Kinematic Analysis and Animation of a

Variable Geometry Truss Robot

by

Dipen P. Gokhale

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Mechanical Engineering

APPROVED:

Dr. C. F. Reinholtz, Chairman

Dr. H. H. Robertshaw,

Dr. A. Myklebust.

December 3, 1987

Blacksburg, Virginia

Kinematic Analysis and Animation of a

Variable Geometry Truss Robot

by

Dipen P. Gokhale

Dr. C. F. Reinholtz, Chairman

Mechanical Engineering

(ABSTRACT)

In this thesis, forward and inverse kinematic equations are developed for a particular type of parallel, closed-loop manipulator known as the Variable Geometry Truss or VGT for short. Widely recognized as adaptive or collapsing structures for space and military applications, VGTs have not received due consideration as robotic manipulators. VGTs undoubtedly represent an important sector of future manipulator applications. VGTs are typically constructed using repeating identical cells or modules and they have exceptional stiffness to weight ratios.

The data obtained from solving the forward kinematic equations is used for animation of the VGT. For animation, three dimensional graphics software, graPHIGS is used. Additionally, the kinematic analysis equations are used to map out workspace of the VGT. An experiment is also carried out to verify the computational results.

Acknowledgements

I would like to take this opportunity to thank my advisor, Dr. Charles Reinholtz, for his guidance throughout the research and preparation of this thesis. I would also like to thank Dr. H. H. Robertshaw and Dr. A. Myklebust for serving on my advisory and examining committee.

In the course of this thesis, a number of fellow students helped in diverse capacities. I would like to thank them all and in particular thanks to , , , , and .

I would also like to thank my parents for their support throughout my education.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction and Literature Review

1.1 Introduction

A new family of robotic manipulators known as Variable Geometry Trusses (VGTs) has evolved in recent years. A VGT is a truss that has some variable length members and therefore is capable of changing geometry. The degrees of freedom of the truss are determined by the number of variable links of the truss. A VGT can be described in simple terms as, a truss that can purposefully vary its geometric configuration by changing length of its variable members. A well known mechanism that can be considered to be a VGT is the Stewart platform. As shown in the Fig. 1, the Stewart platform consists of two plane platforms and six variable length links that connect the two platforms through spheric-prismatic-spheric joints. Stewart's platform has six degrees of freedom and finds applications as an aircraft simulator, and a space vehicle simulator.

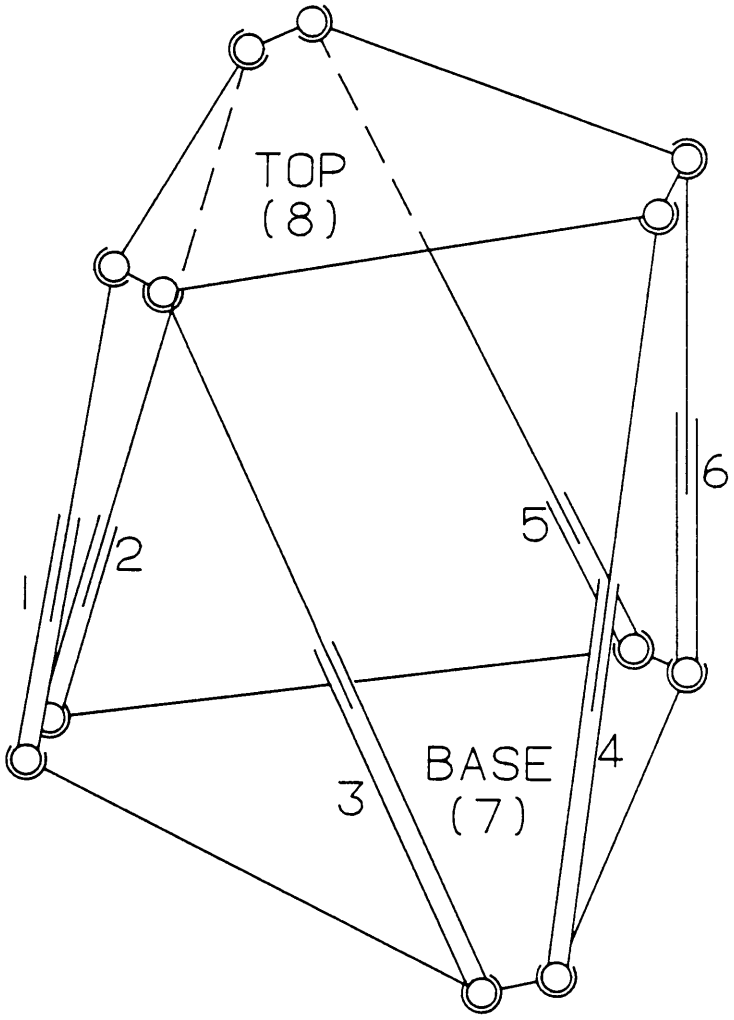
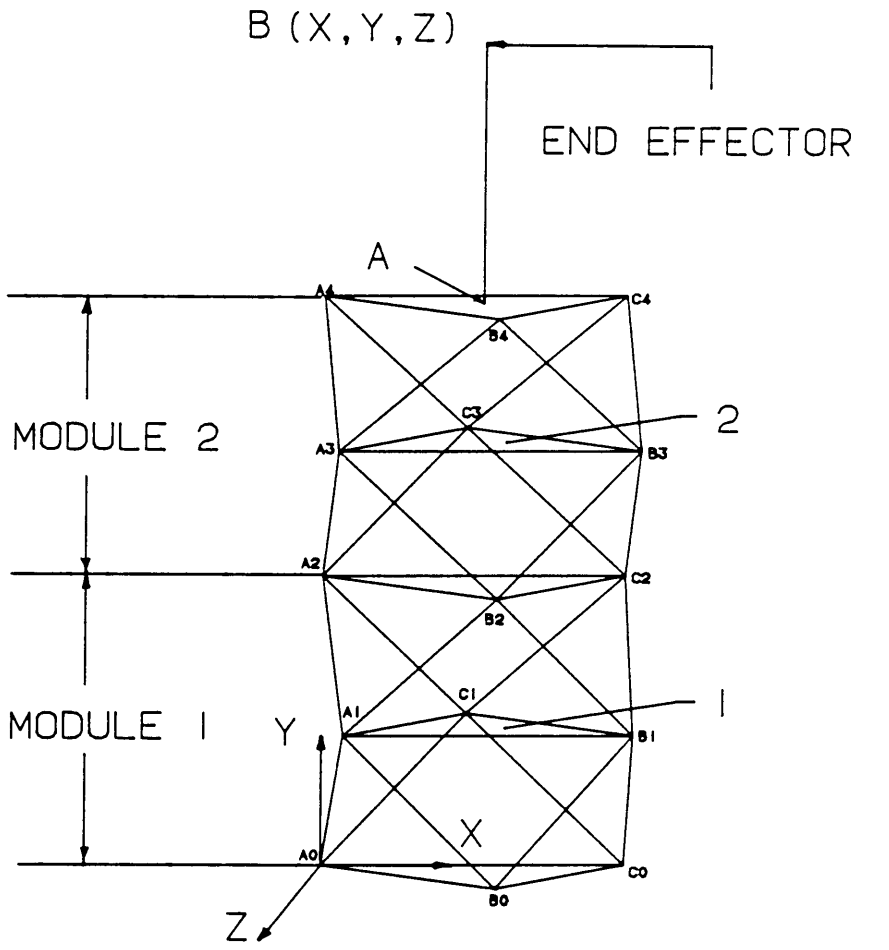


Figure 1. Stewart Platform

VGTs have very high stiffness due to their truss structure. They are also capable of being folded down and stored compactly, which enables a VGT to be folded down during transportation and deployed at the sight of application. These properties make VGTs an ideal candidate as a structural support member for space applications. Various VGT geometries have been studied for their application in space as a structural member. The report by Rockwell International [1] discusses in detail various VGT geometries for their application in space structures. This report discusses various criteria such as high stiffness, strength, and the variable geometry requirement in selection of the deployable members. The report discusses and evaluates eight candidate structures. The report also describes large deployable volumes such as Habitat modules, Tunnels, OTV hangers, in which the VGT is an elementary member. The report by Cox and Nelson [2] is similar to the report by Rockwell International. It discusses a VGT geometry in detail and its potential uses in space. The paper by Miura and Furuya [3] discusses concept of an adaptive structure for space applications. This paper suggests various applications for the adaptive structure such as, structural members in space station supports, large space systems, supports for space antenna etc. The report by Rhodes and Mikulas [4] also discusses a candidate VGT geometry for use in space, its controllablity and kinematics.

Although various VGT geometries have been studied extensively keeping in mind their application in space, a less apparent but equally important application of VGTs as robotic manipulators has not received enough attention. This thesis discusses a specific VGT geometry, NASA's Octahedral VGT, as shown in Fig. 2 for its application as a robotic manipulator.



VARIABLE GEOMETRY TRUSS ROBOT

1) PLANE OF VARIABLE LINK LENGTHS. (L1, L2, L3)

2) PLANE OF HAND ADJUSTABLE LINKS

Figure 2. NASA's Octahedral VGT.

NASA's Octahedral VGT

The VGT shown in Fig. 2, consists of a repeating structure, called a module. In the present case the VGT has two modules. In both the modules the links of the middle, lateral plane are variable in length. The variable links of the first module are changed using motor driven leadscrews, whereas the variable links of the second module are hand adjustable. A stiff rod (AB) is fixed to the top plane of the VGT to act as an end effector. Note that the VGT has only three degrees of freedom, corresponding to the three independently varying, motor driven links. The length of variable links varies from 39 inches to 51 inches. This constraint is mainly due to hardware limitation.

Forward and inverse kinematic solutions are developed for this VGT. Using the kinematic solutions, the workspace of the VGT is examined. The data obtained from the forward kinematic solution is further used in animation of the VGT. For animation, graPHIGS, three dimensional graphics software, is used.

1.2 Literature Review

This literature search focuses on two important areas. The first area is parallel or closed loop robots and the second area is VGT configurations that have been studied for their application as adaptive structures.

Closed loop or parallel robotic manipulators is relatively a new concept as compared to serial or open loop robotic manipulators. Serial robotic manipulators evolved first because of simplicity in their construction. They are extensively used in industry. The advantages of serial manipulators are their long reach, large range of motion i.e. larger workspace, and their capacity to reach into small space due to their compact sizes. But these manipulators also have some serious disadvantages. Serial

manipulators are structurally a cantilever. This cantilever structure limits capacity of the manipulators to carry load. This also poses constraints on the design of the robots in the sense that the joint actuators, which are heavy, must be located near the robot base. These manipulators are also difficult to control due to their kinematic indeterminancy. As an alternative to serial manipulators, parallel manipulators were introduced. These manipulators can be described as robot mechanisms in which the end effector is connected to the ground link through two or more linkages. The main advantages of parallel robots over the serial robots are, their ability to carry large loads due to their high stiffness, which stems from their truss structure. These robots eliminate to a great extent the problem of control due to kinematic indeterminancy as present in serial robots. The main disadvantages of parallel robots are their small range of motion and small length of reach as compared to serial robots. Note that VGTs can be considered to be subset of the parallel, closed loop robot manipulators.

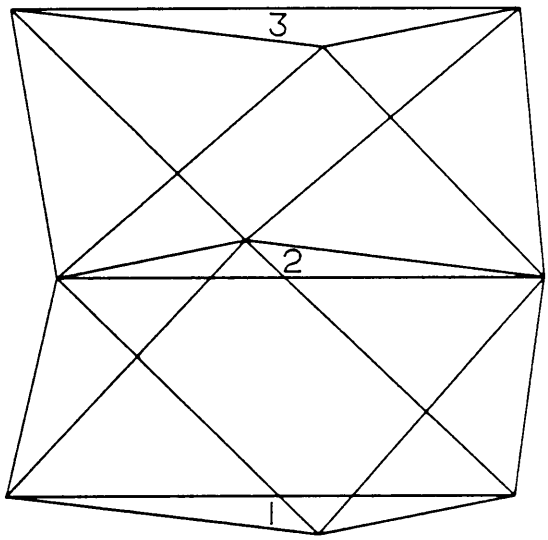
Various kinematic structures have been suggested as candidate geometries for parallel manipulators. Hunt [5] suggested the use of a Stewart platform (a parallel structure) as a robot manipulator. Following Hunt's idea, Fitcher and McDowell [6] presented a review and some preliminary design concepts for parallel manipulators. They have given comparison of serial and parallel manipulators and have suggested various applications for the Stewart platform based manipulator arm (SPMA). Yang and Lee [7] in their paper, have the studied feasibility of platform type robotic manipulators from a kinematic viewpoint. They have considered a specific geometric configuration of the Stewart platform and studied the forward and inverse kinematics problem for that configuration. Using the kinematic solutions they have also studied the range of motion of the robot. Fitcher [8] has examined the kinematics of generalized Stewart platform and has presented a set of equations which cover the gross motion as well the differential motion of the platform. He has also studied the effect of some simplifying assumptions

for the geometric configuration, on the mathematics of kinematic equations of the manipulator. He has considered various Stewart platform configurations that may be used as manipulators.

Besides the Stewart platform, other closed loop, parallel mechanisms have also been studied for their use as a robot manipulator. Hunt [9] has systematically studied the in-parallel actuated robot arm. In this paper, he has reviewed many in-parallel structures for their possible use as a parallel-actuated robot arm. He has also mentioned the possibility of combining serial and parallel mechanisms. Earl and Rooney [10] discuss various kinematic structures for use in robotic manipulator design. In this paper they have considered general properties of these kinematic structures and have given methods for combining two structures. They have also suggested some new designs. Ardayfio and Qiao [11] have studied forward and inverse kinematic solutions for various mechanisms such as 1) the in-parallel actuated robot arm 2) the multiple input robot arm and 3) the parallel and serial input robot arm. Using the kinematic equations they have also developed software for kinematic simulation for the mechanisms.

Trusses of the VGT family have been studied by researchers Sincarsin and Hughes [12], Miura and Furuya [3] for their use as robot manipulators. For solving the kinematic equations they have used projections of the linkages on the base plane. In this thesis for NASA's Octahedral VGT, an intuitive approach is used. Chapter 2 describes formulation and solution of kinematic equations using this approach.

ONE MODULE OF A VGT



1) MIURA'S VGT-

-PLANES 2 AND 3 HAVE VARIABLE LENGTH MEMBERS

-D.O.F. = 6

2) NASA'S OCTAHEDRAL VGT

-ONLY PLANE 2 HAS VARIABLE LENGTH MEMBERS

-D.O.F. = 3

Figure 3. VGT geometry studied by Miura

Chapter 2

Kinematic analysis of NASA's Octahedral VGT

To control a robotic manipulator effectively and to predict its behavior for a set of input parameters (which vary from robot to robot), it is very important to have prior knowledge of the following three aspects of the robot:

- ***Forward kinematic solution :*** Knowledge of the forward kinematic solution for a robot means, to know position and orientation of the end effector of the robot for a given set of input parameters.
- ***Inverse kinematic solution :*** Knowledge of the inverse kinematic solution for a robot enables user to know the values of input parameters for a given position and orientation of the end effector.
- ***Workspace of the robot :*** Workspace of a robot manipulator means the reach of the robot for a given set of geometrical parameters. Knowledge of the workspace helps in effectively adjusting robot position in its operating environment.

This chapter discusses these three important aspects for the NASA's Octahedral VGT.

2.1 Forward Kinematic Solution for NASA's Octahedral VGT

The forward kinematics problem for a robotic manipulator is defined as follows; *computation of the position and orientation of the end effector for given values of the link parameters and the input actuators.* For a typical serial robot manipulator, such as the one shown in Fig. 4, the forward kinematics problem is to find position and orientation of the end effector (AB) for given values of the link parameters (L_1, L_2) and the input actuator parameters (θ_1, θ_2).

For NASA's Octahedral VGT, the forward problem is to calculate the coordinates of the end effector (AB) for a given set of variable link lengths. Note that the problem is solved in parts. This approach is taken because the VGT consists of repeating modules. The forward problem is solved for one module and then repeated for succeeding modules. The forward solution for a module is discussed in following paragraphs.

Figure 5 shows the kinematic diagram of one module of the octahedral VGT on loan from NASA's Langley Research Center. As shown in the figure, the three links of the triangle A1B1C1 are independently variable in length, thus the module has three degrees of freedom. The forward kinematics problem for one module of the VGT can now be defined as; *to find the position (coordinates) of the joints A2, B2 and C2 for a given set of values of variable link lengths A1B1, B1C1, and C1A1 (i.e.* L_1, L_2, *and* L_3 *respectively).*

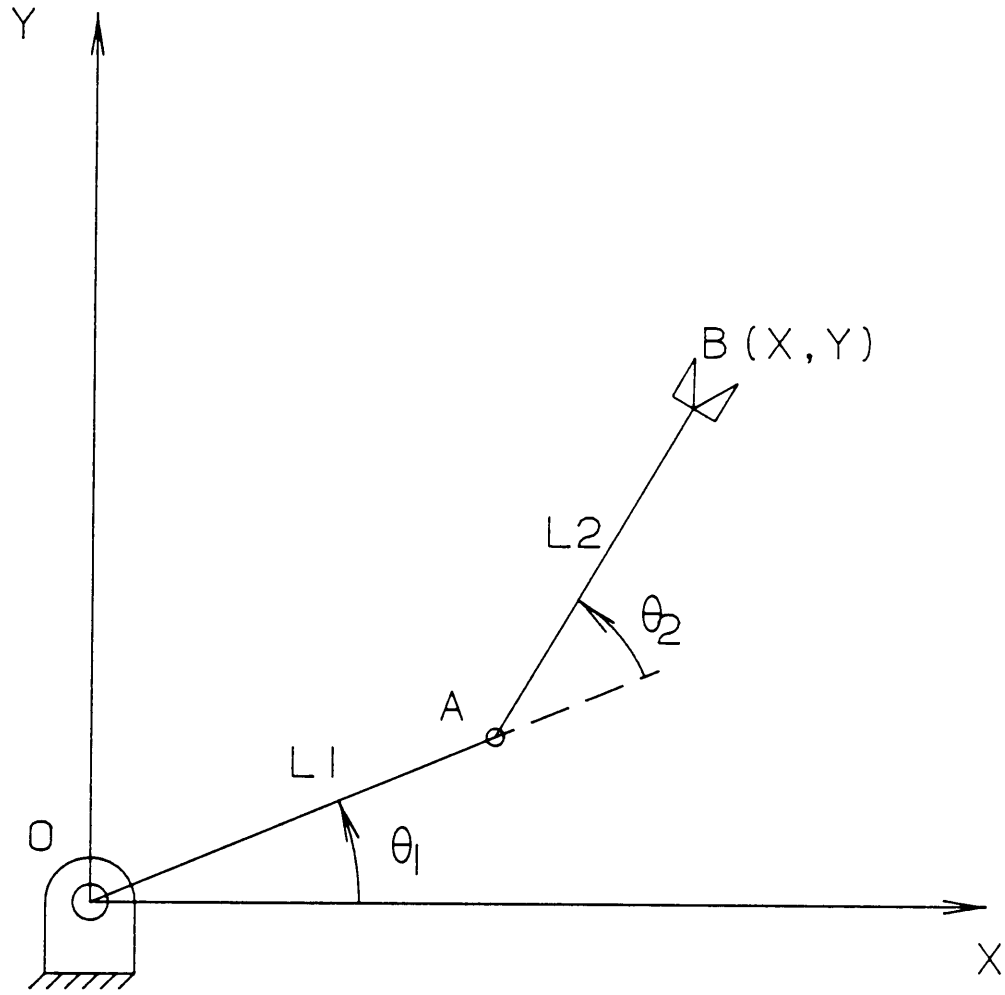
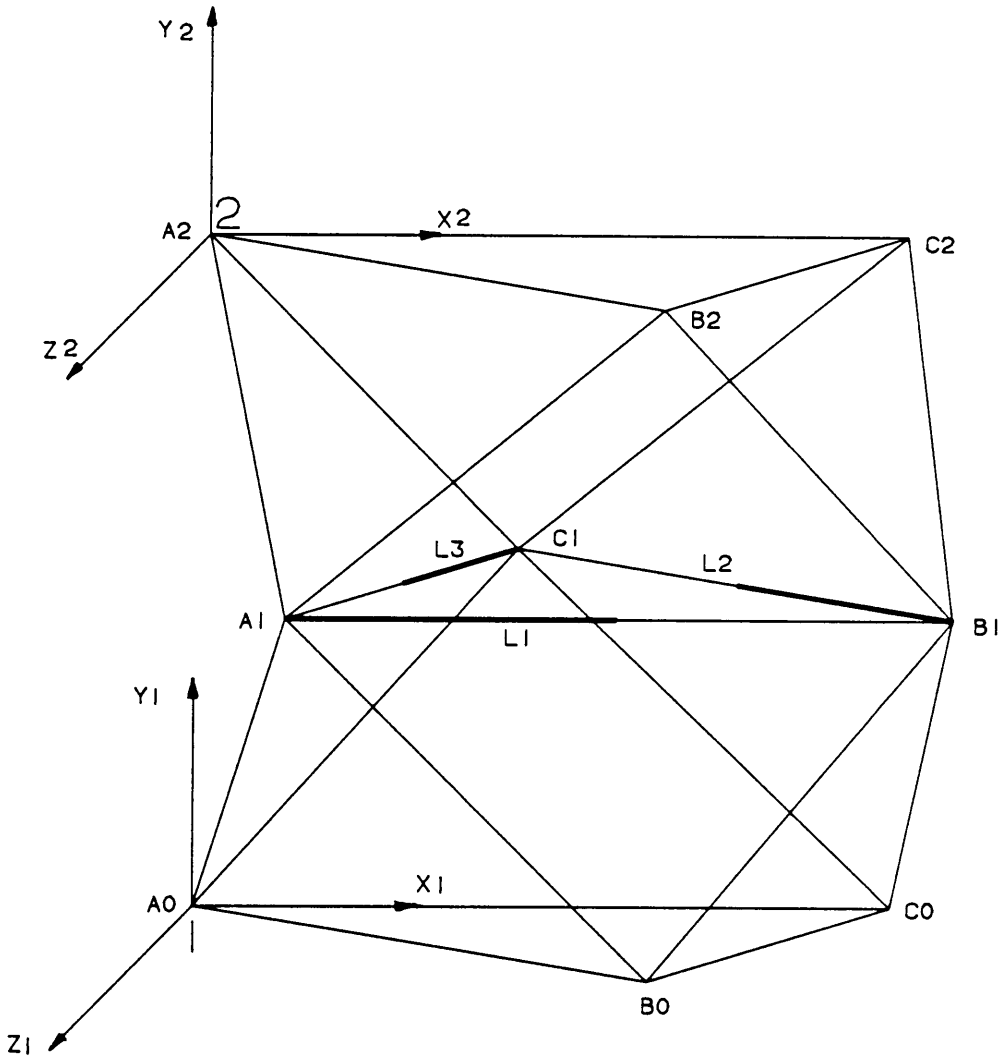


Figure 4. Serial Robot Manipulator.



