Three-dimensional computerized model of an elastic robotic arm

Allison Jeanne Krueger
University of Nevada, Las Vegas

Three-dimensional computerized model of an elastic robotic arm

Krueger, Allison Jeanne, M.S.

University of Nevada, Las Vegas, 1989

THREE DIMENSIONAL COMPUTERIZED MODEL OF AN ELASTIC ROBOTIC ARM

by

Allison Jeanne Krueger

A thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Science

in

Mechanical Engineering

Department of Civil and Mechanical Engineering

University of Nevada, Las Vegas

August, 1989

The thesis of Allison Jeanne Krueger for the degree of Masters of Science in Mechanical Engineering is approved.

[illegible]

Prof. William Culbreth, Ph.D., Chairperson

[illegible]

Prof. Samaan Ladkany, Ph.D., Examining Committee Member

[illegible]

Prof. Mohammed Trabia, Ph.D., Examining Committee Member

[illegible]

Prof. Sahjendra Singh, Ph.D., Examining Committee Member

[illegible]

Prof. Ronald Smith, Ph.D., Graduate College Dean

August, 1989

ABSTRACT

Interactive computer simulation software in the area of robotics is becoming increasingly important for the design of new robots and trajectory planning. The present work involved the creation of a computer simulation software package for an elastic robotic arm. The simulation used a three—link robotic arm controlled by two hydraulic actuators. The computer simulation was unique in four major areas. Using a specialized Silicon Graphics IRIS Workstation, a three—dimensional model of the three—link elastic robotic arm was created. Traditionally, due to the lack of memory and speed in available computers, solid three—dimensional robotic models have been fairly simple and involved static wireframe renditions of the robot. The software simulation developed in the present work was highly interactive with the user. The user is easily able to move the different links, change parameters, and alter dynamic applied forces. Since the second and third links of the arm were elastic, the forces and torques applied to the arm had a definite effect in the form of deformations. The deformations of the robotic arm were modelled and presented on a computer monitor. The kinematics were modelled. Kinematic alterations were possible by user interaction. The user was allowed to adjust the angles of the links by either moving the different links or by changing the magnitude of the forces in the two actuators. By addressing each of these four topics, a more complete computer simulation package was designed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THEORY NOMENCLATURE

A — Cross—sectional area of actuator

${}^{i-1}A_i$ — Coordinate Transformation Matrix

$\vec{a}_0$ — Acceleration of the point Pin #1 = 0.0

dC_1 — Differential Change in Length of Actuator AB

dC_2 — Differential Change in Length of Actuator DE

E — Modulus of elasticity of actuator.

E_i — Modulus of elasticity of segment i

F — Axial load

F^d_{ab} — Dynamic force of actuator ab

F^a_{ab} — Applied force of actuator ab

F^s_{ab} — Static force of actuator ab

F^d_{de} — Dynamic force of actuator de

F^a_{de} — Applied force of actuator de

F^s_{de} — Static force of actuator de.

G_i — The shear modulus of elasticity.

I_{ij}— Moment of Inertia for link i about the j axis

J_i — Polar moment of inertia.

L — level number

L_i — Distance on segment i from Pin #i to Pin #(i+1)

L_1 — P_{12}, the distance between Pins #1 & #2.

L_2 — P_{23}, the distance between Pins #2 & #3.

$M^{l}_{(i+1)y_i}$ — is the moment reaction in the local y_i—axis.

$$M^{l}_{(i+1)y_i} = M^{s}_{(i+1)y}\sin\phi_i + M^{s}_{(i+1)x}\cos\phi_i$$

M^{d}_{ij} — Dynamic moment reaction at Pin #i in the j direction

M^{s}_{ij} — Static moment reaction at Pin #i in the j direction

Q^{d}_{gij} — Inertial force reaction at Pin #i in the j direction

q^{o}_{iji} — Distributed forces per segment i length due to inertial torques in the j direction applied at the Center of Gravity

q_{iji} — Distributed forces per segment i length due to inertial forces in the j direction applied at the Center of Gravity

P — Position Vector

P_{ij} — Distance from Pin #i to Pin #j

P_{igi} — Distance from center of gravity of link i to Pin #i

R — Rotation Matrix

$R^{l}_{(i+1)y_i}$ — The force reaction in the local y_i—axis.

$$R^{l}_{(i+1)y} = R^{s}_{(i+1)y}\cos\phi_i - R^{s}_{(i+1)x}\sin\phi_i.$$

R^{d}_{ij} — Dynamic force reaction at Pin #i in the j direction

R^{s}_{ij} — Static force reaction at Pin #i in the j direction

$r_{2/1}$ — Vector from Pin #1 to Pin #2

$\vec{r}_{3/2}$ — Vector from Pin #2 to Pin #3

T — Homogeneous Transformation Matrix

T^{d}_{gij} — Inertial moment reaction at Pin #i in the j direction

u_{x1} — Axial extension for Segment #1

u_{x2} — Axial extension for Segment #2

V_i — Displacement of segment i in the y direction

V_0 — Velocity of Base, = 0.0

$\vec{V}_1$ — Velocity of Segment #1 at Pin #2

W_i — Displacement of segment i in the z direction

x_i — Linear distance along the x axis where deflection calculations are determined

α_{ij} — Angular acceleration at Pin #i in the j direction

δ — Change in length due to torsional twist

δT — Time step

ϕ_i — Angle i

θ_{x1} — Torsional twist about the x axis for Segment #1

θ_{x2} — Torsional twist about the x axis for Segment #2

θ_{zi} — Slope of displacement for segment i about the z axis

ϑ_{yi} — Slope of displacement for segment i about the y axis

Ω — Angular velocity of Base, ω_0

$\vec{\dot{\Omega}}$ — Angular Acceleration of Base

$\vec{\Omega}_{xyz}$ — The Sum of the Angular Velocity of the Base and Segment #1

ω_{ij} — Angular velocity at Pin #i in the j direction

$\omega_x - \omega_{1x}$

$\omega_y - \omega_{oy} + \omega_{1y}$

$\omega_z - \omega_{1z}$

PROGRAM NOMENCLATURE

AC1 = ACcelerations of partitions of segment #1

AC2 = ACcelerations of partitions of segment #2

AD = ADjust a certain segment

ALPHA = angular accelerations

ANGINC = INCremental value for ANGles

ANGX = ANGle of rotation of the world about X—axis

ANGY = ANGle of rotation of the world about Y—axis

ANGZ = ANGle of rotation of the world about Z—axis

ANG0CNT = ANGle CouNT of the base

ANG1CNT = ANGle CouNT of segment #1

ANG1MN = ANGle #1 MiNimum value

ANG1MX = ANGle #1 MaXimum value

ANG2CNT = ANGle CouNT of segment #2

ANG2MN = ANGle #2 MiNimum value

ANG2MX = ANGle #2 MaXimum value

ANG3CNT = ANGle CouNT of segment #3

ANG3MN = ANGle #3 MiNimum value

ANG3MX = ANGle #3 MaXimum value

ANG4CNT = ANGle CouNT of actuator ab

ANG5CNT = ANGle CouNT of actuator de

ANG6CNT = ANGle CouNT of wing segment

BASEDI = BASE DIameter

BASEHE	= BASE HEight
BASMAS	= BASe MASs
BCMASX	= Base Center of MASs, X—coordinate
BCMASY	= Base Center of MASs, Y—coordinate
BMIX	= Base mass moment of inertia about the X—axis
BMIY	= Base mass moment of inertia about the Y—axis
BMIZ	= Base mass moment of inertia about the Z—axis
BS	= BaSe
BSDT	= BaSe Deformed and Transformed
DPHI1	= differential change in angle one
DPHI2	= differential change in angle two
DT	= Differential Time step
FACTI	= Forces of ACTuators from Input
FABD	= Force of actuator AB, Dynamic
FABS	= Force of actuator AB, Static
FDED	= Force of actuator DE, Dynamic
FDES	= Force of actuator DE, Static
GAMMA	= angle between segment #2 and wing
H1A	= Hydraulic actuator #1 cross—sectional area
H1E	= Hydraulic actuator #1 modulus of Elasticity
H1L	= Hydraulic actuator #1 flag
H1LENG	= Hydraulic actuator #1 LENGth
H1MASS	= Hydraulic actuator #1 MASS
H1MAXL	= Hydraulic actuator #1 MAXimum Length
H1MINL	= Hydraulic actuator #1 MINimum Length
H1THIC	= Hydraulic actuator #1 THICkness
H2A	= Hydraulic actuator #2 cross—sectional area

H2E = Hydraulic actuator #2 modulus of Elasticity

H2L = Hydraulic actuator #2 flag

H2LENG = Hydraulic actuator #2 LENGth

H2MASS = Hydraulic actuator #2 MASS

H2MAXL = Hydraulic actuator #2 MAXimum Length

H2MINL = Hydraulic actuator #2 MINimum Length

H2THIC = Hydraulic actuator #2 THICkness

MD = Moments, Dynamic

MS = Moments, Static

NFACT = Number of current dynamic entry of Forces of ACTuators

NF = Number of input Forces

OMEGA = angular velocities

PB = Pins on Base

PBDT = Pins on Base Deformed and Transformed

PINAX = PIN A on the base, X—coordinate

PINAY = PIN A on the base, Y—coordinate

PINBX = PIN B on segment #1, X—coordinate

PINBY = PIN B on segment #1, Y—coordinate

PINDX = PIN D on segment #1, X—coordinate

PINDY = PIN D on segment #1, Y—coordinate

PINEX = PIN E on the wing segment, X—coordinate

PINEY = PIN E on the wing segment, Y—coordinate

PIN1X = PIN #1, X—coordinate

PIN1Y = PIN #1, Y—coordinate

PIN2X = Pin #2 on Segment #1, X—coordinate

PIN2Y = Pin #2 on Segment #1, Y—coordinate

PIN3X = Pin #3 on Segment #2, X—coordinate

PIN3Y = Pin #3 on Segment #2, Y–coordinate

PNE = adjustment is Postive, Negative, or an Error

PW = Pins on Wing segment

PWD = Pins on Wing segment Deformed

PWDT = Pins on Wing segment Deformed and Transformed

P1 = Pins on Segment #1

P1D = Pins on Segment #1 Deformed

P1DT = Pins on Segment #1 Deformed and Transformed

P2 = Pins on Segment #2

P2D = Pins on Segment #2 Deformed

P2DT = Pins on Segment #2 Deformed and Transformed

QD = Dynamic forces, f=ma

RD = forces, Dynamic Reactions

RS = forces, Static Reactions

S1 = Segment #1

S1A = Segment #1 cross–sectional Area

S1CMAX = Segment #1 Center of MAss, X–coordinate

S1CMAY = Segment #1 Center of MAss, Y–coordinate

S1D = Segment #1 Deformed

S1DE = Segment #1 DEflection

S1DIVN = Number of DIVisions Segment #1 is partitioned into

S1DM = Segment #1 Deflection Multiplier

S1DT = Segment #1 Deformed and Transformed

S1E = Segment #1 modulus of Elasticity

S1G = Segment #1 modulus of rigidity

S1IX = Segment #1 area moment of inertia about the X–axis

S1IY = Segment #1 area moment of inertia about the Y–axis

S1IZ = Segment #1 area moment of inertia about the Z—axis

S1JO = Segment #1 area polar moment of inertia

S1LENG = Segment #1 LENGth

S1MASS = Segment #1 MASS

S1MIX = Segment #1 mass moment of inertia about the X—axis

S1MIY = Segment #1 mass moment of inertia about the Y—axis

S1MIZ = Segment #1 mass moment of inertia about the Z—axis

S1P1X = Pin #1 on Segment #1, X—coordinate

S1P1Y = Pin #1 on Segment #1, Y—coordinate

S1RF = Segment #1 Undeformed and Transformed

S1SL = Segment #1 SLope of deflection

S1THIC = Segment #1 THICkness

S2 = Segment #2

S2A = Segment #2 cross—sectional Area

S2CMAX = Segment #2 Center of MAss, X—coordinate

S2CMAY = Segment #2 Center of MAss, Y—coordinate

S2D = Segment #2 Deformed

S2DE = Segment #2 DEflection

S2DIVN = Number of DIVisions Segment #2 is partitioned into

S2DM = Segment #2 Deflection Multiplier

S2DT = Segment #2 Deformed and Transformed

S2E = Segment #2 modulus of Elasticity

S2G = Segment #2 modulus of rigidity

S2IX = Segment #2 area moment of inertia about the X—axis

S2IY = Segment #2 area moment of inertia about the Y—axis

S2IZ = Segment #2 area moment of inertia about the Z—axis

S2JO = Segment #2 polar moment of inertia

S2LENG = Segment #2 LENGth

S2MASS = Segment #2 MASS

S2MIX = Segment #2 mass moment of inertia about the X—axis

S2MIY = Segment #2 mass moment of inertia about the Y—axis

S2MIZ = Segment #2 mass moment of inertia about the Z—axis

S2P2X = Pin #2 on Segment #2, X—coordinate

S2P2Y = Pin #2 on Segment #2, Y—coordinate

S2RF = Segment #2 Undeformed and Transformed

S2SL = Segment #2 SLope of deflection

S2THIC = Segment #2 THICkness

S3 = Segment #3

S3CMAX = Segment #3 Center of MAss, X—coordinate

S3CMAY = Segment #3 Center of MAss, Y—coordinate

S3D = Segment #3 Deformed

S3DT = Segment #3 Deformed and Transformed

S3LENG = Segment #3 LENGth

S3MASS = Segment #3 MASS

S3P3X = Pin #3 on Segment #3, X—coordinate

S3P3Y = Pin #3 on Segment #3, Y—coordinate

S3RF = Segment #3 Undeformed and Transformed

S3THIC = Segment #3 THICkness

S4CMAX = Segment #4 (load) Center of MAss, X—coordinate

S4CMAY = Segment #4 (load) Center of MAss, Y—coordinate

S4LENG = Segment #4 (load) LENGth

S4MASS = Segment #4 (load) MASS

S4THIC = Segment #4 (load) THICkness

TD = Dynamic Torques, T=I*alpha

VE1 = VElocities of partitions of segment #1

VE2 = VElocities of partitions of segment #2

WING = Wing segment

WINGD = Wing segment Deformed

WINGDT = Wing segment Deformed and Transformed

WINGRF = Wing segment Undeformed and Transformed

WLENG = LENGth of Wing segment

CHAPTER 1
INTRODUCTION

In July, 1987 the University of Nevada, Las Vegas, Department of Civil and Mechanical Enginerring received a research grant from the United States Army to model the dynamic behavior of elastic robot arms.

An elastic robotic arm differs from a rigid robotic arm in both construction application. Rigid segments are typically used to prevent unwanted deformation and oscillations. They also allow an accurate determination to be made of the position of the end—effector through angular encoders located at the joints. In construction, rigid robotic arms have been made out of thick steel or other similar materials. It is common to have a weight—to—load ratio of at least ten to one for rigid robots making them heavy and difficult to move. The operational environment of the rigid robotic arm is not variable since the robot is usually bolted to the floor of a specific work area. This creates a very limited range of applications based on the lack of mobility. An elastic robotic arm is designed to be manufactured out of lightweight material. By building the arm out of lightweight high—strength materials the arm may be made mobile and still highly functional. With the arm constructed out of a more flexible material, new problems arise. Two of the most prevalent problems in elastic robotic arms are oscillatory mode shapes and three—dimensional deformation.

The ARO research grant specified five areas of study: (1) Sensory Perception, (2) Robot Control, (3) Structural Analysis, (4) Elastic Robot Simulation, and (5) System Integration and Evaluation of System Performance under various robot and sensor configurations.

The sensory perception research involved developing sensors to detect deformations, oscillatory frequencies, and mode shapes. The procedures developed are applicable to both elastic and rigid robotic arms. Types of sensors studied were vision, ultrasound range, and end—effector mounted force/torque sensor systems. The output of the sensors served as input for the controller.

Robot control was researched with the following goals: (1) system observation with minimal sensors, (2) optimal performance based on available sensor input, and (3) maximum accuracy and speed while keeping costs down. The goals were to be reached by developing control algorithms and keeping such points as sensor input, costs, and system requirements in view.

Structural Analysis research involved the utilization of high—strength lightweight materials. The structure was required to meet the payload and workspace requirements. Also, the structural design was required to meet the minimization and decoupling requirements of oscillatory mode forms and deformation effects. A successful structural design also improves state controllability. To aid in meeting the structural design requirements, finite element analysis software packages will be referenced.

The fourth area of research was Elastic Robot Simulation. The simulation was to be a dynamic computer—based model. The model was to be based on a three—dimensional, three—link, revolute geometry robotic arm. This allowed a more realistic image to be made of the system required by the Army. The dynamic behavior of the arm was based on applied forces and torques and upon a static and quasi—dynamic analysis. There are four areas that the computer simulation encompassed: (1) user interaction, (2) three—dimensional representation of the arm, (3) the kinematic behavior, and (4) dynamic modelling of the deformation. The goal of this thesis project was to meet the stated four requirements of the elastic robot computer simulation.

The fifth area of research was the System Integration and Evaluation of System Performance under various robot and sensor configuration. This involved the construction of a scale model of the robotic arm and the testing of the arm. It will be tested under various loading conditions and control configurations. The results will then be compared to those of the analytical model.

Physical characteristics of the arm included a reach of 2.8 meters (110.24 in.) and the capability of manipulating a load of 50.00 kg at a maximum acceleration of 9.81 m/s^2. The base, or link one was composed of a stout base rotated by a hydraulic actuator connected through a rack—and—pinion drive. Links two and three are driven by two hydraulic actuators. The first actuator was connected to the base and to link two while the second actuator was connected between links two and three. The links were composed of square hollow sections, 38.1 mm (1.5 in.) square out of grade 46 steel, 6.35 mm (0.25 in.) in wall thickness, see Figure #1.1.

Lightweight robotic arms have applications in various space projects and mobile applications, such as being used from a truck or space workstation. The research completed in these five areas will help to advance the levels of performance and speed of flexible robotic systems.

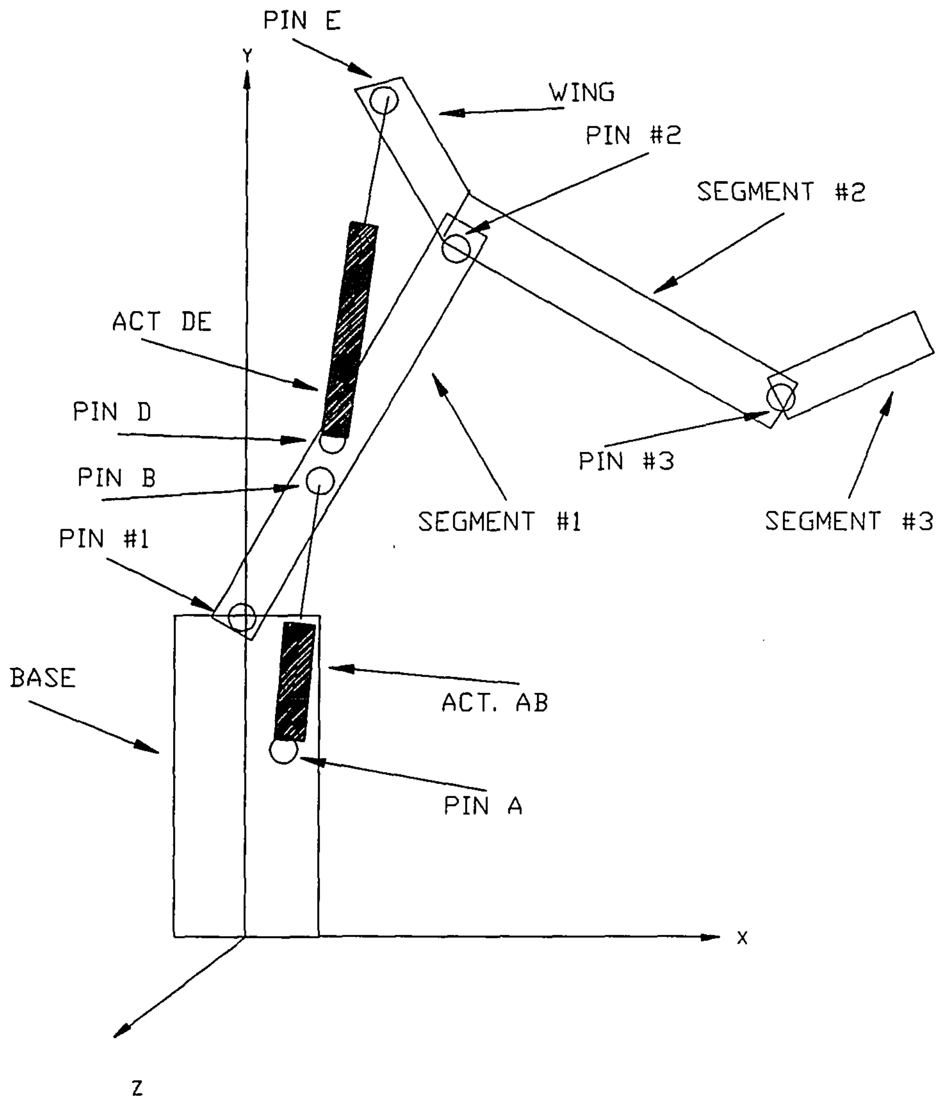


FIGURE 1.1

ELASTIC ROBOT ARM OF THE ARO

CHAPTER 2
HISTORY

Research in the area of computer simulation is relatively new, dating back to the mid 1970's. There are two basic classifications of system modelling. The first classification is for implementation applications. This application requires software which models a process for which the robot will complete in a given work environment. At this point the robotic arm would have already been designed and the modelling software would be used to maximize the efficiency of a layout for the workspace and robotic arm. The second type of modelling software is for design. An engineer would use this software to design a robotic arm for a specific application. This would involve specifying a range of payloads that could be efficiently manipulated. This type of software package would help to eliminate the overdesign typical of rigid robot systems. By using a simulation software package, design parameters of the robot can be varied with dynamic effects reviewed in both graphical and tabular output. The majority of simulation packages that have been written fall into the first classification.

In 1977 Heginbotham at the University of Nottingham in Great Britian designed one of the first computer models(2). This simulation was based on SAMMIE (System for Aiding Man Machine Interaction Evaluation). The computer image of the robot was a wireframe and was used to simiulate an industrial process. From this initial research, various computer simulations have taken on numerous apprearances and capabilities. The appearance and capabilities of a given software package vary widely. Capabilities can range from analytical kinematic and dynamic models to workspace modelling and object collision avoidance verification. The appearance and capabilities related to this thesis are three–dimensional graphics, user interaction, kinematics, and the dynamic modelling of elastic deformation.

Graphical appearances of the robotic arm have ranged in complexity from simple stick figures to two— and three–dimensional images in full color. Two methods for graphical representation exist; using the graphic capabilities of a CAD/CAM or CAE, system and graphical plotting programs.

Wanecke(2), (1978) created a database of two hundred different robots. The graphical interpretation included a topview of machines and obstacles in a polygon shape. Norton(12), (1983) completed a graphical simulation of the Puma robot using an Apple computer. The graphics data was represented using a "layer" format. Norton stored the geometry of each arm of the robot in an independent "layer" of a three dimensional array. Each row of the array layer represented a different node. Each column represented the x, y, and z coordinates. Lines were then drawn connecting the various nodes. Magnenat–Thalman and Thalman(11), (1984) used the programming language, Pascal, to create the MIRA–3D and MIRA–SHADING routines. The software included three–dimensional vector arithmetic, graphical statements, image transformations, and viewing transformations.

The General Electric CAE system SDRC utilizied an array of existing programs along with the drawing capabilities of a CAE system. Although this was among the first to use a color raster display, the system was very limited due to the lack of program interfacing capabilities. Another General Electric simulation package was Robot–Sim, designed by Thomas(20), (1984). As with the previous G.E. software package, this one was also based on interfacing with a CAE system. Objects were shown in a three–dimensional workspace. A multiple of viewing angles were available to the user giving the capability of seeing the entire situation on the screen.

Okino(14), (1985) stated that research problems in areas of CAD/CAM systems based on solid modelling had made slow progress. CAD/CAM systems were initiated in the solid modelling field as a kernel. Okino stated that the solid modellers at this time were considerably slow, incomplete in the areas of drawing tangential curves and surface, and hidden line removal, very limited in the complexity of geometric shapes, provided minimal user interaction interfaces, lacked accuracy, and had minimal effective applications.

As computer hardware has increased in power and capabilities the quality of three—dimensional graphics has also increased. Silicon Graphics, Inc. has produced a system called the IRIS (Integrated Raster Imagining System) Workstation based on the UNIX operating system. There were three pipeline components that the IRIS was designed in in order to increase graphics speed. A Motorola 68020 acted as the system's CPU, the first of the pipeline components.

Herbert and Hoffman(7), (1985) completed their computer simulation using the IRIS Workstation. They created a three—dimensional image of the CMU Direct Drive Arm II by referencing the IRIS's graphics library. The model was used to visualize and edit manipulation tesks of the robot. Full color, hidden line removal, and shading algorithms of the IRIS system were used. The viewpoint could be zoomed in and out as well as rotated and translated about any axis. The objects displayed were in a solid object representation and several objects could appear at once.

Parker(15), (1986) generated a model arm which was described to the system through a variety of parameters including the number and types of joints, axes of rotation, and graphical description of each arm component. The commands were written using FORTRAN. The objects were drawn as a wireframe.

By allowing the user any interactive capabilities, the simulation package becomes increasingly valuable. Norton(13), (1983) also allowed user interaction. His package allowed the user to try a variety of layouts for a parts feeder, fixtures, and workstations. The images were displayed on the screen and then each possible configuration which met the stated requirements were displayed. Objects and fixtures could be redefined with minimal effort.

Work at General Electric(8), (1984) produced a simulation which was concerned with dynamic path errors. After displaying the dynamic path errors, the user could specify arm speeds, settling time, and deceleration rates to meet accuracy requirements. The user could also edit the workspace interactively,. This capability included additions, changes, and deletions of objects and obstacles and repositioning of the manipulator. A library format was used by Thomas(13), (1984) of General Electric. The user could select a robot and gripper from the library.

Herbert and Hoffman(7), (1985) allowed the user pop—up menus and an x—y mouse. This gave the ability to edit the dimensions of a solid and to change the viewpoint.

In Parker(15), (1986) the user was allowed interaction in the form of inputing the parameters of the system from the keyboard. By changing a parameter, although this does not appear to be possible during the running of the simulation, the movement of the robot was altered.

Dave and Jana(1), (1987) set the objective of their package to be user—oriented. The simulation would be used as an educational tool. Through the use of menus, the user could study basic principles of robot kinematics, programming and workcell design. Individual joints as well as the world view could be altered by the user. The wire model was based on the Microbot and was written in Pascal. By referencing the 200 different robot types in the database, Wanecke allowed the user to interact in defining a workspace. After each selection, any obstacle collisions were reported.

Kinematics deals with the position of the elements of the robot manipulator as a function of time irregardless of the forces which are producing the motion. Inman(7), (1984) of the General Electric Company included a general purpose kinematic analysis algorithm. This algorithm gave both the forward and inverse solutions. Thomas(20), (1984) provided the user with accurate information on reach and approach vectors, and singularities during moves.

Parker(14), (1985) used a matrix format to describe the arm. Each element was described using a homogeneous position vector, $P = (p_x, p_y, p_z, 1)$. To alter the coordinates of the given point, the standard matrix transformations were used. There exists a unique matrix transformation for a distinct rotation about each of the three axes, x, y, z. and a translation about any point.

Dave and Jana(1), (1987) created a teaching simulation system which allowed a system to be designed and a full presentation of the kinematic information for the system. Through their menudriven program, the angles of each joint were constantly updated on the screen. The user had a selection of various kinematic information and functions available by specifying a homogeneous transformation matrix, x, y, z, coordinates, absolute joint angles, or incremental angles. Forward and inverse kinematic solutions were also an option to the user.

The majority of simulation packages follow one of the afore mentioned kinematic formats. The kinematic theory is very standardized. The variance from one model to the next has based on the author's desire to manipulate the kinematic data.

Since the majority of simulation packages that have been written fall into the classification of implementation computer models, deformation and dynamic modelling have not been a high concern. As design modelling software becomes increasingly important, the ability to model dynamic effects becomes a highly desired capability.

Thomas(20), (1984) realized the importance of dynamic modelling was. He stated that dynamic models which can provide information such as link masses and inertia, actuator speeds, torque and inertia; gripper and workpiece mass and inertias will be highly important. This information can then lead to determining actual velocities and accelerations of a link and arm deflections from simple beam equations for a link. The dynamic model information and products will help to meet optimal paths, reduced cycle time, and also reduce peak accelerations to extend the life of the robot.

Liegeois(2), (1980) designed a computer simulation system he called M.I.R.E. He manipulated the dynamics by using the Lagrangian equations. Outputs were the load capacities and joint torques. The output was in a tabular form. Gannon(5) and Petroka(17), (1986) at the Naval Postgraduate School both chose to use Lagrangian dynamics for the derivation of the deflection equations.

Derby[3], (1981) wrote a computer program which determined the bending due to gravity and motion accelerations at the calculated positions. The calculations were an option in his computer program. After the arm was positioned and the bending was calculated, the program would list if the members were exceeding their elastic limits. Derby treated the beam as a cantilever beam with a fixed base. He used simple beam analysis to determine the bending, axial deformation and axial twist.

CHAPTER 3
THEORY

3.1 The Computer Graphics Workstation

The computer system used on this project was an IRIS 3130 Workstation constructed by Silicon Graphics, Inc. IRIS is an acronym for Integrated Raster Imaging System. It was a high resolution color graphics computing system. The graphic capabilities encompassed both two— and three—dimensional lines, curves, polygons, and characters. The IRIS hardware can be divided into three components: the Application/Graphics Processor, the Geometry Pipeline, and the Raster Subsystem.

The Applications/Graphics Processor was responsible for controlling the application software, geometry pipeline, and the raster system. To control these three areas, the applications/graphics processor ran the UNIX operating system, various application programs, the graphics library, and I/O software. The graphics library contained more than 200 routines which allowed the user to build images by specifying points, lines, and polygons. Graphics routines were represented in either two— or three—dimensional coordinates which the user could define.

In the next step, the graphics routines were passed through the Geometry Pipeline. In the pipeline the coordinates were transformed, clipped to normalized coordinates, and scaled through transformations to the size of the screen or window. The Geometry Pipeline was composed of twelve VLSI chips. The VLSI chips were referred to as the geometry engines.

The output of the Geometry Pipeline was sent to the third component of the IRIS system, the Raster Subsystem. There were five major hardware components of the Raster Subsystem: Frame Buffer Controller, Bitplane Update Controller, Display Controller, Bitplanes, and the High Resolution Color Monitor. The purpose of the Raster Subsystem was to fill in pixels between line endopoints and polygons, convert character codes into bitmapped characters, perform shading, depth—cueing, and hidden surface removal. The bitplanes acted as storage where a color value for each pixel resided. The graphic drawings were drawn on the screen through the use of the Raster Subsystem. A line which was to be drawn on the screen would be stored in the bitplanes by a patern of color codes for specific pixels. The bitplanes, therefore, controlled what was viewed on the screen.

3.2 User Interaction

The IRIS had four peripherals that allowed the user to interface with a program: (1) Dial Box consisting of eight dials, (2) Switch Box consisting of thirty—two switches, (3) Mouse with three input buttons, and (4) Keyboard consisting of a standard set of keys plus a complete numeric keypad, (See figures 3.1, 3.2, and 3.3.). Silicon Graphics had defined variables for each input of the device: eight dials, thirty—two switches, three mouse keys, and eighty—three keyboard keys. Each input device had a unique variable name and a unique value stored in that variable. See Appendix A of the IRIS User's Guide Volume II Graphics Programming, pages A—22 through A—35 for a complete list. As an example, for Switch #1 the variable name was SW0 and the ASCII value was 111; for the "H" key, the variable was HKEY and the ASCII value was 27; and for Dial #5, the variable is DIAL4 and the ASCII value was 261. See figures 3.1 through 3.2 for identification of input device variables and values.

To use the interactive hardware on the workstation from a FORTRAN program, two steps were followed. First all input devices to be referenced were initialized. This involved changing the status of a given device so that if that device was called upon, a signal would be placed in the events queue and the program would be able to interpret the user's input. This action was completed by using the IRIS command, QDEVIC. Each device name would be sent to the subroutine so that the input device status would be changed to allow input. As an example, if the user wished to use the Dial Box the dials would be initialized by

```
CALL QDEVIC(DIAL0)
CALL QDEVIC(DIAL1)
CALL QDEVIC(DIAL2)
          .
          .
CALL QDEVIC(DIAL6)
CALL QDEVIC(DIAL7)
```

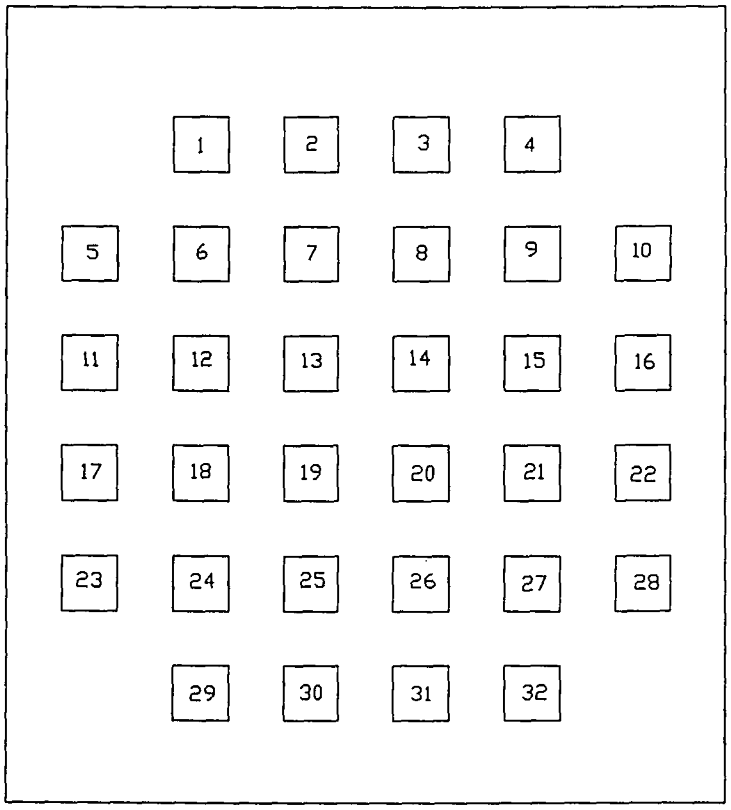


FIGURE 3.1

SWITCH BOX OF THE IRIS 3130

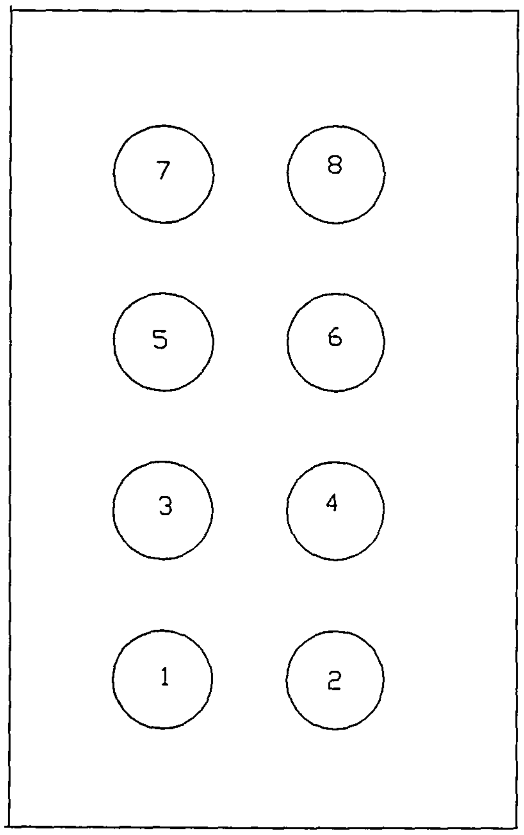


FIGURE 3.2

DIAL BOX OF THE IRIS 3130

The second step to allow the user to interact with a FORTRAN program involved processing the input signal. The input signal placed an entry on a stack in the computer memory, called the events queue. An entry was composed of two parts, the device number and the device value. The device value was stored in the variable, VAL, and the device number in QTYPE. These were both 16—bit integers. To read an entry from the events queue, the IRIS subroutine QREAD was called. QREAD was a function which read the device value of the first entry on the events queue. The programmer could then program an action to be associated with a certain input device. For example, if the user selected the letter "Q", the program would be stopped. Or, by selecting a certain switch, certain information would be written to the screen. These actions would be completed as follows:

```
DEV = QREAD(VAL)
IF(DEV.EQ.QKEY) GO TO 999
```

The term, GO TO 999 would shift the program control to a command to stop the program.

```
IF(DEV.EQSW1) THEN
      WRITE(*,*)' Switch #2 has been selected.'
ENDIF
```

Alternative methods for user interaction I/O devices involved the use of standard FORTRAN commands. Information could be read from a file through formatted and/or unformatted READ statements and information could be written to a file using formatted and/or unformatted WRITE statements.

3.3 Kinematics

The kinematics of a robotic arm may be defined as the analytical study of the geometry of motion of the arm with respect to a fixed reference coordinate system. Two types of kinematic analyses exist. The first type is direct kinematics. Direct kinematics is based upon the given information including joint angles, number of degrees of freedom, and geometric line parameters.The desired output is the position of the end effector. The second type of analysis is inverse kinematics. In inverse kinematics, the desired position of the end—effector and all the link parameters are given. The appropriate joint angles are then calculated to place the end—effector in the desired position. It must also be determined if the angles are valid based on the given specifications. This modelling software prepared for this thesis was based on a direct kinematic analysis.

Robot arms are multilink systems. The position of each link must be described with respect to a fixed reference frame. Since each link could rotate and/or translate with respect to a reference frame, each link would have a unique, local coordinate frame system. The end—product of the direct kinematic analysis was to find a relationship between the local coordinate system of a link and the reference coordinate system. This relationship was described by a homogeneous transformation matrix, T. The homogeneous transformation matrix could be considered as four submatrices:

$$\mathbf{T} = \left[\begin{array}{c|c} R_{3x3} & P_{3x1} \\ \hline f_{1x3} & 1x1 \end{array}\right] = \left[\begin{array}{c|c} \text{Rotation} & \text{Position} \\ \text{Matrix} & \text{Vector} \\ \hline \text{Perspective} & \text{Scaling} \\ \text{Matrix} & \text{Term} \end{array}\right] \tag{3.1}$$

The rotation matrix described rotations about the x, y, and z axes with respect to the reference coordinate system. For example, let OXYZ be the reference coordinate system and OUVW be the local coordinate system. A rotation about the z—axis of 45^o would graphically look like the following:

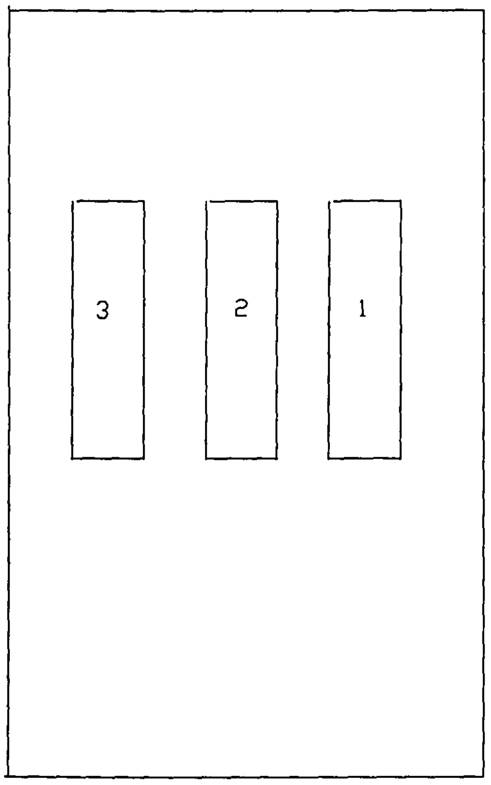


FIGURE 3.3

MOUSE OF THE IRIS 3130

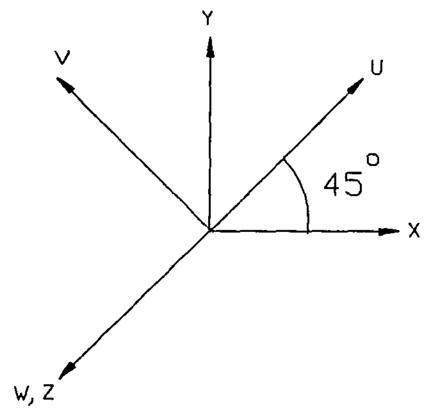


FIGURE 3.4

ROTATION OF THE COORDINATE SYSTEM OUVW 45°

and the rotation matrix would look as follows:

$$\mathbf{R} = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0.0 \\ \sin 45^\circ & \cos 45^\circ & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{3.2}$$

The position vector submatrix expressed a three–dimensional translational relationship between the local coordinate system and the reference coordinate system. A translation of the following

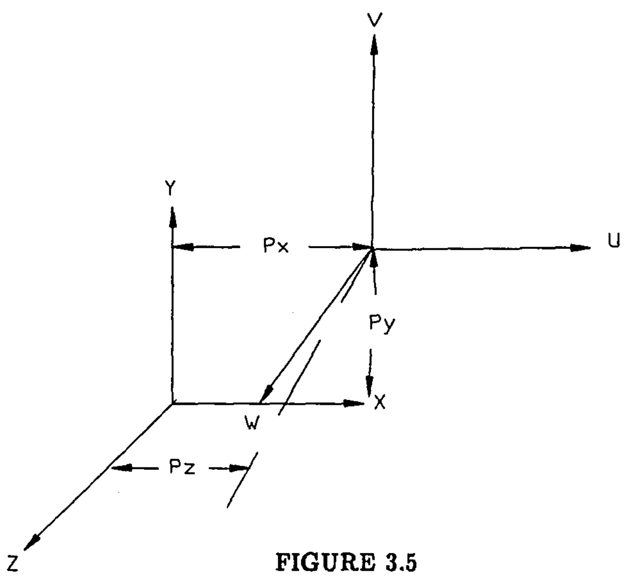


FIGURE 3.5

TRANSLATION OF COORDINATE SYSTEM OUVW

would result in a vector **P** as follows:

$$\mathbf{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1.0 \end{bmatrix} \qquad (3.3)$$

The lower left submatrix is the perspective transformation which is not used in this project and therefore would be a null matrix. The final submatrix, the 1x1 lower right matrix is for scaling. It will be a constant value of 1.0.

An example of a homnogeneous transformation matrix could be based on the following system:

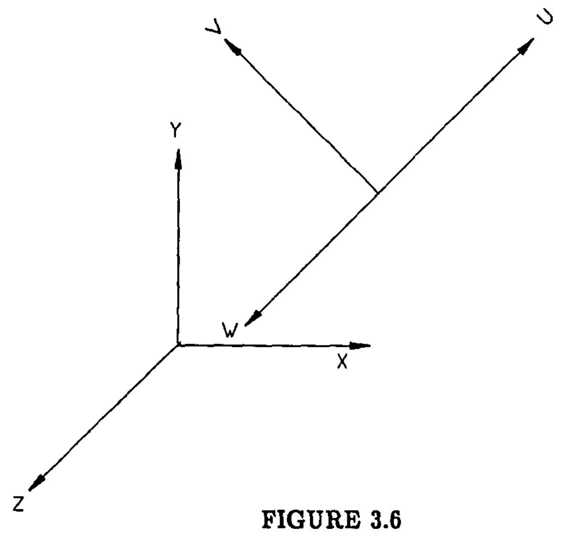


FIGURE 3.6

ROTATION AND TRANSLATION OF COORDINATE SYSTEM OUVW

From the above system, a homogeneous transformation matrix describing the reference frame UVW with respect to the reference frame OXYZ would be

$$T = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ & 0.0 & p_x \\ \sin 45^\circ & \cos 45^\circ & 0.0 & p_y \\ 0.0 & 0.0 & 1.0 & p_z \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{3.4}$$

Now, if a robot arm consisted of multiple links, a description of the relationship between each local coordinate system and the reference system would be required. This could be accomplished by building upon the previous link's relationship. The homogeneous matrix would then describe the location of the ith coordinate frame with respect to the reference coordinate frame. ${}^{0}T_{i}$ is a chain product of successive coordinate transformation matrices, ${}^{i-1}A_{i}$ where ${}^{i-1}A_{i}$ described the relationship between joints i—1 and i. If there were four joints,

$${}^{0}T_{4} = {}^{0}A_{1} * {}^{1}A_{2} * {}^{2}A_{3} * {}^{3}A_{4}. \tag{3.5}$$

A relationship between the reference point, 0, and joint 2 would be found by

$$^{0}T_{2} = {}^{0}A_{1} * {}^{1}A_{2} \tag{3.6}$$

Say a point, P, was described with respect to the local coordinate system of link #2.

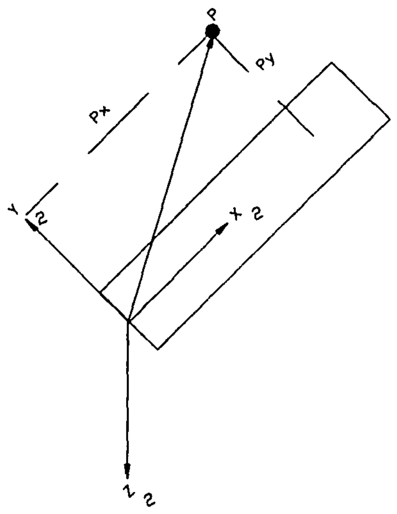


FIGURE 3.7

POINT P DESCRIBED WITH RESPECT TO THE XYZ COORINDATE SYSTEM

The point could be described with respect to the fixed reference frame, OXYZ, by multiplying the position vector by the homogeneous transformation matrix, T, where

$$P = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1.0 \end{bmatrix} \tag{3.7}$$

which is the position vector of point P with respect to coordinate system of link #2 and

$$^{0}T_{2} = {}^{0}A_{1} * {}^{1}A_{2} \tag{3.8}$$

which is the transformation matrix describing the orientation of the coordinate system of link #2 with respect to the OXYZ. Then

$$P_T = T^*P. \tag{3.9}$$

which is the transformed position vector, point P with respect to OXYZ.

3.4 Three Dimensional Graphics

The first step in drawing any object in three–dimensional space was to define the coordinate system by stating a point of reference. To accomplish this on an IRIS Workstation, several steps would be completed. First, the graphics system was initialized. The IRIS subroutine GBEGIN would be the first graphics subroutine called in a program. This subroutine allocated memory for symbol tables and display lists, and initialized hardware. Next, the size of the text and graphics windows would be defined. Two IRIS subroutines were required: PREFPO defined the graphics window coordinates in terms of screen coordinates, TEXTPO defined the text window in terms of screen coordinates.

```
CALL PREFPO(0, 1023, 200, 767)
CALL TEXTPO(0, 1023, 0, 200)
```

The parameters in these subroutines denoted the screen coordinates for the locations of the graphics and text windows. The coordinates were the pixel numbers:

(left,right,bottom,top)

Both the background color of the text window and the color of the text could be defined through the IRIS commands, PAGECO which specified the background color of the text window, and TEXTCO which defined the color of the text in the window. The colors sent to the subroutines were IRIS defined through integer variables. Now that the size of the graphics window had been defined, a scale for the graphics window could be established. By calling the IRIS subroutine ORTHO, the graphics window would be given a three dimensional scale. To give the screen a size of

$$-3.5 \leq X \leq 3.5$$

$$-0.5 \leq Y \leq 5.0$$

$$-3.5 \leq Z \leq 3.5$$

the command ORTHO was called in the following form

```
CALL ORTHO(–3.5, 3.5,–0.5, 5.0,–3.5, 3.5)
```

Any point in space is a unique point based on the unique set of coordinates. The coordinates could be placed in a 4 x 1 matrix called a position vector. The 4 x 1 matrix gives the x, y, and z directional magnitudes which describe the relationship between the reference point and the point being labeled. The value s of the position vector is the scaling factor usually equal to one.

$$P = \begin{bmatrix} P_x \\ P_y \\ P_z \\ s \end{bmatrix} \tag{3.10}$$

For an object to be described in three dimensional space, the programmer would use as many points as required to adequately describe the object. For example, consider the following two shapes:

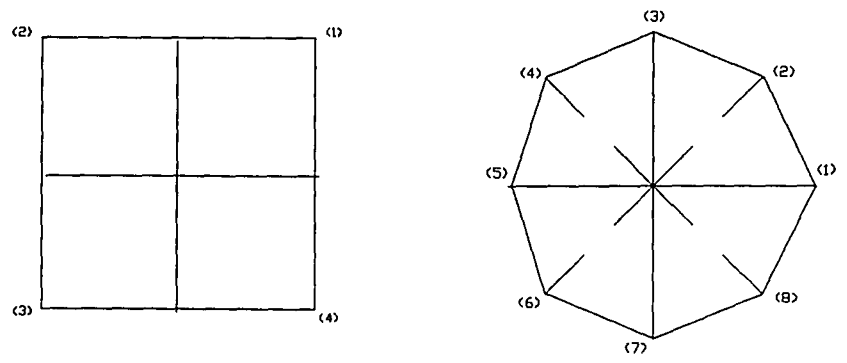


FIGURE 3.8

SQUARE AND OCTAGON SHAPES WITH VERTICES

The square only required four vertices hence only four position vectors were required to describe it. The octagon, had eight vertices and therefore, eight position vectors would be required. The matrix containing the vector information could then be expanded from a (4x1) to a (4x4) for the square, and to a (4x8) for the octagon. So, by stating the reference axis XYZ and then adding four points to the space, a picture similar to the following would be obtained:

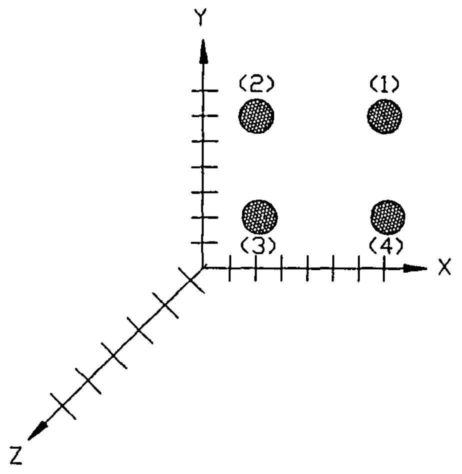


FIGURE 3.9

FOUR POINTS IN SPACE

Each of the four points have their own position vectors:

$$P_1 = \begin{bmatrix}7,6,0,1\end{bmatrix}^T \tag{3.11}$$

$$P_2 = \begin{bmatrix}2,6,0,1\end{bmatrix}^T \tag{3.12}$$

$$P_3 = \begin{bmatrix}2,1,0,1\end{bmatrix}^T \tag{3.13}$$

$$P_4 = \begin{bmatrix}7,1,0,1\end{bmatrix}^T \tag{3.14}$$

The four position vectors could be placed in one matrix that would be a 4 x 4 matrix and take on the form of

$$P = \begin{bmatrix} 7 & 2 & 2 & 7 \\ 6 & 6 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.15}$$

To obtain the figure, the points would then be connected in a systematic manner. For the two examples shown, this would have just required drawing lines from one point to the next point in a sequential manner. Now, if the group of four points were be connected in the order of (1—3—2—4—1) the object drawn would have look like

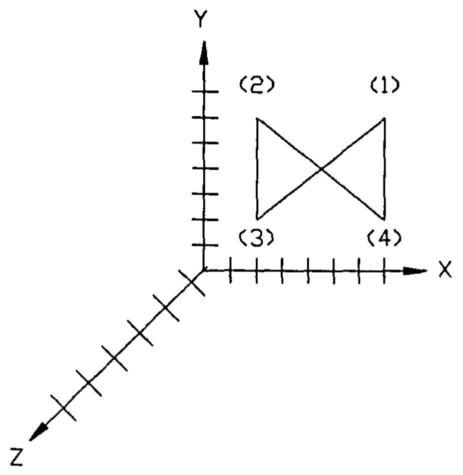


FIGURE 3.10

FOUR POINTS CONNECTED IN THE ORDER (1—3—2—4—1)

If the four points were to have been connected in the order (1—2—3—4—1) a square would have been obtained.

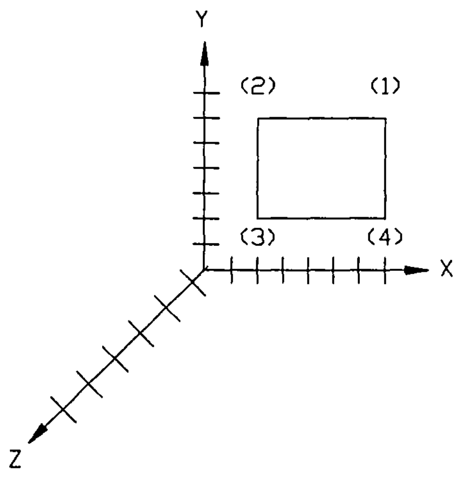


FIGURE 3.11

FOUR POINTS CONNECTED IN THE ORDER (1—2—3—4—1)

Now, consider a three dimensional square cylinder.

