COMPUTER-AIDED KINEMATIC AND DYNAMIC ANALYSES OF FLEXIBLE SPATIAL MANIPULATORS OF ARBITRARY ARCHITECTURE

JINESH JAIN

COMPUTER-AIDED KINEMATIC AND DYNAMIC ANALYSES OF FLEXIBLE SPATIAL MANIPULATORS OF ARBITRARY ARCHITECTURE

By

© Jinesh Jain

A thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Engineering.

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

October 1988

St. John's Newfoundland Canada

ISBN 0-315-50447-1

ABSTRACT

The research work involves the kinematic and dynamic analyses of flexible manipulators of (a) known architecture, (b) arbitrary architecture, and (c) redundant manipulators. The analyses also include the solution of constrained problems. In order to increase the computational efficiencies various efficient numerical schemes have also been studied for solving the dynamic problems.

The constrained displacement analysis of these manipulator has been carried out by successive linear approximation principle where linear and non linear constraints can be handled. The constrained velocity and acceleration analyses have been carried out using quadratic programming principle and simplex method.

In the dynamic analysis, the equations of motion have been obtained using the finite element method. The global system matrices are function of end-effector position and the force vector is a function of kinematic parameters. The dynamic equations are solved using four efficient numerical techniques. The methods of solving the constrained dynamic equations are the Karmarkar's algorithm and the simplex method. To reduce the size of the system matrices, the Guyan reduction technique and component mode synthesis have been used.

ACKNOWLEDGEMENTS

With respect and pleasure I thank my mentor, Professor Anand M. Sharan, for his expert advice and providing the necessary funds and facilities. My work with him has been a valuable personal and professional experience.

Sincere gratitude is expressed to Dr. C. A. Sharpe, Acting Dean of School of Graduate Studies, and Dr. G. R. Peters, Dean of Engineering, for providing the financial support during the course of my graduate work. I will like to express my appreciation and gratitude to Dr. T. R. Chari, Associate Dean of Engineering, for his valuable assistance during this program.

Special thanks are due to my friends, especially Pramod Malgo and Parveen Kalra for their encouragement and unfailing support. The author also wishes to acknowledge Mrs. Wanda Heath for her help and patience in the preparation of this manuscript.

This work is dedicated to the memory of my father,
late Dharam C. Marwari

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

a_j	restricted joint acceleration variables
a_{ij}	coefficients of the variables in the constraint equations
a_N	large penalty
$\{a\}$	approach vector of the end-effector
$[A_0^i]$	4X4 'arm' matrix for proper orientation and positioning of the end-effector
$[A_0^{*i}]$	modified 4X4 'arm' matrix for proper positioning of the end-effector
b_i	components of $\{B\}$ vector
c_k	constant term in the objective function
c_{kp}	orthogonal projection of $[D_k]\{c_k\}$
$\{c_k\}$, $\{c_i\}$	coefficients of the variables in the objective function
c_i	constant functions of joint angles and joint velocities
$[C]$	global damping matrix
$\{d_k\}$	n-dimensional vector
$[d_1, \ldots, d_n]$	components of the diagonal matrix $[D]$
$[D]$, $[D_k]$	diagonal matrix

$\{e_1\}$	first column of an identity matrix
$\{e_k\}$	vector with identity elements
$\{E\}$	new set of variables containing $\{S_\ell^2\}$
$\{f_k\}$	n-dimensional vector
F_1, F_2, F_3	residual functions in the global X_0, Y_0, Z_0 directions
F_{a_1}, F_{a_2}, F_{a_3}	residual joint accelerations functions in the global directions
F_{v_1}, F_{v_2}, F_{v_3}	residual joint velocities functions in the global directions
$\{F_G\}_{in}$	force due to gravity due to nth node on ith link
$\{F_i\}$	generalized force vector
$(F_I)_{inX}$, $(F_I)_{inY}$, $(F_I)_{inZ}$	inertia force due to the nth node on ith link in the global directions
g_k	real number
g_x, g_y, g_z	acceleration components due to gravitational effects
[G]	linear coefficient matrix
H_i	distance from the origin of the (i-1)th coordinate system to the intersection of X_{i-1} axis with the X_i axis along the Z_{i-1} axis
$[H_i]$	Householder matrix

$[I]$	identity matrix
$\tilde{k}_{i\beta\gamma}$	finite element structural stiffness terms
K_1	constant term in quadratic objective function
$[K]$	global stiffness matrix
L	Lagrange function
L_i, D_i, d_i	shortest distance between Z_{i-1} and Z_i axes
$[L]$	lower triangular matrix
$[L_T]$	lower triangular matrix with positive diagonal entries
m_{in}	lumped mass of the nth node on the ith link
$[M]$	global mass matrix
$\{n\}$	normal vector of the end-effector orientation
N	total number of generalized coordinates
NG_i	total number of mode points on ith link
N_ℓ	total number of links
$NP(i)$	total number of perturbation coordinates on ith link
P_i	constant terms in quadratic objective function

$\{p_{in}\}$	perturbation coordinates of the nth node on the ith link
$\{p_k\}$	direction vector
$\{p'_k\}$	normalized orthogonal projection
q_i	generalized coordinates
q_i^L	lower limit on generalized coordinate
q_i^U	upper limit on generalized coordinate
$\{q_i^0\}$	solution vector at the previous configuration
$\{q_i^*\}$	intitial approximate starting vector
$\{\dot{q}_i\}$	generalized joint velocity vector
$\{\ddot{q}_i\}$	generalized joint acceleration vector
$\{q(t)\}$	global displacement vector
$\{q(t)\}^I$	interface displacement coordinates
$\{q(t)\}^F$	free displacement coordinates
$[Q]$	quadratic coefficient matrix
$\{r_i\}$	position vector of a point with respect to the ith link
$\{r_i^0\}$	position vector of a point with respect to global coordinate system

r_k	radius of the largest sphere
$[R_k]$	upper triangular matrix in QR factorization
s	slave degree of freedom
$\{s_i\}$	starting vector in QR decomposition
$\{S_\ell\}$	slack variable vector
$\{S_t\}$	surplus variable vector
$\{t\}$	sliding vector of the end-effector orientation
T_i	constant terms in quadratic objective function
$[T_E]$	transformation matrix for component mode synthesis
$[T_{i-1}^i]$	4X4 Denavit-Hartenberg transformation matrix
U	composite objective function to be minimized
U'	objective function of non-negative artificial variables
U_K	link kinetic energy
U_P	link potential energy
$\{U_1\}$	vector identifying hyperplane span
$(U_K)_{in}$	translational kinetic energy of the ith link due to nth node

$(U_{PG})_{in}$	link potential energy of nth node on ith link due to gravitational effect
$(U_{PE})_{in}$	link potential energy of nth node on the ith link due to elastic effects
v_i	restricted variables
V_{in}	inertial velocity vector of the nth node on the ith link
$\{V\}$	inertial velocity vector containing restricted variables
w_i	constant terms in quadratic objective function
x_i, x_k, x_{kj}	restricted variables
$\{x_k'\}$	restricted variables in the projected domain
$\{\hat{x}_k\}$	vector of restricted variables
$\{y_k\}$	least square solution vector for calculating the projection vector
z_k	objective function
θ_i	joint angle from the X_{i-1} axis to the X_i axis about Z_{i-1} axis in the right hand sense
α_i	angle between Z_{i-1} axis to Z_i axis about the X_i axis, in the right hand sense
α_j	non-negative artificial variables

α_k	step length
λ_k	Karmarkar's artificial variable
α, β	control parameters to determine the accuracy and stability in Newmark integration scheme
$\{\lambda\}$, $\{\xi\}$	Lagrangian multipliers
ρ	2-norm of a vector
σ_k	plausible upper bound for scaling the variables
ξ_k	Karmarkar's new variable equal to Unity
Δt	time step in Newmark integration scheme
$[\phi]$	transformation matrix
$[\phi_E]$	matrix of eigen vector
$\{\dot{\eta}(t)\}$	principal coordinate vector
ℓ_1, m_1, n_1	direction cosines of the local X-axis with respect to inertial coordinate system
ℓ_2, m_2, n_2	direction cosines of the local Y-axis with respect to inertial coordinate system
ℓ_3, m_3, n_3	direction cosines of the local Z-axis with respect to inertial coordinate system
$\{Y\}$, $\{Z\}$	intermediate solution vectors
$[X_i\ Y_i\ Z_i]$	coordinate system of the ith link

$[x^0_{in}, y^0_{in}, z^0_{in}]$	coordinates of the nth node on ith link with respect to inertial coordinate system
$[\dot{x}^0_{in}, \dot{y}^0_{in}, \dot{z}^0_{in}]$	inertial velocity vector of the nth node on the ith link
$[X_0\ Y_0\ Z_0]$	global coordinate system
$[\dot{X}_0\ \dot{Y}_0\ \dot{Z}_0]$	cartesian velocities of the end-effector in the global direction
$[\ddot{X}_0\ \ddot{Y}_0\ \ddot{Z}_0]$	cartesian accelerations of the end-effector in the global direction
$\frac{\partial}{\partial q_i}$	partial derivative with respect to q_i
$\{\ \}$	vector notation
$[\ ]$	matrix notation
$[\ ]^T$	transpose of a matrix
$[\ ]^{-1}$	inverse of a matrix
$\|\ \|_2$	vector norm -2
$\|\ \|$	magnitude of a vector

CHAPTER 1

INTRODUCTION

1.1 Introduction to Robotics

The prime requirement of a modern manufacturing system is to produce a wide range of products in batch production and at the same time ability to incorporate various changes in the product design. This largely demands flexibility in the manufacturing systems. In a mechanized assembly, the transfer line approach may fail when a major design change is required, as this will result in abandoning the special-purpose manufacturing assembly. Flexible automation has always been a core issue in all manufacturing innovations. Numerically controlled production lines had, to a great extent, played a vital role in introducing flexibility in manufacturing. The contemporary robots are programmable, multifunction manipulator designed to perform a variety of tasks through variable programmed motions, a feature that has evolved from the application of numerical control. Fig. 1.1 shows the closed-loop assembly line in a typical robot operated subsystem.

One of the important characteristics of robotic manipulators is the high level of mobility and dexterity which results in a very wide range of potential applications. This is mainly attributed to the mechanical structure of the robot which is made of cantilevered beams called links, connected by hinged joints. This serial linkage

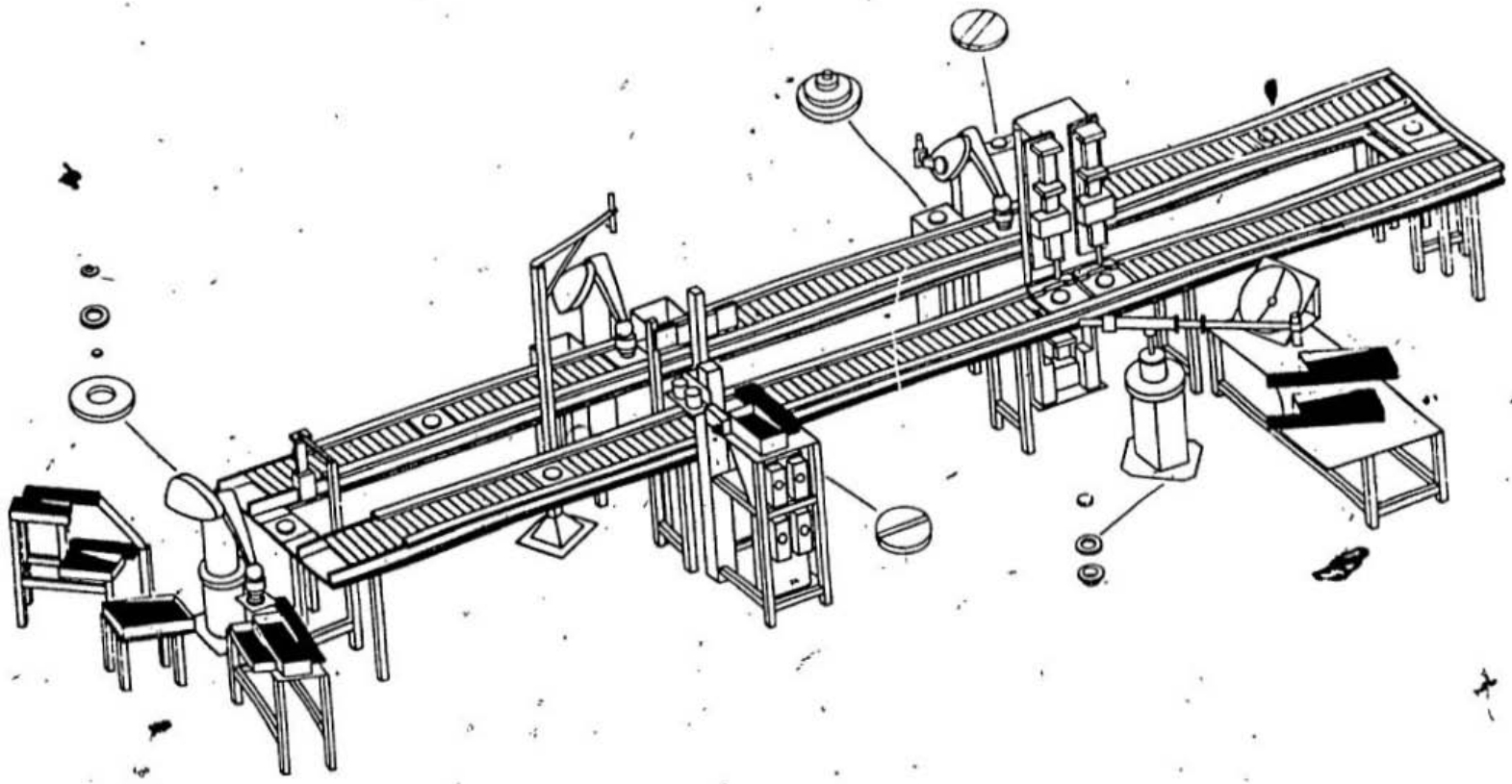

FIG. #1.1: A Typical Robot-Aided Closed Loop Assembly Line [1]

configuration of manipulator arms is described by complex non-linear equations. Unlike the single-input mechanism where there is a single drive, a robot is a multi-input spatial mechanism and consequently requires more sophisticated analysis. The robots presently used in industrial applications have six degrees of freedom. Normally, a minimum of six degrees of freedom are required for a robot to accomplish any task. Out of these six degrees of freedom, three degrees are associated with the robot arms to position itself at the required point in space and the rest three degrees are associated with the wrist as shown in Fig. 1.2 for its proper orientation. An end-effector is a kind of gripper attached at the end of robot wrist to perform useful work. Though desired end-effector motion can be achieved by a six-degree-manipulator but a large portion of the workspace is occupied by the singularity configurations. Singularity arises at a particular robot configuration where the resolved motion rates required to achieve the desired motion in certain directions are enormously high. In such configurations the manipulator's mobility is hampered as it is unable to move in the desired direction. By introducing additional joints, the degrees of freedom can be accordingly increased as degrees of freedom of a manipulator is equal to total number of joints. This in turn introduces redundancy in the system and the additional joints help in avoiding the singularity configurations and still achieve the desired position. Such robots with redundancy are called redundant manipulators. Such manipulators have infinite number of solutions that provide the same motion required for the end-effector. The human arm

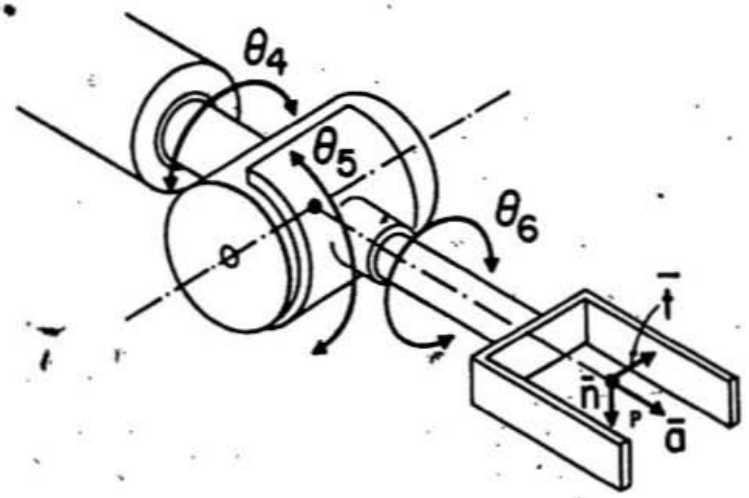

FIG. #1.2: Wrist Coordinate System

has seven degrees of freedom, thus it falls in the category of redundant arms. The study of redundant manipulators is an important research topic in advanced manipulation and there is always a need for efficient numerical techniques for their analyses.

1.2. Literature Survey

1.2.1 Kinematic Analysis of Redundant Manipulators

Kinematic analysis of the robot deals with the analytical study of the spatial configuration of the robot arm as a function of time. For describing the spatial geometry of a rigid link, Denavit and Hartenberg [2] first came up with the 4X4 homogeneous coordinate transformation matrix.

The inverse displacement analysis of robotic manipulators with six degrees of freedom involves the solution of six joint coordinates corresponding to a given position and orientation of the end effector [3-5]. The solution technique can be roughly divided into two categories. In the first type, the analytical methods are used for the manipulators having known architecture. In this type, the equations containing the wrist coordinates can be solved separately from the position coordinates like Stanford manipulator, PUMA-560, T3R3, etc. This is possible if three axes intersect at a point [6]. By such separation, it is possible to get closed-form solutions. Some of the important research work relating to the above mentioned methods can be seen in [7-10] and general kinematic methods in [11-17]. The

existence of a closed-form solution depends on the kinematic structure of the manipulator arms. Thus for robots with redundant arms, the closed form solutions may not be possible to obtain using analytical approaches. Also, for robots, if the joint coordinates cannot be separated, one has to solve for all the unknown joint variables together. This would be a formidable task if one looks for closed form solutions, i.e. in this case three consecutive revolute joints do not meet at a single point. In such situations one way out would be to go for numerical techniques based upon iterative computation schemes. On the other hand, for robots of arbitrary architecture the numerical solutions are also quite useful. As the task of the robot becomes more complicated, its mobility is affected by various types of implicit, explicit, linear or non-linear constraints. Such constrained problems would be quite difficult to solve analytically.

The kinematic link equations of arbitrary robots can correspond to under-determined systems in general. The joint displacements for such type of problems can be determined using methods based on finding a generalized inverse [18], or one can also use Least Square Method (LSM) involving Householder's reflections [19]. However the work in [19] does not include any type of constraints. Another possible problem encountered in solving kinematic equations is the existence of multiple solutions as shown in Fig. 1.3. There are four possible ways for the end effector to reach the point P. For an obstacle free trajectory one seeks a solution which requires minimum movement of the joints to avoid undue oscillations in the mechanism. Such problems can

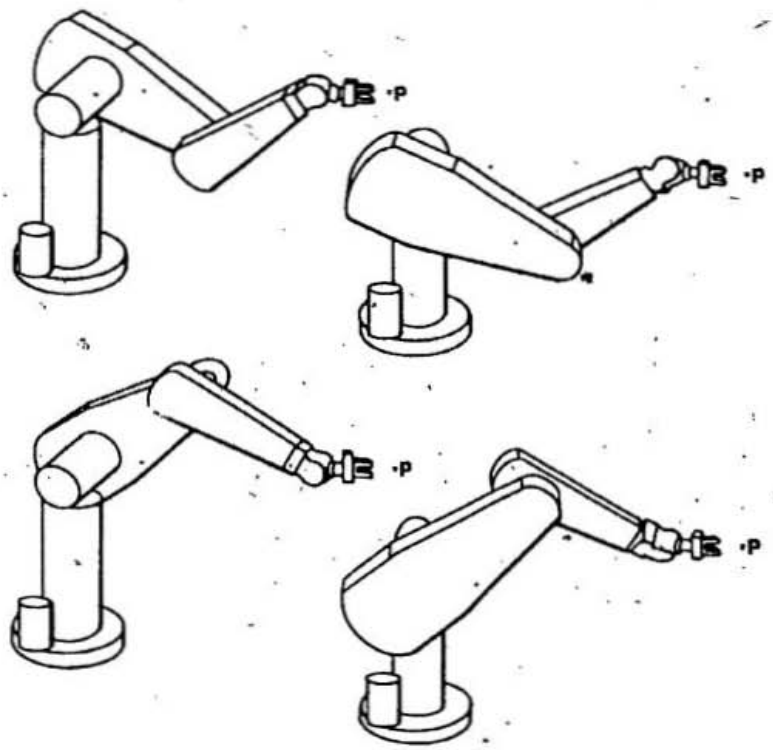

FIG. #1.3: Multiple Solutions of the PUMA 560

be easily handled using methods based on optimization principles such as discussed in [20]. In this work, the required solution was obtained for the minimum change in the joint angles from the previous configuration using a composite objective function defined in terms of two non-linear functions, one of which was defined in terms of cartesian space variables and the other, in terms of joint variables. However, the technique requires a considerable amount of CPU time. Fast computations are required to transform a large number of points along the trajectory into the joint displacement space. Furthermore the computation time becomes critical when transformation is required in real time.

1.2.2 Dynamic Analysis of Robotic Manipulators

The dynamic equations of motion describe the dynamic behaviour of the manipulator. The formulation of robot arm dynamics can be approached by various techniques such as Lagrange-Euler principle [21], the Recursive-Lagrange method [22], the Newton-Euler formulation [23] and the Generalized D'Alembert technique [24]. The derivation of the dynamic equations based on Lagrange-Euler equation is simple and systematic. Uicker [25] derived the equations of motion for modelling multi-loop, multi-degree-of-freedom spatial linkage mechanisms. Another significant work came up using screw coordinate method [26] to derive the equations of motion. Yang [27] came up with 'dual number' approach based on dual vectors and screw calculus. But these methods are only applicable to rigid link dynamics. Winfrey [28] used finite-element

approach to study the dynamic behaviour of flexible mechanisms. In the lumped parameter formulation, Sadler et al [29] analysed a flexible planar mechanism. Dynamic analysis of flexible spatial mechanisms and robotic manipulators using Lagrange equation was carried out by Sunada [30]. Using the finite-element modelling, coordination reduction techniques have been carried out in [31-33]. This helps in considerably reducing the computation time and still retains the accuracy in modelling.

It is equally important to have efficient solution techniques for calculating the dynamic response. The differential equations of motion can be solved by finite difference schemes The system of dynamic equations of motion in the time domain can be expressed as a system of linear algebraic equations using various techniques [34]. On the other hand, the kinematic or dynamic equations can be expressed in the form of under-determined, exact or over- determined system of equations. The solutions of such problems can be found in [35-42].

1.3 Objectives

The robots are used to perform various kinds of tasks like assembly, welding, loading-unloading, painting, drilling, etc. Some of these tasks have to be performed with very high accuracy and fast speed which require accurate analyses of the dynamic effects. The dynamic equation of motion also involves kinematic parameters. Therefore, the objectives of the present work are:

1) The displacement analysis of robots of
   (a) known architecture
   (b) arbitrary architecture, and
   (c) redundant manipulators.
2) The constrained displacement analysis of the above mentioned manipulators.
3) The constrained and unconstrained velocity analyses of the above mentioned manipulators.
4) The constrained and unconstrained acceleration analyses of the above mentioned manipulators.
5) The derivation of dynamic equations of motion of flexible manipulators.
6) The transient response calculation of these manipulators using the Newmark integration scheme.
7) A study of efficient techniques for the solution of resulting dynamic equations.
8) Use of the Linear Programming (L.P.) method and Karmarkar's algorithm in the solution of constrained dynamic problems.
9) A study of condensation techniques to reduce the size of system matrices for enabling economical transient calculations.

In Chapter 2, the kinematic analysis of robotic manipulators have been carried out using (a) successive linear approximation principle, (b) linear programming method, and (c) quadratic programming. In this way the inverse kinematic problem has been solved from the information provided in the cartesian space which include the

constraints.

Next, the dynamic equation of motion are obtained using the finite element analysis in Chapter 3. The resulting matrix differential equation of motion have been reduced to a system of algebraic equations using the Newmark integration scheme. These equations are then solved using various efficient numerical methods and their efficiencies are also compared. The constrained dynamic problems have been solved by linear programming and Karmarkar's algorithm. Finally, the system matrices have been reduced using two dynamic condensation techniques.

CHAPTER 2

THE INVERSE KINEMATIC ANALYSIS OF ROBOTS OF ARBITRARY ARCHITECTURE

2.1 Introduction to Kinematic Analysis

In the kinematic analysis of robotic manipulators, one normally solves two kinds of problems. In the first type, the displacement vector or its first and second order derivatives are specified in the cartesian space and the solutions to such problems are required in the joint space. The variables in the joint space can be vectors consisting of sliding and rotational variables and the corresponding first and second time derivatives. These types of problems are referred to as inverse kinematic problems. On the other hand, if the joint space variables are given and the cartesian space variables are to be found out then these would be called as the direct kinematic problems.

As mentioned in the previous chapter, a given job can be performed by correctly positioning the end-effector with proper orientation. In large number of manipulators the appropriate orientation can be achieved independent of the various arms manipulations. The kinematic equations for such type of manipulators can be de-coupled and hence fewer equations are needed to be solved at a given time. This results in considerable saving of computation time. The inverse kinematic problems are most frequently encountered in actual practice because

the tasks to be performed are normally in the cartesian space and since the manipulation is done through motors, the joint space variables and their derivatives must be calculated at various instants of time.

In the present chapter, the non-linear inverse displacement analysis is carried out using a general technique which is applicable to commonly used manipulators, manipulators of arbitrary architecture and redundant manipulators. After this, the inverse velocity and acceleration analyses have also been carried out by other new techniques for these types of manipulators.

2.2 The Constrained Displacement Analysis of Robotic Manipulators

2.2.1. Solution of Non-linear Displacement Equations Using Successive Linear Approximation

The inverse displacement analysis of robotic manipulators involves solution of joint coordinates from the knowledge of cartesian space variables. These relationships exist in form of non-linear set of equations. In some of the manipulators the wrist coordinates can be solved separately from the end-effector coordinates. Examples of such manipulators are mentioned earlier in Section 1.2.1. On the other hand, the solutions of other kind of manipulators, such as redundant types or of arbitrary architecture, are quite involved and not necessarily known. The solution methods for such kind of manipulators are quite challenging.

In the light of the above discussion, the intent of the work here is to efficiently solve for the joint displacements of manipulators having arbitrary architecture, or redundancy, subjected to linear, non-linear, equality or inequality constraints. This method is based on Successive Linear Approximations [SLA]. In this technique [43], both the non-linear objective function and the non-linear constraints are linearized first and then the solution is obtained by solving a two-phase linear programming problem.

2.2.2 The Denavit-Hartenberg Representation

There exist a kinematic relationship between a pair of adjacent links in an open kinematic chain system. The Denavit-Hartenberg notation [2] is a systematic way of describing this relationship using 4X4 homogeneous transformation matrix. Figure 2.1 shows the relative orientation and position of two adjacent links, link i and link $i - 1$. The Denavit-Hartenberg representation of a robot link depends upon four geometric parameters. Through these four parameters complete description of a revolute or prismatic joint is possible. The four parameters are as follows:

1. L_i is the shortest distance between Z_{i-1} and Z_i axes.
2. θ_i is the joint angle from the X_{i-1} axis to the X_i axis about Z_{i-1} axis in the right hand sense.
3. H_i is the distance from the origin of the $(i - 1)$th coordinate system to the intersection of Z_{i-1} axis with the X_i axis along the Z_{i-1} axis.

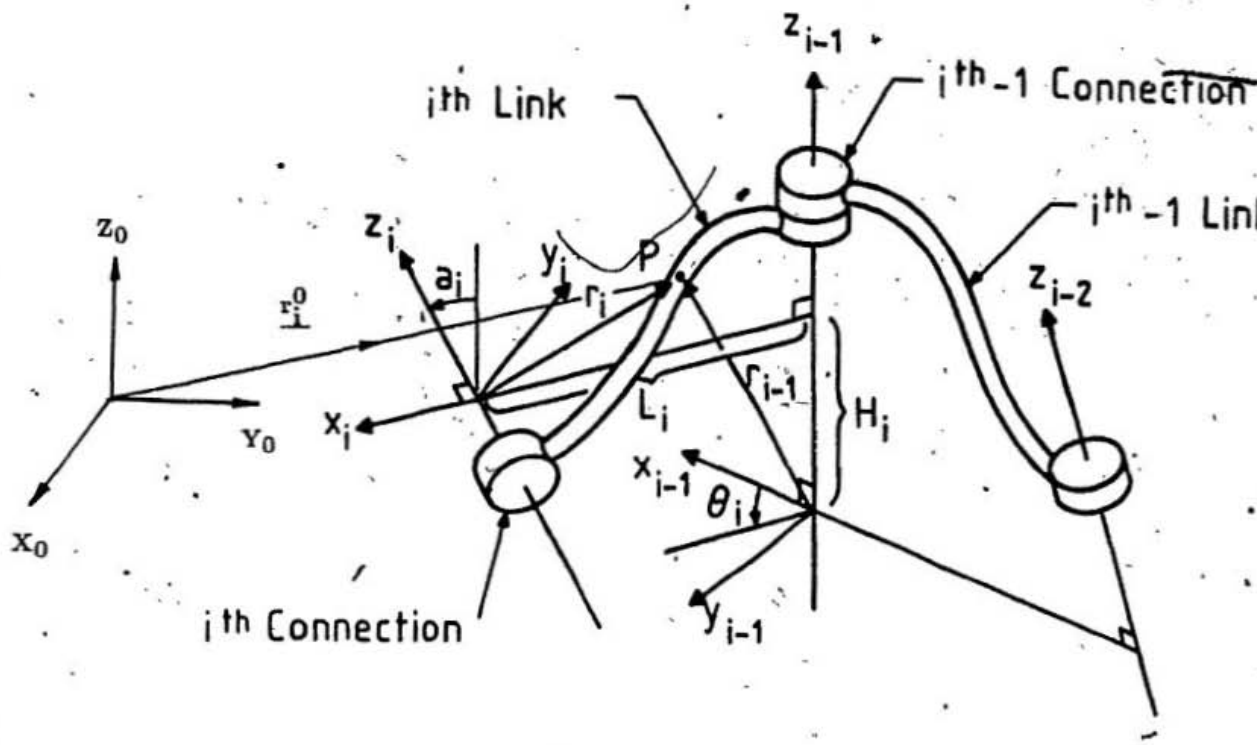

FIG. #2.1: Hartenberg-Denavit Notation

4. α_i is the angle between Z_{i-1} axis to the Z_i axis about the X_i axis, in the right-hand sense.

The time varying parameters for links connected by revolute joints is θ_i. Similarly for a link with prismatic joint, H_i is the time varying parameter. Figure 2.1 also shows the position vectors of a point P with respect to the global coordinate system $(X_0\ Y_0\ Z_0)$, and the moving coordinate system on the ith and (i-1)th links. So the position vector $\underline{r}_i$ can be written as

$$\underline{r}_i = \begin{bmatrix} 1 \\ X_i \\ Y_i \\ Z_i \end{bmatrix} \tag{2.1}$$

Similarly, the position vector of this point with respect to the (i-1)th link will be

$$\underline{r}_{i-1} = \begin{bmatrix} 1 \\ X_{i-1} \\ Y_{i-1} \\ Z_{i-1} \end{bmatrix} \tag{2.2}$$

The transformation matrix relating these two position vectors is given by

$$\underline{r}_{i-1} = [T^i_{i-1}]\underline{r}_i \tag{2.3}$$

where $[T^i_{i-1}]$ is the Denavit-Hartenberg transformation matrix and it is expressed as:

$$[T^i_{i-1}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ L_i\cos\theta_i & \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i \\ L_i\sin\theta_i & \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i \\ H_i & 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix} \quad (2.4)$$

Using Eq. (2.3) one can write

$$\underline{r}^0_i = [T^1_0][T^2_1][T^3_2] \ldots [T^i_{i-1}]\underline{r}_i = [T^i_0]\underline{r}_i \quad (2.5)$$

Each of the transformation matrices $[T^1_0]$, $[T^2_1]$, etc., contain the joint variables which are functions of time.

2.2.3 The Derivation of Displacement Equations

For a given position of the end-effector, the matrix $[T^i_0]$ in Eq. (2.4) contains the joint angles which we will call as the generalized coordinates. These generalized coordinates also include the displacements in the prismatic joints. The vectors $\underline{r}^0_i$ or $\underline{r}_i$ in Eq. (2.5) have four components which include three cartesian space variables. Suppose a rough approximate of the generalized coordinates is made and substituted in Eq. (2.5), then the difference between the vector $\underline{r}^0_i$ and the product of $[T^i_0]$ and $\underline{r}_i$ will not, in general, be equal to a null vector. The first component of the residual vector will always be zero but the other three components may take positive or negative values. Defining three functions F_1, F_2, F_3

corresponding to residues in the global X_0, Y_0, Z_0 directions, one can write analytical expressions for any kind of manipulator with known link parameters. First we will discuss the analytical expressions for standard robots with six degrees of freedom for better understanding of the mathematical formulation and later the redundant manipulators will be studied.

A point to be noted here is that the solutions for the displacement equations for standard robots are well known and a numerical solution would not be desirable in these cases. But, for those cases where the joint space constraint and other linear, non-linear constraint requirements are to be satisfied then the proposed method will be quite effective. Table 2.1 shows the geometric link parameters for T3R3 manipulator shown in Figure 2.2. The three residual functions in the global direction for this manipulator are expressed as:

$$F_1 = L_2\cos q_2\cos q_1 + L_3\cos q_1\cos(q_2 + q_3) - X_0 \qquad (2.6a)$$

$$F_2 = L_2\cos q_2\sin q_1 + L_3\sin q_1\cos(q_2 + q_3) - Y_0 \qquad (2.6b)$$

$$F_3 = L_2\sin q_2 + L_3\sin(q_2 + q_3) - Z_0 \qquad (2.6c)$$

Similarly for Stanford manipulator and PUMA-560, shown in Figures (2.3), (2.4a) and (2.4b) respectively, one can express the functions F_1, F_2 and F_3 as follows.

PUMA-560

$$F_1 = H_3\cos q_1\cos(q_2 + q_3) - L_4\cos q_1\sin(q_2 + q_3) +$$

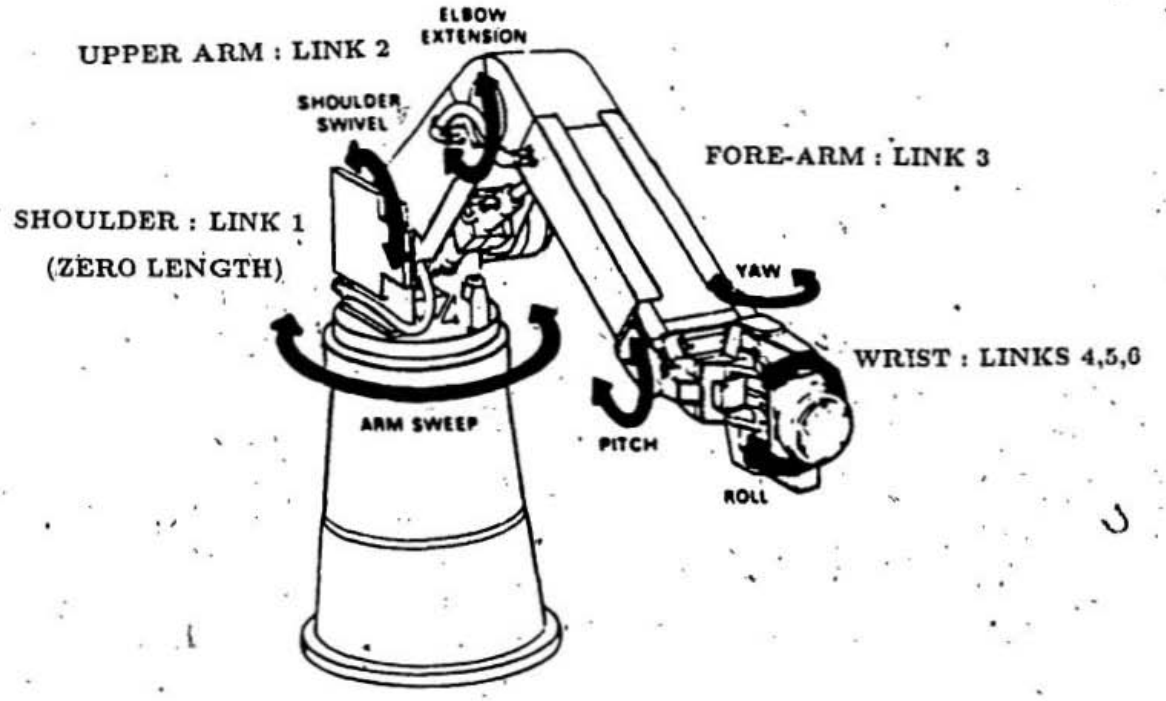

FIG. #2.2: T3R3 Model Robotic Manipulator

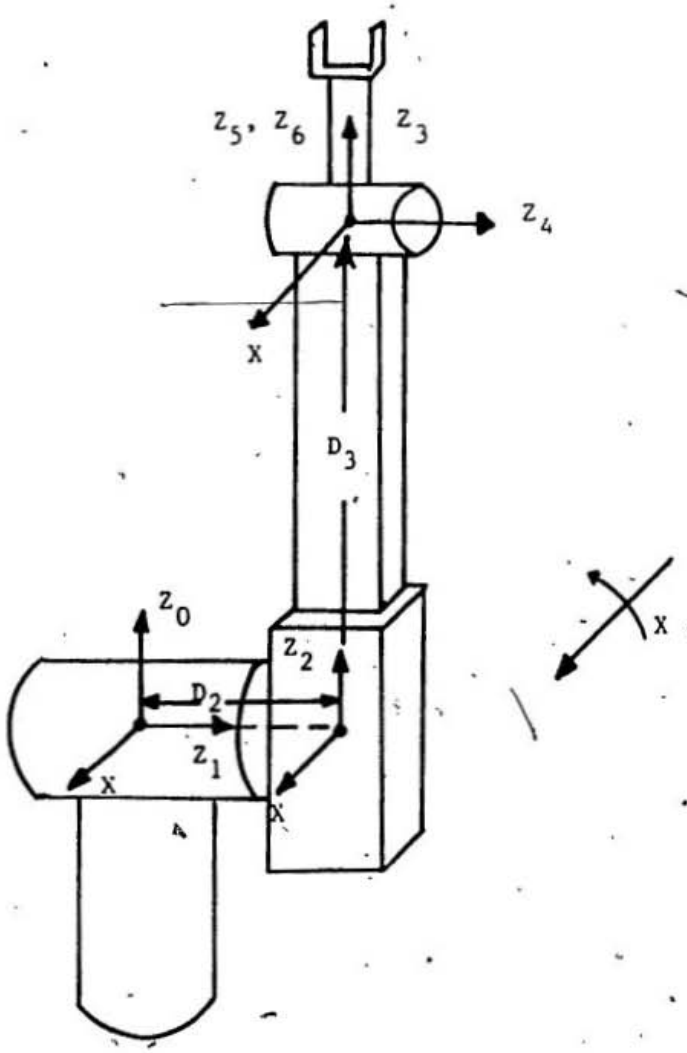

FIG. #2.3: Link Coordinate Frames for the Stanford Manipulator

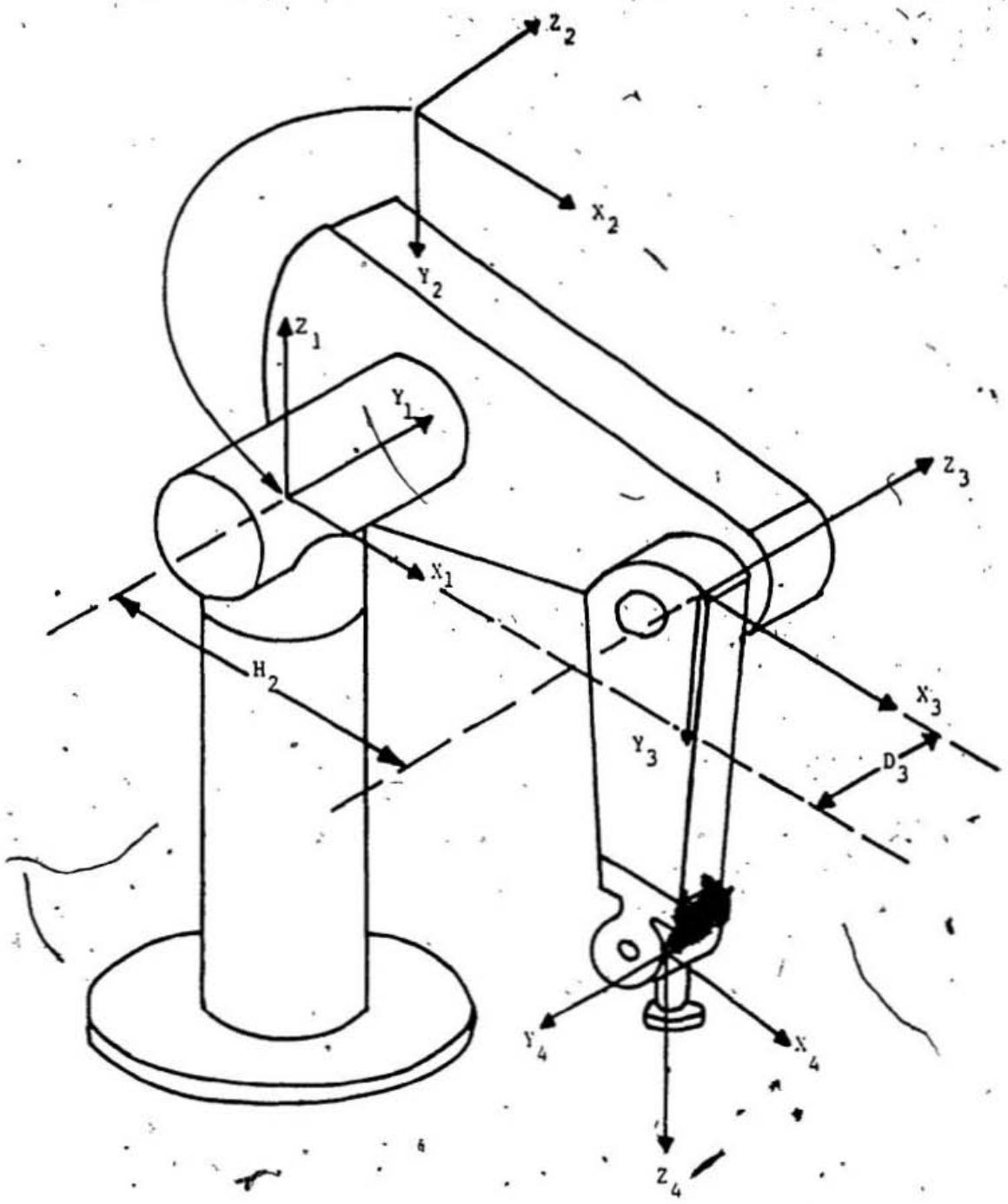

FIG. #2.4a: Link Coordinate Frames for the PUMA-560 Manipulator

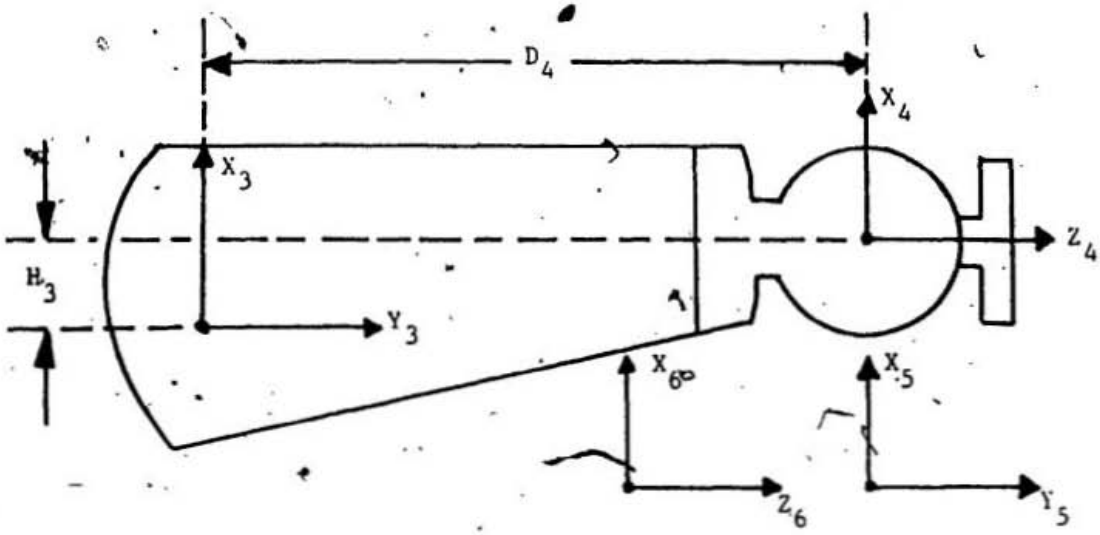

FIG. #2.4b: Coordinate Frame for Forearm of the PUMA-560 Manipulator

TABLE #2.1: Link Parameters of T3R3 Manipulator

Link i	α_i (degrees)	θ_i (degrees)	H_i (m)	D_i (m)
1	90	θ_1	0	0
2	0	θ_2	0	1.016
3	0	θ_3	0	1.511

$$+ H_2\cos q_2\cos q_1 - L_3\sin q_1 - X_0 \qquad (2.7a)$$

$$F_2 = H_3\sin q_1\cos(q_2 + q_3) - L_4\sin q_1\sin(q_2 + q_3) +$$

$$+ H_2\cos q_2\sin q_1 + L_3\cos q_1 - Y_0 \qquad (2.7b)$$

$$F_3 = -H_3\sin(q_2 + q_3) - L_4\cos(q_2 + q_3) -$$

$$- H_2\sin q_2 - Z_0 \qquad (2.7c)$$

Stanford Manipulator

$$F_1 = q_3\cos q_1\sin q_2 - L_2\sin q_1 - X_0 \qquad (2.8a)$$

$$F_2 = q_3\sin q_1\sin q_2 + L_2\cos q_1 - Y_0 \qquad (2.8b)$$

$$F_3 = q_3\cos q_2 - Z_0 \qquad (2.8c)$$

$$q_3 = L_3$$

The geometric link parameters for these manipulators are shown in Table 2.2 and 2.3.

