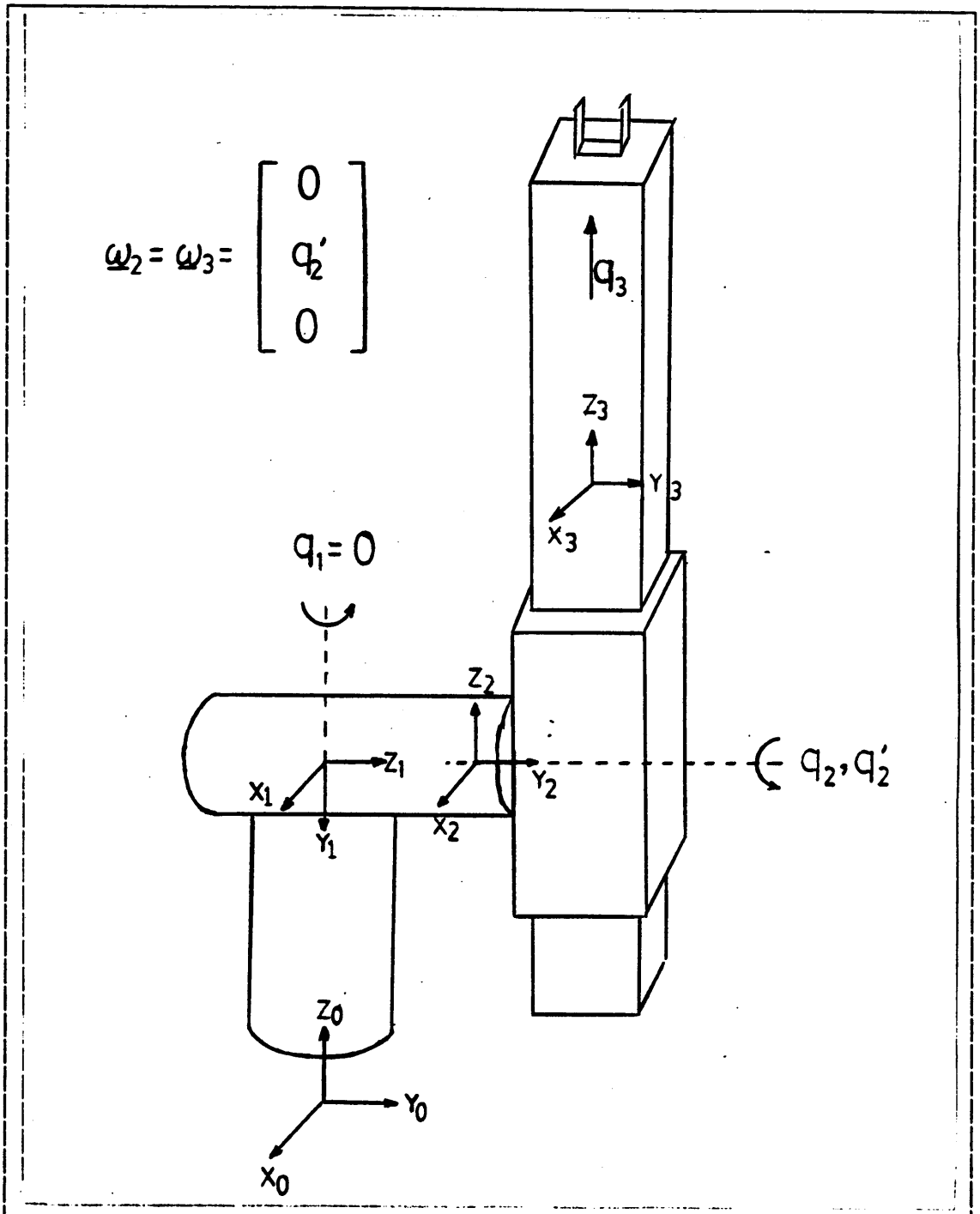


Figure 44. 3-ARM Manipulator: Link co-ordinate systems

APPENDIX D. DYNAMICS SIMULATOR

This version of the simulator includes the control algorithm based on linearised models. The input data to the program consist of geometric and mechanical details of the manipulator, actuator motor data in the form of state variable matrices, the matrices R and Q for computing the optimum input, initial and final position of the manipulator tip, total time specified for the motion and the number of steps the motion is to be performed in. A detailed list of data required is indicated in the input data section.

D.1 INPUT DATA

```
3,  N    =   NUMBER OF LINKS

0,0,1,    JOINT TYPE  ,ROT=0  TRANS=1

1,4,4,    JOINT CLASSIFICATION (SEE APPENDIX A)

9.29,.276,.255,.071,
5.01,.108,.018,.1,     MASS,IXX,IYY,IZZ  FOR EACH LINK
4.25,2.51,2.51,.006,

0,0,0,         0,0,0,EXTERNAL FORCE,MOMENT APPLIED TO
0,0,0,         0,0,0,EACH LINK.MANIPULATOR LOAD MAY BE
0,0,0,         0,0,0,SPECIFIED HERE

0,0,0,             ANGULAR ACCELERATION OF BASE LINK

0,0,+9.80621,      LINEAR ACCELERATION OF BASE LINK

0,-1,0,        0,0,1,JOINT AXIS VECTORS SPECIFIED WRT
0,1,0,         0,0,1,LINK FIXED CO-ORDINATE SYSTEM
0,0,1,         0,0,1,


0,.0175,-.1105,       0,.0175,-.2629 ,VECTORS FROM COG
0,-.1054,0            0,-.1054,0,        OF LINK TO
0,0,-.6447,           0,0,-.6447,       EITHER JOINT

                                        *
0,0,0,         0,0,0,      JOINT VECTORS R  TO BE DEFINED
0,0,1,         0,0,0,      IF THE JOINT CLASSIFICATION IS
0,1,0,         0,1,0,NOT TYPE 1

0,0,1,      JOINT 1 AXIS IN BASE CO-RDINATE SYSTEM

0,.1,-.1,   POSITION VECTOR FROM BASE ORIGIN TO JOINT 1

100,   NUMBER OF STEPS THE MOTION IS PERFORMED IN

1.0,   TOTAL MOTION TIME IN SECONDS

0.01,0,0,
0,.01,0,     MATRIX R FOR OPTIMUM INPUT
0,0,.01,

100000,0,0,0,0,0,
0,100000,0,0,0,0,
```

```
0,0,100000,0,0,0,    MATRIX Q FOR OPTIMUM INPUT
0,0,0,100000,0,0,
0,0,0,0,100000,0,
0,0,0,0,0,100000,

0.30,.1524,.153,INITIAL POSITION OF MANIPULATOR TIP

0.0,.1524,.653, FINAL POSITION OF MANIPULATOR TIP

0,1,0,-46.4,0,88.60,0,-29,ACTUATOR MOTOR STATE
0,1,0,-46.4,0,88.6,0,-29,  VARIABLE MATRICES
0,1,0,-46.4,0,88.6,0,-29,  A(2,2),B(2,1),F(2,1)

1.0,1.0,1.0       MOTOR GEAR RATIOS
```

D.2 PROGRAM LISTING

```
LINKAS: PROC OPTIONS (MAIN);
DCL (D,TIP,LE(3,3),LE1(3,3),LR(3,3),LR1(3,3),BEO(3),
E1T(3),R1T(3),AIM1(3,3),RIT(3),EIT(3),A(3,3),EIF(3),
EI(3),R1(3),SA(3),SAT(3),SB(3),SBT(3),CB(9),CA(9,9),
CX(9),UT1(3,3),UT2(3,3),UT3(3,3),UT4(3,3),UT5(3,3),
UT6(3,3),UT7(3,3),UT8(3,3),E(3),UTI(3,3),PF(3),PS(3),
TI(3,3),V1(3),V2(3),R(3,3),AR1(3,3),EIA(3,3),T1(3,3),
T2(3,3),T3(3,3),T4(3,3),T5(3,3),T6(3,3),T7(3,3),
T8(3,3) ,SLR(3,3),SLR1(3,3),V3(3),V4(3),QD(3),MASS(3),
JI(3,3),WO(3),QDD(3),EO(3),G(3,3),MJG(3,3)) FLOAT(16);
DCL (OMEGA(3,3),ALPHA(3,3,3),THETA(3,3),BETA(3,3,3),
EATA(3,3),AIJ(3,3,3),RO(3),AOI(3,3),
TI1(3,3),TI2(3,3),TI3(3,3),TI4(3,3),
BOI(3,3),TI5(3,3),TI6(3,3),
TI7(3,3),TI8(3,3),BIJ(3,3,3)) FLOAT(16);
DCL (Q1(3),QD1(3),Q(3),P(3),HINV(3,3),H(3,3),SH(3),
D1,D2,CONR,CONQ,TAU(3)) FLOAT(16);
DCL (QA(3),QDA(3),QDDA(3),PK(9,9),PK1(9,9),RHO,PHI(9),
PHIT(9),NUHAT(6,9),NUHAT1(6,9),XK(6),XK1(6),
FHAT(6,6),GHAT(6,3),GHATT(3,6),CAPR(3,3),VS4(3),
CAPQ(6,6),USTAR(3),DELT,TTIME,W1(3,6),W2(3,3),
VS1(6),W6(3,3),VS2(9),W4(9,9),W5(9,9),VS3(9),
CONF(6,6),SIGMA,JQ(101,3),DJQ(101,3),DDJQ(101,3),
CONG(6,3)) FLOAT(16);
DCL (DH(3,3,3),DSH(3,3),DDSH(3,3)) FLOAT(16);
DCL (N9,NL,I,ZETA(3),K,S,SP(3),NL2,NL3) FIXED BIN;
DCL (IF,EDJQ(101,3),USTR(101,3),NTOR(101,3),
EJQ(101,3),JQT(101,3),AJQ(101,3),TAUM(101,3),
ADJQ(101,3))FLOAT(10);
DCL (NGR1,NGC1,NGR2,NGC2,NGR3,NGC3) FIXED BIN;
DCL (NFR1,NFC1,NFR2,NFC2,NFR3,NFC3) FIXED BIN;
DCL(K1,K2,K3,J) FIXED BIN;
DCL(MA1(2,2),MA2(2,2),MA3(2,2),MB1(2),MB2(2),MB3(2),
MF2(2),MF3(2),BIGA(6,6),BIGB(6,3),BIGT(3,6),BIGF(6,3),
BIGX(6),TF(3,3),HTF(3,3),TB(3,3),HTB(3,3),VOLT(3),
GRI(3,3),TA(3,6),HTA(3,6),HTX(3),HTBI(3,3),GR(3),
FG(6,3),HTFG(3,3),AD(6,6),FP(6),MF1(2))FLOAT(16);