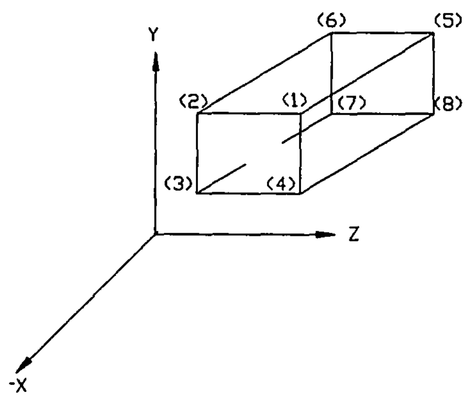


FIGURE 3.12

THREE—DIMENSIONAL SQUARE

There were a total of eight points required to accurately describe the cylinder. The entire cylinder could have also be thought of as a number of smaller square cylinders connected together.

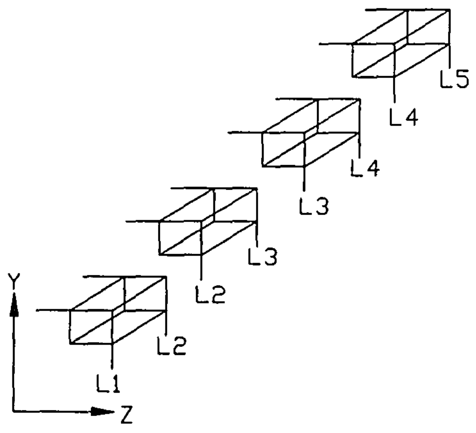


FIGURE 3.13

A SQUARE CYLINDER SUBSECTIONED INTO SMALLER CYLINDERS

Each cross—section division in the y—z plane could be called a level. A level would be described through the coordinates of the four vertices of that cross—section. Each level would describe either the back of one cylinder or the front of the neighboring cylinder.

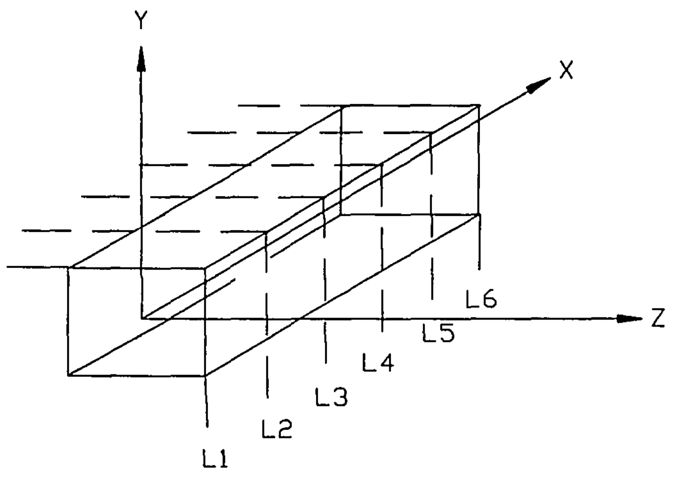


FIGURE 3.14

3—D SQUARE WITH LEVELS

To draw the complete cylinder, the programmer would have simply referenced points of pairs of levels. To draw cylinder 1, you would have reference levels 1 and 2; to draw cylinder 2, you would have reference levels 2 and 3, and so on.

A three—dimensional matrix or array offered a perfect storage medium for such an object. Here, each level could be stored in a different part of the 3—D array. The level number would then correspond to the third index of the array (*,*,L).

For a multi—link robot arm, each link could be described in its respective local coordinate frame. Then, the arm would be assembled by using the homogeneous transformation matrix to transform the position vectors from the local coordinate system to the reference coordinate system. For example, for the given three link arm,

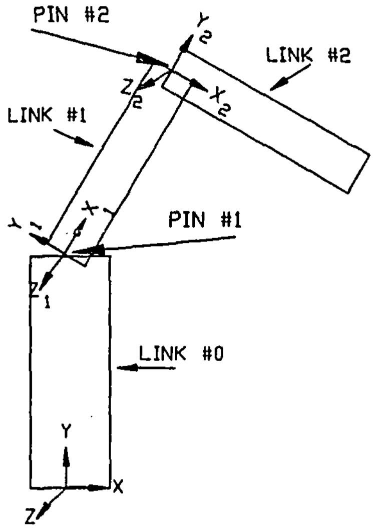


FIGURE 3.15

THREE–LINK SYSTEM

First, links zero, one, and two were described using their local coordinate frames, or,

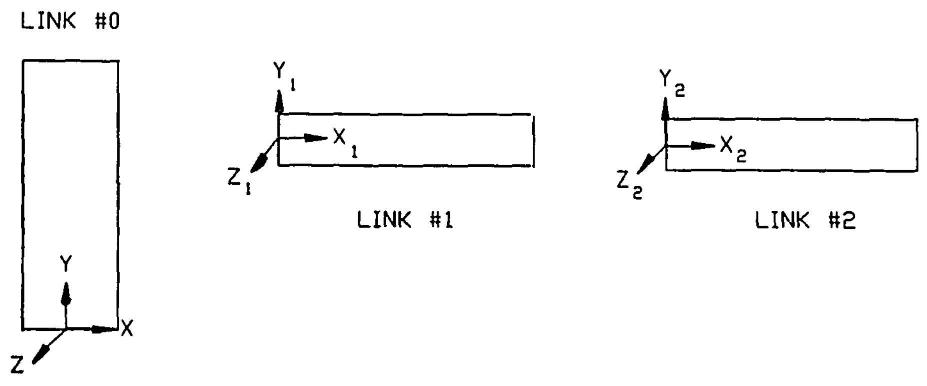


FIGURE 3.16

THREE–LINK SYSTEM IN COMPONENTS

Then, the trannslation and rotation of each local coordinate system with respect to the previous one was specified. This information would complete the A matrices. For example, the local coordinate system of link #0 was the same as the reference. So,

$$A = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{3.16}$$

For link #1, it was desired to have a rotation of $\theta = 30^{\circ}$ about the z—axis. There also existed a translation between the local coordinate system for link #1 and the frame of reference. The translation was the distance from pin #0 to pin #1 or

$$P = \begin{bmatrix} P_{1x} \\ P_{1y} \\ P_{1z} \\ 1.0 \end{bmatrix} \tag{3.17}$$

So,

$$^{0}A_{1} = \begin{bmatrix} \cos 30^{\circ} & -\sin 30^{\circ} & 0.0 & P_{1x} \\ \sin 30^{\circ} & \cos 30^{\circ} & 0.0 & P_{1y} \\ 0.0 & 0.0 & 1.0 & P_{1z} \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{3.18}$$

For link #2, it was desired to have an angle rotation of $\theta = 90^{\circ}$ about the z—axis and the translation was the distance from pin #1 to pin #2 or

$$P = \begin{bmatrix} P_{2x} \\ P_{2y} \\ P_{2z} \\ 1.0 \end{bmatrix} \tag{3.19}$$

So,

$$^{1}A_{2} = \begin{bmatrix} \cos 90^{\circ} & -\sin 90^{\circ} & 0.0 & P_{2x} \\ \sin 90^{\circ} & \cos 90^{\circ} & 0.0 & P_{2y} \\ 0.0 & 0.0 & 1.0 & P_{2z} \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} . \tag{3.20}$$

The transformed points were found as follows:

$$\text{Link0—T} = A_0{*}\text{Link0} \tag{3.21}$$

$$\text{Link1—T} = A_0{*}{}^0A_1{*}\text{Link1} \tag{3.22}$$

$$\text{Link2—T} = A_0{*}{}^0A_1{*}{}^1A_2{*}\text{Link2} \tag{3.23}$$

The final step to reach the desired configuration, Figure 3.16 was to connect the transformed points with line segments. There were two types of objects which the IRIS could draw. The first were line segments. Using the IRIS commands you must have first moved to the beginning of the line segment and then drawn a line to the coordinates of the end of the line segment, or the next point. Both routines required the x, y, and z coordinates of each points. The subroutines were MOVE and DRAW

CALL MOVE(P_{1x},P_{1y},P_{1z})

CALL DRAW(P_{2x},P_{2y},P_{2z})

The second type of shape was a polygon. A polygon could be drawn as an outlined shape, or as a solid. To draw a polygon, the programmer must have first declared an array of size (3,4). The three rows would contain the (x, y, z), coordinates for each of the four points, one to each column. If a polygon contained eight vertices, the array would be of size (3,8). The coordinates of the vertices were stored in the declared array and then the IRIS subroutine was called. For a filled polygon, the call would be

```
CALL POLF(4, POLYSQ)
```

where 4 denoted the number of vertices and POLYSQ was the declared array.

To highlight the 3—D images drawn on the workstation graphics screen, different colors could be used for different entities. The IRIS had a color map which was a table composed of 24—bit RGB values. Each 24—bit value had a unique value or index. The index was stored in the bit planes. Then, when the IRIS was going to draw something, it referenced the color map by the index stored in the bit plane.

A default color map could be referenced by a variable name in which the desired color index was stored in that variable. For example, IRIS had declared the value of 1 to be stored in the variable BLUE. The programmer could call the subroutine COLOR with the variable BLUE and the color blue will be used to draw all subsequent graphic objects.

```
CALL COLOR(BLUE)
```

The programmer could also define a color by using the IRIS subroutine MAPCOL. The subroutine required four parameters, the index value, and then the values for the three guns, red, green, and blue.

```
CALL MAPCOL(111,111,0,111)
```

The calling of the subroutine gave a color which was a combination of red and blue the index 111. Now, if the color subroutine was to be called

```
CALL COLOR (111)
```

that color would be used for drawing.

3.5 Static Analysis

For the robotic arm in Figure 1.1, a static deformation analysis was completed on Segment #1 and Segment #2. The arm would be in an unique geometric configuration based on the angles of each link, the dimensions of the links, and also the location of the pins, (see figure 3.17). Based on the given geometric configuration of the arm, there were three calculations involved with determining the static deflection of the segments:

(1) Determine the Static Reactions

(2) Determine the Virtual Displacements of the Hydraulic Actuators

(3) Determine the Deformation.

<u>3.51 Determine the Static Reactions.</u> Under static analysis, the beam could be considered to be in equilibrium. That is,

$$\Sigma\vec{F} = 0.0 \tag{3.24}$$

$$\Sigma\vec{M} = 0.0 \tag{3.25}$$

For a three–link robotic arm with an end–effector and two actuators, the static reactions would be found at Pin #0, #1, #2, #3, and #4, the point of load, as well as the forces in both actuators. Based on free–body diagrams, Bannoura, (1988) determined the static reactions to be as follows:

Reactions at Pin #0 Due to Static Loading Conditions, (see figures 3.17 and 3.18):

$$R^s_{0x} = R^s_{4x} \tag{3.26}$$

$$R^s_{0y} = -W_0 - W_1 - W_2 - W_3 + R^s_{4y} \tag{3.27}$$

$$R^s_{0z} = R^s_{4z} \tag{3.28}$$

$$M^s_{0x} = (P_{01} + P_{12}\sin\phi_1 + P_{23}\sin\phi_2 + P_{34}\sin\phi_3)R^s_{4z} + M^s_{4x} \tag{3.29}$$

$$M^s_{0y} = -(P_{12}\cos\phi_1 + P_{23}\cos\phi_2 + P_{34}\cos\phi_3)R^s_{4z} + M^s_{4y} \tag{3.30}$$

$$M^s_{0z} = -P_{1g1}\cos\phi_1 {*} W_1 - (P_{12}\cos\phi_1 + P_{2g2}\cos\phi_2)W_2 - (P_{12}\cos\phi_1 + P_{23}\cos\phi_2 + P_{3g3}\cos\phi_3)W_3 +$$

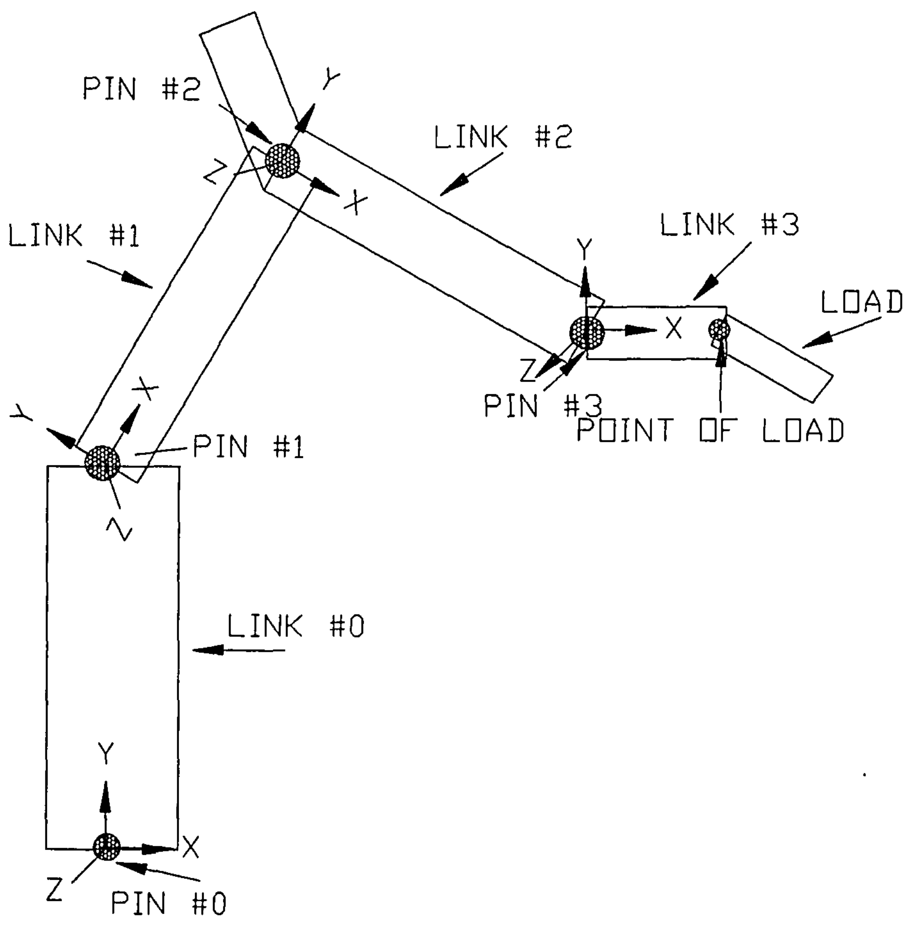


FIGURE 3.17

PIN LOCATIONS FOR FORCE AND MOMENT ANALYSIS

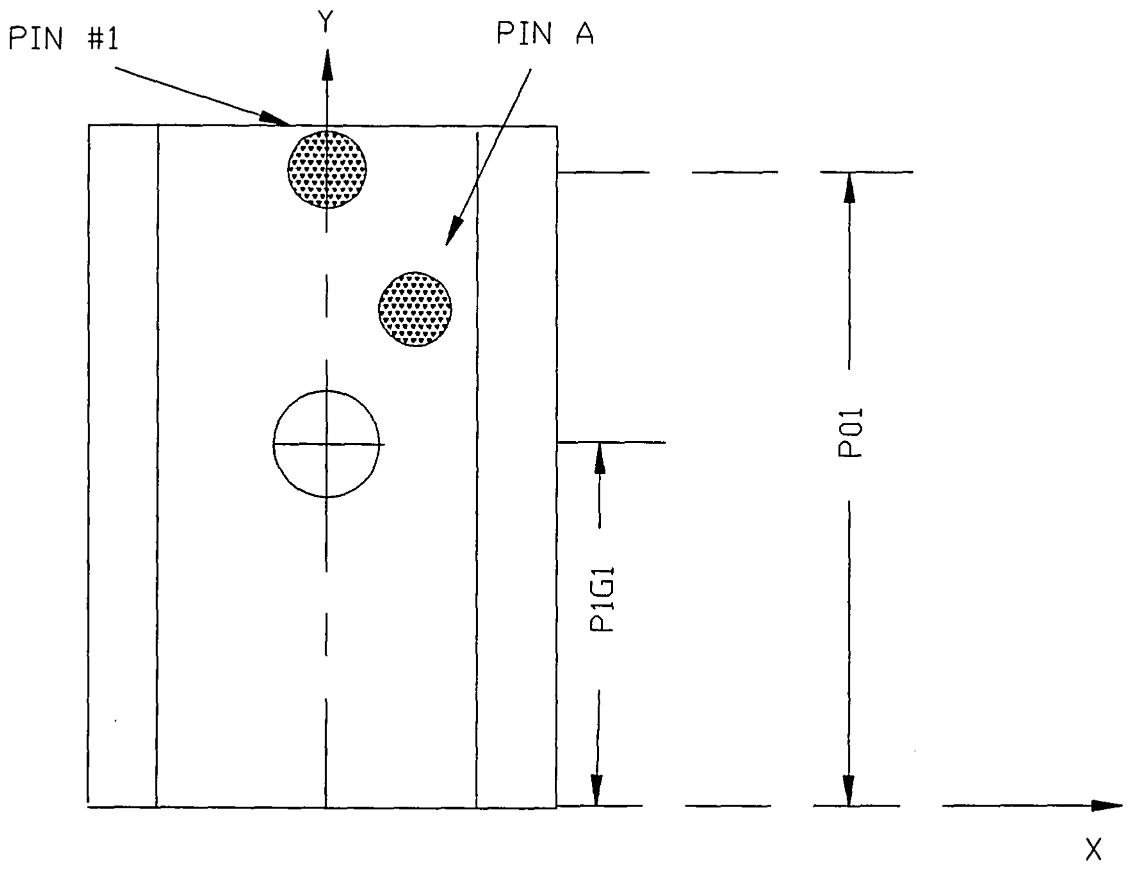


FIGURE 3.18

MOMENT ARM DEFINITIONS OF THE BASE

$$(P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{34}\cos\phi_3)R^s_{4y}-$$
$$(P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+$$
$$P_{34}\sin\phi_3)R^s_{4x}+M^s_{4z} \tag{3.31}$$

Reactions at Pin #1 due to Static Loading Conditions, (see figures 3.17 and 3.19)

$$F^s_{ab}=(P_{1d}\cos\phi_1*F^s_{de}\sin\phi_5-P_{1d}\sin\phi_1*F^s_{de}\cos\phi_5-$$
$$P_{1g1}\cos\phi_1*W_1+P_{12}\cos\phi_1*R^s_{2y}-P_{12}\sin\phi_1*R^s_{2x})/$$
$$(P_{1b}\cos\phi_1\sin\phi_4-P_{1b}\sin\phi_1\cos\phi_4) \tag{3.32}$$

$$R^s_{1x}=F^s_{de}\cos\phi_5-F^s_{ab}\cos\phi_4+R^s_{2x} \tag{3.33}$$

$$R^s_{1y}=F^s_{de}\sin\phi_5-W_1-F^s_{ab}\sin\phi_4+R^s_{2y} \tag{3.34}$$

$$R^s_{1z}=R^s_{2z} \tag{3.35}$$

$$M^s_{1x}=P_{12}\sin\phi_1*R^s_{2z}+M^s_{2x} \tag{3.36}$$

$$M^s_{1y}=-P_{12}\cos\phi_1*R^s_{2z}+M^s_{2y} \tag{3.37}$$

$$M^s_{1z}=0.0 \tag{3.38}$$

Reactions at Pin #2 due to Static Loading Conditions, (see figures 3.17 and 3.20):

$$F^s_{de}=(-P_{23}\cos\phi_2*R^s_{3y}+P_{23}\sin\phi_2*R^s_{3x}+P_{2g2}\cos\phi_2*W_2-M^s_{3z})/$$
$$(P_{2e}\sin\phi_6\cos\phi_5-P_{2e}\cos\phi_6\sin\phi_5) \tag{3.39}$$

$$R^s_{2x}=-F^s_{de}\cos\phi_5+R^s_{3x} \tag{3.40}$$

$$R^s_{2y}=-W_2+R^s_{3y}-F^s_{de}\sin\phi_5 \tag{3.41}$$

$$R^s_{2z}=R^s_{3z} \tag{3.42}$$

$$M^s_{2x}=P_{23}\sin\phi_2*R^s_{3z}+M^s_{3x} \tag{3.43}$$

$$M^s_{2y}=-P_{23}\cos\phi_2*R^s_{3z}+M^s_{3y} \tag{3.44}$$

$$M^s_{2z}=0.0 \tag{3.45}$$

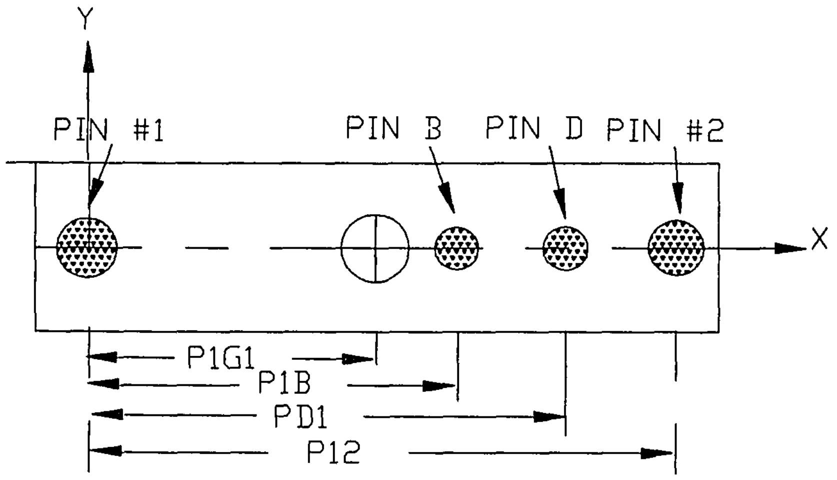


FIGURE 3.19

MOMENT ARM DEFINITIONS OF SEGMENT #1

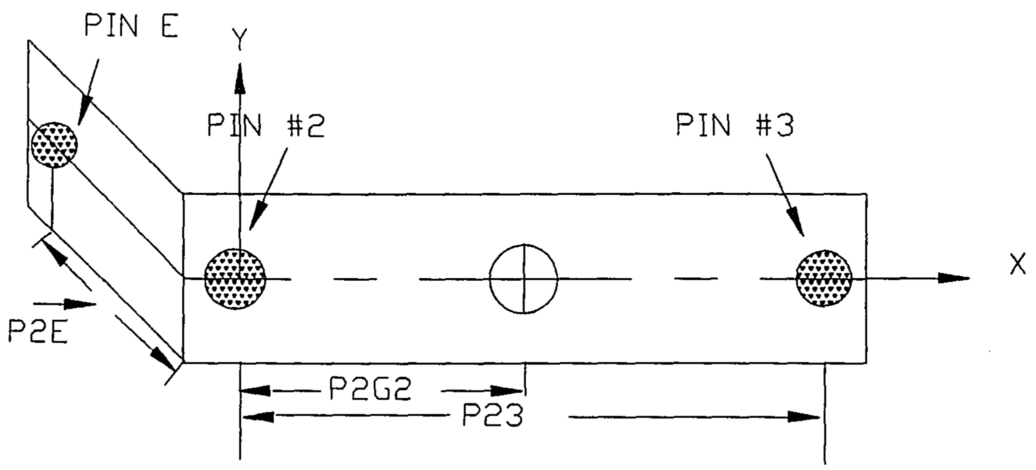


FIGURE 3.20

MOMENT ARM DEFINITIONS OF SEGMENT #2

Reactions at Pin #3 due to Static Loading Conditions, (see figures 3.17 and 3.21):

$$R^s_{3x} = R^s_{4x} \tag{3.46}$$

$$R^s_{3y} = -W_3 + R^s_{4y} \tag{3.47}$$

$$R^s_{3z} = R^s_{4z} \tag{3.48}$$

$$M^s_{3x} = P_{34}\sin\phi_3 * R^s_{4z} + M^s_{4x} \tag{3.49}$$

$$M^s_{3y} = -P_{34}\cos\phi_3 * R^s_{4z} + M^s_{4y} \tag{3.50}$$

$$M^s_{3z} = P_{34}\cos\phi_3 * R^s_{4y} - P_{34}\sin\phi_3 * R^s_{4x} - P_{3g3}\cos\phi_3 * W_3 + M^s_{4z} \tag{3.51}$$

Reactions at Point of Load due to Static Loading Conditions, (see figure 3.17):

$$R^s_{4x} = 0.0 \tag{3.52}$$

$$R^s_{4y} = \text{Weight}_{\text{load}} \tag{3.53}$$

$$R^s_{4z} = 0.0 \tag{3.54}$$

$$M^s_{4x} = -P_{4g4} * \sin\phi_0 * \cos\phi_3 * W_4 \tag{3.55}$$

$$M^s_{4y} = 0.0 \tag{3.56}$$

$$M^s_{4z} = -P_{4g4} * \cos\phi_0 * \cos\phi_3 * W_4 \tag{3.57}$$

Where:

R^s_{ij} — Static force reaction at Pin #i in the j direction

M^s_{ij} — Static moment reaction at Pin #i in the j direction

W_i — Weight of link i

P_{ij} — Distance from Pin #i to Pin #j

P_{igi} — Distance from center of gravity of link i to Pin #i

3.52 Determine the Virtual Displacements of the Hydraulic Actuators. The axial forces of the actuators, F^s_{ab} and F^s_{de}, created virtual displacements in the respective actuator. Virtual displacements in the actuators were products of a change in the fluid pressure in the hydraulic

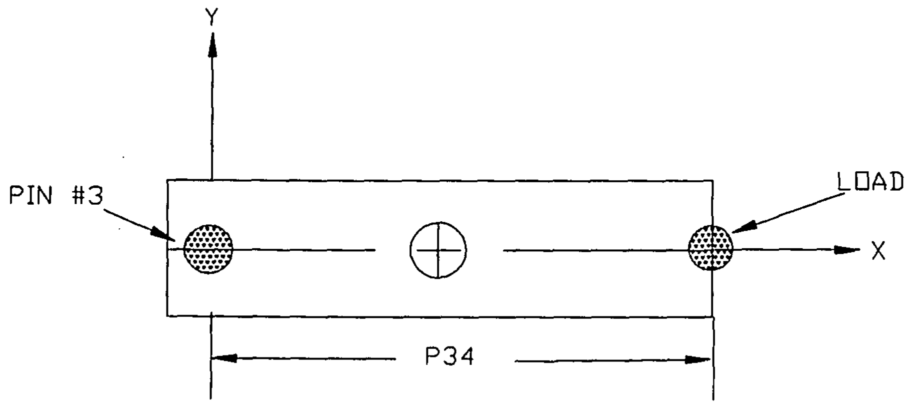


FIGURE 3.21

MOMENT ARM DEFINITIONS OF SEGMENT #3

cylinders. The pressures could be varied by three separate servovalves. A change in length of any actuator resulted in a change in the angle associated with that actuator. For actuator AB, the angles were ϕ_1 and ϕ_4. For actuator DE, the angles effected were ϕ_2, ϕ_5 and ϕ_6. The locations of Segments #1 and #2 were then adjusted due to the differential change in the angles.

The differential change in the angles could be determined by first describing the relationship of the actuator geometry. For actuator AB, (see figure 3.22), the relationship was from the law of cosines,

$$C_1^{\,2} = A_1^{\,2} + B_1^{\,2} - 2A_1B_1\cos(\beta). \tag{3.58}$$

Based on the geometric configuration, the differential change in ϕ_1 was equal to the differential change in β

$$\delta\phi_1 = \delta\beta \tag{3.59}$$

Differentiating the equation by $dC_1/d\beta$ gave an equation which described the differential change of the length of the actuator, C_1 to the change of β.

$$2C_1dC_1 = 2A_1B_1 \sin\beta \, d\beta \tag{3.60}$$

The change of length, dC_1, could also be described based on the deformation as a result of an axial loaded member in compression,

$$dC_1 = \frac{FL}{AE} \tag{3.61}$$

where

F = Axial load equal to F^s_{ab}

L = Length of member equal to C_1

A = Cross–sectional area of actuator

E = Modulus of elasticity of actuator.

Substituting F^s_{ab} and C_1 into equation 3.61 yielded

$$dC_1 = \frac{F^s_{ab}C_1}{AE} \tag{3.62}$$

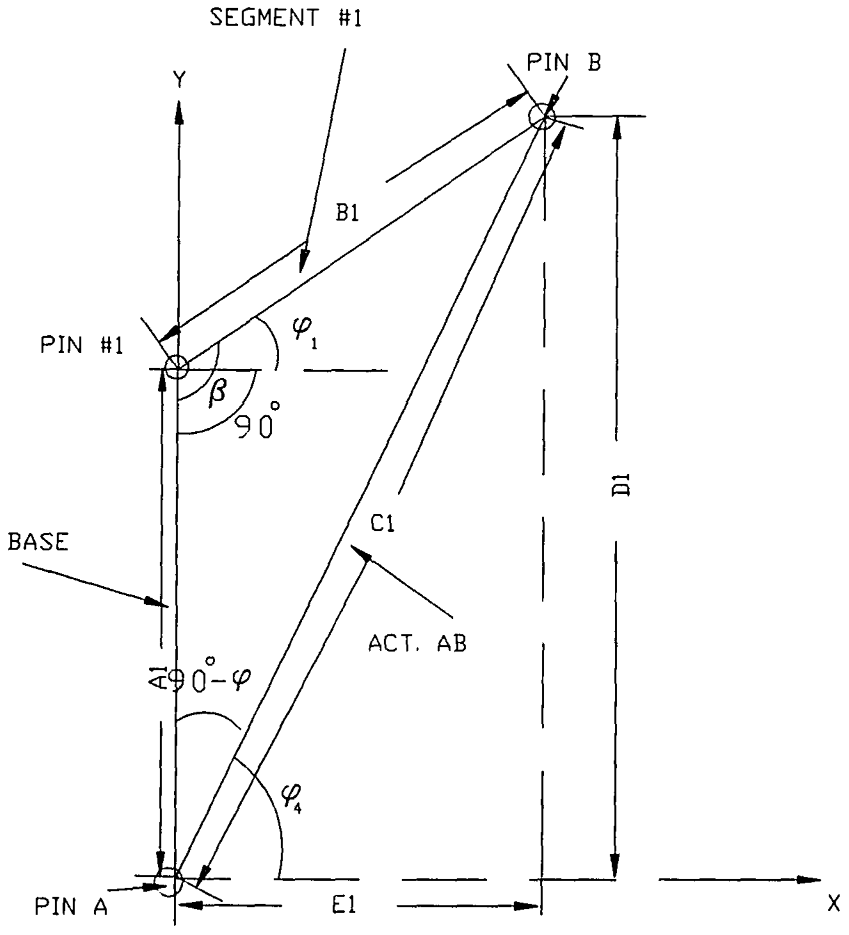


FIGURE 3.22

HYDRAULIC ACTUATOR AB

Therefore, solving equation 3.60 for $d\beta$ yielded

$$d\beta = \frac{C_1}{A_1B_1 \ \sin\beta} dC_1. \tag{3.63}$$

Substituting equation 3.62 into 3.63 yielded

$$d\beta = \frac{F^s_{ab} \ C_1{}^2}{A_1B_1 \ \sin\beta \ AE}. \tag{3.64}$$

The relationship describing the virtual displacement for actuator DE was very similiar, (see figure 3.23). Starting with the law of cosines

$$C_2{}^2 = A_2{}^2 + B_2{}^2 - 2A_2B_2 \cos(\alpha). \tag{3.65}$$

For this geometric configuration, α would need to be determined. Again, from Figure 3.23

$$\kappa = \Pi - \phi_1 \tag{3.66}$$

$$\alpha = 2\Pi - \phi_6 - \kappa \tag{3.67}$$

The differential change in α was equal to the differential change in ϕ_2.

$$d\alpha = d\phi_2. \tag{3.68}$$

The differential equation was

$$2C_2dC_2 = 2A_2B_2 \sin\alpha \, d\alpha \tag{3.69}$$

As with actuator AB, the change of length, dC_2 could also be described based on the deformation as a result of an axial loaded member in compression,

$$dC_2 = \frac{FL}{AE} \tag{3.70}$$

where

F = Axial load equal to F^s_{de}

L = Length of member equal to C_2

A = Cross–sectional area of actuator

E = Modulus of elasticity of actuator.

Substituting F^s_{de} and C_2 into equation 3.70 yielded

$$dC_2 = \frac{F^s_{de} C_2}{AE} \tag{3.71}$$

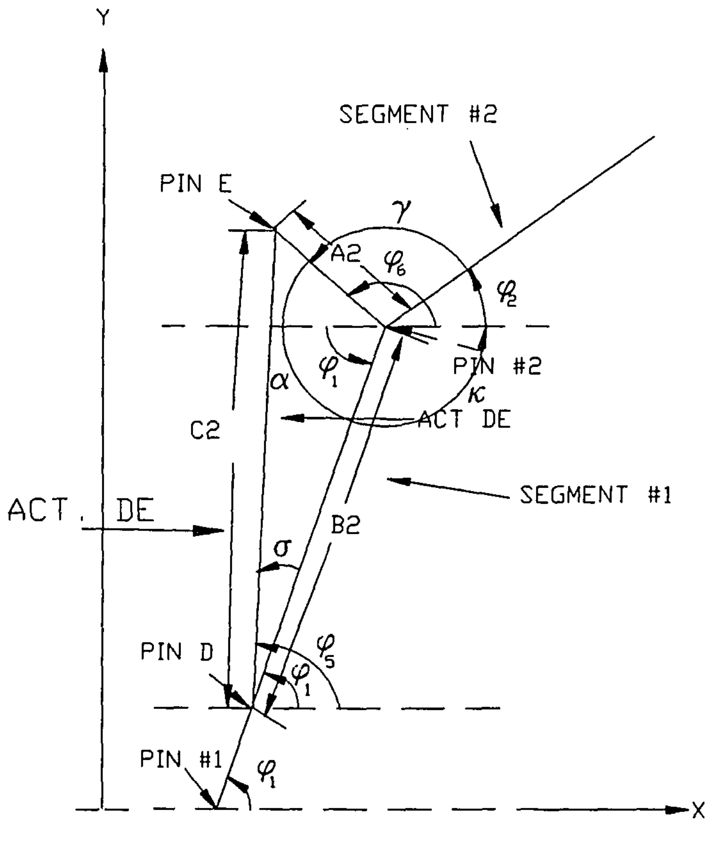


FIGURE 3.23

HYDRAULIC ACTUATOR DE

Therefore, solving equation 3.69 for dα yielded

$$d\alpha = \frac{C_2}{a_2 b_2 \sin\alpha} dC_2. \tag{3.72}$$

Substituting equation 3.71 into 3.72 yielded

$$d\alpha = \frac{F^s_{de} C_2{}^2}{a_2 b_2 \sin\alpha\ AE}. \tag{3.73}$$

The differential change in angles would then need to be added to the respective angles so that the deformation analysis was correct. The static reactions would not need to be recalculated because the static loading condition was considered to be quasi–static.

3.53 Determine the Deformation. The first and second segments of the robotic arm were assumed to be slender flexible elastic beams. Each segment had four types of deformation:

(1) Torsional twist about the x axis

(2) Axial extension in the x direction.

(3) Deflection in the y direction

(4) Deflection in the z direction

Torsional twist and axial extension were based upon the following equations of Bannoura,

$$\theta_{x1} = \frac{T_{g1x}\cos\phi_1 + T_{g1y}\sin\phi_1}{L_1 J_{1x} G_1}\left(L_1 x_1 - \frac{x_1^2}{2}\right) + \frac{M_{2x}\cos\phi_1 + M_{2y}\sin\phi_1}{L_1 J_{1x} G_1}(x_1) \tag{3.74}$$

$$\theta_{x2} = \frac{T_{g2x}\cos\phi_2 + T_{g2y}\sin\phi_2}{L_2 J_{2x} G_2}\left(L_2 x_2 - \frac{x_2^2}{2}\right) + \frac{M_{3x}\cos\phi_2 + M_{3y}\sin\phi_2}{L_2 J_{2x} G_2}(x_2) \tag{3.75}$$

$$u_{x1} = \frac{Q_{g1x}\cos\phi_1 + Q_{g1y}\sin\phi_1}{L_1A_1E_1}\left(L_1x_1 - \frac{x_1^2}{2}\right) + \frac{R_{2x}\cos\phi_1 + R_{2y}\sin\phi_1}{L_1A_1E_1}(x_1) \tag{3.76}$$

$$u_{x2} = \frac{Q_{g2x}\cos\phi_2 + Q_{g2y}\sin\phi_2}{L_2A_2E_2}\left(L_2x_2 - \frac{x_2^2}{2}\right) + \frac{R_{3x}\cos\phi_2 + R_{3y}\sin\phi_2}{L_2A_2E_2}(x_2) \tag{3.77}$$

Since the loading conditions were considered to be static, the above equations simplified to

$$\theta_{x1} = \frac{M^s_{2x}\cos\phi_1 + M^s_{2y}\sin\phi_1}{L_1J_{1x}G_1}(x_1) \tag{3.78}$$

$$\theta_{x2} = \frac{M^s_{3x}\cos\phi_2 + M^s_{3y}\sin\phi_2}{L_2J_{2x}G_2}(x_2) \tag{3.79}$$

$$u_{x1} = \frac{R^s_{2x}\cos\phi_1 + R^s_{2y}\sin\phi_1}{L_1A_1E_1}(x_1) \tag{3.80}$$

$$u_{x2} = \frac{R^s_{3x}\cos\phi_2 + R^s_{3y}\sin\phi_2}{L_2A_2E_2}(x_2) \tag{3.81}$$

Where:

u_{x1} — Axial extension for Segment #1

u_{x2} — Axial extension for Segment #2

θ_{x1} — Torsional twist about the x axis for Segment #1

θ_{x2} — Torsional twist about the x axis for Segment #2

R^s_{ij} — Static force reaction at Pin #i in the j direction

M^s_{ij} — Static moment reaction at Pin #i in the j direction

$L_1 - P_{12}$ which is the distance between Pins #1 & #2.

$L_2 - P_{23}$ which is the distance between Pins #2 & #3.

J_i — Polar moment of inertia.

G_i — Shear modulus of elasticity.

x_1 — The x distance along Segment #1

x_2 — The x distance along Segment #2

The torsional twist also produced a deformation along the x–axis. Based on Figure 3.24

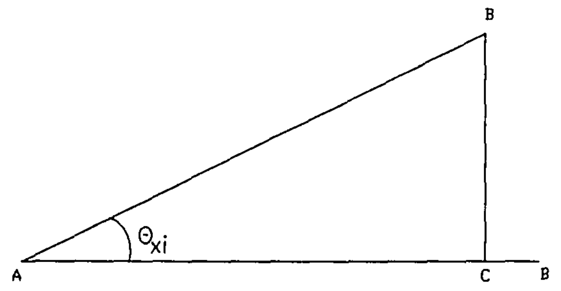


FIGURE 3.24

AXIAL DEFORMATION DUE TO TORSIONAL TWIST

The orginial length was AB. The angle of twist was θ_{xi}. Based on this angle, AB was rotated to the new position. This created a shortening of the linear length, that is, AC < AB

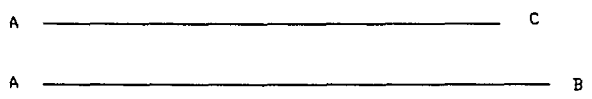


FIGURE 3.25

COMPARASION OF THE LENGTHS OF SEGMENTS AC AND AB

To determine the deformation of the segment,

$$\delta = AB - AC \tag{3.82}$$

the length of AC would be required . By using a trigonometric relationship and given the magnitude of AB,

$$\cos \theta_{xi} = \frac{AC}{AB}, \tag{3.83}$$

$$AC = AB * \cos \theta_{xi}. \tag{3.84}$$

Solving for δ

$$\delta = AB(1 - \cos \theta_{xi}). \tag{3.85}$$

The total deformation along the x axis was then

$$\text{Total Deformation.} = \delta + u_{xi} \tag{3.86}$$

The deflections in the y and z directions were also based on the equations of Bannoura.

$$V_i = \frac{q_{iyi}}{24E_i I_{zi}} \left[x_i^4 - 4L_i x_i^3 + 6L_i^2 x_i^2\right] +$$

$$\frac{q^o_{iyi}}{120E_i I_{zi} L_i} \left[2x_i^5 - 5L_i x_i^4 + 12L_i^3 x_i^2\right] +$$

$$\frac{R^{l}_{(i+1)yi}}{6E_iI_{zi}}(3L_ix_i^2 - x_i^3) + \frac{M_{(i+1)zi}}{2E_iI_{zi}}x_i^2 \tag{3.87}$$

$$\theta_{zi} = \frac{q_{iyi}}{24E_iI_{zi}}\left[4x_i^3 - 12L_ix_i^2 + 12L_ix_i\right] +$$

$$\frac{q^{o}_{iyi}}{20E_iI_{zi}L_i}\left[10x_i^4 - 20L_ix_i^3 + 12L_i^3x_i\right] +$$

$$\frac{R^{l}_{(i+1)yi}}{6E_iI_{zi}}(6L_ix_i - 3x_i^2) - \frac{M_{(i+1)zi}}{E_iI_{zi}}x_i \tag{3.88}$$

$$W_i = \frac{q_{izi}}{24E_iI_{yi}}\left[x_i^4 - 4L_ix_i^3 + 6L_i^2x_i^2\right] +$$

$$\frac{q^{o}_{izi}}{120E_iI_{yi}L_i}\left[2x_i^5 - 5L_ix_i^4 + 12L_i^3x_i^2\right] +$$

$$\frac{R_{(i+1)zi}}{6E_iI_{yi}}(3L_ix_i^2 - x_i^3) -$$

$$\frac{M^{l}_{(i+1)yi}}{2E_iI_{yi}}x_i^2 \tag{3.89}$$

$$\theta_{yi} = -\frac{q_{izi}}{24E_iI_{yi}}\left[4x_i^3 - 12L_ix_i^2 + 12L_i^2x_i\right] +$$

$$\frac{q^{o}_{izi}}{120E_iI_{yi}L_i}\left[10x_i^4 - 20L_ix_i^3 + 24L_i^3x_i\right] +$$

$$\frac{R_{(i+1)zi}}{6E_iI_{yi}}(6L_ix_i - 3x_i^2) - \frac{M^{l}_{(i+1)yi}}{E_iI_{yi}}x_i \tag{3.90}$$

Simplifying the equations based on static loading conditions yielded

$$V_i = \frac{R^{l}_{(i+1)yi}}{6E_iI_{zi}}(3L_ix_i^2 - x_i^3) + \frac{M^{s}_{(i+1)zi}}{2E_iI_{zi}}x_i^2 \tag{3.91}$$

$$\theta_{zi} = \frac{R^{l}_{(i+1)yi}}{6\,E_i I_{zi}}(6L_i x_i - 3x_i^2) - \frac{M^{s}_{(i+1)zi}}{E_i I_{zi}} x_i \qquad (3.92)$$

$$W_i = \frac{R^{s}_{(i+1)zi}}{6E_i I_{yi}}(3L_i x_i^2 - x_i^3) - \frac{M^{l}_{(i+1)yi}}{2E_i I_{yi}} x_i^2 \qquad (3.93)$$

$$\theta_{yi} = \frac{R^{s}_{(i+1)zi}}{6\,E_i I_{yi}}(6L_i x_i - 3x_i^2) - \frac{M^{l}_{(i+1)yi}}{E_i I_{yi}} x_i \qquad (3.94)$$

Where:

V_i — Displacement of segment i in the y direction

θ_{zi} — Slope of displacement for segment i about the z axis

W_i — Displacement of segment i in the z direction

θ_{yi} — Slope of displacement for segment i about the y axis

x_i — Linear distance along the x axis where deflection calculations are determined

L_i — Distance on segment i from Pin #i to Pin #(i+1)

E_i — Modulus of elasticity of segment i

I_{ij} — Moment of about the i axis for segment j

$M^{l}_{(i+1)yi}$ — Moment reaction in the local y_i—axis.

$$M^{l}_{(i+1)yi} = M^{s}_{(i+1)y}\sin\phi_i + M^{s}_{(i+1)x}\cos\phi_i \qquad (3.95)$$

$R^{l}_{(i+1)yi}$ — Force reaction in the local y_i—axis.

$$R^{l}_{(i+1)y} = R^{s}_{(i+1)y}\cos\phi_i - R^{s}_{(i+1)x}\sin\phi_i. \qquad (3.96)$$

The static analysis could be considered in a cyclical format. For a given geometry, the reactions and virtual displacements were determined. Next, the deformation would be determined at incremental steps along the length of both the segments. The first calculation would be at the lower pin, that is at Pin #1 for Segment #1. The deformation would be determined at each interval until the upper pin position was reached, Pin #3 on Segment #2. As the deformation analysis was completed for Segment #2, the analytical cycle would be completed. The next step would be the altering of the geometric configuration of the robot. And, the process would then be continued.

3.6 Dynamic Analysis

A Dynamic deformation analysis was also completed on Segments #1 and #2 of the robotic arm in Figure 1.1. The dynamic analysis was not as simplistic as the static analysis because the beam was not in equilibrium, but instead,

$$\Sigma\vec{F} = m\vec{a} \tag{3.97}$$

$$\Sigma\vec{M} = \vec{I}\vec{\alpha} \tag{3.98}$$

where I is the Inertial Tensor matrix. Therefore, there existed velocities and accelerations associated with the different links. A quasi–static analysis was completed on the links by stepping the robot through very small time increments under the influence of applied actuator forces. These applied actuator forces were determined by the user and placed in the dynamic input parameter file. (See Chapter 5.5). Initial values for the angles, velocities, and accelerations of Segments #1, #2, and the Base were defined to be equal to zero.

In summary, the quasi–static dynamic analysis of the robot was accomplished by using the angles, velocities, and accelerations of each segment along with the applied hydraulic actuator forces to calculate the new position of each segment after a very small time increment. Newton's equations of motion were employed to calculate the new velociotics and accelerations of the segments by using the masses and moments of inertia given in the parameter file. This numerical technique produced solutions that were equal to the exact solutions as the time increment becomes vanishingly small. On the IRIS workstation, a time increment of 1 millisecond was chosen.

In a more complete description of the quasi–static analysis, eight steps were accomplished during every cycle of the analysis. One caculation cycle on the computer corresponded to one time increment of actual robot motion.

(1) **Determine the Static Reactions.** The first step was to calculate the static reactions. These reactions were based on the initial configuration of the arm, the load and the

weight of the various segments. The static reactions acted as a point of reference.

(2) **Determine the Applied and Dynamic Actuator Forces.** The applied actuator forces for time step #1 were read in from the dynamic input parameter file. Based on the static reactions determined in (1) and the applied forces of time step #1, the Dynamic Actuator forces could be determined.

(3) **Determine the Inertial Reactions.** Under the dynamic conditions the $\Sigma\vec{F}=m\vec{a}$ and $\Sigma\vec{M}=\vec{\vec{I}}\vec{\alpha}$. To determine the velocities and accelerations, the angular acceleration α must be determined for each segment. To determine $\vec{\alpha}$, first the sum of the moments must be determined. The Inertial reactions were defined to be $\Sigma\vec{F}$ and $\Sigma\vec{M}$ and were based on the static and dynamic reactions of the arm.

(4) **Determine the Velocities and Accelerations.** Since $\vec{\vec{I}}$ is the inertia tensor and a characteristic of the segments and known, and ΣM was just determined in (3), $\vec{\alpha}$ could be determined for each segment. Next, the angular velocity, $\vec{\omega}$ could be determined from $\vec{\alpha}$. Based on the $\vec{\omega}$ and $\vec{\alpha}$ of the Base and Segments #1 and #2, the linear velocities and accelerations for Segments #1 and #2 could also be determined. The velocities and accelerations determined in this step are for the given time step associated with the applied actuator forces. The magnitude of the time step was specified by the user in the input parameter file.

(5) **Determine the New Position of the Arm in Time.** Based on the time step, the initial angles and the angular velocities and accelerations, any change in the angles could be determined and the new angular position of the segments could as be determined.

(6) **Determine the new Static Reactions.** New static reactions must be calculated based on the new angular positions of the segments.

(7) **Determine the Dynamic Reactions.** The dynamic reactions must also be determined based on the new angular positions of the segments. The inertial reactions were the

sum of the static reactions and the dynamic reactions. Therefore, these reactions must be redetermined.

(8) **Determine the Deformations.** The positions of the segments and the reactions are now accurate for the given time step. The deformations could now be calculated.

After the deformations were determined, the process is returned to (2) where the applied actuator forces for the next time step are obtained from the dynamic input parameter file.

3.61 Determine the Static Reactions. The static reactions would be determined first. This would then allow the dynamic actuator forces to be calculated based on the applied actuator forces which the user had specified in the input file and the static actuator forces. The static reactions were based on the same equations as those used for the static analysis, equations 3.26 through 3.57.

3.62 Determine the Applied and Dynamic Actuator Forces. The input into the dynamic model was the applied actuator forces. The dynamic forces could be determined from the applied forces, and the static forces,

$$F^{d}_{ab} = F^{a}_{ab} - F^{s}_{ab} \tag{3.99}$$

$$F^{d}_{de} = F^{a}_{de} - F^{s}_{de} \tag{3.100}$$

Where:

F^{d}_{ab} — Dynamic force of actuator AB

F^{a}_{ab} — Applied force of actuator AB

F^{s}_{ab} — Static force of actuator AB

F^{d}_{de} — Dynamic force of actuator DE

F^{a}_{de} — Applied force of actuator DE

F^{s}_{de} — Static force of actuator DE.

3.63 Determine the Inertial Reactions. Based on the dynamic reaction equations of Bannoura, the inertial forces, Q^{d}_{ij}, and inertial torques, T^{d}_{ij} were determined.