In a similar fashion analytical expressions can be derived for manipulators with more than six degrees of freedom, which are called redundant manipulators. Figure 2.5 shows a manipulator with seven degrees of freedom. The displacement analysis of this manipulator requires determination of seven joint angles. Table 2.4 shows the geometric link parameters for this manipulator. Unlike standard robots as discussed before, the positional analysis for this manipulator requires solution of four joint angles. The residual functions in this case can be expressed as

TABLE #2.2: Link Parameters of PUMA-560 Manipulator

Link i	α_i (degrees)	θ_i (degrees)	H_i (m)	D_i (m)
1	0	θ_1	0	0
2	-90	θ_2	0.4318	0
3	0	θ_3	0.02032	0.127
4	-90	θ_4	0	0.4318

TABLE #2.3: Link Parameters of Stanford Manipulator

Link i	α_i (degrees)	θ_i (degrees)	H_i (m)	D_i (m)
1	-90	θ_1	0	0
2	90	θ_2	0	0.1524
3	0	θ_3	0	D_3

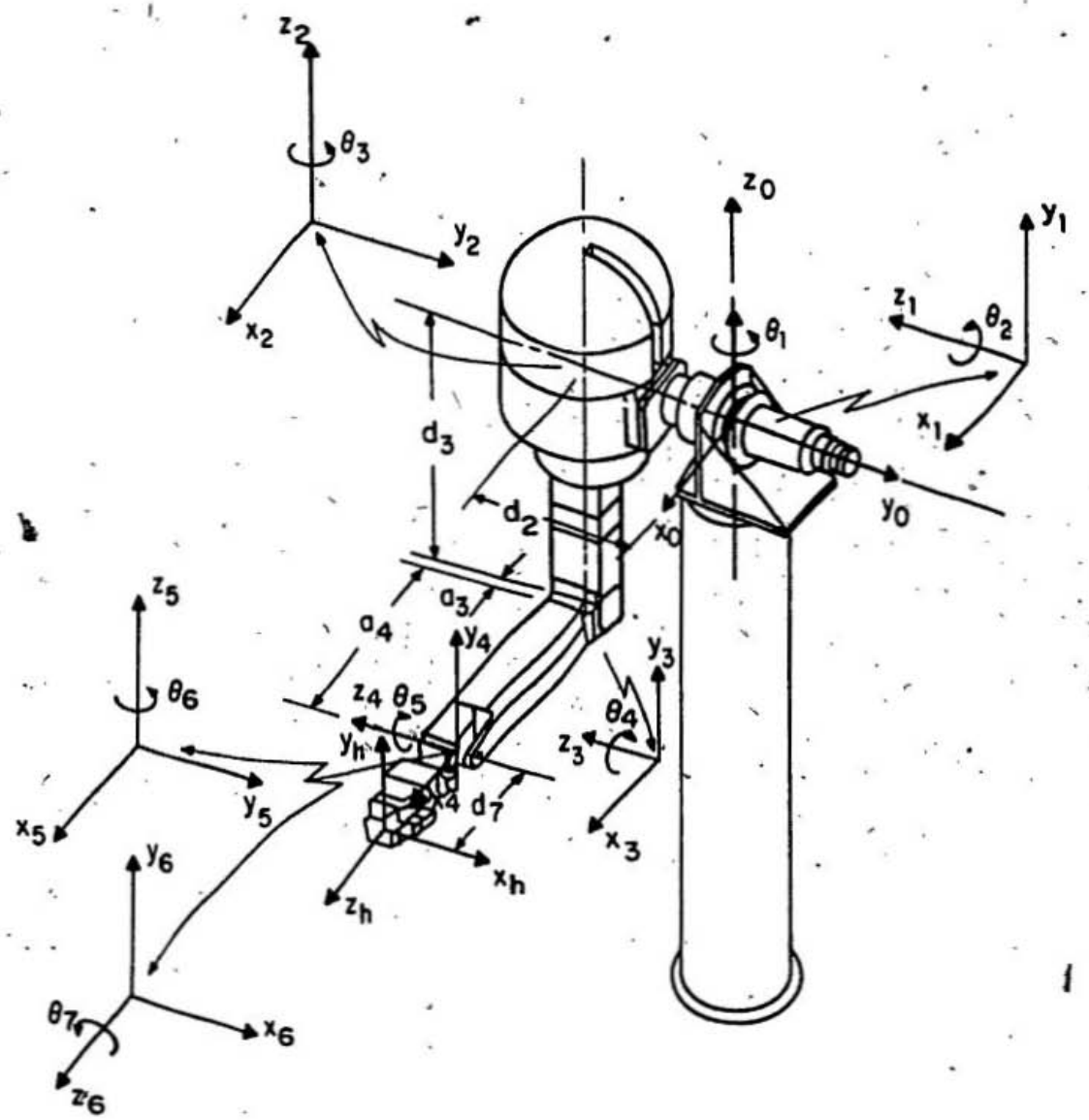

FIG. #2.5: Seven Degrees of Freedom Redundant Manipulator

TABLE #2.4: Link Parameters For Seven-Degree of Freedom Manipulator

Link i	α_i (degrees)	θ_i (degrees)	a_i (m)	d_i (m)
1	90	θ_1	0	0
2	-90	θ_2	0	0.356
3	90	θ_3	0.029	-0.635
4	0	θ_4	0.508	0
5	-90	θ_5	0	0
6	90	θ_6	0	0
7	0	θ_7	0	0

$$F_1 = H_4(\cos q_1 \cos q_2 \cos q_3 \cos q_4 - \sin q_1 \sin q_3 \cos q_4) -$$

$$- - H_4(\sin q_2 \sin q_4 \cos q_1) + H_3(\cos q_1 \cos q_2 \cos q_3 -$$

$$- \sin q_1 \sin q_3) + D_3 \sin q_2 \cos q_1 + D_2 \sin q_1 - X_0 \qquad (2.9a)$$

$$F_2 = H_4(\sin q_1 \cos q_2 \cos q_3 \cos q_4 + \cos q_1 \sin q_3 \cos q_4 -$$

$$- \sin q_1 \sin q_2 \sin q_4) + H_3(\sin q_1 \cos q_2 \cos q_3 +$$

$$+ \sin q_3 \cos q_1) + D_3(\sin q_1 \sin q_2) - D_2 \cos q_1 - Y_0 \qquad (2.9b)$$

$$F_3 = H_4(\sin q_2 \cos q_3 \cos q_4 + \cos q_2 \sin q_4) +$$

$$+ H_3(\sin q_2 \cos q_3) - D_3 \cos q_2 - Z_0 \qquad (2.9c)$$

Another example of a redundant manipulator can be a modification of Stanford robot as shown in Figure 2.6, where the redundancy arises because of the prismatic joint introduced at the base of the robot. The three functions in this case are

$$F_1 = q_3 \cos q_1 \sin q_2 - L_2 \sin q_1 - X_0 , \qquad (2.10a)$$

$$F_2 = q_3 \cos q_2 + L_2 + Y_0 , \text{ and} \qquad (2.10b)$$

$$F_3 = q_3 \sin q_2 \sin q_1 + L_2 \cos q_1 + D_0 - Z_0 . \qquad (2.10c)$$

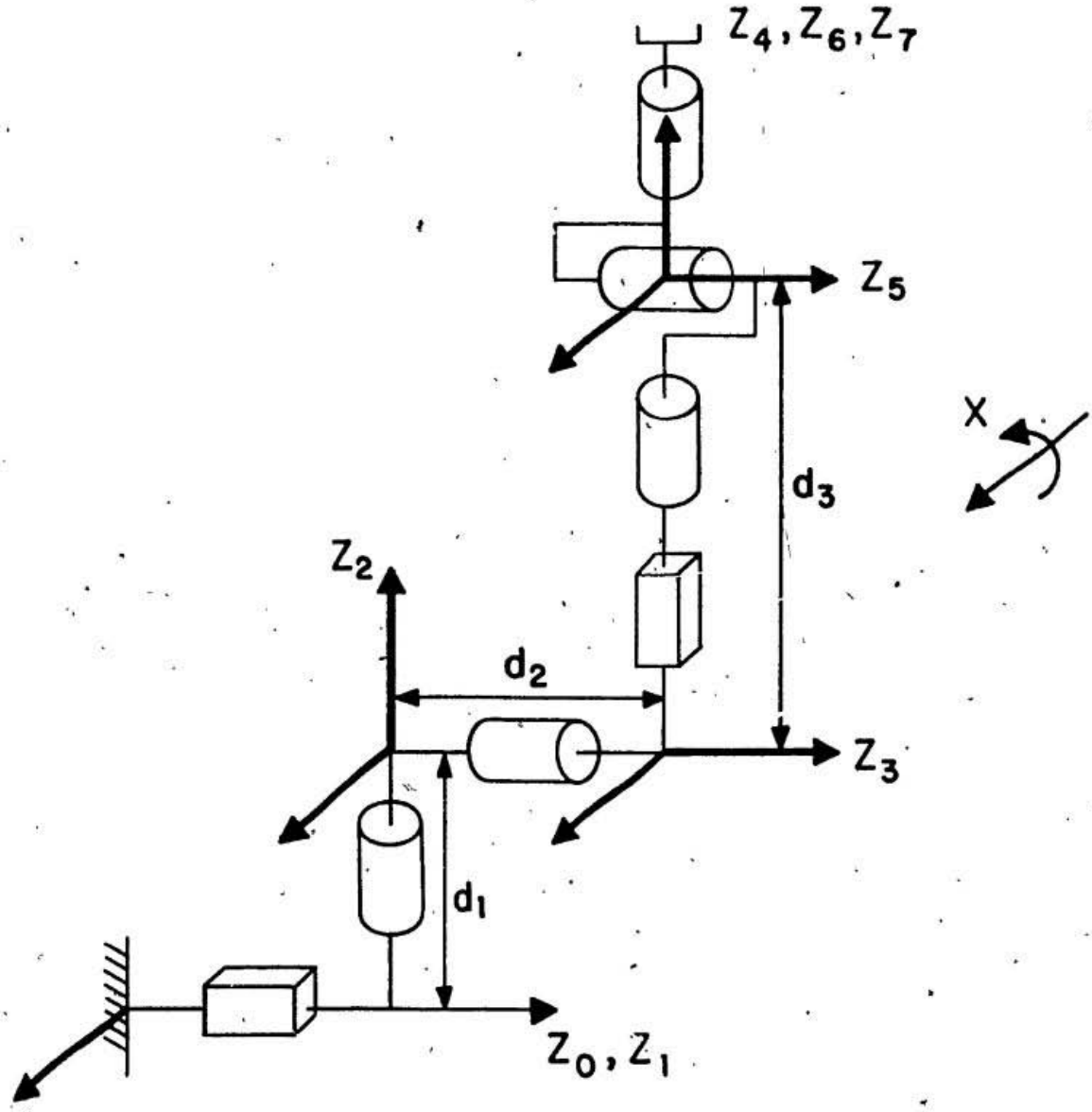

FIG. #2.6: Seven Degrees of Freedom Stanford Manipulator

2.2.4 The Displacement Equations Including the Orientation of the End-Effector.

In order to orient the end-effector properly to perform a task one needs to know the wrist orientations as shown in Figure 1.2. In this figure, $\{n\}$ is the normal vector of the end-effector, $\{t\}$ is the sliding vector and $\{a\}$ is the approach vector. The global coordinates of the point P are X_0, Y_0, Z_0 which can be obtained using the following successive matrix multiplication in the case of T3R3 robot:

$$[T_0^1]\,[T_1^2]\,[T_2^3]\,[T_3^4]\,[T_4^5]\,[T_5^6] = [A_0^6] \tag{2.11}$$

Here the matrix $[A_0^6]$ can be written in the following form.

$$[A_0^6] = \left[\begin{array}{c|ccc} 1 & 0 & 0 & 0 \\ \hline X_0 & n_x & t_x & a_x \\ Y_0 & n_y & t_y & a_y \\ Z_0 & n_z & t_z & a_z \end{array}\right] \tag{2.12}$$

In Eq. (2.12), the 3x3 partitioned matrix corresponds to the orientation of the wrist and $(X_0\ Y_0\ Z_0)$ are the global coordinates of the point P. In actual practice the matrix $[A_0^6]$ is normally known and one has to calculate the joint variables occurring in each of the matrices in the left hand side of Eq. (2.11). However, the unknowns in the matrices $[T_0^1]$, $[T_1^2]$ and $[T_2^3]$ can be solved independent of $[T_3^4]$, $[T_4^5]$ and $[T_5^6]$ because the last three axes intersect at a point. Therefore, for the first three unknowns one can modify Eq. (2.11) as

$$[T_0^1][T_1^2][T_2^3] = [T_0^3] = [A_0^{*6}]$$

$$= \left[\begin{array}{c|ccc} 1 & 0 & 0 & 0 \\ \hline X_0 & 0 & 0 & 0 \\ Y_0 & 0 & 0 & 0 \\ Z_0 & 0 & 0 & 0 \end{array}\right] \qquad (2.13)$$

and solve for the three unknowns.

The details of the method of solution are given in the next section. One substitutes the value of these unknowns in the Eq. (2.11) and solves for the remaining three variables. In the next subsection the actual solution of various manipulators subject to various types of constraints is discussed.

2.2.5 The Solution of the Displacement Equations Subject to Constraints

In the Section 2.2.4 the displacement residual equations for PUMA-560 manipulator were derived and expressed as Eqs. (2.7a) to (2.7c). This manipulator is designed to operate within a constrained work envelope. The constraint space can be defined in terms of the following constraint equations

$$-160^\circ < \theta_1 < 160^\circ$$
$$-225^\circ < \theta_2 < 45^\circ \qquad (2.14)$$
$$-45^\circ < \theta_3 < 225^\circ$$

Similarly, one can express a constraint for stanford manipulator as

$$0.127m \leq D_3 \leq 0.635m \qquad (2.15)$$

If there are any other constraints then those should also be expressed similarly.

One can clearly see that the simultaneous solution of F_1, F_2, and F_3 which are transcendental in nature and subject to constraints expressed in Eqs. (2.14) and (2.15), is quite involved and calls for non-linear methods of solution. One of these methods is the SLA method. If we define an objective function given by

$$U_1 = F_1^2 + F_2^2 + F_3^2 \qquad (2.16)$$

and minimize this function to zero then we would obtain the solution for simultaneous equations expressed as Eq. (2.7a) to (2.7c). However if the constraints are also to be included then one of the ways would be to substitute for the variable occurring in the constraint equations into F_1, F_2 and F_3. Since the constraint equations are of inequality type, this method will be very tedious, but fortunately if we use the SLA method then the constraint equations can be handled separately.

Another type of constraint which is very desirable in the trajectory planning is the minimum movements of the links between two

successive points on a trajectory. Thus the resulting objective function to be minimized can be expressed as:

$$U = \lim_{a_N \to +\infty} (a_N U_1 + U_2) \tag{2.17}$$

where a_N is a large penalty associated with U_1 to assure its minimization. In this equation U_2 is expressed as

$$U_2 = \sum_{\ell=1}^{N_\ell} (q_i - q_i^0) \tag{2.18}$$

Here $\{q_i^0\}$ is the solution vector at the previous configuration. Clearly our objective is to minimize U subject to constraints.

$$\begin{aligned} F_1(q_i) &= 0 \\ F_2(q_i) &= 0 \\ F_3(q_i) &= 0 \\ q_i &\leq 360^\circ, \quad i = 1, 2, 3, \ldots, N_\ell \end{aligned} \tag{2.19}$$

where N_ℓ is the number of links.

The Eq. (2:19) reduces the range of search for the joint variables q_i; for example it would be better if the range of q_1 is limited between 0° and 360° rather than $-\infty$ to $+\infty$ because q_1 occurs in the argument of a cosine function. If we incorporate this constraint the optimum would be found quicker numerically.

In the constraint equation, Eq. (2.19), the angles can be changed depending upon the workspace limitation or obstacles. It is quite possible that generalized variables are constrained by certain lower and upper limits. Under such situations, one should write these constraints in the following form:

$$-q_i + q_i^L \leq 0 \ , \ i = 1, 2, 3, \ldots, N_\ell \tag{2.20}$$

$$q_i - q_i^U \leq 0 \ , \ i = 1, 2, 3, \ldots, N_\ell \tag{2.21}$$

where q_i^L and q_i^U are the lower and upper limits for q_i respectively. The objective function given by Eq. (2.17) is non-linear in nature. Also the three residual functions in global directions are non-linear. One can linearize the function U about an initial starting vector $\underline{q}^*$ defined in the joint space as

$$U = U^* + (q_1 - q_1^*) \frac{\partial U^*}{\partial q_1} + (q_2 - q_2^*) \frac{\partial U^*}{\partial q_2} +$$

$$+ (q_3 - q_3^*) \frac{\partial U^*}{\partial q_3} + \ldots + (q_{N_\ell} - q_{N_\ell}^*) \frac{\partial U^*}{\partial q_{N_\ell}}$$

The above equation can be rewritten as

$$U = q_1 \frac{\partial U^*}{\partial q_1} + q_2 \frac{\partial U^*}{\partial q_2} + q_3 \frac{\partial U^*}{\partial q_3} + \ldots + q_{N_\ell} \frac{\partial U^*}{\partial q_{N_\ell}}$$

$$- \left(q_1^* \frac{\partial U^*}{\partial q_1} + q_2^* \frac{\partial U^*}{\partial q_2} + q_3^* \frac{\partial U^*}{\partial q_3} + \ldots + q_{N_\ell}^* \frac{\partial U^*}{\partial q_{N_\ell}} - U^* \right) \tag{2.22}$$

Similarly one can linearize the constraint equations also as

$$F_i = q_1 \frac{\partial F_i^*}{\partial q_1} + q_2 \frac{\partial F_i^*}{\partial q_2} + q_3 \frac{\partial F_i^*}{\partial q_3} + \ldots + q_{N_\ell} \frac{\partial F_i^*}{\partial q_{N_\ell}}$$

$$- \left(q_1^* \frac{\partial F_i^*}{\partial q_1} + q_2^* \frac{\partial F_i^*}{\partial q_2} + q_3^* \frac{\partial F_i^*}{\partial q_3} + \ldots + q_{N_\ell}^* \frac{\partial F_i^*}{\partial q_{N_\ell}} \right.$$

$$\left. - F_i^* \right) , \quad i = 1, 2, 3, \ldots, N_\ell \qquad (2.23)$$

Since the starting vector $\underline{q}^*$ could be farther away from the optimal vector $(\underline{q})_{opt}$, we have to proceed to the optimum by a sequence of linear approximations. The essence of the scheme is the assumption that the linear approximations used are useful only over a narrow range. Because of this the value of the independent variables may change by only a small amount at each step. Also, the variable steps taken for each of the generalized coordinates would depend on the sensitivity of the objective function for that particular variable. One can take larger steps for the favourable variables. Eq. (2.22) and Eq. (2.23) can be written in the standard form as:

$$U = c_1 q_1 + c_2 q_2 + c_3 q_3 + \ldots + c_{N_\ell} q_{N_\ell} + c_k \qquad (2.24)$$

subject to constraints:

$$\begin{aligned} a_{11} q_1 + a_{12} q_2 + a_{13} q_3 + \ldots + a_{1N_\ell} q_{N_\ell} &= b_1 \\ a_{21} q_1 + a_{22} q_2 + a_{23} q_3 + \ldots + a_{2N_\ell} q_{N_\ell} &= b_2 \\ a_{31} q_1 + a_{32} q_2 + a_{33} q_3 + \ldots + a_{3N_\ell} q_{N_\ell} &= b_3 \end{aligned} \qquad (2.25)$$

where a_{ij}, b_i and c_i are constants. It is understood that one can write the Eq. (2.20) and Eq. (2.21) similar to Eq. (2.25) by adding slack variables [44]. In the simplex algorithm we always start with a set of equations which includes the objective function along with the equality constraints in the canonical form. In fact, one has to optimize Eq. (2.24) without the constant term c_k which would vary from point to point as the global optimum is approached. At the global optimum U should be approaching zero by including the constant term c_k.

To illustrate the concept graphically let us take an example of a function U_1 which is a function of two variables q_1 and q_2. Fig. 2.7 shows the first starting point and the first optimal point in the linearized zone. The optimal solution for the first step is shown to be at the intersection of the imposed constraint equations arising due to the step lengths. In addition, we can also impose similar additional constraints to avoid the possibility of obtaining the multiple solutions, since we are seeking the closest possible solution vector q_i from the previous end-effector position. If this linear approximation step is repeated a few times then we would reach the optimum point.

In the example above, the generalized variables, q_1 and q_2, were all positive. But in some situations they can take negative values i.e. the variables are un-restricted. By using the concept of restricted variables, one can easily modify the objective function and constraint equations in the following form:

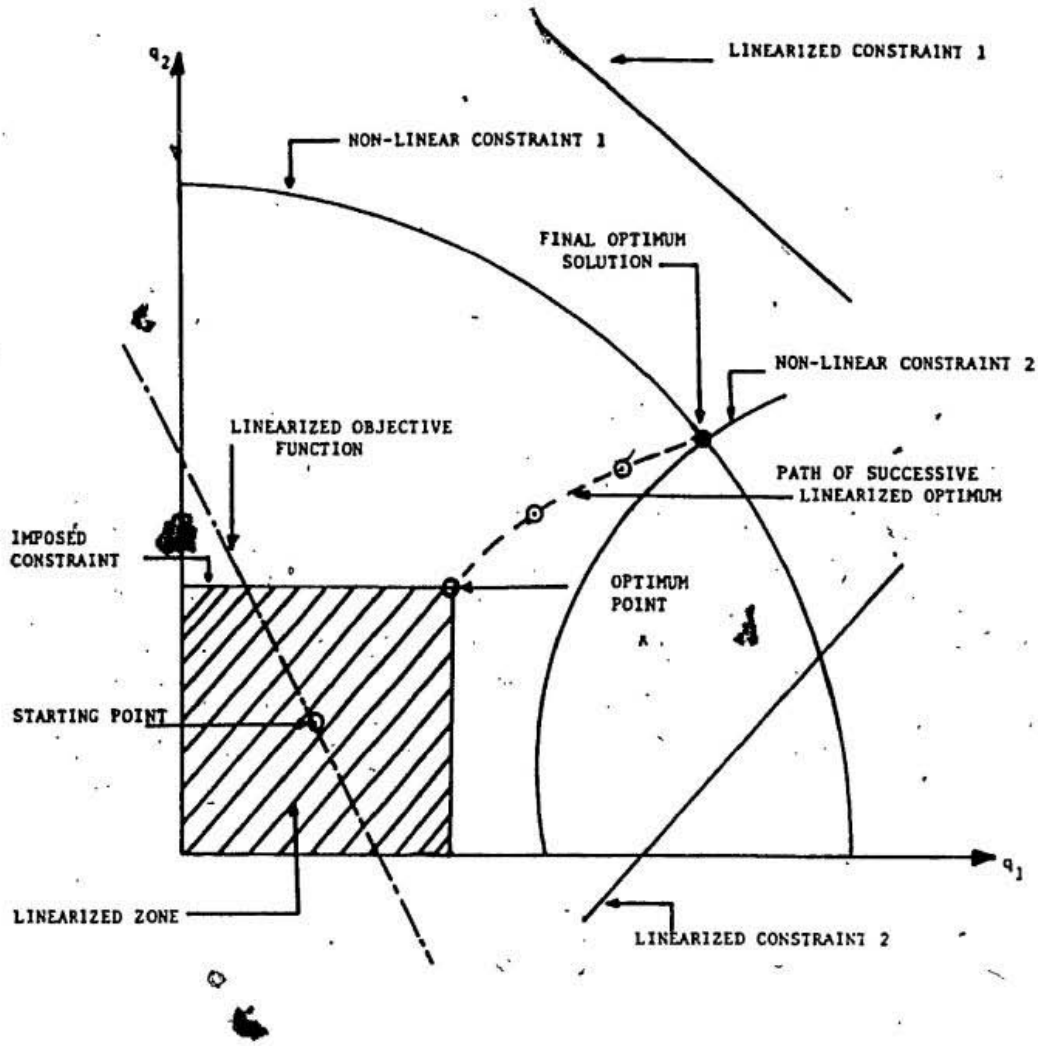

FIG. #2.7:. Programming by Successive Linear Approximations

$$U = (c_1x_1 + c_2x_2) - (c_1x_3 + c_2x_4)$$

$$\text{where } q_1 = x_1 - x_3$$

$$q_2 = x_2 - x_4 \qquad (2.26)$$

$$x_i \geq 0$$

All x_i are restricted variables as they can take values either zero or positive. The constraint equations can also be modified in a similar manner. In the linear programming problem the optimal solution vector exists at one of the vertices of the linearized zone. One starts at one of the vertices and moves along a particular boundary along which the objective function decreases at a maximum rate. Non-negative artificial variables are added to the left hand side of the constraint equations which are either greater or equal to zero. Simultaneously, a large penalty is also introduced in the objective function [44]. Since the objective function is always of minimization type, this artificial variables will not appear in the optimum solution. The slack variables are added whenever these equations are less than or equal to zero. The optimal solution of the linear programming problem is sought in two phases. The first phase is used to find the optimum of the basic feasible solutions. Using this optimum one uses phase II to obtain the global optimum.

The use of the constrained minimization can be understood by taking another example. In this case, one is interested in minimizing the function

$$F(x_1,x_2) = -3x_1^2 - 2x_2^2 \tag{2.27}$$

subject to constraints

$$\begin{aligned} g_1(x_1,x_2) &= x_1^2 + x_2^2 \leq 25 \\ g_2(x_1,x_2) &= 9x_1 - x_2^2 \leq 27 \\ g_3(x_1,x_2) &= x_1^2 + 2x_1 + x_2^2 = 33 \\ x_i &\geq 0,\ i = 1, 2 \end{aligned} \tag{2.28}$$

In this problem both the equality and inequality constraints are present. The constraint equations are modified by adding slack variables in the case of inequality constraints and artificial variables for the equality constraints. This is done after linearizing the constraints and the objective function. These, at the point (1,1), become

$$\begin{aligned} F'(x_1,x_2) &= -6x_1 - 4x_2 + 5 \\ g_1'(x_1,x_2) &= 2x_1 + 2x_2 + x_3 = 27 \\ g_2'(x_1,x_2) &= 9x_1 - 2x_2 + x_4 = 26 \\ g_3'(x_1,x_2) &= 4x_1 + 2x_2 + x_5 = 35 \end{aligned}$$

Here x_3 and x_5 are slack variables and x_5 is the artificial variable. To obtain the optimum, the phase I is used to eliminate x_5 and phase II for x_3 and x_4. The objective function, $F(x_1,x_2)$, was found to be equal to -66 at the point (4,3).

The displacement analysis was carried out for two different tasks to be performed by different manipulators. The trajectories of the two tasks are shown in Figs. 2.8 and 2.9. In Fig. 2.8 the robots perform a spot welding operation by going around a plate (18" X 18"). In Fig. 2.9 these robots measure the various diameters of the stepped- shaft during a turning operation. The detailed description of the trajectory of the stepped-shaft is given in Table 2.5 and 2.6. The results obtained for T3R3 manipulator are shown in Figs. 2.10-2.15. In Fig. 2.10, the angle θ_1 is constant from points 1-11, it decreases rapidly between 11-21; it stays constant between 21-31 and sharply increases from 31-40. The variations of θ_2 and θ_3 in Figs. 2.11 and 2.12 respectively, are non-linear, therefore there will be sharp variations in the angular accelerations. For the second type of work, the variations of θ_1, θ_2 and θ_3 with time are very steep as shown in Figs. 2.13-2.15. The variation of θ_1 corresponding to the stepped shaft for PUMA 560 and for Stanford manipulator are shown in Figs. 2.16 and 2.17 respectively. Clearly, for this task, θ_1 variations are very sharp for all three manipulators. The prismatic joint has almost a constant linear velocity from points 1-11 with a break at the point 6 as shown in Fig. 2.18. Between points 11 and 30, the velocity increases with time, correspondingly there will be variations in the linear accelerations. These variations are in the opposite manner after the point 30.

The efficiency of SLA method can be judged from Table 2.7. In this table the CPU time (using VAX-8800) for one solution using the

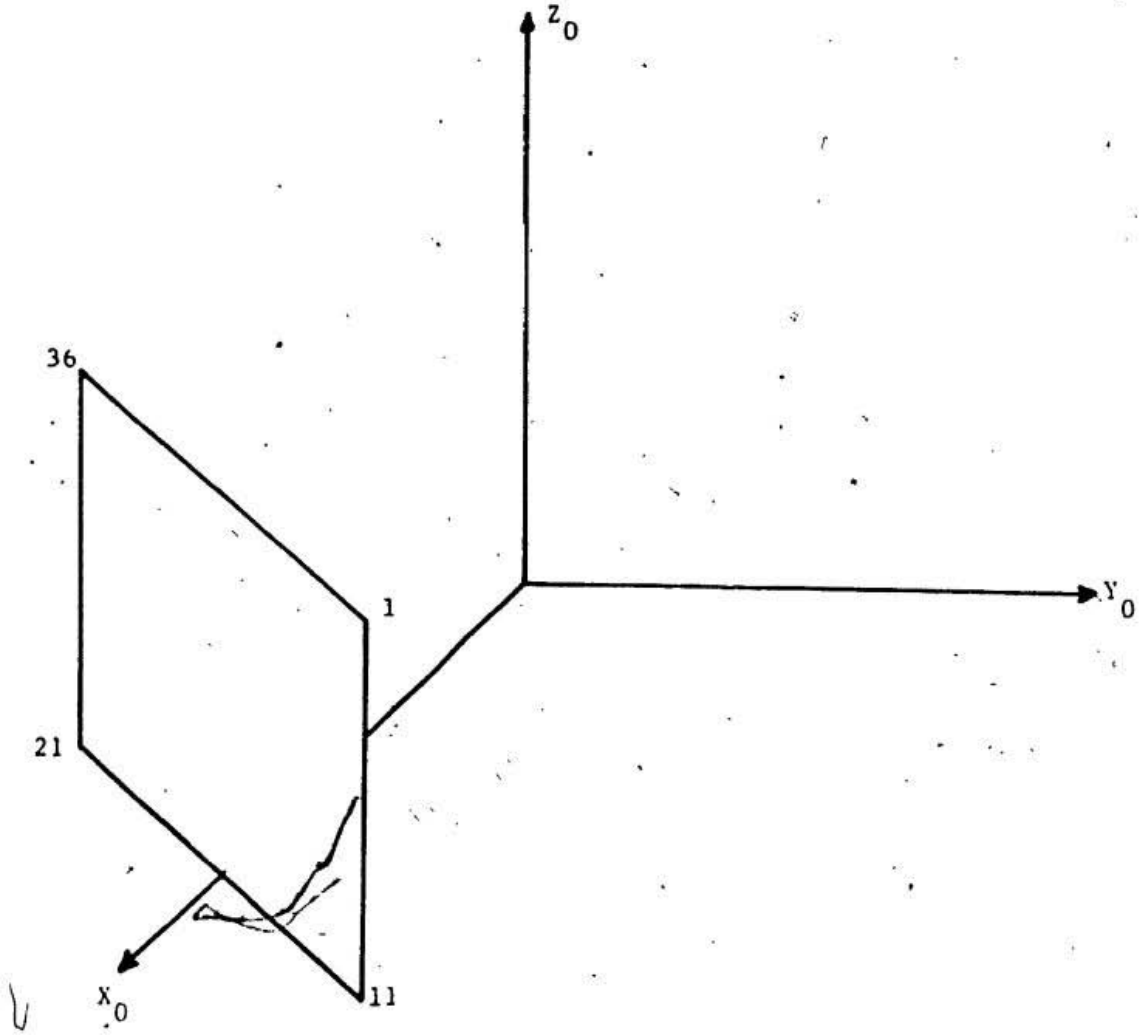

FIG. #2.8: Various Points on the Plate in the Global Coordinate System

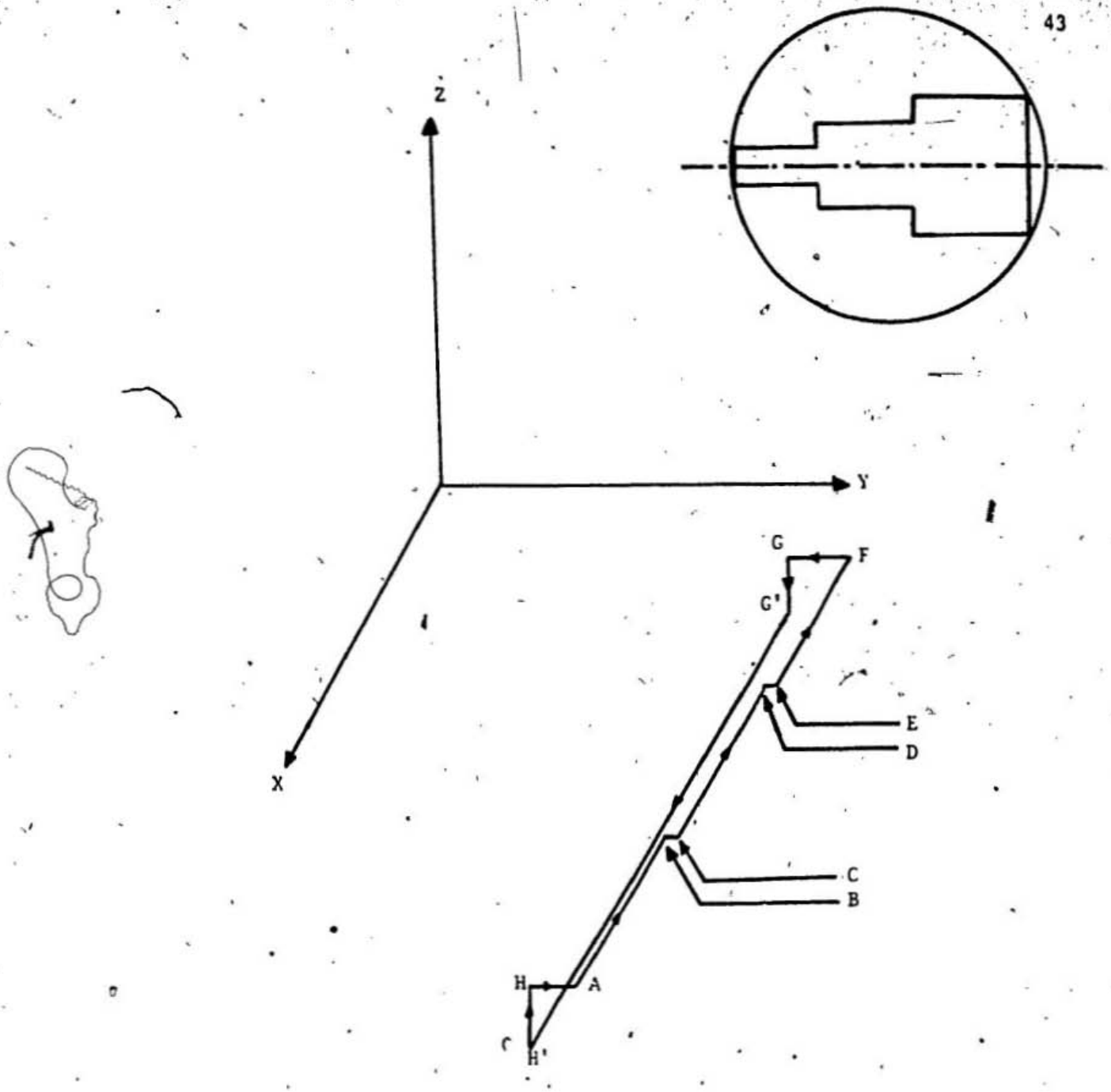

FIG. #2.9: Trajectory for the Stepped -Shaft

TABLE #2.5: The Coordinate of Various Points of the Stepped-Shaft Trajectory for PUMA 560

POINTS	$X_{(m)}$	$Y_{(m)}$	$Z_{(m)}$
H'	0.50	0.2754	0.3
H	0.50	0.2754	0.4
A	0.50	0.3500	0.4
B	0.20	0.3500	0.4
C	0.20	0.3627	0.4
D	-0.10	0.3627	0.4
E	-0.10	0.3754	0.4
F	-0.35	0.3754	0.4
G	-0.35	0.2754	0.4
G'	-0.35	0.2754	0.3

TABLE #2.6: Detailed Description of the Stepped-Shaft Trajectory

Path Description	Details of the End-Effector Trajectory
H' - H	Along Z-axis, to bring the end-effector in the plane of the shaft.
H - A	Along Y-axis, to position the end-effector for performing the job.
A - B, C - D, E - F	Along X-axis, robot performs the job of measuring the diameter at selected points.
B - C, D - E	Along Y-axis, step jump by end-effector to position itself for new configuration.
F - G	Along Y-axis, end-effector retraces back after performing the job.
G - G'	Along Z-axis, end-effector is positioned in the lower z-plane.
G' - H'	Along X-axis, straight sweep by end-effector to come to the initial starting position, indicating the completion of the cycle.

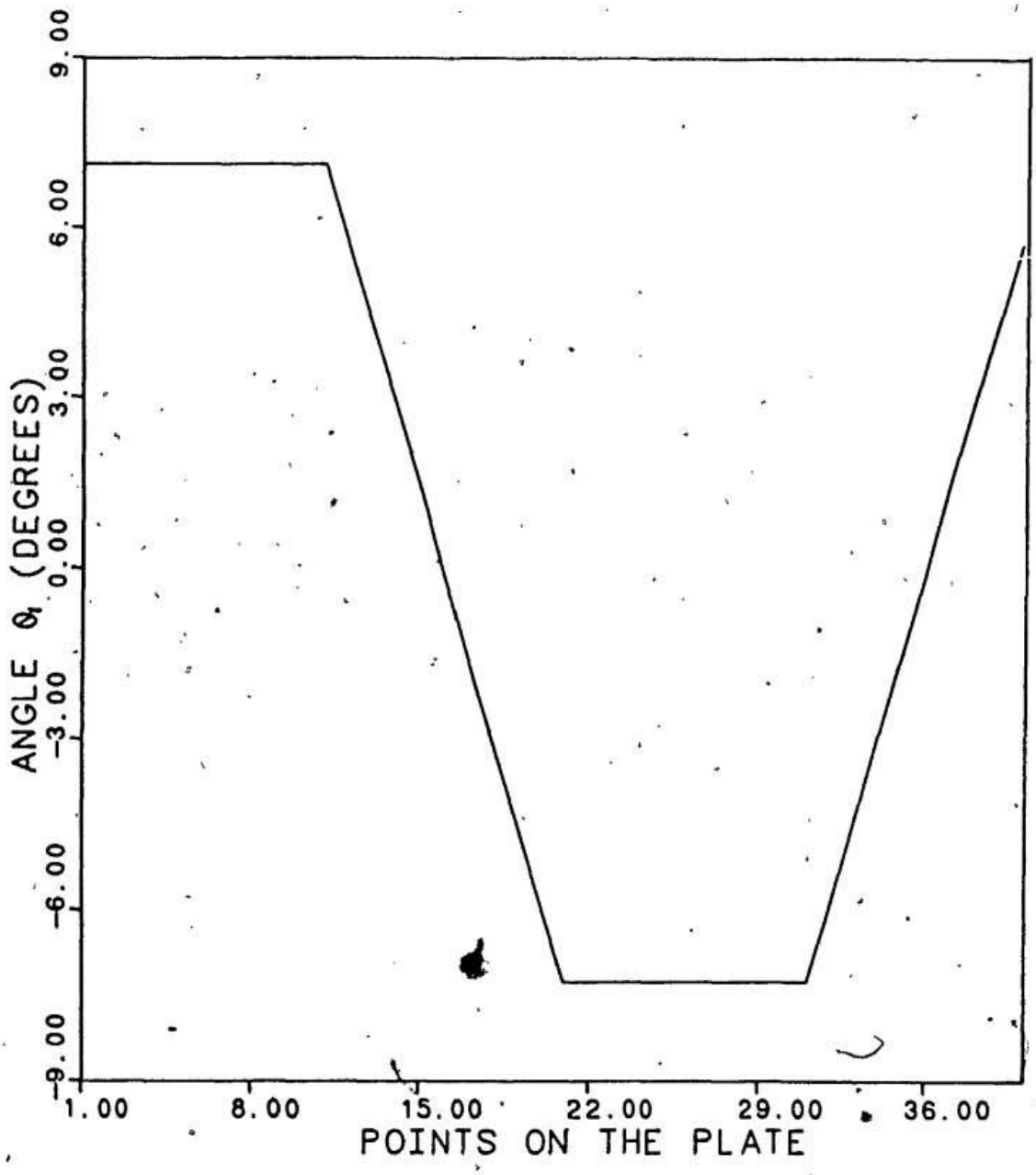

FIG. #2.10: The Variation of θ_1 at Various Points on the Plate for T3R3 Robot

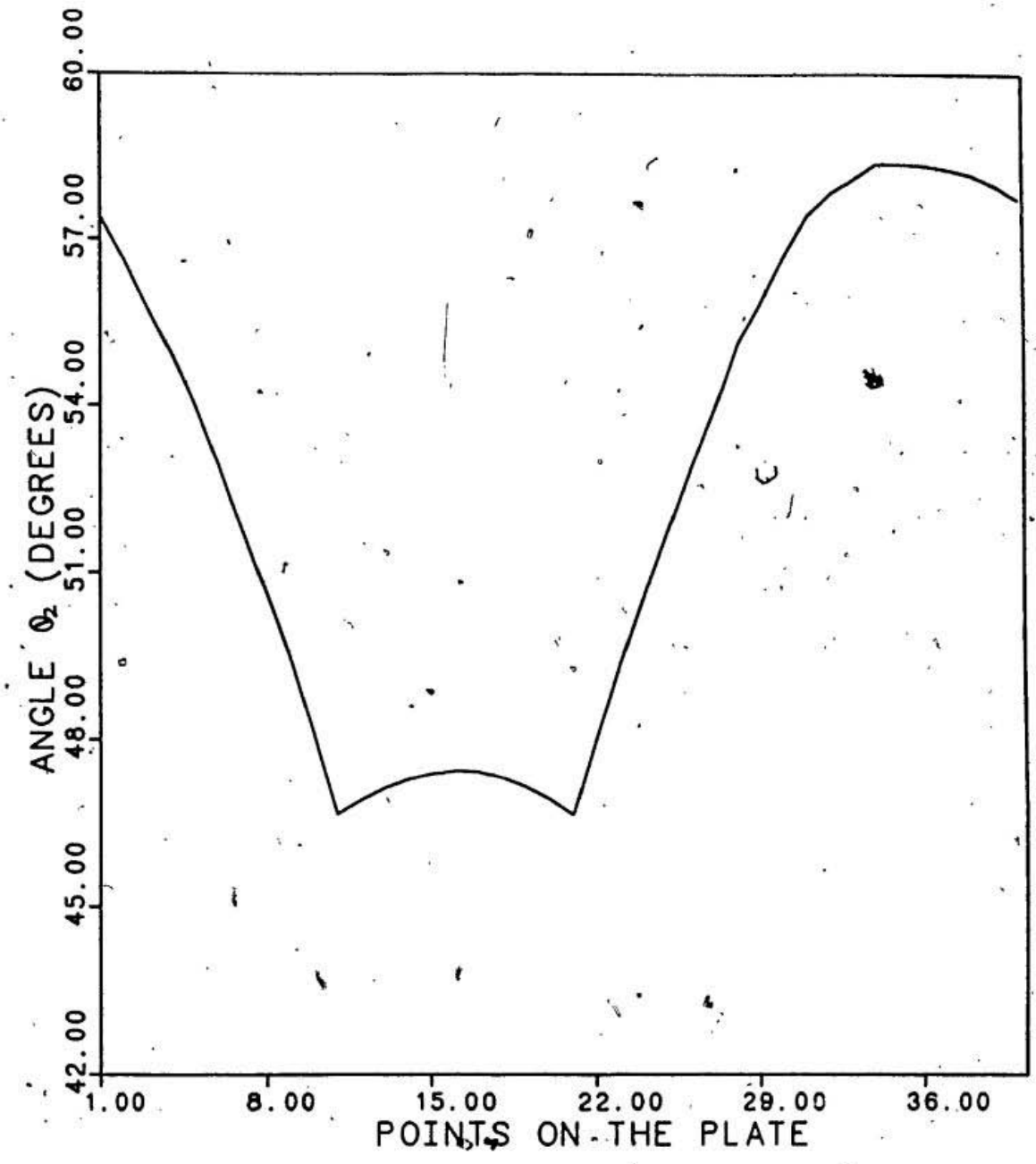

FIG. #2.11: The Variation of θ_2 at Various Points on the Plate for T3R3 Robot

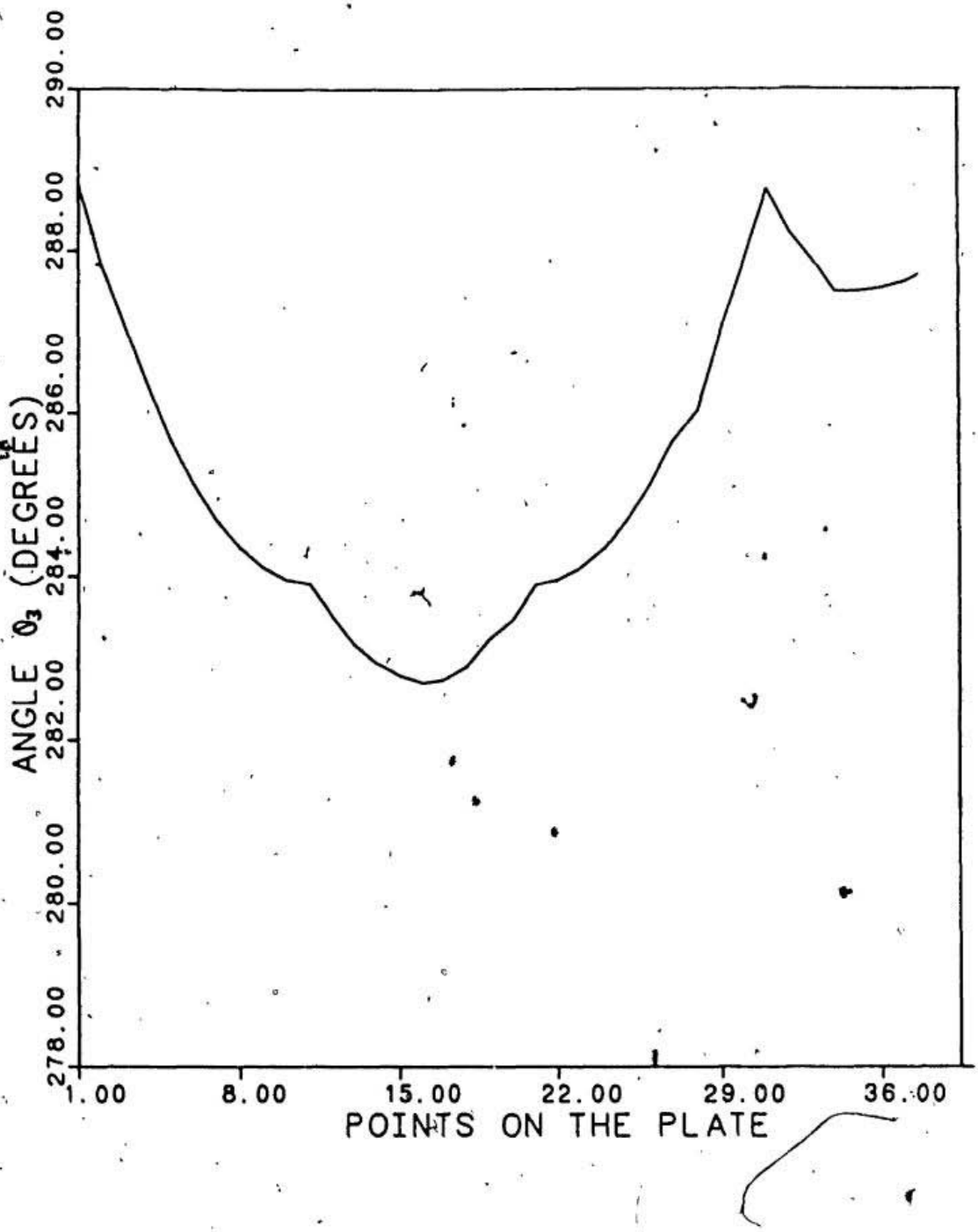

FIG. #2.12: The Variation of θ_3 at Various Points on the Plate for T3R3 Robot

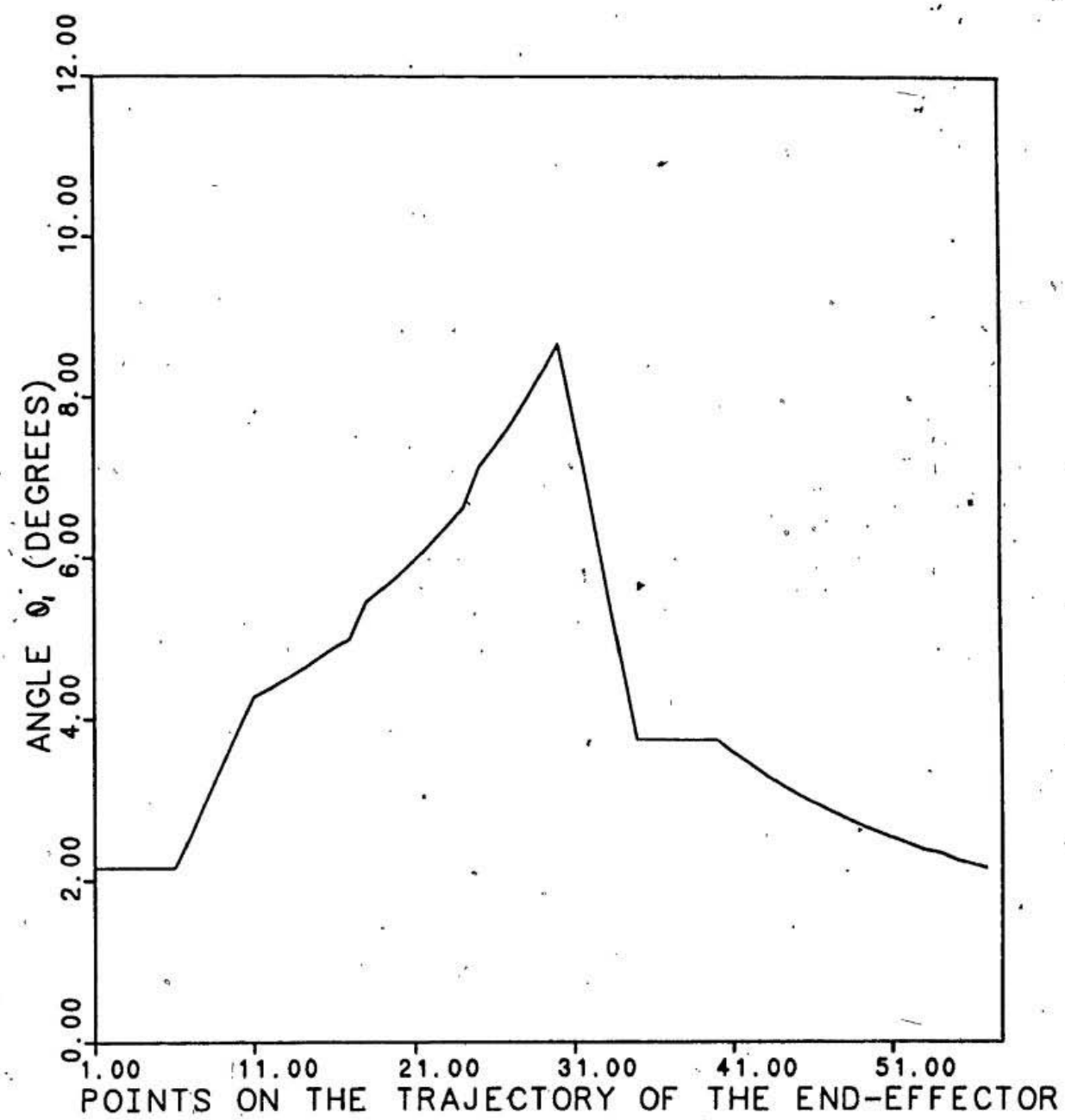

FIG. #2.13: The Variation of θ_1 at Selected Points on the Trajectory of the End-Effector of the T3R3 Robot

