INFORMATION TO USERS

This reproduction was made from a copy of a manuscript sent to us for publication and microfilming. While the most advanced technology has been used to photograph and reproduce this manuscript, the quality of the reproduction is heavily dependent upon the quality of the material submitted. Pages in any manuscript may have indistinct print. In all cases the best available copy has been filmed.

The following explanation of techniques is provided to help clarify notations which may appear on this reproduction.

1. Manuscripts may not always be complete. When it is not possible to obtain missing pages, a note appears to indicate this.

2. When copyrighted materials are removed from the manuscript, a note appears to indicate this.

3. Oversize materials (maps, drawings, and charts) are photographed by sectioning the original, beginning at the upper left hand corner and continuing from left to right in equal sections with small overlaps. Each oversize page is also filmed as one exposure and is available, for an additional charge, as a standard 35mm slide or in black and white paper format.*

4. Most photographs reproduce acceptably on positive microfilm or microfiche but lack clarity on xerographic copies made from the microfilm. For an additional charge, all photographs are available in black and white standard 35mm slide format.*

*For more information about black and white slides or enlarged paper reproductions, please contact the Dissertations Customer Services Department.

University Microfilms International
A Bell & Howell Information Company
300 N. Zeeb Road, Ann Arbor, Michigan 48106
Tsai, Ming-June

WORKSPACE GEOMETRIC CHARACTERIZATION AND MANIPULABILITY OF INDUSTRIAL ROBOTS

The Ohio State University

Ph.D. 1986

University Microfilms International 300 N. Zeeb Road, Ann Arbor, MI 48106
PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark ✓.

1. Glossy photographs or pages
2. Colored illustrations, paper or print
3. Photographs with dark background
4. Illustrations are poor copy
5. Pages with black marks, not original copy
6. Print shows through as there is text on both sides of page
7. Indistinct, broken or small print on several pages ✓
8. Print exceeds margin requirements
9. Tightly bound copy with print lost in spine
10. Computer printout pages with indistinct print
11. Page(s) lacking when material received, and not available from school or author.
12. Page(s) seem to be missing in numbering only as text follows.
13. Two pages numbered. Text follows.
14. Curling and wrinkled pages
15. Dissertation contains pages with print at a slant, filmed as received
16. Other

University Microfilms International
WORKSPACE GEOMETRIC CHARACTERIZATION AND
MANIPULABILITY OF INDUSTRIAL ROBOTS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the Graduate
School of The Ohio State University

By

Ming-June Tsai, B.S.M.E, M.S.W.E.

The Ohio State University
1986

Dissertation Committee:
Dr. Kenneth J. Waldron
Dr. Charles D. Jones
Dr. Gary L. Kinzel
Dr. Chia H. Menq

Approved By

Adviser
Department of
Mechanical Engineering
To my family
ACKNOWLEDGMENTS

I express sincere appreciation to Dr. Kenneth J. Waldron for his guidance and insightful advise throughout the research. Also, I want to express my thanks to my advisory committee members, Dr. Gary L. Kinzel, Dr. Charles D. Jones, and Dr. Chia H. Meng for their suggestions and comments. Special appreciation is extended to National Science Foundation for their grant (No. MEA-8304735) which makes this study possible.
VITA

June 20, 1952 ....... Born - Taiwan, ROC

1971 ................. B.S.M.E, Feng-Chia University
Taichung, Taiwan, ROC

1975-1977 ............. Military Service,
Combined Service Force, Taiwan, ROC

1977-1980 ............. Mechanical Engineer
Taiwan Power Co., Taiwan, ROC

1983 ................. M.S.W.E.
The Ohio State university
Columbus, Ohio

1981-1983 ............. Research Assistant
Department of Welding Engineering
The Ohio State University
Columbus, Ohio

1984-1986 ............. Research Assistant
Department of Mechanical Engineering
The Ohio State University
Columbus, Ohio

PUBLICATIONS

1. Tsai, M.J., and Waldron, K.J.
"Geometric Characteristics of Industrial Robots,"
To be presented in ASME Mechanism Conference, Oct. 5-8,
1986, Columbus, Ohio.

2. Tsai, M.J., and Waldron, K.J.
"Characterization of Manipulator Workspace Geometries,"

3. Vijaykumar, R., Tsai, M.J., and Waldron, K.J.

4. Tsai, C.L., and Tsai, M.J.

5. Tsai, M.J.

FIELDS OF STUDY

Major Field: Mechanical Design of Robot Manipulator

Studies in: From Faculty Members:
Robotics Kenneth J. Waldron
(ME752, EE880) David E. Orin
Kinematics Kenneth J. Waldron
(ME651, 894H, 894K) Kenneth H. Hunt
Welding Design Chon L. Tsai
(ME620, 621, 622, 723, 821) Koichi Masubuchi
Mechanical Design Truman G. Foster
(ME694F, 860, 861) Jack A. Collins
Gary L. Kinzel
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section / Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ACKNOWLEDGMENTS</strong></td>
<td>iii</td>
</tr>
<tr>
<td><strong>VITA</strong></td>
<td>iv</td>
</tr>
<tr>
<td><strong>TABLE OF CONTENTS</strong></td>
<td>vi</td>
</tr>
<tr>
<td><strong>LIST OF TABLES</strong></td>
<td>viii</td>
</tr>
<tr>
<td><strong>LIST OF FIGURES</strong></td>
<td>ix</td>
</tr>
<tr>
<td><strong>Chapter</strong></td>
<td><strong>Page</strong></td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Literature Review</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Scope and Objectives</td>
<td>5</td>
</tr>
<tr>
<td>II. THEORETICAL BACKGROUND</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Introduction of Screw Theory</td>
<td>9</td>
</tr>
<tr>
<td>2.1.1 Plucker Coordinate of Line and Screw</td>
<td>9</td>
</tr>
<tr>
<td>2.1.2 Reciprocal Relationship of Screws</td>
<td>14</td>
</tr>
<tr>
<td>2.1.3 Screw Theory and Linear Algebra</td>
<td>16</td>
</tr>
<tr>
<td>2.2 Jacobian Matrix and Screw Matrix</td>
<td>18</td>
</tr>
<tr>
<td>2.3 Special Configurations of Robot Manipulator</td>
<td>20</td>
</tr>
<tr>
<td>III. WORKSPACE GENERATION</td>
<td>25</td>
</tr>
<tr>
<td>3.1 Properties of Industrial Robot Type Geometries</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Workspace Generation</td>
<td>31</td>
</tr>
<tr>
<td>3.2.1 Geometric Arrangement of Joint Parameters</td>
<td>31</td>
</tr>
<tr>
<td>3.2.2 Reciprocal Conditions</td>
<td>33</td>
</tr>
<tr>
<td>3.2.3 Workspace Generation Algorithm</td>
<td>35</td>
</tr>
<tr>
<td>3.3 Examples</td>
<td>40</td>
</tr>
<tr>
<td>3.4 Hand Orientation Capability</td>
<td>44</td>
</tr>
<tr>
<td>IV. VOLUME AND MOMENT COMPUTATION</td>
<td>47</td>
</tr>
<tr>
<td>4.1 Gaussian Divergence Theorem</td>
<td>48</td>
</tr>
<tr>
<td>4.2 Boundary Generated by RJJS Chain</td>
<td>48</td>
</tr>
<tr>
<td>4.2.1 Case A: Revolute Joint at Joint 2 or 3</td>
<td>48</td>
</tr>
<tr>
<td>4.2.2 Case B: Prismatic Joint at Joint 2 or 3</td>
<td>58</td>
</tr>
<tr>
<td>4.3 Boundary Generated by PJJS Chain</td>
<td>62</td>
</tr>
<tr>
<td>4.3.1 Case C: Revolute Joint at Joint 2 or 3</td>
<td>62</td>
</tr>
<tr>
<td>4.3.2 Case D: Prismatic Joint at Joint 2 or 3</td>
<td>68</td>
</tr>
<tr>
<td>4.4 Elimination of Overlapped Volume</td>
<td>70</td>
</tr>
</tbody>
</table>
4.4.1 Overlapped Sub-volume Due to Re-entrant Surfaces ................................ 71
4.4.2 Overlapped Sub-volume Due to Excessive Motion of Joint 2 ...................... 73
4.4.3 Overlapped Sub-volume Due to Excessive Motion of Joint 1 ...................... 75
4.5 Examples ................................... 76
  4.5.1 Example 1: Cincinnati Milacron T3-746 ... 77
  4.5.2 Example 2: Reis Machine Corp. RR-625/650 79
  4.5.3 Example 3: Citizen Robotics Corp. M-100 . 79
  4.5.4 Example 4: Cincinnati Milacron T3-886 ... 82

V. MANIPULATOR CONFIGURATION AND ASPECT ............... 84
  5.1 Configurations and Aspects of Robot .......... 84
  5.2 Workspace Generation by Aspect Method ....... 87
  5.3 Example 1: Cincinnati Milacron T3-746 ...... 89
  5.4 Example 2: Reis RR-625/650 Robot ............ 94

VI. MANIPULABILITY AND SINGULARITY ...................... 98
  6.1 Singular Value Decomposition ............... 99
  6.2 Definitions of Manipulability .......... 100
  6.3 Singular Configurations of Robot-Arm ....... 102
  6.4 Manipulability Field in Workspace ....... 106
  6.5 Penalty Function for Manipulability ...... 113
  6.6 Modified Manipulability Field .......... 116

VII. DISCUSSION AND CONCLUSION ......................... 122
  7.1 Significance of This Study ............... 122
  7.2 Criteria for evaluation of Workspace Geometry 123
  7.3 Geometric Comparison of Industrial Robots ... 126
  7.4 Conclusion .................................. 129

LIST OF REFERENCES ........................................ 132
APPENDIX A ........................................... 137
APPENDIX B ........................................... 251
APPENDIX C ........................................... 260
APPENDIX D ........................................... 269
LIST OF TABLES

Table                                                                 Page
1. Possible Geometries for Industrial Robots ............ 29
2. Geometric Properties of Four Industrial Robots ..... 127
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Relationship of Two Screw Axes in Space</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>Relationship of Twist and Wrench Screw Systems</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>Some Typical Industrial Robot Type Geometries</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>Typical Workspace Plotting Planes for Robots</td>
<td>32</td>
</tr>
<tr>
<td>5</td>
<td>Reciprocal Conditions of Joints</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>Algorithm for Workspace Boundary Generation</td>
<td>38</td>
</tr>
<tr>
<td>7</td>
<td>Workspace Geometry of CM T3-746</td>
<td>41</td>
</tr>
<tr>
<td>8</td>
<td>Workspace Geometry of Reis RR-625/650</td>
<td>42</td>
</tr>
<tr>
<td>9</td>
<td>Workspace Geometry of Citizen M-100</td>
<td>43</td>
</tr>
<tr>
<td>10</td>
<td>Workspace Geometry of CM T3-886</td>
<td>44</td>
</tr>
<tr>
<td>11</td>
<td>Workspace with Hand Orientation</td>
<td>46</td>
</tr>
<tr>
<td>12</td>
<td>Geometry of Sub-volume Bounded by Torus Portion</td>
<td>49</td>
</tr>
<tr>
<td>13</td>
<td>Sub-Volume Geometry Bounded by Conical Portion</td>
<td>59</td>
</tr>
<tr>
<td>14</td>
<td>Sub-Volume Geometry Bounded by Cylinder Portion</td>
<td>63</td>
</tr>
<tr>
<td>15</td>
<td>Sub-Volume Geometry Bounded by Prism Portion</td>
<td>68</td>
</tr>
<tr>
<td>16</td>
<td>Overlapped Area Due to Re-entrancy</td>
<td>73</td>
</tr>
<tr>
<td>17</td>
<td>Removal of Overlapped Area</td>
<td>74</td>
</tr>
<tr>
<td>18</td>
<td>Overlapped Sub-Volume Due to Joint 2</td>
<td>75</td>
</tr>
<tr>
<td>19</td>
<td>Overlapped Sub-Volume Due to Joint 1</td>
<td>76</td>
</tr>
<tr>
<td>20</td>
<td>Workspace Properties of CM T3-746</td>
<td>78</td>
</tr>
<tr>
<td>21</td>
<td>Workspace Properties of Reis RR-625/650</td>
<td>80</td>
</tr>
</tbody>
</table>
22. Workspace Properties of Citizen M-100 ............ 81
23. Workspace Properties of CM T3-886 ................. 83
24. Operating Barrier of CM T3-746 ..................... 85
25. Manipulator with 3 Links in a Plane ................. 88
26. Aspects of a Manipulator with \( u=v=1 \) ............... 90
27. Joint Space Representation of CM T3-746 ............. 91
28. Aspects of CM T3-746 in Joint Space ................. 92
29. Workspace of CM T3-746 Obtained by Aspects ......... 93
30. Joint Space Representation of RR-625/650 .......... 95
31. Workspace of RR-625/650 from Aspect Method ........ 96
32. Manipulability vs. A Joint Variable ................. 107
33. Manipulability Field for CM T3-746 ................. 109
34. Manipulability Field for Reis RR-625/650 .......... 111
35. Manipulability Field for CM T3-886 ................. 112
36. Penalty Function for Manipulability ................. 115
37. Modified Manipulability Field for CM T3-746 ....... 117
38. Modified Manipulability Field for Reis RR-650 ...... 119
39. Modified Manipulability Field for CM T3-886 ...... 120
CHAPTER I

INTRODUCTION

1.1 Literature Review

In the recent two decades, computer controlled robots and manipulators have been widely used in industries for tasks which are hazardous, monotonous, or tiresome for human beings. For example, mechanical manipulators have been successfully employed for arc welding and painting in the automobile industry, for handling dangerous materials in nuclear power plants, and even for handling objects in outer space for NASA's space shuttle. The study of robot manipulators has become a very active field.

In three dimensional space, a general purpose manipulator needs at least six degrees of freedom in order to perform a task dexterously. Since there are numerous ways to synthesize such a manipulator, the geometric properties of manipulators are very complex. A traditional manipulator design is constructed as an open series kinematic chain. The series kinematic chain used as an manipulator arm has advantages of long reach, large range of
motion, and easy actuation and control. However, there are several disadvantages. For examples, it is structurally a cantilever beam which is inherently not very stiff, and it has many singular configurations and kinematic indeterminacies.

Recently, many optimization techniques have been applied to workspace generation for this type of manipulator. One approach to the development of computer-aided design tools is to attempt to synthesize manipulator geometries based on desired workspace properties. This goal has remained elusive. In fact, efficient algorithms for the generation of a workspace, given the manipulator structure, and for computation of its geometric properties, are still subjects of active research. Roth [1] was one of the first researchers to present work related to the workspace of a manipulator. Shimano and Roth [2] postulated a method to find the maximum distance between two axes while there are intermediate axes in between. Gupta and Roth [3] and Yang and Lee [4] further discussed the general workspace shapes of n-R manipulators. So far, many algorithms have been presented for finding the workspaces of manipulators. Published works in this area can be divided into two groups. In the first group are studies of the geometric properties of manipulators with general geometry [5-10]. In the second group are studies
which concentrate on manipulators with more specialized geometry, such as typical industrial robots\(^*\) [11-19]. Meanwhile, some criteria which are based on geometric optimization techniques have been established for evaluation of a manipulator's performance [18-21].

During the intense recent interest in research in robotics, another important mathematical technique - screw system theory, which had nearly been ignored for several decades, has re-attracted the attention of many kinematicians. In the early 19th century, J. Plucker studied line geometry and developed a method of specifying the position of a line in space by means of six coordinates which can be generalized to describe the rigid body concepts of screws, twists and wrenches. This is the well known "Plucker line coordinate." In 1900, R.S. Ball [22] published his monumental book: "Theory of Screws," which related instantaneous mobility to the joint geometry of a constrained rigid body in three dimensional space.

The nature of screw system theory makes it a very feasible tool for study of spatial mechanisms. Surprisingly, it received little attention by kinematicians

\(^*\)In this study, only the geometry design of industrial robots is of interested. Therefore, the term "manipulator" is treated as synonymous to "robot-arm" or simply "robot"
until the last two decades. The robot industry has gained much benefit from research efforts based on screw theory, chiefly with regard to exploring special configurations of manipulators [23-26]. Workspace boundary finding algorithms, as mentioned above, are mainly based on the discovery of singular configurations of a robot arm. A very popular manipulator control method, known as the "rate control algorithm", is based on the concepts of "motors" and Jacobian matrices which stem from screw theory [27,28]. It is also well known that, at a singular configuration, the control algorithm may break down due to the singularity of the Jacobian matrix. Studies of the geometry-based controllability (or so called manipulability) of the robot hand from the point view of singularity of the Jacobian matrix have been reported [29-34]. Manipulability has been used as a criterion function to be maximized during the control of manipulators with kinematic redundancy [31,34,35]. Recently, more advanced control algorithms, which take the forces or dynamics into consideration, are also available [34-38].

As a result of these different characteristics, the design of a manipulator would depend upon the task it should perform, the position and path accuracy needed, and the workspace to be covered. The general procedure for design of a manipulator is to first optimize the manipulator's
geometry according to the desired performance. One then studies the characteristics and the manipulability of the manipulator so that the overall kinematic performance can be properly evaluated. Finally a control algorithm can be formulated based on the kinematic properties of the manipulator. Therefore, extensive researches on the geometric performance and the controllability of manipulators are greatly needed.

1.2 Scope and Objectives

The topic of this dissertation is the study of the geometric characteristics of robot manipulators, with the focus placed on manipulators with special geometries, i.e. the industrial robot type geometries. In Chapter Two, basic information about line geometry in space and the fundamentals of screw system theory are given. In the Third Chapter, the geometric optimization of robot manipulators is discussed, and some special geometries which are suitable for industrial application are proposed. An algorithm for workspace generation for industrial robots is then presented. This algorithm uses the theory of reciprocal screws for finding the extreme reach of a robot-arm. It can handle robots with revolute joints as well as with prismatic joints.

The Fourth Chapter deals with an algorithm for
computing geometric properties of the workspace, such as its volume, the location of its centroid, and its moments of inertia about the three principal axes. In reference [14] a technique was presented for computation of the volumes of workspaces generated by industrial robot type geometries. This technique utilized a formulation based on Green’s function. In the present study, the computation of higher moments, and other aspects of the Green’s function technique, are explored. The work described in reference [14] is also extended to include geometries which have prismatic joints. In addition to being a very different approach to the problem of the grid scanning techniques used by most other authors, this formulation gives promise of an efficient means of computing volume moments of inertia about principal axes, and even of computing higher moments. This, in turn, gives promise of an ability to characterize the shape of the volume by means of its moments, as is commonly done to characterize shapes for recognition by robot vision systems. The objective is to allow computation of figures of merit, or of other means of characterizing the quality of a given geometry, and hence to provide a basis for geometric optimization.

In Chapter Five the aspects [41] of a manipulator are discussed. The workspace of a given aspect is a sub-set of the total workspace of the robot. The union of all the
workspaces of the aspects is the total workspace of the robot. Therefore, the total workspace of a robot-arm can be found by examining all the sub-workspaces of all the aspects. The workspace of a robot obtained in this way has interior surfaces which are the boundaries of aspects. When mapped into the joint space, those inside surfaces are easily seen to be surfaces produced by joint limits and surfaces for which the minors of the Jacobian are singular. All the re-entrant surfaces traced by the workspace generation algorithm are coincident to some of the interior surfaces generated by aspects. The re-entrant surfaces can also be categorized into two parts: singular surfaces and D-surfaces [39]. Singular surfaces are created by the Jacobian matrix of a robot-arm becoming singular, whereas the D-surfaces are created by joint limits as well. The re-entrant surfaces are of more significance than the aspect surfaces because we can always apply a force normal to those surfaces. They are non-crossable when the robot-arm poses in the configuration in which they were traced. They represent barriers within the workspace which may affect the controllability of the robot-arm.

In Chapter 6 the manipulability of a robot-arm is studied. The manipulability of a robot-arm during operation can be evaluated by means of the singular values of the Jacobian matrix at that instant. This also provides a
quantitative measure of how close the robot-arm is to a singular state. Suggestions about the arrangement of joint parameters to minimize the occurrence of singular surfaces are presented. The D-surfaces, which are inevitable for industrial robots having joint limits, cause the robot to lose one or more degrees of freedom as if they were singular postures. A penalty function is proposed here to be combined with the manipulability index. It forces the modified manipulability to decrease rapidly near to a joint limit. This will enable some advanced control algorithms, which take manipulability as the criterion for singularity avoidance, to keep the robot from operating beyond its joint limits.

In the last chapter, several criteria for evaluation of workspace geometric performance are postulated, and some examples which clearly illustrate the geometric characteristics of manipulators are also presented. Many special features which are unique in this study are also summarized in this chapter. The overall goal of this study is to provide a very efficient means for design and evaluation of a manipulator by use of interactive computer graphic facilities. A software package as a computer-aided design tool allowing a high speed on-line design procedure has also been developed.
CHAPTER II
THEORETICAL BACKGROUND

2.1 Introduction of Screw Theory

The basic concepts of line and screw geometries in space can be found in textbooks and papers relating to the screw theory. A brief review on the fundamentals of screw theory summarized from reference [22] and [24] is given in this section without any proofs.