Inertial reactions at Pin #0:

$$Q^{d}_{g0x} = R^{d}_{0x} - R^{s}_{4x} - Q^{d}_{g1x} - Q^{d}_{g2x} - Q^{d}_{g3x} \tag{3.101}$$

$$Q^d_{g0y} = R^d_{0y} + W_0 + W_1 + W_2 + W_3 - R^s_{4y} - Q^d_{g1y} - Q^d_{g2y} - Q^d_{g3y} \tag{3.102}$$

$$Q^d_{g0z} = R^d_{0z} - R^s_{4z} - Q^d_{g1z} - Q^d_{g2z} - Q^d_{g3z} \tag{3.103}$$

$$T^d_{g0x} = M_{0x} - (P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+ P_{34}\sin\phi_3)R^s_{4z} - M^s_{4x} - P_{0g0}*Q^d_{g0x} - (P_{01}+P_{1g1}\sin\phi_1)Q^d_{g1z} - (P_{01}+P_{12}\sin\phi_1+P_{2g2}\sin\phi_2)Q^d_{g2z} - (P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{3g3}\sin\phi_3)Q^d_{g3z} - T^d_{g1x} - T^d_{g2x} - T^d_{g3x} \tag{3.104}$$

$$T^d_{g0y} = M^d_{0y} + (P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{34}\cos\phi_3)R^s_{4z} - M^s_{4y} + P_{1g1}\cos\phi_1 * Q^d_{g1z} + (P_{12}\cos\phi_1+P_{2g2}\cos\phi_2)Q^d_{g2z} + (P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{3g3}\cos\phi_3)Q^d_{g3z} - T^d_{g1y} - T^d_{g2y} - T^d_{g3y} \tag{3.105}$$

$$T^d_{g0z} = M^d_{0z} + P_{1g1}\cos\phi_1*W_1 + (P_{12}\cos\phi_2+P_{2g2}\cos\phi_2)W_2 + (P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{3g3}\cos\phi_3)W_3 - (P_{12}\cos\phi_1+P_{12}\cos\phi_2+P_{34}\cos\phi_3)R^s_{4y} + (P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{34}\sin\phi_3)R^s_{4x} - M^s_{4z} + P_{0z}*Q^d_{g0x} - P_{1g1}\cos\phi_1*Q^d_{g1y} + (P_{01}+P_{1g1}\sin\phi_1)Q^d_{g1x} - (P_{12}\cos\phi_1+P_{2g2}\cos\phi_2)Q^d_{g2y} + (P_{01}+P_{12}\sin\phi_1+P_{2g2}\sin\phi_2)Q^d_{g2x} - (P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{3g3}\cos\phi_3)Q^d_{g3y} + (P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{3g3}\sin\phi_3)Q^d_{g3x} - T^d_{g1z} - T^d_{g2z} - T^d_{g3z} \tag{3.106}$$

Inertial reactions at Pin #1:

$$Q^d_{g1x} = R^d_{1x} - F^s_{de}\cos\phi_5 + F^s_{ab}\cos\phi_4 - R^s_{2x} - F^d_{de}\cos\phi_6 + F^d_{ab}\cos\phi_4 - R^d_{2x} \tag{3.107}$$

$$\begin{aligned} Q^d_{g1y} &= R^d_{1y} - F^s_{de}\sin\phi_5 + W_1 + F^s_{ab}\sin\phi_4 - R^s_{2y} - \\ &\quad F^d_{de}\sin\phi_6 + F^d_{ab}\sin\phi_4 - R^d_{2y} \end{aligned} \tag{3.108}$$

$$Q^d_{g1z} = R^d_{1z} - R^s_{2z} - R^d_{2z} \tag{3.109}$$

$$\begin{aligned} T^d_{g1x} &= M^d_{1x} - P_{12}\sin\phi_1 * R^s_{2z} - M^s_{2x} - P_{1g1}\sin\phi_1 * Q^d_{g1z} - \\ &\quad P_{12}\sin\phi_1 * R^d_{2z} - M^d_{2x} \end{aligned} \tag{3.110}$$

$$\begin{aligned} T^d_{g1y} &= M^d_{1y} + P_{12}\sin\phi_1 * R^s_{2z} - M^s_{2y} + P_{1g1}\cos\phi_1 * Q^d_{g1z} + \\ &\quad P_{12}\cos\phi_1 * R^d_{2z} - M^d_{2y} \end{aligned} \tag{3.111}$$

$$\begin{aligned} T^d_{g1z} &= -P_{1b} * \sin\phi_1 * F^d_{ab} + P_{1b} * \cos\phi_1 * F^d_{ab} * \sin\phi_4 + \\ &\quad F^d_{de} * \sin\phi_5 * P_{1d} * \cos\phi_1 - F^d_{de} * \cos\phi_5 * P_{1d} * \sin\phi_1 - \\ &\quad W_1 * P_{1g1} * \cos\phi_1 + P_{12} * \cos\phi_1 * R^d_{2y} - \\ &\quad P_{12} * \sin\phi_1 * R^d_{2x} \end{aligned} \tag{3.112}$$

Inertial Reactions at Pin #2:

$$Q^d_{g2x} = R^d_{2x} + F^s_{de}\cos\phi_5 - R^s_{3x} + F^d_{de}\cos\phi_5 - R^d_{3x} \tag{3.113}$$

$$\begin{aligned} Q^d_{g2y} &= R^d_{2y} + W_2 - R^s_{3y} + F^s_{de}\sin\phi_5 + \\ &\quad F^d_{de}\sin\phi_5 - R^d_{3y} \end{aligned} \tag{3.114}$$

$$Q^d_{g2z} = R^d_{2z} - R^s_{3z} - R^d_{3z} \tag{3.115}$$

$$\begin{aligned} T^d_{g2x} &= M^d_{2x} - P_{23}\sin\phi_2 * R^s_{3z} - M^s_{3x} - P_{2g2}\sin\phi_2 * Q^d_{g2z} - \\ &\quad P_{23}\sin\phi_2 * R^d_{3z} - M^d_{3x} \end{aligned} \tag{3.116}$$

$$\begin{aligned} T^d_{g2y} &= M^d_{2y} + P_{23}\cos\phi_2 * R^s_{3z} - M^s_{3y} - P_{2g2}\cos\phi_2 * Q^d_{g2z} + \\ &\quad P_{23}\cos\phi_2 * R^d_{3z} - M^d_{3y} \end{aligned} \tag{3.117}$$

$$T^d_{g2z} = P_{2e} * \cos\phi_6 * F^d_{de} * \sin\phi_5 - P_{2e} * \sin\phi_6 * F^d_{de} * \cos\phi_5 +$$

$$P_{23}*\cos\phi_2*R^d_{3y} - P_{23}*\sin\phi_2*R^d_{3x} -$$
$$W_2*P_{2g2}*\cos\phi_2 \qquad (3.118)$$

Inertial Reactions at Pin #3:

$$Q^d_{g3x} = R^d_{3x} - R^s_{4x} \qquad (3.119)$$
$$Q^d_{g3y} = R^d_{3y} + W_3 - R^s_{4y} \qquad (3.120)$$
$$Q^d_{g3z} = R^d_{3z} - R^s_{4z} \qquad (3.121)$$
$$T^d_{g3x} = M^d_{3x} - P_{34}\sin\phi_3*R^s_{4z} - M^s_{4x} -$$
$$P_{3g3}\sin\phi_3 * Q^d_{g3z} \qquad (3.122)$$
$$T^d_{g3y} = M^d_{3y} + P_{34}\cos\phi_3*R^s_{4z} - M^s_{4y} +$$
$$P_{3g3}\cos\phi_3 * Q^d_{g3z} - M^d_{3y} \qquad (3.123)$$
$$T^d_{g3z} = M^d_{3z} - P_{34}\cos\phi_3*R^s_{4y} + P_{34}\sin\phi_3*R^s_{4x} + P_{3g3}\cos\phi_3*W_3 - M^s_{4z} -$$
$$P_{3g3}\cos\phi_3 * Q^d_{g3y} + P_{3g3}\sin\phi_3 * Q^d_{g3x} \qquad (3.124)$$

Where:

R^d_{ij} — Dynamic force reaction at Pin #i in the j direction

M^d_{ij} — Dynamic moment reaction at Pin #i in the j direction

Q^d_{gij} — Inertial force reaction at Pin #i in the j direction

T^d_{gij} — Inertial moment reaction at Pin #i in the j direction

The inertial reactions were found by initially calculating the terms at Pin #3 and then working in descending pin order.

<u>3.64 Determine the Velocities and Accelerations.</u> Once the inertial terms had been calculated, the angular velocities and accelerations could be determined. Angular accelerations could be extracted from the inertial torques:

$$T^d_{g0x} = \bar{I}_{xx}*\alpha_{0x} \qquad T^d_{g0y} = \bar{I}_{yy}*\alpha_{0y} \qquad T^d_{g0z} = \bar{I}*_{zz}\alpha_{0z} \qquad (3.125)$$
$$T^d_{g1x} = \bar{I}_{xx}*\alpha_{1x} \qquad T^d_{g1x} = \bar{I}_{yy}*\alpha_{1y} \qquad T^d_{g0z} = \bar{I}*_{zz}\alpha_{1z} \qquad (3.126)$$

$$T^d_{g2x} = \bar{I}_{xx} * \alpha_{2x} \qquad T^d_{g2y} = \bar{I}_{yy} * \alpha_{2y} \qquad T^d_{g2z} = \bar{I}^*_{zz} \alpha_{2z} \tag{3.127}$$

The angular velocities were then determined based on the equation

$$\vec{\omega} = \omega_i + \vec{\alpha} * \delta t \tag{3.128}$$

$$\omega_{0x} = \omega_{iox} + \alpha_{0x} * \delta t \qquad \omega_{0y} = \omega_{ioy} + \alpha_{0y} * \delta t$$

$$\omega_{0z} = \omega_{ioz} + \alpha_{0z} * \delta t \tag{3.129}$$

$$\omega_{1x} = \omega_{i1x} + \alpha_{1x} * \delta t \qquad \omega_{1y} = \omega_{i1y} + \alpha_{1y} * \delta t$$

$$\omega_{1z} = \omega_{i1z} + \alpha_{1z} * \delta t \tag{3.130}$$

$$\omega_{2x} = \omega_{i2x} + \alpha_{2x} * \delta t \qquad \omega_{2y} = \omega_{i2y} + \alpha_{2y} * \delta t$$

$$\omega_{2z} = \omega_{i2z} + \alpha_{2z} * \delta t \tag{3.131}$$

Where:

ω_i = Initial angular velocity

ω_{ij} — Angular velocity at Pin #i in the j direction

α_{ij} — Angular acceleration at Pin #i in the j direction

For Segment #1, the velocity was

$$\vec{V}_1 = \vec{V}_0 + \vec{\Omega} \times \vec{r}_{2/1} + (\dot{\vec{r}}_{2/1})_{A_{xyz}}. \tag{3.131}$$

Where

$$(\dot{\vec{r}}_{1/0})_{A_{xyz}} = \vec{\omega}_1 \times \vec{r}_{2/1} \tag{3.132}$$

$\vec{\Omega}$ — Angular velocity of Base, ω_0

$\vec{V}_0$ — Velocity of Base, = 0.0

$\vec{r}_{2/1}$ — Vector from Pin #1 to Pin #2

$\vec{V}_0$ was equal to zero because the point 0 or Pin #0, was defined to be stationary. The vector $\vec{r}_{2/1}$ could also be explained graphically through figure 3.26. Simplifying equation 3.131

$$\vec{V}_1 = \vec{\omega}_0 \times \vec{r}_{2/1} + \vec{\omega}_1 \times \vec{r}_{2/1}. \tag{3.133}$$

The vector $\vec{r}_{2/1}$ was a variable based on the deformation of segment #1. The vector components of $\vec{r}_{2/1}$ are equal to the deformed coordinate information of the various levels of Segment #1

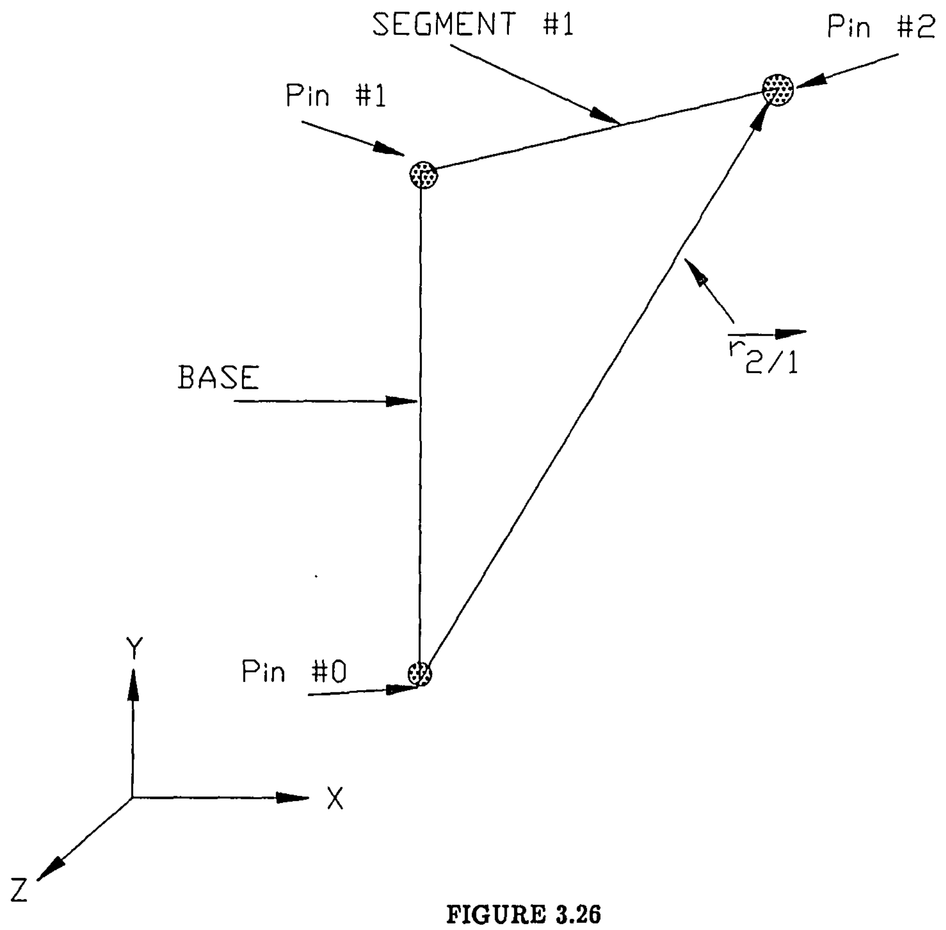


FIGURE 3.26

VECTORS AND PIN LOCATION FOR THE
VELOCITIES AND ACCELERATIONS OF SEGMENT #1

$$\vec{r}_{2/1} = \text{S1DT}(1,1,L)\hat{i} + \text{S1DT}(2,1,L)\hat{j} + \text{S1DT}(3,1,L)\hat{k} \tag{3.134}$$

Where:

L — Level number

The translational velocities were determined at points along the beam equal to each level. For simplicity, let

$$r_{2/1}x = \text{S1DT}(1,1,L)\hat{i} \tag{3.135}$$

$$r_{2/1}y = \text{S1DT}(2,1,L)\hat{j} \tag{3.136}$$

$$r_{2/1}z = \text{S1DT}(3,1,L)\hat{k} \tag{3.137}$$

To complete the calculation for the velocity of Segment #1, the two cross products would be determined.

$$\vec{\omega}_0 \text{ x } \vec{r}_{2/1} = \begin{vmatrix} i & j & k \\ \omega_{0x} & \omega_{0y} & \omega_{0z} \\ r_{1/0}x & r_{1/0}y & r_{1/0}z \end{vmatrix} \tag{3.138}$$

Since the base was not able to rotate about the x or z axes. So, the velocities and accelerations about those axes were zero. This simplified equation 3.151 to

$$\vec{\omega}_0 \text{ x } \vec{r}_{2/1} = r_{2/1}z^*\omega_{0y}\ \hat{i} - r_{2/1}x^*\omega_{0y}\ \hat{k} \tag{3.139}$$

The next cross product was

$$\vec{\omega}_1 \text{ x } \vec{r}_{2/1} = \begin{vmatrix} i & j & k \\ \omega_{1x} & \omega_{1y} & \omega_{1z} \\ r_{1/0}x & r_{1/0}y & r_{1/0}z \end{vmatrix} \tag{3.140}$$

$$\begin{aligned} \vec{\omega}_1 \text{ x } \vec{r}_{2/1} = {} & (r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z})\hat{i} + \\ & (r_{2/1}x^*\omega_{1z} - r_{2/1}z^*\omega_{1x})\hat{j} + \\ & (r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y})\hat{k}. \end{aligned} \tag{3.141}$$

The velocity of Segment #1 then, at a given distance along Segment #1 was the sum of the two cross products:

$$\begin{aligned} \vec{V}_1 = {} & (r_{2/1}z^*\omega_{0y} + r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z})\hat{i} + \\ & (r_{2/1}x^*\omega_{1z} - r_{2/1}z^*\omega_{1x})\hat{j} + \\ & (-r_{2/1}x^*\omega_{0y} + r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y})\hat{k}. \end{aligned} \tag{3.142}$$

The acceleration of Segment #1 was based on the equation

$$\vec{a}_1 = \vec{a}_0 + \dot{\vec{\Omega}} \times \vec{r}_{2/1} + \vec{\Omega} \times (\vec{\Omega} \times \vec{r}_{2/1}) + \\ 2\vec{\Omega} \times (\dot{\vec{r}}_{1/0})_{A_{xyz}} + (\vec{r}_{2/1})_{A_{xyz}} \tag{3.143}$$

Where:

$\vec{a}_0$ — Acceleration of the point Pin #1 = 0.0

$$\dot{\vec{\Omega}} = \dot{\vec{\omega}}_0 = \vec{\alpha}_0 \tag{3.144}$$

$$(\dot{\vec{r}}_{1/0})_{A_{xyz}} = \vec{\omega}_1 \times \vec{r}_{2/1} \tag{3.145}$$

$$(\vec{r}_{2/1})_{A_{xyz}} = \vec{\omega}_1 \times \vec{\omega}_1 \times \vec{r}_{2/1} + \vec{\alpha}_1 \times \vec{r}_{2/1}. \tag{3.146}$$

Substituting equations 3.157 through 3.159 into 3.156 yielded the acceleration to be

$$\vec{a}_1 = \{\vec{\alpha}_0 \times \vec{r}_{2/1}\} + \{\vec{\omega}_0 \times (\vec{\omega}_0 \times \vec{r}_{2/1})\} + \{2\vec{\omega}_0 \times \dot{\vec{r}}_{1/0}\} + \\ \{\vec{\omega}_1 \times \vec{\omega}_1 \times \vec{r}_{2/1}\} + \{\vec{\alpha}_1 \times \vec{r}_{2/1}\} \tag{3.147}$$

The cross—products were:

$$\{\vec{\alpha}_0 \times \vec{r}_{2/1}\} = \\ r_{2/1}z^*\alpha_0 y\ \hat{i} - r_{2/1}x^*\alpha_0 y\ \hat{k} \tag{3.148}$$

$$\{\vec{\omega}_0 \times (\vec{\omega}_0 \times \vec{r}_{2/1})\} = \\ (-r_{2/1}x^*\omega_{0y}^2)\hat{i} + (-r_{2/1}z^*\omega_{0x}^2)\hat{k}. \tag{3.149}$$

$$\{2\vec{\omega}_0 \times \dot{\vec{r}}_{1/0}\} = \\ 2^*\omega_{0y}^2(r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y})\hat{i} + \\ 2^*\omega_{0y}^2(r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z})\hat{k} \tag{3.150}$$

$$\{\vec{\omega}_1 \times \vec{\omega}_1 \times \vec{r}_{2/1}\} = \\ \{\omega_{1y}(r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y}) - \\ \omega_{1z}(r_{2/1}x^*\omega_{1z} - r_{2/1}z^*\omega_{1x})\}\hat{i} \\ \{\omega_{1z}(r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z}) - \\ \omega_{1x}(r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y})\}\hat{j}$$

$$\{\omega_{1x}(r_{2/1}x^*\omega_{1z} - r_{2/1}z^*\omega_{1x}) -$$
$$\omega_{1y}(r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z})\}\hat{k} \tag{3.151}$$

$$\{\vec{\alpha}_1 \times \vec{r}_{2/1}\} =$$
$$\{r_{2/1}z^*\alpha_{1y} - r_{2/1}y^*\alpha_{1z}\}\hat{i} +$$
$$\{r_{2/1}x^*\alpha_{1z} - r_{2/1}z^*\alpha_{1x}\}\hat{j} +$$
$$\{r_{2/1}y^*\alpha_{1x} - r_{2/1}x^*\alpha_{1y}\}\hat{k} \tag{3.152}$$

The velocity of Segment #2 was described by the equation

$$\vec{V}_2 = \vec{V}_1 + \vec{\Omega}_{xyz} \times \vec{r}_{3/2} + (\dot{\vec{r}}_{3/2})_{A_{xyz}}. \tag{3.153}$$

Where:

$\vec{V}_1$ — Velocity of Segment #1 at Pin #2

$\vec{\Omega}_{xyz}$ — The Sum of the Angular Velocity of the Base

and Segment #1, $\omega_{oy} + \omega_{1x} + \omega_{1y} + \omega_{1z}$ (3.154)

$\vec{r}_{3/2}$ — Vector from Pin #2 to Pin #3

$$(\dot{\vec{r}}_{3/2})_{A_{xyz}} = \omega_2 \times r_{3/2} \tag{3.155}$$

The vector $\vec{r}_{3/2}$ could also be explained graphically through figure 3.27. Simplifying equation 3.166

$$\vec{V}_2 = \vec{V}_1 + \vec{\Omega}_{xyz} \times \vec{r}_{3/2} + \vec{\omega}_2 \times \vec{r}_{3/2}. \tag{3.156}$$

The vector $\vec{r}_{3/2}$ was a variable based on the deformation of Segment #2. The vector components of $\vec{r}_{3/2}$ were equal to the deformed coordinate information of the various levels of Segment #2

$$\vec{r}_{3/2} = \text{S2DT}(1,1,L)\hat{i} + \text{S2DT}(2,1,L)\hat{j} + \text{S2DT}(3,1,L)\hat{k} \tag{3.157}$$

Where:

L — level number

The translational velocities was determined at points along the beam equal to each level. For simplicity, let

$$\vec{r}_{3/2}x = \text{S2DT}(1,1,L)\hat{i} \tag{3.158}$$

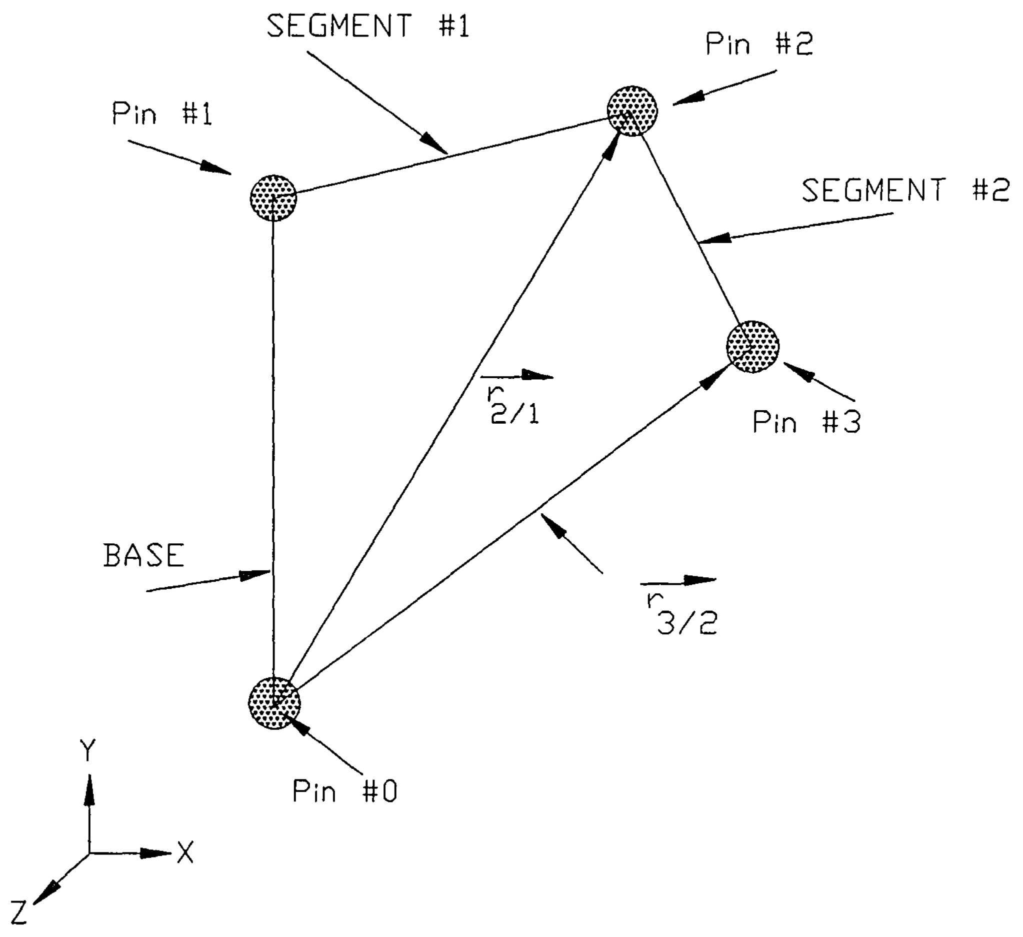


FIGURE 3.27

VECTORS AND PIN LOCATION FOR THE VELOCITIES AND ACCELERATIONS OF SEGMENT #2

$$\vec{r}_{3/2}y = \text{S2DT(2,1,L)}\hat{j} \tag{3.159}$$

$$\vec{r}_{3/2}z = \text{S2DT(3,1,L)}\hat{k} \tag{3.160}$$

Also, for simplicity, Ω_{xyz} was equated to

$$\vec{\Omega}_{xyz} = \omega_x\hat{i} + \omega_y\hat{j} + \omega_z\hat{k} \tag{3.161}$$

Where:

$$\omega_x = \omega_{1x} \tag{3.162}$$

$$\omega_y = \omega_{oy} + \omega_{1y} \tag{3.163}$$

$$\omega_z = \omega_{1z} \tag{3.164}$$

To complete the calculation for the velocity of Segment #2, the two cross–products were determined:

$$\begin{aligned}\vec{\Omega}_{xyz} \times \vec{r}_{3/2} = \\ &(r_{3/2}z^*\omega_y - r_{3/2}y^*\omega_z)\hat{i} + \\ &(r_{3/2}x^*\omega_z - r_{3/2}z^*\omega_x)\hat{j} + \\ &(r_{3/2}y^*\omega_x - r_{3/2}x^*\omega_y)\hat{k}.\end{aligned} \tag{3.165}$$

$$\begin{aligned}\vec{\omega}_2 \times \vec{r}_{3/2} = \\ &(r_{3/2}z^*\omega_{2y} - r_{3/2}y^*\omega_{2z})\hat{i} + \\ &(r_{3/2}x^*\omega_{2z} - r_{3/2}z^*\omega_{2x})\hat{j} + \\ &(r_{3/2}y^*\omega_{2x} - r_{3/2}x^*\omega_{2y})\hat{k}.\end{aligned} \tag{3.166}$$

The velocity of Segment #2 then, at a given distance along Segment #2 was the sum of the two cross–products:

$$\begin{aligned}\vec{V}_2 = (&r_{2/1}z^*\omega_{0y} + r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z} + \\ &r_{3/2}z^*\omega_y - r_{3/2}y^*\omega_z + r_{3/2}z^*\omega_{2y} - r_{3/2}y^*\omega_{2z})\hat{i} + \\ &(r_{2/1}x^*\omega_{1z} - r_{2/1}z^*\omega_{1x} + r_{3/2}x^*\omega_z - \\ &r_{3/2}z^*\omega_x + r_{3/2}x^*\omega_{2z} - r_{3/2}z^*\omega_{2x})\hat{j} + \\ &(-r_{2/1}x^*\omega_{0y} + r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y} + \\ &r_{3/2}y^*\omega_x - r_{3/2}x^*\omega_y + \\ &r_{3/2}y^*\omega_{2x} - r_{3/2}x^*\omega_{2y})\hat{k}.\end{aligned} \tag{3.167}$$

The acceleration of Segment #2 was based on the equation

$$\vec{a}_2 = \vec{a}_1 + \vec{\dot{\Omega}}_{xyz} x\ \vec{r}_{3/2} + \vec{\Omega}_{xyz} x(\vec{\Omega}_{xyz} x\ \vec{r}_{3/2}) + 2\vec{\Omega}_{xyz}\ x\ (\vec{\dot{r}}_{3/2}) + (\vec{r}_{3/2}) \tag{3.168}$$

Where:

$\vec{a}_1$ – Acceleration of Pin #2

$$\vec{\dot{\Omega}}_{xyz} = \vec{\alpha}_2 + \vec{\Omega}_{xyz}\ x\ \vec{\omega}_2 \tag{3.169}$$

$$(\vec{\dot{r}}_{3/2}) = \vec{\omega}_2\ x\ \vec{r}_{3/2} \tag{3.170}$$

$$(\vec{r}_{3/2}) = \vec{\omega}_2\ x\ \vec{\omega}_2\ x\ \vec{r}_{3/2} + \vec{\alpha}_2\ x\ \vec{r}_{3/2}. \tag{3.171}$$

First, it was neccesary to simplify

$$\vec{\dot{\Omega}}_{xyz} = \vec{\alpha}_2 + \vec{\Omega}_{xyz}\ x\ \vec{\omega}_2. \tag{3.172}$$

$$\vec{\Omega}_{xyz}\ x\ \vec{\omega}_2 = \begin{vmatrix} i & j & k \\ \omega_x & \omega_y & \omega_z \\ \omega_{2x} & \omega_{2y} & \omega_{2z} \end{vmatrix} \tag{3.173}$$

$$\vec{\Omega}_{xyz}\ x\ \vec{\omega}_2 = (\omega_y\omega_{2z} - \omega_z\omega_{2y})\hat{i} \quad (\omega_y\omega_{2z} - \omega_z\omega_{2y})\hat{j} \quad (\omega_y\omega_{2z} - \omega_z\omega_{2y})\hat{k} \tag{3.174}$$

Let

$$\dot{\Omega}_x = (\omega_y\omega_{2z} - \omega_z\omega_{2y}) \tag{3.175}$$

$$\dot{\Omega}_y = (\omega_y\omega_{2z} - \omega_z\omega_{2y}) \tag{3.176}$$

$$\dot{\Omega}_z = (\omega_y\omega_{2z} - \omega_z\omega_{2y}) \tag{3.177}$$

Substituting equation 3.184 into 3.181 yielded the acceleration to be

$$\vec{a}_2 = \vec{a}_1 + \{\vec{\dot{\Omega}}_{xyz}\ x\ \vec{r}_{3/2}\} + \{\vec{\Omega}_{xyz}\ x\ (\vec{\Omega}_{xyz}\ x\ \vec{r}_{3/2})\} + \{2\vec{\Omega}_{xyz}\ x\ \vec{\dot{r}}_{3/2}\} + \{\vec{\omega}_2\ x\ \vec{\omega}_2\ x\ \vec{r}_{3/2}\} + \{\vec{\alpha}_2\ x\ \vec{r}_{3/2}\} \tag{3.178}$$

The cross products were:

$$\{\vec{\dot{\Omega}}_{xyz}\ x\ \vec{r}_{3/2}\} = (r_{3/2}z^*\dot{\Omega}_y - r_{3/2}x^*\dot{\Omega}_z)\hat{i} +$$

$$(r_{3/2}x^* \Omega_z - r_{3/2}z^* \Omega_x)\hat{j} +$$
$$(r_{3/2}y^* \Omega_x - r_{3/2}x^* \Omega_y)\hat{k} \qquad (3.179)$$

$$\{\vec{\Omega}_{xyz} \times (\vec{\Omega}_{xyz} \times \vec{r}_{3/2})\} =$$
$$\{\omega_y{}^*(r_{3/2}y^* \omega_x - r_{3/2}x^* \omega_y) -$$
$$\omega_z{}^*(r_{3/2}x^* \omega_z - r_{3/2}z^* \omega_x)\}\ \hat{i} +$$
$$\{\omega_z{}^*(r_{3/2}z^* \omega_y - r_{3/2}y^* \omega_z) -$$
$$\omega_x{}^*(r_{3/2}y^* \omega_x - r_{3/2}x^* \omega_y)\}\hat{j} +$$
$$\{\omega_x{}^*(r_{3/2}x^* \omega_z - r_{3/2}z^* \omega_x) -$$
$$\omega_y{}^*(r_{3/2}z^* \omega_y - r_{3/2}y^* \omega_z)\}\hat{k} \qquad (3.180)$$

$$\{2\vec{\Omega}_{xyz} \times \dot{\vec{r}}_{3/2}\} =$$
$$2^*\{\omega_y{}^*(r_{3/2}y^* \omega_{2x} - r_{3/2}x^* \omega_{2y}) -$$
$$\omega_z{}^*(r_{3/2}x^* \omega_{2z} - r_{3/2}z^* \omega_{2x})\}\hat{i}$$
$$2^*\{\omega_z{}^*(r_{3/2}z^* \omega_{2y} - r_{3/2}y^* \omega_{2z}) -$$
$$\omega_x{}^*(r_{3/2}y^* \omega_{2x} - r_{3/2}x^* \omega_{2y})\}\hat{j}$$
$$2^*\{\omega_x(r_{3/2}x^* \omega_{2z} - r_{3/2}z^* \omega_{2x}) -$$
$$\omega_y{}^*(r_{3/2}z^* \omega_{2y} - r_{3/2}y^* \omega_{2z})\}\hat{k} \qquad (3.181)$$

$$\{\vec{\omega}_2 \times \vec{\omega}_2 \times \vec{r}_{3/2}\} =$$
$$\{\omega_{2y}(r_{3/2}y^* \omega_{2x} - r_{3/2}x^* \omega_{2y}) -$$
$$\omega_{2z}(r_{3/2}x^* \omega_{2z} - r_{3/2}z^* \omega_{2x})\}\hat{i}$$
$$\{\omega_{2z}(r_{3/2}z^* \omega_{2y} - r_{3/2}y^* \omega_{2z}) -$$
$$\omega_{2x}(r_{3/2}y^* \omega_{2x} - r_{3/2}x^* \omega_{2y})\}\hat{j}$$
$$\{\omega_{2x}(r_{3/2}x^* \omega_{2z} - r_{3/2}z^* \omega_{2x}) -$$
$$\omega_{2y}(r_{3/2}z^* \omega_{2y} - r_{3/2}y^* \omega_{2z})\}\hat{k} \qquad (3.182)$$

$$\{\vec{\alpha}_2 \times \vec{r}_{2/1}\} =$$
$$\{r_{3/2}z^* \alpha_{2y} - r_{3/2}y^* \alpha_{2z}\}\hat{i} +$$
$$\{r_{3/2}x^* \alpha_{2z} - r_{3/2}z^* \alpha_{2x}\}\hat{j} +$$

$$\{r_{3/2}y^{*}\alpha_{2x} - r_{3/2}x^{*}\alpha_{2y}\}\hat{k} \tag{3.183}$$

<u>**3.65 Determine the New Position of the Arm in Time.**</u> The next place in time was a function of the time step, the present magnitude of the angles, differential change in angles, the velocities, and the accelerations. The differential changes in angles were based on the same equations as those in the static analysis. For Segment #1, the new location, or angle, was determined from the equation

$$\phi_1 = \phi_1 + \delta\phi_1 + \Omega_1{}^{*}\delta t + 0.5^{*}\forall_1{}^{*}\delta t^2 \tag{3.184}$$

The magnitude of the angular velocity, Ω_1 of Segment #1 was the anular velocity of Segment #1 in the Z—directionn. The magnitude of the angular acceleration,$\forall_1$ of Segment #1 was also the angular acceleration of Segment #1 in the Z—direciton. The differential change in angle ϕ_1 was $\delta\phi_1$ and the time step was δt.

Similarly, for Segment #2,

$$\phi_2 = \phi_2 + \delta\phi_2 + \Omega_2{}^{*}\delta t + 0.5^{*}\forall_2{}^{*}\delta t^2 \tag{3.185}$$

The magnitude of the angular velocity, Ω_2 of Segment #2 was the angualr velocity of Segment #2 in the Z—direciton and the magnitude of the angular acceleration for Segment #2 was the angular acceleration for Segment #2 in the Z—direction.

The differential change in angle ϕ_2 was $\delta\phi_2$.

<u>**3.66 Determine the Dynamic Reactions.**</u> The dynamic reactions were based on the equations :

$$\Sigma F_x = ma_x \qquad \Sigma M_x = \bar{I}_{xx}\alpha_x \tag{3.186}$$

$$\Sigma F_y = ma_y \qquad \Sigma M_y = \bar{I}_{yy}\alpha_y \tag{3.187}$$

$$\Sigma F_z = ma_z \qquad \Sigma M_z = \bar{I}_{zz}\alpha_z . \tag{3.188}$$

Bannoura based his dynamic analysis on the above equations to reach a basic format of

$$\Sigma F = m\vec{a} = Q_g^d \tag{3.189}$$

and

$$\Sigma M = I\vec{\alpha} = T_g^d . \tag{3.190}$$

Where:

m — Mass of link

a — Acceleration of link

α — Angular acceleration of link

I — Mass Moment of Inertia Matrix

Q^d_g — Inertial force reaction

T^d_g — Inertial moment reaction

Dynamic Reactions at Pin #0

The dynamic reactions were found at Pins #0, #1, #2, and #3.

Dynamic Reactions at Pin #0 (See Figures 3.17 and 3.18)

$$R^d_{0x} = R^s_{4x} + Q^d_{g0x} + Q^d_{g1x} + Q^d_{g2x} + Q^d_{g3x} \quad (3.191)$$

$$R^d_{0y} = -W_0 - W_1 - W_2 - W_3 + R^s_{4y} + Q^d_{g0y} + Q^d_{g1y} + Q^d_{g2y} + Q^d_{g3y} \quad (3.192)$$

$$R^d_{0z} = R^s_{4z} + Q^d_{g0z} + Q^d_{g1z} + Q^d_{g2z} + Q^d_{g3z} \quad (3.193)$$

$$\begin{aligned} M^d_{0x} =& (P_{01} + P_{12}\sin\phi_1 + P_{23}\sin\phi_2 + P_{34}\sin\phi_3)R^s_{4z} + M^s_{4x} + P_{0g0} * Q^d_{g0x} + \\ & (P_{01} + P_{1g1}\sin\phi_1)Q^d_{g1z} + \\ & (P_{01} + P_{12}\sin\phi_1 + P_{2g2}\sin\phi_2)Q^d_{g2z} + \\ & (P_{01} + P_{12}\sin\phi_1 + P_{23}\sin\phi_2 + P_{3g3}\sin\phi_3)Q^d_{g3z} + \\ & T^d_{g0x} + T^d_{g1x} + T^d_{g2x} + T^d_{g3x} \end{aligned} \quad (3.194)$$

$$\begin{aligned} M^d_{0y} =& -(P_{12}\cos\phi_1 + P_{23}\cos\phi_2 + P_{34}\cos\phi_3)R^s_{4z} + M^s_{4y} - \\ & P_{1g1}\cos\phi_1 * Q^d_{g1z} - (P_{12}\cos\phi_1 + P_{2g2}\cos\phi_2)Q^d_{g2z} - \\ & (P_{12}\cos\phi_1 + P_{23}\cos\phi_2 + P_{3g3}\cos\phi_3)Q^d_{g3z} + T^d_{g0y} + T^d_{g1y} + \\ & T^d_{g2y} + T^d_{g3y} \end{aligned} \quad (3.195)$$

$$\begin{aligned} M^d_{0z} =& -P_{1g1}\cos\phi_1 * W_1 - (P_{12}\cos\phi_2 + P_{2g2}\cos\phi_2)W_2 - \\ & (P_{12}\cos\phi_1 + P_{23}\cos\phi_2 + P_{3g3}\cos\phi_3)W_3 + \\ & (P_{12}\cos\phi_1 + P_{12}\cos\phi_2 + P_{34}\cos\phi_3)R^s_{4y} - \\ & (P_{01} + P_{12}\sin\phi_1 + P_{23}\sin\phi_2 + P_{34}\sin\phi_3)R^s_{4x} + M^s_{4z} - \end{aligned}$$

$$P_{0z}{*}Q^{d}_{g0x}+P_{1g1}\cos\phi_{1}{*}Q^{d}_{g1y}-(P_{01}+P_{1g1}\sin\phi_{1})Q^{d}_{g1x}+$$
$$(P_{12}\cos\phi_{1}+P_{2g2}\cos\phi_{2})Q^{d}_{g2y}-(P_{01}+P_{12}\sin\phi_{1}+P_{2g2}\sin\phi_{2})Q^{d}_{g2x}+$$
$$(P_{12}\cos\phi_{1}+P_{23}\cos\phi_{2}+P_{3g3}\cos\phi_{3})Q^{d}_{g3y}-$$
$$(P_{01}+P_{12}\sin\phi_{1}+P_{23}\sin\phi_{2}+P_{3g3}\sin\phi_{3})Q^{d}_{g3x}+$$
$$T^{d}_{g0z}+T^{d}_{g1z}+T^{d}_{g2z}+T^{d}_{g3z} \qquad (3.196)$$

Dynamic Reactions at Pin #1 (See Figures 3.17 and 3.19)

$$R^{d}_{1x}=F^{s}_{de}\cos\phi_{5}-F^{s}_{ab}\cos\phi_{4}+R^{s}_{2x}+F^{d}_{de}\cos\phi_{6}+$$
$$Q^{d}_{g1x}-F^{d}_{ab}\cos\phi_{4}+R^{d}_{2x} \qquad (3.197)$$

$$R^{d}_{1y}=F^{s}_{de}\sin\phi_{5}-W_{1}-F^{s}_{ab}\sin\phi_{4}+R^{s}_{2y}+$$
$$F^{d}_{de}\sin\phi_{6}+Q^{d}_{g1y}-F^{d}_{ab}\sin\phi_{4}+R^{d}_{2y} \qquad (3.198)$$

$$R^{d}_{1z}=R^{s}_{2z}+Q^{d}_{g1z}+R^{d}_{2z} \qquad (3.199)$$

$$M^{d}_{1x}=P_{12}\sin\phi_{1}{*}R^{s}_{2z}+M^{s}_{2x}+P_{1g1}\sin\phi_{1}{*}Q^{d}_{g1z}+$$
$$P_{12}\sin\phi_{1}{*}R^{d}_{2z}+T^{d}_{g1x}+M^{d}_{2x} \qquad (3.200)$$

$$M^{d}_{1y}=-P_{12}\sin\phi_{1}{*}R^{s}_{2z}+M^{s}_{2y}-P_{1g1}\cos\phi_{1}{*}Q^{d}_{g1z}-$$
$$P_{12}\cos\phi_{1}{*}R^{d}_{2z}+T^{d}_{g1y}+M^{d}_{2y} \qquad (3.201)$$

$$M^{d}_{1z}=0.0 \qquad (3.202)$$

Dynamic Reactions at Pin #2 (See Figures 3.17 and 3.20)

$$R^{d}_{2x}=-F^{s}_{de}\cos\phi_{5}+R^{s}_{3x}+Q^{d}_{g2x}-F^{d}_{de}\cos\phi_{5}+R^{d}_{3x} \qquad (3.203)$$

$$R^{d}_{2y}=-W_{2}+R^{s}_{3y}-F^{s}_{de}\sin\phi_{5}+Q^{d}_{g2y}-F^{d}_{de}\sin\phi_{5}+R^{d}_{3y} \qquad (3.204)$$

$$R^{d}_{2z}=R^{s}_{3z}+Q^{d}_{g2z}+R^{d}_{3z} \qquad (3.205)$$

$$M^{d}_{2x}=P_{23}\sin\phi_{2}{*}R^{s}_{3z}+M^{s}_{3x}+P_{2g2}\sin\phi_{2}{*}Q^{d}_{g2z}+$$
$$P_{23}\sin\phi_{2}{*}R^{d}_{3z}+T^{d}_{g2x}+M^{d}_{3x} \qquad (3.206)$$

$$M^{d}_{2y} = -P_{23}\cos\phi_{2} * R^{s}_{3z} + M^{s}_{3y} + P_{2g2}\cos\phi_{2} * Q^{d}_{g2z} - P_{23}\cos\phi_{2} * R^{d}_{3z} + T^{d}_{g2y} + M^{d}_{3y} \tag{3.207}$$

$$M^{d}_{2z} = 0.0 \tag{3.208}$$

Dynamic Reactions At Pin #3 (See Figures 3.17 and 3.21)

$$R^{d}_{3x} = R^{s}_{4x} + Q^{d}_{g3x} \tag{3.209}$$

$$R^{d}_{3y} = -W_{3} + R^{s}_{4y} + Q^{d}_{g3y} \tag{3.210}$$

$$R^{d}_{3z} = R^{s}_{4z} + Q^{d}_{g3z} \tag{3.211}$$

$$M^{d}_{3x} = P_{34}\sin\phi_{3} * R^{s}_{4z} + M^{s}_{4x} + P_{3g3}\sin\phi_{3} * Q^{d}_{g3z} + T^{d}_{g3x} \tag{3.212}$$

$$M^{d}_{3y} = -P_{34}\cos\phi_{3} * R^{s}_{4z} + M^{s}_{4y} - P_{3g3}\cos\phi_{3} * Q^{d}_{g3z} + T^{d}_{g3y} \tag{3.213}$$

$$M^{d}_{3z} = P_{34}\cos\phi_{3} * R^{s}_{4y} - P_{34}\sin\phi_{3} * R^{s}_{4x} - P_{3g3}\cos\phi_{3} * W_{3} + M^{s}_{4z} + P_{3g3}\cos\phi_{3} * Q^{d}_{g3y} - P_{3g3}\sin\phi_{3} * Q^{d}_{g3x} + T^{d}_{g3z} \tag{3.214}$$

<u>3.67 Determine the Deformations.</u> As in the static analysis, the dynamic analysis considered Segments #1 and #2 to be slender flexible elastic beams. Four types of deformation existed as a result of the loading, geometry, and material characteristics:

(1) Torsional twist about the x–axis

(2) Axial extension in the x direction.

(3) Deflection in the y direction

(4) Deflection in the z direction

Torsional twist and axial extension were based upon the following equations of Bannoura,

$$\theta_{x1} = \frac{T_{g1x}\cos\phi_{1} + T_{g1y}\sin\phi_{1}}{L_{1} J_{1x} G_{1}}\left(L_{1}x_{1} - \frac{x_{1}^{2}}{2}\right) + \frac{M^{d}_{2x}\cos\phi_{1} + M^{d}_{2y}\sin\phi_{1}}{L_{1} J_{1x} G_{1}}(x_{1}) \tag{3.215}$$

$$\theta_{x2}=\frac{T_{g2x}\cos\phi_2+T_{g2y}\sin\phi_2}{L_2J_{2x}G_2}\left(L_2x_2-\frac{x_2^2}{2}\right)+$$
$$\frac{M^d_{3x}\cos\phi_2+M^d_{3y}\sin\phi_2}{L_2J_{2x}G_2}\,(x_2) \tag{3.216}$$

$$u_{x1}=\frac{Q_{g1x}\cos\phi_1+Q_{g1y}\sin\phi_1}{L_1A_1E_1}\left(L_1x_1-\frac{x_1^2}{2}\right)+$$
$$\frac{R^d_{2x}\cos\phi_1+R^d_{2y}\sin\phi_1}{L_1A_1E_1}\,(x_1) \tag{3.217}$$

$$u_{x2}=\frac{Q_{g2x}\cos\phi_2+Q_{g2y}\sin\phi_2}{L_2A_2E_2}\left(L_2x_2-\frac{x_2^2}{2}\right)+$$
$$\frac{R^d_{3x}\cos\phi_2+R^d_{3y}\sin\phi_2}{L_2A_2E_2}\,(x_2) \tag{3.218}$$

Where:

u_{x1} — Axial extension for Segment #1

u_{x2} — Axial extension for Segment #2

θ_{x1} — Torsional twist about the x axis for Segment #1

θ_{x2} — Torsional twist about the x axis for Segment #2

R^d_{ij} — Static force reaction at Pin #i in the j direction

M^d_{ij} — Static moment reaction at Pin #i in the j direction

L_1 — P_{12} which is the distance between Pins #1 & #2.

L_2 — P_{23} which is the distance between Pins #2 & #3.

J_i — Polar moment of inertia.

G_i — Shear modulus of elasticity.

x_1 — X distance along Segment #1

x_2 — X distance along Segment #2

The function of the dynamic equations for torsional twist and axial extension were the same as that for the static analysis. The dynamic torsional twist also produced a deformation along the x axis. The change in length due to the torsional twist was found through the same algorithm as the static analysis. See figures 3.24 and 3.25.

$$\delta = AB(1 - \cos \theta_{xi}). \tag{3.219}$$

The total deformation along the x axis was then

$$\text{Total Deformation} = \delta + u_{xi}. \tag{3.220}$$

The deflections in the y and z directions were:

$$V_i = \frac{q_{iyi}}{24E_iI_{zi}}\left[x_i^4 - 4L_ix_i^3 + 6L_i^2x_i^2\right] + \frac{q^o_{iyi}}{120E_iI_{zi}L_i}\left[2x_i^5 - 5L_ix_i^4 + 12L_i^3x_i^2\right] + \frac{R^l_{(i+1)yi}}{6E_iI_{zi}}(3L_ix_i^2 - x_i^3) + \frac{M^d_{(i+1)zi}}{2E_iI_{zi}}x_i^2 \tag{3.221}$$

$$\theta_{zi} = \frac{q_{iyi}}{24E_iI_{zi}}\left[4x_i^3 - 12L_ix_i^2 + 12L_ix_i\right] + \frac{q^o_{iyi}}{20E_iI_{zi}L_i}\left[10x_i^4 - 20L_ix_i^3 + 12L_i^3x_i\right] + \frac{R^l_{(i+1)yi}}{6E_iI_{zi}}(6L_ix_i - 3x_i^2) - \frac{M^d_{(i+1)zi}}{E_iI_{zi}}x_i \tag{3.222}$$

$$W_i = \frac{q_{izi}}{24E_iI_{yi}}\left[x_i^4 - 4L_ix_i^3 + 6L_i^2x_i^2\right] + \frac{q^o_{izi}}{120E_iI_{yi}L_i}\left[2x_i^5 - 5L_ix_i^4 + 12L_i^3x_i^2\right] +$$

$$\frac{R^{d}_{(i+1)zi}}{6E_iI_{yi}}(3L_ix_i^2-x_i^3)-\frac{M^{l}_{(i+1)yi}}{2E_iI_{yi}}x_i^2 \tag{3.223}$$

$$\theta_{yi}=-\frac{q_{izi}}{24E_iI_{yi}}\left[4x_i^3-12L_ix_i^2+12L_i^2x_i\right]+\frac{q^0_{izi}}{120E_iI_{yi}L_i}\left[10x_i^4-20L_ix_i^3+24L_i^3x_i\right]+\frac{R^{d}_{(i+1)zi}}{6E_iI_{yi}}(6L_ix_i-3x_i^2)-\frac{M^{l}_{(i+1)yi}}{E_iI_{yi}}x_i \tag{3.224}$$

Where:

$$q^0_{1z1}=\frac{6}{L_1^2}(T_{g1y}\cos\phi_1-T_{g1x}\sin\phi_1) \tag{3.225}$$

$$q^0_{2z2}=\frac{6}{L_2^2}(T_{g2y}\cos\phi_2-T_{g2x}\sin\phi_2) \tag{3.226}$$

$$q^0_{1y1}=\frac{6T_{g1z}}{L_1^2} \tag{3.227}$$

$$q^0_{2y2}=\frac{6T_{g2z}}{L_2^2} \tag{3.228}$$

$$q_{1y1}=\frac{Q_{g1y}\cos\phi_1-Q_{g1x}\sin\phi_1}{L_1}-\frac{W_1}{L_1}\cos\phi_1 \tag{3.229}$$

$$q_{2y2}=\frac{Q_{g2y}\cos\phi_2-Q_{g2x}\sin\phi_2}{L_2}-\frac{W_2}{L_2}\cos\phi_2 \tag{3.230}$$

$$q_{1z1} = \frac{Q_{g1z}}{L_1} \quad (3.231)$$

$$q_{2z2} = \frac{Q_{g2z}}{L_2} \quad (3.232)$$

Where:

V_i — Displacement of segment i in the y direction

θ_{zi} — Slope of displacement for segment i about the z axis

W_i — Displacement of segment i in the z direction

θ_{yi} — Slope of displacement for segment i about the y axis

x_i — Linear distance along the x axis where deflection calculations are determined

q^o_{iji} — Distributed forces per segment i length due to inertial torques in the j direction applied at the center of gravity

q_{iji} — Distributed forces per segment i length due to inertial forces in the j direction applied at the center of gravity

L_i — Distance on segment i from Pin #i to Pin #(i+1)

E_i — Modulus of elasticity of segment i

I_{ij} — Area Moment of Inertia about the i axis for segment j

$M^l_{(i+1)y_i}$ — Moment reaction in the local y_i–axis.

$$M^l_{(i+1)y_i} = M^d_{(i+1)y} \sin\phi_i + M^d_{(i+1)x} \cos\phi_i \quad (3.233)$$

$R^l_{(i+1)y_i}$ — Force reaction in the local y_i–axis.

$$R^l_{(i+1)y} = R^d_{(i+1)y} \cos\phi_i - R^d_{(i+1)x} \sin\phi_i. \quad (3.234)$$

The dynamic analysis could also be considered in a cyclical format. But, the dynamic analysis was dependent upon the user's dynamic input file, **dynin.dat**, which contained different magnitudes of forces for the actuators. Therefore, based on a given pair of applied actuator forces, the eight calculation steps were completed. The deformation analysis was completed over the Segments #1 and #2. The Base, end—effector, Segment #3, and Wing Segment were considered to be rigid. The deformation was determined at incremental steps along the length of both the segments. The first calculation was be at the lower pin, that is at Pin #1 for Segment #1. The deformation would be determined at each interval until the upper pin position was reached, Pin #3 for Segment #2. As the deformation analysis was completed for Segment #2, the analytical cycle would be completed.

CHAPTER 4

PROGRAM IMPLEMENTATON

4.1 Organization of Models

Two computer models have been written. The models were based on a purely static analysis of the robot and on a dynamic analysis. As discussed in Section 3.5, the static loading conditions assumed that the arm was in static equilibrium,

$$\Sigma F = 0.0 \tag{4.1}$$

$$\Sigma M = 0.0 \tag{4.2}$$

This model was called **SMODEL**, (Static Model). The second model analyzed the deflection under dynamic loading conditions. The sum of the forces and moments were then equal to the inertial forces and torques

$$\Sigma\vec{F} = m\vec{a} \tag{4.3}$$

$$\Sigma\vec{M} = I\vec{\alpha} \tag{4.4}$$

This model was called **DMODEL**, (Dynamic Model).

The routines written to accomplish the user interaction, kinematics, three–dimensional graphics, static analysis, and dynamic analysis were basically the same for both models. The main difference was in the global variable file. The variables which were global variables were stored in common blocks. The common blocks were stored in a file called **COMMON**. The dynamic file requires some variables which were not required by the static model. To keep both models independent of each other, two common blocks were created. The common block associated with the static model was named **SCOMMON** and the common block for the dynamic model was named **DCOMMON**. Since the routines associated with the dynamic variables would reference the dcommon block and the static variables would reference the scommon block, two different versions of a subroutine had to be written. All versions associated with the static model have filenames and subroutine names that were prefixed with an "S" and all routines associated with the dynamic model had filenames and subroutine names that were prefixed with a "D".

Subroutines were linked to their calling routine through the FORTRAN statement

```
$INCLUDE"subroutine.f"
```

This allowed the routines to be stored in their own files thereby decreasing the length of the main program. The files were named after the subroutine. For example, the static version of the routine **INSCRN** was stored in the file **sinscrn.f** and the dynamic version was stored in the file **dinscrn.f**.

The routines that were not similar were those associated directly with the static and dynamic analysis. The routine **STATIC** was the control point for the static analysis and the routine **DYNAMIC** was the control point for the dynamic analysis.

STATIC called all routines associated with the determination of the force and moment reactions, angles of actuators, static deformations in the x, y, and z directions, kinematics, graphic images, and user interaction. After the static deflections were found, the position vectors of the deformed coordinates were homogeneously transformed through kinematic equations. For this, **STATIC** called **STFORM**, (Static model — transformations). After the points were kinematically transformed, the objects were ready to be drawn. The routine **SDRROB**, (Static model — draw robot), was called. This routine simple called all the routines required to draw each link. Now, the static analysis was complete and the image was drawn on the screen. The next step was to allow the user to interact with the model. User interaction was accepted through the subroutine **SUSR**, (static, model — user). Based on the type of user interaction, the program control would return to **STATIC** where the entire process would take place again. This continued until the user selected the Q key, to quit or stop the model.

The subroutine **DYNAMIC** followed a format similar to **STATIC**. The major difference was that the analysis steps were encompassed in a Do—loop. The Do—loop was executed as many times as there were entries in the dynamic input file. The dynamic input file, **dynin.dat**, was created by the user. It contained actuator forces as functions of time. By defining the actuator forces, the angles of Segments #1 and #2 were defined. Movement of Segments #1 and #2 was then accomplished by changing the actuator forces.

