TRAJECTORY TRACKING CONTROL OF ROBOTIC MANIPULATORS WITH FLEXIBLE JOINTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

İNANÇ ERDAĞI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF MECHANICAL ENGINEERING

SEPTEMBER 1995

Approval of the Graduate School of Natural and Applied Sciences

[illegible]

Prof. Dr. İsmail Tosun
Director
for

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Rüknettin Oskay
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Kemal Özgören
Co-Supervisor

[illegible]

Assoc. Prof. Dr. S. Kemal İder
Supervisor

Examining Committee Members

Prof. Dr. Kemal Özgören	(Chairman)	[illegible]
Assoc. Prof. Dr. S. Kemal İder	(Supervisor)	[illegible]
Assoc. Prof. Dr. Tuna Balkan		[illegible]
Assoc. Prof. Dr. Aydan Erkmen		[illegible]
Assoc. Prof. Dr. Reşit Soylu		[illegible]

ABSTRACT

TRAJECTORY TRACKING CONTROL OF ROBOTIC MANIPULATORS WITH FLEXIBLE JOINTS

Erdağı , İnanç

M.S. , Department of Mechanical Engineering

Supervisor : Assoc. Prof. Dr. S. Kemal İder

Co–Supervisor : Prof. Dr. Kemal Özgören

September 1995 , 158 pages

The purpose of this thesis is the development of a method for the position control of a robotic manipulator with flexible joints.

The dynamic model of a general flexible joint manipulator is formulated utilizing Lagrange's equations. Then it is shown that this model can be simplified to a fourth order system without loosing its generality.

It is seen that direct extension of inverse dynamics control to flexible joint robots leads to a very complicated control law due to differentiating the equation of motion twice and measuring the accelerations and jerks in addition to the positions and velocities.

In this thesis, an algorithm based on solving the inverse dynamics equations at the acceleration level is developed. Since the control forces can not affect the end-effector accelerations instantaneously due to the elastic media, the acceleration level dynamic equations are singular. To account for the higher order derivative information, methods for solving differential/algebraic equations are utilized. The control law developed leads to a linear and decoupled fourth order error dynamics by feedback of link positions and velocities only.

Finally by using a computer program, written in FORTRAN, control simulations for planar one link and spatial 3-R manipulators are presented.

Keywords : Flexible Joint Manipulators, Position Control, Inverse Dynamics Control, Singular System of Differential Equations.

ÖZ

ELASTİK EKLEMLİ BİR ROBOT KOLUN KONUM KONTROLÜ

Erdağı , İnanç

Yüksek Lisans , Makine Mühendisliği Bölümü

Tez Yöneticisi : Doç. Dr. S. Kemal İder

Ortak Tez Yöneticisi : Prof. Dr. Kemal Özgören

Eylül 1995 , 158 sayfa

Bu tezin amacı elastik eklemli bir robot kolun konum kontrolü için bir metod geliştirmektir.

Elastik eklemli genel bir robotun dinamik modeli, Lagrange denklemleri kullanılarak formüle edilmiştir. Daha sonra, bu modelin genelliğini kaybetmeden, daha basit bir biçimde dördüncü dereceden ifade edilebileceği gösterilmiştir.

Ters dinamik kontrol yasasını, elastik eklemli robotlara direk uygulamanın çok karmaşık bir kontrol yöntemine yol açtığı görülmüştür. Bunun sebebi hareket denklemlerinin iki defa türevinin alınması ve pozisyon ve hız ölçümlerine ek olarak ivme ve ivme hızlarının da ölçülmesidir.

Bu tezde, ters dinamik denklemlerini ivme seviyesinde çözen bir algoritma geliştirilmiştir. Fakat, elastik ortamdan ötürü, kontrol kuvvetleri robot kolun uç noktasının ivmesini anında etkileyemediği için, ivme seviyesindeki dinamik denklemler tekil durumdadır. Daha yüksek türev düzeyindeki bilgileri hesaba katabilmek için, diferansiyel / cebirsel denklemlerin çözümü için kullanılan metodlardan yaralanılmıştır. Geliştirilmiş olan kontrol yasası, sadece uzuv konumlarının ve hızlarının geribeslemesi ile doğrusal ve ayrışık, dördüncü dereceden bir hata dinamiği sonucunu vermiştir.

Son olarak, FORTRAN diliyle yazılmış bir bilgisayar programı kullanılarak 1-uzuvlu planar ve 3-uzuvlu uzaysal robot kollar için yapılmış kontrol simülasyonları sunulmuştur.

Anahtar Kelimeler : Elastik Eklemli Robot Kollar, Konum Kontrolü, Ters Dinamik Kontrol, Tekil Diferansiyel Denklem Sistemleri

ACKNOWLEDGMENTS

The author wishes to express his deepest gratitude and appreciation to his supervisor Assoc. Prof. Dr. S. Kemal İder, for his continuous help, kind supervision and guidance throughout the thesis.

The author is also indebted to Prof. Dr. Kemal Özgören, for his valuable comments and support.

Finally the author would like to thank to his family for their continuous help and encourage and to his friends Assistant S. Savaş Kırımlıoğlu and Assistant S. Emrah Halıcı for their friendship and various helps throughout the thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

θ	Vector of joint variables
θ_i	Joint variable of the ith joint
Φ	Vector of angles of rotation of the actuators
Φ_i	Angle of rotation of the ith actuator
Ψ	Vector of angles of rotation of actuators after gear reduction
Ψ_i	Angle of rotation of the ith actuator after the gear reduction
r_i	Gear ratio of the ith actuator
KE_i^L	Kinetic Energy of the ith link
m_i	mass of the ith link
$\boldsymbol{V}_{ci}^L$	Mass center velocity vector of the ith link expressed in base frame
ω_i^L	Angular velocity of the ith link expressed in base frame
$\boldsymbol{J}_i^L$	Moment of inertia matrix of the ith link
$\boldsymbol{T}_i$	Transformation matrix from reference frame fixed to the ith link to the base frame
$\underline{\boldsymbol{J}}_i^L$	Moment of inertia matrix of the ith link expressed in body frame
$\boldsymbol{W}_{ij}^L$	Velocity influence coefficient vectors of the ith link
Ω_{ij}^L	Angular velocity influence coefficient vectors of the ith link
KE_i^A	Kinetic Energy of the ith actuator
m_i^A	mass of the ith actuator

$\boldsymbol{V}_{ci}^{A}$	Mass center velocity of the ith actuator expressed in base frame
ω_i^{A}	Angular velocity of the ith actuator expressed in base frame
$\boldsymbol{J}_i^{A}$	Moment of inertia matrix of the ith actuator
$\boldsymbol{T}_i^{A}$	Transformation matrix from reference frame fixed to the ith rotor to the base frame
$\underline{\boldsymbol{J}}_i^{A}$	Moment of inertia matrix of the ith actuator expressed in body frame
$\boldsymbol{W}_{ij}^{A}$	Velocity influence coefficient vectors of the ith actuator
α_i	Unit vector along the direction of rotation of the ith actuator in the link frame on which the actuator is placed (i-1th link)
J_i^{A}	Moment of inertia of the rotor about the rotation axis
PE_i^{L}	Potential Energy of the ith link
$\boldsymbol{g}$	Gravitational acceleration vector
$\boldsymbol{p}_i^{L}$	Mass center position vector of the ith link expressed in base frame
PE_i^{A}	Potential Energy of the ith actuator
$\boldsymbol{p}_i^{A}$	Mass center position vector of the ith actuator expressed in base frame
K_i	Joint spring constant of the spring of the ith actuator
$\boldsymbol{K}$	A diagonal matrix whose diagonal elements are K_i
$\boldsymbol{R}$	A diagonal matrix whose diagonal elements are r_j
T_i	Torque applied by the ith actuator
$\boldsymbol{J}^{A}$	A diagonal matrix whose diagonal elements are J_i^{A}
$\boldsymbol{B}$	Vector whose elements are B_i and B_i^{r}
B_i^{r}	Torsional viscous damping coefficient of the ith rotor
$\boldsymbol{T}$	Vector of control actuator torques
$\boldsymbol{\Gamma}$	A diagonal matrix whose diagonal elements are $r_i^2 J_i^{A}$

$\boldsymbol{x}$	Cartesian end-effector position vector
$\boldsymbol{J}$	Manipulator jacobian matrix
$\boldsymbol{u}$	Control input vector
$\boldsymbol{e}$	Cartesian end-effector position error vector
$\boldsymbol{z}$	Calculated acceleration vector
C_1, C_2, C_3, C_4	Diagonal constant feedback gain matrices
h	integration time step
k	integration time step number

CHAPTER I

INTRODUCTION

1.1 Literature Survey

Kinematics, dynamics and control of industrial robots have been studied extensively under the assumption that the robot structure can be modeled as rigid. This brings the advantage of using angular encoders at the actuator shafts or at the joints to get information on the actual position of the end effector in world coordinates in a purely geometric manner. Therefore the control device can use this information directly.

However, in fact, most present day robots exhibit structural flexibility. The existence of flexibilities in the robot structure limits its ability to perform high-precision manipulation. Although in principle, robot flexibility is an unwanted feature, for compliance with the environment, or to deal with an accidental collision, some initial passive flexibility may be especially introduced to the robot structure.

Experimental results reveal that, for a wide variety of robots, joint flexibility is the principal source contributing to overall robot flexibility (Rivin [12]). Flexibilities of the transmission elements like harmonic drives, couplings, belt drives and thin shafts are the reasons of joint flexibility. As a result joint flexibility should be taken into account in the modeling and design of robot controllers if high performance is to be achieved. When joint flexibility is taken into consideration, an n-link robot has 2n degrees of freedom where there are only n actuators to control the motion of the end-effector.

For the modeling and control of robot manipulators with flexible joints, there are different types of strategies developed by various authors.

One approach to the control of flexible joint robots is feedback linearization that yields decoupled linear subsystems for each degree of freedom (Spong [13]; De Luca [2]; Forrest and Babcock [4]). Feedback linearization of flexible joint robots is an extension of inverse dynamics control well known for rigid robots to flexible joint robots. The feedback linearization technique have been criticized for its computational complexity due to differentiating the equations of motion twice, and for its need of measuring the link accelerations and jerks (Spong [13]; Jankowski and Brussel [7]).

Singular perturbation approach has the advantage of order reduction and separation of time scales. In this approach, the difficulties stated above are avoided by decomposing the system into fast subsystem (flexible joints) and slow subsystem (rigid manipulator) (Spong [13]; Chow and Kokotovic [1]; Gogate and Lin [6]). The model obtained exhibits a boundary layer phenomena. The model order is lowered by first neglecting the fast phenomena (boundary layer). Hence first the dynamics of the slow subsystem is considered. Then slow variables are assumed to be constant at the boundary layer, and by boundary layer corrections (calculated in a separate time scale) the effects of fast phenomena are reintroduced. The whole system can be investigated by examining the subsystems. However this approach is valid only when the joint springs are sufficiently stiff so that the flexible joint dynamics is much faster than the manipulator dynamics.

Spong [13], developed the dynamic equations of flexible joints by making the assumptions that the K.E. of the actuators are only due to their own rotation about their rotation axes, and the rotor/gear inertia is symmetric about the rotor axis. The joint elasticity is modeled by a linear torsional spring, placed after the gear reduction. The dynamic equations are obtained from Lagrange's formula . Spong, presented feedback linearization and an integral manifold method as two alternative control strategies. Since the direct relation between the input (motor torques) and output (end-effector position) is fourth order, the feedback linearization is achieved by differentiating the equations of motion twice and by using feedback of link positions, velocities, accelerations and jerks. The integral manifold method is based on singular perturbation approach where the system is divided into slow and fast subsystems after a coordinate transformation. The slow subsystem has the same order with the rigid system, but more accurate in modeling the flexible system than the rigid model. Then fast subsystem is reintroduced by boundary layer corrections, and a control torque which is a combination of controls separately for fast and slow subsystem is applied to the system.

Forrest and Babcock [4] applied inverse dynamics approach to a 2-dof spatial manipulator composed of one prismatic and one revolute joints.

Gogate & Lin [6] also used singular perturbation approach. The variable representing the slow subsystem is the angle between the links and the fast variable is the spring force at the joints. Dynamic equations of the system are obtained by using the Hamilton's Principle. Necessary torques for the control of flexible joint manipulator are obtained by adding a correction term to the torques necessary for the control of rigid robot. By using this torque and assuming that the springs at the joints have a large stiffness, new dynamic equations of the system are obtained. To lower the order of the model, the fast phenomena is neglected. The effects of the fast phenomena are reintroduced to the system as boundary layer corrections, after rewriting the dynamic equations of the system by using new variables, obtained from rearrangement of the previously used system variables.

A discrete time inverse dynamics algorithm is developed by Jankowski and Brussel [7]. A control law involving solution of a set of DAE (Differential Algebraic Equations) is formulated. Since analytical solution of the DAE set is complicated a numerical solution procedure is suggested . A three step numerical solution process is formulated where the infinitely large torques (due to inconsistent conditions and jumps in command accelerations) obtained in the first two steps are disregarded. Hence the control algorithm is valid only if the desired trajectory is smooth everywhere up to the third derivative.

Jankowski and ElMaraghy [8] suggest an inverse dynamics control approach for hybrid position force control of flexible joint manipulators. However, for position control part of the developed control algorithm, measurements of accelerations and jerks are again necessary.

1.2 Object of this thesis

The aim of this study is to develop a control law for position control of flexible joint robots that avoid the difficulties of the existing algorithms. Singular perturbation approach is not considered, since this approach is limited in its applicability due to the assumption that the joint springs are sufficiently stiff.

Direct extension of inverse dynamics control to flexible joint manipulators requires the analytical differentiation of the equations of motion twice and measurements of link accelerations and jerks in addition to the positions and velocities. Hence the controller becomes very complicated causing serious problems for real time implementation.

In this study, in order to avoid the difficulties related to the complexity of the control law, an inverse dynamics control approach that solves the singular inverse dynamics equations at the acceleration level is formulated. It is shown that the dynamic equations of the system form a singular set of differential equations since the control forces and the task accelerations are noncausal due to the elastic media. To account for the higher order derivative information, methods for solving differential / algebraic equations are utilized. The only quantities that should be measured are the joint angles and the joint velocities. The control law developed decouples and linearizes the system, yielding an asymptotically stable fourth order error dynamics. The proposed method is able to track the desired trajectories even if there are initial and intermediate jumps in position, velocity, acceleration and jerks.

CHAPTER II

DYNAMIC MODEL OF A FLEXIBLE JOINT ROBOT

In this chapter a general description of the dynamics of an n-link manipulator with flexible revolute joints, each actuated by a motor through a speed reducer (gear box), will be presented.

Elasticity of the joints is due to many reasons depending on the type of the transmission used. Harmonic drives are commonly used in present day robots and have torsional elasticity which are specified in the manufacturers catalogues. Other components that cause elasticity include elastic couplings, belts and thin shafts.

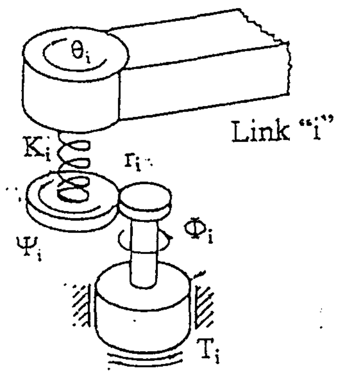


Figure 2.1 Model of the i'th flexible joint

The joint elasticity is modeled as a linear torsional spring with stiffness K_i as shown in Figure 2.1. Because of the additional degrees of freedom introduced due to the elastic coupling of the actuator shafts to the links, the rotors of the actuators are modeled as additional rigid bodies.

Hence a robot with flexible joints consist of 2n rigid bodies: n links and n actuators and has 2n degrees of freedom. Let $\theta = [\theta_1 , ... , \theta_n]^T$ be the vector of joint variables, where θ_i defines the angular position of the ith link with respect to the (i-1)th one and let $\Phi = [\Phi_1 , ... ,\Phi_n]^T$ be the vector of angles of rotation of actuator rotors with Φ_i describing the position of the ith rotor with respect to the (i - 1)th link. At this point another actuator variable is introduced:

$$\Psi_i = \Phi_i / r_i \tag{2.1}$$

where r_i is the gear ratio of the ith transmission. If direct drive actuators are used, then, $r_i = 1$, $(i = 1 , ... , n)$, and $\Psi_i = \Phi_i$.

Moreover, for the formulation of the dynamic model of a robot with flexible joints, several commonly used assumptions will be adopted. First, a symmetric rotor mass distribution around its rotation axis is assumed. Second, damping properties of flexible transmission elements will not be taken into consideration. Finally, it is assumed that the links are rigid.

At the following pages, by using the generalized coordinates stated above the energy functions of the bodies of the manipulator will be written. Then, those functions will be used in Lagrange's Equation to obtain the dynamic equations of the elastic joint manipulator.

2.1 Kinetic Energy of a Link :

Kinetic Energy of a link can be formulated by following the common procedure as follows ;

$$KE_i^L = \frac{1}{2}\, m_i^L\, (V_{ci}^L)^T\, V_{ci}^L + \frac{1}{2}\, (\omega_i^L)^T\, J_i^L\, \omega_i^L \qquad i = 1, \ldots\ldots, n \qquad (2.2)$$

where

m_i^L	is the mass of the ith link
V_{ci}^L	is the mass center velocity vector of the ith link expressed in base frame (fixed reference frame)
J_i^L	is the moment of inertia matrix of the ith link expressed in base frame such that

$$J_i^L = T_i(\theta)\, \underline{J}_i^L\, T_i^T(\theta) \qquad (2.3)$$

where

$T_i(\theta)$ is the transformation matrix from the reference frame fixed to the ith link to the base frame and $\underline{J}_i^L$ is the moment of inertia matrix of the ith link expressed in body frame

ω_i^L is the angular velocity of the ith link expressed in base frame

Translational and angular velocities of a link can be written as follows

$$V_{ci}^L = \sum_{j=1}^{n} W_{ij}^L(\theta)\,\dot{\theta}_j \tag{2.4}$$

$$\omega_i^L = \sum_{j=1}^{n} \Omega_{ij}^L(\theta)\,\dot{\theta}_j \tag{2.5}$$

where

$W_{ij}(\theta)^L$ is the velocity influence coefficient vector and $\Omega_{ij}(\theta)^L$ is the angular velocity influence coefficient vector

When (2.4) and (2.5) are put into equation (2.2) the result is

$$KE_i^L = \frac{1}{2} m_i^L \left(\sum_{j=1}^{n} (W_{ij}^L)^T \dot{\theta}_j \right) \left(\sum_{k=1}^{n} W_{ik}^L \dot{\theta}_k \right) + \frac{1}{2} \left(\sum_{j=1}^{n} (\Omega_{ij}^L)^T \dot{\theta}_j \right) J_i^L \left(\sum_{k=1}^{n} \Omega_{ik} \dot{\theta}_k \right) \tag{2.6}$$

Equation (2.6) can be written more compactly as

$$KE_i^L = \sum_{j=1}^{n} \sum_{k=1}^{n} m_{ijk}^L \; \dot{\theta}_j \; \dot{\theta}_k \qquad (2.7)$$

where

$$m_{ijk}^L = \frac{1}{2} m_i^L \, (W_{ij}^L)^T \, W_{ik}^L + \frac{1}{2} (\Omega_{ij}^L)^T \, J_i^L \, \Omega_{ik}^L \qquad (2.8)$$

It is clear that the total KE of the links is

$$KE^L = \sum_{i=1}^{n} KE_i^L \qquad (2.9)$$

2.2 Kinetic Energy of an Actuator:

For an actuator rotor (where the term actuator is used to mean actuator rotor for short writing), the KE expression and necessary equations for Lagrange's equation can be obtained as follows

$$KE_i^A = \frac{1}{2} m_i^A \, (V_{ci}^A)^T \, V_{ci}^A + \frac{1}{2} (\omega_i^A)^T \, J_i^A \, \omega_i^A \qquad (2.10)$$

where

m_i^A	is the mass of the ith actuator
V_{ci}^A	is the mass center velocity vector of the ith actuator expressed in base frame (fixed coordinates)

J_i^A is the moment of inertia matrix of the ith actuator expressed in base frame such that

$$J_i^A = T_i^A(\theta)\,\underline{J}_i^A\,(T_i^A(\theta))^T \tag{2.11}$$

where

$T_i^A(\theta)$ is the transformation matrix from the frame fixed to the ith rotor to the base frame and $\underline{J}_i^A$ is the moment of inertia matrix of the ith actuator expressed in actuator frame

ω_i^A is the angular velocity of the ith actuator expressed in base frame

Note that the angular velocity of the ith actuator is equal to the angular velocity of the previous link (i-1th link), plus the angular velocity of the actuator relative to the i-1th link on which the actuator is mounted.

The translational and angular velocities of an actuator can be written as follows

$$V_{ci}^A = \sum_{j=1}^{n} W_{ij}^A(\theta)\,\dot{\theta}_j \tag{2.12}$$

$$\omega_i^A = \sum_{j=1}^{n} \Omega_{i-1,j}^L(\theta)\,\dot{\theta}_j + T_{i-1}(\theta)\;\alpha_i\;\dot{\Phi}_i \tag{2.13}$$

where

$W_{ij}^{A}(\theta)$ are the velocity influence coefficient matrices and $\Omega_{i-1,j}^{L}(\theta)$ are the angular velocity influence coefficient matrices of the previous link

α_i is unit vector along the direction of rotation of the ith actuator in the link frame on which the actuator is placed (i-1th link).

Also note that, V_{ci}^{A} does not depend on $\dot{\Phi}_k$, $k = 1 , \ldots\ldots , n$, because V_{ci}^{A} is the translational velocity of the actuator mass center which is a fixed point on the i-1th link. So the actuator mass center velocity can be explained only in terms of θ and $\dot{\theta}$ variables, resulting that V_{ci}^{A} term of the actuators does not include Φ and $\dot{\Phi}$ variables.

If equation (2.12) and (2.13) are put into (2.10) ,

$$KE_i^{A} = \frac{1}{2}\, m_i^{A} \left(\sum_{j=1}^{n} (W_{ij}^{A})^{T} \dot{\theta}_j\right)\left(\sum_{k=1}^{n} W_{ik}^{A} \dot{\theta}_k\right)$$

$$+\frac{1}{2}\left(\sum_{j=1}^{n} (\Omega_{i-1,j}^{L})^{T} \dot{\theta}_j + \alpha_i^{T} T_{i-1}^{T} \dot{\Phi}_i\right) J_i^{A} \left(\sum_{k=1}^{n} \Omega_{i-1,k}^{L} \dot{\theta}_k + T_{i-1}\, \alpha_i\, \dot{\Phi}_i\right) \tag{2.14}$$

is formed, and if equation (2.14) is written in expanded form

$$KE_i^A = \frac{1}{2} m_i^A \left(\sum_{j=1}^{n} (W_{ij}^A)^T \dot{\theta}_j\right)\left(\sum_{k=1}^{n} W_{ik}^A \dot{\theta}_k\right)$$

$$+\frac{1}{2}\left(\sum_{j=1}^{n} (\Omega_{i-1,j}^L)^T \dot{\theta}_j\right) J_i^A \left(\sum_{k=1}^{n} \Omega_{i-1,k}^L \dot{\theta}_k\right)$$

$$+\left(\sum_{j=1}^{n} (\Omega_{i-1,j}^L)^T \dot{\theta}_j\right) J_i^A\ T_{i-1}\,\alpha_i\ \dot{\Phi}_i + \frac{1}{2}\dot{\Phi}_i^2\,\alpha_i^T\, T_{i-1}^T\, J_i^A\, T_{i-1}\,\alpha_i \qquad (2.15)$$

is obtained

Equation (2.15) can be written more compactly by the help of the following equations;

$$\sum_{j=1}^{n}\sum_{k=1}^{n} m_{ijk}^A \dot{\theta}_j \dot{\theta}_k = \frac{1}{2} m_i^A \left(\sum_{j=1}^{n} (W_{ij}^A)^T \dot{\theta}_j\right)\left(\sum_{k=1}^{n} W_{ik}^A \dot{\theta}_k\right)$$

$$+\frac{1}{2}\left(\sum_{j=1}^{n} (\Omega_{i-1,j}^L)^T \dot{\theta}_j\right) J_i^A \left(\sum_{k=1}^{n} \Omega_{i-1,k}^L \dot{\theta}_k\right) \qquad (2.16)$$

$$\sum_{j=1}^{n} \Gamma_{ij}^A \dot{\theta}_j \dot{\Phi}_i = \left(\sum_{j=1}^{n} (\Omega_{i-1,j}^L)^T \dot{\theta}_j\right) J_i^A\ T_i\,\alpha_i\ \dot{\Phi}_i \qquad (2.17)$$

and

$$\frac{1}{2}\dot{\Phi}_i^2\ J_i^A = \frac{1}{2}\dot{\Phi}_i^2\,\alpha_i^T\, T_{i-1}^T\, J_i^A\, T_{i-1}\ \alpha_i \qquad (2.18)$$

Then by using (2.16) , (2.17) , (2.18) in equation (2.15)

$$KE_i^A = \sum_{j=1}^{n} \sum_{k=1}^{n} m_{ijk}^A \dot{\theta}_j \dot{\theta}_k + \sum_{j=1}^{n} \Gamma_{ij}^A \dot{\theta}_j \dot{\Phi}_i + \frac{1}{2} \dot{\Phi}_i^2 J_i^A \tag{2.19}$$

is obtained, where

$$m_{ijk}^A = \frac{1}{2} m_i^A (\boldsymbol{W}_{ij}^A)^T \boldsymbol{W}_{ik}^A + \frac{1}{2} (\Omega_{i-1,j}^L)^T \boldsymbol{J}_i^A \Omega_{i-1,k}^L \tag{2.20}$$

$$\Gamma_{ij}^A = \frac{1}{2} (\Omega_{i-1,j}^L)^T \boldsymbol{J}_i^A \boldsymbol{T}_i \alpha_i + \frac{1}{2} \alpha_i^T \boldsymbol{T}_i^T \boldsymbol{J}_i^A \Omega_{i-1,k}^L \tag{2.21}$$

and J_i^A is the moment of inertia of the rotor about the rotation axis.

The PE equations and their contributions to the Lagrange's Equation are obtained as follows.

2.3 Potential Energy of a Link:

For a rigid link, Potential Energy is the result of gravity and can be written as follows

$$PE_i^L = -\boldsymbol{g}^T m_i^L \boldsymbol{p}_i^L(\theta) \tag{2.22}$$

where

$\boldsymbol{g}$ is the gravitational acceleration vector

$\boldsymbol{p}_i(\theta)^L$ is the link mass center position vector explained in base coordinates

Then the following derivation can be made from (2.22).

$$\frac{\partial PE^L}{\partial\theta_j} = \sum_{i=1}^{n} \frac{\partial PE_i^L}{\partial\theta_j} = -\sum_{i=1}^{n} m_i^L\, \boldsymbol{g}^T\, (\partial \boldsymbol{p}_i^L / \partial\theta_j) \qquad j=1, \ldots, n \qquad (2.23)$$

But note that

$$\partial \boldsymbol{p}_i^L / \partial\theta_j = \boldsymbol{W}_{ij}^L \qquad (2.24)$$

Then (2.23) becomes

$$\frac{\partial PE^L}{\partial\theta_j} = -\sum_{i=1}^{n} m_i^L\, \boldsymbol{g}^T\, \boldsymbol{W}_{ij}^L \qquad (2.25)$$

2.4 Potential Energy of an Actuator:

For an actuator, Potential Energy is the result of, Potential Energy under the effect of gravity and the elastic potential of the joints and can be written as

$$PE_i^A = -\boldsymbol{g}^T m_i^A\, \boldsymbol{p}_i^A + \frac{1}{2}(\Psi_i - \theta_i)\ K_i\ (\Psi_i - \theta_i) \qquad (2.26)$$

where K_i is the joint spring constant of the spring of the ith actuator.

Then similar to previous section

$$\frac{\partial PE^A}{\partial \theta_j} = \sum_{i=1}^{n} \frac{\partial PE_i^A}{\partial \theta_j} = -\sum_{i=1}^{n} m_{ij}^A \, \boldsymbol{g}^T \boldsymbol{W}_{ij}^A - K_i \, (\Psi_i - \theta_i) \qquad j = 1, \ldots, n$$

(2.27)

$$\frac{\partial PE^A}{\partial \Psi_i} = K_i \, (\Psi_i - \theta_i) \qquad i = 1, \ldots, n \qquad (2.28)$$

2.5 Dynamic Equations of a Flexible Joint Manipulator

Using the Kinetic and Potential energies of the links and actuators in Lagrange`s equations the dynamic equations are obtained as given below :

The equations for the first n generalized coordinates , θ_i , $i = 1 , \ldots , n$, are

$$\sum_{i=1}^{n} \sum_{k=1}^{n} [m_{ikj}^L + m_{ikj}^A] \ddot{\theta}_k + \sum_{i=1}^{n} \Gamma_{ij}^A \ddot{\Phi}_i$$

$$+ \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{q=1}^{n} [\frac{\partial m_{ikj}^L}{\partial \theta_q} + \frac{\partial m_{ikj}^A}{\partial \theta_q}] \dot{\theta}_q \dot{\theta}_k + \sum_{i=1}^{n} \sum_{q=1}^{n} \frac{\partial \Gamma_{ij}^A}{\partial \theta_q} \dot{\theta}_q \dot{\Phi}_i$$

$$-\sum_{i=1}^{n} \sum_{q=1}^{n} \sum_{k=1}^{n} [\frac{\partial m_{ijk}^L}{\partial \theta_j} + \frac{\partial m_{ijk}^A}{\partial \theta_j}] \dot{\theta}_q \dot{\theta}_k - \sum_{i=1}^{n} \sum_{q=1}^{n} \frac{\partial \Gamma_{ij}^A}{\partial \theta_j} \dot{\theta}_q \dot{\Phi}_i$$

$$-\frac{1}{2} \sum_{i=1}^{n} \frac{\partial J_i^A}{\partial \theta_j} \dot{\Phi}_i^2 - \sum_{i=1}^{n} m_i^L \, \boldsymbol{g}^T \boldsymbol{W}_{ij}^L - \sum_{i=1}^{n} m_{ij}^A \, \boldsymbol{g}^T \boldsymbol{W}_{ij}^A - K_i \, (\Psi_i - \theta_i) = 0$$

$$j = 1 , \ldots , n \qquad (2.29)$$

The above equation can be arranged and written as

$$\sum_{k=1}^{n} M_{jk}\ \ddot{\theta}_k + \sum_{i=1}^{n} \Gamma_{ij}^{A}\ \ddot{\Phi}_i + Q_j\ - K_j\,(\Psi_j - \theta_j)\ = 0 \qquad j = 1, \ldots\ldots, n \qquad (2.30)$$

where

$$M_{jk} = \sum_{i=1}^{n}\ m_{ikj}^{L} + m_{ikj}^{A} \qquad (2.31)$$

and

$$Q_j = \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{q=1}^{n}\left[\frac{\partial m_{ikj}^{L}}{\partial\theta_q} + \frac{\partial m_{ikj}^{A}}{\partial\theta_q}\right]\dot{\theta}_q\,\dot{\theta}_k + \sum_{i=1}^{n}\sum_{q=1}^{n}\frac{\partial\Gamma_{ij}^{A}}{\partial\theta_q}\ \dot{\theta}_q\,\dot{\Phi}_i$$

$$-\sum_{i=1}^{n}\sum_{q=1}^{n}\sum_{k=1}^{n}\left[\frac{\partial m_{ijk}^{L}}{\partial\theta_j} + \frac{\partial m_{ijk}^{A}}{\partial\theta_j}\right]\dot{\theta}_q\ \dot{\theta}_k - \sum_{i=1}^{n}\sum_{q=1}^{n}\frac{\partial\Gamma_{ij}^{A}}{\partial\theta_j}\ \dot{\theta}_q\,\dot{\Phi}_i$$

$$-\frac{1}{2}\sum_{i=1}^{n}\frac{\partial J_i^{A}}{\partial\theta_j}\ \dot{\Phi}_i^{2} - \sum_{i=1}^{n}\ m_i^{L}\ \boldsymbol{g}^{T}\ \boldsymbol{W}_{ij}^{L} - \sum_{i=1}^{n}\ m_{ij}^{A}\ \boldsymbol{g}^{T}\ \boldsymbol{W}_{ij}^{A} \qquad j = 1, \ldots\ldots, n$$

$$(2.32)$$

In matrix form equation (2.30) can be written as

$$\boldsymbol{M}\ddot{\theta} + \Gamma^{A}\boldsymbol{R}\ddot{\Psi} + \boldsymbol{Q} + \boldsymbol{K}(\theta - \Psi) = 0 \qquad (2.33)$$

Here, for j=1,.....,n , k=1,.......n, $\boldsymbol{M}$ is the matrix whose elements are M_{jk}, Γ^{A} is the matrix whose elements are Γ_{jk}, $\boldsymbol{Q}$ is the vector whose elements are Q_j, $\boldsymbol{K}$

$\boldsymbol{K}$ = diag $[K_j]$ is a diagonal matrix whose diagonal elements are K_j and $\boldsymbol{R}$ = diag $[r_j]$ where r_j is the gear ratio of the j_{th} transmission.

The dynamic equations corresponding to the second n generalized coordinates, Ψ_i are obtained as

$$J_i^A r^2 \ddot{\Psi}_i + \sum_{q=1}^{n} \frac{\partial J_i^A}{\partial \theta_q} r \dot{\theta}_q \dot{\Psi}_i + \sum_{j=1}^{n} \Gamma_{ij}^A \ddot{\theta}_j + \sum_{j=1}^{n} \sum_{q=1}^{n} \frac{\partial \Gamma_{ij}^A}{\partial \theta_q} r \dot{\theta}_q \dot{\theta}_j + K_i (\Psi_i - \theta_i)$$

$$= T_i \qquad i = 1, \ldots\ldots, n \qquad (2.34)$$

The above equation can be arranged and written as

$$J_i^A r^2 \ddot{\Psi}_i + \sum_{j=1}^{n} \Gamma_{ij}^A \ddot{\theta}_j + B_i(\theta, \dot{\theta}, \dot{\Psi}_i) + K_i (\Psi_i - \theta_i) = T_i \qquad (2.35)$$

where

$$B_i = \sum_{q=1}^{n} \frac{\partial J_i^A}{\partial \theta_q} r \dot{\theta}_q \dot{\Psi}_i + \sum_{j=1}^{n} \sum_{q=1}^{n} \frac{\partial \Gamma_{ij}^A}{\partial \theta_q} r \dot{\theta}_q \dot{\theta}_j \qquad (2.36)$$

In matrix form equation (2.35) can be written as

$$\boldsymbol{J}^A \boldsymbol{R}^2 \ddot{\Psi} + (\Gamma^A)^T \ddot{\theta} + \boldsymbol{B} + \boldsymbol{K}(\Psi - \theta) = \boldsymbol{R}\,\boldsymbol{T} \qquad (2.37)$$

where $\boldsymbol{J}^A$ = diag $[J_i^A]$ is a diagonal matrix whose diagonal elements are J_i^A , $\boldsymbol{R}^2$ = diag $[r_i^2]$, $\boldsymbol{B}$ is the vector whose elements are $B_i+B_i^r\, r_i^2\, \dot{\Psi}_i$, where B_i^r is torsional viscous damping coefficient of the ith rotor, and $\boldsymbol{T}$ is the vector of control actuator torques. Notice that only n torques are available to control the system which has 2n degrees of freedom.

Since the joint and rotor angles are related through torsional springs, a direct relation between the output (joint angles, or alternatively end-effector position), and the input (control torques) by eliminating the intermediate variables (rotor angles), can be obtained.

However there are two important remarks that must be made at this point.

First one is that, as the number of the bodies of a manipulator increases, the order of the differential equation between the inputs and outputs also increase. This situation can be better visualized by two examples.

Example 1- One Link Manipulator

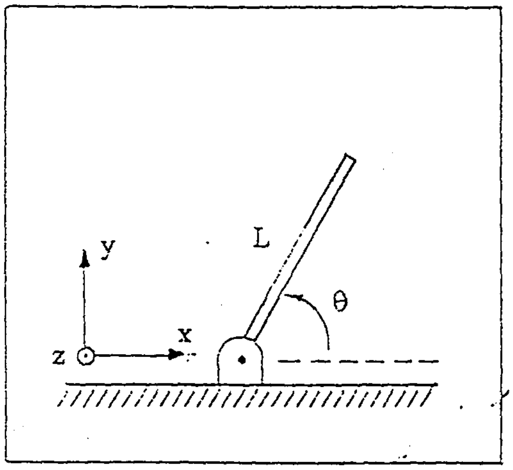


Figure 2.5.1 1R Manipulator

For the manipulator shown in Figure 2.5.1 the equations of motion are

$$I^L_{zz}\,\ddot{\theta} + m^L g\ s\theta\ L/2 - K(\Psi - \theta) = 0 \qquad \text{(E.1)}$$

$$I^A_{zz}\, r^2\, \ddot{\Psi} + K(\Psi - \theta) = T \qquad \text{(E.2)}$$

Then from (E.1)

$$\Psi = \frac{I^L_{zz}\,\ddot{\theta} + \frac{L}{2} m^L g\ c\theta}{K} + \theta \qquad \text{(E.3)}$$

$$\ddot{\Psi} = \frac{I^L_{zz}\,\overset{(4)}{\theta} - \frac{L}{2} m^L (s\theta\ \ddot{\theta} + c\theta\ \dot{\theta}^2)}{K} + \ddot{\theta} \qquad \text{(E.4)}$$

When equations (E.3) and (E.4) are put into equation (E.2) , the following equation is the result

$$\frac{I^{A}_{zz}\, r^2\, I^{L}_{zz}}{K}\overset{(4)}{\theta} + [I^{L}_{zz} - I^{A}_{zz}\, r^2(1-\frac{m^L\, L\, g\, s\theta}{2K})]\,\ddot{\theta} + m^L\, g\,\frac{L}{2}\,(-\frac{I^{A}_{zz}\, r^2}{K}c\theta\ \dot{\theta}^2 + s\theta)$$

$$= T \qquad \text{(E.5)}$$

From (E.5) it is seen that the order of the equation is 4.

Example 2- Three Link Manipulator

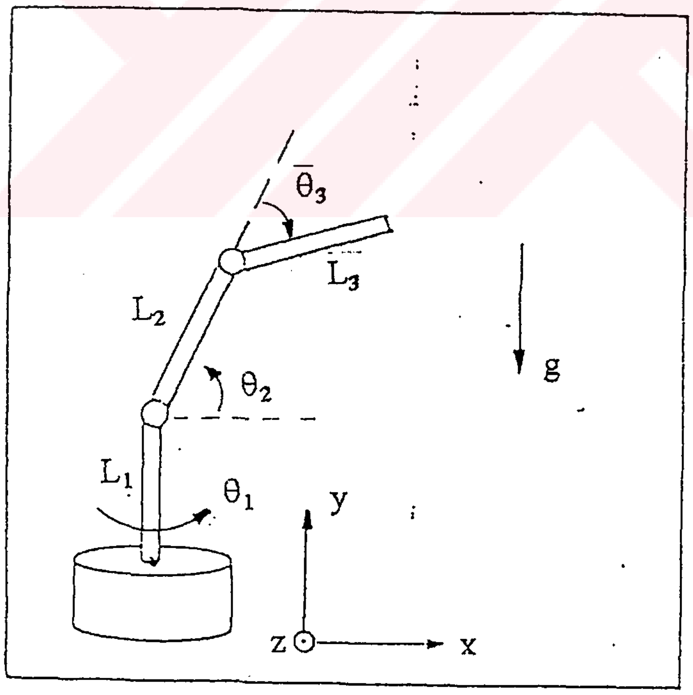


Figure 2.5.2 3R Spatial manipulator

In Figure 2.5.2 there is a 3-link manipulator with flexible joints

Joint coordinates are

$x = [\theta_1, \theta_2, \theta_3, \Psi_1, \Psi_2, \Psi_3]$ where

θ_1 , θ_2 and θ_3 correspond to rigid coordinate variables

Ψ_1 , Ψ_2 and Ψ_3 correspond to elastic coordinate variables

$$KE_{L1} = \frac{1}{2} m^L_1 V^{L\,2}_1 + \frac{1}{2} I^L_1 \omega^{L\,2}_1 \qquad \text{(E.6)}$$

where

$$V^L_1 = 0 \qquad \text{(E.7)}$$

$$\omega^L_1 = \begin{bmatrix} 0 \\ \dot{\theta}_1 \\ 0 \end{bmatrix} \qquad \text{(E.8)}$$

$$PE^L_1 = m^L_1\, g\, L_1 / 2 \qquad \text{(E.9)}$$

$$KE^L_2 = \frac{1}{2} m^L_2 V^{L\,2}_2 + \frac{1}{2} I^L_2 \omega^{L\,2}_2 \qquad \text{(E.10)}$$

where

$$V^L_2 = \begin{bmatrix} \frac{L_2}{2}(-s\theta_1 c\theta_2 \dot{\theta}_1 - c\theta_1 s\theta_2 \dot{\theta}_2) \\ \frac{L_2}{2}(c\theta_2 \dot{\theta}_2) \\ \frac{L_2}{2}(-c\theta_1 c\theta_2 \dot{\theta}_1 + s\theta_1 s\theta_2 \dot{\theta}_2) \end{bmatrix} \qquad \text{(E.11)}$$

$$\omega^{L}{}_{2} = \begin{bmatrix} \dot{\theta}_2 s\theta_1 \\ \dot{\theta}_1 \\ \dot{\theta}_2 c\theta_1 \end{bmatrix} \tag{E.12}$$

$$PE^{L}{}_{2} = m^{L}{}_{2}\, g\, (L_1 + \frac{L_2}{2}\, s\theta_2\,) \tag{E.13}$$

$$KE^{L}{}_{3} = \frac{1}{2}\, m^{L}{}_{2}\, V^{L}{}_{3}{}^{2} + \frac{1}{2}\, I^{L}{}_{2}\, \omega^{L}{}_{3}{}^{2} \tag{E.14}$$

where

$$V^{L}{}_{3} = \begin{bmatrix} \frac{L_3}{2}[-s\theta_1 c\theta_{23}\dot{\theta}_1 - c\theta_1 s\theta_{23}(\dot{\theta}_2+\dot{\theta}_3)] + L_2(-s\theta_1 c\theta_2 \dot{\theta}_1 - c\theta_1 s\theta_2 \dot{\theta}_2) \\ \frac{L_3}{2}(c\theta_{23}(\dot{\theta}_2+\dot{\theta}_3) + L_2(c\theta_2 \dot{\theta}_2) \\ \frac{L_3}{2}(-c\theta_1 c\theta_{23}\dot{\theta}_1 + s\theta_1 s\theta_{23}(\dot{\theta}_2+\dot{\theta}_3) + L_2(-c\theta_1 c\theta_2 \dot{\theta}_1 + s\theta_1 s\theta_2 \dot{\theta}_2) \end{bmatrix} \tag{E.15}$$

$$\omega^{L}{}_{3} = \begin{bmatrix} (\dot{\theta}_2+\dot{\theta}_3)s\theta_1 \\ \dot{\theta}_1 \\ (\dot{\theta}_2+\dot{\theta}_3)c\theta_1 \end{bmatrix} \tag{E.16}$$

$$PE^{L}{}_{3} = m^{L}{}_{3}\, g\, (\, L_1 + L_2\, s\theta_2 + \frac{L_3}{2}\, s\theta_{23}\,) \tag{E.17}$$

$$KE^{A}{}_{1} = \frac{1}{2}\, I^{A}{}_{1}\, \omega^{A}{}_{1}{}^{2} \tag{E.18}$$

where

$$\omega^{A}{}_{1} = \begin{bmatrix} 0 \\ r\dot{\Psi}_1 \\ 0 \end{bmatrix} \tag{E.19}$$

$$PE^{A}{}_{1} = \frac{1}{2} K_1 (\Psi_1 - \theta_1)^2 \tag{E.20}$$

$$KE^{A}{}_{2} = \frac{1}{2} m^{A}{}_{2} V^{A}{}_{2}{}^{2} + \frac{1}{2} I^{A}{}_{2} \omega^{A}{}_{2}{}^{2} \tag{E.21}$$

where

$$V^{A}{}_{2} = \boldsymbol{0} \tag{E.22}$$

$$\omega^{A}{}_{2} = \begin{bmatrix} r\dot{\Psi}_2 s\theta_1 \\ \dot{\theta}_1 \\ r\dot{\Psi}_2 c\theta_1 \end{bmatrix} \tag{E.23}$$

$$PE^{A}{}_{2} = \frac{1}{2} K_2 (\Psi_2 - \theta_2)^2 + m^{A}{}_{2}\, g\, L_1 \tag{E.24}$$

$$KE^{A}{}_{3} = \frac{1}{2} m^{A}{}_{3} V^{A}{}_{3}{}^{2} + \frac{1}{2} I^{A}{}_{3} \omega^{A}{}_{3}{}^{2} \tag{E.25}$$

where

$$V^{A}{}_{3} = \begin{bmatrix} L_2(-s\theta_1 c\theta_2 \dot{\theta}_1 - c\theta_1 s\theta_2 \dot{\theta}_2) \\ L_2(c\theta_2 \dot{\theta}_2) \\ L_2(-c\theta_1 c\theta_2 \dot{\theta}_1 + s\theta_1 s\theta_2 \dot{\theta}_2) \end{bmatrix} \tag{E.26}$$

$$\omega^{A}{}_{3} = \begin{bmatrix} (\dot{\theta}_2 + r\dot{\Psi}_3)s\theta_1 \\ \dot{\theta}_1 \\ (\dot{\theta}_2 + r\dot{\Psi}_3)c\theta_1 \end{bmatrix} \qquad (E.27)$$

$$PE^{A}{}_{3} = \frac{1}{2} K_3 (\Psi_3 - \theta_3)^2 + m^{A}{}_{3}\, g\, (L_1 + L_2\, s\theta_2) \qquad (E.28)$$

When above equations are put into Lagrange’s equation and put into a matrix form, following results are obtained

$$\begin{bmatrix} M_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & M_{22} & M_{23} & 0 & 0 & rI^{A}{}_{3zz} \\ 0 & M_{32} & M_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & r^2 I^{A}{}_{1yy} & 0 & 0 \\ 0 & 0 & 0 & 0 & r^2 I^{A}{}_{2zz} & 0 \\ 0 & rI^{A}{}_{3zz} & 0 & 0 & 0 & r^2 I^{A}{}_{3zz} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \\ \ddot{\Psi}_1 \\ \ddot{\Psi}_2 \\ \ddot{\Psi}_3 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \end{bmatrix} + \begin{bmatrix} K_1(\theta_1 - \Psi_1) \\ K_2(\theta_2 - \Psi_2) \\ K_3(\theta_3 - \Psi_3) \\ K_1(\Psi_1 - \theta_1) \\ K_2(\Psi_2 - \theta_2) \\ K_3(\Psi_3 - \theta_3) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

(E.29)

where

$$M_{11} = \frac{1}{4} m^{L}{}_{2} L_2^2 c^2\theta_2 + m^{L}{}_{3} (\frac{1}{4} L_3^2 c^2\theta_{23} + L_2^2 c^2\theta_2 + L_2 L_3 c\theta_2 c\theta_{23})$$
$$+ \frac{1}{4} m^{A}{}_{3} L_2^2 c^2\theta_2 + I^{L}{}_{1yy} + I^{L}{}_{2xx} s^2\theta_2 + I^{L}{}_{2yy} c^2\theta_2 + I^{L}{}_{3xx} s^2\theta_{23} + I^{L}{}_{3yy} c^2\theta_{23}$$
$$+ I^{A}{}_{2yy} + I^{A}{}_{3xx} s^2\theta_2 + I^{A}{}_{3yy} c^2\theta_2 \qquad (E.30)$$

$$M_{22} = \frac{1}{4} m^{L}{}_{2} L_2^2 + m^{L}{}_{3} (\frac{1}{4} L_3^2 + L_2^2 + L_2 L_3 c\theta_3) + \frac{1}{4} m^{A}{}_{3} L_2^2 + I^{L}{}_{2zz} + I^{L}{}_{3zz}$$
$$+ I^{A}{}_{3zz} \qquad (E.31)$$

$$M_{23} = M_{32} = \frac{1}{2}\, m^L{}_3 \left(\frac{1}{2} L_3{}^2 + L_2 L_3\, c\theta_3 \right) + I^L{}_{3zz} \qquad \text{(E.32)}$$

$$M_{33} = \frac{1}{4}\, m^L{}_3\, L_3{}^2 + I^L{}_{3zz} \qquad \text{(E.33)}$$

$$\begin{aligned} B_1 = {} & 2\, \dot\theta_1 \dot\theta_2 \{ -\frac{1}{4} m^L{}_2 L_2{}^2 c\theta_2 s\theta_2 + \frac{1}{2} m^L{}_3 [-\frac{1}{2} L_3{}^2 c\theta_{23} s\theta_{23} - 2 L_2{}^2 c\theta_2 s\theta_2 \\ & - L_2 L_3 (s\theta_2 c\theta_{23} + c\theta_2 s\theta_{23})] - \frac{1}{4} m^A{}_3 L_2{}^2 c\theta_2 s\theta_2 + I^L{}_{2xx} c\theta_2 s\theta_2 \\ & - I^L{}_{2yy} c\theta_2 s\theta_2 + I^L{}_{3xx} c\theta_{23} s\theta_{23} - I^L{}_{3yy} c\theta_{23} s\theta_{23} + I^A{}_{3xx} c\theta_2 s\theta_2 - I^A{}_{3yy} c\theta_2 s\theta_2 \} \\ & + 2\, \dot\theta_1 \dot\theta_3 \{ \frac{1}{2} m^L{}_3 [-\frac{1}{2} L_3{}^2 c\theta_{23} s\theta_{23} - L_2 L_3 c\theta_2 s\theta_{23}] + I^L{}_{3xx} c\theta_{23} s\theta_{23} \\ & - I^L{}_{3yy} c\theta_{23} s\theta_{23} \} \end{aligned} \qquad \text{(E.34)}$$

$$\begin{aligned} B_2 = {} & \dot\theta_2 \dot\theta_3\, m^L{}_3 L_2 L_3 s\theta_{23} - \frac{1}{2} \dot\theta_3{}^2 m^L{}_3 L_2 L_3 s\theta_3 \\ & - \dot\theta_1{}^2 [-\frac{1}{4} m^L{}_2 L_2{}^2 c\theta_2 s\theta_2 + \frac{1}{2} m^L{}_3 (-\frac{1}{2} L_3{}^2 c\theta_{23} s\theta_{23} - 2 L_2{}^2 c\theta_2 s\theta_2 \\ & - L_2 L_3 s\theta_2 c\theta_{23} - L_2 L_3 c\theta_2 s\theta_{23}) - \frac{1}{4} m^A{}_3 L_2{}^2 c\theta_2 s\theta_2 - I^L{}_{2xx} c\theta_2 s\theta_2 + I^L{}_{2yy} c\theta_2 \\ & s\theta_2 - I^L{}_{3xx} c\theta_{23} s\theta_{23} + I^L{}_{3yy} c\theta_{23} s\theta_{23} + I^A{}_{3yy} c\theta_2 s\theta_2 - I^A{}_{3xx} c\theta_2 s\theta_2] \\ & + \frac{1}{2} m^L{}_2\, g\, c\theta_2 + m^L{}_3\, g\, L_2 c\theta_2 + m^A{}_3\, g\, L_2 c\theta_2 + \frac{1}{2} m^L{}_3\, g\, c\theta_{23} \end{aligned} \qquad \text{(E.35)}$$

$$B_3 = -\frac{1}{2}\,\dot\theta_2\,\dot\theta_3\, m^L{}_3 L_2 L_3 s\theta_3 - \dot\theta_1{}^2\,[\frac{1}{2}\, m^L{}_3\,(-\frac{1}{2}\, L_3{}^2\, c\theta_{23}\, s\theta_{23} - L_2 L_3\, c\theta_2\, s\theta_{23})$$

$$-\frac{1}{2}\, m^L{}_3\, L_2\, L_3\, s\theta_3 - 2\, I^L{}_{3xx}\, c\theta_{23}\, s\theta_{23} + I^L{}_{3yy}\, c\theta_{23}\, s\theta_{23}]$$

$$+\frac{1}{2}\,\dot\theta_2\,\dot\theta_3\, m^L{}_3 L_2 L_3\, s\theta_{23} + \frac{1}{2}\, m^L{}_3\, g\, L_3\, c\theta_{23} \qquad \text{(E.36)}$$

$$B_4 = B_5 = B_6 = 0 \qquad \text{(E.37)}$$

(E.29) can be written in expanded form as

$$M_{11}\ \ddot\theta_1 + B_1 + K_1(\theta_1 - \Psi_1) = 0 \qquad \text{(E.38)}$$

$$M_{22}\,\ddot\theta_2 + M_{23}\,\ddot\theta_3 + r\, I_3{}^A\,\ddot\Psi_3 + B_2 + K_2\,(\theta_2 - \Psi_2) = 0 \qquad \text{(E.39)}$$

$$M_{32}\,\ddot\theta_2 + M_{33}\,\ddot\theta_3 + B_3 + K_3\,(\theta_3 - \Psi_3) = 0 \qquad \text{(E.40)}$$

$$r^2\, I^A{}_{1yy}\,\ddot\Psi_1 + K_1\,(\Psi_1 - \theta_1) = T_1 \qquad \text{(E.41)}$$

$$r^2\, I^A{}_{2zz}\,\ddot\Psi_2 + K_2\,(\Psi_2 - \theta_2) = T_2 \qquad \text{(E.42)}$$

$$r\, I^A{}_{3zz}\,\ddot\theta_2 + r^2\, I^A{}_{3zz}\,\ddot\Psi_3 + K_3\,(\Psi_3 - \theta_3) = T_3 \qquad \text{(E.43)}$$