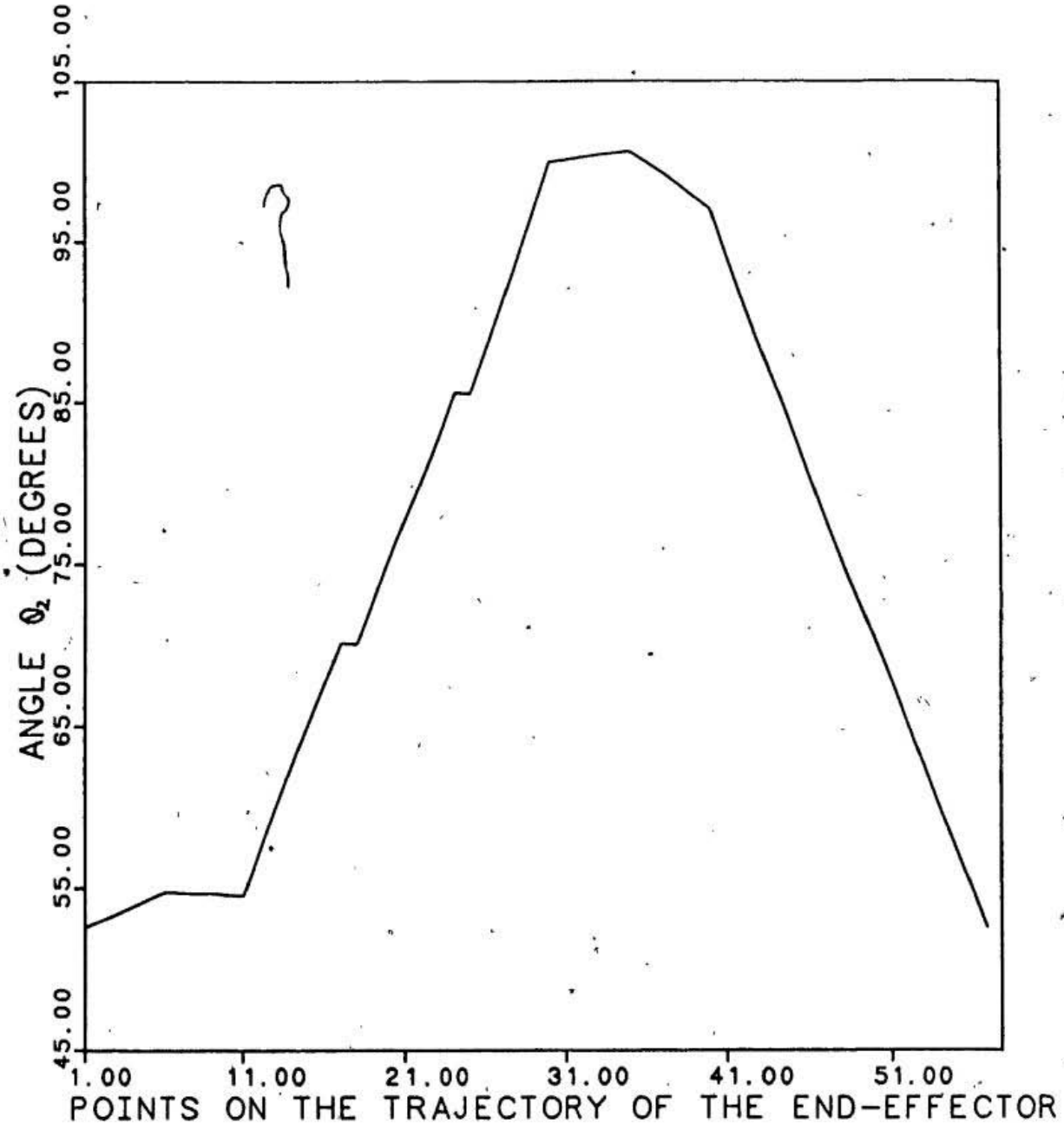

FIG. #2.14: The Variation of θ_2 at Selected Points on the Trajectory of the End-Effector of the T3R3 Robot

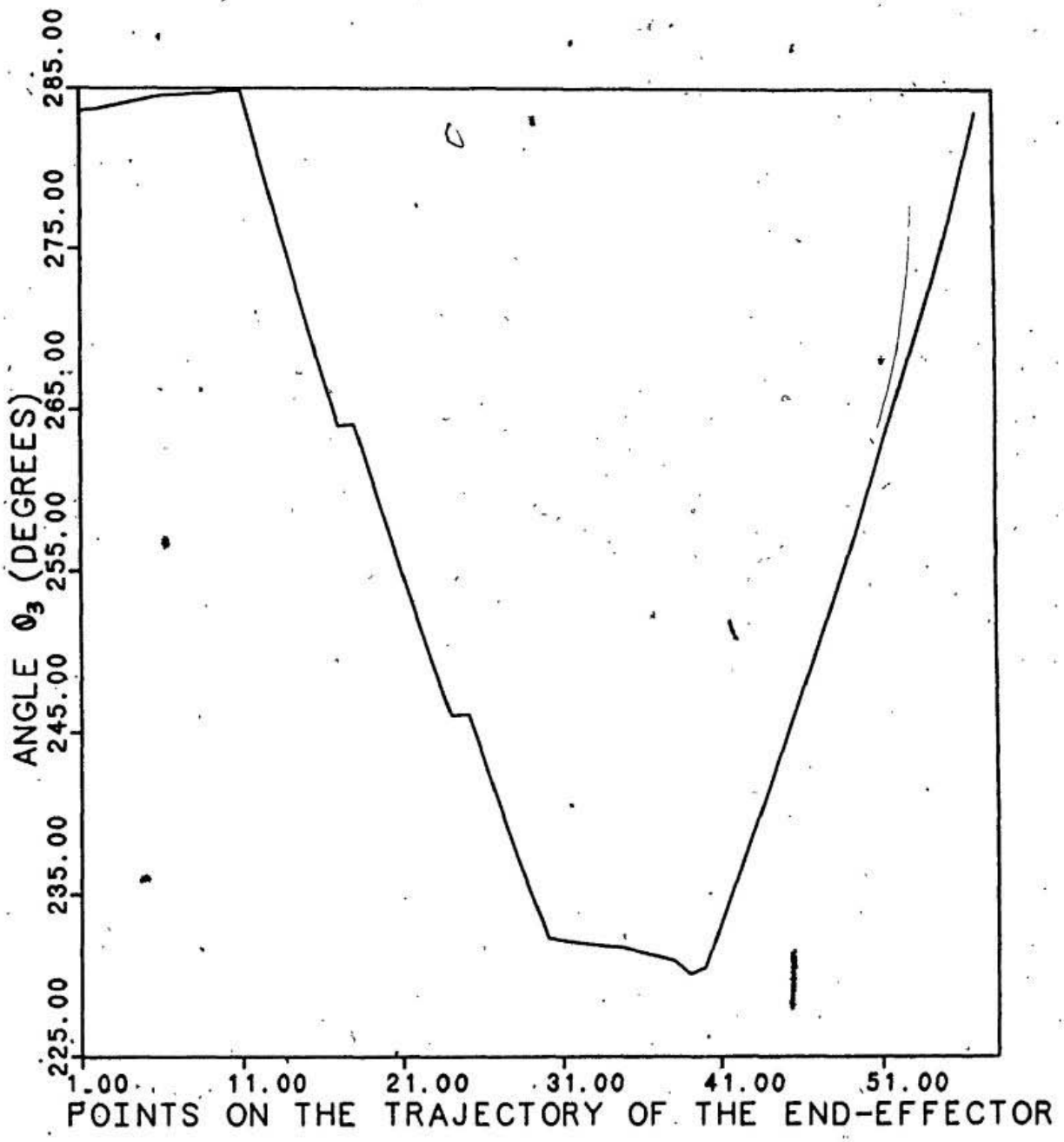

FIG. #2.15 The Variation of θ_3 at Selected Points on the Trajectory of the End-Effector of the T3R3 Robot

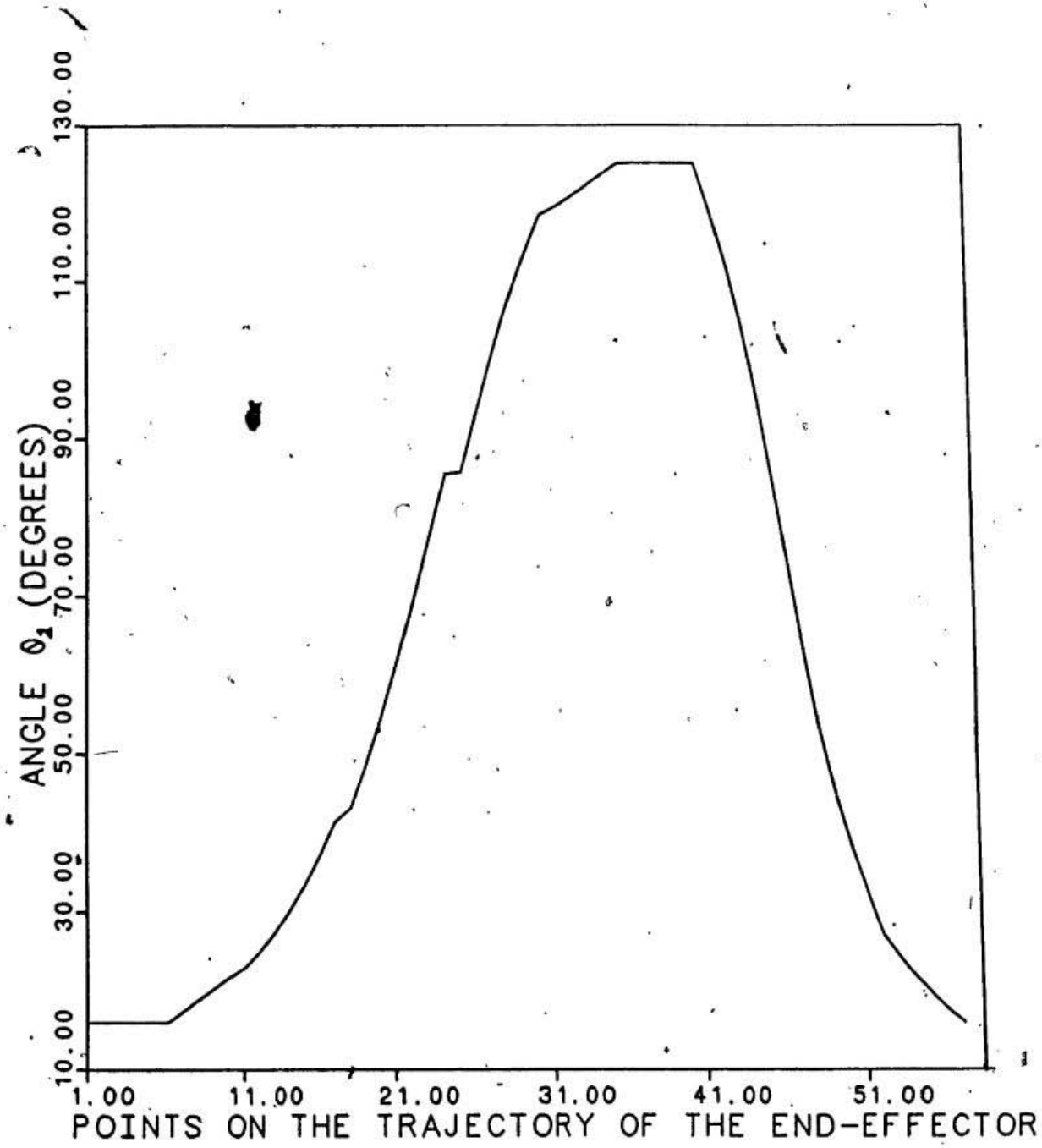

FIG. #2.16: The Variation of θ_1 at Selected Points on the Trajectory of the PUMA-560 Manipulator

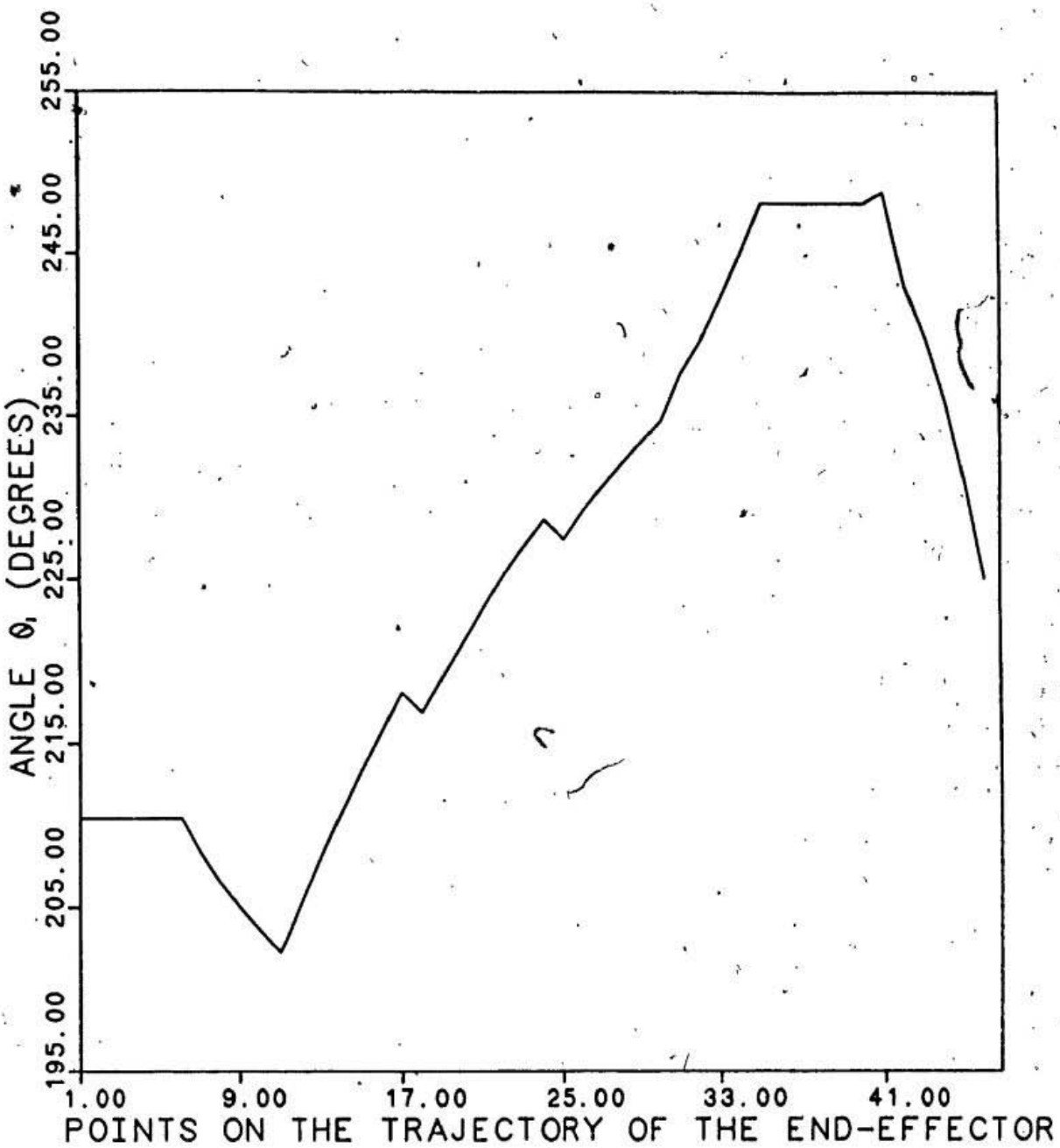

FIG. #2.17: The Variation of θ_1 at Selected Points on the Trajectory of the End-Effector of the Stanford Manipulator

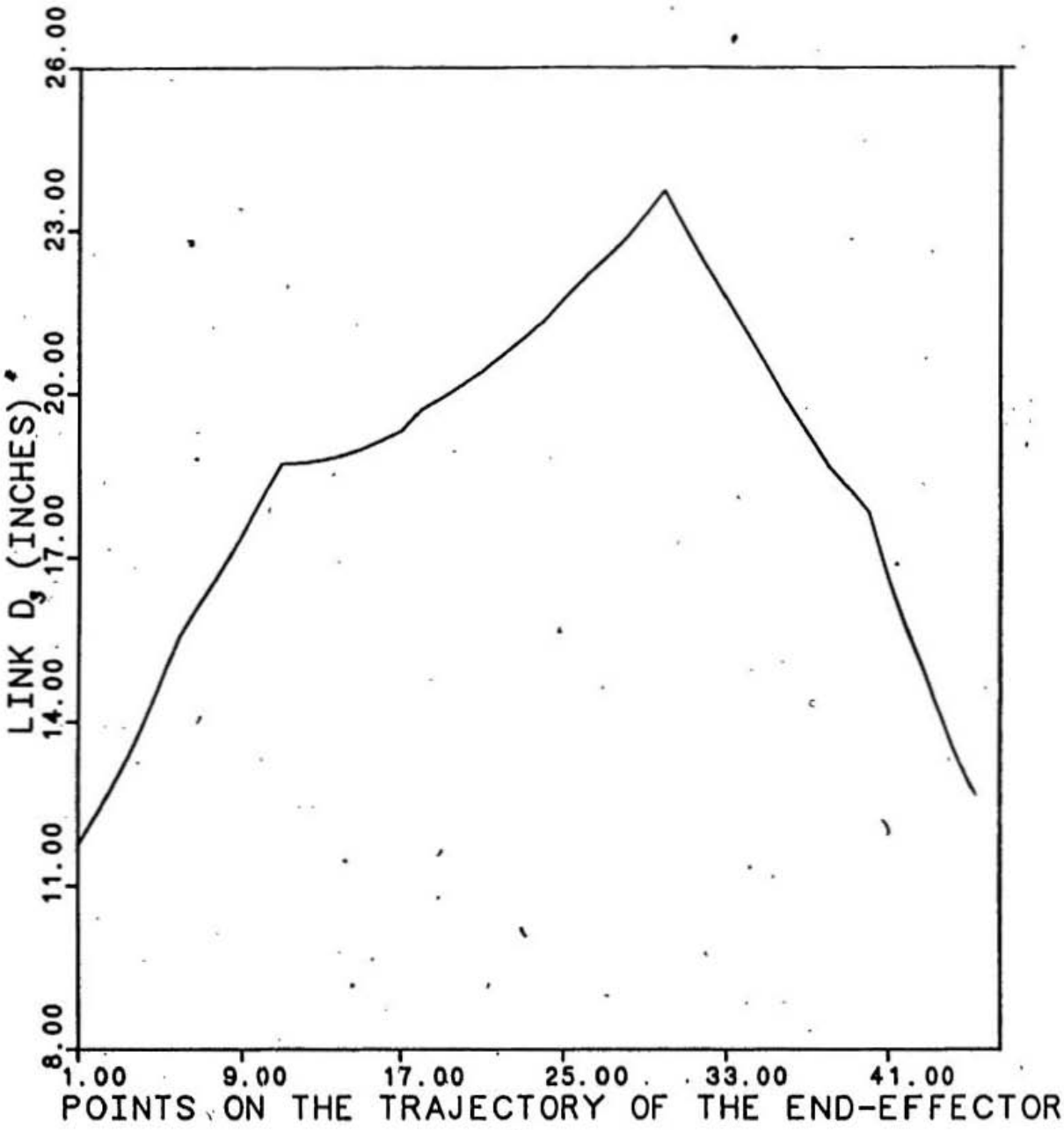

FIG. #2.18: The Variation of D_3 at Selected Points on the Trajectory of the End-Effector of the Stanford Manipulator

TABLE #2.7: CPU Time For One Solution by Various Techniques on Different Manipulators

Type of Robot	Complex Optimization Method* (seconds)	SLA Method $\times 10^3$ (seconds)	LSM Method** $\times 10^3$ (seconds)
T3R3	3	22	35
PUMA-560	3.8	27.5	35
Stanford	2.8	22	35

* The details of this method can be seen in [20].

** The details of this method are given in [18,19].

three methods are shown. It should be noted here that the LSM method does not incorporate any constraint. Clearly, the SLA method is the most efficient among the three. Using the SLA method it took only 0.98 seconds for spot welding of the plate, thus achieving a frequency of approximately 45 solutions per second. Therefore it can be a promising method for on-line systems. Another example involves the solution of all the six joint angles for PUMA 560 manipulator. The homogeneous matrix, describing the position and orientation of the end effector in this case was

$$[A_0^6] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.3 & -0.474 & -0.7892 & -0.6124 \\ 0.2 & 0.6597 & 0.4356 & -0.6124 \\ 0.15 & 0.7500 & -0.4330 & 0.5 \end{bmatrix} \qquad (2.29)$$

CPU time taken by the SLA method was 0.09 seconds and it was 0.14 seconds when LSM was used. As explained earlier, in this example θ_1, θ_2 and θ_3 were obtained first and then the wrist angles θ_4, θ_5, θ_6 were solved. The solution vector obtained (expressed in degrees) was

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} = \begin{bmatrix} 13.06 \\ 40.78 \\ 43.23 \\ 30.71 \\ 63.49 \\ 32.42 \end{bmatrix} \qquad (2.30)$$

Similarly, computations were also carried out for redundant manipula-

tors discussed in previous section. The $[A_0^7]$ for manipulator shown in Fig. 2.5 was

$$[A_0^7] = \begin{bmatrix} 0 & 1 & 0 & 0.20 \\ 1 & 0 & 0 & 0.35 \\ 0 & 0 & -1 & -0.60 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.31}$$

and the solution vector was

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \\ \theta_7 \end{bmatrix} = \begin{bmatrix} 84.211 \\ -12.125 \\ 21.59 \\ 14.19 \\ 76.378 \\ -89.724 \\ -64.60 \end{bmatrix} \tag{2.32}$$

The previous solution vector in this case was

$$\begin{bmatrix} \theta_1^o \\ \theta_2^o \\ \theta_3^o \\ \theta_4^o \\ \theta_5^o \\ \theta_6^o \\ \theta_7^o \end{bmatrix} = \begin{bmatrix} 91.21 \\ -10.99 \\ 16.35 \\ 13.71 \\ 76.796 \\ -89.81 \\ -63.14 \end{bmatrix} \tag{2.33}$$

For modified Stanford manipulator the rotation submatrix was same as given in Eq. (2.31). Mathematically it was represented as

$$[A_0^7] = \begin{bmatrix} 0 & 1 & 0 & 0.1524 \\ -1 & 0 & 0 & 0.0203 \\ 0 & 0 & -1 & -0.1524 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.34}$$

The solution vector obtained was

$$\begin{bmatrix} d_0 \\ \theta_1 \\ \theta_2 \\ d_3 \\ \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} = \begin{bmatrix} 0.248 \\ 205.56 \\ 193.13 \\ 0.417 \\ -64.92 \\ 84.29 \\ -11.94 \end{bmatrix} \tag{2.35}$$

and the previous solution vector here was

$$\begin{bmatrix} d_0^o \\ \theta_1^o \\ \theta_2^o \\ d_3^o \\ \theta_3^o \\ \theta_4^o \\ \theta_5^o \end{bmatrix} = \begin{bmatrix} 0.245 \\ 207.31 \\ 192.032 \\ 0.442 \\ -63.11 \\ 84.43 \\ -10.78 \end{bmatrix} \tag{2.36}$$

2.3 The Constrained Velocity Analysis of Robots of Arbitrary Architecture

2.3.1 Introduction to Velocity Analysis

The sophistication required for the trajectory control of a robotic manipulator is dependent on the complexity of the task to be carried out. This involves the kinematic and dynamic analyses of robotic manipulators. Such a type of analysis requires a complete knowledge of displacement, velocity and acceleration as a function of time in the cartesian space. One of the important aspects in solving these problems is to solve for singularity configuration. This problem arises because at singular points the Jacobian is non-invertible as the rank of the matrix is less than the dimension of the matrix. In addition, if one is dealing with redundant manipulators, then the inverse of the Jacobian is not defined in the regular sense; one has to use the pseudo-inverse technique [45]. In the design of these robots one should also make provisions for additional constraints imposed due to the operational requirements such as workspace limitations, horse-power requirements of the motor, etc.

The objective of the work in this section is to develop a fast and efficient technique for velocity and acceleration analyses of general manipulators subjected to linear and non-linear constraints. In this way one can also handle systems under singularity configurations as well, by imposing constraints on the maximum values of velocities. The kinematic analysis of the redundant type of manipulators can also be performed using the proposed technique.

2.3.2 The Derivation of the Velocity Equations

The equations for obtaining three coordinates of a general point in the case of PUMA-560 manipulator can be written as

$$\begin{aligned} X_0 &= H_3 \cos q_1 \cos(q_2 + q_3) - L_4 \cos q_1 \sin(q_2 + q_3) + \\ &\quad + H_2 \cos q_2 \cos q_1 - L_3 \sin q_1 \\ Y_0 &= H_3 \sin q_1 \cos(q_2 + q_3) - L_4 \sin q_1 \sin(q_2 + q_3) + \\ &\quad + H_2 \cos q_2 \sin q_1 + L_3 \cos q_1 \\ Z_0 &= -H_3 \sin(q_2 + q_3) - L_4 \cos(q_2 + q_3) - H_2 \sin q_2 \end{aligned} \tag{2.37}$$

where q_i are the generalized variables used in place of θ_i. The equations for velocity can be obtained by differentiating the displacement equations i.e. Eq. (2.37) with respect to time. In this way we obtain the equations for velocities as

$$\begin{aligned} \dot{X}_0 &= (-H_3 S q_1 C q_{23} + L_4 S q_1 S q_{23} - H_2 C q_2 S q_1 - L_3 C q_1)\dot{q}_1 \\ &\quad + (-H_3 C q_1 S q_{23} - L_4 C q_1 C q_{23} - H_2 S q_2 C q_1)\dot{q}_2 \\ &\quad + (-H_3 C q_1 S q_{23} - L_4 C q_1 C q_{23})\dot{q}_3 \\ \dot{Y}_0 &= (H_3 C q_1 C q_{23} - L_4 C q_1 S q_{23} + H_2 C q_2 C q_1 - L_3 S q_1)\dot{q}_1 \\ &\quad + (-H_3 S q_1 S q_{23} - L_4 S q_1 C q_{23} - H_2 S q_2 S q_1)\dot{q}_2 \\ &\quad + (-H_3 S q_1 S q_{23} - L_4 S q_1 C q_{23})\dot{q}_3 \end{aligned} \tag{2.38}$$

$$\dot{Z}_0 = 0(q_1) + (-H_3Cq_{23} + L_4Sq_{23} - H_2Cq_2)\dot{q}_2 + (-H_3Cq_{23} + L_4Sq_{23})\dot{q}_3$$

where

$$Cq_i = \cos(q_i)$$

$$Sq_i = \sin(q_i)$$

$$Cq_{ij} = \cos(q_i + q_j)$$

$$Sq_{ij} = \sin(q_i + q_j)$$

Eq. (2.38) can be written in the general form as

$$\begin{aligned} \dot{X}_0 &= a_{11}\dot{q}_1 + a_{12}\dot{q}_2 + \ldots + q_{1N_\ell}\dot{q}_{N_\ell} \\ \dot{Y}_0 &= a_{21}\dot{q}_1 + a_{22}\dot{q}_2 + \ldots + q_{2N_\ell}\dot{q}_{N_\ell} \\ \dot{Z}_0 &= a_{31}\dot{q}_1 + a_{32}\dot{q}_2 + \ldots + q_{3N_\ell}\dot{q}_{N_\ell} \end{aligned} \qquad (2.39)$$

The above velocity equations are linear in nature. Coefficients a_{ij}, which are functions of joint angles, are known after the displacement analyses. Equations, similar to Eq. (2.39), can be obtained for all manipulators by taking the first time derivative of their displacement equations. The detailed expressions of a_{ij} for other manipulators are given in Appendix A. It should be noted here that the proposed methods are general methods and that Eq. (2.39) can be derived for any type of manipulator.

2.4 The Methods of Solution For Velocity Equations

Four general techniques are proposed for the velocity analyses of general manipulators of arbitrary architecture. Depending upon the problem definition, these techniques can efficiently handle various kind of constraints. These are based on

i) Quadratic Programming Principle
ii) Linear Programming Principle
iii) Successive Linear Approximations
iv) Non-linear Complex Optimization

In the next section each technique is discussed in detail.

2.4.1 Velocity Analysis Using Quadratic Programming

For robots of arbitrary architecture, the system of equations, Eq. (2.39) can be under-determined, exact or over-determined i.e. the number of generalized variable q_i can be more than, equal to or less than the number of equations. One of the methods available in the literature to solve such problems is the LSM using Householder transformation [19]. To solve the set of equations for velocities, let us define three residual functions Fv_1, Fv_2 and Fv_3. These functions can be mathematically written as

$$\begin{aligned} Fv_1 &= a_{11}\dot{q}_1 + a_{12}\dot{q}_2 + a_{13}\dot{q}_3 + \ldots + a_{1N_\ell}\dot{q}_{N_\ell} - \dot{X}_0 \\ Fv_2 &= a_{21}\dot{q}_1 + a_{22}\dot{q}_2 + a_{23}\dot{q}_3 + \ldots + a_{2N_\ell}\dot{q}_{N_\ell} - \dot{Y}_0 \\ Fv_3 &= a_{31}\dot{q}_1 + a_{32}\dot{q}_2 + a_{33}\dot{q}_3 + \ldots + a_{3N_\ell}\dot{q}_{N_\ell} - \dot{Z}_0 \end{aligned} \qquad (2.40)$$

In the next step we replace the generalized variable $\dot{q}_i$ by two variables called restricted variables because these can take only positive values, whereas, $\dot{q}_i$ are nonrestricted variables as they can take both positive or negative values. The equation relating all these variables can be written as

$$\dot{q}_i = v_{2i-1} - v_{2i} \qquad i = 1, 2, 3, \ldots, N_\ell \,, \tag{2.41}$$

$$\text{where } v_j \geq 0, \; j = 1, 2, \ldots, N_\ell, \ldots, 2N_\ell$$

To solve Eq. (2.39) let us define a function U such that

$$U = Fv_1^2 + Fv_2^2 + Fv_3^2 \tag{2.42}$$

Combining Eqs. (2.40) and (2.42) one can express U in terms of v_i. For $N_\ell = 3$, the objective function will be

$$\begin{aligned} U = {} & P_1v_1^2 + P_1v_2^2 + P_2v_3^2 + P_2v_4^2 + P_3v_5^2 + P_3v_6^2 + \\ & + T_1(v_1v_2) + T_2(v_3v_4) + T_3(v_5v_6) + T_4(v_1v_3) - T_4(v_1v_4) + \\ & + T_4(v_2v_4) + T_5(v_3v_5) - T_5(v_3v_6) - T_5(v_5v_4) + \\ & + T_5(v_4v_6) + T_6(v_1v_5) - T_6(v_1v_6) - T_6(v_2v_5) + \\ & + T_6(v_2v_6) - \omega_1v_1 + \omega_1v_2 - \omega_2v_3 + \omega_2v_4 - \omega_3v_5 + \\ & + \omega_3v_6 + K_1 \end{aligned} \tag{2.43}$$

In this equation P_i and T_i are functions of a_{ij}, therefore after the displacement analysis these can be considered as constants. Here K_1 is a function of a_{ij}, and end-effector cartesian velocity

components; therefore a constant for a given end-effector displacement and velocity conditions. The various constant terms in Eq. (2.43) are expressed below:

$$P_1 = a_{11}^2 + a_{21}^2 + a_{31}^2$$

$$P_2 = a_{12}^2 + a_{22}^2 + a_{32}^2 \qquad (2.44)$$

$$P_3 = a_{13}^2 + a_{23}^2 + a_{33}^2$$

$$T_1 = -2P_1$$

$$T_2 = -2P_2$$

$$T_3 = -2P_3 \qquad (2.45)$$

$$T_4 = 2(a_{11}a_{12} + a_{21}a_{22} + a_{31}a_{32})$$

$$T_5 = 2(a_{12}a_{13} + a_{22}a_{23} + a_{32}a_{33})$$

$$T_6 = 2(a_{11}a_{13} + a_{21}a_{23} + a_{31}a_{33})$$

$$\omega_1 = 2(Kv_1a_{11} + Kv_2a_{21} + Kv_3a_{31})$$

$$\omega_2 = 2(Kv_1a_{12} + Kv_2a_{22} + Kv_3a_{32}) \qquad (2.46)$$

$$\omega_3 = 2(Kv_1a_{13} + Kv_2a_{23} + Kv_3a_{33})$$

$$Kv_1 = \dot{X}_0$$

$$Kv_2 = \dot{Y}_0$$

$$Kv_3 = \dot{z}_0$$

$$K_1 = Kv_1^2 + Kv_2^2 + Kv_3^2 \qquad (2.47)$$

For a given end-effector velocity profile we can rewrite Eq. (2.43) in the form

$$U = \sum_{j=1}^{6} G_j v_j + \frac{1}{2} \sum_{j=1}^{6} \sum_{k=1}^{6} Q_{jk} v_j v_k \qquad (2.48)$$

If we replace the generalized variable q_i using Eq. (2.41), then velocity equation, Eq. (2.40), can be expressed in the form

$$\sum_{j=1}^{6} A'_{ij} v_j = B'_j , \qquad i = 1, 2, 3$$

$$v_j \geq 0 \qquad j = 1, 2, \ldots, 6 \qquad (2.49)$$

Eqs. (2.48) and (2.49) can be written in the matrix form as Eqs. (2.50) and (2.51) respectively. These equations are:

$$U = \{G\}^T\{v\} + \frac{1}{2}\{v\}^T[Q]\{v\} \qquad (2.50)$$

subject to constraints

$$[A']\{v\} = \{B'\} \qquad (2.51)$$

$$\{v\} \geq \{0\} \qquad (2.52)$$

The expression for quadratic coefficient matrix is

$$[Q] = \begin{bmatrix} 2P_1 & T_1 & T_4 & -T_4 & T_6 & -T_6 \\ T_1 & 2P_1 & -T_4 & T_4 & -T_6 & T_6 \\ T_4 & -T_4 & 2P_2 & T_2 & T_5 & -T_5 \\ -T_4 & T_4 & T_2 & 2P_2 & -T_5 & T_5 \\ T_6 & -T_6 & T_5 & -T_5 & 2P_3 & T_3 \\ -T_6 & T_6 & -T_5 & T_5 & T_3 & 2P_3 \end{bmatrix} \tag{2.53}$$

$\{G\}$ in Eq. (2.50) includes linear coefficient terms and is given by

$$\{G\}^T = \{-\omega_1, \omega_1, -\omega_2, \omega_2, -\omega_3, \omega_3\} \tag{2.54}$$

If we look at the Eq. (2.43), P_i, T_i and ω_i are constants. Therefore the terms occurring in this equation are of two types; the first type are the linear terms having ω_i as the coefficients and the rest are quadratic in nature. The problem given now is to minimize the objective function U given by Eq. (2.50) subject to constraints expressed by Eqs. (2.51) and (2.52). This is a quadratic programming problem and the method of solution can be seen in any text such as [44]. In addition to constraint equations, Eq. (2.51), we can also have some other linear constraints imposed due to the operating conditions or from other sources. Let us write these as

$$[A'']\{v\} \leq \{B''\} \tag{2.55}$$

Using the slack variable vector $\{S_\ell\}$, we can convert these inequality constraint into equality constraint as

$$\underset{n\times(6+n)}{\left[\begin{array}{c|c}[A''] & [I]\end{array}\right]} \left\{\begin{array}{c} v_1 \\ \vdots \\ v_6 \\ \hline S_1 \\ \vdots \\ S_n \end{array}\right\}_{(6+n)\times 1} = \underset{n\times 1}{\{B''\}} \qquad (2.56)$$

To solve this problem we combine Eqs. (2.51) and (2.56) as

$$\underset{(n+3)\times(6+n)}{\left[\begin{array}{c|c}[A'] & 0 \\ \hline [A''] & [I]\end{array}\right]} \left\{\begin{array}{c} v_1 \\ \vdots \\ v_6 \\ \hline S_1 \\ \vdots \\ S_n \end{array}\right\}_{(6+n)\times 1} = \left\{\begin{array}{c} B' \\ \hline B'' \end{array}\right\} = \underset{(n+3)\times 1}{\{B\}}$$

or

$$[A]\{V\} = \{B\} \qquad (2.57)$$

where $\{V\}^T = \{v \ \ldots \ s_\ell\}^T$.

We can also convert Eq. (2.52) into an equality form by including surplus variables as

$$-\{v\} + \{s_t^2\} = \{0\} \qquad (2.58)$$

To solve this minimization problem one can use the Lagrange multiplier technique.

Let L be the Lagrange function, then the objective function can be written as

$$L(\{v\},\{s_\ell\},\{s_t\},\{\lambda\},\{\xi\}) = \{G^T\}\{v\} + \frac{1}{2}(\{v^T\}[Q]\{v\}) + \{\lambda^T\}([A]\{v\} - \{B\}) + \{\xi\}^T(-\{v\} + \{s_t^2\}) \quad (2.59)$$

To minimize the Lagrange function one has to differentiate L with respect to all the variables including λ_i and ξ_i, the Lagrangian multipliers. The necessary conditions as the result of differentiation can be written as

$$\{G\} - \{\xi\} + [Q]\{v\} + [A^T]\{\lambda\} = \{0\} \quad (2.60)$$

$$\lambda_i E_i \geq 0 , \qquad i = 1, 2, 3 \quad (2.61)$$

$$\xi_j v_j \geq 0 , \qquad j = 1, 2, \ldots, 6 \quad (2.62)$$

$$[A^T]\{v\} + \{s_\ell^2\} - \{B\} = \{0\} , \qquad i = 1, 2, 3 . \quad (2.63)$$

$$-\{v\} + \{s_t^2\} = \{0\} , \qquad j = 1, 2, \ldots, 6 \quad (2.64)$$

where

$$\{v\},\ \{\xi\},\ \{E\},\ \{\lambda\} \geq \{0\}$$

and $\{E\}$ is a new set of variables expressed as

$$\{E\} = \{s_i^2\} \geq 0$$

In the equations above, all the equations except Eq. (2.61) and (2.62) are linear. We have a convex objective function, thereby indicating that any local minimum is the global minimum. Also the number of variables become equal to number of equations. Thus any solution which is found to be feasible will be optimum feasible solution. Such types of problem can be solved using phase I of the simplex algorithm by defining the objective function of the form

$$U' = \sum_{j=1}^{6} \alpha_j \tag{2.65}$$

subject to constraints

$$\{G\} - \{\xi\} + [Q]\{v\} + [A^T]\{\lambda\} + \{\alpha\} = \{0\} \tag{2.66}$$

where α_j are non-negative artificial variables.

While solving this problem only phase I computations are required and care is taken regarding the nonlinear constraints given by Eq. (2.61) and (2.62) such that either λ_i or E_i are in the basis. Similarly either ξ_i or v_i should be in the basis.

2.4.2 The Velocity Analysis Using Linear Programming

The linear programming can also be used to solve the velocity problem by expressing the objective function as

$$U = Fv_1 + Fv_2 + Fv_3 \tag{2.67}$$

subject to constraints given by Eq. (2.57). The advantage of expressing the objective function in this form is that there are no non-linear constraints as given by Eq. (2.61) and Eq. (2.62). The other details are similar to the previous method where the simplex technique was used to find optimum. The only difference here is that one has to carry out phase II computations of the simplex method. Additional constraints given by Eq. (2.55) are handled in the same way as before.

2.4.3 Successive Linear Approximation Principle

In the previous two cases the additional constraints imposed due to the robot workspace limitation etc. were linear. If these constraints are non-linear in nature then the previous two methods cannot be used. In that case, one can solve this problem by successively linearizing the non-linear constraint equations and solving the problem using phase II of simplex algorithm at each stage. For example we can linearize the ith non-linear constraint, F_{v_i}, and express it in the linearized form as

$$\begin{aligned} F^{\ell}_{v_i} = {} & \dot{q}_1(\partial F^*_{v_i}/\partial q_1) + \dot{q}_2(\partial F^*_{v_i}/\partial q_2) + \dot{q}_3(\partial F^*_{v_i}/\partial q_3) \\ & - \{\dot{q}^*_1(\partial F^*_{v_i}/\partial \dot{q}_1) + \dot{q}^*_2(\partial F^*_{v_i}/\partial \dot{q}_2) \\ & + \dot{q}^*_3(\partial F^*_{v_i}/\partial \dot{q}_3) - F^*_{v_i}\} \end{aligned} \tag{2.68}$$

where $\{\dot{q}_i^*\}$ are the starting vectors and $F_{v_i}^*$ is the value of function at $\{\dot{q}_i^*\}$. To illustrate this point further, consider a non-linear constraint, $F_{v_i}^{\ell}$, as

$$F_{v_i}^{\ell} = \dot{q}_1^2 + \dot{q}_2^2 + \dot{q}_3^2 \leq 4 \times 10^{-4} \tag{2.69}$$

Now let us choose the starting values as $v_1^* = v_2^* = v_3^* = v_5^* = 0$, $v_4^* = 4 \times 10^{-3}$, and $v_6^* = 1 \times 10^{-3}$. This will result in $\dot{q}_1^* = 0.0$, $\dot{q}_2^* = -4 \times 10^{-3}$, and $\dot{q}_3^* = -1 \times 10^{-3}$.

Then the value of the linearized function, $F_{v_i}^{\ell}$, will be

$$F_{v_i}^{\ell} = \dot{q}_1(0) + \dot{q}_2(-8 \times 10^{-3}) + \dot{q}_3(-2 \times 10^{-3}) \leq 4.17 \times 10^{-4}$$

Similarly, by linearizing each of the additional constraints, we can form a new set of these linearized equation which we can call as

$$[A^m]\{V\} = \{B^m\} \tag{2.70}$$

This has to be combined with Eq. (2.55) and these equations can be solved by either of the two methods mentioned above.

2.5 The Constrained Acceleration Analysis of Robots of Arbitrary Architecture

The acceleration analysis follows on the similar concepts as developed in the previous section for velocity analysis. Differentiating Eq. (2.38) with time we obtain the equations for acceleration for PUMA-560 as

$$\ddot{X}_0 = (-H_3Sq_1Cq_{23} + L_4Sq_1Sq_{23} - H_2Cq_2Sq_1 - L_3Cq_1)\ddot{q}_1$$
$$+ (-H_3Cq_1Sq_{23} - L_4Cq_1Cq_{23} - H_2Sq_2Cq_1)\ddot{q}_2$$
$$+ (-H_3Cq_1Sq_{23} - L_4Cq_1Cq_{23})\ddot{q}_3 + 2(H_3Sq_1Sq_{23}$$
$$+ L_4Sq_1Cq_{23} + H_2Sq_2Sq_1)\dot{q}_1\dot{q}_2 + 2(-H_2Cq_1Cq_{23}$$
$$+ L_4Cq_1Sq_{23})\dot{q}_3\dot{q}_2 + 2(H_3Sq_1Sq_{23} + L_4Sq_1Cq_{23})\dot{q}_3\dot{q}_1$$
$$+ (-H_3Cq_1Cq_{23} + L_4Cq_1Sq_{23} - H_2Cq_2Cq_1 + L_3Sq_1)\dot{q}_1^2$$
$$+ (-H_3Cq_1Cq_{23} + L_4Cq_1Sq_{23} - H_2Cq_2Cq_1)\dot{q}_2^2$$
$$+ (-H_3Cq_1Cq_{23} + L_4Cq_1Sq_{23})\dot{q}_3^2$$

$$\ddot{Y}_0 = (H_3Cq_1Cq_{23} - L_4Cq_1Sq_{23} + H_2Cq_2Cq_1 - L_3Sq_1)\ddot{q}_1$$
$$+ (-H_3Sq_1Sq_{23} - L_4Sq_1Cq_{23} - H_2Sq_2Sq_1)\ddot{q}_2$$
$$+ (-H_3Sq_1Sq_{23} - L_4Sq_1Cq_{23})\ddot{q}_3 + 2(-H_3Cq_1Sq_{23}$$
$$- L_4Cq_1Cq_{23} - H_2Sq_2Cq_1)\dot{q}_1\dot{q}_2 + 2(-H_3Sq_1Cq_{23}$$
$$+ L_4Sq_1Sq_{23})\dot{q}_3\dot{q}_2 + 2(-H_3Cq_1Sq_{23} - L_4Cq_1Cq_{23})\dot{q}_3\dot{q}_1$$
$$+ (-H_3Sq_1Cq_{23} + L_4Sq_1Sq_{23} - H_2Cq_2Sq_1 - L_3Cq_1)\dot{q}_1^2$$
$$+ (-H_3Sq_1Cq_{23} + L_4Sq_1Sq_{23} - H_2Cq_2Sq_1)\dot{q}_2^2$$
$$+ (-H_3Sq_1Cq_{23} + L_4Sq_1Sq_{23})\dot{q}_3^2 \qquad (2.71)$$

$$\ddot{Z}_0 = (0)\ddot{q}_1 + (-H_3Cq_{23} + L_4Sq_{23} - H_2Cq_2)\ddot{q}_2$$
$$+ (-H_3Cq_{23} + L_4Sq_{23})\ddot{q}_3 + 2(H_3Sq_{23} + L_4Cq_{23})\dot{q}_3\dot{q}_2$$
$$+ (H_3Sq_{23} + L_4Cq_{23} + H_2Sq_2)\dot{q}_2^2 + (H_3Sq_{23} + L_4Cq_{23})\dot{q}_3^2$$

For a manipulator with arbitrary architecture Eq. (2.71) can be expressed in the standard form as

$$\begin{aligned}
\ddot{X}_0 &= a_{11}\ddot{q}_1 + a_{12}\ddot{q}_2 + a_{13}\ddot{q}_3 + \ldots + a_{1N_\ell}\ddot{q}_{N_\ell} + C_1 \\
\ddot{Y}_0 &= a_{21}\ddot{q}_1 + a_{22}\ddot{q}_2 + a_{23}\ddot{q}_3 + \ldots + a_{2N_\ell}\ddot{q}_{N_\ell} + C_2 \\
\ddot{Z}_0 &= a_{31}\ddot{q}_1 + a_{32}\ddot{q}_2 + a_{33}\ddot{q}_3 + \ldots + a_{3N_\ell}\ddot{q}_{N_\ell} + C_3
\end{aligned} \qquad (2.72)$$

where C_i are constants as they are functions of joint angles and joint velocities. The expressions for C_i for T3R3 manipulators and Stanford manipulator is given in Appendix A.

Let us define three residual functions Fa_1, Fa_2 and Fa_3 which are mathematically expressed as

$$\begin{aligned}
Fa_1 &= a_{11}\ddot{q}_1 + a_{12}\ddot{q}_2 + a_{13}\ddot{q}_3 + \ldots + a_{1N_\ell}\ddot{q}_{N_\ell} - \ddot{X}_0 + C_1 \\
Fa_2 &= a_{21}\ddot{q}_1 + a_{22}\ddot{q}_2 + a_{23}\ddot{q}_3 + \ldots + a_{2N_\ell}\ddot{q}_{N_\ell} - \ddot{Y}_0 + C_2 \\
Fa_3 &= a_{31}\ddot{q}_1 + a_{32}\ddot{q}_2 + a_{33}\ddot{q}_3 + \ldots + a_{3N_\ell}\ddot{q}_{N_\ell} - \ddot{Z}_0 + C_3
\end{aligned} \qquad (2.73)$$

We can write the objective function for the acceleration analysis as

$$U = Fa_1^2 + Fa_2^2 + F_3^2 \qquad (2.74)$$

which is along the parallel lines as given by Eq. (2.42). Using the concept of restricted variables, we can replace the generalized variables $\ddot{q}_i$ by restricted variables given by equation

$$\ddot{q}_i = a_{2i-1} - a_{2i}, \qquad i = 1, 2, 3 \tag{2.75}$$

$$\text{where } a_j \geq 0, \qquad j = 1, 2, \ldots, 6$$

The $\{B\}$ vector for acceleration analysis will be

$$\{B^T\} = \{Ka_1 \quad Ka_2 \quad Ka_3\}$$

$$\text{where } Ka_1 = C_1 - \ddot{X}_0$$

$$Ka_2 = C_2 - \ddot{Y}_0$$

$$Ka_3 = C_3 - \ddot{Z}_0$$

and

$$K_1 = Ka_1^2 + Ka_2^2 + Ka_3^2$$

The rest of the procedure is same as developed for the velocity analysis.

2.6 Results and Discussions of Velocity and Acceleration Analyses

The velocity and acceleration analyses of T3R3 and PUMA 560 manipulator were carried out while performing a spot welding job around a plate as shown in Fig. (2.8). The velocity and acceleration profiles corresponding to the end-effector's movement between points 1

and 11 shown in Figs. (2.19) and (2.20) respectively. These two robots were shown in Figs. (2.2) and (2.4a-2.4b) and link parameters were given in Tables 2.1 and 2.2 respectively. Eleven points were selected on the trajectory of the plate. The results in the joint space for these robots are shown in Figs. (2.21-2.28). Corresponding to this task, $\dot{\theta}_1$ and $\ddot{\theta}_1$ variations were found to be zero as θ_1 was constant. In these figures, there are very sharp variations in the magnitude of all the variables. The results show that even though the velocity and acceleration profiles in the cartesian space are simple shapes but the calculated joint space variables are of quite complicated shapes. This is because of the nonlinear relationship existing between these two types of variables. Secondly, these sharp variations will pose very stringent conditions in the design of the control systems.