The main programs were **SMODEL** and **DMODEL**. The purpose of the main programs was to calculate forces, torques, deformations and other kinematic values of the robot and to display on the this analytical information on the screen. The programs called initialization subroutines and the analysis routines (see figures 4.1 and 4.2). The initializing responsibilities were completed as follows:

(1) Initialize the screen

(2) Initialize the variables

(3) Initialize the user input devices

by the routine calls

```
CALL INSCRN
CALL INVAR
CALL INUSR
```

The last subroutine called was the deflection analysis routine, **STATIC** or **DYNAMIC**,

```
CALL (STATIC or DYNAMIC).
```

Program control stayed within the analysis routine until the user either stopped the model or in the case of the dynamic analysis, all the actuator input forces had been analyzed.

As the routines are explained in depth, the main name, **invar**, **inscrn**, etc. will be referenced. Any major differences between the static and dynamic models will be cited. To reference the static routine, look under the routine name prefixed by an "S". To reference the dynamic rooutine, look under the routine name prefixed by a "D".

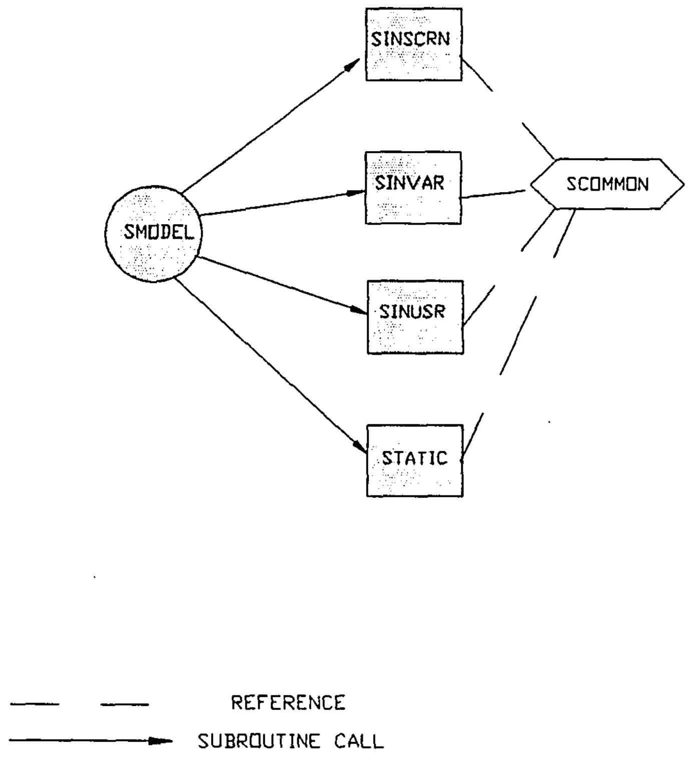


FIGURE 4.1

ORGANIZATION OF STATIC MODEL PROGRAM

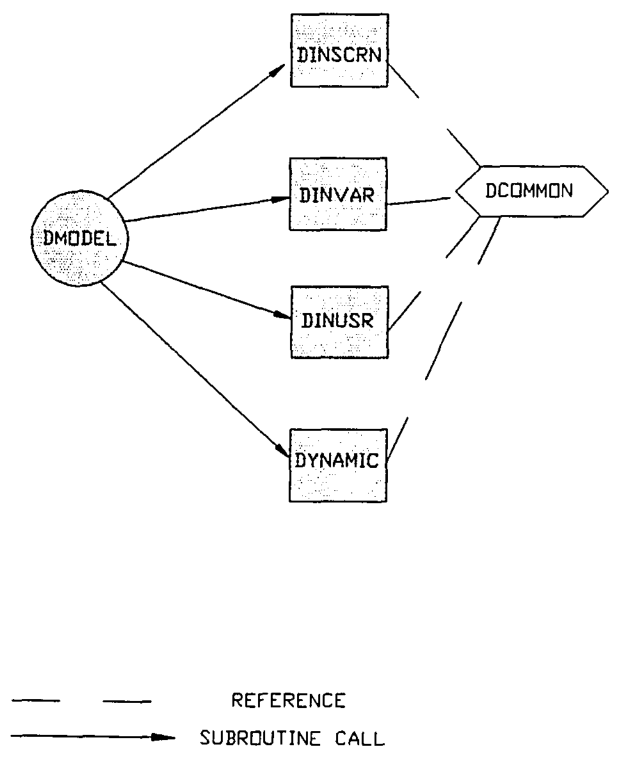


FIGURE 4.2

ORGANIZATION OF DYNAMIC MODEL PROGRAM

4.2 USER INTERACTION

The user could interact with the model through two basic methods. The first method was through the input parameter files. There were two input parameter files, **robots.dat** and **dynin.dat**, Sections 5.3 and 5.4. The first input file was the robot paramcter file. Both models referenced this input file. The second input file was associated with the dynamic model. The user created an input file that contained a list of forces for the two actuators. By varying the forces in the actuators the angles of the segments were altered. The subroutine, **invar**, initialize variables, acted as a control point for all routines associated with reading the parameter file and initializing values of variables. The second method of user interaction was through the hardware.

The robot parameter file allowed the user the greatest control over the program input. The parameter file, **robots.dat** contained a complete description of each link including dimensions and material characteristics as well as initial condition parameters. The input parameter file could be edited by any text editor. On the IRIS, there were two editors, VI, a screen editor and "ed", a line editor. Both the dynamic and static models referenced the file, **robots.dat**.

The first routine called by **invar** was **pread**, (parameter read), (see figure 4.3). **pread** was responsible for opening the input parameter file, **robots.dat** and reading all the values from that file. The file, robots.dat followed an outline type format

A.

1.

a.

b.

2.

a.

b..

B.

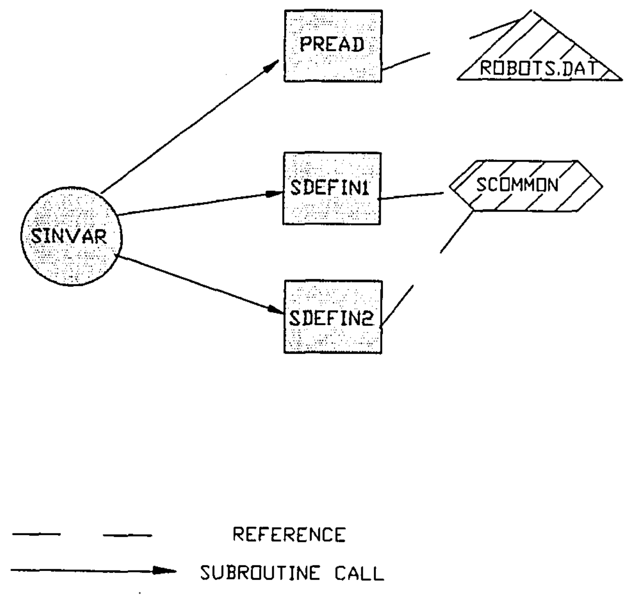


FIGURE 4.3

ORGANIZATION OF THE ROUTINE SINVAR

The outline format controlled how the values were read in and stored. **pread** looked at the first six characters of each line. Values only appeared on lines which were prefaced by a lowercase letter. The lines prefaced by uppercase and numeric lines were descriptive lines telling what link and what attribute of that link were being described. To store the values, two—dimensional arrays were used.

Based on the first two characters, the value was stored in either the array A or B. All information under the heading "A" was stored in the array A and all information under "B" was stored in array B. If a third section of the input parameter file was to be added, an array, 'C', would be required to be defined. By storing the values in arrays at this point, the routine **pread**, could be used by both the static model as well as the dynamic because none of the values were stored as specific global variables. This meant that neither the common block file of the static nor the dynamic model was required for reference. Also since the values were stored in arrays at this point, it was easier to expand the program, if so desired. The dimensions of the arrays were equal to or greater than the number of numeric entries under the uppercase letter and by the largest amount of smaller case letters under a numeric heading. For example, under A, there were eight difference subheadings and the greatest number lowercase letters was twenty under (2.), Segment #1 and also (3.), Segment #2. So, array A was dimensioned to at least, A(8,20). By declaring

```
REAL A(40,40)
```

there was room for expansion. It was assumed that logically, the first group of variables would all be stored in the A array.

The next two spaces of any line in the parameter file determined which row of the array the variable was stored in and the last two spaces determined which column. A formatted read statement read each line of the parameter file. The following was the FORTRAN format statement and read statement:

```
101     FORMAT(3(A2),58X,F12.2)
        READ(1,101)CHAPTER, TITLE, SUBTITLE, VALUE
```

Based on the given line,

b. LENGTH 178

the read statement would have assigned the following values to the variables

CHAPTER = 2 spaces

TITLE = 2 spaces

SUBTITLE = b.

VALUE = 1.78.

The value was stored according to the CHAPTER, TITLE, AND SUBTITLE.

To flag the end of the parameter file, two asteriks appeared in the first two spaces. When CHAPTER equalled "**", the input file was closed and the control of the program returned to **invar**.

The next step was to transfer the values of the arrays into understandable variable names. The routine **definl** (Define, part 1) took the information in the arrays A and B and placed the values into named variables such as SILENG and BASEDI. The values in the arrays were loaded by row into the proper variables:

BASEDI = A(11,1)

BASEHE = A(1,2)

PINIX = A(1,3)

PIN1Y = A(1,4)

BASMAS = A(1,5)

BCMASX = A(1,6)

BCMASY = A(1,7), etc

Since the same parameter file was used for both the static and dynamic models, if there existed a value in the parameter file which was not required for the static model, the routine **sdefinl** ignored the value. If a variables was to be added to the parameter file, the variable name would need to be declared in the common block and then the variable would need to be added in the proper location of the definl routine.

The routine **dinvar** also called the routine **DYREAD** (dynamic read), (see figure 4.4) to read all the dynamic input from the dynamic input file **dynin.dat**. The dynamic input file, **dynin.dat** (see figure 4.4) allowed the user to alter the angles of the segments by changing the forces in the actuators. The user could input a maximum of 998 pairs of forces for actuators AB and DE. This routine was called by the routine **dinvar** after the routine **pread** had been called. **dyread** opened the file **dynin.dat** and read each line through the format and read statements

```
101          FORMAT(1X,13,2(2X, F13.2))
             READ(12,101) I,FD1, FD2.
```

The routine first processed the contents of the variable I represetning the time increment. As long as I was not equal to 999, the values FD1 and FD2 were stored in the two–dimmensional array FACTI. The first row of FACTI stored the forces for actuator AB and the second was for storing for the forces of actuator DE. The variable NFACT stored the number of pairs of actuator forces. During the dynamic analysis, the array containing the actuator forces was referenced. A dynamic analysis was completed for each pair of actuator forces.

Another level of user interaction ocurred through the subroutines **inusr** and **usr**. The first subroutine, **inusr** initialized all input hardware. This involved changing the status of a given device so that if that device was called upon a signal would be added to the events queue. This action was completed by using the IRIS command, QDEVIC. Each device, or variable name had to be sent to the subroutine so that the input device status would be altered.

```
CALL QDEVIC(SW0)
CALL QDEVIC(SW1), etc.
```

If additional input devices were desired, the devices would need to be initialized by simply adding the device name to this file.

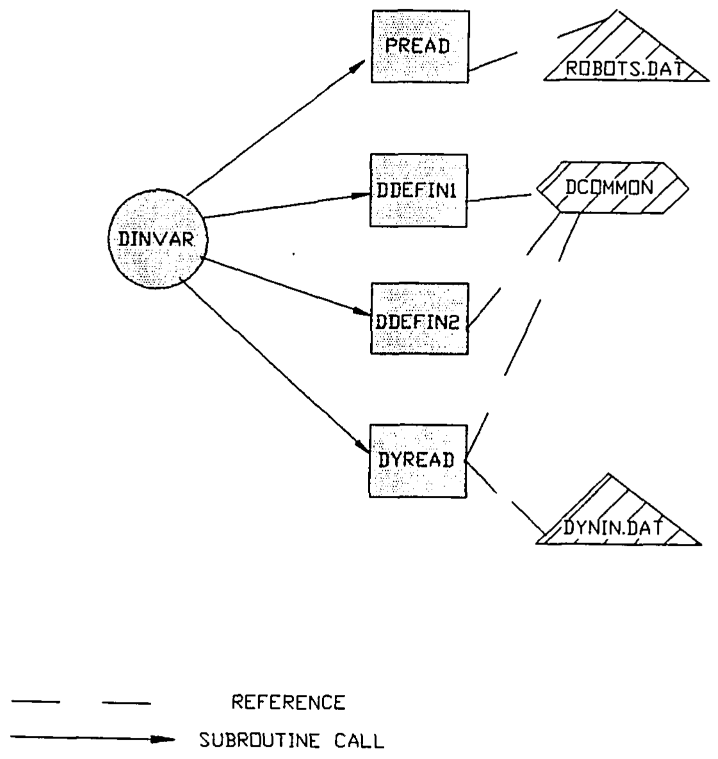


FIGURE 4.4

ORGANIZATION OF THE ROUTINE DINVAR

The second routine written for user interaction was **usr**. This routine carried out an action based on the user interaction device. When the user selected a dial, key or button, a value was sent to the queue. The IRIS function QREAD returned the value that was on the top of the stack. In the following line

DEV = QREAD(VAL)

the variable DEV was the value of the input device which had just been selected by the user. Based on the value of DEV, an action was completed. This action could have been anything from rotating the world coordinates through the mouse to writing the deflections of Pin #2 to an output file. The mouse buttons were responsible for rotating the world, or the user's perspective. The dials were responsible for rotating each element of the arm, and the switches were responsible for allowing information to be written to either the screen or an output file. Each model wrote to a specific output file. For the static model, the output file was **sout.dat**, and for the dynamic, the file was **dout.dat**. Although the dials allowed the user to manipulate the arm by moving the segments, the dynamic model did not allow the user to move Segments #1 or #2 through the dials. This was not allowed because the dynamic model relied on the actuator forces in the dynamic input file to alter the angles of segments #1 and #2. The user interaction capabilities were as follows:

DIAL #1 — Rotate Base Positively

DIAL #2 — Rotate Base Negatively

DIAL #3 — Rotate Segment #1 Positively

DIAL #4 — Rotate Segment #1 Negatively

DIAL #5 — Rotate Segment #2 Positively

DIAL #6 — Rotate Segment #2 Negatively

DIAL #7 — Rotate Segment #3 Positively

DIAL #8 — Rotate Segment #3 Negatively

Mouse #1 — Rotate World about X—Axis

Mouse #2 — Rotate World about Y—Axis

Mouse #3 — Rotate World about Z—Axis

Switch #1 — Angle of Actuator AB

Switch #2 — Static Forces and Moments

Switch #3 — Angular Velocities and Accelerations
of Segment #1

Switch #4 — Segment #1: Deflection and Slope

Switch #5 — Angle of Base

Switch #6 — Angle of Actuator DE

Switch #7 — Dynamic Forces and Moments

Switch #8 — Translational Velocity and Acceleration
of Segment #1

Switch #9 — Segment #2: Deflection and Slope

Switch #10 — Static Forces and Moments written
to an output file

Switch #11 — Angle of Segment #1

Switch #12 — Angle of Wing Segment

Switch #13 — Inertial Forces and Torques

Switch #14 — Angular Velocity and Acceleration
of Segment #2

Switch #15 — Deflection of Pin #2

Switch #16 — All angles written to an output file

Switch #17 — Angle of Segment #2

Switch #18 — Angle of World View

Switch #19 — Force of Actuator AB

Switch #20 — Translational Velocity and Acceleration
of Segment #2

Switch #21 — Deflection of Pin #3

Switch #22 — Velocities and Accelerations of Segment #1

written to an output file

Switch #23 — Angle of Segment #3

Switch #24 — Differential change in the angles of

the actuators

Switch #25 — Force of Actuator DE

Switch #26 — Segment #1: Deflection and Slope

written to an output file

Switch #27 — Segment #2: Deflection and Slope

written to an output file

Switch #28 — Velocities and Accelerations of Segment #2

written to an output file

Switch #29 — Increment Level Number of Segment #1

Switch #30 — Decrement Level Number of Segment #1

Switch #31 — Increment Level Number of Segment #2

Switch #32 — Decrement Level Number of Segment #2

Numeric Pad #0 — Represents Pin #0

Numeric Pad #1 — Represents Pin #1

Numeric Pad #2 — Represents Pin #2

Numeric Pad #3 — Represents Pin #3

Numeric Pad #4 — Represents Pin #4, or Load

As an example of user interaction, say the user wanted to write the static forces and moments to the output file, then, he or she would select switch #10. Then, usr would write the reactions at Pins #0, 1, 2, 3, and the load to the output file. If the user chose to write the reactions to the screen, the user would then need specify which pin, # 0, 1, 2, 3, or 4 by pressing the 0, 1, 2, 3, or 4 of the numeric key pad on the keyboard. The selection of a pin or joint was required because of the limited space in the text window of the screen.

The static and dynamic analysis was completed for Segments #1 and #2 at each level. Switches #29 through #32 allowed the user to increment or decrement counters of the level numbers for Segments #1 and #2. Based on the limited space in the text window, if the user wanted to display information regarding a specific level, such as velocity, acceleration or deformation, the counter for that link would be referenced and then the information at the level number would be written to the screen.

If the user did not wish to continue with the model, by selecting the Q key, the model was terminated. Since the dynamic model does not allow the user to change the angles through the dial box, the user could continue the dynamic analysis by selecting the N key.

4.3 Three Dimensional Graphics

The routine to initialize the graphic capabilities was **inscrn** (intialize screen). As required by the IRIS, the first subroutine called was the IRIS subroutine GBEGIN. Next, the graphics and text windows were defined. The text window was placed on the bottom of the screen. The dimensions of the text window were 1024 pixels by 201 pixels, a height of twelve lines. The graphics window was defined to occupy the upper 1024 x 766 pixels of the workstation screen. The calls to define the screens were

```
CALL PREFPO(0,1023,201,767)
```

the graphics screen and

```
CALL TEXTPO(0,1023,0,200)
```

for the graphics window. Full color capabilities were utilizied on the graphics window. The colors for the text window were chosen to be a blue background with the text written in white.

```
CALL PAGECO(BLUE)
CALL TEXTCO(WHITE)
```

For asthetic reasons, the cursor symbol of the mouse, a red arrow that is usually placed on the screen, was filtered off by the command

```
CALL CURSOF.
```

This prevented the arrow from being drawn on the screen.

The final action of the routine inscrn was to scale the screen. The limits of the x, y, and z axes were selected to be

$$-3.5 \leq X \leq 3.5$$

$$-0.75 \leq Y \leq 5.0$$

$$-3.5 \leq Z \leq 3.5.$$

By calling the routine ORTHO, the screen was scaled

```
CALL ORTHO(-3.5,3.5,-0.75,5.0,-3.5,3.5).
```

This roughly represented the actual workspace in meters, defined by the working ARO elastic robot.

At this point, all graphics initializing tasks have been completed. Now, the coordinates of the vertices of each segment had be defined. The coordinates of the links could be defined based on the parameters specified by the user in the input parameter file, **robots.dat** and the shapes the links would follow. A reference group of coordinates based on the local coordinate system of each segment was determined for each link. This reference group of coordinates would then be deformed and transformed, that is, all analyses would be based upon the initial reference coordinates.

Three–dimensional arrays acted as the medium of storage for all link coordinates and two–dimensional arrays stored all the pin coordinates. The arrays are:

BS = BaSe

BSDT = BaSe Deformed and Transformed

PB = Pins on Base

PBDT = Pins on Base Deformed and Transformed

PW = Pins on Wing segment

PWD = Pins on Wing segment Deformed

PWDT = Pins on Wing segment Deformed and Transformed

P1 = Pins on Segment #1

P1D = Pins on Segment #1 Deformed

P1DT = Pins on Segment #1 Deformed and Transformed

P2 = Pins on Segment #2

P2D = Pins on Segment #2 Deformed

P2DT = Pins on Segment #2 Deformed and Transformed

S1 = Segment #1

S1D = Segment #1 Deformed

S1DT = Segment #1 Deformed and Transformed

S2 = Segment #2

S2D = Segment #2 Deformed

S2DT = Segment #2 Deformed and Transformed

S3 = Segment #3

S3D = Segment #3 Deformed

S3DT = Segment #3 Deformed and Transformed

WING = Wing segment

WINGD = Wing segment Deformed

WINGDT = Wing segment Deformed and Transformed

The routine which initialized and determined all initial coordinates was **defin2**, defin variables, part 2.

4.31 Define the Points of the Base. The local frame of reference for the base was located at the bottom of the base and at the center (see figure 4.5) for a graphical explanation of the variables of the base. The base was considered to be a circular cylindrical shape, but the IRIS did not allow the user to draw a circle in the x—z plane. To compensate, the base could be drawn in a polygonal cylindrical shape of n sides. By increasing the value of n, the cylinder would take on a more circular shape.

The base was drawn with eight sides, an octagonal cylinder. The required a total of sixteen vertices. The first eight vertices were located at the bottom of the cylinder and the second eight were at the top of the cylinder. The base was oriented in the x and z plane is such a manner that vertices 1, 3, 5, and 7 were located on either the x or z axis. Then, vertices 2, 4, 6, and 8 were located at 45° angles from the x— and z—axes, (see figure 4.6).

The variables COORD1 and COORD2 were used to determine the x and z cooridinates of the eight vertices.

$$\text{COORD1} = \text{base radius} = \text{BASEDI}/2 \tag{4.5}$$

$$\text{COORD2} = \text{COORD1*COS(45)} \tag{4.6}$$

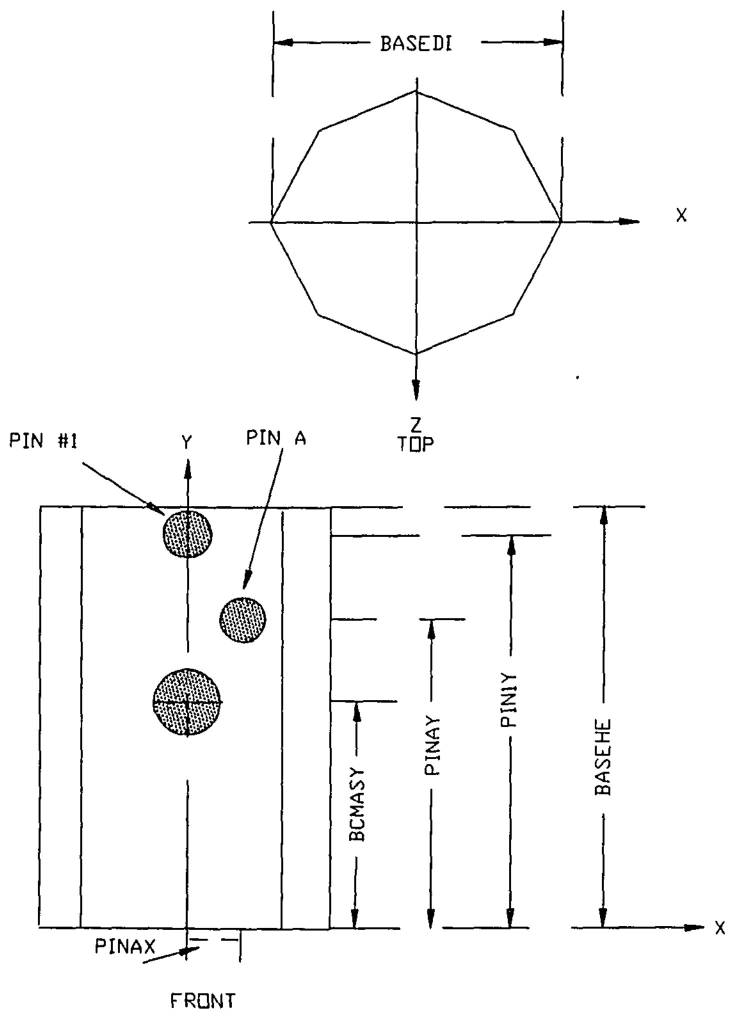


FIGURE 4.5

VARIABLES OF THE BASE

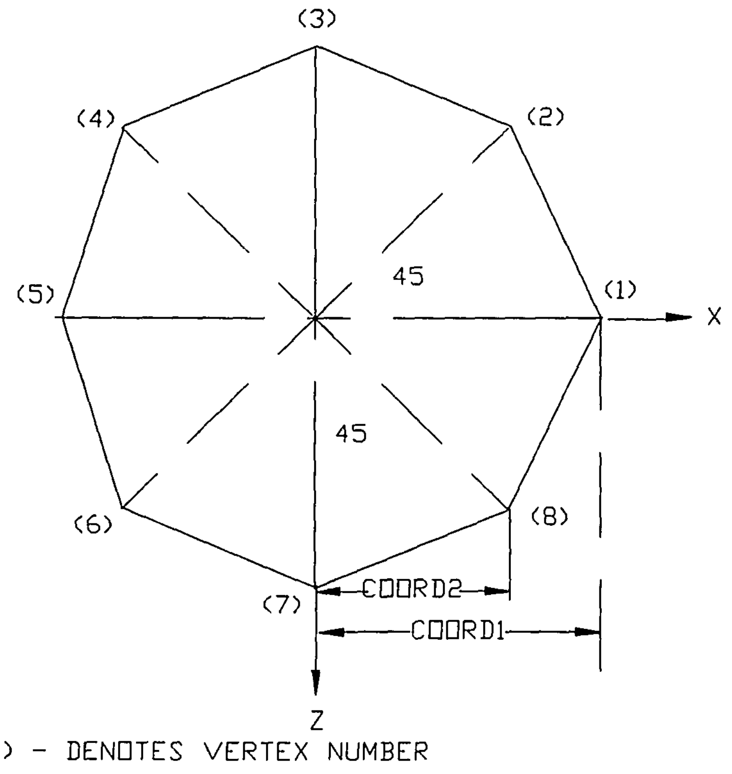


FIGURE 4.6

VERTICES OF THE BASE

From figure 4.6 it was easy to see that vertices 1, 3, 5, and 7 laid directly upon either the x– or z–axes. Therefore, the other coordinate, either the x or z coordinate, was equalled to zero. The y components of the coordinats were either 0.0 or BASEHE.

	X	Z
(1)	COORD1	0.0
(3)	0.0	–COORD1
(5)	–COORD1	0.0
(7)	COORD1	0.0.

The coordinates of vertices 2, 4, 6, and 8 were positive/negative combinations of COORD2:

	X	Z
(2)	COORD2	–COORD2
(4)	–COORD2	–COORD2
(6)	–COORD2	COORD2
(8)	COORD2	COORD2.

The bottom of the base rested at the origin, y = 0.0. The second eight vertices resided at y = BASEHE.

The coorindate information was stored in the array BS which was declared as

```
REAL BS(4,8,100)
```

The four rows represented the x–coordinate, y–cooridinate, z–Coordinate, and Scaling factor of the position vectors, respectively. Each of the eight columns represented a different position vector, that is, each column stored the cooridnate of a different vertex. The third dimension was used to allow each level of the base to be stored. Where Y = 0.0 was level #1 and where Y = BASEHE was level #2. If the base was not considered to be rigid, then the user would have been able to divide the base into subsections and therefore, there would be a variable number of levels. The value of BS(3,3,1) corresponded to the z–coordinate of the third vertex where the height y = 0.0, or level #1. The storing of the cooridnates of the vertices was completed through a Do–Loop. The Do–loop was executed twice. When I = 1, the bottom coordinates were stored and when I = 2, the top coordinates were stored.

```
DO 10 I = 1,2
```

Store the 'x' coordinates of each vertex in the base array.

```
BS(1,1,I)=COORD1
BS(1,5,I)=–COORD1
BS(1,2,I)=COORD2
BS(1,4,I)=–COORD2
BS(1,6,I)=–COORD2
BS(1,8,I)=COORD2
```

Store the 'z' coordinate of each vertex in the base array.

```
BS(3,3,I)=–COORD1
BS(3,7,I)=COORD1
BS(3,2,I)=–COORD2
BS(3,4,I)=–COORD2
BS(3,6,I)=COORD2
BS(3,8,I)=COORD2
```

Store the 'y' coordinate of each vertex in the base array. The 'y' coordinate is a function of the number of levels.

```
DO 11 II=1,8
BS(2,II,I)=(BASEHE)*(I–1)
```

The scaling factor was also initialized to equal one.

```
BS(4,II,I)=1.0
11        CONTINUE
10     CONTINUE
```

Define2 also placed the coorindates of the pions in their proper arrays. The array PB holds all the coordinates of pins placed on the Base. The array was a (4x4) allowing at this time a total of four points to be placed on the Base at this time. If additional pins were to be added, the column size of the array would just be increased. The four rows represented the three components, x, y, and z and the scaling factor of the position vector. The columns are assigned the pins in the following order:

Column #1 = Pin #1

Column #2 = Pin A

The coordinates for the pins were stored as follows:

```
PB(1,1)=0.0
PB(2,1)=PIN1Y
PB(3,1)=0.0
PB(4,1)=1.0
PB(1,2)=PINAX
PB(2,2)=PINAY
PB(4,2)=1.0
```

Pins were not always placed at the center of a link. On a circular link, there was a requirement of an adjustment for the z–coordinate of the pin. The z–coordinate of the pin was a function of the x–coordinate. Pin A was not placed at the center. Two functions were required, one for the domain $(0 \leq x \leq \text{COORD2})$ and one for the domain $(\text{COORD2} \leq x \leq \text{COORD1})$, See Figure 4.7. For $(0 \leq x \leq \text{COORD2})$ the slope was

$$\text{M1}=((\text{COORD2}-\text{COORD1})/\text{COORD2}). \tag{4.7}$$

For $(\text{COORD2} \leq x \leq \text{COORD1})$ the slope was

$$\text{M2}=(-\text{COORD2}/(\text{COORD1}-\text{COORD2})). \tag{4.8}$$

Based on the linear equation of a line,

$$z_1(x) = m_1 x + b_1 \tag{4.9}$$

and

$$z_2(x) = m_2 x + b_2. \tag{4.10}$$

To solve for b_1 and b_2 the boundary conditions were (0,COORD1) and (COORD1,0), respectively. Then,

$$b_1 = \mathrm{COORD1} \tag{4.11}$$

and

$$b_2 = -m_2{}^*\mathrm{COORD1}. \tag{4.12}$$

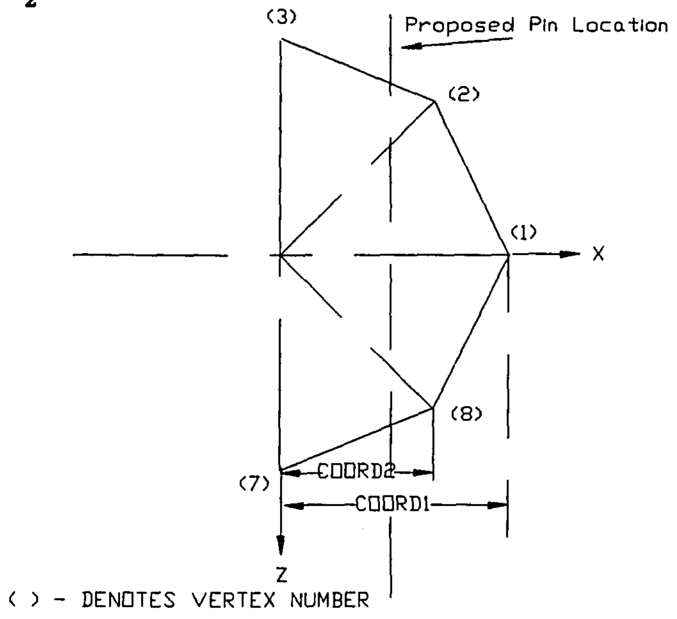


FIGURE 4.7

Z—COORDINATES OF THE PINS OF THE BASE

Therefore, the functions were

$$z_1(x) = m_1x + \mathrm{COORD1} \tag{4.13}$$

and

$$z_2(x) = m_2(x - \mathrm{COORD1}). \tag{4.14}$$

The FORTRAN code was

```
        IF(PB(1,2).GE.0.0.AND.PB(1,2).LE.COORD2) THEN
                PB(3,2)=M1*PB(1,2)+COORD1
        ELSE
                PB(3,2)=M2*(PB(1,2)–COORD1)
        ENDIF.
```

<u>4.32 Define the Points of Segment #1.</u> The next section to be defined was Segment #1 and the pins associated with Segment #1. Everything was described with respect to Segment #1's local coordinate frame (see figure 4.8). The local coordinate frame was described through the variables S1PIX and S1P1Y. The coordinate frame was located where Pin #1 was on Segment #1. The pin coordinates were stored in the array P1 where

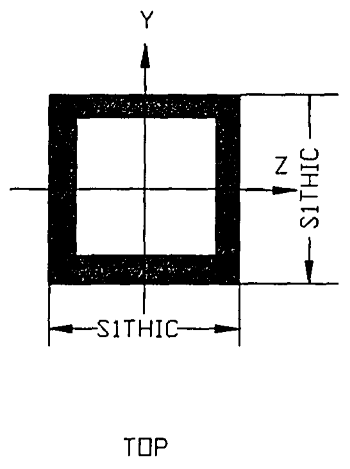
Column #1 = Pin #2

Column #2 = Pin B

Column #3 = Pin D.

All measurements on Segment #1 had to be with respect to the local frame of reference. The measurement of PIN2X was measured from the left most point of the segment. This was not a correct coordinate if the reference point was the location of Pin #1. Therefore, to compensate for the location of Pin #1, and the local reference frame, the cooridnates were defined as follows:

```
        P1(1,1)=PIN2X–S1P1X
        P1(2,1)=PIN2Y–S1P1Y
        P1(3,1)=0.0
        P1(4,1)=1.0
        P1D(4,1)=1.0
        P1DT(4,1)=1.0
        P1(1,2)=PINBX–S1P1X
```



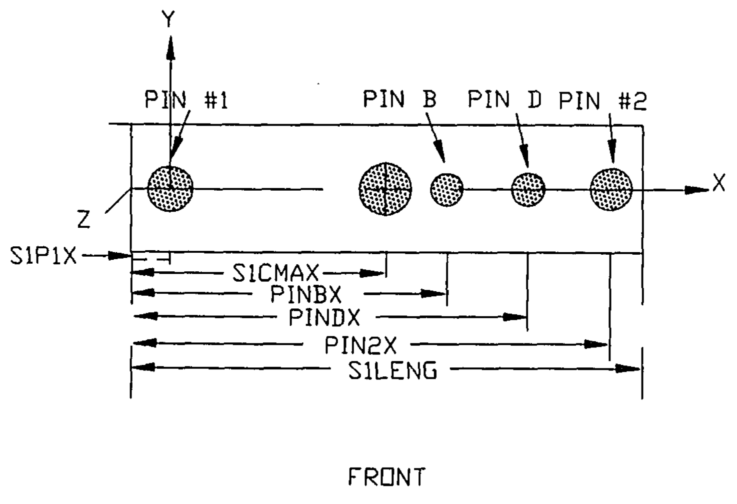


FIGURE 4.8

VARIABLES OF SEGMENT #1

P1(2,2)=PINBY–S1P1Y

P1(3,2)=S1THIC/2

P1(4,2)=1.0

P1D(4,2)=1.0

P1DT(4,2)=1.0

P1(1,3)=PINDX–S1P1X

P1(2,3)=PINDY–S1P1Y

P1(3,3)=S1THIC/2

P1(4,3)=1.0

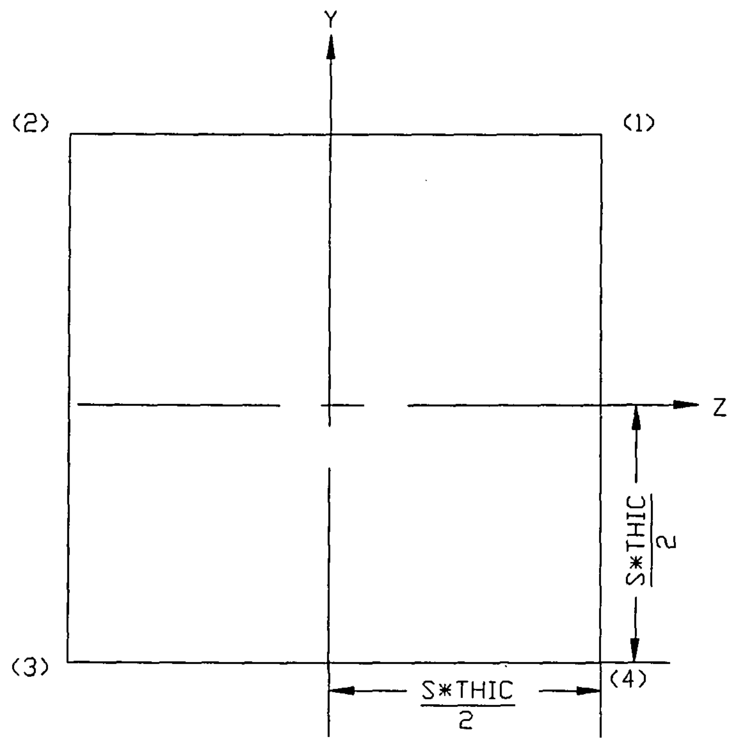
P1D(4,2)=1.0

P1DT(4,2)=1.0

Next, the cooridnate of the vertices were determined. The x–axis was defined as running the length of the segment along the centerline. This segments was considered to be a square hollow cylinder. The y– and z–axes bisected the width and height respectively (see figure 4.9). The number of total vertices was a function of the number of subsections the user had defined, S1DIVN. The user could subdivide segments #1 and #2 into a maximum of 99 subsections. This allowed the deflection analysis to be completed at each subsection line or level. Each subsection could be though of as a rectangular parallelepiped (see figure 3.14). The vertices to draw the first subsection were the four vertices of level #1 and the four vertices of level #2. Similarly, the vertices requried to describe the second rectangle subsection were the vertices of level #2 and the vertices at level #3. So, if the user defined S1DIVN number of subsections, there was S1DIVN+1 or

S1J=S1DIVN+1

number of levels and a total of (S1DIVN+1)*4 number of vertices to describe the segment (see figure 4.10). The width and height were constant throughout the length of the beam. The



( ) - DENOTES VERTEX NUMBER

FIGURE 4.9

VERTICES OF SEGMENTS #1, #2, AND #3

length in the x–direction though, was a function of the level number. By using a Do–loop, the ccordinates of each level were determined. The loop was executed as many times as the number of levels therefore, each interation addressed a specific level. The length of the given level was a function of the iteration number. The scaling factor was also defined to be equal to one.

```
            DO 100 I=1,S1J
```

Determine the x–coordinate and the scaling factor.

```
                DO 101 J=1,4
                 S1(1,J,I)=(S1LENG/S1DIVN)*(I–1)–S1P1X
                 S1(4,J,I)=1.0
                 S1D(4,J,I)=1.0
                 S1DT(4,J,I)=1.0
101         CONTINUE
```

Determine the y–coordinates.

```
                S1(2,1,I)=S1THIC/2
                S1(2,2,I)=S1THIC/2
                S1(2,3,I)=–S1THIC/2
                S1(2,4,I)=–S1THIC/2
```

Determine the z — coordinates

```
                S1(3,1,I)=S1THIC/2
                S1(3,2,I)=–S1THIC/2
                S1(3,3,I)=–S1THIC/2
                S1(3,4,I)=S1THIC/2
100   CONTINUE
```

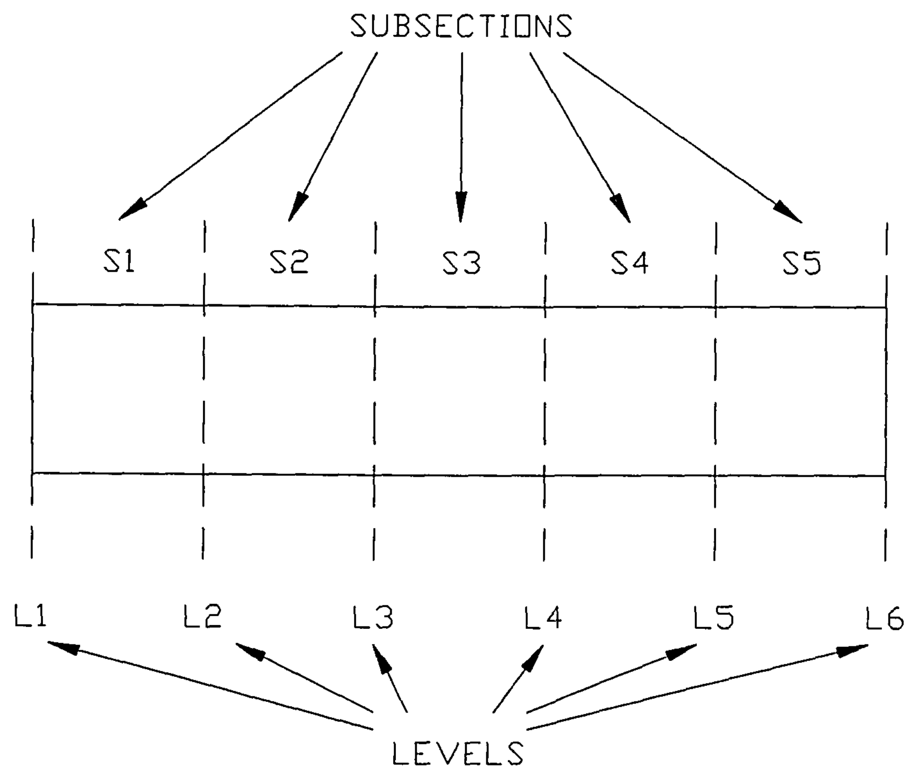


FIGURE 4.10

SUBSECTIONS AND LEVELS OF A SEGMENT

4.33 Define the Points of Segment #2. Segment #2 followed the same general pattern as #1. The Wing and beam segments of Segment #2 (see figure 4.11), were treated sperately. The beam segment was also divided into subsections as defined by the user. The local frame of reference was orented in the same manner (see figure 4.12). Only one pin resided on Segment #2. The coordinates were stored in the array P2, Column #1.

```
          P2(1,1)=PIN3X–S2P2X
          P2(2,1)=PIN3Y–S2P2Y
          P2(3,1)=0.0
          P2(4,1)=1.0
          P2D(4,1)=1.0
          P2DT(4,1)=1.0
```

The total number of subsections was stored in S2DIVN and the total number of levels was then

```
          S2J=S2DIVN+1.
```

Next, the coordinates were defined.

```
          DO 200 I=1,S2J
```

The x–direction coordinates and scaling factors were determined.

```
                DO 201 J=1,4
                 S2(1,J,I)=(S2LENG/S2DIVN)*(I–1)–S2P2X
                 S2(4,J,I)=1.0
                 S2D(4,J,I)=1.0
                 S2DT(4,J,I)=1.0
201       CONTINUE
```

The y — coordinates were determined.

```
                S2(2,1,I)=S2THIC/2
                S2(2,2,I)=S2THIC/2
                S2(2,3,I)=–S2THIC/2
```

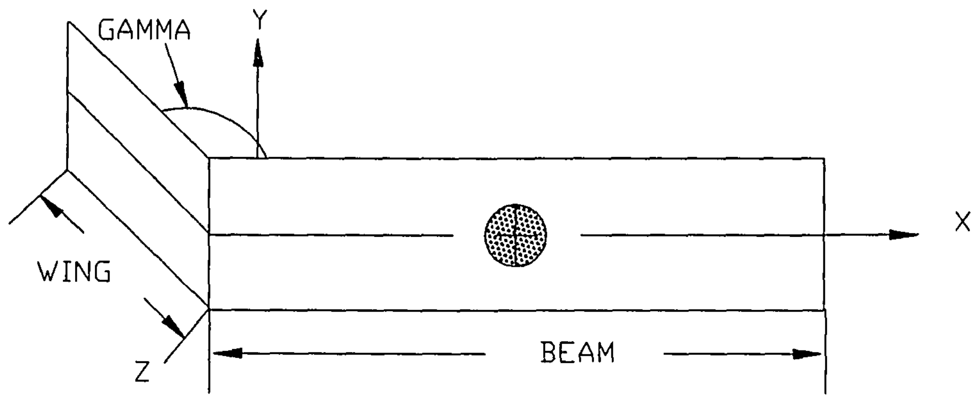
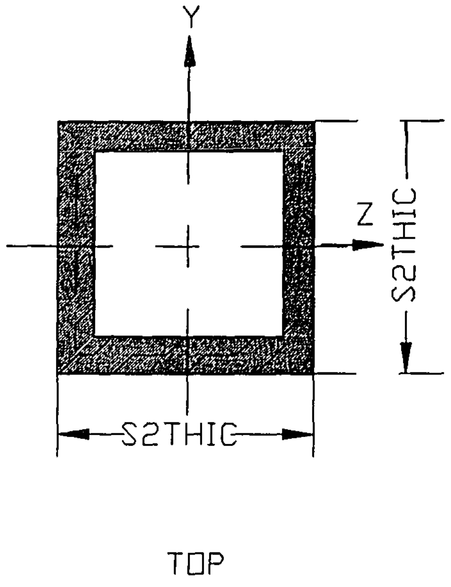


FIGURE 4.11

WING AND BEAM COMPONENTS OF SEGMENT #2



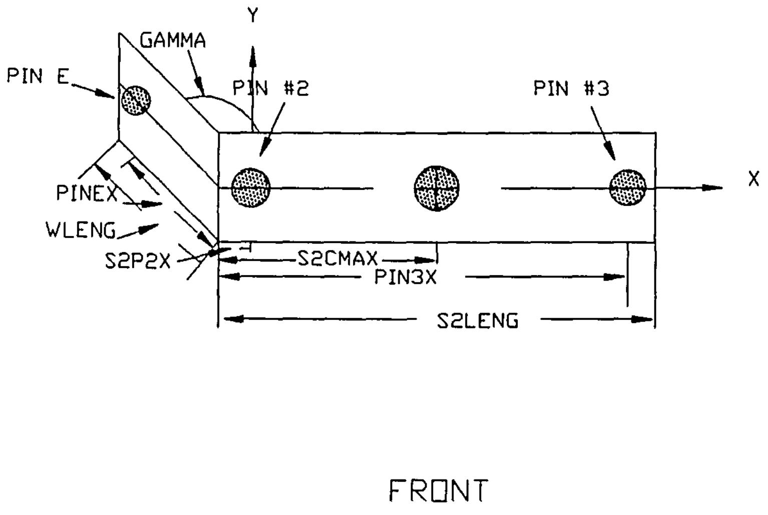


FIGURE 4.12

VARIABLES OF SEGMENT #2

```
          S2(2,4,I)=-S2THIC/2
          S2(3,4,I)=S2THIC/2
200   CONTINUE
```

4.34 Define the Points of the Wing Segment and Segment #3. The Wing segment and Segment #3 were based on the same shape as Segments #1 and #2, but, they were considered to be rigid and therefore were neither subdivided nor analyized. Since neither of these segments could be divided, each had a total number of levels equal to two.

The Wing Segment was the same thickness as Segment #2. The Pin E which connected the top of actuator DE to the Wing resided on the Wing (see figure 4.12). The coordinates for this pin were stored in the array PW in column #1.

```
      PW(1,1)=PINEX
      PW(2,1)=PINEY
      PW(3,1)=0.0
      PW(4,1)=1.0
      PWD(4,1)=1.0
      PWDT(4,1)=1.0
```

The dimensions of the vertices of the Wing Segment were defined as follows:

```
      DO 290 I=1,2
```

The y coordinates were

```
          WING(2,1,I)=S2THIC/2
          WING(2,2,I)=S2THIC/2
          WING(2,3,I)=-S2THIC/2
          WING(2,4,I)=-S2THIC/2
```

The z coordinates were

```
          WING(3,1,I)=S2THIC/2
          WING(3,2,I)=-S2THIC/2
          WING(3,3,I)=-S2THIC/2
          WING(3,4,I)=S2THIC/2
```

The x coordinates and scaling factors were

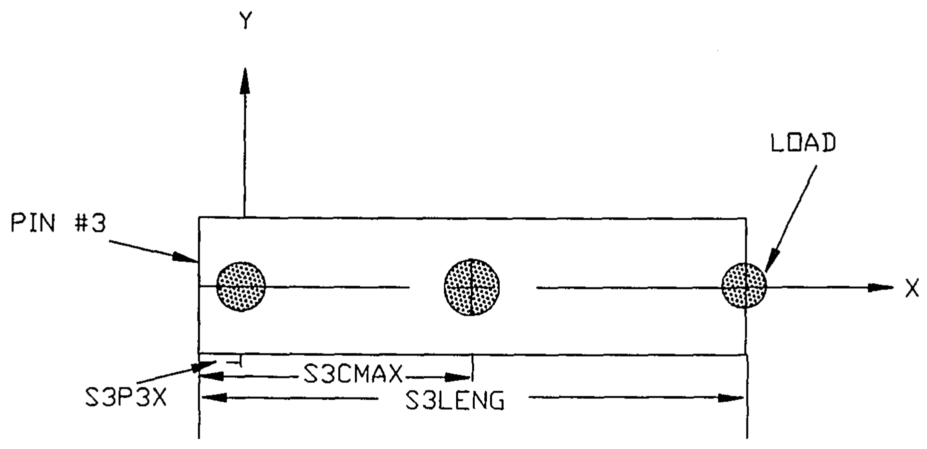
```
          DO 291 J=1,4
           WING(1,J,I)=(I-1)*WLENG
           WING(4,J,I)=1.0
           WINGD(4,J,I)=1.0
           WINGDT(4,J,I)=1.0
291        CONTINUE
290   CONTINUE
```

Segment #3 did not contain any actuator pins. The only pin lcoation which resided on Segment #3 was the location of Pin #3 which connected the end–effector to Segment #2. This location was described for Segment #3 through the variables S3P3X and S3P3Y (see figure 4.13). The only coordinates required then for the segment were the eight vertex coordinates.

```
        DO 300 I=1,2
```

The x–direction coordinates and scaling factor were

```
          DO 301 J=1,4
           S3(1,J,I)=S3LENG*(I-1)-S3P3X
           S3(4,J,I)=1.0
           S3D(4,J,I)=1.0
           S3DT(4,J,I)=1.0
301        CONTINUE
```



FRONT

FIGURE 4.13

VARIABLES OF SEGMENT #3

The y — coordinates were

```
S3(2,1,I)=S3THIC/2
S3(2,2,I)=S3THIC/2
S3(2,3,I)=-S3THIC/2
S3(2,4,I)=-S3THIC/2
```

The z — direction coordinates were

```
S3(3,1,I)=S3THIC/2
S3(3,2,I)=-S3THIC/2
S3(3,3,I)=-S3THIC/2
S3(3,4,I)=S3THIC/2
```

```
300 CONTINUE
```

4.35 Define the Length and Angles of the Hydraulic Actuators. The angles and lengths of actuators AB and DE had to be initially determined. Both were functions of the pin locations and ϕ_1, ϕ_2, ϕ_3, and ϕ_6. First, the length of actuator AB and ϕ_4 were found (see figure 3.22).

```
BETA=1.5706+ANG1CNT
H1A1=PIN1Y-PINAY
H1B1=PINBX-S1P1X
H1D1=H1A1+H1B1+SIN(ANG1CNT)
H1E1=H1B1*COS(ANG1CNT)
H1LENG=SQRT(H1D1**2+H1E1**2)
ANG4CNT=ASIN(H1D1/H1LENG)
```

The length of actuator DE and the magnitude of ϕ_5 were determined (see figure 3.23).

```
KAPPA=1.5706-ANG1CNT
ALPHA=6.2831-ANG6CNT-KAPPA
H2A2=PINEX
H2B2=PIN2X-PINDX
```

```
H2C22=H2A2**2+H2B2**2+2*H2A2*H2B2*COS(KAPPA)

H2LENG=SQRT(H2C22)

SIGMA=ACOS((H2A2**2–H2B2**2–H2C22)/(–2*H2B2*H2LENG))

ANG5CNT=SIGMA+ANG1CNT
```

4.36 Transform the Coordinates. After the links had all been defined, the deflection analysis was initiated. When the magnitude of deflection was calculated, the reference coordinates fo the segments were adjust. The sum was the deformed coordinates of the segments.

$$\begin{bmatrix}\text{Deformed}\\ \text{Coordinates}\end{bmatrix} = \begin{bmatrix}\text{Reference}\\ \text{Coordinates}\end{bmatrix} + \begin{bmatrix}\text{Deformed}\\ \text{Magnitudes}\end{bmatrix}.$$

The deformed coordinates were stored in the arays S1D, S2D, WINGD, and S3D. By storing the deformed coordinates in the above mentioned arrays, the original coordinates were preserved. This was required because a point of reference was required.

After the deflection analysis was completed, the coordinates were transformed. The input to the transformation routine, **TFORM** was the deformed cooridnates. The output was the coordinates which were deformed and transformed. These coordinates were stored in the arrays BSDT, S1DT, S2DT, WINGDT, and S3DT. Now, the links were ready to be drawn on the screen.