DCL (BIGX1(6),BD(6,3),ET(6,6),HT(3,6)) FLOAT(16);
GET LIST (NL);
DO I=1 TO NL;
GET LIST (ZETA(I)); END;
DO I=1 TO NL;
GET LIST (SP(I)); END;
DO I=1 TO NL;
GET LIST(MASS(I),JI(I,1),JI(I,2),JI(I,3)); END;
DO I=1 TO NL;
GET LIST(G(I,*),MJG(I,*)); END;
```

```
GET LIST(EO(*),WO(*));
DO I=1 TO NL;
GET LIST (LE(I,*),LE1(I,*),LR(I,*),LR1(I,*));
END;
DO I=1 TO NL;
GET LIST (SLR(I,*),SLR1(I,*)); END;
GET LIST (BEO(*),RO(*));
GET LIST (NSTEP);
GET LIST (TTIME);
GET LIST (CAPR);
GET LIST (CAPQ);
GET LIST (RHO);
GET LIST(NFR1); GET LIST(NFC1);
GET LIST(NFR2); GET LIST(NFC2);
GET LIST(NFR3); GET LIST(NFC3);
GET LIST(NGR1); GET LIST(NGC1);
GET LIST(NGR2); GET LIST(NGC2);
GET LIST(NGR3); GET LIST(NGC3);
GET LIST (SIGMA);
GET LIST(PS,PF);
GET LIST(MA1,MB1,MF1);
GET LIST(MA2,MB2,MF2);
GET LIST(MA3,MB3,MF3);
GET LIST(GR);
BIGA=0;
DO I=1 TO 2;
DO J=1 TO 2;
BIGA(I,J)=MA1(I,J);
BIGA(I+2,J+2)=MA2(I,J);
BIGA(I+4,J+4)=MA3(I,J);
END;
END;
BIGB=0;
DO I=1 TO 2;
BIGB(I,1)=MB1(I);
BIGB(I+2,2)=MB2(I);
BIGB(I+4,3)=MB3(I);
END;
BIGF=0;
DO I=1 TO 2;
BIGF(I,1)=MF1(I);
BIGF(I+2,2)=MF2(I);
BIGF(I+4,3)=MF3(I);
END;
GRI=0;
BIGT=0;
K1=2;
DO I=1 TO 3;
BIGT(I,K1)=1.0/GR(I);
K1=K1+2;
GRI(I,I)=1.0/GR(I);
```

```
END;
K1=3; K2=6; K3=6;
CALL MULMAT(BIGT,BIGA,TA,K1,K2,K3);
K1=3; K2=6; K3=3;
CALL MULMAT(BIGT,BIGB,TB,K1,K2,K3);
K1=3; K2=6; K3=3;
CALL MULMAT(BIGT,BIGF,TF,K1,K2,K3);
K1=6; K2=3; K3=3;
CALL MULMAT(BIGF,GRI,FG,K1,K2,K3);
CALL TRJPLN(PS,PF);
CALL DISCR2(BIGA,BIGB);
E1T=BEO; R1T=RO; AIM1=0;
DO K=1 TO 3; AIM1(K,K)=1; END;
RIT=LR(1,*); EIT=LE(1,*);
DO I=1 TO NL;
CALL MATMUL (AIM1,E1T,EI);
CALL MATMUL (AIM1,R1T,R1);
CALL DOTVEC (R1,EI,D);
SA=R1-(EI*D);
S=-1;
CALL UNIVEC (SA,SA,S);
CALL DOTVEC (RIT,EIT,D);
SAT=RIT-EIT*D;
S=1;
CALL UNIVEC (SAT,SAT,S);
CALL CROVEC (EI,SA,SB);
CALL CROVEC (EIT,SAT,SBT);
S=1; CALL UNIVEC (SBT,SBT,S);
DO K=1 TO 3;
CB(K)=SA(K); CB(K+3)=EI(K); CB(K+6)=SB(K);
END;
CA=0;
DO K=1 TO 3;
CA(1,K),CA(2,K+3),CA(3,K+6)=SAT(K);
CA(4,K),CA(5,K+3),CA(6,K+6)=EIT(K);
CA(7,K),CA(8,K+3),CA(9,K+6)=SBT(K);
END; N9=9;
CALL SOLLIN (CA,CB,CX,N9);
DO K=1 TO 3;
A(1,K)=CX(K); A(2,K)=CX(K+3); A(3,K)=CX(K+6);
END;
LBL2: IF I=1 THEN UT1=A;
ELSE IF I=2 THEN UT2=A;
ELSE IF I=3 THEN UT3=A;
ELSE IF I=4 THEN UT4=A;
ELSE IF I=5 THEN UT5=A;
ELSE IF I=6 THEN UT6=A;
ELSE IF I=7 THEN UT7=A;
ELSE IF I=8 THEN UT8=A;
IF I=NL THEN GOTO LBL1;
AIM1=A;
```

```
E1T=LE1(I,*);  EIT=LE(I+1,*);
IF SP(I+1)=1 THEN DO;
R1T=LR1(I,*); RIT=LR(I+1,*); END;
ELSE IF SP(I+1)=2 THEN DO;
R1T=SLR1(I,*); RIT=SLR(I+1,*); END;
ELSE IF SP(I+1)=3 THEN DO;
R1T=SLR1(I,*); RIT=LR(I+1,*); END;
ELSE DO; R1T=LR1(I,*); RIT=SLR(I+1,*); END;
LBL1: END;
/*
THIS PROCEDURE COMPUTES THE JOINT SPACE TRAJECTORIES
GIVEN
INITIAL AND FINAL POSITIONS IN CARTESIAN CO-ORDINATES
AND
TOTAL TIME REQUIRED FOR THE TASK
*/
TRJPLN: PROC(P1,P2);
DCL (IL,I) FIXED BIN;
DCL (P1(3),P2(3),DEL(3),P(3),PX,PY,PZ,C2,S2,
XD,ZD) FLOAT(16);
P1=P1+RO; P2=P2+RO;
XD=(P2(1)-P1(1))/TTIME;
ZD=(P2(3)-P1(3))/TTIME;
DEL=(P2-P1)/(NSTEP);
DO IL=1 TO NSTEP+1;
P=P1+(IL-1)*DEL;
PX=P(1); PY=P(2); PZ=P(3);
JQ(IL,3)=SQRT(PX**2+PZ**2);
JQ(IL,2)=ATAN(PX,PZ);
END;
JQ(*,1)=0.0;
DELT=TTIME/NSTEP;
DO I=1 TO NSTEP+1;
C2=COS(JQ(I,2)); S2=SIN(JQ(I,2));
DJQ(I,2)=(C2*XD-S2*ZD)/JQ(I,3);
DJQ(I,3)=S2*XD+C2*ZD;
DDJQ(I,2)=(-2.0*DJQ(I,2)*DJQ(I,3))/JQ(I,3);
DDJQ(I,3)=JQ(I,3)*(DJQ(I,2)**2);
END;
DDJQ(*,1),DJQ(*,1)=0;
END TRJPLN;

/* CONTROL ALGORITHM  */

NL2=NL*2; NL3=NL*3;
AJQ(1,*),QA=JQ(1,*); ADJQ(1,*),QDA=DJQ(1,*);
RO=0;
USTR(1,*),USTAR=0;
DO I=1 TO NSTEP;
QD=DJQ(I,*); Q=JQ(I,*);
QDD=DDJQ(I,*);
```

```
CALL ORIEN;
CALL INERTIA; NTOR(I,*),P=TAU;
DO J=1 TO NL;
XK(J)=QA(J)-Q(J); XK(NL+J)=QDA(J)-QD(J);  END;
IF I=1 THEN GOTO LAB1;
CALL LINEAR;
CALL DISCRET;
FHAT=CONF; GHAT=CONG;
CALL TRANSP(GHAT,GHATT,NL2,NL);
CALL MULMAT(GHATT,CAPQ,W1,NL,NL2,NL2);
CALL MULMAT(W1,GHAT,W2,NL,NL2,NL);
W2=W2+CAPR;
K1=3;
CALL MATINV(W2,W6,K1);
CALL MULVET(FHAT,XK,VS1,NL2,NL2);
CALL MULVET(W1,VS1,VS4,NL,NL2);
CALL MULVET(W6,VS4,USTAR,NL,NL); USTAR=-USTAR;
USTR(I,*)=USTAR;
P=P+USTAR;
LAB1: K1=1;