2.1.1 Plücker Coordinates of a Line and Screw Axis

The Plücker coordinates of a line are denoted by six parameters \((L, M, N; P, Q, R)\). \(L, M, N\) is the direction-ratio of the line and \(P, Q, R\) the resultant moment of the line about the origin \(O\). Plücker line coordinates can be derived in two ways: The homogeneous equation of points and planes in rectangular Cartesian coordinates is written as:

\[
tx + uy + vz + sw = 0 \quad (1)
\]

It can either represent all the points \((x, y, z, w)\) that lie in
a given plane \((t,u,v,s)\) or represent all the planes 
\((t,u,v,s)\) that contain a given point \((x,y,z,w)\). Where \(w\) and
\(s\) are scalar factors; when \(w=0\) is the condition specifying
all the points in the plane at infinity, whereas \(s=0\) is the
condition for all the planes that contain the origin. A
line can either be defined as the joint of two points or the
intersection of two planes. A line can be regarded as a
linear series of \(\omega^1\) points as well as the axis of an axial
pencil of \(\omega^1\) planes. This duality of points and planes in
space enables the Plücker coordinates to be derived in two
ways. Given two points, 1 and 2, the six second-order
determinants of the matrix

\[
\begin{bmatrix}
  w_1 & x_1 & y_1 & z_1 \\
  w_2 & x_2 & y_2 & z_2 \\
\end{bmatrix}
\]

yield the six Plücker "ray" coordinates of the line through
points 1 and 2. These are:

\[
L = \begin{vmatrix} w_1 & x_1 \\ w_2 & x_2 \end{vmatrix} \quad M = \begin{vmatrix} w_1 & y_1 \\ w_2 & y_2 \end{vmatrix} \quad N = \begin{vmatrix} w_1 & z_1 \\ w_2 & z_2 \end{vmatrix} \\
P = \begin{vmatrix} y_1 & z_1 \\ y_2 & z_2 \end{vmatrix} \quad Q = \begin{vmatrix} z_1 & x_1 \\ z_2 & x_2 \end{vmatrix} \quad R = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix}
\]

(2)

The Plücker "axis" coordinates of the line can be obtained
from the intersection of two planes in the same way, but we
write the six parameters in different order: \( (P,Q,R; L,M,N) \). In this study we use "ray" coordinates only.

Since a line is always perpendicular to its moment about the origin 0, we may see that inherent in these six variables is the quadratic identity:

\[
LP + MQ + NR = 0 \tag{3}
\]

Also, since the six coordinate parameters are all direction ratios, we may be interested only in their relative magnitudes. Therefore, they are conventionally normalized by dividing \( (L^2+M^2+N^2)^{0.5} \). Then

\[
L^2 + M^2 + N^2 = 1 \tag{4}
\]

A line in the plane at infinity has coordinates \( (0,0,0; P,Q,R) \). We may normalize it by dividing its three non-zero coordinates by \( (P^2+Q^2+R^2)^{0.5} \). If \( r \) is the perpendicular distance from the origin to a general line, then

\[
(P^2 + Q^2 + R^2)^{0.5} = r(L^2 + M^2 + N^2)^{0.5} \tag{5}
\]

For example, line \( (1,0,0; 0,0,0) \) is the x-axis. Line \( (1,0,1; 0,1,0) \) lies on the plane \( y=0 \) and intersects the z-axis at 1 and x-axis at -1. It is the same as line \( (2,0,2; 0,2,0) \) since their normalized coordinates are the same. The six homogeneous coordinates combined in Equations (3) and (4) give only a total of four independent variables
for a line which agrees with the four variables which define
a line in Cartesian coordinates.

A screw is a line to which a scalar pitch \( h \) with a
length dimension is attached. Any finite displacement of a
rigid body can be represented as a translation along a screw
axis together with a rotation about the screw axis. Since
there are \( \alpha^4 \) lines in space, each line has \( \alpha^1 \) pitches
available for selection, then there are total \( \alpha^5 \) screw axes
in space. If one travels along the screw axis a distance \( h \),
then he will rotate about the axis one revolution. The dual
concepts of instantaneous kinematics and statics of rigid
bodies can be elegantly described by means of the properties
of screw axes.

Statics of a rigid body:

Any given system of forces and couples acting on a
rigid body can be replaced by a resultant wrench - a pair of
parallel vectors consisting of a force \( F \) along the line of
the screw axis and a couple \( C \) about the screw axis.

\[
C = h F
\]

The intensity of the wrench is \( F \); when \( h=0 \) then we may use \( C \)
as the intensity. Letting \( \mathbf{r} \) be the vector from a screw to
the origin, the resultant moment \( M \) of the screw about the
origin is
\[
M = E \times r + hE
\]

**Instantaneous kinematics**

Any given infinitesimal displacement of a body can be expressed as a twist about a screw axis: an angular velocity component \( \omega \) about an instantaneous screw axis together with a translational velocity component \( I \) parallel to \( \omega \).

\[I = h\omega\]  

The amplitude of the twist is \( \omega \); when \( h = \infty \), \( \tau \) may be used as the amplitude. The velocity \( \nu \) of the point instantaneously at the origin is

\[\nu = \omega \times I + h\omega\]

The direction cosines of \( \nu \) or \( \psi \) are \( P + hL, Q + hM, \) and \( R + hN \) (if normalized). If we write

\[P^* = P + hL \quad Q^* = Q + hM \quad R^* = R + hN\]

then a screw \( \xi \) can accordingly be said to have screw coordinates \( (L, M, N; P^*, Q^*, R^*) \). The pitch \( h \) of the screw can be easily obtained as

\[h = \frac{LP^* + MQ^* + NR^*}{L^2 + M^2 + N^2}\]

A screw at infinity has coordinates \((0, 0, 0; P^*, Q^*, R^*)\), and
its pitch can only be infinite. In kinematics it represents a pure translation motion (of a prismatic joint), whereas in statics it may be interpreted as a wrench which is a pure couple.

2.1.2 Reciprocal Relationship of Screws

The perpendicular distance $a$ between two general lines in space is

$$a = \frac{L_1 P_2 + M_1 Q_2 + N_1 R_2 + L_2 P_1 + M_2 Q_1 + N_2 R_1}{\sin \alpha}$$  \hspace{1cm} (12)

The angle $\alpha$ between these two lines is

$$\cos \alpha = \frac{L_1 L_2 + M_1 M_2 + N_1 N_2}{\sin \alpha}$$  \hspace{1cm} (13)

A force $F$ can do no work on a body free only to rotate about a screw axis on line $1$, when the line of action of the force (line 2) intersects the axis of rotation (line 1), namely when

$$L_1 P_2 + M_1 Q_2 + N_1 R_2 + L_2 P_1 + M_2 Q_1 + N_2 R_1 = 0$$  \hspace{1cm} (14)

Given a body free to twist about a screw axis $\$\$ subject to a wrench about another screw axis $\$$' (as shown in Figure 1), the instantaneous rate of working is

$$\text{Power} = F \omega \left( (h + h') \cos \alpha - a \sin \alpha \right)$$  \hspace{1cm} (15)
Figure 1: Relationship of Two Screw Axes in Space

where $h$ and $h'$ are pitches of $\$\$ \$ and $\$'\$' \$ respectively. Expressed in the parameters of Plücker coordinates, the condition in which the wrench on a screw axis $\$'$ $\$' \$ can do no work on the body constrained to a twist about another screw $\$ is

$$(h + h')\cos\alpha - a \sin\alpha = 0,$$ \hspace{1cm} (16)

or represented in Plücker coordinates:

$$LP' + MQ' + NR' + LP + MQ + NR = 0 \hspace{1cm} (17)$$

This is the so-called reciprocity relationship, and the screw axes $\$ and $\$'$ $\$' \$ are said to be mutually reciprocal. In general, a screw system $\$'$ $\$' \$ is said to be reciprocal to
another screw system $ when the following condition is satisfied:

$$\mathbf{s} \cdot \mathbf{s}'^T = 0$$  \hspace{1cm} (18)

Where $\mathbf{s} = [L, M, N; P^A, Q^A, R^A]$ and $\mathbf{s}' = [P^A', Q^A', R^A'; L', M', N']$.

2.1.3 Screw Theory and Linear Algebra

Some basic applications of linear algebra to screw system theory have been presented in references [24, 25, 42]. Some important properties of line and screw geometries are presented here for quick reference.

Given $n$ linearly independent lines, $1 \leq n \leq 5$, an $(n+1)$th line is linearly dependent on them when the rank of the $(n+1)$-by-six matrix comprising the $(n+1)$ lines is $n$. In general,

1. when $n=1$, there are no other lines linearly dependent on it.

2. when $n=2$, there are no other lines linearly dependent on them except when the two lines are coplanar.

3. when $n=3$, there are $\omega^1$ lines linearly dependent on them comprising a regulus, requiring three homogeneous linear conditions. (i.e. three linear constraints)

4. when $n=4$, there are $\omega^2$ lines linearly dependent on them comprising a linear congruence, requiring two homogeneous linear conditions.

5. when $n=5$, there are $\omega^3$ lines linearly dependent
on them comprising a linear complex, requiring one homogeneous linear condition.

6. when \( n = 6 \), all \( \omega^4 \) lines in the space are linearly dependent on them, no restricting condition required.

Similarly, given \( n \) linearly independent screw axes, \( 1 \leq n \leq 5 \), an \((n+1)\)th screw is linearly dependent on them when the rank of the \((n+1)\)-by-six matrix comprising the \((n+1)\) screws is \( n \).

In general,

1. when \( n = 1 \), there are no other screw axes linearly dependent on the given screw axis. The single screw axis comprises a one-system requiring five homogeneous linear conditions.

2. when \( n = 2 \), there are \( \omega^1 \) screw axes linearly dependent on the given screw axes, comprising a two-system, requiring four homogeneous linear conditions in Plucker coordinates to specify them. The screw axes of a two-system generally lie on a cylinder.

3. when \( n = 3, 4, \) or \( 5 \), there are \( \omega^2, \omega^3, \omega^4 \) screws linearly dependent on the given screw axes, comprising a three-, four-, or five-system, requiring three, two, or one homogeneous linear conditions respectively.

4. given six linearly independent screws, all \( \omega^5 \) screws in space are linearly dependent, no restricting conditions are required.

Accordingly, a general purpose robot arm should have 6 linearly independent screw axes in order to perform a task dexterously.

In a series connected kinematic chain, each link acts relative to another about a screw axis, we denote the screw by \( \xi_j \). The first subscript represents the reference link
and the second is the relatively moving link. When a body, acted upon simultaneously in series by \( n \) twists about \( n \) screws \( \theta_1, \theta_2, \ldots, \theta_{(n-1)n} \), is instantaneously at rest relative to the fixed frame, then the vector sum of the \( n \) twists is zero. If the amplitude of a twist about the screw axis \( \theta_{ij} \) is \( \omega_{ij} \), let \( S=(\theta_0, \theta_1, \ldots, \theta_{(n-1)n}) \), and \( \Omega=(\omega_0, \omega_1, \ldots, \omega_{(n-1)n})^T \), then

\[ S\Omega = 0 \quad (19) \]

### 2.2 Jacobian Matrix and Screw Matrix

Considering a general \( n \) link manipulator, letting \( p \), a position vector, be the position of the hand reference point of the manipulator relative to the fixed frame \( p \in \mathbb{R}^n \), and \( g \) the column vector of \( n \) joint variables, \( g \in \mathbb{R}^n \), the relationship between the vectors \( p \) and \( g \) is given as follows.

\[ p = f(g) \quad (20) \]

At an instant, the relationship of hand velocity state \( [\dot{w}, \dot{u}]^T \) and joint rates \( \dot{g} \) is given in motor notation by \([26,28]:(21)\):

\[ \begin{bmatrix} \dot{w} \\ \dot{u} \end{bmatrix} = J\dot{g} \quad (21) \]
where $\omega$ is the end-effector's angular velocity and $\mu$ the translational velocity. $J$ is the Jacobian matrix, and $\dot{q}$ is a column matrix of $n$ joint rates.

When controlling a movement of an industrial robot which operates in the point-to-point mode, it is necessary to generate a smooth trajectory for motion between the initial and final position. One control method has been particularly successful. This is the rate control algorithm. The joint rate computation, on which the method is based, is fulfilled by inverting the Jacobian matrix in Equation (21). Therefore, the Jacobian matrix plays a very important role in the analysis of the controllability of the robot arm.

Since $[\omega, \mu]^T$ is a 6 component vector which represents the resultant twist along a screw $\mathbf{s}_{n0}$ for the end-effector with respect to the fixed frame, we can pretend that there is a hypothetical joint located on the axis of the screw $\mathbf{s}_{n0}$. This hypothetical joint between the end-effector and the base forms a closed-loop spatial kinematic chain. Now let the amplitude of the resultant twist about the screw $\mathbf{s}_{n0}$ to be $\omega_{n0}$. From the Equation (19), extracting the screw $\mathbf{s}_{n0}$ out of the screw matrix one gets:

$$S\dot{q} + \mathbf{s}_{n0}\omega_{n0} = 0$$

or since $\omega_{n0} = -\omega_{0n}$, $\mathbf{s}_{n0} = \mathbf{s}_{0n}$, and the velocity state of
the end-effector with respect to the fixed reference frame is \([\omega, \mu]^T = s_0w_0n\), one then obtains:

\[
\dot{\mathbf{s}} = \begin{bmatrix}
\omega \\
\mu
\end{bmatrix}
\]

Therefore, the Jacobian matrix is no more than a screw matrix constructed by the screws of joint axes. The behavior of the Jacobian matrix can then be discussed by means of screw theory.

### 2.3 Special Configurations of Robot Manipulator

As was discussed in the previous section, the rate control algorithm requires that the Jacobian matrix be inverted in real time. This is possible since almost all industrial robots are specially designed so that their joint axes are all orthogonal or parallel to each other. This greatly simplifies the formation of the Jacobian matrix. By transforming the reference frame to the frame of member 3 or 4, closed-form joint rate solutions can be obtained analytically without actually inverting the 6x6 Jacobian matrix in real time [28]. However, when the manipulator is in a special configuration such that the Jacobian matrix of the manipulator is singular, the joint rates are not obtainable by this control algorithm. Once a robot encounters such a configuration, the system usually just
shuts down, since no correction is implemented for most industrial robots. Therefore, study of the special configurations which may cause the rate control algorithm to break down, is very important.

For a six-jointed manipulator, the six screws in general are linearly independent, and the rank of their $6 \times 6$ screw matrix is 6. There is no screw reciprocal to all of them. However, we can easily encounter a special configuration of the robot arm. The rank of the screw matrix is less than 6. The screw system is degenerate. In this situation, there exists a complemental screw system ($\delta'$) which is reciprocal to the degenerate screw system ($\delta$). The degenerate screw system represents the degrees of freedom available to the end-effector transitorily. The existence of the reciprocal screw system implies that there are un-attainable motions of the end-effector at this moment. The un-attainable degrees of freedom are the orthogonal complements of the available twist degrees of freedom, and the ranks of their screw matrices are $6 - n$ and $n$ respectively. The un-attainable degree of freedom are called the elliptic polars of the wrenches which are reciprocal to the twist freedoms, and their screw matrices have the same rank. The relationship of screw systems of twists and wrenches is summarized from reference [42] and is shown in Figure 2.
The reciprocal screw system may be a one-, two-, three-, or even a four-system depending upon the number of degrees of freedom lost by the robot arm. The order of the reciprocal screw system equals 6 less the rank of the degenerate screw matrix. For example, if $\xi'$ is a single screw axis, the end-effector's freedom is reduced from 6 to 5, and the robot arm is at a "simple" special configuration. The screw axes available for the end-effector are those which are linearly dependent on $\xi$, comprising a five-system with $\xi'$ as their common transversor. If one applies the reciprocal condition to Equation (16), one gets:

$$h = a \tan \alpha - h'$$

(22)
This equation reveals some other properties:

1. Given a pair of parameters \((a, \alpha)\) of a line in space, the pitch \(h\) of a screw along this line is uniquely determined.

2. If the end-effector needs a pure translational velocity \((h=\infty)\), then it can only move in the direction perpendicular to \(\$'\) \((\alpha=90^\circ)\).

3. If \(a=0\) and \(\alpha=90^\circ\), then \(h\) can be any value. i.e. the end-effector can twist with any pitch along any axis intersecting \(\$'\) at right angle.

4. If \(h'=\infty\), the degenerate screw system is the "special five-system" defined in reference [24]. In this case, we may freely take \(\alpha=90^\circ\) or \(h=\infty\), i.e. the end-effector can move in pure translational velocity in any direction, and can twist about any screw with any pitch that lies in any plane perpendicular to the direction of \(\$'\).

5. If \(h'=0\), the end-effector is at the extreme reach of the robot arm in the direction of \(\$'\) [25].

Special configurations of a closed-loop spatial mechanism had been categorized into "stationary configurations" and "uncertainty configurations" in reference [24]. By analogy with the definition of special configurations of closed-loop spatial mechanisms, we may imagine a hypothetical screw which joins the end-effector and the fixed frame to make the open chain closed. If the end-effector's screw axis is inactive and extracted from the 7×6 screw matrix, a stationary configuration occurs when the remaining screw matrix (now, the Jacobian matrix) becomes singular. All the special configurations which make the rate control algorithms break down are then no more than the
stationary configurations with respect to the end-effector's screw.

However, special configurations of a manipulator may not be simple at all. They can be very complex and it is sometime very difficult to discover all the special configurations. The singularity of the screw matrix should be interpreted carefully so as to reveal exactly how a robot arm has moved into a special configuration, and then how to get out of the singularity.
CHAPTER III

WORKSPACE GENERATION

3.1 Properties of Industrial Robot Type Geometries

Industrial robots and manipulators have been widely used in industry for various tasks such as welding, painting, or assembling. Despite the variety of robots used in industry, they have many geometric properties in common. Their geometry and structure should be of optimal design which is a compromise between geometrical compactness and structural stiffness, while keeping the maximum achievable workspace volume. Their orientation capability should, likewise, be optimized. In other words, industrial robots should be well designed according to the optimization criteria, and their geometries must be "special" rather than "general." Therefore, a relatively simple and efficient method for design and evaluation of the geometric properties of industrial type robots is desirable.

As many investigators have remarked [8,15,17,19], the manipulator structure can be subdivided into a regional structure and an orientation structure. Broadly speaking,
for a six degree-of-freedom manipulator, the regional structure, which consists of the inboard three joints and their associated members, determines the workspace shape and volume. The orientation structure, which consists the outboard three joints and members, determines the orientation capability of the end-effector of a manipulator. Some special geometric properties of industrial robot type manipulators can be characterized, according to geometric optimization criteria, by considering these two parts of the system:

**Regional structure:**

It has been pointed out in reference [19] that, for a 6-R manipulator, the workspace volume is maximized when joint axis 1 and 2 intersect orthogonally, and joint axis 3 is parallel to joint axis 2. If only prismatic joints are used, the first three joints should be mutually orthogonal. Since prismatic joints do not permit rotation, their use is limited to the regional structure. In addition, the arrangement of successive joint axes to be either normal to, or parallel to, one another can greatly reduce the computational effort, which will be beneficial for the real time control of the robot.
Orientation structure:

It has been shown [5,19] that, for optimal hand dexterity, the outboard three joints should successively intersect at 90 degrees; joint 4 and joint 6 should be able to rotate through 360 degrees; and joint 5 should be able to rotate through at least 180 degrees. That is, the orientation structure is optimal when the three wrist joint axes are orthogonal and concurrent.

Although the orientation of the hand and the positioning of the reference point are almost always coupled in industrial robots, the reachable workspace can be approximately discovered by exercising the regional structure. If, as has been shown to be optimal [19], the outboard three joint axes are concurrent and the reference point is placed at the point of concurrency, the positioning capability can be totally determined from the regional structure only, and is not affected by the orientation structure. In this situation, the reachable workspace may coincide with the dexterous workspace,
joints are kinematically equivalent to a spherical joint centered on their concurrency point.