4.37 Create the Graphical Output. The routine **DRROB**, draw robot, controlled all the drawing routines. The first routine called was **RAXIS**, reference axis. This routine was the same for both models. It first cleared the screen using the color black by calling the IRIS command

```
CALL COLOR(BLACK)

CALL CLEAR.
```

Next, using the IRIS commands MOVE and DRAW, a three–dimensional reference axis system was drawn. The color white was used to draw the axis system.

```
CALL COLOR(WHITE)
```

The axis was drawn at the three–dimensional coordinates (2.0, 2.0, 2.0). Each axis was 0.2 in length magnitude.

```
CALL MOVE(2.0, 0.0, 2.0)
CALL DRAW(2.2, 0.0, 2.0)
CALL MOVE(2.0, 0.0, 2.0)
CALL DRAW(2.0, 0.2, 2.0)
CALL MOVE(2.0, 0.0, 2.0)
CALL DRAW(2.0, 0.0, 2.2)
```

Each axis was labeled with the appropriate letter, X, Y, or Z.

```
CALL CMOV(2.2, 0.0, 2.0)
CALL CHARST('X', 1)
CALL CMOV(2.0, 0.2, 2.0)
CALL CHARST('Y', 1)
CALL CMOV(2.0, 0.0, 2.2)
CALL CHARST('Z', 1)
```

The drawing of each link was a three part process. First a white outline of where the undeformed segment would lay was drawn. Next, the deformed segment was drawn in two parts. First, a solid filled deformed object was drawn. Then, to differentiate between the sides, a wireframe was overlayed on the deformed solid object. The links were drawn in the following order:

Base

Segment #1

Segment #2

Wing Segment

Segment #3

Pins

Actuators.

4.38 Draw the Base. Graphical The base was drawn as an octagonal cylinder. The solid shape was drawn by the routine DRBS, draw base, solid. Eight rectangular panels, the sides, were drawn. the, the bottom and top octagon shaped panels were drawn. The IRIS subroutine POLF, polygon filled, was used. The routine required the number of vertices, n, defining the polygon, and the varible name of a (3,n) array storing the coorindates of the vertices. To draw the base, two arrays were required:

```
REAL POLYOC(3,8), POLYSQ(3,4).
```

The first array, POLYOC (polygon — octagon) was used to draw the top and bottom. The second array, POLYSQ (polygon — square), was used to draw the rectangular side panels. The solid base was drawn in the color magental.

```
CALL COLOR(MAGENTA)
```

Do—Loops 12 and 13 drew the side panels 1 through 7. A panel was drawn by referencing the same vertices of the bottom level as in the top. For example, side panel #1 consisted of vertices #1 and #2 of levels #1 and #2. Side panel #2 consisted of vertices #2 and #3 of levels #1 and #2.

Panel	Vertices Required
1	1 & 2
2	2 & 3
3	3 & 4
4	4 & 5
5	5 & 6
6	6 & 7
7	7 & 8
8	8 & 1

Do—loop 13 stored the coordinate information of the four vertices by incrementing the row index.

```
              DO 12 J = 1,7
                        DO 13 K = 1,3
                         POLYSQ(K,1) = BSDT(K,J,1)
                         POLYSQ(K,2) = BSDT(K,J,2)
                         POLYSQ(K,3) = BSDT(K,(J+1),2)
                         POLYSQ(K,4) = BSDT(K,(J+1),1)
13                      CONTINUE
```

After all the coordinates have been stored in POLYSQ, the solid polygon was drawn.

```
                        CALL POLF(4,POLYSQ)
```

Do—loop 12 incremented the vertices so that panels one through seven could be drawn.

```
12            CONTINUE
```

Next, the last panel, 8, was drawn

```
              DO 14 K=1,3
                        POLYSQ(K,1) = BSDT(K,8,1)
                        POLYSQ(K,2) = BSDT(K,8,2)
                        POLYSQ(K,3) = BSDT(K,1,2)
                        POLYSQ(K,4) = BSDT(K,1,1)
14            CONTINUE
              CALL POLF(4,POLYSQ)
```

Last, the top and bottom panels were drawn. Do—loop 16 stored the coordinates of the vertices. Then, the shape was drawn. Do—loop 15 controlled which level was drawn. Since only the top and bottom were required to be drawn, the loop was only executed twice. When I = 1 the bottom was drawn and when I = 2 the to was drawn.

```
              DO 15 I=1,2
                        DO 16 J=1,3
                         POLYOC(J,1) = BSDT(J,1,I)
                         POLYOC(J,2) = BSDT(J,2,I)
```

```
                    POLYOC(J,3) = BSDT(J,3,I)
                    POLYOC(J,4) = BSDT(J,4,I)
                    POLYOC(J,5) = BSDT(J,5,I)
                    POLYOC(J,6) = BSDT(J,6,I)
                    POLYOC(J,7) = BSDT(J,7,I)
                    POLYOC(J,8) = BSDT(J,8,I)
16                 CONTINUE
                   CALL POLF(8,POLYOC)
15         CONTINUE
```

The routine **DRBW** (draw base — wire), overlayed a wired frame on the solid object. The color had to be a shade of magenta so that it was visible to the user. The IRIS command MAPCOL would allow the user to create a shade by specifying the intensity of the red, green, and blue guns.

```
           CALL MAPCOL(157, 157, 0, 157)
           IBWIRE=157
           CALL COLOR(IBWIRE)
```

First, the bottom and top octagons were outlined. To outline the shapes lines were drawn to connect the vertices of each level in numerical order. Do—loop 11 incremented the level number while Do—loop 12 incremented the vertex number.

```
           DO 11 J=1,2
                   CALL MOVE(BSDT(1,1,J), BSDT(2,1,J), BSDT(3,1,J))
                   DO 12 I=2,8
                    CALL DRAW(BSDT(1,I,J), BSDT(2,I,J), BSDT(3,I,J))
12                 CONTINUE
```

The conection from vertex (8) to vertex (1) was completed.

```
                   CALL DRAW(BSDT(1,1,J), BSDT(2,1,J), BSDT(3,1,J))
11         CONTINUE
```

To draw the wireframe, lines were drawn from the bottom to the top. Vertex #1 of the bottom was connected to vertex #1 of the top. Do—loop 13 incremented the vertex number.

```
            DO 13 I=1,8
                  CALL MOVE(BSDT(1,I,1), BSDT(2,I,1), BSDT(3,I,1))
                  CALL DRAW(BSDT(1,I,2), BSDT(2,I,2), BSDT(3,I,2))
13          CONTINUE
```

4.39 Draw Segment #1. Next, Segment #1 was drawn. To draw the solid shape, only one array was required.

```
            REAL POLYSQ(3,4).
```

The color light blue, cyan, was used to draw the segment.

```
            CALL COLOR(CYAN)
```

A similar fomrat was used to draw the side panels as in **DRBS**. Because SEgment #1 could be subsectioned, each subsection was completely drawn and then the level number was incremented and the next subsection was drawn. Do—loop 110 incremented the level number from one to the next to last level. This was required becasue to draw the last subsection, the do—loop would have referenced a level number which was out of range. Do—loop 111 incremented the vertices and Do—loop 112 incremented the rows.

```
            DO 110 I=1,S1DIVN
                  DO 111 J = 1,3
                   DO 112 K = 1,3
                   POLYSQ(K,1) = S1DT(K,J,I)
                   POLYSQ(K,2) = S1DT(K,J,(I+1))
                   POLYSQ(K,3) = S1DT(K,(J+1),(I+1))
                   POLYSQ(K,4) = S1DT(K,(J+1),I)
112         CONTINUE
```

Before 111 could be incremented, the polygon was drawn.

```
                CALL POLF(4,POLYSQ)
111         CONTINUE
```

Before the levels can be could be incremented, the fourth panel had to be drawn. Do loop 113 transfered the coordinates to the polygon array so that the fourth side of every subsection could be drawn.

```
               DO 113 K=1,3
                POLYSQ(K,1) = S1DT(K,4,I)
                POLYSQ(K,2) = S1DT(K,4,(I+1))
                POLYSQ(K,3) = S1DT(K,1,(I+1))
                POLYSQ(K,4) = S1DT(K,1,I)
113         CONTINUE
               CALL POLF(4,POLYSQ)
110   CONTINUE
```

After all the subsections had been drawn, the left and right ends had to be drawn. This only required the four vertices of level #1 and be connected to each other and the four vertices of the last level to be connected to each other. the value of the last level is stored in S1J.

```
            S1J=S1DIVN+1
```

Do—loop 114 was only executed twice. It was executed once when I = 1, for the first level or left end. Then, it was incremented by a step of S1DIVN to I = S1J, the last level number where the loop was executed a second time. Do—loop 115 was used to slimline the passing of coordinate information into the polygon array.

```
            DO 114 I=1,S1J,S1DIVN
               DO 115 J=1,3
                POLYSQ(J,1) = S1DT(J,1,I)
                POLYSQ(J,2) = S1DT(J,2,I)
                POLYSQ(J,3) = S1DT(J,3,I)
```

```
                    POLYSQ(J,4) = S1DT(J,4,I)
115         CONTINUE
                    CALL POLF(4,POLYSQ)
114   CONTINUE
```

As with the Base, a wireframe was overlayed on the solid segment shape. The routine **DRS1W** (draw Segment #1 — wire), completed the wire frame of Segment #1. A color that was a slightly different shade of cyan was used.

```
            CALL MAPCOL(150, 0, 150, 150)
            I1WIRE=150
            CALL COLOR(I1WIRE)
```

The wireframe was constructed by connecting the same vertex of each level and also connecting the four vertices of the left end together and then the four vertices of the right end. Do—loops 101 and 102 created the end pieces. The only two levels required for reference for these items were the first and last. Do—loop 101 was executed only twice as with Do—loop 114 fo the solid drawing routine. Do—loop 102 incremented the vertex number so that a line was drawn from (1) to (2) to (3) to (4).

```
            S1J=S1DIVN+1
            DO 101 J=1,S1J,S1DIVN
                    CALL MOVE(S1DT(1,1,J), S1DT(2,1,J), S1DT(3,1,J))
                    DO 102 I=2,4
                     CALL DRAW(S1DT(1,I,J), S1DT(2,I,J), S1DT(3,I,J))
102         CONTINUE
```

The final draw statementthen connected (4) to (1).

```
                    CALL DRAW(S1DT(1,1,J), S1DT(2,1,J), S1DT(3,1,J))
101   CONTINUE
```

Do—loops 103 and 104 drew the lines connecting the same vertex of each level. Do—loop 103 incremented the vertex while Do—loop 104 incremented the level.

```
          DO 103 I=1,4
                    CALL MOVE(S1DT(1,I,1), S1DT(2,I,1), S1DT(3,I,1))
                    DO 104 J=2,S1J
                     CALL DRAW(S1DT(1,I,J), S1DT(2,I,J), S1DT(3,I,J))
104       CONTINUE
103   CONTINUE
```

4.310 Draw Segment #2. Segment #2 was next to be drawn. The same algorithm can be used to draw both the solid and wire forms of Sgment #2 as used for Segment #1. The only difference between the routines is that for Segment #2 the array S2DT and the variables S2DIVN was referenced instead of S1DT and S1DIVN.

To draw the solid form, the routine called was **DRS2S** (draw Segment #2 – solid). Segment #2 was drawn in a solid form using the color yellow.

```
          CALL COLOR(YELLOW)
```

Draw the subsection side panels.

```
          DO 210 I=1,S2DIVN
                   DO 211 J = 1,3
                    DO 212 K = 1,3
                    POLYSQ(K,1) = S2DT(K,J,I)
                    POLYSQ(K,2) = S2DT(K,J,(I+1))
                    POLYSQ(K,3) = S2DT(K,(J+1),(I+1))
                    POLYSQ(K,4) = S2DT(K,(J+1),I)
212        CONTINUE
                    CALL POLF(4,POLYSQ)
211       CONTINUE
```

Now the fourth side panel of the subsection was drawn.

```
                   DO 213 K=1,3
                    POLYSQ(K,1) = S2DT(K,4,I)
                    POLYSQ(K,2) = S2DT(K,4,(I+1))
                    POLYSQ(K,3) = S2DT(K,1,(I+1))
                    POLYSQ(K,4) = S2DT(K,1,I)
213           CONTINUE
                   CALL POLF(4,POLYSQ)
210     CONTINUE
```

Draw the left and right ends.

```
              S2J=S2DIVN+1
              DO 214 I=1,S2J,S2DIVN
                   DO 215 J=1,3
                    POLYSQ(J,1) = S2DT(J,1,I)
                    POLYSQ(J,2) = S2DT(J,2,I)
                    POLYSQ(J,3) = S2DT(J,3,I)
                    POLYSQ(J,4) = S2DT(J,4,I)
215           CONTINUE
                   CALL POLF(4,POLYSQ)
214     CONTINUE
```

The wireframe was drawn by the routine **DRS2W** (draw SEgment #2 — wireframe). The color used was a slightly different hue of yellow.

```
              CALL MAPCOL(175, 175, 175, 0)
              I2WIRE=175
              CALL COLOR(I2WIRE)
```

Again, this algorithm was exactly the same as that found in **DRS1W** except for the variables referenced. First, the left and right ends were outlined.

```
              S2J=S2DIVN+1
              DO 201 J=1,S2J,S2DIVN
                   CALL MOVE(S2DT(1,1,J), S2DT(2,1,J), S2DT(3,1,J))
```

```
                    DO 202 I=2,4
                     CALL DRAW(S2DT(1,I,J), S2DT(2,I,J), S2DT(3,I,J))
202             CONTINUE
                    CALL DRAW(S2DT(1,1,J), S2DT(2,1,J), S2DT(3,1,J))
201     CONTINUE
```

Next the side panels were outlined.

```
                DO 203 I=1,4
                    CALL MOVE(S2DT(1,I,1), S2DT(2,I,1), S2DT(3,I,1))
                    DO 204 J=2,S2J
                     CALL DRAW(S2DT(1,I,J), S2DT(2,I,J), S2DT(3,I,J))
204             CONTINUE
203     CONTINUE
```

4.311 Draw the Wing Segment. Next, the Wing Segment was drawn. The routine **DRWS**, draw Wing Segment — Solid, was called for this action. The routine referenced the color yellow so that the Wing Segment matched Segment #2.

```
                CALL COLOR(YELLOW)
```

This routine differed from DRS1S and DRS2S because the Wing was considered to be rigid and could not be subdivided. Therefore, there were only four side panels to be drawn instead of 4xn(number of divisions). Do–loops 410 and 412 drew the sides one through three.

```
                DO 410 J = 1,3
                    DO 412 K = 1,3
                     POLYSQ(K,1) = WINGDT(K,J,1)
                     POLYSQ(K,2) = WINGDT(K,J,2)
                     POLYSQ(K,3) = WINGDT(K,(J+1),2)
                     POLYSQ(K,4) = WINGDT(K,(J+1),1)
412             CONTINUE
```

```
                    CALL POLF(4,POLYSQ)
410   CONTINUE
```

Now, side panel #4 was drawn by connecting the vertices (1) and (4) of the two levels in Do—loop 413.

```
          DO 413 K=1,3
                    POLYSQ(K,1) = WINGDT(K,4,1)
                    POLYSQ(K,2) = WINGDT(K,4,2)
                    POLYSQ(K,3) = WINGDT(K,1,2)
                    POLYSQ(K,4) = WINGDT(K,1,1)
413   CONTINUE
```

Do—loops 414 and 415 drew the left and right sides.

```
          CALL POLF(4,POLYSQ)
          DO 414 I=1,2
                    DO 415 J=1,3
                     POLYSQ(J,1) = WINGDT(J,1,I)
                     POLYSQ(J,2) = WINGDT(J,2,I)
                     POLYSQ(J,3) = WINGDT(J,3,I)
                     POLYSQ(J,4) = WINGDT(J,4,I)
415       CONTINUE
                    CALL POLF(4,POLYSQ)
414   CONTINUE
```

The routine **DRWW** (draw wing — wireframe), was called next to overlay the wireframe of the Wing Segment. The color used was the same as that used to draw the wireframe of Segment #2.

```
          CALL MAPCOL(175, 175, 175, 0)
          IWWIRE=175
          CALL COLOR(IWWIRE)
```

The side panels were drawn first through Do—loops 401 and 402.

```
          DO 401 J=1,2
            CALL MOVE(WINGDT(1,1,J), WINGDT(2,1,J), WINGDT(3,1,J))
                DO 402 I=2,4
            CALL DRAW(WINGDT(1,I,J), WINGDT(2,I,J), WINGDT(3,I,J))
402       CONTINUE
            CALL DRAW(WINGDT(1,1,J), WINGDT(2,1,J), WINGDT(3,1,J))
401  CONTINUE
```

Finally, the left and right panels were outlined through the Do—loop 403.

```
          DO 403 I=1,4
            CALL MOVE(WINGDT(1,I,1), WINGDT(2,I,1), WINGDT(3,I,1))
            CALL DRAW(WINGDT(1,I,2), WINGDT(2,I,2), WINGDT(3,I,2))
403  CONTINUE
```

4.312 Draw Segment #3. The final segment to be drawn was Segment #3. The routines DRS3S and DRS3W were the same as those for the wing except the variables wee changed respectively. Segment #3 was drawn in the solid form using the color green.

```
          CALL COLOR(GREEN)
```

First, the side panels were drawn.

```
          DO 310 J = 1,3
                DO 312 K = 1,3
                 POLYSQ(K,1) = S3DT(K,J,1)
                 POLYSQ(K,2) = S3DT(K,J,2)
                 POLYSQ(K,3) = S3DT(K,(J+1),2)
                 POLYSQ(K,4) = S3DT(K,(J+1),1)
312       CONTINUE
                CALL POLF(4,POLYSQ)
310  CONTINUE
```

Next, the fourth side panel and the left and right panels were drawn.

```
                DO 313 K=1,3
                        POLYSQ(K,1) = S3DT(K,4,1)
                        POLYSQ(K,2) = S3DT(K,4,2)
                        POLYSQ(K,3) = S3DT(K,1,2)
                        POLYSQ(K,4) = S3DT(K,1,1)
313     CONTINUE
```

Do—loops 314 and 315 drew the left and right sides.

```
                CALL POLF(4,POLYSQ)
                DO 314 I=1,2
                        DO 315 J=1,3
                         POLYSQ(J,1) = S3DT(J,1,I)
                         POLYSQ(J,2) = S3DT(J,2,I)
                         POLYSQ(J,3) = S3DT(J,3,I)
                         POLYSQ(J,4) = S3DT(J,4,I)
315             CONTINUE
                        CALL POLF(4,POLYSQ)
314     CONTINUE
```

A color shade of green was used to draw the wireframe in the routine DRS3W, draw Segment #3 — wireframe.

```
                CALL MAPCOL(107, 0, 107, 0)
                I3WIRE=107
                CALL COLOR(I3WIRE)
```

First, the left and right panels were outlined.

```
                DO 301 J=1,2
                  CALL MOVE(S3DT(1,1,J), S3DT(2,1,J), S3DT(3,1,J))
                  DO 302 I=2,4
                    CALL DRAW(S3DT(1,I,J), S3DT(2,I,J), S3DT(3,I,J))
302               CONTINUE
                    CALL DRAW(S3DT(1,1,J), DT(2,1,J), S3DT(3,1,J))
301   CONTINUE
```

Finally, the side panels were outlined.

```
                  DO 303 I=1,4
                    CALL MOVE(S3DT(1,I,1), S3DT(2,I,1), S3DT(3,I,1))
                    CALL DRAW(S3DT(1,I,2), S3DT(2,I,2), S3DT(3,I,2))
303   CONTINUE
```

<u>4.313 Draw the Pins and Hydraulic Actuators.</u> After all the segments have been drawn, the pis and actuators were drawn. The pins were drawn through the routine **DRPIN** (draw pins). All the pins were drawn in red.

```
                  CALL COLOR(RED)
```

Circles were used to denote a pin. The IRIS circle command does not reference a third dimension, z. Therefore, the pins were drawn in the x—y plane.

First, PINS #1, #2, and #3 were drawn.

```
                  CALL CIRCF(PBDT(1,1), PBDT(2,1), 0.01)
                  CALL CIRCF(P1DT(1,1), P1DT(2,1), 0.01)
                  CALL CIRCF(P2DT(1,1), P2DT(2,1), 0.01)
```

Next, the pins for actuator AB were drawn.

```
                  CALL CIRCF(PBDT(1,2), PBDT(2,2), 0.01)
                  CALL CIRCF(P1DT(1,2), P1DT(2,2), 0.01)
```

Finally, the pins for actuator DE were drawn.

```
                  CALL CIRCF(P1DT(1,3), P1DT(2,3), 0.01)
                  CALL CIRCF(PWDT(1,1), PWDT(2,1), 0.01)
```

The actuators were drawn by the subroutine **DRACT** (draw actuators). The line style could be changed through the IRIS routine RESETL. A solid line was used, but a variety of other line styles were available.

```
RESETLV = 1
CALL RESETL(RESETLV)
```

Next, the thickness of the line was determined by the IRIS command LINEWI. The value in LINEWV denoted the width of the line by the number of pixels. The line for the actuators was a width of three pixels.

```
LINEWV=3
CALL LINEWI(LINEWV)
```

The actuators were drawn by connecting th two pin locations by a line. first, actuator AB was drawn.

```
CALL MOVE(PBDT(1,2),PBDT(2,2),BASEDI/2)
CALL DRAW(P1DT(1,2),P1DT(2,2),S1THIC/2)
```

Finally, actuator De was drawn.

```
CALL MOVE(PBDT(1,2),PBDT(2,2),BASEDI/2)
CALL DRAW(P1DT(1,2),P1DT(2,2),S1THIC/2)
```

The line width was reset before leaving this routine.

```
LINEWV=1
CALL LINEWI(LINEWV)
```

4.4 Kinematics

The kinematics of the robot arm were completed by the routine **tform**, (transform). An **A** matrix was created for each link of the arm and also for the world rotation:

AWO — World Rotation	A2 — Segment #2
A0 — Base	A3 — Segment #3
A1 — Segment #1	Aw — Wing Segment

Each of the **A** matrices were declared as 4x4 real arrays.

REAL AWO(4,4), AW(4,4), A0(4,4), A1(4,4), A2(4,4), A3(4,4)

Input into the routine included the magnitudes of the angles, the deformed coordinates of each link's vertices, and the deformed coordinates of all the pins.

4.41 Transformation of the User's Perspective. The world rotation A matrix, AWO, consisted of no translation but a rotation about each axis, x, y, and z. A rotation matrix for the x—axis was:

$$\mathbf{A}_x = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & \cos\theta_x & \sin\theta_x & 0.0 \\ 0.0 & -\sin\theta_x & \cos\theta_x & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{4.15}$$

The rotation matrix for the y—axis was:

$$\mathbf{A}_y = \begin{bmatrix} \cos\theta_y & 0.0 & -\sin\theta_y & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ \sin\theta_y & 0.0 & \cos\theta_y & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{4.16}$$

The rotation matrix for the z—axis was:

$$\mathbf{A}_z = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0.0 & 0.0 \\ -\sin\theta_z & \cos\theta_z & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{4.17}$$

The rotation submatrix was then the product of the three rotation matrices:

$$AWO = A_x * A_y * A_z \tag{4.18}$$

This matrix allowed the user to rotate the entire object as if they were walking around the arm. The magnitudes of the angles of rotation were stored in the variables ANGX, ANGY, and ANGZ. The elements of the AWO matrix were:

AWO(1,1)=COS(ANGY)*COS(ANGZ)

AWO(2,1)=(COS(ANGZ)*SIN(ANGX)*SIN(ANGY))+(COS(ANGX)*SIN(ANGZ))

AWO(3,1)=(–COS(ANGX)*SIN(ANGY)*COS(ANGZ))+(SIN(ANGX)*SIN(ANGZ))

AWO(1,2)=–COS(ANGY)*SIN(ANGZ)

AWO(2,2)=(–SIN(ANGX)*SIN(ANGY)*SIN(ANGZ))+(COS(ANGX)*COS(ANGZ))

AWO(3,2)=(COSE(ANGX)*SIN(ANGY)*SIN(ANGZ))+(SIN(ANGX)*COS(ANGZ))

AWO(1,3)=SIN(ANGY)

AWO(2,3)=–COS(ANGY)*SIN(ANGX)

AWO(3,3)=COS(ANGX)*COS(ANGY)

AWO(4,4)=1.0

To support the fact that no translation existed, rows one through three of column #4 were equal to zero. Also, no perspective transformations existed as can be seen from the fact that columns one through three of row #4 equalled zero. The scaling factor was equal to one. The rotation of the world coordinates was the first step in building the homogeneous transformation matrix T

$$T = AWO. \tag{4.19}$$

4.42 Transform the Base. The next step was to associate the base's local coordinate frame with the reference coordinate frame, (see figure 4.14). Initially, the two frames were the same, but, as the base was allowed to rotate about the y–axis, this relationship changed. Since the base did not translate about the reference frame, the **A0** matrix was composed only of a rotation about the y–axis. The scaling factor equalled one. The magnitude of the angle of rotation was stored in the variable ANG0CNT.

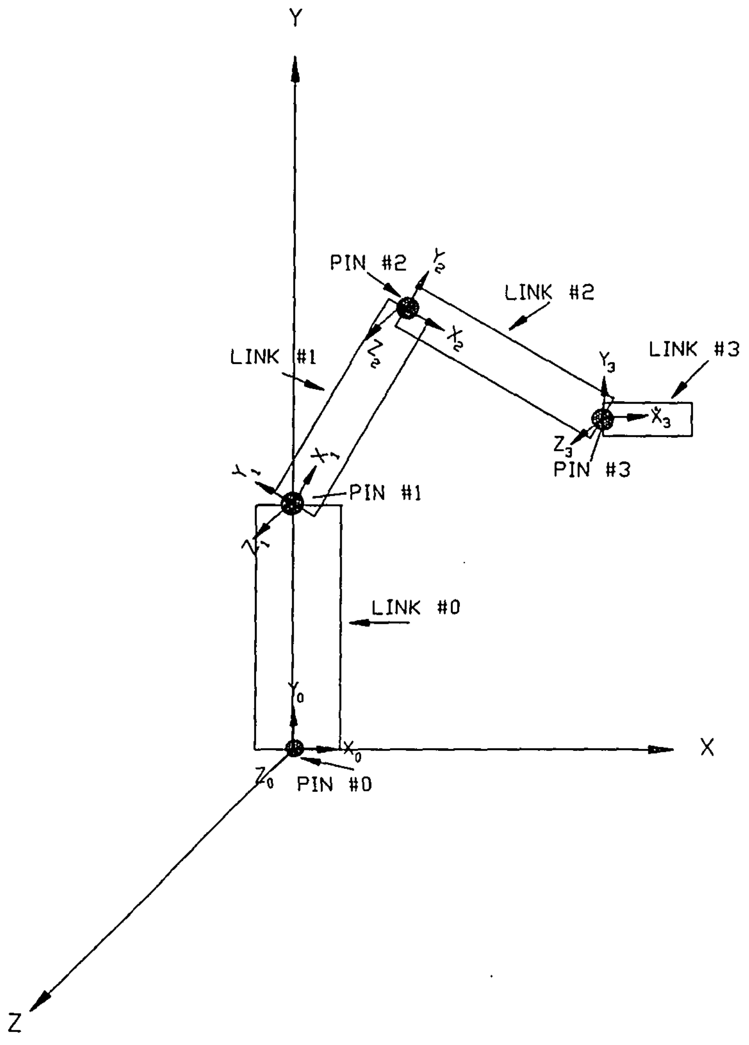


FIGURE 4.14

LOCAL AND REFERENCE COORDINATE FRAMES

$$\mathbf{A0} = \begin{bmatrix} \cos\phi_o & 0.0 & \sin\phi_o & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -\sin\phi_o & 0.0 & \cos\phi_o & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{4.20}$$

The programmed matrix was:

```
A0(1,1)=COS(ANG0CNT)
A0(1,3)=SIN(ANG0CNT)
A0(3,1)=-SIN(ANG0CNT)
A0(3,3)=COS(ANG0CNT)
A0(2,2)=1.0
A0(1,4)=0.0
A0(2,4)=0.0
A0(3,4)=0.0
A0(4,4)=1.0
```

The matrix **T** was given by:

$$\mathbf{T} = \mathrm{AWO}^*\mathrm{AO}. \tag{4.21}$$

To multiply the two matrices, the routine **MAT2MUL** (2–d matrix multiplication) was called. **MAT2MUL** could only multiply arrays which were two–dimensional. Three arrays were sent to **MAT2MUL**. The first two arrays were multiplied in order and the product was placed in the third array. An integer variable specified the number of columns to be multiplied because the arrays which held the pin coordinates were also two–dimensional and were multiplied in this routine.

```
CALL MAT2MUL(AWO,A0,T,4)
```

After **T** had been determined all the base vertices and coordinates of the pins on the base had to be transformed. Based on equation 3.9,

$$\mathrm{P}^{\mathrm{T}} = \mathrm{T}^*\mathrm{P} \text{ or,} \tag{4.22}$$

$$\mathrm{BSDT} = \mathrm{T}^*\mathrm{BS}. \tag{4.23}$$

BSDT stored the base coordinates which had been transformed. The Base coordinates were multiplied by calling the subroutine **MATBMUL**, base matrix multiplication. This routine multiplyed a (4x4) array by a (4x8x2) array. To accomplish this, a Do—Loop was used.

```
      DO 11 I = 1,2
            CALL MATBMUL(T,BS,BSDT,I)
11    CONTINUE
```

The Base was described by labeling the bottom as level #1 and the top as level #2. The integer value I, then repersented the level of coordinates. The pins located on the base were transformed by the call

```
      CALL MAT2MUL(T,PB,PBDT,2).
```

The value two denoted that only two columns contained values.

<u>4.43 Transform Segment #1.</u> Segment #1 was the next link to be transformed. The A matrix, **A1**, was composed of a rotation about the z—axis, a translation equal to the distance Pin #1 was located from the reference point (see figure 4.14) a scaling factor equal to one, and no perspective transformation. The angle of rotation was stored in the variable ANG1CNT. The translation, based on the location of pin #1, was stored in the array locations (1,1), (2,1), and (3,1) of the array PB. The values stored were those originally read from the parameter file, PIN1X and PIN1Y. The z—component of the pin location was zero because the center of the pin was taken to be at z = 0.0.

$$\mathbf{A}_1 = \begin{bmatrix} \cos\phi_1 & \sin\phi_1 & 0.0 & PB(1,1) \\ -\sin\phi_1 & \cos\phi_1 & 0.0 & PB(2,1) \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{4.24}$$

and in the program was:

```
      A1(1,1)=COS(ANG1CNT)
      A1(1,2)=-SIN(ANG1CNT)
      A1(2,1)=SIN(ANG1CNT)
```

```
A1(2,2)=COS(ANG1CNT)

A1(3,3)=1.0

A1(1,4)=PB(1,1)

A1(2,4)=PB(2,1)

A1(3,4)=PB(3,1)

A1(4,4)=1.0.
```

The **T** matrix was then given by:

$$\mathbf{T} = \text{AWO*A0*A1.} \tag{4.45}$$

From the translation of the Base, the matrix T had previously been defined as:

$$\mathbf{T} = \text{AWO*A0.} \tag{4.26}$$

So, subsstituting equation 4.26 into 4.25 yielded

$$\mathbf{T} = \text{T*A1.} \tag{4.27}$$

The matrix multiplication routine would not allow the product to be stored in an array that was also being used for the multiplication. Therefore, a temporary variable, TA was used.

$$\text{TA} = \text{T*A1.} \tag{4.28}$$

Then,

$$\mathbf{T} = \text{TA.} \tag{4.29}$$

To transfer the contents of TA into T after the multiplication had taken place the matrix routine **MXTRAN** (matrix transfer) was called. This routine transfered the contents of the second array into the first.

```
CALL MAT2MUL(T,A1,TA,4)

CALL MXTRAN(T,TA).
```

To transform the vertices and pins to their new coordinates of Segment #1, calls to **MATSMUL** and **MAT2MUL**, respectivley were made. **MATSMUL** (matrix multiplicaton of segments), multiplied arrays which were three–dimensional, (4x4100). Since the user could divide the segment into smaller link segments, the index value of 100 allowed the user to subdivide the segment into 99 sections. Each subsection was marked by a level. Then, each level's coordinate information was stored on the appropriate third dimension of the array. The number of subsections or for Segment #1 was stored in the variable S1DIVN. Each level had to be transformed. A Do–loop was called which was incremented from 1 to (S1DIVN+1). A value of one waa added to S1DIVN because if the user divides the segment into five subsections, there would be a total of six levels (see figure 4.10). MAT2MUL transformed the pins into the reference coordinate system. There were a total of three pins on Segment #1, stored as follows:

Column #1	Pin #2
Column #2	Pin B
Column #3	Pin D

The following FORTRAN staatements transformed the pins and vertices of Segment #1 from the their local coordinate system to the global coordinate system with its origin at the base of the robot.

```
      CALL MAT2MUL(T,P1D,P1DT,3)
      S1J=S1DIVN+1
      DO 101 I=1,S1J
             CALL MATSMUL(T,S1D,S1DT,I)
101   CONTINUE
```

<u>4.44 Transform Segment #2.</u> Segment #2 was the next to be transformed. There were two parts of Segment #2, the actual beam, and the Wing (see figure 4.11). The coordinates of the beam were stored in the array S2D and the coordinates of the Wing are stored in WINGD and were all based on the local coordinate system origin at Pin #2. The local x–axis extended along Segment #2's length and the z–axis was perpendicular to Pin #2. A2 was composed of a rotation about the z–axis by a value of ANG2CNT, a scaling factor of one, no perspective transformation, and the translation. The translation was the coordinates of Pin #2 (see figure 4.14) which were the deformed coordinates and were stored in the array P1D, first column.

$$A_2 = \begin{bmatrix} \cos\phi_2 & \sin\phi_2 & 0.0 & P1D(1,1) \\ -\sin\phi_2 & \cos\phi_2 & 0.0 & P1D(2,1) \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{4.30}$$

which was programmed as:

```
A2(1,1)=COS(ANG2CNT)

A2(1,2)=-SIN(ANG2CNT)

A2(2,1)=SIN(ANG2CNT)

A2(2,2)=COS(ANG2CNT)

A2(3,3)=1.0

A2(1,4)=P1D(1,1)

A2(2,4)=P1D(2,1)

A2(3,4)=P1D(3,1)

A2(4,4)=1.0.
```

The beam coordinates could be transformed by T which was now

$$T = AWO*A0*A1*A2. \tag{4.31}$$

Based on the substitution of equation 4.26 into 4.25

$$TA=T*A2 \tag{4.32}$$

So,

```
CALL MAT2MUL(T,A2,TA,4)
```

Then, TA was substituted back into T

```
CALL MXTRAN(T,TA).
```

As with Segment #1, the coordinates of Segment #2 were multiplied by calling **MATSMUL** and the pins were multiplied by calling **MAT2MUL.**

```
CALL MAT2MUL(T,P1D,P1DT,3)
```

Segment #2 could also be subdivided based on the number of subsections the user selected. S2DIVN contained the user defined number of subdivisions which meant there were S2DIVN+1 number of levels.

```
      S2J=S2DIVN+1
      DO 202 I=1,S2J
            CALL MATSMUL(T,S2D,S2DT,I)
202   CONTINUE
```

<u>4.45 Transform the Wing Segment.</u> The Wing segment differed from Segment #2 by the angle gamma (see figure 4.12). There was no translational relationship between the Wing Segment and the beam segment of Segment #2. Therefore, the matrix AW only contained a rotation about the z–axis through the angle GAMMA and a scaling factor of one.

$$A_W = \begin{bmatrix} \cos\gamma & \sin\gamma & 0.0 & 0.0 \\ -\sin\gamma & \cos\gamma & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \quad (4.33)$$

```
      AW(1,1)=COS(GAMMA)
      AW(1,2)=-SIN(GAMMA)
      AW(2,1)=SIN(GAMMA)
      AW(2,2)=COS(GAMMA)
      AW(3,3)=1.0
      AW(4,4)=1.0.
```

The T matrix should now have been

$$T = AWO^*A0^*A1^*A2^*AW \quad (4.34)$$

or

$$TA=T^*AW. \quad (4.35)$$

The multiplication was completed and then the TA into T transfer of,

```
            CALL MAT2MUL(T,AW,TA,4)
            CALL MXTRAN(T,TA).
```

Since the Wing was considered to be rigid, the user could not subdivide the segment.

Therefore, the coordinates of the Wing could be thought of as being in two levels. A Do—Loop which was completed twice will be used to transform all of the Wing Segment. The Pin E resides on the Wing Segment. MAT2MUL will be called to transform Pin E.

```
      CALL MAT2MUL(T,PWD,PWDT,1)
      DO 201 I=1,2
            CALL MATSMUL(T,WINGD,WINGDT,I)
201   CONTINUE
```

<u>4.46 Transform Segment #3.</u> The final link was Segment #3. This link was connected to the end of the beam of Segment #2, not the Wing. The A matrix for Sgment #3 consisted of a rotation about the local z—axis, a scaling factor of one, and a translation equal to the coordinates of Pin #3 (see figure 4.14).

$$A_3 = \begin{bmatrix} \cos\phi_3 & \sin\phi_3 & 0.0 & 0.0 \\ -\sin\phi_3 & \cos\phi_3 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{4.36}$$

```
      A3(1,1)=COS(ANG3CNT)
      A3(1,2)=-SIN(ANG3CNT)
      A3(2,1)=SIN(ANG3CNT)
      A3(2,2)=COS(ANG3CNT)
      A3(3,3)=1.0
      A3(1,4)=P2D(1,1)
      A3(2,4)=P2D(2,1)
      A3(3,4)=P2D(3,1)
      A3(4,4)=1.0.
```

The homogeneous transformation matrix, T, for Segment #3 was

$$T = AWO*A0*A1*A2*A3. \tag{4.37}$$

But, at the present time,

$$T = AWO*A0*A1*A2*AW. \tag{4.38}$$

The new T could not be found by simply multiplying

$$TA = T*A3 \tag{4.39}$$

because Segment #3 was not related to the Wing Segment. Therefore, T had to be reconstructed by the following commands

```
          CALL MAT2MUL(AWO,A0,T,4)
          CALL MAT2MUL(T,A1,TA,4)
          CALL MXTRAN(T,TA).
          CALL MAT2MUL(T,A2,TA,4)
          CALL MXTRAN(T,TA).
          CALL MAT2MUL(T,A3,TA,4)
          CALL MXTRAN(T,TA).
```

Now, the coordinates of Segment #3 could be transformed by

```
      DO 301 I=1,2
          CALL MATSMUL(T,S3D,S3DT,I)
301   CONTINUE.
```

At the completion of the multiplication of Segment #3, the kinematics were completed and the links were now ready to be drawn.

4.5 Static Analysis

Static analysis was controlled by the routine STATIC. The static conditions were dependent upon the geometry of the robotic arm. The geometric configuration was dependent upon the angles of the base and three segments which were altered by the user. The interactive input from the user was obtained by turning the eight dials of the Dial Box, the angles of the Base and three segments were adjusted either positively or negatively in magnitude by the eight dials. Each link was given an initial angle in the parameter file so that the analysis could be completed when the program was started. The static analysis was completed through the following eight steps:

(1) Determine the Static Reactions **(STAREAC)**

(2) Determine the Differential Change in Angles and the Virtual Displacement of the Actuators **(DADJANG)**

(3) Determine the Deflections, Torsional Twist, and Axial Extension

(4) Complete the Kinematic Analysis **(SDEFLX, SDEFLY, SDEFLZ)**

(5) Draw the Three–Dimensional Robotic Arm Image **(SDRROB)**

(6) Allow User Interaction (SUSR)

(7) Based on user interaction, stop or continue the model.

(8) If model is continued, Check Validity of the Altered Angle (SANGLC)

The eight steps were in a pseudo Do–loop, 100. If the segment angle input by the user exceeded the limits defined by the user in the parameter file, the user was instructed of the error and asked for a new value. If the user decided to stop the program, the variable PNE was then set equal to 999. The analysis was continued when the command

```
GO TO 100
```

was completed.

```
100     CALL STAREAC
        CALL SADJANG
        CALL SDEFLX
        CALL SDEFLY
        CALL SDEFLZ
        CALL STFORM
        CALL SDRROB
102     CALL SUSR
        IF(PNE.EQ.999.0) GO TO 999                    (Stop the program)
        CALL SANGLC
        IF(PNE.EQ.2.0)THEN
                GO TO 102
        ENDIF
        GO TO 100
```

4.51 Determine the Static Reactions. The static reactions were determined through the routine **STAREAC** (static reactions). All reactions were programmed based on equations 3.26 through 3.57 which were determined by Bannoura. The magnitudes of the static forces were stored in the array RS and the magnitudes of the static moments were stored in the array MS. Both arrays were declared as follows:

```
REAL*8 RS(3,5), MS(3,5), FABS, FDES.
```

The three rows of the arrays stored the x, y, and z components of the force or moment reactions at pins #0, 1, 2, 3, and the the load, #4. The five columns represented the five pins. As an example, the force in the y–direction at pin #2 would have been stored in RS(2,3).

To conserve computation time and minimize the length of the subroutine, the moment arms, see figures 3.17 through 3.21, weights, and the trigonometric functions were determined in the beginning of the program.

Moment arms:

P01=PIN1Y

P12=PIN2X–S1P1X

P23=PIN3X–S2P2X

P34=S3LENG–S3P3X

P2E=PINEX

P1B=PINBX–S1P1X

PD1=PINDX–S1P1X

Centers of gravity moment arms:

P0G0=BASEHE*0.5

P1G1=S1CMAX–S1P1X

P2G2=S2CMAX–S2P2X

P3G3=P34

P4G4=S4CMAX

Weights:

W0=GRAVITY*BASMAS

W1=GRAVITY*S1MASS

W2=GRAVITY*S2MASS

W3=GRAVITY*S3MASS

W4=GRAVITY*S4MASS

Trigonometric functions:

CO0=COS(ANG0CNT)

SI0=SIN(ANG0CNT)

CO1=COS(ANG1CNT)

SI1=SIN(ANG1CNT)

The values stored in the variables ANG2CNT, ANG3CNT, and ANG6CNT were angles with respect to the x–axis of the previous member. All analysis routines required that the angles be measured with respect to the horizontal. To compensate, the correct angle of a given segment was the sum of that angle and all previous angles.

$$\phi_2^T = \phi_1 + \phi_2 \tag{4.40}$$

$$\phi_6^T = \phi_6 + \phi_1 \tag{4.41}$$

$$\phi_3^T = \phi_1 + \phi_2 + \phi_3. \tag{4.42}$$

```
CO2=COS(ANG1CNT+ANG2CNT)

SI2=SIN(ANG1CNT+ANG2CNT)

CO3=COS(ANG1CNT+ANG2CNT+ANG3CNT)

SI3=SIN(ANG1CNT+ANG2CNT+ANG3CNT)

CO4=COS(ANG4CNT)

SI4=SIN(ANG4CNT)

CO5=COS(ANG5CNT)

SI5=SIN(ANG5CNT)

CO6=COS(ANG1CNT+ANG6CNT)

SI6=SIN(ANG1CNT+ANG6CNT)
```

The reactions are solved by first determining the reactions of the load and working backwards to Pin #3, Pin #2, Pin #1, and then, Pin #0.

The theoretical equations for the load were:

$$R_{4x}^{s} = 0.0 \tag{4.43}$$

$$R_{4y}^{s} = \text{Weight}_{\text{load}} \tag{4.44}$$

$$R^{s}{}_{4z} = 0.0 \tag{4.45}$$

$$M_{4x}^{s} = -\,P_{4g4}{*}\sin\phi_0{*}\cos\phi_3{*}W_4 \tag{4.46}$$

$$M_{4y}^{s} = 0.0 \tag{4.47}$$

$$M_{4z}^{s} = -\,P_{4g4}{*}\cos\phi_0{*}\cos\phi_3{*}W_4 \tag{4.48}$$

$$R_{0x}^{s} = R^{s}{}_{4x} \tag{4.49}$$

The program lines representing these equations for the load were:

RS(1,5)=0.0

RS(2,5)=–W4

RS(3,5)=0.0

MS(1,5)=–P4G4*SI0*CO3*W4

MS(2,5)=0.0

MS(3,5)=–P4G4*CO0*CO3*W4

The theoretical equations for Pin #3 were:

$$R^s_{3x} = R^s_{4x} \tag{4.50}$$

$$R^s_{3y} = -W_3 + R^s_{4y} \tag{4.51}$$

$$R^s_{3z} = R^s_{4z} \tag{4.52}$$

$$M^s_{3x} = P_{34} \sin\phi_3 {*} R^s_{4z} + M^s_{4x} \tag{4.53}$$

$$M^s_{3y} = -P_{34} \cos\phi_3 {*} R^s_{4z} + M^s_{4y} \tag{4.54}$$

$$M^s_{3z} = P_{34} \cos\phi_3 {*} R^s_{4y} - P_{34} \sin\phi_3 {*} R^s_{4x} - P_{3g3} \cos\phi_3 {*} W_3 + M^s_{4z} \tag{4.55}$$

The program lines reperesenting these equations at Pin #3 were:

RS(1,4)=RS(1,5)

RS(2,4)=RS(2,5)–W3

RS(3,4)=RS(3,5)

MS(1,4)=P34*SI3*RS(3,5)+MS(1,5)

MS(2,4)=–P34*CO3*RS(3,5)+MS(2,5)

M4ZT1=P34*CO3*RS(2,5)

M4ZT2=P34*SI3*RS(1,5)

M4ZT3=P34*CO3*W3

MS(3,4)=M4ZT1–M4ZT2–M4ZT3+MS(3,5)

Theoretical equation for Actuator DE was:

$$F^{s}_{de}=(-P_{23}\cos\phi_{2}*R^{s}_{3y}+P_{23}\sin\phi_{2}*R^{s}_{3x}-P_{2g2}\cos\phi_{2}*W_{2}-M^{s}_{3z})/$$
$$(P_{2e}\sin\phi_{6}\cos\phi_{5}-P_{2e}\cos\phi_{6}\sin\phi_{5}) \qquad (4.56)$$

The program line respresenting this equation for Actuator force DE was:

```
FDET1=P23*(-CO2*RS(2,4)+SI2*RS(1,4))

FDET2=P2G2*CO2*W2-MS(3,4)

FDET3=P2E*(SI6*CO5-SI5*CO6)

FDES=(FDET1+FDET2)/FDET3
```

The theoretical equations for Pin #2 were:

$$R^{s}_{2x}=-F^{s}_{de}\cos\phi_{5}+R^{s}_{3x} \qquad (4.57)$$
$$R^{s}_{2y}=-W_{2}+R^{s}_{3y}-F^{s}_{de}\sin\phi_{5} \qquad (4.58)$$
$$R^{s}_{2z}=R^{s}_{3z} \qquad (4.59)$$
$$M^{s}_{2x}=P_{23}\sin\phi_{2}*R^{s}_{3z}+M^{s}_{3x} \qquad (4.60)$$
$$M^{s}_{2y}=-P_{23}\cos\phi_{2}*R^{s}_{3z}+M^{s}_{3y} \qquad (4.61)$$
$$M^{s}_{2z}=0.0 \qquad (4.62)$$

The program lines representing these equations for Pin #2 were:

```
RS(1,3)=-FDES*CO5+RS(1,4)

RS(2,3)=-W2+RS(2,4)-FDES*SI5

RS(3,3)=RS(3,4)

MS(1,3)=P23*SI2*RS(3,4)+MS(1,4)

MS(2,3)=-P23*CO2*RS(3,4)+MS(2,4)

MS(3,3)=0.0
```

The theory equations for Actuator AB was:

$$F^s_{ab} = (P_{1d}\cos\phi_1 * F^s_{de}\sin\phi_5 - P_{1d}\sin\phi_1 * F^s_{de}\cos\phi_5 -$$
$$P_{1g1}\cos\phi_1 * W_1 + P_{12}\cos\phi_1 * R^s_{2y} - P_{12}\sin\phi_1 * R^s_{2x})/$$
$$(P_{1b}\cos\phi_1\sin\phi_4 - P_{1b}\sin\phi_1\cos\phi_4) \tag{4.63}$$

The program lines representing this equation for Actuator AB was:

```
FABT1=FDES*PD1*(CO1*SI5-CO5*SI1)

FABT2=-P1G1*CO1*W1+P12*(RS(2,3)*CO1-RS(1,3)*SI1)

FABT3=P1B*(CO1*SI4-SI1*CO4)

FABS=(FABT1+FABT2)/FABT3
```

The theoretical equations for Pin #1 were:

$$R^s_{1x} = F^s_{de}\cos\phi_5 - F^s_{ab}\cos\phi_4 + R^s_{2x} \tag{4.64}$$
$$R^s_{1y} = F^s_{de}\sin\phi_5 - W_1 - F^s_{ab}\sin\phi_4 + R^s_{2y} \tag{4.65}$$
$$R^s_{1z} = R^s_{2z} \tag{4.66}$$
$$M^s_{1x} = P_{12}\sin\phi_1 * R^s_{2z} + M^s_{2x} \tag{4.67}$$
$$M^s_{1y} = -P_{12}\cos\phi_1 * R^s_{2z} + M^s_{2y} \tag{4.68}$$
$$M^s_{1z} = 0.0 \tag{4.69}$$

The program lines representing these equations for Pin #1 were:

```
RS(1,2)=FDES*CO5-FABS*CO4+RS(1,3)

RS(2,2)=FDES*SI5-W1-FABS*SI4+RS(2,3)

RS(3,2)=RS(3,3)

MS(1,2)=P12*SI1*RS(3,3)+MS(1,3)

MS(2,2)=-P12*CO1*RS(3,3)+MS(2,3)
```

The theoretical equations for Pin #0 were:

$$R^s_{ox} = R^s_{4x} \tag{4.70}$$
$$R^s_{0y} = -W_0 - W_1 - W_2 - W_3 + R^s_{4y} \tag{4.71}$$

$$R^s_{0z}=R^s_{4z} \tag{4.72}$$

$$M^s_{0x}=(P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+ \\ P_{34}\sin\phi_3)R^s_{4z}+M^s_{4x} \tag{4.73}$$

$$M^s_{0y}=-(P_{12}\cos\phi_1+P_{23}\cos\phi_2+ \\ P_{34}\cos\phi_3)R^s_{4z}+M^s_{4y} \tag{4.74}$$

$$M^s_{0z}=-P_{1g1}\cos\phi_1{*}W_1-(P_{12}\cos\phi_1+P_{2g2}\cos\phi_2)W_2- \\ (P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{3g3}\cos\phi_3)W_3+ \\ (P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{34}\cos\phi_3)R^s_{4y}- \\ (P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{34}\sin\phi_3)R^s_{4x}+M^s_{4z} \tag{4.75}$$

The program lines representing these equations for Pin #0 were:

```
RS(1,1)=RS(1,5)

RS(2,1)=RS(2,5)–W0–W1–W2–W3

RS(3,1)=RS(3,5)

MS(1,1)=(P01+P12*SI1+P23*SI2+P34*SI3)*RS(3,5)+MS(1,5)

MS(2,1)=–(P12*CO1+P23*CO2+P34*CO3)*RS(3,5)+MS(2,5)

M0ZT1=–P1G1*CO1*W1

M0ZT2=–(P12*CO1+P2G2*CO2)*W2

M0ZT3=–(P12*CO1+P23*CO2+P3G3*CO3)*W3

M0ZT4=(P12*CO1+P23*CO2+P34*CO3)*RS(2,5)

M0ZT5=–(P01+P12*SI1+P23*SI2+P34*SI3)*RS(1,5)+MS(3,5)

MS(3,1)=M0ZT1+M0ZT2+M0ZT3+M0ZT4+M0ZT5
```

4.52 Determine the Differential Change in the Angles and Virtual Displacements of the Actuators. The virtual displacements of the actuators were found in the routine **SADJANG** (static model, adjust angles). The program was based on equations 3.61 and 3.70. First, the differential change due to actuator AB was determined. The length of the actuator was required.

```
BETA = 1.5706+ANG1CNT
H1A1=PIN1Y–PINAY
H1B1=PINBX–S1P1X
H1D1=H1A1+H1B1*SIN(ANG1CNT)
H1E1=H1B1*COS(ANG1CNT)
H1C12=H1A1**2+H1B1**2–(2*H1A1*H1B1*COS(BETA))
H1LENG=SQRT(H1C12)
```

From the length of the actuator AB, the angle of the actuator, ϕ_4 could be determined.

```
ANG4CNT=ASIN(H1D1/H1LENG)
```

Based on equation 3.61, the differential change DPHI1, based on axial compression was calculated.

```
DPHI1T1=H1C12*FABS
DPHI1T2=H1A1*H1B1*SIN(BETA)*H1A*H1E
DPHI1=DPHI1T1/(DPHI1T2*57.3)
```

The differential change affected ϕ_1 so, ϕ_1 had to be adjusted by the magnitude DPHI1.

```
ANG1CNT=ANG1CNT+DPHI1
```

Next, the length of Actuator DE was determined.

```
KAPPA=3.1459–ANG1CNT
ALPHA=6.28318–(ANG1CNT+ANG6CNT)–KAPPA
H2A2=PINEX
H2B2=PIN2X–PINDX
H2C22=H2A2**2+H2B2**2–2*H2A2*H2B2*COS(KAPPA)
H2LENG=SQRT(H2C22)
```

The differential change of ϕ_2, DPHI2, based on equation 3.70, was then a function of the axial compression of Actuator DE.

```
SIGMA=ACOS((H2A2**2–H2B2**2–H2C22)/(2*H2B2*H2LENG))
ANG5CNT=SIGMA+ANG1CNT
```

DPHI2T1=H2C22*FDES

DPHI2T2=H2A2*H2B2*H2A*H2E*SIN(ALPHA)

DPHI2=DPHI2T1/(DPHI2T2*57.3)

Both ϕ_2 and ϕ_6 were effected by the $\delta\phi_2$.