1 -GLOBAL COORDINATE SYSTEM

(COORDINATE SYSTEM OF BASE MODULE)

2 - LOCAL COORDINATE SYSTEM

(COORDINATE SYSTEM OF MODULE 2)

Figure 5. Base Module of NASA's Octahedral VGT.

As mentioned in the previous chapter, the forward problem has been solved using projections of the links on the base plane for robots of the VGT family [13, 4] . Although this basic approach is correct, the equations to be solved become unnecessarily complicated. In the present thesis, a more intuitive approach is used to reduce the complexity of the equations to be solved.

Three RSSR Mechanism Representation of the VGT

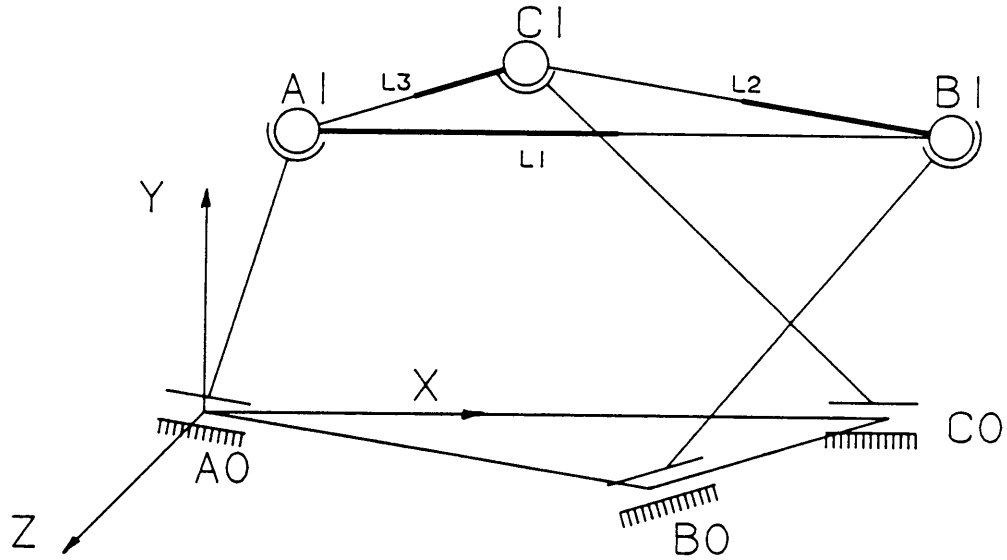
Figure 6 shows the lower half of the VGT of Fig. 5 modeled as a ten link mechanism consisting of three revolute joints, three prismatic joints and six spheric joints. This mechanism has six degrees of freedom as given by the Kutzbach equation, [15] ,

$$\text{Degrees of Freedom} = 6(10 - 1) - 6(5) - 6(3) = 6$$

Out of these six degrees of freedom, three degrees of freedom are idle rotations of the two-link chains A1B1, B1C1 and C1A1 between the spheric joints, hence effectively the mechanism has only three degrees of freedoms. This gives a kinematic approximation of VGT which is equivalent in terms of mobility. This mechanism can also be visualized as consisting of three RSSR mechanisms A0-A1-B1-B0, B0-B1-C1-C0, and C0-C1-A1-A0, each having one degree of freedom. The reason for doing so is that the RSSR mechanisms can be easily analyzed using known solution methods.

For the three RSSR mechanisms with a given set of lengths (L_1, L_2, and L_3) between the S-S joints, the unknowns are the three angles (θ_1, θ_2, and θ_3) the links A0-A1, B0-B1, and C0-C1 make with the ground plane (plane A0B0C0). Thus, the data and unknowns for the forward problem using three RSSR mechanism representation of VGT are:

THREE RSSR MECHANISM APPROXIMATION OF THE VGT



THE THREE RSSR MECHANISMS ARE:

1) A0-A1-B1-B0

2) B0-B1-C1-C0

3) C0-C1-A1-A0

Figure 6. Three RSSR Mechanism approximation of VGT.

Unknowns - ($\theta_1, \theta_2, \theta_3$).

Input Data - (L1, L2, L3).

The constraint used in solving for the unknowns is that all of the three RSSR mechanisms should assemble simultaneously.

Kinematic Analysis of the RSSR mechanism

This section discusses kinematic analysis for one RSSR mechanism (mechanism A0-A1-B1-B0). Equations using the known length between the two spheric joints are written [16]. This concept is then simultaneously applied to all three of RSSR mechanisms that comprise the VGT.

All of the links of the RSSR mechanism shown in Fig. 7 are defined in length. The unknowns of this mechanism are the angles the two grounded links make with the ground (angles θ_1, θ_2). These two unknown angles can be correlated using the known value of link length between the two spheric joints as given by the following equation,

$$L_1^2 - (\bar{a} - \bar{b}) \bullet (\bar{a} - \bar{b}) = 0 \qquad [2.1]$$

where,

L_1 is the known length between the spheric joints.

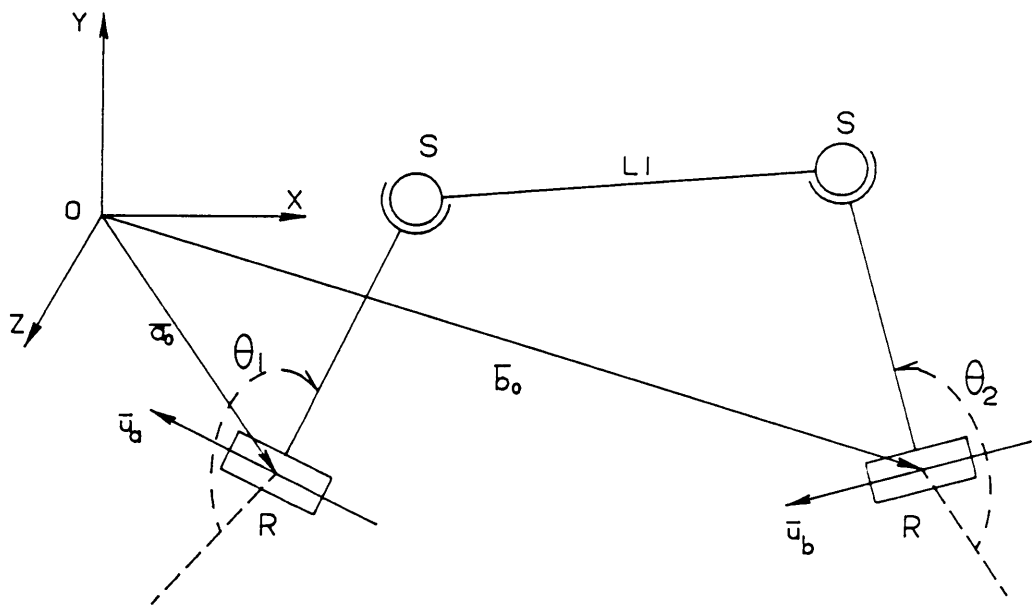
$\bar{a}$ and $\bar{b}$ are the position vectors of the two spheric joints.

Vectors $\bar{a}$ and $\bar{b}$ in equation (2.1) can be expressed in terms of the input and output angles as follows:

$$\bar{a} = [R_{\theta_1, \bar{u}_a}](\bar{a}_1 - \bar{a}_0) + (\bar{a}_0)$$

$$\bar{b} = [R_{\theta_2, \bar{u}_b}](\bar{b}_1 - \bar{b}_0) + (\bar{b}_0)$$

ANALYSIS OF ONE RSSR MECHANISM



L1 IS THE KNOWN LINK LENGTH

θ_1 ; θ_2 ARE THE UNKNOWN ANGLES

Figure 7. RSSR Mechanism.

where,

$\bar{a}_0$, $\bar{b}_0$ are the vector locations of the revolute joints of the mechanism with respect to the global coordinate system,

$\bar{a}_1$, $\bar{b}_1$ are the predefined starting positions of the spheric joints for $\theta_1 = 0$ and $\theta_2 = 0$,

Note that, $L_1^2 = (\bar{a}_1 - \bar{b}_1) \bullet (\bar{a}_1 - \bar{b}_1)$ and,

$[R_{\theta,\bar{u}}]$ is the axis rotation matrix expressing rotation around the unit vector $\bar{u}$ by an amount θ, [15].

The terms in equation (2.1) can be expanded and the equation can be written as,

$$L_1^2 - (\bar{a} \bullet \bar{a}) - 2(\bar{a} \bullet \bar{b}) + (\bar{b} \bullet \bar{b}) = 0 \qquad [2.2]$$

Expanding the terms containing the vector $\bar{b}$ in the equation (2.2) gives:

$$2(\bar{a} \bullet \bar{b}) = 2((\bar{a} \bullet \bar{b}_0) + \bar{a} \bullet [R_{\theta_2,\bar{u}_b}](\bar{b}_1 - \bar{b}_0)) \qquad [2.3a]$$

and,

$$(\bar{b} \bullet \bar{b}) = (\bar{b}_0 \bullet \bar{b}_0) + 2\bar{b}_0 \bullet [R_{\theta_2,\bar{u}_b}](\bar{b}_1 - \bar{b}_0)$$

$$+ \{[R_{\theta_2,\bar{u}_b}](\bar{b}_1 - \bar{b}_0)\} \bullet \{[R_{\theta_2,\bar{u}_b}](\bar{b}_1 - \bar{b}_0)\} \qquad [2.3b]$$

The equation (2.3*b*) can further be reduced to,

$$(\bar{b} \bullet \bar{b}) = (\bar{b}_0 \bullet \bar{b}_0) + 2\bar{b}_0 \bullet [R_{\theta_2,\bar{u}_b}](\bar{b}_1 - \bar{b}_0) + (\bar{b}_1 - \bar{b}_0) \bullet (\bar{b}_1 - \bar{b}_0) \qquad [2.3c]$$

Substituting the equations (2.3*a*) and (2.3*c*) in equation (2.2) we get,

$$(\bar{a} \bullet \bar{a}) - 2(\bar{a} \bullet \bar{b}_0) + 2(\bar{b}_0 - \bar{a})[R_{\theta_2,\bar{u}_b}](\bar{b}_1 - \bar{b}_0)$$

$$+ (\bar{b}_0 \bullet \bar{b}_0) + (\bar{b}_1 - \bar{b}_0) \bullet (\bar{b}_1 - \bar{b}_0) - (\bar{a}_1 - \bar{b}_1) \bullet (\bar{a}_1 - \bar{b}_1) = 0 \qquad [2.4]$$

This equation can further be expanded and written as,

$$(\bar{a} \cdot \bar{a}) + (\bar{b}_0 \cdot \bar{b}_0) + (\bar{b}_1 - \bar{b}_0) \cdot (\bar{b}_1 - \bar{b}_0) - 2(\bar{a} \cdot \bar{b}_0) - (\bar{a}_1 - \bar{b}_1) \cdot (\bar{a}_1 - \bar{b}_1)$$

$$+ 2(\bar{b}_0 - \bar{a})\{([I] - [Q_{\bar{u}_b}])(\bar{b}_1 - \bar{b}_0) \cos\theta_2 + [P_{\bar{u}_b}](\bar{b}_1 - \bar{b}_0) \sin\theta_2\}$$

$$+ [2(\bar{b}_0 - \bar{a})]\{[Q_{\bar{u}_b}](\bar{b}_1 - \bar{b}_0)\} = 0 \qquad [2.5]$$

Equation (2.5) can be written in simplified form as,

$$E \cos\theta_2 + F \sin\theta_2 + G = 0 \qquad [2.6]$$

where,

E, F and G are functions of θ_1 as given below:

$$E = (\bar{a} - \bar{b}_0) \cdot [I - Q_{\bar{u}_b}](\bar{b}_1 - \bar{b}_0)$$

$$F = (\bar{a} - \bar{b}_0) \cdot [P_{\bar{u}_b}](\bar{b}_1 - \bar{b}_0)$$

$$G = (\bar{a} - \bar{b}_0) \cdot [Q_{\bar{u}_b}](\bar{b}_1 - \bar{b}_0)$$

$$+ 0.5 \times [(\bar{a}_1 - \bar{b}_1) \cdot (\bar{a}_1 - \bar{b}_1) - (\bar{a} - \bar{b}_0) \cdot (\bar{a} - \bar{b}_0) - (\bar{b}_1 - \bar{b}_0) \cdot (\bar{b}_1 - \bar{b}_0)]$$

and where,

$$[P_{\bar{u}_b}] = \begin{bmatrix} 0 & -u_{bz} & u_{by} \\ u_{bz} & 0 & -u_{bx} \\ -u_{by} & u_{bx} & 0 \end{bmatrix}$$

$$[Q_{\bar{u}_b}] = \begin{bmatrix} u_{bx}^2 & u_{bx}u_{by} & u_{bx}u_{bz} \\ u_{bx}u_{by} & u_{by}^2 & u_{by}u_{bz} \\ u_{bx}u_{bz} & u_{bx}u_{bz} & u_{bz}^2 \end{bmatrix}$$

u_{bx}, u_{by} and u_{bz} are the three direction cosines of the unit vector $\bar{u}_b$.

Equation (2.6) relates the unknown angles θ_1, θ_2 and can be expressed as,

$$\text{FUN1}(\theta_1, \theta_2) = E\cos\theta_2 + F\sin\theta_2 + G = 0 \qquad [2.7]$$

Following the same analysis procedure for the other RSSR loops (mechanism B0-B1-C1-C0 and mechanism C0-C1-A1-A0) two more equations are formed as given below:

$$\text{FUN2}(\theta_2, \theta_3) = E1\cos\theta_3 + F1\sin\theta_3 + G1 = 0 \qquad [2.8]$$

where,

$$E1 = (\bar{b} - \bar{c}_0) \cdot [I - Q_{\bar{u}_c}](\bar{c}_1 - \bar{c}_0)$$

$$F1 = (\bar{b} - \bar{c}_0) \cdot [P_{\bar{u}_c}](\bar{c}_1 - \bar{c}_0)$$

$$G1 = (\bar{b} - \bar{c}_0) \cdot [Q_{\bar{u}_c}](\bar{c}_1 - \bar{c}_0)$$

$$+ 0.5 \times [(\bar{b}_1 - \bar{c}_1) \cdot (\bar{b}_1 - \bar{c}_1) - (\bar{b} - \bar{c}_0) \cdot (\bar{b} - \bar{c}_0) - (\bar{c}_1 - \bar{c}_0) \cdot (\bar{c}_1 - \bar{c}_0)]$$

and,

$$\text{FUN3}(\theta_3, \theta_1) = E2\cos\theta_2 + F2\sin\theta_2 + G2 = 0 \qquad [2.9]$$

where,

$$E2 = (\bar{c} - \bar{a}_0) \bullet [I - Q_{\bar{u}_a}](\bar{a}_1 - \bar{a}_0)$$

$$F2 = (\bar{c} - \bar{a}_0) \bullet [P_{\bar{u}_a}](\bar{a}_1 - \bar{a}_0)$$

$$G2 = (\bar{c} - \bar{a}_0) \bullet [Q_{\bar{u}_a}](\bar{a}_1 - \bar{a}_0)$$

$$+ 0.5 \times [(\bar{c}_1 - \bar{a}_1) \bullet (\bar{c}_1 - \bar{a}_1) - (\bar{c} - \bar{a}_0) \bullet (\bar{c} - \bar{a}_0) - (\bar{a}_1 - \bar{a}_0) \bullet (\bar{a}_1 - \bar{a}_0)]$$

The three RSSR mechanism representation of the VGT approximation gives three equations in terms of three unknowns. Equations (2.7), (2.8) and (2.9) are transcendental in form and cannot be easily solved in closed form. For this reason, an iterative method is used for solving these equations. In this thesis the Newton-Raphson root finding method for multivariables is used for solving the equations. This procedure is outlined below.

- First, the values of unknown variables are guessed as, $(\theta_1, \theta_2, \theta_3)^T$
- Using the known values L_1, L_2, and L_3 for link lengths and θ_1, θ_2, and θ_3 for the face angles, values of the functions FUN1, FUN2 and FUN3 are calculated.
- Depending on values of FUN1, FUN2 and FUN3 corrections in the initial guesses θ_1, θ_2, and θ_3 are calculated from,

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix} = - [J]^{-1} \begin{bmatrix} FUN1 \\ FUN2 \\ FUN3 \end{bmatrix}$$

where,

$[J]^{-1}$: is the inverse of the jacobian matrix, the jacobian matrix is given as,

$$[J] = \begin{bmatrix} \frac{\delta FUN1}{\delta\theta_1} & \frac{\delta FUN1}{\delta\theta_2} & 0 \\ 0 & \frac{\delta FUN2}{\delta\theta_2} & \frac{\delta FUN2}{\delta\theta_3} \\ \frac{\delta FUN3}{\delta\theta_1} & 0 & \frac{\delta FUN3}{\delta\theta_3} \end{bmatrix}$$

The values of the initial guesses are then modified to give new guesses as

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} + \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{bmatrix}$$

Corrections in the values of the initial guesses are carried out until the values of all the functions FUN1, FUN2, and FUN3 are less than a given tolerance limit (in this case 10^{-5}). If the number of iterations exceeds a specified maximum number of iterations, then the iterations are halted and it is assumed that the assembly is not possible. When the solution does converge, the corresponding values of the face angles are solution to the three simultaneous equations.

The face angles (θ_1, θ_2, θ_3) are then used in calculation of coordinates of joints **A1**, **B1** and **C1** from,

$$\mathbf{A1} = [R_{\theta_1, \bar{u}_a}](\bar{a}_1 - \bar{a}_0) + \bar{a}_0$$

$$\mathbf{B1} = [R_{\theta_2, \bar{u}_b}](\bar{b}_1 - \bar{b}_0) + \bar{b}_0$$

$$\mathbf{C1} = [R_{\theta_3, \bar{u}_c}](\bar{c}_1 - \bar{c}_0) + \bar{c}_0$$

where,

$\bar{u}_a =$ unit vector along $A0B0$.

$\bar{u}_b =$ unit vector along $B0C0$.

$\bar{u}_c =$ unit vector along $C0A0$.

For calculating the coordinates of the joint **A2**, it is required to know angle between planes (A0A1C1 & A2A1C1). Due to symmetry of the VGT geometry planes A0A1C1 and A2A1C1 make equal angle α with the plane A1B1C1. Since the coordinates of joints **A0**, **A1**, **B1**, and **C1** are known, the angle α can easily be found as,

$$\alpha = \cos^{-1}\left(\frac{\bar{n} \bullet \bar{n}_1}{|\bar{n}|\,|\bar{n}_1|}\right)$$

where,

$$\bar{n} = \frac{\mathbf{B1A1} \times \mathbf{C1B1}}{|\mathbf{B1A1} \times \mathbf{C1B1}|}$$, $\bar{n}$ is unit vector normal to plane A1B1C1

and,

$$\bar{n}_1 = \frac{\mathbf{A1A0} \times \mathbf{C1A1}}{|\mathbf{A1A0} \times \mathbf{C1A1}|}$$, $\bar{n}_1$ is unit vector normal to plane A0B1C1

Using value of angle α, the coordinates of joints **A2** can be found as

$$\mathbf{A2} = [R_{2\alpha,\, \bar{u}_{a1}}](\bar{a}_0 - \bar{a}_1) + \bar{a}_1$$

where,

$\bar{u}_{a1} =$ unit vector along $C1A1$.

Following the same analysis procedure angles β between planes B0B1A1 and A1B1C1, and γ between planes C0C1B1 and A1B1C1 are found as,

$$\beta = \cos^{-1}\left(\frac{\bar{n} \bullet \bar{n}_2}{|\bar{n}|\,|\bar{n}_2|}\right)$$

$$\gamma = \cos^{-1}\left(\frac{\bar{n} \bullet \bar{n}_3}{|\bar{n}|\,|\bar{n}_3|}\right)$$

where,

$\bar{n}_2 = \dfrac{\mathbf{B1B0} \times \mathbf{A1B1}}{|\mathbf{B1B0} \times \mathbf{A1B1}|}$, $\bar{n}_2$ is unit vector normal to plane B0B1A1.

and,

$\bar{n}_3 = \dfrac{\mathbf{C1C0} \times \mathbf{B1C1}}{|\mathbf{C1C0} \times \mathbf{B1C1}|}$, $\bar{n}_3$ is unit vector normal to plane C0C1B1.

Following the same procedure the as in calculation of **A2** the coordinates of joints **B2** and **C2** are found as

$$\mathbf{B2} = [R_{2\beta,\,\bar{u}_{b1}}](\bar{b}_0 - \bar{b}_1) + \bar{b}_1$$

$$\mathbf{C2} = [R_{2\gamma,\,\bar{u}_{c1}}](\bar{c}_0 - \bar{c}_1) + \bar{c}_1$$

where,

$\bar{u}_{b1}$ = unit vector along $B1A1$.

and,

$\bar{u}_{c1}$ = unit vector along $C1B1$.

This completes the forward kinematic solution for one module.