From (E.40)

$$\Psi_3 = \frac{M_{32}\,\ddot\theta_2 + M_{33}\,\ddot\theta_3 + B_3}{K_3} + \theta_3 \qquad \text{(E.44)}$$

$$\ddot{\Psi}_3 = \frac{\ddot{M}_{32}\ddot{\theta}_2 + 2\dot{M}_{32}\overset{(3)}{\theta}_2 + M_{32}\overset{(4)}{\theta}_2 + \ddot{M}_{33}\ddot{\theta}_3 + 2\dot{M}_{33}\overset{(3)}{\theta}_3 + M_{33}\overset{(4)}{\theta}_3 + \ddot{B}_3}{K_3} + \ddot{\theta}_3 \tag{E.45}$$

From (E.38)

$$\Psi_1 = \frac{M_{11}\ddot{\theta}_1 + B_1}{K_1} + \theta_1 \tag{E.46}$$

$$\ddot{\Psi}_1 = \frac{\ddot{M}_{11}\ddot{\theta}_1 + 2\dot{M}_{11}\overset{(3)}{\theta}_1 + M_{11}\overset{(4)}{\theta}_1 + \ddot{B}_1}{K_1} + \ddot{\theta}_1 \tag{E.47}$$

From (E.39)

$$\begin{aligned}\Psi_2 = \theta_2 + \frac{1}{K_2}\{ & M_{22}\ddot{\theta}_2 + M_{23}\ddot{\theta}_3 \\ & + I_3^A r\left[\frac{\ddot{M}_{32}\ddot{\theta}_2 + 2\dot{M}_{32}\overset{(3)}{\theta}_2 + M_{32}\overset{(4)}{\theta}_2 + \ddot{M}_{33}\ddot{\theta}_3 + 2\dot{M}_{33}\overset{(3)}{\theta}_3 + M_{33}\overset{(4)}{\theta}_3 + \ddot{B}_3}{K_3} + \ddot{\theta}_3\right] \\ & + B_2\}\end{aligned} \tag{E.48}$$

$$\begin{aligned}\ddot{\Psi}_2 = \ddot{\theta}_2 + \frac{1}{K_2}\ [& \ddot{\theta}_2(\ddot{M}_{22} + \frac{r}{K_3} I^A{}_{3zz}\overset{(4)}{M}_{32}) + \overset{(3)}{\theta}_2(2\dot{M}_{22} + 4\frac{r}{K_3} I^A{}_{3zz}\overset{(3)}{M}_{32}) \\ & + \overset{(4)}{\theta}_2(M_{22} + 3\frac{r}{K_3} I^A{}_{3zz}\ddot{M}_{32} + 3\ddot{M}_{32}) + \overset{(5)}{\theta}_2(4\dot{M}_{32}) + \overset{(6)}{\theta}_2 M_{32}\end{aligned}$$

$$
\begin{aligned}
&+\ddot{\theta}_3 \left(\ddot{M}_{23} + \frac{r}{K_3} I^A{}_{3zz} \overset{(3)}{M}_{33} \right) + \overset{(6)}{\theta}_3 \left(2 \dot{M}_{23} + 4 \frac{r}{K_3} I^A{}_{3zz} \overset{(3)}{M}_{33} \right) \\
&+\overset{(4)}{\theta}_3 \left(M_{23} + 3 \frac{r}{K_3} I^A{}_{3zz} \ddot{M}_{33} + 3 \ddot{M}_{33} + r I^A{}_{3zz} \right) + \overset{(5)}{\theta}_3 (4 \dot{M}_{32}) + \overset{(6)}{\theta}_2 M_{33} \\
&+\ddot{B}_2] + \ddot{\theta}_2 \qquad \text{(E.49)}
\end{aligned}
$$

When equations (E.44) , (E.45) , (E.46) , (E.47) , (E.48) and (E.49) are put into equations (E.41) , (E.42) and (E.43), following equations are obtained

$$
T_1 = r^2 I^A{}_{1yy} \left[\frac{\ddot{M}_{11} \ddot{\theta}_1 + 2 \dot{M}_{11} \overset{(3)}{\theta}_1 + M_{11} \overset{(4)}{\theta}_1 + \ddot{B}_1}{K_1} \right] + \ddot{\theta}_1 I^A{}_{1yy} + M_{11} \ddot{\theta}_1 + B_1 + B_4 \qquad \text{(E.50)}
$$

$$
\begin{aligned}
T_2 = {} & r^2 I^A{}_{2zz} \{ \ddot{\theta}_2 + \frac{1}{K_2} [\ddot{\theta}_2 (\ddot{M}_{22} + \frac{r}{K_3} I^A{}_{3zz} \overset{(4)}{M}_{32}) + \overset{(3)}{\theta}_2 (2 \dot{M}_{22} + 4 \frac{r}{K_3} I^A{}_{3zz} \overset{(3)}{M}_{32}) \\
&+ \overset{(4)}{\theta}_2 (M_{22} + 3 \frac{r}{K_3} I^A{}_{3zz} \ddot{M}_{32} + 3 \ddot{M}_{32}) + \overset{(5)}{\theta}_2 (4 \dot{M}_{32}) + \overset{(6)}{\theta}_2 M_{32} \\
&+ \ddot{\theta}_3 (\ddot{M}_{23} + \frac{r}{K_3} I^A{}_{3zz} \overset{(3)}{M}_{33}) + \overset{(3)}{\theta}_3 (2 \dot{M}_{23} + 4 \frac{r}{K_3} I^A{}_{3zz} \overset{(3)}{M}_{33}) \\
&+ \overset{(4)}{\theta}_3 (M_{23} + 3 \frac{r}{K_3} I^A{}_{3zz} \ddot{M}_{33} + 3 \ddot{M}_{33} + r I^A{}_{3zz}) + \overset{(5)}{\theta}_3 (4 \dot{M}_{32}) \\
&+ \overset{(6)}{\theta}_2 M_{33} + \ddot{B}_2] + \ddot{\theta}_2 \} + M_{22} \ddot{\theta}_2 + M_{23} \ddot{\theta}_3 \\
&+ I_3{}^A \left[\frac{\ddot{M}_{32} \ddot{\theta}_2 + 2 \dot{M}_{32} \overset{(3)}{\theta}_2 + M_{32} \overset{(4)}{\theta}_2 + \ddot{M}_{33} \ddot{\theta}_3 + 2 \dot{M}_{33} \overset{(3)}{\theta}_3 + M_{33} \overset{(4)}{\theta}_3 + \ddot{B}_3}{K_3} + \ddot{\theta}_3 \right] \\
&+ B_2 + B_5 \qquad \text{(E.51)}
\end{aligned}
$$

$$T_3 = r^2 I^A{}_{3zz} \left[\frac{\ddot{M}_{32}\ddot{\theta}_2 + 2\dot{M}_{32}\overset{(3)}{\theta}_2 + M_{32}\overset{(4)}{\theta}_2 + \ddot{M}_{33}\ddot{\theta}_3 + 2\dot{M}_{33}\overset{(3)}{\theta}_3 + M_{33}\overset{(4)}{\theta}_3 + \ddot{B}_3}{K_3} + \ddot{\theta}_3\right]$$

$$+ M_{32}\ddot{\theta}_2 + M_{33}\ddot{\theta}_3 + r I^A{}_{3zz}\ddot{\theta}_2 + B_3 + B_6 \qquad (E.52)$$

Equations (E.5) , (E.50) , (E.51) and (E.52) result in that for a 3-link flexible joint manipulator the order of the dynamic equations is 6. However in case of 1-link manipulator the order of the equation was 4. In short it can be concluded that, as the number of the bodies increases, the order of the dynamic equations also increase depending on the direction of the joints (e.g. for a 2-link planar flexible joint manipulator the order is also 6).

The physical reason of this situation is because the forces are transmitted through elastic members, which cause time delay in the force transmission.

The second remark is that, in most applications the actuator rotors have large velocity and small mass, due to the speed reduction mechanisms. For this reason the rotational Kinetic Energy of the actuators can be approximated by neglecting the angular velocities of the links. With this assumption the inertia coupling terms disappear yielding a fourth order input-output relation as shown below.

The actual rotor angles are Φ_i. The variables Ψ_i ($\Psi_i = \Phi_i / r_i$, r_i = gear ratio) are the rotor angles after the gear reduction, hence are of the same order of magnitude with θ_i. this results that Φ_i is generally r_i times larger than θ_i.

By using the above information, the KE of the actuator can be written as follows ;

$$KE_i^A = \frac{1}{2} m_i^A (V_{ci}^A)^T V_{ci}^A + \frac{1}{2} (\underline{\omega}_i^A)^T \underline{J}_i^A \ \underline{\omega}_i^A \tag{2.38}$$

where

m_i^A	is the mass of the i th actuator
V_{ci}^A	is the mass center velocity vector of the i th actuator expressed in base frame (fixed coordinates)
$\underline{J}_i^A$	is the moment of inertia matrix of the i th actuator expressed in actuator frame
$\underline{\omega}_i^A$	is the angular velocity of the i th actuator expressed in actuator frame

Note that it was previously stated that the translational velocity of the i th actuator can be explained in terms of the θ variables (up to θ_{i-1}). So that the mass of the i th actuator can be included into the mass of the i-1 th link. This means, the first term of the equation (2.38) will be included in the KE of the link i-1.

The above procedure reduces equation (2.38) into

$$KE_i^A = \frac{1}{2} (\underline{\omega}_i^A)^T \underline{J}_i^A \ \underline{\omega}_i^A \tag{2.39}$$

(2.39) can be written in expanded form as

$$KE_i^A = \frac{1}{2} [\omega_{i1}\ \omega_{i2}\ \omega_{i3}] \begin{bmatrix} I^A_{ixx} & 0 & 0 \\ 0 & I^A_{iyy} & 0 \\ 0 & 0 & I^A_{izz} \end{bmatrix} \begin{bmatrix} \omega_{i1} \\ \omega_{i2} \\ \omega_{i3} \end{bmatrix} \tag{2.40}$$

$$KE_i^A = \frac{1}{2} (I^A_{ixx}\,\omega_{i1}^2 + I^A_{iyy}\,\omega_{i2}^2 + I^A_{izz}\,\omega_{i3}^2) \tag{2.41}$$

However note that one of these principal axes is the rotor axis. The angular velocity in that direction is equal to $\dot{\Phi}_i$ plus the component of the angular velocity of i-1th link in that direction. Since $\dot{\Phi}_i$ is larger than the components of the angular velocity of the i-1th link in the order of r (gear ratio), neglecting the terms other than $\dot{\Phi}_i$. Then the KE of the rotor reduces to

$$KE_i^A = \frac{1}{2} J_i^A \dot{\Phi}_i^2 \tag{2.42}$$

where J_i^A is the moment of inertia along the rotation axis of the actuator.

Then the 2n equations of motion can be written as

$$\sum_{k=1}^{n} M_{jk}\,\ddot{\theta}_k + Q_j - K_j(\Psi_j - \theta_j) = 0 \qquad j = 1, \ldots\ldots, n \tag{2.43}$$

and

$$J_i^A r^2 \ddot{\Psi} + K_i (\Psi_i - \theta_i) = T_i \quad (2.44)$$

where

$$M_{jk} = \sum_{i=1}^{n} m_{ikj}^{L} \quad (2.45)$$

$$Q_j = \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{q=1}^{n} \frac{\partial m_{ikj}^{L}}{\partial \theta_q} \dot{\theta}_q \dot{\theta}_k - \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{q=1}^{n} \frac{\partial m_{ijk}^{L}}{\partial \theta_j} \dot{\theta}_q \dot{\theta}_k - \sum_{i=1}^{n} m_i^{L} g^{T} W_{ij}^{L}$$

$$j = 1, \ldots, n \quad (2.46)$$

and m_{ikj}^L in equation (2.45) becomes

$$m_{ikj}^{L} = \frac{1}{2} m_i^{L} (W_{ij}^{L})^{T} W_{ik}^{L} + \frac{1}{2} (\Omega_{ij}^{L})^{T} J_i^{L} \Omega_{ik}^{L} + \frac{1}{2} m_i^{A} (W_{ij}^{A})^{T} W_{ij}^{A} \quad (2.47)$$

Therefore with the assumptions that the rotor mass distribution is symmetric about its rotation axis and that the rotational Kinetic Energy of the rotor is mainly due to its own rotation, the inertia and gravitational coupling terms between the joint and rotor variables disappear. These assumptions are common in all flexible joint manipulator models in the literature and are valid for most manipulators used in practice. The advantage of the above equations is that the direct relationship between the input and the output is 4 regardless of the number of links and the flexible joints of the manipulator.

Equations (2.43) and (2.44) can be expressed in matrix form as below

$$\boldsymbol{M}(\theta)\,\ddot{\theta} + \boldsymbol{Q}(\theta,\dot{\theta}) + \boldsymbol{K}(\theta - \Psi) = 0 \qquad (2.48)$$

$$\boldsymbol{I^r}\,\ddot{\Psi} + \boldsymbol{B^r}\,\dot{\Psi} - \boldsymbol{K}(\theta - \Psi) = \boldsymbol{T} \qquad (2.49)$$

where $\boldsymbol{M}(\theta)$ is the symmetric positive definite inertia matrix corresponding to the joint degrees of freedom and $\boldsymbol{Q}(\theta, \dot{\theta})$ contains centrifugal, Coriolis and gravitational terms. $\boldsymbol{M}$ and $\boldsymbol{Q}$ turn out to be same as those for the n link manipulator with the exception that the rotor masses (as point masses) and the stator masses are also included as part of the corresponding links. $\boldsymbol{K}$, $\boldsymbol{I^r}$ and $\boldsymbol{B^r}$ are diagonal matrices: $\boldsymbol{K} = \text{diag}\,[K_i]$, $\boldsymbol{I^r} = \text{diag}\,[r_i^2\, J_i^A]$ and $\boldsymbol{B^r} = \text{diag}\,[r_i^2 B_i^r]$.

CHAPTER III

INVERSE DYNAMICS CONTROL

The control problem for robot manipulators is a servo problem in which the end-effector is made to track the translational and rotational reference values representing the desired trajectory. The joint inputs may be joint torques (like in this case), joint forces or they may be inputs to the actuators, for example, voltage inputs to the motors, depending on the model used for the controller design. The commanded motion may be specified either in the operation space or joint space.

In this paper the task formulation is made in the operational space. Since up to this point all the derivations are made in joint space, in the following paragraphs the relation between the operational and joint space is presented.

Let x_j , j = 1, ... , m represent the Cartesian end effector position variable. Then the generalized function relating each coordinate of the end effector to the joint coordinates, θ_n , can be written as

$$x_j = f_j(\theta_1, \ldots\ldots, \theta_n) \tag{3.1}$$

where m is the number of Cartesian coordinates and n corresponds to the number of joints of the particular manipulator.Taking the total differential of the equation (3.1) yields

$$\dot{x}_j = \sum_{i=1}^{n} J_{ji}\,\dot{\theta}_i \tag{3.2}$$

where

$$J_{ji} = \frac{\partial f_j}{\partial \theta_i} \tag{3.3}$$

in matrix form equation (3.2) can be written as

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\dot{\theta} \tag{3.4}$$

where $\boldsymbol{J}$ is the manipulator Jacobian matrix

At the acceleration level (3.4) can be written as

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}\ddot{\theta} + \dot{\boldsymbol{J}}\dot{\theta} \tag{3.5}$$

The above equation will help in obtaining the relation between the input and output of the system. The dynamic equations of the system obtained in the previous chapter as equation (2.47) is rewritten below for convenience.

$$M(\theta)\,\ddot{\theta} + Q(\theta,\dot{\theta}) + K\,(\theta - \Psi) = 0 \tag{3.6a}$$

$$I^r\,\ddot{\Psi} + B^r\,\dot{\Psi} - K\,(\theta - \Psi) = T \tag{3.6b}$$

Also it was stated that the order of the equation which relates the output x(t) to input T is 4. This can be shown by substituting the second derivatives of (3.6 a) and (3.5) into (3.6 b) as follows:

$$\frac{d}{dt}\left(M\ddot{\theta} + Q + K(\theta - \Psi) \right) = \dot{M}\ddot{\theta} + M\overset{(3)}{\theta} + \dot{Q} + K(\dot{\theta} - \dot{\Psi}) \tag{3.7}$$

$$\frac{d}{dt}(\dot{M}\ddot{\theta} + M\overset{(3)}{\theta} + \dot{Q} + K(\dot{\theta} - \dot{\Psi}) = \ddot{M}\ddot{\theta} + 2\dot{M}\overset{(3)}{\theta} + \ddot{Q} + K(\ddot{\theta} - \ddot{\Psi}) \tag{3.8}$$

From (3.5)

$$\overset{(3)}{x} = 2\dot{J}\ddot{\theta} + J\overset{(3)}{\theta} + \ddot{J}\dot{\theta} \tag{3.9}$$

$$\overset{(4)}{x} = \overset{(3)}{J}\dot{\theta} + 3\ddot{J}\ddot{\theta} + 3\dot{J}\overset{(3)}{\theta} + J\overset{(4)}{\theta} \tag{3.10}$$

(3.7) , (3.8) and (3.10) lead to

$$\dot{\Psi} = K^{-1}(\dot{M}\ddot{\theta} + M\overset{(3)}{\theta} + \dot{Q}) + \dot{\theta} \tag{3.11}$$

$$\ddot{\Psi} = K^{-1}(\ddot{M}\ddot{\theta} + 2\dot{M}\overset{(3)}{\theta} + M\overset{(4)}{\theta} + \ddot{Q}) + \ddot{\theta} \tag{3.12}$$

$$\overset{(4)}{\theta} = J^{-1}(\overset{(4)}{x} - 3\dot{J}\overset{(3)}{\theta} - 3\ddot{J}\ddot{\theta} - \overset{(3)}{J}\dot{\theta}) \tag{3.13}$$

when (3.11) , (3.12) are substituted into equation (3.6 b)

$$I^r[K^{-1}(\ddot{M}\ddot{\theta}+2\dot{M}\overset{(3)}{\theta}+M\overset{(4)}{\theta}+\ddot{Q})+\ddot{\theta}] + B^r[K^{-1}(\dot{M}\ddot{\theta}+M\overset{(3)}{\theta}+\dot{Q})+\dot{\theta}] + M\ddot{\theta}+Q = T \tag{3.14}$$

and when (3.13) is substituted into (3.14) it is brought into the final form as

$$\begin{aligned} T = {} & I^r K^{-1} M J^{-1}[(\overset{(4)}{x} - 3\dot{J}\overset{(3)}{\theta} - 3\ddot{J}\ddot{\theta} - \overset{(3)}{J}\dot{\theta}) + 2\dot{M}\overset{(3)}{\theta} + \ddot{M}\ddot{\theta} + \ddot{Q}] + I^r\ddot{\theta} \\ & + B^r K^{-1}(M\overset{(3)}{\theta} + \dot{M}\ddot{\theta} + \dot{Q}) + B^r\dot{\theta} + M\ddot{\theta} + Q \end{aligned} \tag{3.15}$$

Note that in (3.15) ; $\overset{(3)}{\theta}$, $\ddot{\theta}$, $\dot{\theta}$ and θ can be written in terms of $\overset{(3)}{x}$, $\ddot{x}$, $\dot{x}$ and x by using (3.1) and its derivatives.

Since the equation which shows the relation between the input and output is available it can be used for inverse dynamics control of the system.

The idea of inverse dynamics is to seek a nonlinear feedback control law

$$T = f(x, \dot{x}, \ddot{x}, \overset{(3)}{x}) \tag{3.16}$$

which is when substituted into equation (3.15), results in a linear and decoupled closed loop system.

(3.15) can be written as

$$T = P\overset{(4)}{x} + R(x,\dot{x},\ddot{x},\overset{(3)}{x}) \tag{3.17}$$

If T is chosen according to the following control law

$$T = Pu + R \tag{3.18}$$

Then the system reduces to

$$\overset{(4)}{x} = u \tag{3.19}$$

The term u represents a new input to the system which is yet to be chosen.

Since this is a fourth order system u is chosen as

$$u = \overset{(4)}{x}{}^{d} + C_1(\overset{(3)}{x}{}^{d} - \overset{(3)}{x}) + C_2(\ddot{x}^{d} - \ddot{x}) + C_3(\dot{x}^{d} - \dot{x}) + C_4(x^{d} - x) \tag{3.20}$$

The superscript d denotes the desired values and C_1, C_2, C_3, C_4 are diagonal feedback gain matrices.

The measured quantities are θ , $\dot{\theta}$, $\ddot{\theta}$, $\overset{(3)}{\theta}$. In the absence of modeling error, the application of the torques, so calculated, to the system leads to the following fourth order error dynamics.

$$\overset{(4)}{e} + C_1 \overset{(3)}{e} + C_2 \ddot{e} + C_3 \dot{e} + C_4 e = 0 \tag{3.21}$$

where

$$e = x^d - x \tag{3.22}$$

Asymptotic stability is achieved by selection of proper values for the constants C_{ij} ; i = 1 , ... , 4 , j = 1 , ... , n for the decoupled and linearized system.

It is seen that the above control scheme requires the analytical derivation of the robot dynamic equation (3.6 a) and the acceleration level task equation (3.5) twice. Hence in equation (3.15) $\dot{M}$, $\ddot{M}$, $\dot{Q}$, $\ddot{Q}$, $\ddot{J}$ and $\overset{(3)}{J}$ are needed for the calculation of the control torques.

However, this is not practical. Because the complexity grows exponentially with the order of the differential equations of motion of the inverse dynamics controller and with the number of robot links. Since all these calculations should be performed on-line, with a small sampling period, practical real time implementation of the full inverse dynamics controller for n>2 does not seem possible using present day computers. In addition the second drawback of the above method is the need for measurement of accelerations and jerks. Although the calculation of higher order time derivatives of system variables can be done on the basis of measured positions and velocities of actuators and links, the formulas for such an evaluation are also very complicated.

CHAPTER IV

INVERSE DYNAMICS CONTROL USING SINGULAR ACCELERATION LEVEL EQUATIONS

To eliminate the difficulties related to the complexity of the control law, an inverse dynamics control approach utilizing the dynamic equations at the acceleration level only needs to be formulated.

If $\ddot{x}$ in equation (3.5) is replaced by a new control variable z that represents the " calculated acceleration " such that

$$u = \ddot{z} \tag{4.1}$$

where u is given by equation (3.20), then the equations (2.48), (2.49) and (3.15) can be written in augmented form as below ;

$$\begin{bmatrix} M & 0 & 0 \\ 0 & I^r & -I \\ J & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\Psi} \\ T \end{bmatrix} = \begin{bmatrix} -\mathbf{Q} - \mathbf{K}(\theta - \Psi) \\ -\mathbf{B}^r \dot{\Psi} + \mathbf{K}(\theta - \Psi) \\ -\dot{\mathbf{J}}\dot{\theta} + \mathbf{z} \end{bmatrix} \tag{4.2}$$

The accelerations $\ddot{\theta}$ and $\ddot{\Psi}$ that appear as unknowns in equation (4.2), in addition to the control torques are identical to the accelerations of the real system in the absence of modelling error, and do not have any other physical significance.

First, the solution of $\ddot{\theta}$, $\ddot{\Psi}$ and $\boldsymbol{T}$ from equation (4.2) is not a straightforward task. This is because the acceleration and torque coefficient matrix on the LHS of equation (4.2) is singular, hence equations (4.2) represent a singular set of differential equations. The physical reason of the singularity is that, because the location of the actuators and the end-effector are connected through elastic joints, the control forces do not have an instantaneous effect on the task acceleration. In fact as seen by equation (3.15) the control forces have an instantaneous effect to second derivatives of the task accelerations.

For these reasons, a different approach is suggested for solving the inverse dynamics problem. As digital controllers are generally used to control robots, the use of numerical methods conceived for the solution of differential-algebraic equations (DAE) are applied here.

Let's consider the differential-algebraic equation (DAE)

$$\boldsymbol{F}(\mathrm{t}, \boldsymbol{y}, \dot{\boldsymbol{y}}) = \boldsymbol{0} \qquad (4.3)$$

where $\boldsymbol{F}$, $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$ are s-dimensional vectors. Many of these problems can be solved conveniently and economically using numerical Ordinary Differential Equation (ODE) methods.

The idea of using ODE methods for solving DAE systems directly was introduced in Gear and Petzold [5], and is best illustrated by considering the simplest possible algorithm, based on the backward Euler method. In this method the derivative $y(t_{k+1})$ at time t_{k+1} is approximated by a backward difference of $y(t)$.

$$\dot{y}_{k+1} = (y_{k+1} - y_k) / h \tag{4.4}$$

where h is the integration time step and k is the time step number.

The resulting system of nonlinear equations is solved for y_{k+1} ,

$$\boldsymbol{F} (t_{k+1}, y_{k+1}, (y_{k+1} - y_k) / h) = 0 \tag{4.5}$$

In this way the solution is advanced from time t_k to time t_{k+1}. When this method is applied to the DAE set obtained in the previous chapters the solution converges.

In the control problem, when it is possible to calculate the control vector $\boldsymbol{T}$, the time derivative of $\boldsymbol{T}$ is usually out of the realm of interest. The DAE set which consists of equations (4.2) satisfies this requirement. The variables entering such a DAE set are y and $\boldsymbol{T}$, where

$$y = [\, \theta^T, \Psi^T, \dot{\theta}^T, \dot{\Psi}^T]^T \tag{4.6}$$

According to the above explanations equation (4.4) leads to rewriting equations of (4.2) at time t_{k+1} as follows :

$$\boldsymbol{M}\,(\dot{\theta}_{k+1}-\dot{\theta}_k)\,/\,\mathrm{h}+\boldsymbol{Q}+\boldsymbol{K}\,(\mathrm{h}\dot{\theta}_{k+1}+\theta_k)-\boldsymbol{K}\,(\mathrm{h}\dot{\Psi}_{k+1}+\Psi_k)=\boldsymbol{0} \tag{4.7}$$

$$\boldsymbol{\Gamma}\,(\dot{\Psi}_{k+1}-\dot{\Psi}_k)\,/\,\mathrm{h}+\boldsymbol{B'}\,\dot{\Psi}_{k+1}-\boldsymbol{K}\,(\mathrm{h}\dot{\theta}_{k+1}+\theta_k)+\boldsymbol{K}\,(\mathrm{h}\dot{\Psi}_{k+1}+\Psi_k)=\boldsymbol{T}_{k+1} \tag{4.8}$$

$$\boldsymbol{J}\,(\dot{\theta}_{k+1}-\dot{\theta}_k)\,/\,\mathrm{h}+\dot{\boldsymbol{J}}\,\dot{\theta}_{k+1}=z_{k+1} \tag{4.9}$$

where, $\boldsymbol{M}(\mathrm{h}\dot{\theta}_{k+1}+\theta_k)$, $\boldsymbol{Q}(\mathrm{h}\dot{\theta}_{k+1}+\theta_k,\dot{\theta}_{k+1})$, $\boldsymbol{J}(\mathrm{h}\dot{\theta}_{k+1}+\theta_k)$ and $\dot{\mathrm{J}}\,(\mathrm{h}\dot{\theta}_{k+1}+\theta_k,$ $\dot{\theta}_{k+1})$ also depend on $\dot{\theta}_{k+1}$.

Equations (4.7) , (4.8) and (4.9) are a set of 3n algebraic equations from which the 3n unknowns $\dot{\theta}_{k+1}$, $\dot{\Psi}_{k+1}$ and $\boldsymbol{T}_{k+1}$ can be solved. It is necessary to obtain z that appears in equation (4.9) by analytical integration of equation (3.20) to avoid measurements of accelerations and jerks. Usually the desired motion is specified as piecewise smooth functions. Let $x(\mathrm{t})^d$ be smooth up to the third derivative in the time interval $\mathrm{t_a} < \mathrm{t} < \mathrm{t_b}$. Then integration of equation (3.20) twice leads to

$$\begin{aligned} z - z_a - \dot{z}_a(\mathrm{t}-\mathrm{t_a}) = {} & \ddot{x}^d - \ddot{x}^d_a - \overset{(3)}{x}{}^d_a(\mathrm{t}-\mathrm{t_a}) + C_1(\dot{x}^d - \dot{x}^d_a - \ddot{x}^d_a(\mathrm{t}-\mathrm{t_a})) \\ & - C_1(\dot{x} - \dot{x}_a - \ddot{x}_a(\mathrm{t}-\mathrm{t_a})) + C_2\,(x^d - x^d_a - \dot{x}^d_a(\mathrm{t}-\mathrm{t_a})) \\ & - C_2\,(x - x_a - \dot{x}_a(\mathrm{t}-\mathrm{t_a})) + \int_{\mathrm{t_a}}^{\mathrm{t}}\int_{\mathrm{t_a}}^{\mathrm{t}} \ddot{w}\,\mathrm{dt}\,\mathrm{dt} \qquad \text{for } \mathrm{t_a} < \mathrm{t} < \mathrm{t_b} \end{aligned} \tag{4.10}$$

where

$$\ddot{w} = C_3\,(\dot{x}^d - \dot{x}) + C_4\,(x^d - x) \tag{4.11}$$

Equation (4.10) can be written as below

$$z = \ddot{x}^{d} + C_1(\dot{x}^{d} - \dot{x}) + C_2\,(x^{d} - x) + \int_{t_a}^{t}\int_{t_a}^{t} \{C_3\,(\dot{x}^{d} - \dot{x}) + C_4\,(x^{d} - x)\}\,\mathrm{dt}\,\mathrm{dt} + b \tag{4.12}$$

where

$$\begin{aligned} b = {} & z_a + \dot{z}_a(t - t_a) - \ddot{x}^{d}_{a} - \overset{(3)}{x}{}^{d}_{a}(t - t_a) - C_1\,[(\ddot{x}^{d}_{a} - \ddot{x}_a)(t - t_a) + \dot{x}^{d}_{a} - \dot{x}_a\,] \\ & - C_2\,[\,x^{d}_{a} - x_a + (\dot{x}^{d} - \dot{x}_a)(t - t_a)] \end{aligned} \tag{4.13}$$

Then z_{k+1} in equation (4.9) is obtained as

$$\begin{aligned} z_{k+1} = {} & z_a + \dot{z}_a(t_{k+1} - t_a) + \ddot{x}^{d}_{k+1} - \ddot{x}^{d}_{a} - \overset{(3)}{x}{}^{d}_{a}(t_{k+1} - t_a) + C_1(\dot{x}^{d} - \dot{x}^{d}_{a} - \ddot{x}^{d}_{a}(t - t_a)) \\ & - C_1(\dot{x}_{k+1} - \dot{x}_a - \ddot{x}_a(t_{k+1} - t_a)) + C_2\,(x^{d}_{k+1} - x^{d}_{a} - \dot{x}^{d}_{a}(t_{k+1} - t_a)) \\ & - C_2\,(x_{k+1} - x_a - \dot{x}_a(t_{k+1} - t_a)) + \ddot{w}_{k+1}\,h^2 + \dot{w}_k\,h + w_k \qquad \text{for } t_a < t_{k+1} < t_b \end{aligned} \tag{4.14}$$

where

$$\ddot{w}_{k+1} = C_3\,(\dot{x}^{d}_{k+1} - \dot{x}_{k+1}) + C_4\,(x^{d}_{k+1} - x_{k+1}) \tag{4.15}$$

It is important to note that , because the inverse dynamics problem is noncausal, i.e. the control torques can not effect the end effector accelerations instantaneously, any jumps in the calculated accelerations z cause very large input torques to be found from equations (4.7) , (4.8) , (4.9). To avoid torque saturations and resulting nonlinearities, the integration constants are set to match z and $\dot{z}$ at the boundaries of the desired motion specifications by freely selecting the integration constants accordingly. This is achieved by choosing $z_a = z(t_a^+) = z(t_a^-)$ and $\dot{z}_a = \dot{z}(t_a^+) = \dot{z}(t_a^-)$. If the system starts from rest, then $z_0 = 0$, $\dot{z}_0 = 0$.

Calculation of the input torques that linearize and decouple the system the system in the absence of modeling error can be done in the following order.

Equation (4.9) represents n nonlinear equations from which $\dot{\theta}_{k+1}$ can be solved. Functional iteration can be conveniently used for this purpose. Substitution of equations (4.12) and (4.4) into equation (4.9) yields

$$\begin{aligned}\dot{\theta}_{k+1} = {} & \left(J\frac{1}{h} + \dot{J} + C_1 J + C_2 h J + C_3 h^2 J + C_4 h^3 J\right)^{-1} \Big[J\frac{1}{h}\dot{\theta}_k + z_a + \dot{z}_a(t_{k+1} - t_a) \\ & + \ddot{x}^d_{k+1} - \ddot{x}^d_a - \overset{(3)}{x}{}^d_a(t_{k+1} - t_a) + C_1(\dot{x}^d - \dot{x}^d_a - \ddot{x}^d_a(t - t_a)) \\ & - C_1(\dot{x}_{k+1} - \dot{x}_a - \ddot{x}_a(t_{k+1} - t_a)) + C_2(x^d_{k+1} - x^d_a - \dot{x}^d_a(t_{k+1} - t_a)) \\ & - C_2(x_{k+1} - x_a - \dot{x}_a(t_{k+1} - t_a)) + \ddot{w}_{k+1} h^2 + \dot{w}_k h + w_k \Big] \qquad (4.16)\end{aligned}$$

Starting with $\dot{\theta}_k$ for $\dot{\theta}_{k+1}$ in the right hand side, equation (4.16) is solved for new $\dot{\theta}_{k+1}$. Iteration continues until the norm of the difference between successive iterations is less than a small number ε. It has been observed in the numerical simulations that for $\varepsilon = 10^{-10}$, at most 3 iterations were needed. Note that if the desired motions are specified in the joint space, then J becomes identity matrix and $\dot{J}$ zero, hence equation (4.9) becomes linear for $\dot{\theta}_{k+1}$.

Then equation (4.7) is used for finding $\dot{\Psi}_{k+1}$. Finally the linearizing control torques T_{k+1} are computed from equation (4.8).

The quantities that should be measured for the solution of the equations (4.7) , (4.8) and (4.9) are the joint angles θ_k and the joint velocities $\dot{\theta}_k$. Measurements of the intermediate variables, rotor angles and rotor velocities, are not necessary since they should be calculated in accordance with the consistency of the dynamic model. To this end, equation (4.7) is written for the previous step, where for $\dot{\theta}_k$, the measured quantities are used instead of the calculated ones, as

$$\boldsymbol{M}\frac{1}{h}(\dot{\theta}_k - \dot{\theta}_{k-1}) + \boldsymbol{Q} + \boldsymbol{K}(h\dot{\theta}_k + \theta_{k-1}) - \boldsymbol{K}(h\dot{\Psi}_k + \Psi_{k-1}) = \boldsymbol{0} \qquad (4.17)$$

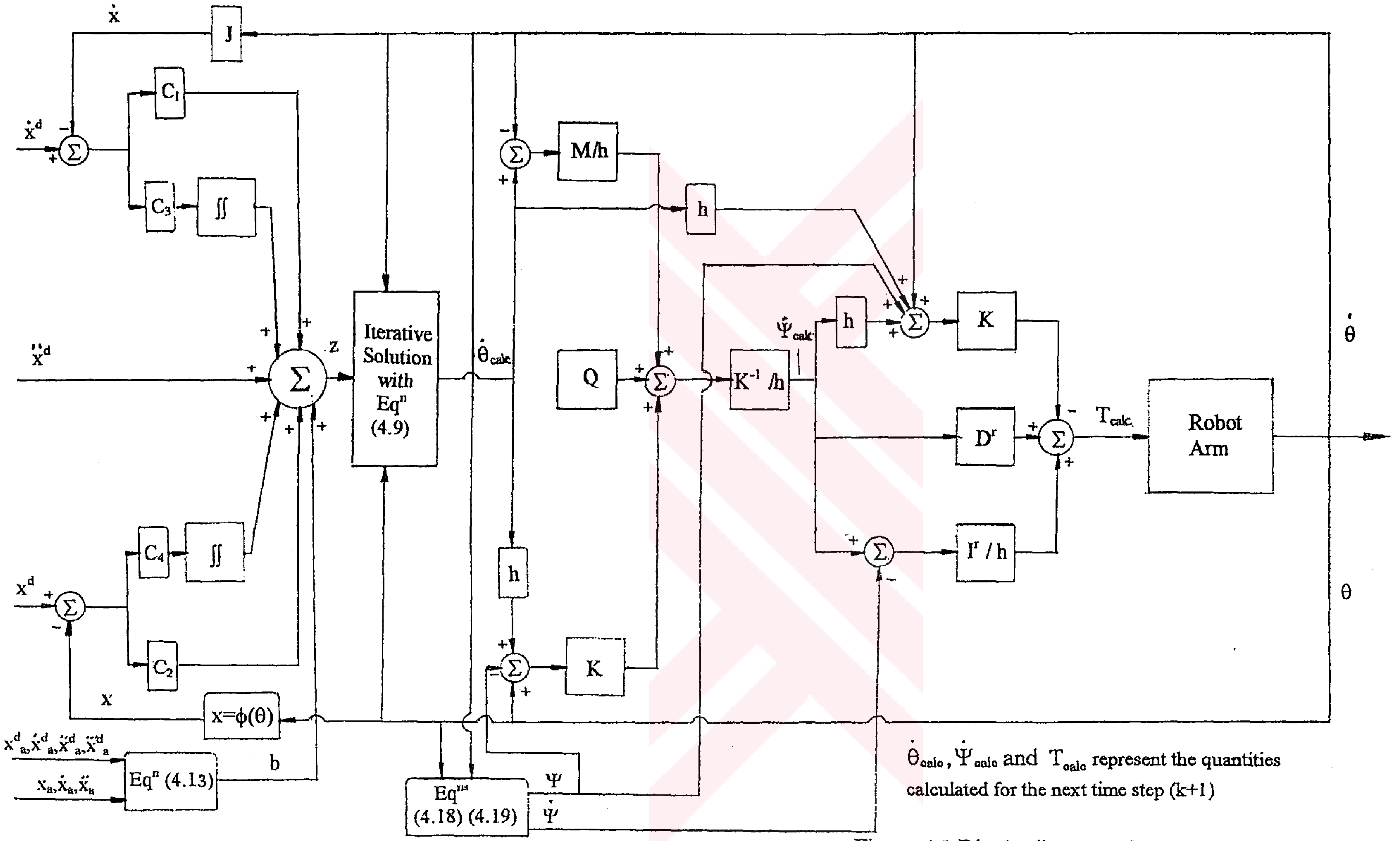
Solution for $\dot{\Psi}_k$ yields

$$\dot{\Psi}_k = \frac{1}{h}\boldsymbol{K}^{1}[\boldsymbol{M}\frac{1}{h}(\dot{\theta}_k - \dot{\theta}_{k-1}) + \boldsymbol{Q}] + \dot{\theta}_k + \frac{1}{h}(\theta_{k-1} - \Psi_{k-1}) \qquad (4.18)$$

Then Ψ_k is obtained as

$$\Psi_k = h\dot{\Psi}_k + \Psi_{k-1}\,. \qquad (4.19)$$

The control algorithm explained above can be represented in a block diagram form as given in figure 4.1.



$\dot{\theta}_{calc}$, $\dot{\Psi}_{calc}$ and T_{calc} represent the quantities calculated for the next time step (k+1)

Figure 4.1 Block diagram of the developed control law

CHAPTER V

CONTROL SIMULATIONS

In this chapter, the performance of the control law, explained in Chapters 3 and 4, is tested by using a computer program written in FORTRAN, and the results are presented. The computer program makes use of the influence coefficient matrices to generate the equations of motion of rigid and flexible joint robot arms , when topological, geometrical and mass properties are given. At each sampling time step the control torques are calculated using the algorithm given in Chapter IV. Then the forward dynamics that represent the actual system is solved as a result of the applied control torques. The actual joint accelerations so obtained are numerically integrated using a predictor-corrector algorithm to yield the positions and velocities.

The performance of the control law is tested for various desired motion trajectories. Both initial errors and modeling errors are considered.

Control gains are chosen according to ITAE criteria. Note, it was shown that dynamic equations of a flexible joint manipulator are 4th order. For a fourth order system, the optimal form of the closed loop transfer function is

$$\frac{Y(s)}{R(s)} = \frac{\alpha^4}{s^4 + 2.1\alpha\ s^3 + 3.4\alpha^2\ s^2 + 2.7\alpha^3\ s + \alpha^4} \tag{5.1}$$

So the control gains used in the control law, explained in chapter 4, are

$$C_1 = 2.1\ \alpha \tag{5.2a}$$

$$C_2 = 3.4\ \alpha^2 \tag{5.2b}$$

$$C_3 = 2.7\ \alpha^3 \tag{5.2c}$$

$$C_4 = \alpha^4 \tag{5.2d}$$

To facilitate the selection of the value of α, equation (5.1) can be written by factoring the polynomials as

$$\frac{Y(s)}{R(s)} = \frac{1}{(\frac{s}{\alpha} + 0.424\alpha \pm j\ 1.263\alpha)\ (\frac{s}{\alpha} + 0.626\alpha \pm j\ 0.4141\alpha)} \tag{5.3}$$

where

$$(\frac{s}{\alpha} + a\alpha \pm jb\alpha) = (\frac{s}{\alpha} + a\alpha + jb\alpha).\ (\frac{s}{\alpha} + a\alpha - jb\alpha) \qquad \text{for compactness}$$

As it can be seen from equation (5.3), this fourth order system has 2 natural frequencies as

$$w_{n1} = \sqrt{0.424^2 + 1.263^2}\ \alpha = 1.33\ \alpha \tag{5.4}$$

$$w_{n2} = \sqrt{0.626^2 + 0.4141^2}\ \alpha = 0.75\ \alpha \tag{5.5}$$

Simulations are made on 1-Link Planar and 3-R Spatial manipulators whose all joints are flexible. During simulations viscous effects are neglected.

In fact, the dynamic equations of both manipulators were given as examples in Chapter 2. However, the dynamic equations of the manipulators were not written by taking into account the assumptions which were explained after obtaining the equations. These assumptions do not change the dynamic equations of the 1-link manipulator but they do change the dynamic equations of the 3-link manipulator. So the dynamic equations of 3-R Spatial manipulator will be rewritten.

Both manipulators and simulations performed on them will be explained in the following sections.

5.1 Case Study 1: 1-R Planar Manipulator

The one link flexible joint manipulator is shown in Figure 2.5.1. In this simple system gravity will not be considered assuming that the system operates in the horizontal plane.

Using Example 1 in Chapter II, the dynamic equations of the 1-link flexible joint manipulator are ;

$$I^L_{zz}\,\ddot{\theta} + K\,(\theta - \Psi) = 0 \qquad \text{(E.1)}$$

$$I^A_{zz}\, r^2\, \ddot{\Psi} + K\,(\Psi - \theta) = T \qquad \text{(E.2)}$$

Then open loop natural frequencies of the system can be found by using the free vibration eigenvalue problem as

$$w^2 = K \, . \left(\frac{1}{I^L_{zz}} + \frac{1}{r^2\, I^A_{zz}}\right) \qquad (5.1.4)$$

However since the flexibility of the system is modeled in the control system, the frequencies of vibrations in closed loop operation are the natural frequencies of the closed loop system given by equations (5.4) and (5.5). The sampling frequency is chosen to be at least 10 times the larger natural frequency of the closed loop system. Smaller sampling frequencies cause instability and divergence of the control forces and response.

The system parameters used in the simulations are

m^L	$= 9$ kg
m^A	$= 1.3\ 10^{-2}$ kg
L	$= 1$ m
K	$= 5000$ N / m
α	$= 40$, 50 & 60 rad / sec
I^L_{zz}	$= 3$ kg.m^2

$I^A_{zz} = 0.23\ 10^{-4}$ kg.m^2

$r = 100$

Initially the link is horizontal with $\theta_{in} = 0°$. The desired motion is a step input in joint space as 10°, i.e. $\theta^d = 10°$.

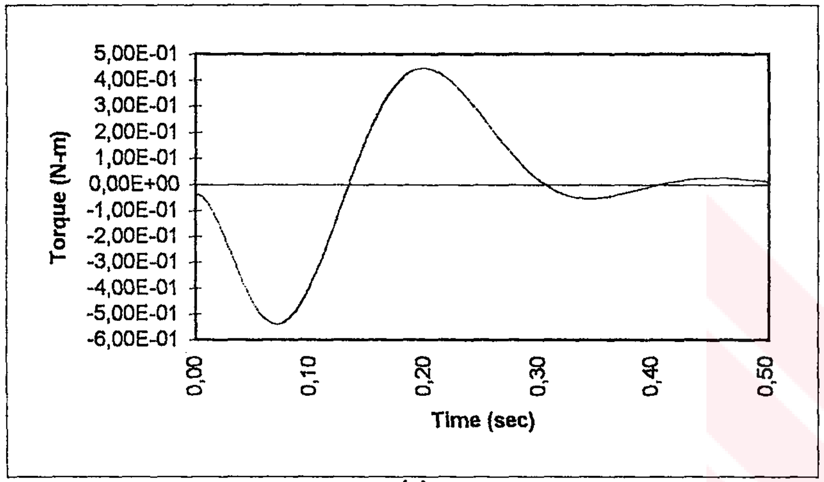
During simulations 2 different sets of control gains are used and modeling error in mass properties is also considered.

The simulation results are presented in figures 5.1.2–5.1.7. In each figure there are three graphs. The order of the graphs is as; the first graph (a) shows the torque applied by the actuator, the second graph (b) shows the angular difference between the rigid and flexible system variables, and the third graph (c) shows the desired motion and the response of rigid and flexible system variables.

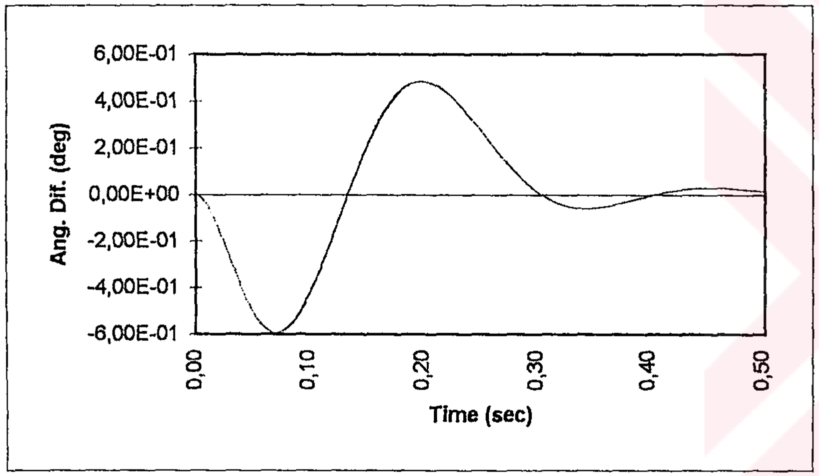
The response with $\alpha = 20$ is given in Figure 5.1.2, when there is no modeling error. In Figure 5.1.3, $\alpha = 20$ and the model mass properties are 10 % smaller than actual. A small steady-state error occurs when there is modeling error ($e_{ss} = 0.001$ deg).