ANG2CNT=ANG2CNT+DPHI2

ANG6CNT=ANG6CNT+DPHI2

At this point, the angles, considered to be true to the given conditions, and the force and moment reactions were known.

4.53 Determine the Deflections, Torsional Twist, and Axial Extension.

The torsional twist, axial extension, and deflections could now be determined. The arrays which stored the deformation effects were:

S1DE — Segment #1, deformations

S2DE — Segment #2, deformations

S1SL — Segment #1, slopes of deformations

S2SL — Segment #2, slopes of deformations.

As with the arrays which held the magnitudes of the reactions, the four deformation arrays are also declared as (3,100). The three rows are for the x, y, and z deflection components. Since segments #1 and #2 could be divided into subsections, the deformation was determined at each level between the lower and upper pin locations of the segment. The 100 columns acted as storage for each level. The location S2DE(2,70), for example, stored the deflection of segment #2 in the y direction at level #70.

The torsional twist and axial extension were considered only to act in the x direction. The magnitudes of these deformations were determined in the routine SDEFLX (static model, x deflection). The cosine and sine of the angles were found initially to save space and computation time. Again, all angles were measured with respect to the horizontal.

CO1=COS(ANG1CNT)

SI1=SIN(ANG1CNT)

```
CO2=COS(ANG1CNT+ANG2CNT)
SI2=SIN(ANG1CNT+ANG2CNT)
```

XTEMP was a variable used to store XI, the value of the x distance, from the last round of computations. DEFL was a variable used to store the deflection determined in the last round. The variable DELTA stored the change in length due to torsional twist, U1X stored the axial extension, P2XD stored the deformation at the location of Pin #2, and P3XD stored the deformation at the location of Pin #3. Initially, all variables were zero.

```
XTEMP=0.0
DEFL=0.0
DELTA=0.0
U1X=0.0
P2XD=0.0
P3XD=0.0
```

Do—loop 100 was responsible for determining the deformation along Segment #1. The deformations were calculated at each level number, so the limits of the Do—loop were from 1 to (S1DIVN+1), the last level. Once Pin #2 was reached, I was set equal to (S1DIVN+1) so that the deformation calculations were stopped for Segment #1.

```
DO 100 I=1,(S1DIVN+1)
```

The x—distance, which was incremented to equal the distance of each level number, was determined. The deflection and slope were calculated at each level

```
XI=(S1LENG/S1DIVN)*(I–1)
```

The first calculation was performed at the location of Pin #1, so, if the x—distance was less than the distance of pin #1, the calculations were not completed and the loop was incremented.

```
IF(XI.LT.S1P1X) GO TO 199
```

First, the axial extension was determined.

$$u_{x1} = \frac{R^s_{2x}\cos\phi_1 + R^s_{2y}\sin\phi_1}{L_1 A_1 E_1}(x_1) \qquad (4.76)$$

The programmed equations were:

```
U1XT1=(RS(1,3)*CO1+RS(2,3)*SI1)*XI

U1XT2=P1(1,1)*S1A*S1E

U1X=U1XT1/U1XT2
```

The torsional twist was determined based on the equation from Bannoura of

$$\theta_{x1} = \frac{M^s_{2x}\cos\phi_1 + M^s_{2y}\sin\phi_1}{L_1 J_{1x} G_1}(x_1) \qquad (4.77)$$

which became

```
TH1XT1=(MS(1,3)*CO1+MS(2,3)*SI1)*XI

TH1XT2=(P1(1,1)*S1JO*S1G)

TH1X=TH1XT1/(TH1XT2*57.3)
```

in the program. Based on the torsional twist, there was a change in the x–components of the segment. But, if the torsional twist angle is too small, then the change was considered equal to zero.

```
DELTA=XI*(1–COS(TH1X))
```

The next section determined the deflection associated with the pins on Segment #1. If a pin was located between the present x location, XI, and the previous x location, XTEMP the deflection was interpolated for that pin. First, Pin #2 was considered. If the location of Pin #2 was reached, then the deformation of the pin was stored in the variables P2XD so that segments #2, #3, and the wing would be adjusted.

The value S1DM was a multiplier defined by the user so that the deflection values could be exaggerated for graphical presentation purposes. The multiplier was not used in the storage of analytical information.

```
        IF(P1(1,1).EQ.XI) THEN
          P1D(1,1)=P1(1,1)+(DELTA+U1X)*S1DM
          P2XD=(DELTA+U1X)*S1DM
        ENDIF
        IF(P1(1,1).GT.XTEMP.AND.P1(1,1).LT.XI) THEN
          DEFLT=((DELTA+U1X-DEFL)/(XI-XTEMP))*(P1(1,1)-XTEMP)
          P1D(1,1)=P1(1,1)+(DEFLT*S1DM)
          P2XD=DEFLT*S1DM
        ENDIF
```

Determine if pin B lies within the range.

```
        IF(P1(1,2).EQ.XI) P1D(1,2)=P1(1,2)+(DELTA+U1X)*S1DM
        IF(P1(1,2).GT.XTEMP.AND.P1(1,2).LT.XI) THEN
          DEFLT=((DELTA+U1X-DEFL)/(XI-XTEMP))*(P1(1,2)-XTEMP)
          P1D(1,2)=P1(1,2)+(DEFLT*S1DM)
        ENDIF
```

Determine if pin D lies within the range.

```
        IF(P1(1,3).EQ.XI) P1D(1,3)=P1(1,3)+(DELTA+U1X)*S1DM
        IF(P1(1,3).GT.XTEMP.AND.P1(1,3).LT.XI) THEN
          DEFLT=((DELTA+U1X-DEFL)/(XI-XTEMP))*(P1(1,3)-XTEMP)
          P1D(1,3)=P1(1,3)+(DEFLT*S1DM)
        ENDIF
```

The slope, deflection, and x coordinate were now ready to be stored in the proper array and/or temporary variables.

```
        S1SL(1,I)=TH1X
        S1DE(1,I)=DELTA+U1X
199     XTEMP=XI
        DEFL=DELTA+U1X
```

Next, the vertices of segment #1 were adjusted through Do—loop 101.

```
                        DO 101 J=1,4
                         S1D(1,J,II)=S1(1,J,II)+((DELTA+U1X)*S1DM)
101          CONTINUE
```

If Pin #2 was reached on the last calculation, then that was the last calculation. At this point though, the remaining level coordinates needed to be transferred to the array S1D. To accomplished this, the level number, I, was incremented. This was the first level of coordinates which were not deformed. The last level was still (S1DIVN+1).

```
             IF(XI.GE.P1(1,1)) THEN
```

Transfer the coordinates of any level after Pin #2 to the array S1D.

```
             DO 198 II=(I+1),(S1DIVN+1)
                  DO 198 J=1,4
                  S1D(1,J,II)=S1(1,J,II)+((DELTA+U1X)*S1DM)
      198    CONTINUE
             I=S1DIVN+1
             ENDIF
```

Next, the deformation for Segment #2 was determined. Do—loop 200 completed the calculations for every level number between Pin #2 and Pin #3. The algorithm was the same as that for Segment #1. First, the temporary variables were reset to zero.

```
             XTEMP=0.0
             DEFL=0.0
             DO 200 I=1,(S2DIVN+1)
                  XI=(S2LENG/S2DIVN)*(I–1)
                  IF(XI.LT.S2P2X) GO TO 299
                  XI2=XI*XI
```

The axial extension, U2X, was determined. The equation from Bannoura was:

$$u_{x2} = \frac{R^{s}_{3x}\cos\phi_2 + R^{s}_{3y}\sin\phi_2}{L_2 A_2 E_2}(x_2) \tag{4.78}$$

The programmed equation was:

```
U2XT1=(RS(1,4)*CO2+RS(2,4)*SI2)*XI

U2XT2=(P2(1,1)*S2A*S2E)

U2X=(U2XT1/U2XT2)
```

The torsional twist, TH2X, was determined based on the equation

$$\theta_{x2} = \frac{M^{s}_{3x}\cos\phi_2 + M^{s}_{3y}\sin\phi_2}{L_2 J_{2x} G_2}(x_2) \tag{4.79}$$

from Bannoura which became

```
TH2XT1=(MS(1,4)*CO2+MS(2,4)*SI2)*XI

TH2XT2=(P2(1,1)*S2JO*S2G)

TH2X=TH2XT1/(TH2XT2*57.3)
```

in the program. The axial deformation due to the torsional twist was determined.

```
DELTA=XI*(1-(COS(TH2X))
```

The vertices of Segment #2 were adjusted.

```
          DO 201 J=1,4

            S2D(1,J,I)=S2(1,J,I)+(DELTA+U2X)*S2DM + P2XD

201       CONTINUE
```

Only one pin was located on Segment #2, Pin #3. Once the deformation for Pin #3 was determined, the value of the deformation was stored in P3XD.

```
                    IF(P2(1,1).EQ.XI) THEN
                     P2D(1,1)=P2(1,1)+(DELTA+U2X)*S2DM+P2XD
                     P3XD=(DELTA+U2X)*S2DM+P2XD
                    ENDIF
                    IF(P2(1,1).GT.XTEMP.AND.P2(1,1).LT.XI) THEN
                      DEFLT=((DELTA+U2X–DEFL)/(XI–XTEMP))*(P2(1,1)–XTEMP)
                      P2D(1,1)=P2(1,1)+(DEFLT*S2DM)+P2XD
                      P3XD=DEFLT*S2DM+P2XD
                    ENDIF
```

The values for the slope and deflection were stored in the proper arrays. The present deflection value and x–coordinate were also stored in the temporary variables.

```
                    S2SL(1,I)=TH2X
                    S2DE(1,I)=DELTA+U2X
299                 XTEMP=XI
                    DEFL=DELTA+U2X
```

If Pin #3 was reached on the last calculation, then that was the last calculation. At this point though, the remaining level coordinates must be transferred to the array S2D. To accomplished this, the level number, I, was incremented. This was the first level of coordinates which were not deformed. The last level was still (S2DIVN+1).

```
                IF(XI.GE.P2(1,1)) THEN
```

Transfer the coordinates of any level after Pin #3 to the array S2D.

```
                DO 298 II=(I+1),(S2DIVN+1)
                    DO 298 J=1,4
                S2D(1,J,II)=S2(1,J,II)+((DELTA+U2X)*S2DM)
      298   CONTINUE
            I=S2DIVN+1
            ENDIF
200   CONTINUE
```

Segment #3 and the Wing segment were adjusted.

```
            DO 300 I=1,2
                  DO 300 J=1,4
                   S3D(1,J,I)=S3(1,J,I)
300   CONTINUE
            DO 400 I=1,2
                  DO 400 J=1,4
                   WINGD(1,J,I)=WING(1,J,I)
400   CONTINUE
```

The x coordinate of Pin E was adjusted.

```
            PWD(1,1)=PW(1,1)
```

This concluded the calculations of deformation in the x direction.

The deflection in the y direction was determined in the routine SDEFLY. This routine followed the same basic algorithm as SDEFLX. The differences layed in the deformation equations. RL1 and RL2 were the force reactions in the local y—axis.

The theoretical equations of these forces were:

$R^{l}_{(i+1)y_i}$ — is the force reaction in the local y_i—axis.

$$R^{l}_{(i+1)y} = R^{s}_{(i+1)y}\cos\phi_i - R^{s}_{(i+1)x}\sin\phi_i. \tag{4.80}$$

The program lines representing these equations were:

```
            RL1=RS(2,3)*CO1–RS(1,3)*SI1
            RL2=RS(2,4)*CO2–RS(1,4)*SI2
```

The theoretical equations for the deformation in the y direction were:

$$V_i = \frac{R^l_{(i+1)yi}}{6E_i I_{zi}}(3L_i x_i^2 - x_i^3) + \frac{M^s_{(i+1)zi}}{2E_i I_{zi}} x_i^2 \tag{4.81}$$

$$\theta_{zi} = \frac{R^l_{(i+1)yi}}{6E_i I_{zi}}(6L_i x_i - 3x_i^2) - \frac{M^s_{(i+1)zi}}{E_i I_{zi}} x_i \tag{4.82}$$

The program lines representing these equations for the deformation in the y direction for Segment #1 were:

```
V1T1=RL1/(6*S1E*S1IZ)

V1T2=3*P1(1,1)*XI2–XI3

V1=V1T1*V1T2

TH1ZT1=RL1/(6*S1E*S1IZ)

TH1ZT2=6*P1(1,1)*XI–3*XI2

TH1Z=TH1ZT1*TH1ZT2
```

The programmed equations for the deformation in the y direction of Segment #2 were:

```
V2T1=RL2/(6*S2E*S2IZ)

V2T2=3*P2(1,1)*XI2–XI3

V2T3=(MS(3,4)*XI2)/(2*S2E*S2IZ)

V2=V2T1*V2T2+V2T3

TH2ZT1=RL2/(6*S2E*S2IZ)

TH2ZT2=6*P2(1,1)*XI–3*XI2

TH2ZT3=(MS(3,4)*XI)/(S2E*S2IZ)

TH2Z=TH2ZT1*TH2ZT2–TH2ZT3
```

The deflection in the z direction was determined in the routine SDEFLZ. This routine also followed the basic algorithm presented in the routine DEFLFX. The difference in the routines was in the equations of deformation. ML1 and ML2 were the moment reactions in the local y — axis. From theory, the equations were:

$M^{l}_{(i+1)y_i}$ — is the moment reaction in the local y_i–axis.

$$M^{l}_{(i+1)y_i} = M^{s}_{(i+1)y}\sin\phi_i + M^{s}_{(i+1)x}\cos\phi_i \tag{4.83}$$

The programmed equations were:

```
ML1=MS(2,3)*SI1+MS(1,3)*CO1

ML2=MS(2,4)*SI2+MS(1,4)*CO2
```

The theoretical equations for the deformation in the z direction were:

$$W_i = \frac{R^{s}_{(i+1)zi}}{6E_i I_{yi}}(3L_i x_i^2 - x_i^3) - \frac{M^{l}_{(i+1)yi}}{2E_i I_{yi}} x_i^2 \tag{4.84}$$

$$\theta_{yi} = \frac{R^{s}_{(i+1)zi}}{6\ E_i I_{yi}}(6L_i x_i - 3x_i^2) - \frac{M^{l}_{(i+1)yi}}{E_i I_{yi}} x_i \tag{4.85}$$

The programmed equations for Segment #1 were:

```
W1T1=RS(3,3)/(6*S1E*S1IY)

W1T2=3*P1(1,1)*XI2–XI3

W1T3=(ML1*XI2)/(2*S1E*S1IY)

W1=W1T1*W1T2–W1T3

TH1YT1=RS(3,3)/(6*S1E*S1IY)

TH1YT2=6*P1(1,1)*XI–3*XI2

TH1YT3=(ML1*XI)/(S1E*S1IZ)

TH1Y=TH1YT1*TH1YT2–TH1YT3
```

For Segment #2, the programmed equations were:

```
W2T1=RS(3,4)/(6*S2E*S2IY)

W2T2=3*P2(1,1)*XI2–XI3

W2T3=(ML2*XI2)/(2*S2E*S2IY)

W2=W2T1*W2T2–W2T3

TH2YT1=RS(3,4)/(6*S2E*S2IY)

TH2YT2=6*P2(1,1)*XI–3*XI2
```

```
TH2YT3=(ML2*XI)/(S2E*S2IZ)

TH2Y=TH2YT1*TH2YT2–TH2YT3
```

4.54 Complete the Kinematic Analysis. At this point, all of the deformed coordinates have been stored in the arrays S1D, S2D, S3D, WINGD, P1D, P2D, and PWD. The next step was to transform the coordinates to their respective location based on the angles of the segments and the location of the connecting pins. The routine STFORM completed the kinematic analysis. Section 4.3 Kinematics, gave a full explanation of the program.

4.55 Draw the Three–Dimensional Robotic Arm Image. The output of the routine STFORM contained the transformed coordinates. The segments could now be drawn on the screen. All of the drawing tasks were completed through the routine SDRROB (static model, draw robot). This routine simply acted as a manager for controlling when each drawing routines were called. As discussed in Section 4.4 **Three–Dimensional Graphics**, there were specific drawing routines for each link.

4.56 Allow User Interaction. After the image was completely drawn on the screen, the user was allowed to interact with the model. The routine SUSR (static model user interaction) completed different actions which were dependent upon which dial, button, or key the user selected. Section 4.2 **User Interaction** discussed the user capabilities.

4.57 Check the Validity of the Altered Angle. If the user decided to change the angle of one of the segments through the dials, the new angle was checked before any calculations were completed again. This check of validity was based on the limits of the allowable angles the user had specified in the parameter file. The user was allowed to specify the maximum and minimum angles of each segment as well as the maximum and minimum lengths of the actuators. The routine SANGLC (static model) angle check, then compared the possible new angle with the user defined limits. The computed GO TO statement allows the checking to be kept to a minimum. If only Segment #3 has been altered, then the angles of Segments #1 and #2 were still valid. The varible AD contained the number of the segment which has just been altered.

```
        GO TO (100, 200, 300, 400) AD
```

The variable PNE was a flag which determined if the change was a increment or a decrement of the angle. If AD was equal to zero, then the control of the routine went to the next line after the GO TO and the base was altered.

```
        IF(PNE.EQ.1.0) THEN
              ANG0CNT=ANG0CNT+ANGINC
        ELSE
              ANG0CNT=ANG0CNT–ANGINC
        ENDIF
```

The base is allowed to rotate 360°. Therefore, the only check is to keep the angle under 360°.

```
        IF(ANG0CNT.GT.6.2832) ANG0CNT=ANG0CNT–6.2832
        IF(ANG0CNT.LT.0.0) ANG0CNT=ANG0CNT+6.2832
```

If the base was altered, no other angle needed to be adjusted. The program control then returned to the routine STATIC.

```
        GO TO 999
```

The next angle that was checked was that of Segment #1, ANG1CNT. First, the angle was altered and the new value was temporarily stored. Since an alteration of Segment #1 also altered Segment #2 and Segment #3, the validity of both actuators as well as the validity of all the angles had to be checked. If one was determined to be invalid, then, the temporary values were not passed back into the angle variables.

```
  100   IF(PNE.EQ.1.0) THEN
              ANG1T=ANG1CNT+ANGINC
        ELSE
              ANG1T=ANG1CNT–ANGINC
        ENDIF
```

The length of Actuator AB was determined and compared with its allowable limits.

```
          BETA = 1.5706+ANG1T
          H1A1=PIN1Y–PINAY
          H1B1=PINBX–S1P1X
          H1D1=H1A1+H1B1*SIN(ANG1T)
          H1E1=H1B1*COS(ANG1T)
          H1C1=SQRT(H1E1**2+H1D1**2)
          H1LENG=H1C1
          IF(H1LENG.LT.H1MINL.AND.H1LENG.GT.H1MAXL) THEN
               PNE=2.0
               WRITE(*,*)' ACTUATOR 1 AT LIMIT', H1LENG
```

If the length was out of bounds, then, this routine was terminated and the control returned to the STATIC routine which then returned to allow the user to interact again.

```
               GO TO 999
          ENDIF
```

If the actuator AB length was within allowable limits, then, the new angle of Segment #1 was checked.

```
          IF(ANG1T.LT.ANG1MX) GO TO 101
               PNE=2.0
               WRITE(*,*)' ANGLE ONE AT LIMIT: ', (ANG1T*57.3)
               GO TO 999
    101   IF(ANG1T.GT.ANG1MN) GO TO 102
          PNE=2.0
          WRITE(*,*)' ANGLE ONE AT LIMIT: ', (ANG1T*57.3)
          GO TO 999
```

If the length of Actuator AB and the temporary value of the angle of Segment #1 were within the limits, then, the temporary variable for ϕ_4 could be determined.

```
          ANG4T=ASIN(H1D11/H1LENG)
```

Next, the angle of Segment #2 was checked. If Segment #2 had been the segment originally altered, this would have been the first group of commands altered. As with ϕ_1, if Segment #2 was the one adjusted, the angles ϕ_2 and ϕ_6 were adjusted and those values were stored in a temporary variables.

```
200         IF(PNE.EQ.1.0) THEN
             ANG2T=ANG2CNT+ANGINC
             ANG6T=ANG6CNT+ANGINC
      ELSE
             ANG2T=ANG2CNT−ANGINC
            ANG6T=ANG6CNT−ANGINC
      ENDIF
```

Next, the length of Actuator DE was computed. As with Actuator AB, if the length was out of range, then no values were changed and the program returned to the user interaction subroutine.

```
      KAPPA=1.5706−ANG1CNT
      ALPHA=6.28318−ANG6T−KAPPA
      H2A2=PINEX
      H2B2=PIN2X−PINDX
      H2C22=H2A2**2+H2B2**2−2*H2A2*H2B2*COS(KAPPA)
      H2LENG=SQRT(H2C22)
      IF(H2LENG.LT.H2MINL.AND.H2LENG.GT.H2MAXL) THEN
            WRITE(*,*)' ACTUATOR 2 AT LIMIT', H2LENG
            PNE=2.0
            GO TO 999
      ENDIF
```

Next, the new value of Segment #2 was checked.

```
        IF(ANG2T.LT.ANG2MX) GO TO 201
        PNE=2.0
        WRITE(*,*)' ANGLE TWO AT LIMIT: ', (ANG2T*57.3)
        GO TO 999
201     IF(ANG2T.GT.ANG2MN) GO TO 202
        PNE=2.0
        WRITE(*,*)' ANGLE TWO AT LIMIT: ', (ANG2T*57.3)
        GO TO 999
202
```

Finally, Segment #3 was altered and checked.

```
300     IF(PNE.EQ.1.0) THEN
                ANG3CNT=ANG3CNT+ANGINC
        ELSE
                ANG3CNT=ANG3CNT–ANGINC
        ENDIF
        IF(ANG3CNT.LT.ANG3MX) GO TO 301
        PNE=2.0
        ANG3CNT=ANG3CNT–ANGINC
        GO TO 999
301             IF(ANG3CNT.GT.ANG3MN) THEN
        PNE=2.0
        ANG3CNT=ANG3CNT+ANGINC
```

If ϕ_3 had been determined to be valid, then all temporary variables were passed to the angle variables. If the user altered the world view of the robot, then, the magnitude of the angles were checked to stay below 360^o.

```
400    IF(ANGX.GT.6.283)ANGX=ANGX–6.283
       IF(ANGY.GT.6.283)ANGY=ANGY–6.283
       IF(ANGZ.GT.6.283)ANGZ=ANGZ–6.283
```

At the completion of the routine SANGLC, the entire static analysis had been completed. The next step would be to either return to the user interaction routine or return to the top of the routine STATIC to start the static analysis process for the new configuration.

4.6 Dynamic Analysis

The dynamic analysis was controlled by the routine **DYNAMIC**. The conditions were dependent upon the geometry of the arm. But, the geometry of the arm was dependent upon the forces of the actuators which the user had input through the dynamic input file, **dynin.dat**. There were a total of twelve steps required to complete the dynamic analysis, to allow user interaction, and to produce a three–dimensional graphics image:

(1) Determine the Static Reactions (**DSTREAC**)

(2) Determine the Applied and Dynamic Actuator Forces (**DACTFOR**)

(3) Determine the Inertial Reactions (**DYNINER**)

(4) Determine the Velocities and Accelerations (**DVELAC**)

(5) Determine the New Angles Based on the New Time Step (**DADJANG**)

(6) Determine the new Static Reactions (**DSTREAC**)

(7) Determine the Dynamic Reactions (**DYNREAC**)

(8) Determine the Deformation (**DDEFLX, DDEFLY, DDEFLZ**)

(9) Complete the Kinematic Analysis (**DTFORM**)

(10) Draw the Three–Dimensional Robotic Arm Image (**DDRROB**)

(11) Allow User Interactions. (**DUSR**)

(12) Based on the number of input actuator forces or the user interaction, continue or stop the model.

The twelve steps were placed in a Do–loop in the routine **DYNAMIC**:

```
        CALL DSTREAC
DO 100 NF=1,NFACT
        CALL DACTFOR
        CALL DYNINER
        CALL DVELAC
        CALL DADJANG
```

```
          IF(PNE.EQ.2.0) THEN
            WRITE(*,*)' RETURNING TO CALCULATE THE NEXT
            ACTUATOR FORCE'
            GO TO 100
          ENDIF
          CALL DSTREAC
          CALL DYNREAC
          CALL DDEFLX
          CALL DDEFLY
          CALL DDEFLZ
          CALL DTFORM
          CALL DDRROB
          CALL DUSR
          IF(PNE.EQ.999.0) GO TO 999    (Stop the model)
  100     CONTINUE
```

The number of times the Do—loop was completed was based on the number of entries in the dynamic input file, **dynin.dat**. This value was stored in the variable NFACT. If the new angles were not within the limits or the length of the actuators based on the new angles were not within the limits, then the analysis was not completed for the given applied actuator forces. The control returns to determine the next pair of applied and dynamic actuator forces.

4.61 Determine the Static Reactions. The static reactions were determined in the routine **DSTREAC** (dynamic model, static reactions). This routine was based on the static reactions of equations 3.99 through 3.100 which were also used in the routine **STAREAC**. The routine is basically the same as **STAREAC**. The only difference was that **STAREAC** was called in the static model and referenced the common block, **SCOMMON**, and the routine

DSTREAC was called in the dynamic model and referenced the dynamic common block, **DCOMMON**.

4.62 Determine the Applied and Dynamic Actuator Forces. After the static reactions had been determined, the routine **DACTFOR** (dynamic model actuator forces) could determine the applied and dynamic forces of the actuators. The applied forces were the actuator forces from the input file **dynin.dat**. The dynamic forces were, based on equation ——, the difference of the applied and static forces:

FABD = FACTI(1,NF) – FABS

FDED = FACTI(2,NF) – FDES

The variable NF was the number the Do–loop of the DYNAMIC routine was equal to at that time. NF could then be used to access the array FACTI which held the actuator forces the user had specified in the dynamic input file.

4.63 Determine the Inertial Reactions. Based on the new dynamic actuator forces, the inertial reactions, $\vec{Q} = m\vec{a}$ and $\vec{T} = I\vec{\alpha}$ could now be determined by the routine **DYNINER** (dynamic model, inertial reactions). The reactions were determined by solving Bannoura's dynamic reaction equations for the inertial terms, QD and TD. QD and Td were both real arrays used to store the magnitudes of the inertial reactions. QD stored the forces while TD stored the moments. Both were declared to be (3,4) arrays where the three rows stored the (x,y,z) components of the reactions and the columns were each a different pin location. As an example, TD(2,3) would have stored the inertial torque in the y direction at Pin #2. The reactions were determined first at Pin #3, then Pin #2, #1 and finally #0. The routine DYNINER followed the same basic outline as the static reaction routines. First, the cosine and sine of the angles, weights, and moment arms were determined. The theoretical equations for Pin #3 were:

$$Q^{d}_{g3x} = R^{d}_{3x} - R^{s}_{4x} \qquad (4.86)$$

$$Q^{d}_{g3y} = R^{d}_{3y} + W_{3} - R^{s}_{4y} \qquad (4.87)$$

$$Q^{d}_{g3z} = R^{d}_{3z} - R^{s}_{4z} \tag{4.88}$$

$$\begin{aligned} T^{d}_{g3x} = {} & M^{d}_{3x} - P_{34}\sin\phi_3 * R^{s}_{4z} - M^{s}_{4x} - \\ & P_{3g3}\sin\phi_3 * Q^{d}_{g3z} \end{aligned} \tag{4.89}$$

$$T^{d}_{g3y} = M^{d}_{3y} + P_{34}\cos\phi_3 * R^{s}_{4z} - M^{s}_{4y} + P_{3g3}\cos\phi_3 * Q^{d}_{g3z} \tag{4.90}$$

$$\begin{aligned} T^{d}_{g3z} = {} & M^{d}_{3z} - P_{34}\cos\phi_3 * R^{s}_{4y} + P_{34}\sin\phi_3 * R^{s}_{4x} + \\ & P_{3g3}\cos\phi_3 * W_3 - M^{s}_{4z} - P_{3g3}\cos\phi_3 * Q^{d}_{g3y} + \\ & P_{3g3}\sin\phi_3 * Q^{d}_{g3x} \end{aligned} \tag{4.91}$$

The program equivalent was:

```
QD(1,4)=RD(1,4)–RS(1,5)

QD(2,4)=RD(2,4)+W3–RS(2,5)

QD(3,4)=RD(3,4)–RS(3,5)

TD(1,4)=MD(1,4)–(P34*RS(3,5)+P3G3*QD(3,4))*SI3–MS(1,5)

TD(2,4)=MD(2,4)+(P34*RS(3,5)+P3G3*QD(3,4))*CO3–MS(2,5)

M3ZT1=(P34*RS(2,5)–P3G3*W3+P3G3*QD(2,4))*CO3

M3ZT2=(P34*RS(1,5)+P3G3*QD(1,4))*SI3

TD(3,4)=MD(3,4)–M3ZT1+M3ZT2–MS(3,5)
```

The theoretical equations for Pin #2 were:

$$Q^{d}_{g2x} = R^{d}_{2x} + F^{s}_{de}\cos\phi_5 - R^{s}_{3x} + F^{d}_{de}\cos\phi_5 - R^{d}_{3x} \tag{4.92}$$

$$\begin{aligned} Q^{d}_{g2y} = {} & R^{d}_{2y} + W_2 - R^{s}_{3y} + F^{s}_{de}\sin\phi_5 + \\ & F^{d}_{de}\sin\phi_5 - R^{d}_{3y} \end{aligned} \tag{4.93}$$

$$Q^{d}_{g2z} = R^{d}_{2z} - R^{s}_{3z} - R^{d}_{3z} \tag{4.94}$$

$$\begin{aligned} T^{d}_{g2x} = {} & M^{d}_{2x} - P_{23}\sin\phi_2 * R^{s}_{3z} - M^{s}_{3x} - P_{2g2}\sin\phi_2 * Q^{d}_{g2z} - \\ & P_{23}\sin\phi_2 * R^{d}_{3z} - M^{d}_{3x} \end{aligned} \tag{4.95}$$

$$T^d_{g2y} = M^d_{2y} + P_{23}\cos\phi_2 * R^s_{3z} - M^s_{3y} - P_{2g2}\cos\phi_2 * Q^d_{g2z} + P_{23}\cos\phi_2 * R^d_{3z} - M^d_{3y} \tag{4.96}$$

$$T^d_{g2z} = 0.0 \tag{4.97}$$

The program lines representing the equations were:

```
QD(1,3)=RD(1,3)+(FDES+FDED)*CO5–RS(1,4)–RD(1,4)

QD(2,3)=RD(2,3)+(FDES+FDED)*SI5+W2–RS(2,4)–RD(2,4)

QD(3,3)=RD(3,3)–RS(3,4)–RD(3,4)

M2XT1=(P23*RS(3,4)+P2G2*QD(3,3)+P23*RD(3,4))*SI2

TD(1,3)=MD(1,3)–M2XT1–MS(1,4)–MD(1,4)

M2YT1=(–P23*RS(3,4)+P2G2*QD(3,3)–P23*RD(3,4))*CO2

TD(2,3)=MD(2,3)–M2YT1–MS(2,4)–MD(2,4)

TD(3,3)=0.0
```

The theoretical equations for Pin #1 were:

$$Q^d_{g1x} = R^d_{1x} - F^s_{de}\cos\phi_5 + F^s_{ab}\cos\phi_4 - R^s_{2x} - F^d_{de}\cos\phi_6 + F^d_{ab}\cos\phi_4 - R^d_{2x} \tag{4.98}$$

$$Q^d_{g1y} = R^d_{1y} - F^s_{de}\sin\phi_5 + W_1 + F^s_{ab}\sin\phi_4 - R^s_{2y} - F^d_{de}\sin\phi_6 + F^d_{ab}\sin\phi_4 - R^d_{2y}$$

(4.99)

$$Q^d_{g1z} = R^d_{1z} - R^s_{2z} - R^d_{2z} \tag{4.100}$$

$$T^d_{g1x} = M^d_{1x} - P_{12}\sin\phi_1 * R^s_{2z} - M^s_{2x} - P_{1g1}\sin\phi_1 * Q^d_{g1z} - P_{12}\sin\phi_1 * R^d_{2z} - M^d_{2x} \tag{4.101}$$

$$T^d_{g1y} = M^d_{1y} + P_{12}\cos\phi_1 * R^s_{2z} - M^s_{2y} + P_{1g1}\cos\phi_1 * Q^d_{g1z} + P_{12}\cos\phi_1 * R^d_{2z} - M^d_{2y} \tag{4.102}$$

$$T^d_{g1z} = 0.0 \tag{4.103}$$

The program equivalent was:

```
R1XT1=-(FABS+FABD)*CO4+FDES*CO5+FDED*CO6

QD(1,2)=RD(1,2)-R1XT1-RS(1,3)-RD(1,3)

R1YT1=FDES*SI5+FDED*SI6-(FABS+FABD)*SI4-W1

QD(2,2)=RD(2,2)-R1YT1-RS(2,3)-RD(2,3)

QD(3,2)=RD(3,2)-RS(3,3)-RD(3,3)

M1XT1=((RS(3,3)+RD(3,3))*P12+P1G1*QD(3,2))*SI1

TD(1,2)=MD(1,2)-M1XT1-MS(1,3)-MD(1,3)

M1YT1=-((RS(3,3)+RD(3,3))*P12+P1G1*QD(3,2))*CO1

TD(2,2)=MD(2,2)-M1YT1-MS(2,3)-MD(2,3)

TD(3,2)=0.0
```

The theoretical equations for Pin #0 were:

$$Q^d_{g0x} = R^d_{0x} - R^s_{4x} - Q^d_{g1x} - Q^d_{g2x} - Q^d_{g3x} \tag{4.104}$$

$$Q^d_{g0y} = R^d_{0y} + W_0 + W_1 + W_2 + W_3 - R^s_{4y} - Q^d_{g1y} - Q^d_{g2y} - Q^d_{g3y} \tag{4.105}$$

$$Q^d_{g0z} = R^d_{0z} - R^s_{4z} - Q^d_{g1z} - Q^d_{g2z} - Q^d_{g3z} \tag{4.106}$$

$$T^d_{g0x} = M_{0x} - (P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{34}\sin\phi_3)R^s_{4z} - M^s_{4x} - P_{0g0}*Q^d_{g0x} - (P_{01}+P_{1g1}\sin\phi_1)Q^d_{g1z} - (P_{01}+P_{12}\sin\phi_1+P_{2g2}\sin\phi_2)Q^d_{g2z} - (P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{3g3}\sin\phi_3)Q^d_{g3z} - T^d_{g1x} - T^d_{g2x} - T^d_{g3x} \tag{4.107}$$

$$T^d_{g0y} = M^d_{0y} + (P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{34}\cos\phi_3)R^s_{4z} - M^s_{4y} + P_{1g1}\cos\phi_1 * Q^d_{g1z} + (P_{12}\cos\phi_1+P_{2g2}\cos\phi_2)Q^d_{g2z} + (P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{3g3}\cos\phi_3)Q^d_{g3z} - T^d_{g1y} - T^d_{g2y} - T^d_{g3y} \tag{4.108}$$

$$T^d_{g0z} = M^d_{0z} + P_{1g1}\cos\phi_1 * W_1 + (P_{12}\cos\phi_2+P_{2g2}\cos\phi_2)W_2 + (P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{3g3}\cos\phi_3)W_3 -$$

$$(P_{12}\cos\phi_1+P_{12}\cos\phi_2+P_{34}\cos\phi_3)R^s_{4y} +$$
$$(P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{34}\sin\phi_3)R^s_{4x} - M^s_{4z} +$$
$$P_{0z}{*}Q^d_{g0x} - P_{1g1}\cos\phi_1{*}Q^d_{g1y} + (P_{01}+P_{1g1}\sin\phi_1)Q^d_{g1x} -$$
$$(P_{12}\cos\phi_1+P_{2g2}\cos\phi_2)Q^d_{g2y} + (P_{01}+P_{12}\sin\phi_1+P_{2g2}\sin\phi_2)Q^d_{g2x} -$$
$$(P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{3g3}\cos\phi_3)Q^d_{g3y} +$$
$$(P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{3g3}\sin\phi_3)Q^d_{g3x} -$$
$$T^d_{g1z} - T^d_{g2z} - T^d_{g3z} \tag{4.109}$$

The program equivalent was:

```
QD(1,1)=RD(1,1)–RS(1,5)–QD(1,2)–QD(1,3)–QD(1,4)

QD(2,1)=RD(2,1)+(W0+W1+W2+W3)–RS(2,5)–QD(2,2)–QD(2,3)–QD(2,4)

QD(3,1)=RD(3,1)–RS(3,5)–QD(3,2)–QD(3,3)–QD(3,4)

M0XT1=(P01+P12*SI1+P23*SI2+P34*SI3)*RS(3,5)

M0XT2=(P01+P1G1*SI1)*QD(3,2)

M0XT3=(P01+P12*SI1+P2G2*SI2)*QD(3,3)

M0XT4=(P01+P12*SI1+P23*SI2+P3G3*SI3)*QD(3,4)

M0XT5=TD(1,2)+TD(1,3)+TD(1,4)

M0XT6=P0G0*QD(1,1)+MS(1,5)

TD(1,1)=MD(1,1)–M0XT1–M0XT2–M0XT3–M0XT4–M0XT5–M0XT6

M0YT1=–(P12*CO1+P23*CO2+P34*CO3)*RS(3,5)

M0YT2=–(P12*CO1+P2G2*CO2)*QD(3,3)

M0YT3=–(P12*CO1+P23*CO2+P3G3*CO3)*QD(3,4)

M0YT4=TD(2,2)+TD(2,3)+TD(2,4)

M0YT5=MS(2,5)+P1G1*QD(3,2)

TD(2,1)=MD(2,1)–M0YT1–M0YT2–M0YT3–M0YT4–M0YT5

M0ZT1=–P1G1*CO1*W1–(P12+P2G2)*CO2*W2

M0ZT2=–(P12*CO1+P23*CO2+P3G3*CO3)*W3

M0ZT3=(P12*CO1+P23*CO2+P34*CO3)*RS(2,5)
```

```
M0ZT4=-(P01+P12*SI1+P23*SI2+P34*SI3)*RS(1,5)
M0ZT5=P0G0*QD(1,1)
M0ZT6=P1G1*CO1*QD(2,2)-(P01+P1G1*SI1)*QD(1,2)
M0ZT7=(P12*CO1+P2G2*CO2)*QD(2,3)
M0ZT8=-(P01+P12*SI1+P2G2*SI2)*QD(1,3)
M0ZT9=(P12*CO1+P23*CO2+P3G3*CO3)*QD(2,4)
M0ZT10=(P01+P12*SI1+P23*SI2+P3G3*SI3)*QD(1,4)
M0ZT11=MS(3,5)+TD(3,2)+TD(3,3)+TD(3,4)
M0ZT12=M0ZT1+M0ZT2+M0ZT3+M0ZT4+M0ZT5+M0ZT6
M0ZT13=M0ZT7+M0ZT8+M0ZT9+M0ZT10+M0ZT11
TD(3,1)=MD(3,1)-M0ZT12-M0ZT13  .
```

4.64 Determine the Velocities and Accelerations. The velocities and accelerations of the Segments #1 and #2 were determined next. The inertial torques were a product of the moment of inertia tensor and the angular acceleration of the segment. Angular accelerations could be extracted from the inertial torques:

$$T^d_{g0x} = \bar{I}_{xx}*\alpha_{0x} \qquad T^d_{g0y} = \bar{I}_{yy}*\alpha_{0y} \qquad T^d_{g0z} = \bar{I}_{zz}*\alpha_{0z} \tag{4.110}$$

$$T^d_{g1x} = \bar{I}_{xx}*\alpha_{1x} \qquad T^d_{g1x} = \bar{I}_{yy}*\alpha_{1y} \qquad T^d_{g0z} = \bar{I}_{zz}*\alpha_{1z} \tag{4.111}$$

$$T^d_{g2x} = \bar{I}_{xx}*\alpha_{2x} \qquad T^d_{g2y} = \bar{I}_{yy}*\alpha_{2y} \qquad T^d_{g2z} = \bar{I}_{zz}*\alpha_{2z} \tag{4.112}$$

The angular velocities were then determined based on the equation

$$\vec{\omega} = \omega_i + \vec{\alpha}*\delta t \tag{4.113}$$

$$\omega_{0x} = \omega_{iox} + \alpha_{0x}*\delta t \qquad \omega_{0y} = \omega_{ioy} + \alpha_{0y}*\delta t$$

$$\omega_{0z} = \omega_{ioz} + \alpha_{0z}*\delta t \tag{4.114}$$

$$\omega_{1x} = \omega_{i1x} + \alpha_{1x}*\delta t \qquad \omega_{1y} = \omega_{i1y} + \alpha_{1y}*\delta t$$

$$\omega_{1z} = \omega_{i1z} + \alpha_{1z}*\delta t \tag{4.115}$$

$$\omega_{2x} = \omega_{i2x} + \alpha_{2x}*\delta t \qquad \omega_{2y} = \omega_{i2y} + \alpha_{2y}*\delta t$$

$$\omega_{2z} = \omega\alpha_{i2z} + \alpha_{2z}*\delta t \tag{4.116}$$

The angular accelerations were stored in the array ALPHA and the angular velocities were stored in the array OMEGA. Both arrays were (3,3) where the three rows stored the (x,y,z) components of the terms and the three rows were for the base, Segment #1, and Segment #2. The equations of the program were:

For the base:

```
ALPHA(1,1)=TD(1,1)/BIX
ALPHA(2,1)=TD(2,1)/BIY
ALPHA(3,1)=TD(3,1)/BIZ
OMEGA(1,1)=ALPHA(1,1)*DT + OMEGA(1,1)
OMEGA(2,1)=ALPHA(2,1)*DT + OMEGA(2,1)
OMEGA(3,1)=ALPHA(3,1)*DT + OMEGA(3,1)
```

For Segment #1:

```
ALPHA(1,2)=TD(1,2)/S1IX
ALPHA(2,2)=TD(2,2)/S1IY
ALPHA(3,2)=TD(3,2)/S1IZ
OMEGA(1,2)=ALPHA(1,2)*DT + OMEGA(1,2)
OMEGA(2,2)=ALPHA(2,2)*DT + OMEGA(2,2)
OMEGA(3,2)=ALPHA(3,2)*DT + OMEGA(3,2)
```

For Segment #2

```
ALPHA(1,3)=TD(1,3)/S2IX
ALPHA(2,3)=TD(2,3)/S2IY
ALPHA(3,3)=TD(3,3)/S2IZ
OMEGA(1,3)=ALPHA(1,3)*DT + OMEGA(1,3)
OMEGA(2,3)=ALPHA(2,3)*DT + OMEGA(2,3)
OMEGA(3,3)=ALPHA(3,3)*DT + OMEGA(3,3)
```

For Segment #1, the velocity was based on equation 3.131:

$$\vec{V}_1 = \vec{\omega}_0 \times \vec{r}_{2/1} + \vec{\omega}_1 \times \vec{r}_{2/1}. \tag{4.117}$$

The vector $\vec{r}_{2/1}$ was a variable based on the deformation of Segment #1. Since the velocity and acceleration were only determined over the distance form Pin #1 to Pin #2 on Segment #1, $\vec{r}_{2/1}$ was first set equal to the deformed location of Pin #1:

```
R21X=PBDT(1,1)
R21Y=PBDT(2,1)
R21Z=PBDT(3,1)
```

The velocity and acceleration were then determined at each level between the two pins. The vector $\vec{r}_{2/1}$ was then a function of the level number, I:

```
R21X=S1DT(1,1,I)
R21Y=S1DT(2,1,I)
R21Z=S1DT(3,1,I)
```

The final calculations on Segment #1 were at the location of Pin #2. At this point, the vector $\vec{r}_{2/1}$ was set equal to the deformed coordinates of Pin #2:

```
R21X=P1DT(1,1)
R21Y=P1DT(2,1)
R21Z=P1DT(3,1)
```

Two cross–products required for Segment #1 were determined next. In theory,

$$\begin{aligned}\vec{\omega}_1 \times \vec{r}_{2/1} = {} & (r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z})\hat{i} + \\ & (r_{2/1}x^*\omega_{1z} - r_{2/1}z^*\omega_{1x})\hat{j} + \\ & (r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y})\hat{k}.\end{aligned} \tag{4.118}$$

The program lines were:

```
W1R21X=R21Z*OMEGA(2,2)–R21Y*OMEGA(3,2)
W1R21Y=R21X*OMEGA(3,2)–R21Z*OMEGA(1,2)
W1R21Z=R21Y*OMEGA(1,2)–R21X*OMEGA(2,2)
```

The second cross–product in theory was:

$$\vec{\omega}_0 \times \vec{r}_{2/1} = (r_{2/1}z^*\omega_{0y})\hat{i} + (-r_{2/1}x^*\omega_{0y})\hat{k}. \tag{4.119}$$

In the program, the equations were:

```
WOR21X=R21Z*OMEGA(2,1)

WOR21Z=–R21X*OMEGA(2,1)
```

The velocity of Segment #1 at a given distance along Segment #1 was the sum of the two cross—products. In theory:

$$\vec{V}_1 = (r_{2/1}z^*\omega_{0y} + r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z})\hat{i} + (r_{2/1}x^*\omega_{1z} - r_{2/1}z^*\omega_{1x})\hat{j} + (-r_{2/1}x^*\omega_{0y} + r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y})\hat{k}. \tag{4.120}$$

In the program, the velocity of Segment #1 was stored in the array VE. The three rows stored the (x,y,z) components while the columns each held a different pin or level:

```
VE1(1,J)=WOR21X+W1R21X

VE1(2,J)=W1R21Y

VE1(3,J)=WOR21Z+W1R21Z
```

The acceleration of Segment #1 was based on the equation

$$\vec{a}_1 = \{\vec{\alpha}_0 \times \vec{r}_{2/1}\} + \{\vec{\omega}_0 \times (\vec{\omega}_0 \times \vec{r}_{2/1})\} + \{2\vec{\omega}_0 \times \dot{\vec{r}}_{1/0}\} + \{\vec{\omega}_1 \times \vec{\omega}_1 \times \vec{r}_{2/1}\} + \{\vec{\alpha}_1 \times \vec{r}_{2/1}\} \tag{4.121}$$

The cross—products were:

In analytical theory:

$$\{\vec{\alpha}_0 \times \vec{r}_{2/1}\} = r_{2/1}z^*\alpha_0y\ \hat{i} - r_{2/1}x^*\alpha_0y\ \hat{k} \tag{4.122}$$

In the program:

```
A0R21X=R21Z*ALPHA(2,1)

A0R21Z=–R21X*ALPHA(2,1)
```

In theory:

$$\{\vec{\omega}_0 \times (\vec{\omega}_0 \times \vec{r}_{2/1})\} = (-r_{2/1}x^*\omega_{0y}^{2})\hat{i} + (-r_{2/1}z^*\omega_{0y}^{2})\hat{k}. \tag{4.123}$$

In the program:

```
W0W0R1X=–R21X*OMEGA(2,1)**2

W0W0R1Z=–R21Z*OMEGA(2,1)**2
```

In theory:

$$\{2\vec{\omega}_0 \times \dot{\vec{r}}_{1/0}\} = 2^*\omega_{0y}(r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y})\hat{i} + 2^*\omega_{0y}(r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z})\hat{k} \tag{4.124}$$

In the program:

```
W02R10X=2*OMEGA(2,1)*(R21Y*OMEGA(1,2)–R21X*OMEGA(2,2))

W02R10Z=2*OMEGA(2,1)*(R21Z*OMEGA(2,2)–R21Y*OMEGA(3,2))
```

In theory:

$$\begin{aligned}\{\vec{\omega}_1 \times \vec{\omega}_1 \times \vec{r}_{2/1}\} = &\{\omega_{1y}(r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y}) - \\ &\omega_{1z}(r_{2/1}x^*\omega_{1z} - r_{2/1}z^*\omega_{1x})\}\hat{i} \\ &\{\omega_{1z}(r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z}) - \\ &\omega_{1x}(r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y})\}\hat{j} \\ &\{\omega_{1x}(r_{2/1}x^*\omega_{1z} - r_{2/1}z^*\omega_{1x}) - \\ &\omega_{1y}(r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z})\}\hat{k}\end{aligned} \tag{4.125}$$

In the program:

```
W1W1R1X=OMEGA(2,2)*(R21Y*OMEGA(1,2)–R21X*OMEGA(2,2))–

+OMEGA(3,2)*(R21X*OMEGA(3,2)–R21Z*OMEGA(1,2))

W1W1R1Y=OMEGA(3,2)*(R21Z*OMEGA(2,2)–R21Y*OMEGA(3,2))–

+OMEGA(1,2)*(R21Y*OMEGA(1,2)–R21X*OMEGA(2,2))
```

W1W1R1Z=OMEGA(1,2)*(R21X*OMEGA(3,2)–R21Z*OMEGA(1,2))–

+OMEGA(2,2)*(R21Z*OMEGA(2,2)–R21Y*OMEGA(3,2))

In theory:

$$\{\vec{\alpha}_1 \times \vec{r}_{2/1}\} =$$
$$\{r_{2/1}z^*\alpha_{1y} - r_{2/1}y^*\alpha_{1z}\}\hat{i} +$$
$$\{r_{2/1}x^*\alpha_{1z} - r_{2/1}z^*\alpha_{1x}\}\hat{j} +$$
$$\{r_{2/1}y^*\alpha_{1x} - r_{2/1}x^*\alpha_{1y}\}\hat{k} \qquad (4.126)$$

In the program:

```
      A1R21X=R21Z*ALPHA(2,2)–R21Y*ALPHA(3,2)
      A1R21Y=R21X*ALPHA(3,2)–R21Z*ALPHA(1,2)
      A1R21Z=R21Y*ALPHA(1,2)–R21X*ALPHA(2,2).
C
C     Sum up the acceleration terms
C
      AC1(1,J)=A0R21X+W0W0R1X+W02R21X+W1W1R1X+A1R21X
      AC1(2,J)=W1W1R1Y+A1R21Y
      AC1(3,J)=A0R21Z+W0W0R1Z+W02R21Z+W1W1R1Z+A1R21Z
```

The velocity of Segment #2 was described by the equation

$$\vec{V}_2 = \vec{V}_1 + \vec{\Omega}_{xyz} \times \vec{r}_{3/2} + \vec{\omega}_2 \times \vec{r}_{3/2}. \qquad (4.127)$$

The vector $\vec{r}_{3/2}$ was similar to that of $r_{2/1}$ for the calculations of Segment #1. The first calculation was at the location of Pin #2. The vector was:

```
      R32X=P1DT(1,1)
      R32Y=P1DT(2,1)
      R32Z=P1DT(3,1)
```

The next calculations for Segment #2 were at the different levels of Segment #2. The vector $\vec{r}_{3/2}$ was then a function of the level number:

```
R32X=S2DT(1,1,J)

R32Y=S2DT(2,1,J)

R32Z=S2DT(3,1,J)
```

The final calculation was at the location of Pin #3:

```
R32X=P2DT(1,1)

R32Y=P2DT(2,1)

R32Z=P2DT(3,1)
```

$\vec{\Omega}_{xyz}$ in theory was given by:

$$\vec{\Omega}_{xyz} = \omega_x\hat{i} + \omega_y\hat{j} + \omega_z\hat{k} \tag{4.128}$$

Where:

$$\omega_x = \omega_{1x} \tag{4.129}$$

$$\omega_y = \omega_{oy} + \omega_{1y} \tag{4.130}$$

$$\omega_z = \omega_{1z} \tag{4.131}$$

In the program, these became:

```
WXYZX=OMEGA(1,2)

WXYZY=OMEGA(2,1)+OMEGA(2,2)

WXYZZ=OMEGA(3,2)
```

To complete the calculation for the velocity of Segment #1, the two cross–products were determined:

In theory:

$$\begin{aligned}\vec{\Omega}_{xyz} \times \vec{r}_{3/2} = \\ &(r_{3/2}z^*\omega_y - r_{3/2}y^*\omega_z)\hat{i} + \\ &(r_{3/2}x^*\omega_z - r_{3/2}z^*\omega_x)\hat{j} + \\ &(r_{3/2}y^*\omega_x - r_{3/2}x^*\omega_y)\hat{k}.\end{aligned} \tag{4.132}$$

In the program:

```
WXYZR2X=R32Z*WXYZY–R32Y*WXYZZ

WXYZR2Y=R32X*WXYZZ–R32Z*WXYZX

WXYZR2Z=R32Y*WXYZX–R32X*WXYZY
```

In theory:

$$\begin{aligned}\vec{\omega}_2 \text{ x } \vec{r}_{3/2} = & \\ & (r_{3/2}z^*\omega_{2y} - r_{3/2}y^*\omega_{2z})\hat{i} + \\ & (r_{3/2}x^*\omega_{2z} - r_{3/2}z^*\omega_{2x})\hat{j} + \\ & (r_{3/2}y^*\omega_{2x} - r_{3/2}x^*\omega_{2y})\hat{k}.\end{aligned} \tag{4.133}$$

In the program:

```
W2R21X=R32Z*OMEGA(2,3)–R32Y*OMEGA(3,3)

W2R21Y=R32X*OMEGA(3,3)–R32Z*OMEGA(1,3)

W2R21Z=R32Y*OMEGA(1,3)–R32X*OMEGA(2,3)
```