The forward kinematic solution for the second module is same as that of the first module. The second module is located on the top of the first module i.e. the base plane of the second module coincides with the top plane of the first module. The following method is used for calculating the coordinates of the second module in the global coordinate system {1}:

- The forward problem is solved for the second module in the local coordinate system {2} using the input data L_{12}, L_{22} and L_{32}.
- To calculate the coordinates of the second module in global coordinate system {1}, a transformation matrix $[T]_2^1$ is formed as given below.

First, unit vectors $\bar{u}_{a2}$, $\bar{u}_{b2}$, $\bar{u}_{c2}$ along X, Y and Z axes of coordinate system {2} are calculated as,

$$\bar{u}_{a2} = \frac{\mathbf{C2A2}}{|\mathbf{C2A2}|}$$

$$\bar{u}_{b2} = \frac{\mathbf{A2C2} \times \mathbf{B2A2}}{|\mathbf{A2C2} \times \mathbf{B2A2}|}$$

$$\bar{u}_{c2} = \bar{u}_{a2} \times \bar{u}_{b2}$$

These unit vectors are used to form the rotation matrix $[R]_2^1$.

$$[R]_2^1 = \begin{bmatrix} u_{a2x} & u_{b2x} & u_{c2x} \\ u_{a2y} & u_{b2y} & u_{c2y} \\ u_{a2z} & u_{b2z} & u_{c2z} \end{bmatrix}$$

Then the transformation vector locating the origin of the system {2} relative to system {1} is calculated as,

$$\bar{t} = (\bar{a}_2 - \bar{a}_0)$$

Combining the transformation vector and the rotation matrix and writing the result in homogeneous coordinate form gives the transformation matrix $[T]_2^1$ as

$$[T]_2^1 = \begin{bmatrix} u_{a2x} & u_{b2x} & u_{c2x} & t_x \\ u_{a2y} & u_{b2y} & u_{c2y} & t_y \\ u_{a2z} & u_{b2z} & u_{c2z} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using $[T]_2^1$ matrix the coordinates of a joint of the second module are calculated as,

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} = [T]_2^1 \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix}$$

This completely solves the forward problem for both the modules and defines coordinates of joints of both the modules in global coordinate system {1} .

It is envisioned that some VGT manipulators will have many more modules, as compared to two modules of the VGT under study. These more complex devices can be analyzed using the theory presented above.

The coordinates of the tip (point B) of the end effector (AB) can now be easily calculated as,

$$\text{Coordinates of the tip} = \frac{(\mathbf{A4} + \mathbf{B4} + \mathbf{C4})}{3} + L \times \bar{u}_{b4}$$

where,

L = length of the rod,

$\bar{u}_{b4}$ = is unit vector normal to plane A4B4C4.

2.2 *Inverse Kinematics Solution for NASA's Octahedral VGT*

In robotics, the inverse problem is defined as follows; *given the position and orientation of the end effector of the manipulator, calculate all possible sets of joint angles that could be used to attain this position and orientation.* For a typical robotic manipulator such as the one shown in the Fig. 3, the inverse problem is defined as finding the link parameters (θ_1, θ_2) for a given position and orientation of the end effector.

In the present case, for NASA's octahedral VGT, the inverse problem reduces to finding the variable link lengths (*L*1, *L*2, *L*3), for a given set of coordinates (X, Y, Z) of the tip (point B) of the beam. The input-data and the unknowns of the inverse problem for the VGT are:

Unknowns - L1, L2, L3

Input Data - X, Y, Z

The method used for solving the inverse kinematic problem for NASA's Octahedral VGT is outlined below.

- First, starting guesses are made for the variable link lengths L_1, L_2, and L_3.
- Using the values of the initial guesses for the lengths of the variable links, the forward problem is solved and the coordinates of the tip of the beam ($X1$, $Y1$, $Z1$) are calculated.
- These coordinates are then used to form the three functions F1, F2 and F3 :

$$F1(L1, L2, L3) = X - X1$$

$$F2(L1, L2, L3) = Y - Y1$$

$$F3(L1, L2, L3) = Z - Z1$$

where,

X, Y, Z are the desired coordinates of the tip of the beam (point B).

X1, Y1 and Z1 are the actual coordinates of the tip (point B), for the given values of L1, L2 and L3.

The values of the functions F1, F2 and F3 are compared with the specified tolerance limit (in this case 10^{-5}). If the function values are less than the tolerance limit then the solution has converged, and the corresponding values of the link lengths $L_1, L_2,$ and L_3 is the solution vector. If any of the three function values is greater than the tolerance limit, or if the number of iterations is less than specified maximum number of iterations, then corrections are made in the initial guesses as follows,

$$\begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix} + \begin{bmatrix} \Delta L_1 \\ \Delta L_2 \\ \Delta L_3 \end{bmatrix}$$

according to the function values and the forward program is solved again.

Note that the problem of finding corrections in the link lengths is very sensitive to the value of the initial guesses. For this reason the Newton-Raphson root finding method is not used. In the present thesis the Quasi-Newton iteration method (Appendix A) is used instead. This method is able to converge even when the initial guesses are far from the solution.

2.3 *Workspace of NASA's Octahedral VGT*

The workspace of a robotic manipulator can be defined as follows; *the volume of space that the end effector of the manipulator can reach.* Workspace can also be interpreted as the space in which a kinematic solution exists for the robotic manipulator. The workspace of a manipulator can be further divided into *Dextrous Workspace* and *Reachable Workspace* . Dextrous workspace means that the robot manipulator can reach the volume with all possible orientations of the end effector, whereas the reachable workspace is the volume of space which the end effector can reach with at least one orientation.

The VGT under consideration has only three degrees of freedom. This limits the scope of calculation of the workspace to *reachable workspace* . The method used in the calculation of workspace is described below.

As mentioned in the previous chapter, the range of the three variable links is between 39 inches to 51 inches. For generating the workspace, the three links were varied from one extreme value to the other extreme value in the following two ways:

- One link is fixed in length in the range and the other two links are varied in length from one extreme to the other extreme value and,

- Two links are fixed in length in the range and the third link is varied in length from one extreme to the other extreme.

The corresponding coordinates of the end effector are then used for generating the workspace. Figures 8, 9, 10 and 11 show plots of the workspace.

As can be seen from the plots, the workspace has eight apexes. These eight apexes correspond to the eight extreme positions of the link lengths as given in Table 1.

Table 1. Eight extreme values of link lengths

No.	L1 inches	L2 inches	L3 inches
1	39	39	39
2	39	39	51
3	39	51	51
4	51	51	51
5	51	51	39
6	51	39	39
7	51	39	51
8	39	51	39

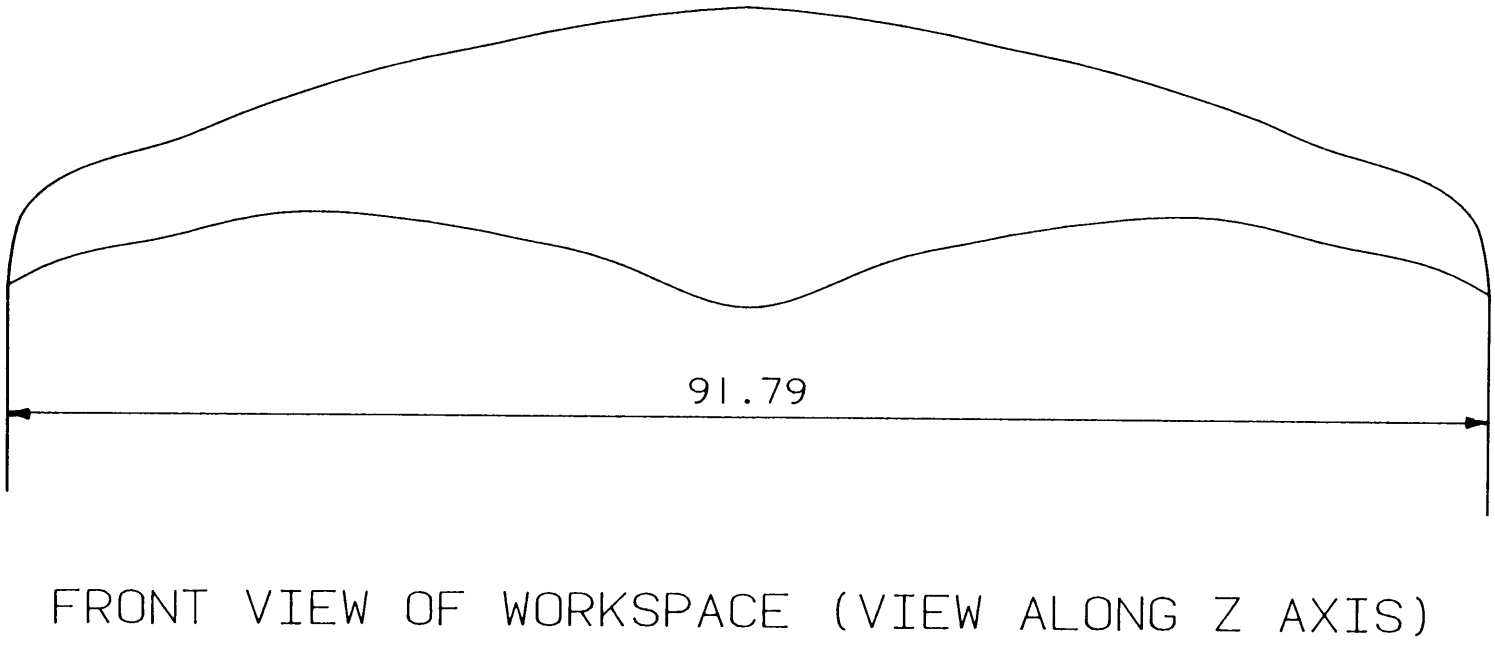


Figure 8. Front view of the workspace of the VGT.

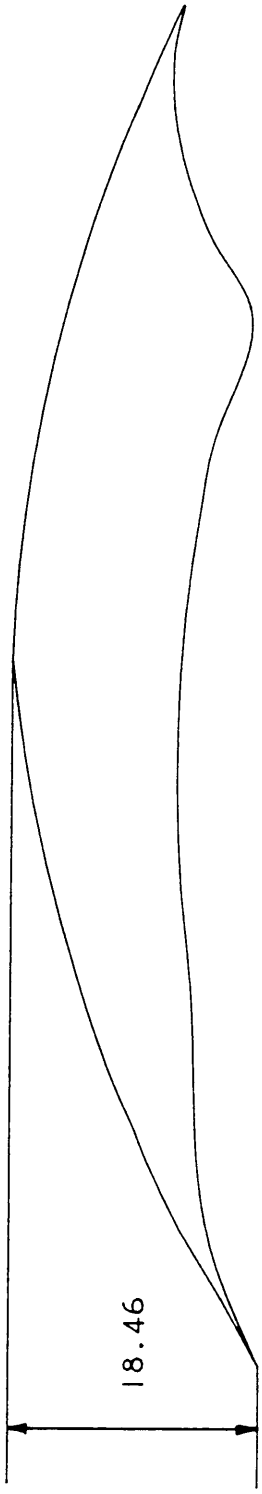


Figure 9. Side view of the workspace of the VGT.

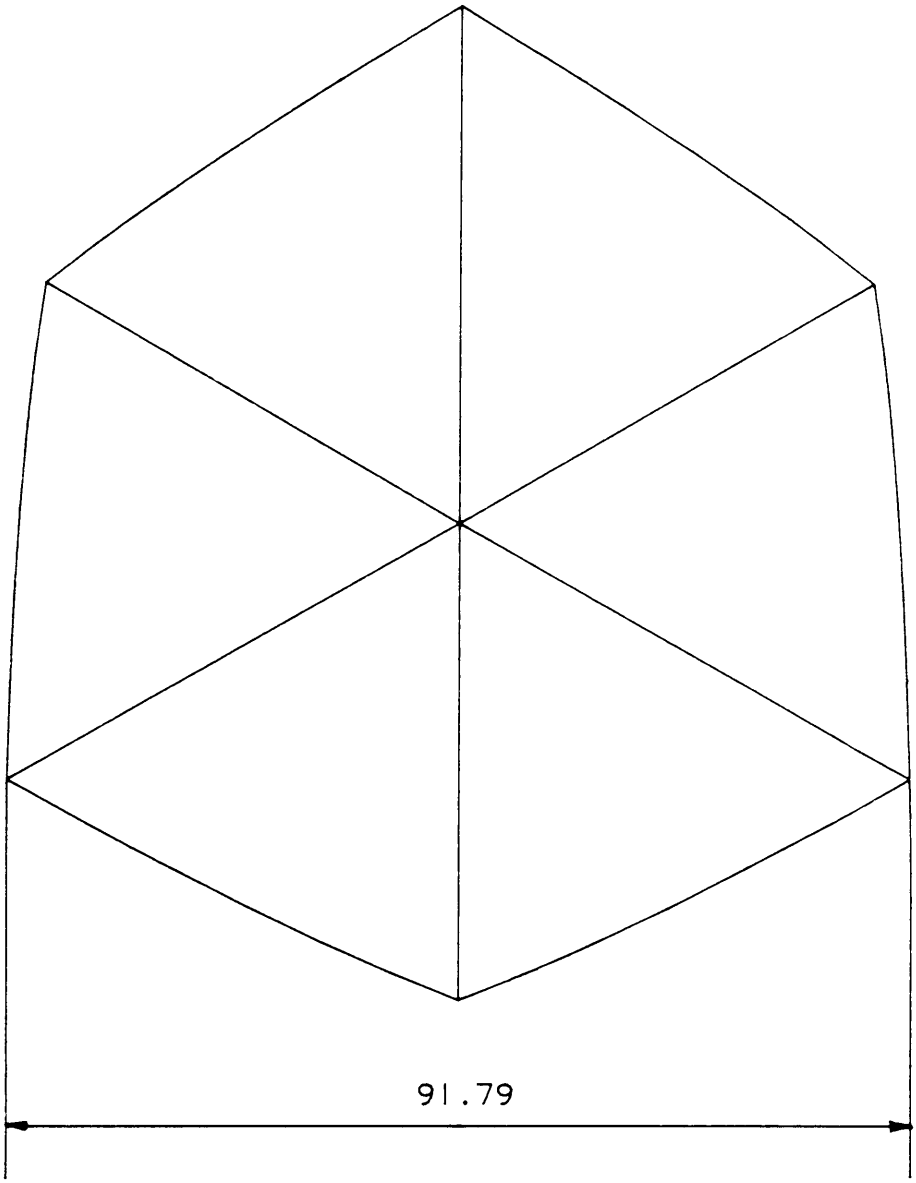


Figure 10. Top view of the workspace of the VGT.

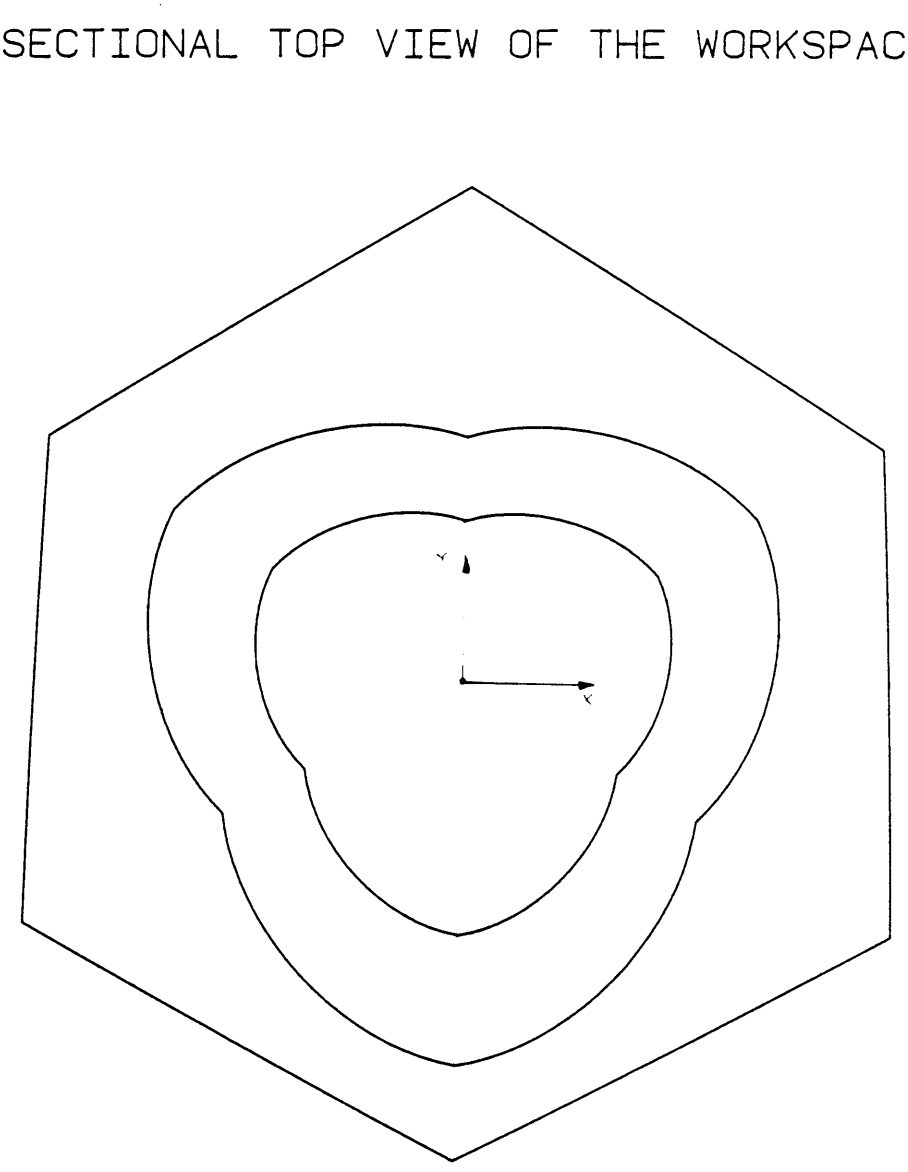


Figure 11. Sectional top view of the workspace of the VGT.

Chapter 3

Computer Implementation of Forward and Inverse Kinematic Solutions.

This Chapter discusses the computer implementation of the forward and inverse kinematic solutions developed in Chapter 2. The logic of the forward and inverse solutions is explained with the help of flowcharts and descriptions of main subroutines. The software is written in Fortran 77 and is run on a mainframe computer (IBM 4341-12). For the purpose of displaying the forward solution on computer screen, graphics software graPHIGS is used.

Implementation of the Forward Kinematic Solution

Figure 12 shows the flowchart of the forward kinematics program. This flowchart explains the forward kinematic solution for both modules of VGT. As shown in the flowchart, the computer solves the forward problem for one module and calculates

transformation matrix for transforming coordinates of the second module from the local coordinate system {2} to the global coordinate system {1}. The forward problem is then solved for the second module and its coordinates are transformed using the transformation matrix.

Figure 13 shows the flowchart of the forward kinematics program for one module of the NASA's Octahedral VGT. The functions of the main subroutines used in the program are explained below.

Subroutine VGTRUS defines coordinates of the base triangle. It then calls the NEWTON subroutine. Coordinates of joints of the module are calculated using the data obtained from the NEWTON subroutine. VGTRUS then passes data to the graPHIGS routines for drawing the VGT geometry on a IBM 5080 graphics terminal. The VGT geometry is drawn module by module i.e. as soon as coordinates of all the joints of a module are calculated that module is drawn.

Subroutine NEWTON solves the three functions FUN1, FUN2, and FUN3 described in Chapter 2, using the Newton-Raphson method. The solution vector is the three face angles θ_1, θ_2, and θ_3 . This output is then passed back to the VGTRUS subroutine for calculation of the joint coordinates.

Subroutines ANYRS, ANYRS1, and ANYRS2 are used to form and evaluate the three functions FUN1, FUN2 and FUN3. These subroutines also calculate partial derivatives of the functions with respect to the three variables θ_1, θ_2, and θ_3 using a finite difference method. The function values and the partial derivatives are then passed to the NEWTON routine for further computation.

Computer Animation of NASA's Octahedral VGT

Figure 14 shows flowchart of the logic used in animation of the VGT. For animation, graPHIGS software is used [22]. graPHIGS software is based on the

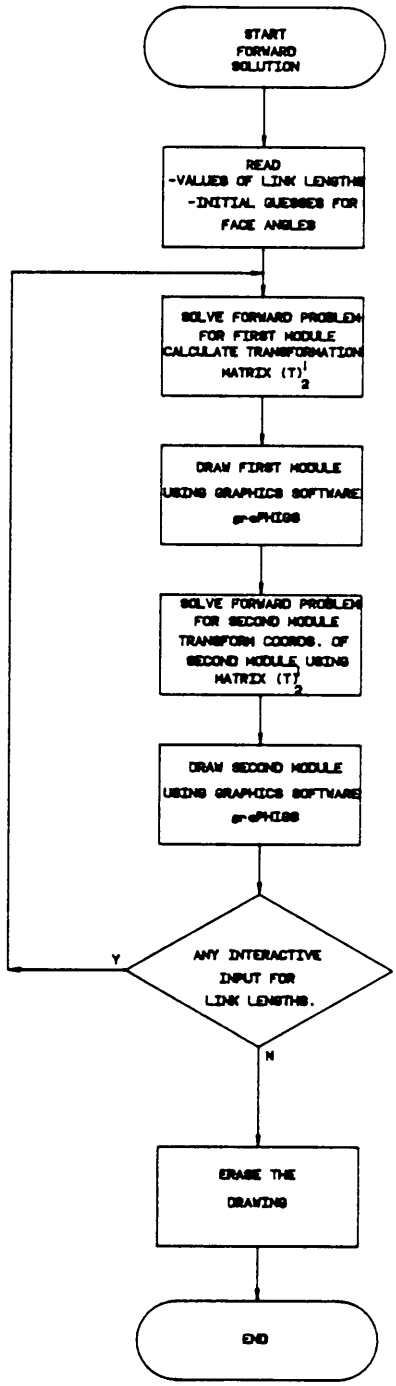


Figure 12. Flowchart of Forward solution for VGT.