Therefore, an ideal structure of an industrial robot can be represented as being of JJJS type. Here J stands for a joint which can be either prismatic (P) or revolute (R); and S is a ball joint which represents the optimal orientation structure. Since each of first three joints can be either prismatic or revolute, there are a total of eight possible manipulator geometries which fall within the restrictions we have postulated. Table 1 lists all the eight possible manipulator geometries with suggested joint parameters. The notation for joint parameters follow that of reference [5], and is shown in Figure 5. It is worthy of note that, in order to minimize the occurrence of voids or holes within the workspace, it is desirable to make the link length zero for a prismatic joint, while the offset of the member is preferred to be zero for consecutively parallel revolute joints. It has been found that, in a recent survey, almost all the commercially available industrial robots belong to this category. Figure 3 lists the geometries of some commercial available industrial robots.
Table 1: Possible Geometries for Industrial Robot

<table>
<thead>
<tr>
<th>Joint Type</th>
<th>Previous</th>
<th>Next</th>
<th>$a_1$</th>
<th>$r_1$</th>
<th>$a_2$</th>
<th>$\theta_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-P</td>
<td>0</td>
<td>2-P</td>
<td>0</td>
<td>variable</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-R</td>
<td>0</td>
<td>variable</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1-R</td>
<td>0</td>
<td>2-P</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-R</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>variable</td>
</tr>
<tr>
<td>2-P</td>
<td>1-P</td>
<td>3-P</td>
<td>constant</td>
<td>variable</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-R</td>
<td>constant</td>
<td>variable</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1-R</td>
<td>1-P</td>
<td>3-P</td>
<td>constant</td>
<td>variable</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-R</td>
<td>constant</td>
<td>variable</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2-R</td>
<td>1-P</td>
<td>3-P</td>
<td>constant</td>
<td>constant</td>
<td>90</td>
<td>variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-R</td>
<td>constant</td>
<td>constant</td>
<td>0</td>
<td>variable</td>
</tr>
<tr>
<td>1-R</td>
<td>1-P</td>
<td>3-P</td>
<td>constant</td>
<td>constant</td>
<td>90</td>
<td>variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-R</td>
<td>constant</td>
<td>constant</td>
<td>0</td>
<td>variable</td>
</tr>
<tr>
<td>3-P</td>
<td>2-P</td>
<td>4-R</td>
<td>constant</td>
<td>variable</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-R</td>
<td>constant</td>
<td>variable</td>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>3-R</td>
<td>2-P</td>
<td>4-R</td>
<td>constant</td>
<td>constant</td>
<td>0</td>
<td>variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-R</td>
<td>constant</td>
<td>constant</td>
<td>90/0</td>
<td>variable</td>
</tr>
<tr>
<td>4-R</td>
<td>3-P</td>
<td>5-R</td>
<td>constant</td>
<td>constant</td>
<td>90</td>
<td>variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3-R</td>
<td>constant</td>
<td>constant</td>
<td>90</td>
<td>variable</td>
</tr>
<tr>
<td>5-R</td>
<td>4-R</td>
<td>6-R</td>
<td>constant</td>
<td>constant</td>
<td>90</td>
<td>variable</td>
</tr>
<tr>
<td>6-R</td>
<td>5-R</td>
<td>0</td>
<td>constant</td>
<td>0</td>
<td>constant</td>
<td>variable</td>
</tr>
</tbody>
</table>

Notations: $a_1$ = link length, $r_1$ = offset, $\theta_1$ = joint angle, $\phi_1$ = twist angle.
Figure 3: Some Typical Industrial Robot Type Geometries
3.2 Workspace Generation

3.2.1 Geometric Arrangement of Joint Parameters

For a JJJS industrial type robot, the orientation of the hand and the positioning of the reference point are weakly coupled, and the reachable workspace can be determined by the regional structure alone. It is also easy to see that the workspace for a JJJS manipulator is symmetric about axis 1. Thus the workspace boundary generating curves can be described by the extreme motion of the remaining kinematic chain (the -JJS chain) on a plane containing joint axis 1. This generating plane can be identified as the XZ plane if we let joint axis 1 be coincident with the z-axis of the fixed reference frame. This is the circular projection of the workspace into the XZ plane as defined by Lee [20]. Figure 4 shows the two typical planes for plotting the workspaces of manipulators whose geometries belong to this category.

The whole workspace is obtained by rotating the extreme motion trajectory about axis 1. This is also true when the first joint is prismatic except that the generating plane is now perpendicular to axis 1, and it is the XY plane of the fixed reference frame. Thus the workspace generating algorithm must trace the generating curves of the various possible -JJS chains on the generating plane.
In this study, the manipulator joint parameters are arranged to keep the kinematic chain on the generating plane for its entire motion. The extra freedoms (yaw and roll) of the equivalent spherical joint are not considered since the yaw freedom will allow the hand reference point to move into or away from the generating plane, whereas the roll freedom produces spin of the hand. They will not affect the
generation of the workspace boundary in the generating plane. In summary, the joint parameters are restricted as follows: either the length or the joint offset (or both) of any given link is zero. All twist angles of revolute joints are either zero or 90 degrees. And all joint angles of prismatic joints are also either zero or 90 degrees.

It is a convenience to normalize the length of the manipulator to unity. That is

$$\sum_{i=1}^{6} (a_i + r_i) = 1$$

(23)

where $a_i$ is the link length of member $i$, $r_i$ the offset length of member $i$. Remember that one or the other of $a_i$ or $r_i$ is zero for each link. If the first joint is revolute, the link length of this joint is conveniently set to zero since it does not affect the profile of the workspace in the generating plane.

3.2.2 Reciprocal Conditions

The workspace boundary generation algorithm is based on the fact that a manipulator is always in a singular configuration when the hand reference point of the robot is on the workspace boundary. The configuration is such that the joint axes are all reciprocal to a zero pitch wrench axis [2,7,25]. It is easy to see that if any joint axis is
not reciprocal to the applied zero pitch wrench (a force), work will be done causing the joint to move until there is no moment about the joint. In this situation, the robot-arm is in the extreme reach position along the line of the applied force, and all the joint axes are reciprocal to the applied force.

Equation (16) shows the reciprocal condition of joint axis $\mathbf{h}$ and the wrench axis $\mathbf{h}'$.

$$(h + h') \cos \alpha - a \sin \alpha = 0$$

Since the pitch of the applied wrench is zero, i.e. $h' = 0$, then

$$h \cos \alpha - a \sin \alpha = 0$$

(24)

or

$$\tan \alpha = \frac{h}{a}$$

(25)

To apply this equation, one may consider two types of joint commonly used in the industrial robot: a revolute joint and a prismatic joint.

1. Revolute Joints

For revolute joints, $h = 0$, then

$$a \sin \alpha = 0$$

(26)
The condition will be met when $a=0$ or $a=0^\circ$ or $180^\circ$. i.e. either the wrench axis intersects the joint axis or the wrench axis is parallel to the joint axis.

2. Prismatic Joint

For prismatic joints, $h=\infty$, then

$$\tan a = \infty$$

The condition is satisfied when $a=90^\circ$. i.e. the wrench axis should be perpendicular to the joint axis.

The reciprocal conditions for both revolute joints and prismatic joints are graphically shown in Figure 5.

3.2.3 Workspace Generation Algorithm

If one applies a force (a wrench of zero pitch) in any direction to the reference point on the end-effector of a robot, the robot-arm will move in that direction until it reaches an extreme position. Considering only the kinematic chain on the generating plane, all the joint axes (joints JJS) must intersect the wrench axis. They belong to a degenerate screw system, and the robot loses a degree of freedom. This is a geometrically singular position. The screw system remains unchanged when the kinematic chain moves in the generating plane about joint 2. Therefore, the
Figure 5: Reciprocal Conditions of Joints

points on the trajectory of the reference point all represent singular positions. The set of all points generated by a motion sequence may be called a singular surface. Other segments of the surface are obtained by the
following procedure. Let us assume that the motion of the kinematic chain is caused by rotation of the applied force vector, which continuously rotates in a uniform direction. If joint 2 is moved to its joint limit and is then inactive, the remaining kinematic chain is kept moving in the direction of rotation of the applied force. This is accomplished by testing the direction of the moment, created by the applied force, acting on joint 3. Joint 3 is movable if the direction of the acting moment is within its permitted motion range. In this case, the wrench axis is still reciprocal to all the active joint axes, and the generated trajectory is then guaranteed to be at the extreme reach of the remaining kinematic chain. If all the three joints are moved to either of their limits, the wrench axis then rotates itself to change the direction of the applied force. In this way, the kinematic chain may move back to the opposite joint motion limits by testing the directions of the acting moment again. The workspace boundary is then completed when all joint variables return to their original values.

In order to ensure that the generated motion trajectories satisfy the condition for a boundary surface, it is necessary to keep all the active joint axes reciprocal to the wrench axis during execution of a motion sequence. This can be done by assigning a motion priority for each
joint. Since the motion about the most inboard joint will not change the reciprocal configuration, the most inboard joint has highest priority. i.e. the joint 2. It is followed by joint 3, and joint 4 has the lowest motion priority. The algorithm for generation of the workspace boundary is flow-charted in Figure 6.

![Flowchart of Algorithm for Workspace Boundary Generation](image)

**Figure 6:** Algorithm for Workspace Boundary Generation

In this way, since the workspace boundaries may be generated when one or two joints are inactive due to joint limits, the algorithm allows generation of re-entrant
surfaces within the workspace. These re-entrant singular surfaces play an important role in controllability, and will be discussed later. The boundary generation algorithm also presents no problem in handling kinematic chains containing prismatic joints. It can even be expanded to find the working boundary of a planar kinematic chain with an arbitrary number of links.

It should be noticed that, unless the wrench axis is reciprocal to all the joint axes, the workspace generation condition is not met. The wrench will create moments about those joints axes which are not reciprocal to the wrench axis. The moments will cause those joints to move until the reciprocal condition is satisfied or until they are inactive due to reaching joint limits. Under this circumstance, one should always check the reciprocal condition of each joint before a motion sequence is executed.

After each joint has been tested for movability, a flag can be set. The execution of motion sequences is done by checking the flags according to the order of motion priorities. Once a motion has been executed, the flags are all reset for a new geometric configuration. Care should be taken when the link members are in an unstable configuration (folded or flattened), the sign of resultant moment may be reversed for the inner joints depending upon the direction of the force applied.
3.3 Examples

To illustrate the capability of this workspace generation algorithm, four industrial robots, which are very different in geometry, have been selected as examples. The four industrial robots will be used as examples throughout this study. Since the length of the end-effector is task dependant, its length may vary from case to case. Also, the workspaces of industrial robots provided by manufacturers are presented in different ways. For easy comparison, the length of the end-effector is standardized to be 15% of the total manipulator length in the following examples. For robots with three prismatic joints, the length of the end-effector is set to zero since it is usually relatively small when compared to the whole structure.

Example 1: Cincinnati Milacron T3-746

Figure 7 shows the workspace geometry of a typical industrial manipulator - Cincinnati Milacron T3-746, a general purpose 6-R robot-arm. The lengths for the 2nd and the 3rd links are 44 inches and 55 inches respectively. The first joint can rotate through 270 degrees. The rotation ranges for joints 2, 3, and 4 are 60 to 147, 30 to 135, and -119 to 119 degrees respectively. The outer boundary of the workspace of this robot is a portion of a sphere.
Example 2: Reis Machine Corp. RR-625/650

Figure 8 shows the workspace shape of the Reis Machine Corporation RR 625/650 industrial robot, an 1P-5R manipulator. The travel length of the first joint is 48 inches. The lengths of links 2 and 3 are 28 inches and 20 inches respectively. The motion ranges of joints 2, 3, and 4 are -90 to 180, -175 to 175, and -135 to 135 degrees respectively. The outer boundary of the workspace of this robot is a part of a cylindrical column.
Figure 8: Workspace Geometry of Reis RR-625/650

Example 3: Citizen Robotics Corp. M-100

Figure 9 presents the workspace boundary of the Citizen Robotic Corporation M-100. The link lengths for the 2nd and 3rd links are 17.4 inches and 18 inches respectively. The motion ranges for the first four joints are 300 degrees for the 1st joint, -45 to 225 degrees for the 2nd joint, -90 to 90 degrees for the 3rd joint, and -45 to 45 degrees for the 4th joint. The workspace of this robot is a spherical shell.
Example 4: Cincinnati Milacron T3-886

Figure 10 shows the workspace boundary in both the generating plane and the horizontal plane of the Cincinnati Milacron T3-886. This is a general purpose 3P-3R industrial robot. The regional structure of this robot consists of three orthogonal prismatic joints for higher positional accuracy and more rigidity in structure for carrying higher pay loads in the end-effector. The travel lengths of the first three joints are 40", 36", and 40" respectively. The workspace of this robot is a rectangular prism.
3.4 Hand Orientation Capability

The orientation capability of the end-effector is a very important operational characteristic of a manipulator. Simply tracing the workspace boundary does not guarantee that the end effector can be placed with the reference point coinciding with a given point in the workspace with a desired orientation. This problem is treated in this study by providing a capability for tracing workspace profiles with the end-effector in a specific orientation. The algorithm for tracing the workspace for a specified hand orientation is very simple. Since the end-effector now
should not react to the direction of the applied force, the force is then applied to the joint 3, the joint connecting the end-effector to the inboard members. The same algorithm is then used as in Figure 6 except that there is no need to set and check flag 3 for joint 4. However, the workspace of a robot with a specific hand orientation generated by this algorithm applies only to hand orientation with respect to the generating plane. Once the kinematic chain sweeps about joint axis 1, the hand orientation is no longer the original orientation with respect to the fixed reference frame.

Figure 11 shows the workspace of a manipulator with three links in the generating plane and its third link at zero degree hand orientation with respect to the generating plane. The workspace shown in this figure is essentially the workspace of a manipulator with only two identical links except that there is a translational relationship between the two workspaces. The workspace of the former can be obtained by translating the workspace of the latter the length of link 3 in the direction of its hand orientation. The workspace of the manipulator with any other hand orientation with respect to the generating plane can be obtained by moving the original workspace around a circle with radius the length of link 3 on the plane.
Figure 11: Workspace with Hand Orientation
CHAPTER IV

VOLUME AND MOMENT COMPUTATION

Once the workspace boundary is traced in the generating plane, the workspace volume is then obtained as follow:

\[ V = A_g L \]  

(28)

where \( A_g \) is the area inside the generating curve and \( L \) is the motion range of the first joint. If the first joint is a revolute, \( L = \bar{r} \theta \), where \( \bar{r} \) is the distance of the centroid of that area from the 1st axis, and \( \theta \) is the sweep angle of the 1st joint. If the first joint is prismatic, \( L \) is the travel length of that joint. However, this equation is not convenient for use because there are difficulties in computing the areas of the segments as well as the locations of their centroids. Fortunately, a property of the workspace boundary is that it is a piecewise smooth curve composed of circular arcs and/or straight line segments. This makes it possible to use an elegant computation technique based on Gauss' divergence theorem. This theorem allows a closed form formulation for workspace volumes, their centroids, and higher order moments of inertia.
4.1 Gaussian Divergence Theorem

The Gauss' divergence theorem states that, for a closed bounded region of space $T$, whose boundary is composed of a piecewise smooth surface $S$, if $U(P)$ is a vector function of an arbitrary point, $P$, and $U(P)$ is continuous up to its first derivative in a domain containing $T$, then

$$
\iiint_T \nabla \cdot U \, dV = \iint_S U \cdot n \, dA
$$

where $n$ is an unit vector in the direction of the outward normal of $S$ with respect to $T$. This formula makes it possible to transform from a volume integral into a surface integral or from a surface integral into a contour integral.

4.2 Boundary Generated by an RJJS Chain

4.2.1 Case A: Revolute Joint at Joint 2 or 3

For a segment of the generating curve which is a circular arc, which may result from the use of a revolute at joint 2 or 3, the use of this relationship to compute centroid locations and volumes (Equations 30-36) had been presented in reference [9]. Those results are summarized and included here for completeness.

Referring to Figure 12, the sub-volume bounded by a circular arc boundary is a portion of torus. For the
conical flank, $\mathbf{p} \cdot \mathbf{n} = 0$, ao only the circular arc segment needs to be considered. On the arc, let $\mathbf{p} = \mathbf{p}_c + r_a$, $\mathbf{n} = r_a$, and $d\mathbf{s} = \text{rad} \psi$. So

$$\mathbf{p} \cdot \mathbf{n} = \mathbf{p}_c \cdot r_a + r_a = p_c \cos(\psi - \phi_c) + r_a$$

Figure 12: Geometry of Sub-volume Bounded by Torus Portion

(1) Area

Let $\mathbf{U} = \mathbf{p} = x_1 + z_1$, then $\mathbf{v} \cdot \mathbf{U} = 2$

$$\iint \mathbf{v} \cdot \mathbf{U} d\mathbf{A} = 2 \iint d\mathbf{A} = 2 A_s = \int \mathbf{p} \cdot \mathbf{n} d\mathbf{s}$$

$$A_s = r_a \left[ p_c \sin(\psi - \phi_c) + \sin(\phi_c - \psi_0) + r_a (\psi - \psi_0) \right] / 2$$

(30)
(2) Volume

Let \( \mathbf{V} = \mathbf{p} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} \),

\[ \mathbf{V} = \frac{\partial}{\partial x} i + \frac{\partial}{\partial y} j + \frac{\partial}{\partial z} k \]

then \( \mathbf{V} \cdot \mathbf{U} = 3 \). Hence from Equation (29)

\[ \iiint \mathbf{V} \cdot \mathbf{U} \, dV = 3V = \iint \mathbf{P} \cdot \mathbf{n} \, dA \]

\[ V = \frac{1}{3} \iint \mathbf{P} \cdot \mathbf{n} \, dA \]

\[ = \frac{r_a(\theta_1 - \theta_0)}{12} \left[ 4p_c^2 \cos \phi_c \left[ \sin(\psi_1 - \phi_c) - \sin(\psi_0 - \phi_c) \right] \right. \]

\[ \left. + 6r_a p_c (\psi_1 - \psi_0) \cos \phi_c + 4r_a^2 (\sin \psi_1 - \sin \psi_0) \right] \]

\[ + r_a p_c \left[ \sin(2\psi_1 - \phi_c) - \sin(2\psi_0 - \phi_c) \right] \] (31)

(3) The radius of the centroid of area \( A_s \) with respect to the joint axis \( l \) can be evaluated by means of the equation:

\[ \overline{r} = \frac{V}{A_s(\theta_1 - \theta_0)} \] (32)

(4) The position of the area centroid in the direction of the joint axis \( l \) (z-axis) is given by:
let \( U = zp = z(x'i + y'i + zk) \),

\[
V = \frac{\partial}{\partial x} i + \frac{\partial}{\partial z} k
\]

then \( V \cdot U = 3z \), \( \iint V \cdot \mathbf{n} dA = 3zA = \int \mathbf{p} \cdot \mathbf{n} dA \)

\[
\bar{z}_A = \frac{\frac{1}{3} \int zp \cdot \mathbf{n} dA}{A_s}
\]

\[
= \frac{r_a}{12} \left[ 4r_c^2 \sin \phi_c \left( \sin(\psi_i - \phi_c) - \sin(\psi_o - \phi_c) \right) \right. \\
+ 6r_c (\psi_i - \psi_o) \sin \phi_c - 4r_c \left( \cos \psi_i - \cos \psi_o \right) \\
- \left. r_a p_c [\cos(2\psi_i - \phi_c) - \cos(2\psi_o - \phi_c)] \right], \quad (33)
\]

and

\[
\bar{z} = \frac{\bar{z}_A}{A_s} \quad (34)
\]

(5) The position of the volume centroid in the z-direction is obtained from the following formulations:

let \( U = zp = z(x'i + y'i + zk) \), \( V \cdot U = 4z \).

\[
z = p_c \sin \phi_c + r_a \sin \psi, \quad \text{and}
\]

\[
\iiint V \cdot U \, dV = 4Vz = \iint zp \cdot \mathbf{n} \, dA,
\]

then
\[
V_z = \frac{1}{4} \int \int z \mathbf{p} \cdot \mathbf{n} \, dA
\]

\[
V_z = \frac{r_a (\theta_1 - \theta_0)}{48} \left[ 12r_a p_c^2 (\psi_1 - \psi_0) \sin 2\phi_c \\
-15r_a^2 p_c [\cos (\psi_1 - \phi_c) - \cos (\psi_0 + \phi_c)] \\
-3r_a (r_a^2 + p_c^2) \cos 2\psi_1 - \cos 2\psi_0 \\
+6p_c^3 \sin 2\phi_c [\sin (\psi_1 - \phi_c) - \sin (\psi_0 - \phi_c)] \\
-r_a^2 p_c [\cos (3\psi_1 - \phi_c) - \cos (3\psi_0 - \phi_c)] \right], \quad (35)
\]

and

\[
\bar{z} = \frac{V_z}{V} \quad (36)
\]