Table 2.8 shows CPU times taken for velocity analysis of T3R3 robot while using these methods. here were no additional constraints such as given by Eq. (2.55) in these results. One can see that the CPU times taken for the LSM and the simplex method are same but the advantage in using the simplex method is the possibility of handling linear and non-linear constraints. Time taken using quadratic programming is slightly more than the other two methods. The present results show that linear programming method can be successfully used in the velocity and acceleration analyses. Since this method can handle linear and nonlinear constraints, it is possible to solve advance problems in trajectory path planning such as obstacle avoidance and singularities

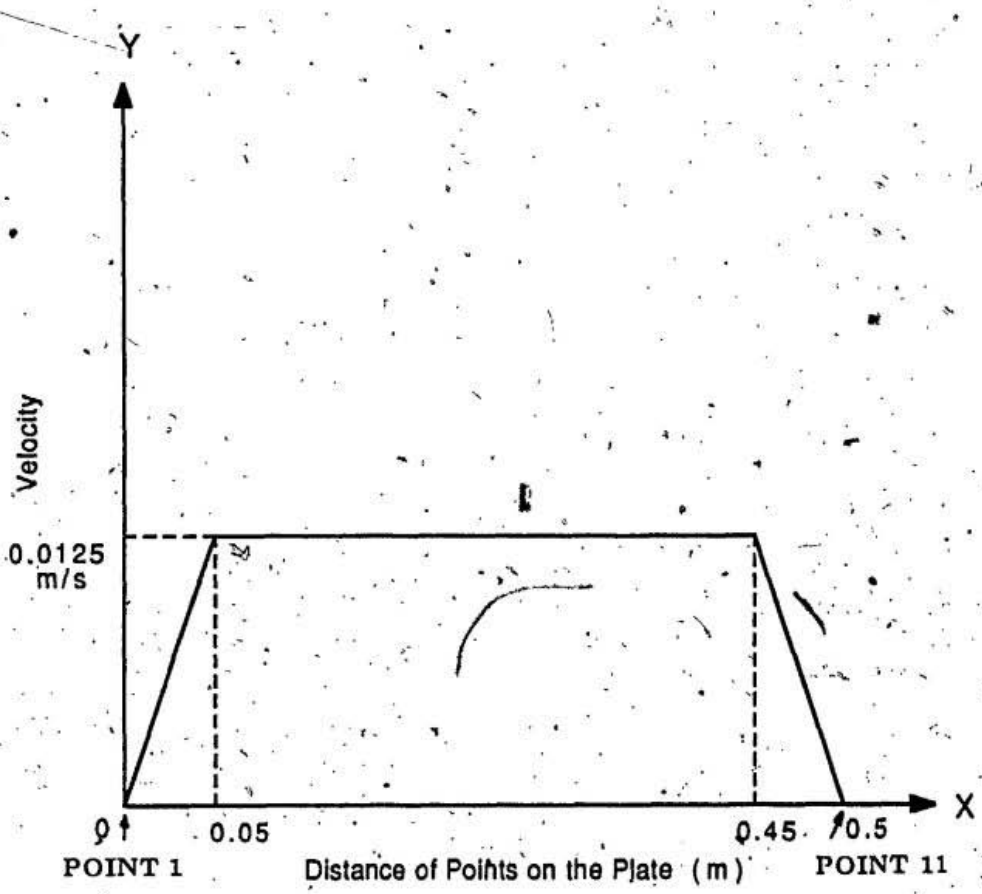

FIG. #2.19: Velocity Profile of the End-Effector while Traversing the Trajectory of the Plate

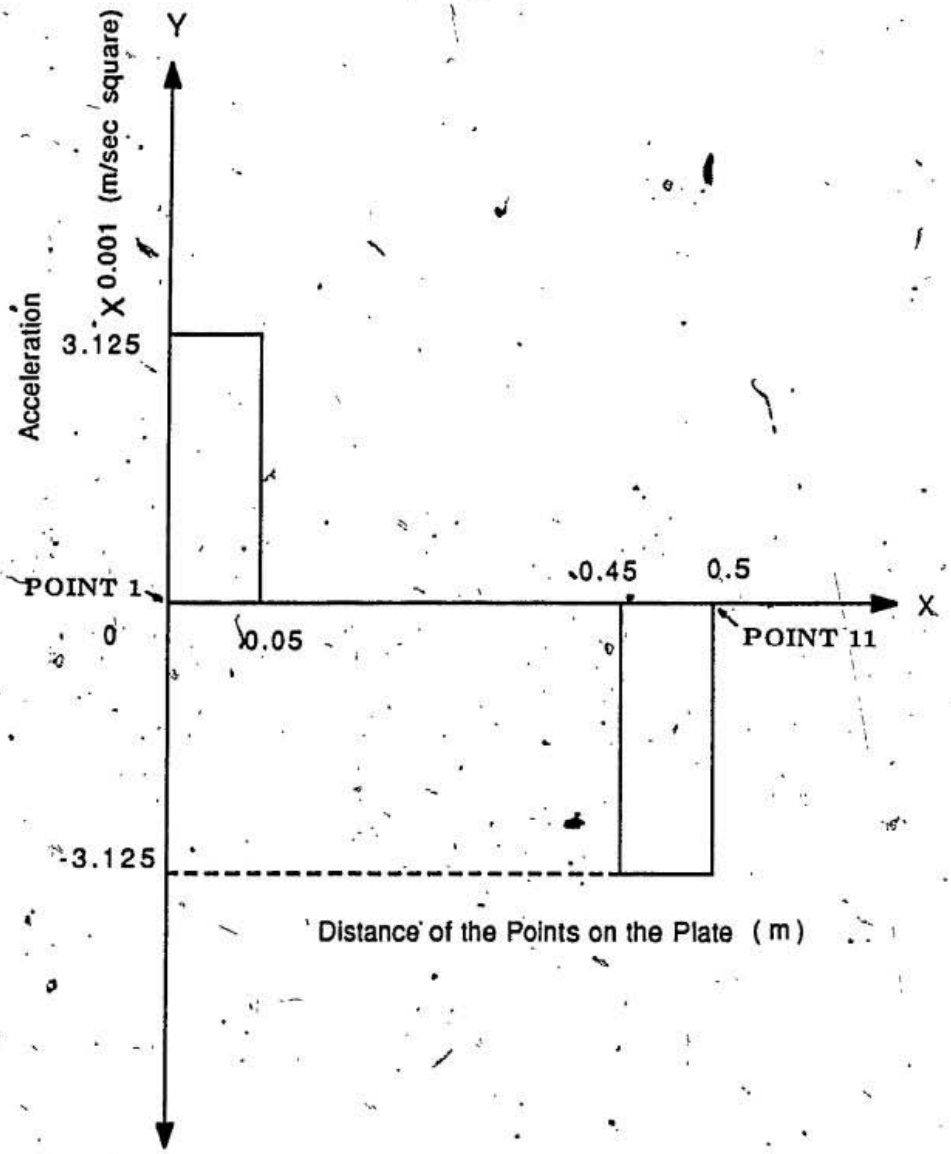

FIG. # 2.20: Acceleration Profile of the End-Effector while Traversing the Trajectory of the Plate

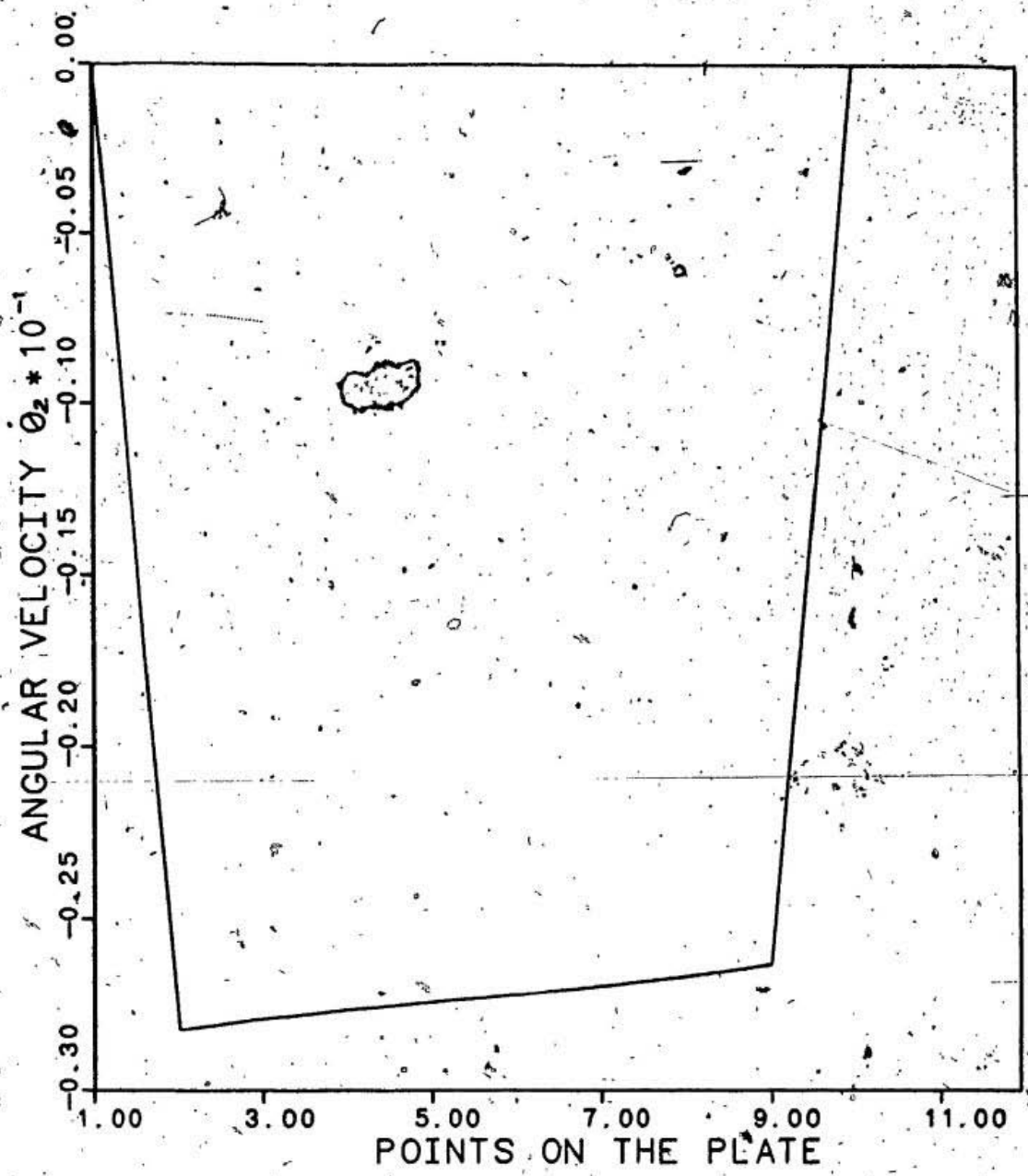

FIG. #2.21: The Variation of $\dot{\theta}_2$ at Various Points on the Plate for PUMA-560 Manipulator

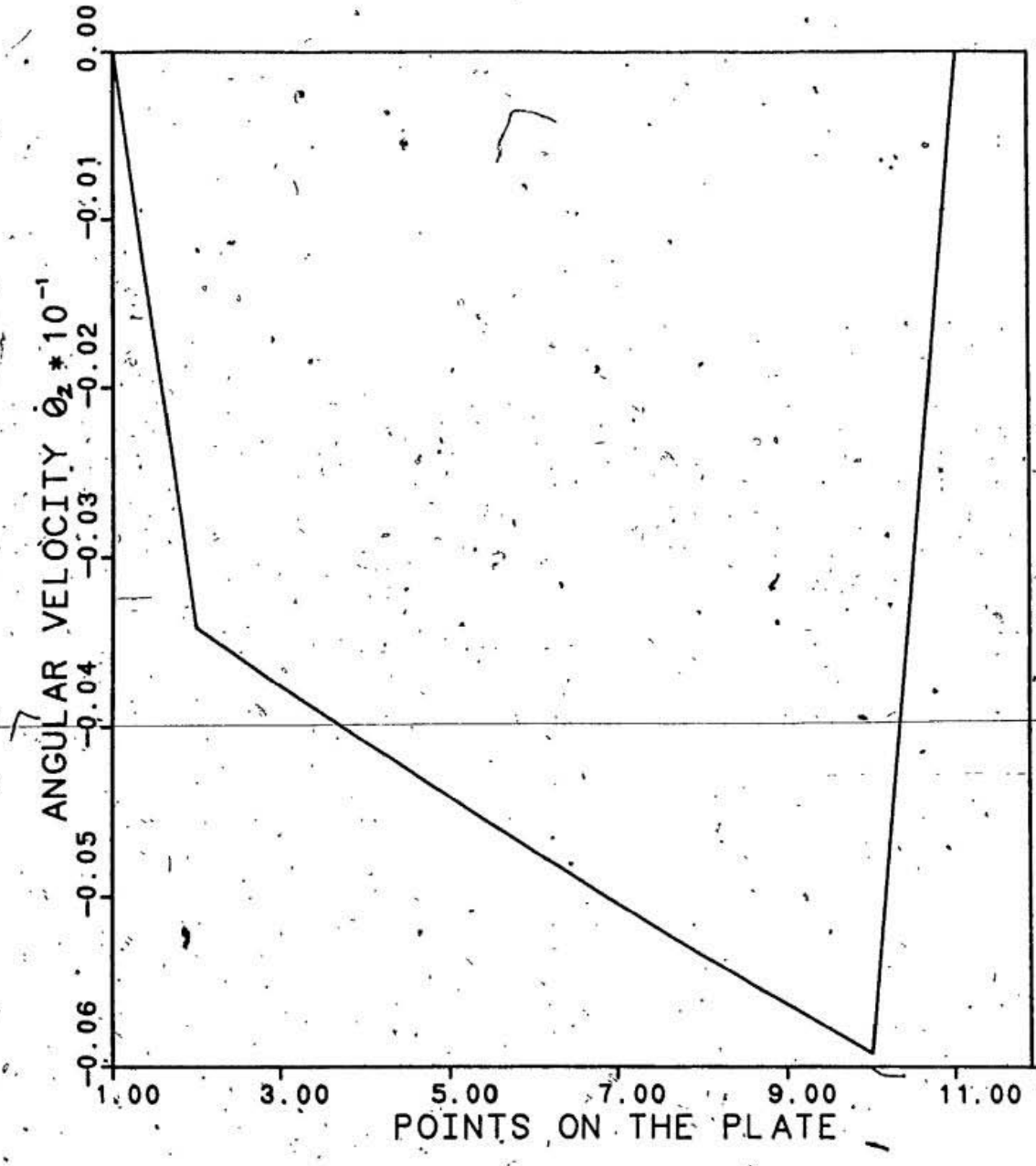

FIG. #2.22: The Variation of $\dot{\theta}_2$ at Various Points on the Plate for T3R3 Robot

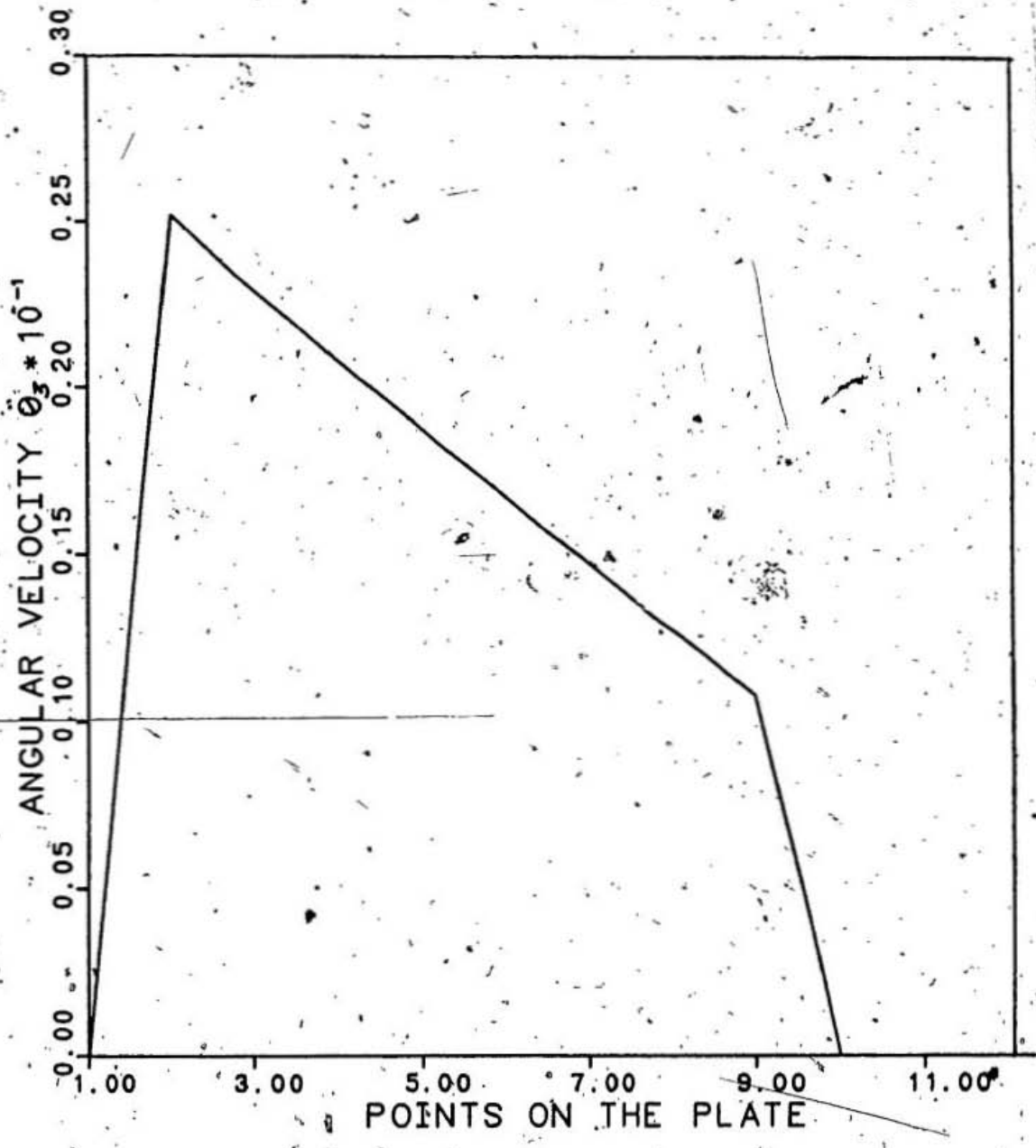

FIG. #2.23: The Variation of $\dot{\theta}_3$ at Various Points on the Plate for PUMA-560 Manipulator

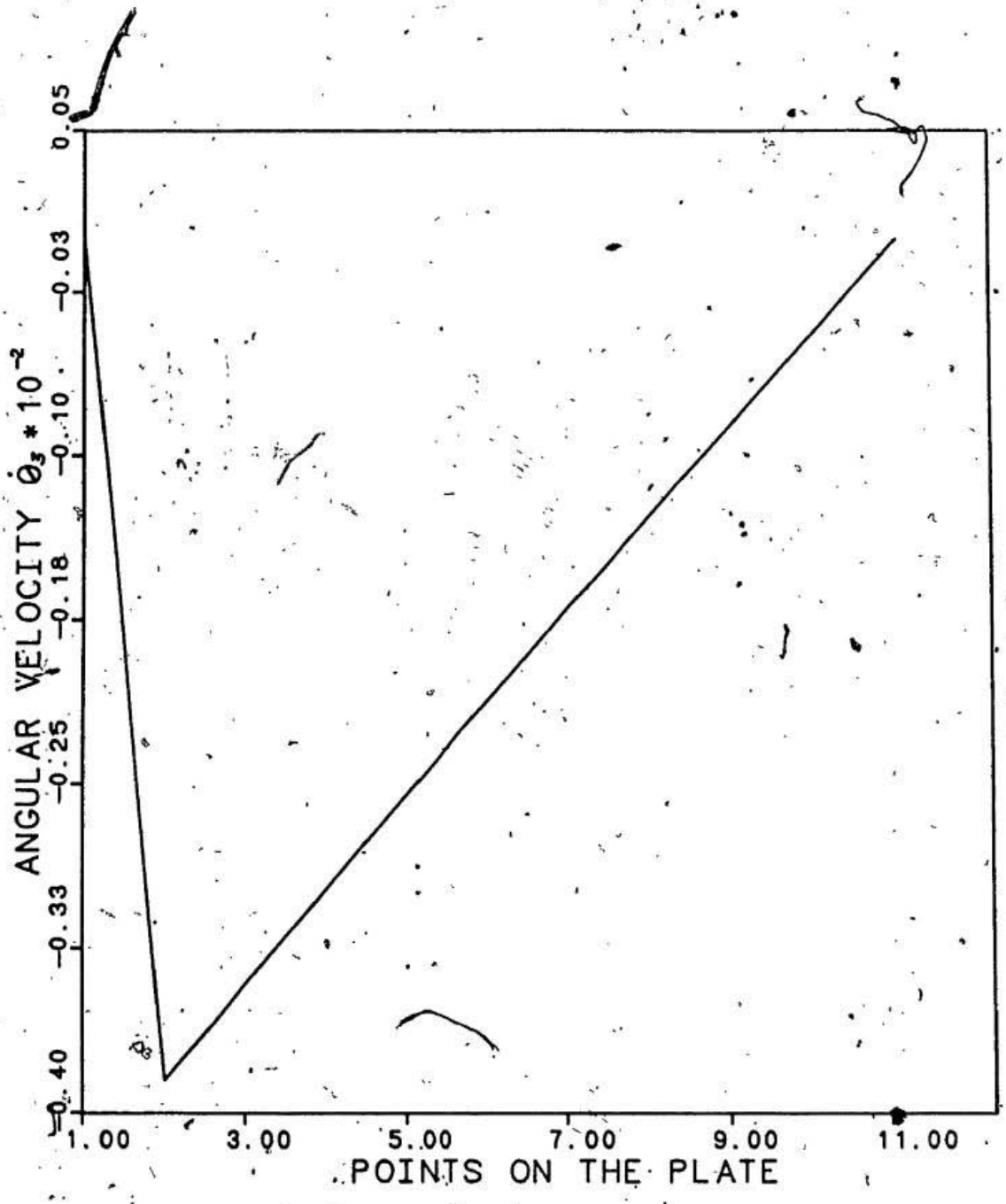

FIG. #2.24: The Variation of $\dot{\theta}_3$ at Various Points on the Plate for T3R3 Robot

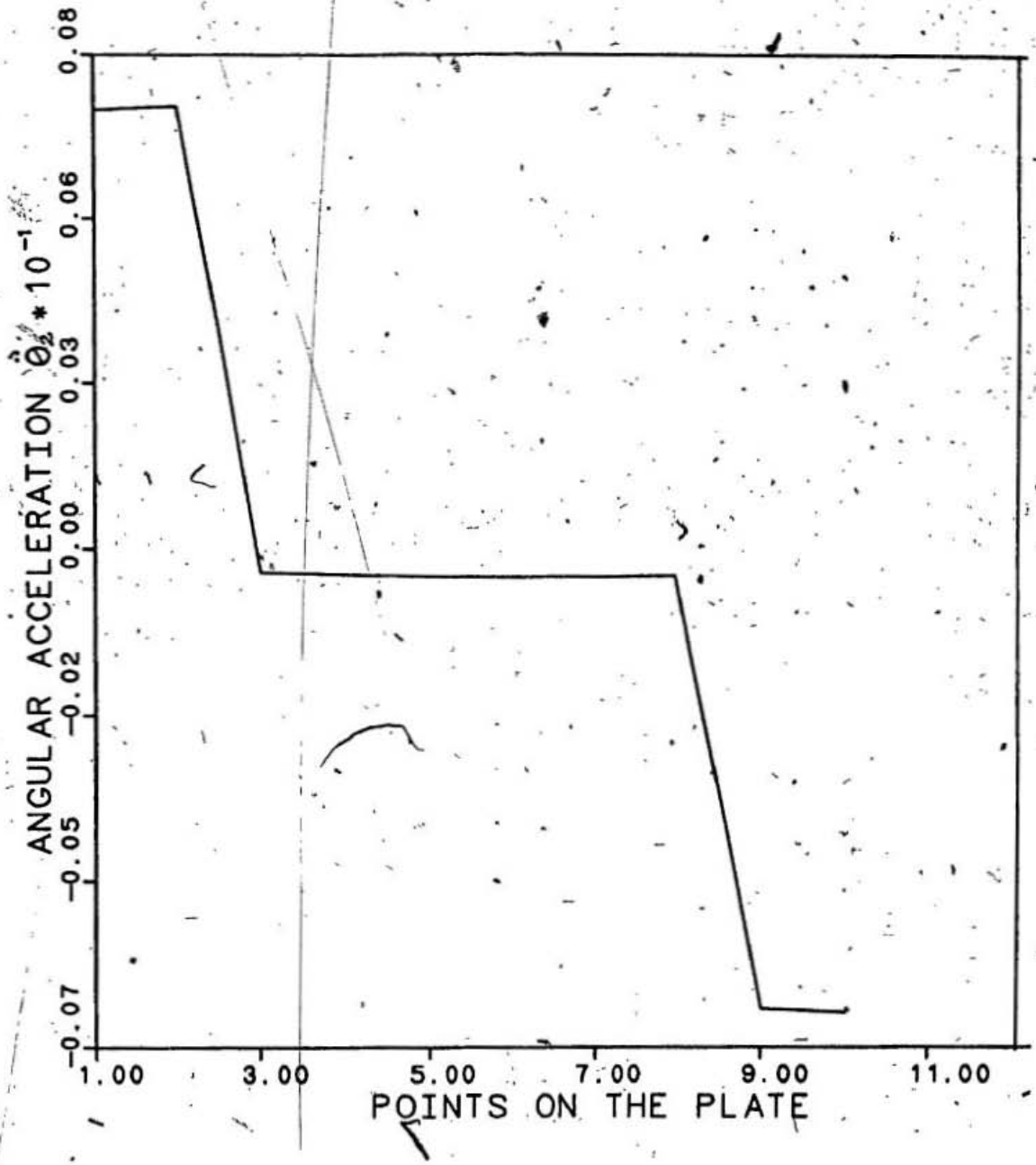

FIG. #2.25: The Variation of $\ddot{\theta}_2$ at Various Points on the Plate for PUMA-560 Manipulator

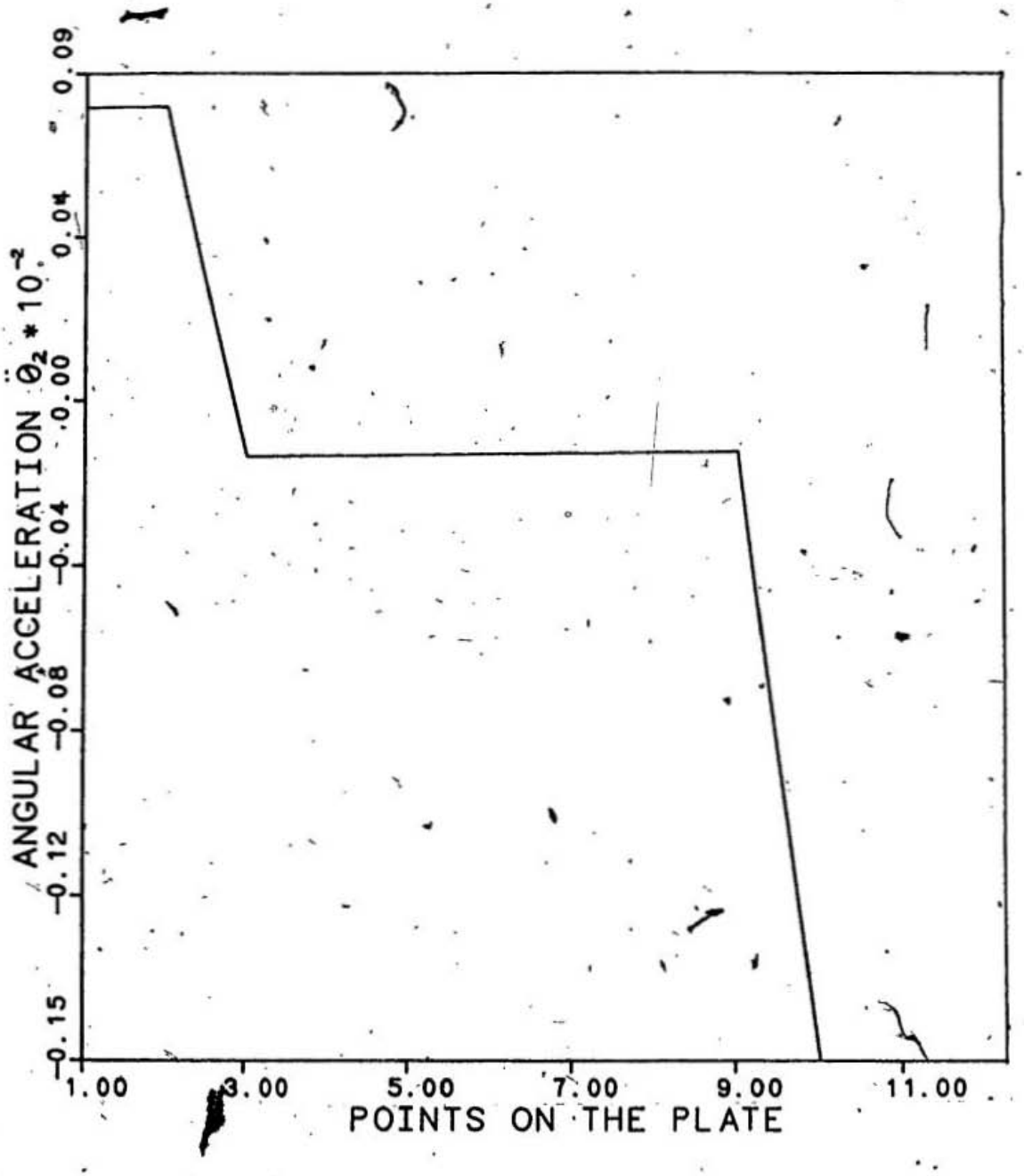

FIG. #2.26: The Variation of $\ddot{\theta}_2$ at Various Points on the Plate for T3R3 Robot

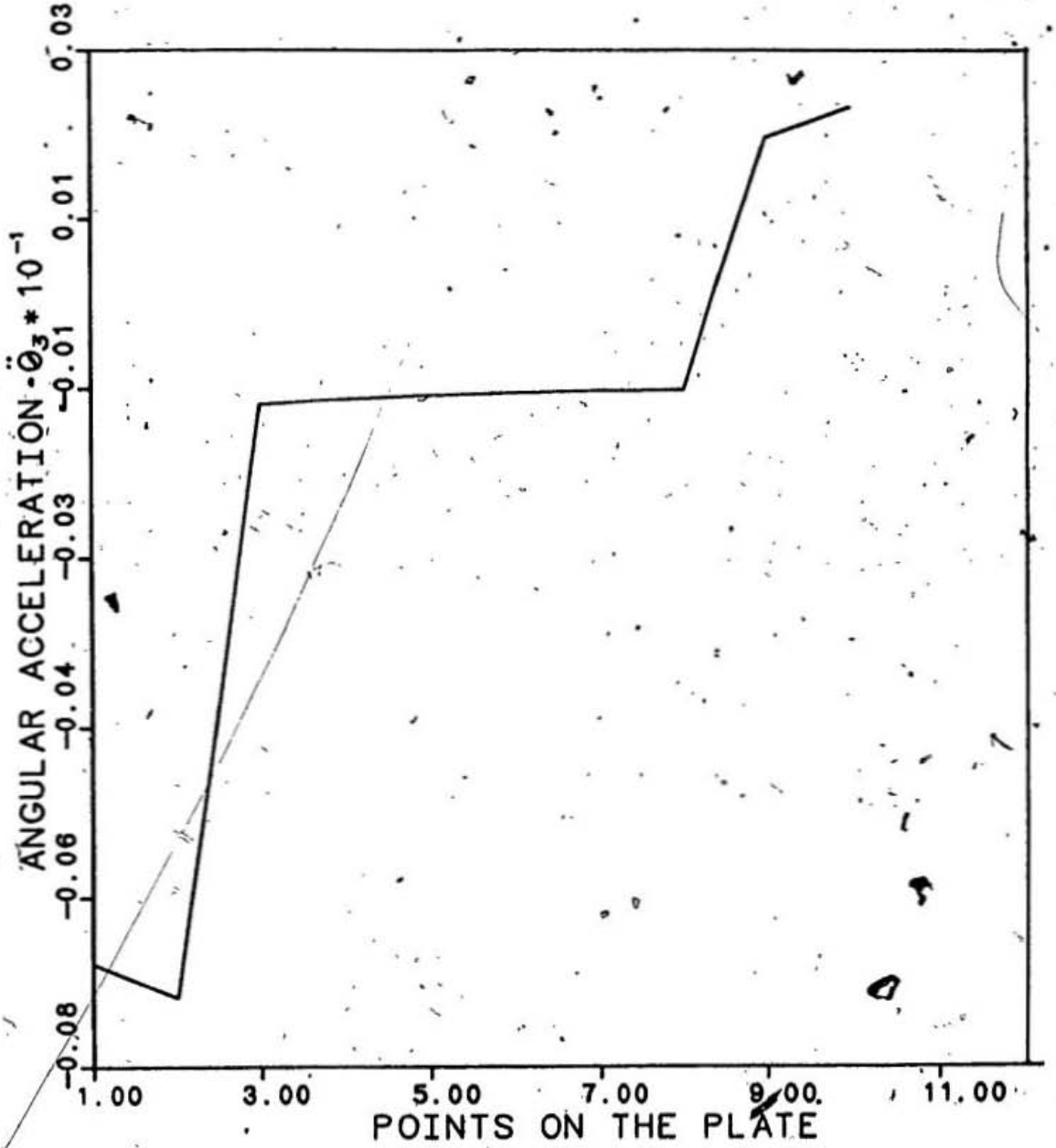

FIG. #2.27: The Variation of $\ddot{\theta}_3$ at Various Points on the Plate for PUMA-560 Manipulator

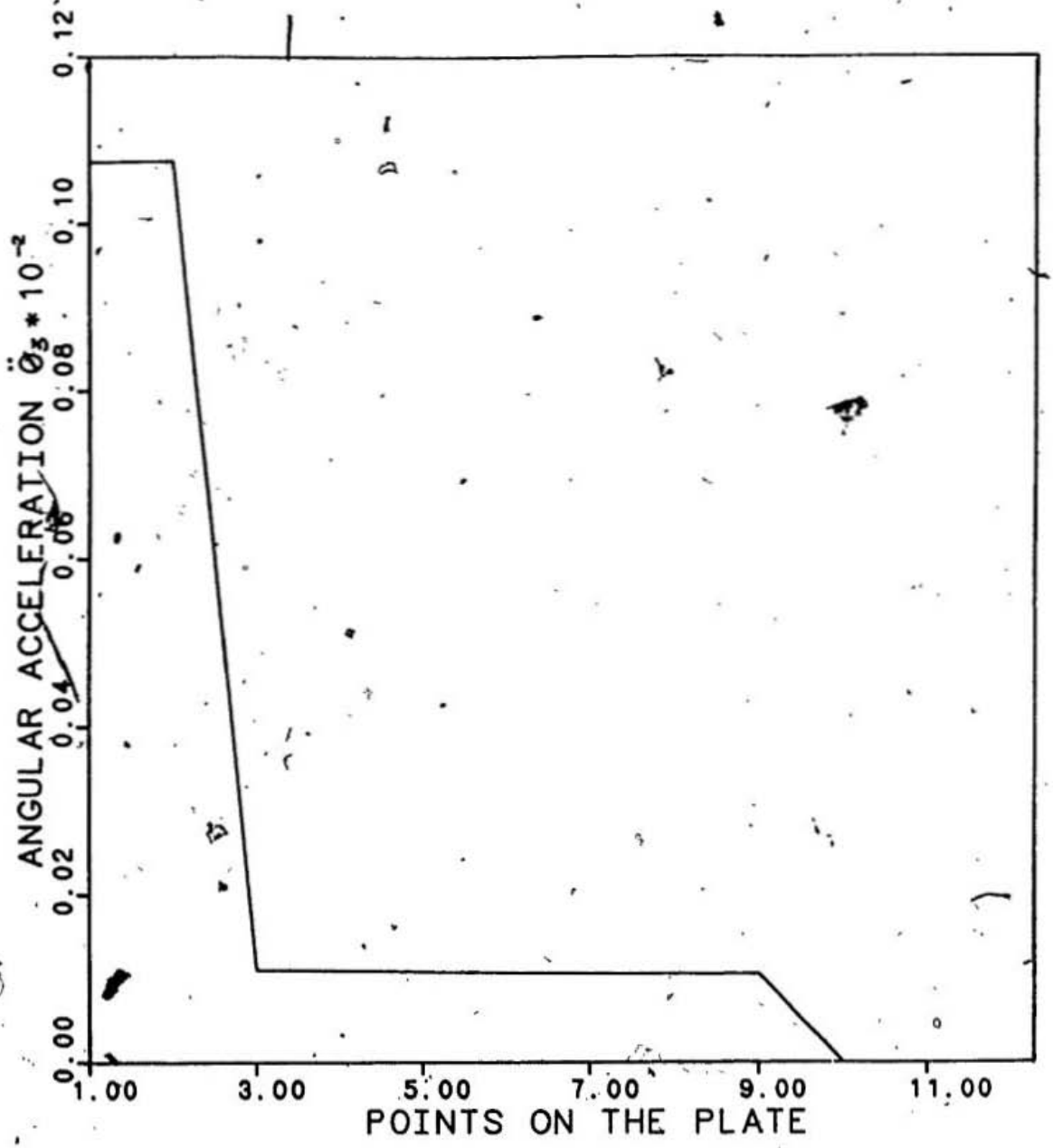

FIG. #2.28: The Variation of $\ddot{\theta}_3$ at Various Points on the Plate for T3R3 Robot

TABLE #2.8: CPU Time For One Solution by Various Methods

Method	Time
Complex Optimization	2.8 secs
L.S.M.	0.015 secs
Simplex	0.015 secs
Quadratic Programming	0.018 secs

using the methods discussed in this work.

The singularities can be of different types such as at the points where accelerations take very high values or those where the inverse of the Jacobian does not exist. These situations can be easily expressed mathematically in this work by writing constraint equations. In addition, one can use these techniques for redundant manipulators also.

2.7 Conclusions

In this chapter the displacement equations of robots of known, or arbitrary architecture and also of redundant manipulators were obtained using the Denavit-Hartenberg matrices. The relationships between the cartesian space and joint space variables were of non-linear nature. In addition, in some cases additional constraints were also imposed. The solutions of the displacement equations were carried out using the SLA technique. After this the joint velocity and acceleration equations were derived where the Jacobian matrix was also involved. The solution methods in these cases were, (a) the quadratic programming, and (b) the simplex method. The non-linear constraints were handled by the SLA technique. One can draw the following conclusion based on the work on this chapter:

1) The displacement equations of any type of manipulators can be arrived at using the Denavit-Hartenberg matrices.
2) The SLA method can be successfully used to solve unconstrained or constrained displacement equations.

3) The SLA method is more efficient than the complex optimization method or the LSM method and this method (SLA) can be used for on line systems.

4) The quadratic programming method can be used to solve unconstrained or linearly constrained joint velocity and acceleration equations of robots of any type.

5) The simplex method can also be used to solve for the problems mentioned just above.

6) The SLA method can be used for solving joint velocity and acceleration equations subjected to non-linear constraints for all class of manipulators.

7) The singular configurations can also be handled easily by the SLA method.

CHAPTER 3

THE DYNAMIC ANALYSIS OF ROBOTIC MANIPULATORS

3.1 Introduction to Dynamic Analysis

It has been shown by several authors [46-47] that the dynamic equations of motion of robotic manipulators also involve kinematic parameters such as joint space variables and their derivatives. Keeping this in mind the detailed kinematic analysis of various types of manipulators was carried out in Chapter 2. There exists a large volume of literature for manipulator dynamics with rigid arms, but for high accuracy one has to also include the flexibility of the various links [30,46]. This is especially true since the robotic manipulators are much more flexible than the machine tools. One of the methods of including the flexibility of the links is by using the finite element method and it is discussed in the next section. If one uses the finite element method, the resulting system matrices are very large and require efficient techniques to solve for the time dependent response. Several methods have been discussed in this chapter and their computational efficiencies are compared. To handle constrained problems, linear programming and the Karmarkar's algorithm are also discussed where by one can obtain the dynamic response of the constrained systems. Finally, to reduce the size of system matrices the Guyan's reduction technique and the component mode synthesis have been discussed in detail.

3.2 Dynamic Equations of Motion for Flexible Manipulators

3.2.1 The Global System Matrices

The global mass matrix can be obtained by writing the Lagrange's dynamic equation of motion for an element and then assembling the resulting elemental matrices. However, it is possible to assemble the elemental equation for a given link and then link by link assembly can be performed for the whole system. The advantage in using this scheme is that the stiffness, damping and force matrices are dependent on velocities and accelerations of various nodes. Since the nodes are on various links, which are in motion due to the angular accelerations imparted by the corresponding drive motors, therefore it would be quite logical to write dynamic equations link by link.

The Lagrange's equation of motion for ith link is expressed as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = F_i \ , \ i = 1, \ \ldots, \ N \qquad (3.1)$$

where, L = Lagrangian function which is expressed as

$$L = U_k - U_p$$

U_k = link kinetic energy

U_p = link potential energy

q_i = generalized coordinates

$\dot{q}_i$ = the first time derivative of generalized coordinates

F_i = generalized forces

N = total number of generalized coordinates

Dividing a link into number of elements, the translational kinetic energy of the nth node of the ith link is given by

$$(U_k)_{in} = \frac{1}{2} m_{in} [(\dot{x}^0_{in})^2 + (\dot{y}^0_{in})^2 + (\dot{z}^0_{in})^2]$$

$$= \frac{1}{2} m_{in} \underline{v}^T_{in} \underline{v}_{in} \qquad (3.2)$$

where

$$\underline{v}_{in} = \begin{Bmatrix} 0 \\ \dot{x}^0_{in} \\ \dot{y}^0_{in} \\ \dot{z}^0_{in} \end{Bmatrix} = \text{inertial velocity vector for the nth node on the ith link.}$$

m_{in} = the mass of the nth node of the ith link

The total kinetic energy of the ith link having NG_i nodes is given by

$$(U_k)_i = \sum_{n=1}^{NG_i} (U_k)_{in}$$

$$= \sum_{n=1}^{NG_i} \frac{1}{2} m_{in} \underline{v}^T_{in} \underline{v}_{in} \qquad (3.3)$$

The potential energy for the ith link due to the structural elasticity is given by

$$(Up_E)_i = \frac{1}{2}\{p_i\}^T[\bar{K}_i]\{p_i\}$$

$$= \frac{1}{2}\sum_{\beta=1}^{NP(i)}\sum_{\gamma=1}^{NP(i)} \bar{K}_{i\beta\gamma}\, p_{i\beta}\, p_{i\gamma} \qquad (3.4)$$

where $\bar{K}_{i\beta\gamma}$ is the finite-element structural stiffness term,

$\{p_{in}\}$ = perturbation coordinates of the nth node on the ith link as shown in Fig. 3.1.

$= \{r'_{in}\} - \{r_{in}\}$

NP(i) = 6 × NG(i) (there are six degrees of freedom at each node).

The potential energy of the nth node point of the ith link due to gravitational effect is given by

$$(Up_G)_{in} = -m_{in}(g_x x^0_{in} + g_y y^0_{in} + g_z z^0_{in}) \qquad (3.5)$$

where (g_x, g_y, g_z) = acceleration components due to gravitational effects, and

$(x^0_{in}, y^0_{in}, z^0_{in})$ = coordinates of the nth node of the ith link in terms of the inertial coordinate system.

Since $g_x = g_y = 0$, therefore Eq. (3.5) can be simplified as

$$(Up_G)_{in} = -m_{in}\, g_z z^0_{in} \qquad (3.6)$$

The total potential energy of the link is the sum of the potential energies due to elastic and gravitational effects and is expressed as

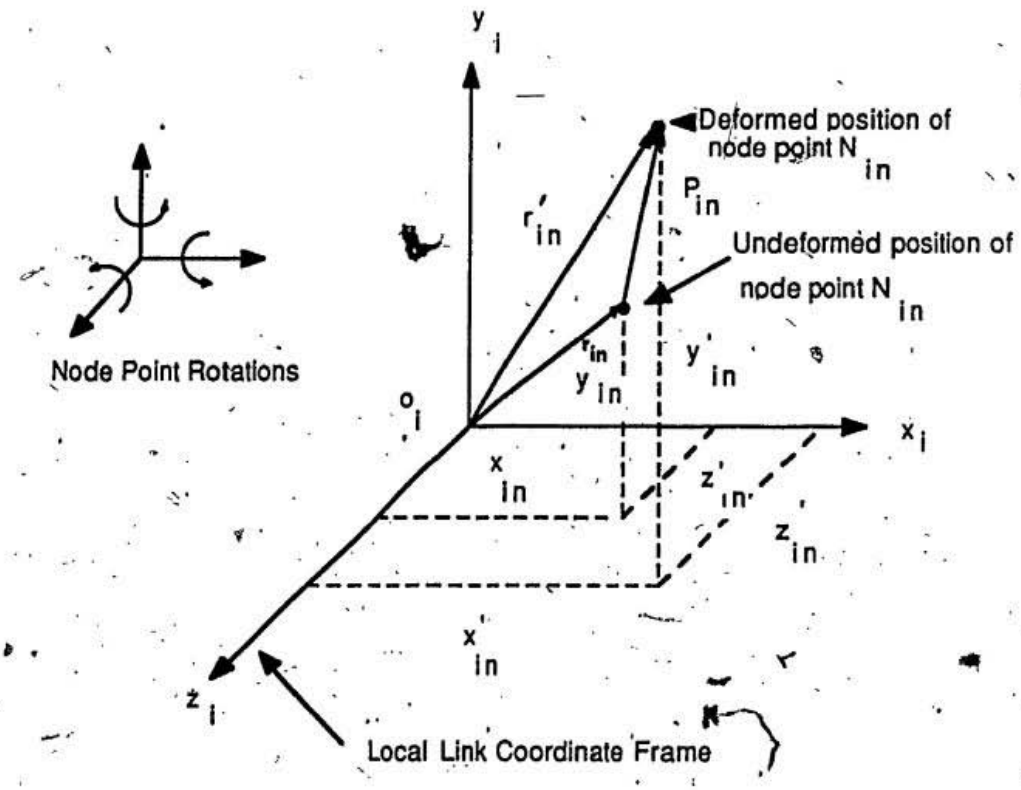

FIG. #3.1: Perturbation Coordinates of the nth Node on the ith Link

$$(Up)_i = (Up_E)_i + (Up_G)_i \qquad (3.7)$$

By substituting the expressions for kinetic energy from Eq. (3.3) and potential energy from Eq. (3.7), into the Lagrange's equation of motion given by Eq. (3.1), the matrix form of the dynamic equation can be written as [46].

$$[M]\{\ddot{q}(t)\} + [C]\{\dot{q}(t)\} + [K]\{q(t)\} = \{F(t)\} \qquad (3.8)$$

It should be noted that to obtain any of the global matrices, in addition to the elemental matrices which are given in the Appendix B, one also needs the three dimensional matrix of direction cosines. This matrix is also given in the Appendix B.

3.2.2 The Global Force Vector

3.2.2.1 The Nodal Gravity Forces

The force vector shown in Eq. (3.8) consists of several terms, one of which is due to gravitational acceleration. In the absence of any external forces or any other kind of motion, the deflection of the system will be due to these forces.

Let $(F_G)_{in}$ be the force due to gravity on nth node on the ith link, then

$$(F_G)_{in} = -m_{in} g_z \qquad (3.9)$$

where, m_{in} is the lumped mass of the nth node of the ith link, g_z is the acceleration due to gravity = 9.81 m/s^2.

As explained earlier, for the vertical gravity force, the components of g in x and y directions will be zero.

3.2.2.2 The Nodal Inertia Forces

When a robot is tracking an assigned path it is subjected to inertia forces arising due to angular accelerations and angular velocities of links. Therefore, in order to calculate the inertia forces at each node, one has to have the prior knowledge of kinematic parameters such as angular velocities and accelerations at that point.

Referring to acceleration analysis discussed in Section 2.5, one can note that in Eq. (2.72), there are terms which are functions of joint velocities and there are also terms which are functions of joint accelerations. Since Eq. (2.72) is expressed with reference to global coordinate system, so the resulting inertia forces will also be in this system.

To calculate the inertia forces, following steps should be performed:

Step 1: For a given end-effector kinematic parameters, the inverse kinematic analysis is carried out to calculate joint displacements, velocities and accelerations.

Step 2: The robot arm is divided into finite number of elements. The number of elements depends upon how accurately one wants to model the manipulator.

Step 3: At each node the accelerations are calculated using Eq. (2.72).

Step 4: The acceleration components calculated at each node from Step 3 are multiplied with appropriate lumped masses of the elements to obtain the inertia forces given by

$$(F_I)_{inx} = -m_{in}(a^i_{11}\ddot{q}_1 + a^i_{12}\ddot{q}_2 + \ldots + a^i_{1N_\ell}\ddot{q}_{N_\ell} + c^i_1)$$

$$(F_I)_{iny} = -m_{in}(a^i_{21}\ddot{q}_1 + a^i_{22}\ddot{q}_2 + \ldots + a^i_{2N_\ell}\ddot{q}_{N_\ell} + c^i_2) \quad (3.10)$$

$$(F_I)_{inz} = -m_{in}(a^i_{31}\ddot{q}_1 + a^i_{32}\ddot{q}_2 + \ldots + a^i_{3N_\ell}\ddot{q}_{N_\ell} + c^i_3)$$

where, $(F_I)_{inx}$ is the inertia force due to the nth node on ith link in the global x direction,

$(F_I)_{iny}$ is the inertia force due to the nth node on ith link in the global y direction, and

$(F_I)_{inz}$ is the inertia force due to the nth node on ith link in the global z direction.