The steady-state error and tracking errors are reduced when α is increased. However larger α incerases the control torques and overshoots. Simulations with $\alpha = 40$ are given in figures 5.1.4 and 5.1.5 (with 10 % modeling error). Simulations with $\alpha = 60$ are given in figures 5.1.6 and 5.1.7 (with 10 % modeling error). The steady-state errors become negligibly small. The responses with $\alpha = 60$

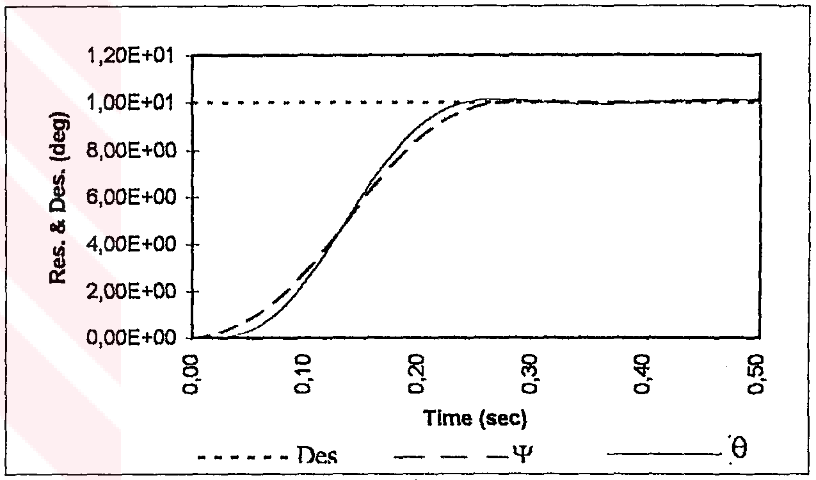
show good performance both for the cases without and with modeling errors, while the control torques are also at acceptable levels.



(a)

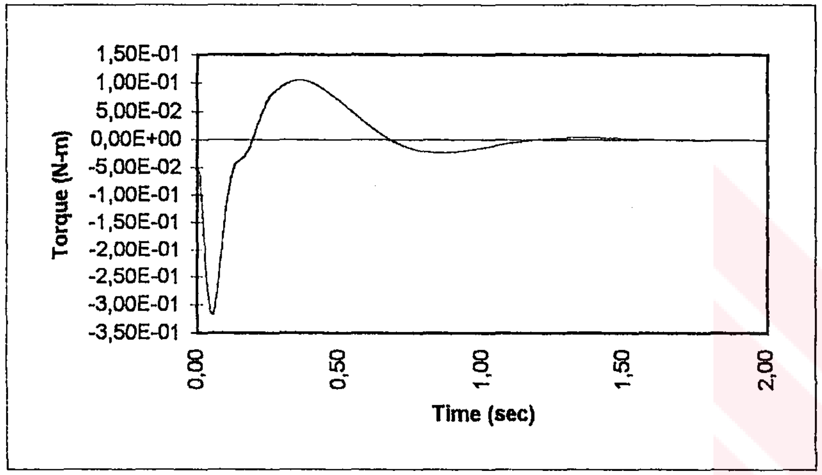


(b)

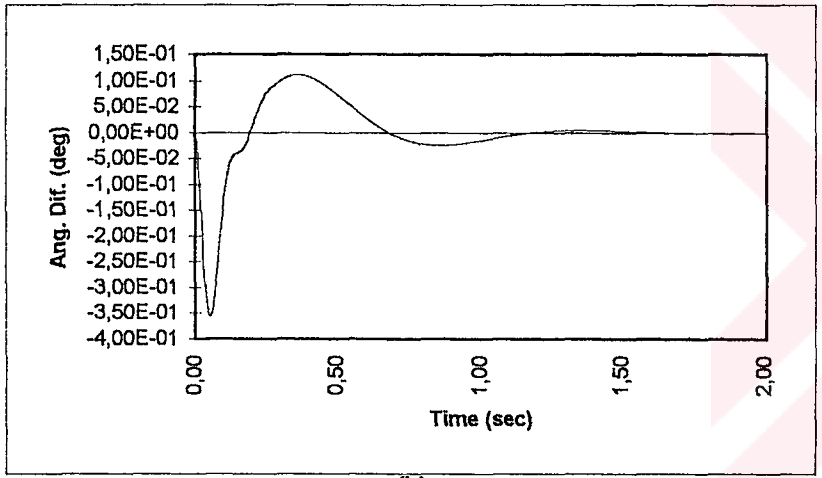


(c)

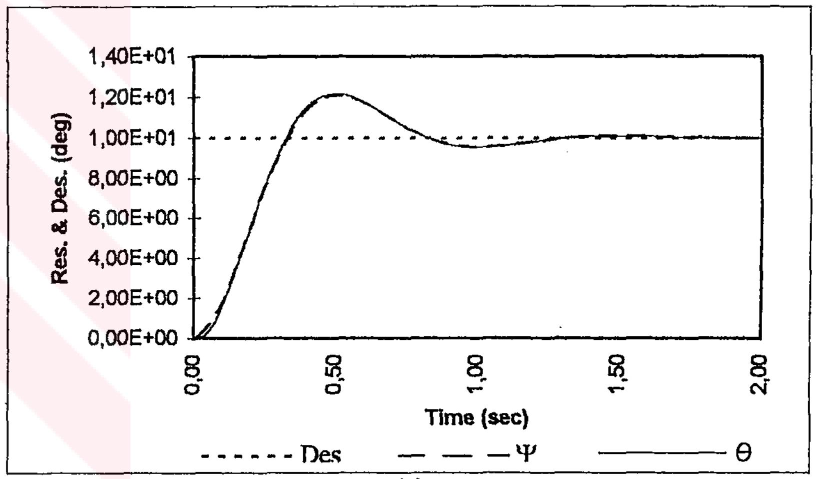
Figure 5.1.1 1R , α = 20



(a)

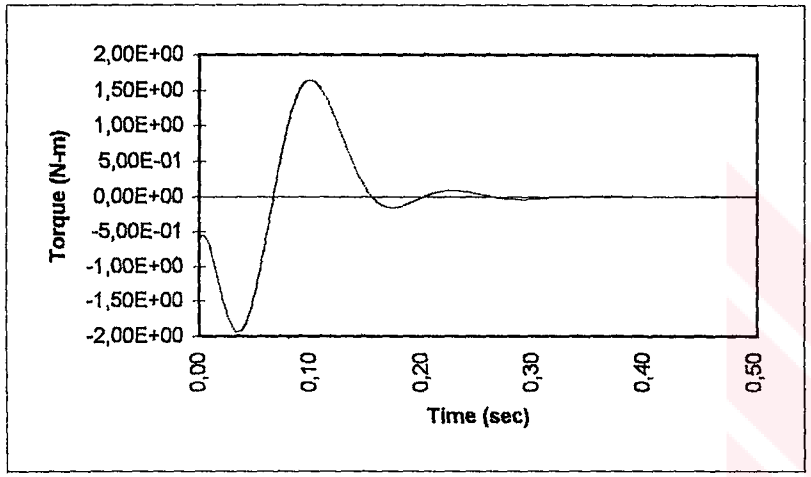


(b)

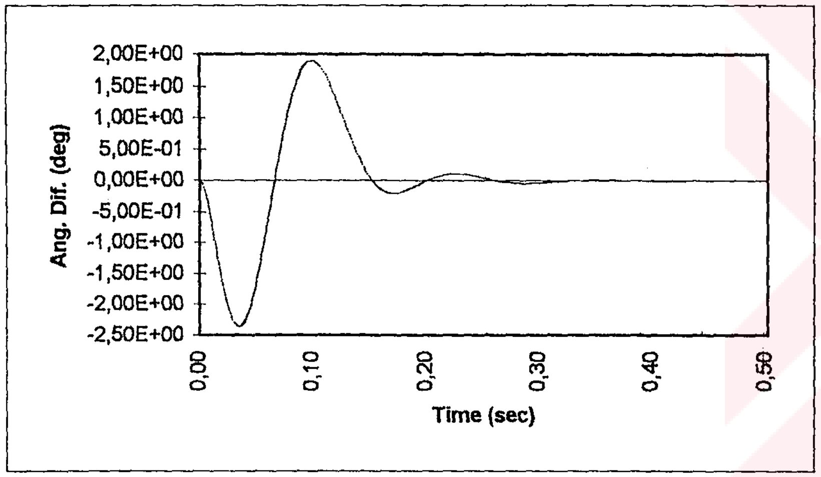


(c)

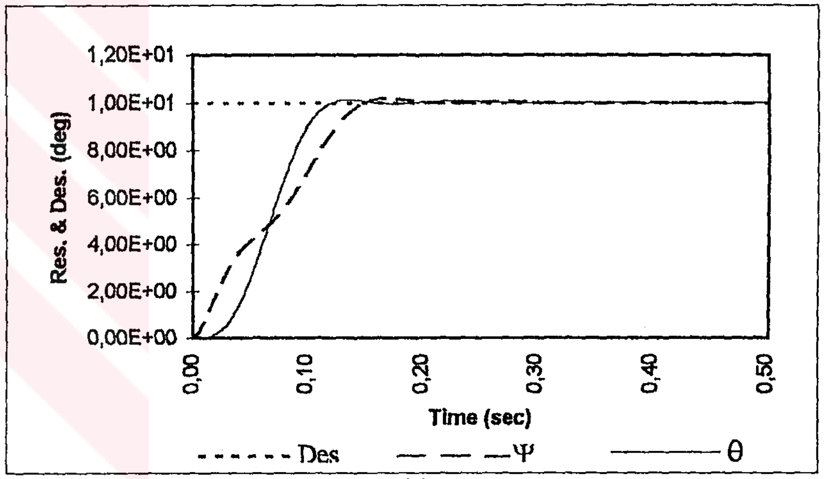
Figure 5.1.2 1R , α = 20 , with modeling error



(a)

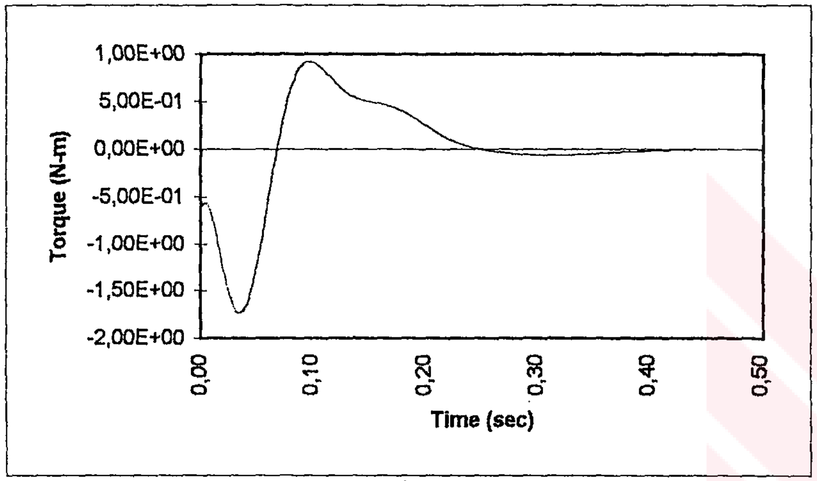


(b)

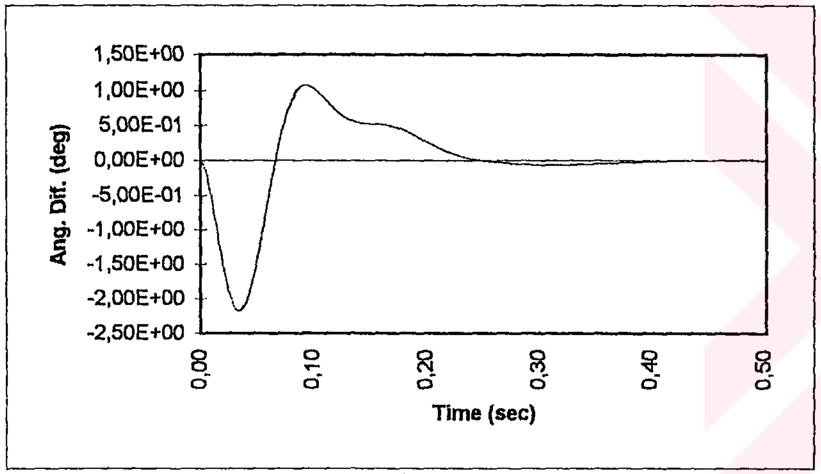


(c)

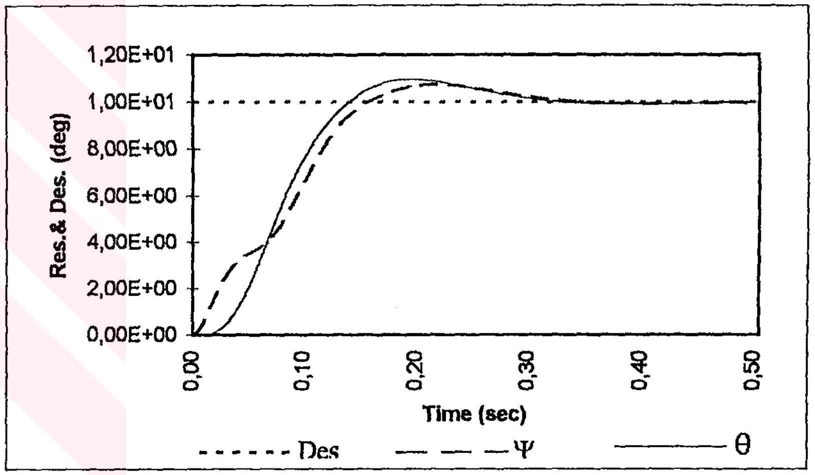
Figure 5.1.3 1R , $\alpha = 40$



(a)

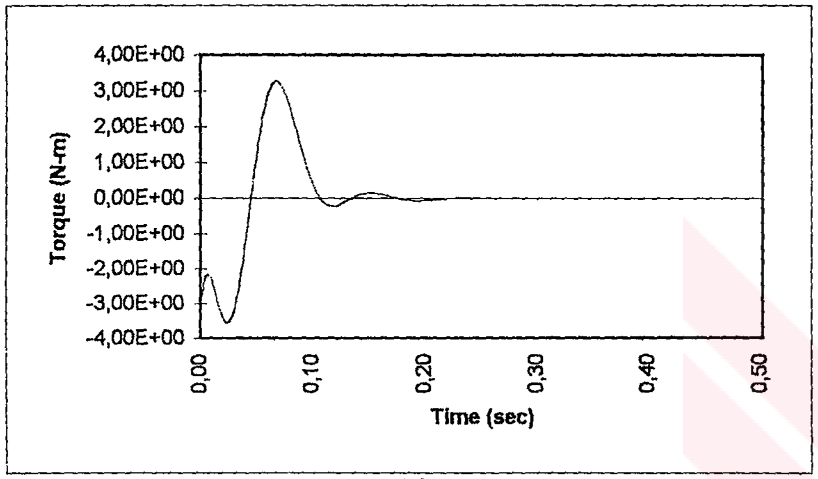


(b)

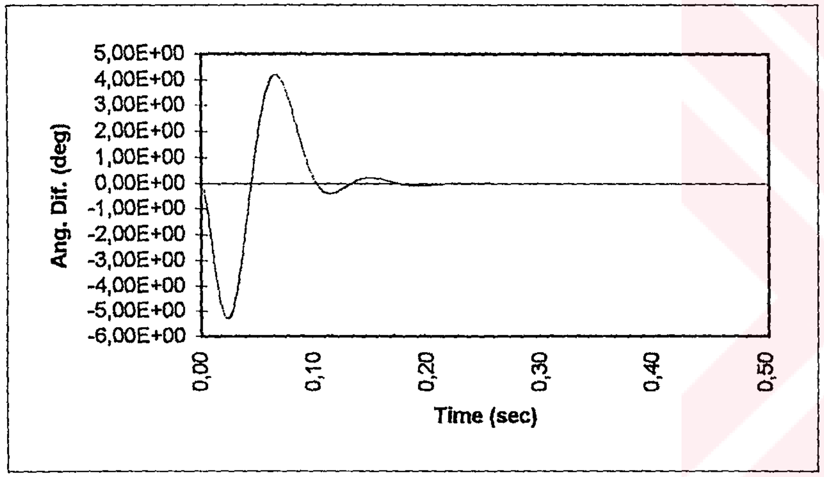


(c)

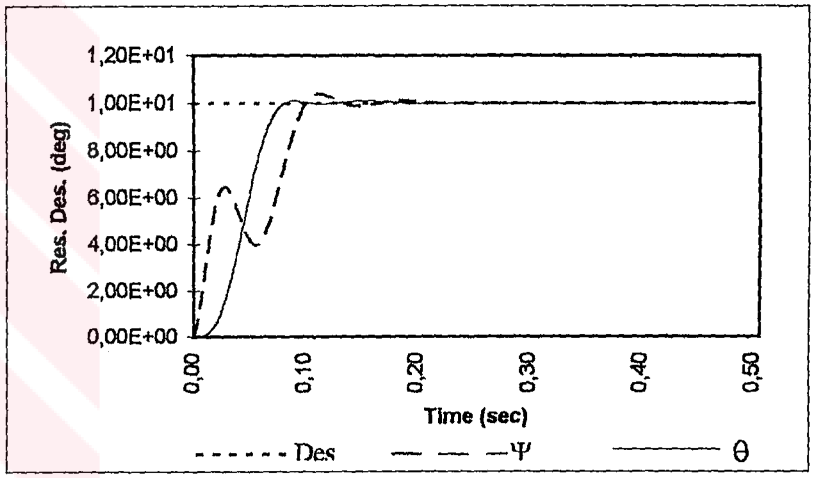
Figure 5.1.4 1R , α = 40 , with modeling error



(a)

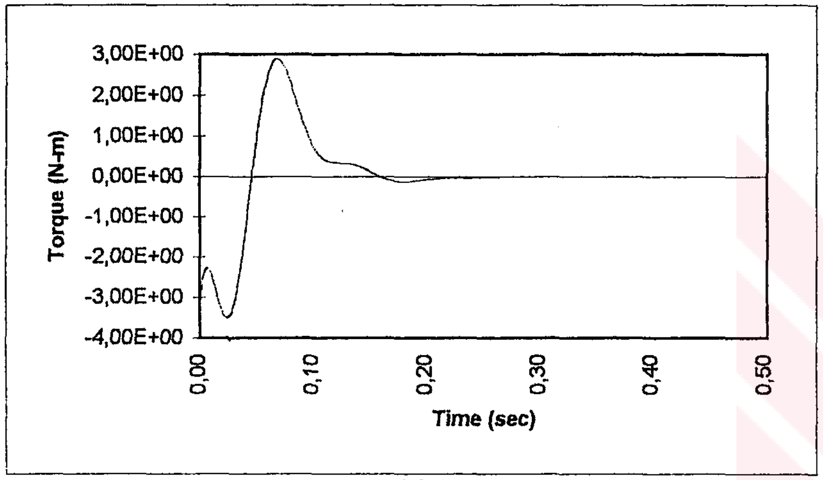


(b)

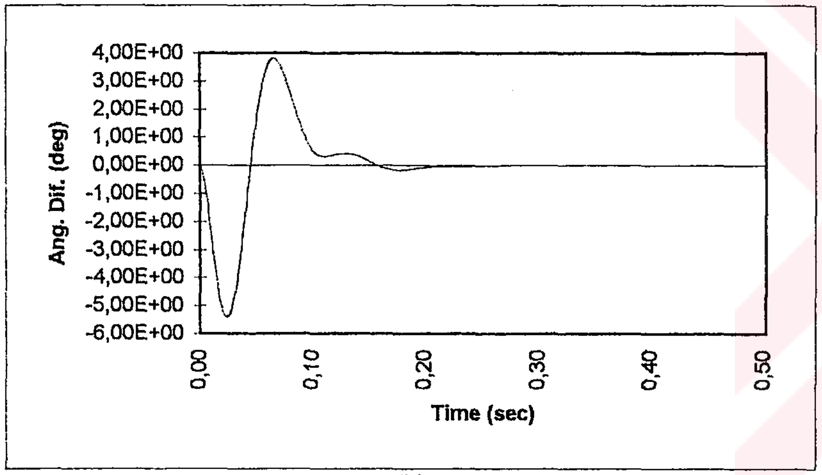


(c)

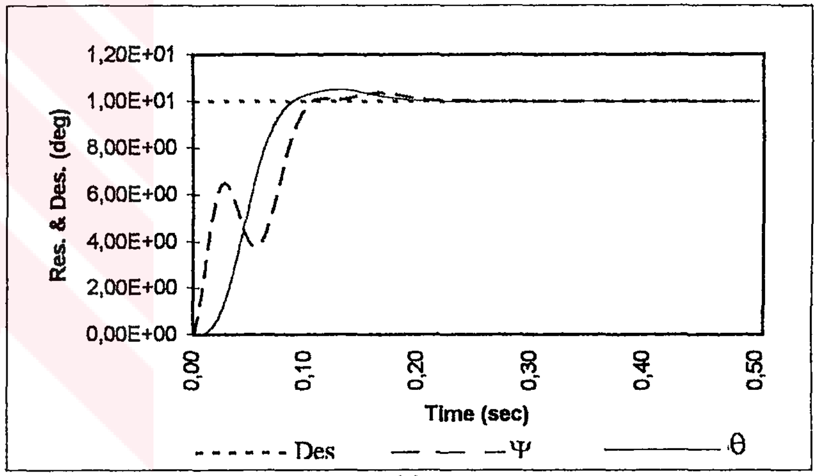
Figure 5.1.5 IR , $\alpha = 60$



(a)



(b)



(c)

Figure 5.1.6 1R , $\alpha = 60$, with modeling error

5.2 Case Study 2: 3-R Spatial Manipulator

A 3-R spatial robotic manipulator is shown in Figure 2.5.2. This manipulator configuration is one of the mostly used configurations in practical applications.

Using the assumptions related to the dynamics of the rotors, the dynamic equations of the 3-link manipulator can be written as follows ;

$$\begin{bmatrix} M_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & M_{22} & M_{23} & 0 & 0 & 0 \\ 0 & M_{32} & M_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & r^2\, I^A{}_{1yy} & 0 & 0 \\ 0 & 0 & 0 & 0 & r^2\, I^A{}_{2zz} & 0 \\ 0 & 0 & 0 & 0 & 0 & r^2\, I^A{}_{3zz} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \\ \ddot{\Psi}_1 \\ \ddot{\Psi}_2 \\ \ddot{\Psi}_3 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ B_5 \\ B_6 \end{bmatrix} + \begin{bmatrix} K_1(\theta_1 - \Psi_1) \\ K_2(\theta_2 - \Psi_2) \\ K_3(\theta_3 - \Psi_3) \\ K_1(\Psi_1 - \theta_1) \\ K_2(\Psi_2 - \theta_2) \\ K_3(\Psi_3 - \theta_3) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

(5.2.1)

where

$$M_{11} = \frac{1}{4}\, m^L{}_2\, L_2{}^2\, c^2\theta_2 + m^L{}_3\, (\frac{1}{4}\, L_3{}^2\, c^2\theta_{23} + L_2{}^2\, c^2\theta_2 + L_2\, L_3\, c\theta_2\, c\theta_{23})$$

$$+ \frac{1}{4}\, m^A{}_3\, L_2{}^2\, c^2\theta_2 + I^L{}_{1yy} + I^L{}_{2xx}\, s^2\theta_2 + I^L{}_{2yy}\, c^2\theta_2 + I^L{}_{3xx}\, s^2\theta_{23} + I^L{}_{3yy}\, c^2\theta_{23}$$

(5.2.2)

$$M_{22} = \frac{1}{4}\, m^L{}_2\, L_2{}^2 + m^L{}_3\, (\frac{1}{4}\, L_3{}^2 + L_2{}^2 + L_2\, L_3\, c\theta_3) + \frac{1}{4}\, m^A{}_3\, L_2{}^2 + I^L{}_{2zz} + I^L{}_{3zz}$$

(5.2.3)

$$M_{23} = M_{32} = \frac{1}{2}\, m^L_3 \left(\frac{1}{2} L_3^2 + L_2 L_3\, c\theta_3 \right) + I^L_{3zz} \tag{5.2.4}$$

$$M_{33} = \frac{1}{4}\, m^L_3 L_3^2 + I^L_{3zz} \tag{5.2.5}$$

$$\begin{aligned} B_1 = {} & 2\, \dot{\theta}_1 \dot{\theta}_2 \{ -\frac{1}{4} m^L_2 L_2^2\, c\theta_2\, s\theta_2 + \frac{1}{2} m^L_3 [-\frac{1}{2} L_3^2\, c\theta_{23}\, s\theta_{23} - 2 L_2^2\, c\theta_2\, s\theta_2 \\ & - L_2 L_3 (s\theta_2\, c\theta_{23} + c\theta_2\, s\theta_{23})] - \frac{1}{4} m^A_3 L_2^2\, c\theta_2\, s\theta_2 + I^L_{2xx}\, c\theta_2\, s\theta_2 \\ & - I^L_{2yy}\, c\theta_2\, s\theta_2 + I^L_{3xx}\, c\theta_{23}\, s\theta_{23} - I^L_{3yy}\, c\theta_{23}\, s\theta_{23} \} \\ & + 2\, \dot{\theta}_1 \dot{\theta}_3 \{ \frac{1}{2} m^L_3 [-\frac{1}{2} L_3^2\, c\theta_{23}\, s\theta_{23} - L_2 L_3\, c\theta_2\, s\theta_{23}] + I^L_{3xx}\, c\theta_{23}\, s\theta_{23} \\ & - I^L_{3yy}\, c\theta_{23}\, s\theta_{23} \} \end{aligned} \tag{5.2.6}$$

$$\begin{aligned} B_2 = {} & \dot{\theta}_2 \dot{\theta}_3\, m^L_3 L_2 L_3\, s\theta_{23} - \frac{1}{2} \dot{\theta}_3^2\, m^L_3 L_2 L_3\, s\theta_3 \\ & - \dot{\theta}_1^2 [-\frac{1}{4} m^L_2 L_2^2\, c\theta_2\, s\theta_2 + \frac{1}{2} m^L_3 (-\frac{1}{2} L_3^2\, c\theta_{23}\, s\theta_{23} - 2 L_2^2\, c\theta_2\, s\theta_2 \\ & - L_2 L_3\, s\theta_2\, c\theta_{23} - L_2 L_3\, c\theta_2\, s\theta_{23}) - \frac{1}{4} m^A_3 L_2^2\, c\theta_2\, s\theta_2 - I^L_{2xx}\, c\theta_2\, s\theta_2 + I^L_{2yy}\, c\theta_2 \\ & s\theta_2 - I^L_{3xx}\, c\theta_{23}\, s\theta_{23} + I^L_{3yy}\, c\theta_{23}\, s\theta_{23}] \\ & + \frac{1}{2} m^L_2\, g\, c\theta_2 + m^L_3\, g\, L_2\, c\theta_2 + m^A_3\, g\, L_2\, c\theta_2 + \frac{1}{2} m^L_3\, g\, c\theta_{23} \end{aligned} \tag{5.2.7}$$

$$B_3 = -\frac{1}{2}\dot{\theta}_2\dot{\theta}_3 m^L{}_3 L_2 L_3 s\theta_3 - \dot{\theta}_1{}^2 [\frac{1}{2} m^L{}_3 (-\frac{1}{2} L_3{}^2 c\theta_{23} s\theta_{23} - L_2 L_3 c\theta_2 s\theta_{23})$$

$$-\frac{1}{2} m^L{}_3 L_2 L_3 s\theta_3 - 2 I^L{}_{3xx} c\theta_{23} s\theta_{23} + I^L{}_{3yy} c\theta_{23} s\theta_{23}]$$

$$+\frac{1}{2}\dot{\theta}_2\dot{\theta}_3 m^L{}_3 L_2 L_3 s\theta_{23} + \frac{1}{2} m^L{}_3 g L_3 c\theta_{23} \qquad (5.2.8)$$

$$B_4 = B_5 = B_6 = 0 \qquad (5.2.9)$$

For this manipulator two different trajectories are commanded, which are explained in two different sections.

In both cases the manipulator is at position (θ_{1in} = -30°, θ_{2in} = 120°, θ_{3in} = -150°) which corresponds to initial end-effector position (0.31699 m , 0.61603 m, 0.18301 m).

When there is no initial position error the desired motion starts from the initial position, i.e. $x_0^d = 0.31699$ m, $y_0^d = 0.61603$ m, $z_0^d = 0.18301$ m. The initial position errors and modeling errors considered in the numeraical experiments are as follows;

Initial Position Errors are

$x_0^d = 0.30$ m, for Trajectory 1 (Cycloidal deployment motion)

$y_0^d = 0.70$ m for Trajectory 1

$z_0^d = 0.30$ m for Trajectory 1

$x_0^d = 0.41699$ m, for Trajectory 2 (Circular trajectory)

$y_0^d = 0.61603$ m for Trajectory 2

$z_0^d = 0.18301$ m for Trajectory 2

i.e., there is, -0.01699 m for "Trajectory 1", 0.1 m for "Trajectory 2", position error in x-direction; 0.08397 m for "Trajectory 1", 0.1 m for "Trajectory 2", position error in y-direction; -0.11699 m for "Trajectory 1" position error in z-direction.

Modeling Error: Modeled mass properties are 10 % less than actual .

Results of the simulations are presented after each section. Each simulation is presented by four graphs. The first graph (a) shows the actual and desired motions of the end-effector in X, Y, and Z coordinates, the second graph (b) shows the error between the actual and desired positions of the end-effector, the third graph (c) shows the torque applied by the each actuator, and the fourth graph (d) shows the deflections of the torsional springs (i.e. the angular difference between the rigid and flexible system variables of each actuator).

5.2.1 Trajectory 1

The first commanded motion is given in operational space . This motion is a cycloidal deployment motion where the end effector is desired to deploy from position (x_0^d, y_0^d, z_0^d) to (x_0^d, $y_0^d+\Delta y$, $z_0^d+\Delta z$). The period of the desired motion is T.

$$x^d(t) = x_0^d$$

$$y^d(t) = y_0^d + \frac{\Delta y}{T}\left(t - \frac{T}{2\pi}\left(\sin\frac{2\pi\, t}{T}\right)\right.$$

$$z^d(t) = z_0^d + \frac{\Delta z}{T}\left(t - \frac{T}{2\pi}\left(\sin\frac{2\pi\, t}{T}\right)\right.$$

The input data used in the simulations are given as follows;

$m^L{}_1$	$= 18$ kg
$m^L{}_2$	$= 9$ kg
$m^L{}_3$	$= 9$ kg
$m^A{}_1$	$= 3.9\ 10^{-2}$ kg
$m^A{}_2$	$= 1.3\ 10^{-2}$ kg
$m^A{}_3$	$= 1.3\ 10^{-2}$ kg
L_1	$= 0.25$ m
L_2	$= 1$ m
L_3	$= 1$ m
Δx	$= 0$ m
Δy	$= 0.5$ m
Δz	$= -0.5$ m

$K_1 = K_2 = K_3 = 5000$ N / m

α $= 50$ & 70 rad / sec

T $= 0.5$ & 1 sec

Simulation results can be summarized as follows:

The response with $\alpha = 50$ is given in Figure 5.2.1.1. Fig. 5.2.1.2 shows the resultant graphs if the assumptions made in Chapter II were not taken into account. As it can be seen from both figures, the assumptions are valid and do not cause changes in the simulations. The initial and final spring deflections are nonzero because of the gravitational forces. Initial position errors cause larger initial torques to be applied and larger initial angular differences (Fig. 5.2.1.3). Presence of modeling error causes larger torques to be applied, larger angular differences, and larger tracking errors during motion. Also a steady-state position error occurs. Fig. 5.2.1.4 shows that presence of both errors cause increases at applied torques, increases at spring deflections both at the beginning of the motion and during motion, larger tracking errors and steady-state position errors.

Simulations with $\alpha = 70$ are given in Figures from 5.2.1.5 to 5.2.1.8. An increase in α decreases tracking errors but increases applied torques and angular differences.

Figures from 5.2.1.9 to 5.2.1.16 show simulation results when $T = 1$ sec. Increasing T from 0.5 sec to 1 sec decreases tracking errors, applied torques and angular differences significantly.

It should be noted that, the differences between the patterns of the control torques and the elastic deflections show the importance of including the flexible joint dynamics into the model.

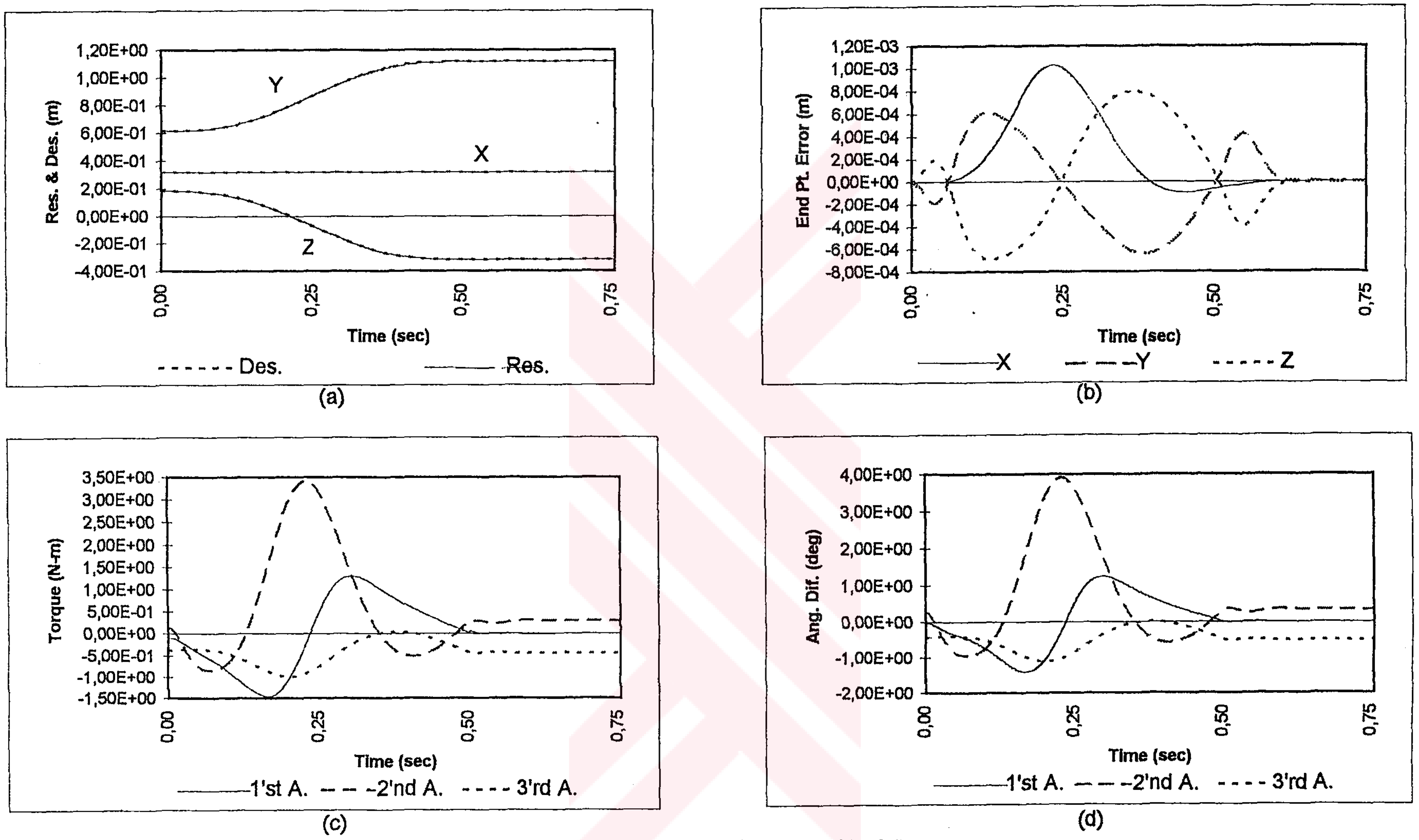


Figure 5.2.1.1 3R , Tr. 1 , $\alpha = 50$, T = 0.5 sec

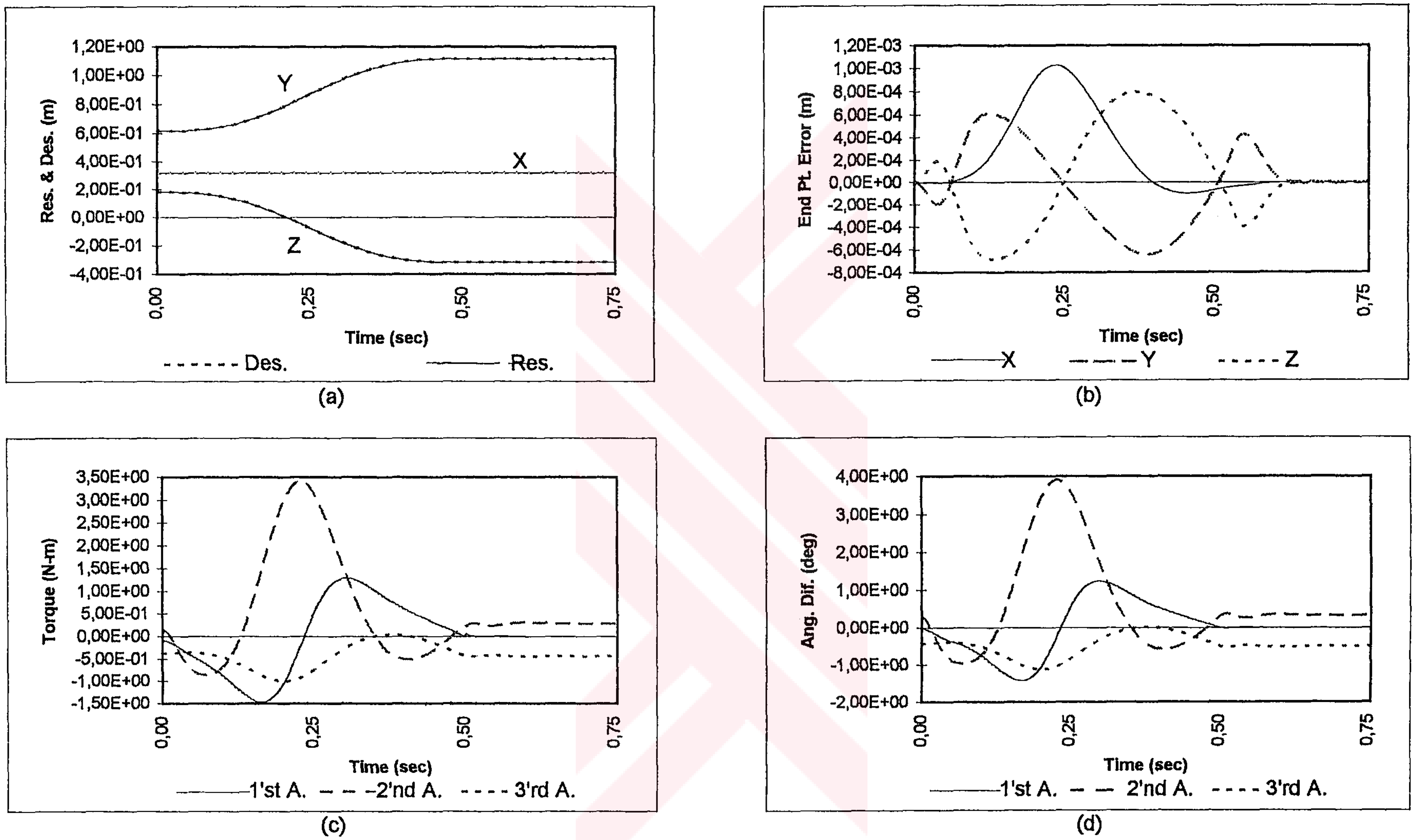
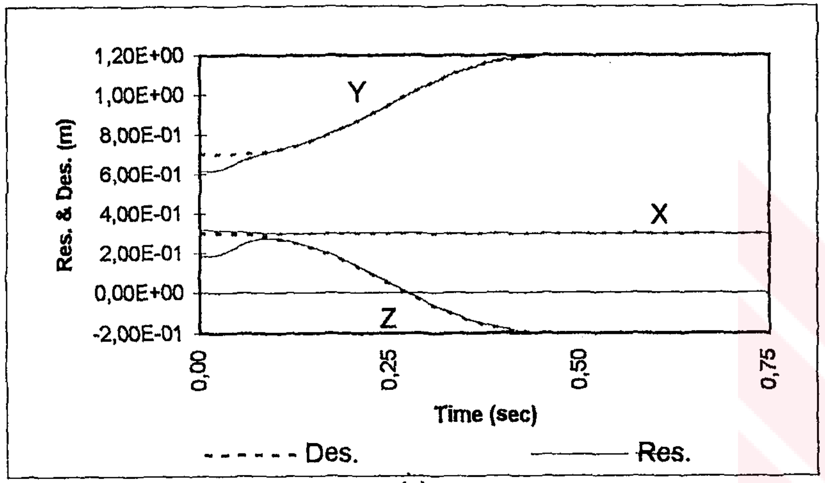
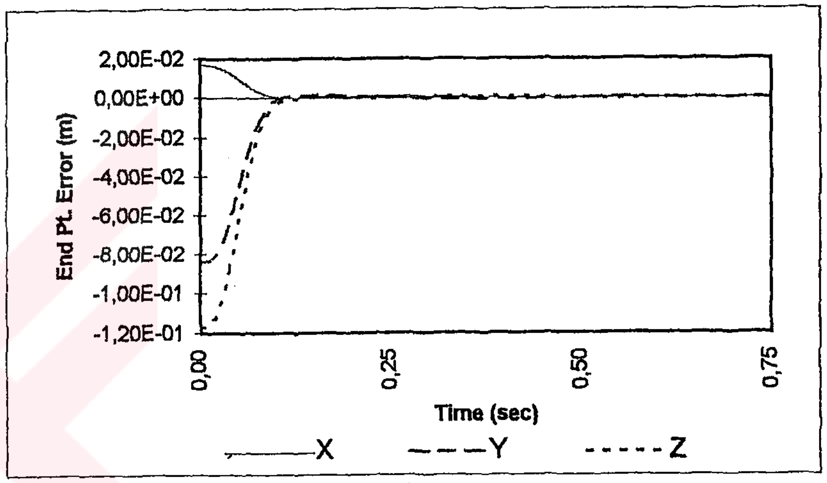


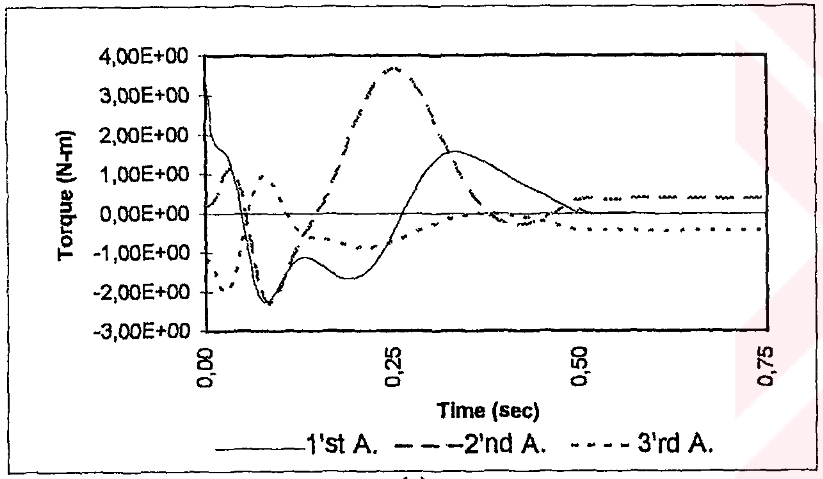
Figure 5.2.1.2 $\alpha = 50$, T = 0.5 sec without considering the assumptions explained in Ch. II



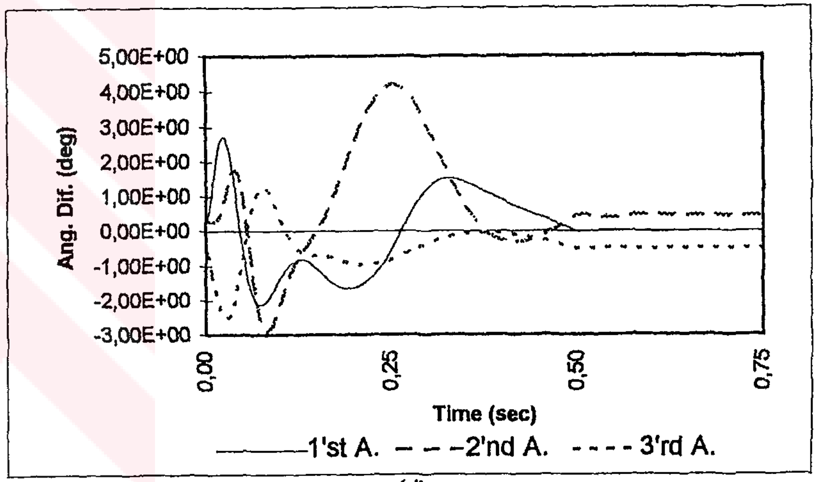
(a)



(b)



(c)



(d)

Figure 5.2.1.3 3R , Tr. 1 , $\alpha = 50$, $T = 0.5$ sec , with initial position error

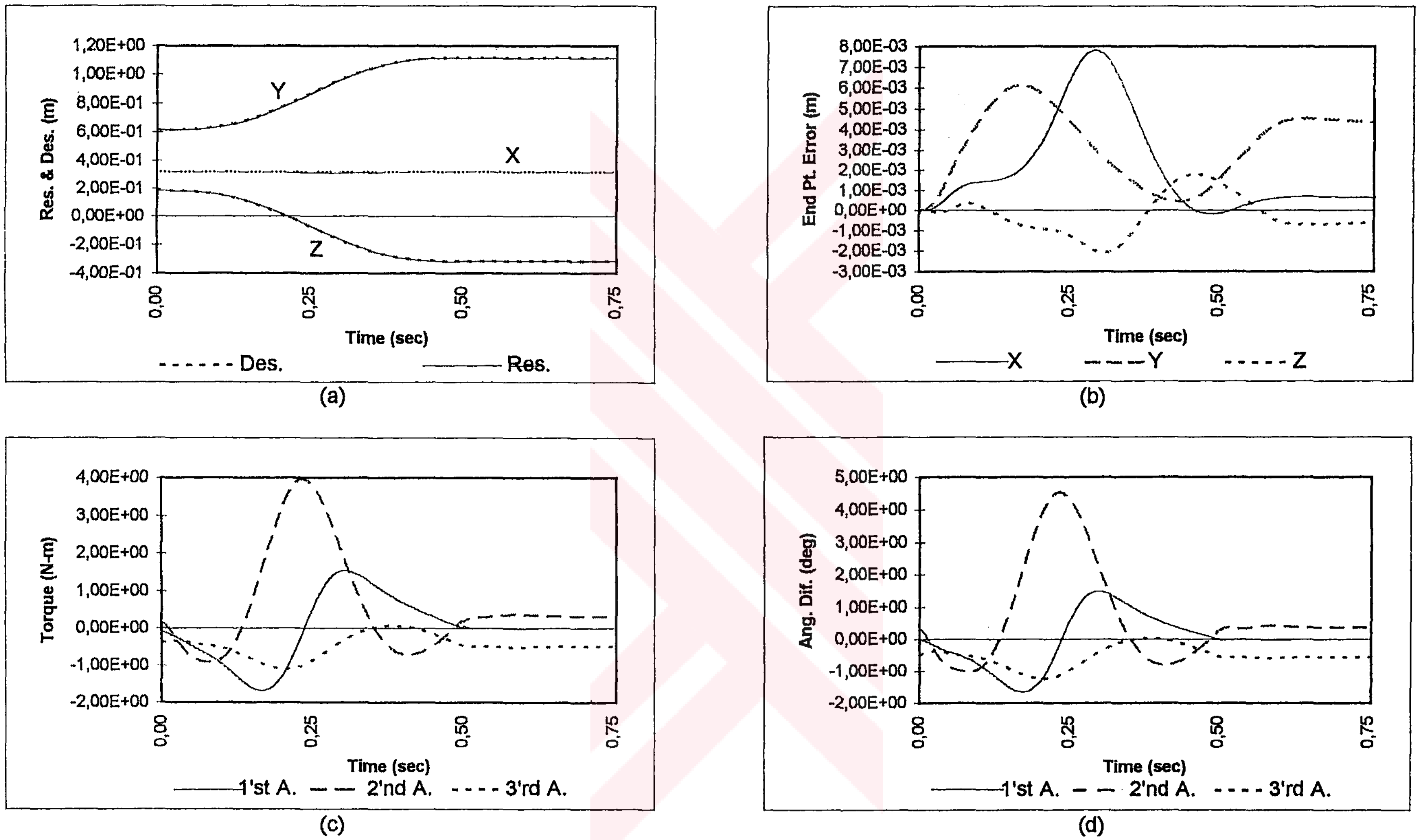
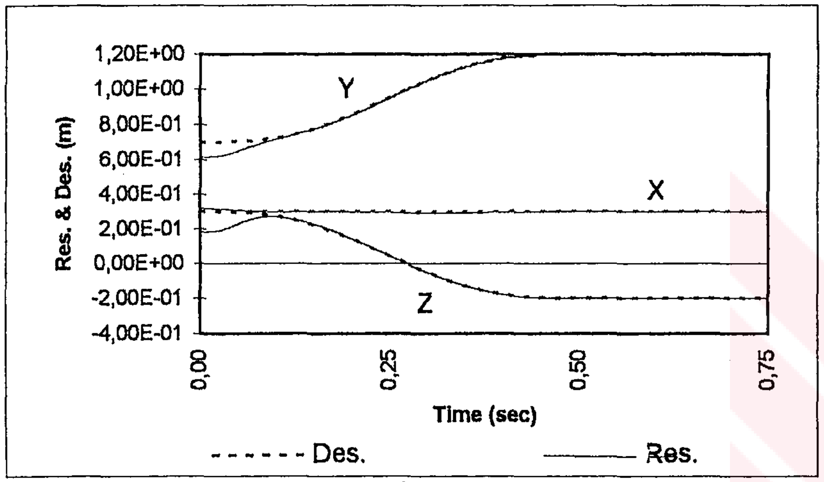
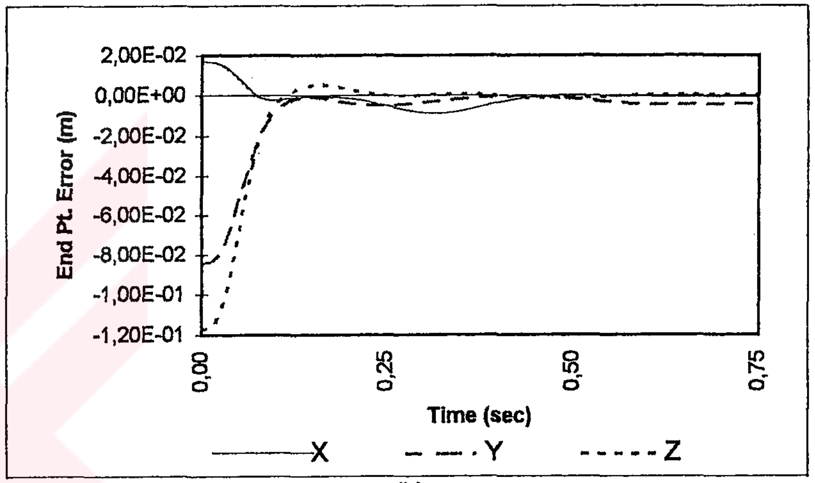