(6) If the first joint has motion limits, the workspace is no longer a complete axisymmetric volume. The position of the volume centroid in the \( x \)-direction can be similarly obtained as follows:

\[
\bar{x} = \frac{V_x}{V} \quad , \quad (37)
\]

where
\[ \bar{V}_x = \int \int x \, dv = \frac{1}{4} \int \int x \, dA \]

\[ = \int_{\theta_o}^{\theta_1} \int_{\psi_o}^{\psi_1} (p_c \cos \phi_c + r_a \cos \psi)^2 \cos \theta [p_c \cos (\psi - \phi_c) + r_a] \, d\psi \, d\theta \]

\[ = \frac{\sin \psi_1 - \sin \psi_o}{48} \left[ 3r_a^3 \left( \sin 2\psi_1 - \sin 2\psi_o + 2(\psi_1 - \psi_o) \right) \right. \]

\[ + 3p_c^2 r_a \sin 2\phi_c (\cos 2\psi_o - \cos 2\psi_1) \]

\[ + 6p_c^2 r_c \cos^2 \phi_c [4(\psi_1 - \psi_o) + (\sin 2\psi_1 - \sin 2\psi_o)] \]

\[ + 12p_c^3 \cos^2 \phi_c \left[ \sin (\psi_1 - \phi_c) - \sin (\psi_o - \phi_c) \right] \]

\[ + p_c r_a \sin^2 (3\psi_1 - \phi_c) - \sin^2 (3\psi_o - \phi_c) \]

\[ + 21p_c r_a^2 \sin (\psi_1 - \phi_c) - \sin (\psi_o - \phi_c) \]

\[ + 12p_c r_a^2 \cos \phi_c [\sin \psi_1 - \sin \psi_o]. \]  

(38)

(7) In order to get the moment of inertia about axis 1, put

\[ \bar{U} = r^2 p = r^2 (r \hat{r} + r \hat{\theta} + z \hat{z}). \]

In cylindrical polar coordinates,

\[ \bar{V} = \frac{\partial}{\partial r} r \hat{r} + \frac{\partial}{\partial \theta} \theta \hat{\theta} + \frac{\partial}{\partial z} z \hat{z} \]

so, \( \bar{V} \cdot \bar{U} = 5r^2 \) and
Hence, referring to Figure 12; and applying Equation (39):

\[
I_z = \frac{1}{5} \int r^2 \rho \cdot dA = \frac{1}{5} \int_{\theta_o}^{\theta_1} \int r^2 \rho \cdot r \, d\theta \, d\psi
\]

\[
= (\theta_1 - \theta_0) M_r,\quad (40)
\]

where

\[
M_r = \frac{1}{5} \int_{\psi_o}^{\psi_1} r^3 \rho \cdot \alpha \, d\psi.
\]

Integration of \( M_r \) using the geometry shown in Figure 12 yields:
\[
M_r = \frac{r_a}{240} \left[ 6p_c r_a^2 \right. \\
\left. + 3p_c r_a \right] \left[ 3p_c^2 + 9r_a^2 \right] \left[ \sin(\phi_c + 2\psi_1) - \sin(\phi_c + 2\psi_0) \right] \\
+ 3p_c r_a \left[ 6p_c^2 + 9r_a^2 \right] \left[ \sin(\phi_c - 2\psi_0) - \sin(\phi_c - 2\psi_1) \right] \\
+ 6p_c^2 \left[ 3p_c^2 + 12r_a^2 \right] \left[ \sin(2\phi_c - \psi_0) - \sin(2\phi_c - \psi_1) \right] \\
+ 2r_a^2 \left[ 3p_c^2 + 2r_a^2 \right] \left[ \sin 3\psi_1 - \sin 3\psi_0 \right] \\
+ 30p_c r_a (\psi_1 - \psi_0) \left[ p_c^2 \cos 3\phi_c + 3(p_c^2 + r_a^2) \cos \phi_c \right] \\
+ 6 \left[ 3p_c^4 + 21p_c^2 r_a^2 + 6r_a^4 \right] (\sin \psi_1 - \sin \psi_0) \\
+ 6p_c^4 \left[ \sin(4\phi_c - \psi_0) - \sin(4\phi_c - \psi_1) \right] \\
+ 1.5p_c r_a^3 \left[ \sin(4\psi_1 - \phi_c) - \sin(4\psi_0 - \phi_c) \right] \\
+ 9p_c^3 r_a \left[ \sin(3\phi_c - 2\psi_0) - \sin(3\phi_c - 2\psi_1) \right] \\
+ 6p_c^2 r_a^2 \left[ \sin(3\psi_1 - 2\phi_c) - \sin(3\psi_0 - 2\phi_c) \right]. \tag{41}
\]

(8) The moment of inertia about the y-axis of the fixed frame is

\[
I_y = \int \int \int \rho_y^2 dV, \quad \text{where} \quad \rho_y^2 = x^2 + z^2 = (r\cos \theta)^2 + z^2.
\]

Put

\[
\rho = \rho_y = (x^2 + z^2)(x \dot{y} + y \dot{x} + z \dot{z}), \quad z = p_c \sin \phi_c + r_a \sin \psi,
\]
then \( \mathbf{y} \cdot \mathbf{y} = 5(x^2 + z^2) = 5r_y^2 \), and

\[
I_y = \frac{1}{5} \iiint U \cdot \mathbf{n} \, dA = \frac{1}{5} \iiint (x^2 + y^2) \mathbf{P} \cdot \mathbf{n} \, dA
\]

\[
= \frac{1}{5} \iiint r^2 \cos^2 \theta \mathbf{P} \cdot \mathbf{n} \, dA + \frac{1}{5} \iiint z^2 \mathbf{P} \cdot \mathbf{n} \, dA
\]

\[
= M_r \int_{\theta_\text{lo}}^{\theta_\text{hi}} \cos^2 \theta \, d\theta + M_z,
\]

(42)

where

\[
M_z = \frac{1}{5} \iiint z^2 \mathbf{P} \cdot \mathbf{n} \, dA
\]

is integrated in closed form using the geometry of Figure 12 as follows:
$$M_z = \frac{r_a}{240} [12p_c^2 (p_c^2 + 4r_a^2)\sin 2\phi_c (\cos \psi_o - \cos \psi_1)
- 30p_c^2 r_a^2 [\sin (\psi_1 + 2\phi_c) - \sin (\psi_o + 2\phi_c)]
+ 6p_c r_a (2p_c^2 + 3r_a^2) \sin \phi_c (\cos 2\psi_o - \cos 2\psi_1)
- 3p_c r_a (p_c^2 + 4r_a^2) [\sin (2\psi_1 + \phi_c) - \sin (2\psi_o + \phi_c)]
- 6p_c^4 \sin (\psi_1 - 4\phi_c) - \sin (\psi_o - 4\phi_c)
+ 1.5p_c r_a^3 [\sin (4\psi_1 - \phi_c) - \sin (4\psi_o - \phi_c)]
+ 3p_c^3 r_a [\sin (2\psi_1 - 3\phi_c) - \sin (2\psi_o - 3\phi_c)]
+ 2p_c^2 r_a^2 [\sin (3\psi_1 - 2\phi_c) - \sin (3\psi_o - 2\phi_c)]
+ 30p_c r_a [(p_c^2 + r_a^2) \cos \phi_c - p_c^2 \cos 3\phi_c] (\psi_1 - \psi_o)
+ 6(p_c^4 + 7p_c^2 r_a^2 + 2r_a^4) (\sin \psi_1 - \sin \psi_o)
- 2r_a^2 (\sin 3\psi_1 - \sin 3\psi_o) [3p_c^2 + 2r_a^2], \quad (43)$$

and

$$f_{\psi_1} \cos^2 \theta \, d\theta = (\theta_1 - \theta_o)/2 + (\sin 2\theta_1 - \sin 2\theta_o)/4. \quad (44)$$

(9) The moment of inertia about the x-axis of the fixed frame is
\[ I_x = \int \frac{r_x^2}{y^2 + z^2} dV, \quad \text{where} \quad r_x^2 = y^2 + z^2 = (r \sin \theta)^2 + z^2. \]

Similarly, \( I_x \) can be formulated as:

\[ I_x = M_r \int_{\theta_0}^{\theta_1} \sin^2 \theta d\theta + M_z, \quad (45) \]

where

\[ \int_{\theta_0}^{\theta_1} \sin^2 \theta d\theta = \frac{1}{2} \left( \theta_1 - \theta_0 \right) \left[ \frac{1}{2} \sin 2\theta_1 - \sin 2\theta_0 \right]. \quad (46) \]

or

\[ I_x + I_y = I_z + 2M_z. \quad (47) \]

### 4.2.2 Case B: Prismatic Joint at Joint 2 or 3

For a straight line segment in the generating curve, which may result from the use of a prismatic joint at joint 2 or 3, the geometry is a sub-volume of a cone as shown in Figure 13. Again, from Figure 13, at both flanks \( p \cdot n = 0 \). Only the straight line segment need be considered. Clearly \( p \cdot n = h \), the distance from the origin \( O \) to the line containing the straight line segment. Given the angle \( \alpha \) of the line segment to the horizontal, one can evaluate \( h \) as follows:

Let point \((x_0, z_0)\) be the starting point of the line segment, then
Figure 13: Sub-Volume Geometry Bounded by Conical Portion

\[ r = x_o + l \cos \alpha, \quad z = z_o + l \sin \alpha, \quad ds = dl, \quad dA = r \, d\theta \, dl, \]

\[ \phi = \tan^{-1} \frac{z_o}{x_o}, \quad \text{and} \quad r_p = (x_o^2 + z_o^2)^{1/2}, \text{then} \]

\[ h = OP \sin(\alpha - \phi) = r_p \sin(\alpha - \phi). \quad (48) \]

(1) The area generated by the straight segment is given by:

let \( p = x_i + y_j, \quad V \cdot U = 2, \text{and} \)

\[ \iint V \cdot UdA = \iint 2dA = \int p \cdot nds, \]

then,
The volume of the segment bounded by the straight segment is given by:

let \( p = U = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}, \quad \mathbf{v} \cdot \mathbf{U} = 3, \) and \( dA = r\,d\theta\,dl \)

\[
\iiint_{S} \mathbf{v} \cdot \mathbf{dV} = \iiint_{T} 3\,dV = \iint_{S} p \cdot \mathbf{dA}, \text{ then}
\]

\[
V = \frac{1}{3} \iint_{S} p \cdot \mathbf{dA} = \frac{hL}{3} (\theta_1 - \theta_0)(x_0 + \frac{L}{2} \cos \alpha) . \tag{50}
\]

(3) The radius of centroid \( \bar{R} \) of area \( A_\beta \) with respect to the joint axis 1 of this case is the same as that given in Case A, Equation (32).

(4) The position of the centroid of area in the direction of the first joint axis (z axis) can be obtained from Equation (34) and the following equation:

\[
\bar{z}_{A_\beta} = \frac{1}{3} \int z p \cdot \mathbf{dA} = \frac{hL}{3} (z_0 + \frac{L}{2} \sin \alpha). \tag{51}
\]

(5) The position of the volume centroid in the z-axis
direction can also be calculated from Equation (35) and:

\[ V_\dddot{z} = \frac{1}{4} \int \int \int_{E} \dddot{z} \cdot ndA \]

\[ = \frac{h}{4} \int_{\theta_{1}}^{\theta_{0}} (x_{0} + \frac{h \cos \theta}{2}) (z_{0} + \frac{h \sin \theta}{2}) d\theta d\theta \]

\[ = \frac{h L}{4} ((\theta_{1} - \theta_{0}) (x_{0} z_{0} + \frac{x_{0} L}{2} \sin \alpha + \frac{z_{0} L}{2} \cos \alpha + \frac{L^{2}}{3} \cos \alpha \sin \alpha) \cdot \cos \alpha \sin \alpha) \cdot \cos \alpha \sin \alpha). \tag{52} \]

(6) Likewise, the position of the volume centroid in the x-direction can be evaluated from:

\[ V_x = \frac{h L}{4} ((\theta_{1} - \theta_{0}) (x_{0} z_{0} + \frac{x_{0} L}{2} \sin \alpha + \frac{z_{0} L}{2} \cos \alpha + \frac{L^{2}}{3} \cos \alpha \sin \alpha) \cdot \cos \alpha \sin \alpha). \tag{53} \]

(7) The moment of inertia \( I_z \) about the axis of symmetry is given by Equation (40), where \( M_r \) is different from the previous one and is evaluated as follows:

\[ M_r = \frac{1}{5} \int_{0}^{L} r^{3} P \cdot nd\lambda = \frac{h}{5} \int_{0}^{L} (x_{0} + \frac{h \cos \alpha}{2})^{3} d\lambda \]

\[ = h L (x_{0}^{3} + 1.5 x_{0}^{2} L \cos \alpha + x_{0} L^{2} \cos^{2} \alpha + .25 L^{3} \cos^{3} \alpha) / 5. \tag{54} \]
(8) The moment of inertia about the y axis of the fixed reference frame is given by Equation (42), and \( M_z \) now becomes:

\[
M_z = \frac{1}{5} \iint z^2 \rho \cdot \nabla \, dA
\]

\[
= \frac{1}{5} \int_0^\theta_1 \int_0^L (z_0 + \ell \sin \alpha)^2 h(x_0 + \ell \cos \alpha) \, dl
\]

\[
= \frac{hL}{120} (\theta_1 - \theta_0) \left[ 24x_0^2 + 12Lz_0^2 \cos \alpha \right] + 24x_0z_0 \ell \sin \alpha
\]

\[
+ 8L^2 (x_0 \sin^2 \alpha + z_0 \sin 2\alpha) + 3L^3 \sin 2\alpha \sin \alpha].
\]  

(55)

(9) The moment of inertia about the x axis of the fixed reference frame is given in Equation (45), where \( M_r \) and \( M_z \) are given in Equations (54) and (55) respectively.

4.3 Boundary Generated by PJJS Chain

4.3.1 Case C: Revolute Joint at Joint 2 or 3

If the first joint is prismatic, the generating curve sweeps out a non-circular cylinder, rather than a volume of revolution. The geometry, for a circular arc boundary segment, is shown in Figure 14. Let
Figure 14: Sub-Volume Geometry Bounded by Cylinder Portion

\[ \mathbf{p} = x\mathbf{i} + y\mathbf{j} = \mathbf{u}, \text{ then } \mathbf{u} \cdot \mathbf{u} = 2. \]

And from Figure 14, we have

\[ \mathbf{p} \cdot \mathbf{n} = p_c \cos(\psi - \phi_c) + r_a, \quad ds = r_a \, d\psi, \text{ and} \]

\[ x = p_c \cos \phi_c + r_a \cos \psi, \]

\[ y = p_c \sin \phi_c + r_a \sin \psi \]

(1) The cross sectional area is
\[ \int_a \int b \sin A = \int a \int b dA = \int e \cdot dA \]

\[ A_s = r_a [p_c (\sin \psi_1 - \phi_c) + \sin (\phi_c - \psi_c)] + r_a (\psi_1 - \psi_o) / 2 \]  \hspace{1cm} \text{(56)}

(2) The volume is obtained simply by taking the product of the cross-sectional area and the travel (length \( W \)) of the first joint.

\[ V = A_s W \]  \hspace{1cm} \text{(57)}

(3) The location of the volume centroid is coincident with the location of the area centroid of the generating plane at the mid-point of the travel of joint 1. Thus

\[ \bar{z} = 0 \]

The position of the volume centroid in the \( x, y \) coordinates can be evaluated from the following equations by the method presented in Equation (34):
let \( V = y \widehat{y} = y(x_1 + y_1) \), then \( V \cdot U = 3y \).

\[
\iint V \cdot UdA = \iint 3ydA = \iint y \cdot \widehat{y} dA.
\]

\[
\bar{y}_A = \frac{\iint ydA}{\iint \widehat{y}dA} = \frac{1}{3} \iint y \cdot \widehat{y} dA
\]

\[
= \frac{ra}{12} \left[ 4pc^2 \sin \phi_c [\sin(\psi_1 - \phi_c) - \sin(\psi_0 - \phi_c)]
+ 6ra (\psi_1 - \psi_0) \sin \phi_c - 4ra^2 (\cos \psi_1 - \cos \psi_0) \right]
- ra p_c [\cos(2\psi_1 - \phi_c) - \cos(2\psi_0 - \phi_c)].
\]  

Likewise,

\[
\bar{x}_A = \frac{ra}{12} \left[ 4pc^2 \cos \phi_c [\sin(\psi_1 - \phi_c) - \sin(\psi_0 - \phi_c)]
+ 4ra^2 (\sin \psi_1 - \sin \psi_0) + 6p \cdot r_a \cos \phi_c (\psi_1 - \psi_0)
+ ra p_c [\sin(2\psi_1 - \phi_c) - \sin(2\psi_0 - \phi_c)] \right].
\]  

(4) The moment of inertia with respect to the z axis is:

\[
I_z = \iiint r^2_z dV, \text{ where } r^2_z = x^2 + y^2.
\]

Let

\[
V = r^2_z (x_1 + y_1), \text{ then } V \cdot U = 4 r^2_z .
\]
\[ \iiint \mathbf{V} \cdot \mathbf{dV} = \iiint 4r_z^2 \mathbf{dV} = 4I_z. \] So

\[ I_z = \frac{1}{4} \iiint r_z^2 \mathbf{dA} \]

\[ = \frac{1}{4} \iint x^2 \mathbf{dA} + \frac{1}{4} \iint y^2 \mathbf{dA} \]

\[ = M_x + M_y. \] (60)

\[ M_x \] and \[ M_y \] can be evaluated by integration as in previous cases, giving:

\[ M_x = \frac{raW}{48} \left[ 12p_c^3 \cos^2 \phi_c \left[ \sin(\psi_1 - \phi_c) - \sin(\psi_o - \phi_c) \right] \right. \]

\[ + 3r_a^3 \left[ \sin 2\psi_1 - \sin 2\psi_o + 2(\psi_1 - \psi_o) \right] \]

\[ + 6p_c^2 r_a \cos \phi_c \left[ \sin(2\psi_1 - \phi_c) - \sin(2\psi_o - \phi_c) \right] \]

\[ + 24p_c^2 r_a \cos^2 \phi_c (\psi_1 - \psi_o) \]

\[ + p_c r_a^2 \left[ \sin(3\psi_1 - \phi_c) - \sin(3\psi_o - \phi_c) \right] \]

\[ + 30p_c r_a^2 \cos \phi_c (\sin \psi_1 - \sin \psi_o) \]

\[ + 3p_c r_a^2 \left[ \sin(\psi_1 - \phi_c) - \sin(\psi_o - \phi_c) \right], \] (61)

and
The moment of inertia about the x-axis of the fixed reference frame is:

\[ I_x = \iiint r_x^2 \, dV = \iiint (y^2 + z^2) \, dV \]

\[ = \iiint y^2 \, dV + \iiint z^2 \, dV \]

\[ = M_y + M_z. \]  \hspace{1cm} (63) \]

Since \( dV = A_s \, dz \), we have

\[ M_z = \iiint z^2 \, dV = \int_{-w/2}^{w/2} z^2 A_s \, dz \]

\[ = \frac{1}{12} A_s W^3. \]  \hspace{1cm} (64) \]

\[
M_y = \frac{r_a \, W}{48} [12p_c^3 \sin^2 \phi_c [\sin(\psi_1 - \phi_c) - \sin(\psi_0 - \phi_c)]
+ 3r_a^3 [\sin 2\psi_1 - \sin 2\psi_0 + 2(\psi_1 - \psi_0)]
- 6p_c^2 r_a \sin \phi_c [\cos(2\psi_1 - \phi_c) - \cos(2\psi_0 - \phi_c)]
+ 24p_c^2 r_a \sin^2 \phi_c (\psi_1 - \psi_0)
- p_c r_a^2 [\sin(3\psi_1 - \phi_c) - \sin(3\psi_0 - \phi_c)]
- 30p_c r_a^2 \sin \phi_c (\cos \psi_1 - \cos \psi_0)
+ 3p_c r_a^2 [\sin(\psi_1 - \phi_c) - \sin(\psi_0 - \phi_c)]. \]  \hspace{1cm} (62) \]
Similarly, the moment of inertia about the $y$ axis of the fixed reference frame can be evaluated by

$$I_y = \iiint r_y^2 dV$$

$$= \iiint x^2 dV + \iiint z^2 dV$$

$$= M_x + M_z.$$  \hspace{1cm} (65)

### 4.3.2 Case D: Prismatic Joint at Joint 2 or 3

The generating curve for this case is a straight line segment, and the workspace volume is a triangular prism as shown in Figure 15. Let $p=x_i+y_j=U$, then $\mathbf{v} \cdot \mathbf{U}=2$ and $p \cdot \mathbf{n}=h$, where $h$ is given in Equation (48).