DO J=1 TO 3;
BIGX(K1)=GR(J)*QA(J);
BIGX(K1+1)=GR(J)*QDA(J);
K1=K1+2;
END;
K1=3; K2=3; K3=3;
CALL MULMAT(H,TF,HTF,K1,K2,K3);
K1=3; K2=3; K3=3;
CALL MULMAT(HTF,GRI,HTFG,K1,K2,K3);
K1=3; K2=3; K3=3;
CALL MULMAT(H,TB,HTB,K1,K2,K3);
K1=3;
CALL MATINV(HTB,HTBI,K1);
HTFG=-HTFG;
DO K=1 TO 3;
HTFG(K,K)=1+HTFG(K,K);
END;
K1=3; K2=3;
CALL MULVET(HTFG,P,VOLT,K1,K2);
K1=3; K2=3; K3=6;
CALL MULMAT(H,TA,HTA,K1,K2,K3);
K1=3; K2=6;
CALL MULVET(HTA,BIGX,HTX,K1,K2);
HTX=VOLT-SH-HTX;
K1=3; K2=3;
CALL MULVET(HTBI,HTX,VOLT,K1,K2);
K1=6; K2=3;
CALL MULVET(FG,P,FP,K1,K2);
CALL NEXSTP(BIGX,VOLT,FP,BIGX1);
K1=1;
DO J=1 TO 3;
```

```
QA(J)=BIGX1(K1)/GR(J);
QDA(J)=BIGX1(K1+1)/GR(J);
K1=K1+2;
END;
AJQ(I+1,*)=QA;
ADJQ(I+1,*)=QDA;
TAUM(I,*)=VOLT;
END;
TAUM(NSTEP+1,*)=TAUM(NSTEP,*);
USTR(NSTEP+1,*)=USTR(NSTEP,*);

/* COMPUTES INERTIAL MOMENT AND FORCE */


INERTIA: PROC;
DCL (OMEGO(3),THETO(3),OMEG1(3),
V1(3),V2(3),V3(3), RII(3),RII1(3),
EATO(3),E(3),OMEGI(3),V4(3),V5(3),
V6(3),V7(3),V8(3),
TI(3,3),SUM,T(3,3),
D1,D2,D3,LAMDA(3,3),F(3,3),
M(3,3),D,RJI(3),GAMJ(3,3),LJ(3,3),
RF(3),RM(3)) FLOAT(16);
DCL (I,J,K,L) FIXED BIN;
OMEGO=0;
DO I=1 TO NL;
IF ZETA(I)=1 THEN DO;
OMEGA(I,*)=OMEGO; END;
ELSE DO;
OMEGA(I,*)=OMEGO+QD(I)*EIA(I,*); END;
OMEGO=OMEGA(I,*); END;
THETO,OMEG1=0;
DO I=1 TO NL;
IF ZETA(I)=0 THEN DO;
V2,ALPHA(I,I,*)=EIA(I,*);
CALL CROVEC(OMEG1,V2,V3);
THETA(I,*)=THETO+QD(I)*V3; END;
ELSE DO;
ALPHA(I,I,*)=0;
THETA(I,*)=THETO; END;
IF I=1 THEN GOTO LBL6;
DO J=1 TO I-1;
ALPHA(I,J,*)=ALPHA(I-1,J,*);
END;
LBL6:THETO=THETA(I,*); OMEG1=OMEGA(I,*);
END;
RII=R(1,*); RII1=RO; EATO,THETO,OMEG1=0;
E=EIA(1,*); OMEGI=OMEGA(1,*);
DO I=1 TO NL;
IF ZETA(I)=0 THEN DO;
CALL CROVEC(E,RII,V1);
```

```
BETA(I,I,*)=V1;
V2=RII-RII1;
CALL CROVEC(THETO,V2,V3);
CALL CROVEC(OMEG1,E,V4);
CALL CROVEC(V4,RII,V5);
CALL CROVEC(OMEGI,RII,V6);
CALL CROVEC(OMEGI,V6,V7);
CALL CROVEC(OMEG1,RII1,V6);
CALL CROVEC(OMEG1,V6,V8);
EATA(I,*)=EATO+V3+QD(I)*V5+V7-V8; END;
ELSE DO;
BETA(I,I,*)=E; V1=RII-RII1;
CALL CROVEC(THETO,V1,V2);
CALL CROVEC(OMEG1,E,V3);
CALL CROVEC(OMEGI,RII,V4);
CALL CROVEC(OMEGI,V4,V5);
CALL CROVEC(OMEG1,RII1,V6);
CALL CROVEC(OMEG1,V6,V7);
EATA(I,*)=EATO+V2+2*QD(I)*V3+V5-V7;
END;
IF I=1 THEN GOTO LBL7;
DO J=1 TO I-1;
V1=ALPHA(I-1,J,*); V2=RII-RII1;
CALL CROVEC(V1,V2,V3);
BETA(I,J,*)=BETA(I-1,J,*)+V3;
END;
LBL7: IF I=NL THEN GOTO LBL8;
EATO=EATA(I,*); RII=R(I+1,*);
RII1=AR1(I,*);E=EIA(I+1,*);
OMEGI=OMEGA(I+1,*); OMEG1=OMEGA(I,*);
THETO=THETA(I,*);
LBL8:END;
DO I=1 TO NL;
AOI(I,*)=-MASS(I)*EATA(I,*);
DO J=1 TO I;
AIJ(I,J,*)=-MASS(I)*BETA(I,J,*);
END;
END;
DO I=1 TO NL;
IF I=1 THEN TI=T1;
ELSE IF I=2 THEN TI=T2;
ELSE IF I=3 THEN TI=T3;
ELSE IF I=4 THEN TI=T4;
ELSE IF I=5 THEN TI=T5;
ELSE IF I=6 THEN TI=T6;
ELSE IF I=7 THEN TI=T7;
ELSE IF I=8 THEN TI=T8;
DO J=1 TO 3; DO K=1 TO 3; SUM =0;
DO L=1 TO 3;
SUM =SUM+TI(J,L)*TI(K,L)*JI(I,L);
END;
```

```
T(J,K)=SUM;
END;
END;
IF I=1 THEN TI1=T;
ELSE IF I=2 THEN TI2=T;
ELSE IF I=3 THEN TI3=T;
ELSE IF I=4 THEN TI4=T;
ELSE IF I=5 THEN TI5=T;
ELSE IF I=6 THEN TI6=T;
ELSE IF I=7 THEN TI7=T;
ELSE IF I=8 THEN TI8=T;
DO J=1 TO I;
V1=ALPHA(I,J,*); CALL MATMUL(T,V1,V2);
BIJ(I,J,*)=-V2;
END;
V1=THETA(I,*); CALL MATMUL(T,V1,V2);
BOI(I,*)=-V2;
V1=TI(*,1); V2=TI(*,2); V3=TI(*,3);
V4=OMEGA(I,*);
CALL DOTVEC(V4,V1,D1); CALL DOTVEC(V4,V2,D2);
CALL DOTVEC(V4,V3,D3);
V5(1)=(JI(I,2)-JI(I,3))*D2*D3;
V5(2)=(JI(I,3)-JI(I,1))*D3*D1;
V5(3)=(JI(I,1)-JI(I,2))*D1*D2;
CALL MATMUL(TI,V5,V6);
LAMDA(I,*)=V6;
BOI(I,*)=BOI(I,*)+LAMDA(I,*);
END;
/* COMPUTE INERTIAL FORCE AND MOMENT */
DO I=1 TO NL;
F(I,*)=AOI(I,*)-MASS(I)*WO;
DO J=1 TO I;
F(I,*)=AIJ(I,J,*)*QDD(J) +F(I,*);
END;
END;
DO I=1 TO NL;
IF I=1 THEN TI=TI1;
ELSE IF I=2 THEN TI=TI2;
ELSE IF I=3 THEN TI=TI3;
ELSE IF I=4 THEN TI=TI4;
ELSE IF I=5 THEN TI=TI5;
ELSE IF I=6 THEN TI=TI6;
ELSE IF I=7 THEN TI=TI7;
ELSE IF I=8 THEN TI=TI8;
CALL MATMUL (TI,EO,V1);
M(I,*)=BOI(I,*)-V1;
DO J=1 TO I;
M(I,*)=M(I,*)+BIJ(I,J,*)*QDD(I);
END;
END;
/* COMPUTING APPLIED FORCE/TORQUE
```

```
   GIVEN THE ACCELERATIONS */
DO J=1 TO NL;