Step 5: The global force vector is formed by adding the forces due to the gravity given by Eq. (3.9) and the inertia forces expressed in Eq. (3.10).

Step 6: The dynamic equation of motion, given by Eq. (3.8), is solved by applying appropriate boundary conditions. This results in modification of the global matrices.

3.2.3 The Methods of Solution of the Dynamic Equations

The differential equation of motion given by Eq. (3.8) can be solved using several numerical integration schemes. The approach used in the present study was the Newmark integration scheme [48]. It is possible to vary the parameters in this scheme to obtain desired or optimum stability and accuracy characteristics. Two basic assumptions are to be followed while carrying out the step by step integration in

this scheme. First, the scheme holds true for only certain discrete interval of time, say Δt and secondly, a certain variation of displacement and its time derivatives is assumed within the selected time step Δt. For example, the acceleration can be considered as constant. One of the most critical features to be considered while carrying out direct integration analysis is the number of time steps required for accurately detailing the dynamic behaviour of the structure. Too large a time step can result in erroneous solution and on the other hand, very small time step may be undesireable as this will increase the computation cost. The steps to be followed can be found in any standard text [34,48]. It is observed that the maximum CPU time is taken while finding the inverse of the matrix

$$[A_1] = \left[\frac{1}{\alpha(\Delta t)^2}[M] + \frac{\beta}{\alpha\Delta t}[C] + [K]\right]$$

for the damped systems or for the undamped systems finding the inverse of the corresponding matrix given by

$$\left[\frac{1}{\alpha(\Delta t)^2}[M] + [K]\right] \qquad (3.11)$$

where α, β are the stability parameters. For every time step this inverse computation of Eq. (3.11) is to be carried out. Also, as the size of the mass and stiffness matrices increases, the cost of the inverse computation shoots up. For accurately detailing the dynamic model of any flexible manipulator, the number of elements should be

large. This in turn increases the size of [M] and [K] and hence the computation time. Thus there is always a need for more efficient techniques.

Using this scheme one reduces Eq. (3.8) to the following form

$$[A_1]\{q\} = \{B_1\} \tag{3.12}$$

The problem remains now is to obtain the vector $\{q\}$ in the above equation. The most obvious method would be to find the inverse of the matrix $[A_1]$ and obtain the solution in the form

$$\{q\} = [A_1]^{-1}\{B_1\} \tag{3.13}$$

However, to calculate the inverse of the matrix $[A_1]$, large number of mathematical operations are required. This is why one seldom calculates the inverse of this matrix. One can obtain the solution using several other algorithms which are more efficient and are discussed in the subsequenct sub-sections.

3.2.3.1 The LDL^T Decomposition Using Skyline Storage Scheme

If all the leading principal submatrices of $[A_1]$ in Eq. (3.12) are non-singular then there exists a lower triangular matrix [L] and a diagonal matrix $[D] = \text{diag}\,[d_1, \ldots, d_n]$ such that

$$[A_1] = [L][D][L^T] \tag{3.14}$$

The matrix [L] is expressed as

$$[L] = [L_1][L_2] \ldots [L_{n-1}] \tag{3.15}$$

The general formula for forming $[L_i]$ is

$$[L_i] = \begin{bmatrix} 1 & & & 0 \\ & 1 & & \\ 0 & \ell_{i+1,i} & & \\ & \ell_{i+2,i} & 0 & \\ & \ell_{n_i} & \cdots\cdots & 1 \end{bmatrix} \qquad (3.16)$$

where

$$\ell_{i+j,i} = -\frac{A_{1i+j,i}}{A_{1ii}}$$

Also the matrix $[L_i]$ has all the columns zero except the ith column below the diagonal. All the diagonal terms are equal to one. The matrix $[L_i]^{-1}$ is obtained by just changing the signs of ith column of $[L_i]$ but the diagonal terms do not change their signs.

The vector $\{q\}$ is obtained by solving three sequence of equations given by

$$[L]\{Y\} = [L_1] \ \ldots \ [L_{n-1}]\{Y\} = \{B_1\} \qquad (3.17)$$

or $$\{Y\} = [L_{n-1}]^{-1} \ \ldots \ [L_1]^{-1}\{B_1\}$$

as the first step. Then the equations

$$[D]\{Z\} = \{Y\} \quad \text{and} \qquad (3.18)$$

$$[L]^T\{q\} = \{Z\} \qquad (3.19)$$

are solved. The number of mathematical operations required for solving Eqs. (3.17), (3.18) and (3.19) are $n^2/2$, n, $n^2/2$ respectively for a $(n \times n)$ matrix. However, even further reduction in computation is possible for matrices shown in the Fig. (3.2). In this figure, the skyline of the matrix $[A_1]$ is clearly depicted. Now if we refer to Eq. (3.17), the terms above the skyline need not be computed because the numerator is zero. If we compare a banded matrix and a skyline matrix as shown in Fig. (3.2), one can see that the skyline method is even more efficient than the banded matrix computation. This is because in the band storage scheme the zeros above the skyline are also stored and need to be computed.

3.2.3.2 The QR Decomposition Using Householder Transformations

This technique can be used where the matrix $[A_1]$ is square or rectangular, i.e. the system is unique or over-determined. Since the solution technique is a bit involved, it can be best illustrated by taking an example. Let us say we have

$$\begin{bmatrix} 5 & -4 & 1 & 0 \\ -4 & 6 & -4 & 1 \\ 1 & -4 & 6 & -4 \\ 0 & 1 & -4 & 5 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad (3.20)$$

To solve this problem one starts with a vector $\{s_1\} = \{5 \ -4 \ 1 \ 0\}^T$, which is the first column of the matrix $[A_1]$. Next we define a vector $\{U_1\}$ using

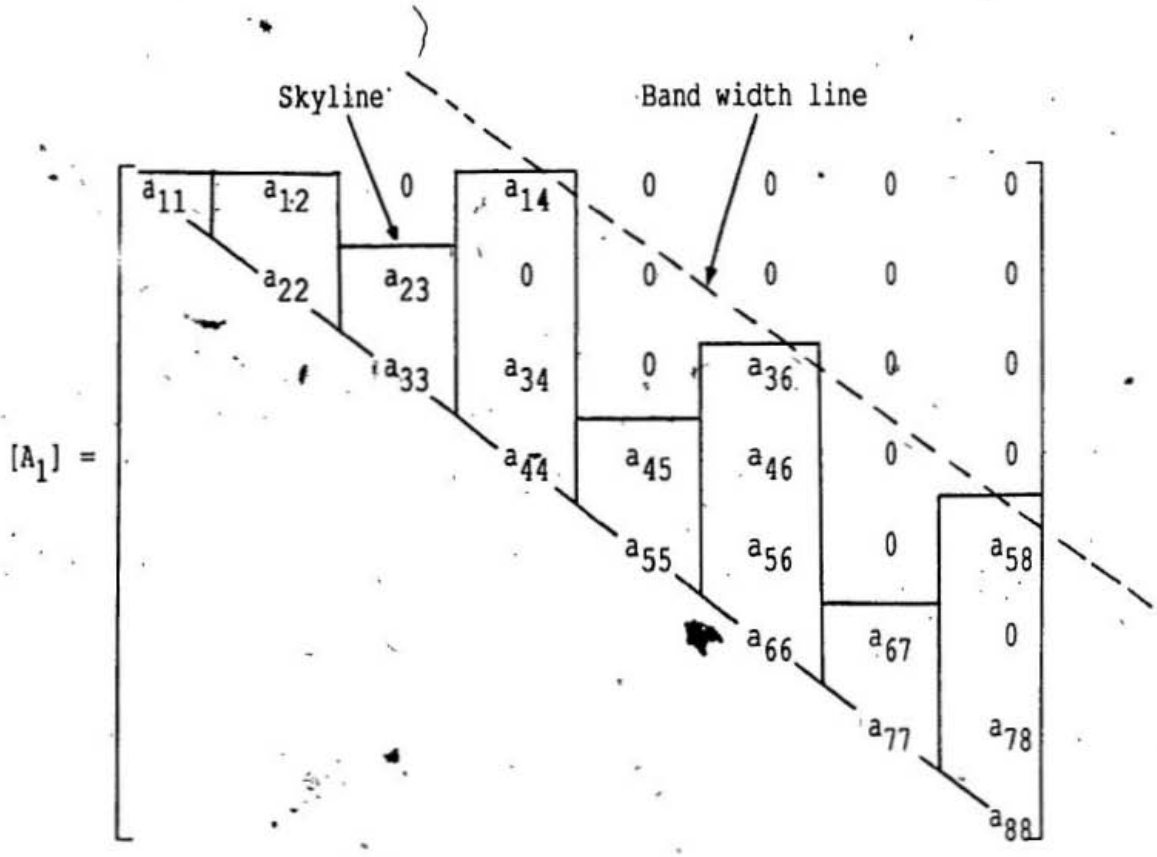

FIGURE #3.2: The Skyline Storage of $[A_1]$

$$\{U_1\} = \{s_1\} + \rho\{e_1\} \tag{3.21}$$

where, $\rho = \|s_1\|_2 = \sqrt{\{s_1\}^T\{s_1\}}$ and $\{e_1\}$ is the first column of a unit matrix [I]. Then we can calculate a matrix $[H_1]$ expressed as

$$[H_1] = [I] - \frac{2\{u_1\}\{u_1\}^T}{\{u_1\}^T\{u_1\}} \tag{3.22}$$

$$\begin{bmatrix} -0.7715 & 0.6172 & -0.154 & 0 \\ 0.6172 & 0.7849 & 0.0537 & 0 \\ -0.154 & 0.537 & 0.98 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This is a symmetric and orthogonal matrix. Pre-multiplying left hand side of Eq. (3.20) by $[H_1]$ we get

$$[H_1][A_1] = \begin{bmatrix} -6.48074 & 7.40656 & -4.166191 & 1.234427 \\ 0 & 2.2584 & -2.20005 & 0.569914 \\ 0 & -3.00646 & 5.550012 & -3.892478 \\ 0 & 1.0000 & -4.0000 & 5.0000 \end{bmatrix}$$

In the second step we form $\{s_2\}$ using the second column of the matrix $[[H_1][A_1]]$ where the term above the diagonal is made equal to zero. Thus $\{s_2\}^T$ will be equal to

$$\{0 \quad 2.02 \quad -3.006 \quad 1\}$$

we calculate $\{U_2\}$ using Eq. (3.21) and continue the procedure till the last column. For the problem at hand we will have

$$[H_3][H_2][H_1][A_1] = \begin{bmatrix} -6.488074 & 7.40656 & -4.16619 & 1.23442 \\ 0 & -3.76069 & 6.68568 & -4.74835 \\ 0 & 0 & -2.63523 & 4.21637 \\ 0 & 0 & 0 & 0.389249 \end{bmatrix}$$

on the left hand side and

$$[H_3][H_2][H_1]\{B_1\} = \begin{bmatrix} 0.617213 \\ -0.379869 \\ -0.421637 \\ 0.544949 \end{bmatrix}$$

on the right hand side. The solution vector $\{q\}$ is obtained by back substitution and in the present case it will be equal to

$$= \begin{bmatrix} 1.62 \\ 2.58 \\ 2.41 \\ 1.41 \end{bmatrix}$$

3.2.3.3 The Cholesky Decomposition

If $[A_1]$ is a positive definite symmetric matrix then there exists a lower triangular matrix $[L_T]$, with positive diagonal entries such that

$$[A_1] = [L_T][L_T]^T\{q\} = \{B_1\} \qquad (3.23)$$

Eq. (3.23) follows from the fact that there exist a LDL^T transformation given by Eq. (3.14). Since all elements of the diagonals matrix are positive, the matrix $[L_T] = [L]\ \text{diag}(d_{11}^{1/2}, \ldots, d_{nn}^{1/2})$ is real, lower triangular, has positive diagonal entries, and satisfies Eq. (3.23). This decomposition is known as Cholesky decomposition and $[L_T]$ is referred to as the Cholesky triangle. Equation (3.23) can be solved in two steps. Let us rewrite this equation as

$$[L_T]([L_T]^T\{q\}) = \{B_1\} \qquad (3.24)$$

$$\text{or} \quad [L_T]\{Y\} = \{B_1\} \qquad (3.25)$$

$$\text{where} \quad [L_T]^T\{q\} = \{Y\} \qquad (3.26)$$

In the first step one solves for $\{Y\}$ in Eq. (3.25) by forward substitution and in the second step Eq. (3.26) is solved for $\{q\}$ using backward substitution.

3.2.4 Forced Time Response of Robotic Manipulators

To obtain the global system of equations one has to start with the elemental matrices. The details of an element are shown in Fig. 3.3. Here there are six degree of freedom at each node which corresponds to three displacements and three rotations. Therefore the size of the elemental matrices will be 12X12. The T3R3 manipulator shown in Fig. 2.2 was divided into eleven elements out of which the forearm had six elements. The fixed boundary conditions were applied to the base of

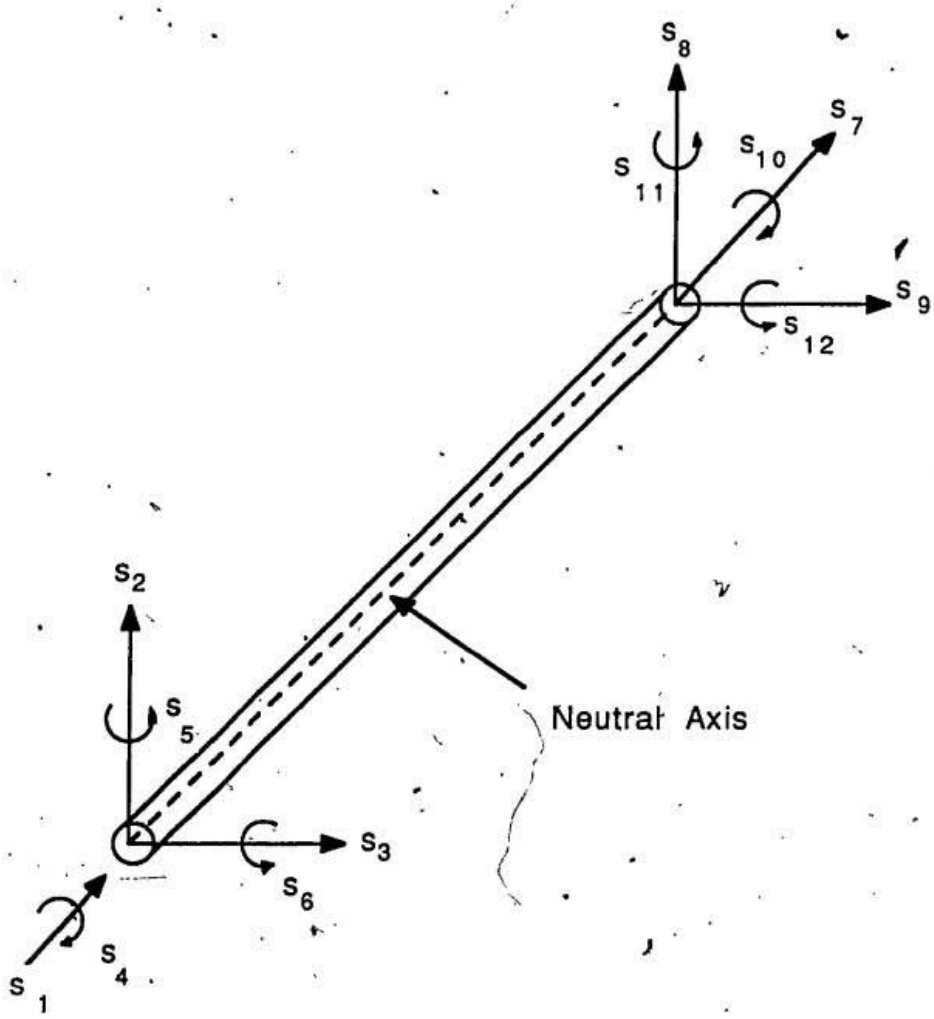

FIG. #3.3: Three-Dimensional Beam Element with Six Degree of Freedom at each Node

the robot. The various other details regarding this robot are given in the Table 3.1. The dynamic response was calculated for a welding operation on one side of the square plate. The end-effector velocity and acceleration profiles in the cartesian space are shown in the Figs. (2.19) and (2.20). The response calculated at the end-effector position are shown in Figs. (3.4) to (3.6). The response curves clearly depict the oscillatory nature of the system. The forcing functions are given in Eqs. (3.9) and (3.10) where c_i' are due to the angular velocities and $\ddot{q}_i$ are due to the angular accelerations. Since we have seen in the Chapter 2 that both the angular velocities and angular accelerations vary very sharply with time, therefore these forcing functions will undergo large variations.

The efficiencies of various methods discussed in earlier section are shown in Table 3.2. Among all the methods the LDL^T method with skyline storage is the most efficient. The QR iteration is uneconomical but its utility will become clearer when we solve an overdetermined system later on in this chapter.

3.2.5 The Karmarkar's Algorithm And Solution of Overdetermined Systems

The solution of overdetermined systems involves solution of

$$\underset{(m\times n)}{[A_k]}\ \underset{(n\times 1)}{\{x_k\}} = \underset{(m\times 1)}{\{B_k\}}\ ,\ m > n \qquad (3.27)$$

These kinds of systems can occur if we are solving dynamic equations such as given in Eq. (3.12) along with some added linear constraints.

TABLE #3.1: The Geometric Dimensions of T3R3 Robot.

Parameters	Link 1	Link 2	Link 3
Description	Shoulder	Upper Arm	Forearm
Angle (Degrees)	θ_1	θ_2	θ_3
Length (m)	0	1.016	1.5113
Cross-Section	-	Hollow Square	Hollow Circle
Outer Dimension (m)	-	0.1524	0.1524
Inner Dimension (m)	-	0.1143	0.1143
Material	-	Steel	Aluminium Alloy

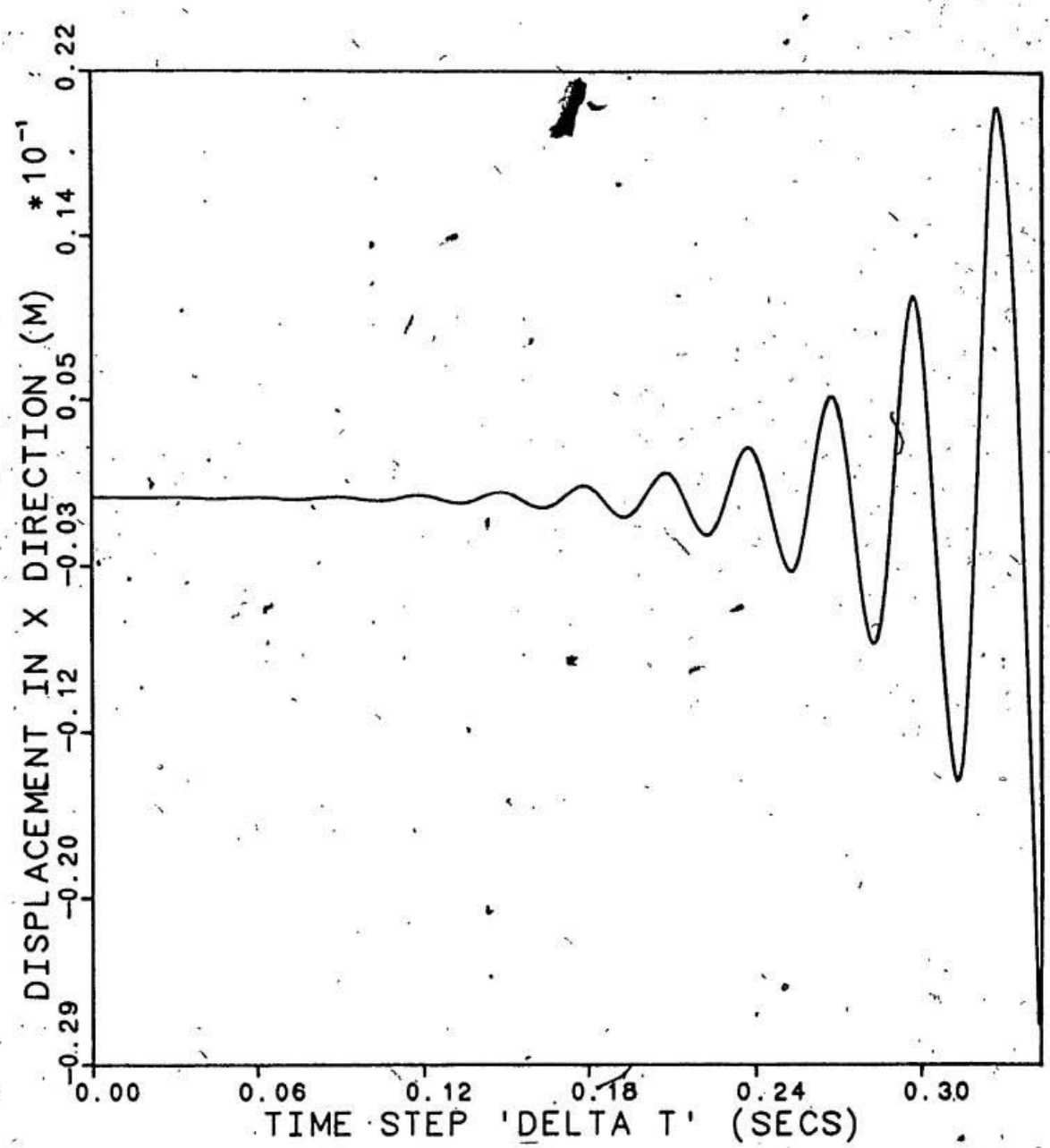

FIG. #3.4: Response Curve for End-Effector Displacement in Global X Direction

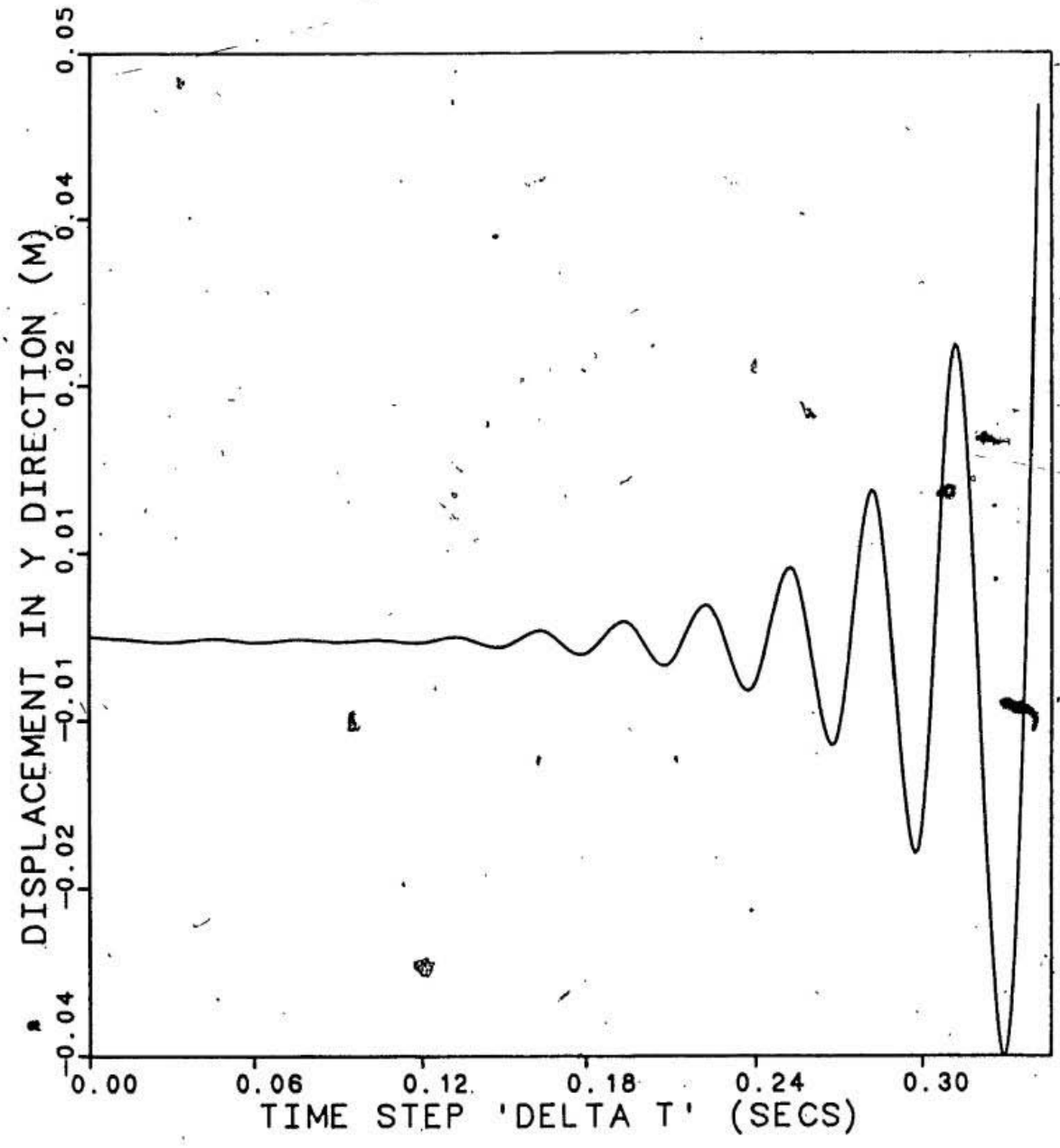

FIG. #3.5: Response Curve for End-Effector Displacement in Global Y Direction

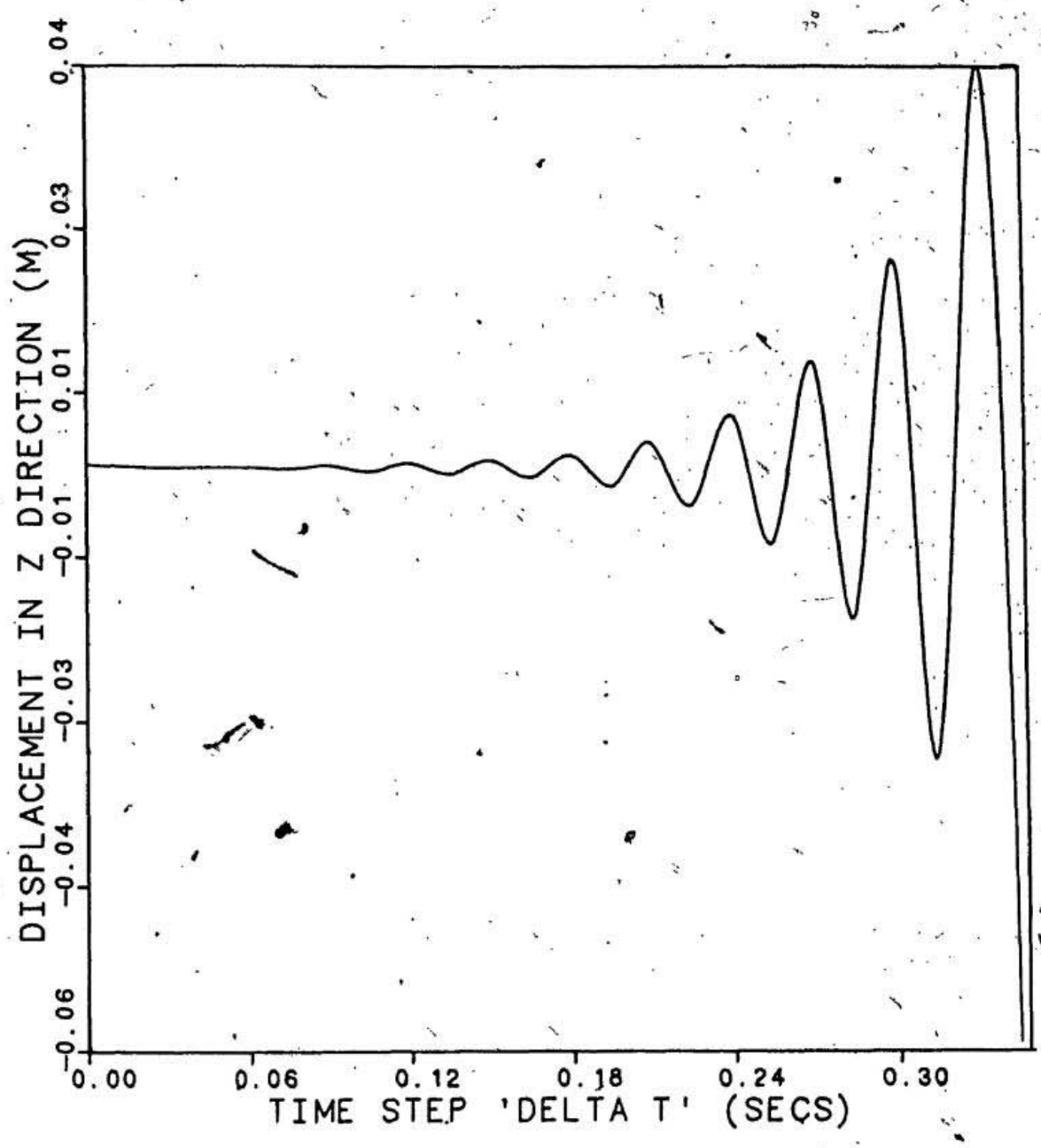

FIG. #3.6: Response Curve for End-Effector Displacement in Global Z Direction

TABLE #3.2: The Computational Efficiencies of Various Methods

Number	Methods	CPU Time (seconds)
1.	LDL^T with skyline	10.97
2.	LDL^T without skyline	13.84
3.	QR Decomposition	19.41
4.	Cholesky Factorization	15.25

These types of systems can also be there when one is solving for robots of arbitrary architecture and redundant manipulators. Such types of problems can be solved by Least Square Analysis, where Eq. (3.27) can be written, after pre-multiplying by $[A_k]^T$, as

$$[A_k]^T[A_k]\{x_k\} = [A_k]^T\{B_k\} \tag{3.28}$$

$$\{x_k\} = [A_k]^T[A_k]^{-1}[A_k]^T\{B_k\} \tag{3.29}$$

However, Eq. (3.27) can also be solved by formulating it as a L.P. problem where the objective function can be written as

Minimize

$$z_k = \sum_{j=1}^{n} c_j \, x_{kj} \tag{3.30}$$

$$\text{where} \quad c_j = \left(\sum_{i=1}^{m} a_{ij}\right)$$

subject to constraints given by Eq. (3.27). The system of equations given by the constraint equation above can be very large or small. For example, in the finite element analysis of large systems the constraint matrix $[A_k]$ will be very large. On the other hand, in the kinematic analysis of redundant manipulators, this matrix will be of very small size. In recently published papers [36-41], comparative studies have been done for solving such linear programming problems. Therefore, this method is discussed with sufficient details in this

chapter.

Before getting started with Karmarkar's algorithm let us review L.P. in general. The geometrical interpretation reveals two important features about L.P. The first feature is that an optimal solution always lies at a vertex where the vertices are determined by the intersection of N linearly independent hyperplanes defined by the constraints. Secondly, if a vertex is not optimal, then one of its neighbours has the value of objective function less than this for a minimization problem. The ellipsoid method had been proposed earlier by the Russian mathematicians Iudin and Neimirovski as a means of solving nonlinear optimization problems [49]. The proposed technique could actually be applied to solve the problem of finding feasible solution to a L.P. problem.

The ellipsoid method starts with a large ellipsoid containing the entire feasible region. The feasibility problem is solved if the point located at the center of the ellipsoid is feasible. The presence of inequality constraint separates the center point from the feasible region. This separating hyperplane can be used to construct another ellipsoid with smaller volume which still contains all of the feasible region. Since the volume of these ellipsoids decrease at every iteration, eventually the feasible region will be tightly bounded by a smaller ellipsoid whose center point is feasible. At first, the ellipsoid method was perceived as presenting the first real challenge to the simplex method. But soon it was discovered that

enormous computational efforts were involved in large scale problems and the method proved to be inferior to simplex in practice.

The major discovery came in form of a new polynomial-time algorithm for L.P. problem by Karmarkar [42] in May, 1984. Claims were made that numerical results proved that the new method was up to 50 times faster than the simplex algorithm. Unfortunately, no details on the computer implementation and experimental results were released by the author.

3.2.5.1 The Mathematical Formulation of the Karmarkar's Algorithm

The linear programming problem in Karmarkar's framework is expressed in the following homogeneous form:

Minimize

$$Z_k = \{c_k\}^T\{x_k\} \tag{3.31}$$

subject to

$$[A_k]\{x_k\} = 0 \tag{3.32}$$

$$\{e_k\}^T\{x_k\} = 1 \,, \; \{x_k\} \geq 0 \tag{3.33}$$

where $\{e_k\}$ is a vector with all its element as 1, $[A_k]$ is a $m \times n$ matrix

The minimum of Z_k is always equal to zero. Let $[D_k]$ be a diagonal matrix given by $\text{diag}(x_{k1}, \ldots, x_{kn})$.

Karmarkar employed projective transformation described by the following expression

$$\{x_k\} <-> \frac{[D_k]\{x_k\} + \{d_k\}}{\{f_k\}^T\{x_k\} + g_k} \tag{3.34}$$

Here $[D_k]$ is a non-singular matrix; $\{d_k\}$ and $\{f_k\}$ are n-dimensional vectors and g_k is a real number.

This transformation maps hyperplanes corresponding to constraint boundaries in the original space, into hyperplanes in the projected region. Fig. 3.7 shows a feasible region with a point $\bar{x}_k$ near the boundary. The projective transformation can be physically understood by considering the light source and the image or shadow created by it. With the proper rotation, it is possible to get the corresponding point $\hat{x}_k$ to lie near the center of the projected region. Here a steepest-descent direction can be found in this projected region. By back projection a direction in the original space is obtained. Since the search direction is a property intrinsic to the problem, Karmarkar's method is far superior than the existing techniques. Using the above projective transformation we have the following mapping equation

$$\{x_k'\} = \frac{[D_k]^{-1}\{x_k\}}{\{e_k\}^T[D_k]^{-1}\{x_k\}} \tag{3.35}$$

and in the original space it is

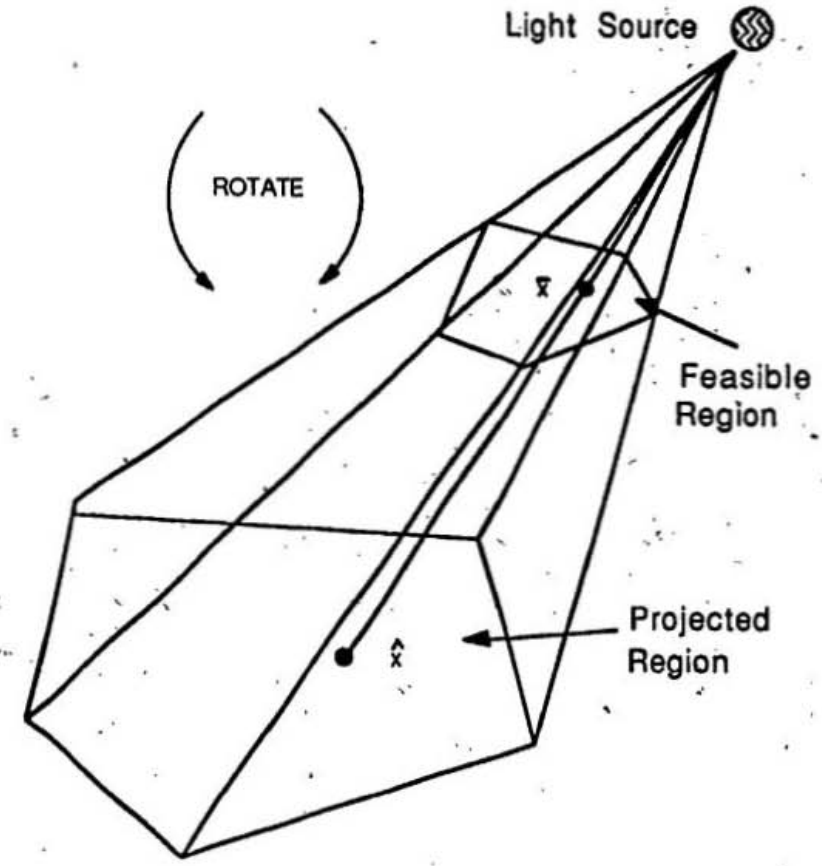

FIG. #3.7: Physical Interpretation of Projective Transformation in Karmarkar's Algorithm

$$\{x_k\} = \frac{[D_k]\{x_k'\}}{\{e_k\}^T[D_k]\{x_k'\}} \tag{3.36}$$

Thus the objective function and the constraints become

Minimize

$$Z_k = \frac{\{c_k\}^T[D_k]\{x_k'\}}{\{e_k\}^T[D_k]\{x_k'\}} \tag{3.37}$$

subject to

$$[A_k][D_k]\{x_k'\} = 0 \tag{3.38}$$

and

$$\{e\}^T\{x_k'\} = 1 \,, \qquad \{x_k'\} \geq 0 \tag{3.39}$$

From the definition of $[D_k]$, the points $\bar{x}_k$ in x_k-space are mapped onto the points $\left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$ in x_k'-space. Specifically, the algorithm generates a sequence of points $x_k'^{(1)}, x_k'^{(2)}, \ldots, x_k'^{(k)}$ where $x_{kj}'^{(1)} > 0$ $(j = 1, \ldots, n)$ as follows:

1. Define $[D_k] = \text{diag}\,(x_k'^{(k)}, \ldots, x_{kn}'^{(k)})$, and

2. $$[B_k] = \begin{bmatrix} [A_k][D_k] \\ \hline \{e_k\}^T \end{bmatrix} \tag{3.40}$$

3. Compute the orthogonal projection of $[D_k]\{c_k\}$ as:

$$\{c_{kp}\} = \left[[I] - [B_k]^T([B_k][B_k]^T)^{-1}[B_k]\right][D_k]\{c_k\} \tag{3.41}$$

$\{c_{kp}\}$ projects a steep-ascent direction into its constraint matrix $[B_k]$.

4. Normalize $\{c_{kp}\}$ and scale it by the radius of the largest sphere which can be inscribed in the simplex given by Eq. (3.39) to produce the correct direction vector. Thus we get

$$\{p_k'\} = \frac{\{c_{kp}\}}{|c_{kp}|}, \text{ and} \tag{3.42}$$

$$\{p_k\} = \{p_k'\} \cdot \frac{1}{r_k} \tag{3.43}$$

where r_k = radius of the largest sphere

$$= \frac{1}{\sqrt{n(n-1}}$$

5. Take a step length of α_k to a new feasible point

$$\{x_k'\} = \frac{1}{n}\{e_k\} - \alpha_k\{p_k\} \tag{3.44}$$

Karmarkar originally suggested that the value of 'α_k' can be set to $\frac{1}{4}$ but later on it was found that the value can be modified for less iterations.

6. Applying inverse projective transformation i.e., $\{x_k'\}$ back into $\{x_k\}$ space we get.

$$x_k^{(k+1)} = \frac{[D_k]\{x_k'\}}{\{e_k\}^T[D_k]\{x_k'\}} \tag{3.45}$$

7. Apply a stopping or termination criterion for every iteration. If the new point satisfies the termination criterion stop; otherwise, set $k = k + 1$ and repeat the above steps.

3.2.5.2 Transformation of the Standard L.P. to Karmarkar's Framework

All L.P. problems are more commonly expressed in a canonical form such as -

$$\text{Minimize } Z_k = \{c_k\}^T\{x_k\}$$

subject to

$$[A_k]\{x_k\} = \{B_k\} \tag{3.46}$$

$$\{x_k\} \geq 0$$

By scaling the variables by some plausible upper bounds on their sum, we have

$$[A_k]\{\bar{x}_k\} = \{\bar{B}_k\} = \frac{\{B_k\}}{\sigma_k}, \text{ and} \tag{3.47}$$

$$\{e\}^T\{\bar{x}_k\} + \{\bar{x}_{k(n+1)}\} = 1$$

Karmarkar also suggested that a new variable be defined corresponding to the right hand side of Eq. (3.47), and forcing this variable to be 1. Thus we will have

$$[A_k]\{\bar{x}_k\} - \left(\frac{\{B_k\}}{\sigma_k}\right)\xi_k = 0, \tag{3.48}$$

$$\xi_k = 1, \text{ and} \tag{3.49}$$

$$\{e_k^T\}\{\bar{x}_k\} + \bar{x}_{k(n+1)} = 1 \tag{3.50}$$

The last two constraints can be replaced as

$$\{e_k^T\}\{\bar{x}_k\} - \xi_k + \{\bar{x}_{k(n+1)}\} = 0 \text{ , and} \qquad (3.51)$$

$$\{e_k^T\}\{\bar{x}_k\} + \xi_k + \{\bar{x}_{k(n+1)}\} = 2 \qquad (3.52)$$

Scaling all the variables by $\frac{1}{2}$ we get

$$\begin{bmatrix} [A_k] & -\frac{\{B_k\}}{\sigma_k} & 0 \\ \{e_k^T\} & -1 & 1 \\ \{e_k^T\} & 1 & 1 \end{bmatrix} \{\hat{x}_k\} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad (3.53)$$

and a solution to Eq. (3.46) may be recovered as

$$\{x_{kj}\} = 2\sigma_k\{\hat{x}_{kj}\} \text{ , } \quad j = 1, \ldots, n \qquad (3.54)$$

The algorithm needs the construction of an artificial column to obtain a starting interior solution. This involves adding a new column and variable to Eq. (3.32). This can be achieved in the following equations:

$$[A_k]\{x_k\} + \{d_k\}\lambda_k = 0 \text{ , and} \qquad (3.55)$$

$$\{e_k\}^T\{x_k\} + \lambda_k = 1$$

such that $\frac{\{e\}}{(n+1)}$ is a feasible solution, and then minimize λ_k to zero or close to zero.

Thus Eq. (3.53) becomes

$$\begin{bmatrix} [A_k] & -\dfrac{\{B_k\}}{\sigma_k} & 0 & \{d_k\} \\ \{e_k^T\} & -1 & 1 & -\delta_k \\ \{e_k\}^T & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ \lambda_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad (3.56)$$

where, $[A_k]$ is still $m \times n$.

We used the starting solution vector as

$$\hat{x}_{kj} = \frac{1}{4n} \qquad (j = 1, \ldots, n) \text{ , and}$$

$$\hat{x}_{k(n+1)} = \hat{x}_{k(n+2)} = \lambda_k = \frac{1}{4} \qquad (3.57)$$

The artificial coefficient values are:

$$\{d_k\} = \frac{\{B_k\}}{\sigma_k} - \frac{[A_k]\{e_k\}}{n} \text{ , and } \delta_k = -1 \qquad (3.58)$$

The major computational effort is consumed in the calculation of the projective transformation in the step

$$\{C_{kp}\} = \left[[I] - [B_K]^T([B_K][B_K]^T)^{-1}[B_K]\right][D_K]\{C_K\}$$

Inverse of $([B_k][B_k]^T)$ is to be calculated at every step. Basically it can be broken up into using LDL^T transformation, as

$$[B_k][B_k]^T = \begin{bmatrix} [A_k][D_k]^2[A_k]^T & [0] \\ [0] & [n] \end{bmatrix} \qquad (3.59)$$

The only quantity which is changing is the diagonal matrix $[D_k]$.

Karmarkar suggested to use updated LU decomposition for finding the inverse but for a large system it proved to be pretty inferior to the simplex algorithm. The advantage was taken of the symmetricity of $[B_k][B_k]^T$ and skyline procedure was used to find the inverse. It was also realised that apart from the inverse computation, the matrix multiplication also consumed a good amount of CPU time. Partitioning of matrices, as shown in Table 3.3, resulted in avoiding lot of zeros in the computation. But the most efficient of all the approaches originated from least square problem. It turns out that computation in step 3 is equivalent to finding the residual of a least squares problem in the following way

$$([B_k][B_k]^T)^{-1}[B_k][D_k]\{C_k\} = \{\bar{y}_k\} \qquad (3.60)$$

which can be simplified to

$$[D_k]\{C_k\} = [B_k]^T\{\bar{y}_k\} \qquad (3.61)$$

Therefore, Eq. (3.41) reduces to

$$\{C_{kp}\} = [D_k]\{C_k\} - [B_k]^T\{\bar{y}_k\} \qquad (3.62)$$

This tremendously avoids the multiplications and inverse calculation. The most efficient way of getting the least squares solution is through QR decomposition using Householder transformations discussed in Section 3.2.3.2. The matrix $[B_k^T]$ is decomposed as

TABLE #3.3: A Typical Partitioned Matrix

$$
\left[\begin{array}{ccc|ccc|ccc}
a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & a_{1,6} & & & \\
a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & & 0 & \\
a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & & & \\
\hline
 & & & a_{4,7} & a_{4,8} & a_{4,9} & a_{4,10} & a_{4,11} & a_{4,12} \\
 & 0 & & a_{5,7} & a_{5,8} & a_{5,9} & a_{5,10} & a_{5,11} & a_{5,12} \\
 & & & a_{6,7} & a_{6,8} & a_{6,9} & a_{6,10} & a_{6,11} & a_{6,12} \\
\hline
 & & & & & & a_{7,13} & a_{7,14} & a_{7,15} & a_{7,16} & a_{7,17} & a_{7,18} \\
 & 0 & & & 0 & & a_{8,13} & a_{8,14} & a_{8,15} & a_{8,16} & a_{8,17} & a_{8,18} \\
 & & & & & & a_{9,13} & a_{9,14} & a_{9,15} & a_{9,16} & a_{9,17} & a_{9,18}
\end{array}\right]
$$

$$[Q_k][B_k]^T = \begin{bmatrix} R_k \\ 0 \end{bmatrix} \tag{3.63}$$

To illustrate the mathematical theory developed above, let us take an example.

A simple problem of solving two simultaneous equations was considered to explain the algorithm and how the variables change in different domains. We want to solve the set of equations

$$\begin{aligned} x_1 + 2x_2 &= 3 \\ x_1 + x_2 &= 2 \end{aligned} \tag{3.64}$$

Using Eq. (3.30) the objective function is to minimize $Z_k = 2x_1 + 3x_2$ subject to constraints defined in Eq. (3.64). Therefore we have

$$[A_k] = \begin{bmatrix} 1 & 2 \\ 1 & 1 \end{bmatrix}_{2X2}$$

$$[B_k] = \begin{Bmatrix} 3 \\ 2 \end{Bmatrix}_{2X1}, \text{ and } \{C_k\} = \begin{Bmatrix} 2 \\ 3 \end{Bmatrix}_{2X1}$$

The vector $\{d_k\}$ is calculated from Eq. (3.58) as

$$\{d_k\} = \begin{bmatrix} 3/\sigma_k - 3 \\ 2/\sigma_k - 2 \end{bmatrix}_{2X1}$$

The final expression from Eq. (3.56) is

$$\begin{bmatrix} 1 & 2 & -3/\sigma_k & 0 & \left(\frac{3}{\sigma_k} - 3\right) \\ 1 & 1 & -2/\sigma_k & 0 & \left(\frac{2}{\sigma_k} - 2\right) \\ 1 & 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}_{4\times5}$$

The starting value was selected as

$$\begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{Bmatrix} = \begin{Bmatrix} 1/4n \\ 1/4n \\ 1/4 \\ 1/4 \\ 1/4 \end{Bmatrix} = \begin{Bmatrix} 1/8 \\ 1/8 \\ 1/4 \\ 1/4 \\ 1/4 \end{Bmatrix}, \text{ where } n = 2$$

Following the steps given in Section 3.2.5.1 one can obtain the final solution. The values of various variables in different domains are discussed in Table 3.4.

The following parameters were used for taking the step lengths

$$\alpha_k = 0.15$$

radius of the sphere $r_k = 0.2236068$

Plausible upper bound $\sigma_k = 6$

TABLE #3.4: Values of Various Variables in Different Domains

DOMAIN	VECTOR COMPONENTS	ITERATION NUMBER	
		1	2
Projected	$C_{kp}(1)$	0.1331325	0.1468506
	$C_{kp}(2)$	0.2785761	0.248077
	$x'_k(1)$	0.2145520	0.2165479
	$x'_k(2)$	0.1732559	0.1908471
Karmarkar	$\hat{x}_k(1)$	0.0073411	0.008331
	$\hat{x}_k(2)$	0.0082697	0.008271
Original	$x_k(1)$	0.8809345	0.9997215
	$x_k(2)$	0.9923701	0.9925203
	Objective Function	0.0081	0.00052

3.2.5.3 Application to Robotics

Karmarkar's algorithm moves faster when number of variables increases. Both in kinematic and dynamic analyses of robots, one has to occasionally solve equality constraints of the type

$$[A_k]\{X_k\} = \{B_k\}$$

A large number of data points describing the joint displacements of the manipulator arm were obtained. The task to be performed was that of spot welding one side of the plate shown in Fig. 2.8. Since resolved motion rate is non-restricted, concept of restricted variables is used. Thus for three-degree-of-freedom manipulator arm, there are six variables. A linear system containing as large as 528 variables was taken and solved for 1056 restricted variables. The formation of $[A_k]$ was similiar to shown in Table 3.3. Table 3.5 shows the CPU times comparison of the simplex and the Karmarkar's algorithm. Karmarkar's method is more efficient than the simplex method as the number of variables increases. Using sophisticated data structures and mathematical tricks, the Karmarkar's algorithm can be much faster which is a challenging future research topic in this area. Formation of $[A_k]$ and solution of a large number of points in one step is quite useful in trajectory planning. There are number of ways in which a particular trajectory for a task can be achieved. To find the optimal path, all possible trajectories are taken into consideration and requird data points for these trajectories are generated. The SLA method can be used to find all possible paths. Now, for the selected

TABLE #3.5: The Computational Efficiencies of the Two Techniques

Technique	CPU Times for Number of Variables Equal to:		
Simplex	264 (seconds)	528 (seconds)	1056 (seconds)
	7.86	56.80	502.87
Karmarkar	12.42	84.89	417.49

trajectory, inverse kinematic analysis can be carried out for all the trajectories. It is possible that the constraints may not be satisfied, hence the second best optimal path is taken and the inverse analyses can be carried out. This whole optimization can be carried out in one step by using a composite objective function and well-defined constraint equations using the Karmarkar's algorithm.