1. **Cross-Sectional Area**

$$A_s = 0.5 \int p \cdot \mathbf{n} ds = 0.5hL.$$  \hspace{1cm} (66)

2. **Volume**

$$V = 0.5hLW.$$  \hspace{1cm} (67)

3. **Position of volume centroid**
\begin{align*}
\bar{z} &= 0, \\
\bar{x} &= \frac{2}{3}(x_o + \frac{L}{2} \cos \alpha), \\
\bar{y} &= \frac{2}{3}(y_o + \frac{L}{2} \sin \alpha). 
\end{align*}
(4) The moments of Inertia about the three principal axes of the fixed reference frame are given in Equations (63), (65), and (60) respectively.

\[ I_x = M_x + M_y, \]
\[ I_y = M_x + M_z, \]
\[ I_z = M_x + M_y. \]

However, the values of \( M_x, M_y, \) and \( M_z \) are different, and they are given as follows:

\[ M_x = \frac{1}{4} \int x^2 \rho \cdot n \, dA = \frac{h \cdot w/2 \cdot L}{4} \int_0^{\pi/2} (x_0 + L \cos \alpha)^2 \, d\theta \]
\[ = \frac{hWL}{4} \left( x_0^2 + x_0L \cos \alpha + \frac{L^2}{3} \cos^2 \alpha \right), \quad (70) \]

\[ M_y = \frac{1}{4} \int y^2 \rho \cdot n \, dA \]
\[ = \frac{hWL}{4} \left( y_0^2 + y_0L \sin \alpha + \frac{L^2}{3} \cos^2 \alpha \right), \quad (71) \]

and
\[ M_z = \iiint_T z^2 \, dV = \int_{-w/2}^{w/2} z A_z \, dz \]

\[ = \frac{hL}{24} \, W^3. \quad (72) \]

4.4 Elimination of Overlapped Volume

As indicated in the previous chapter, the workspace generation algorithm developed in this study allows re-entrancy of the robot-arm to generate contours inside the workspace. The sub-volumes bounded by those re-entrant surfaces are overlapped volumes. In order to get a net workspace volume for a robot-arm, those overlapped sub-volumes should be eliminated before the volume integration is performed. In fact, there are three different types of overlapped sub-volume, created due to excessive motion of joints, which should be removed.

4.4.1 Overlapped Sub-volume Due to Re-entrant Surfaces

Actually, the overlapped re-entrant surfaces are of three types:

1. The total surface is within the workspace volume.
2. The surface is partially inside the workspace volume, and partially on the workspace boundary.
3. The total surface is coincident with the workspace boundaries.

It is very difficult to develop a workspace generation algorithm which can remove the overlapped sub-volume automatically. However, the Gaussian contour integration permits a positive value of area bounded by a contour line along one direction while yielding a negative value of area when integrated along the other direction of the contour. The area bounded by the workspace boundary in the generating plane should be integrated along one direction so that a net positive value of area is obtained. Any other properties of the workspace such as volume and moments of inertia should follow the same sign convention as that of area integration.

The third type of re-entrant surfaces which are multiply traced workspace boundaries should be integrated only once at the consistent direction. The other two types of re-entrant surfaces should be identified interactively by picking up the surfaces totally inside the workspace and surfaces partially on the workspace boundary. The surfaces which are totally within the workspace are then removed. Any surface partially on the workspace boundary should intersect another surface which also has a portion on the workspace boundary. Therefore, there are two intersecting points between two such surfaces in the generating plane. Only one intersecting point in the generating plane is on
the boundary and again should be picked up interactively. Both intersecting surfaces can then be divided into two portions. The portions which lie inside the workspace volume can now be eliminated.

Figure 16: Overlapped Area Due to Re-entrancy

Figure 16 shows the work area of a two-link manipulator. The shaded area can be reached by two robot configurations and is, therefore, an overlapped area. Figure 17 shows the net workspace of the manipulator after removing the
4.4.2 Overlapped Sub-volume Due to Excessive Motion of Joint 2

The second case of an overlapped sub-volume is due to excessive motion of joint 2. This type of overlapping can be visualized in the generating plane. The case is likely to occur when the motion range of joint 2 exceeds 180
degrees and the joint limit of joint 3 is beyond ±90 degrees.

Figure 18: Overlapped Sub-Volume Due to Joint 2

As shown in the shaded area of Figure 18, the "head" of the workspace in the generating plane intersects the "tail" of the workspace. This type of overlapped volume can be removed interactively as discussed previously.
4.4.3 Overlapped Sub-volume Due to Excessive Motion of Joint 1

The third case of an overlapped sub-volume is due to excessive motion of joint 1. This type of overlapping can be visualized in the horizontal plane. It will occur when the motion range of joint 1 exceeds 180 degrees and the workspace volume intersects the axis of joint 1. The overlapped volume can be calculated as the area bounded by the other side of the first joint axis times the length from the area centroid to this axis and the angles \((2\theta_1 - 2\pi)\) as shown in the shaded area of Figure 19.

Figure 19: Overlapped Sub-Volume Due to Joint 1
4.5 Examples

The workspaces of the four industrial robots have been found in Chapter 2. The workspace properties can now be evaluated by the integration algorithm in this Chapter. Notice that the lengths of the end-effectors have been standardized to be 15% of the total manipulator length except for the 3P-3R robot. All the workspace properties are evaluated after removing all the overlapped regions.

4.5.1 Example 1: Cincinnati Milacron T3-746

Figure 20 shows the workspace geometric properties of the manipulator – Cincinnati Milacron T3-746. The total workspace volume is 3,014130 cubic inches. Its normalized volume index (NVI) equals to 0.455. The location of the volume centroid relative to the fixed frame is:

\[ \bar{x} = 0.15324, \quad \bar{y} = 0.0, \quad \bar{z} = 0.22253. \]

The radii of gyration of the workspace volume about the principal axes are: \( r_x = 0.54677, \quad r_y = 0.61260, \) and \( r_z = 0.60125. \)
Figure 20: Workspace Properties of CM T3-746
4.5.2 Example 2: Reis Machine Corp. RR-625/650

Figure 21 shows the workspace volume of the Reis Machine Corporation RR 625/650 industrial robot. The workspace volume is 819,708 cubic inches. The normalized volume index is 0.136. The location of the volume centroid relative to the fixed frame is:

\[ \bar{x} = 0.052336, \quad \bar{y} = 0.052336, \quad \bar{z} = 0.0. \]

The radii of gyration of the workspace volume about the principal axes are: \( r_x = 0.31211, \quad r_y = 0.33535, \quad \text{and} \quad r_z = 0.38838. \)

4.5.3 Example 3: Citizen Robotics Corp. M-100

Figure 22 presents the workspace boundary in two views of the Citizen Robotics Corporation M-100. The workspace volume is 224,802 cubic inches, and its NVI is 0.744. The Figure also shows a big void in the middle of workspace. The location of the volume centroid relative to the fixed frame is:

\[ \bar{x} = 0.0, \quad \bar{y} = 0.0, \quad \bar{z} = 0.017912. \]

The radii of gyration of the workspace volume about the principal axes are: \( r_x = 0.686, \quad r_y = 0.686, \quad \text{and} \quad r_z = 0.696. \)
Figure 21: Workspace Properties of Reis RR-625/650
**Figure 22: Workspace Properties of Citizen M-100**

<table>
<thead>
<tr>
<th>D-Index</th>
<th>Line Lbl</th>
<th>Offset</th>
<th>Jt. Ang.</th>
<th>Twist</th>
<th>Posn</th>
<th>Total Volume</th>
<th>Moment of In.</th>
<th>Cross-Sec. Area</th>
<th>Radius of Gy.</th>
<th>Volumes Centroid</th>
<th>Area Centroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-R</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>196.000</td>
<td>0.167E+61</td>
<td>0.146E+61</td>
<td>0.167E+61</td>
<td>0.167E+61</td>
<td>0.167E+61</td>
</tr>
<tr>
<td>2-R</td>
<td>0.010</td>
<td>0.000</td>
<td>0.000</td>
<td>-45.000</td>
<td>0.000</td>
<td>286.000</td>
<td>0.857E+60</td>
<td>0.868E+60</td>
<td>0.857E+60</td>
<td>0.868E+60</td>
<td>0.857E+60</td>
</tr>
<tr>
<td>3-R</td>
<td>0.020</td>
<td>0.000</td>
<td>0.000</td>
<td>-90.000</td>
<td>0.000</td>
<td>0.167E+61</td>
<td>0.799E+61</td>
<td>0.000</td>
<td>0.799E+61</td>
<td>0.000</td>
<td>0.799E+61</td>
</tr>
<tr>
<td>4-R</td>
<td>0.030</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.167E+61</td>
<td>0.303E+60</td>
<td>---</td>
<td>0.303E+60</td>
<td>---</td>
<td>0.303E+60</td>
</tr>
</tbody>
</table>
4.5.4 Example 4: Cincinnati Milacron T3-886

Figure 23 shows the workspace of Cincinnati Milacron T3-886. The workspace volume is 115,200 cubic inches. The normalized volume index is 0.0072442. The location of the volume centroid relative to the fixed frame is:

\[
\bar{x} = 0.12393, \quad \bar{y} = 0.11530, \quad \bar{z} = 0.0.
\]

The workspace shape of this robot is a rectangular prism shape with radii of gyration about the principal axes: \(r_x = 0.199\), \(r_y = 0.248\), and \(r_z = 0.199\).
Figure 23: Workspace Properties of CM T3-886
CHAPTER V
MANIPULATOR CONFIGURATION AND ASPECT

5.1 Configurations and Aspects of Robot

As can be seen in Figure 7, 8, and 9, the workspace generation algorithm traces the workspace boundaries as well as the re-entrant surfaces. If all the six joints of a manipulator do not reach their joint limits, the algorithm traces a singular surface since a wrench axis is reciprocal to all the 6 joint axes. If any of the 6 joints reaches its joint limit and then becomes inactive, the algorithm will trace the surfaces that are created by the extreme reach of the remaining kinematic chain. Those surfaces can be either on the workspace boundaries or inside the workspace volume. Since a wrench can always be applied normal to those surfaces, those surfaces are not crossable for the robot in the configuration in which they were traced. The non-crossable inside surfaces are defined as the D-surfaces in reference [39]. As shown in Figure 24, the small arrows represent the direction of the applied wrench when the manipulator is in the configuration that traces the
surfaces. The robot-arm then cannot move further in the direction indicated.

Figure 24: Operating Barrier of CM T3-746

All those surfaces traced by the workspace generation algorithm indicate singular positions or operating barriers for a manipulator in a given configuration. However, a general geometry 6-R manipulator can theoretically reach a point within its workspace with as many as 32 different
configurations. Most of the configurations are not attainable due to the limitations of the geometric design and the restrictions of the joint motions. The existence of different configurations capable of reaching points in the workspace can be discovered by the study of manipulator's "aspect."

Definition of an Aspect

For an n jointed manipulator, the coordinate relationship between the hand position \( p \) and the joint variable \( q \) is given in Equation (20) and the relationship between hand velocity and joint rates is given in Equation (21). Let \( D \) be the admissible domain in the joint space

\[ D \in \mathbb{R}^n, \quad D = \{ q | q_{\text{imin}} < q_i < q_{\text{imax}}, [i=1,n] \}, \]

and \( \Delta = \{ \delta_1, \delta_2, \ldots, \delta_k \} \), a set of m-order minors of \( J \) where \( k = \binom{n}{m} \). An aspect \( A \) of a robot is defined as a set of points in \( \mathbb{R}^n \) [41], such that

\[ A \subseteq D \]

\[ \forall q \in A, \quad \delta_i(q) \neq 0, \quad i \leq k \]  \( (73) \)

Therefore, an aspect of a robot can be interpreted as a set of joint variables such that the manipulator can reach points inside the workspace at one configuration without...
hitting a joint limit.

5.2 Workspace Generation by Aspect Method

In cartesian space, if only the positioning of a point is of interest, three parameters can specify the point, then $p \in \mathbb{R}^3$. The reachable workspace of a manipulator is no more than the union of images, by the $f(p)$ operator, of all the robot's aspects in the joint space of the regional structure. Because the boundaries of aspects are hypersurfaces which partition $\mathbb{D}$ into aspects or which limit the aspects, the workspace boundaries can be determined by the boundaries of all the aspects of the robot's regional structure. If the structure of an industrial robot is of an optimal JJJS design, the workspace of such a manipulator can be discussed by the extreme reach of the -JJS chain in the generating plane as discussed in chapter 3. The reachable point $P$ in $\mathbb{R}^3$ is then decomposed to a point $P$ in $\mathbb{R}^2$, i.e. in the generating plane. The situation is similar to a kinematically redundant chain (3 joints) in a two-dimensional plane. Referring to Figure 25, the position of the point $P$ in terms of the joint variables is given by:

$$
\begin{align*}
    x_P &= a_1 \cos \theta_1 + a_2 \cos \theta_{12} + a_3 \cos \theta_{123} \\
    y_P &= a_1 \sin \theta_1 + a_2 \sin \theta_{12} + a_3 \sin \theta_{123}
\end{align*}
$$

where $a$ is the link length, $C$ and $S$ stand for cosine and
The sine function respectively. $\theta_{123}$ means $(\theta_1 + \theta_2 + \theta_3)$. The Jacobian matrix can be obtained by differentiating the transformation matrix:

$$J = \begin{bmatrix}
-vS\theta_{123} - uS\theta_{12} - S\theta_1 & -vS\theta_{123} - uS\theta_{12} & -vS\theta_{123} \\
vC\theta_{123} + uC\theta_{12} + C\theta_1 & vC\theta_{123} + uC\theta_{12} & vC\theta_{123}
\end{bmatrix}$$  \hspace{1cm} (75)

where $v = a_3/a_1$ and $u = a_2/a_1$. Solving the conditions for the 3 sets of second order minors of $J$ to be zeros, one gets:

$$vS\theta_{23} + uS\theta_2 = 0,$$  \hspace{1cm} (76)

$$uS\theta_3 + S\theta_{23} = 0,$$  \hspace{1cm} (77)

and
For a given set of link length ratios $v$ and $u$, one may plot the 3 conditions in joint coordinate space. Each condition gives a family of surfaces in the joint space. Those surfaces will repeatedly occur at intervals of $2\pi$. Limiting the values of joint angles to lie between $-\pi$ and $\pi$, each surface defines a boundary of an aspect. Together with the joint limit surfaces, the total assemblage of aspects of a given manipulator can be discovered.

Figure 26 shows the aspects of a manipulator with 3 links in the joint space. The link length ratios are set to $v=u=1$. In this figure, the 3 families of surfaces (which include the surfaces of $\theta_2=\pm\pi$ and $\theta_3=\pm\pi$) together with the joint limit surfaces $\theta_1=\pm\pi$ totally define 12 aspects. If the boundaries of all the aspects are mapped onto the cartesian space by Equation (74), the workspace of the manipulator can be plotted in the generating plane.

5.3 Example 1: Cincinnati Milacron T3-746

The workspace geometry of the Cincinnati Milacron T3-746 has been discussed in Chapter 3, and was shown in Figure 7. However, it is very interesting to compare the workspace boundaries generated by both methods. The representation of the robot's aspects in joint space is
Figure 26: Aspects of a Manipulator with $u=v=1$

obtained in Figure 27 with the link length ratios $v=1.2487$ and $u=0.3968$. The three conditions in Equations (76)-(78) yield three families of curved surfaces in the cube of joint space. The joint limits are so carefully selected that the manipulator has only three aspects as can be seen in Figure 28. The three aspects are then mapped into the Cartesian space to get their workspaces. The total workspace is then obtained from the union of the workspaces of the three
Figure 27: Joint Space Representation of CM T3-746

aspects. Figure 29 shows the total workspace obtained from the aspect method. The mapping from the joint space to the Cartesian space is multi-valued. For any point in the joint space, there is a point in Cartesian coordinate corresponding to it, but for any point in the workspace, there may be more than one corresponding point in joint space.

The workspace boundary obtained from the aspect method
Figure 28: Aspects of CM T3-746 in Joint Space

is essentially the same as that generated from the algorithm in Chapter 3. Comparing Figure 7 with Figure 29, the aspect method generates more inside surfaces than the re-entrant surfaces generated from the workspace algorithm. It is interesting to notice that all the re-entrant surfaces are coincident to some of the inside surfaces. This is true because each edge of an aspect represents a joint limit or a degenerate m-order minor of the Jacobian matrix. In m dimensional space, the conditions that make m-order minors of the Jacobian matrix singular are those in which the robot
Figure 29: Workspace of CM T3-746 Obtained by Aspects

loses a degree of freedom in the m-dimensional operating space. The corresponding surfaces in the workspace are the sets of points at which the robot loses a degree of freedom either due to joint limits or due to degenerate m-order minors in the Jacobian. Remembering that the workspace algorithm traces the extreme reach of the sub-kinematic chains in the generating plane while a wrench axis is
applied, those surfaces are also created due to joint limits and singularities. Therefore, they belong to the set generated by the aspect method. The only difference between inside surfaces and re-entrant surfaces is that the re-entrant surfaces are non-crossable, whereas the inside surfaces may be crossable for the robot in that configuration. For example, points 15 and 16 in Figure 28 are the operating barrier points in the joint space for which $\theta_4$ is at both limits, $\theta_3$ is at its minimum, and $\theta_2$ is at its maximum. However, both points, when mapped into the cartesian workspace as shown in Figure 29, are inside the workspace boundary. They are no longer the operating barrier points since no wrench can be applied to the corresponding robot configuration without moving the end-effector away from the points. Therefore, the workspace generation algorithm traces fewer, yet more important, re-entrant surfaces.

5.4 Example 2: Reis RR-625/650 Robot

Figure 30 shows the aspects of Reis RR-625/650 robot in the joint space. As mentioned before, the three families of aspect surfaces concentrate on positions in which $\theta_2$ and $\theta_3$ are 0 and $\pi$. Unfortunately, the joint limits of this robot are selected symmetrically relative to the reference frame. There are 12 aspects found in the joint space. When the 12
Figure 30: Joint Space Representation of Reis RR-625/650

aspects are mapped into the Cartesian space, there are too many inside surfaces within the workspace to be identified as the boundaries of aspects. Figure 31 shows the workspace of this robot obtained by the aspect mapping. The workspace boundary is the same as that in Figure 8, but it has very complex inside surfaces. The workspace obtained in this way should be carefully studied so that all aspects can be labelled to identify voids or holes, if any, within the workspace.
If a robot has many aspects in joint space, each aspect may cover different areas of the workspace. The control of the robot from one point of an aspect to another point of another aspect involves change of aspects. The change of aspect will occur on the hypersurfaces which divide the aspects. Those surfaces are created due to the degeneration of the Jacobian minors. Although the degeneration of Jacobian minors (in this case, 2x2 matrix) does not
necessarily make the whole Jacobian matrix (a 6x6 matrix) singular, the tendency is very high. Therefore, the change of aspect may cause the robot move into a singular configuration. Besides, the more aspects in the workspace, the greater the chance that D-surfaces occur, which is not desirable. From this view point, it is better to minimize the occurrence of aspects. This goal can be easily achieved with the help of aspect mapping in the joint space. After the three families of aspect surfaces have been plotted, one may select a less dense hypersurface region for setting the joint limits. Figure 28 is a good example in comparing with Figure 30.
CHAPTER VI

MANIPULABILITY AND SINGULARITY

A manipulator is called locally maneuverable if the rank of the Jacobian is equal to 6 [30]. When the manipulator is in a singular configuration, the Jacobian matrix is degenerate and is not invertible. The manipulator is not locally maneuverable since rate control algorithms do not work in this situation. However, when the Jacobian matrix is ill-conditioned and the robot arm is near a singular configuration, the maneuverability of the robot is very poor. i.e. a great deal of joint effect can only cause a very small velocity output at the end-effector. Sometimes, there may be a situation in which the joint efforts needed to generate a desired hand velocity will be well beyond the capability of the actuators. Therefore, a quantitative measure of how close the robot arm is to a singular state is helpful in controlling the actuators. The quantitative measurement of the controllability based on the robot’s geometry is the so-called “manipulability.” Many recently developed control algorithms take the manipulability as the criterion function to be optimized.
during the operation of the manipulator. Once the manipulability decreases below a pre-set value, the control algorithms should take special action to avoid getting into a special configuration, in order to protect the actuators.

6.1 Singular Value Decomposition

Singular value decomposition (SVD) is a very powerful tool for solving linear systems of the familiar form:

\[ Ax = b \]  

(79)

Here \( A \) is an \( m \times n \) matrix, \( m \) can be greater than, equal to, or less than \( n \). The method provides very useful information about the condition of matrix \( A \). The definition and some basic properties of SVD are summarized from reference [40] as follows:

For a matrix \( A \in \mathbb{R}^{m \times n} \), there exits orthogonal matrices \( U \in \mathbb{R}^{m \times m} \) and \( V \in \mathbb{R}^{n \times n} \) such that

\[ A = UV^T \]  

(80)

where

\[ \Sigma = \begin{bmatrix} S & 0 \\ 0 & 0 \end{bmatrix} \quad \text{an \( m \times n \) matrix} \]

and \( S = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r, \ldots, \sigma_m) \), a diagonal matrix composed of singular values \( (\sigma_1, \sigma_2, \ldots, \sigma_m) \) of \( A \) with \( \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \); \( r \)
is the rank of $A$; If $r < m$, then $\sigma_{r+1}=\sigma_{r+2}=\ldots=\sigma_m=0$. The singular values of $A$ are the positive square roots of the eigenvalues of $A^T A$. Columns of the $U$ matrix are the left singular vectors of $A$, or the orthonormal eigenvectors of $A A^T$, whereas columns of the $V$ matrix are the right singular vectors of $A$, or the orthonormal eigenvectors of $A^T A$. $\sigma_m$ gives a dependable measure of how far a matrix is from matrices of lesser rank.