GAMJ(J,*)=F(J,*)+G(J,*);
LJ(J,*)=M(J,*)+MJG(J,*);
END;
DO I=1 TO NL;
E=EIA(I,*);
IF ZETA(I)=1 THEN DO;
RF=0; DO J=I TO NL;
RF=RF-GAMJ(J,*);
END;
CALL DOTVEC(E,RF,D); TAU(I)=D;
END;
ELSE DO;
RM,RJI=0; DO J=I TO NL; RJI=RJI+R(J,*);
V1=GAMJ(J,*); CALL CROVEC(RJI,V1,V2);
RM=-LJ(J,*)-V2+RM;
RJI=RJI-AR1(J,*);
END;
CALL DOTVEC(E,RM,D); TAU(I)=D;  END; END;
/* COMPUTING CO-EFFICIENT MATRICES OF THE
DIFFERENTIAL EQUATIONS */
DO I=1 TO NL;
IF ZETA(I)=1 THEN DO;
V2=EIA(I,*);
DO K=1 TO NL;
V1=0;
DO J=K TO NL;
V1=V1-AIJ(J,K,*);
END;
CALL DOTVEC(V1,V2,D);
H(I,K)=D;
END;
V1=0;
DO J=I TO NL;
V1=V1+AOI(J,*)-MASS(J)*WO+G(J,*);
END;
V1=-V1;
CALL DOTVEC(V1,V2,D);
SH(I)=D;
END;
ELSE DO;
V5=EIA(I,*);
DO K=1 TO NL; V6=0;
DO J=K TO NL;
CALL VECTOR(J,I,RJI);
V3=AIJ(J,K,*);
CALL CROVEC(RJI,V3,V4);
V6=V6+BIJ(J,K,*)+V4;
LBL5: END;
V6=-V6;
```

```
CALL DOTVEC(V6,V5,D);
H(I,K)=D;
END;
V6,V3,RJI=0;
DO J=I TO NL;
IF J=1 THEN TI=TI1;
ELSE IF J=2 THEN TI=TI2;
ELSE IF J=3 THEN TI=TI3;
ELSE IF J=4 THEN TI=TI4;
ELSE IF J=5 THEN TI=TI5;
ELSE IF J=6 THEN TI=TI6;
ELSE IF J=7 THEN TI=TI7;
ELSE IF J=8 THEN TI=TI8;
RJI=RJI+R(J,*);
V2=AOI(J,*)-MASS(J)*WO+G(J,*);
CALL CROVEC(RJI,V2,V3);
V6=V6+V3+BOI(J,*)+MJG(J,*);
CALL MATMUL(TI,EO,V4); V6=V6-V4;
END;
V6=-V6;
CALL DOTVEC(V3,V5,D);
SH(I)=D;
END;
END;
END INERTIA;

/*LINEARISED MODEL  CONSTRUCTION */

LINEAR: PROC;
DCL (K,J,EKJ,L,N,M,ELI,I,IN,EKI1) FIXED BIN;
DCL (DALPHA(3,3,3,3),V1(3),V2(3),V3(3),V4(3),V5(3),
DBETA(3,3,3,3),V6(3),V7(3),V8(3),V9(3),V10(3),V11(3),
DAIJ(3,3,3,3),T(3,3),SUM,DT(3,3),V12(3),V13(3),V14(3),
DBIJ(3,3,3,3),DEI(3),D1,D2,RJI(3),DOMEG(3,3,3),
RJK(3),DRJ(3),DALF(3,3,3),D3,
DGAM(3,3,3),DGAM1(3,3,3),DRIJ(3),
DBET(3,3,3),DAJO(3,3,3),V15(3),V16(3),V17(3),V18(3),
V19(3),V20(3),V21(3),V22(3),V23(3),AJ(3,3),
XJ(3),DAJ(3,3),C1,C2,DOMEGT(3),DXJ(3),DLAMDA(3,3,3),
DTJ(3,3),DBJO(3,3,3),DRJI(3),D,
DDOMEG(3,3,3),DDALF(3,3,3),DDBET(3,3,3),
DDAJO(3,3,3),V24(3),TJ(3,3),C3,DDXJ(3),DDLAMD(3,3,3),
DDBJO(3,3,3),JJ(3)) FLOAT(16);
DO L=1 TO NL;
DO K=1 TO NL;
DO J=K TO NL;
IFK=J THEN DO;
CALL DAXIS(L,J,V1);
DALPHA(L,J,J,*)=V1*(1-ZETA(J));
V2=R(J,*);
CALL DIFPOS(L,J,J,V3);
```

```
V4=EIA(J,*);
CALL CROVEC(V1,V2,V5);
CALL CROVEC(V4,V3,V6);
DBETA(L,J,K,*)=(V5+V6)*(1-ZETA(J)) + V1*ZETA(J);
DAIJ(L,J,K,*)=-MASS(J)*DBETA(L,J,K,*);
END;
ELSE DO;
V7=R(J,*);
V8=AR1(J-1,*);
CALL DIFPOS(L,J,J,V9);  IN=J-1;
CALL DIFPOS(L,IN,J,V10);
V11,DALPHA(L,J,K,*)=DALPHA(L,IN,K,*);
V12=ALPHA(IN,K,*);  V13=V7-V8; V14=V9-V10;
CALL CROVEC(V12,V14,V15);
CALL CROVEC(V11,V13,V16);
DBETA(L,J,K,*)=DBETA(L,IN,K,*)+V15+V16;
DAIJ(L,J,K,*)=-MASS(J)*DBETA(L,J,K,*);
END;
END;
END;
END;
DO L=1 TO NL;
DO K=1 TO NL;
DO J=K TO NL;
IF J=1 THEN TJ=TI1;
ELSE IF J=2 THEN TJ=TI2;
ELSE IF J=3 THEN TJ=TI3;
ELSE IF J=4 THEN TJ=TI4;
ELSE IF J=5 THEN TJ=TI5;
ELSE IF J=6 THEN TJ=TI6;
ELSE IF J=7 THEN TJ=TI7;
ELSE IF J=8 THEN TJ=TI8;
IF J=1 THEN T=T1;
ELSE IF J=2 THEN T=T2;
ELSE IF J=3 THEN T=T3;
ELSE IF J=4 THEN T=T4;
ELSE IF J=5 THEN T=T5;
ELSE IF J=6 THEN T=T6;
ELSE IF J=7 THEN T=T7;
ELSE IF J=8 THEN T=T8;
IF L <= J THEN ELI=1; ELSE ELI=0;
V1=EIA(L,*);
V2=T(*,1);
V3=T(*,2);
V4=T(*,3);
CALL CROVEC(V1,V2,V5); V5=V5*(1-ZETA(L))*ELI;
CALL CROVEC(V1,V3,V6); V6=V6*(1-ZETA(L))*ELI;
CALL CROVEC(V1,V4,V7); V7=V7*(1-ZETA(L))*ELI;
DO I=1 TO 3;
DO N=1 TO 3;
SUM =0;
```

```
DO M=1 TO 3;
IF M=1 THEN V8=V5;
ELSE IF M=2 THEN V8=V6;
ELSE IF M=3 THEN V8=V7;
IF M=1 THEN V9=V2;
ELSE IF M=2 THEN V9=V3;
ELSE IF M=3 THEN V9=V4;
SUM=SUM+ (V8(I)*V9(N)+V9(I)*V8(N))*JI(J,M);
END;
DT(I,N)=SUM;
END;
END;
V11=ALPHA(J,K,*); V13=DALPHA(L,J,K,*);
CALL MATMUL(DT,V11,V12);
CALL MATMUL(TJ,V13,V14);
DBIJ(L,J,K,*)=-V12-V14;
END;
END;
END;
DO L=1 TO NL;
DO I=1 TO NL;
V1=EIA(I,*);
CALL DAXIS(L,I,DEI);
DO K=1 TO NL;
IF ZETA(I)=1 THEN DO;
V2,V3=0;
DO J=K TO NL;
V2=V2+AIJ(J,K,*);
V3=V3+DAIJ(L,J,K,*);
END;
CALL DOTVEC(DEI,V2,D1);
CALL DOTVEC(V1,V3,D2);
DH(L,I,K)=-D1-D2;
END;
ELSEDO;
V2,V3=0;
DO J=K TO NL;
CALL VECTOR(J,I,RJI);
V4=AIJ(J,K,*);
CALL CROVEC(RJI,V4,V5);
V2=V2+BIJ(J,K,*)+V5;
V6=DAIJ(L,J,K,*);
CALL CROVEC(RJI,V6,V7);