3.2.6 The Dynamic Coordinate Reduction Techniques

We have seen earlier that to obtain the dynamic deflections using Eq. (3.8), large system matrices are involved. In addition, in the calculation of transient response, for very small Δt, repeated solutions are necessary. Therefore, it is quite desirable that some dynamic coordinate reduction schemes be investigated. Two such schemes were discussed in [50]. Since robotic manipulators are different from machine tool structures, these require special consideration because all the system matrices are functions of the positions of the end-effector. At first, a brief mathematical background necessary to understand these techniques is discussed and then the dependence of the natural frequencies as a function of end-effector position is discussed in the following section.

3.2.6.1 The Guyan's Reduction Technique

In Guyan's reduction technique [32], total degrees of freedom are separated into master degrees of freedom and slave degrees of freedom. The

slave degrees of freedom are selected by scanning the K_{ii}/M_{ii} ratio. The ith degree of freedom for which this ratio is largest is selected as the first 'slave'. This separation discards the higher frequencies and relatively accurate representation of lower modes is possible in the retained system. If 'm' denotes the master degrees of freedom and 's' denotes the slave degrees of freedom, then after the separation the dynamic equation, Eq. (3.8), can be written as

$$\begin{bmatrix} [M_{mm}] & [M_{ms}] \\ [M_{sm}] & [M_{ss}] \end{bmatrix} \begin{bmatrix} \ddot{q}_m(t) \\ \ddot{q}_s(t) \end{bmatrix} + \begin{bmatrix} [C_{mm}] & [C_{ms}] \\ [C_{sm}] & [C_{ss}] \end{bmatrix} \begin{bmatrix} \dot{q}_m(t) \\ \dot{q}_s(t) \end{bmatrix}$$

$$+ \begin{bmatrix} [K_{mm}] & [K_{ms}] \\ [K_{sm}] & [K_{ss}] \end{bmatrix} \begin{bmatrix} q_m(t) \\ q_s(t) \end{bmatrix} = \begin{bmatrix} F_m(t) \\ F_s(t) \end{bmatrix} \qquad (3.64)$$

The coordinate transformation matrix $[\phi]$ is given by

$$[\phi] = \begin{bmatrix} [I] \\ \hline -[K_{ss}]^{-1} [K_{ms}]^T \end{bmatrix} \qquad (3.65)$$

Therefore, relationship between $[q_s(t)]$ and $[q_{(t)}]$ can be written as

$$\underset{n\times 1}{[q(t)]} = \underset{n\times 1}{\begin{bmatrix} q_m(t) \\ \hline q_s(t) \end{bmatrix}} = \underset{n\times m}{[\phi]} \quad \underset{m\times 1}{[q_m(t)]} =$$

$$\underset{n\times m}{\begin{bmatrix} [I] \\ \hline -[K_{ss}]^{-1} [K_{ms}]^T \end{bmatrix}} \underset{m\times 1}{[q_m(t)]} \qquad (3.66)$$

and

$$[q_s(t)] = -[K_{ss}]^{-1}[K_{ms}]^T[q_m(t)] \tag{3.67}$$

Thus, the dynamic equation of motion of the condensed system is written as

$$[M_{cs}][\ddot{q}_m(t)] + [C_{cs}][q_m(t)] + [K_{cs}][q_m(t)\}] = [F_{cs}] \tag{3.68}$$

where

$$[M_{cs}] = [\phi]^T[M][\phi] ,$$

$$[C_{cs}] = [\phi]^T[C][\phi] , \tag{3.69}$$

$$[K_{cs}] = [\phi]^T[K][\phi] , \text{ and}$$

$$\{F_{cs}\} = [\phi]^T\{F\}$$

3.2.6.2 The Component Mode Synthesis

This technique can be used also for reducing the size of the matrices. This method has been extensively used in the area of aeronautical engineering since early sixties. In this technique the displacement vector $[q(t)]$ is written as

$$[q(t)] = \begin{bmatrix} q(t)^I \\ \hline q(t)^F \end{bmatrix} \tag{3.70}$$

where $[q(t)^I]$ are the interface coordinates and $[q(t)^F]$ are the

free coordinates. The static behaviour of the system can be expressed as

$$\left[\begin{array}{c|c} K^{II} & K^{IF} \\ \hline K^{FI} & K^{FF} \end{array}\right] \left[\begin{array}{c} q(t)^I \\ \hline q(t)^F \end{array}\right] = \left[\begin{array}{c} f(t)^I \\ \hline f(t)^F \end{array}\right] \tag{3.71}$$

where

$$[q(t)^F] = [q(t)^F]^I + [q(t)^F]^M \tag{3.72}$$

$$[q(t)^F]^M = [\phi_E][\eta(t)] \tag{3.73}$$

Using Eq. (3.71) one can write

$$\{q(t)^F\} = -[K^{FF}]^{-1}[K^{FI}][q(t)^I] + [\phi_E][\eta(t)] \tag{3.74}$$

In Eq. (3.73), $[\phi_E]$ is the matrix of eigen vector and $[\eta(t)]$ is the principal coordinate vector. Using the above relations, one can derive the transformation matrix as

$$[T_E]_{(m+n)\times(n+r)} = \left[\begin{array}{c|c} \underset{n\times n}{[I]} & \underset{n\times r}{[0]} \\ \hline -[K^{FF}]^{-1}[K^{FI}] & [\Phi_E] \\ \underset{m\times m}{} \quad \underset{m\times n}{} & \underset{m\times r}{} \end{array}\right] \tag{3.75}$$

where r = master degrees of freedom,

m_2 = number of free coordinates, and

n = number of interface coordinates.

Thus the transformation between $[q(t)]$ and the reduced coordinate

$[q_c(t)]$ is given by

$$[q(t)] = [T_E][q_c(t)] \tag{3.76}$$

Using the above transformation, one can derive the dynamic equation of motion as

$$[M_{dd}][\ddot{q}(t)] + [C_{dd}][\dot{q}(t)] + [K_{dd}][q(t)] = [f(t)_d] \tag{3.77}$$

where

$$\begin{aligned} [M_{dd}] &= [T_E]^T[M][T_E] \\ [C_{dd}] &= [T_E]^T[C][T_E] \\ [K_{dd}] &= [T_E]^T[K][T_E] \\ \{f(t)_d\} &= [T_E]^T\{f(t)\} \end{aligned} \tag{3.78}$$

Transformation matrix is of the order of $(m + n) \times (n + r)$, the reduced mass, stiffness and damping matrices are of the order of $(n + r) \times (n + r)$.

3.2.6.3 Results and Discussions

In the Fig. 3.8, the moving coordinate system (x_3, y_3, z_3) is located at the end effector. The system matrices i.e. [M], [K], etc. vary depending on the location of the end effector when measured with respect to the global coordinate system (X_0, Y_0, Z_0). The undamped

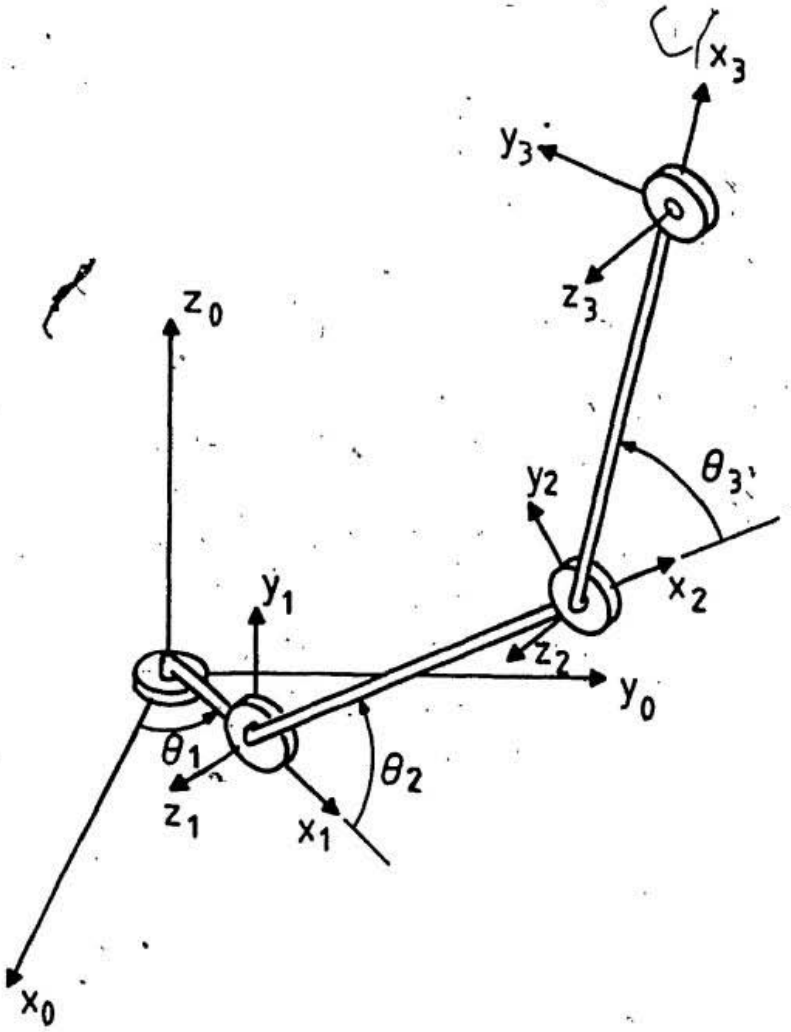

FIG. #3.8: Kinematic Model of the T3R3 Robot

natural frequencies for the manipulators were calculated corresponding to three end effector positions and are shown in Tables 3.6 to 3.8. In the Table 3.6, the total degrees of freedom are 66 and the master degrees of freedom vary between 8 and 15. To represent the first four modes one needs to have 8 degrees of freedom in either of the two reduction techniques. But one can notice that if the master degrees of freedom are 9, then the component mode results show an error. On the other hand, to include the first six modes only 10 degrees of freedom are necessary in case of component mode, whereas, the results do not converge even up to 15 degrees of freedom in the case of Guyan's reduction. Therefore, the component mode yields better results for this particular position of the end-effector. In the Table 3.7, to represent the first four modes, one needs to include 8 degrees of freedom for both reduction techniques. The Guyan's reduction results converge very well but the component mode results show considerable errors as the number of master degrees are increased. However, if first 6 modes are to be included, the component mode results accurately represent these modes corresponding to the master degree of freedom of 10; whereas, the results obtained by the Guyan's reduction show a very large error. If the master degrees of freedom are increased from 10, the Guyan's reduction results converge slowly whereas, the deviations in the component mode results at first, increase and then decrease very rapidly. So overall, the component mode results, even in this case, are better. In the Table 3.8, the results are similar to those in the Table 3.7.

TABLE #3.6 The Comparison of Undamped Natural Frequencies of the Robotic Manipulator by Various Condensation Techniques at $X_0 = 2.0$, $Y_0 = 0.25$ and $Z_0 = 0.5$

Number of Master Degrees of Freedom	The First Natural Frequency (Hertz)		The Second Natural Frequency (Hertz)		The Third Natural Frequency (Hertz)		The Fourth Natural Frequency (Hertz)		The Fifth Natural Frequency (Hertz)		The Sixth Natural Frequency (Hertz)	
	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode
8	21.76	21.75	22.67	22.65	70.88	70.90	76.50	76.23	322.72	325.61	534.30	312.87
9	21.76	21.77	22.67	22.65	70.88	71.76	76.55	124.65	322.72	288.49	533.72	400.29
10	21.756	21.755	22.676	22.662	70.859	70.816	76.417	76.257	244.596	225.679	533.562	260.685
11	21.755	21.755	22.676	22.662	70.859	70.806	76.358	76.257	230.732	225.679	533.540	260.462
12	21.755	21.753	22.676	22.662	70.859	70.806	76.324	76.257	226.501	225.535	533.535	260.462
13	21.753	21.753	22.676	22.662	70.859	70.806	76.313	76.257	225.707	225.535	533.511	260.350
14	21.753	21.753	22.676	22.662	70.859	70.805	76.312	76.257	225.640	225.472	533.495	260.350
15	21.753	21.753	22.664	22.662	70.846	70.805	76.310	76.257	225.634	225.472	293.553	260.307
66	21.753	21.753	22.660	22.660	70.803	70.803	76.257	76.257	225.454	225.454	260.288	260.288

TABLE # 3.7 The Comparison of Undamped Natural Frequencies of the Robotic Manipulator by Various Condensation Techniques at. $X_0 = 2.0$, $Y_0 = 0.125$ and $Z_0 = 0.5$

Number of Master Degrees of Freedom	The First Natural Frequency (Hertz)		The Second Natural Frequency (Hertz)		The Third Natural Frequency (Hertz)		The Fourth Natural Frequency (Hertz)		The Fifth Natural Frequency (Hertz)		The Sixth Natural Frequency (Hertz)	
	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode
8	21.76	21.74	22.68	22.66	70.68	70.71	76.45	76.12	323.09	312.68	539.19	325.24
9	21.76	21.74	22.68	22.66	70.65	71.54	76.45	124.89	323.09	287.66	538.50	399.83
10	21.75	21.74	22.68	22.66	70.65	70.58	76.25	76.12	230.69	225.14	538.46	259.84
11	21.75	21.74	22.68	22.66	70.65	71.54	76.20	124.51	226.51	287.65	538.46	394.06
12	21.75	21.74	22.68	22.66	70.65	71.30	76.21	107.80	226.99	267.50	538.44	371.97
13	21.75	21.74	22.68	22.66	70.65	71.09	76.20	94.73	225.23	262.61	538.45	324.45
14	21.75	21.74	22.68	22.66	70.65	70.91	76.20	84.72	225.16	260.84	538.44	270.27
15	21.75	21.74	22.67	22.66	70.64	70.76	76.20	78.33	225.16	237.95	333.03	260.48
66	21.74	21.74	22.66	22.66	70.57	70.57	76.12	76.12	224.92	224.92	259.92	259.47

TABLE # 3.8 The Comparison of Undamped Natural Frequencies of the Robotic Manipulator by Various Condensation Techniques at, $X_0 = 1.0$, $Y_0 = 0.25$ and $Z_0 = 0.5$

Number of Master Degrees of Freedom	The First Natural Frequency (Hertz)		The Second Natural Frequency (Hertz)		The Third Natural Frequency (Hertz)		The Fourth Natural Frequency (Hertz)		The Fifth Natural Frequency (Hertz)		The Sixth Natural Frequency (Hertz)	
	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode	Guyan's Reduction	Component Mode
8	24.59	24.57	25.47	25.46	66.78	66.68	70.85	70.64	360.42	317.58	506.15	347.52
9	24.59	24.58	25.47	25.45	66.70	68.38	70.85	155.90	360.42	288.30	505.53	430.12
10	24.57	24.56	25.47	25.45	66.75	66.68	70.74	70.64	238.94	232.22	505.39	258.08
11	24.57	24.57	25.47	25.45	66.75	68.37	70.71	155.71	234.12	285.52	505.40	424.92
12	24.57	24.56	25.47	25.45	66.75	68.03	70.71	128.30	234.63	265.06	505.37	400.14
13	24.57	24.56	25.47	25.45	66.75	67.70	70.71	102.89	232.52	259.36	505.37	324.86
14	24.57	24.56	25.47	25.45	66.75	66.99	70.71	182.53	232.45	258.39	505.37	263.82
15	24.57	24.56	25.47	25.45	66.73	66.99	70.71	72.67	232.44	239.29	332.22	258.2
66	24.56	24.56	25.45	25.45	66.67	66.67	70.64	70.64	231.96	231.96	257.98	257.98

From these results one can see that one can use either of these two techniques in the coordinate reduction scheme and select the better technique depending upon the number of modes to be retained. In summary, the lower modes are retained, in the case of Guyan's reduction technique, by calculating only the ratios of K_{ii}/M_{ii} (diagonal terms) of the global matrices. On the other hand, the transformation matrix as shown in Eq. (3.75), in the case of component mode synthesis, contains the lower modes of the free coordinates. The calculation of these lower modes involves the matrices $[M^{FF}]$ and $[K^{FF}]$. Thus the modal information based on very large number of free coordinates is already included in the transformation matrix.

3.3 Conclusions

In this chapter the dynamic equations of motion were obtained using the finite element method. The transient response of the system was calculated using Newmark integration scheme. The solutions of the linear system of equations were carried out by: (a) LDL^T method with skyline approach, (b) Cholesky decomposition, (c) QR decomposition using Householder transformations, and (d) LDL^T method. The solution of the constrained dynamic problems was analysed using the linear programming and Karmarkar's method. After this, the reduction in the sizes of the system matrices was obtained by (a) Guyan's reduction technique, and, (b) component mode synthesis. Based on the analysis of this chapter the following conclusions can be drawn:

1. The finite element analysis can be successfully used to derive the dynamic equation of motion of flexible manipulators.
2. LDL^T technique with skyline storage is the most efficient among the 4 methods used in this chapter.
3. The L.P. method and as well as Karmarkar's algorithm can be used to solve the constrained dynamic problems.
4. The L.P. method is more efficient for number of variables less than 1056.
5. Overall the component mode synthesis yields better results than the Guyan reduction technique.

CHAPTER 4

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

4.1 Conclusions

The objective of this work was to carry out the kinematic and dynamic analysis of flexible manipulators of arbitrary architecture and subjected to various linear and non-linear constraints. The kinematic analysis was divided into three parts which were: (a) displacement analysis, (b) velocity analysis, and (c) acceleration analysis of the three dimensional mechanisms. These problems were solved using the SLA method, simplex method and quadratic programming [51,52]. The dynamic analysis was carried out using the finite element method. The dynamic equations were then solved in the time-domain using the Newmark integration scheme and four other numerically efficient techniques. The solution of the constrained dynamic problems involved the use of simplex method and Karmarkar algorithm. In order to achieve greater computational efficiencies, two dynamic condensation techniques were also discussed. The conclusions that can be drawn out of this investigation are:

1. The displacement equations of any type of manipulators can be arrived at using the Denavit-Hartenberg matrices.
2. The SLA method can be successfully used to solve unconstrained or constrained displacement equations.

3. The SLA method is more efficient than the complex optimization method or the LSM method and this method can be used for on line systems.
4. The quadratic programming method can be used to solve unconstrained or linearly constrained joint velocity and acceleration equations of robots of any type.
5. The simplex method can also be used to solve for the problems mentioned just above.
6. The SLA method can be used for solving joint velocity and acceleration equations subjected to non-linear constraints for all class of manipulators.
7. The singular configurations can also be handled easily by the SLA method.
8. The finite element analysis can be successfully used to derive the dynamic equation of motion for flexible manipulators.
9. LDL^T technique with skyline storage is the most efficient among the 4 methods used in Chapter 3.
10. The L.P. method and as well as Karmarkar's algorithm can be used to solve the constrained dynamic problems.
11. The L.P. method is more efficient for number of variables less than 1056.
12. Overall, the component mode synthesis yields better results than the Guyan's reduction technique.

4.2 Limitations and Suggestions for Future Work

The studies carried out here had the following limitations:

1. In the derivation of dynamic equations, the link length were assumed constant because of the rotary joint. In the more general approach, the link lengths can be allowed to be varied as in the case of prismatic joints and corresponding changes can be included in the kinematic and dynamic analyses.
2. In the present work multi-loop linkages were not considered. This would be an important future research work.
3. The design of optimal control system is an important challenging work for future.
4. In the present work, consideration of back lash, and other manufacturing errors were not included in the analysis.
5. Computational efficiencies in the constrained dynamic problems can be further increased by several new research work being carried out in the optimization area.

REFERENCES

1. Critchlow, A., "Introduction to Robotics", Macmillan Publishing Company, NY, 1985.

2. Denavit, S., and Hartenberg, R. S., "A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices", J. Appl. Mech. 22, Trans. ASME 77, 215-221, 1955.

3. Craig, J. J., "Introduction to Robotics: Mechanics and Control", Stanford University, pp. 103-119, Addison-Wesley, 1986.

4. Koren, Y., "Robotics for Engineers", pp. 83-85, McGraw-Hill, NY, 1985.

5. Paul, R. P., "Robot Manipulators: Mathematics, Programming and Control", pp. 65-73, MIT Press, Massachussets, 1981.

6. Asada, H., and Slottine, J. J. E., "Robot Analysis and Control", pp. 15-49, Wiley Interscience Publications, 1986.

7. Pieper, D., "The Kinematics of Manipulators Under Computer Control", Ph.D. Thesis, Stanford University, 1968.

8. Featherstone, R., "Position and Velocity Transformations Between Robot End-Effector Coordinates and Joint Angles", Int. J. Robotics Research, 2(2): 35-45, 1983.

9. Hollerbach, J. M., and Sahar, G., "Wrist-Partitioned, Inverse Kinematic Accelerations and Manipulator Dynamics", Int. J. Robotics Research, Vol. 2, pp. 66-76, Winter 1983.

10. Paul, R. P., and Zhang, H., "Computationally Efficient Kinematics for Manipulators with spherical wrists based on the Homogeneous Transformation Representation". Kinematics of Robot Manipulators., Edited by McCarthy, J. M., pp. 30-42, The MIT Press, (1987).

11. Duffy, J., and Derby, S., "Displacement Analysis of a Spatial, 7R Mechanism: A Generalized Lobster's Arm." Journal of Mech. Des., Trans. ASME, 101(2): 224-231, 1979.

12. Duffy, J., and Crane, C., "A Displacement Analysis of the General Spatial 7-link, FR Mechanism". Mech-Mach. Theory, 15(3): pp. 153-159, 1980.

13. Tsai, L. W., and Morgan, A. P., "Solving the Kinematics of the Most General Six and Five Degrees of Freedom Manipulators by Continuation Methods", ASME paper 84-DET 20, Cambridge, Mass: ASME Design Eng. Tech. Conference.

14. Kostantinov, M. S., and Markov, M. D., "Discrete Positions Method in Kinematics and Control of Spatial Linkages", Mech-Mach. Theory 15(1): 47-60, 1980.

15. Albala, H., and Angeles, J., "Numerical Solution to the Input-Output Displacement Equation of the General 7R Spatial Mechanism". Proc. 5th World Congress, Theory of Mach. Mechanisms, pp. 1008-1011, 1979.

16. Uicker, J. J., Jr., Denavit, J., and Hartenberg, R. S., "An Iterative Method For the Displacement Analysis of Spatial Mechanisms", J. Appl. Mech., Trans. ASME, Vol. 86, pp. 309-316, 1964.

17. Alizade, R., Duffy, J. and Hatiyere, E. T. "Mathematical Models for Analysis and Synthesis of Spatial Mechanisms", Mech-Mach Theory 18(5): 301-328, 1983.

18. Bouillion, T. L., and Odell, L., "Generalized Inverse Matrices", New York Willey Intersciences, 1971.

19. Angeles, J., "On the Numerical Solution of the Inverse Kinematic Problem", Int. J. of Robotics Research, 2(4): 21-37, 1985.

20. Subbiah, M. and Sharan, A. M., "The Non-linear Displacement Analysis of Robot Manipulators Using the Complex Optimization Method", Mech-Mach. Theory, Vol. 22, No. 1, pp. 89-95, 1987.

21. Bejczy, A. K., "Robot Arm Dynamics and Control", Technical Memo 33-669. Jet Propulsion Laboratory.

22. Hollerbach, J. M., "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity", IEEE Trans. on Systems, Man, and Cybernatics, Vol. SMC-10, (11): 730-736, 1980.

23. Luh, J. Y. S., et al. "On-line Computational Scheme for Mechanical Manipulators", Trans. of ASME, Journal of Dynamics Systems, Measurements, and Control, (120): 69-76, 1980.

24. Lee, C. S. G., et al, "An Efficient Formulation of Robot Arm Dynamics for Control Analysis and Manipulator Design", Technical Report RSD-TR8-82, Center for Robotics and Integrated Manufacturing, University of Michigan.

25. Uicker, J. J., "On the Dynamic Analysis of Spatial Linkages Using 4X4 Matrices", Ph.D. Dissertation, Northwestern University, Evanston, Illinois, 1965.

26. Woo, L. S., and Freudenstein, F. "Dynamic Analysis of Mechanisms Using Screw Coordinates", Trans. of the ASME, Journal of Engineering for Industry, 93(1): 273-276, 1971.

27. Yang, A. T. "Inertia Force Analysis of Spatial Mechanisms", Trans. of the ASME, Journal of Engineering for Industry 93(1): 27-33, 1971.

28. Winfrey, R. C., "Dynamics of Mechanisms with Elastic Links", Ph.D. Thesis, University of California, Los Angeles, 1969.

29. Sadler, J. P., and Sandor, G. N. "A Lumped Parameter Approach to Vibration and Stress Analysis of Elastic Linkages", Trans. of the ASME, Journal of Engineering for Industry 92(2): 549-557, 1973.

30. Sunada, W. H., "Dynamic Analysis of Flexible Spatial Mechanisms and Robotic Manipulators", Ph.D. Dissertation, University of California, Los Angeles, 1981.

31. Hurty, W. C., "Dynamic Analysis of Structural Systems Using Component Modes", AIAA Journal, 3(4), 1965.

32. Guyan, R. J., "Reduction of Stiffness and Mass Matrices", AIAA Journal, 3:380, 1965.

33. Subbiah, M., "The Computer-Aided Structural Design of Robotic Manipulators", M.Eng. Thesis, Memorial University of Newfoundland, July 1986.

34. Rao, S. S., "Mechancial Vibrations", Addison-Wesley Publication Company, 1986.

35. Dantzig, G. B., "Linear Programming and Extensions", Princeton University Press, Princeton, New Jersey, 1963.

36. Barnes, E. R., "A Variation on Karmarkar's Algorithm For Solving Linear Programming Problems", Mathematical Programming 36: 174-182, 1986.

37. Murthy, K. G., "The Gravitational Method For Linear Programming", Opsearch 23(4): 206-214, 1986.

38. Gill, P. E., Murray, W., Saunders, M. A., Tomlin, J. A., Wright, M. H. "On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method", Mathematical Programming 36: 183-209, 1986.

39. Gay, D. M., "A Variant of Karmarkar's Linear Programming Algorithm for Problems in Standard Form", Mathematical Programming 37: 81-90, 1987.

40. Mehrotra, S., "Variants of Karmarkar's Algorithm: Theoretical Complexity and Practical Implementation", Ph.D. Dissertation, Columbia University, 1987.

41. Lustig, I. J., "A Practical Approach to Karmarkar's Algorithm", System Optimization Laboratory, Technical Report 85-5, Stanford University, 1985.

42. Karmarkar, N. K., "A New Polynomial-time Algorithm for Linear Progamming", Combinatorica 4, 373-395, 1984.

43. Griffith, R. E., and Stewart, R. A. "A Nonlinear Programming Technique for Optimization of Continuous Processing Systems", Management Science, 7(379), 1961.

44. Rao, S. S., "Optimization, Theory and Applications", pp. 113-115, Wiley Eastern, New Delhi 1978.

45. Chevallerau, C. and Khalil, W., "Efficient Method for the Calculation of the Pseudo Inverse Kinematic Problem", IEEE-Int. Conference on Robotics and Automation, Vol. 3, 1987.

46. Book, W. J., "Recursive Lagrangian Dynamics of Flexible Manipulator Arms", International Journal of Robotics Research, 3(3): 87-101, 1984 Fall.

47. Naganathan, G., and Soni, A. H. "Coupling Effects of Kinematics and Flexibility in Manipulators", International Journal of Robotics Research 6(1): 75-85, 1987 Spring.

48. Bathe, K. and Wilson, E. L. "Numerical Methods in Finite Element Analysis", Prentice-Hall, Inc., New Jersey, 1976.

49. Bland, R. G., Goldfard, D., and Todd, M. J. "The ellipsoid method: a survey", Operation Research 29, 1039-1091, 1981.

50. Subbiah, M., Sharan, A. M., and Jain, Jinesh, "A Study of the Dynamic Condensation Techniques for the Machine Tools and Robotic Manipulators", Mechanism and Machine Theory 23(1): 63-69, 1988.

51. Jain, Jinesh, and Sharan, A. M. "The inverse Velocity and Acceleration Analyses of Robots of Arbitrary Architecture Subjected to various types of Constraints". Accepted for publication in Mechanism and Machine Theory, June 1988.

52. Jain, Jinesh, and Sharan, A. M. "The constrained Velocity and Acceleration Analyses of Robots of Arbitrary Architecture Using the Quadratic Programming". Ninth Symposium on Engineering Applications of Mechanisms, London, Canada, May 1988.

APPENDIX A

DETAILS OF CONSTANTS USED IN THE KINEMATIC ANALYSIS

Various constants for the T3R3 manipulator are:

$$a_{11} = -L_2 Cq_2 Sq_1 - L_3 Cq_{23} Sq_1$$

$$a_{12} = -L_2 Sq_2 Cq_1 - L_3 Sq_{23} Cq_1$$

$$a_{13} = -L_3 Sq_{23} Cq_1$$

$$a_{21} = L_2 Cq_2 Cq_1 + L_3 Cq_{23} Cq_1$$

$$a_{22} = -L_2 Sq_2 Sq_1 - L_3 Sq_{23} Sq_1$$

$$a_{23} = -L_3 Sq_{23} Sq_1$$

$$a_{31} = 0$$

$$a_{32} = L_2 Cq_2 + L_3 Cq_{23}$$

$$a_{33} = L_3 Cq_{23}$$

$$c_1 = 2[L_2 Sq_1 Sq_2 + L_3 Sq_{23} Sq_1]\dot{q}_1\dot{q}_2$$

$$-2[L_3 Cq_1 Cq_{23}]\dot{q}_2\dot{q}_3 + 2[L_3 Sq_1 Sq_{23}]\dot{q}_1\dot{q}_3$$

$$-[L_2 Cq_1 Cq_2 + L_3 Cq_{23} Cq_1]\dot{q}_1^2$$

$$-[L_2 Cq_1 Cq_2 + L_3 Cq_{23} Cq_1]\dot{q}_2^2 - [L_3 Cq_{23} Cq_1]\dot{q}_3^2$$

$$c_2 = -2[L_2Sq_2Cq_1 + L_3Sq_{23}Cq_1]\dot{q}_1\dot{q}_2$$
$$-2[L_3Sq_1Cq_{23}]\dot{q}_2\dot{q}_3 - 2[L_3Cq_1Sq_{23}]\dot{q}_1\dot{q}_3$$
$$-[L_2Sq_1Cq_2 + L_3Cq_{23}Sq_1]\dot{q}_1^2 - [L_2Sq_1Cq_2 + L_3Cq_{23}Sq_{23}Sq_1]\dot{q}_2^2$$
$$-[L_3Cq_{23}Sq_1]\dot{q}_3^2$$

$$c_3 = -2[L_3Sq_{23}]\dot{q}_2\dot{q}_3 - [L_2Sq_2 + L_3Sq_{23}]\dot{q}_2^2$$
$$-[L_3Sq_{23}]\dot{q}_3^2$$

Similarly for Stanford Manipulator they are:

$$a_{11} = -q_3Sq_2Sq_1 - L_2Cq_1$$

$$a_{12} = q_3Cq_1Cq_2$$

$$a_{13} = Cq_1Sq_2$$

$$a_{21} = q_3Cq_1Sq_2 - L_2Cq_1$$

$$a_{22} = q_3Sq_1Cq_2$$

$$a_{23} = Sq_1Sq_2$$

$$a_{31} = 0$$

$$a_{32} = -q_3Sq_2$$

$$a_{33} = Cq_2$$

$$c_1 = -2(q_3 Sq_1 Cq_2)\dot{q}_1\dot{q}_2 + 2(Cq_1 Cq_2)\dot{q}_2\dot{q}_3$$

$$-2(Sq_1 Sq_1)\dot{q}_1\dot{q}_3 + (-q_3 Sq_2 Cq_1 + L_2 Sq_1)\dot{q}_1^2 - (q_3 Cq_1 Sq_2)\dot{q}_3^2$$

$$c_2 = 2(q_3 Cq_1 Cq_2)\dot{q}_1\dot{q}_2 + 2(Sq_1 Cq_2)\dot{q}_2\dot{q}_3$$

$$+ 2(Cq_1 Sq_2)\dot{q}_1\dot{q}_3 + (-q_3 Sq_1 Sq_2 + L_2 Sq_1)\dot{q}_1 + (-q_3 Sq_1 Sq_2)\dot{q}_2^2$$

$$c_3 = -2(Sq_2)\dot{q}_3\dot{q}_2 - q_3 Cq_2\dot{q}_2^2$$

APPENDIX B

THE DETAILS OF THE ELEMENTAL AND TRANSFORMATION MATRICES

The finite element stiffness and mass matrices and the transformation matrix are given by

$$[m_e] = \sigma A \ell$$

$$\begin{bmatrix}
\frac{1}{3} & & & & & & & & & & & \\
0 & \frac{13}{35} & & & & \text{SYMMETRIC} & & & & & & \\
0 & 0 & \frac{13}{35} & & & & & & & & & \\
0 & 0 & 0 & \frac{I_x}{3A} & & & & & & & & \\
0 & 0 & \frac{-11\ell}{210} & 0 & \frac{\ell^2}{105} & & & & & & & \\
0 & \frac{11\ell}{210} & 0 & 0 & 0 & \frac{\ell^2}{105} & & & & & & \\
\frac{1}{6} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & & & & & \\
0 & \frac{9}{70} & 0 & 0 & 0 & \frac{13\ell}{420} & 0 & \frac{13}{35} & & & & \\
0 & 0 & \frac{9}{70} & 0 & \frac{-13\ell}{420} & 0 & 0 & 0 & \frac{13}{35} & & & \\
0 & 0 & 0 & \frac{I_x}{6A} & 0 & 0 & 0 & 0 & 0 & \frac{I_x}{3A} & & \\
0 & 0 & \frac{13\ell}{420} & 0 & \frac{-\ell^2}{140} & 0 & 0 & 0 & \frac{11\ell}{210} & 0 & \frac{\ell^2}{105} & \\
0 & \frac{-13\ell}{420} & 0 & 0 & 0 & \frac{-\ell^2}{140} & 0 & \frac{-11\ell}{210} & 0 & 0 & 0 & \frac{\ell^2}{105}
\end{bmatrix} \qquad \text{(B.1)}$$

$$[k_e] = \frac{E}{\ell}\begin{bmatrix}
A & & & & & & & & & & & \\
0 & \frac{12I_z}{\ell^2} & & & & & & & & & & \\
0 & 0 & \frac{12I_y}{\ell^2} & & & & & \text{SYMMETRIC} & & & & \\
0 & 0 & 0 & \frac{GI_x}{E} & & & & & & & & \\
0 & 0 & \frac{-6I_y}{\ell} & 0 & 4I_y & & & & & & & \\
0 & \frac{6I_z}{\ell} & 0 & 0 & 0 & 4I_z & & & & & & \\
-A & 0 & 0 & 0 & 0 & 0 & A & & & & & \\
0 & \frac{-12I_z}{\ell^2} & 0 & 0 & 0 & \frac{-6I_z}{\ell} & 0 & \frac{12I_z}{\ell^2} & & & & \\
0 & 0 & \frac{-12I_y}{\ell^2} & 0 & \frac{6I_y}{\ell} & 0 & 0 & 0 & \frac{12I_y}{\ell^2} & & & \\
0 & 0 & 0 & \frac{-GI_x}{E} & 0 & 0 & 0 & 0 & 0 & \frac{GI_x}{E} & & \\
0 & 0 & \frac{-6I_y}{\ell} & 0 & 2I_y & 0 & 0 & 0 & \frac{6I_y}{\ell} & 0 & 4I_y & \\
0 & \frac{6I_z}{\ell} & 0 & 0 & 0 & 2I_z & 0 & \frac{-6I_z}{\ell} & 0 & 0 & 0 & 4I_z
\end{bmatrix} \tag{B.2}$$

$$
[TR] = \begin{bmatrix}
\ell_1 & m_1 & n_1 & & & & & & & & & \\
\ell_2 & m_2 & n_2 & & & & & & & & 0 & \\
\ell_3 & m_3 & n_3 & & & & & & & & & \\
 & & & \ell_1 & m_1 & n_1 & & & & & & \\
 & & & \ell_2 & m_2 & n_2 & & & & & & \\
 & & & \ell_3 & m_3 & n_3 & & & & & & \\
 & & & & & & \ell_1 & m_1 & n_1 & & & \\
 & & & & & & \ell_2 & m_2 & n_2 & & & \\
 & & & & & & \ell_3 & m_3 & n_3 & & & \\
 & & & & & & & & & \ell_1 & m_1 & n_1 \\
 & 0 & & & & & & & & \ell_2 & m_2 & n_2 \\
 & & & & & & & & & \ell_3 & m_3 & n_3
\end{bmatrix} \qquad (B.3)
$$

APPENDIX C

COMPUTER PROGRAM LISTING AND DESCRIPTION

Implementation of the techniques discussed in Chapters 2 and 3, were developed in Fortran Code. IMSL subroutines were called for matrix operations.

Program 'SIMPLEX' performs Linear Programming using revised simplex technique.

Program 'DISCON' performs the non-linear displacement analysis of robots of arbitrary architecture. At every successive linear approximation, subroutine simplex is called to check the optimum conditions. The optimum solution vector is based on minimum movement of the links and also satisfying all incorporated constraints.

Program 'KARMAR' is based on Karmarkar's new polynomial - time algorithm. It performs the velocity and acceleration analyses of robots for a large network of data points. Projective transformation is carried out using QR iteration and Householder reflections. The Flow Chart shown in Fig. C.1 shows the step-by-step program description.

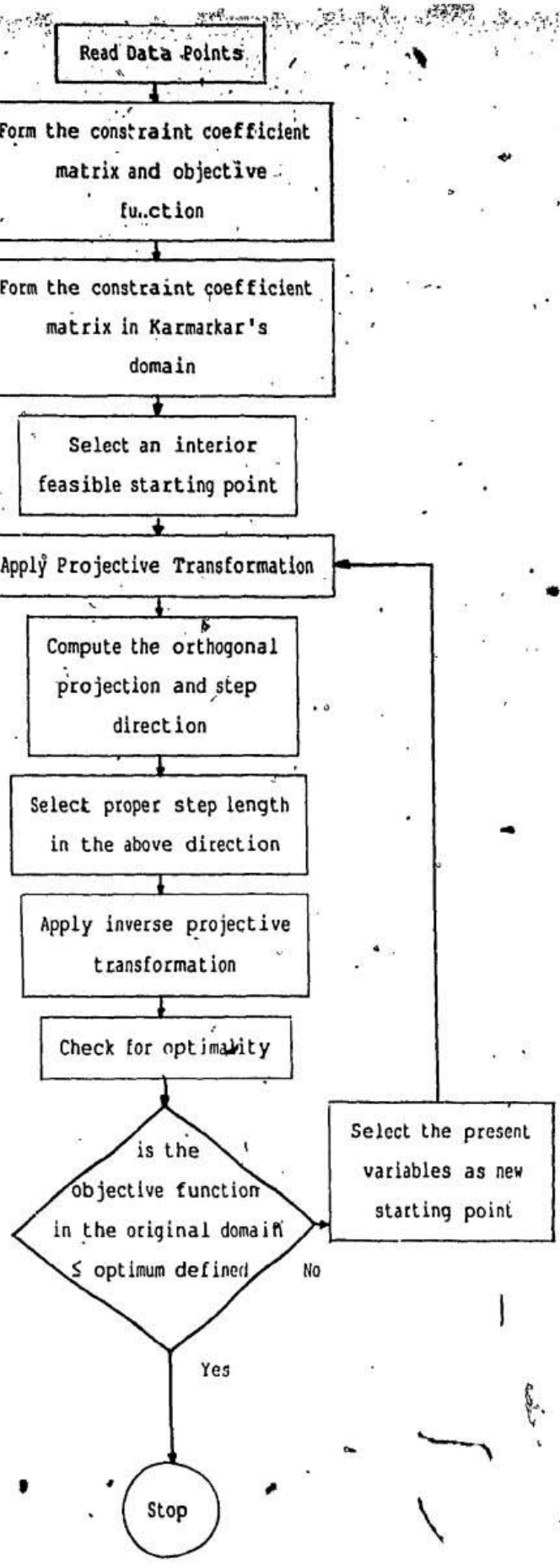

FIG. C.1: Flow Chart of the Karmarkar's Algorithm

Program 'QUARD' performs the inverse constrained kinematic analysis of robots of arbitrary architecture using Wolfe's algorithm. It can handle linear constraints. The objective function to be optimized is non-linear in nature.