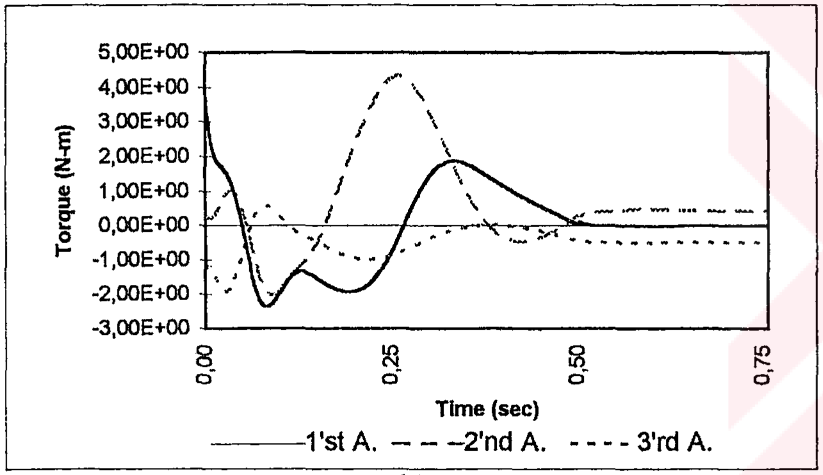
Figure 5.2.1.4 3R , Tr. 1 , $\alpha = 50$, $T = 0.5$ sec , with modeling error



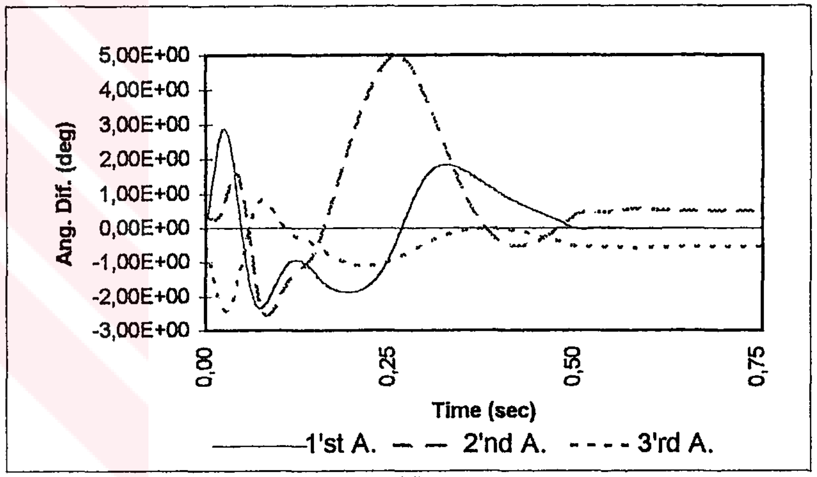
(a)



(b)



(c)



(d)

Figure 5.2.1.5 3R , Tr. 1 , $\alpha = 50$; T = 0.5 sec with modeling error , with initial position error

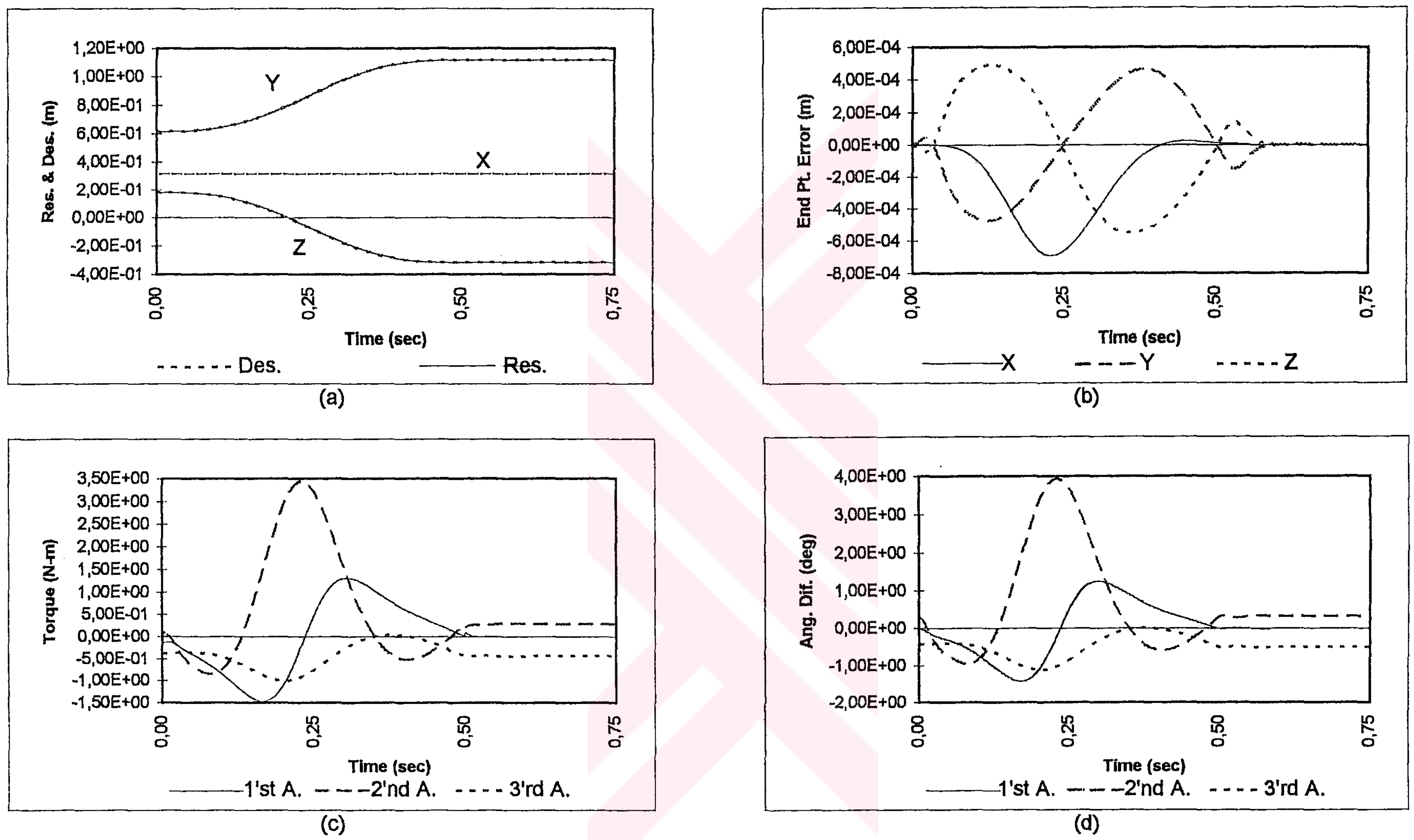
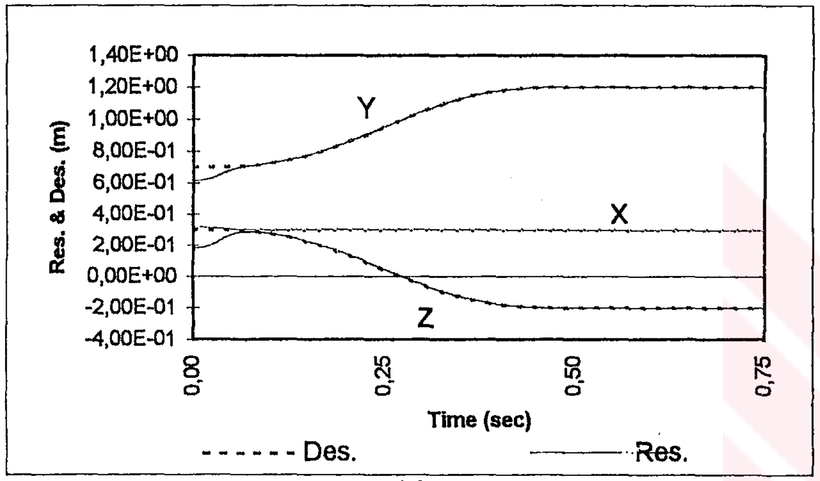
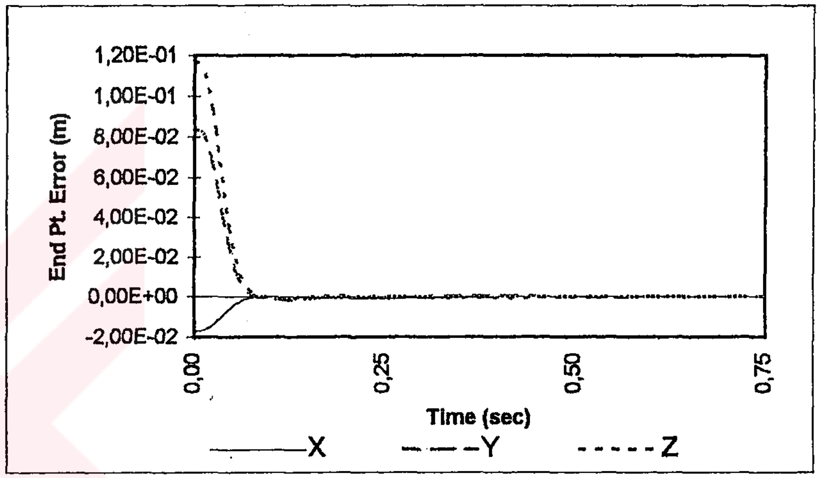


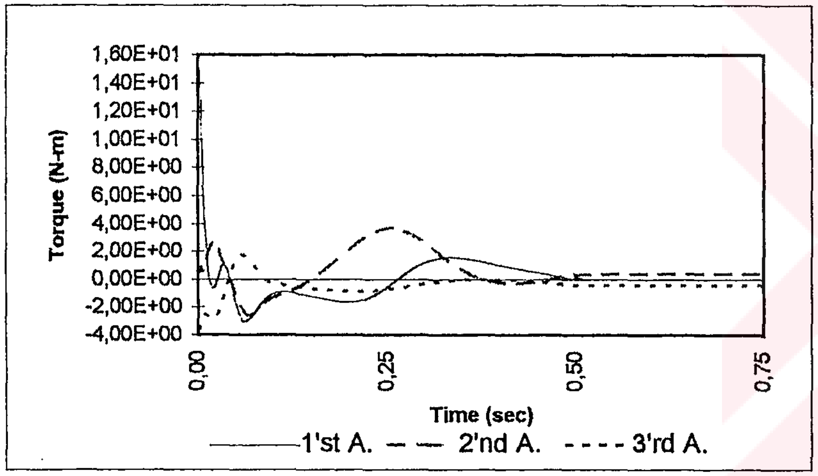
Figure 5.2.1.6 3R , Tr. 1 , $\alpha = 70$, $T = 0.5$ sec



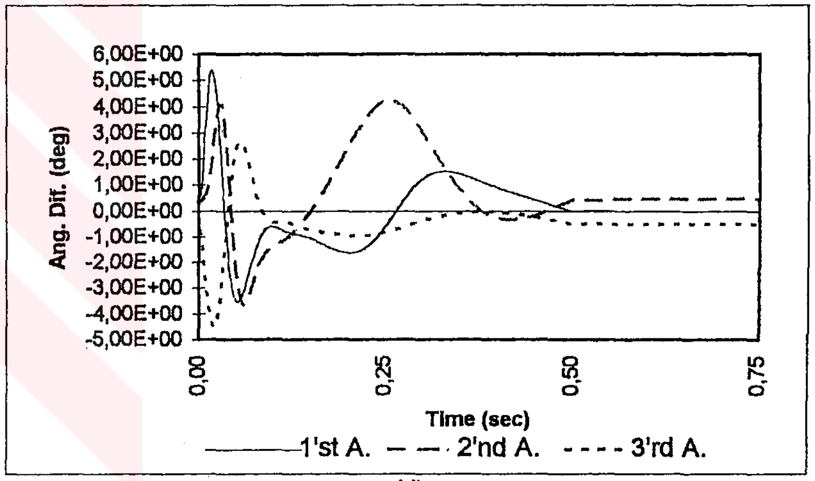
(a)



(b)



(c)



(d)

Figure 5.2.1.7 3R , Tr. 1 , $\alpha = 70$, $T = 0.5$ sec , with initial position error

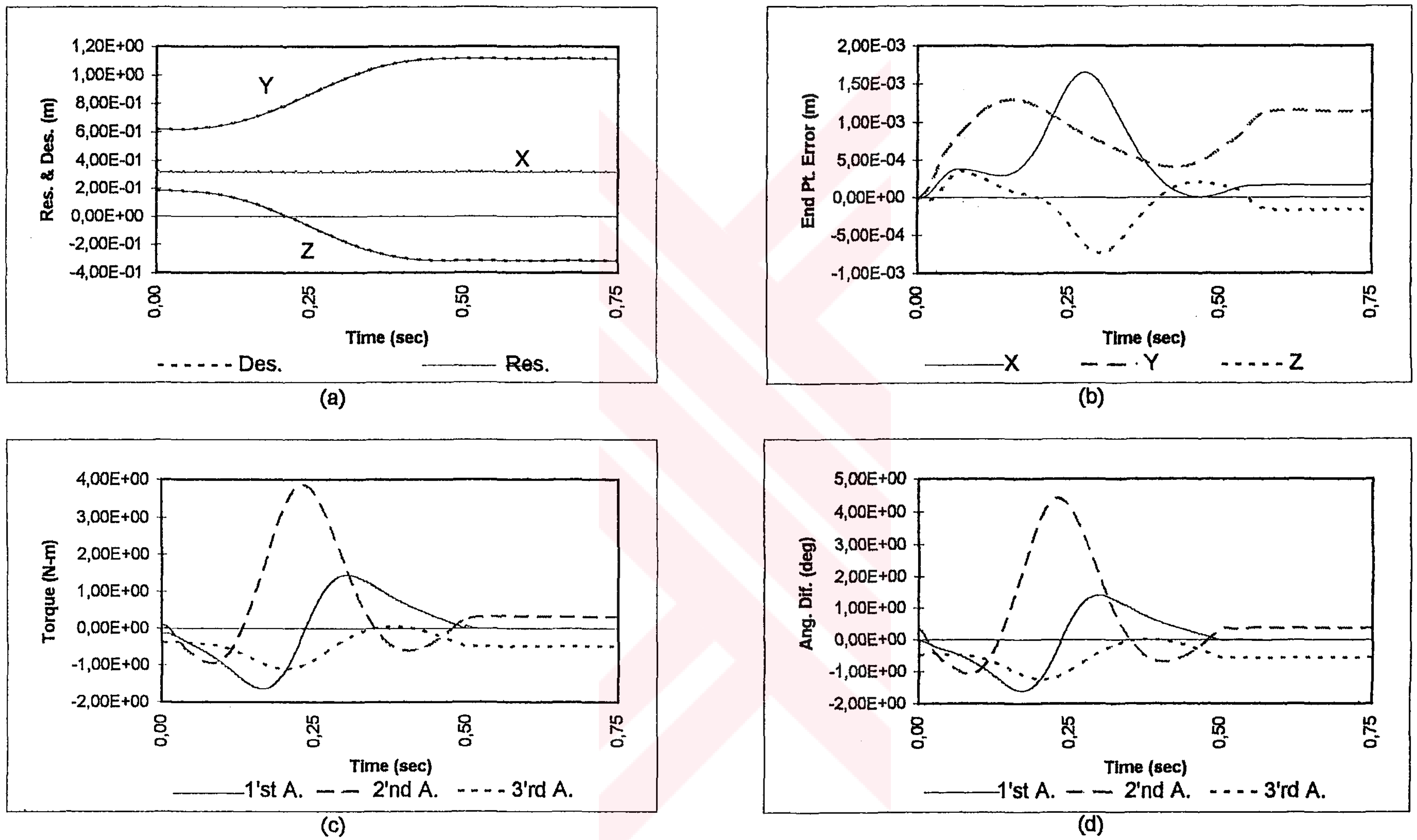
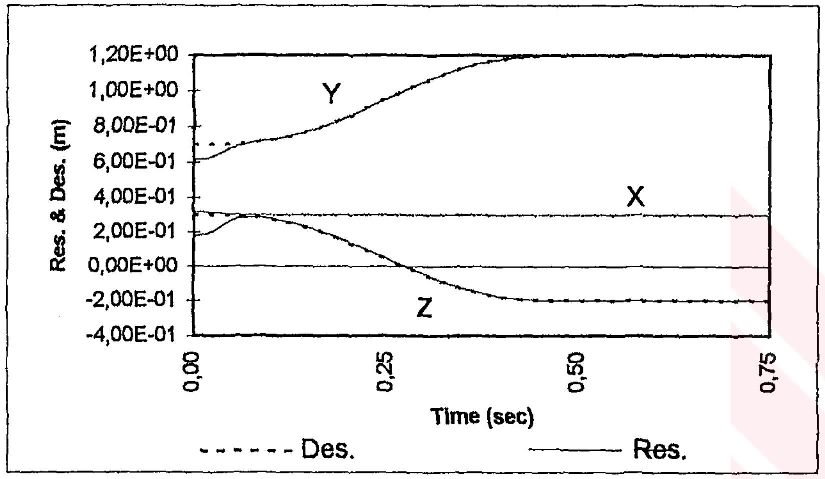
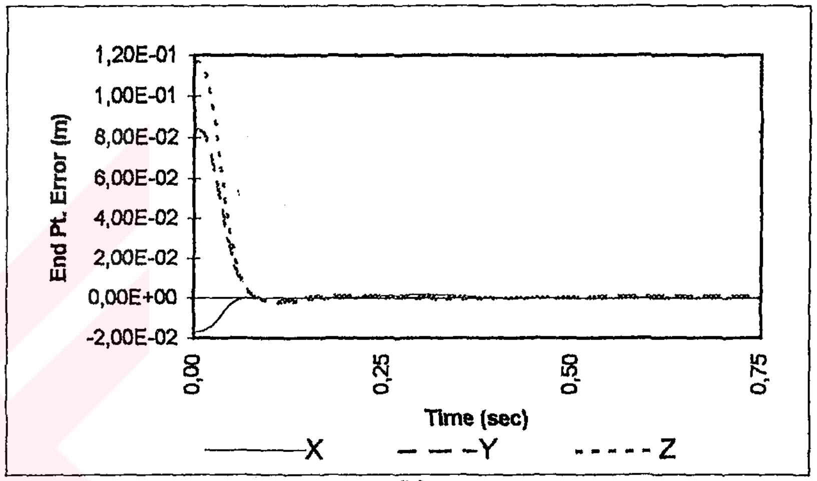


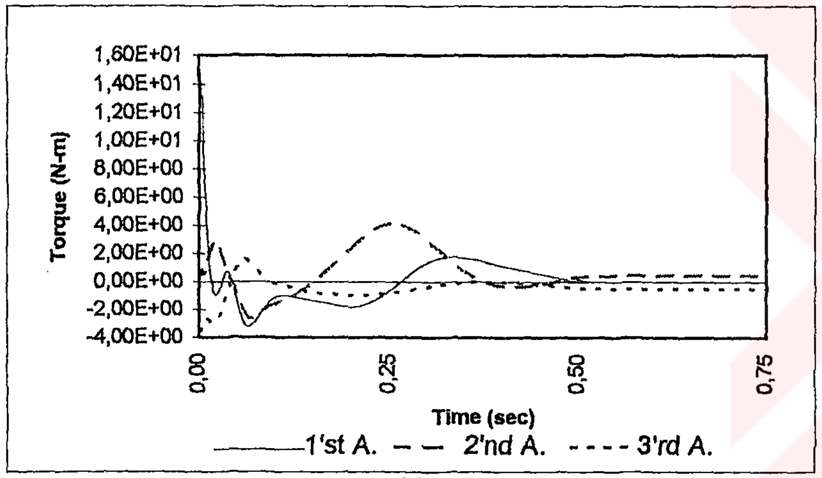
Figure 5.2.1.8 3R, Tr. 1, $\alpha = 70$, $T = 0.5$ sec, with modeling error



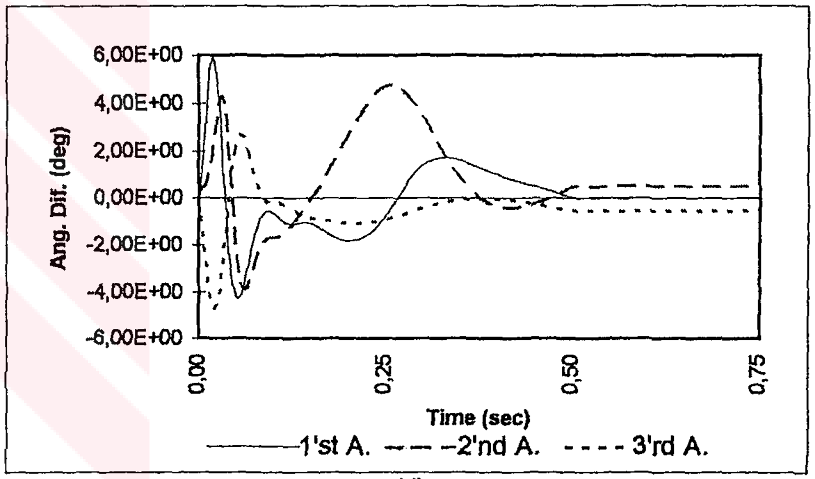
(a)



(b)



(c)



(d)

Figure 5.2.1.9 3R, Tr. 1, $\alpha = 70$ $T = 0.5$ *sec* , *with modeling error* , with initial position Error

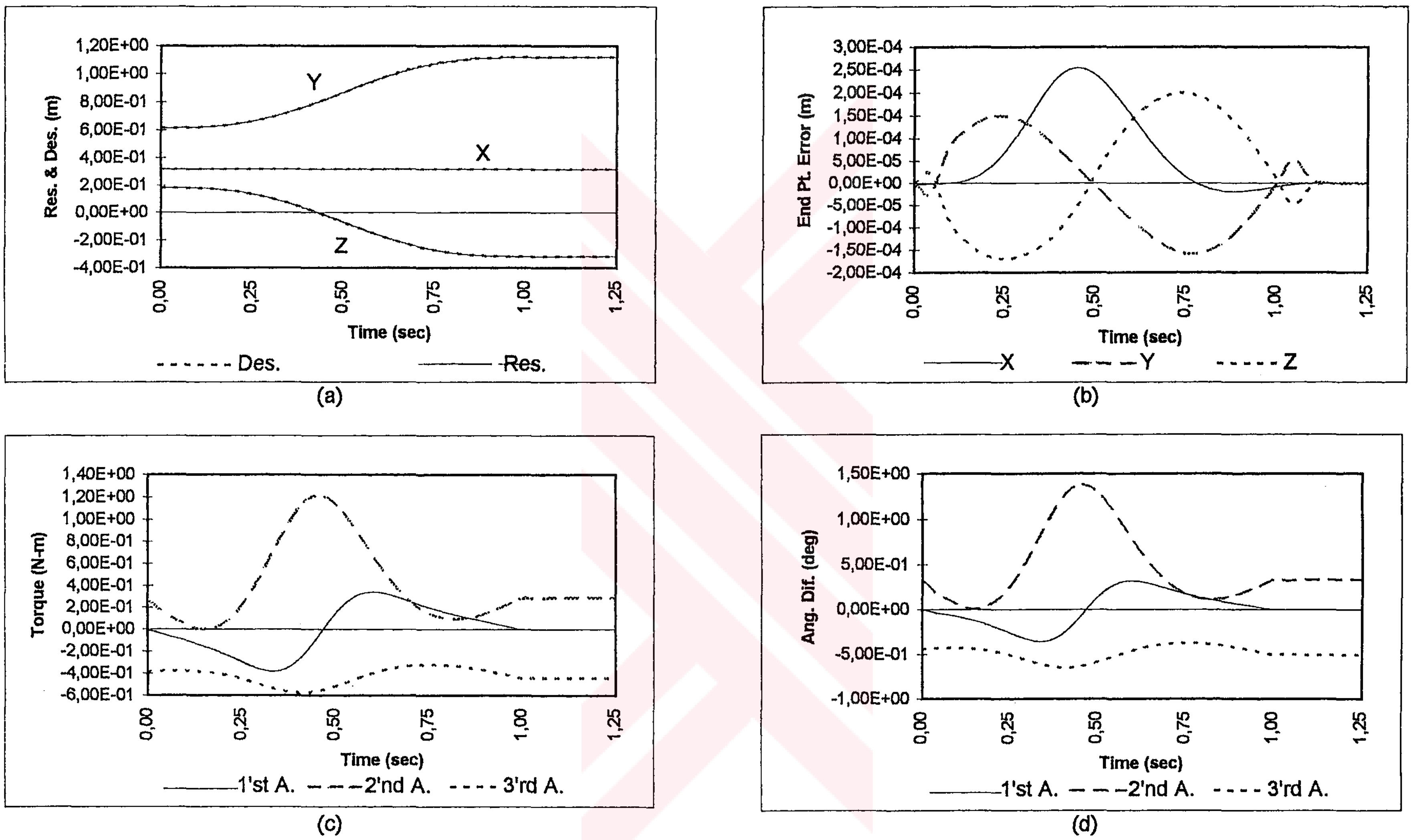


Figure 5.2.1.10 3R , Tr. 1 , $\alpha = 50$, T = 1 sec

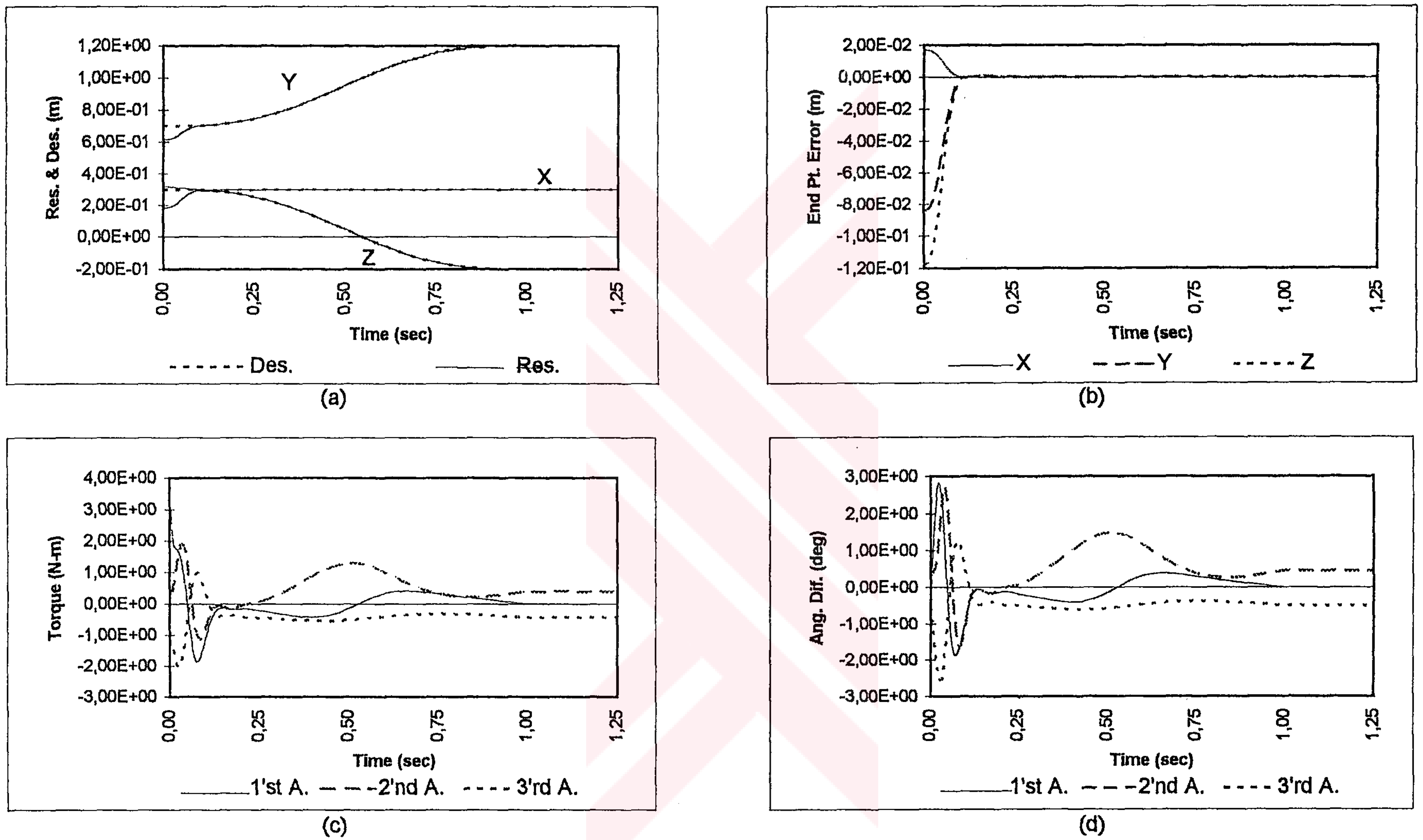


Figure 5.2.1.11 3R , Tr. 1 , $\alpha = 50$, $T = 1$ sec , with initial position error

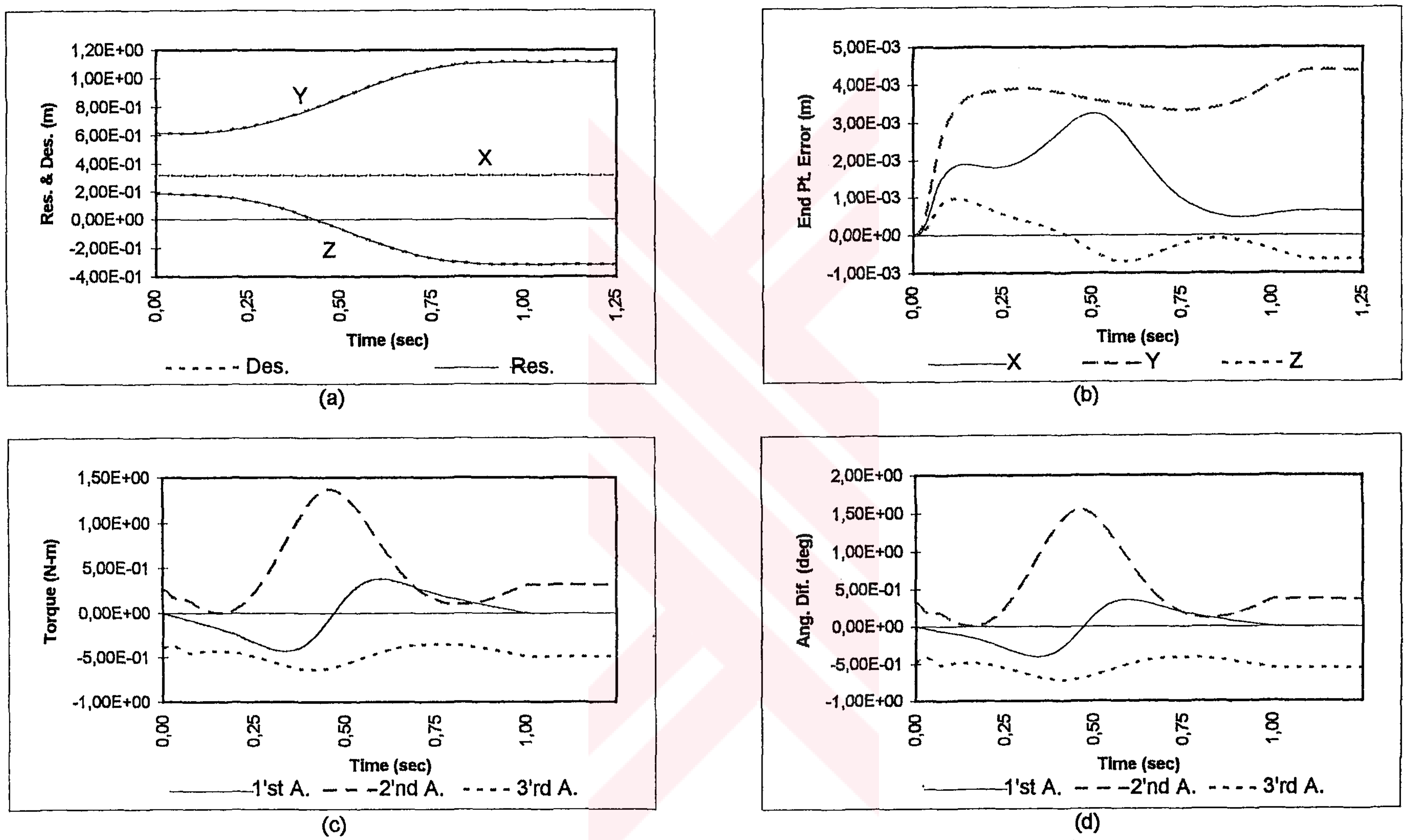


Figure 5.2.1.12 3R , Tr. 1 , $\alpha = 50$, T = 1 sec , with modeling error

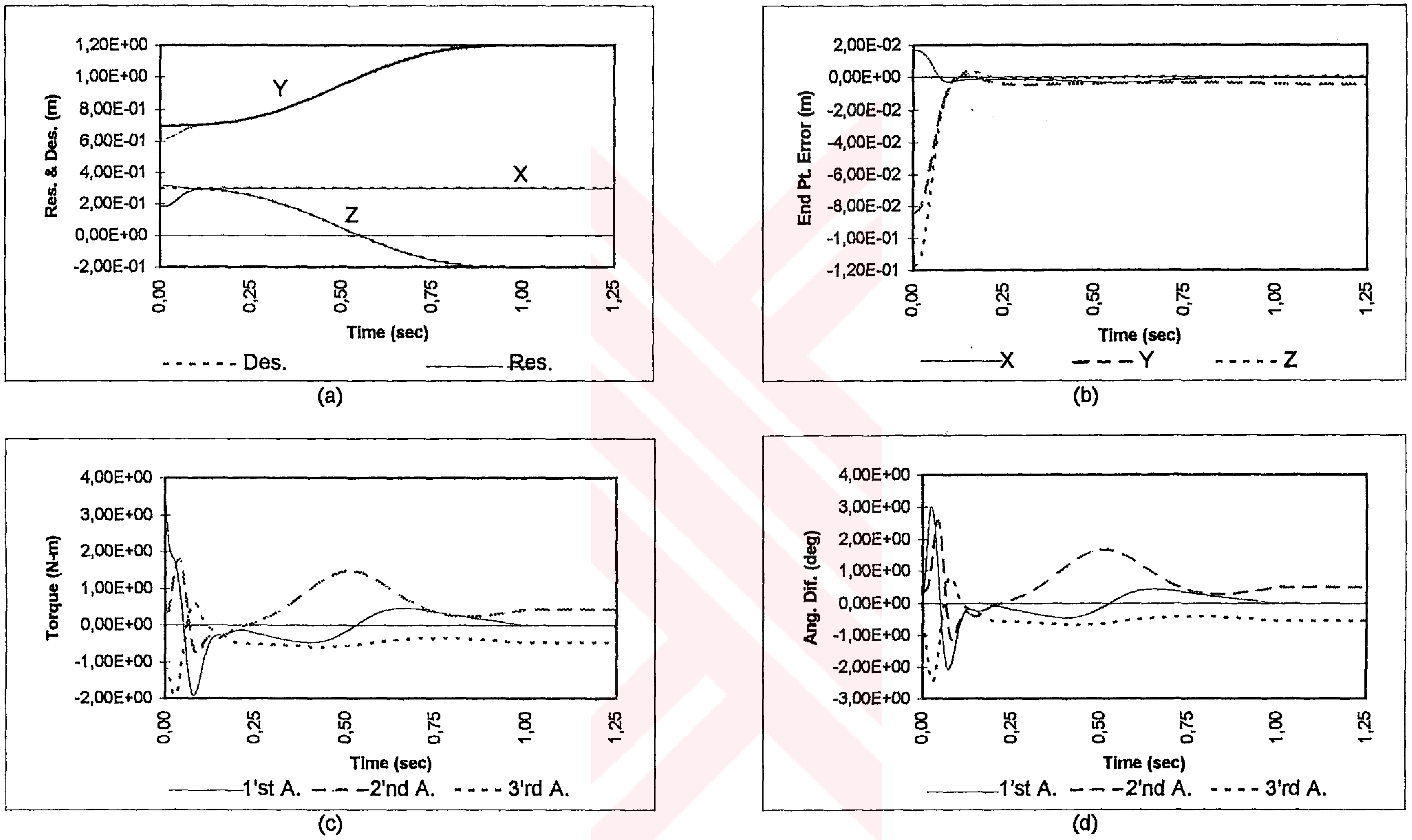
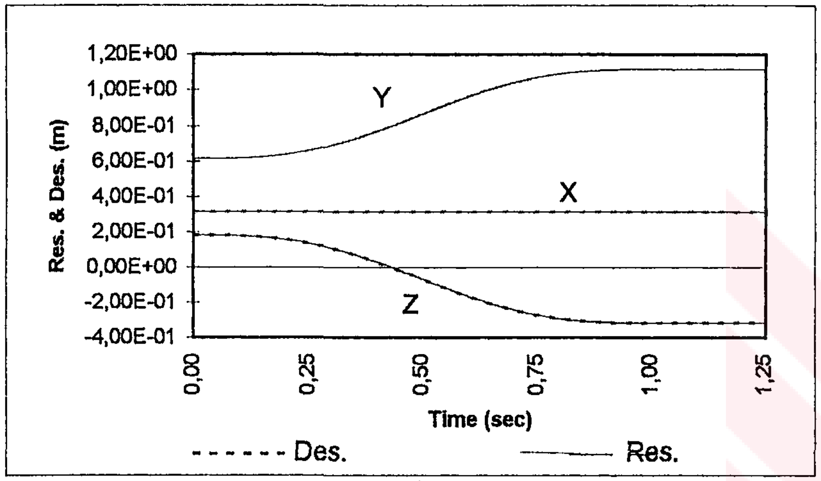
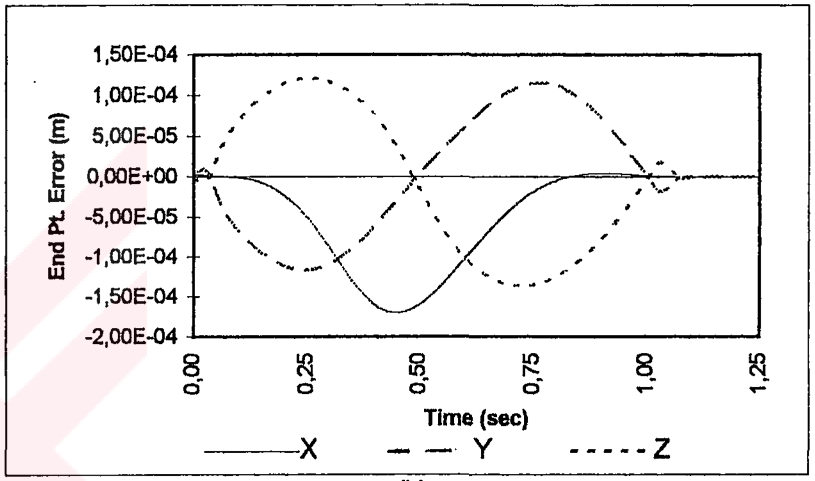


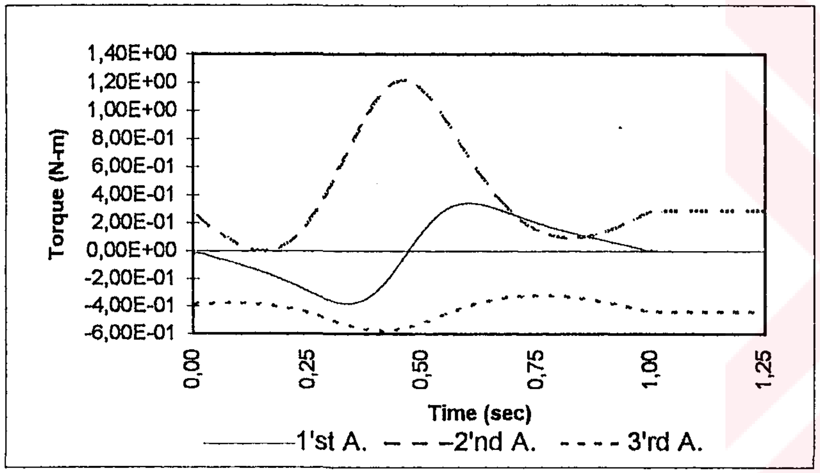
Figure 5.2.1.13 3R , Tr. 1 , $\alpha = 50$, T = 1 sec , with modeling error , with initial position error



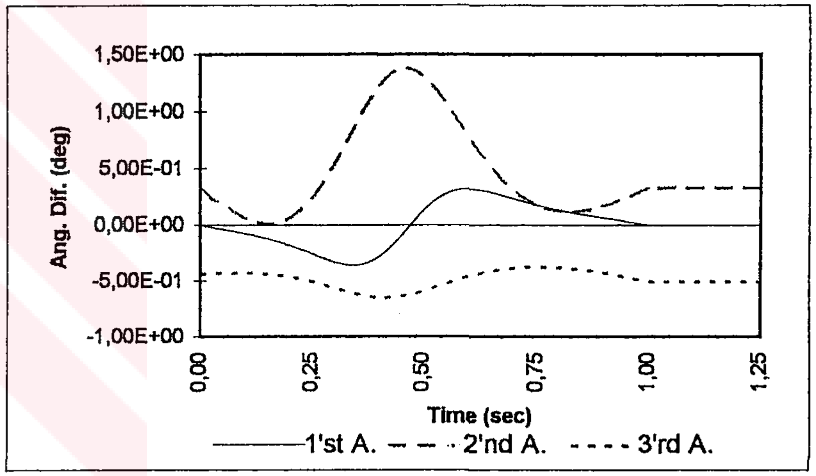
(a)



(b)

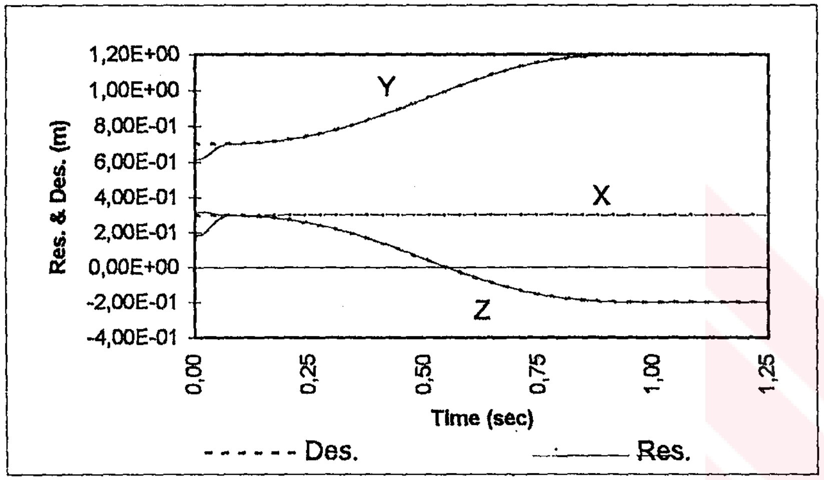


(c)

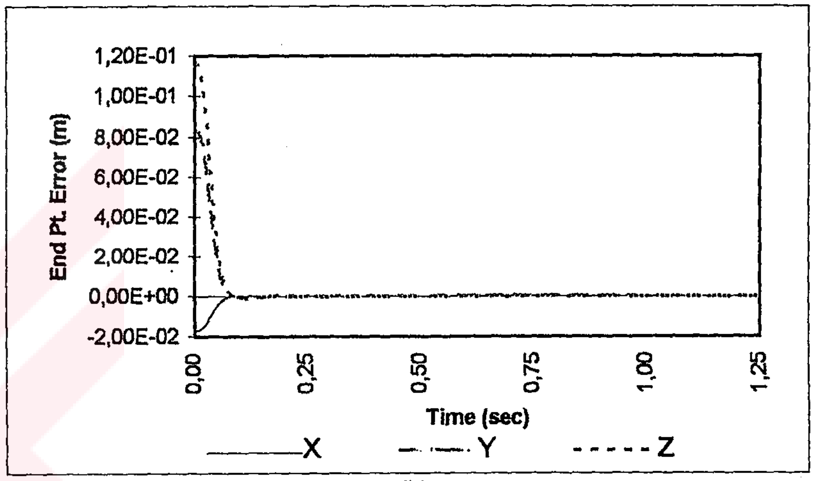


(d)

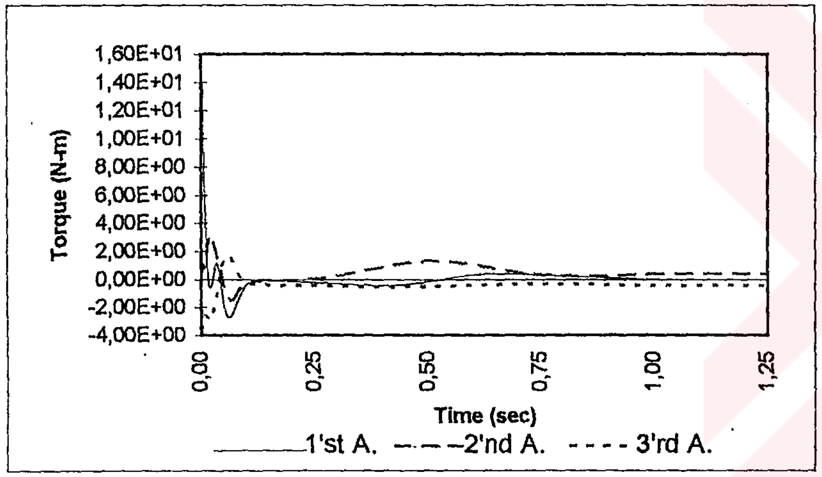
Figure 5.2.1.14 3R , Tr. 1 , $\alpha = 70$, T = 1 sec



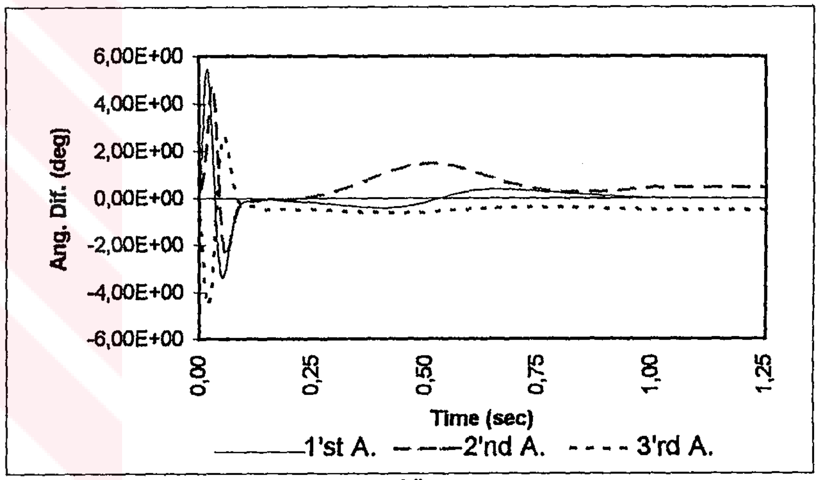
(a)



(b)

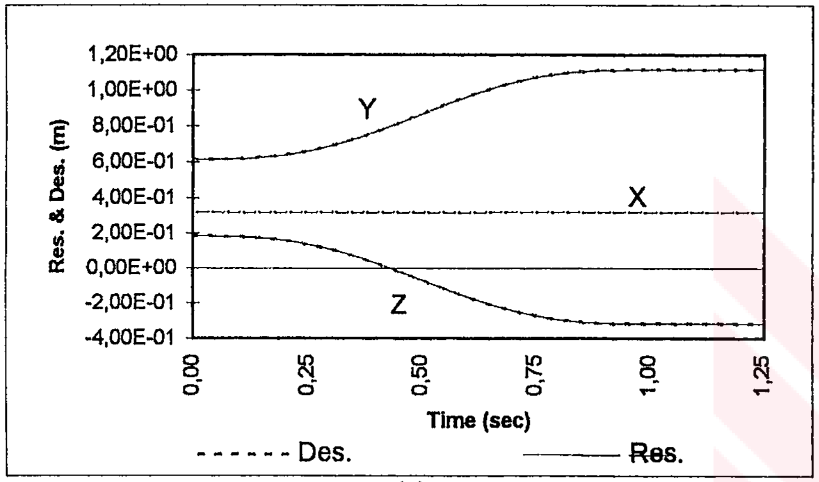


(c)

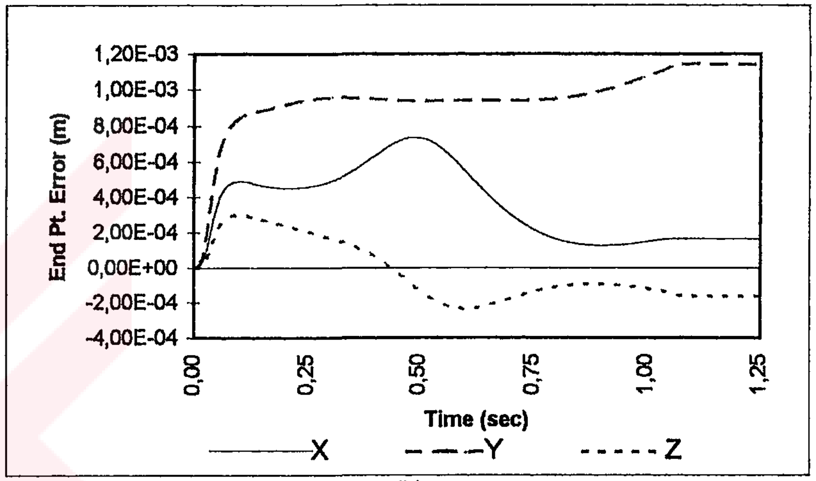


(d)