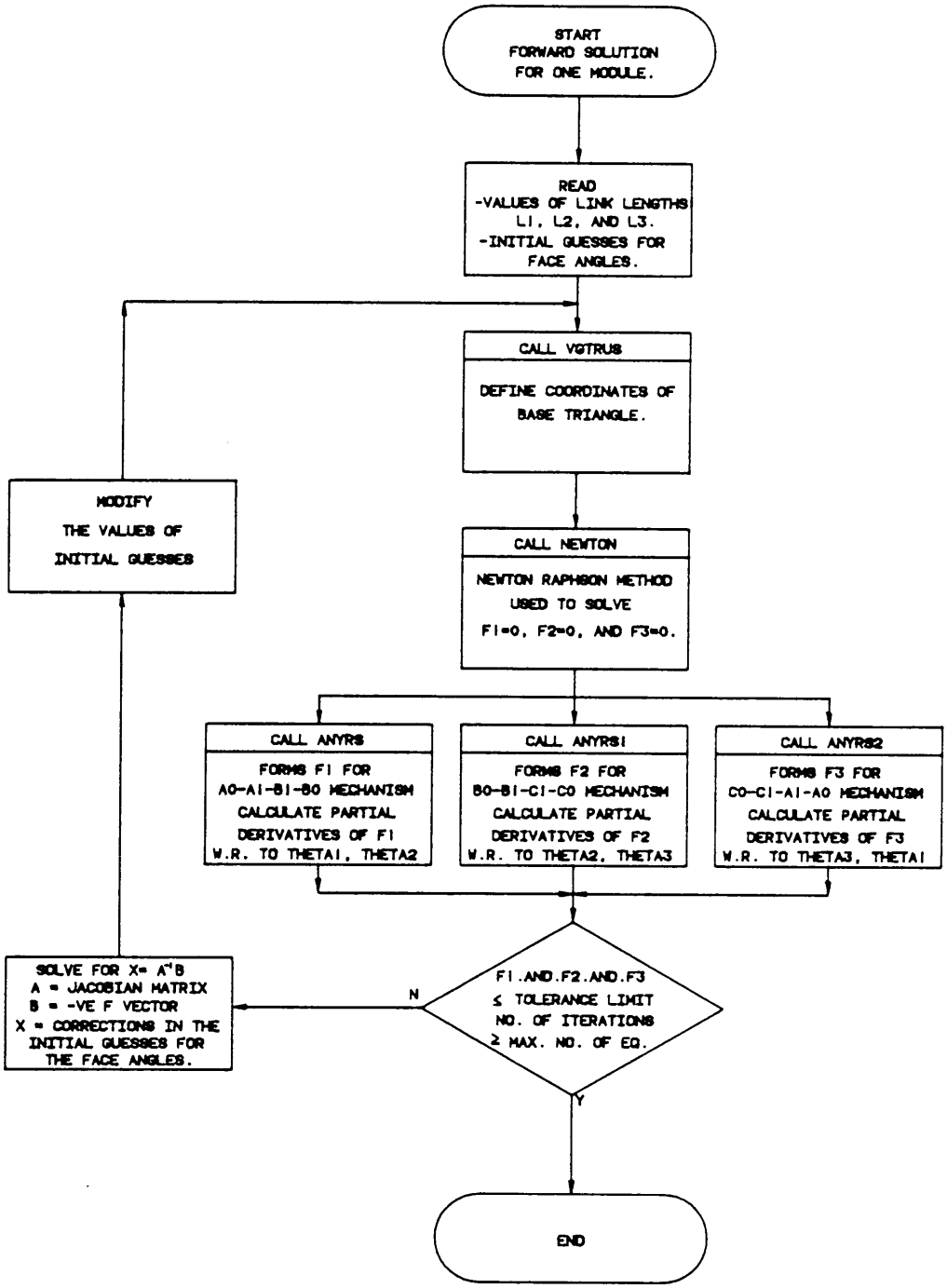


Figure 13. Flowchart of Forward solution for one module.

proposed 3-D graphics standard PHIGS. The main reasons in selection of this graphics software are:

- PHIGS is device independent.
- PHIGS is three dimensional graphics software. This aspect is very important to completely represent the three dimensional VGT geometry.
- PHIGS provides a high level of interactivity, thus it is easy to use interactive data input. The data is input using logical input devices.
- PHIGS can display and manipulate hierarchical data-structures and there is a dynamic relationship between the databases. This aspect is very important in animation of VGT.

The sequence used for animation is as follows:

First the VGT geometry is drawn for predefined link lengths. The program then waits for twenty seconds and checks for any graphical input. The graphical inputs used are, valuators to input the link lengths and choice to exit out of the animation sequence. If any graphical input is read within twenty seconds then the program immediately processes that input, else automatically shuts off any further interaction and stops displaying the VGT geometry. If the input is a changed value of link lengths, that set of link lengths is input to the forward program. The forward program calculates the new geometry for these link lengths, the original geometry is erased and the new geometry is displayed. This sequence is fast, and it gives the effect of animation of the VGT. If the input is choice to exit out of the animation sequence then the geometry is erased and the program is terminated.

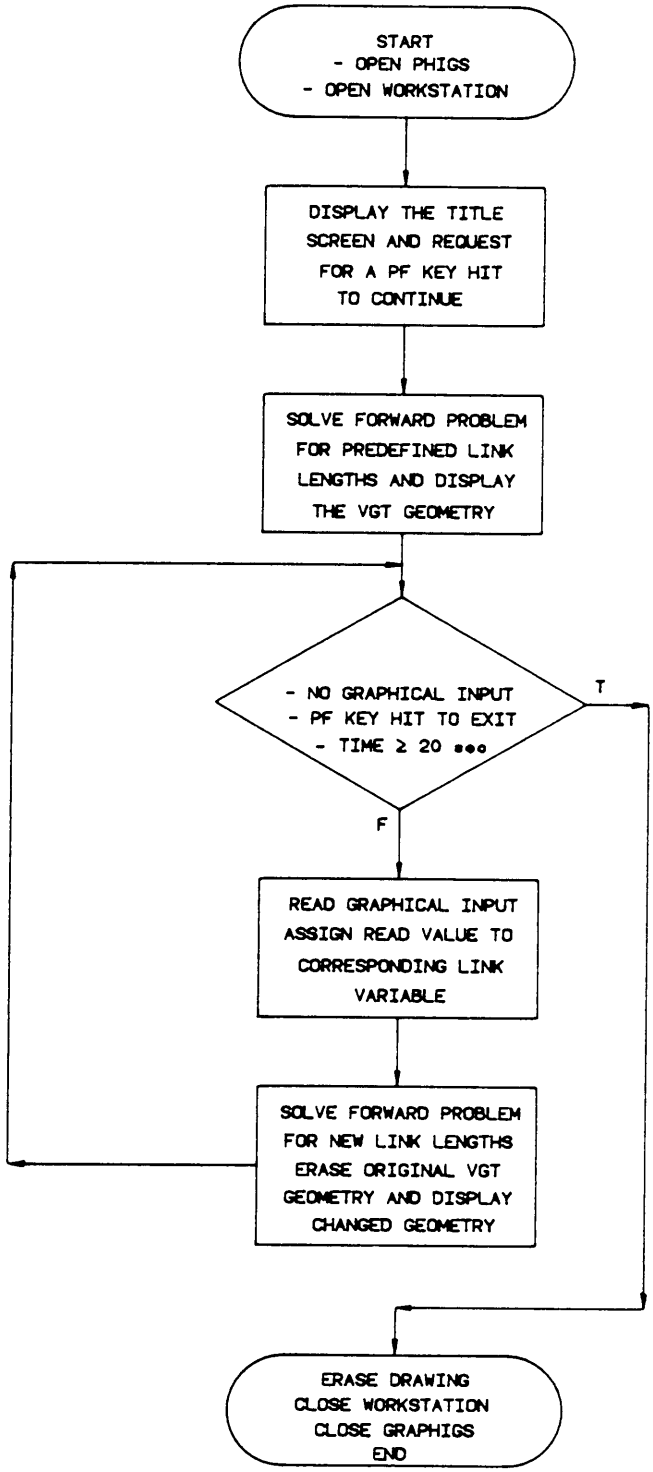


Figure 14. Flowchart for Animation sequence of VGT.

Implementation of the Inverse kinematic solution

Figure 15 shows flowchart of the inverse kinematic solution. As shown in the figure, the forward problem is first solved for initial guesses of the link lengths. Three functions F1, F2 and F3 are then formed. These functions are solved iteratively using the Quasi-Newton method. The solution of the inverse problem gives the link length vector. The inverse kinematic program can be used for interactive input, or input can be read from a data file.

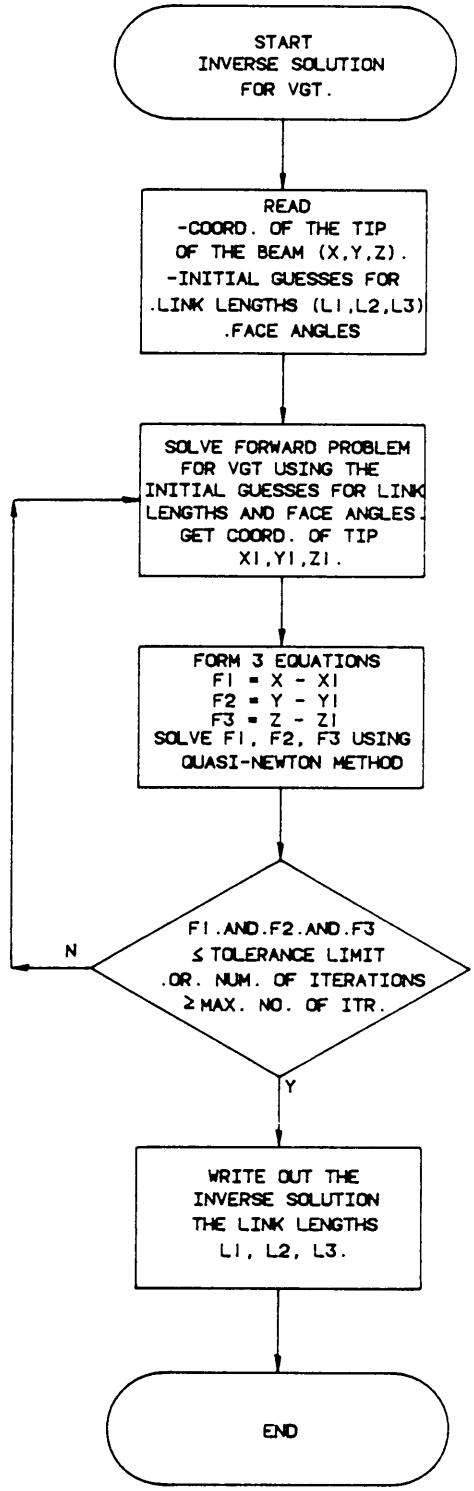


Figure 15. Flowchart of Inverse solution of VGT.

Chapter 4

Experimental Verification of VGT Kinematics

The forward and inverse kinematic solution results were tested on NASA's octahedral VGT. The experimental setup, the experiment and the results obtained are discussed in this chapter.

4.1 Experimental Setup

For the experiment, the VGT of Fig. 2, was turned upside down as shown in Fig. 16. In the experiment, the range of variable links was from 40 inches to 50 inches. The length of the hand-adjustable links was fixed at 46.5 inches. A bar of length 77.5 inches was fixed to the top plane of VGT to act as an end effector. A marking pen was fixed to the free end of the rod, so that the path taken by the end effector could be traced on paper.

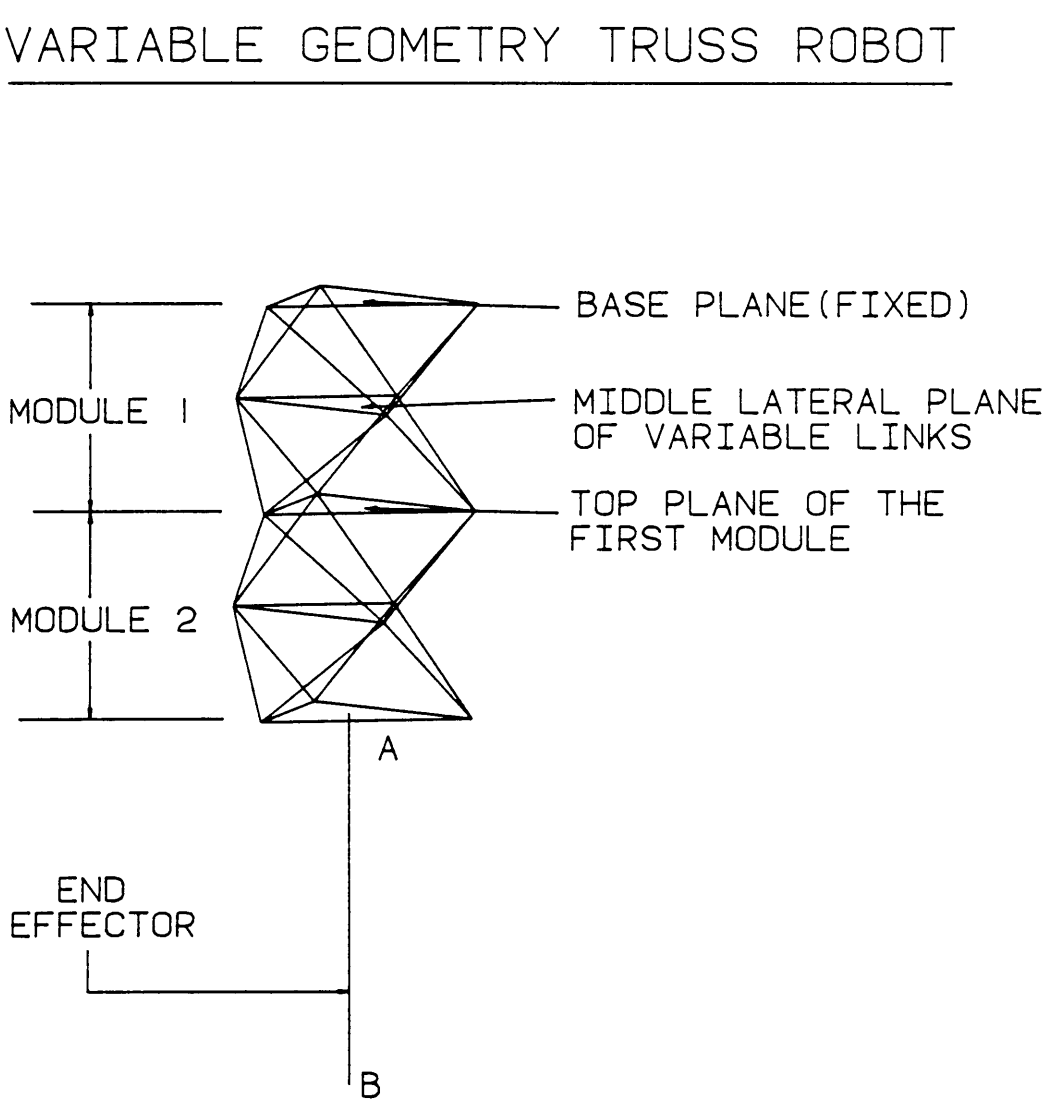


Figure 16. Arrangement of the VGT for the experiment.

Link actuator Mechanism

The length of the variable links is changed using leadscrews driven by small D.C. motors, associated with each of the link. Figure 17 shows the diagram of the actuator mechanism.

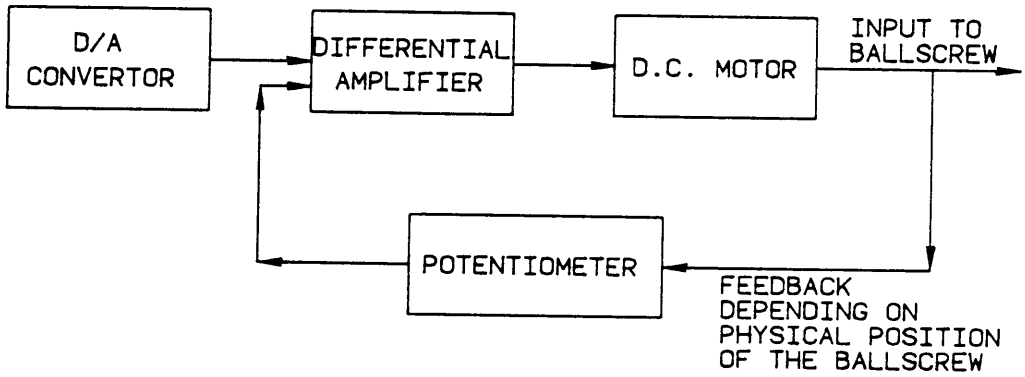


Figure 17. Link Actuator Mechanism

As shown in the figure, the motor is driven by the output of the differential amplifier. The input of the differential amplifier is the difference of voltage between the output of the D/A convertor and the voltage across the potentiometer. The voltage across each potentiometer is a function of length of the corresponding variable link.

4.2 *Experiment*

The aim of the experiment carried out was: *to demonstrate effective and accurate control of the end effector through the forward and inverse kinematic solution.* For this purpose it was decided that a set of letters of known, preselected dimensions be drawn with the marking pen attached at the end of the rod. The letters NASA were selected for this purpose. The letters NA and SA were required to be written on two different planes separated by a small step in the vertical direction. This was done to demonstrate that all the three degrees of freedom of the end of the rod could be controlled. These letters were first drawn on the computer using CADAM software. The height of the letters was fixed at five inches. These letters were then divided into points. Coordinates of these points were the calculated in the global coordinate system which was located at the base of the first module. These coordinates were then used as input to the inverse kinematics program (Appendix C). Solution of the inverse problem gave link lengths corresponding to each digitized point.

The values of the link lengths were then input to a computer program that converts each link length value into a corresponding voltage value. These digital voltage values were then converted into analog voltages using a digital to analog (D/A) convertor. The analog voltages were then used as input to a differential amplifier, which compares the voltages across the potentiometer and the output of the D/A convertor. The output of the differential amplifier then turns the motor, which in turn rotates the leadscrew associated with it. Thus the link lengths were changed and the end effector assumed the specified position in the workspace.

4.3 Results

Figure 18 shows the photograph of the NASA letters drawn by the marking pen attached to end effector. It was found that the dimensions of the letters used as input to the program and the dimensions of the output, the letters drawn on paper, matched with each other accurately.

For the purpose of calculating the numerical value of accuracy, one more experiment was carried out. In this case a simple planar geometry, a circle, was drawn following the same procedure as in the experiment described above. In this case the circle perimeter was divided into 360 points.

Table 2 lists the inaccuracy in the radius of the circle at twelve points. Note that the accuracy calculations are done in the local coordinate system which is located at the center of the circle.

Table 2. Inaccuracy in Forward and Inverse Kinematic Solutions.

Angle from X Axis	Inaccuracy in %
0	3.125
30	4.5
60	4.75
90	3.125
120	4.75
150	4.375
180	3.25
210	2.0
240	2.5
270	3.125
300	2.5
330	1.25

As can be seen from the table above the forward and inverse kinematic solutions are accurate up to 95.25%.

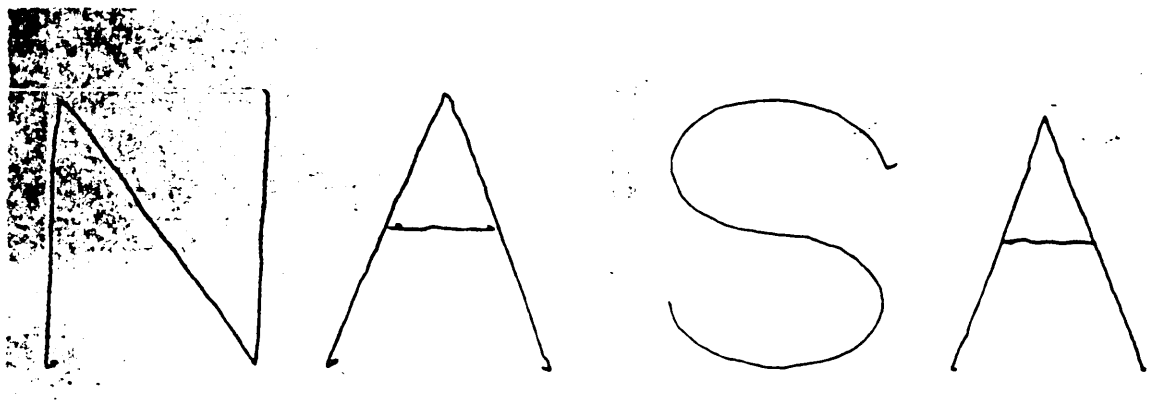


Figure 18. Letters NASA drawn by the end effector.