The velocity of Segment #2 then, at a given distance along Segment #2 was the sum of the two cross products. It was stored in the array VE2 which has defined the same as VE1:

In theory:

$$\begin{aligned}\vec{V}_2 = & (r_{2/1}z^*\omega_{0y} + r_{2/1}z^*\omega_{1y} - r_{2/1}y^*\omega_{1z} + \\ & r_{3/2}z^*\omega_y - r_{3/2}y^*\omega_z + r_{3/2}z^*\omega_{2y} - r_{3/2}y^*\omega_{2z})\hat{i} + \\ & (r_{2/1}x^*\omega_{1z} - r_{2/1}z^*\omega_{1x} + r_{3/2}x^*\omega_z - \\ & r_{3/2}z^*\omega_x + r_{3/2}x^*\omega_{2z} - r_{3/2}z^*\omega_{2x})\hat{j} + \\ & (-r_{2/1}x^*\omega_{0y} + r_{2/1}y^*\omega_{1x} - r_{2/1}x^*\omega_{1y} + \\ & r_{3/2}y^*\omega_x - r_{3/2}x^*\omega_y + \\ & r_{3/2}y^*\omega_{2x} - r_{3/2}x^*\omega_{2y})\hat{k}.\end{aligned} \tag{4.134}$$

In the program:

```
VE2(1,I)=VE1(1,P2J)+WXYZR2X+W2R21X

VE2(2,I)=VE1(2,P2J)+WXYZR2Y+W2R21Y

VE2(3,I)=VE1(3,P2J)+WXYZR2Z+W2R21Z
```

The acceleration of Segment #2 was based on the equation

$$a_2 = a_1 + \{\vec{\dot{\Omega}}_{xyz} \times \vec{r}_{3/2}\} + \{\vec{\Omega}_{xyz} \times (\vec{\Omega}_{xyz} \times \vec{r}_{3/2})\} + \{2\vec{\Omega}_{xyz} \times \vec{\dot{r}}_{3/2}\} +$$
$$\{\vec{\omega}_2 \times \vec{\omega}_2 \times \vec{r}_{3/2}\} + \{\vec{\alpha}_2 \times \vec{r}_{3/2}\} \qquad (4.135)$$

The cross producst were:

First, $\vec{\dot{\Omega}}_{xyz}$ was determined. In theory,

$$\vec{\dot{\Omega}}_{xyz} = \vec{\alpha}_2 + \vec{\Omega}_{xyz} \times \vec{\omega}_2 \text{ and}$$
$$\vec{\Omega}_{xyz} \times \vec{\omega}_2 = (\omega_y\omega_{2z} - \omega_z\omega_{2y})\hat{i}$$
$$(\omega_y\omega_{2z} - \omega_z\omega_{2y})\hat{j}$$
$$(\omega_y\omega_{2z} - \omega_z\omega_{2y})\hat{k} \qquad (4.136)$$

In the program:

```
AXYZX=OMEGA(3,3)*(OMEGA(2,1)+OMEGA(2,2))
+—OMEGA(3,2)*OMEGA(2,3)+ALPHA(1,3)
AXYZY=OMEGA(3,2)*OMEGA(1,3)–OMEGA(3,3)*OMEGA(1,2)
++ALPHA(2,3)
AXYZZ=OMEGA(2,3)*OMEGA(1,2)–OMEGA(1,3)*(OMEGA(2,1)
++OMEGA(2,2))+ALPHA(3,3)
```

In theory:

$$\{\vec{\dot{\Omega}}_{xyz} \times \vec{r}_{3/2}\} =$$
$$(r_{3/2}z^*\dot{\Omega}_y - r_{3/2}x^*\dot{\Omega}_z)\hat{i} +$$
$$(r_{3/2}x^*\dot{\Omega}_z - r_{3/2}z^*\dot{\Omega}_x)\hat{j} +$$
$$(r_{3/2}y^*\dot{\Omega}_x - r_{3/2}x^*\dot{\Omega}_y)\hat{k} \qquad (4.137)$$

In the program:

```
AXYZR2X=R21Z*AXYZY–R21Y*AXYZZ
AXYZR2Y=R21X*AXYZZ–R21Z*AXYZX
AXYZR2Z=R21Y*AXYZX–R21X*AXYZY
```

In theory:

$$\{\vec{\Omega}_{xyz} \times (\vec{\Omega}_{xyz} \times \vec{r}_{3/2})\} =$$

$$\{\omega_y{*}(r_{3/2}y{*}\omega_x - r_{3/2}x{*}\omega_y) -$$
$$\omega_z{*}(r_{3/2}x{*}\omega_z - r_{3/2}z{*}\omega_x)\}\ \hat{i} +$$
$$\{\omega_z{*}(r_{3/2}z{*}\omega_y - r_{3/2}y{*}\omega_z) -$$
$$\omega_x{*}(r_{3/2}y{*}\omega_x - r_{3/2}x{*}\omega_y)\}\hat{j} +$$
$$\{\omega_x{*}(r_{3/2}x{*}\omega_z - r_{3/2}z{*}\omega_x) -$$
$$\omega_y{*}(r_{3/2}z{*}\omega_y - r_{3/2}y{*}\omega_z)\}\hat{k} \qquad (4.138)$$

In the program:

```
WXYZ2RX=WXYZR3Z*WXYZY–WXYZR3Y*WXYZZ

WXYZ2RY=WXYZR3X*WXYZZ–WXYZR3Z*WXYZX

WXYZ2RZ=WXYZR3Y*WXYZX–WXYZR3X*WXYZY
```

In theory:

$$\{2\vec{\Omega}_{xyz} \times \dot{\vec{r}}_{3/2}\} =$$

$$2{*}\{\omega_y{*}(r_{3/2}y{*}\omega_{2x} - r_{3/2}x{*}\omega_{2y}) -$$
$$\omega_z{*}(r_{3/2}x{*}\omega_{2z} - r_{3/2}z{*}\omega_{2x})\}\hat{i}$$
$$2{*}\{\omega_z{*}(r_{3/2}z{*}\omega_{2y} - r_{3/2}y{*}\omega_{2z}) -$$
$$\omega_x{*}(r_{3/2}y{*}\omega_{2x} - r_{3/2}x{*}\omega_{2y})\}\hat{j}$$
$$2{*}\{\omega_x(r_{3/2}x{*}\omega_{2z} - r_{3/2}z{*}\omega_{2x}) -$$
$$\omega_y{*}(r_{3/2}z{*}\omega_{2y} - r_{3/2}y{*}\omega_{2z})\}\hat{k} \qquad (4.139)$$

In the program:

```
W2XYZRX=2*(WXYZY*W2R32Z–WXYZZ*W2R32Y)

W2XYZRY=2*(WXYZZ*W2R32X–WXYZX*W2R32Z)

W2XYZRZ=2*(WXYZX*W2R32Y–WXYZY*W2R32X)
```

In theory:

$$\{\vec{\omega}_2 \times \vec{\omega}_2 \times \vec{r}_{3/2}\} =$$
$$\{\omega_{2y}(r_{3/2}y^*\omega_{2x} - r_{3/2}x^*\omega_{2y}) -$$
$$\omega_{2z}(r_{3/2}x^*\omega_{2z} - r_{3/2}z^*\omega_{2x})\}\hat{i}$$
$$\{\omega_{2z}(r_{3/2}z^*\omega_{2y} - r_{3/2}y^*\omega_{2z}) -$$
$$\omega_{2x}(r_{3/2}y^*\omega_{2x} - r_{3/2}x^*\omega_{2y})\}\hat{j}$$
$$\{\omega_{2x}(r_{3/2}x^*\omega_{2z} - r_{3/2}z^*\omega_{2x}) -$$
$$\omega_{2y}(r_{3/2}z^*\omega_{2y} - r_{3/2}y^*\omega_{2z})\}\hat{k} \tag{4.140}$$

In the program:

```
W2W2R3X=W2R32Z*OMEGA(2,3)–W2R32Y*OMEGA(3,3)

W2W2R3Y=W2R32X*OMEGA(3,3)–W2R32Z*OMEGA(1,3)

W2W2R3Z=W2R32Y*OMEGA(1,3)–W2R32X*OMEGA(2,3)
```

In theory:

$$\{\vec{\alpha}_2 \times \vec{r}_{2/1}\} =$$
$$\{r_{3/2}z^*\alpha_{2y} - r_{3/2}y^*\alpha_{2z}\}\hat{i} +$$
$$\{r_{3/2}x^*\alpha_{2z} - r_{3/2}z^*\alpha_{2x}\}\hat{j} +$$
$$\{r_{3/2}y^*\alpha_{2x} - r_{3/2}x^*\alpha_{2y}\}\hat{k} \tag{4.141}$$

```
A2R32X=R32Z*ALPHA(2,3)–R32Y*ALPHA(3,3)

A2R32Y=R32X*ALPHA(3,3)–R32Z*ALPHA(1,3)

A2R32Z=R32Y*ALPHA(1,3)–R32X*ALPHA(2,3)
```

The acceleration of Segment #2 at a given location was then the sum of the cross–products. The acceleration was stored in the array AC2. This array was declared in the same manner as that of VE1.

```
AC2(1,I)=AC1(1,IP2)+AXYZR2X+WXYZ2RX+W2XYZRX+A2R21X+W2W2R2X

AC2(2,I)=AC1(2,IP2)+AXYZR2Y+WXYZ2RY+W2XYZRY+A2R21Y+W2W2R2Y

AC2(3,I)=AC1(3,IP2)+AXYZR2Z+WXYZ2RZ+W2XYZRZ+A2R21Z+W2W2R2Z.
```

<u>4.65 Determine the New Angles Based on the New Time Step</u>. The angles of Segments #1 and #2 have been altered based on the forces in the actuators, the angular velocities and the angular accelerations. To calculate the degree of change, the routine **DADJANG**, dynamic model, adjust angles, was called. The routine determined the differential change of the angles due to the actuator forces and also the new angles based on the differential change, time step, acceleration and velocity of that segment. The length of the actuators and the differentials change in lengths were determined in the same manner as in the static model. For Segment #1, the equations in theory were:

$$\phi_1 = \phi_1 + \delta\phi_1 + \Omega_1 * \delta t + 0.5 * \forall_1 * \delta t^2. \tag{4.142}$$

$$\Omega_1 = \omega_{1z} \tag{4.143}$$

$$\forall_1 = \alpha_{1z} \tag{4.144}$$

In the program, the equations were:

```
ANG1T=ANG1CNT+DPHI1+OMEGA(3,2)*DT+0.5*ALPHA(3,2)*DT**2
```

Similarly, for Segment #2,

$$\phi_2 = \phi_2 + \delta\phi_2 + \Omega_2 * \delta t + 0.5 * \forall_2 * \delta t^2. \tag{4.145}$$

$$\Omega_2 = \omega_{2z} \tag{4.146}$$

$$\forall_2 = \alpha_{2z} \tag{4.147}$$

In the program, the equations were:

```
ANG2T=ANG2CNT+DPHI2+OMEGA(3,3)*DT+0.5*ALPHA(3,3)*DT**2
```

If the length of the actuators or the new angles were not within the limits of the system, then an error code was sent back to the routine **DYNAMIC**. This routine was very similar to the two routines **SANGLC** and **SADJANG** in the static model.

<u>4.66 Determine the Dynamic Reactions.</u> The dynamic reactions were determined based on the dynamic reactions equations of Bannoura. The routine **DYNREAC**, dynamic reactions, followed the same general format as DYNINER and DSTREAC. The difference was only in the equations. The magnitudes of the reactions were stored in the arrays RD for the forces and MD for the moments. As with all other reaction arrays, these were also declared to be real and

of size (3,4). The reactions were first determined at Pin #3 , then Pin #2, Pin #1, and then Pin #0. The equations for Pin #3 in theory are:

$$R^d_{3x} = R^s_{4x} + Q^d_{g3x} \tag{4.148}$$

$$R^d_{3y} = -W_3 + R^s_{4y} + Q^d_{g3y} \tag{4.149}$$

$$R^d_{3z} = R^s_{4z} + Q^d_{g3z} \tag{4.150}$$

$$M^d_{3x} = P_{34}\sin\phi_3 * R^s_{4z} + M^s_{4x} + P_{3g3}\sin\phi_3 * Q^d_{g3z} + T^d_{g3x} \tag{4.151}$$

$$M^d_{3y} = -P_{34}\cos\phi_3 * R^s_{4z} + M^s_{4y} - P_{3g3}\cos\phi_3 * Q^d_{g3z} + T^d_{g3y} \tag{4.152}$$

$$M^d_{3z} = P_{34}\cos\phi_3 * R^s_{4y} - P_{34}\sin\phi_3 * R^s_{4x} - P_{3g3}\cos\phi_3 * W_3 + M^s_{4z} + P_{3g3}\cos\phi_3 * Q^d_{g3y} - P_{3g3}\sin\phi_3 * Q^d_{g3x} + T^d_{g3z} \tag{4.153}$$

The programlines were:

```
RD(1,4)=RS(1,5)+QD(1,4)

RD(2,4)=-W3+RS(2,5)+QD(2,4)

RD(3,4)=RS(3,5)+QD(3,4)

MD(1,4)=(P34*RS(3,5)+P3G3*QD(3,4))*SI3+MS(1,5)+TD(1,4)

MD(2,4)=-(P34*RS(3,5)+P3G3*QD(3,4))*CO3+MS(2,5)+TD(2,4)

M3ZT1=(P34*RS(2,5)-P3G3*W3+P3G3*QD(2,4))*CO3

M3ZT2=(P34*RS(1,5)+P3G3*QD(1,4))*SI3

MD(3,4)=M3ZT1-M3ZT2+MS(3,5)+TD(3,4)
```

The theoretical equations for Pin #2 were:

$$R^d_{2x} = -F^s_{de}\cos\phi_5 + R^s_{3x} + Q^d_{g2x} - F^d_{de}\cos\phi_5 + R^d_{3x} \tag{4.154}$$

$$R^d_{2y} = -W_2 + R^s_{3y} - F^s_{de}\sin\phi_5 + Q^d_{g2y} - F^d_{de}\sin\phi_5 + R^d_{3y} \tag{4.155}$$

$$R^d_{2z} = R^s_{3z} + Q^d_{g2z} + R^d_{3z} \tag{4.156}$$

$$M^d_{2x} = P_{23}\sin\phi_2 * R^s_{3z} + M^s_{3x} + P_{2g2}\sin\phi_2 * Q^d_{g2z} + P_{23}\sin\phi_2 * R^d_{3z} + T^d_{g2x} + M^d_{3x} \tag{4.157}$$

$$M^d_{2y} = -P_{23}\cos\phi_2 * R^s_{3z} + M^s_{3y} + P_{2g2}\cos\phi_2 * Q^d_{g2z} - P_{23}\cos\phi_2 * R^d_{3z} + T^d_{g2y} + M^d_{3y} \quad (4.158)$$

$$M^d_{2z} = 0.0 \quad (4.159)$$

The program lines were:

```
RD(1,3)=-(FDES+FDED)*CO5 +RS(1,4)+QD(1,3)+RD(1,4)

RD(2,3)=-(FDES+FDED)*SI5-W2+RS(2,4)+QD(2,3)+RD(2,4)

RD(3,3)=RS(3,4)+QD(3,3)+RD(3,4)

M2XT1=(P23*RS(3,4)+P2G2*QD(3,3)+P23*RD(3,4))*SI2

MD(1,3)=M2XT1+MS(1,4)+TD(1,3)+MD(1,4)

M2YT1=(-P23*RS(3,4)+P2G2*QD(3,3)-P23*RD(3,4))*CO2

MD(2,3)=M2YT1+MS(2,4)+TD(2,3)+MD(2,4)

MD(3,3)=0.0
```

The theoretical equations for Pin #1 were:

$$R^d_{1x} = F^s_{de}\cos\phi_5 - F^s_{ab}\cos\phi_4 + R^s_{2x} + F^d_{de}\cos\phi_6 + Q^d_{g1x} - F^d_{ab}\cos\phi_4 + R^d_{2x} \quad (4.160)$$

$$R^d_{1y} = F^s_{de}\sin\phi_5 - W_1 - F^s_{ab}\sin\phi_4 + R^s_{2y} + F^d_{de}\sin\phi_6 + Q^d_{g1y} - F^d_{ab}\sin\phi_4 + R^d_{2y} \quad (4.161)$$

$$R^d_{1z} = R^s_{2z} + Q^d_{g1z} + R^d_{2z} \quad (4.162)$$

$$M^d_{1x} = P_{12}\sin\phi_1 * R^s_{2z} + M^s_{2x} + P_{1g1}\sin\phi_1 * Q^d_{g1z} + P_{12}\sin\phi_1 * R^d_{2z} + T^d_{g1x} + M^d_{2x} \quad (4.163)$$

$$M^d_{1y} = -P_{12}\sin\phi_1 * R^s_{2z} + M^s_{2y} - P_{1g1}\cos\phi_1 * Q^d_{g1z} - P_{12}\cos\phi_1 * R^d_{2z} + T^d_{g1y} + M^d_{2y} \quad (4.164)$$

$$M^d_{1z} = 0.0 \quad (4.165)$$

The program lines were:

```
R1XT1=-(FABS+FABD)*CO4+FDES*CO5+FDED*CO6
RD(1,2)=R1XT1+RS(1,3)+RD(1,3)+QD(1,2)
R1YT1=FDES*SI5+FDED*SI6-(FABS+FABD)*SI4-W1
RD(2,2)=R1YT1+RS(2,3)+QD(2,2)+RD(2,3)
RD(3,2)=RS(3,3)+QD(3,2)+RD(3,3)
M1XT1=((RS(3,3)+RD(3,3))*P12+P1G1*QD(3,2))*SI1
MD(1,2)=M1XT1+MS(1,3)+MD(1,3)+TD(1,2)
M1YT1=-((RS(3,3)+RD(3,3))*P12+P1G1*QD(3,2))*CO1
MD(2,2)=M1YT1+MS(2,3)+MD(2,3)+TD(2,2)
MD(3,2)=0.0
```

The theoretical equations for Pin #0 were:

$$R^d_{0x}=R^s_{4x}+Q^d_{g0x}+Q^d_{g1x}+Q^d_{g2x}+Q^d_{g3x} \tag{4.166}$$

$$R^d_{0y}=-W_0-W_1-W_2-W_3+R^s_{4y}+Q^d_{g0y}+Q^d_{g1y}+Q^d_{g2y}+Q^d_{g3y} \tag{4.167}$$

$$R^d_{0z}=R^s_{4z}+Q^d_{g0z}+Q^d_{g1z}+Q^d_{g2z}+Q^d_{g3z} \tag{4.168}$$

$$\begin{aligned}M^d_{0x}=&(P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{34}\sin\phi_3)R^s_{4z}+M^s_{4x}+P_{0g0}*Q^d_{g0x}+\\&(P_{01}+P_{1g1}\sin\phi_1)Q^d_{g1z}+\\&(P_{01}+P_{12}\sin\phi_1+P_{2g2}\sin\phi_2)Q^d_{g2z}+\\&(P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{3g3}\sin\phi_3)Q^d_{g3z}+\\&T^d_{g0x}+T^d_{g1x}+T^d_{g2x}+T^d_{g3x}\end{aligned} \tag{4.169}$$

$$\begin{aligned}M^d_{0y}=&-(P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{34}\cos\phi_3)R^s_{4z}+M^s_{4y}-\\&P_{1g1}\cos\phi_1 * Q^d_{g1z}-(P_{12}\cos\phi_1+P_{2g2}\cos\phi_2)Q^d_{g2z}-\\&(P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{3g3}\cos\phi_3)Q^d_{g3z}+T^d_{g0y}+T^d_{g1y}+\\&T^d_{g2y}+T^d_{g3y}\end{aligned} \tag{4.170}$$

$$\begin{aligned}M^d_{0z}=&-P_{1g1}\cos\phi_1*W_1-(P_{12}\cos\phi_2+P_{2g2}\cos\phi_2)W_2-\\&(P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{3g3}\cos\phi_3)W_3+\end{aligned}$$

$$\begin{aligned}
&(P_{12}\cos\phi_1+P_{12}\cos\phi_2+P_{34}\cos\phi_3)R^s_{4y}- \\
&(P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{34}\sin\phi_3)R^s_{4x}+M^s_{4z}- \\
&P_{0z}{*}Q^d_{g0x}+P_{1g1}\cos\phi_1{*}Q^d_{g1y}-(P_{01}+P_{1g1}\sin\phi_1)Q^d_{g1x}+ \\
&(P_{12}\cos\phi_1+P_{2g2}\cos\phi_2)Q^d_{g2y}-(P_{01}+P_{12}\sin\phi_1+P_{2g2}\sin\phi_2)Q^d_{g2x}+ \\
&(P_{12}\cos\phi_1+P_{23}\cos\phi_2+P_{3g3}\cos\phi_3)Q^d_{g3y}- \\
&(P_{01}+P_{12}\sin\phi_1+P_{23}\sin\phi_2+P_{3g3}\sin\phi_3)Q^d_{g3x}+ \\
&T^d_{g0z}+T^d_{g1z}+T^d_{g2z}+T^d_{g3z}
\end{aligned} \tag{4.171}$$

The program lines were:

```
RD(1,1)=RS(1,5)+QD(1,1)+QD(1,2)+QD(1,3)+QD(1,4)

RD(2,1)=-(W0+W1+W2+W3)+RS(2,5)+QD(2,1)+QD(2,2)+QD(2,3)+QD(2,4)

RD(3,1)=RS(3,5)+QD(3,1)+QD(3,2)+QD(3,3)+QD(3,4)

M0XT1=(P01+P12*SI1+P23*SI2+P34*SI3)*RS(3,5)

M0XT2=(P01+P1G1*SI1)*QD(3,2)

M0XT3=(P01+P12*SI1+P2G2*SI2)*QD(3,3)

M0XT4=(P01+P12*SI1+P23*SI2+P3G3*SI3)*QD(3,4)

M0XT5=TD(1,1)+TD(1,2)+TD(1,3)+TD(1,4)

M0XT6=P0G0*QD(1,1)+MS(1,5)

MD(1,1)=M0XT1+M0XT2+M0XT3+M0XT4+M0XT5+M0XT6

M0YT1=-(P12*CO1+P23*CO2+P34*CO3)*RS(3,5)

M0YT2=-(P12*CO1+P2G2*CO2)*QD(3,3)

M0YT3=-(P12*CO1+P23*CO2+P3G3*CO3)*QD(3,4)

M0YT4=TD(2,1)+TD(2,2)+TD(2,3)+TD(2,4)

M0YT5=MS(2,5)+P1G1*QD(3,2)
```

```
MD(2,1)=M0YT1+M0YT2+M0YT3+M0YT4+M0YT5

M0ZT1=-P1G1*CO1*W1-(P12+P2G2)*CO2*W2

M0ZT2=-(P12*CO1+P23Z*CO2+P3G3*CO3)*W3

M0ZT3=(P12*CO1+P23*CO2+P34*CO3)*RS(2,5)

M0ZT4=-(P01+P12*SI1+P23*SI2+P34*SI3)*RS(1,5)

M0ZT5=P0G0*QD(1,1)

M0ZT6=P1G1*CO1*QD(2,2)-(P01+P1G1*SI1)*QD(1,2)

M0ZT7=(P12*CO1+P2G2*CO2)*QD(2,3)

M0ZT8=-(P01+P12*SI1+P2G2*SI2)*QD(1,3)

M0ZT9=(P12*CO1+P23*CO2+P3G3*CO3)*QD(2,4)

M0ZT10=(P01+P12*SI1+P23*SI2+P3G3*SI3)*QD(1,4)

M0ZT11=MS(3,5)+TD(3,1)+TD(3,2)+TD(3,3)+TD(3,4)

M0ZT12=M0ZT1+M0ZT2+M0ZT3+M0ZT4+M0ZT5+M0ZT6

M0ZT13=M0ZT7+M0ZT8+M0ZT9+M0ZT10+M0ZT11

MD(3,1)=M0ZT12+M0ZT13
```

4.67 Determine the Deformation. The final caluclations are to determine the deformation of Segments #1 and #2. The routines **DDEFLX**, **DDEFLY**, and **DDEFLZ** calculate the deformatin in the x, y, and z directions, respectively. Each of the dynamic deflection routines are based on their static model counterparts, **SDELFX**, **SDEFLY**, and **SDEFLZ**. The differences between each pair of routines lie within the equations. In the static models, the deformation equations were simplified based on only static loading conditions. But, the dynamic model's deformation equations cannot be simplified.

The equations of theory for axial extension in the x direction and torsional twist for Segment #1 were:

$$u_{x1} = \frac{Q_{g1x}\cos\phi_1 + Q_{g1y}\sin\phi_1}{L_1 A_1 E_1}\left(L_1 x_1 - \frac{x_1}{2}\right) +$$

$$\frac{R^d_{2x}\cos\phi_1+R^d_{2y}\sin\phi_1}{L_1A_1E_1}(x_1) \tag{4.172}$$

$$\theta_{x1}=\frac{T_{g1x}\cos\phi_1+T_{g1y}\sin\phi_1}{L_1J_{1x}G_1}(L_1x_1-\frac{x_1^2}{2})+$$

$$\frac{M^d_{2x}\cos\phi_1+M^d_{2y}\sin\phi_1}{L_1J_{1x}G_1}(x_1) \tag{4.173}$$

The program lines representing the equations were:

```
U1XT1=(QD(1,2)*CO1+QD(2,2)*SI1)*(P1(1,1)*XI–0.5*XI)

U1XT2=(RD(1,3)*CO1+RD(2,3)*SI1)*XI

U1XT3=(P1(1,1)*S1A*S1E)

U1X=(U1XT1+U1XT2)/U1XT3

TH1XT1=(TD(1,2)*CO1+TD(2,2)*SI1)*(P1(1,1)*XI–0.5*XI2)

TH1XT2=(MD(1,3)*CO1+MD(2,3)*SI1)*XI

TH1XT3=(P1(1,1)*S1JO*S1G)

TH1X=(TH1XT1+TH1XT2)/(TH1XT3*57.3)
```

For Segment #2, the equations of theory for axial extension and torsional twist were:

$$u_{x2}=\frac{Q_{g2x}\cos\phi_2+Q_{g2y}\sin\phi_2}{L_2A_2E_2}(L_2x_2-\frac{x_2}{2})+$$

$$\frac{R^d_{3x}\cos\phi_2+R^d_{3y}\sin\phi_2}{L_2A_2E_2}(x_2) \tag{4.174}$$

$$\theta_{x2}=\frac{T_{g2x}\cos\phi_2+T_{g2y}\sin\phi_2}{L_2J_{2x}G_2}(L_2x_2-\frac{x_2}{2})+$$

$$\frac{M^d_{3x}\cos\phi_2+M^d_{3y}\sin\phi_2}{L_2J_{2x}G_2}(x_2) \tag{4.175}$$

The program lines were:

```
U2XT1=(QD(1,3)*CO2+QD(2,3)*SI2)*(P2(1,1)*XI—0.5*XI)
U2XT2=(RD(1,4)*CO2+RD(2,4)*SI2)*XI
U2XT3=(P2(1,1)*S2A*S2E)
U2X=((U2XT1+U2XT2)/U2XT3)
TH2XT1=(TD(1,3)*CO2+TD(2,3)*SI2)*(P2(1,1)*XI—0.5*XI2)
TH2XT2=(MD(1,4)*CO2+MD(2,4)*SI2)*XI
TH2XT3=(P2(1,1)*S2JO*S2G)
TH2X=(TH2XT1+TH2XT2)/(TH2XT3*57.3)
```

The equations of theory for deflection in the y direction, V_i and the slope of the deflection were:

$$V_i = \frac{q_{iyi}}{24E_iI_{zi}}\left[x_i^4 - 4L_ix_i^3 + 6L_i^2x_i^2\right] + \frac{q^o_{iyi}}{120E_iI_{zi}L_i}\left[2x_i^5 - 5L_ix_i^4 + 12L_i^3x_i^2\right] + \frac{R^l_{(i+1)yi}}{6E_iI_{zi}}(3L_ix_i^2 - x_i^3) + \frac{M^d_{(i+1)zi}}{2E_iI_{zi}}x_i^2 \tag{4.176}$$

$$\theta_{zi} = \frac{q_{iyi}}{24E_iI_{zi}}\left[4x_i^3 - 12L_ix_i^2 + 12L_ix_i\right] + \frac{q^o_{iyi}}{20E_iI_{zi}L_i}\left[10x_i^4 - 20L_ix_i^3 + 12L_i^3x_i\right] + \frac{R^l_{(i+1)yi}}{6E_iI_{zi}}(6L_ix_i - 3x_i^2) + \frac{M^d_{(i+1)zi}}{E_iI_{zi}}x_i \tag{4.177}$$

The equations for $R^l_{(i+1)}$, q_{iyi}, q^o_{iyi} in theory were:

$$q^0_{1y1} = \frac{6T_{g1z}}{L_1^2} \tag{4.178}$$

$$q^0_{2y2} = \frac{6T_{g2z}}{L_2^2} \quad (4.179)$$

$$q_{1y1} = \frac{Q_{g1y}\cos\phi_1 - Q_{g1x}\sin\phi_1}{L_1} - \frac{W_1}{L_1}\cos\phi_1 \quad (4.180)$$

$$q_{2y2} = \frac{Q_{g2y}\cos\phi_2 - Q_{g2x}\sin\phi_2}{L_2} - \frac{W_2}{L_2}\cos\phi_2 \quad (4.181)$$

The program lines were:

```
QOD1Y=(6*TD(3,2))/S1L2

QOD2Y=(6*TD(3,3))/S2L2

QD1Y=(QD(2,2)*CO1–QD(1,2)*SI1–W1*CO1)/S1LENG

QD2Y=(QD(2,3)*CO2–QD(1,3)*SI2–W2*CO2)/S2LENG

RL1=RD(2,3)*CO1–RD(1,3)*SI1

RL2=RD(2,4)*CO2–RD(1,4)*SI2
```

The program lines representing the equations for the deflection and slope in the y direction for Segment #1 were:

```
V1T1=QD1Y/(24*S1E*S1IZ)

V1T2=XI4–4*P1(1,1)*XI3+6*S1L2*XI2

V1T3=QOD1Y/(120*S1E*S1IZ*P1(1,1))

V1T4=2*XI5–5*P1(1,1)*XI4+12*S1L3*XI2

V1T5=RL1/(6*S1E*S1IZ)

V1T6=3*P1(1,1)*XI2–XI3

V1T7=(MD(3,3)*XI2)/(2*S1E*S1IZ)

V1=V1T1*V1T2+V1T3*V1T4+V1T5*V1T6+V1T7

TH1ZT1=QD1Y/(24*S1E*S1IZ)

TH1ZT2=4*XI3–12*P1(1,1)*XI2+12*P1(1,1)*XI

TH1ZT3=QOD1Y/(120*S1E*S1IZ*P1(1,1))

TH1ZT4=10*XI4–20*P1(1,1)*XI3+12*S1L3*XI
```

```
TH1ZT5=(RL1*(6*p1(1,1)*XI–3*XI2))/(6*S1E*S1IZ)

TH1ZT6=(MD(3,3)*XI)/(S1E*S1IZ)

TH1Z=TH1ZT1*TH1ZT2+TH1ZT3*TH1ZT4+TH1ZT5+TH1ZT6
```

The program line representing equations for the deflection and slope in the y direction for Segment #2 were:

```
V2T1=QD2Y/(24*S2E*S2IZ)

V2T2=XI4–4*P2(1,1)*XI3+6*S2L2*XI

V2T3=QOD2Y/(120*S2E*S2IZ*P2(1,1))

V2T4=2*XI5–5*p2(1,1)*XI4+12*S2L3*XI2

V2T5=RL2/(6*S2E*S2IZ)

V2T6=3*P2(1,1)*X12–XI3

V2T7=(MD(3,4)*XI2)/(2*S2E*S2IZ)

V2=V2T1*V2T2+V2T3*V2T4+V2T5*V2T6+V2T7+P2YD

TH2ZT1=QD2Y/(24*S2E*S2IZ)

TH2ZT2=4*XI3–12*P2(1,1)*XI2+12*P2(1,1)*XI

TH2ZT3=QOD2Y/(120*S2E*S2IZ*P2(1,1))

TH2ZT4=10*XI4–20*P2(1,1)*XI3+12*S2L3*XI

TH2ZT5=(RL2*(6*P2(1,1)*XI–3*XI2))/(6*S2E*S2IZ)

TH2ZT6=(MD(3,4)*XI)/(S2E*S2IZ)

TH2Z=TH2ZT1*TH2ZT2+TH2ZT3*TH2ZT4+TH2ZT5+TH2ZT6
```

The equations of theory for deflection in the z direction, W_i, and the slope of defelction were:

$$W_i = \frac{q_{i\,z\,i}}{24E_i I_{y\,i}} \left[x_i^4 - 4L_i x_i^3 + 6L_i^2 x_i^2\right] +$$

$$\frac{q^o_{i\,z\,i}}{120E_i I_{y\,i} L_i} \left[2x_i^5 - 5L_i x_i^4 + 12L_i^3 x_i^2\right] +$$

$$\frac{R^{d}_{(i+1)zi}}{6E_iI_{yi}}(3L_ix_i^2-x_i^3)-\frac{M^{l}_{(i+1)yi}}{2E_iI_{yi}}x_i^2 \tag{4.182}$$

$$\theta_{yi}=-\frac{q_{izi}}{24E_iI_{yi}}\left[4x_i^3-12L_ix_i^2+12L_i^2x_i\right]+$$

$$\frac{q^{o}_{izi}}{120E_iI_{yi}L_i}\left[10x_i^4-20L_ix_i^3+24L_i^3x_i\right]+$$

$$\frac{R^{d}_{(i+1)zi}}{6E_iI_{yi}}(6L_ix_i-3x_i^2)-\frac{M^{l}_{(i+1)yi}}{E_iI_{yi}}x_i \tag{4.183}$$

The equations for $M^{l}_{(i+1)}$, q_{izi}, q^{o}_{izi} in theory were:

$$q^{0}_{1z1}=\frac{6}{L_1^2}(T_{g1y}\cos\phi_1-T_{g1x}\sin\phi_1) \tag{4.184}$$

$$q^{0}_{2z2}=\frac{6}{L_2^2}(T_{g2y}\cos\phi_2-T_{g2x}\sin\phi_2) \tag{4.185}$$

$$q_{1z1}=\frac{Q_{g1z}}{L_1} \tag{4.186}$$

$$q_{2z2}=\frac{Q_{g2z}}{L_2} \tag{4.187}$$

The program lines were:

```
ML1=MD(2,3)*SI1+MD(1,3)*CO1

ML2=MD(2,4)*SI2+MD(1,4)*CO2

QOD1Z=(6/S1L2)*(TD(2,2)*CO1–TD(1,2)*SI1)

QOD2Z=(6/S2L2)*(TD(2,3)*CO2–TD(1,3)*SI2)

QD1Z=QD(3,2)/S1LENG

QD2Z=QD(3,3)/S2LENG
```

The program lines representing the equations for the deflection and slope in the z direction for Segment #1 were:

```
W1T1=QD1Z/(24*S1E*S1IY)
```

```
W1T2=XI4-4*P1(1,1)*XI3+6*S1L2*XI2

W1T3=QOD1Z/(120*S1E*S1IY*P1(1,1))

W1T4=2*XI5-5*P1(1,1)*XI4+12*S1L3*XI2

W1T5=RD(3,3)/(6*S1E*S1IY)

W1T6=3*P1(1,1)*XI2-XI3

W1T7=(ML1*XI2)/(2*S1E*S1IY)

W1=W1T1*W1T2+W1T3*W1T4+W1T5*W1T6+W1T7

TH1YT1=QD1Z/(24*S1E*S1IY)

TH1YT2=(4*XI3-12*P1(1,1)*XI2+12*S1L2*XI)

TH1YT3=QOD1Y/(120*S1E*S1IY*P1(1,1)

TH1YT4=10*XI4-20*P1(1,1)*XI3+24*S1L3*XI

TH1YT5=RD(3,3)/(6*S1E*S1IY)

TH1YT6=6*P1(1,1)*XI-3*XI2

TH1YT7=(ML1*XI)/(S1E*S1IY)

TH1Y=TH1YT1*TH1YT2+TH1YT3*TH1YT4

        +TH1YT5*TH1YT6+TH1YT7
```

The program lines representing equations for the deflection and slope in the z direction for Segment #2 were:

```
W2T1=QD2Z/(24*S2E*S2IY)

W2T2=XI4-4*P2(1,1)*XI3+6*S2L2*XI2

W2T3=QOD2Z/(120*S2E*S2IY*P2(1,1))

W2T4=2*XI5-5*P2(1,1)*XI4+12*S2L3*XI2

W2T5=RD(3,4)/(6*S2E*S2IY)

W2T6=3*P2(1,1)*XI2-XI3

W2T7=(ML2*XI2)/(2*S2E*S2IY)

W2=W2T1*W2T2+W2T3*W2T4+W2T5*W2T6+W2T7+P2ZD

TH2YT1=QD2Z/(24*S2E*S2IY)
```

```
TH2YT2=4*XI3–12*P2(1,1)*XI2+12*S2L2*XI

TH2YT3=QOD2Y/(120*S2E*S2IY*P2(1,1))

TH2YT4=10*XI4–20*P2(1,1)*XI3+24*S2L3*XI

TH2YT5=RD(3,4)/(6*S2E*S2IY)

TH2YT6=6*P2(1,1)*XI–3*XI2

TH2YT7=(ML2*XI)/(S2E*S2IY)

TH2Y=TH2YT1*TH2YT2+TH2YT3*TH2YT4

        +TH2YT5*TH2YT6+TH2YT7
```

At this point, the deformed coordinates have been stored in the arrays S1D, S2D, S3D, WINGD, P1D, P2D, and PWD.

4.68 Complete the Kinematic Analysis. The next step was to transform the coordinates through the kinematic analysis. This kinematic analysis was completed through the routine **DTFORM**, dynamic model, transformations. Section **4.3 Kinematics** gave a full explanation. The output of this routine were the transformed deformed coordinates which could be used to draw the robot image.

4.69 Draw the Three–Dimensional Robotic Arm Image. The robot image was drawn through the control of the routine DDRROB, dynamic model, draw robot. From this routine, all the routines which drew the links were called. As explained in Seciton 4.4 **Three–Dimensional Graphics**, there were specific routines responsible for drawing each component of the robot.

4.610 Allow User Interactions. Once the user was given the graphics output of the program, the routine DUSR, dynamic model, user interaction, was called to allow analytical information to be extracted at the user's request. This routine was described in Section 4.2 **User Interactions.**

CHAPTER 5
USER DIRECTIONS

The first step in using either of the models is to move to the correct subdirectory. The directory where the models are is

```
\usr\aro\solid .
```

To move to that subdirectory type

```
cd \usr\aro\solid .
```

5.1 Static Model

1. Edit input parameter file in the vi editor by the command:

   ```
   vi robots.dat
   ```

   Follow the vi commands to complete the editing.

2. Initiate the Static Model through the command:

   ```
   ./smodel
   ```

3. When a session is complete, preserve the output file for that session by using the command:

   ```
   copy sout.dat filename
   ```

5.2 Dynamic Model

1. Edit input parameter file in the vi editor by the command:

   ```
   vi robots.dat
   ```

Follow the vi commands to complete the editing.

2. Initiate the Dynamic Model through the command:

   ./dmodel

3. When a session is complete, preserve the output file for that session by using the command:

   copy dout.dat filename

5.3 VI Editor Commands

Arrow Keys — → ↑ ← ↓ — Move the cursor to the desired location

'x' — deletes the character the cursor is positioned over

'a' — adds or appends after the position of the cursor

'i' — inserts after the position of the cursor

'esc' — ends the appending or inserting mode

':x' — saves the changes in the file and ends the session with vi

5.4 User Interaction Capabilities

5.41 Angle Interaction

DIAL #1 — Rotate Base Positively

DIAL #2 — Rotate Base Negatively

DIAL #3 — Rotate Segment #1 Positively

DIAL #4 — Rotate Segment #1 Negatively

DIAL #5 — Rotate Segment #2 Positively

DIAL #6 — Rotate Segment #2 Negatively

DIAL #7 — Rotate Segment #3 Positively

DIAL #8 — Rotate Segment #3 Negatively

Mouse #1 — Rotate World about X—Axis

Mouse #2 — Rotate World about Y—Axis

Mouse #3 — Rotate World about Z—Axis

Switch #1 — Angle of Actuator AB

Switch #5 — Angle of Base

Switch #6 — Angle of Actuator DE

Switch #11 — Angle of Segment #1

Switch #12 — Angle of Wing Segment

Switch #16 — All angles written to an output file

Switch #17 — Angle of Segment #2

Switch #18 — Angle of World View

Switch #23 — Angle of Segment #3

Switch #24 — Differential change in the angles of the actuators

5.42 Deformation

Switch #4 — Segment #1: Deflection and Slope

Switch #9 — Segment #2: Deflection and Slope

Switch #15 — Deflection of Pin #2

Switch #21 — Deflection of Pin #3

Switch #26 — Segment #1: Deflection and Slope written to an output file

Switch #27 — Segment #2: Deflection and Slope written to an output file

5.43 Forces and Moments

Switch #7 — Dynamic Forces and Moments

Switch #2 — Static Forces and Moments

Switch #10 — Static Forces and Moments written to an output file

Switch #13 — Inertial Forces and Torques

Switch #19 — Force of Actuator AB

Switch #25 — Force of Actuator DE

Numeric Pad #0 — Represents Pin #0

Numeric Pad #1 — Represents Pin #1

Numeric Pad #2 — Represents Pin #2

Numeric Pad #3 — Represents Pin #3

Numeric Pad #4 — Represents Pin #4, or Load

5.44 Segment Levels

Switch #29 — Level Number of Segment #1

Switch #30 — Level Number of Segment #2

Switch #31 — Increment Level Number

Switch #32 — Decrement Level Number

5.45 Program Control

'N' — Starts the dynamic analysis on the next pair of actuator forces

'Q' — Stops the model

5.46 Velocities and Accelerations

Switch #3 — Angular Velocities and Accelerations of Segment #1

Switch #8 — Translational Velocity and Acceleration of Segment #1

Switch #14 — Angular Velocity and Acceleration of Segment #2

Switch #20 — Translational Velocity and Acceleration

of Segment #2

Switch #22 — Velocities and Accelerations of Segment #1

written to an output file

Switch #28 — Velocities and Accelerations of Segment #2

written to an output file

5.5 Input Parameter File ROBOTS.DAT

ROBOT PARAMETER FILE — ROBOTS.DAT
ROBOT ARM SPECIFICATIONS

A.	ARM DIMENSIONS (Angles — Degrees, Units — SI)			
1.	Base Dimensions(all meas'ts w.r.t. center bottom of base)			
	a.	Base Diameter		0.36
	b.	Base Height		2.198
	c.	Position of Pin #1 on the Base	(X)	0.0
	d.		(Y)	2.149
	e.	Mass		252.30
	f.	Center of Mass,	(X)	0.0
	g.		(Y)	0.75
	h.	Mass Moment of Inertia about X—Axis		0.221
	i.	Mass Moment of Inertia about Y—Axis		0.44
	j.	Mass Moment of Inertia about Z—Axis		0.221
2.	Segment #1 Dimensions (all measurements w.r.t. pin #1)			
	a.	Thickness		0.038
	b.	Length		1.78
	c.	Position of Pin #1 on Segment 1	(X)	0.10
	d.		(Y)	0.0
	e.	Mass		72.66
	f.	Minimum Angle w.r.t. Horizontal		—30.0
	g.	Maximum Angle w.r.t. Horizontal		30.0
	h.	Center of Mass,	(X)	0.887
	i.		(Y)	0.0021
	j.	Position of Pin #2,	(X)	1.6
	k.		(Y)	0.0
	l.	Number of Divisions		10.0
	m.	Modulus of Elasticity (in G Pa)		192.05
	n.	Modulus of Rigidity (in G Pa)		86.0
	o.	Deflection Multiplier		1.0
	p.	Cross—Sectional Area		0.0087
	q.	Mass Moment of Inertia about X—Axis		0.7166
	r.	Mass Moment of Inertia about Y—Axis		29.43
	s.	Mass Moment of Inertia about Z—Axis		29.21
	t.	Area Moment of Inertia about Y—Axis		1.4E—7
	u.	Area Moment of Inertia about Z—Axis		1.4E—7
	v.	Area Polar Moment of Inertia		2.8E—7
3.	Segment #2 Dimensions (all measurements w.r.t. pin #2)			
	a.	Thickness		0.038
	b.	Length		0.9144
	c.	Position of Pin #2 on Segment 2	(X)	0.10
	d.		(Y)	0.0
	e.	Mass		11.9
	f.	Minimum Angle w.r.t. Segment #1		—90.0
	g.	Maximum Angle w.r.t. Segment #1		7.0
	h.	Center of Mass,	(X)	0.375
	i.		(Y)	0.0
	j.	Position of Pin #3,	(X)	0.821
	k.		(Y)	0.0
	l.	Number of Divisions		10.0
	m.	Modulus of Elasticity (in G Pa)		192.05

n.	Modulus of Rigidity (in G Pa)		86.0
o.	Deflection Multiplier		1.0
p.	Cross—Sectional Area		0.0087
q.	Mass Moment of Inertia about X—Axis		0.0008
r.	Mass Moment of Inertia about Y—Axis		1.681
s.	Mass Moment of Inertia about Z—Axis		1.681
t.	Area Moment of Inertia about Y—Axis		1.4E—7
u.	Area Moment of Inertia about Z—Axis		1.4E—7
v.	Area Polar Moment of Inertia		2.8E—7
4. Wing Segment Dimensions (all measurements w.r.t. pin #2)			
a.	Angle of Wing Segment w.r.t. Segment #2		120.0
b.	Wing Length		0.4191
5. End Effector, Segment #3 (all measurements w.r.t. pin #3)			
a.	Thickness		0.1
b.	Length		0.104
c.	Mass		4.55
d.	Center of Mass,	(X)	0.0
e.		(Y)	0.0
f.	Minimum Angle w.r.t. Segment #2		—90.0
g.	Maximum Angle w.r.t. Segment #2		90.0
h.	Position of Pin #3 on Segment 3	(X)	0.01
i.		(Y)	0.0
6. Object, Segment #4			
a.	Thickness		0.15
b.	Length		0.944
c.	Mass		28.56
d.	Center of Mass,	(X)	0.4572
e.		(Y)	0.0
7. Hydraulic Cylinder #1 Dimensions			
a.	Thickness		0.0762
b.	Maximum Length		1.9144
c.	Minimum Length		0.4572
d.	Position of Pin #A on Base,	(X)	0.178
e.		(Y)	1.464
f.	Position of Pin #B on Segment #1,	(X)	0.333
g.		(Y)	0.0
h.	Mass		1.3
i.	Cross—Sectional Area		0.005
j.	Modulus of Elasticity		200.0
8. Hydraulic Cylinder #2 Dimensions			
a.	Thickness		0.076
b.	Maximum Length		1.829
c.	Minimum Length		0.915
d.	Position of Pin #D on Segment #1,	(X)	0.108
e.		(Y)	0.0
f.	Position of Pin #E on Wing Segment,	(X)	0.3875
g.		(Y)	0.0
h.	Mass		1.2
i.	Cross—Sectional Area		0.005
j.	Modulus of Elasticity		200.0

B. PROGRAM INITIAL VALUES

1. Initial Display Parameters

a.	Angle Increment	5.0
b.	Time Step	0.01
c.	Initial value of Base	0.0
d.	Initial angle of Segment #1	0.0
e.	Initial angle of Segment #2	0.0
f.	Initial angle of Segment #3	0.0

**

All distances are in terms of local ditances measured from the reference or the reference pin of each segment. All angles are in degrees. The suggested unit system is SI.

5.6 Dynamic Input File, DYNIN.DAT

ARO DYNAMIC INPUT FILE

I	FAB (N)	FDE (N)
1	−6841.18	964.92
2	−6841.18	964.92
3	−6841.18	964.92
4	−6841.18	964.92
5	−6841.18	0.0
6	−6841.18	0.0
7	−6841.18	0.0
8	−6841.18	0.0
999		
x123	1234567890.12	1234567890.12

CHAPTER 6
RESULTS

The results of the modeling system were classified in the four areas of research: Three–Dimensional Graphics, Kinematics, User–Interaction, and Deformation Analysis. Results were compiled in both analytical and graphic forms.

6.1 Three–Dimensional Graphics

A three–dimensional image was created on the screen through the Silicon Graphics IRIS 3130 Graphics Workstation. The image, (See Figure 6.1) was based on the dimensions specified in the parameter file, **robots.dat**. As an example of the graphic capabilities, the color of the base was changed to white. This produced the image in Figure 6.2.

6.2 Kinematics

The kinematic analysis resulted in a discrete configuration of the arm. The configuration of the arm would change based on the angles of the links and also the dimensions of each link. Figure 6.3 shows a configuration consisting of the following angles:

$\phi_0 = 0^\circ$ $\qquad$ $\phi_1 = 30^\circ$

$\phi_2 = -60^\circ$ $\qquad$ $\phi_3 = 0^\circ$

$\gamma = 120^\circ$

FIGURE 6.1

THREE DIMENSIONAL COMPUTERIZED IMAGE ON THE IRIS 3130

FIGURE 6.2

WHITE COMPUTERIZED IMAGE ON THE IRIS 3130

FIGURE 6.3

ARM WITH ϕ_0=0, ϕ_1=30, ϕ_2=–60, ϕ_3=0, ϕ_6=120

6.3 User Interaction

The small blue window at the bottom of the screen was a text window. The user could extract information regarding the given configuration of the arm or the results of the analysis of the arm. For example, given the configuration in Figure 6.3, by selecting Switches #5, 11, and 17 the angles for Segments #1 and #2 would be written to the screen. Another mode of user interaction would be changing the the World Perspective by selecting the one or a combination of the Mouse Keys. Figure 6.4 shows a change in World Perspective with the angles written to the text window through selecting Switch #18.

The third example of results of user interaction is through the input parameter file, (See Figure 6.5). In Figure 6.6 the height of the Base was 2.198m and the thickness was 0.36 m.