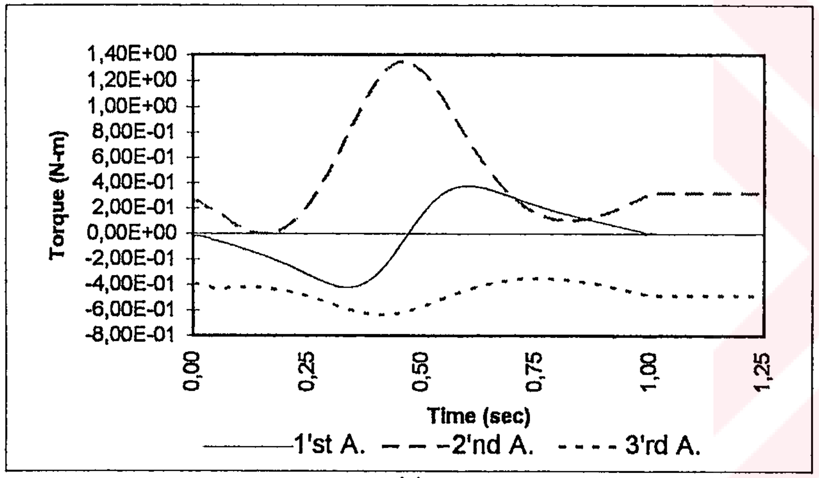
Figure 5.2.1.15 3R , Tr. 1 , $\alpha = 70$, $T = 1$ sec , with initial position error



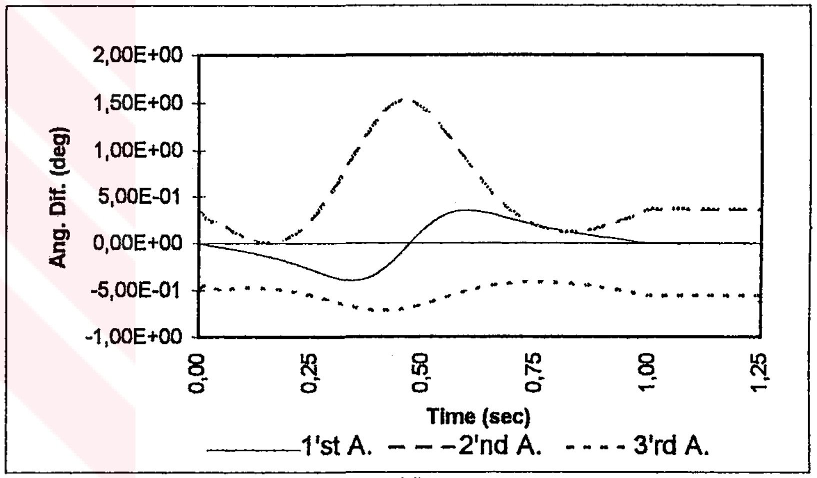
(a)



(b)

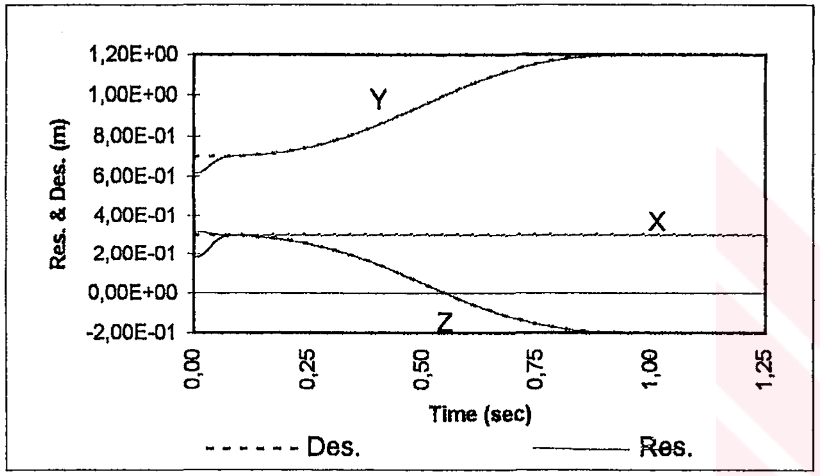


(c)

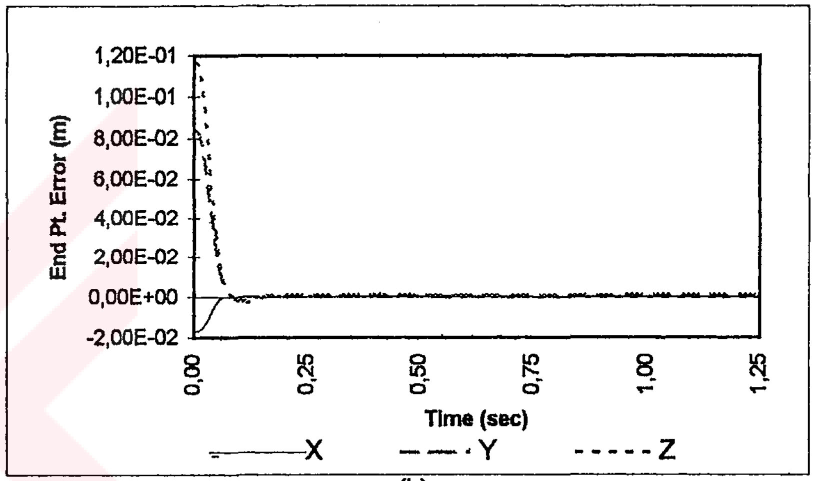


(d)

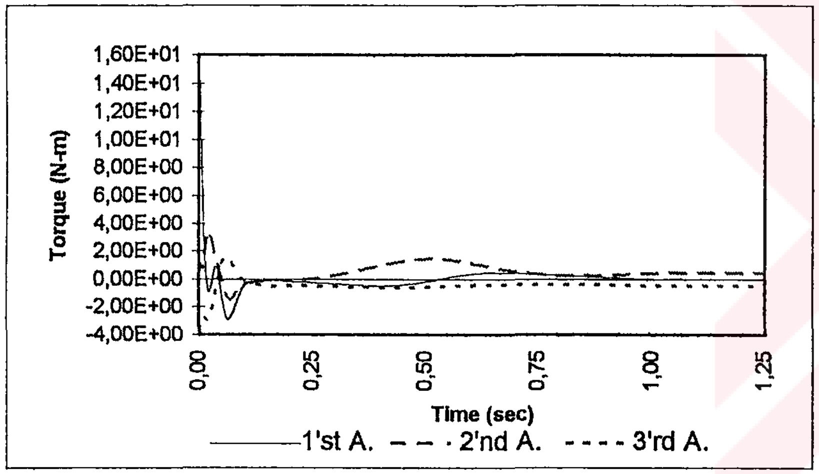
Figure 5.2.1.16 3R , Tr. 1 , $\alpha = 70$, $T = 1$ sec , with modeling error



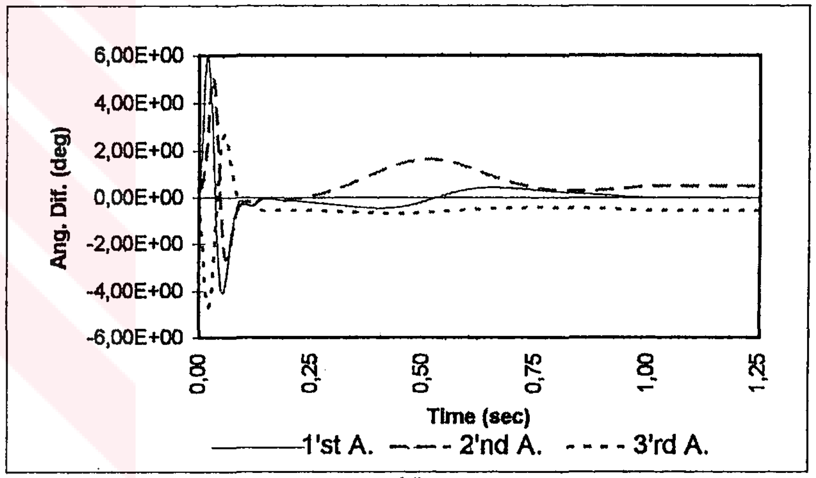
(a)



(b)



(c)



(d)

Figure 5.2.1.17 3R , Tr. 1 , $\alpha = 70$ T = 1 sec , with modeling error , with initial position error

5.2.2 Trajectory 2

Second commanded motion to the manipulator is a circular trajectory. The end-effector is desired to make a circular motion with radius R in yz-plane about center (x_0^d, y_0^d -R, z_0^d) starting from position (x_0^d, y_0^d, z_0^d).

$$x^d(t) = x_0^d$$

$$y^d(t) = y_0^d + R\,(\cos\beta - 1)$$

$$z^d(t) = z_0^d + R\sin\beta$$

$$\beta(t) = \pi\,(1 - \cos \pi\, t / T)$$

In the simulation two different set of control gains are used. Modeling error in mass properties and initial position error are also taken into consideration.

The input data used in the simulations are given as follows;

$m^L{}_1$	= 18 kg
$m^L{}_2$	= 9 kg
$m^L{}_3$	= 9 kg
$m^A{}_1$	= 3.9 10^{-2} kg
$m^A{}_2$	= 1.3 10^{-2} kg
$m^A{}_3$	= 1.3 10^{-2} kg
L_1	= 0.25 m

$L_2 = 1$ m

$L_3 = 1$ m

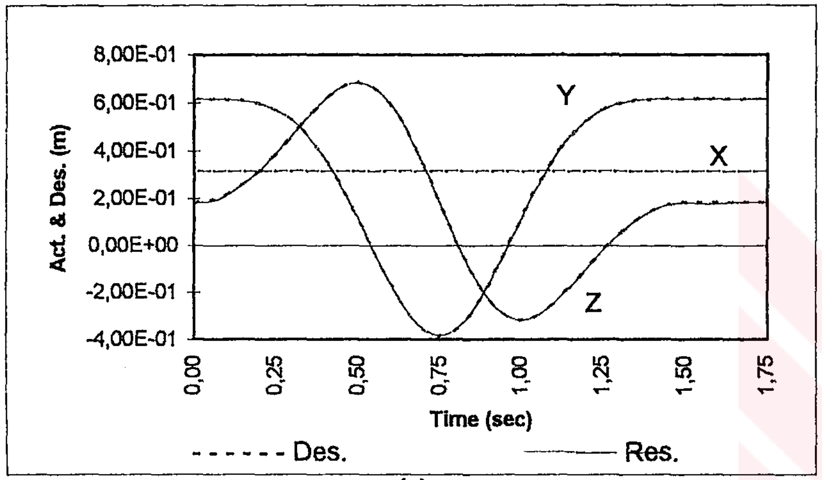
$K_1 = K_2 = K_3 = 5000$ N / m

$\alpha = 30$ & 50 rad / sec

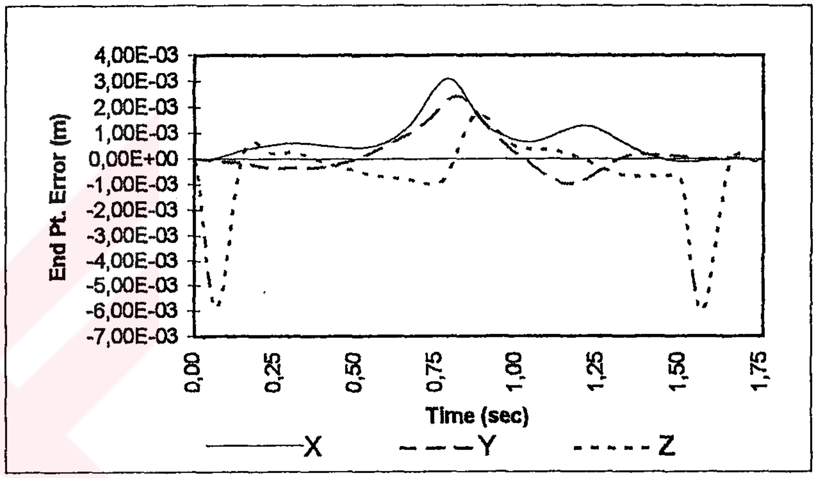
$T = 1.5$ sec

Simulation results can be summarized as follows;

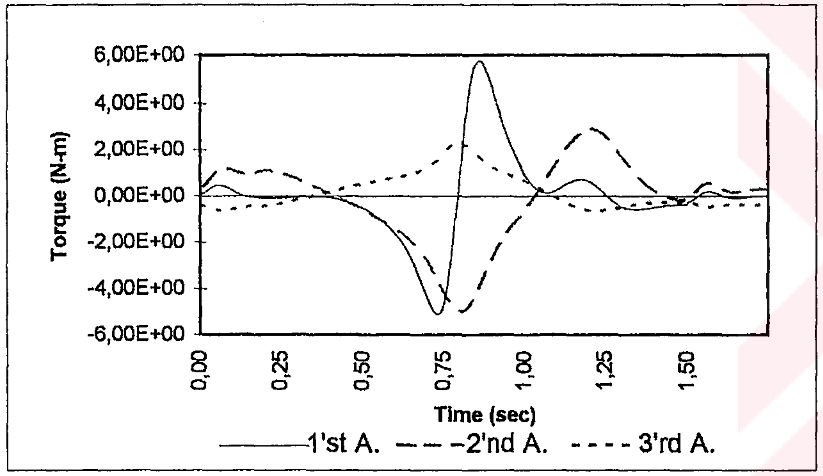
The response with $\alpha = 30$ is given in Fig. 5.2.2.1, when there is no initial position error and modeling error. In Fig. 5.2.2.2 initial positional errors in x and y directions are taken into account. Initial position error causes larger initial torques to be applied and larger initial angular differences happen. In Fig. 5.2.2.3, modeling error (10 % error in mass properties) is present. This error results larger tracking error and a small steady-state position error at the end-effector. In Fig. 5.2.2.4, both modeling and initial position error are present. Graph 5.2.2.4a shows that end-effector can not follow the desired trajectory properly. Again there happens larger initial torques, angular differences, tracking errors and a steady state position error at the end-effector. Figures from 5.2.2.5 to 5.2.2.8 show that, when $\alpha = 50$, end-effector can follow the desired trajectoy better. However this increase in α causes larger torques to be applied (in acceptable level) when there is initial position error or modeling error.



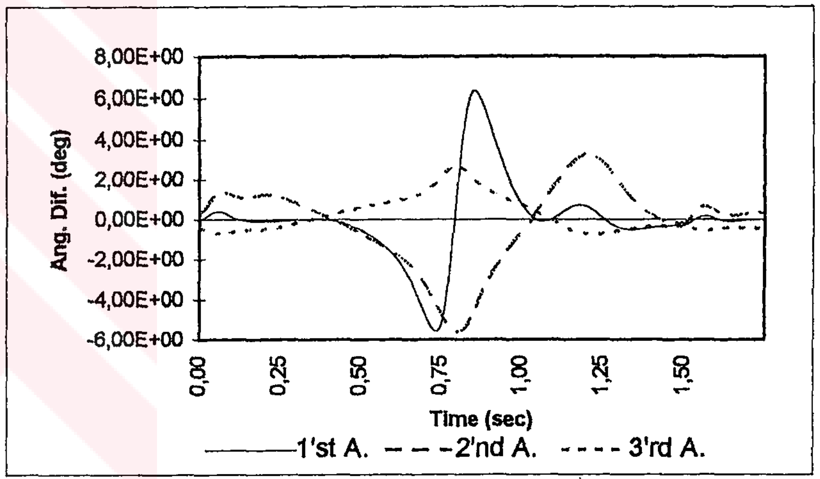
(a)



(b)



(c)



(d)

Figure 5.2.2.1 3R , Tr. 2 , $\alpha = 30$, T = 1.5 sec

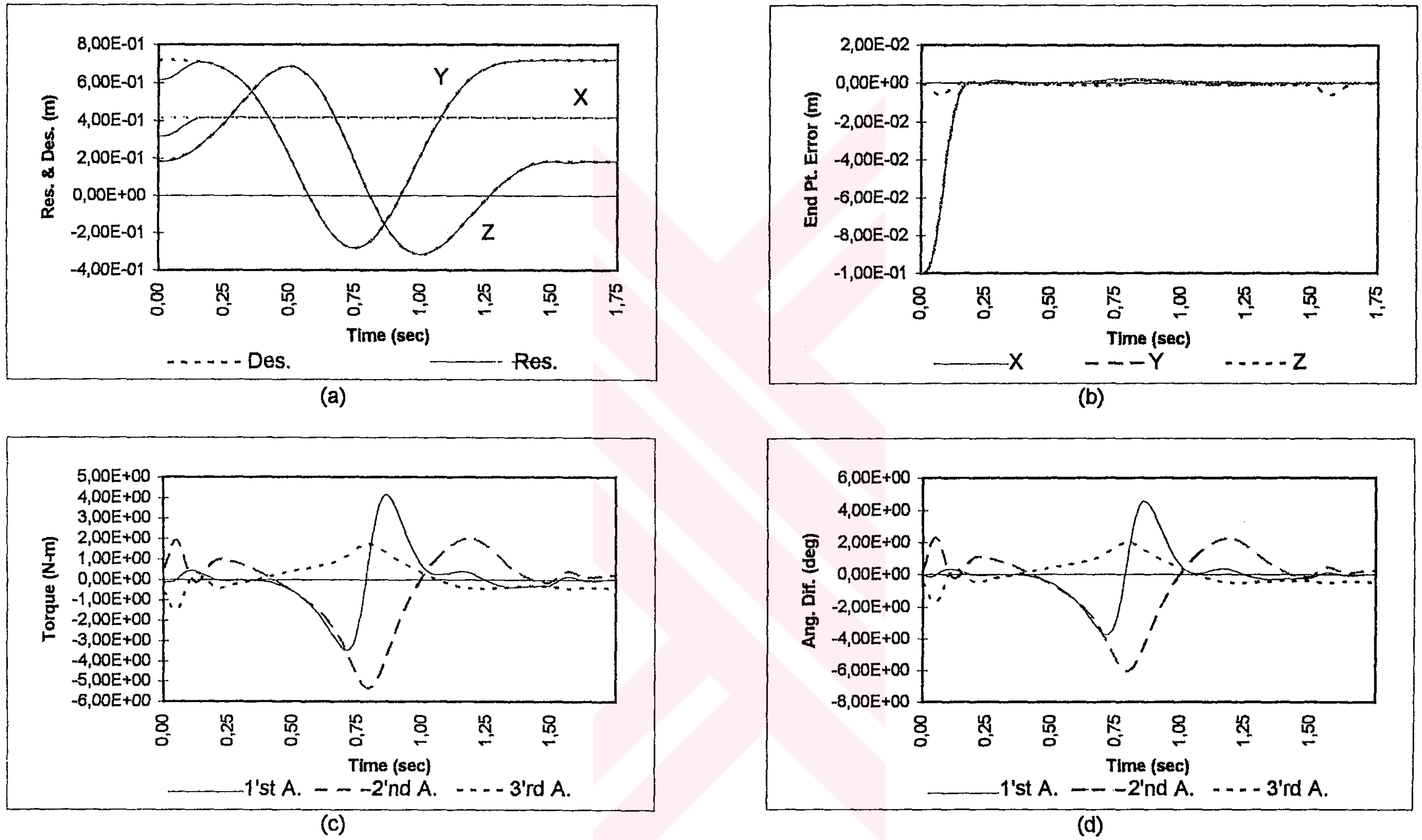
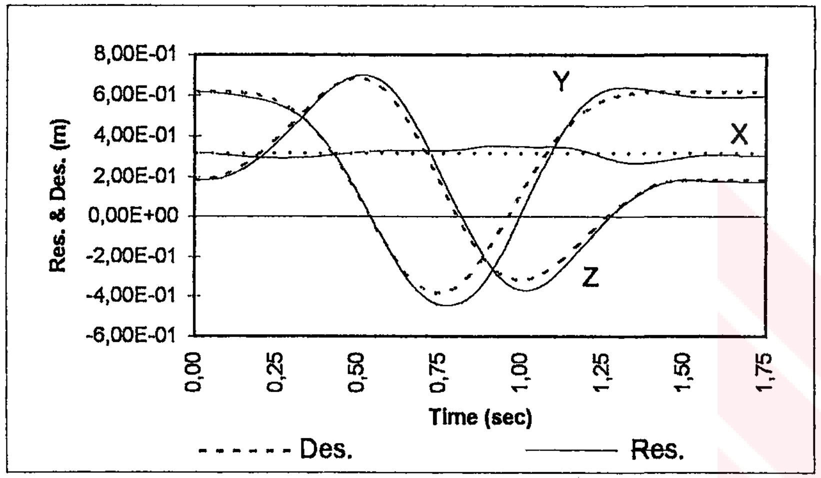
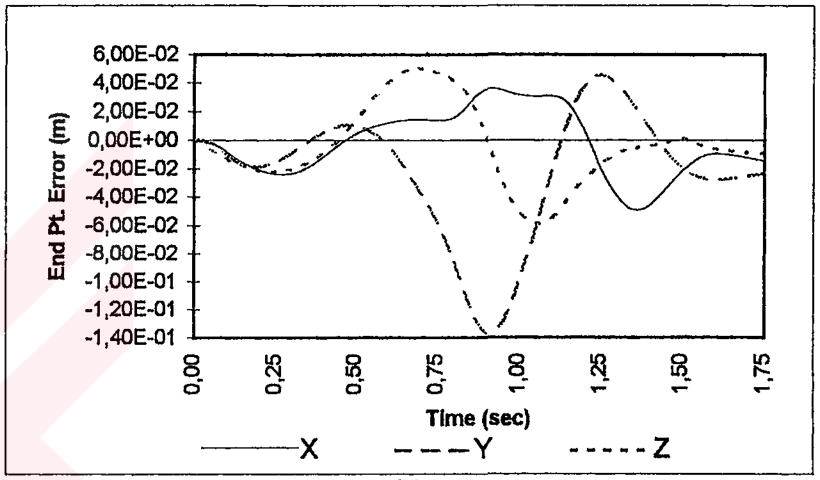


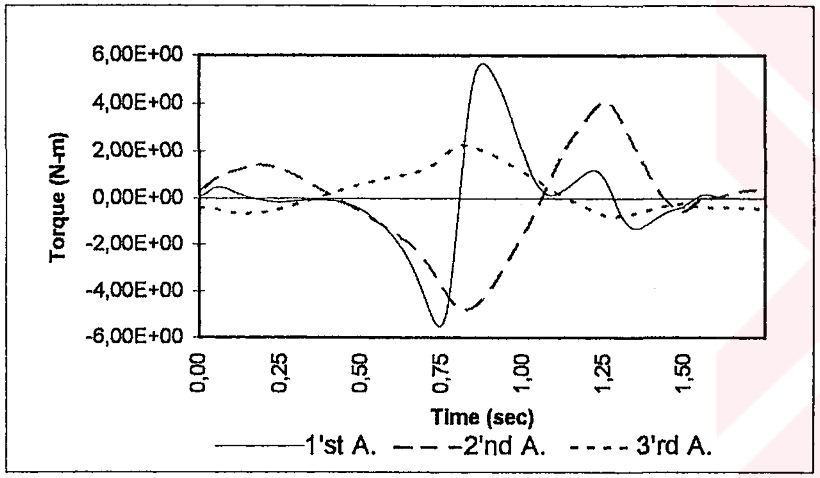
Figure 5.2.2.2 3R, Tr. 2, $\alpha = 30$, $T = 1.5$ sec, with initial position error



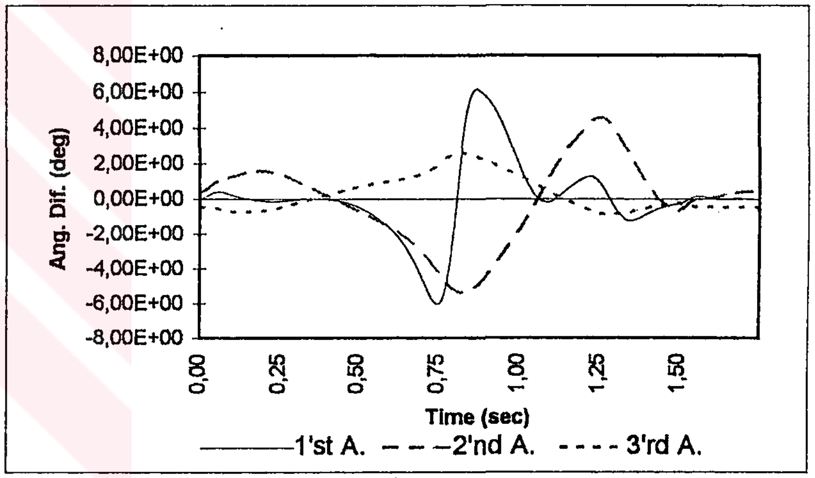
(a)



(b)

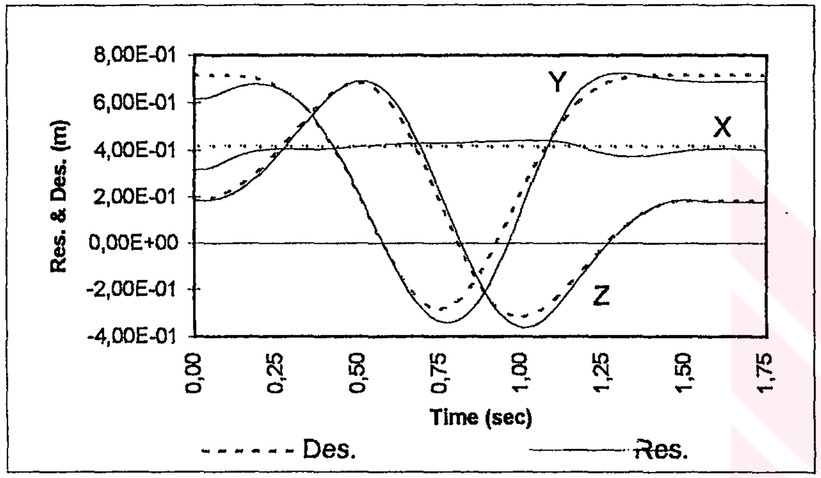


(c)

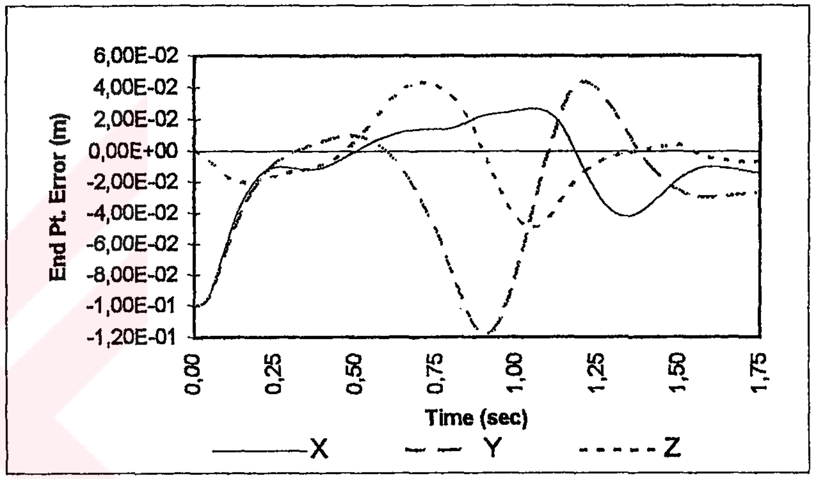


(d)

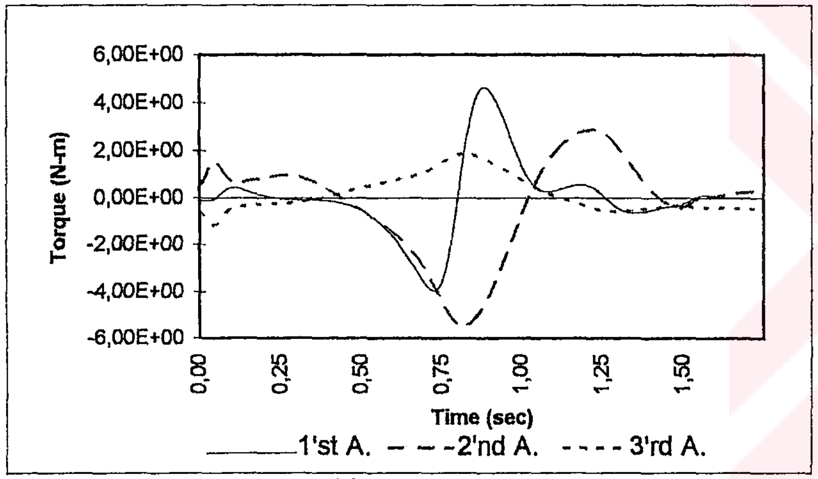
Figure 5.2.2.3 3R , Tr. 2 , $\alpha = 30$, $T = 1.5$ sec , with modeling error



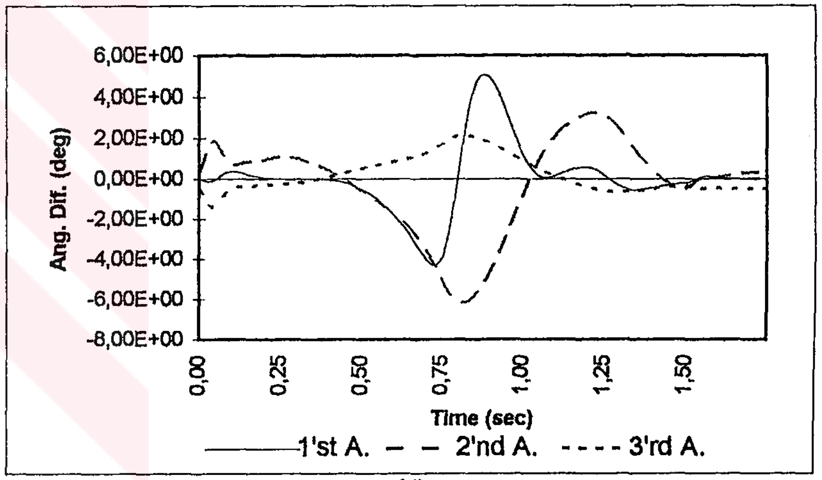
(a)



(b)



(c)



(d)

Figure 5.2.2.4 3R , Tr. 2 , $\alpha = 30$, T = 1.5 sec with modeling error , with initial position error

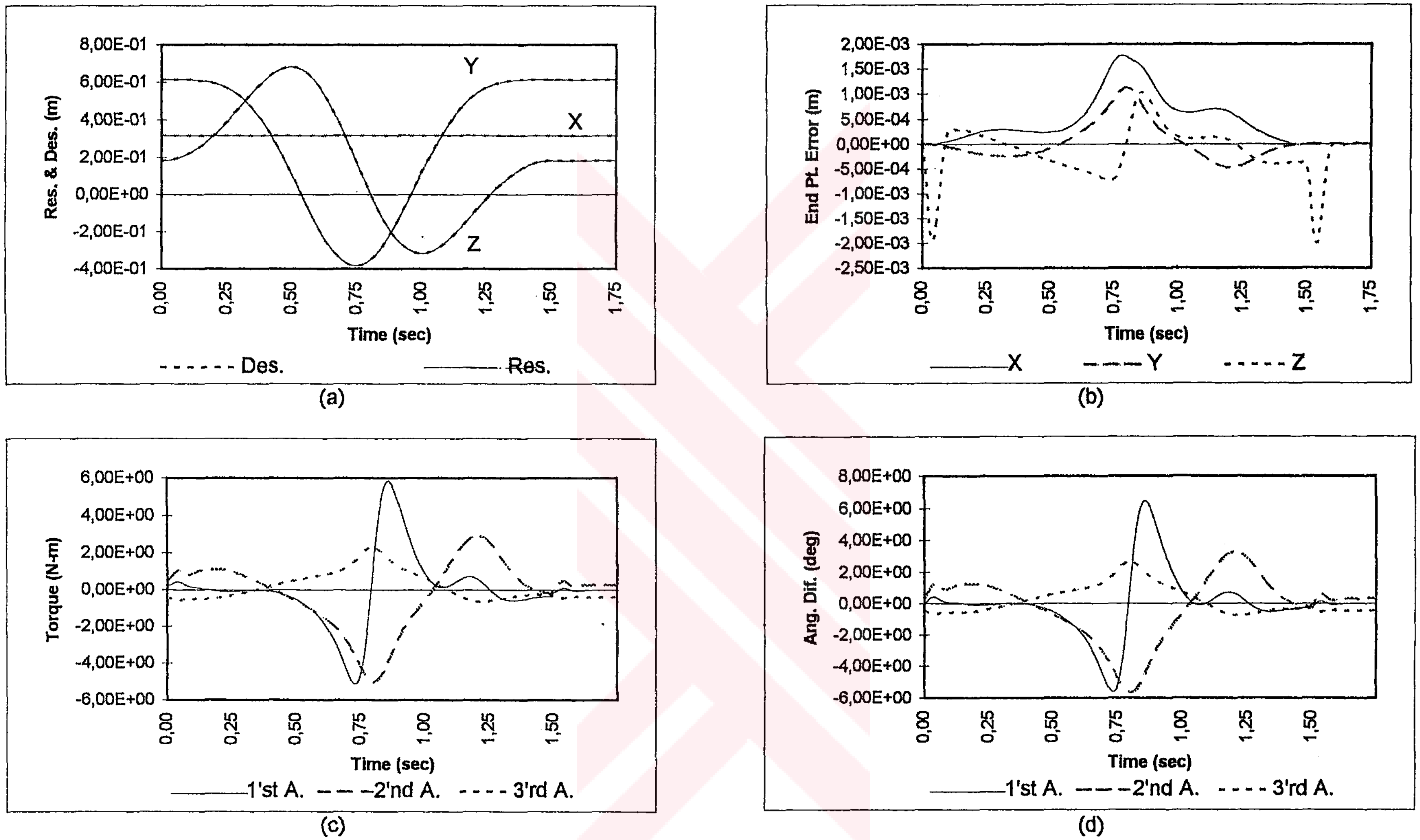
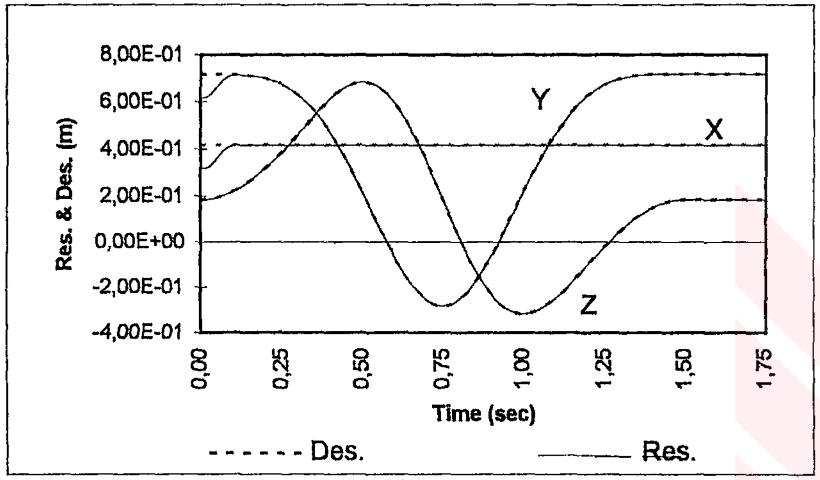
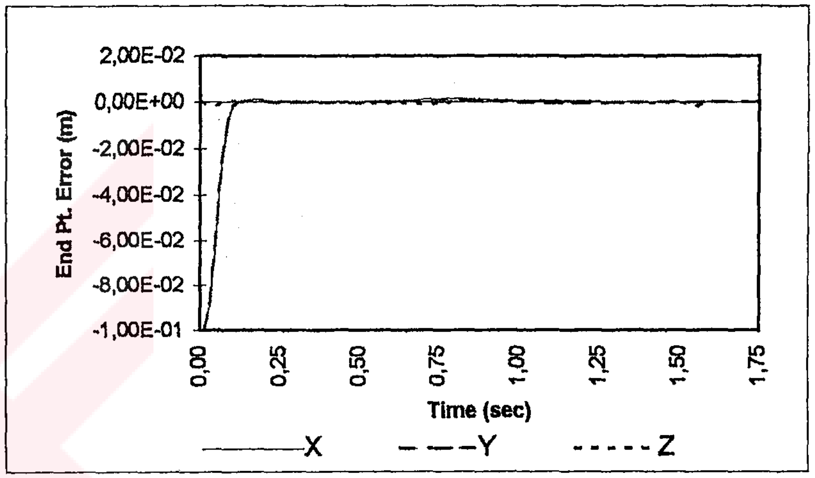


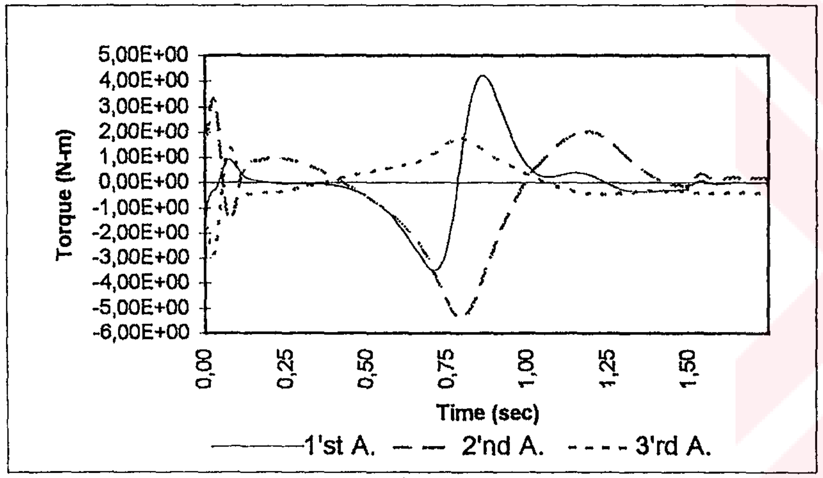
Figure 5.2.2.5 3R , Tr. 2 , $\alpha = 50$, T = 1.5 sec



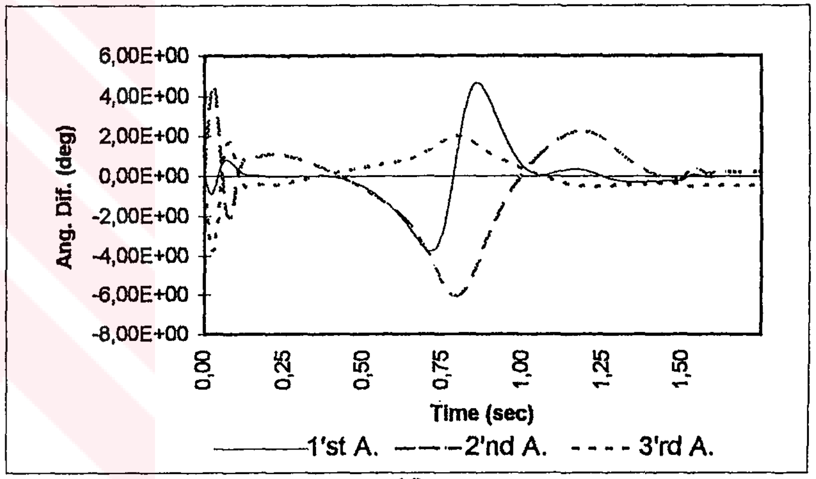
(a)



(b)

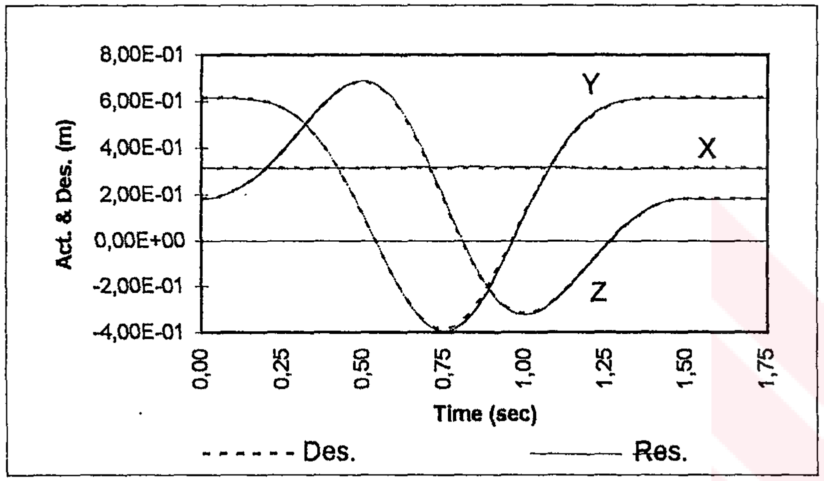


(c)

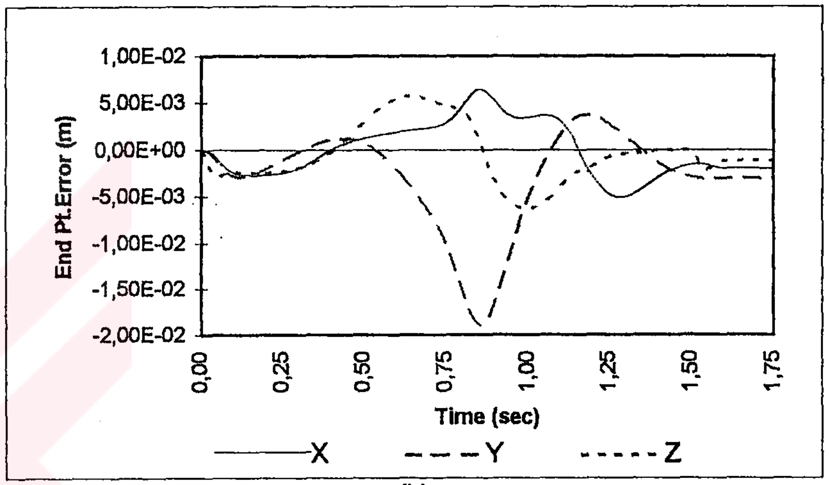


(d)

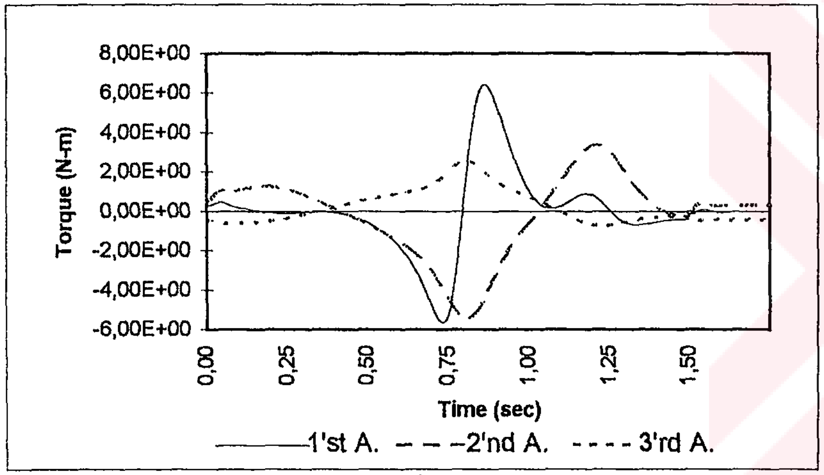
Figure 5.2.2.6 3R , Tr. 2 , $\alpha = 50$, T = 1.5 sec , with initial position error



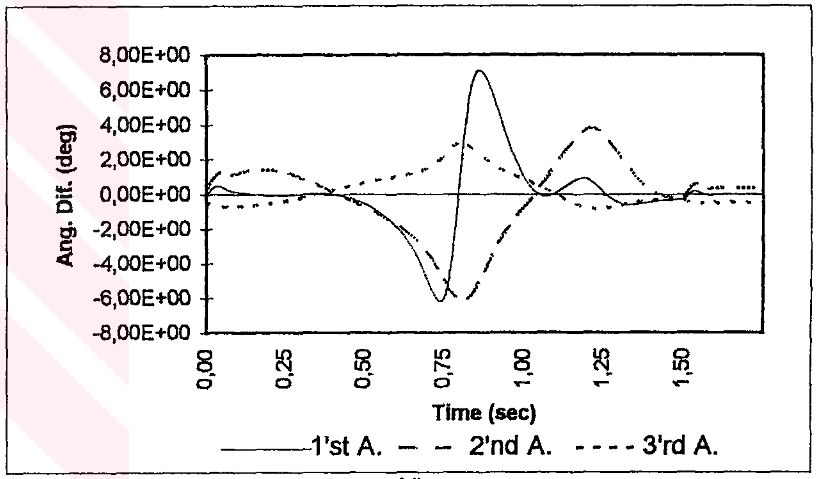
(a)



(b)

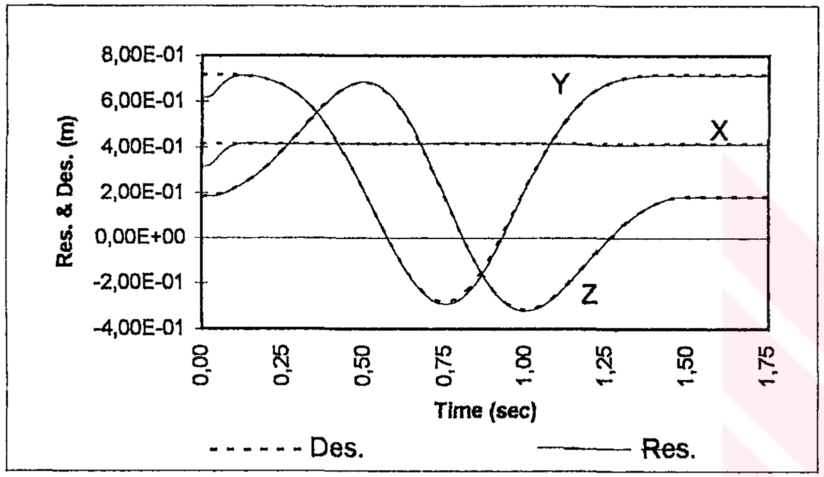


(c)

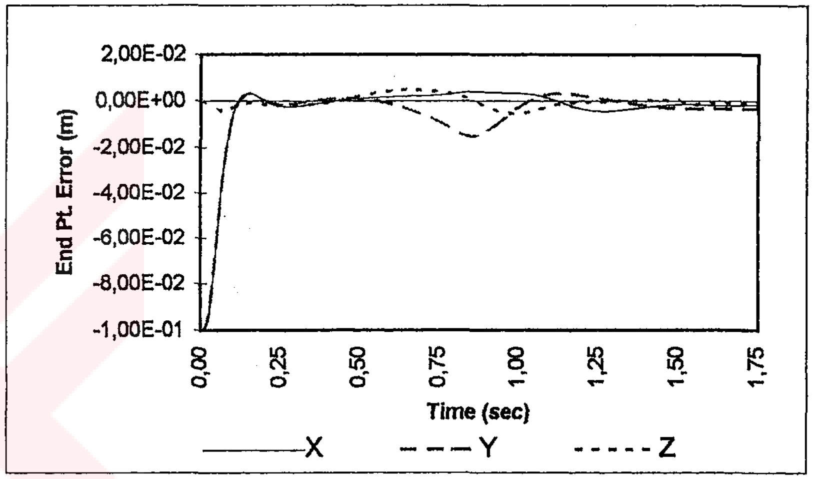


(d)

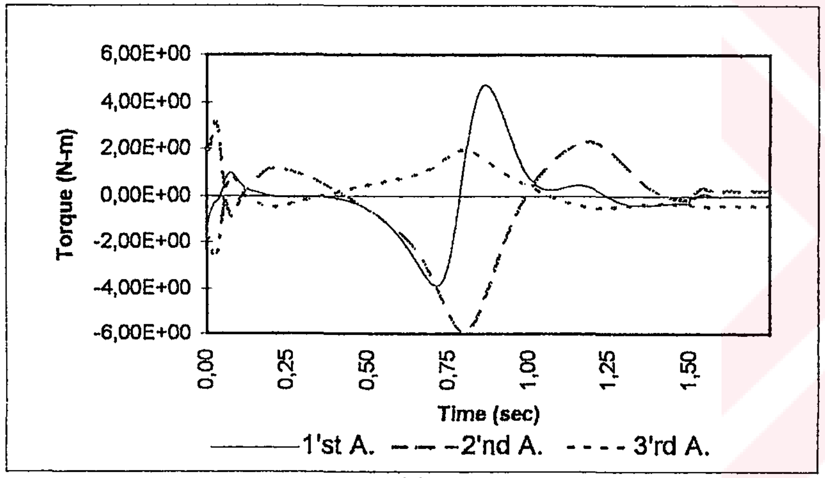
Figure 5.2.2.7 3R , Tr. 2 , $\alpha = 50$, $T = 1.5$ sec , with modeling error



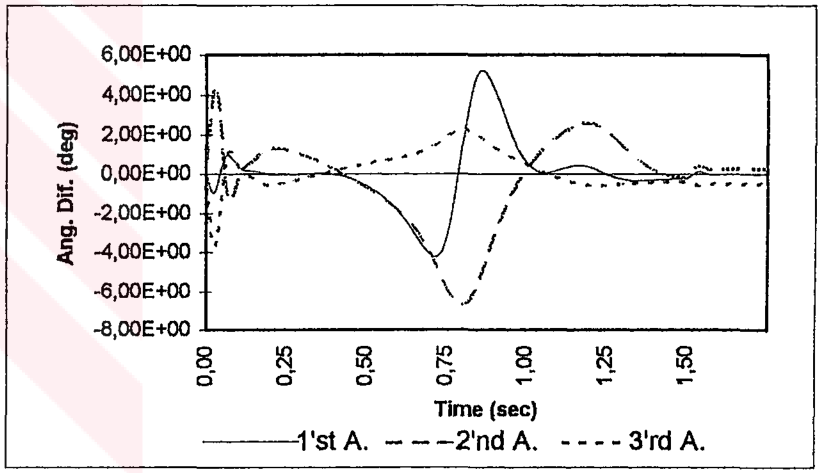
(a)



(b)



(c)



(d)

Figure 5.2.2.8 3R , Tr. 2 , $\alpha = 50$, T = 1.5 sec , with modeling error , with initial position error

Simulation results are presented in table form at tables 4.1, 4.2, 4.3 to enable the reader make comparisions and see the relations between the graphs.