CALL DIFPOS(L,J,I,V8);
CALL CROVEC(V8,V4,V9);
V3=V3+DBIJ(L,J,K,*)+V9+V7;
END;
CALL DOTVEC(DEI,V2,D1);
CALL DOTVEC(V1,V3,D2);
DH(L,I,K)=-D1-D2;
END;
```

```
END;
END;
END;
DOMEG=0;
DO L=1 TO NL;
DO I=1 TO NL;
CALL DAXIS(L,I,DEI);
IF I=1 THEN GOTO L7;
DOMEG(L,I,*)=DOMEG(L,I-1,*)+QD(I)*DEI*(1-ZETA(I));
GOTO L4;
L7: DOMEG(L,I,*)=QD(I)*DEI*(1-ZETA(I));
L4: END;
END;
DALF=0;
DO L=1 TO NL;
DO I=1 TO NL;
IF I=1 THEN GOTO L9;
V1=DOMEG(L,I-1,*); V2=EIA(I,*); V3=OMEGA(I-1,*);
CALL DAXIS(L,I,DEI);
CALL CROVEC(V3,DEI,V4);
CALL CROVEC(V1,V2,V5);
DALF(L,I,*)=DALF(L,I-1,*)+QD(I)*(V4+V5)*(1-ZETA(I));
GOTO L5;
L9: CALL DAXIS(L,I,DEI);
DALF(L,I,*)=QD(I)*DEI*(1-ZETA(I));
L5:END;
END;
DO L=1 TO NL;
DO J=1 TO NL;
V1=R(J,*);
V2=OMEGA(J,*);
V3=DOMEG(L,J,*);
CALL DIFPOS(L,J,J,V11);
CALL CROVEC(V2,V1,V7);
CALL CROVEC(V3,V7,V8);
CALL CROVEC(V3,V1,V9);
CALL CROVEC(V2,V9,V10);
CALL CROVEC(V2,V11,V12);
CALL CROVEC(V2,V12,V13);
DGAM(L,J,*)=V8+V10+V13;
END;
END;DGAM1=0;
DO L=1 TO NL;
DO J=1 TO NL;
IF J=1 THEN GOTO L6;
V1=OMEGA(J-1,*);
V2=AR1(J-1,*);
V3=DOMEG(L,J-1,*); IN=J-1;
CALL DIFPOS(L,IN,J,V6);
CALL CROVEC(V1,V2,V7);
CALL CROVEC(V3,V7,V8);
```

```
CALL CROVEC(V3,V2,V9);
CALL CROVEC(V1,V9,V10);
CALL CROVEC(V1,V6,V11);
CALL CROVEC(V1,V11,V12);
DGAM1(L,J-1,*)=V8+V10+V12;
L6:END;
END;
DAJO,DBET=0;
DO L=1 TO NL;
DO J=1 TO NL;
IF J=1 THEN GOTO L8;
V1=DBET(L,J-1,*);
V2=DALF(L,J-1,*);
V3=R(J,*);
V4=AR1(J-1,*);
V5=THETA(J-1,*);
CALL DIFPOS(L,J,J,V6);
CALL DIFPOS(L,IN,J,V7);
V8=DOMEG(L,J-1,*);
V13=EIA(J,*);
V9=OMEGA(J-1,*);
CALL DAXIS(L,J,V10);
V11=DGAM1(L,J-1,*);
V12=DGAM(L,J,*);
V14=V3-V4;
CALL CROVEC(V2,V14,V15);
V16=V6-V7;
CALL CROVEC(V5,V16,V17);
CALL CROVEC(V8,V13,V18);
CALL CROVEC(V18,V3,V19);
CALL CROVEC(V9,V10,V20);
CALL CROVEC(V20,V3,V21);
CALL CROVEC(V9,V13,V22);
CALL CROVEC(V22,V6,V23);
DBET(L,J,*)=V1+V15+V17+QD(J)*(V19+V21+V23)*(1-ZETA(J))
+2*QD(J)*(V18+V20)*ZETA(J)-V11+V12;
DAJO(L,J,*)=-MASS(J)*DBET(L,J,*);
GOTO L16;
L8:DBET(L,1,*)=DGAM(L,1,*);
DAJO(L,1,*)=-MASS(1)*DBET(L,1,*);
L16: END;
END;
DO I=1 TO NL;
DO J=1 TO NL;
IF J=1 THEN AJ=T1;
ELSE IF J=2 THEN AJ=T2;
ELSE IF J=3 THEN AJ=T3;
ELSE IF J=4 THEN AJ=T4;
ELSE IF J=5 THEN AJ=T5;
ELSE IF J=6 THEN AJ=T6;
ELSE IF J=7 THEN AJ=T7;
```

```
ELSE IF J=8 THEN AJ=T8;
IF J=1 THEN JJ=JI(1,*);
ELSE IF J=2 THEN JJ=JI(2,*);
ELSE IF J=3 THEN JJ=JI(3,*);
IF I<=J THEN ELI=1; ELSE ELI=0;
V1=AJ(*,1);
V2=AJ(*,2);
V3=AJ(*,3);
V4=OMEGA(J,*);
CALL DOTVEC(V4,V1,D1);
CALL DOTVEC(V4,V2,D2);
CALL DOTVEC(V4,V3,D3);
XJ(1)=(JJ(2)-JJ(3))*D2*D3;
XJ(2)=(JJ(3)-JJ(1))*D3*D1;
XJ(3)=(JJ(1)-JJ(2))*D1*D2;
V5=EIA(I,*);
CALL CROVEC(V5,V1,V6);
CALL CROVEC(V5,V2,V7);
CALL CROVEC(V5,V3,V8);
DAJ(*,1)=V6*(1-ZETA(I))*ELI;
DAJ(*,2)=V7*(1-ZETA(I))*ELI;
DAJ(*,3)=V8*(1-ZETA(I))*ELI;
V9=DOMEG(I,J,*);
V10=OMEGA(J,*);
V11=DAJ(*,1);
V12=DAJ(*,2);
V13=DAJ(*,3);
CALL DOTVEC(V9,V1,C1);
CALL DOTVEC(V10,V11,C2);
DOMEGT(1)=C1+C2;
CALL DOTVEC(V9,V2,C1);
CALL DOTVEC(V10,V12,C2);
DOMEGT(2)=C1+C2;
CALL DOTVEC(V9,V3,C1);
CALL DOTVEC(V10,V13,C2);
DOMEGT(3)=C1+C2;
DXJ(1)=(DOMEGT(2)*D3+D2*DOMEGT(3))*(JJ(2)-JJ(3));
DXJ(2)=(DOMEGT(3)*D1+D3*DOMEGT(1))*(JJ(3)-JJ(1));
DXJ(3)=(DOMEGT(1)*D2+D1*DOMEGT(2))*(JJ(1)-JJ(2));
CALL MATMUL(AJ,DXJ,V14);
CALL MATMUL(DAJ,XJ,V15);
DLAMDA(I,J,*)=V14+V15;
END;
END;
DO I=1 TO NL;
DO J=I TO NL;
IF J=1 THEN TJ=T1;
ELSE IF J=2 THEN TJ=T2;
ELSE IF J=3 THEN TJ=T3;
ELSE IF J=4 THEN TJ=T4;
ELSE IF J=5 THEN TJ=T5;
```

```
ELSE IF J=6 THEN TJ=T6;
ELSE IF J=7 THEN TJ=T7;
ELSE IF J=8 THEN TJ=T8;
IF J=1 THEN T=TI1;
ELSE IF J=2 THEN T=TI2;
ELSE IF J=3 THEN T=TI3;
ELSE IF J=4 THEN T=TI4;
ELSE IF J=5 THEN T=TI5;
ELSE IF J=6 THEN T=TI6;
ELSE IF J=7 THEN T=TI7;
ELSE IF J=8 THEN T=TI8;
IF I<=J THEN ELI=1; ELSE ELI=0;
V1=EIA(I,*);
V2=TJ(*,1);
V3=TJ(*,2);
V4=TJ(*,3);
CALL CROVEC(V1,V2,V5);
CALL CROVEC(V1,V3,V6);
CALL CROVEC(V1,V4,V7);
DO L=1 TO 3;
DO N=1 TO 3;
SUM=0;
DO M=1 TO 3;
IF M=1 THEN V8=V5;
ELSE IF M=2 THEN V8=V6;
ELSE IF M=3 THEN V8=V7;
IF M=1 THEN V9=V2;
ELSE IF M=2 THEN V9=V3;
ELSE IF M=3 THEN V9=V4;
V10=V8*(1-ZETA(I))*ELI;
SUM=SUM+(V10(L)*V9(N)+V9(L)*V10(N))*JI(J,M);