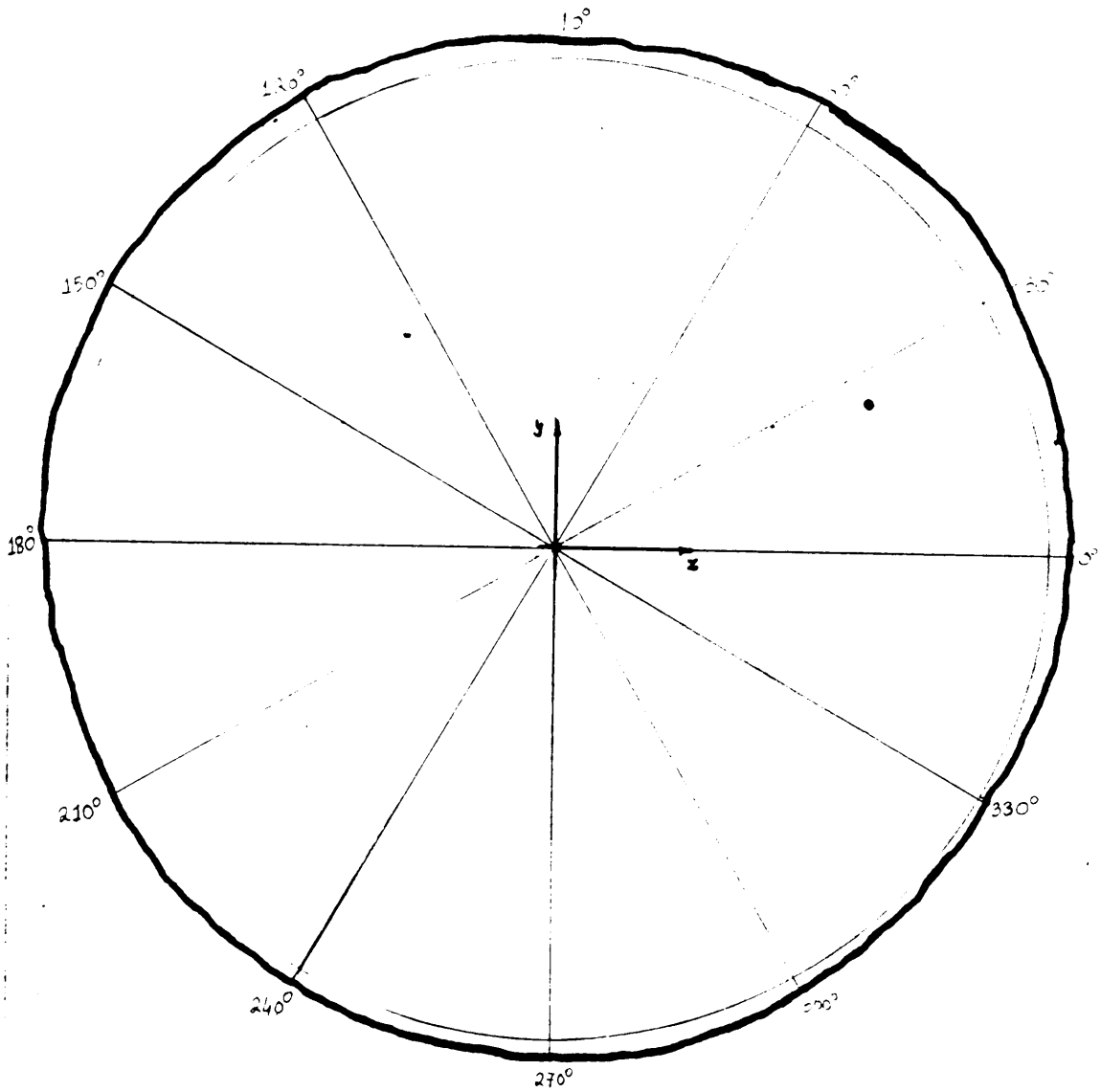


Figure 19. Circle geometry used in accuracy calculation.

Chapter 5

Conclusions and Recommendations

The main purpose of this thesis was to study the feasibility of using the NASA's Octahedral VGT at Virginia Tech as a robotic manipulator. For this purpose kinematic equations were developed, and the workspace of the VGT was computed. An experiment was run on the VGT to check the accuracy of the results of the kinematic solution. Following are the conclusions drawn from the results.

5.1 Conclusions

1) From the results obtained, it can be seen that the forward and inverse solutions are accurate up to 95.25%. The possible sources of inaccuracy are,

- Clearances in the joints of the truss.

- Digital to Analog voltage conversion process.
- Approximation of motion by the actuator mechanism between two digitized points.
- Vibrations of the end effector and deflection at the support of the stiff rod.

2) It is obvious from the shape and dimensions of the workspace that, although the length and width of the workspace of the VGT are large, the depth of the workspace is small. For the VGT to be used as a robotic manipulator, the range of the variable links must be increased substantially so that the depth of the workspace which is a function of the range of the variable links will also increase proportionately.
Note that the range of length of the variable links is limited mainly because,

- The VGT has to accommodate the leadscrew and the actuator mechanism that controls the variable links. This puts a lower limit on the link lengths, as the lengths cannot be reduced below the actual physical dimension of the actuators.
- As the link lengths increase, the face angles α, β and γ start decreasing. But the angles cannot become negative as it would cause interference between the structural members. Thus the lower values of the face angles put higher limits on the link lengths.

5.2 Recommendations

1) In the present case, out of the six variable links only three are actuated using motor driven leadscrews. This limits the scope of the kinematic solution. Also it limits the calculation of the workspace of the VGT to the reachable workspace only. In future,

the three variable links of the second module will also be actuated. This will substantially increase potential of the VGT for its likely use as a robotic manipulator.

2) For the VGT under study, in a module only three links are variable in length. This means that a module has only three degrees of freedom. If other links, such as the links of the top lateral plane are made variable in length, the mobility of a module will increase from three to six. Besides the links of the lateral, horizontal triangle being variable in length, the links of other triangles can also be actuated as in the case of the Stewart platform.

Note that as the number of variable links increases, dexterity of the truss increases. But this also complicates the problem of control. Effective and accurate control in such cases is a challenging task.

References

1. Rockwell International, "Development of Deployable Structures for Large Space Platform Systems", *Interim Report Volume 1. SSD 82-0121-1 (Contract NAS8-34677)* August 1982.

2. Cox, R. L., and Nelson, R. A., "Development of Deployable Structures for Large Space Platform Systems," *Rep. NO. 2-32300/2R-53215(Contract NAS8-34678),* Vought Corp., October 1982.

3. Miura, K., and Furuya, H., "An Adaptive Structure Concept For Future Space Applications", IAF-85-211 *Proceedings of the 36th congress of the International* Astronautical Federation, Pergamon Press, 1985.

4. Rhodes, M. D., and Mikulas Jr., M. M., "Deployable Controllable Geometry Truss Beam", *NASA Technical Memorandum 86366,* June 1985.

5. Hunt, K. H., **Kinematic Geometry of Mechanisms,** Oxford University Press, London, 1978.

6. Fitcher, E. F., and Mcdowell, E. D., *A Novel Design for Robot Arm,* Advances in Computer Technology, an ASME publication, pp. 250- 256.

7. Yang, D.C.H., and Lee, T. W., *Feasibility Study of A Platform Type of Robotic Manipulator from a Kinematic Viewpoint,* Trans. ASME, Journal of Mechanisms, Transmissions and Automation in Design, Vol. 106, No. 2, June 1984, pp. 191-198.

8. Fitcher, E. F., *Kinematics of a Parallel Connection Manipulator,* ASME paper No. 84-DET-45, 1984.

9. Hunt, K. H., *Structural Kinematics of In-parallel Actuated Robot Arms,* ASME paper 82-DET-105, 1982.

10. Earl, C. F., and Rooney, J., *Some Kinematic Structures for Robot Manipulator Designs,* Trans. ASME, Journal of Mechanisms , Transmissions and Automation in Design, Vol. 105, 1983 pp. 15-22.

11. Ardayfio, D. D., and Qiao, D., *Kinematic Simulation of Novel Robotic Mechanisms having Closed Kinematic Chains,* ASME paper 85-DET-81.

12. Sincarsin, W. G., and Hughes, P. C.: "Trussarm candidate Geometries", Dynacon Report 28-611/0401, Dynacon Enterprises Limited, 5050 Dufferin Street, Suite 200, Downsview, Ontario, Canada, 1987.

13. Miura, Koryo., Furuya, H., and Suzuki, K.: *Variable Geometry Truss and Its Application to Deployable Truss and Space Crane Arm,* 'Acta Astronautica,' 1985, VOL.12, NO. 7-8.(IAF-84-394, Lausanne, Oct. 1984.)

14. Miura, K., *Design and Operation of a Deployable Truss Structure* , Paper No. 4, NASA CP-2311, 18th Aerospace Mechanisms Symposium, NASA Goddard Space Flight Center, Greenbelt, Maryland, (2-4 May 1984).

15. Mabie, H. H., and Reinholtz, C. F.: **Mechanisms and Dynamics of Machinery** , 4th Ed., John Wiley and Sons NY, 1987.

16. Suh, C. H., and Radcliffe, C. W., **Kinematics and Mechanisms Design** , Robert E. Krieger Publishing Company, Malabar, Florida, Reprint Ed. 1983.

17. Craig, J. J., **Introduction to Robotics, Mechanics and Control,** Addison-Wesley Publishing Co. (1986).

18. Paul, R. P., **Robot Manipulators,** MIT Press, Cambridge (USA) and London (1983).

19. Johnson, L. W., and Riess, R. D., **Numerical Analysis** Adddison-Wesley Publishing company (1982).

20. IBM programmer's Reference for graPHIGS, SC33-8104-0, Program No. 5668- 792.

21. IBM Messages and Codes for graPHIGS, SC33-8105-0, Program No. 5668-792.

22. IBM Understanding graPHIGS, SC33-8102-0, Program No. 5668-792.

23. IBM Writing Applications for graPHIGS, SC33-8103-0, Program No. 5668-792

Appendix A
Quasi-Newton Method

In this appendix the Quasi-Newton method used in the solution of the inverse problem is discussed.

Given data :-

1) A set of nonlinear functions F(U) as

$$F(U) = \begin{bmatrix} F1(U) \\ F2(U) \\ \cdots \\ \cdots \\ \cdots \\ FN(U) \end{bmatrix} = \begin{bmatrix} F1(x1, x2, ..., xn) \\ F2(x1, x2, ..., xn) \\ ..... \\ ..... \\ ..... \\ FN(x1, x2, ..., xn) \end{bmatrix}$$

2) Initial Guesses as U0 = $(x1, x2, ..., xn)^T$

Goal :-

To find solution vector S, such that F(S) = θ , where θ is a null matrix.

Solution Method :-

a) Calculate F(U0) using initial guess (U0) for the unknowns.

b) Check F(U0) against the specified tolerance limit. If F(U0) is less than the specified tolerance limit, then it means the solution has been reached and the corresponding values of the link lengths are the solution for the given input coordinates.

If this criteria is not satisfied then proceed as follows,

1) Calculate a descent direction p_k using the following equation,

$$p_k = -J(U0)^T F(U0)$$

2) Calculate λ_k such that the following equation is satisfied using this value

$$\phi(U0 + \lambda_k p_k) < \phi(U0) + 10^{-4}\lambda_k \nabla\phi(U0)^T p_k$$

where,

$$\phi(U0) = \frac{||F(U0)||^2}{2}$$

Using the two values p_k, and λ_k calculated above, correction is made in the initial guess vector as $U0 = U0 + \lambda_k p_k$.

Then compare the number of iterations with the maximum number of iterations. If the number of iterations is greater than maximum number of iterations then stop the iterations. This means that the solution is not possible. If the number of iterations is less than the maximum number of iterations then again repeat the steps (a) & (b) using the corrected values of initial guesses.

Note that, the Quasi-Newton method differs from the Newton-Raphson method in two ways

1. It finds the descent direction, i.e. decides the direction in which to proceed for finding the roots and,

2. It calculates the amount of "step" to be taken in the descent direction.

More detailed description of the method can be found in a book by Johnson and Riess [19].