6.2 Definitions of Manipulability

Some definitions of manipulability have been proposed. They are all based on the algebraic properties of the Jacobian matrix. Among them, the application of the singular value decomposition method to the Jacobian matrix shows the greatest promise.

**Definition of Manipulability**

Yoshikawa [31] proposed

$$w \equiv [\text{det}(JJ^T)]^{1/2} = \sigma_1 \cdot \sigma_2 \ldots \sigma_m$$  \hspace{1cm} (81)  

for a quantitative measure of maneuverability of a robot arm. Clearly, $w$ is proportional to the volume of an $m$ dimensional ellipsoid with the major axes along the eigenvectors of $JJ^T$ (the columns of $U$ matrix) and with the length of the major axes corresponding to $\sigma_1, \sigma_2, \ldots, \sigma_m$. 

respectively. However, the volume of the ellipsoid is sensitive to the scale of the singular values. An alternative method is to measure the shape of the ellipsoid. The shape of the ellipsoid can be detected by the magnitude of $\sigma_m/\sigma_1$. This is the ratio of the smallest length to the largest length of the major axes of the ellipsoid. Therefore, the ratio not only reveals the slenderness of the ellipsoid but also normalizes the scale factor of the singular values. Since the manipulability is a measure of the "nearness" of the Jacobian to degeneracy, we can measure the shape of the ellipsoid as well as its volume.

The quantity $\sigma_m/\sigma_1$ is the reciprocal of the condition number $\text{cond}(J)=|J|\cdot|J^{-1}|$, where $J^{-1}$ is the inverse or the pseudo-inverse of $J$. The manipulability is then re-defined by Togai as [33]:

$$\omega \equiv \frac{1}{\text{cond}(J)} = \frac{1}{|J||J^{-1}|} = \frac{\sigma_m}{\sigma_1} = \frac{\sigma_{\text{min}}}{\sigma_{\text{max}}} \quad (82)$$

From this definition, $\omega$ is normalized to a value $0<\omega<1$. When $\omega=1$, the robot arm is at the best possible condition to operate. The points within the workspace that maximize the manipulability are called "isotropic" points. However, there may be many different robot postures which can reach an isotropic point, each posture may have different value of $\omega$. The worst case in operating a robot is when $\omega=0$. i.e.
when the robot arm is in a singular configuration.

6.3 Singular Configurations of Robot-Arm

As was stated earlier, in three dimensional space, a general purpose robot needs 6 linearly independent joints to provide 6 degrees of freedom. Considering a 6 joint manipulator, if at any instant the six screws happen to belong to a five-system, the rank of the Jacobian matrix is decreased by 1. The robot is then in a singular configuration. In general, for any n screws of a 6 jointed robot, 2≤n≤6, if the n screws happen to belong to a (n-1) order screw system, then the n screws are linearly dependent. The rank of the total 6x6 matrix will decrease by 1. Therefore, the singularities of a robot can be discovered by finding the degenerate sub-screw systems. In general, five screws lie in a linear complex, the lines of four screws belong to a linear congruence, the lines of three screws lie on a regulus, and two screws define a cylindroid. If the lines of five screws can be included in a linear congruence, the five screws are linearly dependent, or if any three screws of a robot belong to a cylindroid, the robot is in a singular configuration.

To discover the singularities of a manipulator, let us consider its the regional structure and orientation structure separately.
Singularities of Orientation Structure

The orientation (wrist) structure consists of three revolute joints for any type of industrial robot. The optimal design is for all three joint axes to be concurrent. Since the pitch of a revolute joint is zero, the three system they define lies in a bundle of lines with the center at the point of concurrency. If the three joints axes belong to a cylindroid (in this case, a planar pencil of screws), the wrist becomes singular. That is, when all three joint axes lie in a plane. This will most commonly happen when joint axis 4 and joint axis 6 become collinear. No matter where the joint 5 is placed, all three joints lie in a plane. However, this is also the case in which two screws belong to a one-system.

Another wrist singularity occurs when joint 6 is placed so that it parallel to joint axes 2, 3, and 4 (If they are so arranged). The four parallel screws define a special three-system (the fifth special three-system in C243). The four joints act as a spatial four-bar linkage, and they define a planar motion perpendicular to the four joint axes. In those two singular cases, joint 5 plays a very important role since it determines the alignment of joint 6. Therefore, the motion range of joint 5 should be limited so that joint 6 will not enter a singular position. Both cases
can be termed as the wrist singularity. Reference [19] presents the detailed arrangements of joint limits for the wrist structure to avoid the wrist singularity.

**Singularities of Regional Structure**

The regional structure of an industrial robot involves three joints which are generally placed in parallel with or orthogonal to one another. If link length 2 is not zero, the regional structure will not encounter a singular configuration. However, when we consider the whole structures, many robots are designed in such a way that joint axis 4 is parallel to the axes of joints 2 and 3. In general, the three joint axes define a three system. There may be a chance that joint 3 is fully extended (or is fully folded). In this case, the three joint axes lie in one plane. The screw system they belong to is a special two-system. This is a degenerate cylindroid. Therefore, the robot-arm loses a degree of freedom. This is a so-called elbow type singularity since joint 3 behaves like the elbow joint in a human arm. An easy way to avoid the elbow type singularity is to limit the rotation range of joint 3 to be slightly greater than 0 degrees to slightly less than 180 degrees.

If joints 1 and 2 are revolutes and intersect at a point, they define a planar pencil of screws. The plane in
which the planar pencil lies contains joint axes 1 and 2 and is called the shoulder plane. If the wrist center happens to lie in the shoulder plane, they totally define a planar field of screws in the shoulder plane along with bundle of screws passing through the wrist center. This is the first special four-system defined in reference [24] with the two reciprocal screw pitches $h_\alpha = h_\beta = 0$. Since the total of five screws belong to a degenerate congruence, the robot is in a singular configuration. This is the, so-called, shoulder type singularity.

However, if the three wrist joint axes are not concurrent, the screw systems are more complex, but the singularities still exist. The shoulder type of singularity is more difficult to eliminate by limiting the rotation range of joints alone. The combination of link lengths and joint limits should be exercised in order to get a singular free shoulder structure while keeping the workspace volume at a maximum. This is possible by the help of the workspace generation algorithm presented in this study, since the shoulder type singularity will occur when the workspace of a manipulator in the generating plane intersects the first joint axis. By systematically studying the effects of link lengths and joint limits of links 2 and 3 on the workspace shape and volume, one may obtain optimal sets of joint parameters which give maximum workspace volume and which are
free from the shoulder type of singularity.

6.4 Manipulability Field in Workspace

As we know, when a robot is in a singular position, the manipulability of the robot is zero. However, it is interesting and worthwhile to study the way in which the manipulability goes to zero when a singular point is reached. A simpler way to study this is to exercise a joint variable while keeping all other joint parameters constant. Figure 32 shows four typical relationships of manipulability vs. a joint variable. The manipulabilities throughout this study are calculated according to the Equation (82). In Figure 32-(a), the singularities occur at both ends of the joint limits. The singular condition is commonly found in joint 3 for an elbow type singularity. To avoid this type of singularity, one may set the joint limits so that the joint will reach its motion limits before the robot encounters the singularities. Figure 32-(b) shows the joint moving the robot into a singular configuration in the middle of its motion. This may be a typical shoulder type singularity. It is very difficult to predict when the shoulder type singularity will occur while exercising only one joint. Figure 32-(c) shows a desirable shape in which the manipulability is highest in the middle of joint limits and is kept constant as long as possible. The singularity
Figure 32: Manipulability vs. A Joint Variable

does not occur during the motion of this joint. Figure 32-(d) shows the manipulability in this case is independent of the joint motion. This is true for the first joint and the last joint of a robot. Since the rotation of the whole
arm about the joint 1 will not change the the relative joint positions. Similarly, rotation about joint axis 6 does not change any position of the preceding joints.

Sometimes we may like to see how the manipulability changes within the workspace. However, many points in the workspace can be reached by the end-effector in many different robot postures. Each robot posture may have a different value of manipulability. This problem can be overcome by plotting the manipulability field in the workspace of a robot's aspect. By exercising joints 2 and 3 of a robot while keeping all other joint parameters constant, the manipulability field can be plotted in the workspace cross-section, e.g. in the generating plane. Figure 33 shows such a manipulability field for the Cincinnati Milacron T3-746 robot in the workspace cross-section. The manipulability is at the maximum around the middle of the workspace. Actually, the manipulability is larger when all revolute joint angles are near 90 degrees and all prismatic joints are at their zero positions. Figure 33-(a) is a two-dimensional plot, in which the contour lines are the manipulability. The manipulabilities from 0.0 to 0.1615 are plotted by 15 equal contour lines. Figure 33-(b) is a three-dimensional plot, in which the height of the "island" is the manipulability. Both figures show that shoulder type singularities occur in the workspace.
15 equally spaced contour lines form 0.0 to 0.1615

Figure 33: Manipulability Field for CM T3-746
where it intersects the first joint axis (near the upper cusp shape of the workspace).

Figure 34 shows the manipulability field in an aspect of the Reis Machine Corporation RR-625/650 robot in the workspace cross-section. The manipulability is at the maximum along the center line of workspace. This is when joint 3 is at 90 degrees. Figure 34-(a) shows the two-dimensional manipulability field, whereas Figure 34-(b) is a three-dimensional plot, in which the height of the "stadium" is the manipulability. The manipulabilities from 0.0 to 0.07128 are plotted by means of 10 equally spaced contour lines, symmetrically distributed on both side of the center line. Both figures show the elbow type singularities occur in the workspace boundaries in which the elbow is fully stretched and is fully folded.

Figure 35 shows the manipulability field of the Cincinnati Milacron T3-886 robot in the workspace cross-section in the generating plane. The manipulability is at a maximum at one edge of the workspace. This is because we use this corner as the origin of the fixed reference frame. The manipulability is at a maximum (in this case, the isotropic point) at the origin where the offsets of all three prismatic joints are zeros. The manipulability decreases gradually when the wrist center travels away from the origin. This is not surprising
10 equally spaced contour lines form 0.0 to 0.07128

Figure 34: Manipulability Field for Reis RR-625/650
15 equally spaced contour lines form 0.7123 to 1.0

Figure 35: Manipulability Field for CM T3-886
because when the wrist center is placed at an infinite distance, the three joint axes of the wrist structure are lines at infinity. In this situation, the first three components of the three wrist screws in Plücker coordinates [L,M,N; P^*,Q^*,R^*] can be normalized to zeros, which are linearly dependent on the first three joint axes. Therefore, the Jacobian of the screw matrix becomes singular when the wrist center is at infinity. This is not likely to happen since no such structure exists, but this explains why the manipulability decreases while the wrist is traveling away from the origin. Figure 35-(a) is a two-dimensional plot. The manipulabilities from 0.7123 to 1.0 are plotted by 15 equally spaced contour lines. Figure 35-(b) is a three-dimensional plot, in which the height of the "cube" is the manipulability. The use of three orthogonal joints as the regional structure and 3 revolute joints as the wrist structure can significantly increase the manipulability and eliminate shoulder and elbow type singularities.

6.5 Penalty Function for Manipulability

As discussed in the previous chapter, there are hypersurfaces within the workspace volume. These are the singular surfaces and the joint limit surfaces. The significance of singular surfaces has been discussed by the study of the manipulability field. Likewise, a robot-arm
loses a degree of freedom when a joint limit is reached. Especially when a robot is in the configuration that it traces a D-surface, the robot-arm can not move through the surface. Therefore, the joint limit surfaces behave like singular surfaces. Unfortunately, all industrial robots have joint limits. In these cases, the control algorithm may try to drive the robot beyond the joint limits. A proper adjustment of manipulability is needed to enable control algorithms which use manipulability as a criterion function for singularity avoidance, to detect closeness to a joint limit. A penalty function is introduced for this purpose. The penalty function is designed to force the manipulability to decrease very rapidly in the vicinity of a joint limit, while keeping the manipulability unchanged when a joint limit is a reasonable distance away. To this end, we propose a penalty function of the form:

\[ P = 1 - \exp\left[-k \sum_{i=1}^{n} \left(\frac{\theta_{i} - \theta_{imin}}{\theta_{imax} - \theta_{imin}}\right)^{2}\right] \]  

where \( P \) is the penalty function to be multiplied by the manipulability, \( k \) is a scale factor which can be freely adjusted to get a desired value of \( P \). Figure 36 shows the effect on the penalty function of different \( k \) values. Figure 36-(a) shows the penalty function for \( k=1 \) applied to a rectangular function. By adjusting the \( k \) value, a desired
shape of penalty function can be obtained. Figure 36-(d) shows an acceptable penalty function with $k=100$.

Figure 36: Penalty Function for Manipulability
6.6 Modified Manipulability Field

Applying the penalty function $P$ to the manipulability calculation, the formulae become:

$$w = P \left[ \det(J J^T)^{0.5} \right],$$  \hspace{1cm} (84)

$$w = P \frac{\sigma_m}{\sigma_1}$$  \hspace{1cm} (85)

The effect of applying the penalty function to the manipulability calculation can be visualized by plotting the manipulability fields. In three dimensional plots in which the height of the graph is the modified manipulability, the highest regions are the regions that control algorithms tend to use most.

Figure 37 shows the modified manipulability field with the penalty function for the Cincinnati Milacron T3-746 robot in the workspace cross-section. The manipulability is again at a maximum around the middle of workspace. Comparing this plot with Figure 33, the manipulabilities around the middle of the workspace are nearly unchanged after applying the penalty function. However, the manipulabilities on the workspace boundaries are forced to zero by the penalty function. Again, Figure 37-(a) shows a two-dimensional plot, whereas Figure 37-(b) is a three-dimensional plot. The modified manipulabilities from
15 equally spaced contour lines form 0.0 to 0.15931

Figure 37: Modified Manipulability Field for CM T3-746
0.0 to 0.1615 are plotted by 15 equal contour lines. Both figures show zero manipulability around the cusp shape of the upper corner of the workspace. This is because there is a shoulder type singularity which occurs around the cusp of the workspace, and the workspace is so small within the cusp. The manipulabilities within this area have been forced to zero.

Figure 38 shows the modified manipulability field of Reis Machine Corporation RR-625/650 robot in the workspace cross-section. Again, the manipulability is at a maximum around the middle of workspace. It is hardly changed after applying the penalty function. Figure 38-(a) shows the two-dimensional plot of the modified manipulability field, whereas Figure 38-(b) is a three-dimensional plot. The modified manipulabilities from 0.0 to 0.07105 are plotted by 10 equally spaced contour lines. In addition to the elbow type singularities which force the manipulabilities at two edges of workspace to be zero, the manipulabilities along the "head" and the "tail" of the "stadium" shape workspace are now forced to zero since joint 2 reaches its limits on these boundaries.

Figure 39 shows the modified manipulability field of the Cincinnati Milacron T3-886 robot. The manipulability is now at a maximum in the middle but near one corner of the workspace. This is due to the corner is at the joint limits
10 equally spaced contour lines form 0.0 to 0.07105

Figure 38: Modified Manipulability Field for Reis RR-650
15 equally spaced contour lines form 0.0 to .822

Figure 39: Modified Manipulability Field for CM T3-886
joints of 2 and 3. The manipulability decreases gradually in the middle of the "platform," and decreases very shapely around the four edges. The modified manipulabilities now from 0.0 to 0.8922 are plotted using 15 equally spaced contour lines.
CHAPTER VII

DISCUSSION AND CONCLUSION

7.1 Significance of This Study

In order to utilize numerical optimization techniques and, indeed, in order to perform rational comparisons between designs, it is necessary to be able to numerically characterize the geometric performance of manipulators. For optimization purposes, in particular, it is necessary to have a relatively small number of numerical measures which provide meaningful information about the workspace geometry. In view of the highly complex geometry of many manipulator workspaces this is not easy. A well established method for characterizing complex shapes is to use measures such as the volume, the location of the center of mass, the moments of inertia about axes through the center of mass, and possibly higher order moments. This technique, for example, is widely used in robotic vision systems, for the identification of parts with irregular shapes. Therefore, the primary objective in the present work is to devise efficient means of calculating these measures. Although
many investigators have published methods of computing the volume of the workspace almost all of these are essentially square counting methods with very limited accuracy, particularly around the edges of the workspace which may have cuspidal geometry. The method presented here is a closed form method in which the accuracy attainable is only limited by the accuracy of the computer used and it is a mathematical formulation which provides a foundation for further theoretical development. The computation of the centroids and the higher moments, as presented here, is a good example of the potential of this approach to the volume calculation for generating additional scientific developments.

7.2 Criteria for evaluation of Workspace Geometry

Several important features included in this study can be used as criteria for characterizing the workspace geometry. They are:

1. The workspace volume:

The value of the workspace volume is a direct evaluation of the design of a manipulator structure. It can be normalized to the volume of a sphere with unit radius. This is the maximum theoretical achievable volume for a manipulator with it total length normalized to unity. The
normalized volume is defined as the "Normalized Volume Index" (NVI) in reference [20].

\[ \text{NVI} = \frac{V}{\frac{4}{3} \pi l^3} = \frac{V}{4.1888} \]  

(86)

This is a kinematic performance index which provides the ratio of actual manipulator workspace volume to the theoretical maximum achievable volume. A manipulator with revolute joints is considered to be more geometrically efficient than one with prismatic joints in the sense of utilizing its total link length.

2. The workspace shape:

The irregular shapes of workspace volumes make it difficult to evaluate kinematic performance of a manipulator quantitatively. The centroid of the volume reveals the geometric center of the workspace. However, many an industrial robot yields a very thin shell workspace which has its volume centroid close to the origin. The use of higher order moments then becomes a useful technique for characterization of workspace shape. The moment of inertia sometimes is represented by a radius of gyration to provide a dimension in length for easy comparison. The radius of gyration is an index of the workspace distribution with respect to the principle axes of the fixed reference frame.
3. **Hand orientation capability:**

Simple calculation of the workspace volume does not mean that the whole volume is available for a specific task which requires the end effector to be at a given orientation. Typically, the workspace with a specified hand orientation may be much smaller than the overall workspace and will even disappear for some combinations of hand orientation and length. This program provides an unique capability for evaluating the available volume for a robot with its hand orientation specified in the generating plane.

4. **Special surfaces within workspace:**

As was stated earlier, the algorithm for finding the workspace boundary in this study allows for re-entrancy of boundary surfaces within the workspace. The existence of these surfaces, which indicate operating barriers within the workspace, plays an important role in the control of the robot-arm. It is possible to minimize the occurrence of singularities by carefully selecting an optimal set of joint limits while retaining the maximum workspace or desired workspace shape. This goal can be achieved with the help of aspect mapping. The fewer the aspects of a manipulator, the smaller the number of re-entrant surfaces. The existence of re-entrant surfaces within the workspace of an industrial
robot could be used as a criterion to characterize the manipulator geometry.

5. Manipulability of robot-arm:

If a manipulator is at a configuration near a singular surface, its manipulability is very poor. Manipulability of a specific configuration reveals the nearness of the configuration to a special configuration. A criterion can be established according to the capability of the actuators and velocity requirements at the end-effector to define an allowance of the "closeness" to a special configuration. The D-surfaces, which cause the robot to lose degrees of freedom, behave like singular surfaces. With the penalty function, the manipulability can also detect nearness to a joint limit. Special action can then be taken to avert the robot-arm from approaching those special configurations.

7.3 Geometric Comparison of Industrial Robots

There are four industrial robots selected as examples to illustrate the evaluation of workspace properties. Remembering the length of the end-effector in the four industrial robots had been standardized to be 15% of the total manipulator length for easy comparison (except the 3P-3R robot). The geometric properties of the above-mentioned four industrial robots are listed in Table
Table 2: Geometric Properties of Four Industrial Robots

<table>
<thead>
<tr>
<th>Property</th>
<th>Model</th>
<th>Cincinnati Milacron T3-746</th>
<th>Citizen Robot Co. M-100</th>
<th>Reis Machine Corporation RR-625/650</th>
<th>Cincinnati Milacron T3-886</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>6-R</td>
<td>6-R</td>
<td>1P-5R</td>
<td>3P-3R</td>
<td></td>
</tr>
<tr>
<td>Volume</td>
<td>3,014,130</td>
<td>224,802</td>
<td>819,708</td>
<td>115,200</td>
<td></td>
</tr>
<tr>
<td>NVI</td>
<td>0.45544</td>
<td>0.7444</td>
<td>0.13584</td>
<td>0.007244</td>
<td></td>
</tr>
<tr>
<td>Volume</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>0.15324</td>
<td>0.0</td>
<td>0.052336</td>
<td>0.12393</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>0.0</td>
<td>0.0</td>
<td>0.052336</td>
<td>0.11538</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0.22253</td>
<td>0.017912</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Radius of</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gyration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>0.54677</td>
<td>0.68557</td>
<td>0.31211</td>
<td>0.19916</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>0.61260</td>
<td>0.68557</td>
<td>0.33535</td>
<td>0.24816</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0.60125</td>
<td>0.69643</td>
<td>0.38838</td>
<td>0.19917</td>
<td></td>
</tr>
</tbody>
</table>

2.