END;
DTJ(L,N)=SUM;
END;
END;
V11=THETA(J,*);
CALL MATMUL(DTJ,V11,V12);
V13=DALF(I,J,*);
CALL MATMUL(T,V13,V14);
DBJO(I,J,*)=-(V12+V14)+DLAMDA(I,J,*);
END;
END;
DO L=1 TO NL;
DO I=1 TO NL;
CALL DAXIS(L,I,DEI);
V8=EIA(I,*);
IF ZETA(I)=0 THEN DO;
V1,V2=0;
DO J=I TO NL;
CALL VECTOR(J,I,RJI);
CALL DIFPOS(L,J,I,DRJI);
```

```
V3=-MASS(J)*(EATA(J,*)+WO)+G(J,*);
CALL CROVEC(RJI,V3,V4);
CALL CROVEC(DRJI,V3,V5);
V1=V1+V4+BOI(J,*)+MJG(J,*);
V6=DAJO(L,J,*);
CALL CROVEC(RJI,V6,V7);
V2=V2+V7+DBJO(L,J,*)+V5;
END;
CALL DOTVEC(DEI,V1,D1);
CALL DOTVEC(V8,V2,D2);
DSH(I,L)=-D1-D2;
END;
ELSE DO;
V1,V2=0;
DO J=I TO NL;
V1=V1-MASS(J)*(EATA(J,*)+WO)+G(J,*);
V2=V2+DAJO(L,J,*);
END;
CALL DOTVEC(DEI,V1,D1);
CALL DOTVEC(V8,V2,D2);
DSH(I,L)=-D1-D2;
END;
END;
END;
DDOMEG=0;
DDOMEG(1,1,*)=EIA(1,*)*(1-ZETA(1));
DO I=1 TO NL;
DO J=1 TO NL;
V1=EIA(J,*);
IF I=J THEN EKJ=1; ELSE EKJ=0;
IF J=1 THEN GOTO L11;
DDOMEG(I,J,*)=DDOMEG(I,J-1,*)+V1*(1-ZETA(J))*EKJ;
L11: END;
END;
DDALF(*,1,*)=0;
DO I=1 TO NL;
DO J=1 TO NL;
V1=EIA(J,*);
IF J=I THEN EKJ=1;
ELSE EKJ=0;
IF J=1 THEN GOTO L12;
V2=OMEGA(J-1,*);
V3=DDOMEG(I,J-1,*);
CALL CROVEC(V2,V1,V4);
CALL CROVEC(V3,V1,V5);
DDALF(I,J,*)=DDALF(I,J-1,*)+
+(V4*EKJ+QD(J)*V5)*(1-ZETA(J));
L12: END;
END;
DO I=1 TO NL;
DO J=1 TO NL;
```

```
IF J=1 THEN GOTO L14;
V1=DDALF(I,J-1,*);
V2=R(J,*);
V3=AR1(J-1,*);
V4=DDOMEG(I,J-1,*);
V5=OMEGA(J-1,*);
V6=DDOMEG(I,J,*);
V7=OMEGA(J,*);
IF I=J THEN EKI1=1;
ELSE EKI1=0;
IF I<= (J-1) THEN EKJ=1;
ELSE EKJ=0;
V13=V2-V3;
CALL CROVEC(V1,V13,V8);
CALL CROVEC(V5,V3,V9);
CALL CROVEC(V4,V9,V10);
CALL CROVEC(V4,V3,V11);
CALL CROVEC(V5,V11,V12);
CALL CROVEC(V7,V2,V14);
CALL CROVEC(V6,V14,V15);
CALL CROVEC(V6,V2,V16);
CALL CROVEC(V7,V16,V17);
V18=EIA(J,*);
CALL CROVEC(V5,V18,V19);
CALL CROVEC(V19,V2,V20);
V20=V20*(1-ZETA(J));
CALL CROVEC(V5,V18,V21);
V20=V21*2*ZETA(J)+V20;
CALL CROVEC(V4,V18,V22);
CALL CROVEC(V22,V2,V23);
V23=V23*QD(J)*(1-ZETA(J));
CALL CROVEC(V4,V18,V24);
V24=V24*2*ZETA(J)*QD(J);
DDBET(I,J,*)=DDBET(I,J-1,*)+V8-V10-V12+V15+V17+V20*EKI1+
(V23+V24)*EKJ;
DDAJO(I,J,*)=-MASS(J)*DDBET(I,J,*);
GOTO L13;
L14: V1=DDOMEG(I,1,*);
V2=OMEGA(1,*);
V3=R(1,*);
CALL CROVEC(V2,V3,V4);
CALL CROVEC(V1,V4,V5);
CALL CROVEC(V1,V3,V6);
CALL CROVEC(V2,V6,V7);
DDBET(I,1,*)=V5+V7;
DDAJO(I,1,*)=-MASS(1)*DDBET(I,1,*);
L13:END;
END;
DO I=1 TO NL;
DO J=1 TO NL;
IF J=1 THEN AJ=T1;
```

```
ELSE IF J=2 THEN AJ=T2;
ELSE IF J=3 THEN AJ=T3;
ELSE IF J=4 THEN AJ=T4;
ELSE IF J=5 THEN AJ=T5;
ELSE IF J=6 THEN AJ=T6;
ELSE IF J=7 THEN AJ=T7;
ELSE IF J=8 THEN AJ=T8;
IF J=1 THEN TJ=TI1;
ELSE IF J=2 THEN TJ=TI2;
ELSE IF J=3 THEN TJ=TI3;
ELSE IF J=4 THEN TJ=TI4;
ELSE IF J=5 THEN TJ=TI5;
ELSE IF J=6 THEN TJ=TI6;
ELSE IF J=7 THEN TJ=TI7;
ELSE IF J=8 THEN TJ=TI8;
V1=AJ(*,1);
V2=AJ(*,2);
V3=AJ(*,3);
V4=DDOMEG(I,J,*);
V5=OMEGA(J,*);
CALL DOTVEC(V4,V1,D1);
CALL DOTVEC(V4,V2,D2);
CALL DOTVEC(V4,V3,D3);
CALL DOTVEC(V5,V1,C1);
CALL DOTVEC(V5,V2,C2);
CALL DOTVEC(V5,V3,C3);
DDXJ(1)=(D2*C3+C2*D3)*(JI(J,2)-JI(J,3));
DDXJ(2)=(D3*C1+C3*D1)*(JI(J,3)-JI(J,1));
DDXJ(3)=(D1*C2+C1*D2)*(JI(J,1)-JI(J,2));
CALL MATMUL(AJ,DDXJ,V6);
DDLAMD(I,J,*)=V6;
V7=DDALF(I,J,*);
CALL MATMUL(TJ,V7,V8);
DDBJO(I,J,*)=-V8+V6;
END;
END;
DO L=1 TO NL;
DO I=1 TO NL;
V1=EIA(I,*);
V2=0;
IF ZETA(I)=0 THEN DO;
DO J=I TO NL;
CALL VECTOR(J,I,RJI);
V3=DDAJO(L,J,*);
CALL CROVEC(RJI,V3,V4);
V2=V2+V4+DDBJO(L,J,*);
END;
END;
ELSE DO;
DO J=I TO NL;
V2=V2+DDAJO(L,J,*);
```

```
END;
END;
CALL DOTVEC(V1,V2,D);
DDSH(I,L)=-D;
END;
END;
DAXIS:PROC(L1,L2,V);
DCL (V(3),V1(3),V2(3)) FLOAT(16);
DCL (L1,L2) FIXED BIN;
IF (L1>=L2 | ZETA(L1)=1) THEN DO;
V=0; GOTO LA1; END;
V1=EIA(L1,*); V2=EIA(L2,*);
CALL CROVEC(V1,V2,V);
V=V*(1-ZETA(L1));
LA1: END DAXIS;
DIFPOS: PROC(L1,L2,L3,V);
DCL (V(3),RIJ(3),RIL(3),V1(3),V2(3),V3(3)) FLOAT(16);
DCL (L1,L2,L3,ELI,ELJ) FIXED BIN;
IF L1<=L2 THEN ELI=1;
ELSE ELI=0;
IF L1<=(L3-1) THEN ELJ=1;
ELSE ELJ=0;
CALL VECTOR(L2,L3,RIJ);
CALL VECTOR(L2,L1,RIL);
V1=EIA(L1,*);
CALL CROVEC (V1,RIJ,V2);
CALL CROVEC (V1,RIL,V3);
V=(V2*ELJ+V3*(ELI-ELJ))*(1-ZETA(L1)) +