Appendix B

Forward Program Listing

```
C THIS PROGRAM SOLVES THE FORWARD KINEMATICS PROBLEM FOR THE OCTAHEDRAL
C VGT AT VIRGINIA TECH ON LOAN FROM NASA'S LANGLEY RESEARCH CENTER.
C THE COORDINATE DATA WHICH IS THE OUTPUT OF THE PROGRAM IS USED IN
C ANIMATION OF THE VGT. FOR ANIMATION A THREE DIMENSIONAL GRAPHICS
C SOFTWARE, PHIGS IS USED.
C PROGRAMMER : D. P. GOKHALE.
C S.S. NO. 225 39 4388.


C DECLARATION OF TYPE OF VARIABLES AND CONSTANTS.

      INTEGER IWSID, ISTART, INUM
      INTEGER DEVID, STATUS, CHOICE, BREAK
      INTEGER BLACK, WHITE, RED, GREEN, BLUE, GRAY, MAGENT, YELLOW, CYAN
      INTEGER PRFORM, HIGHER, NOECHO, ECHO, CLIP, OFF, ON
      INTEGER VIEW0, VIEW1, VIEW2, VIEW3
      INTEGER VALUT, EVENT, REQ
      INTEGER VGTR
      INTEGER TYPE
      INTEGER INFLAG, NDEV, DEV(1), UNITS
      INTEGER ASIZE(3)
      INTEGER I, J, K, L, M, N
      INTEGER LNGT
      INTEGER NUM, IKFLAG

      REAL POS2(2)
      REAL PRP(3), DIST, NEAR, FAR
      REAL WINDO1(6), WINDO2(6)
      REAL VPR1(6), VPR2(6), VPR3(6)
      REAL COLORS(27)
      REAL CSIZE(3), AREA1(6)
      REAL*8 AL1(10), AL2(10), AL3(10)
      REAL*4 AL11, AL12, AL13, AL14, AL15, AL16
      REAL*8 ALPHA(10), BETA(10), GAMMA(10)
      REAL*8 AX(3), BX(3), CX(3)
      REAL XYZ1(3,4)

      CHARACTER*8 ERFILE, WSTYPE, CONNID
      CHARACTER*41 MSG1
      CHARACTER*41 MSG
      CHARACTER*15 TXT, TXT1, TXT2, TXT3, TXT4, TXT5

C
C THE FOLLOWING VARIABLES ARE USED IN THE INPUT DEVICE INITIALIZATION.
C
      INTEGER ICHOI, ECHO1, ECHO2, DATAL, DATA1(35)
      INTEGER VECHO, VDATAL
      INTEGER PPATH(3), ACLASS(1)

      REAL IPOS(3)
      REAL IVAL1, IVAL2, IVAL3, IVAL4, IVAL5, IVAL6, IVAL7, IVAL8
      REAL PAREA(6)

      CHARACTER*16 VADAT1

C
C VALUES USED TO MANAGE THE EVENT INPUT.
C
      INTEGER CLASS,DEVICE


C
C THIS WRITE STATEMENT IS USED FOR ECHO TYPE FOUR, USED IN THE VALUATOR
C INPUT. THE DATA IS WRITTEN IN A FILE IN UNFORMATTED FORM SO IT IS
C AUTOMATICALLY CONVERTED INTO BINARY FORM. IT IS THEN READ IN THE
C BINARY FORM INTO THE VADAT1 VARIABLE. WITH THE HELP OF THE VALUATOR
C TYPE FOUR WE CAN CORRELATE THE REVOLUTIONS OF THE DIAL AND THE VALUE
```

```
C CHANGE.
C
      WRITE(12)1,0,0,1
      REWIND(12)
      READ(12)VADAT1


C---------------------------------------------------------------------
C
C DATA DECLARATION AND DECLARATION OF CONSTANTS.
C
C---------------------------------------------------------------------

C DEFINING THE ERROR FILE.
      DATA ERFILE/'SYSPRINT'/

C DEFINING THE WORKSTATION TYPE.
      DATA WSTYPE/'5080    '/

C DEFINING THE CONNECTION IDENTIFIER.
      DATA CONNID/'IBM5080 '/

C DEFINING THE WORKSTATION IDENTIFIER.
      DATA IWSID/1/

C DEFINING THE START INDEX FOR THE COLOR TABLE.
      DATA ISTART/0/

C DEFINING THE NUMBER OF ENTRIES TO LOAD INTO THE COLOR TABLE.
      DATA INUM/9/

C DEFINING THE ARRAY USED TO LOAD THE COLOR TABLE.

C                BLACK - THE BACKGROUND COLOR.
      DATA COLORS/ 0.0, 0.0, 0.0,

C                     WHITE
     &             1.0, 1.0, 1.0,

C                     RED
     &             1.0, 0.0, 0.0,

C                     GREEN
     &             0.0, 1.0, 0.0,

C                     BLUE
     &             0.0, 0.0, 1.0,

C                     GRAY
     &             0.35, 0.35, 0.35,

C                     MAGENTA
     &             1.0, 0.0, 1.0,

C                     YELLOW
     &             1.0, 1.0, 0.0,

C                     CYAN
     &             0.0, 1.0, 1.0/

C COLOR TABLE DATA DECLARATION.
      DATA BLACK,WHITE,RED,GREEN,BLUE,GRAY,MAGENT,YELLOW,CYAN/
     >        0,   1,   2,   3,    4,   5,   6,      7,   8 /

C DEFINING THE HIGHER PRIORITY FLAG.
      DATA HIGHER/1/
```

```
C DEFINING THE NOECHO SWITCH.
      DATA NOECHO/1/

C DEFINING THE ECHO SWITCH.
      DATA ECHO/2/

C DEFINING THE THE CLIP ON VALUE.
      DATA CLIP/2/

C DEFINING THE ACTIVE OFF VALUE.
      DATA OFF/1/

C DEFINING THE ACTIVE ON VALUE.
      DATA ON/2/

C DEFINING THE VIEW ID.
      DATA VIEW1, VIEW2, VIEW3/1, 2, 3/

C DEFINING THE GRAPHICAL MODES.
      DATA EVENT, REQ/3, 1/

C DEFINING THE PERFORM UPDATE FLAG.
      DATA PRFORM/2/

C DEFINING THE STATUS VALUE FOR A BREAK ON CHOICE.
      DATA BREAK/1/

C DEFINING THE PROJECTION TYPE, IN THIS CASE PARALLEL PROJECTION TYPE IS
C SELECTED.
      DATA TYPE/1/

C DEFINING THE PROJECTION REFERENCE POINT.
      DATA PRP/3.0, 0.0, 3.0/

C DEFINING THE VIEW PLANE DISTANCE.
      DATA DIST/50/

C DEFINING THE EXTENT OF THE NEAR AND THE FAR CLIPPING PLANES.
      DATA NEAR, FAR/100.0, -100.0/

C DEFINING THE THE VALUATOR DEVICE ECHO DATA.
      DATA VDATAL/16/

C THE FOLLOWING DATA IS USED FOR INPUT DEVICE INITIALIZATION.
      DATA ICHOI, ECHO1, ECHO2/ 1, 1, 2/
      DATA DATAL/140/
      DATA DATA1/32, 0, 0,     1,1,1,1,
     &                       1,1,1,1,1,1,1,
     &                       1,1,1,1,1,1,1,
     &                       1,1,1,1,1,1,1,
     &                       1,1,1,1,1,1,1,
     &                         1,1,1,2/

C WINDOW AND VIEWPORT EXTENT INFORMATION.
      DATA WINDO1/-150., 150., -150., 150., -150., 150./
      DATA WINDO2/-200., 200., -200., 200., -200., 200./
      DATA VPR1/0.0, 1.0, 0.0, 1.0, 0.0, 1.0/
      DATA VPR2/0.0, 1.0, 0.0, 1.0, 0.0, 1.0/

C THE FOLLOWING DATA DECLARES THE CHARACTER STRINGS TO BE DISPLAYED
C ON THE GRAPHICS SCREEN.
      DATA LNGT/41/
      DATA MSG/'PRESS LIGHTED KEY TO EXIT'/
      DATA MSG1/'PRESS LIGHTED KEY TO CONTINUE'/
      DATA TXT/'ANIMATION'/
      DATA TXT1/'OF'/
      DATA TXT2/'A'/
```

```
      DATA TXT3/'VARIABLE'/
      DATA TXT4/'GEOMETRY'/
      DATA TXT5/'ROBOT'/


C-------------------------------------------------------------------------
C THE FOLLOWING BLOCK DEFINES THE VARIABLES FOR THE VARIABLE GEOMETRY
C TRUSS.
C-------------------------------------------------------------------------

C DEFINING THE TOTAL NUMBER OF MODULES OF THE V.G.T.
      NUM = 2

C DEFINING THE FLAG OPTION. THIS FLAG IS USED IN THE VGTRUS SUBROUTINE.
      IKFLAG = 0

C DEFINING THE INITIAL LENGTH OF THE VARIABLE LINKS.
      DATA AL11,AL12,AL13/44.5,44.5,44.5/
      DATA AL14,AL15,AL16/46.5,46.5,46.5/
      DO 25 I = 1, NUM
      AL1(I) = 39.01
      AL2(I) = 39.01
      AL3(I) = 39.01
25    CONTINUE

      AL1(2) = 46.5
      AL2(2) = 46.5
      AL3(2) = 46.5

C THE STARTING VALUES FOR ANGLES THETA-A, THETA-B, THETA-C.
      DO 15 I = 1, NUM
      ALPHA(I) = 0.9
      BETA(I) = 0.9
      GAMMA(I) = 0.9
15    CONTINUE

C DEFINING THE INITIAL VALUE FOR THE VALUATOR.
      IVAL1 = 32.0

C
C-------------------------------------------------------------------------
C OPENING AND INITIALIZATION OF PHIGS.
C-------------------------------------------------------------------------
C

C CALL TO OPEN GRAPHICS.
      CALL GPOPPH(ERFILE, 0)

C CALL TO OPEN A 5080 GRAPHICS WORKSTATION.
      CALL GPOPWS(IWSID, CONNID, WSTYPE)

C TO OBTAIN THE ACTUAL DISPLAY SURFACE SIZE OF THE WORKSTATION WHICH IS
C USED IN CALCULATION OF THE ECHO AREA.
      CALL GPQADS(IWSID, ERRIND, 1, CSIZE, ASIZE)
      IF(ERRIND.NE.0)GOTO 10

C THE ECHO AREA IS DEFINED IN THE FOLLOWING BLOCK.
      AREA1(1) = 0.0
      AREA1(2) = CSIZE(1)
      AREA1(3) = 0.95 * CSIZE(2)
      AREA1(4) = CSIZE(2)
      AREA1(5) = 0.0
      AREA1(6) = CSIZE(3)

C CALL TO SET COLOR MODEL TO RGB.
      CALL GPCML(IWSID, 1)
```

```
C LOADING THE COLOR TABLE INTO THE WORKSTATIONS MEMORY.
      CALL GPCR(IWSID, ISTART, INUM, COLORS)

C ACTIVATE THE FONT FOR USE IN THE TITLE. FONT NUMBER 11 IS USED.
      CALL GPACFO(IWSID, 1, 11)

C---------------------------------------------------------------------
C
C THE FOLLOWING BLOCK DISPLAYS THE INITIAL TITLE PAGE.
C
C---------------------------------------------------------------------

C CALL TO OPEN STRUCTURE NUMBER 1.
      CALL GPOPST(1)

C CALLS TO SET THE CHARACTERISTICS OF THE TEXT TO BE DISPLAYED.
      CALL GPTXPR(1)
      CALL GPCHH(16.)
      CALL GPTXFO(11)
      CALL GPTXCI(WHITE)

C CALLS TO SET THE CHARACTERISTICS OF THE BACKGROUND AREA.
      CALL GPIS(2)

C THE FOLLOWING DATA DEFINES THE EXTENT OF THE SHADOW AREA.
      DATA XYZ1/-125., -125., 0., 115., -125., 0., 115.,
     &          115.,  0.,  -125.,  115.,  0./

C TO SET THE COLOR OF THE SHADOW.
      CALL GPICI(GRAY)
      CALL GPPG3(1,4,3,XYZ1)

C TO DEFINE THE EXTENT OF THE BACKGROUND AREA.
      XYZ1(1,1) = -120.
      XYZ1(2,1) = -120.
      XYZ1(3,1) =  0.0
      XYZ1(1,2) =  120.
      XYZ1(2,2) = -120.
      XYZ1(3,2) =  0.0
      XYZ1(1,3) =  120.
      XYZ1(2,3) =  120.
      XYZ1(3,3) =  0.0
      XYZ1(1,4) = -120.
      XYZ1(2,4) =  120.
      XYZ1(3,4) =  0.0

C TO SET THE COLOR OF THE BACKGROUND AREA.
      CALL GPICI(BLUE)
      CALL GPPG3(1,4,3,XYZ1)

C CALLS TO WRITE OUT THE TEXT OF THE TITLE.
      POS2(1) = -78.
      POS2(2) =  73.
      CALL GPTX2(POS2, 15, TXT)

      POS2(1) = -20.
      POS2(2) =  40.
      CALL GPTX2(POS2, 15, TXT1)

      POS2(1) = -11.
      POS2(2) = 10.
      CALL GPTX2(POS2, 15, TXT2)

      POS2(1) = -68.
      POS2(2) = -20.
      CALL GPTX2(POS2, 15, TXT3)
```

```
      POS2(1) = -68.
      POS2(2) = -50.
      CALL GPTX2(POS2, 15, TXT4)

      POS2(1) = -41.
      POS2(2) = -80.
      CALL GPTX2(POS2, 10, TXT5)

C CALL TO CLOSE THE TITLE DISPLAY STRUCTURE.
      CALL GPCLST

C THIS BLOCK ASSOCIATES THE TRUSS STRUCTURE WITH VIEW1.
      CALL GPARV(IWSID,VIEW1,1,1.0)
      CALL GPVMP3(IWSID,VIEW1,WINDO1,VPR1,TYPE,PRP,DIST,NEAR,FAR)
      CALL GPVCH(IWSID,VIEW1,CLIP,CLIP,CLIP,OFF,0,ON,WHITE,ON)
      CALL GPUPWS(IWSID,PRFORM)


C REQUEST A PF KEY HIT FOR CONTINUATION OF THE PROGRAM.
      CALL GPMSG(IWSID, 41, MSG1)
      CALL GPRQCH(IWSID,1,STATUS,CHOICE)
      CALL GPMSG(IWSID, 0, MSG1)
      IF(STATUS.EQ.BREAK)GOTO 10
      CALL GPDRV(IWSID,VIEW1,1)
      CALL GPEST(1)


C THE FOLLOWING BLOCK DISPLAYS THE VGT GEOMETRY FOR PREDEFINED LINK
C LENGTHS.
C CALL THE TRUSS SUBROUTINE TO CALCULATE THE COORDINATES OF THE NODES.
C THE SUBROUTINE IS CALLED IN A DO LOOP TO CONSTRUCT THE MODULES.
      DO 1 I = 1,NUM
      CALL VGTRUS(AL1(I),AL2(I),AL3(I),
     &            ALPHA(I),BETA(I),GAMMA(I),NUM,IKFLAG)
1     CONTINUE

C THIS BLOCK ASSOCIATES THE TRUSS STRUCTURE WITH VIEW1.
      CALL GPARV(IWSID,VIEW1,1,1.0)
      CALL GPARV(IWSID,VIEW1,2,1.0)
      CALL GPVMP3(IWSID,VIEW1,WINDO2,VPR1,TYPE,PRP,DIST,NEAR,FAR)
      CALL GPVCH(IWSID,VIEW1,CLIP,CLIP,CLIP,OFF,0,ON,WHITE,ON)
      CALL GPUPWS(IWSID,PRFORM)
      CALL GPDRV(IWSID,VIEW1,1)

C BLOCK FOR INTERACTIVE GRAPHICAL INPUT.
C THE FOLLOWING BLOCK INITIALIZES THE VALUATOR DEVICE AND ASKS FOR
C AN INPUT FROM THE OPERATOR.
      DO 20 I = 1,6
      CALL GPVLMO(IWSID,I,1,2)
      CALL GPINVL(IWSID,I,39.01,4,AREA1,39.0,51.0,16,VADAT1)
      CALL GPVLMO(IWSID,I,3,2)
      AREA1(3) = AREA1(3) - 0.01
20    CONTINUE

C THIS BLOCK INITIALIZES THE CHOICE DEVICE.
      CALL GPCHMO(IWSID,1,1,2)
      AREA1(3) = 0.0
      CALL GPINCH(IWSID,1,1,2,AREA1,DATAL,DATA1)
      CALL GPCHMO(IWSID,1,3,2)

C REQUEST A PF KEY HIT TO EXIT OUT OF THE DO LOOP.
C DISPLAY A MESSAGE REQUESTING INPUT.
      CALL GPMSG(IWSID,LNGT,MSG)

C CALL TO EMPTY THE STRUCTURE.
30    CALL GPEST(1)
```

```
C THIS BLOCK READS THE GRAPHICAL INPUT AND CORRESPONDINGLY PROCESSES IT.
C DEFINING THE TIME LIMIT.
      TIME = 20.0
      CALL GPAWEV(TIME,J,CLASS,DEVICE)
      IF(CLASS.EQ.3)GOTO 40

C THIS BLOCK LETS USER TO USE CHOICE TO EXIT OUT OF THE ANIMATION
C LOOP.
      CALL GPGTCH(CHOICE)
      IF(CHOICE.EQ.32)GOTO 10
      CALL GPDAST
      GOTO 30

C THIS BLOCK PROCESSES VALUATOR INPUT .
40    CONTINUE
      IF(DEVICE.EQ.1)CALL GPGTVL(AL11)
      IF(DEVICE.EQ.2)CALL GPGTVL(AL12)
      IF(DEVICE.EQ.3)CALL GPGTVL(AL13)
      IF(DEVICE.EQ.4)CALL GPGTVL(AL14)
      IF(DEVICE.EQ.5)CALL GPGTVL(AL15)
      IF(DEVICE.EQ.6)CALL GPGTVL(AL16)

C DOWNLOADING THE NEW VALUES.
      AL1(1) = AL11
      AL2(1) = AL12
      AL3(1) = AL13
      AL1(2) = AL14
      AL2(2) = AL15
      AL3(2) = AL16

C TO TEST IF THE THREE LENGTHS OF THE LATERAL TRIANGLE SATISFY THE
C BASIC TRIANGLE RULE.
      IF((AL11 + AL12).LT.(AL13 - 2.5))GOTO 205
      IF((AL12 + AL13).LT.(AL11 - 2.5))GOTO 205
      IF((AL13 + AL11).LT.(AL12 - 2.5))GOTO 205
      IF((AL14 + AL15).LT.(AL16 - 2.5))GOTO 205
      IF((AL15 + AL16).LT.(AL14 - 4.0))GOTO 205
      IF((AL16 + AL14).LT.(AL15 - 4.0))GOTO 205

C CALL THE VGTRUS SUBROUTINE TO DRAW THE FIRST MODULE.
C TO SET THE IKFLAG TO EQUAL TO ZERO.
      IKFLAG = 0

C TO CALL THE VGTRUS SUBROUTINE IN A DO LOOP TO CONSTRUCT THE MODULES.
      DO 2 I = 1,NUM
      CALL VGTRUS(AL1(I),AL2(I),AL3(I),
     &            ALPHA(I),BETA(I),GAMMA(I),NUM,IKFLAG)
2     CONTINUE

C THIS BLOCK ASSOCIATES THE TRUSS STRUCTURE WITH VIEW1.
      CALL GPARV(IWSID,VIEW1,1,1)
      CALL GPARV(IWSID,VIEW1,2,1)
      CALL GPVMP3(IWSID,VIEW1,WINDO2,VPR1,TYPE,PRP,DIST,NEAR,FAR)
      CALL GPVCH(IWSID,VIEW1,CLIP,CLIP,CLIP,OFF,0,ON,WHITE,ON)
      CALL GPUPWS(IWSID,PRFORM)
      CALL GPDRV(IWSID,VIEW1,1)
      CALL GPDAST
      GOTO 30

205   CONTINUE
      WRITE(15,*)'THE THREE VARIABLE LINKS DO
     >            NOT FOLLOW THE BASIC TRIANGLE LAW'

10    CONTINUE

C CALL TO DEACTIVATE THE WORKSTATION.
      CALL GPDAST
```

```
C CALL TO CLOSE GRAPHICS.
      CALL GPCLPH

      STOP
      END



C-----------------------------------------------------------------------
C     SUBROUTINE VGTRUS CALCULATES COORDINATES OF JOINTS OF THE VGT
C     FOR GIVEN LENGTHS OF THE VARIABLE LINKS.
C-----------------------------------------------------------------------

      SUBROUTINE VGTRUS(AL1,AL2,AL3,
     &                  ALPHA,BETA,GAMMA,NUM,IKFLAG)


C TYPE AND DATA DECLARATION.
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/TRIAL/UA(3),UB(3),UC(3),A0(3),B0(3),C0(3),
     &             A1MA0(3),B1MB0(3),C1MC0(3),A1B1MG,A1C1MG,C1B1MG

      COMMON PI

      DIMENSION AJ(3),BJ(3),CJ(3)
      DIMENSION PM(3,3),QM(3,3),QIM(3,3),RM(3,3,2)
      DIMENSION A(3,3),B(3)
      DIMENSION ANORM1(3),ANORM2(3),ANORM3(3),ANORM4(3)
      DIMENSION BJMAJ(3),CJMAJ(3),CJMBJ(3)
      DIMENSION A0MAJ(3),B0MAJ(3),C0MBJ(3),AJMA0(3)
      DIMENSION AJPCJ(3),AJPBJ(3),BJPCJ(3)
      DIMENSION A0MACJ(3),B0MABJ(3),C0MBCJ(3)
      DIMENSION ANODE7(3),ANODE8(3),ANODE9(3)
      DIMENSION UD(3),UE(3),UF(3)
      DIMENSION UAT(3),UBT(3),UN(3),UZ(3),TRAN(3,3),D(3)
      REAL*4    XYZ1(3,13),XYZ2(3,10),XYZ3(3,3),XYZ4(3,2)
      DATA IN,IOUT /12,10/

C  SQUARING THE LINK LENGTHS FOR EASE OF CALCULATIONS.
      A1B1MG = AL1 * AL1
      A1C1MG = AL2 * AL2
      C1B1MG = AL3 * AL3

C
C DEFINING THE CONSTANTS OF THE PROGRAM.
C

C THE TOLERANCE LIMIT, FOR NEWTON-RAPHSON METHOD.
      TOL = 0.00001

C DEFINING THE VALUE OF THE TRIGONOMETRIC CONSTANT PI.
      PI = 3.141592654

C REDEFINING THE POSITION OF REVOLUTE JOINT A0.
      A0(1)= 0.0
      A0(2)= 0.0
      A0(3)= 0.0

C DEFINING THE POSITION OF REVOLUTE JOINT B0.
      B0(1)=23.25
      B0(2)=0.0
      B0(3)=40.27018128

C DEFINING THE POSITION OF REVOLUTE JOINT C0.
      C0(1)=46.5
      C0(2)=0.0
```

```
      C0(3)=0.0

C  THE POSITION OF VECTOR (A1 - A0).
      A1MA0(1) = -10.45864780
      A1MA0(2) =   0.0
      A1MA0(3) =  32.8850964

C  THE POSITION OF VECTOR (B1 - B0).
      B1MB0(1) =  33.7086478
      B1MB0(2) =   0.0
      B1MB0(3) =  -7.38509064

C  THE POSITION OF VECTOR (C1 - C0).
      C1MC0(1) = -23.25
      C1MC0(2) =   0.0
      C1MC0(3) = -25.5

      IF(IKFLAG.NE.0)GOTO 1111

C  THE UNIT VECTOR ASSOCIATED WITH A0.
      UA(1) = -0.5
      UA(2) =  0.0
      UA(3) = -0.86602543

C  THE UNIT VECTOR ASSOCIATED WITH B0.
      UB(1) = -0.5
      UB(2) =  0.0
      UB(3) =  0.86602543

C  THE UNIT VECTOR ASSOCIATED WITH C0.
      UC(1) =  1.0
      UC(2) =  0.0
      UC(3) =  0.0

1111  CONTINUE

C CALLING THE NEWTON ROUTINE FOR SOLVING FOR VALUES OF THE ANGLES.
      CALL NEWTON (ALPHA,BETA,GAMMA,TOL,K)

C CONVERTING ALPHA BETA GAMMA INTO POSITIVE VALUES.
      ALPHA = DABS(ALPHA)
      BETA = DABS(BETA)
      GAMMA = DABS(GAMMA)

C TO FIND COORDINATES OF THE VERTEX AJ.
      CALL RMAXIS (UA,ALPHA,RM,2)
      CALL ROTATE (AJ,A0,RM,A1MA0,2)

C TO FIND COORDINATES OF THE VERTEX BJ.
      CALL RMAXIS (UB,BETA,RM,2)
      CALL ROTATE (BJ,B0,RM,B1MB0,2)

C TO FIND COORDINATES OF THE VERTEX CJ.
      CALL RMAXIS (UC,GAMMA,RM,2)
      CALL ROTATE (CJ,C0,RM,C1MC0,2)

C TO CALCULATE NORMAL TO THE LATERAL VARIABLE PLANE.
      DO 7 I = 1,3
      BJMAJ(I) = BJ(I) - AJ(I)
7     CJMBJ(I) = CJ(I) - BJ(I)
      CALL CROSS (ANORM1,BJMAJ,CJMBJ)

C
C-----------------------------------------------------------------------
C     COORDINATES OF NODE 7 ARE CALCULATED IN THE FOLLOWING BLOCK.
C-----------------------------------------------------------------------
C
```

```
      DO 8 I = 1,3
      CJMAJ(I) = CJ(I) - AJ(I)
    8 AJMA0(I) = AJ(I) - A0(I)
      CALL CROSS (ANORM2,AJMA0,CJMAJ)

C TO CALCULATE THE ANGLE BETWEEN THE LATERAL AND SIDE PLANES.
      ANUM1 = DOT (ANORM1,ANORM2)
      DEN1 = DABS(DSQRT (ANORM1(1)**2 + ANORM1(2)**2 + ANORM1(3)**2)) *
     $       DABS(DSQRT (ANORM2(1)**2 + ANORM2(2)**2 + ANORM2(3)**2))
      THETA = DABS(DACOS (ANUM1/DEN1))
      IF(THETA.GT.PI)THETA = 2.*PI - THETA
      THETA = 2. * THETA

      DO 9 I = 1,3
      AJPCJ(I) = (AJ(I) + CJ(I)) / 2.
    9 A0MACJ(I) = A0(I) - AJPCJ(I)

C TO FIND UNIT VECTOR ALONG THE LINK CJAJ.
      DO 10 I = 1,3
   10 UD(I) = CJMAJ(I)/(DSQRT(CJMAJ(1)**2 + CJMAJ(2)**2 + CJMAJ(3)**2))
      CALL RMAXIS (UD,THETA,RM,2)
      CALL ROTATE (ANODE7,AJPCJ,RM,A0MACJ,2)

C
C-----------------------------------------------------------------------
C     COORDINATES OF NODE 8 ARE CALCULATED IN THE FOLLOWING BLOCK.