Example 1 is the Cincinnati Milacron T3-746 robot. Figure 7 and Figure 20 shows the workspace geometry of this general purpose 6-R robot arm. Example 2 is the Reis Machine Corporation RR-625/650 robot. It is a 1P-5R manipulator as shown in Figures 8 and 21. The smaller workspace volume (NVI=0.136) reveals that it is less geometrically efficient than a 6-R robot (referring to Figures 20 and 22) since its first joint is prismatic. Figure 8 also shows that there are many re-entrant surfaces
within the workspace volume. Example 3 is the Citizen Robotics Corporation M-100 robot. This is also a general purpose 6-R manipulator arm. The geometry of workspace is shown in Figure 9 and Figure 22. The workspace NVI is 0.744. Example 4 is the Cincinnati Milacron T3-886 robot. This is a 3P-3R industrial robot. The normalized volume index of this robot is very small \(\text{NVI}=0.0072442\) when compared to that of 6-R robots because three orthogonal joint axes are used.

Comparing these, the Citizen Robot M-100 has many interesting geometrical properties. It has the largest NVI \(\text{NVI}=0.744\), and the location of its volume centroid is very close to the origin. However, there is a big void inside the workspace volume, and the geometry of the workspace is a thin shell of a sphere centered at the origin as can be seen in Figure 22. Such a special workspace geometry can be characterized by its radii of gyration about the principal axes: \(r_x=0.686\), \(r_y=0.686\), and \(r_z=0.696\). The large radii of gyration reveal that the distribution of the workspace volume is more than a distance of two thirds of the total link length away from the origin. As is expected, the compactness of its workspace volume is characterized by its radii of gyration, which are the highest values among the four manipulators in Table 2.

Another interesting case is the Cincinnati Milacron
T3-886 robot. Its normalized volume index is very small (NVI=0.007244) in spite of the fact that its actual workspace volume is not small. This result indicates that the NVI is possibly misleading when dealing with structures which have two or three prismatic joints. This is not surprising, since its definition assumes a basically spherical workspace geometry. On the other hand, the low NVI in this case correctly indicates that a relatively large structure is needed to generate unit workspace volume. The shape of this workspace is a rectangular prism, as shown in Figure 23, which is a very compact geometry. The compactness of the workspace can be characterized by its relatively small radii of gyration \( r_x = 0.199, r_y = 0.248, r_z = 0.199 \). That is, the lower the radius of gyration, the more compact the workspace volume.

7.4 Conclusion

Although this study is limited to manipulators with specialized geometries, it does provide a very efficient and accurate method to characterize and evaluation of a manipulator workspace geometry. The program developed so far is found to be applicable to most industrial robots, except for some manipulators with geometries which have voids or holes in their workspaces. In such cases, one may use this program to generate a 'gross' volume from which one
may then subtract the volumes of the holes or voids to obtain a net workspace volume.

In summary, some special contributions to the design and evaluation of a manipulator workspace geometry are presented in this study. These are listed as follows:

1. The use of reciprocal screw theory enables us to provide a very efficient workspace boundary generation algorithm for a kinematic chain, which is expandable to an arbitrary number of links, with either revolute or prismatic joints. The workspace boundary generated by this algorithm is piecewise continuous consisting of straight line segments and circular arcs. This is very different from most pixel search methods which generate zig-zag workspace boundaries.

2. The application of the Gaussian Divergence Theorem to the evaluation of workspace properties makes closed-form formulations possible. This provides an efficient and accurate computation. The solutions are exact with the accuracy limited only by the computer's truncation and round-off errors. The formulation is considered to be far more elegant than any of the pixel counting methods, and should be used as the standard solution.

3. The workspace geometry of a robot can be characterized by the workspace volume, the location of its volume centroid, and its moments of inertia about the fixed reference frame. These relatively few parameters provide
very comprehensive information about the workspace geometry of a given manipulator. Furthermore, the existence of D-surfaces within the workspace also reveals operational barriers of the robot-arm within its workspace. The D-surfaces can be used as a criterion to characterize the manipulability of the robot-arm.

4. The program is capable of generating a workspace boundary with the end-effector's orientation specified in the generating plane. This will enable us to calculate the workspace volume for a robot required to perform a task in which the robot hand should be kept in a desired orientation with respect to the generating plane.

5. Study of the manipulability of a robot arm, with or without joint limit constraints, facilitates direct application of this program to design and controllability analysis of industrial robots. One may select an optimal set of joint parameters to minimize the occurrence of geometric singularities as well as to minimize the change of robot aspects by manipulability analysis and aspect mapping.

6. The use of an interactive computer program for design and evaluation of geometric performance of a given industrial robot allows a very high speed, high accuracy method of design which lends itself much better than batch type programs do.
LIST OF REFERENCES


9. Cwiakala, M., and Lee, T. W., "Generation and
Evaluation of a Manipulator Workspace Based on Optimum Path Search," ASME Paper No. 84-DET-185.


19. Vijaykumar, R., Tsai, M. J., and Waldron, J. K.,


APPENDIX A

List of program "WORKSP.FOR"
**MAIN PROGRAM**

PROGRAM TO CALCULATE WORK SPACE OF A 4 DEGREE OF FREEDOM
ROBOT ARM (I.E. ONLY REGIONAL LINKS and hand are CONSIDERED).

WRITTEN BY: HING 0. TSAI

DATE: APRIL 17, 1984

REVISED: JUNE 20, 1984 -- HAND ORIENTATION
JULY 27, 1984 -- BOUNDRY & VOLUME
AUG. 29, 1984 -- MOMENT OF INERTIA
OCT. 10, 1984 -- PRISMATIC JOINTS (2,3,4)
DEC. 09, 1984 -- 1ST JOINT PRISMATIC
JAN. 30, 1985 -- REMOVAL OF RE-ENTRY BOUNDARIES
OCT. 30, 1985 -- PLTPAK VERSION
JAN. 06, 1986 -- MANIPULABILITY

JULY 10, 1986 -- ADD 5TH JOINT (REDUNDANT)

The program generates workspace of kinematic chain of JJS
In the generating plane, then you can eliminate the interior
arcs or lines interactively to got a net workspace boundary.
The workspace properties such as the cross-sectional area,
volume, and the location of volume centroid, and moments of
inertia can be obtained by selecting the "VOLUME" function
in main menu. Other functions provided by this program are
1. Draw links for various positions of motion sequence.
2. Draw boundaries for 2, 3, and 4 links chain in the plane.
3. Hand orientation of link 3 w.r.t. the generating plane.
4. Calculate the manipulability index of a robot posture.

Notice that some functions do not yet implemented for number
of joint > 6. those are: Hand orientation, Manipulability.
The hand orientation function is also not work if joint 4 is
a sliding joint.
The joint parameters are to be arranged so that joints 2,3,4
lie in the generating plane. Total link length is normalized
to unit.

**JTYPE**: Joint type, either R for revolute or P for prismatic
**MAKER**: the robot manufacturer
**MODEL**: the model of the robot
**S,IL**: Intermediate character value used to hold data
**IIV**: logical variable, if true, write a data file
**RF**: logical variable, if true, Hand orientation is specified
**ROTA**: logical array, if true, the motion sequence is
generated by a revolute joint
**INACT**: logical array, if true, the joint is inactive
**COMP**: logical variable, if true, the robot is in an unstable
configuration under the applied force
**RATIO**: the total physical link length
**A(I)**: the array to hold lengths of joint members
**BL(I)**: the array to hold maximum lengths of joint members
**PI**: constant PI=3.14159
**RTH(I)**: the motion range of joint I
**TH(I)**: the angle between joint i-1 to i
**Twist(I)**: constant angle between two prismatic joints
**LAST**: counter for each small motion increment, variable
**MAX**: the maximum step of motion increment
**MM**: counter for each motion sequence, variable
**NSE**: maximum number of motion sequences
LOOP(1): the starting position of motion sequence 1
IANG: the degree of hand orientation to the horizontal
Jnum: number of joint
IDIR: motion direction 1-Increase 2-decrease joint variable
LL: COUNTER FOR EVERY FIVE MOTION SETP
BD(1): the force direction of every FIVE step 1
IP:
ICOUNT(1):
JJ:
Sweep: the motion range of 1st joint
THMIN(1): the minimum motion position of joint 1
THMAX(1): the maximum motion position of joint 1
THS(1): the minimum motion position of joint 1 (degree)
THB(1): the maximum motion position of joint 1 (degree)
ALPHA(1): the twist angle between joint 1-1 to 1
THETA(1): the joint angle of joint 1
OFFSET(1): the offset of joint 1
LENGTH(1): the link length 1
TITLE: character array to write the subroutine used to screen
CX1(1), CY1(1): the x.y coordinates of joint 2 in motion step 1
CX2(1), CY2(1): the x.y coordinates of joint 3 in motion step 1
CX3(1), CY3(1): the x.y coordinates of joint 4 in motion step 1
CX4(1), CY4(1): the x.y coordinates of joint 5 in motion step 1
TPX(1), TPY(1): the x.y coordinates of hand in motion step 1

Variables used in volume integration (except for local variables):
The following parameters refers to figures in Dissertation:
PC(1): if prismatic, y coordinate of starting position
RA(1): if revolute, the radius of rotation
PHY(1): if prismatic, the angle between a vector from
SHI01(1): if prismatic, the motion range of this sequence
SH11(1): if prismatic, the angle between the starting position
to the origin and horizontal
TVOL1: total volume (NV1)
TARCA: total cross-sectional area in the generating plane
TVCX: x location of total area centroid (radius of rotation)
TVZ: z location of total volume centroid
TVA: z location of total area centroid
TVX: x location of total volume centroid
TVH: x location of total volume centroid
TVH: x location of total volume centroid
TVH1Y: total volume moment of inertia about x axis
TVH1Z: total volume moment of inertia about y axis
TVH1Z: total volume moment of inertia about z axis
PX: radius of gyration about x axis
PY: radius of gyration about y axis
PZ: radius of gyration about z axis

PROGRAM Wo-kSPLOGICAL RF.IW.COMP,ROTA.INACT
CHARACTER * 10 MODEL
CHARACTER*20 MAKER
CHARACTER*35 TITLE
CHARACTER JTYPE, IL
COMMON /LENGTH/RATIO,A(4),BL(4),P1,PTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /XY/CX1(999),CX2(999),CX3(999),CX4(999),CY1(999),CY2(999),
CY3(999),CX4(999),CY4(999),TPX(999),TPY(999),FDIR(999),BD(38B)
COMMON /SUM/PC(601),RA(60),PHY(60),SH1(60),SH2(60),SH3(60),
COMMON /ITG/LAST,MM,LOOP(60),NSE,IA,LL,IP,IDIR,ICOUNT(30),JJ
COMMON /CHR/MODEL,MAKER,OTYPE(6)
COMMON /LOG/TV.RF,CMP,ROTA(6B),INACT(6)
CALL KINITZ(5.20)
TITLE='MAIN PROGRAM'
P1=4.*ATAN(1.)
CALL KINPUT
IP=1
ICOUNT(1)=1
LL=1
MM=1
LAST=1
RF=.FALSE.
CALL SEQUE(*170,*280)
IP=IP+1
LOOP(MM)=LAST
LAST=LAST-1
ICOUNT(IP)=LAST
NSE=MM=1
MAX=LAST
150 CALL FRAME(TITLE)
CALL SCREEN
CALL SELECT
CALL LISPAR
CALL KOCAT(X,Y,1L)
IF(I.LT.10) GO TO 9
IF(Y.LT.475.) THEN
IF(Y.LT.475.) THEN
GOTO 150
ELSE IF(Y.LE.528. .AND. Y.GT.475.) THEN
ELIF(Y.LE.528. .AND. Y.GT.475.) THEN
GOTO 140
ELSE IF(Y.LE.565. .AND. Y.GT.528.) THEN
ELSE IF(Y.LE.565. .AND. Y.GT.528.) THEN
GOTO 130
ELSE IF(Y.LE.602. .AND. Y.GT.565.) THEN
ELSE IF(Y.LE.602. .AND. Y.GT.565.) THEN
GOTO 130
ELSE IF(Y.LE.639. .AND. Y.GT.602.) THEN
ELSE IF(Y.LE.639. .AND. Y.GT.602.) THEN
GOTO 180
ELSE IF(Y.LE.676. .AND. Y.GT.639.) THEN
ELSE IF(Y.LE.676. .AND. Y.GT.639.) THEN
GOTO 160
ELSE IF(Y.LE.713. .AND. Y.GT.676.) THEN
ELSE IF(Y.LE.713. .AND. Y.GT.676.) THEN
GOTO 190
ELSE IF(Y.LE.713.) THEN
ELSE IF(Y.LE.713.) THEN
GOTO 120
ELSE GOTO 110
ENDIF
140 CALL HANDLE(*170,*280)
130 CALL EOUNT
GO TO 110
160 CALL AMNULN:
GO TO 110
280 CALL KTEX1('WARNING: CHECK ROTATING SEQUENCE 78.B)
CALL KMOVAB(358,-38.)
CALL KTEX1('WARNING: CHECK ROTATING SEQUENCE 1.8)
CALL KMOVAB(2,-8.)
SUBROUTINE ANLINK

THIS SUBROUTINE ANIMATES THE ROBOT ARM.
THE PARAMETERS TO BE TRANSFERRED ARE:

1 - DRAW ALL CONFIGURATIONS FOR ALL SEQUENCES
2 - DRAW ALL LINKS OF A MOTION SEQUENCE
3 - DRAW ONE LINK AT EVERY STARTING SEQUENCE
4 - DRAW THE FIRST LINK OF THE n(th) SEQUENCE

****

CALL KTEXT( 'HIT <CR> TO CONTINUE', 0 )
CALL KPAUSE( 0.0 )
CALL KTXSIZE( 0.012 )
GO TO 170

170 CALL VOLUME( 300., 150. )
190 CALL MANIP
GOTO 150
120 CONTINUE
CALL KFINISH
STOP
END

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

SUBROUTINE ANLINK

LOGICAL IW, RF, COMP, ROA, INACT
CHARACTER IL
COMMON / LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON / AHSL1/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /XY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999),
1 CY3(999),CX4(999),CY4(999),TPX(999),TPY(999),FDIR(999),BD(300)
COMMON / SUM/PC(60),RA(60),PHY(60),SHI(60),SHI1(60)
COMMON /ITG/LAST.MM,LOOP(60),NSE,IANG.LL,IP,IDIR,ICOUNT(30),JJ
1 MAXJNUM
COMMON /LOG/IW,RF,COMP,ROA(60.),INACT(6)
CALL KWPRINT(0.,1.,0.0292,0.7324)
CALL KTXSIZE(0.012)
CALL KMOVAB(-2100.,160.)
CALL KTEXT( 1 - DRAW ALL LINKS OF ALL SEQUENCES', 1 )
CALL KTEXT( 2 - DRAW ALL LINKS OF A SEQUENCE', 1 )
CALL KTEXT( 3 - DRAW FIRST LINK OF ALL SEQUENCES', 1 )
CALL KTEXT( 4 - DRAW FIRST OF A SEQUENCE: ', 1 )
CALL KREAD(IHELP, 'ILHELP', '1')
IF( IL.EQ.'1') THEN
DO 200 I=1,MAX
IF( BL(4).EQ.0.) THEN
CALL KMOVAB(CX2(I),CY2(I))
ELSE
CALL KMOVAB(CX4(I),CY4(I))
ENDIF
CALL KDRVAB(CX2(I),CY2(I))
CALL KDRWAB(CX1(I),CY1(I))
CALL KDRWAB(0.,0.)
END
CONTINUE
CALL KLINE(1)
CALL KTEXT( 'DRAW ARROW HEAD', 0.9 )
CALL KREAD(IL,IHELP, 'ILHELP', 'N')
IF( IL.EQ.'Y') THEN
DO 300 I=1,MAX

IF(BL(4).EQ.0.) THEN
IF(RF) THEN
CALL ARROW(CX2(I),CY2(I),50.,FD1R(I),0)
ELSE
CALL ARROW(CX3(I),CY3(I),50.,FD1R(I),0)
ENDIF
ELSE
CALL ARROW(CX4(I),CY4(I),50.,FD1R(I),0)
ENDIF
CONTINUE
ELSE ENDIF
ELSE IF(IL.EQ.2) THEN
CALL KTEXT(' ENTER SEQUENCE NO',0)
CALL KREAD(1,NI,251,0)
DO 250 I=LOOP(NI),LOOP(NI+1)
IF(BL(4).EQ.0.) THEN
CALL KMOVAB(CX3(I),CY3(I))
ELSE
CALL KMOVAB(CX4(I),CY4(I))
CALL KDRAW(CX3(I),CY3(I))
ENDIF
CALL KDRAW(0.,0.)
ENDIF
CONTINUE
ELSE IF(IL.EQ.3) THEN
DO 10 J=1,NSE
CALL HAND(J)
10 CONTINUE
ELSE
CALL KTEXT(' ENTER O TO EXIT',1)
1 CALL KREAD(1,NI,1,0)
2 CALL KDRAW(CX3(I),CY3(I))
IF(NI.EQ.0 .OR. NI.GT.NSE) THEN
GOTO 250
ELSE
CALL HAND(NI)
ENDIF
GOTO 2
ENDIF
350 CALL KVUPD(0.,1.,0.,0.,76171875)
CALL KWINDD(0.,1024.,0.,780.)
RETURN

FUNCTION ANOM(ANG)

THIS FUNCTION NORMALIZE AN ANGLE SO THAT ITS OUTPUT VALUE
IS -PI <= ANOM <= PI

FUNCTION ANOM(ANG)
PI=4.*ATAN1(I.)
TEMP=ANG
IF(TEMP.LE.-3.1416) TEMP=TEMP+2.*PI
IF(TEMP.GT.3.1415) TEMP=TEMP-2.*PI
ANOM=TEMP
RETURN
SUBROUTINE ARROW(X, Y, RR, DIR, N)
THIS SUBROUTINE TO DRAW A ARROW FROM POINT(X, Y) WITH THE DIRECTION 'DIR' (RADIUS) TO THE HORIZONTAL AND LENGTH 'RR'.

SUBROUTINE ARROW(X, Y, RR, DIR, N)
CHARACTER*2 S
XDP=RR*COS(DIR)
YDP=RR*SIN(DIR)
IF(RR.LT.100.) THEN
  RT=0.4*RR
ELSE IF(RR.LT.300.) THEN
  RT=0.2*RR
ELSE IF(RR.LT.600.) THEN
  RT=0.1*RR
ELSE
  RT=0.05*RR
ENDIF
IF(RT.GT.50.) RT=50.
IF(RT.LT.15.) RT=15.
SDR=SIN(DIR)*2.36)*RT
CDR=2*COS(DIR)*2.36)*RT
SX=X*KOR
SY=Y*YDR
CALL KLNTYP('SOLID')
CALL KMOVAB(X, Y)
CALL KDRWRL(CDR, SDR)
CALL KMOVAB(X, Y)
CALL KDRWRL(CDR2, SDR2)
CALL KMOVAB(X, Y)
IF(N.EQ.0) THEN
  WRITE(UNIT+S, FMT=111) N
  FORMAT(13)
  CALL KTEXT(S, 0)
ELSE
ENDIF
RETURN
END

SUBROUTINE EOTHVW
THIS SUBROUTINE DRAWS TWO VIEWS OF WORKING SPACE BOUNDARY. THESE ARE PROJECTIONS ON X-Z AND X-Y PLANE.