V1*(ELI-ELJ)*ZETA(L1);
END DIFPOS;
END LINEAR;
VECTOR:PROC(N1,N2,V);
DCLV(3) FLOAT(16);
DCL (N1,N2,L) FIXED BIN;
IF N1=N2 THEN DO;
V=R(N1,*);
RETURN;
END;
IF N1< (N2-1) THEN DO;
V=AR1(N1,*);
DO L=N1+1 TO N2;
IF L=N2 THEN GOTO LBL25;
V=V+AR1(L,*)-R(L,*);
LBL25: END;
RETURN;
END;
IF N1=(N2-1) THEN DO;
V=AR1(N1,*); RETURN; END;
IF N1=(N2+1) THEN DO;
V=-AR1(N2,*)+R(N1,*)+R(N2,*);
RETURN; END;
```

```
ELSE IF N1>N2 THEN DO;
V=R(N2,*);
DO L=N2 TO N1;
IF L=N1 THEN GOTO LBL26;
V=V-AR1(L,*)+R(L+1,*);
LBL26:END;
RETURN; END;
END VECTOR;
ORIEN:PROC;
DCL (TIP,UTI(3,3),V1(3),V2(3),V4(3)) FLOAT(16);
DCL (L,J) FIXED BIN;
EIA(1,*),E=BEO;
DO L=1 TO NL;
IF L=1 THEN T1=UT1;
ELSE IF L=2 THEN T2=UT2;
ELSE IF L=3 THEN T3=UT3;
ELSE IF L=4 THEN T4=UT4;
ELSE IF L=5 THEN T5=UT5;
ELSE IF L=6 THEN T6=UT6;
ELSE IF L=7 THEN T7=UT7;
ELSE IF L=8 THEN T8=UT8;
END;
DO L=1 TO NL;
TIP=Q(L);  IF TIP=0 THEN GOTO LBLA;
DO J=L TO NL;
IF J=1 THEN UTI=T1;
ELSE IF J=2 THEN UTI=T2;
ELSE IF J=3 THEN UTI=T3;
ELSE IF J=4 THEN UTI=T4;
ELSE IF J=5 THEN UTI=T5;
ELSE IF J=6 THEN UTI=T6;
ELSE IF J=7 THEN UTI=T7;
ELSE IF J=8 THEN UTI=T8;
IF ZETA(L)=0 THEN DO;
CALL ROTATE (UTI,E,TIP,TI);
IF J=1 THEN T1=TI;
ELSE IF J=2 THEN T2=TI;
ELSE IF J=3 THEN T3=TI;
ELSE IF J=4 THEN T4=TI;
ELSE IF J=5 THEN T5=TI;
ELSE IF J=6 THEN T6=TI;
ELSE IF J=7 THEN T7=TI;
ELSE IF J=8 THEN T8=TI;
END;
END;
LBLA: IF L=1 THEN TI=T1;
ELSE IF L=2 THEN TI=T2;
ELSE IF L=3 THEN TI=T3;
ELSE IF L=4 THEN TI=T4;
ELSE IF L=5 THEN TI=T5;
ELSE IF L=6 THEN TI=T6;
```

```
ELSE IF L=7 THEN TI=T7;
ELSE IF L=8 THEN TI=T8;
IF L=NL THEN GOTO LBLB;
V1=LE1(L,*);
CALL MATMUL (TI,V1,E); EIA(L+1,*)=E;
LBLB: END;
DO L=1 TO NL;
V1=LR(L,*); V3=LR1(L,*);
IF L=1 THEN TI=T1;
ELSE IF L=2 THEN TI=T2;
ELSE IF L=3 THEN TI=T3;
ELSE IF L=4 THEN TI=T4;
ELSE IF L=5 THEN TI=T5;
ELSE IF L=6 THEN TI=T6;
ELSE IF L=7 THEN TI=T7;
ELSE IF L=8 THEN TI=T8;
CALL MATMUL (TI,V1,V2);
CALL MATMUL (TI,V3,V4);
IF ZETA(L)=1 THEN R(L,*)=V2+Q(L)*EIA(L,*);
ELSE R(L,*)=V2;
AR1(L,*)=V4;
END;
END ORIEN;
ROTATE: PROC(A,EX,Q,RA);
DCL I FIXED BIN;
DCL (A(3,3),EX(3),Q,RA(3,3),CQ,SQ,V1(3),V2(3),D)
FLOAT(16);
CQ=COS(Q); SQ=SIN(Q);
DO I=1 TO 3;
V1=A(*,I);
CALL CROVEC (EX,V1,V2);
CALL DOTVEC (EX,V1,D);
RA(*,I)=V1*CQ+(1-CQ)*D*EX+V2*SQ;
END;
END ROTATE;
MATMUL: PROC (M1,M2,M3);
DCL (M1(3,3),M2(3),M3(3),SUM) FLOAT(16);
DCL (I,K) FIXED BIN;
DO I=1 TO 3;
SUM=0; DO K=1 TO 3;
SUM=SUM+M1(I,K)*M2(K); END;
M3(I)=SUM; END;
END MATMUL;
DOTVEC: PROC(V1,V2,RD);
DCL (V1(3),V2(3),RD) FLOAT(16);
DCL I FIXED BIN;
RD=0;
DO I=1 TO 3; RD=RD+V1(I)*V2(I); END;
END DOTVEC;
CROVEC: PROC (V1,V2,V3);
DCL (V1(3),V2(3),V3(3)) FLOAT(16);
```

```
V3(1)=V1(2)*V2(3)-V1(3)*V2(2);
V3(2)=V1(3)*V2(1)-V1(1)*V2(3);
V3(3)=V1(1)*V2(2)-V1(2)*V2(1);
END CROVEC;
UNIVEC: PROC (V1,V2,S);
DCL (V1(3),V2(3),MAG) FLOAT(16);
DCL S FIXED BIN;
MAG=SQRT(V1(1)**2+V1(2)**2+V1(3)**2);
V2=V1/MAG*S;
END UNIVEC;
MULVET: PROC(M1,V1,V2,L1,L2);
DCL(M1(*,*),V1(*),V2(*),SUM) FLOAT(16);
DCL(I,J,L1,L2) FIXED BIN;
DO I=1 TO L1;
SUM=0; DO J=1 TO L2;
SUM=SUM+M1(I,J)*V1(J); END;
V2(I)=SUM; END;
END MULVET;
VETMUL: PROC(V1,M1);
DCL (V1(9),M1(9,9)) FLOAT(16);
DCL (I,J) FIXED BIN;
DO I=1 TO 9;  DO J=1 TO 9;
M1(I,J)=V1(I)*V1(J); END; END;
END VETMUL;
MULMAT: PROC(M1,M2,M3,L1,L2,L3);
DCL(M1(*,*),M2(*,*),M3(*,*),SUM) FLOAT(16);
DCL(I,J,K,L1,L2,L3) FIXED BIN;
DO I=1 TO L1; DO J=1 TO L3; SUM=0;
DO K=1 TO L2; SUM=SUM+M1(I,K)*M2(K,J); END;
M3(I,J)=SUM;
END; END;
END MULMAT;
MATINV: PROC(M,MINV,L);
DCL(BS(L),BX(L)) FLOAT(16);
DCL(M(*,*),MINV(*,*)) FLOAT(16);
DCL (I,J,L) FIXED BIN;
DO I=1 TO L;
BS=0; BS(I)=1;
CALL SOLLIN(M,BS,BX,L);
MINV(*,I)=BX;
END;
END MATINV;
DOT:PROC(V1,V2,S);
DCL (V1(9),V2(9),S,SUM) FLOAT(16);
DCL K FIXED BIN;  SUM=0;
DO K=1 TO 9; SUM=SUM+V1(K)*V2(K); END; S=SUM;
END DOT;

DISCRET:PROC;
DCL (TEMP(3,3),TEMP1(3,3),V1(3),HP(3,3),A21(3,3),
A22(3,3),A(6,6),B(6,3)) FLOAT(16);
```

```
DCL (I,J) FIXED BIN;
I=3;
CALL MATINV(H,HINV,I);
DO I=1 TO NL;
TEMP=DH(I,*,*);
CALL MULVET(TEMP,QDD,V1,NL,NL);
TEMP1(*,I)=V1;
END;
HP=TEMP1+DSH;
CALL MULMAT(HINV,HP,A21,NL,NL,NL);