C-----------------------------------------------------------------------
C
      DO 12 I = 1,3
   12 B0MAJ(I) = B0(I) - AJ(I)
      CALL CROSS (ANORM3,B0MAJ,BJMAJ)

      ANUM2 = DOT (ANORM1,ANORM3)
      DEN2 = DABS(DSQRT (ANORM1(1)**2 + ANORM1(2)**2 + ANORM1(3)**2)) *
     $       DABS(DSQRT (ANORM3(1)**2 + ANORM3(2)**2 + ANORM3(3)**2))
      THETA1 = DABS(DACOS (ANUM2/DEN2))
      IF(THETA1.GT.PI)THETA1 = 2.* PI - THETA1
      THETA1 = 2. * THETA1

      DO 13 I = 1,3
      AJPBJ(I) = (AJ(I) + BJ(I)) / 2.
   13 B0MABJ(I) = B0(I) - AJPBJ(I)

C TO FIND UNIT VECTOR ALONG THE LINK BJAJ.
      DO 14 I =1,3
   14 UE(I) = -BJMAJ(I)/(DSQRT(BJMAJ(1)**2 + BJMAJ(2)**2 + BJMAJ(3)**2))
      CALL RMAXIS (UE,THETA1,RM,2)
      CALL ROTATE (ANODE8,AJPBJ,RM,B0MABJ,2)

C
C-----------------------------------------------------------------------
C     COORDINATES OF NODE 9 ARE CALCULATED IN THE FOLLOWING BLOCK.
C-----------------------------------------------------------------------
C
      DO 16 I = 1,3
   16 C0MBJ(I) = C0(I) - BJ(I)
      CALL CROSS (ANORM4,C0MBJ,CJMBJ)

      ANUM3 = DOT (ANORM1,ANORM4)
      DEN3 = DABS(DSQRT (ANORM1(1)**2 + ANORM1(2)**2 + ANORM1(3)**2)) *
     $       DABS(DSQRT (ANORM4(1)**2 + ANORM4(2)**2 + ANORM4(3)**2))
      THETA2 = DABS(DACOS (ANUM3/DEN3))
      IF(THETA2.GT.PI)THETA2 = 2.*PI - THETA2
      THETA2 = 2. * THETA2

      DO 17 I = 1,3
      BJPCJ(I) = (BJ(I) + CJ(I)) / 2.
```

```
  17  COMBCJ(I) = CO(I) - BJPCJ(I)

C TO FIND UNIT VECTOR ALONG THE LINK CJBJ.
      DO 18 I =1,3
  18  UF(I) = -CJMBJ(I)/(DSQRT(CJMBJ(1)**2 + CJMBJ(2)**2 + CJMBJ(3)**2))
      CALL RMAXIS (UF,THETA2,RM,2)
      CALL ROTATE (ANODE9,BJPCJ,RM,COMBCJ,2)

C CALL TO CHECK THE IKFLAG.
      IF(IKFLAG.EQ.0)GOTO 31
C TO CALCULATE THE UNIT VECTORS ALONG THE SIDE LENGTHS OF THE TOP
C LATERAL PLANE.
      DO 49 I = 1, 3
      UAT(I) = (XYZ3(I,1) - XYZ3(I,3))/46.5
      UBT(I) = (XYZ3(I,2) - XYZ3(I,1))/46.5
49    CONTINUE

C CALLING CROSS SUBROUTINE TO CALCULATE NORMAL TO THE TOP PLANE.
      CALL CROSS(UN,UAT,UBT)
      UN(2) = UN(2)/(SIN(PI/3.))

C TO CALL THE CROSS SUBROUTINE TO CALCULATE UNIT VECTOR ALONG Z AXIS OF
C THE MOVING COORDINATE SYSTEM.
      CALL CROSS(UZ,UN,UAT)

C TO DEFINE THE ELEMENTS OF THE ROTATION MATRIX.
      TRAN(1,1) = -UAT(1)
      TRAN(2,1) = -UAT(2)
      TRAN(3,1) = -UAT(3)
      TRAN(1,2) =  UN(1)
      TRAN(2,2) =  UN(2)
      TRAN(3,2) =  UN(3)
      TRAN(1,3) =  UZ(1)
      TRAN(2,3) =  UZ(2)
      TRAN(3,3) =  UZ(3)

C TO DEFINE THE TRANSLATION VECTOR.
      D(1) = XYZ3(1,1) - AO(1)
      D(2) = XYZ3(2,1) - AO(2)
      D(3) = XYZ3(3,1) - AO(3)

C CALLING THE MATRIX MULTIPLICATION SUBROUTINE TO TRANSFORM THE
C VERTEX COORDINATES FROM LOCAL TO GLOBAL SYSTEMS.
      CALL MATMUL(TRAN,AJ,D)
      CALL MATMUL(TRAN,BJ,D)
      CALL MATMUL(TRAN,CJ,D)
      CALL MATMUL(TRAN,ANODE7,D)
      CALL MATMUL(TRAN,ANODE8,D)
      CALL MATMUL(TRAN,ANODE9,D)

C TO DEFINE BASE COORDINATES FOR THE REPEATING MODULES.
      DO 3 I = 1,3
      AO(I) = XYZ3(I,1)
      BO(I) = XYZ3(I,2)
      CO(I) = XYZ3(I,3)
3     CONTINUE

31    CONTINUE


C THE FOLLOWING BLOCK DOWNLOADS THE COORDINATES OF THE JOINTS.
      DO 999 I =1,3
      XYZ1(I,1) = AO(I)
      XYZ1(I,2) = BO(I)
      XYZ1(I,3) = CO(I)
      XYZ1(I,4) = AO(I)
      XYZ1(I,5) = AJ(I)
```

```
      XYZ1(I,6) = BJ(I)
      XYZ1(I,7) = CJ(I)
      XYZ1(I,8) = AJ(I)
      XYZ1(I,9) = BO(I)
      XYZ1(I,10) = BJ(I)
      XYZ1(I,11) = CO(I)
      XYZ1(I,12) = CJ(I)
      XYZ1(I,13) = AO(I)
      XYZ2(I,1) = ANODE7(I)
      XYZ2(I,2) = ANODE8(I)
      XYZ2(I,3) = ANODE9(I)
      XYZ2(I,4) = ANODE7(I)
      XYZ2(I,5) = CJ(I)
      XYZ2(I,6) = ANODE9(I)
      XYZ2(I,7) = BJ(I)
      XYZ2(I,8) = ANODE8(I)
      XYZ2(I,9) = AJ(I)
      XYZ2(I,10) = ANODE7(I)
      XYZ3(I,1) = ANODE7(I)
      XYZ3(I,2) = ANODE8(I)
      XYZ3(I,3) = ANODE9(I)
      XYZ4(I,1) = (ANODE7(I) + ANODE8(I) + ANODE9(I))/3.0
      XYZ4(I,2) = 77.75 * UN(I) + XYZ4(I,1)
999   CONTINUE

C CALL TO OPEN A STRUCTURE FOR DRAWING THE VGT.
      CALL GPOPST(1)
C CALL TO DRAW POLYLINES DRAWING THE V.G.T.
      CALL GPLWSC(10)
      CALL GPPL3(13,3,XYZ1)
      CALL GPPL3(10,3,XYZ2)

C CHECKING THE IKFLAG TO DETERMINE IF THE MODULE TO BE DRAWN IS THE LAST
C MODULE.
C INCREMENTING THE FLAG.
      IKFLAG = IKFLAG + 1
      IF(IKFLAG.NE.NUM)GOTO 1000

C CALL TO FILL UP THE TOP SURFACE OF THE TOP MODULE WITH BLUE COLOR.
C TO DEFINE ATTRIBUTES OF THE FILL AREA.
      CALL GPEF(2)
      CALL GPECI(1)
      CALL GPIS(2)
      CALL GPICI(4)
      CALL GPPG3(1,3,3,XYZ3)
      CALL GPPL3(2,3,XYZ4)
1000  CONTINUE
      CALL GPCLST

      RETURN
      END


C-----------------------------------------------------------------------
C
C     SUBROUTINE ANYRS - FORMS FUN1 AND CALCULATES PARTIAL DERIVATIVES.
C
C-----------------------------------------------------------------------

      SUBROUTINE ANYRS (ALPHA,BETA,GAMMA,FUNC1,DIFF1,DIF1)

C
C   INPUT STATEMENTS
C
C TYPE DECLARATION.
      IMPLICIT REAL*8(A-H,O-Z)
```

```
C DECLARING THE COMMON BLOCK.
      COMMON/TRIAL/UA(3),UB(3),UC(3),AO(3),BO(3),CO(3),
     &            A1MAO(3),B1MBO(3),C1MCO(3),A1B1MG,A1C1MG,C1B1MG

      COMMON PI

C DECLARING THE DIMENSIONS.

      DIMENSION AJ(3),BJ(3),CJ(3)
      DIMENSION PM(3,3),QM(3,3),QIM(3,3),RM(3,3,2)
      DIMENSION T1(3),T2(3),T3(3),T4(3),T5(3),T6(3),T7(3),T8(3)
      DIMENSION V1(3),V2(3),V3(3),V4(3),V5(3),V6(3)

C     POSITION ANALYSIS OF AO-A1-B1-BO MECHANISM.

      DALPH = 0.00000001
      ALPH = ALPHA + DALPH
      DBET = 0.00000001
      BET = BETA + DBET
C
      DO 24 I = 1,3
      T1(I) = A1MAO(I)
      T4(I) = AO(I) - BO(I)
   24 T2(I) = B1MBO(I)

      CALL RMAXIS (UA,ALPH,RM,2)
      CALL ROTATE (AJ,AO,RM,T1,2)
      DO 25 I = 1,3
      V1(I) = AJ(I) - AO(I)
   25 V2(I) = AJ(I) - BO(I)

      CALL RMAXIS (UA,ALPHA,RM,2)
      CALL ROTATE (AJ,AO,RM,T1,2)
      DO 26 I = 1,3
      T5(I) = AJ(I) - AO(I)
   26 T3(I) = AJ(I) - BO(I)

      CALL PMTX (UB,PM)
      CALL QMTX (UB,QM)
      CALL QIMTX (UB,QM,QIM)
      CALL MTXVEC (T6,QIM,T2)
      E = DOT (V2,T6)
      E1 = DOT (T3,T6)
      CALL MTXVEC (T7,PM,T2)
      F = DOT (V2,T7)
      F1 = DOT (T3,T7)
      CALL MTXVEC (T8,QM,T2)
      G = DOT(V2,T8) + .5 * (A1B1MG - DOT(V2,V2) - DOT(T2,T2))
      G1 = DOT(T3,T8) + .5 * (A1B1MG - DOT(T3,T3) - DOT(T2,T2))
      APP1 = E * DCOS(BETA) + F * DSIN(BETA) + G
      FUNC1 = E1 * DCOS(BETA) + F1 * DSIN(BETA) + G1
      DIFF1 = (APP1 - FUNC1) / DALPH
      APB = E1 * DCOS(BET) + F1 * DSIN(BET) + G1
      DIF1 = (APB - FUNC1) / DBET
C
      RETURN
      END

C-----------------------------------------------------------------------
C
C     SUBROUTINE ANYRS1 - FORMS FUN2 AND CALCULATES PARTIAL DERIVATIVES.
C
C-----------------------------------------------------------------------

      SUBROUTINE ANYRS1 (ALPHA,BETA,GAMMA,FUNC2,DIFF2,DIF2)
```

```
C
C   INPUT STATEMENTS
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/TRIAL/UA(3),UB(3),UC(3),AO(3),BO(3),CO(3),
     &            A1MAO(3),B1MBO(3),C1MCO(3),A1B1MG,A1C1MG,C1B1MG
      COMMON PI

      DIMENSION AJ(3),BJ(3),CJ(3)
      DIMENSION PM(3,3),QM(3,3),QIM(3,3),RM(3,3,2)
      DIMENSION S1(3),S2(3),S3(3),S4(3),S5(3),S6(3),S7(3),S8(3)
      DIMENSION V1(3),V2(3),V3(3),V4(3),V5(3),V6(3)

C     POSITION ANALYSIS OF CO-C1-A1-AO MECHANISM .

      DALPH = 0.000000001
      ALPH = ALPHA + DALPH
      DGAMA = 0.000000001
      GAMA = GAMMA + DGAMA
C
      DO 24 I = 1,3
      S1(I) = C1MCO(I)
      S4(I) = CO(I) - AO(I)
   24 S2(I) = A1MAO(I)

      CALL RMAXIS (UC,GAMA,RM,2)
      CALL ROTATE (CJ,CO,RM,S1,2)
      DO 25 I = 1,3
      V3(I) = CJ(I) - CO(I)
   25 V4(I) = CJ(I) - AO(I)

      CALL RMAXIS (UC,GAMMA,RM,2)
      CALL ROTATE (CJ,CO,RM,S1,2)
      DO 26 I = 1,3
      S5(I) = CJ(I) - CO(I)
   26 S3(I) = CJ(I) - AO(I)

      CALL PMTX (UA,PM)
      CALL QMTX (UA,QM)
      CALL QIMTX (UA,QM,QIM)
      CALL MTXVEC (S6,QIM,S2)
      E = DOT (V4,S6)
      E1 = DOT (S3,S6)
      CALL MTXVEC (S7,PM,S2)
      F = DOT (V4,S7)
      F1 = DOT (S3,S7)
      CALL MTXVEC (S8,QM,S2)
      G = DOT(V4,S8) + .5 * (A1C1MG - DOT(V4,V4) - DOT(S2,S2))
      G1 = DOT(S3,S8) + .5 * (A1C1MG - DOT(S3,S3) - DOT(S2,S2))
      APP2 = E * DCOS(ALPHA) + F * DSIN(ALPHA) + G
      FUNC2 = E1 * DCOS(ALPHA) + F1 * DSIN(ALPHA) + G1
      DIFF2 = (APP2 - FUNC2) / DGAMA
      APG = E1 * DCOS(ALPH) + F1 * DSIN(ALPH) + G1
      DIF2 = (APG - FUNC2) / DALPH
C
      RETURN
      END

C-----------------------------------------------------------------------
C
C     SUBROUTINE ANYRS2 - FORMS FUN3 AND CALCULATES PARTIAL DERIVATIVES.
C
C-----------------------------------------------------------------------

      SUBROUTINE ANYRS2 (ALPHA,BETA,GAMMA,FUNC3,DIFF3,DIF3)
```

```
C
C   INPUT STATEMENTS
C
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/TRIAL/UA(3),UB(3),UC(3),AO(3),BO(3),CO(3),
     &            A1MAO(3),B1MBO(3),C1MCO(3),A1B1MG,A1C1MG,C1B1MG
      COMMON PI

      DIMENSION AJ(3),BJ(3),CJ(3)
      DIMENSION PM(3,3),QM(3,3),QIM(3,3),RM(3,3,2)
      DIMENSION W1(3),W2(3),W3(3),W4(3),W5(3),W6(3),W7(3),W8(3)
      DIMENSION V1(3),V2(3),V3(3),V4(3),V5(3),V6(3)

C     POSITION ANALYSIS OF BO-B1-C1-CO MECHANISM.

      DGAMA = 0.000000001
      GAMA = GAMMA + DGAMA
      DBET = 0.000000001
      BET = BETA + DBET
C
      DO 24 I = 1,3
      W1(I) = B1MBO(I)
      W4(I) = BO(I) - CO(I)
   24 W2(I) = C1MCO(I)

      CALL RMAXIS (UB,BET,RM,2)
      CALL ROTATE (BJ,BO,RM,W1,2)
      DO 25 I = 1,3
      V5(I) = BJ(I) - BO(I)
   25 V6(I) = BJ(I) - CO(I)

      CALL RMAXIS (UB,BETA,RM,2)
      CALL ROTATE (BJ,BO,RM,W1,2)
      DO 26 I = 1,3
      W5(I) = BJ(I) - BO(I)
   26 W3(I) = BJ(I) - CO(I)

      CALL PMTX (UC,PM)
      CALL QMTX (UC,QM)
      CALL QIMTX (UC,QM,QIM)
      CALL MTXVEC (W6,QIM,W2)
      E = DOT (V6,W6)
      E1 = DOT (W3,W6)
      CALL MTXVEC (W7,PM,W2)
      F = DOT (V6,W7)
      F1 = DOT (W3,W7)
      CALL MTXVEC (W8,QM,W2)
      G = DOT(V6,W8) + .5*(C1B1MG - DOT(V6,V6) - DOT(W2,W2))
      G1 = DOT(W3,W8) + .5*(C1B1MG - DOT(W3,W3) - DOT(W2,W2))
      APP3 = E * DCOS(GAMMA) + F * DSIN(GAMMA) + G
      FUNC3 = E1 * DCOS(GAMMA) + F1 * DSIN(GAMMA) + G1
      DIFF3 = (APP3 - FUNC3) / DBET
      APB = E1 * DCOS(GAMA) + F1 * DSIN(GAMA) + G1
      DIF3 = (APB - FUNC3) / DGAMA
C
      RETURN
      END


C-----------------------------------------------------------------------
C   FUNCTION DOT(V1,V2) COMPUTES VECTOR DOT PRODUCT V1*V2.
C-----------------------------------------------------------------------

      FUNCTION DOT (V1,V2)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION V1(3),V2(3)
      DOT = 0.0
```

```
      DO 10 I = 1,3
   10 DOT = DOT + V1(I)*V2(I)
      RETURN
      END

C-----------------------------------------------------------------------
C   SUBROUTINE CROSS COMPUTES THE CROSS PRODUCT OF V1 AND V2
C-----------------------------------------------------------------------

      SUBROUTINE CROSS (ANORM,V1,V2)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION V1(3),V2(3),ANORM(3)
      ANORM(1) = V1(2)*V2(3) - V1(3)*V2(2)
      ANORM(2) = -(V1(1)*V2(3) - V1(3)*V2(1))
      ANORM(3) = V1(1)*V2(2) - V1(2)*V2(1)
      RETURN
      END

C-----------------------------------------------------------------------
C   SUBROUTINE ROTATE
C   ROTATES VECTOR (P1MQ1) TO (PJMQJ)=(RM)*(P1MQ1)
C   COMPUTES PJ GIVEN P1,Q1,QJ,RM
C-----------------------------------------------------------------------

      SUBROUTINE ROTATE (PJ,QJ,RM,P1MQ1,J)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION PJ(3),QJ(3),P1MQ1(3),RM(3,3,2)
      DO 10 I = 1,3
   10 PJ(I) = RM(I,1,J)*(P1MQ1(1))+RM(I,2,J)*(P1MQ1(2))+
     $        RM(I,3,J)*(P1MQ1(3))+QJ(I)
      RETURN
      END

C-----------------------------------------------------------------------
C   SUBROUTINE ROTVEC
C   COMPUTES V2 = (RM)*V1
C-----------------------------------------------------------------------

      SUBROUTINE ROTVEC (V2,RM,V1,J)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION V2(3),RM(3,3,J),V1(3)
      DO 10 I = 1,3
      V2(I) = 0.0
      DO 10 K = 1,3
   10 V2(I) = RM(I,K,J) * V1(K) + V2(I)
      RETURN
      END

C-----------------------------------------------------------------------
C   SUBROUTINE PMTX
C   COMPUTES P-MATRIX ELEMENTS
C-----------------------------------------------------------------------

      SUBROUTINE PMTX (U,PM)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION U(3),PM(3,3)
      PM(1,1) = 0.0
      PM(1,2) = -U(3)
      PM(1,3) = U(2)
      PM(2,1) = U(3)
      PM(2,2) = 0.0
      PM(2,3) = -U(1)
      PM(3,1) = -U(2)
      PM(3,2) = U(1)
      PM(3,3) = 0.0
      RETURN
      END
```

```
C-----------------------------------------------------------------------
C   SUBROUTINE QMTX
C   COMPUTES Q-MATRIX ELEMENTS
C-----------------------------------------------------------------------

      SUBROUTINE QMTX (U,QM)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION U(3),QM(3,3)
      QM(1,1) = U(1)*U(1)
      QM(1,2) = U(1)*U(2)
      QM(1,3) = U(1)*U(3)
      QM(2,1) = U(2)*U(1)
      QM(2,2) = U(2)*U(2)
      QM(2,3) = U(2)*U(3)
      QM(3,1) = U(3)*U(1)
      QM(3,2) = U(3)*U(2)
      QM(3,3) = U(3)*U(3)
      RETURN
      END

C-----------------------------------------------------------------------
C   SUBROUTINE QIMTX
C   COMPUTES (I-Q) MATRIX ELEMENTS
C-----------------------------------------------------------------------

      SUBROUTINE QIMTX (U,QM,QIM)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION U(3),QM(3,3),QIM(3,3)
      DO 10 I = 1,3
      DO 10 J = 1,3
   10 QIM(I,J) = -QM(I,J)
      QIM(1,1) = 1. + QIM(1,1)
      QIM(2,2) = 1. + QIM(2,2)
      QIM(3,3) = 1. + QIM(3,3)
      RETURN
      END

C-----------------------------------------------------------------------
C   SUBROUTINE MTXVEC
C   RETURNS PRODUCT OF A MATRIX AND A VECTOR IN TEMP
C-----------------------------------------------------------------------

      SUBROUTINE MTXVEC (TEMP,A,VEC)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(3,3),VEC(3),TEMP(3)
      DO 10 I = 1,3
      TEMP(I) = 0.0
      DO 10 J = 1,3
   10 TEMP(I) = TEMP(I) + A(I,J) * VEC(J)
      RETURN
      END

C-----------------------------------------------------------------------
C   SUBROUTINE RMAXIS
C   COMPUTES ROTATION MATRIX ELEMENTS IN TERMS OF ANGLE PHI ABOUT U
C-----------------------------------------------------------------------

      SUBROUTINE RMAXIS (U,PHI,RM,J)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION RM(3,3,2),U(3)
      COMMON /PRNTR/PRNT
      LOGICAL PRNT
      C = DCOS (PHI)
      S = DSIN (PHI)
      V = 1. - C
      RM(1,1,J) = U(1)*U(1)*V+C
```

```
      RM(1,2,J) = U(1)*U(2)*V-U(3)*S
      RM(1,3,J) = U(1)*U(3)*V+U(2)*S
      RM(2,1,J) = U(1)*U(2)*V+U(3)*S
      RM(2,2,J) = U(2)*U(2)*V+C
      RM(2,3,J) = U(2)*U(3)*V-U(1)*S
      RM(3,1,J) = U(1)*U(3)*V-U(2)*S
      RM(3,2,J) = U(2)*U(3)*V+U(1)*S
      RM(3,3,J) = U(3)*U(3)*V+C
      RETURN
      END

C-----------------------------------------------------------------------
C  SUBROUTINE NEWTON - SOLVES THE THREE FUNCTIONS FUN1, FUN2 AND FUN3
C  ITERATIVELY USING NEWTON-RAPHSON METHOD.
C-----------------------------------------------------------------------

      SUBROUTINE NEWTON (ALPHA,BETA,GAMMA,TOL,K)

      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION UA(3),AO(3),A1MAO(3),AJ(3)
      DIMENSION UB(3),BO(3),B1MBO(3),BJ(3)
      DIMENSION UC(3),CO(3),C1MCO(3),CJ(3)
      DIMENSION A(3,3),B(3)

C
C CALLING THE ANALYSIS SUBROUTINE FOR CONSTRUCTING FUNCTION EQUATIONS
C FOR INPUT TO THE NEWTON SUBROUTINE.
C
      DO 10 K = 1,500
      CALL ANYRS (ALPHA,BETA,GAMMA,FUNC1,DIFF1,DIF1)
      CALL ANYRS1(ALPHA,BETA,GAMMA,FUNC2,DIFF2,DIF2)
      CALL ANYRS2(ALPHA,BETA,GAMMA,FUNC3,DIFF3,DIF3)
      IC = 0
      IF (DABS(FUNC1) .LT. TOL) IC = IC + 1
      IF (DABS(FUNC2) .LT. TOL) IC = IC + 1
      IF (DABS(FUNC3) .LT. TOL) IC = IC + 1
      IF (IC .EQ. 3) RETURN
C
      A(1,1) = DIFF1
      A(1,2) = DIF1
      A(1,3) = 0.0
      A(2,1) = DIF2
      A(2,2) = 0.0
      A(2,3) = DIFF2
      A(3,1) = 0.0
      A(3,2) = DIFF3
      A(3,3) = DIF3
C
      B(1) = -FUNC1
      B(2) = -FUNC2
      B(3) = -FUNC3

      D = A(2,2) * A(3,3) - A(2,3) * A(3,2)
      D1 = A(2,1) * A(3,3) - A(3,1) * A(2,3)
      D2 = A(2,1) * A(3,2) - A(3,1) * A(2,2)
      DET = A(1,1) * D - A(1,2) * D1 + A(1,3) * D2
      DETX = B(1) * D - A(1,2) * (B(2) * A(3,3) - B(3) * A(2,3)) +
     $       A(1,3) * (B(2) * A(3,2) - B(3) * A(2,2))
      DETY = A(1,1)*(B(2)*A(3,3) - B(3)*A(2,3)) - B(1)*D1 +
     $       A(1,3)*(B(3)*A(2,1) - B(2)*A(3,1))
      DETZ = A(1,1)*(B(3)*A(2,2) - B(2)*A(3,2)) + B(1)*D2 -
     $       A(1,2)*(B(3)*A(2,1) - B(2)*A(3,1))
C
      IF (DET .EQ. 0) THEN
      WRITE(6,*)'THE DETERMINANT IS ZERO'
      GOTO 10
      ENDIF
```

```
      ALPHA = ALPHA + DETX / DET
      BETA = BETA + DETY / DET
      GAMMA = GAMMA + DETZ / DET
   10 CONTINUE

      RETURN
      END

C-------------------------------------------------------------------------
C THIS SUBROUTINE TRANSFORMS A VECTOR FROM ONE COORDINATE SYSTEM TO THE
C OTHER SEPARATED BY VECTOR D.
C-------------------------------------------------------------------------

      SUBROUTINE MATMUL(A,B,D)

C TO DECLARE THE TYPE OF THE VARIABLES.
      REAL*8 A(3,3),B(3,1),C(3,1),D(3)
      INTEGER I,K

      DO 10 I = 1,3
      C(I,1) = 0.0
      DO 10 K = 1,3
      C(I,1) = C(I,1) + A(I,K) * B(K,1)
10    CONTINUE

C TO DOWNLOAD THE C VECTOR INTO B VECTOR AND ADD THE TRANSFORMATION
C VECTOR.
      DO 20 I = 1,3
      B(I,1) = C(I,1) + D(I)
20    CONTINUE
      C(I,1) = 0.0
      RETURN
      END
```