Table 4.1 Simulation Results for 1-R

	Steady State Ang. Error	Time Initial Error Reduces to 2%	Max. Torque Applied By Act.
α=20			
no error	-	0,231	-5,39E-01
modeling error	1,60E-02	0,25	-3,15E-01
α=40			
no error	-	0,15	-1,94
modeling error	8,40E-03	0,156	-1,74
α=60			
no error	-	0,101	-3,56
modeling error	1,00E-03	0,106	-3,5

Table 4.2 Simulation Results for 3-R Tr.1

	Max. End Pt. Posn. Error in			Steady State Posn. Error in			Time Initial Error Reduces to 2%			Max. Torque Applied By Act.		
	x-dir	y-dir	z-dir	x-dir	y-dir	z-dir	x-dir	y-dir	z-dir	1	2	3
		(m)			(m)			(sec)			(N.m)	
α=50 , T=0.5 sec												
no error	1,00E-03	-5,45E-03	1,27E-03	-	-	-	-	-	-	-1,46E+00	3,42E+00	-9,94E-01
initial error	-	-	-	-	-	-	0,092	0,096	0,094	3,83E+00	3,68E+00	-2,01E+00
modeling error	-7,99E-03	-6,99E-03	2,48E-03	-9,90E-04	-6,03E-03	9,90E-04	-	-	-	-1,68E+00	3,97E+00	-1,11E+00
both errors	-	-	-	-5,50E-04	-4,40E-03	4,20E-04	0,096	0,1	0,098	3,83E+00	4,34E+00	-1,91E+00
α=70 , T=0.5 sec												
no error	-6,92E-04	-4,74E-04	-5,48E-04	-	-	-	-	-	-	-1,47E+00	3,44E+00	-9,96E-01
initial error	-	-	-	-	-	-	0,094	0,086	0,096	1,50E+01	3,69E+00	-3,68E+00
modeling error	1,65E-03	1,30E-03	3,51E-04	1,64E-04	1,14E-03	-1,64E-04	-	-	-	-1,63E+00	3,86E+00	-1,10E+00
both errors	-	-	-	1,38E-04	1,15E-03	9,21E-05	0,1	0,088	0,09	1,50E+01	4,14E+00	-3,68E+00
α=50 , T=1 sec												
no error	1,00E-05	-5,02E-03	6,73E-04	-	-	-	-	-	-	-3,80E-02	1,22E+00	-5,75E-01
initial error	-	-	-	-	-	-	0,088	0,092	0,092	3,92E+00	1,90E+00	-2,04E+00
modeling error	2,99E-03	-8,39E-03	-1,39E-03	-9,90E-04	-6,03E-03	9,90E-04	-	-	-	-4,27E-01	1,37E+00	-6,35E-01
both errors	-	-	-	-5,50E-04	-4,40E-03	4,20E-04	0,09	0,094	0,094	3,92E+00	1,80E+00	-1,93E+00
α=70 , T=1 sec												
no error	-1,70E-04	-1,17E-04	-1,36E-04	-	-	-	-	-	-	-3,81E-01	1,22E+00	-5,75E-01
initial error	-	-	-	-	-	-	0,09	0,092	0,09	1,51E+01	3,20E+00	-3,68E+00
modeling error	7,35E-04	1,10E-03	3,00E-04	1,64E-04	1,14E-03	-1,64E-04	-	-	-	-4,21E-01	1,35E+00	-6,33E-01
both errors	-	-	-	1,38E-04	1,15E-03	-9,22E-05	0,072	0,088	0,088	1,51E+01	3,42E+00	-3,68E+00

Table 4.3 Simulation Results for 3-R Tr.2

	Max. End Pt. Posn. Error in			Steady State Posn. Error in			Time Initial Error Reduces to 2%			Max. Torque Applied By Act.		
	x-dir	y-dir	z-dir	x-dir	y-dir	z-dir	x-dir	y-dir	z-dir	1	2	3
		(m)			(m)			(sec)			(N.m)	
α=30 , T=1.5 sec												
no error	-3,08E-03	-2,38E-03	-6,00E-03	-	-	-	-	-	-	5,77	-4,98	2,29
initial error	-	-	-	-	-	-	0,146	0,13	-	4,16	-5,33	1,74
modeling error	-4,92E-02	-1,37E-01	-5,88E-02	-1,35E-02	-2,37E-02	-8,83E-03	-	-	-	5,66	-4,8	2,26
both errors	-	-	-	-1,40E-02	-2,69E-02	-6,92E-03	0,502	0,308	-	4,64	-5,47	1,85
α=50 , T=1.5 sec												
no error	1,76E-03	1,12E-03	-1,97E-03	-	-	-	-	-	-	5,8	-5,02	2,32
initial error	-	-	-	-	-	-	0,098	0,1	-	4,29	-5,37	-2,74
modeling error	6,39E-03	-1,90E-02	-6,30E-03	-1,92E-03	-3,01E-03	-1,09E-03	-	-	-	6,42	-5,45	2,56
both errors	-	-	-	-1,85E-03	-3,50E-03	-7,83E-04	0,102	0,108	-	4,73	-6	-2,76

CHAPTER VI

CONCLUSIONS

This thesis presented an inverse dynamics control algorithm for trajectory tracking control of flexible joint robots. The control torques that yield asymptotic stability are calculated by using acceleration level dynamic equations. Since the acceleration level equations form a singular set of differential equations, the higher order derivative information are accounted in the numerical solution procedure. The proposed method requires feedback of position and velocities only, avoiding acceleration and jerk measurements. Also analytical differentiations of the equations of motion are avoided and the amount of on line calculations are greatly reduced.

Since the control forces can not make an instantaneous effect on the end-effector accelerations due to the elastic media, a jump in the end-effector command accelerations requires infinitely large control forces. The algorithm avoids jumps in command accelerations and the resulting force saturations, by using matching conditions at the time points where discontinuities of the desired motion occur. Since the command accelerations are determined by integrating the command jerk rates that decouple and linearize the system, the integration constants are selected freely such that the command acceleration and jerks are matched at the discontinuities.

The numerical simulations of 3-R Spatial manipulator show good tracking performances even if there are large modeling errors. The proposed method is also able to track the desired trajectories in the presence of initial and intermediate jumps in the desired positions, velocities, accelerations and jerks.

Suggestions for Future Work :

1-) Implementation to real robotic systems

2-) Extension of the algorithm for position control to hybrid position / force control.

LIST OF REFERENCES

1. Chow, Joe H. and Kokotovic, Petar V., 1978. "Two-Time-Scale Feedback Design of a Class of Nonlinear Systems", IEEE Transactions on Automatic Control , Vol. AC-23, No.3, pp.438-443.

2. De Luca, A., 1988. "Control Properties of Robot Arms with Joint Elasticity", in Analysis and Control of Nonlinear Systems, Byrnes,C. I., Martin, C. F. and Saeks, R. E., Editors, pp.61-70.

3. D`Souza, A. Frank, 1988. "Design of Control Systems", Prentice-Hall International Editions, USA.

4. Forrest, M. G. and Babcock, S. M., 1987. "Inverse Dynamics Position Control of a Compliant Manipulator", IEEE Journal of Robotics and Automation , Vol.RA-3, No.1, pp.75-83.

5. Gear, C. W. and Petzold. R., 1984. " ODE Methods for the Solution of Differential/Algebraic Systems", SIAM Journal of Numerical Analysis , Vol.21, No.4, pp.716-728.

6. Gogate, Sachın and Lin, Yueh-Jaw, 1993. "Formulation and Control of Robots With Link and Joint Flexibility", Robotica , Vol.11, pp.273-282.

7. Jankowski, Krzysztof P. and Van Brussel Hendrik, 1992. "An Approach to Discrete Inverse Dynamics Control of Flexible-Joint Robots", IEEE Transactions on Robotics and Automation , Vol.8, No.5, pp.651-658.

8. Jankowski, Krzysztof P. and ElMaraghy, Hoda A., 1994. "Inverse Dynamics and Feedforward Controllers for High Precision Position/Force Tracking Control of Flexible Joint Robots", Robotica , Vol.12, pp.227-241.

9. İder, S. K., "Lecture Notes on Flexible Multibody System Dynamics", Unpublished, METU, Ankara.

10. Ogata, Katsuhiko, 1990. "Modern Control Engineering", Prentice-Hall International Editions, USA.

11. Özgören, M.K., "Lecture Notes on Robotics", Unpublished, METU, Ankara.

12. Rivin, E. I., 1985. "Effective Rigidity of Robot Structure", Proceedings of 1985 American Control Conference , pp.381-382.

13. Spong, M. W., 1987. "Modeling and Control of Elastic Joint Robots", Journal of Dynamic Systems, Measurement, and Control , Vol.109, pp.310-319.