A21=-A21;
CALL MULMAT(HINV,DDSH,A22,NL,NL,NL);
A22=-A22;
A=0; B=0;
DO I=1 TO NL;
DO J=1 TO NL;
A(I+NL,J)=A21(I,J); B(I+NL,J)=HINV(I,J);
A(I+NL,J+NL)=A22(I,J); END; A(I,I+NL)=1; END;
NL2=NL*2;
CALL EXPON(A,B,DELT,NL2,NL,CONF,CONG);
EXPON:PROC(A1,B1,T1,L1,L2,F1,G1);
DCL (A1(6,6),AT(6,6),AT1(6,6),AT2(6,6),AT3(6,6),
ID(6,6),
B1(6,3),G1(6,3),F1(6,6),T1,RJ,ET(6,6)) FLOAT(16);
DCL(L1,L2,J,NL2,NL) FIXED BIN;
ID=0; NL2=L1; NL=L2;
DO J=1 TO NL2; ID(J,J)=1; END;
AT=A1*T1; AT1=ID+AT/16;
DO J=3 TO 15;  RJ=18-J;
AT2=AT/RJ;
CALL MULMAT(AT2,AT1,AT3,NL2,NL2,NL2);
AT1=AT3+ID;
END;
AT2=A1*(T1**2)/2;
CALL MULMAT(AT2,AT1,ET,NL2,NL2,NL2);
ET=ID*T1 + ET;
CALL MULMAT(A,ET,F1,NL2,NL2,NL2);
F1=F1+ID;
CALL MULMAT(ET,B1,G1,NL2,NL2,NL);
END EXPON;
END DISCRET;
SOLLIN: PROC (A,B,X,N);
DCL (A(*,*),B(*),X(*)) FLOAT(16);
DCL (AUG(9,10),PIVOT,TEMP,Q1) FLOAT(16);
DCL (I,K,L,IPIVOT,IP1,K1,K2,K3,
K4,K5,N,NM1,NP1) FIXED BIN;
NP1=N+1; NM1=N-1;
DO I=1 TO N;
DO K=1 TO N;
AUG(I,K)=A(I,K); END; END;
DO I=1 TO N; AUG(I,NP1)=B(I); END;
```

```
DO K=1 TO NM1;
PIVOT =0;
DO L=K TO N;
TEMP=ABS(AUG(L,K));
IF PIVOT >= TEMP THEN GOTO LB1;
PIVOT=TEMP;
IPIVOT=L;
LB1:END;
IF PIVOT=0 THEN GOTO LB2;
ELSE IF IPIVOT=K THEN GOTO LB3;
DO K1=K TO NP1;
TEMP=AUG(K,K1);
AUG(K,K1)=AUG(IPIVOT,K1);
AUG(IPIVOT,K1)=TEMP;
END;
LB3: IP1=K+1;
DO K2=IP1 TO N;
Q1=-AUG(K2,K)/AUG(K,K);
AUG(K2,K)=0;
DO K3=IP1 TO NP1;
AUG(K2,K3)=Q1*AUG(K,K3)+AUG(K2,K3);
END;
END;
END;
IF AUG(N,N)=0 THEN GOTO LB2;
X(N)=AUG(N,NP1)/AUG(N,N);
DO K4=1 TO NM1;
Q1=0;
DO K5=1 TO K4;
Q1=Q1+AUG(N-K4,NP1-K5)*X(NP1-K5);
END;
X(N-K4)=(AUG(N-K4,NP1)-Q1)/AUG(N-K4,N-K4);
END;
GOTO LB4;
LB2: PUT SKIP LIST ('ERROR IN SOLLIN');
RETURN;
LB4: END SOLLIN;
TRANSP: PROC(M1,M2,L1,L2);
DCL (M2(*,*),M1(*,*)) FLOAT(16);
DCL (I,J,L1,L2) FIXED BIN;
DO I=1 TO L2; DO J=1 TO L1;
M2(I,J)=M1(J,I); END; END;
END TRANSP;
NEXSTP: PROC(X,U,C,X1);
DCL (X(6),C(6),U(3),X1(6),V1(6),
V2(6),V3(6)) FLOAT(16);
DCL (K,L) FIXED BIN;
K=6; L=6;
CALL MULVET(AD,X,V1,K,L);
K=6; L=3;
CALL MULVET(BD,U,V2,K,L);
```

```
K=6; L=6;
CALL MULVET(ET,C,V3,K,L);
X1=V1+V2+V3;
END NEXSTP;
DISCR2:PROC(A,B);
DCL (A(6,6),AT(6,6),AT1(6,6),AT2(6,6),AT3(6,6),
B(6,3),RJ,ID(6,6)) FLOAT(16);
DCL(K1,K2,J,K3) FIXED BIN;
ID=0;
DO J=1 TO 6; ID(J,J)=1; END;
AT=A*DELT; AT1=ID+AT/36;
DO J=3 TO 35;  RJ=38-J;
AT2=AT/RJ;
K1=6; K2=6; K3=6;
CALL MULMAT(AT2,AT1,AT3,K1,K2,K3);
AT1=AT3+ID;
END;
AT2=A*(DELT**2)/2;
K1=6; K2=6; K3=6;
CALL MULMAT(AT2,AT1,ET,K1,K2,K3);
ET=ID*DELT + ET;
K1=6; K2=6; K3=6;
CALL MULMAT(A,ET,AD,K1,K2,K3);
AD=AD+ID;
K1=6; K2=6; K3=3;
CALL MULMAT(ET,B,BD,K1,K2,K3);
END DISCR2;
JQT=JQ;
EJQ=JQT-AJQ;
TTOR=NTOR+USTR;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,JQT(I,1)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,AJQ(I,1)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,EJQ(I,1)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,JQT(I,2)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,AJQ(I,2)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,EJQ(I,2)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,JQT(I,3)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,AJQ(I,3)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,EJQ(I,3)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,NTOR(I,1)); END;
DO I=1 TO NSTEP+1; IF=I-1;
```

```
PUT SKIP LIST (IF,USTR(I,1)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,TTOR(I,1)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,NTOR(I,2)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,USTR(I,2)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,TTOR(I,2)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,NTOR(I,3)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,USTR(I,3)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,TTOR(I,3)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,TAUM(I,1)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,TAUM(I,2)); END;
DO I=1 TO NSTEP+1; IF=I-1;
PUT SKIP LIST (IF,TAUM(I,3)); END;
END LINKAS;
```

The vita has been removed from the scanned document