Program 'FORCEAN' calculates the force vector for the carrying out the forced response of 3 link manipulator and Program 'NEWCON' performs the forced response analysis using Newmark-Beta integration scheme. The global mass and stiffness matrix is formed by 'FRECON'. The finite element analysis and natural frequency calculation is carried out in 'FRECON'.

```
C*****************************************************************
C SUBROUTINE SIMPLEX
C THIS SUBROUTINE PERFORMS LINEAR PROGRAMMING BY REVISED
C SIMPLEX TECHNIQUE BY CALLING TWO OTHER SUBROUTINES
C SUBROUTINE OPTIMUM AND SUBROUTINE SIMITER
C*****************************************************************
      SUBROUTINE SIMPLEX (N,M,A,B,C,X,U,W)
      DIMENSION A(M,1), B(1), C(1), X(1), W(1)
      COMMON /OPTI/ KO
      COMMON /SIMPLEX/ NSTOP,IDATA,NNDEX
C.... DEFAULT VALUES FOR PROGRAMMING PARAMETERS
      DATA NSTOP,IDATA,NNDEX/0,0,0/
      IF (NSTOP.NE.0) GO TO 1
      NSTOP=4*M+10
C.... NOTE THAT W MUST HAVE A DIMENSION OF M*(6+M)
1     I1=1+M*M
      I2=I1+M
      I3=I2+M
      I4=I3+M
      I5=I4+M
      CALL OPTIMUM (N,M,A,B,C,X,U,W(1),W(I1),W(I2),W(I3),W(I4),W(I5))
      RETURN
      END
      SUBROUTINE OPTIMUM (N,M,A,B,C,X,U,WORKA,WORKB,WORKC,WORKD,WORKE,WOR
     1KF)
      DIMENSION X(1), A(M,1), B(1), C(1), WORKA(1), MO(2), WORKB(1),
WOR
     1KC(1), WORKD(1), WORKE(1), WORKF(1)
      COMMON /OPTI/ KO
      COMMON /SIMPLEX/ NSTOP,IDATA,NNDEX
C     SUBROUTINE OPTIMUM IS USED AS A MEANS TO CALCULATE
C     A VALUE OF THE OBJECTIVE FUNCTION AT THE OPTIMUM CONDITIONS
C     OR IF THE SOLUTION IS NOT VALID THIS SUBROUTINE THEN OUTPUTS
C     THE DIAGNOSTIC MESSAGES
C     THE ACTUAL ITERATIVE PROCESS OF THE REVISED SIMPLEX TECHNIQUE
IS
```

```
C     PERFORMED IN SUBROUTINE SIMITER
      IF (IDATA.NE.1) GO TO 1
      WRITE (6,18) IDATA
      WRITE (6,24) N
      WRITE (6,23) M
      WRITE (6,19) NSTOP
      WRITE (6,20) (B(J),J=1,M)
      WRITE (6,21) (C(I),I=1,N)
      WRITE (6,22) ((A(I,J),J=1,N),I=1,M)
1     DO 2 I=1,M
      WORKB(I)=0.0
      WORKC(I)=0.0
      WORKD(I)=0.0
      WORKE(I)=0.0
      WORKF(I)=0.0
      DO 2 J=1,M
2     WORKA(IM(I,J,M))=0.
      CALL SIMITER (M,N,MO,X,WORKA,A,B,C,WORKB,WORKC,WORKD,WORKE,WORKF)
      IF (MO(1).GT.5) GO TO 6
      MODE1=MO(1)+1
      GO TO (8,3,4,3,5,6), MODE1
3     WRITE (6,14)
3     GO TO 7
4     WRITE (6,15)
      GO TO 7
5     WRITE (6,16) MO(2)
      GO TO 7
6     WRITE (6,17) MO(2)
7     KO=1
      GO TO 10
8     U=0.0
      DO 9 J=1,N
      U=U+C(J)*X(J)
9     CONTINUE
      IF (NNDEX.GT.0) GO TO 10
      WRITE (6,11)
```

```
      WRITE (6,12) U
      WRITE (6,13) (I,X(I),I=1,1)
10    RETURN
11    FORMAT (1HO,22X,36HOPTIMUM  SOLUTION  FOUND  BY  SIMPLE/23X,36H---
     1--------------------------------/)
12    FORMAT (20X,12HMINIMUM  U =,E16.8//)
13    FORMAT (25X,2HX(,I2,3H) =,E16.8)
14    FORMAT (1X,44H NO FEASIBLE SOLUTION CAN BE FOUND BY SIMPLE)
15    FORMAT (1HO,43HTHE SIMPLEX ROUTINE FOUND UNBOUNDED OPTIMUM)
16    FORMAT (1HO,97HTHE MAXIMUM ALLOWABLE NO OF ITERATIONS FOR
SIMPLEX
     1HAS BEEN EXCEEDED,--SOLUTION IS STILL FEASIBLE/1HO,17HNO OF
ITERAT
     2IONS=,I5)
17    FORMAT (1HO,85HNO FEASIBLE SOLUTION EXISTS FOR SIMPLEX-PROGRAM
STO
     1PPED ON ALLOWABLE NO OF ITERATIONS/1HO,17HNO OF ITERATIONS=,I5)
18    FORMAT (61HOINPUT DATA IS PRINTED OUT FOR IDATA=1 ONLY. .
. .   ID
     1ATA  =,I6)
19    FORMAT (61HONUMBER OF ITERATIONS PERMITTED. . . . . .
. .   NS
     1TOP  =,I6)
20    FORMAT (61HORIGHT HAND SIDE OF SIMPLEX ARRAY. . . . . .
. .    B
     1(M)  =,//(5E16.8))
21    FORMAT (61HOCOEFFICIENTS OF SIMPLEX OBJECTIVE FUNCTION. .
. .    C
     1(N)  =,//(5E16.8))
22    FORMAT (61HOCOEFFICIENTS OF SIMPLEX CONSTRAINT EQUATIONS.
. .  A(M
     1,N)  =,//(5E16.8))
23    FORMAT (61HONUMBER OF EQUATIONS . . . . . .
. . . .
     1. M  =,I6)
24    FORMAT (61HONUMBER OF INDEPENDENT VARIABLES . . . . . .
```

```
     1  N  =,I6)
        END
        SUBROUTINE SIMITER (M,N,KO,KB,E,A,B,C,P,X,Y,PE,JH)
        DIMENSION B(1), C(1), E(1), KO(2), KB(1), A(M,1), P(1), X(1),
    1Y(1)
       1, PE(1), JH(1)
        COMMON /SIMPLEX/ NSTOP,IDATA,NNDEX
        EQUIVALENCE (XX,LL)
        LOGICAL FEAS,VER,NEG,TRIG,KQ,ABSC
C       THE PURPOSE OF THE SUBROUTINE SIMP IS TO PERFORM THE ITERATIVE
C       METHOD OF LINEAR PROGRAMMING KNOWN AS THE SIMPLEX METHOD
        ITER=0
        NUMVR=0
        NUMPV=0
        TEXP=.5**16
        NVER=M/2+5
        M2=M*M
        FEAS=.FALSE.
        DO 2 J=1,N
        KB(J)=0
        KQ=.FALSE.
        DO 1 I=1,M
        IF (A(I,J).EQ.0.0) GO TO 1
        IF (KQ.OR.A(I,J).LT.0.0) GO TO 2
        KQ=.TRUE.
    1   CONTINUE
        KB(J)=1
    2   CONTINUE
        DO 3 I=1,M
        JH(I)=-1
    3   CONTINUE
        NUMVR=NUMVR+1
        INVC=0
    4   VER=.TRUE.
        TRIG=.FALSE.
```

```
      DO 5 I=1,M2
      E(I)=0.0
5     CONTINUE
      MM=1
      DO 6 I=1,M
      E(MM)=1.0
      PE(I)=0.0
      X(I)=B(I)
      IF (JH(I).NE.0) JH(I)=-1
      MM=MM+M+1
6     CONTINUE
      DO 13 JT=1,N
      IF (KB(JT).EQ.0) GO TO 13
      GO TO 29
7     TY=0.0
      KQ=.FALSE.
      DO 12 I=1,M
      IF (JH(I).NE.-1.OR.ABS(Y(I)).LE.TPIV) GO TO 12
      IF (KQ) GO TO 9
      IF (X(I).EQ.0.) GO TO 8
      IF (ABS(Y(I)/X(I)).LE.TY) GO TO 12
      TY=ABS(Y(I)/X(I))
      GO TO 11
8     KQ=.TRUE.
      GO TO 10
9     IF (X(I).NE.0..OR.ABS(Y(I)).LE.TY) GO TO 12
10    TY=ABS(Y(I))
11    IR=I
12    CONTINUE
      KB(JT)=0
      IF (TY.LE.0.) GO TO 13
      GO TO 42
13    CONTINUE
      DO 14 I=1,M
      IF (JH(I).EQ.-1) JH(I)=0
      IF (JH(I).EQ.0) FEAS=.FALSE.
```

```
14    CONTINUE
15    VER=.FALSE.
      NEG=.FALSE.
      IF (FEAS) GO TO 17
      FEAS=.TRUE.
      DO 16 I=1,M
      IF (X(I).LT.0.0) GO TO 19
      IF (JH(I).EQ.0) FEAS=.FALSE.
16    CONTINUE
      IF (.NOT.FEAS) GO TO 20
17    DO 18 I=1,M
      P(I)=PE(I)
      IF (X(I).LT.0.) X(I)=0.
18    CONTINUE
      ABSC=.FALSE.
      GO TO 26
19    FEAS=.FALSE.
      NEG=.TRUE.
20    DO 21 J=1,M
      P(J)=0.
21    CONTINUE
      ABSC=.TRUE.
      DO 25 I=1,M
      MM=I
      IF (X(I).GE.0.0) GO TO 23
      ABSC=.FALSE.
      DO 22 J=1,M
      P(J)=P(J)+E(MM)
      MM=MM+M
22    CONTINUE
      GO TO 25
23    IF (JH(I).NE.0) GO TO 25
      IF (X(I).NE.0.) ABSC=.FALSE.
      DO 24 J=1,M
      P(J)=P(J)-E(MM)
      MM=MM+M
```

```
24    CONTINUE
25    CONTINUE
26    JT=0
      BB=0.0
      DO 28 J=1,N
      IF (KB(J).NE.0) GO TO 28
      DT=0.0
      DO 27 I=1,M
      DT=DT+P(I)*A(I,J)
27    CONTINUE
      IF (FEAS) DT=DT+C(J)
      IF (ABSC) DT=-ABS(DT)
      IF (DT.GE.BB) GO TO 28
      BB=DT
      JT=J
28    CONTINUE
      IF (JT.LE.0) GO TO 49
      IF (ITER.GE.NSTOP) GO TO 48
      ITER=ITER+1
      ITER=ITER+1
29    DO 30 I=1,M
      Y(I)=0.0
30    CONTINUE
      LL=0
      COST=C(JT)
      DO 33 I=1,M
      AIJT=A(I,JT)
      IF (AIJT.EQ.0.) GO TO 32
      COST=COST+AIJT*PE(I)
      DO 31 J=1,M
      LL=LL+1
      Y(J)=Y(J)+AIJT*E(LL)
31    CONTINUE
32    LL=LL+M
33    CONTINUE
      YMAX=0.0
```

```
      DO 34 I=1,M
      YMAX=AMAX1(ABS(Y(I)),YMAX)
34    CONTINUE
      TPIV=YMAX*TEXP
      IF (VER) GO TO 7
      RCOST=YMAX/BB
      IF (TRIG.AND.BB.GE.-TPIV) GO TO 49
      TRIG=.FALSE.
      IF (BB.GE.-TPIV) TRIG=.TRUE.
      IR=0
      AA=0.0
      KQ=.FALSE.
      DO 38 I=1,M
      IF (X(I).NE.0.0.OR.Y(I).LE.TPIV) GO TO 38
      IF (JH(I).EQ.0) GO TO 36
      IF (KQ) GO TO 38
35    IF (Y(I).LE.AA) GO TO 38
      GO TO 37
36    IF (KQ) GO TO 35
      KQ=.TRUE.
37    AA=Y(I)
      IR=I
38    CONTINUE
      IF (IR.NE.0) GO TO 41
      AA=1.0E+20
C     FIND MINIMUM PIVOT AMONG POSITIVE EQUATIONS
      DO 39 I=1,M
      IF (Y(I).LE.TPIV.OR.X(I).LE.0.0.OR.Y(I)*AA.LE.X(I)) GO TO
39
      AA=X(I)/Y(I)
      IR=I
39    CONTINUE
      IF (.NOT.NEG) GO TO 41
C     FIND PIVOT AMONG NEGATIVE EQUATIONS, IN WHICH X/Y IS LESS
THAN THE
C     MINIMUM X/Y IN THE POSITIVE EQUATIONS, THAT HAS THE LARGEST
```

```
C     ABSF(Y)
      BB=-TPIV
      DO 40 I=1,M
      IF (X(I).GE.0..OR.Y(I).GE.BB.OR.Y(I)*AA.GT.X(I)) GO TO 40
      BB=Y(I)
      IR=I
40    CONTINUE
C     TEST FOR NO PIVOT ROW
41    IF (IR.LE.0) GO TO 47
      IA=JH(IR)
      IF (IA.GT.0) KB(IA)=0
42    NUMPV=NUMPV+1
      JH(IR)=JT
      KB(JT)=IR
      YI=-Y(IR)
      Y(IR)=-1.0
      LL=0
      DO 45 J=1,M
      L=LL+IR
      IF (E(L).NE.0.0) GO TO 43
      LL=LL+M
      GO TO 45
43    XY=E(L)/YI
      PE(J)=PE(J)+COST*XY
      E(L)=0.0
      DO 44 I=1,M
      LL=LL+1
      E(LL)=E(LL)+XY*Y(I)
44    CONTINUE
45    CONTINUE
C     TRANSFORM X
      XY=X(IR)/YI
      DO 46 I=1,M
      XOLD=X(I)
      X(I)=XOLD+XY*Y(I)
      IF (.NOT.VER.AND.X(I).LT.0..AND.XOLD.GE.0.) X(I)=0.
```

```
46    CONTINUE
      Y(IR)=-YI
      X(IR)=-XY
      IF (VER) GO TO 13
      IF (NUMPV.LE.M) GO TO 15
C     TEST FOR INVERSION ON THIS ITERATION
      INVC=INVC+1
      IF (INVC.EQ.NVER) GO TO 4
      GO TO 15
47    IF (.NOT.FEAS.OR.RCOST.LE.-1000.) GO TO 49
      K=2
      GO TO 50
48 K=4
GO TO 50
C     FEASIBLE OR INFEASIBLE SOLUTION
49    K=0
50    IF (.NOT.FEAS) K=K+1
      DO 51 J=1,N
      XX=0.0
      KBJ=KB(J)
      IF (KBJ.NE.0) XX=X(KBJ)
      KB(J)=LL
51    CONTINUE
      KO(1)=K
      KO(2)=ITER
      RETURN
      END
FUNCTION IM(I,J,M)
IM=M*(J-1)+I
RETURN
END
```

```
C*****************************************************************C
C ***PROGRAM DISCON ****                                           C
C NON LINEAR DISPLACEMENT ANALYSES OF ROBOTS OF ARBITRARY          C
C ARCHITECTURE                                                     C
C IN THE PRESENT PROGRAM CESAR ROBOT IS ANALYSED                   C
C*****************************************************************
      DIMENSION Y(7),PL(4),R(3),Z(4),PY(4)
      DIMENSION A(11,12),B(11),C(12),X(12),W(2000)
      DIMENSION A01(4,4),A12(4,4),A23(4,4),A34(4,4),A04(4,4)
      DIMENSION TEMP1(4,4),TEMP2(4,4),WK(1000)
      DIMENSION OF(3,3),T04INV(4,4),T04IN(3,3),ORT(3,3)
            COMMON /SIMPLE/ NSTOP,IDATA,NNDEX
            COMMON /OPTI/ KO
            DATA NSTOP,IDATA,NNDEX/0,0,0/
      TR=22./(7.*180.)
      N=12
      M=11
C APPROXIMATE STARTING SOLUTION
      Y(1)=150.*TR
      Y(2)=360.*TR
      Y(3)=50.*TR
      Y(4)=36.*TR
C SUCCESSIVE APPROXIMATION LIMITS
      DO 23 I=1,4
   23 PL(I)=1.75
      C1=COS(Y(1))
      C2=COS(Y(2))
      C3=COS(Y(3))
      C4=COS(Y(4))
      S1=SIN(Y(1))
      S2=SIN(Y(2))
      S3=SIN(Y(3))
      S4=SIN(Y(4))
C POSITION OF A POINT ON THE PLANNED TRAJECTORY OF THE
C THE TASK IN GLOBAL COORDINATES
      R(3)=-.6
```

```
      R(2)=.40
      R(1)=.2
      DO 1000 KKK=1,10
      60 TR=22./(7.*180.)
      C INPUT LINK PARAMETERS
      D2=.356
      D3=.635
      A3=.029
      A4=.508
      C RESIDUAL FUNCTIONS AND LINEARIZED DISPLACEMENT COMPONENTS
      F10=A4*(C1*C2*C3*C4-S1*S3*C4)-A4*S2*S4*C1+A3*C1*C2*C3
           1 -A3*S1*S3+D3*C1*S2+D2*S1-R(1)
      DELF11=A4*(-S1*C2*C3*C4-C1*S3*C4)+A4*S2*S4*S1-A3*S1*C2*C3
           1 -A3*C1*S3-D3*S1*S2+D2*C1
      DELF12=A4*(-C1*S2*C3*C4)-A4*C2*S4*C1-A3*C1*S2*C3
           1 +D3*C1*C2
      DELF13=A4*(-C1*C2*S3*C4-S1*C3*C4)-A3*C1*C2*S3
           1 -A3*S1*C3
      DELF14=A4*(-C1*C2*C3*S4+S1*S3*S4)-A4*S2*C4*C1
      F20=A4*(S1*C2*C3*C4+C1*S3*C4)-A4*S1*S2*S4+A3*S1*C2*C3
           1 +A3*C1*S3+D3*S1*S2-D2*C1-R(2)
      DELF21=A4*(C1*C2*C3*C4-S1*S3*C4)-A4*C1*S2*S4+A3*C1*C2*C3
           1 -A3*S1*S3+D3*C1*S2+D2*S1
      DELF22=A4*(-S1*S2*C3*C4)-A4*S1*C2*S4-A3*S1*S2*C3
           1 +D3*S1*C2
      DELF23=A4*(-S1*C2*S3*C4+C1*C3*C4)-A3*S1*C2*S3
           1 +A3*C1*C3
      DELF24=A4*(-S1*C2*C3*S4-C1*S3*S4)-A4*S1*S2*C4
      F30=A4*S2*C3*C4+A4*C2*S4+A3*S2*C3-D3*C2-R(3)
      DELF31=0.0
      DELF32=A4*C2*C3*C4-A4*S2*S4+A3*C2*C3+D3*S2
      DELF33=-A4*S2*S3*C4-A3*S2*S3
      DELF34=-A4*S2*C3*S4+A4*C2*C4
      C LINEARIZED EQUATION FOR MINIMUM MOVEMENT OF LINKS
      DELF41=2.*(Y(1)-PY(1))
      DELF42=2.*(Y(2)-PY(2))
```

```
      DELF43=2.*(Y(3)-PY(3))
      DELF44=2.*(Y(4)-PY(4))
C     AC=LARGE WEIGHTED NUMBER
      AC=1000.
C     COMPONENTS OF LINEARIZED OBJECTIVE FUNCTION
      DELG1=((2.*F10*DELF11+2.*F20*DELF21+2.*F30*DELF31)*AC+DELF41)
      DELG2=((2.*F10*DELF12+2.*F20*DELF22+2.*F30*DELF32)*AC+DELF42)
      DELG3=((2.*F10*DELF13+2.*F20*DELF23+2.*F30*DELF33)*AC+DELF43)
      DELG4=((2.*F10*DELF14+2.*F20*DELF24+2.*F30*DELF34)*AC+DELF44)
      DO 10 I=1,11
      DO 10 J=1,12
   10 A(I,J)=0.0
      A(1,1)=DELF11
      A(1,2)=DELF12
      A(1,3)=DELF13
      A(1,4)=DELF14
      A(2,1)=DELF21
      A(2,2)=DELF22
      A(2,3)=DELF23
      A(2,4)=DELF24
      A(3,1)=DELF31
      A(3,2)=DELF32
      A(3,3)=DELF33
      A(3,4)=DELF34
      A(4,1)=1.
      A(4,5)=-1.
      A(5,2)=1.
      A(5,6)=-1.
      A(6,3)=1.
      A(6,7)=-1.
      A(7,4)=1.
      A(7,8)=-1.
      A(8,1)=1.
      A(8,9)=1.
      A(9,2)=1.
      A(9,10)=1.
```

```
      A(10,3)=1.
      A(10,11)=1.
      A(11,4)=1.
      A(11,12)=1.
      B(1)=-F10+(Y(1)*DELF11+Y(2)*DELF12+Y(3)*DELF13+Y(4)*DELF14)
      B(2)=-F20+(Y(1)*DELF21+Y(2)*DELF22+Y(3)*DELF23+Y(4)*DELF24)
      B(3)=-F30+(Y(1)*DELF31+Y(2)*DELF32+Y(3)*DELF33+Y(4)*DELF34)
      B(4)=Y(1)-(PL(1)*TR)
      B(5)=Y(2)-(PL(2)*TR)
      B(6)=Y(3)-(PL(3)*TR)
      B(7)=Y(4)-(PL(4)*TR)
      B(8)=(PL(1)*TR)+Y(1)
      B(9)=(PL(2)*TR)+Y(2)
      B(10)=(PL(3)*TR)+Y(3)
      B(11)=(PL(4)*TR)+Y(4)
      DO 20 I=1,12
   20 C(I)=0.0
      C(1)=DELG1
      C(2)=DELG2
      C(3)=DELG3
      C(4)=DELG4
C     SUBROUTINE SIMPLEX IS CALLED
      CALL SIMPLEX(N,M,A,B,C,X,U,W)
      DO 301 I=1,4
      PY(I)=X(I)
  301 Y(I)=X(I)
      XXO=Y(1)/TR
      YYO=Y(2)/TR
      ZZO=Y(3)/TR
      UUO=Y(4)/TR
      C1=COS(Y(1))
      C2=COS(Y(2))
      C3=COS(Y(3))
      C4=COS(Y(4))
      S1=SIN(Y(1))
      S2=SIN(Y(2))
```

```
S3=SIN(Y(3))
S4=SIN(Y(4))
F3=A4*S2*C3*C4+A4*C2*S4+A3*S2*C3-D3*C2-R(3)
F2=A4*(S1*C2*C3*C4+C1*S3*C4)-A4*S1*S2*S4+A3*S1*C2*C3
     1 +A3*C1*S3+D3*S1*S2-D2*C1-R(2)
F1=A4*(C1*C2*C3*C4-S1*S3*C4)-A4*S2*S4*C1+A3*C1*C2*C3
     1 -A3*S1*S3+D3*C1*S2+D2*S1-R(1)
C CHECK FOR OPTIMUM
C FOR THE NONLINEAR OBJECTIVE FUNCTION
OBJ=F1**2+F2**2+F3**2
WRITE(6,*)OBJ
UTOL=1.E-06
UDIF=OBJ-UTOL
IF(UDIF) 40,40,50
50 GO TO 60
40 WRITE(6,*)OBJ
WRITE(6,*)XXO,YYO,ZZO,UUO.
C ANALYSES OF WRIST ANGLES
DO 75 I=1,4
DO 75 J=1,4
AO1(I,J)=0.0
A12(I,J)=0.0
A23(I,J)=0.0
75 A34(I,J)=0.0
AO1(1,1)=C1
AO1(1,3)=S1
AO1(2,1)=S1
AO1(2,3)=-C1
AO1(3,2)=1.
AO1(4,4)=1.
A12(1,1)=C2
A12(1,3)=-S2
A12(2,1)=S2
A12(2,3)=C1
A12(3,2)=-1.
A12(3,4)=D2
```

```
      A12(4,4)=1.
      A23(1,1)=C3
      A23(1,3)=S3
      A23(1,4)=A3*C3
      A23(2,1)=S3
      A23(2,3)=-C3
      A23(2,4)=A3*S3
      A23(3,2)=1.
      A23(3,4)=-D3
      A23(4,4)=1.
      A34(1,1)=C4
      A34(1,2)=-S4
      A34(1,4)=A4*C4
      A34(2,1)=S4
      A34(2,3)=C4
      A34(2,4)=A4*S4
      A34(3,3)=1.
      A34(4,4)=1.
      CALL VMULFF(A01,A12,4,4,4,4,4,TEMP1,4,IER)
      CALL VMULFF(TEMP1,A23,4,4,4,4,4,TEMP2,4,IER)
      CALL VMULFF(TEMP2,A34,4,4,4,4,4,A04,4,IER)
      IDGT=4
      CALL LINV2F(A04,4,4,T04INV,IDGT,WK,IER)
      DO 76 I=1,3
      DO 76 J=1,3
      ORT(I,J)=0.0
   76 T04IN(I,J)=T04INV(I,J)
C     ROTATION SUBMATRIX OF THE ARM MATRIX
      ORT(1,2)=1.
      ORT(2,1)=1.
      ORT(3,3)=-1.
      CALL VMULFF(T04IN,ORT,3,3,3,3,3,OF,3,IER)
      Y(5)=ATAN(OF(2,3)/OF(1,3))
      C5=COS(Y(5))
      S5=SIN(Y(5))
      Y(6)=ATAN((C5*OF(1,3)+S5*OF(2,3))/OF(3,3))
```

```
Y(7)=ATAN((-S5*OF(1,1)+C5*OF(2,1))/(-S5*OF(1,2)+C5*OF(2,2)))
TH5=Y(5)/TR
TH6=Y(6)/TR
TH7=Y(7)/TR
WRITE(6,*)'WRIST',TH5,TH6,TH7
C NEXT POINT ON THE PLANNED TRAJECTORY IS SELECTED
DELR=.05
R(2)=R(2)-DELR
1000 CONTINUE
STOP
END
```

```
C*************************************************************C
C PROGRAM ***KARMKR***                                        C
C THIS PROGRAM PERFORMS THE VELOCITY ANALYSES OF ROBOT        C
C FOR A LARGE NETWORK OF DATA POINTS                          C
C IT USES KARMARKAR'S ALGORITHM OF LINEAR PROGRAMMING         C
C PROJECTIVE TRANSFORMATION IS CARRIED OUT USING              C
C QR ITERATION AND HOUSE HOLDER REFLECTIONS                   C
C ALFA IS THE STEP FOR EVERY ITERATION                        C
C RADII IS THE RADIUS OF SPHERE FOR EVERY JUMP                C
C SCLE=PLAUSIBLE UPPER BOUND ON ROWS OF A                     C
C*************************************************************C
  PARAMETER (LAL=132,LM=134,LMM=135,LN=267,LR=3,LC=6,LALT=264)
REAL B(LMM,LN),C(LN,1),TEMP1(LMM,LMM),AAA(132,264)
REAL TEMP4(LN,LN),BBT(135,135),BTY(267,1),RES(267,1)
REAL TEMP5(LN,LN),FX(LN),Y(LAL),YBAR(267,1),PR(132,1)
REAL DIFF(LN,LN),DELTA(LN,1),DC(267,1),DCC(267),CC(135)
REAL BT(LN,LMM),PELTAI(LN,1),DELGV1(66),DELGV2(66),DELGV3(66)
REAL AX(LN,1),X(LN,1),AA(LM,LN),PROD(LN,1),CX(1,LN),DELGV4(66)
REAL A1(33,66),TEMP8(1,LN),TEMP7(LN,1),TEMP9(1,1)
REAL APROD(LN,1),Y1(33),RV(3),BB(LAL),A3(33,66),A4(33,66)
REAL A2(33,66),ANGV(LAL),Y2(33),Y3(33)
REAL Y4(33),TEMP22(LMM,LN),W(100000)
INTEGER IPVT(LMM)
REAL QR(LN,LMM),QRAUX(LMM),CONORM(LMM),QB(1),QTB(LN),AAX(1)
COMMON/WORKSP/RWKSP
REAL RWKSP(36875)
C*********************************************
C READ THE DATA POINTS FROM OUT1.DAT
C*********************************************
OPEN(UNIT=18,FILE='OUT1.DAT',TYPE='OLD')
READ(18,*)(Y(I),I=1,LAL)
CALL IWKIN(36875)
TR=22./(180.*7.)
DO 219 I=1,33
219 Y1(I)=Y(I)*TR
DO 220 I=34,66
```

```
220 Y2(I-33)=Y(I)*TR
DO 221 I=67,99
221 Y3(I-66)=Y(I)*TR
DO 222 I=100,132
222 Y4(I-99)=Y(I)*TR
M=134
MM=M+1
RV(1)=0.
RV(3)=0.
RV(2)=-0.0127
DO 21 I=1,3
21 BB(I)=RV(I)
DO 24 J=3,129,3
DO 22 I=1,3
22 BB(I+J)=RV(I)
24 CONTINUE
TN=267.
ALFA=.99
SCLE=BB(2)*10000.
RADII=1./((TN*(TN-1.))**.5)
N=267
AN=264.
C*********************************************
C A FEASIBLE STARTING POINT
C*********************************************
DO 3043 I=1,AN
DO 3043 J=1,1
3043 X(I,J)=1./(4.*AN)
DO 3054 I=265,267
J=1
3054 X(I,J)=1./4.
C*********************************************
C INPUT LINK PARAMETERS
C*********************************************
2111 D2=1.016
D3=1.5113
```

```
C*********************************************
C HERE FORMATION OF CONSTRAINT COEFFCIENT MATRIX STARTS
C DELGVI ARE THE COEFFICIENT OF OBJECTIVE FUNCTIONS
C*********************************************
CALL CONSTANT(Y1,D2,D3,A1,DELGV1)
CALL CONSTANT(Y2,D2,D3,A2,DELGV2)
CALL CONSTANT(Y3,D2,D3,A3,DELGV3)
CALL CONSTANT(Y4,D2,D3,A4,DELGV4)
DO 499 I=1,LM
DO 499 J=1,LN
499 AA(I,J)=0.0
DO 3005 I=1,33
DO 3005 J=1,66
3005 AA(I,J)=A1(I,J)
DO 500 I=1,33
'DO 500 J=1,66'
500 AA(I+33,J+66)=A2(I,J)
DO 501 I=1,33
DO 501 J=1,66
501 AA(I+66,J+132)=A3(I,J)
DO 502 I=1,33
DO 502 J=1,66
502 AA(I+99,J+198)=A4(I,J)
DO 3006 I=1,LAL
J=LALT+1
3006 AA(I,J)=-BB(I)/SCLE
DO 3008 I=1,LAL
SUM=0.0
DO 3007 J=1,LALT
SUM=SUM+AA(I,J)
3007 CONTINUE
AA(I,LN)=(BB(I)/SCLE)-(SUM/AN)
3008 CONTINUE
DO 3009 I=133,134
DO 3009 J=1,LALT
3009 AA(I,J)=1.
```

```
      AA(133,265)=-1.
      AA(133,266)=1.
      AA(133,267)=-1.
      AA(134,265)=1.
      AA(134,266)=1.
      AA(134,267)=1.
      DO 508 I=1,LN
      J=1
      508 C(I,J)=0.0
      DO 2954 I=1,66
      DO 2954 J=1,1
      2954 C(I,J)=DELGV1(I)
      DO 505 I=1,66
      J=1
      505 C(I+66,J)=DELGV2(I)
      DO 506 I=1,66
      J=1
      506 C(I+132,J)=DELGV3(I)
      DO 507 I=1,66
      J=1
      507 C(I+198,J)=DELGV4(I)
      C**********************************************
      C KARMARKAR'S TIME POLYNOMIAL STARTS
      C**********************************************
      1 DO 744 I=1,M
      DO 745 J=1,N
      IF(AA(I,J).EQ.0.0) GO TO 745
      B(I,J)=AA(I,J)*X(J,1)
      745 CONTINUE
      744 CONTINUE
      DO 6 J=1,N
      6 B(MM,J)=1.0
      DO 29 I=1,MM
      DO 29 J=1,N
      29 BT(J,I)=B(I,J)
      DO 39 I=1,N
```

```
      DC(I,1)=X(I,1)*C(I,1)
   39 DCC(I)=DC(I,1)
      TOL=1E-06
      NRA=N
      NCA=MM
      LDA=N
      LDQR=N
      KBASIS=135
      LOGICAL=.TRUE.
      PIVOT=.TRUE.
C*********************************************
C IMSL SUBROUTINE
C LQRRR PERFORMS THE QR ITERATION USING
C HOUSE HOLDER REFLECTIONS
C*********************************************
      CALL ISET(NCA,0,IPVT,1)
      CALL LQRRR(NRA,NCA,BT,LDA,PIVOT,IPVT,QR,LDQR,QRAUX,
     1  CONORM)
        IPATH=00100
C*********************************************
C LQRSL PERFORMS THE LEAST SQUARE SOLUTION USING FACTORISED
C Q AND R MATRIX
C*********************************************
      CALL LQRSL(NRA,KBASIS,QR,LDQR,QRAUX,DC,IPATH,QB,QTB,YBAR
     1  ,RES,AAX)
C*********************************************
C SUBTRACTION BY UNIT VECTOR
C*********************************************
      DO 678 I=1,N
  678 BTY(I,1)=0.0
      DO 679 I=1,N
      DO 679 J=1,MM
  679 BTY(I,1)=BTY(I,1)+BT(I,J)*YBAR(J,1)
      DO 40 I=1,N
   40 PELTAI(I,1)=DC(I,1)-BTY(I,1)
C*********************************************
```

```
C SELECTING THE STEP SIZE
C*********************************************
SUM=0.0
DO 320 I=1,N
320 SUM=SUM+PELTAI(I,1)**2
DELMAG=SQRT(SUM)
DO 104 I=1,N
DO 104 J=1,1
104 DELTA(I,J)=(PELTAI(I,J))/DELMAG
TLAMDA=ALFA*RADII
DO 33 I=1,N
DO 33 J=1,1
33 PROD(I,J)=TLAMDA*DELTA(I,J)
C*********************************************
C SOLUTION IN KARMARKAR'S DOMAIN
C*********************************************
DO 45 I=1,N
DO 45 J=1,1
AX(I,J)=(1./TN)-PROD(I,J)
45 CONTINUE
DO 4670 I=1,N
4670 TEMP7(I,1)=X(I,1)*AX(I,1)
PP=0.0
DO 60 I=1,N
60 PP=PP+TEMP7(I,1)
C*********************************************
C SOLUTION IN TRANSFORMED SPACE
C*********************************************
DO 331 I=1,N
DO 331 J=1,1
331 X(I,J)=TEMP7(I,J)/PP
C*********************************************
C ACTUAL SOLUTION AFTER ITERATION
C*********************************************
DO 310 I=1,LALT
310 FX(I)=2.*SCLE*X(I,1)
```

```
SUM=0.0
DO 140 I=1,LALT
140 SUM=SUM+C(I,1)*FX(I)
BSUM=0.0
DO 141 I=1,LAL
141 BSUM=BSUM+BB(I)
C*********************************************
C CHECK FOR OPTIMUM. A STOPPING CRITERION IS SELECTED
C*********************************************
OBJV=(-SUM-BSUM)**2
WRITE(6,*)'OBJV',OBJV
IF(OBJV.LT.1E-7)GO TO 123
GO TO 1
123 J=1
C*********************************************
C CALCULATION OF NONRESTRICTED VARIABLES
C*********************************************
DO 476 I=1,LAL
ANGV(I)=FX(J)-FX(J+1)
476 J=J+2
WRITE(6,*)'ANGV',ANGV
122 WRITE(6,*)'OBJV',OBJV
STOP
END
C*****************************************************************C
C SUBROUTINE CONSTANT FORMS GLOBAL CONSTRAINT COEFFCIENT   C
C MATRIX AND CALCULATES THE COEFFCIENT OF OBJECTIVE        C
C FUNCTION                                                 C
C*****************************************************************C
SUBROUTINE CONSTANT(Y,D2,D3,A,DELGV)
DIMENSION Y(33),A(33,66),DELGV(66)
DIMENSION CON1(3,3),CON2(3,3),CON3(3,3),CON4(3,3)
DIMENSION CON5(3,3),CON6(3,3),CON7(3,3),CON8(3,3)
DIMENSION CON9(3,3),CON10(3,3),CON11(3,3),CON12(3,3)
CON1(1,1)=-D2*COS(Y(2))*SIN(Y(1))-D3*SIN(Y(1))*COS(Y(2)+Y(3))
CON1(1,2)=-D2*COS(Y(1))*SIN(Y(2))-D3*COS(Y(1))*SIN(Y(2)+Y(3))
```

```
      CON1(1,3)=-D3*COS(Y(1))*SIN(Y(2)+Y(3))
      CON1(2,1)=D2*COS(Y(2))*COS(Y(1))+D3*COS(Y(1))*COS(Y(2)+Y(3))
      CON1(2,2)=-D2*SIN(Y(2))*SIN(Y(1))-D3*SIN(Y(1))*SIN(Y(2)+Y(3))
      CON1(2,3)=-D3*SIN(Y(1))*SIN(Y(2)+Y(3))
      CON1(3,1)=0.0
      CON1(3,2)=D2*COS(Y(2))+D3*COS(Y(2)+Y(3))
      CON1(3,3)=D3*COS(Y(2)+Y(3))
      CON2(1,1)=-D2*COS(Y(5))*SIN(Y(4))-D3*SIN(Y(5))*COS(Y(5)+Y(6))
      CON2(1,2)=-D2*COS(Y(4))*SIN(Y(5))-D3*COS(Y(4))*SIN(Y(5)+Y(6))
      CON2(1,3)=-D3*COS(Y(4))*SIN(Y(5)+Y(6))
      CON2(2,1)=D2*COS(Y(5))*COS(Y(4))+D3*COS(Y(4))*COS(Y(5)+Y(6))
      CON2(2,2)=-D2*SIN(Y(5))*SIN(Y(4))-D3*SIN(Y(4))*SIN(Y(5)+Y(6))
      CON2(2,3)=-D3*SIN(Y(4))*SIN(Y(5)+Y(6))
      CON2(3,1)=0.0
      CON2(3,2)=D2*COS(Y(5))+D3*COS(Y(5)+Y(6))
      CON2(3,3)=D3*COS(Y(5)+Y(6))
      CON3(1,1)=-D2*COS(Y(8))*SIN(Y(7))-D3*SIN(Y(7))*COS(Y(8)+Y(9))
      CON3(1,2)=-D2*COS(Y(7))*SIN(Y(8))-D3*COS(Y(7))*SIN(Y(8)+Y(9))
      CON3(1,3)=-D3*COS(Y(7))*SIN(Y(8)+Y(9))
      CON3(2,1)=D2*COS(Y(8))*COS(Y(7))+D3*COS(Y(7))*COS(Y(8)+Y(9))
      CON3(2,2)=-D2*SIN(Y(8))*SIN(Y(7))-D3*SIN(Y(7))*SIN(Y(8)+Y(9))
      CON3(2,3)=-D3*SIN(Y(7))*SIN(Y(8)+Y(9))
      CON3(3,1)=0.0
      CON3(3,2)=D2*COS(Y(8))+D3*COS(Y(8)+Y(9))
      CON3(3,3)=D3*COS(Y(8)+Y(9))
      CON4(1,1)=-D2*COS(Y(11))*SIN(Y(10))-D3*SIN(Y(10))*COS(Y(11)+Y(12))
      CON4(1,2)=-D2*COS(Y(10))*SIN(Y(11))-D3*COS(Y(10))*SIN(Y(11)+Y(12))
      CON4(1,3)=-D3*COS(Y(10))*SIN(Y(11)+Y(12))
      CON4(2,1)=D2*COS(Y(11))*COS(Y(10))+D3*COS(Y(10))*COS(Y(11)+Y(12))
      CON4(2,2)=-D2*SIN(Y(11))*SIN(Y(10))-D3*SIN(Y(10))*SIN(Y(11)+Y(12))
      CON4(2,3)=-D3*SIN(Y(10))*SIN(Y(11)+Y(12))
      CON4(3,1)=0.0
      CON4(3,2)=D2*COS(Y(11))+D3*COS(Y(11)+Y(12))
      CON4(3,3)=D3*COS(Y(11)+Y(12))
      CON5(1,1)=-D2*COS(Y(14))*SIN(Y(13))-D3*SIN(Y(13))*COS(Y(14)+Y(15))
      CON5(1,2)=-D2*COS(Y(13))*SIN(Y(14))-D3*COS(Y(13))*SIN(Y(14)+Y(15))
```

```
CON5(1,3)=-D3*COS(Y(13))*SIN(Y(14)+Y(15))
CON5(2,1)=D2*COS(Y(14))*COS(Y(13))+D3*COS(Y(13))*COS(Y(14)+Y(15))
CON5(2,2)=-D2*SIN(Y(14))*SIN(Y(13))-D3*SIN(Y(13))*SIN(Y(14)+Y(15))
CON5(2,3)=-D3*SIN(Y(13))*SIN(Y(14)+Y(15))
CON5(3,1)=0.0
CON5(3,2)=D2*COS(Y(14))+D3*COS(Y(14)+Y(15))
CON5(3,3)=D3*COS(Y(14)+Y(15))
CON6(1,1)=-D2*COS(Y(17))*SIN(Y(16))-D3*SIN(Y(16))*COS(Y(17)+Y(18))
CON6(1,2)=-D2*COS(Y(16))*SIN(Y(17))-D3*COS(Y(16))*SIN(Y(17)+Y(18))
CON6(1,3)=-D3*COS(Y(16))*SIN(Y(17)+Y(18))
CON6(2,1)=D2*COS(Y(17))*COS(Y(16))+D3*COS(Y(16))*COS(Y(17)+Y(18))
CON6(2,2)=-D2*SIN(Y(17))*SIN(Y(16))-D3*SIN(Y(16))*SIN(Y(17)+Y(18))
CON6(2,3)=-D3*SIN(Y(16))*SIN(Y(17)+Y(18))
CON6(3,1)=0.0
CON6(3,2)=D2*COS(Y(17))+D3*COS(Y(17)+Y(18))
CON6(3,3)=D3*COS(Y(17)+Y(18))
CON7(1,1)=-D2*COS(Y(20))*SIN(Y(19))-D3*SIN(Y(19))*COS(Y(20)+Y(21))
CON7(1,2)=-D2*COS(Y(19))*SIN(Y(20))-D3*COS(Y(19))*SIN(Y(20)+Y(21))
CON7(1,3)=-D3*COS(Y(19))*SIN(Y(20)+Y(21))
CON7(2,1)=D2*COS(Y(20))*COS(Y(19))+D3*COS(Y(19))*COS(Y(20)+Y(21))
CON7(2,2)=-D2*SIN(Y(20))*SIN(Y(19))-D3*SIN(Y(19))*SIN(Y(20)+Y(21))
CON7(2,3)=-D3*SIN(Y(19))*SIN(Y(20)+Y(21))
CON7(3,1)=0.0
CON7(3,2)=D2*COS(Y(20))+D3*COS(Y(20)+Y(21))
CON7(3,3)=D3*COS(Y(20)+Y(21))
CON8(1,1)=-D2*COS(Y(23))*SIN(Y(21))-D3*SIN(Y(22))*COS(Y(23)+Y(24))
CON8(1,2)=-D2*COS(Y(22))*SIN(Y(23))-D3*COS(Y(22))*SIN(Y(23)+Y(24))
CON8(1,3)=-D3*COS(Y(22))*SIN(Y(23)+Y(24))
CON8(2,1)=D2*COS(Y(23))*COS(Y(22))+D3*COS(Y(22))*COS(Y(23)+Y(24))
CON8(2,2)=-D2*SIN(Y(23))*SIN(Y(22))-D3*SIN(Y(22))*SIN(Y(23)+Y(24))
CON8(2,3)=-D3*SIN(Y(22))*SIN(Y(23)+Y(24))
CON8(3,1)=0.0
CON8(3,2)=D2*COS(Y(23))+D3*COS(Y(23)+Y(24))
CON8(3,3)=D3*COS(Y(23)+Y(24))
CON9(1,1)=-D2*COS(Y(26))*SIN(Y(25))-D3*SIN(Y(25))*COS(Y(26)+Y(27))
CON9(1,2)=-D2*COS(Y(25))*SIN(Y(26))-D3*COS(Y(25))*SIN(Y(26)+Y(27))
```

```
CON9(1,3)=-D3*COS(Y(25))*SIN(Y(26)+Y(27))
CON9(2,1)=D2*COS(Y(26))*COS(Y(25))+D3*COS(Y(25))*COS(Y(26)+Y(27))
CON9(2,2)=-D2*SIN(Y(26))*SIN(Y(25))-D3*SIN(Y(25))*SIN(Y(26)+Y(27))
CON9(2,3)=-D3*SIN(Y(25))*SIN(Y(26)+Y(27))
CON9(3,1)=0.0
CON9(3,2)=D2*COS(Y(26))+D3*COS(Y(26)+Y(27))
CON9(3,3)=D3*COS(Y(26)+Y(27))
CON10(1,1)=-D2*COS(Y(29))*SIN(Y(28))-D3*SIN(Y(28))*COS(Y(29)+Y(30))
CON10(1,2)=-D2*COS(Y(28))*SIN(Y(29))-D3*COS(Y(28))*SIN(Y(29)+Y(30))
CON10(1,3)=-D3*COS(Y(28))*SIN(Y(29)+Y(30))
CON10(2,1)=D2*COS(Y(29))*COS(Y(28))+D3*COS(Y(28))*COS(Y(29)+Y(30))
CON10(2,2)=-D2*SIN(Y(29))*SIN(Y(28))-D3*SIN(Y(28))*SIN(Y(29)+Y(30))
CON10(2,3)=-D3*SIN(Y(28))*SIN(Y(29)+Y(30))
CON10(3,1)=0.0
CON10(3,2)=D2*COS(Y(29))+D3*COS(Y(29)+Y(30))
CON10(3,3)=D3*COS(Y(29)+Y(30))
CON11(1,1)=-D2*COS(Y(32))*SIN(Y(31))-D3*SIN(Y(31))*COS(Y(32)+Y(33))
CON11(1,2)=-D2*COS(Y(31))*SIN(Y(32))-D3*COS(Y(31))*SIN(Y(32)+Y(33))
CON11(1,3)=-D3*COS(Y(31))*SIN(Y(32)+Y(33))
CON11(2,1)=D2*COS(Y(32))*COS(Y(31))+D3*COS(Y(31))*COS(Y(32)+Y(33))
CON11(2,2)=-D2*SIN(Y(32))*SIN(Y(31))-D3*SIN(Y(31))*SIN(Y(32)+Y(33))
CON11(2,3)=-D3*SIN(Y(31))*SIN(Y(32)+Y(33))
CON11(3,1)=0.0
CON11(3,2)=D2*COS(Y(32))+D3*COS(Y(32)+Y(33))
CON11(3,3)=D3*COS(Y(32)+Y(33))
DO 10 I=1,33
DO 10 J=1,66
10 A(I,J)=0.0
DO 11 I=1,3
K=1
DO 12 J=1,3
A(I,K)=CONI(I,J)
12 K=K+2
11 CONTINUE
DO 13 I=1,3
K=2
```

```
      DO 14 J=1,3
      A(I,K)=-1.*CON1(I,J)
   14 K=K+2
   13 CONTINUE
      DO 15 I=1,3
      K=1
      DO 16 J=1,3
      A(I+3,K+6)=CON2(I,J)
   16 K=K+2
   15 CONTINUE
      DO 17 I=1,3
      K=2
      DO 18 J=1,3
      A(I+3,K+6)=-CON2(I,J)
   18 K=K+2
   17 CONTINUE
      DO 19 I=1,3
      K=1
      DO 20 J=1,3
      A(I+6,K+12)=CON3(I,J)
   20 K=K+2
   19 CONTINUE
      DO 21 I=1,3
      K=2
      DO 22 J=1,3
      A(I+6,K+12)=-CON3(I,J)
   22 K=K+2
   21 CONTINUE
      DO 23 I=1,3
      K=1
      DO 24 J=1,3
      A(I+9,K+18)=CON4(I,J)
   24 K=K+2
   23 CONTINUE
      DO 25 I=1,3
      K=2
```

```
      DO 26 J=1,3
      A(I+9,K+18)=-CON4(I,J)
      26 K=K+2
      25 CONTINUE
      DO 27 I=1,3
      K=1
      DO 28 J=1,3
      A(I+12,K+24)=CON5(I,J)
      28 K=K+2
      27 CONTINUE
      DO 29 I=1,3
      K=2
      DO 30 J=1,3
      A(I+12,K+24)=-CON5(I,J)
      30 K=K+2
      29 CONTINUE
      DO 31 I=1,3
      K=1
      DO 32 J=1,3
      A(I+15,K+30)=CON6(I,J)
      32 K=K+2
      31 CONTINUE
      DO 33 I=1,3
      K=2
      DO 34 J=1,3
      A(I+15,K+30)=-CON6(I,J)
      34 K=K+2
      33 CONTINUE
      DO 35 I=1,3
      K=1
      DO 36 J=1,3
      A(I+18,K+36)=CON7(I,J)
      36 K=K+2
      35 CONTINUE
      DO 37 I=1,3
      K=2
```

```
      DO 38 J=1,3
      A(I+18,K+36)=-CON7(I,J)
   38 K=K+2
   37 CONTINUE
      DO 39 I=1,3
      K=1
      DO 40 J=1,3
      A(I+21,K+42)=CON8(I,J)
   40 K=K+2
   39 CONTINUE
      DO 41 I=1,3
      K=2
      DO 42 J=1,3
      A(I+21,K+42)=-CON8(I,J)
   42 K=K+2
   41 CONTINUE
      DO 43 I=1,3
      K=1
      DO 44 J=1,3
      A(I+24,K+48)=CON9(I,J)
   44 K=K+2
   43 CONTINUE
      DO 45 I=1,3
      K=2
      DO 46 J=1,3
      A(I+24,K+48)=-CON9(I,J)
   46 K=K+2
   45 CONTINUE
      DO 47 I=1,3
      K=1
      DO 48 J=1,3
      A(I+27,K+54)=CON10(I,J)
   48 K=K+2
   47 CONTINUE
      DO 49 I=1,3
      K=2
```

```
      DO 50 J=1,3
      A(I+27,K+54)=-CON10(I,J)
   50 K=K+2
   49 CONTINUE
      DO 51 I=1,3
      K=1
      DO 52 J=1,3
      A(I+30,K+60)=CON11(I,J)
   52 K=K+2
   51 CONTINUE
      DO 53 I=1,3
      K=2
      DO 54 J=1,3
      A(I+30,K+60)=-CON11(I,J)
   54 K=K+2
   53 CONTINUE
      J=1
      DO 1000 I=1,6,2
      SUM=0.0
      DO 9000 K=1,3
 9000 SUM=SUM+CON1(K,J)
      DELGV(I)=-1.*SUM
      DELGV(I+1)=SUM
 1000 J=J+1
      J=1
      DO 1200 I=7,12,2
      SUM=0.0
      DO 1300 K=1,3
 1300 SUM=SUM+CON2(K,J)
      DELGV(I)=-1.*SUM
      DELGV(I+1)=SUM
 1200 J=J+1
      J=1
      DO 1400 I=13,18,2
      SUM=0.0
      DO 1500 K=1,3
```

```
1500 SUM=SUM+CON3(K,J)
DELGV(I)=-1.*SUM
DELGV(I+1)=SUM
1400 J=J+1
J=1
DO 1600 I=19,24,2
SUM=0.0
DO 1700 K=1,3
1700 SUM=SUM+CON4(K,J)
DELGV(I)=-1.*SUM
DELGV(I+1)=SUM
1600 J=J+1
J=1
DO 7000 I=25,30,2
SUM=0.0
DO 7100 K=1,3
7100 SUM=SUM+CON5(K,J)
DELGV(I)=-1.*SUM
DELGV(I+1)=SUM
7000 J=J+1
J=1
DO 7200 I=31,36,2
SUM=0.0
DO 7300 K=1,3
7300 SUM=SUM+CON6(K,J)
DELGV(I)=-1.*SUM
DELGV(I+1)=SUM
7200 J=J+1
J=1
DO 7400 I=37,42,2
SUM=0.0
DO 7500 K=1,3
7500 SUM=SUM+CON7(K,J)
DELGV(I)=-1.*SUM
DELGV(I+1)=SUM
7400 J=J+1
```

```
      J=1
      DO 7600 I=43,48,2
      SUM=0.0
      DO 7700 K=1,3
 7700 SUM=SUM+CON8(K,J)
      DELGV(I)=-1.*SUM
      DELGV(I+1)=SUM
 7600 J=J+1
      J=1
      DO 7800 I=49,54,2
      SUM=0.0
      DO 7900 K=1,3
 7900 SUM=SUM+CON9(K,J)
      DELGV(I)=-1.*SUM
      DELGV(I+1)=SUM
 7800 J=J+1
      J=1
      DO 8000 I=55,60,2
      SUM=0.0
      DO 8100 K=1,3
 8100 SUM=SUM+CON10(K,J)
      DELGV(I)=-1.*SUM
      DELGV(I+1)=SUM