14. Spong, M. W. and Vidyasagar, M., 1989. "Robot Dynamics and Control", John Wiley & Sons Inc., USA.

APPENDIX

SIMULATION PROGRAM

The simulation program used in the thesis for the simulation of the motion of flexible joint robot arms is given below:

```
C ************************************************************************
C              MAIN PROGRAM
C*** THIS PROGRAM GENERATES THE EQUATIONS OF MOTION OF MULTIBODY
SYSTEMS
C WITH OR WITHOUT JOINT STIFFNESSES, WITH OR WITHOUT CLOSED LOOPS
C BY CALLING SUBROUTINES INPUT, UPDATE, SHIFT, PANGV, OMEGA,SHIFTD,
C PANGVD, PVEL, PVELD, MASSQ AND FORCE.
C*** CONSTRAINT EQUATIONS FOR DESIRED MOTIONS AND CLOSED LOOPS ARE
C GENERATED BY SUBROUTINES CONSTR AND CLOOP.
C*** CONTROL FORCES ARE FOUND USING BACWARD DIFFERENCE CONTROL LAW
C USING DESIRED ACCELERATIONS CALCULATED BY SUBROUTINES CONG AND
ABSPOS.
C*** FORWARD DYNAMICS IS SOLVED USING PREDICTOR CORRECTOR METHOD
BY
C SUBROUTINES DAMSDE AND F.
C ************************************************************************
C
   IMPLICIT REAL*8 (A-H,O-Z)
   LOGICAL STAR
   DIMENSION Y(60),YP(60)
C    THE FIRST NTOT OF Y ARE GEN. SPEEDS, LAST NTOT GEN. COORDINATES
   COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
   COMMON/INP/ N,IRF(14)
   COMMON/PRIN0/ IPR0(40),NPRI
   COMMON/PRIN/ IPR(40)
   COMMON/FL/ IFL(40)
   COMMON/CNST3/ NCST,NMNC
   COMMON/CNST4/ NLOOP,NCADD
   COMMON/CNST5/ LOOP(5,2),REND(5,2,3),NDIR(5,3),NQCON(5,2)
   COMMON/FOR1/ FMJOIN(30)
   COMMON/CNSTA/ NDIRA(4,3),NNANG(4),NANGV
   COMMON/STIF/ GROSK(30,30)
   COMMON/SD2/ GROSD(30,30)
   COMMON/PDISP/ KPOIN,KNPOIN,PPOIN(3)
   COMMON/GF1/ GENFI(3,30),GENFE(3,30),GENFC(30),RHS(30)
   COMMON/DIFF1/ TDEL
   COMMON/NEWA/ AA(15,30)
   CHARACTER*4 TIT(22), TEXT
   COMMON/MAT2/ R(14,3),RR(14,3,3)
   COMMON/MAT3/ RMASS(14)
   DIMENSION RN(14,3),RRN(14,3,3),RMASSN(14),YBEF(60)
```

```
COMMON/CNST1/CE(15,30)
COMMON/CNST7/BDOT(15,30),AAIBD(30,30)
   COMMON/MASS1/GM(30,30)
   COMMON/FOR2/EFOR(30)
   COMMON/DIFF2/YBEFN(60)
   DIMENSION YSTART(60),YPRE(60)
   COMMON/CONG1/GDOT(15)
   COMMON/CONG2/APB(3),APDB(3)
   DIMENSION B(30,15),BBT(30,60),BBTIB(15,30),ANIU(15),BTL(30)
   COMMON/GEAR/IGEAR,NGEAR,MIND(10,2),FACT(10)
C
   DATA TIT/'CONS','TIME','JOFM','DISP','INVE','INPO',
  1  'COMM','CNEW','STIF','DAMP','EJFM','EULR',
  2  'BDIF','FEED','LUMP','DELC',/
   DATA Y/60*0./,YP/60*0./
   DATA REL,ABST,IFLAG/2*0.000001,1/
   OPEN(5,FILE='FLEX.INP',STATUS='OLD')
   OPEN(6,FILE='FLEX.OUT',STATUS='UNKNOWN')
   OPEN(11,FILE='FLEX.ANG',STATUS='UNKNOWN')
   OPEN(12,FILE='FLEX.VEL',STATUS='UNKNOWN')
   OPEN(14,FILE='FLEX.ABS',STATUS='UNKNOWN')
   OPEN(17,FILE='FLEX.DSR',STATUS='UNKNOWN')
   OPEN(15,FILE='FLEX.ACC',STATUS='UNKNOWN')
   OPEN(16,FILE='FLEX.DIF',STATUS='UNKNOWN')
C
   PRINT*,'PLEASE WAIT, PROGRAM IS RUNNING'
   DO 8 I=1,40
   IPR0(I)=0
  8 IFL(I)=0
   NPRI=0
C
   CALL INPUT
C
C  INITIALIZATIONS
   NTIT=17
   TSTART=0.
   DO 503 I=1,NTOT*2
 503 YSTART(I)=0.
   IGEAR=0
   IEUL=0
   NLOOP=0
   NCADD=0
   NTC=0
   NCST=0
   KPOIN=0
   NTOT=NBC
   NMNC=NTOT
   ND=NTOT*2
   STAR=.FALSE.
   IPCOUN=0
   DO 9 I=1,40
  9 IPR(I)=IPR0(I)
   DO 30 I=1,NTOT
 30 FMJOIN(I)=0.
   DO 842 I=1,30
 842 BTL(I)=0.
   DO 820 I=1,N
   RMASSN(I)=RMASS(I)
   DO 820 J1=1,3
```

```
      RN(I,J1)=R(I,J1)
      DO 820 J2=1,3
  820 RRN(I,J1,J2)=RR(I,J1,J2)
C
      CALL PERMUT
      CALL TRE
C
C  READ THE EXECUTION CARDS
   10 READ(5,11,END=70) TEXT
      WRITE(6,115) TEXT
  115 FORMAT(2X,A4)
   11 FORMAT(A4)
      DO 12 I=1,NTIT
      IF(TEXT.EQ.TIT(I)) GO TO 13
   12 CONTINUE
      WRITE(6,100)
  100 FORMAT(///' KEYWORD ERROR IN CARD')
   70 STOP
   13 II=I
      GO TO (60,61,64,65,66,140,10,
     1   212,301,302,303,307,315,502,570,580),II
C
C  CONSTRAINT EQUATIONS
C  NTC=0 IF CONSTR. EQNS. DO NOT HAVE EXPLICIT TIME DEPENDENT TERMS.
C   OTHERWISE 1. (NTC=1 ALSO WHEN THERE IS MODIFIED CONSTRAINT.)
C  NCST=# OF CONSTR. EQNS.. FIRST ONES ARE DUE TO LOOPS.
C  THE REST NCADD NUMBER OF CONSTR. EQNS. WILL BE EXPLICITILY WRITTEN
C   INTO B AND BDOT IN S. CONSTR (AND INTO TCD IN S. F).
C  NDIR SHOWS WHICH DIRECTIONS IN R THE POINT IS CONSTRAINED.
C   FREE DIRECTIONS ARE 0.
   60 IFL(3)=1
      READ(5,*) NCST,NTC,NLOOP,NANGV,NCADD
      WRITE(6,*) NCST,NTC,NLOOP,NANGV,NCADD
      NMNC=NTOT-NCST
      IF(NTC.GT.0) IFL(3)=2
      IF(NLOOP.EQ.0) GO TO 14
      DO 15 I=1,NLOOP
      READ(5,*) (NDIR(I,J),J=1,3)
      WRITE(6,*) (NDIR(I,J),J=1,3)
      DO 15 IND=1,2
      READ(5,*) LOOP(I,IND)
      WRITE(6,*) LOOP(I,IND)
      K=LOOP(I,IND)
      IF(K.EQ.0) GO TO 15
      READ(5,*) (REND(I,IND,J),J=1,3)
   15 CONTINUE
   14 IF(NANGV.EQ.0) GO TO 17
      DO 18 I=1,NANGV
      READ(5,*) (NDIRA(I,J),J=1,3),NNANG(I)
   18 WRITE(6,*) (NDIRA(I,J),J=1,3),NNANG(I)
   17 GO TO 10
C
C
C EULER METHOD FOR INTEGRATION
  307 IEUL=1
      GO TO 10
C
C BACKWARD EULER METHOD FOR FLEXIBLE CONTROL
  310 IFL(30)=1
```

```
      READ(5,*)EPSL,MITER
      WRITE(6,*)EPSL,MITER
      GO TO 10
C
C  APPLIED JOINT MOMENTS AND FORCES
   64 IFL(10)=1
      GO TO 10
C
C  TIME DEPENDENT EXTERNAL JOINT FORCES, MOMENTS SPECIFIED IN SUBR. EJF
  303 IFL(5)=1
      GO TO 10
C
C  INITIAL VELOCITIES
   66 READ(5,*) (Y(I),I=1,NTOT)
      WRITE(6,*) (Y(I),I=1,NTOT)
      GO TO 10
C
C  INITIAL JOINT COORDINATES. SHOULD BE USED FOR INITIAL DEFLECTION
C  OF JOINT SPRINGS
   67 READ(5,*) (Y(I+NTOT),I=1,NBC1)
      READ(5,*) (YSTART(I+NTOT),I=1,NBC1)
      WRITE(6,*) (Y(I+NTOT),I=1,NBC1)
      WRITE(6,*) (YSTART(I+NTOT),I=1,NBC1)
      DO 68 I=1,NBC1
      Y(I+NTOT)=Y(I+NTOT)*3.1415926/180.
   68 YSTART(I+NTOT)=YSTART(I+NTOT)*3.1415926/180.
      GO TO 10
C
C  ABSOLUTE DISPLACEMENTS OF A POINT REQUIRED
   65 IFL(6)=1
      READ(5,*) KPOIN
      IF(IRF(KPOIN).EQ.0) READ(5,*)(PPOIN(J),J=1,3)
      IF(IRF(KPOIN).GT.0) READ(5,*)KNPOIN
      GO TO 10
C
C  INPUTTING STIFFNESS COEFFICIENTS CORR. TO EACH R. B. COORDINATES
  301 READ(5,*) ((GROSK(I,J),J=1,NBC),I=1,NBC)
      WRITE(6,721)
  721 FORMAT(/,'STIFFNESS MATRIX')
      DO 722  I=1,NBC
  722 WRITE(6,723) (GROSK(I,J),J=1,NBC)
  723 FORMAT(1X,8E13.5)
      IFL(13)=1
      GO TO 10
C
C  INPUTTING DAMPING COEFFICIENTS CORR. TO EACH R. B. COORDINATES
  302 READ(5,*) (GROSD(I,I),I=1,NBC)
      IFL(13)=1
      GO TO 10
C
C CONTROL FORCES, ARBITRARY DIRECTION
  210 IFL(12)=1
      DO 211 I=1,15
      DO 211 J=1,30
  211 AA(I,J)=0.
      CALL CONA
      GO TO 10
C
C  GEAR RATIO=G:1
```

```
C  INDICES SHOULD CORRESPOND TO ROTOR DOF'S
C  IF INDICES ARE EQUAL, FACT SHOULD BE G*G
C  IF INDICES ARE NOT EQUAL, FACT SHOULD BE ZERO OR G
 580 IGEAR=1
    READ(5,*) NGEAR
    DO 581 I=1,NGEAR
    READ(5,*) MIND(I,1),MIND(I,2),FACT(I)
 581 WRITE(6,*) MIND(I,1),MIND(I,2),FACT(I)
    GO TO 10
C
C  FEEDBACK CONTROL
C  FEED SHOULD BE USED TOGETHER WITH BDIF AND CNEW (AND POSSIBLY
C  WITH HYBD).
C  IT IS ASSUMED THAT NCST=# OF CONTROL FORCES, I.E. NO CLOSED CHAIN
C  CONSTRAINTS.
 502 IFL(32)=1
    READ(5,*) TDEL1,INTM,PERC
    TDELTA=TDEL
    GO TO 10
C  INTM=1 FOR BDIF, INTM=2 FOR DAMSDE
C  IF BDIF IS USED: SET TDELTA=TDEL, TDEL1 IS NOT USED
C  IF DAMSDE IS USED: SET TDELTA=TDEL, AND TDEL1 IS THE
C     TIME STEP FOR THE SIMULATION OF THE REAL SYSTEM USING THE
C     CONTROL FORCES. IT SHOULD BE A FRACTION OF TDELTA.
C
C  STARTING SIMULATION
  61 READ(5,*) T,TFIN,TDEL
    WRITE(6,101) T,TFIN,TDEL
 101 FORMAT(//' INITIAL TIME, FINAL TIME, INCREMENT:',3E12.4)
    TOUT=TDEL+T
C
  22 CONTINUE
    PRINT*,T
    DO 220 I=1,NTOT*2
    YBEFN(I)=Y(I)
 220 YBEF(I)=Y(I)
    ITERN=0
    IF(IFL(30).EQ.1) T=T+TDEL
  23 CONTINUE
C
    CALL UPDATE(Y,T)
C
    CALL SHIFT
    IF(IFL(6).EQ.1.AND.ITERN.EQ.0) CALL ABSPOS(Y,T)
    CALL PR01(STAR,IPCOUN)
    CALL PANGV
    CALL OMEGA(Y)
    CALL SHIFTD
    CALL PANGVD
    CALL PVEL
    CALL PVELD
    IF(NCST.GE.1) CALL CONSTR(Y,YP,T)
C BACKWARD EULER
    IF(IFL(30).EQ.0) GO TO 312
    IF(ITERN.GE.1) GO TO 312
    DO 318 I=1,NCST
 318 GDOT(I)=0.
    CALL CONG(Y,YP,T)
 312 CONTINUE
```

```
C END BACKWARD EULER
C PRINTING GENERALIZED FORCES
      IF(IPR(3).EQ.0) GO TO 114
      DO 110 I=1,NTOT
  110 GENFC(I)=GENFI(1,I)-GENFE(1,I)
      IF(IFL(10).LT.2) GO TO 117
      WRITE(6,116) (GENFC(I),I=1,NTOT)
  116 FORMAT(1X,6E14.6)
      GO TO 114
  117 WRITE(6,113)
  113 FORMAT(/' FC, FL*, MA, Q, FL, S, EXT')
      DO 112 I=1,NTOT
  112 WRITE(6,111) GENFC(I),(GENFI(J,I),J=2,3),(GENFE(J,I),J=2,3)
  111 FORMAT(1X,7E12.4)
  867 FORMAT(1X,F9.5,8E13.5)
  114 CONTINUE
C END PRINTING GENERALIZED FORCES
C BACKWARD EULER
      IF(IFL(30).EQ.0) GO TO 311
      CALL F(T,Y,YP)
      DO 221 I=1,NTOT
  221 Y(I)=YP(I)
      DO 222 I=1,NTOT
      YP(I)=(Y(I)-YBEF(I))/TDEL
  222 Y(I+NTOT)=YBEF(I+NTOT)+Y(I)*TDEL
      SUM=0.
      DO 223 I=1,NTOT
  223 SUM=SUM+(Y(I)-YP(I+NTOT))**2.
      DO 224 I=1,NTOT
  224 YP(I+NTOT)=Y(I)
      IF(SUM.LE.EPSL.OR.ITERN.GE.MITER) GO TO 225
      ITERN=ITERN+1
      GO TO 23
  225 CONTINUE
      IF(IPR(1).GT.0) WRITE(6,226)ITERN,SUM
  226 FORMAT(/'ITERATION=',I3,3X,'SSQ=',E13.4)
      DO 870 I=1,NTOT*2
  870 YPRE(I)=Y(I)
      DO 323 I=1,NTOT
      SUM=0.
      DO 324 J=1,NTOT
  324 SUM=SUM+GM(I,J)*Y(J)
  323 RHS(I)=RHS(I)-SUM+GENFE(1,I)
C     IF(IPR(1).GT.0) CALL OUTP(T,Y,YP,0)
      IF(T.GT.TFIN) STOP
      GO TO 330
  311 CONTINUE
C END BACKWARD EULER
C
      TOUT=T+TDEL
   21 CALL DAMSDE(ND,Y,T,TOUT,REL,ABST,IFLAG,YP)
      IF(IFLAG.EQ.3) GO TO 21
      IF(IPR(1).GT.0) CALL OUTP(T,Y,YP,IFLAG)
C     IF(IFL(6).EQ.1) CALL ABSPOS(Y,T)
      IF(IFLAG.LT.4) GO TO 24
      CALL OUTP(T,Y,YP,IFLAG)
      STOP
   24 CONTINUE
C
```

```
      IF(T.GE.TFIN) STOP
      DO 325 I=1,NTOT
      SUM=0.
      DO 326 J=1,NTOT
  326 SUM=SUM+GM(I,J)*YP(I)
  325 RHS(I)=-SUM+GENFE(1,I)-GENFI(3,I)
  330 CONTINUE
C
      IF (NCST.EQ.0) GO TO 349
      DO 342 I=1,NCST
      DO 342 J=1,NTOT
  342 B(I,J)=CE(I,J)
      IF(IFL(12).EQ.0) GO TO 339
      DO 338 I=1,NCST
      DO 338 J=1,NTOT
  338 B(I,J)=AA(I,J)
  339 CONTINUE
      DO 340 I=1,NCST
      DO 340 J=1,NCST
      SUM=0.
      DO 341 L=1,NTOT
  341 SUM=SUM+B(I,L)*B(J,L)
  340 BBT(I,J)=SUM
      DO 343 I=1,NCST
      DO 344 J=1,NCST
  344 BBT(I,J+NCST)=0.
  343 BBT(I,I+NCST)=1.
      NP=2*NCST
      CALL ELIM(BBT,NCST,NP)
      DO 345 I=1,NCST
      DO 345 J=1,NTOT
      SUM=0.
      DO 346 L=1,NCST
  346 SUM=SUM+BBT(I,L+NCST)*B(L,J)
  345 BBTIB(I,J)=SUM
      DO 347 I=1,NCST
      SUM=0.
      DO 348 L=1,NTOT
  348 SUM=SUM+BBTIB(I,L)*RHS(L)
  347 ANIU(I)=SUM
      WRITE(10,867) T,(ANIU(I)/100.,I=1,NCST)
  349 CONTINUE
      STAR=.TRUE.
      IF(IFL(32).EQ.0) GO TO 22
C
      IFLAG=1
      DO 504 I=1,NTOT*2
      Y(I)=YSTART(I)
  504 YBEFN(I)=YSTART(I)
      T=TSTART
      IFL(12)=0
      IFL(30)=0
C
      IF (INTM.NE.1) GO TO 800
      IFL(30)=1
      ITERN=0
      T=T+TDEL
  800 CONTINUE
C
```

```
      NCSTST=NCST
  520 CALL UPDATE(Y,T)
      CALL SHIFT
C     IF (IFL(6).EQ.1) CALL ABSPOS(Y,T)
      CALL PANGV
      CALL OMEGA(Y)
      CALL SHIFTD
      CALL PANGVD
      CALL PVEL
      CALL PVELD
      IFL(10)=1
      DO 521 I=1,NTOT
      SUM=0.
      DO 522 J=1,NCSTST
  522 SUM=SUM+AA(J,I)*ANIU(J)
  521 FMJOIN(I)=-SUM
C
      DO 821 I=1,N
      RMASS(I)=RMASSN(I)*PERC
      DO 821 J1=1,3
      R(I,J1)=RN(I,J1)*PERC
      DO 821 J2=1,3
  821 RR(I,J1,J2)=RRN(I,J1,J2)*PERC
C
      CALL MASSQ(Y,YP)
      CALL FORC(T)
      WRITE(6,888)(EFOR(I),I=1,NTOT)
C
      IF (INTM.NE.1) GO TO 801
      CALL F(T,Y,YP)
      DO 802 I=1,NTOT
  802 Y(I)=YP(I)
      DO 803 I=1,NTOT
      YP(I)=(Y(I)-YSTART(I))/TDEL
  803 Y(I+NTOT)=YSTART(I+NTOT)+Y(I)*TDEL
      SUM=0.
      DO 804 I=1,NTOT
  804 SUM=SUM+(Y(I)-YP(I+NTOT))**2.
      DO 805 I=1,NTOT
  805 YP(I+NTOT)=Y(I)
      IF(SUM.LE.EPSL.OR.ITERN.GE.MITER) GO TO 806
      ITERN=ITERN+1
      GO TO 520
  806 CONTINUE
      IF(IPR(1).EQ.0) GO TO 811
      WRITE(6,226) ITERN,SUM
      CALL OUTP(T,Y,YP,0)
  811 CONTINUE
      DO 807 I=1,NTOT
      SUM=0.
      DO 808 J=1,NTOT
  808 SUM=SUM+GM(I,J)*Y(J)
  807 RHS(I)=RHS(I)-SUM+GENFE(1,I)
      GO TO 809
  801 CONTINUE
C
      IF ((T+1.D-8).GE.TSTART+TDELTA) GO TO 809
      TOUT=T+TDEL1
  699 CALL DAMSDE(ND,Y,T,TOUT,REL,ABST,IFLAG,YP)
```

```
      IF(IPR(1).GT.0) CALL OUTP(T,Y,YP,IFLAG)
      IF(IFLAG.EQ.3) GOTO 699
      IF(IFLAG.GT.3) STOP
      GO TO 520
  809 CONTINUE
C
      DO 510 I=1,NTOT*2
  510 YSTART(I)=Y(I)
      TSTART=T
      DO 822 I=1,N
      RMASS(I)=RMASSN(I)
      DO 822 J1=1,3
      R(I,J1)=RN(I,J1)
      DO 822 J2=1,3
  822 RR(I,J1,J2)=RRN(I,J1,J2)
      DO 542 I=1,NTOT
  542 FMJOIN(I)=0.
C CALCULATION OF THE ROTOR ANGLES FROM THE EQNS OF MOTION
      CALL MASSQ(Y,YP)
      NTOTF=NTOT/2
      DO 890 I=1,NTOTF
      I1=I*2-1
      I2=I*2
C      Y(NTOT+I1)=YPRE(NTOT+I1)
      Y(NTOT+I1)=(GENFI(1,I2)+BTL(I2)-EFOR(I2))/GROSK(I1,I1)+Y(NTOT+I2)
C 890 Y(I1)=YPRE(I1)
  890 Y(I1)=(Y(NTOT+I1)-YBEF(NTOT+I1))/TDEL
      WRITE(6,889) (Y(NTOT+I),I=1,NTOT)
  889 FORMAT(' *****NEW ANG=',6E14.5)
      WRITE(6,886) (Y(I),I=1,NTOT)
  886 FORMAT(' *****NEW VEL=',6E14.5)
C END CALCULATION OF ROTOR ANGLES
      IFL(12)=1
      IFL(30)=1
      IFL(10)=0
      NCST=NCSTST
      NMNC=NTOT-NCST
      GO TO 22
C
      END
C
C ****************************************************************
      SUBROUTINE CONG(Y,YP,T)
C ****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/BAUM/ PABS(3)
      COMMON/CNST1/ B(15,30)
      COMMON/CNST3/ NCST,NMNC
      COMMON/CNST7/ BDOT(15,30),AAIBD(30,30)
      COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
      DIMENSION VP(3),AP(3),APD(3),APDD(3)
      DIMENSION Y(60),YP(60)
      COMMON/CONG1/ GDOT(15)
      COMMON/CONG2/ APB(3),APDB(3)
      COMMON/DIFF1/ TDEL
C
      ALP=50.
      C1=2.1*ALP
      C2=3.4*ALP*ALP
      C3=2.7*ALP*ALP*ALP
      C4=ALP**4.
      DO 20 I=1,NCST
      VP(I)=0.
      AP(I)=0.
      DO 20 J=1,NTOT
      VP(I)=VP(I)+B(I,J)*Y(J)
   20 AP(I)=AP(I)+B(I,J)*YP(J)+BDOT(I,J)*Y(J)
      DO 21 I=1,NCST
      APD(I)=(AP(I)-APB(I))/TDEL
   21 APDD(I)=(APD(I)-APDB(I))/TDEL
      PI=3.1415926
      TT=0.5
      VAR=2.*PI/TT
      DTX=0.
      DTY=0.5
      DTZ=-0.5
      IF (T.GE.TT) GO TO 30
      GZ=0.3000000+DTZ/TT*(T-TT*DSIN(VAR*T)/2./PI)
      GY=0.7000000+DTY/TT*(T-TT*DSIN(VAR*T)/2./PI)
      GX=0.30000
C      GZ=0.1830100+DTZ/TT*(T-TT*DSIN(VAR*T)/2./PI)
C      GY=0.6160300+DTY/TT*(T-TT*DSIN(VAR*T)/2./PI)
C      GX=0.31699
      GDZ=DTZ/TT*(1.-DCOS(VAR*T))
      GDY=DTY/TT*(1.-DCOS(VAR*T))
      GDX=0.
      GDDZ=DTZ/TT*VAR*DSIN(VAR*T)
      GDDY=DTY/TT*VAR*DSIN(VAR*T)
      GDDX=0.
      GDDDZ=DTZ/TT*VAR*VAR*DCOS(VAR*T)
      GDDDY=DTY/TT*VAR*VAR*DCOS(VAR*T)
      GDDDX=0.
      GDDDDZ=-DTZ/TT*VAR**3.*DSIN(VAR*T)
      GDDDDY=-DTY/TT*VAR**3.*DSIN(VAR*T)
      GDDDDX=0.
      GO TO 40
   30 CONTINUE
      GZ=0.3000000+DTZ
      GY=0.7000000+DTY
      GX=0.30000
C      GZ=0.1830100+DTZ
C      GY=0.6160300+DTY
C      GX=0.31699
      GDX=0.
      GDY=0.
      GDZ=0.
      GDDX=0.
      GDDY=0.
      GDDZ=0.
      GDDDX=0.
      GDDDY=0.
      GDDDZ=0.
      GDDDDX=0.
      GDDDDY=0.
      GDDDDZ=0.
   40 CONTINUE
      WRITE(17,105) T,GX,GY,GZ
      WRITE(18,105) T,(GX-PABS(1)),(GY-PABS(2)),(GZ-PABS(3))
```

```
 105 FORMAT(1X,F9.5,3E15.7)
   APDD1=GDDDDX+C1*(GDDDX-APD(1))+C2*(GDDX-AP(1))
  1 +C3*(GDX-VP(1))+C4*(GX-PABS(1))
   APDD2=GDDDDY+C1*(GDDDY-APD(2))+C2*(GDDY-AP(2))
  1 +C3*(GDY-VP(2))+C4*(GY-PABS(2))
   APDD3=GDDDDZ+C1*(GDDDZ-APD(3))+C2*(GDDZ-AP(3))
  1 +C3*(GDZ-VP(3))+C4*(GZ-PABS(3))
   GDOT(1)=APDD1*TDEL*TDEL+APD(1)*TDEL+AP(1)
   GDOT(2)=APDD2*TDEL*TDEL+APD(2)*TDEL+AP(2)
   GDOT(3)=APDD3*TDEL*TDEL+APD(3)*TDEL+AP(3)
   DO 22 I=1,NCST
   APB(I)=AP(I)
 22 APDB(I)=APD(I)
   RETURN
   END
C *****************************************************************
   SUBROUTINE CONA
C *****************************************************************
   IMPLICIT REAL*8 (A-H,O-Z)
   COMMON/NEWA/AA(15,30)
   AA(1,1)=1.
   AA(2,3)=1.
   AA(3,5)=1.
   RETURN
   END
□

C *****************************************************************
   SUBROUTINE INPUT
C *****************************************************************
   IMPLICIT REAL*8 (A-H,O-Z)
   COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
   COMMON/COOR2/ IBC1(21),IBC2(21),KIBC1(21,2),KIBC2(21,2)
   COMMON/RIG1/ ANG0(14,3)
   COMMON/RIG0/ KBOL(14,3,21)
   COMMON/RIG2/ QRIG(14,3)
   COMMON/INP/ N
   COMMON/TOPO/ LBA(14)
   COMMON/MAT2/ R(14,3),RR(14,3,3)
   COMMON/MAT3/ RMASS(14)
   COMMON/PRIN0/ IPR(40),NPRI
   COMMON/FL/ IFL(40)
   COMMON/WEIG/ WGT(14,3)
C
C MAXIMUM NUMBERS:
C  # OF BODIES=14
C  # OF OF BODY RB ROT OR TRANS COOR=30
C
C NBC1 : TOTAL RB ROTATION COORDINATES
C NBC2 : TOTAL RB TRANSLATION COORDINATES
C NBC=NBC1+NBC2
C IBC1(J),J=1,NBC1 : INDEX IN 3N OF THE J TH RB ROTATION COOR (FOR RED)
C KIBC1(J,K),J=1,NBC1;K=1,2 : IF K=1, THE BODY NUMBER OF THE J TH RB
C   ROTATION COOR.; IF K=2, THE COOR NUMBER OUT OF 1,2,3 (FOR UPDATE)
C IBC2, KIBC2 : SAME AS ABOVE FOR RB TRANSLATION
C ANG0 : INITIAL RB EULER ANGLES BETWEEN EACH BODY'S JOINT AXES PAIRS.
C KBOL : BOOLEAN MATRICES FOR RB COORDINATES FOR ALL BODIES.
C R(IBOD,I) = RHO * FIRST MOM OF AREA, FOR RIGID BODY IBOD
C RR(IBOD,I)= RHO * SECOND MOM OF AREA, FOR RIGID BODY IBOD
```

```
C  RMASS(I) : MASS OF I TH BODY WHETHER RIGID
C
      CHARACTER*4 TITLE(9), TEXT
      DIMENSION IBC(6),IPRI(40)
      DATA TITLE/'TOPO','PROP','ANGL','COOR','PRIN','WEIG',
     1  'END ','COMM'/
C
      NTITLE=8
      DO 52 I=1,14
      DO 52 J=1,3
   52 WGT(I,J)=0.
    8 READ(5,9) TEXT
    9 FORMAT(A4)
      DO 10 I=1,NTITLE
      IF(TEXT.EQ.TITLE(I)) GO TO 11
   10 CONTINUE
      WRITE(6,500)
  500 FORMAT(///' ERROR RETURN FROM INPUT : KEYWORD ERROR')
      STOP
   11 II=I
      GO TO (101,102,103,104,105,106,108,8),II
C
C  READING THE TOPOLOGY
C
  101 READ(5,*) N
      READ(5,*) (LBA(I),I=1,N)
      DO 72 I=1,N
   72 IRF(I)=0
      WRITE(6,502) (LBA(I),I=1,N)
  502 FORMAT(' LBA :',14I3)
      WRITE(6,503) (IRF(I),I=1,N)
  503 FORMAT(' IRF :',14I3)
      GO TO 8
C
C  READING THE PROPERTIES OF THE BODIES
C
  102 DO 21 K=1,N
      READ(5,*) IBOD
C
      READ(5,*) (QRIG(IBOD,I),I=1,3),RMASS(IBOD),
     1 (R(IBOD,I),I=1,3),((RR(IBOD,I,J),J=1,3),I=1,3)
      WRITE(6,504) IBOD,(QRIG(IBOD,I),I=1,3),RMASS(IBOD),
     1 (R(IBOD,I),I=1,3),((RR(IBOD,I,J),J=1,3),I=1,3))
  504 FORMAT(/' BODY',I3/' D =',3E10.4/' MASS= ',E10.4/
     1  ' R =',3E10.4/' RR =',3(T8,3E10.4/))
C
C  FORMATION OF BODY RIGID BODY COORDINATE BOOLEAN MATRICES
   70 NOOFR=0
      NRT=N*3
      DO 26 I=1,N
      DO 27 K=1,3
      DO 65 L=1,NRT
   65 KBOL(I,K,L)=0
   27 KBOL(I,K,K+NOOFR)=1
      IF(IPR(5).EQ.0) GO TO 26
      WRITE(6,524) I
  524 FORMAT(/' KBOL FOR BODY',I3)
      DO 66 K=1,3
   66 WRITE(6,523) (KBOL(I,K,L),L=1,NRT)
```

```
 26 NOOFR=NOOFR+3
    GO TO 8
C
C  INITIAL ANGLES
C
 103 WRITE(6,514)
    DO 30 I=1,N
    READ(5,*) (ANG0(I,J),J=1,3)
    WRITE(6,508) I,(ANG0(I,J),J=1,3)
    DO 30 J=1,3
  30 ANG0(I,J)=ANG0(I,J)*3.1415926/180.
 514 FORMAT(/' INITIAL EULER ANGLES:')
 508 FORMAT(' BODY',I3,' :',3E10.4)
    GO TO 8
C
C   REDUCTION  OF  RIGID  BODY  RELATIVE  ROTATION  AND  TRANSLATION
COORDINATES
C   AND CALCULATING NBC1, NBC2, NBC.
C
 104 NBC1=0
    NBC2=0
    WRITE(6,509)
    DO 40 K=1,N
    J=(K-1)*3
    READ(5,*) (IBC(I),I=1,6)
    WRITE(6,510) (IBC(I),I=1,6)
    DO 41 L=1,3
    IF(IBC(L).EQ.0) GO TO 42
    NBC1=NBC1+1
    IBC1(NBC1)=J+L
    KIBC1(NBC1,1)=K
    KIBC1(NBC1,2)=L
  42 IF(IBC(L+3).EQ.0) GO TO 41
    NBC2=NBC2+1
    IBC2(NBC2)=J+L
    KIBC2(NBC2,1)=K
    KIBC2(NBC2,2)=L
  41 CONTINUE
  40 CONTINUE
    NBC=NBC1+NBC2
 509 FORMAT(/' RIGID BODY ALLOWED COORDINATES:')
 510 FORMAT(2X,6I3)
    WRITE(6,511)(I,IBC1(I),KIBC1(I,1),KIBC1(I,2),I=1,NBC1)
 511 FORMAT(/' ROTATION COOR, IBC1, KIBC1 =',(T30,4I5/))
    WRITE(6,512)(I,IBC2(I),KIBC2(I,1),KIBC2(I,2),I=1,NBC2)
 512 FORMAT(/' TRANSLATION COOR, IBC2, KIBC2 =',(T35,4I5/))
    GO TO 8
C
C  PRINTING INFORMATION
 105 READ(5,*) NPRI,(IPRI(I),I=1,NPRI)
    DO 50 I=1,NPRI
    I1=IPRI(I)
  50 IPR(I1)=1
    IF(IPR(10).GT.0) READ(5,*) IPR(10)
    GO TO 8
C
C  WEIGHT (GRAV IS GRAVITATIONAL ACC, G)
 106 IFL(2)=1
    READ(5,*) GRAV,UN1,UN2,UN3
```

```
      WRITE(6,530) UN1,UN2,UN3
  530 FORMAT(/' UNIT VECTOR ALONG GRAV. ACC. =',3E13.4)
      DO 51 I=1,N
      WGT(I,1)=GRAV*UN1
      WGT(I,2)=GRAV*UN2
   51 WGT(I,3)=GRAV*UN3
      GO TO 8
C
  108 RETURN
      END
C
C
C ****************************************************
      SUBROUTINE ABSPOS(Y,T)
C ****************************************************
C  DETERMINING ABSOLUTE POSITION OF THE REQUESTED POINT
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/PDISP/ KPOIN,KNPOIN,PPOIN(3)
      COMMON/FLEXD/ D(14,3),DD(14,3)
      COMMON/RIG4/ AKSI(14,3),AKSID(14,3)
      COMMON/CHIF/ SOK(14,3,3),SOKS(14,3,3)
      DIMENSION Y(60),PERR(3)
      COMMON/TREE/ KTREE(14,14)
      COMMON/PRIN/ IPR(40)
      COMMON/BAUM/ PABS(3)
C
      DO 10 I=1,3
      SUM=0.
      DO 11 J=1,3
   11 SUM=SUM+SOK(KPOIN,I,J)*PPOIN(J)
   10 PABS(I)=SUM
      K=KPOIN
   21 KB=KTREE(K,1)
      IF(KB.NE.0) GO TO 20
      DO 12 I=1,3
   12 PABS(I)=PABS(I)+AKSI(K,I)
      GO TO 30
   20 DO 13 I=1,3
      SUM=0.
      DO 14 J=1,3
   14 SUM=SUM+SOKS(K,I,J)*AKSI(K,J)+SOK(KB,I,J)*D(K,J)
   13 PABS(I)=PABS(I)+SUM
      K=KB
      GO TO 21
C
   30 CONTINUE
      IF(IPR(1).EQ.0.AND.IPR(2).EQ.0) RETURN
C     WRITE(6,100) T,(PABS(I),I=1,3)
C 100 FORMAT(' T=',E12.4,' ABS:',3E14.6)
      WRITE(14,101)T,(PABS(I),I=1,3)
  101 FORMAT(1X,F9.5,3E15.7)
      RETURN
      END
C
C
      SUBROUTINE PR01(STAR,ICOUN)
      LOGICAL STAR
      COMMON/PRIN0/ IPR0(40),NPRI
      COMMON/PRIN/ IPR(40)
      ICOUN=ICOUN+1
      IF(ICOUN-IPR0(10)) 611,612,612
  611 IF(.NOT.STAR) RETURN
      DO 614 I=1,40
  614 IPR(I)=0
      RETURN
  612 DO 615 I=1,40
  615 IPR(I)=IPR0(I)
      ICOUN=0
      RETURN
      END
C
C ***************************************************************
      SUBROUTINE UPDATE(Y,T)
C ***************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
C  THIS SUBROUTINE MAKES THE NECESSARY UPDATING AFTER INTEGRATION
STEP.
C  FIRST INITIALIZATION IS ALSO DONE HERE.
C
C  OUTPUTS
      COMMON/FLEXD/ D(14,3),DD(14,3)
      COMMON/RIG3/ ANG(14,3)
      COMMON/RIG4/ AKSI(14,3),AKSID(14,3)
C  INPUTS
      DIMENSION Y(60)
      COMMON/RIG1/ ANG0(14,3)
      COMMON/FLEX4/ KBOOL(4,33,75)
      COMMON/TREE/ KTREE(14,14)
      COMMON/COOR2/ IBC1(21),IBC2(21),KIBC1(21,2),KIBC2(21,2)
      COMMON/RIG2/ QRIG(14,3)
      COMMON/INP/ N
      COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
      COMMON/PRIN/ IPR(40)
      COMMON/FL/ IFL(40)
      COMMON/PDISP/ KPOIN,KNPOIN,PPOIN(3)
C
      DO 37 K=1,N
      DO 37 I=1,3
      ANG(K,I)=ANG0(K,I)
      AKSI(K,I)=0.
   37 AKSID(K,I)=0.
C     WRITE(6,211)
  211 FORMAT(' ')
C
      DO 40 K=1,N
      KB=KTREE(K,1)
      IF(KB.EQ.0) GO TO 40
      D(K,I)=QRIG(K,I)
   43 DD(K,I)=0.
   40 CONTINUE
C
C  UPDATING ANG
      IF(NBC1.EQ.0) GO TO 52
      DO 50 I=1,NBC1
      I1=KIBC1(I,1)
      I2=KIBC1(I,2)
   50 ANG(I1,I2)=ANG0(I1,I2)+Y(NTOT+I)
      IF(IPR(4).EQ.0) GO TO 52
      DO 55 K=1,N
   55 WRITE(6,106) K,(ANG(K,I),I=1,3)
  106 FORMAT(/' BODY',I3/' R.B. ANG=',3E12.4)
C
C  UPDATING AKSI, AKSID
   52 IF(NBC2.EQ.0) GO TO 53
      DO 51 I=1,NBC2
      I1=KIBC2(I,1)
      I2=KIBC2(I,2)
      AKSI(I1,I2)=Y(NTOT+NBC1+I)
   51 AKSID(I1,I2)=Y(NBC1+I)
      IF(IPR(4).EQ.0) GO TO 53
      DO 56 K=1,N
      WRITE(6,107) K,(AKSI(K,I),I=1,3)
   56 WRITE(6,108) (AKSID(K,I),I=1,3)
  107 FORMAT(/' BODY',I3/' KSI=',3E12.4)
  108 FORMAT(' KSIDOT=',3E12.4)
   53 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE PERMUT
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/PER/ E(3,3,3)
C SET THE STANDARD PERMUTATION SYMBOL
      DO 11 NN=1,3
      DO 11 I=1,3
      DO 11 M=1,3
      E(NN,I,M)=-1.
      IF((NN.EQ.M).OR.(NN.EQ.I).OR.(I.EQ.M))E(NN,I,M)=0.
      IF((NN.EQ.1).AND.(I.EQ.2).AND.(M.EQ.3))E(NN,I,M)=1.
      IF((NN.EQ.2).AND.(I.EQ.3).AND.(M.EQ.1))E(NN,I,M)=1.
      IF((NN.EQ.3).AND.(I.EQ.1).AND.(M.EQ.2))E(NN,I,M)=1.
  11  CONTINUE
      RETURN
      END
C
C ****************************************************
      SUBROUTINE TRE
C ****************************************************
      COMMON/TREE/ KTREE(14,14)
      COMMON/TOPO/ LBA(14)
      COMMON/INP/ N
C
      DO 10 I=1,N
   10 KTREE(I,1)=LBA(I)
      RETURN
      END
C
C
C ******************************************************
      SUBROUTINE SHIFT
C *********************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
```

```
C  THIS SUBROUTINE CALCULATES THE SHIFTERS SOK AND SOK* FOR ALL BODY K
C
C  OUTPUTS
      COMMON/CHIF/ SOK(14,3,3),SOKS(14,3,3)
      DIMENSION ST(3,3),S(3,3),PMAT(3,3)
C  INPUTS
      COMMON/INP/ N
      COMMON/TREE/ KTREE(14,14)
      COMMON/RIG3/ ANG(14,3)
      COMMON/PRIN/ IPR(40)
C
C  SOKS(K) WHEN KB=0 IS I, BUT NOT FORMED SINCE NEVER USED.
      DO 20 K=1,N
      KB=KTREE(K,1)
      IF(KB.EQ.0) GO TO 33
      DO 22 I=1,3
      DO 22 J=1,3
      ST(I,J)=SOK(KB,I,J)
   22 SOKS(K,I,J)=SOK(I,J)
      GO TO 30
C
   33 DO 12 I=1,3
      DO 12 J=1,3
   12 ST(I,J)=0.
      DO 13 I=1,3
   13 ST(I,I)=1.
   30 CONTINUE
C
      ALF=ANG(K,1)
      BET=ANG(K,2)
      GAM=ANG(K,3)
      CALL CH(ALF,BET,GAM,S)
      CALL PRODS(ST,S,PMAT)
      DO 31 I=1,3
      DO 31 J=1,3
      ST(I,J)=PMAT(I,J)
   31 SOK(K,I,J)=PMAT(I,J)
   20 CONTINUE
C
      IF(IPR(12).EQ.0) RETURN
      DO 40 K=1,N
      WRITE(6,100) K,((SOK(K,I,J),J=1,3),I=1,3)
   40 WRITE(6,101) K,((SOKS(K,I,J),J=1,3),I=1,3)
  100 FORMAT(/' BODY',I3,' SOK',(T15,3E10.4))
  101 FORMAT(/' BODY',I3,' SOKS',(T15,3E10.4))
      RETURN
      END
C
C
      SUBROUTINE CH(ALF,BET,GAM,S)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION S(3,3)
C
C FORMULATION OF THE SHIFTER
C    WRITE (6,100) ALF,BET,GAM
C 100 FORMAT(' ALF,BET,GAM=',3E12.4)
       S(1,1)=DCOS(GAM)*DCOS(BET)
       S(1,2)=-DCOS(BET)*DSIN(GAM)
       S(1,3)=DSIN(BET)
      S(2,1)=DCOS(ALF)*DSIN(GAM)+DSIN(ALF)*DSIN(BET)*DCOS(GAM)
      S(2,2)=DCOS(ALF)*DCOS(GAM)-DSIN(ALF)*DSIN(BET)*DSIN(GAM)
      S(2,3)=-DSIN(ALF)*DCOS(BET)
      S(3,1)=DSIN(ALF)*DSIN(GAM)-DCOS(ALF)*DSIN(BET)*DCOS(GAM)
      S(3,2)=DSIN(ALF)*DCOS(GAM)+DCOS(ALF)*DSIN(BET)*DSIN(GAM)
      S(3,3)=DCOS(ALF)*DCOS(BET)
C
      RETURN
      END
C
C *******************************************************
      SUBROUTINE PRODS(SI,S,PMAT)
C *******************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION SI(3,3),S(3,3),PMAT(3,3)
      DO 12 N=1,3
      DO 12 I=1,3
      SUM=0.
      DO 10 J=1,3
   10 SUM=SUM+SI(I,J)*S(J,N)
   12 PMAT(I,N)=SUM
      RETURN
      END
C
C
C ****************************************************
      SUBROUTINE PANGV
C ****************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
C THIS SUBROUTINE CALCULATES PARTIAL ANGULAR VELOCITY ARRAYS  WNIU,
C   MIU, AND MIUS.
C ALSO TNIU IS CALCULATED (PARTIAL VELOCITY ARRAY FOR KSI D.).
C
C OUTPUTS
      COMMON/PANG1/ WNIU(14,3,21),TNIU(14,3,21)
      DIMENSION NIU(14,3,55)
      REAL*8  NIU
C INPUTS
      COMMON/INP/ N
      COMMON/RIG0/ KBOL(14,3,21)
      COMMON/FNUM2/ NELEM(4),NNODE(4)
      COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
      COMMON/COOR2/ IBC1(21),IBC2(21),KIBC1(21,2),KIBC2(21,2)
      COMMON/TREE/ KTREE(14,14)
      COMMON/CHIF/ SOK(14,3,3),SOKS(14,3,3)
      COMMON/PRIN/ IPR(40)
C
      N3=3*N
      DO 10 K=1,N
      KB=KTREE(K,1)
      IF(KB.NE.0) GO TO 20
C CALCULATIONS WHEN KB=0
      DO 11 J=1,3
      DO 9 I=1,N3
    9 NIU(K,J,I)=KBOL(K,J,I)
      GO TO 10
C
C CALCULATIONS FOR REMAINING BODIES
```

```
  20 DO 21 I=1,N3
     DO 21 J=1,3
  21 NIU(K,J,I)=NIU(KB,J,I)
     DO 24 I=1,N3
     DO 24 J=1,3
     DO 24 M=1,3
  24 NIU(K,J,I)=NIU(K,J,I)+KBOL(K,M,I)*SOKS(K,J,M)
  10 CONTINUE
C
C  REDUCING NIU FOR ANGULAR VELOCITY : WNIU
     IF(NBC1.EQ.0) GO TO 42
     DO 40 I=1,NBC1
     L=IBC1(I)
     DO 40 K=1,N
     DO 40 M=1,3
  40 WNIU(K,M,I)=NIU(K,M,L)
C  REDUCING NIU FOR KSI : TNIU
  42 IF(NBC2.EQ.0) GO TO 43
     DO 41 I=1,NBC2
     L=IBC2(I)
     DO 41 K=1,N
     DO 41 M=1,3
  41 TNIU(K,M,I)=NIU(K,M,L)
  43 CONTINUE
C
     IF (IPR(11).EQ.0) RETURN
     DO 50 K=1,N
     WRITE(6,102) K,(L,(WNIU(K,M,L),M=1,3),L=1,NBC1)
  50 WRITE(6,103) K,(L,(TNIU(K,M,L),M=1,3),L=1,NBC2)
 102 FORMAT(/' BODY',I3,(T10,'WNIU',I2,3X,3E10.4))
 103 FORMAT(/' BODY',I3,(T10,'TNIU',I2,3X,3E10.4))
     RETURN
     END
C
C
C  **********************************************************
     SUBROUTINE OMEGA(Y)
C  ************************************************************
     IMPLICIT REAL*8 (A-H,O-Z)
C
C  COMPUTING ABSOLUTE ANGULAR VELOCITIES OF BODY REFERENCE AXIS
C
C  OUTPUTS
     COMMON/OMEG/ OMEGK(14,3),OMEGKS(14,3)
C  INPUTS
     COMMON/INP/ N
     COMMON/PANG1/ WNIU(14,3,21),TNIU(14,3,21)
     DIMENSION Y(60)
     COMMON/TREE/ KTREE(14,14)
     COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
C
     DO 10 K=1,N
     KB=KTREE(K,1)
     DO 11 M=1,3
     OMEGK(K,M)=0.
     OMEGKS(K,M)=0.
     IF(NBC1.EQ.0) GO TO 20
     DO 12 L=1,NBC1
  12 OMEGK(K,M)=OMEGK(K,M)+WNIU(K,M,L)*Y(L)
```

```
      IF(KB.EQ.0) GO TO 11
      IF(NBC1.EQ.0) GO TO 11
      DO 13 L=1,NBC1
   13 OMEGKS(K,M)=OMEGKS(K,M)+WNIU(KB,M,L)*Y(L)
   11 CONTINUE
   10 CONTINUE
      RETURN
      END
C
C
C ******************************************************
      SUBROUTINE SHIFTD
C ******************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
C CALCULATING SHIFTER DERIVATIVES SOKD AND SOKSD, FOR ALL BODIES K
C
C OUTPUTS
      COMMON/CHIFD/ SOKD(14,3,3),SOKSD(14,3,3)
      DIMENSION WOK(3,3),WOKS(3,3),SK(3,3),SKS(3,3)
      DIMENSION PMATK(3,3),PMATKS(3,3)
C INPUTS
      COMMON/CHIF/ SOK(14,3,3),SOKS(14,3,3)
      COMMON/OMEG/ OMEGK(14,3),OMEGKS(14,3)
      COMMON/INP/ N
      COMMON/PER/ E(3,3,3)
      COMMON/PRIN/ IPR(40)
C
      DO 10 K=1,N
      DO 12 I=1,3
      DO 12 J=1,3
      SK(I,J)=SOK(K,I,J)
      SKS(I,J)=SOKS(K,I,J)
      WOK(I,J)=0.
      WOKS(I,J)=0.
      DO 12 IN=1,3
      WOK(I,J)=WOK(I,J)-E(I,J,IN)*OMEGK(K,IN)
   12 WOKS(I,J)=WOKS(I,J)-E(I,J,IN)*OMEGKS(K,IN)
      CALL PRODS(WOK,SK,PMATK)
      CALL PRODS(WOKS,SKS,PMATKS)
      DO 15 I=1,3
      DO 15 J=1,3
      SOKD(K,I,J)=PMATK(I,J)
   15 SOKSD(K,I,J)=PMATKS(I,J)
   10 CONTINUE
C
      IF(IPR(12).EQ.0) RETURN
      DO 40 K=1,N
      WRITE(6,102) K,(OMEGK(K,I),I=1,3)
      WRITE(6,103) ((WOK(I,J),J=1,3),I=1,3)
      WRITE(6,100) K,((SOKD(K,I,J),J=1,3),I=1,3)
   40 WRITE(6,101) K,((SOKSD(K,I,J),J=1,3),I=1,3)
  102 FORMAT(/' BODY',I3,' OMEGK',2X,3E12.4)
  103 FORMAT(/' WOK=',(T10,3E12.4))
  100 FORMAT(/' BODY',I3,' SOKD',(T20,3E12.4))
  101 FORMAT(/' BODY',I3,' SOKSD',(T20,3E12.4))
      RETURN
      END
C
```

```
C ******************************************************
    SUBROUTINE PANGVD
C ******************************************************
    IMPLICIT REAL*8 (A-H,O-Z)
C
C THIS SUBROUTINE CALCULATRES WNIUD, MIUD, MIUSD AND TNIUD
C
C OUTPUTS
    COMMON/PANG1D/ WNIUD(14,3,21),TNIUD(14,3,21)
    DIMENSION NIUD(14,3,42)
    REAL*8  NIUD
C INPUTS
    COMMON/INP/ N
    COMMON/RIG0/ KBOL(14,3,21)
    COMMON/FNUM2/ NELEM(4),NNODE(4)
    COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
    COMMON/COOR2/ IBC1(21),IBC2(21),KIBC1(21,2),KIBC2(21,2)
    COMMON/TREE/ KTREE(14,14)
    COMMON/CHIFD/ SOKD(14,3,3),SOKSD(14,3,3)
    COMMON/PRIN/ IPR(40)
C
    N3=3*N
    DO 10 K=1,N
    KB=KTREE(K,1)
    IF(KB.NE.0) GO TO 20
C  CALCULATIONS WHEN KB=0
    DO 9 I=1,N3
  9 NIUD(K,J,I)=0.
    GO TO 10
C
C  CALCULATIONS FOR REMAINING BODIES
 20 DO 21 I=1,N3
    DO 21 J=1,3
 21 NIUD(K,J,I)=NIUD(KB,J,I)
    DO 24 I=1,N3
    DO 24 J=1,3
    DO 24 M=1,3
 24 NIUD(K,J,I)=NIUD(K,J,I)+KBOL(K,M,I)*SOKSD(K,J,M)
 10 CONTINUE
C
C  REDUCING NIU FOR ANGULAR VELOCITY : WNIU
    IF(NBC1.EQ.0) GO TO 42
    DO 40 I=1,NBC1
    L=IBC1(I)
    DO 40 K=1,N
    DO 40 M=1,3
 40 WNIUD(K,M,I)=NIUD(K,M,L)
C  REDUCING NIU FOR KSI : TNIU
 42 IF(NBC2.EQ.0) GO TO 43
    DO 41 I=1,NBC2
    L=IBC2(I)
    DO 41 K=1,N
    DO 41 M=1,3
 41 TNIUD(K,M,I)=NIUD(K,M,L)
 43 CONTINUE
C
    IF (IPR(11).EQ.0) RETURN
    DO 50 K=1,N
    WRITE(6,102) K,(L,(WNIUD(K,M,L),M=1,3),L=1,NBC1)
 50 WRITE(6,103) K,(L,(TNIUD(K,M,L),M=1,3),L=1,NBC2)
102 FORMAT(/' BODY',I3,(T10,'WNIUD',I2,3X,3E10.4))
103 FORMAT(/' BODY',I3,(T10,'TNIUD',I2,3X,3E10.4))
    RETURN
    END
C
C
C ****************************************************
    SUBROUTINE PVEL
C ****************************************************
    IMPLICIT REAL*8 (A-H,O-Z)
C
C THIS SUBROUTINE CALCULATES PARTIAL VELOCITY VECTOR COMPONENTS
C GA, GDRIG
C
C OUTPUTS
    COMMON/PVEL1/ GA(14,3,21),GDRIG(14,3,21,3)
C INPUTS
    COMMON/INP/ N
    COMMON/TREE/ KTREE(14,14)
    COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
    COMMON/PRIN/ IPR(40)
C
    DO 10 K=1,N
    DO 10 I=1,3
    DO 9 J=1,NBC1
  9 GA(K,I,J)=0.
C
    DO 20 K=1,N
    KB=KTREE(K,1)
    IF(KB.EQ.0) GO TO 100
    DO 21 I=1,3
    DO 22 J=1,NBC1
 22 GA(K,I,J)=GA(KB,I,J)
C
    CALL MULT(1,K,KB)
C
100 CONTINUE
    CALL MULT(2,K,KB)
 20 CONTINUE
C
    RETURN
    END
C
C
C ****************************************************
    SUBROUTINE PVELD
C ****************************************************
    IMPLICIT REAL*8 (A-H,O-Z)
C
C THIS SUBROUTINE CALCULATES PARTIAL VELOCITY VECTOR COMPONENT
C DERIVATIVES GAD, GDRIGD
C
C OUTPUTS
    COMMON/PVEL3/ GAD(14,3,21),GDRIGD(14,3,21,3)
C INPUTS
    COMMON/INP/ N
    COMMON/TREE/ KTREE(14,14)
    COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
```

```
      COMMON/PRIN/ IPR(40)
C
      DO 10 K=1,N
      DO 10 I=1,3
      DO 9 J=1,NBC1
    9 GAD(K,I,J)=0.
C
      DO 20 K=1,N
      KB=KTREE(K,1)
      IF(KB.EQ.0) GO TO 100
      DO 21 I=1,3
      DO 22 J=1,NBC1
   22 GAD(K,I,J)=GAD(KB,I,J)
C
      CALL MULT(3,K,KB)
C
  100 CONTINUE
      CALL MULT(4,K,KB)
   20 CONTINUE
      RETURN
      END
C
C
C ************************************************
      SUBROUTINE MULT(NCODE,K,KB)
C ************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
C THIS SUBROUTINE PERFORMS THE MULTIPLICATIONS FOR THE GENERATION OF
C THE PARTIAL VELOCITY VECTOR COMPONENTS.
C
C OUTPUTS
      COMMON/PVEL1/ GA(14,3,21),GDRIG(14,3,21,3)
      COMMON/PVEL3/ GAD(14,3,21),GDRIGD(14,3,21,3)
C INPUTS
      COMMON/FLEXD/ D(14,3),DD(14,3)
      COMMON/RIG4/ AKSI(14,3),AKSID(14,3)
      COMMON/CHIF/ SOK(14,3,3),SOKS(14,3,3)
      COMMON/CHIFD/ SOKD(14,3,3),SOKSD(14,3,3)
      COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
      COMMON/PANG1/ WNIU(14,3,21),TNIU(14,3,21)
      COMMON/PANG1D/ WNIUD(14,3,21),TNIUD(14,3,21)
      DIMENSION VAR2(3,3),VAR3(3,3),VAR2D(3,3),VAR3D(3,3),
     1 VAR5(3,3,42),VAR5D(3,3,42),VAR7(3,3,3),VAR7D(3,3,3)
      COMMON/PER/ E(3,3,3)
C
      DO 20 I=1,3
      DO 21 J=1,3
      VAR2(I,J)=0.
      VAR2D(I,J)=0.
      VAR3(I,J)=0.
   21 VAR3D(I,J)=0.
      DO 20 J=1,3
      DO 23 L=1,3
      VAR7(I,J,L)=0.
   23 VAR7D(I,J,L)=0.
      DO 24 L=1,NBC1
      VAR5(I,J,L)=0.
   24 VAR5D(I,J,L)=0.
```

```
C
      GO TO (10,12,14,16),NCODE
C
   10 DO 100 I1=1,3
      DO 100 I2=1,3
      DO 100 I3=1,3
      DO 100 I4=1,3
      VAR2(I3,I4)=VAR2(I3,I4)+E(I3,I4,I2)*AKSI(K,I1)*SOKS(K,I2,I1)
  100 VAR3(I3,I4)=VAR3(I3,I4)+E(I3,I4,I2)*D(K,I1)*SOK(KB,I2,I1)
      DO 101 M=1,3
      DO 101 I=1,3
      DO 102 L=1,NBC1
  102 GA(K,M,L)=GA(K,M,L)+WNIU(KB,I,L)*(VAR2(M,I)+VAR3(M,I))
      RETURN
C
   12 DO 117 M=1,3
      DO 117 I1=1,3
      DO 117 I2=1,3
      DO 117 I3=1,3
  117 VAR7(M,I1,I2)=VAR7(M,I1,I2)+SOK(K,I3,I2)*E(M,I1,I3)
      DO 110 I1=1,3
      DO 110 I2=1,3
      DO 110 M=1,3
      DO 111 L=1,NBC1
  111 VAR5(M,I2,L)=VAR5(M,I2,L)+VAR7(M,I1,I2)*WNIU(K,I1,L)
C
      DO 114 M=1,3
      DO 114 I2=1,3
      DO 115 L=1,NBC1
  115 GDRIG(K,M,L,I2)=VAR5(M,I2,L)
      RETURN
C
C  CALCULATION OF THE DERIVATIVES
C
   14 DO 200 I1=1,3
      DO 200 I2=1,3
      DO 200 I3=1,3
      DO 200 I4=1,3
      VAR2(I3,I4)=VAR2(I3,I4)+E(I3,I4,I2)*AKSI(K,I1)*SOKS(K,I2,I1)
      VAR3(I3,I4)=VAR3(I3,I4)+E(I3,I4,I2)*D(K,I1)*SOK(KB,I2,I1)
      VAR2D(I3,I4)=VAR2D(I3,I4)+E(I3,I4,I2)*(AKSID(K,I1)*SOKS(K,I2,I1)
     1   +AKSI(K,I1)*SOKSD(K,I2,I1))
  200 VAR3D(I3,I4)=VAR3D(I3,I4)+E(I3,I4,I2)*(DD(K,I1)*SOK(KB,I2,I1)
     1    +D(K,I1)*SOKD(KB,I2,I1))
      DO 201 M=1,3
      DO 201 I=1,3
      DO 202 L=1,NBC1
  202 GAD(K,M,L)=GAD(K,M,L)+WNIU(KB,I,L)*(VAR2D(M,I)+VAR3D(M,I))
     1    +WNIUD(KB,I,L)*(VAR2(M,I)+VAR3(M,I))
      RETURN
C
   16 DO 217 M=1,3
      DO 217 I1=1,3
      DO 217 I2=1,3
      DO 217 I3=1,3
      VAR7(M,I1,I2)=VAR7(M,I1,I2)+SOK(K,I3,I2)*E(M,I1,I3)
  217 VAR7D(M,I1,I2)=VAR7D(M,I1,I2)+SOKD(K,I3,I2)*E(M,I1,I3)
      DO 210 I1=1,3
      DO 210 I2=1,3
   DO 210 M=1,3
   DO 211 L=1,NBC1
   VAR5(M,I2,L)=VAR5(M,I2,L)+VAR7(M,I1,I2)*WNIU(K,I1,L)
 211 VAR5D(M,I2,L)=VAR5D(M,I2,L)+VAR7D(M,I1,I2)*WNIU(K,I1,L)
   1   +VAR7(M,I1,I2)*WNIUD(K,I1,L)
C
   DO 214 M=1,3
   DO 214 I2=1,3
   DO 215 L=1,NBC1
 215 GDRIGD(K,M,L,I2)=VAR5D(M,I2,L)
   RETURN
   END
C
C
      SUBROUTINE ELIM(AB,N,NP)
      DIMENSION AB(30,60)
C****
   IMPLICIT REAL*8 (A-H,O-Z)
C    DET= AB(1,1)*(AB(2,2)*AB(3,3)-AB(3,2)*AB(2,3))
C   1 -AB(1,2)*(AB(2,1)*AB(3,3)-AB(3,1)*AB(2,3))
C   2 +AB(1,3)*(AB(2,1)*AB(3,2)-AB(3,1)*AB(2,2))
C    WRITE(6,60) DET
C  60 FORMAT(' DET=',F14.4)
C****
C THIS SUBROUTINE SOLVES A SET OF LINEAR EQUATIONS.
C THE GAUSS ELIMINATION METHOD IS USED ,WITH PARTIAL PIVOTING.
C MULTIPLE RIGHT HAND SIDES ARE PERMITTED,THEY SHOULD BE SUPLIED
C AS COLUMNS THAT AUGMENT THE COEFFICIENT MATRIX.
C PARAMETERS ARE-
C    AB   COEFFICIENT MATRIX AUGMENTED WITH R.H.S VECTORS
C    N    NUMBER OF EQUATIONS
C    NP   TOTAL NUMBER OF COLUMNS IN THE AUGMENTED MATRIX.
C    NDIM  FIRST DIMENSION OF MATRIX AB IN THE CALLING PROGRAM.
C  THE SOLUTION  VECTOR(S) ARE RETURNED IN THE AUGMENTED
C  COLUMNS OF AB.
C
C BEGIN  THE REDUCTION
    NDIM=N
    NM1=N-1
    DO 35 I = 1,NM1
C FIND THE ROW NUMBERS OF THE PIVOT ROW.WE WILL THEN
C INTERCHANGE ROWS TO PUT THE PIVOT ELEMENT ON THE DIAGONAL .
      IPVT = I
      IP1 = I + 1
      DO 10 J =IP1,N
        IF(DABS(AB(IPVT,I)).LT.DABS(AB(J,I))) IPVT = J
 10     CONTINUE
C CHECK TO BE SURE THE PIVOT ELEMENT IS NOT TOO SMALL,IF SO
C PRINT A MESSAGE AND RETURN.
        IF(DABS(AB(IPVT,I)).LT. 1.D-5) GO TO 99
C NOW INTERCHANGE ,EXECPT IF THE PIVOT ELEMENT IS ALREADY ON
C THE DIAGONAL ,DON`T NEED TO.
         IF(IPVT.EQ.I) GO TO 25
       DO 20 JCOL = I,NP
        SAVE = AB(I,JCOL)
        AB(I,JCOL)=AB(IPVT,JCOL)
        AB(IPVT,JCOL)=SAVE
 20     CONTINUE
C NOW REDUCE ALL ELEMENTS BELOW THE DIAGONAL IN THE I-TH ROW.CHECK
```

```
C CAN FIRST TO SEE IF A ZERO ALREADY PRESENT.IF SO,
C CAN SKIP REDUCTION FOR THAT ROW.
 25        DO 32 JROW = IP1,N
          IF(AB(JROW,I).EQ.0) GO TO 32
          RATIO = AB(JROW,I)/AB(I,I)
         DO 30 KCOL = IP1,NP
          AB(JROW,KCOL)=AB(JROW,KCOL)-RATIO*AB(I,KCOL)
 30        CONTINUE
 32      CONTINUE
 35     CONTINUE
C WE STILL NEED TO CHECK A(N,N) FOR SIZE.
      IF(DABS(AB(N,N)).LT.1.D-5) GO TO 99
C NOW WE BACK SUBSTITUTE
      NP1 = N + 1
      DO 50 KCOL = NP1,NP
       AB(N,KCOL)= AB(N,KCOL)/AB(N,N)
      DO 45 J=2,N
        NVBL= NP1-J
        L = NVBL + 1
        VALUE=AB(NVBL,KCOL)
       DO 40 K= L,N
         VALUE=VALUE-AB(NVBL,K)*AB(K,KCOL)
 40      CONTINUE
       AB(NVBL,KCOL)=VALUE/AB(NVBL,NVBL)
 45    CONTINUE
 50    CONTINUE
      RETURN
C MESSAGE FOR A NEAR SINGULAR MATRIX
 99   WRITE(6,100)
 100  FORMAT('1',10X,'SOLUTION NOT FEASIBLE.ANEAR ZERO PIVOT WAS
   $  ENCOUTERED.')
      RETURN
      END
C
C.....................................................
          SUBROUTINE OUTP(T,Y,YP,IFLAG)
    IMPLICIT REAL*8 (A-H,O-Z)
C.....................................................
          DIMENSION Y(60),YP(60),YANG(30)
          COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
CC
          DO 10 I=1,NBC1
  10        YANG(I)=Y(NTOT+I)*180./3.1415926
          DO 11 I=1,NBC2
  11        YANG(NBC1+I)=Y(NTOT+NBC1+I)
C
          WRITE(6,105)IFLAG,T
C          WRITE(6,104)T
          WRITE(6,100)
          WRITE(6,101)(YANG(I),I=1,NTOT)
    WRITE(16,101)T,YANG(2)-YANG(1),YANG(4)-YANG(3),YANG(6)-YANG(5)
          WRITE(11,201)T,(YANG(I),I=1,NBC)
          WRITE(6,102)
          WRITE(6,101)(Y(I),I=1,NTOT)
          WRITE(12,201)T,(Y(I),I=1,NBC)
          WRITE(6,103)
          WRITE(6,101)(YP(I),I=1,NTOT)
          WRITE(15,201)T,(YP(I),I=1,NTOT)
          WRITE(6,106)
```

```
C
C
C
 201   FORMAT(1X,F9.5,12E13.5)
 104   FORMAT(10X,'TIME=',F12.4)
 100   FORMAT(10X,'DISPLACEMENT')
 101   FORMAT(3X,6E12.4)
 102   FORMAT(10X,'VELOCITY')
 103   FORMAT(10X,'ACCELERATION')
 105   FORMAT(/10X,'IFLAG=',I2,4X,'TIME=',F10.4)
 106   FORMAT(' ')
C
      RETURN
      END
C.................................................................
    SUBROUTINE DAMSDE(NEQN,Y,T,TOUT,RELERR,ABSERR,IFLAG,YP)
C.................................................................
    IMPLICIT REAL*8 (A-H,O-Z)
C  SUBROUTINE  DE  INTEGRATES A SYSTEM OF UP TO 20 FIRST ORDER ORDINARY
C  DIFFERENTIAL EQUATIONS OF THE FORM
C        DY(I)/DT = F(T,Y(1),Y(2),...,Y(NEQN))
C        Y(I) GIVEN AT  T .
C  THE SUBROUTINE INTEGRATES FROM  T  TO  TOUT .  ON RETURN THE
C  PARAMETERS IN THE CALL LIST ARE INITIALIZED FOR CONTINUING THE
C  INTEGRATION.  THE USER HAS ONLY TO DEFINE A NEW VALUE  TOUT
C  AND CALL  DE  AGAIN.
C
C  DE  CALLS TWO CODES, THE INTEGRATOR  STEP  AND THE INTERPOLATION
C  ROUTINE  INTRP .  STEP  USES A MODIFIED DIVIDED DIFFERENCE FORM OF
C  THE ADAMS PECE FORMULAS AND LOCAL EXTRAPOLATION.  IT ADJUSTS THE
C   ORDER AND STEP SIZE TO CONTROL THE LOCAL ERROR.  NORMALLY EACH
CALL
C  TO  STEP  ADVANCES THE SOLUTION ONE STEP IN THE DIRECTION OF  TOUT .
C  THOUGH NEVER BEYOND T + 10*(TOUT-T), AND CALLS  INTRP  TO
C  INTERPOLATE THE SOLUTION AT  TOUT .  AN OPTION IS PROVIDED TO STOP
C  THE INTEGRATION AT  TOUT  BUT IT SHOULD BE USED ONLY IF IT IS
C  IMPOSSIBLE TO CONTINUE THE INTEGRATION BEYOND  TOUT .
C
C  THIS CODE IS COMPLETELY EXPLAINED AND DOCUMENTED IN THE TEXT,
C  COMPUTER SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS:  THE INITIAL
C  VALUE PROBLEM  BY L. F. SHAMPINE AND M. K. GORDON.
C
C  THE PARAMETERS FOR  DE  ARE:
C    F – SUBROUTINE F(T,Y,YP) TO EVALUATE DERIVATIVES YP(I)=DY(I)/DT
C    NEQN – NUMBER OF EQUATIONS TO BE INTEGRATED
C    Y(*) – SOLUTION VECTOR AT  T
C    T – INDEPENDENT VARIABLE
C    TOUT – POINT AT WHICH SOLUTION IS DESIRED
C        RELERR,ABSERR  –  RELATIVE  AND  ABSOLUTE  ERROR  TOLERANCES  FOR
LOCAL
C       ERROR TEST.  AT EACH STEP THE CODE REQUIRES
C         ABS(LOCAL ERROR) .LE. ABS(Y)*RELERR + ABSERR
C       FOR EACH COMPONENT OF THE LOCAL ERROR AND SOLUTION VECTORS
C    IFLAG – INDICATES STATUS OF INTEGRATION
C
C  FIRST CALL TO DE –
C
C  THE USER MUST PROVIDE STORAGE IN HIS CALLING PROGRAM FOR THE ARRAY