Appendix C
Inverse Program Listing

```
*************************************************************************

C THIS PROGRAM USES QUISI-NEWTON ALGORITHM TO SOLVE A SET OF
C SIMULTANEOUS EQUATIONS.
C PROGRAMMER : D. P. GOKHALE.

*************************************************************************

C VARIABLE TYPE AND DATA DECLARATION.
      COMMON/GUESS1/X, Y, Z,ALPHA,BETA,GAMMA

      REAL*8 UO(12),TOL,TYPX(12),H(12),F(12),PHI,U(12)
      REAL*8 X,Y,Z,ALPHA,BETA,GAMMA
      REAL*8 FJAC(12,12),PDIR(12),LAMDAK
      INTEGER N,MAXK,RMIN,KITR,IFLAG,TIME

*************************************************************************

C TO DEFINE THE VALUES OF ALPHA, BETA, AND GAMMA.
      ALPHA = 1.0
      BETA =  1.0
      GAMMA = 1.0

C TO DEFINE THE NUMBER OF EQUATIONS TO BE SOLVED.
      N = 3

C TO READ THE COORDINATES OF THE TIP OF THE BEAM.
      WRITE(6,*)'ENTER THE POINT COORDINATES'

C TO READ IN THE INITIAL GUESSES FOR THE LINK LENGTHS.
      WRITE(6,*)'ENTER THE INITIAL GUESSES FOR THE LINK LENGTHS.'
      UO(1) = 45.
      UO(2) = 45.
      UO(3) = 45.

C TO DEFINE THE MAXIMUM NUMBER OF ITERATIONS TO BE PERFORMED.
      MAXK = 100

C TO DEFINE THE TOLERANCE LIMIT.
      TOL = 0.001

C TO DEFINE THE TYPICAL VALUE OF XJ.
      TYPX(1) = 46.5
      TYPX(2) = 46.5
      TYPX(3) = 46.5

C TO DEFINE THE MINIMUM NUMBER OF LINE SEARCH STEPS'
      RMIN = 100

      KITR = 1

32    DO 19 J = 1,N
      IF(UO(J).GE.0.0D0)THEN
      IFLAG = 1
      ELSE
      IFLAG = -1
      ENDIF

      H(J) = 0.0D0
19    H(J)= 0.02D0*(DMAX1(DABS(UO(J)),TYPX(J)))*IFLAG

      LAMDAK = 1.0D0

      CALL FCN(UO,N,F)
      CALL EVLPHI(F,N,PHI)
      CALL JACOB(H,N,UO,F,FJAC)
      CALL NEWTDR(FJAC,N,F,PDIR)
```

```
      CALL LINESR(RMIN,N,F,PHI,FJAC,UO,U,PDIR,LAMDAK)

      DO 15 I =1,N
15    UO(I) = U(I)

      CALL FCN(UO,N,F)
      CALL EVLPHI(F,N,PHI)
      IF(PHI.LT.TOL)GOTO 31
      KITR = KITR + 1
      IF(KITR.GT.MAXK)GOTO 31

      IF((UO(1)+UO(2)).LT.UO(3))GOTO 35
      IF((UO(2)+UO(3)).LT.UO(1))GOTO 35
      IF((UO(3)+UO(1)).LT.UO(2))GOTO 35
      IF(UO(1).LT.0.0)GOTO 35
      IF(UO(2).LT.0.0)GOTO 35
      IF(UO(3).LT.0.0)GOTO 35
      IF(UO(1).GT.51.0.OR.UO(1).LT.39.)GOTO 36
      IF(UO(2).GT.51.0.OR.UO(2).LT.39.)GOTO 36
      IF(UO(3).GT.51.0.OR.UO(3).LT.39.)GOTO 36

      GOTO 32

36    CONTINUE
      WRITE(8,*)'THIS LINK LENGTH IS BEYOND LENGTH LIMIT FOR V.G.T.'

35    CONTINUE
      WRITE(8,*)'TRY AGAIN THIS POINT IS OUT OF THE WORKSPACE'

31    CONTINUE

C TO WRITE OUT THE SOLUTION OF THE INVERSE PROBLEM.
      WRITE(8,30)U(1),U(2),U(3)
30    FORMAT(F5.2,F5.2,F5.2)

20    CONTINUE

      STOP
      END

C-------------------------------------------------------------------------
C THIS SUBROUTINE FORMS THE THREE FUNCTIONS F1, F2, AND F3.
C-------------------------------------------------------------------------

      SUBROUTINE  FCN(U,N,F)

      COMMON/GUESS1/X, Y, Z,ALPHA,BETA,GAMMA

      REAL*8 U(12),F(12),ALPHA,BETA,GAMMA,XYZ1(3),XYZ2(3),X,Y,Z
      REAL*8 LEN1,LEN2,LEN3,ALPHA1,BETA1,GAMMA1
      INTEGER N

C CALLING THE NEWTON ROUTINE FOR CALCULATION OF THE COORDINATES.
      IKFL = 0

      CALL VGTRUS(U(1),U(2),U(3),ALPHA,BETA,GAMMA,XYZ1,IKFL)
      IKFL = 1

C TO DEFINE THE VARIABLES FOR DRAWING THE SECOND MODULE.
      ALPHA1 = 1.055899
      BETA1 = 1.055899
      GAMMA1 = 1.055899

      LEN1 = 46.5
      LEN2 = 46.5
      LEN3 = 46.5
```

```
      CALL VGTRUS(LEN1,LEN2,LEN3,ALPHA1,BETA1,GAMMA1,XYZ2,IKFL)

      DO 1 I =1,N
1     F(I) = 0.0D0

C FORMING THE THREE FUNCTIONS.
      F(1)= XYZ2(1) - X
      F(2)= XYZ2(2) - Y
      F(3)= XYZ2(3) - Z

      RETURN
      END

C-----------------------------------------------------------------------
C THIS SUBROUTINE FORMS PHI USING FUNCTION VALUES.
C-----------------------------------------------------------------------

      SUBROUTINE EVLPHI(F,N,PHI)

      REAL*8 F,PHI
      INTEGER N

      PHI = 0.0D0

      DO 2 I = 1,N
2     PHI = PHI + 0.5D0 * F(I) * F(I)

      RETURN
      END

C-----------------------------------------------------------------------
C THIS SUBROUTINE CALCULATES THE JACOBIAN MATRIX.
C-----------------------------------------------------------------------

      SUBROUTINE JACOB(H,N,UO,F,FJAC)

      REAL*8 UO(12),FJAC(12,12),H(12),UR(12),FR(12),F(12)
      INTEGER N

      DO 1 I = 1,N
      DO 2 J = 1,N
2     UR(J) = UO(J)
      UR(I) =  UO(I) + H(I)
      CALL FCN(UR,N,FR)
      DO 1 K = 1,N
1     FJAC(K,I) = (FR(K) - F(K)) / H(I)

      RETURN
      END

C-----------------------------------------------------------------------
C SUBROUTINE GAUSS SOLVES THE SET OF EQUATIONS USING THE GAUSS
C ELIMINATION METHOD.
C-----------------------------------------------------------------------

      SUBROUTINE GAUSS(A,B,X,N,MAINDM,IERROR,RNORM)

      REAL*8 A(MAINDM,MAINDM),B(MAINDM),X(MAINDM),AUG(50,51),RSQ
      REAL*8 RNORM,RESI

      NM1 = N - 1
      NP1 = N + 1

C SETUP AUGMENTED MATRIX FOR AX = B
      DO 2 I = 1,N
      DO 1 J = 1,N
```

```
      AUG(I,J) = A(I,J)
1     CONTINUE
      AUG(I,NP1) = B(I)
2     CONTINUE

C THE OUTER LOOP USES ELEMENTARY ROW OPERATIONS TO TRANSFORM THE
C AUGMENTED MATRIX TO ECHELON FORM.

      DO 8 I = 1,NM1

C SEARCH FOR THE LARGEST ENTRY IN THE COLUMN1, ROWS I THROUGH N.

C IPIVOT IS THE ROW INDEX OF THE LARGEST ENTRY.
      PIVOT = 0.0D0

      DO 3 J = I,N
      TEMP = ABS(AUG(J,I))
      IF(PIVOT.GE.TEMP)GOTO 3
      PIVOT = TEMP
      IPIVOT = J
3     CONTINUE
      IF(PIVOT.EQ.0.0D0)GOTO 13
      IF(IPIVOT.EQ.I)GOTO 5

C INTERCHANGE ROW I AND IPIVOT

      DO 4 K = I,NP1
      TEMP = AUG(I,K)
      AUG(I,K) = AUG(IPIVOT,K)
      AUG(IPIVOT,K) = TEMP

4     CONTINUE

C ZERO ENTRIES (I+1,I),(I+2,I),.....,(N,I) IN THE AUGMENTED MATRIX.
5     IP1 = I + 1
      DO 7 K = IP1,N
      Q = -AUG(K,I)/AUG(I,I)
      AUG(K,I) = 0.0D0
      DO 6 J = IP1,NP1
      AUG(K,J) = Q * AUG(I,J) + AUG(K,J)
6     CONTINUE
7     CONTINUE
8     CONTINUE
      IF(AUG(N,N).EQ.0.0D0)GOTO 13

C BACK SOLVE TO OBTAIN A SOLUTION TO AX = B
      X(N) = AUG(N,NP1)/AUG(N,N)
      DO 10 K = 1,NM1
      Q = 0.0D0
      DO 9 J = 1,K
      Q = Q + AUG(N-K,NP1-J)*X(NP1-J)
9     CONTINUE
      X(N-K) = (AUG(N-K,NP1) - Q )/AUG(N-K,N-K)
10    CONTINUE

C CALCULATE THE NORM OF THE RESIDUAL VECTOR, B - AX.
C SET IERROR = 1 AND RETURN.
      RSQ = 0.0D0
      DO 12 I = 1,N
      Q = 0.0D0
      DO 11 J = 1,N
      Q = Q + A(I,J) * X(J)
11    CONTINUE
      RESI = B(I) - Q
      RMAG = DABS(RESI)
      RSQ = RSQ + RMAG**2
12    CONTINUE
```

```
      RNORM = DSQRT(RSQ)
      IERROR = 1

      RETURN

C ABNORMAL RETURN IF THE MATRIX A IS SINGULAR.
13    IERROR = 2

      RETURN
      END

C-----------------------------------------------------------------------
C THIS SUBROUTINE CALCULATES THE NEWTON DIRECTION.
C-----------------------------------------------------------------------

      SUBROUTINE NEWTDR(FJAC,N,F,PDIR)

      REAL*8 FJAC(12,12),F(12),PDIR(12),F1(12),RNORM
      INTEGER N,IERROR

      DO 1 I = 1,N
1     F1(I) = -1.0D0 * F(I)
      CALL GAUSS(FJAC,F1,PDIR,N,12,IERROR,RNORM)

      RETURN
      END


C-----------------------------------------------------------------------
C SUBROUTINE GRDPHI
C-----------------------------------------------------------------------

      SUBROUTINE GRDPHI(F,FJAC,N,DELPHI)

      REAL*8 F(12),DELPHI(12),FJAC(12,12)
      INTEGER N

      DO 1 J = 1,N
1     DELPHI(J) = 0.0D0
      DO 2 J = 1,N
      DO 3 I = 1,N
      DELPHI(J) = DELPHI(J) + F(I) * FJAC(I,J)
3     CONTINUE
2     CONTINUE

      RETURN
      END

C-----------------------------------------------------------------------
C SUBROUTINE LINESEARCH.
C-----------------------------------------------------------------------

      SUBROUTINE LINESR(RMIN,N,F,PHI,FJAC,U0,U2,PDIR,LAMDAK)

      REAL*8 U0(12),LAMDAK,PDIR(12),U2(12),PHI,DELPH,R
      REAL*8 F1(12),FT1(12),PHI1,DELPHI(12),LAMDA,FJAC(12,12)

      INTEGER N,IND

      PHI1 = 0.0D0
12    R = LAMDAK
      DO 1 I = 1,N
      F1(I) = 0.0D0
      U2(I) = 0.0D0
1     U2(I) = U0(I) + R * PDIR(I)
      CALL FCN(U2,N,F1)
      CALL EVLPHI(F1,N,PHI1)
```

```
      CALL GRDPHI(F,FJAC,N,DELPHI)
      DOT = 0.0D0
      DO 10 I = 1,N
10    DOT = DOT + DELPHI(I) * PDIR(I)
      DOT1 = PHI + 0.0001D0 * R * DOT
      IF(PHI1.LT.DOT1)GOTO 14
      IND = IND + 1
      IF(IND.GT.RMIN)GOTO 14
      LAMDA = -0.5D0*DOT*R**2/(PHI1 - PHI - R*DOT)
      LAMDAK = DMAX1(LAMDA,R/10.0D0)
      GOTO 12

14    RETURN
      END

C-----------------------------------------------------------------------
C SUBROUTINE VGTRUS CALCULATES COORDINATES OF THE JOINTS OF THE VGT.
C-----------------------------------------------------------------------

      SUBROUTINE VGTRUS(AL1,AL2,AL3,ALPHA,BETA,GAMMA,XYZ,IKFL)

      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/TRIAL/UA(3),UB(3),UC(3),A0(3),B0(3),C0(3),
     &             A1MA0(3),B1MB0(3),C1MC0(3),A1B1MG,A1C1MG,C1B1MG

      COMMON PI

      DIMENSION AJ(3),BJ(3),CJ(3)
      DIMENSION PM(3,3),QM(3,3),QIM(3,3),RM(3,3,2)
      DIMENSION A(3,3),B(3)
      DIMENSION ANORM1(3),ANORM2(3),ANORM3(3),ANORM4(3)
      DIMENSION BJMAJ(3),CJMAJ(3),CJMBJ(3)
      DIMENSION A0MAJ(3),B0MAJ(3),C0MBJ(3),AJMA0(3)
      DIMENSION AJPCJ(3),AJPBJ(3),BJPCJ(3)
      DIMENSION A0MACJ(3),B0MABJ(3),C0MBCJ(3)
      DIMENSION ANODE7(3),ANODE8(3),ANODE9(3)
      DIMENSION UD(3),UE(3),UF(3)
      DIMENSION UAT(3),UBT(3),UN(3),UZ(3),TRAN(3,3),D(3)
      DIMENSION XYZ(3),XYZ1(3),XYZ2(3),XYZ3(3)

C  SQUARING THE LINK LENGTHS FOR EASE OF CALCULATIONS.
      A1B1MG = AL1 * AL1
      A1C1MG = AL2 * AL2
      C1B1MG = AL3 * AL3

C DEFINING THE CONSTANTS OF THE PROGRAM.

C THE TOLERANCE LIMIT, FOR NEWTON-RAPHSON METHOD.
      TOL = 0.0000001

C DEFINING THE VALUE OF THE TRIGONOMETRIC CONSTANT PI.
      PI = 3.141592654

C REDEFINING THE POSITION OF REVOLUTE JOINT A0.
      A0(1)= 0.0
      A0(2)= 0.0
      A0(3)= 0.0

C DEFINING THE POSITION OF REVOLUTE JOINT B0.
      B0(1)=23.25
      B0(2)=0.0
      B0(3)=40.27018128

C DEFINING THE POSITION OF REVOLUTE JOINT C0.
      C0(1)=46.5
      C0(2)=0.0
      C0(3)=0.0
```

```
C  THE POSITION OF VECTOR (A1 - A0).
      A1MA0(1) = -10.45864780
      A1MA0(2) =   0.0
      A1MA0(3) =  32.8850964

C  THE POSITION OF VECTOR (B1 - B0).
      B1MB0(1) =  33.7086478
      B1MB0(2) =   0.0
      B1MB0(3) =  -7.38509064

C  THE POSITION OF VECTOR (C1 - C0).
      C1MC0(1) = -23.25
      C1MC0(2) =   0.0
      C1MC0(3) = -25.5

      IF(IKFL.EQ.1)GOTO 1111

C  THE UNIT VECTOR ASSOCIATED WITH A0.
      UA(1) = -0.5
      UA(2) =  0.0
      UA(3) = -0.86602543

C  THE UNIT VECTOR ASSOCIATED WITH B0.
      UB(1) = -0.5
      UB(2) =  0.0
      UB(3) =  0.86602543

C  THE UNIT VECTOR ASSOCIATED WITH C0.
      UC(1) =  1.0
      UC(2) =  0.0
      UC(3) =  0.0
1111  CONTINUE

C CALLING THE NEWTON ROUTINE FOR SOLVING FOR VALUES OF THE ANGLES.
      CALL NEWTON (ALPHA,BETA,GAMMA,TOL,K)

C TO FIND COORDINATES OF THE VERTEX AJ.
      CALL RMAXIS (UA,ALPHA,RM,2)
      CALL ROTATE (AJ,A0,RM,A1MA0,2)

C TO FIND COORDINATES OF THE VERTEX BJ.
      CALL RMAXIS (UB,BETA,RM,2)
      CALL ROTATE (BJ,B0,RM,B1MB0,2)

C TO FIND COORDINATES OF THE VERTEX CJ.
      CALL RMAXIS (UC,GAMMA,RM,2)
      CALL ROTATE (CJ,C0,RM,C1MC0,2)

C TO CALCULATE NORMAL TO THE LATERAL VARIABLE PLANE.
      DO 7 I = 1,3
      BJMAJ(I) = BJ(I) - AJ(I)
7     CJMBJ(I) = CJ(I) - BJ(I)
      CALL CROSS (ANORM1,BJMAJ,CJMBJ)

C-----------------------------------------------------------------------
C
C     COORDINATES OF NODE 7 ARE CALCULATED IN THE FOLLOWING BLOCK.
C
C-----------------------------------------------------------------------

      DO 8 I = 1,3
      CJMAJ(I) = CJ(I) - AJ(I)
   8  AJMA0(I) = AJ(I) - A0(I)
      CALL CROSS (ANORM2,AJMA0,CJMAJ)

C TO CALCULATE THE ANGLE BETWEEN THE LATERAL AND SIDE PLANES.
```

```
      ANUM1 = DOT (ANORM1,ANORM2)
      DEN1 = DABS(DSQRT (ANORM1(1)**2 + ANORM1(2)**2 + ANORM1(3)**2)) *
     $       DABS(DSQRT (ANORM2(1)**2 + ANORM2(2)**2 + ANORM2(3)**2))
      THETA = DABS(DACOS (ANUM1/DEN1))
      IF(THETA.GT.PI)THETA = 2.*PI - THETA
      THETA = 2. * THETA

      DO 9 I = 1,3
    9 AOMACJ(I) = AO(I) - AJ(I)

C TO FIND UNIT VECTOR ALONG THE LINK CJAJ.
      DO 10 I = 1,3
   10 UD(I) = CJMAJ(I)/(DSQRT(CJMAJ(1)**2 + CJMAJ(2)**2 + CJMAJ(3)**2))
      CALL RMAXIS (UD,THETA,RM,2)
      CALL ROTATE (ANODE7,AJ,RM,AOMACJ,2)
C
C-----------------------------------------------------------------------
C
C     COORDINATES OF NODE 8 ARE CALCULATED IN THE FOLLOWING BLOCK.
C
C-----------------------------------------------------------------------
C
      DO 12 I = 1,3
   12 BOMAJ(I) = BO(I) - AJ(I)
      CALL CROSS (ANORM3,BOMAJ,BJMAJ)

      ANUM2 = DOT (ANORM1,ANORM3)
      DEN2 = DABS(DSQRT (ANORM1(1)**2 + ANORM1(2)**2 + ANORM1(3)**2)) *
     $       DABS(DSQRT (ANORM3(1)**2 + ANORM3(2)**2 + ANORM3(3)**2))
      THETA1 = DABS(DACOS (ANUM2/DEN2))
      IF(THETA1.GT.PI)THETA1 = 2.* PI - THETA1
      THETA1 = 2. * THETA1

      DO 13 I = 1,3
   13 BOMABJ(I) = BO(I) - BJ(I)

C TO FIND UNIT VECTOR ALONG THE LINK BJAJ.
      DO 14 I =1,3
   14 UE(I) = -BJMAJ(I)/(DSQRT(BJMAJ(1)**2 + BJMAJ(2)**2 + BJMAJ(3)**2))
      CALL RMAXIS (UE,THETA1,RM,2)
      CALL ROTATE (ANODE8,BJ,RM,BOMABJ,2)
C
C-----------------------------------------------------------------------
C
C     COORDINATES OF NODE 9 ARE CALCULATED IN THE FOLLOWING BLOCK.
C
C-----------------------------------------------------------------------
C
      DO 16 I = 1,3
   16 COMBJ(I) = CO(I) - BJ(I)
      CALL CROSS (ANORM4,COMBJ,CJMBJ)

      ANUM3 = DOT (ANORM1,ANORM4)
      DEN3 = DABS(DSQRT (ANORM1(1)**2 + ANORM1(2)**2 + ANORM1(3)**2)) *
     $       DABS(DSQRT (ANORM4(1)**2 + ANORM4(2)**2 + ANORM4(3)**2))
      THETA2 = DABS(DACOS (ANUM3/DEN3))
      IF(THETA2.GT.PI)THETA2 = 2.*PI - THETA2
      THETA2 = 2. * THETA2

      DO 17 I = 1,3
   17 COMBCJ(I) = CO(I) - CJ(I)

C TO FIND UNIT VECTOR ALONG THE LINK CJBJ.
      DO 18 I =1,3
   18 UF(I) = -CJMBJ(I)/(DSQRT(CJMBJ(1)**2 + CJMBJ(2)**2 + CJMBJ(3)**2))
      CALL RMAXIS (UF,THETA2,RM,2)
      CALL ROTATE (ANODE9,CJ,RM,COMBCJ,2)
```

```
C CALL TO CHECK THE IKFL.
      IF(IKFL.EQ.0)GOTO 31

C TO CALCULATE THE UNIT VECTORS .
      DO 49 I = 1,3
      UAT(I) = (XYZ1(I) - XYZ3(I))/46.5
      UBT(I) = (XYZ2(I) - XYZ1(I))/46.5
49    CONTINUE

C TO CALL THE CROSS SUBROUTINE TO CALCULATE THE NORMAL.
      CALL CROSS(UN,UAT,UBT)
      UN(2) = UN(2)/SIN(PI/3.)

C CALL THE CROSS SUBROUTINE TO CALCULATE THE UNIT VECTOR ALONG THE Z AX.
      CALL CROSS(UZ,UN,UAT)

C TO DEF. THE ROTN. MAT.
      TRAN(1,1) = -UAT(1)
      TRAN(2,1) = -UAT(2)
      TRAN(3,1) = -UAT(3)
      TRAN(1,2) = UN(1)
      TRAN(2,2) = UN(2)
      TRAN(3,2) = UN(3)
      TRAN(1,3) = UZ(1)
      TRAN(2,3) = UZ(2)
      TRAN(3,3) = UZ(3)

C TO DEF. THE TRANS. VECTOR.
      D(1) = XYZ1(1) - A0(1)
      D(2) = XYZ1(2) - A0(2)
      D(3) = XYZ1(3) - A0(3)

C CALLING THE MATRIX MULTIPLICATION SUBROUTINE.
      XYZ(1) = (ANODE7(1) + ANODE8(1) + ANODE9(1))/3.0
      XYZ(2) = (ANODE7(2) + ANODE8(2) + ANODE9(2))/3.0
      XYZ(3) = (ANODE7(3) + ANODE8(3) + ANODE9(3))/3.0
      CALL MATMUL(TRAN,XYZ,D)
      XYZ(1) = XYZ(1) + 80. * UN(1)
      XYZ(2) = XYZ(2) + 80. * UN(2)
      XYZ(3) = XYZ(3) + 80. * UN(3)

31    CONTINUE

      DO 999 I =1,3
      XYZ1(I) = ANODE7(I)
      XYZ2(I) = ANODE8(I)
      XYZ3(I) = ANODE9(I)
999   CONTINUE

      RETURN
      END
```

The vita has been removed from the scanned document