 8000 J=J+1
      J=1
      DO 8200 I=61,66,2
      SUM=0.0
      DO 8300 K=1,3
 8300 SUM=SUM+CON11(K,J)
      DELGV(I)=-1.*SUM
      DELGV(I+1)=SUM
 8200 J=J+1
      RETURN
      END
```

```
C*************************************************************C
C PROGRAM *** QUARD ***                                       C
C QUADRATIC PROGRAM BY THE WOLFE METHOD                       C
C IT PERFORMS THE INVERSE CONSTRAINED KINEMATIC               C
C ANALYSES OF ROBOTS OF GENERAL ARCHITECTURE                  C
C*************************************************************C
      PARAMETER (M=3,N=6)
      DIMENSION A(M,N),B(M),C(N,N),P(N),T(M+N,2*M+3*N+1)
      DIMENSION COST(2*M+3*N+1),TT(2*M+3*N+1)
      DIMENSION DIFF(2*M+3*N+1),PRFIT(2*M+3*N+1),RATIO(M+N+4)
      DIMENSION IB(M+N),III(2*M+3*N),OPP(M+N),TITLE(20)
      DIMENSION Y(3),RV(3)
  100 FORMAT(1X,10F10.3)
  102 FORMAT(/18X,4HC(J),3X,9(1X,F8.3,1X)/(14X,10(F9.2,1X)))
  103 FORMAT(6X,4HC(I),3X,4HT(I),10(4X,I2,4X)/(17X,10(4X,I2,4X)))
  104 FORMAT(2X,F8.2,2X,F11.2,1X,9(1X,F8.2,1X)/(14X,10(1X,F8.2,1X)))
  105 FORMAT(//5X,4HZ(J),3X,F11.2,1X,9(F9.2,1X)/14X,10(F9.2,1X))
  106 FORMAT(//5X,3HC-Z,6X,10(F9.2,1X)/14X,10(F9.2,1X))
  107 FORMAT(//9X,26HTHE MINIMUM VALUE OF Z IS=, E16.8)
  108 FORMAT(//9X,37HTHE OPTIMUM POINTS ARE PRINTED BELOW )
  109 FORMAT(//9X,43HTHE REST OF THE VARIABLES ARE EQUAL TO ZERO)
  110 FORMAT(9X,I2,1X,4X,E16.8)
  111 FORMAT(1H1,4X,5HTABLE,3X,I3)
  112 FORMAT(13X,36HTHE OBJECTIVE FUNCTION IS UNBOUNDED.)
  113 FORMAT(9X,4HC-Z(, I2, 1X, 2H)=, E16.8)
  114 FORMAT(//3X,5HIB(I))
  115 FORMAT(//9X,25HTHE OPPORTUNITY COSTS ARE/)
  116 FORMAT(//9X,42HTHE REST OF THE OPPORTUNITY COSTS ARE ZERO)
  117 FORMAT(5X,8HRATIO(I))
  118 FORMAT(//4X,17HNUMBERS 1 THROUGH, I3, 32H ARE ORDINARY VARIABLES
     1 , NUMBERS,I3,8H THROUGH,I3,17H ARE LAGRANGIANS.)
  119 FORMAT(8X,3HIPR,7X,3HIPC,7X,3HKCK)
  120 FORMAT(7X,4HTEST,7X,5HPIVOT,6X,9HDIFF(IPC))
  121 FORMAT(8I10)
  122 FORMAT(/4X,7HNUMBERS,I3,8H THROUGH,I3,20H ARE SLACKS, NUMBERS,
     1 I3,8H THROUGH,I3,15H ARE GRADIENTS.)
```

```
300 FORMAT(4I10)
302 FORMAT(3X,I4,F20.6)
128 FORMAT(//,6X,4H  I ,8X,4HX(I)//)
ITMAX=1000
MTR=1
D2=1.016
D3=1.5113
TR=22./(7.*180.)
Y(1)=7.122*TR
Y(2)=56.6709*TR
Y(3)=287.877*TR
RV(1)=0.0
RV(2)=0.
RV(3)=-0.0125
CON11=-D2*GOS(Y(2))*SIN(Y(1))-D3*SIN(Y(1))*COS(Y(2)+Y(3))
CON12=-D2*COS(Y(1))*SIN(Y(2))-D3*COS(Y(1))*SIN(Y(2)+Y(3))
CON13=-D3*COS(Y(1))*SIN(Y(2)+Y(3))
CON21=D2*COS(Y(2))*COS(Y(1))+D3*COS(Y(1))*COS(Y(2)+Y(3))
CON22=-D2*SIN(Y(2))*SIN(Y(1))-D3*SIN(Y(1))*SIN(Y(2)+Y(3))
CON23=-D3*SIN(Y(1))*SIN(Y(2)+Y(3))
CON31=0.0
CON32=D2*COS(Y(2))+D3*COS(Y(2)+Y(3))
CON33=D3*COS(Y(2)+Y(3))
999 DO 623 I=1,M
DO 623 J=1,N
623 A(I,J)=0.0
A(1,1)=CON11
A(1,2)=-CON11
A(1,3)=CON12
A(1,4)=-CON12
A(1,5)=CON13
A(1,6)=-CON13
A(2,1)=CON21
A(2,2)=-CON21
A(2,3)=CON22
A(2,4)=-CON22
```

```
A(2,5)=CON23
A(2,6)=-CON23
A(3,1)=CON31
A(3,2)=-CON31
A(3,3)=CON32
A(3,4)=-CON32
A(3,5)=CON33
A(3,6)=-CON33
DO 624 J=1,M
624 B(J)=0.0
B(1)=RV(1)
B(2)=RV(2)
B(3)=RV(3)
DO 625 I=1,N
DO 625 J=1,N
625 C(I,J)=0.0
C(1,1)=CON11**2+CON21**2+CON31**2
C(2,2)=CON11**2+CON21**2+CON31**2
C(3,3)=CON12**2+CON22**2+CON32**2
C(4,4)=CON12**2+CON22**2+CON32**2
C(5,5)=CON13**2+CON23**2+CON33**2
C(6,6)=CON13**2+CON23**2+CON33**2
C(2,1)=-(CON11**2+CON21**2+CON31**2)
C(1,2)=-(CON11**2+CON21**2+CON31**2)
C(1,3)=CON11*CON12+CON21*CON22+CON31*CON32
C(3,1)=CON11*CON12+CON21*CON22+CON31*CON32
C(1,4)=-(CON11*CON12+CON21*CON22+CON31*CON32)
C(4,1)=-(CON11*CON12+CON21*CON22+CON31*CON32)
C(1,5)=CON13*CON11+CON21*CON23+CON31*CON33
C(5,1)=CON13*CON11+CON21*CON23+CON31*CON33
C(1,6)=-(CON13*CON11+CON21*CON23+CON31*CON33)
C(6,1)=-(CON13*CON11+CON21*CON23+CON31*CON33)
C(2,3)=-(CON11*CON12+CON21*CON22+CON31*CON32)
C(2,4)=CON11*CON12+CON21*CON22+CON31*CON32
C(4,2)=CON11*CON12+CON21*CON22+CON31*CON32
C(2,5)=-(CON13*CON11+CON21*CON23+CON31*CON33)
```

```
C(2,6)=CON13*CON11+CON21*CON23+CON31*CON33
C(5,2)=-(CON13*CON11+CON21*CON23+CON31*CON33)
C(6,2)=CON13*CON11+CON21*CON23+CON31*CON33
C(3,2)=-(CON11*CON12+CON21*CON22+CON31*CON32)
C(3,4)=-(CON12**2+CON22**2+CON32**2)
C(3,5)=CON13*CON12+CON22*CON23+CON32*CON33
C(3,6)=-(CON13*CON12+CON22*CON23+CON32*CON33)
C(4,3)=-(CON12**2+CON22**2+CON32**2)
C(5,3)=CON13*CON12+CON22*CON23+CON32*CON33
C(6,3)=-(CON13*CON12+CON22*CON23+CON32*CON33)
C(4,5)=-(CON13*CON12+CON22*CON23+CON32*CON33)
C(5,4)=-(CON13*CON12+CON22*CON23+CON32*CON33)
C(4,6)=CON13*CON12+CON22*CON23+CON32*CON33
C(6,4)=CON13*CON12+CON22*CON23+CON32*CON33
C(5,6)=-(CON13**2+CON23**2+CON33**2)
C(6,5)=-(CON13**2+CON23**2+CON33**2)
DO 626 I=1,N
626 P(I)=0.0
P(1)=-2.*(RV(1)*CON11+RV(2)*CON21+RV(3)*CON31)
P(3)=-2.*(RV(1)*CON12+RV(2)*CON22+RV(3)*CON32)
P(5)=-2.*(RV(1)*CON13+RV(2)*CON23+RV(3)*CON33)
P(2)=2.*(RV(1)*CON11+RV(2)*CON21+RV(3)*CON31)
P(4)=2.*(RV(1)*CON12+RV(2)*CON22+RV(3)*CON32)
P(6)=2.*(RV(1)*CON13+RV(2)*CON23+RV(3)*CON33)
MP1=M+1
MM1=M-1
NP1=N+1
NP2=N+2
MN=M+N
MNM1=MN-1
MNP1=MN+1
MNP2=MN+2
NV=MN+N
NVP1=NV+1
NVP2=NV+2
NY=NV+M
```

```
      NYP1=NY+1
      NYP2=NY+2
      NZ=NY+N
      NC=NZ+1
      NZP2=NZ+2
      DO 180 I=1,MN
      DO 180 J=1,NC
  180 T(I,J)=0.0
      DO 182 I=1,M
  182 T(I,1)=B(I)
      DO 183 I=MP1,MN
      J=I-M
  183 T(I,1)=-P(J)
      DO 184 I=1,M
      DO 184 J=1,N
      JP1=J+1
  184 T(I,JP1)=A(I,J)
      DO 185 I=1,N
      DO 185 J=1,N
      IPM=I+M
      JP1=J+1
  185 T(IPM,JP1)=2.*C(I,J)
      DO 186 I=MP1,MN
      IMM=I-M
      DO 186 J=NP2,MNP1
      JMN=J-N-1
  186 T(I,J)=A(JMN,IMM)
      DO 187 I=1,MN
      IJ=I+NYP1
      DO 187 J=NYP2,NC
      IF(J-IJ)187,179,187
  179 T(I,J)=1.
  187 CONTINUE
      DO 188 I=MP1,MN
      IJ=I-M+MNP1
      DO 188 J=MNP2,NC
```

```
      IF(J-IJ)188,178,188
  178 T(I,J)=-1.
  188 CONTINUE
      DO 208 I=1,MN
      OPP(I)=T(I,1)
  208 CONTINUE
      DO 340 J=1,NZ
  340 COST(J)=0.0
      DO 189 I=1,M
      J=NP1+I
  189 COST(J)=T(I,1)
      DO 190 J=NYP2,NC
  190 COST(J)=9999.
      NN=NZ-MN
      DO 25 KK=1,NZ
   25 III(KK)=KK
      DO 1 I=1,MN
    1 IB(I)=NN+I
      K=0
C ITERATON START
   19 K=K+1
      DO 2 J=1,NC
    2 PRFIT(J)=0.
      DO 3 J=1,NC
      SUM=0.
      DO 4 I=1,MN
      JJ=IB(I)+1
    4 SUM=SUM+COST(JJ)*T(I,J)
      PRFIT(J)=SUM
    3 DIFF(J)=COST(J)-PRFIT(J)
      IF(MTR)555,666,555
  555 WRITE(6,111)K
c555 WRITE(6,*)K
C PRINT TABLE IF DESIRED
      WRITE(6,102)(COST(J),J=2,NC)
      WRITE(6,103)(III(KK),KK=1,NZ)
```

```
WRITE(6,*)(COST(J),J=2,NC)
WRITE(6,*)(III(KK),KK=1,NZ)
DO 26 I=1,MN
JJ=IB(I)+1
26 WRITE(6,104)COST(JJ),(T(I,J),J=1,NC)
WRITE(6,105)(PRFIT(J),J=1,NC)
WRITE(6,106)(DIFF(J),J=2,NC)
print*,'*****************************************'
C FIND THE PIVOT ELEMENT ----T(IPR,IPC)
666 IPC=0
TEST=0.
C FIND THE VARAIBLE WITH THE LARGEST PROFIT
DO 5 I=2,NC
235 IF(DIFF(I)-TEST)6,5,5
6 TEST=DIFF(I)
IPC=I
5 CONTINUE
IF(IPC)99,99,7
7 KCK=0
DO 8 I=1,MN
IF(T(I,IPC))32,32,20
20 RATIO(I)=T(I,1)/T(I,IPC)
GO TO 8
32 KCK=KCK+1
IF(KCK-MN)21,31,21
21 RATIO(I)=1.E20
8 CONTINUE
C REMOVE LIMITING VARIABLE
DO 9 I=1,MN
IF(RATIO(I))9,10,10
10 IF(RATIO(I).GT.10000.)RATIO(I)=10000.
TEST=RATIO(I)
IPR=I
GO TO 11
9 CONTINUE
11 DO 12 I=1,MN
```

```
      IF(TEST-RATIO(I))12,12,13
   13 TEST=RATIO(I)
      IPR=I
   12 CONTINUE
C START PIVOTING AND INTRODUCING NEW VARIABLE INTO SOLUTION
      PIVOT=T(IPR,IPC)
      DO 15 J=1,NC
   15 T(IPR,J)=T(IPR,J)/PIVOT
      DO 171 I=1,MN
      IF(I-IPR)17,171,17
   17 DO 18 J=1,NC
   18 TT(J)=T(IPR,J)*T(I,IPC)/T(IPR,IPC)
      DO 172 J=1,NC
  172 T(I,J)=T(I,J)-TT(J)
  171 CONTINUE
      COST(IPR)=COST(IPC)
      IB(IPR)=IPC-1
C TRACE OUTPUT IF DESIRED
      IF(MTR-1)205,205,86
   86 WRITE(6,114)
      print*,'check for '
      DO 87 I=1,MN
   87 WRITE(6,300)I,IB(I)
      WRITE(6,119)
      WRITE(6,121)IPR,IPC,KCK
      WRITE(6,120)
      WRITE(6,100)TEST,PIVOT,DIFF(IPC)
      WRITE(6,117)
      DO 88 I=1,MN
   88 WRITE(6,302)I,RATIO(I)
C RECOMPUTE COSTS
  205 DO 176 J=1,NYP1
  176 COST(J)=0.
      DO 197 I=1,MN
      IF(IB(I)-MN)192,192,195
  192 JJ=IB(I)+MNP1
```

```
      GO TO 198
  195 IF(IB(I)-NY)196,196,197
  196 JJ=IB(I)-MNM1
      GO TO 198
  198 COST(JJ)=T(I,1)
  197 CONTINUE
      IF(K-ITMAX)19,99,99
   99 SUM=0.
      DO 201 I=1,MN
      IN=IB(I)
      IF(IN.GT.N)GO TO 201
      SUM=SUM+P(IN)*T(I,1)
  201 CONTINUE
      FRST=SUM
      SUM=0.
      DO 202 I=1,MN
      DO 202 J=1,MN
      IN=IB(I)
      IF(IN.GT.N)GO TO 202
      JN=IB(J)
      IF(JN.GT.N)GO TO 202
      SUM=SUM+C(IN,JN)*T(I,1)*T(J,1)
  202 CONTINUE
      SCND=SUM
      OBJ=FRST+SCND
      WRITE(6,107)OBJ
      WRITE(6,118)N,NP1,MN
      WRITE(6,122)MNP1,NY,NYP1,NZ
      WRITE(6,108)
      WRITE(6,128)
      DO 28 I=1,MN
      IF(T(I,1))27,28,27
   27 WRITE(6,110)IB(I),T(I,1)
   28 CONTINUE
      WRITE(6,109)
      WRITE(6,115)
```

```
      DO 53 I=1,MN
      IF(OPP(I))52,53,52
   52 WRITE(6,113)I,OPP(I)
   53 CONTINUE
      WRITE(6,116)
   31 WRITE(6,*)
      GO TO 30
  830 PRINT*,'ITERATION LIMIT EXCEEDED'
   30 STOP
      END
```

```
C*****************************************************************C
C PROGRAM *** FORCEAN ***                                         C
C THIS PROGRAM CALCULATES THE FORCE VECTOR FOR                    C
C STUDYING THE FORCE RESPONSE OF T3R3 ROBOT                       C
C*****************************************************************C
      REAL FI(42),Y(3),ANGACC(3),ANGV(3),AAL(3),AANGV(3)
REAL AL(3),FIM(84),F(66),AANGACC(3),YY(3)
REAL FFT(66,10),FR(66,10)
OPEN(UNIT=26,FILE='DIVEAC.DAT',TYPE='NEW')
DO 1000 K=1,10
CALL PARAMETER(AAL,YY,AANGV,AANGACC)
DO 1 IL=2,4
INDEX=1
CALL CONDITION(IL,NG,RNG,NJAP)
DO 10 IG=1,NG
CALL DATA(IG,RNG,IL,AAL,YY,AANGV,AANGACC,AL,Y,ANGV,ANGACC)
D2=AL(2)
D3=AL(3)
ACON11=-D2*COS(Y(2))*SIN(Y(1))-D3*SIN(Y(1))*COS(Y(2)+Y(3))
ACON12=-D2*COS(Y(1))*SIN(Y(2))-D3*COS(Y(1))*SIN(Y(2)+Y(3))
ACON13=-D3*COS(Y(1))*SIN(Y(2)+Y(3))
ACON21=D2*COS(Y(2))*COS(Y(1))+D3*COS(Y(1))*COS(Y(2)+Y(3))
ACON22=-D2*SIN(Y(2))*SIN(Y(1))-D3*SIN(Y(1))*SIN(Y(2)+Y(3))
ACON23=-D3*SIN(Y(1))*SIN(Y(2)+Y(3))
ACON31=0.0
ACON32=D2*COS(Y(2))+D3*COS(Y(2)+Y(3))
ACON33=D3*COS(Y(2)+Y(3))
T11=-(D2*COS(Y(2))*COS(Y(1))+D3*COS(Y(1))*COS(Y(2)+Y(3)))
     1 *(ANGV(1)**2)
T12=-(D2*COS(Y(1))*COS(Y(2))+D3*COS(Y(1))*COS(Y(2)+Y(3)))
     1 *(ANGV(2)**2)
T13=-(D3*COS(Y(1))*COS(Y(2)+Y(3)))*(ANGV(3)**2)
T14=(D2*SIN(Y(1))*SIN(Y(2))+D3*SIN(Y(1))*SIN(Y(2)+Y(3)))
     1 *2.*ANGV(1)*ANGV(2)
T15=-(D3*COS(Y(1))*COS(Y(2)+Y(3)))*2.*ANGV(2)*ANGV(3)
T16=(D3*SIN(Y(1))*SIN(Y(2)+Y(3)))*2.*ANGV(3)*ANGV(1)
```

```
T21=(-D2*COS(Y(2))*SIN(Y(1))-D3*SIN(Y(1))*COS(Y(2)+Y(3)))
     1 *(ANGV(1)**2)
T22=(-D2*COS(Y(2))*SIN(Y(1))-D3*SIN(Y(1))*COS(Y(2)+Y(3)))
     1 *(ANGV(2)**2)
T23=(-D3*SIN(Y(1))*COS(Y(2)+Y(3)))*(ANGV(3)**2)
T24=(-D2*SIN(Y(2))*COS(Y(1))-D3*COS(Y(1))*SIN(Y(2)+Y(3)))
     1 *2.*ANGV(1)*ANGV(2)
T25=(-D3*SIN(Y(1))*COS(Y(2)+Y(3)))*ANGV(2)*ANGV(3)*2.
T26=(-D3*COS(Y(1))*SIN(Y(2)+Y(3)))*2.*ANGV(3)*ANGV(1)
T31=0.0
T32=-(D2*SIN(Y(2))+D3*SIN(Y(2)+Y(3)))*ANGV(2)**2
T33=(-D3*SIN(Y(2)+Y(3)))*(ANGV(3)**2)
T34=2.*(-D3*SIN(Y(2)+Y(3)))*ANGV(3)*ANGV(2)
INDEX=INDEX+INC
FI(INDEX)=-(T11+T12+T13+T14+T15+T16-ACON11*ANGACC(1)
     1 -ACON12*ANGACC(2)-ACON13*ANGACC(3))-9.81
FI(INDEX+1)=-(T21+T22+T23+T24+T25+T26-ACON21*ANGACC(1)
     1 -ACON22*ANGACC(2)-ACON23*ANGACC(3))-9.81
FI(INDEX+2)=-(T32+T33+T34-ACON31*ANGACC(1)-ACON32*ANGACC(2)
     1 -ACON33*ANGACC(3))-9.81
CALL UFIM(FI,IL,FIM)
INC=6
10 CONTINUE
1 CONTINUE
DO 3 I=25,30
3 FIM(I)=FIM(I)+FIM(I+6)+FIM(I+48)
DO 5 I=31,36
5 FIM(I)=FIM(I+48)
DO 40 I=7,72
40 F(I-6)=FIM(I)
DO 61 I=61,63
61 F(I)=F(I)*30.429/5.429
DO 41 I=1,66
41 FR(I,K)=F(I)
1000 CONTINUE
JJ=0
```

```
      DO 44 KI=1,66
      DO 43 I=1,10
      J=JJ+1
      FFT(J,I)=FR(J,I)
      WRITE(26,*)FFT(J,I)
      43 CONTINUE
      JJ=JJ+1
      44 CONTINUE
      STOP
      END
      SUBROUTINE DATA(IG,RNG,IL,AAL,YY,AANGV,AANGACC,AL,
     1  Y,ANGV,ANGACC)
      DIMENSION AL(3),Y(3),ANGV(3),ANGACC(3)
      DIMENSION AAL(3),YY(3),AANGV(3),AANGACC(3)
      REAL RIG,RNG
      RIG=IG
      DO 1 I=1,3
      AL(I)=AAL(I)
      Y(I)=YY(I)
      ANGV(I)=AANGV(I)
      1 ANGACC(I)=AANGACC(I)
      IF(IL.EQ.2)THEN
      AL(2)=(AAL(2)/(RNG-1.))*(RIG-1.)
      AL(3)=0.0
      Y(3)=0.0
      ANGV(3)=0.0
      ANGACC(3)=0.0
      ELSE
      IF(IL.EQ.3)THEN
      AL(2)=AAL(2)
      AL(3)=(AAL(3)/(RNG-1.))*(RIG-1.)
      ELSE
      IF(RIG.EQ.1.)THEN
      AL(3)=.15*RIG
      ELSE
      AL(3)=0.0
```

```
      ENDIF
      ENDIF
      ENDIF
C     PRINT*,'AL',AL(1),AL(2),AL(3)
      RETURN
      END
      SUBROUTINE CONDITION(IL,NG,RNG,NJAP)
      IF(IL.EQ.2)THEN
      NG=5
      RNG=5.
      NJAP=30
      ELSE
      IF(IL.EQ.3)THEN
      NG=7
      RNG=7.
      NJAP=42
      ELSE
      NG=2
      RNG=2.
      NJAP=12
      ENDIF
      ENDIF
      RETURN
      END
      SUBROUTINE PARAMETER(AL,Y,ANGV,ANGACC)
      DIMENSION AL(3),Y(3),YY(100),ANGV(3),ANGACC(3)
      OPEN(UNIT=7,FILE='DATA.DAT',TYPE='OLD')
      READ(7,*)(YY(I),I=1,9)
      TR=22./(180.*7.)
      AL(1)=0.0
      AL(2)=1.016
      AL(3)=1.5113
      DO 2 I=1,3
    2 Y(I)=YY(I)*TR
      DO 3 I=4,6
    3 ANGV(I-3)=YY(I)
```

```
      DO 4 I=7,9
    4 ANGACC(I-6)=YY(I)
      RETURN
      END
      SUBROUTINE UFIM(FI,IL,FIM)
      DIMENSION FI(42),FIM(84)
      IF(IL.EQ.2)THEN
      AM=20.1289
      DO 1 I=1,30
    1 FIM(I)=FI(I)*am
      ELSE
      IF(IL.EQ.3)THEN
      AM=5.429
      DO 2 I=31,72
    2 FIM(I)=FI(I-30)*am
      ELSE
      AM=30.429
      DO 3 I=73,84
    3 FIM(I)=FI(I-72)*am
      ENDIF
      ENDIF
      RETURN
      END
```

```
C*****************************************************************C
C PROGRAM ###NEWCON###                                            C
C NEWCON PERFORMS THE FORCE RESPONSE ANALYSIS OF T3R3             C
C ROBOT USING NEWMARK-BETA INTEGRATION SCHEME                     C
C GMM IS THE GLOBAL MASS MATRIX AND GKK IS GLOBAL                 C
C STIFFNESS MATRIX FORMED BY PROGRAM FRECON                       C
C*****************************************************************C
DIMENSION GMM(100,100),GKK(100,100),X(66,31),XD(66,31),
     1 XDD(66,31),F(66),GMKINV(66,66),TMK(66,66),CON1(66,31),
     1 CON2(66),TMKINV(66,66),WKAREA(10000),GMMCON(66),SUMF(66),
     1 TX(66,1),Y(660)
REAL RWKSP(19771)
COMMON/WORKSP/RWKSP
OPEN(UNIT=14,FILE='MASS.DAT',TYPE='OLD')
OPEN(UNIT=15,FILE='STIFF.DAT',TYPE='OLD')
READ(14,*)((GMM(I,J),J=1,66),I=1,66)
READ(15,*)((GKK(I,J),J=1,66),I=1,66)
CALL IWKIN(19771)
ALFA=1./4.
BETA=1./2.
TIME=.0299/2.
K=1
DO 1 I=1,66
X(I,K)=0.0
XD(I,K)=0.0
XDD(I,K)=0.0
1 CONTINUE
C DELT1=TIME STEP FOR NEWMARK SCHEME
delt1=delt1+time
DELT=DELT+TIME
DO 1000 K=2,24
C LSQRC IS A LEAST SQUARE CURVE FITTING PROGRAM
CALL LSQRC(DELT1,F)
C WRITE(31,*)DELT1,F(61)
C WRITE(32,*)DELT1,F(62)
C WRITE(33,*)DELT1,F(63)
```

```
      DO 2 I=1,66
      DO 2 J=1,66
    2 TMK(I,J)=(1./(ALFA*DELT**2))*GMM(I,J)+GKK(I,J)
      IDGT=4
      DO 3 I=1,66
      CON1(I,K)=(1./(ALFA*DELT**2))*X(I,K-1)+(1./(ALFA*DELT))*
     1 XD(I,K-1)+((1./(2.*ALFA))-1.)*XDD(I,K-1)
    3 CON2(I)=CON1(I,K)
      CALL MRRRR(66,66,GMM,66,66,1,CON2,66,66,1,GMMCON,66)
      DO 4 II=1,66
    4 SUMF(II)=GMMCON(II)+F(II)
      NJ=66
C     SKYLINE SOLVES THE SYSTEM OF EQUATION AX=B
      CALL SKYLINE(TMK,NJ,SUMF,TX)
      DO 5 I=1,66
    5 X(I,K)=TX(I,1)
      DO 6 I=1,66
      XDD(I,K)=(1./(ALFA*DELT**2))*(X(I,K)-X(I,K-1))-
     1 (1./(ALFA*DELT))*XD(I,K-1)-((1./(2.*ALFA))-1.)*XDD(I,K-1)
      XD(I,K)=XD(I,K-1)+(1.-BETA)*DELT*XDD(I,K-1)+BETA*DELT*
     1 XDD(I,K)
    6 CONTINUE
C     WRITE(21,*)delt1,X(61,K)
C     WRITE(22,*)delt1,X(62,K)
C     WRITE(23,*)delt1,X(63,K)
      delt1=delt1+time
 1000 CONTINUE
      STOP
      END
C#####################################################################C
C SUBROUTINE LSQRC IS CURVE FITTING PROGRAM BASED ON                  C
C LEAST SQUARE METHOD                                                C
C IT READS THE DATA FROM DIVEAC.DAT                                  C
C #####################################################################C
      SUBROUTINE LSQRC(TIME,F)
      DIMENSION X(10),Y(660),YFIT(660),A(3),F(66)
```

```
      DATA M,NI,NF/10,2,3/
      OPEN(UNIT=11,FILE='DIVEAC.DAT',TYPE='OLD')
      REWIND (11)
      C CALL FOR TIME STEPS OF DIVEAC.DAT
      CALL XXX(X)
      DO 74 KKK=1,66
      READ(11,*)(Y(I),I=1,M)
      DO 80 N=NI,NF
      NP=N+1
      CALL LSFIT(X,Y,A,M,N,NP)
      SUMSQ=0.
      DO 20 K=1,M
      YFIT(K)=A(1)
      TEMP=X(K)
      DO 10 I=2,N
      YFIT(K)=YFIT(K)+A(I)*TEMP
            TEMP=TEMP*X(K)
       10   CONTINUE
      SUMSQ=SUMSQ+(Y(K)-YFIT(K))**2
       20   CONTINUE
       80   CONTINUE
      PX=TIME
      C CALCULATES THE INTERMEDIATE POINTS
      CALL XINTER(A,PX,PY)
      F(KKK)=PY
      74 CONTINUE
      RETURN
      END
            SUBROUTINE LSFIT(X,Y,A,M,N,NP)
      DIMENSION X(M),Y(M),A(N),COEFF(12,13),SUmX(18)
      NM2=2*(N-1)
      DO 10 L=1,NM2
      SUMX(L)=0.
       10 CONTINUE
      DO 20 I=1,N
      COEFF(I,NP)=0.
```

```
 20     CONTINUE
DO 50 K=1,M
TEMP=X(K)
DO 30 L=1,NM2
SUMX(L)=SUMX(L)+TEMP
TEMP=TEMP*X(K)
 30  CONTINUE
TEMP=Y(K)
DO 40 I=1,N
COEFF(I,NP)=COEFF(I,NP)+TEMP
TEMP=TEMP*X(K)
 40     CONTINUE
 50 CONTINUE
DO 80 I=1,N
DO 70 J=1,N
IF(I.GT.1.OR.J.GT.1)GOTO 60
COEFF(I,J)=M
GOTO 70
 60 L=I+J-2
COEFF(I,J)=SUMX(L)
 70 CONTINUE
 80 CONTINUE
CALL GAEL(COEFF,A,N,NP)
RETURN
END
       SUBROUTINE GAEL(A,X,N,NP)
DIMENSION A(12,13),X(N)
L=N-1
DO 60 K=1,L
KP1=K+1
JJ=K
BIG=ABS(A(K,K))
DO 10 I=KP1,N
AB=ABS(A(I,K))
IF(AB.LE.BIG)GOTO 10
BIG=AB
```

```
      JJ=K
       10 CONTINUE
      IF(JJ.EQ.K)GOTO 30
      DO 20 J=K,NP
      TEMP=A(JJ,J)
      A(JJ,J)=A(K,J)
      A(K,J)=TEMP
       20    CONTINUE
       30    DO 50 I=KP1,N
      QUOT=A(I,K)/A(K,K)
      DO 40 J=KP1,NP
      A(I,J)=A(I,J)-QUOT*A(K,J)
       40 CONTINUE
       50  CONTINUE
       60 CONTINUE
      X(N)=A(N,NP)/A(N,N)
      DO 80 NN=1,L
      SUM=0.
      I=N-NN
      IP1=I+1
      DO 70 J=IP1,N
      SUM=SUM+A(I,J)*X(J)
      70  CONTINUE
      X(I)=(A(I,NP)-SUM)/A(I,I)
      80 CONTINUE
      RETURN
      END
      SUBROUTINE XINTER(A,PX,PY)
      DIMENSION A(3)
      PY=A(1)+A(2)*PX+A(3)*PX**2
      RETURN
      END.
      SUBROUTINE XXX(X)
      DIMENSION X(10)
      DELT=.03937
         X(1)=0.0
```

```
X(2)=DELT
   X(3)=DELT+X(2)
   X(4)=DELT+X(3)
X(5)=DELT+X(4)
   X(6)=DELT+X(5)
   X(7)=DELT+X(6)
   X(8)=DELT+X(7)
   X(9)=DELT+X(8)
   X(10)=DELT+X(9)
RETURN
END
```

```
C*****************************************************************C
C PROGRAM FRECON TO CALCULATE THE NATURAL FREQUENCY OF      C
C ROBOTIC MANIPULATOR                                       C
C       USING FINITE ELEMENT TECHNIQUE                          C
C THE GIVEN ROBOT IS DIVIDED INTO 11 BEAM ELEMENTS          C
C AT EACH NODE THERE IS 6 DEGREE OF FREEDOM                 C
C ROBOT HAS 66 DEGREES OF FREEDOM                           C
C*****************************************************************C
DIMENSION A1(3),B1(3),C1(3),D1(3),GM(100,100),GK(100,100),
     1  GM1(100,100),GK1(100,100),GM2(100,100),GK2(100,100),P(3),
     1  GM3(100,100),GK3(100,100),GM23(100,100),GK23(100,100)
COMMON/BLK1/OMEGA(100)
OPEN (UNIT=16,FILE='OUTPUT.DAT',TYPE='NEW')
OPEN (UNIT=17,FILE='STIFF.DAT',TYPE='NEW')
OPEN (UNIT=18,FILE='MASS.DAT',TYPE='NEW')
P(1)=0.
P(2)=0.
P(3)=10.
A1(1)=0.
A1(2)=0.
A1(3)=0.
B1(1)=.53571
B1(2)=0.0384
B1(3)=.8626
C1(1)=.3903
C1(2)=.024
C1(3)=.8986
D1(1)=2.
D1(2)=.25/2.
D1(3)=.5
NO1=4
NO2=1
NO3=6
CALL LINK(A1,B1,NO1,2.E11,.01016,7800.,6.14591E-5,3.07295E-5,
     1  3.07295E-5,GM1,GK1,P)
CALL LINK(C1,B1,NO2,6.894E10,0.00798,2700.,3.62E-5,1.81E-5,
```

```
     1  1.81E-5,GM2,GK2,P)
CALL LINK(B1,D1,NO3,6.894E10,0.00798,2700.,3.62E-5,1.81E-5,
     1  1.81E-5,GM3,GK3,P)
CALL GLOBAL(NO2,GM2,GK2,NO3,GM3,GK3,GM23,GK23)
NO23=(NO2+NO3)*6+6
CALL ASSMBLY(NO1,GM1,GK1,NO23,GM23,GK23,GM,GK)
N=(NO1+NO2+NO3)*6+6
CALL FREQUENCY(GM,GK,N)
WRITE(16,*)' THE NATURAL FREQUENCIES OF THE ROBOTIC MANIPULATOR
ARE'
DO 14 I=1,N-6
WRITE(16,*)'SNO(I)=',I,'FREQUENCY=',OMEGA(I)
14 CONTINUE
STOP
END
C*****************************************************************C
C SUBROUTINE LINK CALCULATES THE  LINK STIFFNESS AND       C
C MASS MATRIX IN GLOBAL DIRECTION                          C
C*****************************************************************C
SUBROUTINE LINK(A1,B1,N,E,A,DEN,AIX,AIY,AIZ,GM,GK,P)
DIMENSION AK(12,12),AM(12,12),T(12,12),A1(3),B1(3),
     1  TEMP1(12,12),GK(100,100),GM(100,100),EK(12,12),EM(12,12),
     1  AMM(25,12,12),AKK(25,12,12),P(3)
DO 5 I=1,100
DO 5 J=1,100
GM(I,J)=0.
5 GK(I,J)=0.
ANEW=0.3
G=E/(2.*(1.+ANEW))
TTLEN=SQRT((B1(1)-A1(1))**2+(B1(2)-A1(2))**2+(B1(3)-
     1  A1(3))**2)
ALEN=TTLEN/N
CALL STIFF(ALEN,E,G,A,AIX,AIY,AIZ,AK)
CALL MASS(ALEN,A,DEN,AIX,AM)
C CALL TRANSFORM(A1,B1,T)
ICODE=ICODE+1
```

```
      IF(ICODE.EQ.1) THEN
      CALL ALL1(A1,B1,T,P)
      ELSE
      CALL ALL(A1,B1,T,P)
      ENDIF
      CALL VMULFM(T,AK,12,12,12,12,12,TEMP1,12,IER)
      CALL RGMPRD(TEMP1,T,EK,12,12,12)
      CALL VMULFM(T,AM,12,12,12,12,12,TEMP1,12,IER)
      CALL RGMPRD(TEMP1,T,EM,12,12,12)
      DO 10 I=1,N
      DO 10 J=1,12
      DO 10 K=1,12
      AMM(I,J,K)=EM(J,K)
   10 AKK(I,J,K)=EK(J,K)
      I3=0
      DO 20 I=1,N*6-5,6
      I3=I3+1
      I1=I-1
      DO 20 I2=1,12
      DO 20 J2=1,12
      GM(I1+I2,I1+J2)=GM(I1+I2,I1+J2)+AMM(I3,I2,J2)
        GK(I1+I2,I1+J2)=GK(I1+I2,I1+J2)+AKK(I3,I2,J2)
   20 CONTINUE
      RETURN
      END
      SUBROUTINE ALL(B1,D1,T,P)
      DIMENSION B1(3),D1(3),TR2(4,4),P(3),XP(4,1),XP0(4,1)
      DIMENSION T(12,12),TR3(4,4)
      CALL DCS(B1,D1,TR2,P)
      CALL LOCALP(B1,D1,XP)
      CALL CHECK(TR2,TR3,XP,XP0,P)
      CALL GLOBTRA(TR3,T)
      RETURN
      END
      SUBROUTINE ALL1(B1,D1,T,P)
      DIMENSION B1(3),D1(3),TR2(4,4),P(3),XP(4,1),XP0(4,1)
```

```
      DIMENSION T(12,12),TR3(4,4)
      CALL DCS(B1,D1,TR2,P)
      CALL LOCALP1(B1,D1,XP)
      CALL CHECK(TR2,TR3,XP,XP0,P)
      CALL GLOBTRA(TR3,T)
      RETURN
      END
      SUBROUTINE LOCALP1(O,A,X)
      DIMENSION O(3),A(3),X(4,1)
      TR=22./(180.*7.)
      THETA1=ATAN(A(3)/(SQRT(A(1)**2+A(2)**2)))
      THETA2=90.*TR-THETA1
      AR3=1.016/COS(THETA2)
      X(1,1)=(10.*1.016/AR3)-1.016
      X(2,1)=10.*SIN(THETA2)
      X(3,1)=0.0
      X(4,1)=1.
      RETURN
      END
      SUBROUTINE DCS(A1,B1,TR1,P)
      DIMENSION A1(3),B1(3),TR1(4,4),P(3)
      CA11=B1(1)-A1(1)
      CA12=B1(2)-A1(2)
      CA13=B1(3)-A1(3)
      CA21=P(1)-A1(1)
      CA22=P(2)-A1(2)
      CA23=P(3)-A1(3)
      COX=CA12*CA23-CA13*CA22
      COY=CA13*CA21-CA11*CA23
      COZ=CA11*CA22-CA12*CA21
      XYZ=SQRT(COX**2+COY**2+COZ**2)
      AB=SQRT((B1(1)-A1(1))**2+(B1(2)-A1(2))**2+(B1(3)-A1(3))**2)
      AL1=(B1(1)-A1(1))/AB
      AM1=(B1(2)-A1(2))/AB
      AN1=(B1(3)-A1(3))/AB
      AL3=COX/XYZ
```

```
AM3=COY/XYZ
AN3=COZ/XYZ
AL2=AM3*AN1-AN3*AM1
AM2=AN3*AL1-AL3*AN1
AN2=AL3*AM1-AM3*AL1
tr1(1,4)=B1(1)
tr1(2,4)=B1(2)
tr1(3,4)=B1(3)
tr1(4,4)=1.
TR1(1,1)=AL1
TR1(1,2)=AL2
TR1(1,3)=AL3
TR1(2,1)=AM1
TR1(2,2)=AM2
TR1(2,3)=AM3
TR1(3,1)=AN1
TR1(3,2)=AN2
TR1(3,3)=AN3
DO 200 I=1,3
DO 200 J=1,3
200 WRITE(6,*)'DCS','I=',I,'J=',J,TR1(I,J)
RETURN
END
SUBROUTINE LOCAEP(B1,D1,XP)
DIMENSION B1(3),D1(3),XP(4,1)
TR=22./(180.*7.)
al2=sqrt(d1(1)**2+d1(2)**2)
al1=sqrt(b1(1)**2+b1(2)**2)
s1=al2*(b1(3)-d1(3))/(al2-al1)
theta=atan(s1/al2)
alfa=90.*tr-theta
s2=10.-(s1+d1(3))
xp(2,1)=s2*sin(alfa)
s3=s2*cos(alfa)
xp(1,1)=-(s3+(al2/cos(theta)))
xp(3,1)=0.0
```

```
      xp(4,1)=1.
      RETURN
      END
      SUBROUTINE GLOBTRA(TR3,T)
      DIMENSION TR3(4,4),T(12,12)
      DO 1 I=1,3
      DO 1 J=1,3
    1 T(I,J)=TR3(J,I)
      N=0
      DO 10 I=1,3
      N=N+3
      DO 10 J=1,3
      DO 10 KK=1,3
      T(J+N,KK+N)=T(J,KK)
   10 CONTINUE
      RETURN
      END
      SUBROUTINE CHECK(A,B,XP,XPO,P)
C     HERE CHECK IS APPLIED FOR CORRECT SIGN OF D.C.S
      DIMENSION A(4,4),X(4,1),B(4,4),XP(4,1),XPO(4,1)
      DIMENSION UDIFF(3),P(3)
      CALL VMULFF(A,xP,4,4,1,4,4,xp0,4,IER)
C     PRINT*,'GLOBAL POINT P FROM IST SET',XPO(1,1),XPO(2,1),XPO(3,1)
      UTOL=1.E-02
      DO 12 I=1,3
      UDIFF(I)=ABS(XPO(I,1)-P(I))
      IF(UDIFF(I).LE.UTOL) GO TO 20
      GO TO 30
   20 CONTINUE
   12 CONTINUE
      GO TO 40
   30 DO 11 I=1,3
      DO 11 J=2,3
   11 A(I,J)=-1.*A(I,J)
   40 CONTINUE
      DO 50 I=1,4
```

```
      DO 50 J=1,4
   50 B(I,J)=A(I,J)
      CALL VMULFF(B,XP,4,4,1,4,4,XPO,4,IER)
C     PRINT*,'GLOBAL POINT P FROM 2ND SET',XPO(1,1),XPO(2,1),XPO(3,1)
      RETURN
      END
C*****************************************************************C
C SUBROUTINE FREQUENCY CALCULATES THE NATURAL FREQUENCY OF THE C
C THE ROBOTIC MANIPULATOR BY FORMING THE DYNAMIC MATRIX        C
C*****************************************************************C
      SUBROUTINE FREQUENCY(GM,GK,N)
      DIMENSION GMM(100,100),GKK(100,100),GM(100,100),GK(100,100),
     1  WKAREA(100000),ALAMDA(100,100),GKKINV(100,100)
      COMPLEX  TILAMDA(100,100),EIGVAL(100),EIGVEC(100,100)
      COMMON/BLK1/OMEGA(100)
C       FIXED BOUNDARY CONDITIONS ARE APPLIED
      DO 30 I=1,N-6
      DO 30 J=1,N-6
      GMM(I,J)=GM(I+6,J+6)
   30 GKK(I,J)=GK(I+6,J+6)
      WRITE(17,*)((GKK(I,J),J=1,N-6),I=1,N-6)
      WRITE(18,*)((GMM(I,J),J=1,N-6),I=1,N-6)
      NN=N-6
      IDGT=4
      CALL LINV2F(GKK,NN,100,GKKINV,IDGT,WKAREA,IER)
      CALL VMULFF(GKKINV,GMM,NN,NN,NN,100,100,ALAMDA,100,IER)
      DO 40 I=1,NN
      DO 40 J=1,NN
   40 TILAMDA(I,J)=CMPLX(ALAMDA(I,J))
C DETERMINATION OF EIGEN VALUES
      IJOB=1
      CALL EIGCC(TILAMDA,NN,100,IJOB,EIGVAL,EIGVEC,100,WKAREA,IER)
      DO 50 I=1,NN
   50 EIGVAL(I)=1./EIGVAL(I)
      DO 70 I=1,NN
      OMEGA(I)=SQRT(REAL(EIGVAL(I)))
```

```
      OMEGA(I)=OMEGA(I)*7./(2.*22.)
   70 CONTINUE
      DO 80 I=1,NN
      DO 80 J=I+1,NN
      IF (OMEGA(I).GT.OMEGA(J)) THEN
      TEMP=OMEGA(I)
      OMEGA(I)=OMEGA(J)
      OMEGA(J)=TEMP
      END IF
   80 CONTINUE
      RETURN
      END
C#####################################################################C
C SUBROUTINE STIFF CALCULATES THE STIFFNESS MATRIX                    C
C#####################################################################C
      SUBROUTINE STIFF(AL,E,G,A,AIX,AIY,AIZ,AK)
      DIMENSION AK(12,12)
      AK(1,1)=E*A/AL
      AK(2,2)=12.*E*AIZ/AL**3
      AK(3,3)=12.*E*AIY/AL**3
      AK(4,4)=G*AIX/AL
      AK(5,3)=-6.*E*AIY/AL**2
      AK(5,5)=4.*E*AIY/AL
      AK(6,2)=6.*E*AIZ/AL**2
      AK(6,6)=4.*E*AIZ/AL
      AK(7,1)=-E*A/AL
      AK(7,7)=A*E/AL
      AK(8,2)=-12.*E*AIZ/AL**3
      AK(8,6)=-6.*E*AIZ/AL**2
      AK(8,8)=12.*E*AIZ/AL**3
      AK(9,3)=-12.*E*AIY/AL**3
      AK(9,5)=6.*E*AIY/AL**2
      AK(9,9)=12.*E*AIY/AL**3
      AK(10,4)=-G*AIX/AL
      AK(10,10)=G*AIX/AL
      AK(11,3)=-6.*E*AIY/AL**2
```

```
      AK(11,5)=2.*E*AIY/AL
      AK(11,9)=6.*E*AIY/AL**2
      AK(11,11)=4.*E*AIY/AL
      AK(12,2)=6.*E*AIZ/AL**2
      AK(12,6)=2.*E*AIZ/AL
      AK(12,8)=-6.*E*AIZ/AL**2
      AK(12,12)=4.*E*AIZ/AL
      DO 10 I=1,11
      DO 10 J=I+1,12
   10 AK(I,J)=AK(J,I)
      RETURN
      END
C*****************************************************************C
C SUBROUTINE MASS CALCULATES THE MASS MATRIX                      C
C*****************************************************************C
      SUBROUTINE MASS(AL,A,DEN,AIX,AM)
      DIMENSION AM(12,12)
      AM(1,1)=1./3.
      AM(2,2)=13./35.
      AM(3,3)=13./35.
      AM(4,4)=AIX/3./A
      AM(5,3)=-11.*AL/210.
      AM(5,5)=AL**2/105.
      AM(6,2)=11.*AL/210.
      AM(6,6)=AL**2/105.
      AM(7,1)=1./6.
      AM(7,7)=1./3.
      AM(8,2)=9./70.
      AM(8,6)=13.*AL/420.
      AM(8,8)=13./35.
      AM(9,3)=9./70.
      AM(9,5)=-13.*AL/420.
      AM(9,9)=13./35.
      AM(10,4)=AIX/(6.*A)
      AM(10,10)=AIX/(3.*A)
      AM(11,3)=13.*AL/420.
```

```
AM(11,5)=-AL**2/140.
AM(11,9)=11.*AL/210.
AM(11,11)=AL**2/105.
AM(12,2)=-13.*AL/420.
AM(12,6)=-AL**2/140.
AM(12,8)=-11.*AL/210.
AM(12,12)=AL**2/105.
DO 10 I=1,11
DO 10 J=I+1,12
10 AM(I,J)=AM(J,I)
DO 20 I=1,12
DO 20 J=1,12
20 AM(I,J)=AM(I,J)*DEN*A*AL
RETURN
END
C*****************************************************************C
C SUBROUTINE TRANSFORM CALCULATES THE DIRECTION COSINES OF     C
C THE LINKS AND GET THE ELEMENTAL MASS AND STIFFNESS MATRIX    C
C INTO GLOBAL COORDINATES                                      C
C*****************************************************************C
SUBROUTINE TRANSFORM(A,B,AT)
DIMENSION A(3),B(3),AT(12,12)
AX=A(1)
AY=A(2)
AZ=A(3)
BX=B(1)
BY=B(2)
BZ=B(3)
PX=0.
PY=10.
PZ=0.
ABX=BX-AX
ABY=BY-AY
ABZ=BZ-AZ
ABLEN=SQRT(ABX*ABX + ABY*ABY + ABZ*ABZ)
CX=ABX/ABLEN
```

```
CY=ABY/ABLEN
CZ=ABZ/ABLEN
CONS=SQRT(CX*CX+CZ*CZ)
XP=CX*PX + CY*PY + CZ*PZ
YP=-CX*CY*PX/CONS + CONS*PY - CY*CZ*PZ/CONS
ZP=-CZ*PX/CONS + CX*PZ/CONS
ALFA = ATAN(ZP/YP)
  AT(1,1)=CX
AT(1,2)=CY
AT(1,3)=CZ
AT(2,1)=(-CX*CY*COS(ALFA)-CZ*SIN(ALFA))/CONS
AT(2,2)=CONS*COS(ALFA)
AT(2,3)=(-CY*CZ*COS(ALFA)+CX*SIN(ALFA))/CONS
AT(3,1)=(CX*CY*SIN(ALFA)-CZ*COS(ALFA))/CONS
AT(3,2)=-CONS*SIN(ALFA)
AT(3,3)=(CY*CZ*SIN(ALFA)+CX*COS(ALFA))/CONS
N=0
DO 10 I=1,3
N=N+3
DO 10 J=1,3
DO 10 KK=1,3
10 AT(J+N,KK+N)=AT(J,KK)
RETURN
END
C*****************************************************************C
C SUBROUTINE GLOBAL ADD THE MOTOR AND GRIPPER MASSES TO    C
C THE THE RESPECTIVE LINK (TO THE DIAGONAL ELEMENT)        C
C*****************************************************************C
SUBROUTINE GLOBAL(N2,GM2,GK2,N3,GM3,GK3,GM23,GK23)
DIMENSION GM2(100,100),GK2(100,100),GM3(100,100),GK3(100,100),
     1  GM23(100,100),GK23(100,100),TEMP1(100),TEMP2(100)
DO 5 I=1,100
DO 5 J=1,100
GM23(I,J)=0.
5 GK23(I,J)=0.
BMOMAS=50./(N2+1)
```

```
GRIMAS=25.
C  THE MOTOR MASES ARE ADDED
DO 7 I=1,(N2+1)*6,6
DO 7 II=1,3
7 GM2(II+I-1,II+I-1)=GM2(II+I-1,II+I-1)+BMOMAS
C  THE GRIPPER MASSES ARE ADDED
GM3(N3*6+1,N3*6+1)=GM3(N3*6+1,N3*6+1)+GRIMAS
GM3(N3*6+2,N3*6+2)=GM3(N3*6+2,N3*6+2)+GRIMAS
GM3(N3*6+3,N3*6+3)=GM3(N3*6+3,N3*6+3)+GRIMAS
DO 10 I=1,N2*6+6
DO 10 J=1,N2*6+6
GM23(I,J)=GM2(I,J)
10 GK23(I,J)=GK2(I,J)
DO 20 I=1,N3*6+6
DO 20 J=1,N3*6+6
GM23(N2*6+I,N2*6+J)=GM23(N2*6+I,N2*6+J)+GM3(I,J)
20 GK23(N2*6+I,N2*6+J)=GK23(N2*6+I,N2*6+J)+GK3(I,J)
N=(N2+N3)*6+6
DO 30 I=1,6
DO 30 J=1,N
TEMP1(J)=GM23(I,J)
TEMP2(J)=GK23(I,J)
GM23(I,J)=GM23(I+6,J)
GK23(I,J)=GK23(I+6,J)
GM23(I+6,J)=TEMP1(J)
30 GK23(I+6,J)=TEMP2(J)
DO 40 J=1,6
DO 40 I=1,N
TEMP1(I)=GM23(I,J)
TEMP2(I)=GK23(I,J)
GM23(I,J)=GM23(I,J+6)
GK23(I,J)=GK23(I,J+6)
GM23(I,J+6)=TEMP1(I)
40 GK23(I,J+6)=TEMP2(I)
RETURN
END
```

```
C*****************************************************************C
C SUBROUTINE ASSMBLY CALCULATES THE GLOBAL MASS AND STIFFNESS
C MATRIX                                                          C
C*****************************************************************C
      SUBROUTINE ASSMBLY(N1,GM1,GK1,N,GM23,GK23,GM,GK)
      DIMENSION GM1(100,100),GK1(100,100),GM23(100,100),
     1  GK23(100,100),GM(100,100),GK(100,100)
      DO 5 I=1,100
      DO 5 J=1,100
      GM(I,J)=0.
    5 GK(I,J)=0.
      DO 20 I=1,N1*6+6
      DO 20 J=1,N1*6+6
      GM(I,J)=GM1(I,J)
   20 GK(I,J)=GK1(I,J)
      DO 30 I=1,N
      DO 30 J=1,N
      GM(N1*6+I,N1*6+J) =GM(N1*6+I,N1*6+J)+GM23(I,J)
   30 GK(N1*6+I,N1*6+J)=GK(N1*6+I,N1*6+J)+GK23(I,J)
      RETURN
      END
      SUBROUTINE RGMPRD(A,C,R,N,M,L)
      REAL A(1),C(1),R(1)
      IR=0.0
      IK=-M
      DO 13 K=1,L
      IK=IK+M
      DO 13 J=1,N
      IR=IR+1
      JI=J-N
      IC=IK
      R(IR)=0
      DO 13 I=1,M
      JI=JI+N
      IC=IC+1
   13 R(IR)=R(IR)+A(JI)*C(IC)
```

```
      RETURN
      END
```