C  IN THE CALL LIST, Y(NEQN), DECLARE  F  IN AN EXTERNAL STATEMENT,
C  SUPPLY THE SUBROUTINE F(T,Y,YP) TO EVALUATE
C    DY(I)/DT = YP(I) = F(T,Y(1),Y(2),...,Y(NEQN))
C  AND INITIALIZE THE PARAMETERS:
C    NEQN – NUMBER OF EQUATIONS TO BE INTEGRATED
C    Y(*) – VECTOR OF INITIAL CONDITIONS
C    T – STARTING POINT OF INTEGRATION
C    TOUT – POINT AT WHICH SOLUTION IS DESIRED
C    RELERR,ABSERR – RELATIVE AND ABSOLUTE LOCAL ERROR TOLERANCES
C    IFLAG – +1,-1.  INDICATOR TO INITIALIZE THE CODE.  NORMAL INPUT
C       IS +1.  THE USER SHOULD SET IFLAG=-1 ONLY IF IT IS
C       IMPOSSIBLE TO CONTINUE THE INTEGRATION BEYOND  TOUT .
C  ALL PARAMETERS EXCEPT  F ,  NEQN  AND  TOUT  MAY BE ALTERED BY THE
C  CODE ON OUTPUT SO MUST BE VARIABLES IN THE CALLING PROGRAM.
C
C  OUTPUT FROM  DE –
C
C    NEQN – UNCHANGED
C    Y(*) – SOLUTION AT  T
C    T – LAST POINT REACHED IN INTEGRATION.  NORMAL RETURN HAS
C       T = TOUT .
C    TOUT – UNCHANGED
C        RELERR,ABSERR  –  NORMAL  RETURN  HAS  TOLERANCES  UNCHANGED.
IFLAG=3
C       SIGNALS TOLERANCES INCREASED
C    IFLAG = 2 – NORMAL RETURN.  INTEGRATION REACHED  TOUT
C        = 3 – INTEGRATION DID NOT REACH  TOUT  BECAUSE ERROR
C            TOLERANCES TOO SMALL.  RELERR ,  ABSERR  INCREASED
C            APPROPRIATELY FOR CONTINUING
C        = 4 – INTEGRATION DID NOT REACH  TOUT  BECAUSE MORE THAN
C            MAXNUM  STEPS NEEDED
C        = 5 – INTEGRATION DID NOT REACH  TOUT  BECAUSE EQUATIONS
C            APPEAR TO BE STIFF
C        = 6 – INVALID INPUT PARAMETERS (FATAL ERROR)
C       THE VALUE OF  IFLAG  IS RETURNED NEGATIVE WHEN THE INPUT
C       VALUE IS NEGATIVE AND THE INTEGRATION DOES NOT REACH  TOUT ,
C       I.E., -3, -4, -5.
C
C  SUBSEQUENT CALLS TO  DE –
C
C  SUBROUTINE  DE  RETURNS WITH ALL INFORMATION NEEDED TO CONTINUE
C  THE INTEGRATION.  IF THE INTEGRATION REACHED  TOUT , THE USER NEED
C  ONLY DEFINE A NEW  TOUT  AND CALL AGAIN.  IF THE INTEGRATION DID NOT
C  REACH  TOUT  AND THE USER WANTS TO CONTINUE, HE JUST CALLS AGAIN.
C  THE OUTPUT VALUE OF  IFLAG  IS THE APPROPRIATE INPUT VALUE FOR
C  SUBSEQUENT CALLS.  THE ONLY SITUATION IN WHICH IT SHOULD BE ALTERED
C  IS TO STOP THE INTEGRATION INTERNALLY AT THE NEW  TOUT , I.E.,
C  CHANGE OUTPUT  IFLAG=2  TO INPUT  IFLAG=-2 .  ERROR TOLERANCES MAY
C   BE  CHANGED  BY  THE  USER  BEFORE  CONTINUING.   ALL  OTHER  PARAMETERS
MUST
C  REMAIN UNCHANGED.
C
   LOGICAL START,CRASH,STIFF
   DIMENSION Y(60),PSI(20),A(60),IA(60)
   DIMENSION YY(60),WT(60),PHI(60,16),P(60),YP(60),YPOUT(60)
C   EXTERNAL F
C
C*********************************************************************
C*  THE ONLY MACHINE DEPENDENT CONSTANT IS BASED ON THE MACHINE UNIT
*
```

```
C* ROUNDOFF ERROR  U  WHICH IS THE SMALLEST POSITIVE NUMBER SUCH THAT
*
C* 1.0+U .GT. 1.0 . U MUST BE CALCULATED AND FOURU=4.0*U INSERTED *
C* IN THE FOLLOWING DATA STATEMENT BEFORE USING DE . THE ROUTINE    *
C* MACHIN CALCULATES U . FOURU AND TWOU=2.0*U MUST ALSO BE    *
C* INSERTED IN SUBROUTINE STEP BEFORE CALLING DE .          *
   DATA FOURU/3.8E-6/
C***********************************************************************
C
C  THE CONSTANT  MAXNUM  IS THE MAXIMUM NUMBER OF STEPS ALLOWED IN
ONE
C  CALL TO DE . THE USER MAY CHANGE THIS LIMIT BY ALTERING THE
C  FOLLOWING STATEMENT
   DATA MAXNUM/500/
C
C       ***       ***       ***
C
   IF(NEQN .LT. 1              ) GO TO 10
C
C  TEST FOR IMPROPER PARAMETERS                         D
C
   IF(T .EQ. TOUT) GO TO 10
   IF(RELERR .LT. 0. .OR. ABSERR .LT. 0.)GO TO 10
   EPS = AMAX1(RELERR,ABSERR)
   IF(EPS .LE. 0.)GO TO 10
   IF(IFLAG .EQ. 0) GO TO 10
   ISN = ISIGN(1,IFLAG)
   IFLAG = IABS(IFLAG)
   IF(IFLAG .EQ. 1) GO TO 20
   IF(T .NE. TOLD) GO TO 10
   IF(IFLAG .GE. 2 .AND. IFLAG .LE. 5) GO TO 20
 10 IFLAG = 6
   GOTO 140
C
C  ON EACH CALL SET INTERVAL OF INTEGRATION AND COUNTER FOR NUMBER
OF
C  STEPS. ADJUST INPUT ERROR TOLERANCES TO DEFINE WEIGHT VECTOR FOR
C
 20 DEL = TOUT - T
   ABSDEL = ABS(DEL)
   TEND = T + 10.0 *DEL
   IF(ISN .LT. 0) TEND = TOUT
   NOSTEP = 0
   KLE4 = 0
   STIFF = .FALSE.
   RELEPS = RELERR/EPS
   ABSEPS = ABSERR/EPS
   IF(IFLAG .EQ. 1) GO TO 30
   IF(ISNOLD .LT. 0) GO TO 30
   IF(DELSGN*DEL .GT. 0.)GO TO 50
C
C  ON START AND RESTART ALSO SET WORK VARIABLES X AND YY(*), STORE THE
C  DIRECTION OF INTEGRATION AND INITIALIZE THE STEP SIZE
C
 30 START = .TRUE.
   X = T
   DO 40 L = 1,NEQN
 40  YY(L) = Y(L)
   DELSGN = SIGN(1. ,DEL)
```

```
      H=AMAX1( ABS(TOUT-X),FOURU* ABS(X))
      H= SIGN(H,TOUT-X)
C
C   IF ALREADY PAST OUTPUT POINT, INTERPOLATE AND RETURN
C
   50 IF( ABS(X-T).LT. ABSDEL) GO TO 60
C
      CALL INTRP(X,YY,TOUT,Y,YP,NEQN,KOLD,PHI,PSI)
      IFLAG = 2
      T = TOUT
      TOLD = T
      ISNOLD = ISN
      GOTO 140
C
C   IF CANNOT GO PAST OUTPUT POINT AND SUFFICIENTLY CLOSE,
C   EXTRAPOLATE AND RETURN
C
   60 IF(ISN .GT. 0  .OR.  ABS(TOUT-X) .GE. FOURU* ABS(X))GO TO 80
      H = TOUT - X
      CALL F(X,YY,YP)
      DO 70 L = 1,NEQN
   70   Y(L) = YY(L) + H*YP(L)
      IFLAG = 2
      T = TOUT
      TOLD = T
      ISNOLD = ISN
      GOTO 140
C
C   TEST FOR TOO MUCH WORK
C
   80 IF(NOSTEP .LT. MAXNUM) GO TO 100
      IFLAG = ISN*4
      IF(STIFF) IFLAG = ISN*5
      DO 90 L = 1,NEQN
   90   Y(L) = YY(L)
      T = X
      TOLD = T
      ISNOLD = 1
      GOTO 140
C
C   LIMIT STEP SIZE, SET WEIGHT VECTOR AND TAKE A STEP
C
  100 H = SIGN(AMIN1( ABS(H), ABS(TEND-X)     ),H)
      DO 110 L = 1,NEQN
  110   WT(L) =RELEPS* ABS(YY(L)) + ABSEPS
C
      CALL STEP(X,YY,NEQN,H,EPS,WT,START,
     1  HOLD,K,KOLD,CRASH,PHI,P,YP,PSI)
C
C
C   TEST FOR TOLERANCES TOO SMALL
C
      IF(.NOT.CRASH) GO TO 130
      IFLAG = ISN*3
      RELERR = EPS*RELEPS
      ABSERR = EPS*ABSEPS
      DO 120 L = 1,NEQN
  120   Y(L) = YY(L)
      T = X
   TOLD = T
   ISNOLD = 1
   GOTO 140
C
C  AUGMENT COUNTER ON WORK AND TEST FOR STIFFNESS
C
 130 NOSTEP = NOSTEP + 1
   KLE4 = KLE4 + 1
   IF(KOLD .GT. 4) KLE4 = 0
   IF(KLE4 .GE. 50) STIFF = .TRUE.
   GO TO 50
 140 RETURN
   END
C
C
C.........................................................
    SUBROUTINE STEP(X,Y,NEQN,H,EPS,WT,START,HOLD,K,KOLD,
   1  CRASH,PHI,P,YP,PSI)
   IMPLICIT REAL*8 (A-H,O-Z)
C.........................................................
C  CALLED BY == DE ==
C
C  SUBROUTINE  STEP  INTEGRATES A SYSTEM OF FIRST ORDER ORDINARY
C  DIFFERENTIAL EQUATIONS ONE STEP, NORMALLY FROM X TO X+H, USING A
C    MODIFIED DIVIDED DIFFERENCE FORM OF THE ADAMS PECE FORMULAS.
LOCAL
C  EXTRAPOLATION IS USED TO IMPROVE ABSOLUTE STABILITY AND ACCURACY.
C  THE CODE ADJUSTS ITS ORDER AND STEP SIZE TO CONTROL THE LOCAL ERROR
C  PER UNIT STEP IN A GENERALIZED SENSE.  SPECIAL DEVICES ARE INCLUDED
C    TO CONTROL ROUNDOFF ERROR AND TO DETECT WHEN THE USER IS
REQUESTING
C  TOO MUCH ACCURACY.
C
C  THIS CODE IS COMPLETELY EXPLAINED AND DOCUMENTED IN THE TEXT,
C  COMPUTER SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS:  THE INITIAL
C  VALUE PROBLEM  BY L. F. SHAMPINE AND M. K. GORDON.
C
C
C  THE PARAMETERS REPRESENT:
C    X -- INDEPENDENT VARIABLE
C    Y(*) -- SOLUTION VECTOR AT X
C    YP(*) -- DERIVATIVE OF SOLUTION VECTOR AT  X  AFTER SUCCESSFUL
C       STEP
C    NEQN -- NUMBER OF EQUATIONS TO BE INTEGRATED
C    H -- APPROPRIATE STEP SIZE FOR NEXT STEP.  NORMALLY DETERMINED BY
C       CODE
C    EPS -- LOCAL ERROR TOLERANCE.  MUST BE VARIABLE
C    WT(*) -- VECTOR OF WEIGHTS FOR ERROR CRITERION
C    START -- LOGICAL VARIABLE SET .TRUE. FOR FIRST STEP,  .FALSE.
C       OTHERWISE
C    HOLD -- STEP SIZE USED FOR LAST SUCCESSFUL STEP
C    K -- APPROPRIATE ORDER FOR NEXT STEP (DETERMINED BY CODE)
C    KOLD -- ORDER USED FOR LAST SUCCESSFUL STEP
C    CRASH -- LOGICAL VARIABLE SET .TRUE. WHEN NO STEP CAN BE TAKEN,
C       .FALSE. OTHERWISE.
C  THE ARRAYS  PHI, PSI  ARE REQUIRED FOR THE INTERPOLATION SUBROUTINE
C  INTRP .  THE ARRAY  P  IS INTERNAL TO THE CODE.
C
C  INPUT TO  STEP
```

```
C
C   FIRST CALL -
C
C  THE USER MUST PROVIDE STORAGE IN HIS DRIVER PROGRAM FOR ALL ARRAYS
C  IN THE CALL LIST, NAMELY
C
C   DIMENSION Y(NEQN),WT(NEQN),PHI(NEQN,16),P(NEQN),YP(NEQN),PSI(12)
C
C  THE USER MUST ALSO DECLARE  START  AND  CRASH  LOGICAL VARIABLES
C  AND  F  AN EXTERNAL SUBROUTINE, SUPPLY THE SUBROUTINE  F(X,Y,YP)
C  TO EVALUATE
C    DY(I)/DX = YP(I) = F(X,Y(1),Y(2),...,Y(NEQN))
C  AND INITIALIZE ONLY THE FOLLOWING PARAMETERS:
C    X - INITIAL VALUE OF THE INDEPENDENT VARIABLE
C    Y(*) - VECTOR OF INITIAL VALUES OF DEPENDENT VARIABLES
C    NEQN - NUMBER OF EQUATIONS TO BE INTEGRATED
C    H - NOMINAL STEP SIZE INDICATING DIRECTION OF INTEGRATION
C        AND MAXIMUM SIZE OF STEP.  MUST BE VARIABLE
C    EPS - LOCAL ERROR TOLERANCE PER STEP.  MUST BE VARIABLE
C    WT(*) - VECTOR OF NON-ZERO WEIGHTS FOR ERROR CRITERION
C    START - .TRUE.
C
C  STEP  REQUIRES THE L2 NORM OF THE VECTOR WITH COMPONENTS
C  LOCAL ERROR(L)/WT(L)  BE LESS THAN  EPS  FOR A SUCCESSFUL STEP.  THE
C  ARRAY  WT  ALLOWS THE USER TO SPECIFY AN ERROR TEST APPROPRIATE
C  FOR HIS PROBLEM.  FOR EXAMPLE,
C    WT(L) = 1.0  SPECIFIES ABSOLUTE ERROR,
C         = ABS(Y(L))  ERROR RELATIVE TO THE MOST RECENT VALUE OF THE
C              L-TH COMPONENT OF THE SOLUTION,
C         = ABS(YP(L))  ERROR RELATIVE TO THE MOST RECENT VALUE OF
C              THE L-TH COMPONENT OF THE DERIVATIVE,
C         = AMAX1(WT(L),ABS(Y(L)))  ERROR RELATIVE TO THE LARGEST
C              MAGNITUDE OF L-TH COMPONENT OBTAINED SO FAR,
C         = ABS(Y(L))*RELERR/EPS + ABSERR/EPS  SPECIFIES A MIXED
C              RELATIVE-ABSOLUTE TEST WHERE  RELERR  IS RELATIVE
C              ERROR,  ABSERR  IS ABSOLUTE ERROR AND  EPS =
C              AMAX1(RELERR,ABSERR) .
C
C   SUBSEQUENT CALLS -
C
C  SUBROUTINE  STEP  IS DESIGNED SO THAT ALL INFORMATION NEEDED TO
C  CONTINUE THE INTEGRATION, INCLUDING THE STEP SIZE  H  AND THE ORDER
C  K , IS RETURNED WITH EACH STEP.  WITH THE EXCEPTION OF THE STEP
C  SIZE, THE ERROR TOLERANCE, AND THE WEIGHTS, NONE OF THE PARAMETERS
C  SHOULD BE ALTERED.  THE ARRAY  WT  MUST BE UPDATED AFTER EACH STEP
C  TO MAINTAIN RELATIVE ERROR TESTS LIKE THOSE ABOVE.  NORMALLY THE
C  INTEGRATION IS CONTINUED JUST BEYOND THE DESIRED ENDPOINT AND THE
C  SOLUTION INTERPOLATED THERE WITH SUBROUTINE  INTRP .  IF IT IS
C  IMPOSSIBLE TO INTEGRATE BEYOND THE ENDPOINT, THE STEP SIZE MAY BE
C  REDUCED TO HIT THE ENDPOINT SINCE THE CODE WILL NOT TAKE A STEP
C  LARGER THAN THE  H  INPUT.  CHANGING THE DIRECTION OF INTEGRATION,
C  I.E., THE SIGN OF  H , REQUIRES THE USER SET  START = .TRUE. BEFORE
C  CALLING  STEP  AGAIN.  THIS IS THE ONLY SITUATION IN WHICH  START
C  SHOULD BE ALTERED.
C
C  OUTPUT FROM  STEP
C
C   SUCCESSFUL STEP -
C
```

```
C  THE SUBROUTINE RETURNS AFTER EACH SUCCESSFUL STEP WITH  START  AND
C  CRASH  SET .FALSE.. X  REPRESENTS THE INDEPENDENT VARIABLE
C  ADVANCED ONE STEP OF LENGTH  HOLD  FROM ITS VALUE ON INPUT AND  Y
C  THE SOLUTION VECTOR AT THE NEW VALUE OF  X . ALL OTHER PARAMETERS
C  REPRESENT INFORMATION CORRESPONDING TO THE NEW  X  NEEDED TO
C  CONTINUE THE INTEGRATION.
C
C    UNSUCCESSFUL STEP -
C
C  WHEN THE ERROR TOLERANCE IS TOO SMALL FOR THE MACHINE PRECISION,
C  THE SUBROUTINE RETURNS WITHOUT TAKING A STEP AND  CRASH = .TRUE..
C  AN APPROPRIATE STEP SIZE AND ERROR TOLERANCE FOR CONTINUING ARE
C  ESTIMATED AND ALL OTHER INFORMATION IS RESTORED AS UPON INPUT
C  BEFORE RETURNING. TO CONTINUE WITH THE LARGER TOLERANCE, THE USER
C  JUST CALLS THE CODE AGAIN. A RESTART IS NEITHER REQUIRED NOR
C  DESIRABLE.
C
      LOGICAL START,CRASH,PHASE1,NORND
      DIMENSION Y(60),WT(60),PHI(60,16),P(60),YP(60),PSI(20)
      DIMENSION ALPHA(20),BETA(20),SIG(20),W(20),V(20),G(20),
     1  GSTR(13),TWO(13),A(60),IA(60)
C   EXTERNAL F
C***********************************************************************
C*  THE ONLY MACHINE DEPENDENT CONSTANTS ARE BASED ON THE MACHINE
UNIT *
C*  ROUNDOFF ERROR  U  WHICH IS THE SMALLEST POSITIVE NUMBER SUCH THAT
*
C*  1.0+U .GT. 1.0  . THE USER MUST CALCULATE  U  AND INSERT         *
C*  TWOU=2.0*U  AND  FOURU=4.0*U  IN THE DATA STATEMENT BEFORE CALLING *
C*  THE CODE.  THE ROUTINE  MACHIN  CALCULATES  U .                 *
      DATA TWOU,FOURU/1.9E-6, 3.8E-6/
C***********************************************************************
      DATA TWO/2.0,4.0,8.0,16.0,32.0,64.0,128.0,256.0,512.0,1024.0,
     1  2048.0,4096.0,8192.0/
      DATA GSTR/.500,.083300,.041700,.026400,.018800,.014300,.011400,
     1 .936E-2,.789E-2,.679E-2,.592E-2,.524E-2,.468E-2/
      DATA G(1),G(2)/1.000,0.500/,SIG(1)/1.000/
C
C
C     ***    BEGIN BLOCK 0    ***
C  CHECK IF STEP SIZE OR ERROR TOLERANCE IS TOO SMALL FOR MACHINE
C  PRECISION.  IF FIRST STEP, INITIALIZE PHI ARRAY AND ESTIMATE A
C  STARTING STEP SIZE.
C            ***
C
C  IF STEP SIZE IS TOO SMALL, DETERMINE AN ACCEPTABLE ONE
C
      CRASH = .TRUE.
      IF( ABS(H) .GE. FOURU* ABS(X)) GO TO 5
      H = SIGN(FOURU* ABS(X),H)
      RETURN
    5 P5EPS = 0.500*EPS
C
C  IF ERROR TOLERANCE IS TOO SMALL, INCREASE IT TO AN ACCEPTABLE VALUE
C
      ROUND = 0.
      DO 10 L = 1,NEQN
   10   ROUND = ROUND + (Y(L)/WT(L))**2
      ROUND =TWOU* SQRT(ROUND)
      IF(P5EPS .GE. ROUND) GO TO 15
      EPS = 2.0*ROUND*(1.0 + FOURU)
      RETURN
   15 CRASH = .FALSE.
      IF(.NOT.START) GO TO 99
C
C  INITIALIZE.  COMPUTE APPROPRIATE STEP SIZE FOR FIRST STEP
C
      CALL F(X,Y,YP)
      SUM = 0.
      DO 20 L = 1,NEQN
        PHI(L,1) = YP(L)
        PHI(L,2) = 0.
   20   SUM = SUM + (YP(L)/WT(L))**2
      SUM = SQRT(SUM)
      ABSH = ABS(H)
      IF(EPS .LT. 16.0*SUM*H*H) ABSH = 0.2500* SQRT(EPS/SUM)
      HH=AMAX1(ABSH,FOURU* ABS(X))
      H= SIGN(HH,H)
      HOLD = 0.
      K = 1
      KOLD = 0
      START = .FALSE.
      PHASE1 = .TRUE.
      NORND = .TRUE.
      IF(P5EPS .GT. 100.0*ROUND) GO TO 99
      NORND = .FALSE.
      DO 25 L = 1,NEQN
   25   PHI(L,15) = 0.
   99 IFAIL = 0
C     ***    END BLOCK 0    ***
C
C     ***    BEGIN BLOCK 1    ***
C  COMPUTE COEFFICIENTS OF FORMULAS FOR THIS STEP.  AVOID COMPUTING
C  THOSE QUANTITIES NOT CHANGED WHEN STEP SIZE IS NOT CHANGED.
C              ***
C
  100 KP1 = K+1
      KP2 = K+2
      KM1 = K-1
      KM2 = K-2
C
C  NS IS THE NUMBER OF STEPS TAKEN WITH SIZE H, INCLUDING THE CURRENT
C  ONE.  WHEN K.LT.NS, NO COEFFICIENTS CHANGE
C
      IF(H .NE. HOLD) NS = 0
C NEXT LINE DIFFER IN BOOK & HERE. LINE COMMENTED & LINE FROM BOOK
INSER
C     IF(NS .LE. KOLD) NS=NS+1
      NS = MIN0(NS + 1,KOLD + 1)
      NSP1 = NS+1
      IF (K .LT. NS) GO TO 199
C
C  COMPUTE THOSE COMPONENTS OF ALPHA(*),BETA(*),PSI(*),SIG(*) WHICH
C  ARE CHANGED
C
      BETA(NS) = 1.
      REALNS = NS
      ALPHA(NS) =1.  /REALNS
   TEMP1 = H*REALNS
   SIG(NSP1) = 1.
   IF(K .LT. NSP1) GO TO 110
   DO 105 I = NSP1,K
    IM1 = I-1
    TEMP2 = PSI(IM1)
    PSI(IM1) = TEMP1
    BETA(I) = BETA(IM1)*PSI(IM1)/TEMP2
    TEMP1 = TEMP2 + H
    ALPHA(I) = H/TEMP1
    REALI = I
 105   SIG(I+1) = REALI*ALPHA(I)*SIG(I)
 110 PSI(K) = TEMP1
C
C  COMPUTE COEFFICIENTS G(*)
C
C  INITIALIZE V(*) AND SET W(*).  G(2) IS SET IN DATA STATEMENT
C
   IF(NS .GT. 1) GO TO 120
   DO 115 IQ = 1,K
    TEMP3 = IQ*(IQ+1)
    V(IQ) =1. /TEMP3
 115   W(IQ) = V(IQ)
   GO TO 140
C
C  IF ORDER WAS RAISED, UPDATE DIAGONAL PART OF V(*)
C
 120 IF(K .LE. KOLD) GO TO 130
   TEMP4 = K*KP1
   V(K) =1. /TEMP4
   NSM2 = NS-2
   IF(NSM2 .LT. 1) GO TO 130
   DO 125 J = 1,NSM2
    I = K-J
 125   V(I) = V(I) - ALPHA(J+1)*V(I+1)
C
C  UPDATE V(*) AND SET W(*)
C
 130 LIMIT1 = KP1 - NS
   TEMP5 = ALPHA(NS)
   DO 135 IQ = 1,LIMIT1
    V(IQ) = V(IQ) - TEMP5*V(IQ+1)
 135   W(IQ) = V(IQ)
   G(NSP1) = W(1)
C
C  COMPUTE THE G(*) IN THE WORK VECTOR W(*)
C
 140 NSP2 = NS + 2
   IF(KP1 .LT. NSP2) GO TO 199
   DO 150 I = NSP2,KP1
    LIMIT2 = KP2 - I
    TEMP6 = ALPHA(I-1)
    DO 145 IQ = 1,LIMIT2
 145    W(IQ) = W(IQ) - TEMP6*W(IQ+1)
 150   G(I) = W(1)
 199   CONTINUE
C     ***    END BLOCK 1    ***
C
C     ***    BEGIN BLOCK 2    ***
```

```
C  PREDICT A SOLUTION P(*), EVALUATE DERIVATIVES USING PREDICTED
C  SOLUTION, ESTIMATE LOCAL ERROR AT ORDER K AND ERRORS AT ORDERS K,
C  K-1, K-2 AS IF CONSTANT STEP SIZE WERE USED.
C             ***
C
C  CHANGE PHI TO PHI STAR
C
   IF(K .LT. NSP1) GO TO 215
   DO 210 I = NSP1,K
    TEMP1 = BETA(I)
    DO 205 L = 1,NEQN
 205    PHI(L,I) = TEMP1*PHI(L,I)
 210   CONTINUE
C
C  PREDICT SOLUTION AND DIFFERENCES
C
 215 DO 220 L = 1,NEQN
    PHI(L,KP2) = PHI(L,KP1)
    PHI(L,KP1) = 0.
 220   P(L) = 0.
   DO 230 J = 1,K
    I = KP1 - J
    IP1 = I+1
    TEMP2 = G(I)
    DO 225 L = 1,NEQN
     P(L) = P(L) + TEMP2*PHI(L,I)
 225    PHI(L,I) = PHI(L,I) + PHI(L,IP1)
 230   CONTINUE
   IF(NORND) GO TO 240
   DO 235 L = 1,NEQN
    TAU = H*P(L) - PHI(L,15)
    P(L) = Y(L) + TAU
 235   PHI(L,16) = (P(L) - Y(L)) - TAU
   GO TO 250
 240 DO 245 L = 1,NEQN
 245   P(L) = Y(L) + H*P(L)
 250 XOLD = X
   X = X + H
   ABSH = ABS(H)
   CALL F(X,P,YP)
C
C  ESTIMATE ERRORS AT ORDERS K,K-1,K-2
C
 251 ERKM2 = 0.
   ERKM1 = 0.
   ERK = 0.
   DO 265 L = 1,NEQN
    TEMP3 =1. /WT(L)
    TEMP4 = YP(L) - PHI(L,1)
    IF(KM2)265,260,255
 255   ERKM2 = ERKM2 + ((PHI(L,KM1)+TEMP4)*TEMP3)**2
 260   ERKM1 = ERKM1 + ((PHI(L,K)+TEMP4)*TEMP3)**2
 265   ERK = ERK + (TEMP4*TEMP3)**2
   IF(KM2)280,275,270
 270 ERKM2 = ABSH*SIG(KM1)*GSTR(KM2)* SQRT(ERKM2)
 275 ERKM1 = ABSH*SIG(K)*GSTR(KM1)* SQRT(ERKM1)
 280 TEMP5 = ABSH* SQRT(ERK)
   ERR = TEMP5*(G(K)-G(KP1))
   ERK = TEMP5*SIG(KP1)*GSTR(K)
   KNEW = K
C
C  TEST IF ORDER SHOULD BE LOWERED
C
   IF(KM2)299,290,285
 285 IF(AMAX1(ERKM1,ERKM2) .LE. ERK) KNEW = KM1
   GO TO 299
 290 IF(ERKM1 .LE. .500*ERK)KNEW = KM1
C
C  TEST IF STEP SUCCESSFUL
C
 299 IF(ERR .LE. EPS) GO TO 400
C    ***    END BLOCK 2    ***
C
C    ***    BEGIN BLOCK 3    ***
C  THE STEP IS UNSUCCESSFUL.  RESTORE  X, PHI(*,*), PSI(*) .
C  IF THIRD CONSECUTIVE FAILURE, SET ORDER TO ONE.  IF STEP FAILS MORE
C  THAN THREE TIMES, CONSIDER AN OPTIMAL STEP SIZE.  DOUBLE ERROR
C    TOLERANCE  AND  RETURN  IF  ESTIMATED  STEP  SIZE  IS  TOO  SMALL  FOR
MACHINE
C  PRECISION.
C           ***
C
C  RESTORE X, PHI(*,*) AND PSI(*)
C
   PHASE1 = .FALSE.
   X = XOLD
   DO 310 I = 1,K
    TEMP1 =1.  /BETA(I)
    IP1 = I+1
    DO 305 L = 1,NEQN
 305     PHI(L,I) = TEMP1*(PHI(L,I) - PHI(L,IP1))
 310   CONTINUE
   IF(K .LT. 2) GO TO 320
   DO 315 I = 2,K
 315   PSI(I-1) = PSI(I) - H
C
C  ON THIRD FAILURE, SET ORDER TO ONE.  THEREAFTER, USE OPTIMAL STEP
C  SIZE
C
 320 IFAIL = IFAIL + 1
   TEMP2 = 0.5
   IF(IFAIL - 3) 335,330,325
 325 IF(P5EPS .LT. .2500*ERK)TEMP2 = SQRT(P5EPS/ERK)
 330 KNEW = 1
 335 H = TEMP2*H
   K = KNEW
   IF( ABS(H) .GE. FOURU* ABS(X)) GO TO 340
   CRASH = .TRUE.
   H = SIGN(FOURU* ABS(X),H)
   EPS = EPS + EPS
   RETURN
 340 GO TO 100
C    ***    END BLOCK 3    ***
C
C    ***    BEGIN BLOCK 4    ***
C  THE STEP IS SUCCESSFUL.  CORRECT THE PREDICTED SOLUTION, EVALUATE
C  THE DERIVATIVES USING THE CORRECTED SOLUTION AND UPDATE THE
C  DIFFERENCES.  DETERMINE BEST ORDER AND STEP SIZE FOR NEXT STEP.
```

```
C           ***
 400 KOLD = K
    HOLD = H
C
C   CORRECT AND EVALUATE
C
    TEMP1 = H*G(KP1)
    IF(NORND) GO TO 410
    DO 405 L = 1,NEQN
     RHO = TEMP1*(YP(L) - PHI(L,1)) - PHI(L,16)
     Y(L) = P(L) + RHO
 405  PHI(L,15) = (Y(L) - P(L)) - RHO
    GO TO 420
 410 DO 415 L = 1,NEQN
 415  Y(L) = P(L) + TEMP1*(YP(L) - PHI(L,1))
 420 CALL F(X,Y,YP)
C
C   UPDATE DIFFERENCES FOR NEXT STEP
C
    DO 425 L = 1,NEQN
     PHI(L,KP1) = YP(L) - PHI(L,1)
 425  PHI(L,KP2) = PHI(L,KP1) - PHI(L,KP2)
    DO 435 I = 1,K
     DO 430 L = 1,NEQN
 430    PHI(L,I) = PHI(L,I) + PHI(L,KP1)
 435  CONTINUE
C
C   ESTIMATE ERROR AT ORDER K+1 UNLESS:
C    IN FIRST PHASE WHEN ALWAYS RAISE ORDER,
C    ALREADY DECIDED TO LOWER ORDER,
C    STEP SIZE NOT CONSTANT SO ESTIMATE UNRELIABLE
C
    ERKP1 = 0.
    IF(KNEW .EQ. KM1  .OR.  K .EQ. 12) PHASE1 = .FALSE.
    IF(PHASE1) GO TO 450
    IF(KNEW .EQ. KM1) GO TO 455
    IF(KP1 .GT. NS) GO TO 460
    DO 440 L = 1,NEQN
 440  ERKP1 = ERKP1 + (PHI(L,KP2)/WT(L))**2
    ERKP1 = ABSH*GSTR(KP1)* SQRT(ERKP1)
C
C   USING ESTIMATED ERROR AT ORDER K+1, DETERMINE APPROPRIATE ORDER
C   FOR NEXT STEP
C
    IF(K .GT. 1) GO TO 445
    IF(ERKP1 .GE. .500*ERK)GO TO 460
    GO TO 450
 445 IF(ERKM1 .LE. AMIN1(ERK,ERKP1)) GO TO 455
    IF(ERKP1 .GE. ERK  .OR.  K .EQ. 12) GO TO 460
C
C   HERE ERKP1 .LT. ERK .LT. AMAX1(ERKM1,ERKM2) ELSE ORDER WOULD HAVE
C   BEEN LOWERED IN BLOCK 2.  THUS ORDER IS TO BE RAISED
C
C   RAISE ORDER
C
 450 K = KP1
    ERK = ERKP1
    GO TO 460
C
```

```
C  LOWER ORDER
C
 455 K = KM1
    ERK = ERKM1
C
C  WITH NEW ORDER DETERMINE APPROPRIATE STEP SIZE FOR NEXT STEP
C
 460 HNEW = H + H
    IF(PHASE1) GO TO 465
    IF(P5EPS .GE. ERK*TWO(K+1)) GO TO 465
    HNEW = H
    IF(P5EPS .GE. ERK) GO TO 465
    TEMP2 = K+1
    R = (P5EPS/ERK)**(1.0/TEMP2)
    HNEW=ABSH*AMAX1(0.500,AMIN1(0.900,R))
    HNEW = SIGN(AMAX1(HNEW,FOURU* ABS(X)),H)
 465 H =  HNEW
    RETURN
C     ***    END BLOCK 4    ***
    END
C
C
C
    SUBROUTINE INTRP(X,Y,XOUT,YOUT,YPOUT,NEQN,KOLD,PHI,PSI)
    IMPLICIT REAL*8 (A-H,O-Z)
C
C  THE METHODS IN SUBROUTINE  STEP  APPROXIMATE THE SOLUTION NEAR  X
C  BY A POLYNOMIAL.  SUBROUTINE  INTRP  APPROXIMATES THE SOLUTION AT
C   XOUT   BY EVALUATING  THE  POLYNOMIAL  THERE.   INFORMATION  DEFINING
THIS
C  POLYNOMIAL IS PASSED FROM  STEP  SO  INTRP  CANNOT BE USED ALONE.
C
C  THIS CODE IS COMPLETELY EXPLAINED AND DOCUMENTED IN THE TEXT,
C  COMPUTER SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS:  THE INITIAL
C  VALUE PROBLEM  BY L. F. SHAMPINE AND M. K. GORDON.
C
C  INPUT TO INTRP --
C
C  THE USER PROVIDES STORAGE IN THE CALLING PROGRAM FOR THE ARRAYS IN
C  THE CALL LIST
    DIMENSION Y(1),YOUT(1),YPOUT(1),PHI(60,1),PSI(20)
C  AND DEFINES
C    XOUT -- POINT AT WHICH SOLUTION IS DESIRED.
C  THE REMAINING PARAMETERS ARE DEFINED IN  STEP  AND PASSED TO  INTRP
C  FROM THAT SUBROUTINE
C
C  OUTPUT FROM  INTRP --
C
C    YOUT(*) -- SOLUTION AT  XOUT
C    YPOUT(*) -- DERIVATIVE OF SOLUTION AT  XOUT
C  THE REMAINING PARAMETERS ARE RETURNED UNALTERED FROM THEIR INPUT
C  VALUES.  INTEGRATION WITH  STEP  MAY BE CONTINUED.
C
    DIMENSION G(13),W(13),RHO(13)
    DATA G(1)/1.0/,RHO(1)/1.0/
C
    HI = XOUT - X
    KI = KOLD + 1
    KIP1 = KI + 1
```

```
C
C  INITIALIZE W(*) FOR COMPUTING G(*)
C
    DO 5 I = 1,KI
     TEMP1 = I
  5  W(I) =1. /TEMP1
    TERM = 0.
C
C  COMPUTE G(*)
C
    DO 15 J = 2,KI
     JM1 = J - 1
     PSIJM1 = PSI(JM1)
     GAMMA = (HI + TERM)/PSIJM1
     ETA = HI/PSIJM1
     LIMIT1 = KIP1 - J
     DO 10 I = 1,LIMIT1
 10     W(I) = GAMMA*W(I) - ETA*W(I+1)
     G(J) = W(1)
     RHO(J) = GAMMA*RHO(JM1)
 15   TERM = PSIJM1
C
C  INTERPOLATE
C
    DO 20 L = 1,NEQN
     YPOUT(L) = 0.
 20   YOUT(L) = 0.
    DO 30 J = 1,KI
     I = KIP1 - J
     TEMP2 = G(I)
     TEMP3 = RHO(I)
     DO 25 L = 1,NEQN
      YOUT(L) = YOUT(L) + TEMP2*PHI(L,I)
 25     YPOUT(L) = YPOUT(L) + TEMP3*PHI(L,I)
 30   CONTINUE
    DO 35 L = 1,NEQN
 35   YOUT(L) = Y(L) + HI*YOUT(L)
    RETURN
    END
C
C
C ****************************************************
    SUBROUTINE PUD
C ****************************************************
    IMPLICIT REAL*8 (A-H,O-Z)
C
    COMMON/COOR1/ NBC1,NBC2,NBC,N
    COMMON/CNST1/ CE(15,30)
    COMMON/CNST2/ D2TH(25,30)
    COMMON/CNST3/ M,NM
    COMMON/PRIN/ IPR(40)
    COMMON/FL/ IFL(40)
    DIMENSION B(30,15),HB(30,15),H(30,30),HH(30,30)
    DIMENSION U(30),W(30),X(30)
    COMMON/NEWA/ AA(15,30)
    DATA W/30*0./
C
    DO 201 I=1,N
    W(I)=0.
```

```
      DO 201 J=1,M
  201 B(I,J)=CE(J,I)
C
      IF(IFL(12).EQ.0) GO TO 50
      DO 51 I=1,N
      DO 51 J=1,M
   51 B(I,J)=AA(J,I)
   50 CONTINUE
C
      IPRI=IPR(16)
      IF(IPR(17).EQ.1) IPRI=2
      IF(IPRI.LT.1) GO TO 18
      WRITE(6,11)
      DO 21 I=1,N
   21 WRITE(6,12) I,(B(I,J),J=1,M)
   18 CONTINUE
      DO 40 I=1,N
      DO 41 J=1,M
   41 HB(I,J)=B(I,J)
      DO 42 J=1,N
   42 HH(I,J)=0.
   40 HH(I,I)=1.
C
      DO 80 K=1,M
      DO 23 I=1,N
   23 X(I)=HB(I,K)
      IF(K.EQ.1) GO TO 44
      K1=K-1
      DO 24 I=1,K1
   24 W(I)=X(I)
   44 CONTINUE
      TOT=0.
      DO 25 I=K,N
   25 TOT=TOT+X(I)*X(I)
C  IF K TH ELEMENT OF K TH W IS ZERO, CONSTRAINTS NOT LINEARLY INDEP.
      IF(TOT.GE.0.1E-10) GO TO 202
      WRITE(6,203)
  203 FORMAT(/' CONSTR MATRIX LESS THAN FULL RANK')
      STOP
  202 CONTINUE
      W(K)=TOT**0.5
      TOT=0.
      DO 26 I=1,N
      U(I)=W(I)-X(I)
   26 TOT=TOT+U(I)*U(I)
      TOT1=TOT**0.5
C  X AND W MAY BE SAME. IN THIS CASE H=IDENTITY MATRIX
      IF(TOT1.EQ.0.) GO TO 28
      DO 27 I=1,N
   27 U(I)=U(I)/TOT1
   28 CONTINUE
C
      DO 31 I=1,N
      DO 30 J=1,N
   30 H(I,J)=-2.*U(I)*U(J)
   31 H(I,I)=H(I,I)+1.
C
      CALL AMULT(H,HB,N,M,N)
      CALL AMULT(H,HH,N,N,N)
```

```
C
      IF(IPRI.LE.1) GO TO 80
      WRITE(6,81) K
      WRITE(6,13)
      WRITE(6,14) (W(I),I=1,N)
      WRITE(6,15)
      WRITE(6,14) (U(I),I=1,N)
      WRITE(6,32)
      DO 39 I=1,N
   39 WRITE(6,12) I,(H(I,J),J=1,N)
      WRITE(6,33)
      DO 34 I=1,N
   34 WRITE(6,12) I,(HB(I,J),J=1,M)
      IF(K.EQ.1) GO TO 80
      WRITE(6,35)
      DO 36 I=1,N
   36 WRITE(6,12) I,(HH(I,J),J=1,N)
C
   80 CONTINUE
C
      M1=M+1
      DO 43 I=M1,N
      I1=I-M1+1
      DO 43 J=1,N
      D2TH(I1,J)=HH(I,J)
      IF(ABS(HH(I,J)).LT.1.E-6) D2TH(I1,J)=0.
   43 CONTINUE
C
C
   60 IF(IPRI.EQ.0) RETURN
      WRITE(6,37)
      DO 38 I=1,N
   38 WRITE(6,12) I,(D2TH(J,I),J=1,NM)
C
      RETURN
   11 FORMAT(//10X,'THE CONSTRAINT MATRIX, B')
   12 FORMAT(1X,'R',I2,':',8E12.4/5X,8F10.4)
   81 FORMAT(/5X,'*** TRANSFORMATION STEP',I3)
   13 FORMAT(/10X,'TRANSPOSE OF W')
   14 FORMAT(8(10X,8F12.4/))
   15 FORMAT(/10X,'TRANSPOSE OF U')
   32 FORMAT(/10X,'TRANSFORMATION MATRIX, H')
   33 FORMAT(/10X,'TRANSFORMED CONSTRAINT MATRIX, HB')
   35 FORMAT(/10X,'PRODUCT OF TRANSFORMATION MATRICES, HH')
   37 FORMAT(/10X,'TRANSFORMATION MATRIX FOR VARIABLES, HTD2')
      END
C
      SUBROUTINE AMULT(A,B,NA,MB,MA)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION A(30,1),B(30,1),C(30,30)
C
      DO 10 L=1,MB
      DO 10 I=1,NA
      TOT=0.
      DO 11 J=1,MA
   11 TOT=TOT+A(I,J)*B(J,L)
   10 C(I,L)=TOT
      DO 20 I=1,MA
      DO 20 J=1,MB
```

```
  20 B(I,J)=C(I,J)
     RETURN
     END
C
C
C ***********************************************
     SUBROUTINE CONSTR(Y,YP,T)
C ***********************************************
     IMPLICIT REAL*8 (A-H,O-Z)
C
     DIMENSION Y(60),YP(60)
     COMMON/CNST4/ NLOOP,NCADD
     COMMON/CNSTA/ NDIRA(4,3),NNANG(4),NANGV
     COMMON/CNST1/ B(15,30)
     COMMON/CNST7/ BDOT(15,30),AAIBD(30,30)
     COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
     COMMON/CNST3/ NCST,NMNC
     COMMON/PRIN/ IPR(40)
C
     COMMON/DIFF1/ TDEL
     DIMENSION PACC(10),DACC(10),BB(10)
C
     DIMENSION Q(30),S(30),XB(15,30),XBDOT(15,30)
C
     DO 10 I=1,NCST
     DO 10 J=1,NTOT
     XB(I,J)=0.
     XBDOT(I,J)=0.
     B(I,J)=0.
  10 BDOT(I,J)=0.
     NCST1=NCST-NCADD
C
     IF ((NLOOP+NANGV).GT.0) CALL CLOOP
C
C  INSERT THE LAST NCADD ROWS FOR B AND BDOT HERE
     IF (NCADD.EQ.0) GO TO 11
     DO 30 I=1,NTOT
     Q(I)=Y(I+NTOT)
  30 S(I)=Y(I)
     CALL CONEQ (Q,S,T,XB,XBDOT)
     DO 31 I=1,NCADD
     DO 31 J=1,NTOT
     B(NCST1+I,J)=XB(NCST1+I,J)
  31 BDOT (NCST1+I,J)=XBDOT(NCST1+I,J)
C
  11 CONTINUE
C
     WRITE(6,101)
 101 FORMAT(' CONSTRAINT MATRIX')
     DO 32 I=1,NCST
  32 WRITE(6,100) (B(I,J),J=1,NTOT)
 100 FORMAT(1X,6E13.5)
C
     IF(NMNC.GT.0) CALL PUD
     RETURN
     END
C
C
C ********************************************************
```

```
      SUBROUTINE CLOOP
C ********************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
C THIS SUBROUTINE GENERATES ROWS OF B AND BDOT FOR CLOSED LOOPS AND
C  AND PRESCRIBED MOTIONS
C
C OUTPUTS
      COMMON/CNST1/ B(15,30)
      COMMON/CNST7/ BDOT(15,30),AAIBD(30,30)
C INPUTS
      COMMON/CNST3/ NCST,NMNC
      COMMON/CNST4/ NLOOP,NCADD
      COMMON/CNST5/ LOOP(5,2),REND(5,2,3),NDIR(5,3),NQCON(5,2)
      COMMON/INP/ N
      COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
      DIMENSION BVEC(2,3,30),BVECD(2,3,30)
      COMMON/PANG1/ WNIU(14,3,21),TNIU(14,3,21)
      COMMON/PANG1D/ WNIUD(14,3,21),TNIUD(14,3,21)
      COMMON/PVEL1/ GA(14,3,21),GDRIG(14,3,21,3)
      COMMON/PVEL3/ GAD(14,3,21),GDRIGD(14,3,21,3)
      INTEGER PHIKCN
      COMMON/CNSTA/ NDIRA(4,3),NNANG(4),NANGV
C
      NCOUNT=0
      IF(NLOOP.EQ.0) GO TO 50
      DO 10 ILOOP=1,NLOOP
      DO 11 ISIDE=1,2
      K=LOOP(ILOOP,ISIDE)
      IF(K.NE.0) GO TO 13
      DO 15 L=1,NTOT
      DO 15 M=1,3
      BVEC(ISIDE,M,L)=0.
   15 BVECD(ISIDE,M,L)=0.
      GO TO 11
   13 CONTINUE
      DO 12 M=1,3
   14 IF(NBC1.EQ.0) GO TO 19
      DO 24 L=1,NBC1
      SUM1=0.
      SUM2=0.
      DO 25 M1=1,3
      SUM1=SUM1+GDRIG(K,M,L,M1)*REND(ILOOP,ISIDE,M1)
   25 SUM2=SUM2+GDRIGD(K,M,L,M1)*REND(ILOOP,ISIDE,M1)
      BVEC(ISIDE,M,L)=GA(K,M,L)+SUM1
   24 BVECD(ISIDE,M,L)=GAD(K,M,L)+SUM2
   19 IF(NBC2.EQ.0) GO TO 12
      DO 18 L=1,NBC2
      BVEC(ISIDE,M,L+NBC1)=TNIU(K,M,L)
   18 BVECD(ISIDE,M,L+NBC1)=TNIUD(K,M,L)
   12 CONTINUE
   11 CONTINUE
C
      DO 30 M=1,3
      IF(NDIR(ILOOP,M).EQ.0) GO TO 30
      NCOUNT=NCOUNT+1
      DO 31 L=1,NTOT
      B(NCOUNT,L)=BVEC(1,M,L)-BVEC(2,M,L)
   31 BDOT(NCOUNT,L)=BVECD(1,M,L)-BVECD(2,M,L)
   30 CONTINUE
C
   10 CONTINUE
   50 IF(NANGV.EQ.0) GO TO 52
      DO 53 IANG=1,NANGV
      K=NNANG(IANG)
      DO 54 M=1,3
      NCOUNT=NCOUNT+1
      IF(NBC1.EQ.0) GO TO 55
      DO 56 L=1,NBC1
      B(NCOUNT,L)=WNIU(K,M,L)
   56 BDOT(NCOUNT,L)=WNIUD(K,M,L)
   54 CONTINUE
   53 CONTINUE
   52 CONTINUE
      IF(NCOUNT.EQ.(NCST-NCADD)) GO TO 40
      WRITE(6,100) NCOUNT
      STOP
  100 FORMAT(/10X,'ERROR RETURN:',I3,' CONSTRAINTS')
   40 RETURN
      END
C
C
C ******************************************************
      SUBROUTINE MASSQ(Y,YP)
C ******************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
C THIS SUBROUTINE COMPUTES THE MASS MATRIX GM, AND THE QUADRATIC
C VELOCITY VECTOR G.
C
C OUTPUTS
      DIMENSION AB(30,60),GG(30),GROSKR(30,30)
      COMMON/MASS1/ GM(30,30)
      COMMON/MASS2/ G(30)
      COMMON/MASS3/ AAI(30,30)
      COMMON/STIF1/ AAIKF(30,30)
      COMMON/CNST7/ BDOT(15,30),AAIBD(30,30)
      COMMON/FOR2/ EFOR(30)
      COMMON/SD3/ AAID(30,30)
      DIMENSION BDOTY(15)
C INPUTS
      COMMON/MAT1/ SSB(2,3,3,33,33),SB(4,3,33)
      COMMON/PANG1/ WNIU(14,3,21),TNIU(14,3,21)
      COMMON/PANG1D/ WNIUD(14,3,21),TNIUD(14,3,21)
      COMMON/D23/ NCOPS,NAPS,NTPS
      COMMON/MAT3/ RMASS(14)
      COMMON/FNUM2/ NELEM(4),NNODE(4)
      DIMENSION Y(60),YP(60),VI(3),VJ(3,3,12),VJR(3,3),VG(3,21,3,33)
      DIMENSION VB(3,20,3,33),VGR(3,21,3),VBR(3,20,3)
      COMMON/INP/ N,IRF(14)
      COMMON/COOR1/ NBC1,NBC2,NBC,NMODE,NTOT
      COMMON/MAT2/ R(14,3),RR(14,3,3)
      COMMON/PRIN/ IPR(40)
      COMMON/FL/ IFL(40)
      COMMON/CNST1/ B(15,30)
      COMMON/CNST2/ D2TH(25,30)
      COMMON/CNST3/ NCST,NMNC
      COMMON/STIF/ GROSK(30,30)
      COMMON/PVEL1/ GA(14,3,21),GD(2,3,21,3,33),GDRIG(14,3,21,3)
      COMMON/PVEL2/ BA(14,3,20),BD(2,3,20,3,33),BDRIG(14,3,20,3)
      COMMON/PVEL3/ GAD(14,3,21),GDD(2,3,21,3,33),GDRIGD(14,3,21,3)
      COMMON/PVEL4/ BAD(14,3,20),BDD(2,3,20,3,33),BDRIGD(14,3,20,3)
      COMMON/WEIG/ WGT(14,3)
      COMMON/SD2/ GROSD(30,30)
      DIMENSION GROSDR(30,30)
      COMMON/GF1/ GENFI(3,30),GENFE(3,30),GENFC(30),RHS(30)
      COMMON/DIFF1/ TDEL
      COMMON/DIFF2/ YBEF(60)
      COMMON/HYBR1/ AL(5),ALD(5),ALB(5),ALDB(5)
      COMMON/HYBR2/IHYBR(5),NHYBR
      COMMON/GEAR/IGEAR,NGEAR,MIND(10,2),FACT(10)
      COMMON/IMPC2/ DIRIMP(30)
      COMMON/IMPC3/ IMP01,RESTI
      DIMENSION ZL(30,60)
C
      DO 8 I1=1,NTOT
      G(I1)=0.
      EFOR(I1)=0.
      DO 8 I2=1,NTOT
    8 GM(I1,I2)=0.
C
      DO 10 K=1,N
C
C  COMPUTE  THE FIRST PART OF GM AND G
C
      IF(NBC1.EQ.0) GO TO 25
      DO 17 L=1,NBC1
      DO 11 LR=1,NBC1
      SUM=0.
      DO 12 M=1,3
   12 SUM=SUM+GA(K,M,L)*GA(K,M,LR)
   11 GM(L,LR)=SUM*RMASS(K)+GM(L,LR)
      IF(NBC2.EQ.0) GO TO 17
      DO 13 LR=1,NBC2
      SUM=0.
      DO 14 M=1,3
   14 SUM=SUM+GA(K,M,L)*TNIU(K,M,LR)
   13 GM(L,LR+NBC1)=SUM*RMASS(K)+GM(L,LR+NBC1)
   17 CONTINUE
   25 IF(NBC2.EQ.0) GO TO 27
      DO 22 L=1,NBC2
      DO 18 LR=1,NBC2
      SUM=0.
      DO 19 M=1,3
   19 SUM=SUM+TNIU(K,M,L)*TNIU(K,M,LR)
   18 GM(L+NBC1,LR+NBC1)=GM(L+NBC1,LR+NBC1)+SUM*RMASS(K)
   27 CONTINUE
C
      DO 30 M=1,3
      VI(M)=0.
      IF(NBC1.EQ.0) GO TO 34
      DO 31 L=1,NBC1
   31 VI(M)=VI(M)+GAD(K,M,L)*Y(L)
   34 IF(NBC2.EQ.0) GO TO 30
      DO 32 L=1,NBC2
   32 VI(M)=VI(M)+TNIUD(K,M,L)*Y(NBC1+L)
   30 CONTINUE
```

```
      DO 43 M=1,3
      IF(NBC1.EQ.0) GO TO 44
      DO 40 L=1,NBC1
      EFOR(L)=EFOR(L)+GA(K,M,L)*WGT(K,M)*RMASS(K)
   40 G(L)=G(L)+GA(K,M,L)*VI(M)*RMASS(K)
   44 IF(NBC2.EQ.0) GO TO 43
      DO 41 L=1,NBC2
      EFOR(L+NBC1)=EFOR(L+NBC1)+TNIU(K,M,L)*WGT(K,M)*RMASS(K)
   41 G(L+NBC1)=G(L+NBC1)+TNIU(K,M,L)*VI(M)*RMASS(K)
   43 CONTINUE
C
C  COMPUTE THE REST OF GM AND G
C
      IF(NBC1.EQ.0) GO TO 110
      DO 102 M=1,3
      DO 102 L=1,NBC1
      DO 102 IH=1,3
      SUM=0.
      DO 103 IQ=1,3
  103 SUM=SUM+GDRIG(K,M,L,IQ)*RR(K,IQ,IH)
      VGR(M,L,IH)=SUM
      DO 104 LR=1,NBC1
  104 GM(L,LR)=GM(L,LR)+VGR(M,L,IH)*GDRIG(K,M,LR,IH)
  110 CONTINUE
      IF(NBC1.EQ.0) GO TO 118
      DO 121 L=1,NBC1
      DO 122 LR=1,NBC1
      DO 122 M=1,3
      DO 122 IQ=1,3
  122 GM(L,LR)=GM(L,LR)+(GA(K,M,L)*GDRIG(K,M,LR,IQ)+GA(K,M,LR)*
     1    GDRIG(K,M,L,IQ))*R(K,IQ)
      IF(NBC2.EQ.0) GO TO 119
      DO 123 LR=1,NBC2
      DO 123 M=1,3
      DO 123 IQ=1,3
  123 GM(L,LR+NBC1)=GM(L,LR+NBC1)+GDRIG(K,M,L,IQ)*TNIU(K,M,LR)*
     1    R(K,IQ)
  119 IF(NMODE.EQ.0) GO TO 121
      DO 124 LS=1,NMODE
      DO 124 M=1,3
      DO 124 IQ=1,3
  124 GM(L,LS+NBC)=GM(L,LS+NBC)+(GA(K,M,L)*BDRIG(K,M,LS,IQ)+
     1    BA(K,M,LS)*GDRIG(K,M,L,IQ))*R(K,IQ)
  121 CONTINUE
  118 CONTINUE
C
      DO 130 M=1,3
      DO 130 IQ=1,3
      VJR(M,IQ)=0.
      IF(NBC1.EQ.0) GO TO 130
      DO 132 L=1,NBC1
  132 VJR(M,IQ)=VJR(M,IQ)+GDRIGD(K,M,L,IQ)*Y(L)
  130 CONTINUE
      DO 134 M=1,3
      DO 134 IQ=1,3
      IF(NBC1.EQ.0) GO TO 135
      DO 136 L=1,NBC1
      EFOR(L)=EFOR(L)+GDRIG(K,M,L,IQ)*R(K,IQ)*WGT(K,M)
  136 G(L)=G(L)+(GA(K,M,L)*VJR(M,IQ)+GDRIG(K,M,L,IQ)*VI(M))*
```

```
   1    R(K,IQ) + VGR(M,L,IQ)*VJR(M,IQ)
 135 IF(NBC2.EQ.0) GO TO 134
     DO 138 L=1,NBC2
 138 G(L+NBC1)=G(L+NBC1)+TNIU(K,M,L)*VJR(M,IQ)*R(K,IQ)
 134 CONTINUE
  10 CONTINUE
C
C  FORMING THE SYMMETRIC PART OF GM
C
     IF (NBC1.EQ.0) GO TO 922
     DO 924 L=1,NBC1
     IF (NBC2.EQ.0) GO TO 924
     DO 201 LR=1,NBC2
 201 GM(NBC1+LR,L)=GM(L,NBC1+LR)
 924 CONTINUE
 922 CONTINUE
C
C **** MULTIPLY ROTOR INERTIAS BY GEAR RATIO SQUARE (FOR DIAGONAL
TERMS),
C    OR BY GEAR RATIO OR ZERO (FOR THE COUPLING TERMS)
     IF(IGEAR.EQ.0) GO TO 950
     DO 951 I=1,NGEAR
     GM(MIND(I,1),MIND(I,2))=GM(MIND(I,1),MIND(I,2))*FACT(I)
     IF(MIND(I,1).EQ.MIND(I,2)) GO TO 951
 951 CONTINUE
 950 CONTINUE
C **** END ROTOR
C
C  PRINTING GM AND G
     IF(IPR(14).EQ.0) GO TO 215
     WRITE(6,212)
     DO 210 I=1,NTOT
 210 WRITE(6,211) I,(GM(I,J),J=1,NTOT)
     WRITE(6,213)
     WRITE(6,214) (G(I),I=1,NTOT)
 213 FORMAT(/' QUADRATIC VELOCITY VECTOR')
 212 FORMAT(/' MASS MATRIX')
 211 FORMAT(' R',I2,(T7,5D14.7))
 214 FORMAT(2X,10E12.5)
 215 CONTINUE
C
C  GENERALIZED FORCES
     DO 360 I=1,3
     DO 360 J=1,NTOT
     GENFI(I,J)=0.
 360 GENFE(I,J)=0.
     DO 350 I=1,NTOT
     SUM=0.
     DO 351 J=1,NTOT
 351 SUM=SUM+GM(I,J)*YP(J)
     GENFI(2,I)=SUM
     GENFI(3,I)=G(I)
 350 GENFI(1,I)=SUM+G(I)
     DO 352 I=1,NTOT
     SUM=0.
     DO 353 J=1,NTOT
C
 353 SUM=SUM+GROSK(I,J)*YBEF(NTOT+J)+GROSD(I,J)*Y(J)
C
```

```
      GENFE(2,I)=-SUM
  352 GENFE(1,I)=-SUM
C  END GENERALIZED FORCES
C
C  BACKWARD DIFF: GM=GM+STIF, G=G+MY
      IF(IFL(30).EQ.0) GO TO 410
      DO 411 I=1,NTOT
      SUM=0.
      DO 412 J=1,NTOT
  412 SUM=SUM+GM(I,J)*YBEF(J)
  411 G(I)=G(I)-SUM/TDEL
      WRITE(6,213)
      WRITE(6,214) (G(I),I=1,NTOT)
C**
      IF(NCST.EQ.NHYBR) GO TO 432
      DO 430 I=1,NTOT
      SUM=0.
      DO 431 J=1,NHYBR
  431 SUM=SUM+AL(J)*B(IHYBR(J),I)
  430 G(I)=G(I)+SUM
      WRITE(6,213)
      WRITE(6,214) (G(I),I=1,NTOT)
  432 CONTINUE
      DO 413 I=1,NTOT
      DO 413 J=1,NTOT
  413 GM(I,J)=GM(I,J)/TDEL+GROSK(I,J)*TDEL
C
      IF(IPR(14).EQ.0) GO TO 421
      WRITE(6,212)
      DO 420 I=1,NTOT
  420 WRITE(6,211) I,(GM(I,J),J=1,NTOT)
      WRITE(6,213)
      WRITE(6,214) (G(I),I=1,NTOT)
  421 CONTINUE
  410 CONTINUE
      DO 438 I=1,NTOT
  438 RHS(I)=-G(I)
C**
C  END BACKWARD DIFFERENCE
C
C  IF NCST>0 PREMULTIPLY GM BY D2TH
C  FORM MATRIX AB FOR INVERSION OF MASS MATRIX
C  PREMULTIPLY G BY D2TH TO FORM GG
C  PREMULTIPLY GROSK BY D2TH TO FORM GROSKR
      IF(NCST.EQ.0) GO TO 220
      DO 223 I=1,NMNC
      DO 221 J=1,NTOT
      SUM=0.
      DO 222 L=1,NTOT
  222 SUM=SUM+D2TH(I,L)*GM(L,J)
  221 AB(I,J)=SUM
      SUM=0.
      DO 224 L=1,NTOT
  224 SUM=SUM+D2TH(I,L)*G(L)
  223 GG(I)=SUM
C
C  BACKWARD DIFFERENCE
      IF(IFL(30).EQ.0) GO TO 414
C     IF(VCON.EQ.1) GO TO 414
```

```
      DO 415 I=1,NCST
      DO 415 J=1,NTOT
  415 AB(I+NMNC,J)=B(I,J)/TDEL+BDOT(I,J)
      GO TO 416
  414 CONTINUE
C  END BACKWARD DIFFERENCE
C
      DO 225 I=1,NCST
      DO 225 J=1,NTOT
  225 AB(I+NMNC,J)=B(I,J)
  416 CONTINUE
      DO 240 I=1,NMNC
      DO 240 J=1,NTOT
      SUM=0.
      SUM1=0.
      DO 241 L=1,NTOT
      SUM1=SUM1+D2TH(I,L)*GROSD(L,J)
  241 SUM=SUM+D2TH(I,L)*GROSK(L,J)
      GROSDR(I,J)=SUM1
  240 GROSKR(I,J)=SUM
      GO TO 230
C
C  IF NCST=0, FORM MATRIX AB FOR INVERSION OF GM, SET GG=G
  220 DO 231 I=1,NTOT
      GG(I)=G(I)
      DO 231 J=1,NTOT
  231 AB(I,J)=GM(I,J)
      DO 244 J=1,NTOT
      DO 244 I=1,NTOT
      GROSDR(I,J)=GROSD(I,J)
  244 GROSKR(I,J)=GROSK(I,J)
C
  230 CONTINUE
      DO 232 I=1,NTOT
      DO 233 J=1,NTOT
  233 AB(I,J+NTOT)=0.
  232 AB(I,I+NTOT)=1.
C
C  INVERT THE GENERALIZED MASS MATRIX GM : AAI
      NP=NTOT*2
C
      IF(IPR(18).EQ.0) GO TO 305
      WRITE(6,300)
      DO 302 I=1,NTOT
  302 WRITE(6,211) I,(AB(I,J),J=1,NP)
  305 CONTINUE
  300 FORMAT(/' GEN. MASS MATRIX, CONSTRAINED')
C
      CALL ELIM(AB,NTOT,NP)
      DO 234 I=1,NTOT
      DO 234 J=1,NTOT
  234 AAI(I,J)=AB(I,J+NTOT)
C  sF
      IF(IPR(18).EQ.0) GO TO 306
      WRITE(6,303)
      DO 304 I=1,NTOT
  304 WRITE(6,211) I,(AAI(I,J),J=1,NTOT)
  306 CONTINUE
  303 FORMAT(/' MASS MATRIX INVERSE')
```

```
C
C MULTIPLY AAI BY GG AND STORE BACK INTO G (THE CONTENTS OF G ARE NOW
C  THE VALUES OF THE ACCELERATIONS DUE TO THE QUAD VEL GEN FORCES).
   DO 235 L=1,NTOT
   SUM=0.
   DO 236 J=1,NMNC
236 SUM=SUM+AAI(L,J)*GG(J)
235 G(L)=SUM
   IF(IPR(14).EQ.0) GO TO 216
   WRITE(6,217)
217 FORMAT(/'ACC DUE TO Q')
   WRITE(6,214) (G(I),I=1,NTOT)
216 CONTINUE
C
C PREMULTIPLY THE STIFFNESS AND DAMPING MATRICES BY AAI
   DO 242 L=1,NTOT
   DO 242 I=1,NTOT
   SUM=0.
   SUM1=0.
   DO 243 J=1,NMNC
   SUM1=SUM1+AAI(L,J)*GROSDR(J,I)
243 SUM=SUM+AAI(L,J)*GROSKR(J,I)
   AAID(L,I)=SUM1
242 AAIKF(L,I)=SUM
C MULTIPLY -AAI BY BDOT TO FORM AAIBD
   IF(NCST.EQ.0) RETURN
C BACKWARD EULER
   IF(IFL(30).EQ.0) GO TO 417
   DO 418 I=1,NTOT
   DO 418 J=1,NTOT
   SUM=0.
   DO 419 L=1,NCST
419 SUM=SUM+AAI(I,NMNC+L)*B(L,J)
418 AAIBD(I,J)=SUM/TDEL
   RETURN
417 CONTINUE
C END BACKWARD EULER
   DO 251 I=1,NTOT
   DO 251 J=1,NTOT
   SUM=0.
   DO 252 L=1,NCST
252 SUM=SUM+AAI(I,NMNC+L)*BDOT(L,J)
251 AAIBD(I,J)=-SUM
   RETURN
   END
C
C ***********************************************
   SUBROUTINE FORC(T)
C ***********************************************
   IMPLICIT REAL*8 (A-H,O-Z)
C
C OUTPUTS
   COMMON/EXTF/ GF(30)
   DIMENSION A1(30)
C INPUTS
   COMMON/FOR1/ FMJOIN(30)
   COMMON/FOR2/ EFOR(30)
   COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
   COMMON/CNST2/ D2TH(25,30)
```

```
      COMMON/CNST3/ NCST,NMNC
      COMMON/FL/ IFL(40)
      COMMON/MASS3/ AAI(30,30)
      COMMON/PRIN/ IPR(40)
      COMMON/GF1/ GENFI(3,30),GENFE(3,30),GENFC(30),RHS(30)
C
C  EFOR : GENERALIZED FORCES DUE TO WEIGHT
C  FMJOIN : JOINT MOMENTS AND JOINT FORCES
C  FMJOIN COMES FROM MAIN, EFOR COMES FROM MASSQ.
C  MAKE CHANGES ON FMJOIN HERE:
C     IF(T.GE.0.02) FMJOIN(1)=0.
C
      DO 10 I=1,NTOT
   10 GF(I)=FMJOIN(I)+EFOR(I)
      IF(IPR(14).GT.0) WRITE(6,100) (GF(I),I=1,NTOT)
  100 FORMAT(/' GENERALIZED EXT FORCES'/(2X,8E12.4))
C
C  GENERALIZED FORCES
      DO 11 I=1,NTOT
      GENFE(3,I)=GF(I)
   11 GENFE(1,I)=GENFE(1,I)+GF(I)
C  END GENERALIZED FORCES
C
      IF(NCST.EQ.0) GO TO 22
      DO 20 I=1,NMNC
      SUM=0.
      DO 21 L=1,NTOT
   21 SUM=SUM+D2TH(I,L)*GF(L)
   20 A1(I)=SUM
      GO TO 23
   22 DO 24 I=1,NTOT
   24 A1(I)=GF(I)
   23 DO 25 L=1,NTOT
      SUM=0.
      DO 26 J=1,NMNC
   26 SUM=SUM+AAI(L,J)*A1(J)
   25 GF(L)=SUM
      IF(IPR(14).GT.0) WRITE(6,101) (GF(I),I=1,NTOT)
  101 FORMAT(/' ACC DUE TO EXTERNAL FORCES'/(2X,8E12.4))
      RETURN
      END
C
C
C  ****************************************************
      SUBROUTINE F(T,Y,YP)
C  ****************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION Y(60),YP(60)
      COMMON/COOR1/ NBC1,NBC2,NBC,NTOT
      COMMON/MASS3/ AAI(30,30)
      COMMON/MASS2/ G(30)
      COMMON/CNST7/ BDOT(15,30),AAIBD(30,30)
      COMMON/CONG1/GDOT(15)
      COMMON/CNST3/ NCST,NMNC
      DIMENSION TCD(15)
      COMMON/FL/ IFL(40)
      COMMON/STIF1/ AAIKF(30,30)
      COMMON/EXTF/ GF(30)
      COMMON/SD3/ AAID(30,30)
      COMMON/DIFF1/ TDEL
      COMMON/DIFF2/ YBEF(60)
C
      DIMENSION Q(30),S(30)
C
      DO 10 L=1,NTOT
      YP(L)=0.
C  FOR EULER ANGLES INSERT TRANSFORMATION HERE
   10 YP(L+NTOT)=Y(L)
      DO 11 L=1,NTOT
   11 YP(L)=-G(L)
C
C  GENERALIZED FORCES DUE TO WEIGHT
      IF((IFL(2)+IFL(5)+IFL(10)).EQ.0) GO TO 21
      DO 19 L=1,NTOT
   19 YP(L)=YP(L)+GF(L)
C
C  GENERALIZED FORCES DUE TO STIFFNESS AND DAMPING
   21 IF(IFL(13).EQ.0.AND.NMODE.EQ.0) GO TO 22
      IF(IFL(30).EQ.0) GO TO 39
      DO 40 L=1,NTOT
      SUM=0.
      DO 41 J=1,NTOT
   41 SUM=SUM-AAIKF(L,J)*YBEF(NTOT+J)-AAID(L,J)*YBEF(J)
   40 YP(L)=YP(L)+SUM
      GO TO 22
   39 CONTINUE
      DO 17 L=1,NTOT
      SUM=0.
      DO 18 J=1,NTOT
   18 SUM=SUM-AAIKF(L,J)*Y(NTOT+J)-AAID(L,J)*Y(J)
   17 YP(L)=YP(L)+SUM
C
C  CONTRIBUTION DUE TO DERIVATIVES OF CONSTRAINT EQUATIONS
   22 IF(NCST.EQ.0) GO TO 23
      IF(IFL(30).EQ.0) GO TO 35
      DO 36 L=1,NTOT
      SUM=0.
      DO 37 J=1,NTOT
   37 SUM=SUM+AAIBD(L,J)*YBEF(J)
   36 YP(L)=YP(L)+SUM
      GO TO 38
   35 CONTINUE
      DO 12 L=1,NTOT
      SUM=0.
      DO 13 J=1,NTOT
   13 SUM=SUM+AAIBD(L,J)*Y(J)
   12 YP(L)=YP(L)+SUM
   38 CONTINUE
      IF(IFL(3).EQ.1) GO TO 24
C  ACCELERATION PROFILES
      IF(IFL(30).EQ.1.OR.IFL(32).EQ.1) GO TO 25
      DO 14 J=1,NCST
   14 GDOT(J)=0.
      CALL CONG(Y,YP,T)
   25 CONTINUE
      DO 15 L=1,NTOT
      SUM=0.
      DO 16 J=1,NCST
   16 SUM=SUM+AAI(L,NMNC+J)*GDOT(J)
   15 YP(L)=YP(L)+SUM
   24 CONTINUE
C
      RETURN
      END
□
```