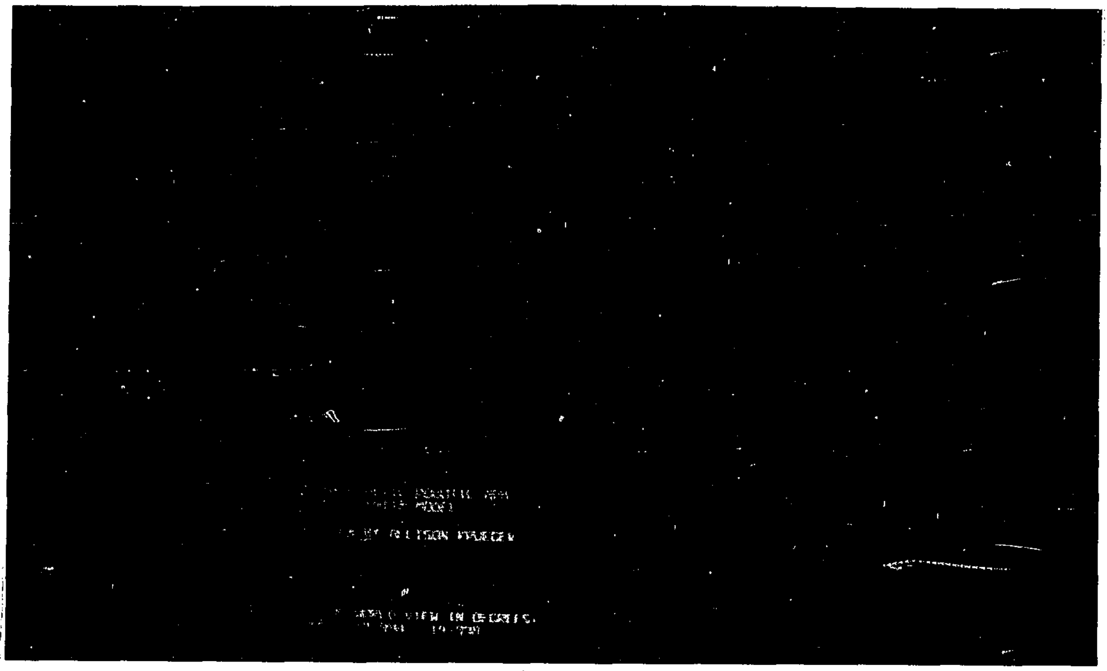
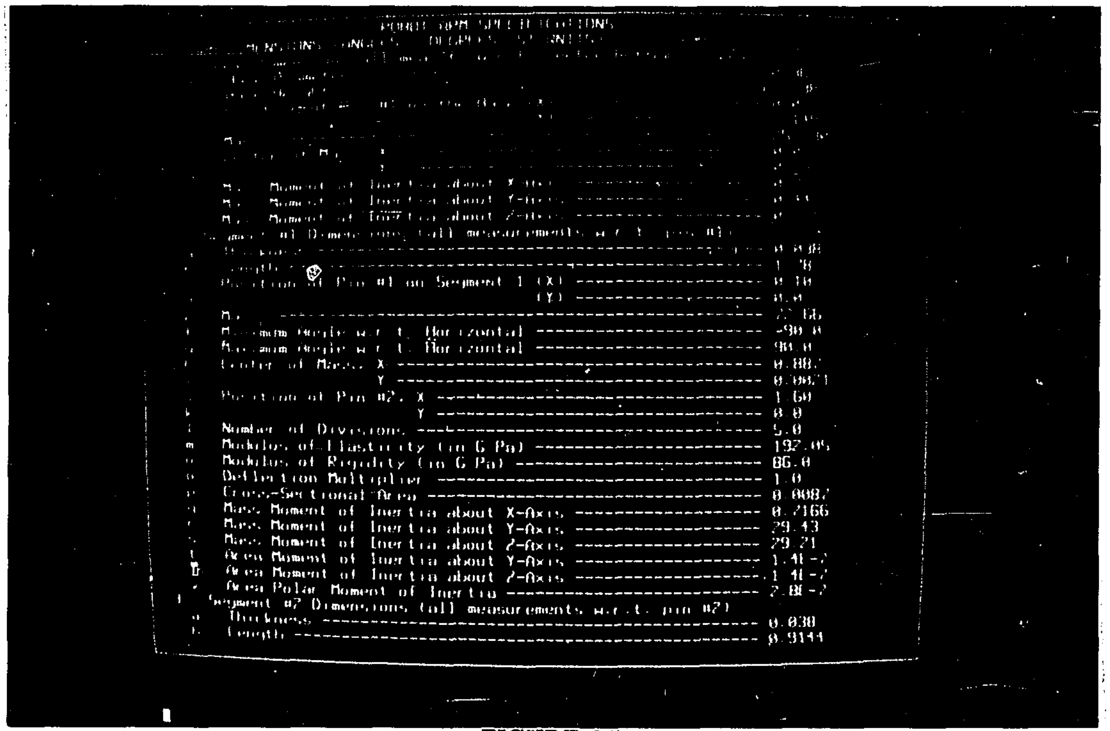


FIGURE 6.4

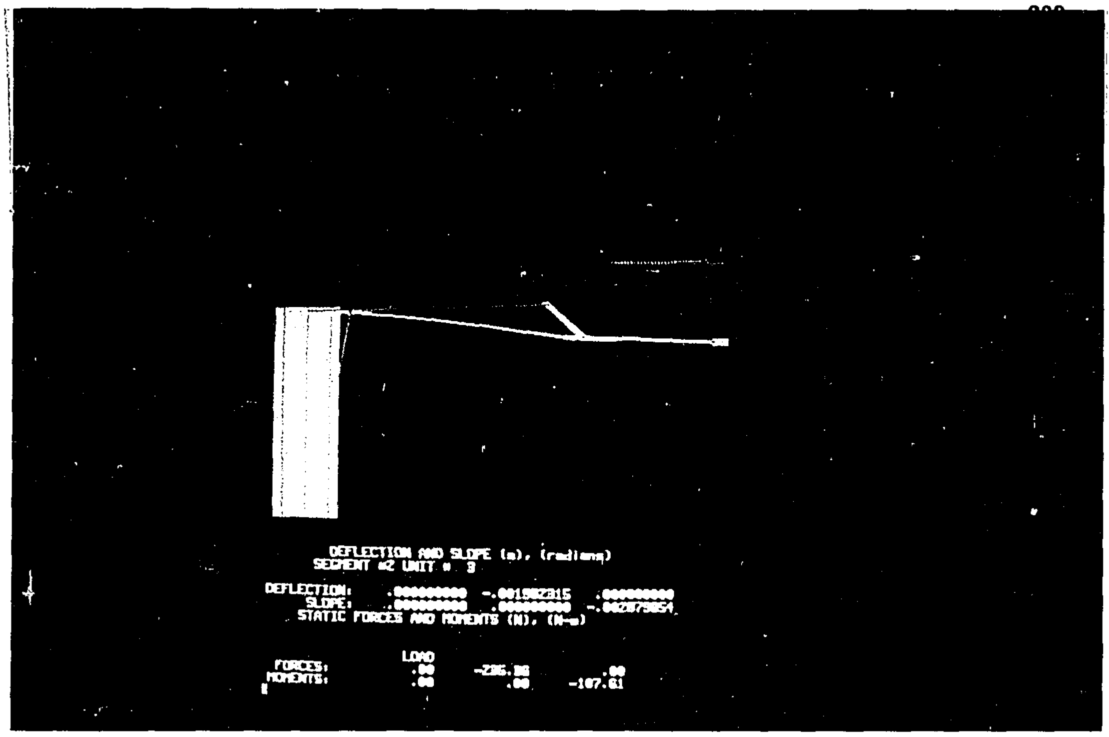
CHANGE OF WORLD PERSPECTIVE



```
                         ROBOT ARM SPECIFICATIONS
        DIMENSIONS [illegible] DEGREES [illegible]
  [illegible]
     M[illegible] Moment of Inertia about X-Axis ------------------------- [illegible]
     M[illegible] Moment of Inertia about Y-Axis ------------------------- [illegible]
     M[illegible] Moment of Inertia about Z-Axis ------------------------- [illegible]
  [illegible] Dimensions (all measurements w.r.t. pin #1)
     Thickness ------------------------------------------------------- [illegible]
     Length ---------------------------------------------------------- 1.[illegible]
     Position of Pin #1 on Segment 1 (X) -------------------------- [illegible]
                                     (Y) -------------------------- [illegible]
     M[illegible] ------------------------------------------------------- [illegible]
     Minimum Angle w.r.t. Horizontal ------------------------------ -90.0
     Maximum Angle w.r.t. Horizontal ------------------------------ 90.0
     Center of Mass, X -------------------------------------------- 0.88[illegible]
                     Y -------------------------------------------- 0.[illegible]
     Position of Pin #2, X ---------------------------------------- 1.60
                         Y ---------------------------------------- 0.0
     Number of Divisions ------------------------------------------ 5.0
     Modulus of Elasticity (in G Pa) ------------------------------ 197.0[illegible]
     Modulus of Rigidity (in G Pa) -------------------------------- 86.0
     Deflection Multiplier ---------------------------------------- 1.0
     Cross-Sectional Area ----------------------------------------- 0.008[illegible]
     Mass Moment of Inertia about X-Axis -------------------------- 0.2166
     Mass Moment of Inertia about Y-Axis -------------------------- 29.43
     Mass Moment of Inertia about Z-Axis -------------------------- 29.21
     Area Moment of Inertia about Y-Axis -------------------------- 1.4E-[illegible]
     Area Moment of Inertia about Z-Axis -------------------------- 1.4E-[illegible]
     Area Polar Moment of Inertia --------------------------------- 2.8E-[illegible]
  Segment #2 Dimensions (all measurements w.r.t. pin #2)
     Thickness ------------------------------------------------------- 0.030
     Length ---------------------------------------------------------- 0.9144
```

FIGURE 6.5

INPUT PARAMETER FILE ROBOTS.DAT



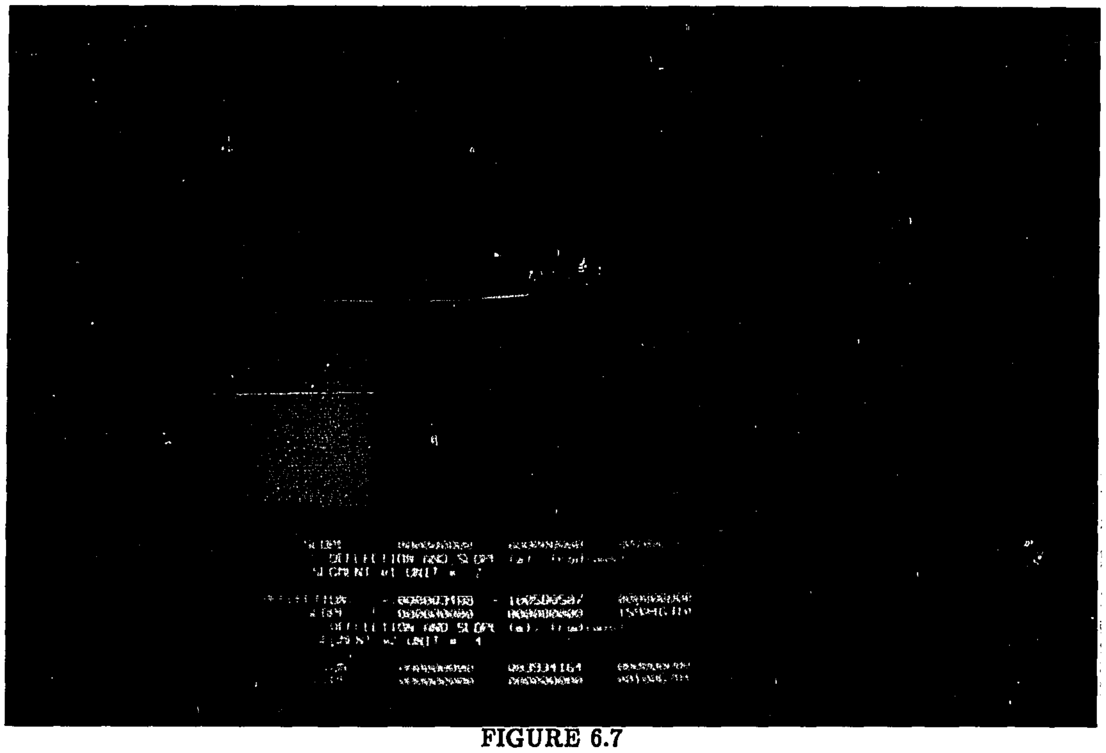
```
      DEFLECTION AND SLOPE (m), (radians)
    SEGMENT #2 UNIT # 9

DEFLECTION:     .000000000  -.001982315  .000000000
     SLOPE:     .000000000   .000000000  -.002879854
   STATIC FORCES AND MOMENTS (N), (N-m)

                   LOAD
  FORCES:           .00     -296.88        .00
 MOMENTS:           .00         .00    -107.81
```

FIGURE 6.6

BASE WITH DIMENSIONS OF HEIGHT=2.198m AND WIDTH=0.36m



```
 SLOPE          [illegible]       [illegible]      [illegible]
      DEFLECTION AND SLOPE (m), (radians)
    SEGMENT #1 UNIT # 7

DEFLECTION     -.000007193   -.10250567       [illegible]
 SLOPE           .000000000   .000000000      [illegible]
      DEFLECTION AND SLOPE (m), (radians)
    SEGMENT #2 UNIT # 4

 [illegible]    [illegible]   [illegible]     [illegible]
 [illegible]    [illegible]   [illegible]     [illegible]
```

FIGURE 6.7

BASE WITH DIMENSIONS OF HEIGHT=1.25m AND WIDTH=0.75m

By changing the height to 1.25m and the thickness to 0.75m a new robotic arm image is obtained. Figure 6.7 will show the new arm.

Analytical information such as the reactions at each of the pins can also be extracted. By selecting Pin # and the numeric keypad number 4, the reactions of the load were written to the screen (see Figure 6.6).

6.4 Deformation Analysis

The static deflection results predicted by the program smodel.f were compared with the measured deflections from experimental conditions of the robot arm shown in Figure 6.1. The configuration of the arm is based on the following angles:

$\phi_0 = 0^\circ$ $\qquad\qquad \phi_1 = 0^\circ$

$\phi_2 = 0^\circ$ $\qquad\qquad \phi_3 = 0^\circ$

$\gamma = 120^\circ$

The experiment was conducted by placing two micrometers under the beams of Segments #1 and #2 as shown in Figure 6.8. To similate the load, small steel plates of the dimension (6x4x3/4)in or (0.15x0.10x0.02)m weighing an average of 2.41 kg were placed in the bucket. A zero load was considered to be based on the mass of Segment #3, the rope, and the bucket. The load was then increased and decreased by adding or removing the steel plates. Each plate was numbered so that the exact mass was calculated (See Table 6.1).

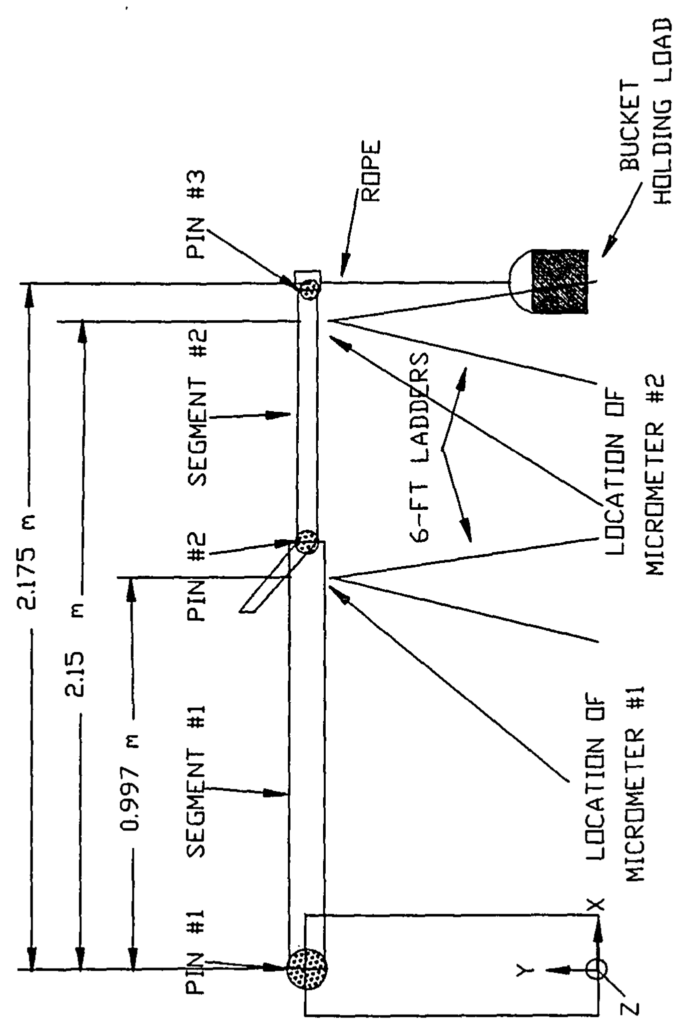


FIGURE 6.8
EXPERIMENTAL SETUP FOR MEASURING STATIC DEFLECTION

TABLE 6.1 MASS OF OBJECTS FOR EXPERIMENTAL MEASUREMENTS OF STATIC DEFLECTION

ITEM	MASS (KG)	LOAD (N)
SEG'T #3	4.55	44.620
ROPE	0.63	6.178
BUCKET	0.887	8.698
PLATE #10	2.254	22.104
PLATE #9	2.271	22.271
PLATE #8	2.275	22.310
PLATE #7	2.241	21.977
PLATE #6	2.256	22.124
PLATE #5	2.194	21.516
PLATE #4	2.258	22.143
PLATE #3	2.238	21.947
PLATE #2	2.268	22.241
PLATE #1	2.236	21.928

The first micrometer gave the deflection measurement associated with Segment #1 located just to the left of Pin #2. Micrometer #2 measured the deflection associated with Segment #2 located just to the left of Pin #3, (see Figure 6.8). The measurements taken are summarized in Table 6.2.

TABLE 6.2 EXPERIMENTAL MEASUREMENTS OF STATIC DEFLECTIONS

DEFLECTION SEGMENT #1 (mm)	DEFLECTION SEGMENT #2 (mm)	TIME (SEC)	LOAD (N)
** Constant Loading Conditions (Drift)			
1.44	0.56	0	59.5
1.55	0.71	30	59.5
1.63	0.84	60	59.5
** Increase in Loading			
2.59	2.87	108	81.6
3.46	4.87	131	103.9
4.32	6.83	150	126.2
5.21	8.78	165	148.2
6.07	10.74	181	170.3
6.91	12.62	195	191.8
7.80	14.55	210	213.9
8.68	16.56	225	235.9
9.54	18.54	240	258.1
10.44	20.55	259	280.1

** Constant Loading Conditions (Drift)			
10.54	20.70	270	280.1
10.67	21.00	300	280.1
10.81	21.25	330	280.1
** Decrease in Loading			
10.61	19.74	345	258.1
9.45	18.10	360	235.9
8.75	16.50	380	213.9
8.02	14.78	400	191.8
7.32	13.17	415	170.3
6.57	11.46	435	148.2
5.80	9.65	450	126.2
5.06	7.92	480	103.9
4.29	6.14	500	81.6
3.54	4.34	525	59.5
** Constant Loading Conditions (Drift)			
3.56	4.39	540	59.5
3.63	4.51	570	59.5
3.71	4.62	600	59.5
3.80	4.75	630	59.5

The calculations from the static model were tabulated in Table 6.3:

TABLE 6.3 STATIC DEFLECTIONS PREDICTED FROM STATIC MODEL SMODEL.F

LOAD (N)	DEFLECTION SEGMENT #1 (mm)	DEFLECTION SEGMENT #2 (mm)
59.5	12.1	17.7
81.6	13.8	21.8
103.9	15.6	26.0
126.2	17.3	30.2
148.2	19.0	34.3
170.3	20.7	38.5
191.8	22.4	42.5
213.9	24.1	46.6
235.9	25.8	50.8
258.1	27.5	54.9
280.1	29.2	59.1

Relative deflections for both the experimental data and the calculated data were then compared in Table 6.4.

TABLE 6.4 CHANGES IS EXPERIMENTAL DEFLECTIONS VS. CHANGES IN CALCULATED DEFLECTIONS

	EXPERIMENTAL		CALCULATED	
CHANGE LOAD	DEFLECTION SEGMENT #1 (mm)	DEFLECTION SEGMENT #2 (mm)	DEFLECTION SEGMENT #1 (mm)	DEFLECTION SEGMENT #2 (mm)
0.0	0.0	0.0	0.0	0.0
22.0	0.86	2.03	1.7	4.1
22.3	0.96	2.00	1.7	4.2
22.3	0.87	1.96	1.7	4.2
22.0	0.87	1.95	1.7	4.1
21.5	0.86	1.97	1.7	4.2
21.5	0.84	1.87	1.7	4.0
22.1	0.89	1.94	1.7	4.1
22.0	0.89	2.01	1.7	4.2
22.2	0.86	1.98	1.7	4.1
22.0	0.90	2.01	1.7	4.2

The relative deflections for Segment#2 that were measured from the experiment are very close to those calculated from the model. However, the experimentally measured deflections for Segment #1 do not coincide with those calculated from the model. The slight deviation in the two sets can be explained through four points:

(1) Drift
(2) Change of reactions at end of cantilever beam
(3) Change in lengths in cantilever beam
(4) Percent of beam which is considered to be elastic

The drift of the beam is associated with due to the load on the beam as a function of time. As can be see in Table 6.2, there were three periods of constant load where drift was recorded. The change in reactions is based on the fact that in the experiment the two segments, Segment #1 and Segment #2 were considered to be a cantilever beam, (See Figure 6.9) with one set of reactions effecting the entire beam. But, in the model, the equations

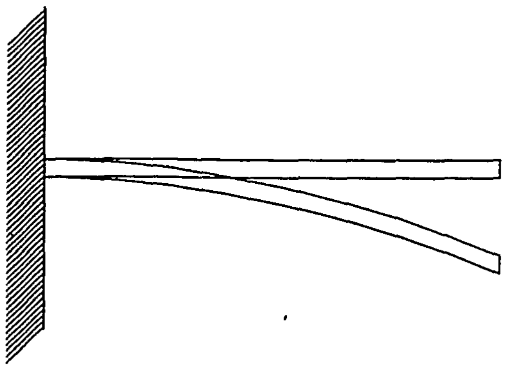


FIGURE 6.9

SEGMENTS #1 AND #2 AS ONE CANTILEVER BEAM

of deflections were based on each segment being a seperate cantilever beam with different reactions at the end of the beam, (See Figure 6.10).

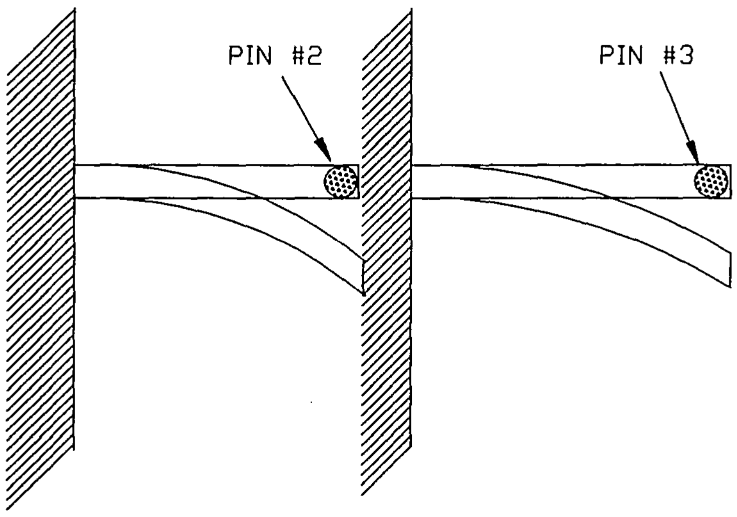


FIGURE 6.10

SEGMENTS #1 AND #2 AS TWO CANTILEVER BEAMS

This created a change in the measured and calculated values based on the change in reactions. The difference in configuration also changed the lengths of the cantilever beams as well as the location of the calculated deflection. This in turn also altered the measured and calculated values.

The percentage of the beams which were considered to be elastic also effected the deflections. The experimental beam contained a lower percentage of elasticity than the calculated value.

For a graphical interpretation of severe deflection see figure 6.11 inwhich the modulus of elasticity was considerably less than that of either steel or aluminum.

FIGURE 6.11

ROBOTIC ARM WITH SEVERE DEFLECTION

6.5 Dynamic Analysis

As an example of the output from the dynamic robot simulation program, DMODEL, the file DYNIN.DAT was supplied with actuator forces adequate to maintain the robot near static equilibrium. In this configuration, link #1 and link #2 were both horizontal. The time

step was set to 0.01 seconds. At time t = 0.04 seconds, the force in hydraulic actuator DE was dropped to zero causing the second link to fall. The program calculated the resulting angular acceleration of both links, computed the deflections, velocities, and resulting positions, and displayed all of this information on the computer screen through the robot
model.

In the example executed on the IRIS workstation, the elastic robot described in table 5.5 was used with an applied load of 250 N. Data acquired during this simulation was as follows:

During static equilibrium, t = 0.03 seconds:

Applied force in actuator AB (N)	–6841.2
Applied force in actuator DE (N)	964.9
Resulting dynamic moment about pin #1 (N–m)	–546.0
Resulting dynamic moment about pin #2 (N–m)	–30.7
Angular velocity of link #1 (rad/s)	–0.76
Angular velocity of link #2 (rad/s)	–0.79
Angular acceleration of link #1 (rad/s^2)	–18.69
Angular acceleration of link #2 (rad/s^2)	–18.28
Total vertical deflection of link #1 (mm)	–194.3
Total vertical deflection of link #2 (mm)	–6.52

After the force in actuator DE becomes zero, t = 0.04 seconds:

Applied force in actuator AB (N)	–6841.2
Applied force in actuator DE (N)	0.0
Resulting dynamic moment about pin #1 (N–m)	–539.5
Resulting dynamic moment about pin #2 (N–m)	–392.2
Angular velocity of link #1 (rad/s)	–0.94
Angular velocity of link #2 (rad/s)	–3.12
Angular acceleration of link #1 (rad/s^2)	–18.47
Angular acceleration of link #2 (rad/s^2)	–233.35
Total vertical deflection of link #1 (mm)	–196.7
Total vertical deflection of link #2 (mm)	–6.13

CHAPTER 7
CONCLUSIONS AND RECOMMENDATIONS

Four major areas of study have been investigated in this project. Each area has been successfully completed through the use of the programming language FORTRAN77 and the Silicon Graphics IRIS 3130 Computer Graphics Workstation. The success of this project was dependent upon the programming organization of the tasks associated with each area. As work is continued on this topic, the framework of the organization as well as the prefix labels of subroutines and variables should be kept in mind. This will be the key to keeping the models uniform.

In the area of three–dimensional graphics, it is suggested that future work be in the area of shading, user–controlled scaling, and hidden line removal. In the area of user interaction, it is suggested that the user be allowed to change the shape of the segments and vary the number of segments. Additional analytical information could also be extracted to the screen or output file. In the area deflection analysis, future work is suggested in the areas of real–time analysis and also the use of feedback information from such sensors as strain gauges or lasers.

BIBILOGRAPHY

1. Dave, R. N., and A. Jana. Development of A PC—Based Robotic Simulation Package. Proceedings of the 1987 ASME International Computers in Engineering Conference and Exhibition. August 9—13, 1987. New York: ASME, 1987.

2. Derby, S. J.. Computer Graphics Robot Simulation Programs: A Comparasion. Robotics Research and Advanced Applications. (Presented at the Winter Annual Meeting of the American Society of Mechanical Engineers.) November 14—19, 1982. New York: ASME, Dynamic Systems and Controls Division, 1982.

3. Derby, Stephen James. Kinematic Elasto—Dynamic Analysis and Computer Graphics Simulation of General Purpose Robot Maniupulators. Disseration, Rensselaer Polytechnic Institute 1981. New York: Troy, 1981.

4. Fu, K. S., R. C. Gonzalez, and C. S. G. Lee. ROBOTICS: Control, Sensing, Vision, and Intelligence. New York: McGraw Hill, 1987.

5. Gannon, Kevin P.. "Modeling and Experimental Validation of a Single—Link Flexible Manipulator." Thesis, Naval Postgraduate School, Monterey, California, 1986.

6. Gere, James, M. and Stephen P. Timoshenko. Mechanics of Materials, 2ed. Belmont: Wadsworth, 1984.

7. Herbert, Martial and Regis Hoffman. Real—Time Graphic Simulation of Robotic Manipulation Using Solid Models. Proceedings — Trends and Applications, 1985: Utilizing Computer Graphics. May 20—22, 1985. Washington D.C.: IEEE, 1985.

8. Imam, I.,L. W. Sweet, J. E. Davis, M. C. Good, and K. L. Strobel. Simulation and Display of Dynamic Path Errors for Robotic Motion Off—Line Programming. Robots 8, Conference Proceedings. (Volume 1; Applications for Today.) June 4—7, 1984. Michigan: SME, 1984.

9. IRIS User's Guide Volume I Graphics Programming. Mountain View: Silicon Graphics, 1986.

10. IRIS User's Guide Volume II Graphics Programming. Mountain View: Silicon Graphics, 1986.

11. Magnenat—Thalmann, N. and D. Thalmann. Animated Types and Actor Types in Computer Simulation and Animation. Simulation in Strongly Typed Languages: ADA, PASCAL,SIMULA, Proceedings of the Conference. February 2—4, 1984. California: Society forComputer Simulation, Simulation Series 13 (1984) 51—56.

12. Nickel, Randy. "The IRIS Workstation." IEEE Computer Graphics and Applications, 4 (1984) 30—34.

13. Norton, R. L.. Graphic Simulation of Puma Robot Motions On The Apple Computer. Computers in Engineering 1983, Proceedings of the 1983 International Conference and Exhibit. (Volume 2: Robotics Theory and Applications; Computers in Education.) August 7—11, 1983. New York: ASME, 1983.

14. Okino, Norio. New Geometric Modeller TIPS/GS For Geometric Simulation. KnowledgeEngineering and Computer Modelling in CAD, Proceedings of CAD86: SeventhInternational Conference on the Computer as a Design Tool. September 2—5, 1986.London: Butterworths, 1986.

15. Parker, J. R.. "A Graphics Based Robot Simulation." Transactions of the Society for Computer Simulation, 3 (1986), 125—134.

16. Parker, James R.. "Simulating A Robot Arm Using Graphics and Animation." AI, Graphics and Simulation, (published by Society for Computer Simulation), (1985) 58—61.

17. Petroka, Robert P.. "Computer Simulation and Experimental Validation of a Dynamic Model(Equivalent Rigid Link System) on a Single—Link Flexible Manipulator." Thesis, Naval Postgraduate School Monterey, California, 1986.

18. Sweet, Larry W., J. E. Davis, M. C. Good, I. Imam, and K. L. Strobel. Simulation of Off—Line—Programmed Robot Motions. 2nd Biennial International Machine Tool Technical Conference. September 5—13, 1984. Virginia: National Machine Tool Builders' Association, 1984.

19. Tabani, Iqbal and Akbar Montaser. "Robot Motion and Task Planning: Simulation and Programming of a Robot Arm." Analytical Instrumentation, 16 (1987), 385—398.

20. Thomas, Bradley S.. Graphic 3—D Simulation of Robot Workcells. 2nd Biennial International Machine Tool Technical Conference. September 5—13, 1984. Virginia: National Machine Tool Builders' Association, 1984.

21. Tsujido, Yoshinori, Norio Kodaira, and Michitaka Oshima. Realtime Motion Simulator of Robots. Proceedings of "83 International Conference on Advanced Robotics. September 12—13, 1983. Tokyo, Japan: Japan Industrial Robot Association, 1983.

APPENDIX A

NOMENCLATURE

The following variables are listed in the order in which they appear in the dynamic com It was chosen to reference the dynamic common block because it contains all variables reference common block plus variables which are unique to the dynamic model.

```
COMMON /BASE/ BASEDI, BASEHE, PIN1X, PIN1Y, BASMAS,
+BCMASX, BCMASY, BMIX, BMIY, BMIZ, BS, BSDT, PB, PBDT
```

BASEDI = BASE DIameter

BASEHE = BASE HEight

PIN1X = PIN #1, X—coordinate

PIN1Y = PIN #1, Y—coordinate

BASMAS = BASe MASs

BCMASX = Base Center of MASs, X—coordinate

BCMASY = Base Center of MASs, Y—coordinate

BMIX = Base mass moment of inertia about the X—axis

BMIY = Base mass moment of inertia about the Y—axis

BMIZ = Base mass moment of inertia about the Z—axis

BS = BaSe

BSDT = BaSe Deformed and Transformed

PB = Pins on Base

PBDT = Pins on Base Deformed and Transformed

a.	Base Diameter		BASEDI
b.	Base Height		BASEHE
c.	Position of Pin #1 on the Base	(X)	PIN1X
d.		(Y)	PIN1Y
e.	Mass		BASMAS
f.	Center of Mass, (X)		BCMASX
g.		(Y)	BCMASY
h.	Mass Moment of Inertia about X—Axis		BMX
i.	Mass Moment of Inertia about Y—Axis		BMIY
j.	Mass Moment of Inertia about Z—Axis		BMIZ
k.	3—D (4,8,100) Array holds 3—D coordinate information of the eight base vertices		BS
l.	3—D (4,8,100) Array holds 3—D coordinate information which has been transformed using the D—H transformation matrix A0, BSDT = BS*A0		BSDT
m.	2—D (4,4) Array holds coordinate information of the location of each pin on the base. Pins included are Pin #1 in column #1 and Pin A in column #2		PB
n.	2—D (4,4) Array holds coordinate information which has been transformed using the D—H transformation matrix A0, PBDT= PB*A0		PBDT

```
COMMON /SEG1/ S1THIC, S1LENG, S1P1X, S1P1Y, S1MASS,
+ANG1MN, ANG1MX, S1CMAX, S1CMAY, PIN2X, PIN2Y, S1DIVN,
+S1E, S1G, S1A, S1MIX, S1MIY, S1MIZ, S1IY, S1IZ, S1JO,
+S1, S1D, S1DT, S1RF, P1, P1D, P1DT
```

S1THIC = Segment #1 THICkness

S1LENG = Segment #1 LENGth

S1P1X = Pin #1 on Segment #1, X–coordinate

S1P1Y = Pin #1 on Segment #1, Y–coordinate

S1MASS = Segment #1 MASS

ANG1MN = ANGle #1 MiNimum value

ANG1MX = ANGle #1 MaXimum value

S1CMAX = Segment #1 Center of MAss, X–coordinate

S1CMAY = Segment #1 Center of MAss, Y–coordinate

PIN2X = Pin #2 on Segment #1, X–coordinate

PIN2Y = Pin #2 on Segment #1, Y–coordinate

S1DIVN = Number of DIVisions Segment #1 is partitioned into

S1E = Segment #1 modulus of Elasticity

S1G = Segment #1 modulus of rigidity

S1A = Segment #1 cross–sectional Area

S1MIX = Segment #1 mass moment of inertia about the X–axis

S1MIY = Segment #1 mass moment of inertia about the Y–axis

S1MIZ = Segment #1 mass moment of inertia about the Z–axis

S1IY = Segment #1 area moment of inertia about the Y–axis

S1IZ = Segment #1 area moment of inertia about the Z–axis

S1JO = Segment #1 polar moment of inertia

S1 = Segment #1

S1D = Segment #1 Deformed

S1DT = Segment #1 Deformed and Transformed

S1RF = Segment #1 Undeformed and Transformed

P1 = Pins on Segment #1

P1D = Pins on Segment #1 Deformed

P1DT = Pins on Segment #1 Deformed and Transformed

a.	Thickness		S1THIC
b.	Length		S1LENG
c.	Position of Pin #1 on Segment 1	(X)	S1P1X
d.		(Y)	S1P1Y
e.	Mass		S1MASS
f.	Minimum Angle w.r.t. Horizontal		ANG1MN
g.	Maximum Angle w.r.t. Horizontal		ANG1MX
h.	Center of Mass,	(X)	S1CMAX
i.		(Y)	S1CMAY
j.	Position of Pin #2,	(X)	PIN2X
k.		(Y)	PIN2Y
l.	Number of Divisions		S1DIVN
m.	Modulus of Elasticity (in G Pa)		S1E
n.	Modulus of Rigidity (in G Pa)		S1G
o.	Cross—Sectional Area		S1A
p.	Mass Moment of Inertia about X—Axis		S1MIX
q.	Mass Moment of Inertia about Y—Axis		S1MIY
r.	Mass Moment of Inertia about Z—Axis		S1MIZ
s.	Area Moment of Inertia about Y—Axis		S1IY

t.	Area Moment of Inertia about Z—Axis	S1IZ
u.	Polar Moment of Inertia	S1JO
v.	3—D (4,4,100) Array holds 3—D coordinate information of the four vertices of Segment #1	S1
w.	3—D (4,4,100) Array holds 3—D coordinate information which has been altered due to deformation effects on Segment #1	S1D
x.	3—D (4,4,100) Array holds 3—D coordinate information which has been transformed using the D—H transformation matrix product of (A0*A1), S1RF = S1*(A0*A1)	S1RF
x.	3—D (4,4,100) Array holds 3—D coordinate information which has been transformed using the D—H transformation matrix product of (A0*A1), S1DT = S1D*(A0*A1)	S1DT
z.	2—D (4,4) Array holds coordinate information of the location of each pin on Segment #1. Pins included are Pin #2 in column #1 Pin B in column #2, and Pin D in column #3	P1
aa.	2—D (4,4) Array holds coordinate information of pins which has been altered due to deformation effects on Segment #1	P1D
ab.	2—D (4,4) Array holds coordinate information which has been transformed using the D—H transformation matrix (A0*A1), P1DT= P1D*(A0*A1)	P1DT

```
COMMON /SEG2/ S2THIC, S2LENG, S2P2X, S2P2Y, S2MASS,
+ANG2MN, ANG2MX, S2CMAX, S2CMAY PIN3X, PIN3Y, S2DIVN,
+S2E, S2G, S2A, S2MIX, S2MIY, S2MIZ, S2IY, S2IZ, S2JO,
+S2, S2D, S2DT, S2RF, P2, P2D, P2DT
```

S2THIC = Segment #2 THICkness

S2LENG = Segment #2 LENGth

S2P2X = Pin #2 on Segment #2, X–coordinate

S2P2Y = Pin #2 on Segment #2, Y–coordinate

S2MASS = Segment #2 MASS

ANG2MN = ANGle #2 MiNimum value

ANG2MX = ANGle #2 MaXimum value

S2CMAX = Segment #2 Center of MAss, X–coordinate

S2CMAY = Segment #2 Center of MAss, Y–coordinate

PIN3X = Pin #3 on Segment #2, X–coordinate

PIN3Y = Pin #3 on Segment #2, Y–coordinate

S2DIVN = Number of DIVisions Segment #2 is partitioned into

S2E = Segment #2 modulus of Elasticity

S2G = Segment #2 modulus of rigidity

S2A = Segment #2 cross–sectional Area

S2MIX = Segment #2 mass moment of inertia about the X–axis

S2MIY = Segment #2 mass moment of inertia about the Y–axis

S2MIZ = Segment #2 mass moment of inertia about the Z–axis

S2IY = Segment #2 area moment of inertia about the Y–axis

S2IZ = Segment #2 area moment of inertia about the Z–axis

S2JO = Segment #2 polar moment of inertia

S2 = Segment #2

S2D = Segment #2 Deformed

S2DT = Segment #2 Deformed and Transformed

S2RF = Segment #2 Undeformed and Transformed

P2 = Pins on Segment #2

P2D = Pins on Segment #2 Deformed

P2DT = Pins on Segment #2 Deformed and Transformed

a.	Thickness		S2THIC
b.	Length		S2LENG
c.	Position of Pin #2 on Segment #2	(X)	S2P2X
d.		(Y)	S2P2Y
e.	Mass		S2MASS
f.	Minimum Angle w.r.t. Segment #1		ANG2MN
g.	Maximum Angle w.r.t. Segment #1		ANG2MX
h.	Center of Mass,	(X)	S2CMAX
i.		(Y)	S2CMAY
j.	Position of Pin #3,	(X)	PIN3X
k.		(Y)	PIN3Y
l.	Number of Divisions		S2DIVN
m.	Modulus of Elasticity (in M Pa)		S2E
n.	Modulus of Rigidity (in M Pa)		S2G
o.	Cross–Sectional Area		S2A
p.	Mass Moment of Inertia about X—Axis		S2MIX
q.	Mass Moment of Inertia about Y—Axis		S2MIY
r.	Mass Moment of Inertia about Z—Axis		S2MIZ

s.	Area Moment of Inertia about Y—Axis	S2IY
t.	Area Moment of Inertia about Z—Axis	S2IZ
u.	Polar Moment of Inertia	S2JO
v.	3—D (4,4,100) Array holds 3—D coordinate information of the four vertices of Segment #2	S2
w.	3—D (4,4,100) Array holds 3—D coordinate information which has been altered due to deformation effects on Segment #1 and Segment #2	S2D
x.	3—D (4,4,100) Array holds 3—D coordinate information which has been transformed using the D—H transformation matrix product of (A0*A1*A2), S2DT = S2D*(A0*A1*A2)	S2DT
y.	3—D (4,4,100) Array holds 3—D coordinate information which has been transformed using the D—H transformation matrix product of (A0*A1*A2), S2RF = S2*(A0*A1*A2)	S2DT
z.	2—D (4,4) Array holds coordinate information of the location of each pin on Segment #2. Pins included are Pin #3 in column #1	P2
aa.	2—D (4,4) Array holds coordinate information of pins which has been altered due to deformation effects on Segment #1 and Segment #2	P2D
ab.	2—D (4,4) Array holds coordinate information which has been transformed using the D—H transformation matrix (A0*A1*A2), P2DT= P2D*(A0*A1*A2)	P2DT

COMMON /WINGS/WLENG, WING, WINGD, WINGDT, WINGRF, PW, PWD,
+PWDT

WLENG = LENGth of Wing segment

WING = Wing segment

WINGD = Wing segment Deformed

WINGDT = Wing segment Deformed and Transformed

WINGRF = Wing Segment Undeformed and Transformed

PW = Pins on Wing segment

PWD = Pins on Wing segment Deformed

PWDT = Pins on Wing segment Deformed and Transformed

a.	Wing Length	WLENG
b.	3–D (4,4,100) Array holds 3–D coordinate information of the four vertices of Wing segment	WING
c.	3–D (4,4,100) Array holds 3–D coordinate information which has been altered due to deformation effects on Segment #1 and Segment #2	WINGD
d.	3–D (4,4,100) Array holds 3–D coordinate information which has been transformed using the D–H transformation matrix product of (A0*A1*A2*AW), WINGDT = WINGD*(A0*A1*A2*AW)	WINGDT
e.	3–D (4,4,100) Array holds 3–D coordinate information which has been transformed using the D–H transformation matrix product of (A0*A1*A2*AW), WINGRF = WING*(A0*A1*A2*AW)	WINGRF

f.	2–D (4,4) Array holds coordinate information of the location of each pin on Wing segment. Pins included are Pin E in column #1	PW
g.	2–D (4,4) Array holds coordinate information of pins which has been altered due to deformation effects on Segment #1 and Segment #2	PWD
h.	2–D (4,4) Array holds coordinate information which has been transformed using the D–H transformation matrix (A0*A1*A2*AW), WINGDT= WINGD*(A0*A1*A2*AW)	PWDT

```
COMMON /SEG3/ S3THIC, S3LENG S3P3X, S3P3Y, S3MASS,
+ANG3MN, ANG3MX, S3CMAX, S3CMAY, S3, S3D, S3DT, S3RF
```

S3THIC = Segment #3 THICkness

S3LENG = Segment #3 LENGth

S3P3X = Pin #3 on Segment #3, X—coordinate

S3P3Y = Pin #3 on Segment #3, Y—coordinate

S3MASS = Segment #3 MASS

ANG3MN = ANGle #3 MiNimum value

ANG3MX = ANGle #3 MaXimum value

S3CMAX = Segment #3 Center of MAss, X—coordinate

S3CMAY = Segment #3 Center of MAss, Y—coordinate

S3 = Segment 3#

S3D = Segment #3 Deformed

S3DT = Segment #3 Deformed and Transformed

S3RF = Segment #3 Undeformed and Transformed

a.	Thickness		S3THIC
b.	Length		S3LENG
c.	Position of Pin #3 on Segment 3	(X)	S3P3X
d.		(Y)	S3P3Y
e.	Mass		S3MASS
f.	Minimum Angle w.r.t. Segment #2		ANG3MN
g.	Maximum Angle w.r.t. Segment #2		ANG3MX
h.	Center of Mass,	(X)	S3CMAX
i.		(Y)	S3CMAY

j.	3—D (4,4,100) Array holds 3—D coordinate information of the four vertices of Segment #3	S3
k.	3—D (4,4,100) Array holds 3—D coordinate information which has been altered due to deformation effects on Segment #1 and Segment #2	S3D
l.	3—D (4,4,100) Array holds 3—D coordinate information which has been transformed using the D—H transformation matrix product of (A0*A1*A2*A3), S3DT = S3D*(A0*A1*A2*A3)	S3DT
m.	3—D (4,4,100) Array holds 3—D coordinate information which has been transformed using the D—H transformation matrix product of (A0*A1*A2*A3), S3RF = S3*(A0*A1*A2*A3)	S3RF

COMMON /SEG4/ S4THIC, S4LENG, S4MASS, S4CMAX, S4CMAY

S4THIC = Segment #4 (load) THICkness

S4LENG = Segment #4 (load) LENGth

S4MASS = Segment #4 (load) MASS

S4CMAX = Segment #4 (load) Center of MAss, X—coordinate

S4CMAY = Segment #4 (load) Center of MAss, Y—coordinate

a.	Thickness		S4THIC
b.	Length		S4LENG
c.	Mass		S4MASS
d.	Center of Mass,	(X)	S4CMAX
e.		(Y)	S4CMAY

COMMON /HYD1/ H1THIC, H1MAXL, H1MINL, PINAX,

+PINAY, PINBX, PINBY, H1MASS, H1A, H1E, H1LENG

H1THIC = Hydraulic actuator #1 THICkness

H1MAXL = Hydraulic actuator #1 MAXimum Length

H1MINL = Hydraulic actuator #1 MINimum Length

PINAX = PIN A on the base, X—coordinate

PINAY = PIN A on the base, Y—coordinate

PINBX = PIN B on segment #1, X—coordinate

PINBY = PIN B on segment #1, Y—coordinate

H1MASS = Hydraulic actuator #1 MASS

H1A = Hydraulic actuator #1 cross—sectional area

H1E = Hydraulic actuator #1 modulus of Elasticity

H1LENG = Hydraulic actuator #1 LENGth

a.	Thickness		H1THIC
b.	Maximum Length		H1MAXL
c.	Minimum Length		H1MINL
d.	Position of Pin #A on Base,	(X)	PINAX
e.		(Y)	PINAY
f.	Position of Pin #B on Segment #1,	(X)	PINBX
g.		(Y)	PINBY
h.	Mass		H1MASS
i.	Cross—Sectional Area		H1A
j.	Modulus of Elasticity		H1E
k.	Length of actuator		H1LENG

COMMON /HYD2/ H2THIC, H2MAXL, H2MINL, PINDX,
+PINDY, PINEX, PINEY, H2MASS, H2LENG, H2A, H2E

H2THIC = Hydraulic actuator #2 THICkness

H2MAXL = Hydraulic actuator #2 MAXimum Length

H2MINL = Hydraulic actuator #2 MINimum Length

PINDX = PIN D on segment #1, X—coordinate

PINDY = PIN D on segment #1, Y—coordinate

PINEX = PIN E on the wing segment, X—coordinate

PINEY = PIN E on the wing segment, Y—coordinate

H2MASS = Hydraulic actuator #2 MASS

H2A = Hydraulic actuator #2 cross—sectional area

H2E = Hydraulic actuator #2 modulus of Elasticity

H2LENG = Hydraulic actuator #2 LENGth

a.	Thickness		H2THIC
b.	Maximum Length		H2MAXL
c.	Minimum Length		H2MINL
d.	Position of Pin #D on Segment #1,	(X)	PINDX
e.		(Y)	PINDY
f.	Position of Pin #E on Wing Segment,	(X)	PINEX
g.		(Y)	PINEY
h.	Mass		H2MASS
i.	Cross—Sectional Area		H2A
j.	Modulus of Elasticity		H2E
k.	Length of actuator		H2LENG

COMMON /FORCES/ FABD, FABS, FDES, FDED, FACTI, MD, MS, NFACT,
+NF, QD, RD, RS, TD

FABD = Force of actuator AB, Dynamic

FABS = Force of actuator AB, Static

FDED = Force of actuator DE, Dynamic

FDES = Force of actuator DE, Static

FACTI = Forces of ACTuators from Input

MD = Moments, Dynamic

MS = Moments, Static

NFACT = Number of current dynamic entry of Forces of ACTuators

NF = Number of input Forces

QD = Dynamic forces, f=ma

RD = forces, Dynamic Reactions

RS = forces, Static Reactions

TD = Dynamic Torques, T=I*alpha

a.	Dynamic force of actuator AB	FABD
b.	Static force of actuator AB	FABS
c.	Dynamic force of actuator DE	FDED
d.	Static force of actuator DE	FDES
e.	2—D, (2,100), Array holds the values of the forces in the actuators that the user may input in using the model. Row 1 will contain forces for AB and Row 2 will contain forces for DE	FACTI

f.	2—D (3,5), Array holds the dynamic moment reactions at pins #0, 1, 2, 3, and the load (4). Rows one thru three hold the x, y, and z moments respectively. Pin #0 moments are stored in column #1, Pin #1 in column #2, and so on.	MD
g.	2—D (3,5), Array holds the static moment reactions at #0, 1, 2, 3, and the load (4). Rows one thru three hold the x, y, and z moments respectively. Pin #0 moments are stored in column #1, Pin #1 in column #2, and so on	MS
h.	Number of dynamic force entries the dynamic input file contained.	NFACT
i.	The current dynamic force entry analysis is based on	NF
j.	2—D (3,5), Holds the dynamic force values, where in dynamics, sum of the forces = ma, of pins #0, 1, 2, and 3. Rows one thru three hold the x, y, and z forces respectively. Pin #0 forces are stored in column #1, Pin #1 in column #2, and so on.	QD
k.	2—D (3,5), Holds the dynamic force reactions at pins #0, 1, 2, 3, and the load (4). Rows one thru three hold the x, y, and z forces respectively. Pin #0 forces are stored in column #1, Pin #1 in column #2, and so on.	RD

l. 2—D (3,5), Holds the static force reactions at pins #0, 1, 2, 3, and the load (4). Rows one thru three hold the x, y, and z forces respectively. Pin #0 forces are stored in column #1, Pin #1 in column #2, and so on. RS

m. 2—D, (3,5), Holds the dynamic torques, where sum of the moments = I*alpha, at pins #0, 1, 2, and 3. Rows one thru three hold the x, y, and z torques respectively. Pin #0 torques are stored in column #1, Pin #1 in column #2, and so on TD

COMMON /VELACC/DT, OMEGA, ALPHA, VE1, VE2, AC1, AC2

AC1 = ACcelerations of partitions of segment #1

AC2 = ACcelerations of partitions of segment #2

ALPHA = angular accelerations

DT = Differential Time step

OMEGA = angular velocities

VE1 = VElocities of partitions of segment #1

VE2 = VElocities of partitions of segment #2

a. 2—D, (3,100), Array holds the x, y, and z accelerations found at each partition starting from the location of pin #1 and continuing through each partition of segment #1 until pin #2 is reached. Rows one thru three hold the x, y, and z accelerations respectively AC1

b. 2—D, (3,100), Array holds the x, y, and z accelerations found at each partition starting from the location of pin #2 and continuing through each partition of segment #2 until pin #3 is reached. Rows one thru three hold the x, y, and z accelerations respectively AC2

c. 2—D, (3,3), Array holds the x, y, and z angular accelerations of the base, segment #1 and segment #2. Rows one thru three hold the x, y, and z accelerations respectively and column #1 contains

	base information, column #2 segment #1 information and column #3 segment #2 information	ALPHA
d.	differential time step	DT
e.	2–D, (3,3), Array holds the x, y, and z angular velocities of the base, segment #1 and segment #2. Rows one thru three hold the x, y, and z velocities respectively and column #1 contains base information, column #2 segment #1 information and column #3 segment #2 information	OMEGA
f.	2–D, (3,100), Array holds the x, y, and z velocities found at each partition starting from the location of pin #1 and continuing through each partition of segment #1 until pin #2 is reached. Rows one thru three hold the x, y, and z velocities respectively	VE1
g.	2–D, (3,100), Array holds the x, y, and z velocities found at each partition starting from the location of pin #2 and continuing through each partition of segment #2 until pin #3 is reached. Rows one thru three hold the x, y, and z velocities respectively	VE2

COMMON /SLDEFL/S1SL, S1DE, S2SL, S2DE, S1DM, S2DM

S1DM = Segment #1 Deflection Multiplier

S1DE = Segment #1 DEflection

S1SL = Segment #1 SLope of deflection

S2DM = Segment #2 Deflection Multiplier

S2DE = Segment #2 DEflection

S2SL = Segment #2 SLope of deflection

a.	A multiplier used for graphic display purposes to enhance the acquired deflection of segment #1	S1DM
b.	2–D, (3,100), Array holds the deflection in the x, y, and z directions found at each partition; Rows one thru three hold the x, y, and z deflection and each column stores the information for each partition of segment #1	S1DE
c.	2–D, (3,100), Array holds the slope of the deflection in the x, y, and z directions found at each partition; Rows one thru three hold the x, y, and z deflection and each column stores the information for each partition of segment #1	S1SL
d.	A multiplier used for graphic display purposes to enhance the acquired deflection of segment #2	S2DM

e.	2—D, (3,100), Array holds the deflection in the x, y, and z directions found at each partition; Rows one thru three hold the x, y, and z deflection and each column stores the information for each partition of segment #2	S2DE
f.	2—D, (3,100), Array holds the slope of the deflection in the x, y, and z directions found at each partition; Rows one thru three hold the x, y, and z deflection and each column stores the information for each partition of segment #2	S2SL

COMMON /TRANS/AD, PNE

AD = ADjust a certain segment

PNE = adjustment is Postive, Negative, or an Error

a.	A flag used to signal which element of the robotic system the user has rotated	AD
b.	A flag used to specify if the rotation is positive or negative or if the alteration would be past the limits of the segment the error flag is set	PNE

```
COMMON /ANGINF/  ANGINC, ANGX, ANGY, ANGZ, ANG0CNT,
+ANG1CNT, ANG2CNT, ANG3CNT, ANG4CNT, ANG5CNT, ANG6CNT,
+DPHI1, DPHI2, GAMMA
```

ANGINC = INCremental value for ANGles

ANGX = ANGle of rotation of the world about X—axis

ANGY = ANGle of rotation of the world about Y—axis

ANGZ = ANGle of rotation of the world about Z—axis

ANG0CNT = ANGle CouNT of the base

ANG1CNT = ANGle CouNT of segment #1

ANG2CNT = ANGle CouNT of segment #2

ANG3CNT = ANGle CouNT of segment #3

ANG4CNT = ANGle CouNT of actuator ab

ANG5CNT = ANGle CouNT of actuator de

ANG6CNT = ANGle CouNT of wing segment

DPHI1 = differential change in angle one

DPHI2 = differential change in angle two

GAMMA = angle between segment #2 and wing

a.	Incremental value to be used for altering angles of rotation	ANGINC
b.	Angle of world rotation about the X—axis	ANGX
c.	Angle of world rotation about the Y—axis	ANGY
d.	Angle of world rotation about the Z—axis	ANGZ
e.	Angle of rotation of base about the Y—axis	ANG0CNT
f.	Angle of rotat'n of Seg #1 about the Z—axis	ANG1CNT

g.	Angle of rotat'n of Seg #2 about the Z—axis	ANG2CNT
h.	Angle of rotat'n of Seg #3 about the Z—axis	ANG3CNT
i.	Angle of rotat'n of Act AB about the Z—axis	ANG4CNT
j.	Angle of rotat'n of Act DE about the Z—axis	ANG5CNT
k.	Angle of rotation of Wing about the Z—axis	ANG6CNT
l.	Differential change in angle of segment #1 due to material parameters and applied forces	DPHI1
m.	Differential change in angle of segment #2 due to material parameters and applied forces	DPHI2
n.	Angle of Wing Segment w.r.t. Segment #2	GAMMA