SUBROUTINE EOTHVW
CHARACTER*20 NAME
CHARACTER*10 TITLE
CHARACTER*10 MODEL
COMMON /ANGLE/ SWEEP, THMIN(4), THMAX(4), THS(4), THB(4)
COMMON /XY/ CX1(999), CX2(999), CX3(999), CY1(999), CY2(999).
COMMON LENGTH/RATIO,A14,BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON ITG/LAST,MM,LOOP60,NSE,LANG,LL,IP,DIR,ICOUNT(30),JJ

1 COMMON /CHR/MODEL,MAKER,JTYPE(6)
     TITLE='MAIN/VOLUME'
     CALL FRAMEITITLE)
     CALL KVWRT(0.518,.95,0.259,0.691)
     CALL SCALE('V')
     AMIN=TPX(1)
     AMAX=TPX(1)
     DO 100 I=1,LAST-1
        IF(AMIN.GT.TPX(I+1)) AMIN=TPX(I+1)
        IF(AMAX.LT.TPX(I+1)) AMAX=TPX(I+1)
     CONTINUE
     AM=AMAX(ABS(AMIN),ABS(AMAX))
     AN=AMIN(ABS(AMIN),ABS(AMAX))
     GAM1=SWEEP/2.
     IF(JTYPE(1).EQ.‘P’) THEN
        CALL IMOVABI(AMIN,GAM1)
        CALL IDRWAIE(AMAX,GAM1)
        CALL IDRAE(AMAX,GAM2)
        CALL IDPW(E(AMAX,GAM1)
        ELSE
        IF(SWEEP.LT.180.) THEN
           TH2=SWEEP/2. - Pi/180.
        ELSE
           TH2=(180.-SWEEEP/2.)*Pi/180.
        ENDIF
        TH3=180.-SWEEEP/2.
        TH4=180.-SWEEEP/2.
        TH5=SWEEEP-180. + TH3
        TH6=TH3+180.
        SY=AN*COS(TH2)
        SV=AN*SIN(TH2)
        EX=AM*COS(TH2)
        EY=AM*SIN(TH2)
        IF(AMAX.LT.0.) THEN
           IF(SWEEP.LT.180.) THEN
              CALL IMOVAB(BX,SY)
              CALL IDPWAB(SX,SY)
              CALL IDRAE(BX,-BY)
              CALL IDPWAB(-SX,-SY)
              CALL CIRCLE(O,0,-AM,TH3,TH4,2)
              CALL CIRCLE(O,0,AN,TH3,TH4,2)
           ELSE
              CALL IMOVAB(BX,SY)
              CALL IDPWAB(SX,SY)
              CALL IDRAE(BX,-BY)
              CALL IDPWAB(-SX,-SY)
              CALL CIRCLE(O,0,-AM,TH3,TH4,2)
              CALL CIRCLE(O,0,AN,TH3,TH4,2)
           ENDIF
        ELSE IF(AMIN.GT.0.) THEN
           IF(SWEEP.LT.180.) THEN
              CALL IMOVAB(BX,SY)
              CALL IDPWAB(SX,SY)
              CALL IMOVAB(BX,-BY)
              CALL IDPWAB(SX,-SY)
              CALL CIRCLE(O,0,-AM,TH3,TH4,2)
              CALL CIRCLE(O,0,-AN,TH3,TH4,2)
           ELSE
              CALL IMOVAB(BX,SY)
              CALL IDPWAB(SX,SY)
              CALL IMOVAB(BX,-BY)
              CALL IDPWAB(SX,-SY)
              CALL CIRCLE(O,0,-AM,TH3,TH4,2)
              CALL CIRCLE(O,0,-AN,TH3,TH4,2)
           ENDIF
        ELSE
CALL KORVAB(SX,-SY)
CALL CIRCLE(0.,0.,AM,-SWEEP/2.,SWEEP/2.,2.)
CALL CIRCLE(0.,0.,AN,-SWEEP/2.,SWEEP/2.,2.)
ELSE
CALL KMOVAB(-BX,BY)
CALL KORVAB(-SX,SY)
CALL KMOVAB(-BX,-BY)
CALL KORVAB(-SX,-SY)
CALL CIRCLE(0.,0.,AM,-TH5,TH5,2.)
CALL CIRCLE(0.,0.,AN,-TH5,TH5,2.)
ENDIF
ELSE IF(SWEEP.GT.180.) THEN
IF(ABS(AMIN).EQ.AN) THEN
CALL KMOVAB(-DX,BY)
CALL KORVAB(-SX,SY)
CALL LNITYP('DOTTED')
CALL KORVAB(SX,-SY)
CALL KMOVAB(-BX,-BY)
CALL LNITYP('SOLID')
CALL KORVAB(SX,-SY)
CALL CIRCLE(O.,0.,AM,-TH5,TH5,0)
CALL CIRCLE(O.,0.,AN,TH3,TH3,1)
CALL CIRCLE(O.,0.,AN,TH3,TH3,0)
CALL CIRCLE(O.,0.,AN,TH3,TH3,1)
ENDIF
ELSE IF(ABS(AMIN).EQ.AN) THEN
CALL KMOVAB(BX,BY)
CALL KORVAB(SX,SY)
CALL LNITYP('DOTTED')
CALL KORVAB(SX,-SY)
CALL KMOVAB(BX,-BY)
CALL LNITYP('SOLID')
CALL KORVAB(SX,-SY)
CALL CIRCLE(A.,A.,AM,TH3,TH4,0)
CALL CIRCLE(A.,A.,AN,TH6,TH4,1)
CALL CIRCLE(A.,A.,AN,TH6,TH4,1)
CALL CIRCLE(A.,A.,AN,TH6,TH4,1)
ENDIF
ELSE IF(ABS(AMIN).EQ.AN) THEN
CALL KMOVAB(BX,BY)
CALL KORVAB(SX,SY)
CALL CIRCLE(O.,0.,AM,-SWEEP/2.,SWEEP/2.,0)
CALL CIRCLE(O.,0.,AN,TH3,TH4,0)
ELSE
CALL KMOVAB(-BX,BY)
CALL KORVAB(SX,-SY)
CALL KMOVAB(-CX,-BY)
CALL KORVAB(SX,SY)
CALL CIRCLE(O.,0.,AM,TH3,TH4,0)
CALL CIRCLE(O.,0.,AN,-SWEEP/2.,SWEEP/2.,0)
ENDIF
ENDIF
CALL KVWPRT(0.,0.,0.472,0.,0.259,0.691)
CALL SCALE('Z ')
CALL BOUND(O.,0.)
CALL KVWPRT(O.,1.,O.,0.,0.,7617875)
This subroutine draws the boundary of the robot workspace in 3D according to menu selection.

**SUBROUTINE BOUND**

**CHARACTER IL**

**COMMON**
- SWEEP, THMIN, THMAX, THS, THB
- COMMON /XY/I0009, CX2, CX3, CY2, CY3, CX4, CV1, CV2, CV3
- COMMON /ITG/LAST, MM, LOOP, NSE, IANG, LL, IP, IDIR, ICOUNT
- MAX
- COMMON /IAT/NUMS, NL

**CALL**
- KWINDO(1024.,0.,768.)
- KTEXT('WORKSPACE OF THE MANIPULATOR', 0)
- KMOVAB(265., 165.)
- KTEXT(MAKER, 0)
- KMOVAB(610., 165.)
- KTEXT(MODEL, 0)
- KMOVAB(180., 720.)
- KTEXT('CROSS SECTION', 0)
- KMOVAB(700., 720.)
- KTEXT('TOP VIEW', 0)
- RETURN

**END**

This subroutine draws the motion trajectory of joint 4 in the generating plane. If I.I.EQ.0 then draw the force direction applied to.
the trajectory. This subroutine also draw dotted line for un-reachable boundary due to joint limit for a hand orientation.

SUBROUTINE BOUND4(KK)
COMMON /XY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999),
1 CY3(999),CX4(999),CY4(999),TPX(999),TPY(999),FDIR(999),BD(300)
COMMON /ITG/LAST,MH,LOOP(60),NSET,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /IAT/INUM(60),NUMS(60),NL(300)
K=1
CALL KMOVAD(CX4(1),CY4(1))
DO 101 J=1,IP-1
100 CONTINUE
RETURN
END

This subroutine draw motion trajectory of Joint 3 in the generating plane by dashed line type.

SUBROUTINE BOUND3
COMMON /XY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999),
1 CY3(999),CX4(999),CY4(999),TPX(999),TPY(999),FDIR(999),BD(300)
COMMON /ITG/LAST,MH,LOOP(60),NSET,IANG,LL,IP,IDIR,ICOUNT(30),J3
1 MAX
COMMON /IAT/INUM(60),NUMS(60),NL(300)
CALL KLNTYP(‘DASHED’)
CALL KMOVAD(CX3(1),CY3(1))
DO 101 J=1,MAT
100 CONTINUE
CALL KLNTYP(‘SOLID’)

This subroutine draws motion trajectory of Joint 2 in the generating plane by dotted line type.

Subroutine BOUND2
COMMON /XY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999),
1 CY3(999),CX4(999),CY4(999),TPX(999),TPY(999),FDIR(999),BD(300)
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
CALL KLNTYP('DOTTED')
CALL KMOVAB(CX2(I),CY2(I))
DO 101 J=1,MAX
CALL :DRWAB(CX2(J),CY2(J))
101 CONTINUE
CALL KLNTYP('SOLID')
RETURN
END

This subroutine draws motion trajectory of hand reference point, so that it only represents the real boundary after removal of the interior motion sequences.

MM - total motion sequence after removal of overlapped area
J - draws arrow head in the progressing direction
I - draws arrow
D - no draws
k - if 0 draws real line
K - if not 0 draw dotted line

Subroutine BOUND(T,J,LK)
COMMON /XY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999),
1 CY3(999),CX4(999),CY4(999),TPX(999),TPY(999),FDIR(999),BD(300)
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
DO 100 J=1,MH-1
CALL DRWLOP(I,J,LK)
100 CONTINUE
RETURN
END

This subroutine draws boundary generated by loop N
I - draws arrow head in the progressing direction
I - draws arrow
I - no draws
L - if 0 draws real line
I - if not 0 draw dotted line
N - the sequence to be drawn and written out

Subroutine DRWLOP(N,I,L)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THE(4)
COMMON /XY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999),
1 CY3(999),CX4(999),CY4(999),TPX(999),TPY(999),FDIR(999),BD(300)
COMMON /ITG/LAST,MM,LOOP(60),NSE,IANG,LL,IP,IDIR,ICOUNT(30),J3
CALL KMOVAB(TPX(LOOP(N),TPY(LOOP(N)))
DO 200 K = LOOP(N), LOOP(N+1)-2
   IF (I, GE, 1) THEN
      H = MOD(K+2, 3)
   IF (H, EQ, 0) THEN
      AA = TPY(K+1) - TPY(K)
      BB = TPX(K+1) - TPX(K)
      IF (AA, EQ, 0 .AND. BB, EQ, 0) THEN
         DIS = SQRT (AA**2 + BB**2)
         ANG = ATAN2(AA, BB)
         CALL ARROW(TPX(K), TPY(K), DIS, ANG)
      ELSE
         CALL ARROW(TPX(K), TPY(K), DIS, ANG)
      ENDIF
      ENDIF
      CALL KMOVAB(TPX(K+1), TPY(K+1))
   ELSE
      IF (I, GE, 1) THEN
         CALL KLNTYP('DOTTED')
         CALL KDRWAB(TPX(K+1), TPY(K+1))
      ELSE
         CALL KLNTYP('SOLID')
         CALL KDRWAB(TPX(K+1), TPY(K+1))
      ENDIF
      ENDIF
   ELSE
      IF (I, GE, 1) THEN
         CALL KLNTYP('DOTTED')
         CALL KDRWAB(TPX(K+1), TPY(K+1))
      ELSE
         CALL KLNTYP('SOLID')
         CALL KDRWAB(TPX(K+1), TPY(K+1))
      ENDIF
      ENDIF
   ENDIF
200 CONTINUE
   CALL KLNTYP('SOLID')
RETURN
END

C: CALCUL.FOR
C
C THIS SUBROUTINE CALCULATE POSITIONS AND ANGLES OF
C robot-arm for small motion increments.
C INPUT: TH(I), A(I), I=1..4, where TH(I) are the joint angles
C of joints 2, 3, 4, and 5; A(I) are the link length or
C offset of joint 2, 3, 4, 5, and length of end-effector.
C output: POX(I), POY(I), the coordinates of joint I
C RO(I): distance from joint 2 to I,
C RO(2): distance from joint 2 to 4,
C RO(3): distance from joint 3 to 5,
C RO(4)*2 TO TIP,
C RO(5)=3 TO TIP,
C RO(6)=4 TO TIP,
C AFA(I): angle between line connecting joints 2,3 to
C the horizontal
C AFA(2): joints 2,4
C AFA(3): joints 2,5
C AFA(4): joints 3,4
C AFA(5): joints 4,5
SUBROUTINE CALCU1R0X.R0Y.R0.RO,AFA

EXTERNAL ANOM
CHARACTER*20 MAKER
CHARACTER*10 MODEL
CHARACTER*10 TYPE
COMMON /CHR/MODEL,MAKER,TYPE
COMMON /LENGTH/RAT1O(1),BL(4),PI,RTH(4),TH(4),TWIST(4)
DIMENSION COSA(10),SINA(10),ROX(4),ROY(4),ROU(6),AFA(10)

C

IF any link length is zero, we should still retain the direction
of the link for a prismatic joint. So set the link length to be
a small value.

IF(A(1).EQ.0.) A(1)=1.E-5
IF(A(2).EQ.0.) A(2)=1.E-5
IF(A(3).EQ.0.) A(3)=1.E-5
IF(A(4).EQ.0.) A(4)=1.E-5
ROX(1)=A(1)*COS(TH(1))
ROY(1)=A(1)*SIN(TH(1))
ROX(2)=ROX(1)+A(2)*COS(TH(2))
ROY(2)=ROY(1)+A(2)*SIN(TH(2))
ROX(3)=ROX(2)+A(3)*COS(TH(1)+TH(2)+TH(3))
ROY(3)=ROY(2)+A(3)*SIN(TH(1)+TH(2)+TH(3))
ROX(4)=ROY(3)+A(4)*COS(TH(1)+TH(2)+TH(3)+TH(4))
ROY(4)=ROY(3)+A(4)*SIN(TH(1)+TH(2)+TH(3)+TH(4))
RO(1)=SQR((ROX(2)**2+ROY(2)**2))
RO(2)=SQR((ROX(3)**2+ROY(3)**2))
RO(3)=SQR((ROX(4)**2+ROY(4)**2))
RO(4)=SQR((ROX(5)**2+ROY(5)**2))
RO(5)=SQR((ROX(6)**2+ROY(6)**2))
COSA(2)=ROX(2)/RO(1)
COSA(3)=ROX(3)/RO(2)
COSA(4)=ROX(4)/RO(3)
COSA(5)=ROX(5)/RO(4)
COSA(6)=ROX(6)/RO(5)
COSA(7)=ROX(7)/RO(6)
COSA(8)=ROX(8)/RO(7)
COSA(9)=ROX(9)/RO(8)
COSA(10)=ROX(10)/RO(9)
COSA(1)=ROX(1)/RO(1)
COSA(2)=ROX(2)/RO(2)
COSA(3)=ROX(3)/RO(3)
COSA(4)=ROX(4)/RO(4)
COSA(5)=ROX(5)/RO(5)
COSA(6)=ROX(6)/RO(6)
COSA(7)=ROX(7)/RO(7)
COSA(8)=ROX(8)/RO(8)
COSA(9)=ROX(9)/RO(9)
COSA(10)=ROX(10)/RO(10)
COSA(1)=ROX(1)/RO(1)
COSA(2)=ROX(2)/RO(2)
COSA(3)=ROX(3)/RO(3)
COSA(4)=ROX(4)/RO(4)
COSA(5)=ROX(5)/RO(5)
COSA(6)=ROX(6)/RO(6)
COSA(7)=ROX(7)/RO(7)
COSA(8)=ROX(8)/RO(8)
COSA(9)=ROX(9)/RO(9)
COSA(10)=ROX(10)/RO(10)
SINA(1) = (ROY(4) - ROY(3))/A(4)
AFA(1) = TH(1)
DO 10 I = 2, 10
IF(SINA(1) .EQ. 0.) SINA(1) = 1.E-30
IF(ABS(1 - COSA(1)) .LT. 1.E-7) THEN
   AFA(1) = 0.
ELSE
   AFA(1) = 2.*ATAN((1 - COSA(1))/SINA(1))
END IF
10 IF(JTYPE(1) .EQ. 'P') AFA(1) = TWIST(1)
IF(JTYPE(3) .EQ. 'P') AFA(4) = TWIST(2) + AFA(1)
IF(JTYPE(4) .EQ. 'P') AFA(5) = TWIST(3) + AFA(4)
IF(ABS(TH(2)) .LT. 0.001) KL = 1
IF(ABS(TH(3)) .LT. 0.001) KM = 1
IF(KL .EQ. 1) OR (KM .EQ. 1)
   AFA(2) = AFA(1)
   AFA(5) = AFA(4)
   AFA(3) = AFA(2)
DO 30 I = 1, 4
   IF(RO(1) .LE. 0.001) A(1) = 0.
   IF(RO(3) .LE. 0.001) A(3) = 0.
   IF(RO(5) .LE. 0.001) A(5) = 0.
30 CONTINUE
AFA(1) = ANO11(AFA(1))
AFA(3) = ANO11(AFA(3))
40 CONTINUE
RETURN
END

SUBROUTINE CALC1
SUBROUTINE CALCULATE POSITIONS AND ANGLES OF A robot with the orientation of the hand specified in the generating plane.
The program good for 3 links robot with the third link to be the end-effector.

SUBROUTINE CALC2(ROX, ROY, RO, AFA)
SUBROUTINE CALCULATE POSITIONS AND ANGLES OF A robot with the orientation of the hand specified in the generating plane.
The program good for 3 links robot with the third link to be the end-effector.

INPUT: A(I), LENGTH OF EACH LINK.
TH(1), TH(2), ANGLE OF LINKS 1, 2
AFA(5), HAND ORIENTATION.

SUBROUTINE CALCU2(ROX, ROY, RO, AFA)
EXTERNAL ANO11
CHARACTER*20 MAKER
CHARACTER*10 MODEL
CHARACTER JTYPE
COMMON /CHP/MODEL, MAKER, JTYPE(6)
COMMON /LENGTH/RATIO(4), BL(4), PI, RTH(4), TH(4), TWIST(4)
COMMON /CHP/MODEL, MAKER, JTYPE(6)
DIMENSION COSA(6), SINA(6), ROX(4), ROY(4), RO(6), AFA(10)
BET1 = TH(1)
BET2 = TH(2)
ROX(1) = 0.
ROY(1) = 0.
IF(A(1) .EQ. 0.) A(1) = 1.E-3
IF(A(2) .EQ. 0.) A(2) = 1.E-3
ROX(1) = A(1)*COS(BET1)
ROY(1) = A(1) * SIN(2)
ROY(2) = ROY(1) * A(2) * COS(2) * SIN(3) * A(2) * COS(3) * SIN(2)
ROY(3) = ROY(2) * A(3) * COS(AFA(5))
ROY(4) = ROY(3) * ROY(2) = A(3) * SIN(AFA(5))
ROY(5) = ROY(4) * A(2) * COS(3) = 2 * ROY(2) = 2
ROY(6) = ROY(5) = 2 * ROY(3) = 2
ROY(7) = ROY(6) * (ROY(6) - ROY(2)) = 2 * ROY(3) - ROY(1) = 2

COS(1) = ROY(1)/A(1)
SINA(1) = ROY(1)/A(1)
IF (ROY(1) < 1.E-10) THEN
    IF (ROY(2) < 1.E-10) THEN
        AFA(1) = 0.
    ELSE
        AFA(1) = 2. * ATAN((-1 - COS(1))/SINA(1))
    END IF
ELSE
    AFA(1) = 2. * ATAN((-1 - COS(1))/SINA(1))
END IF
ELSE
    AFA(1) = 2. * ATAN((-1 - COS(1))/SINA(1))
END IF

DO 30 1 = 1, 4

AFA(1) = AFA(1) + ANOM(AFA(5) - AFA(4))

IF (TH(2) < 1.E-5) THEN
    IF (SINA(1) < 1.E-5) THEN
        AFA(1) = 1.E-5
    ELSE
        AFA(1) = 2. * ATAN((-1 - COS(1))/SINA(1))
    END IF
ELSE
    AFA(1) = 2. * ATAN((-1 - COS(1))/SINA(1))
END IF

CONTINUE

TH(3) = ANOM(AFA(5) - AFA(4))

IF (TH(2) < 1.E-5) THEN
    IF (F(TH(2) < 1.E-5) THEN
        AFA(1) = 1.E-5
    ELSE
        AFA(1) = 2. * ATAN((-1 - COS(1))/SINA(1))
    END IF
ELSE
    AFA(1) = 2. * ATAN((-1 - COS(1))/SINA(1))
END IF

CONTINUE

IF (TH(3) < 1.E-5) THEN
    IF (SINA(1) < 1.E-5) THEN
        AFA(1) = 1.E-5
    ELSE
        AFA(1) = 2. * ATAN((-1 - COS(1))/SINA(1))
    END IF
ELSE
    AFA(1) = 2. * ATAN((-1 - COS(1))/SINA(1))
END IF

CONTINUE

RETURN

END

--------------------------------------------------------------------------

SUBROUTINE CASEPP

THIS SUBROUTINE ELIMINATES RE-ENTRY SEQUENCE FOR TWO INTERSECTING SEQUENCES generated by a P.P. chain which are PARTIALLY ON THE BOUNDARY.
SUBROUTINE CASEPP(I,J)
EXTERNAL ANOM
BYTE AL(2)
CHARACTER IL
CHARACTER*35 TITLE
LOGICAL CW1,CW2
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /XY/CX(999),CY(999),CX2(999),CX3(999),CY3(999),CY2(999),
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
NUM2=3*INT((DIS2/PHY(J))*10)
NUM1=3*INT((DIS1/PHY(J))*10)
NUMH=3*INT((DISH/PHY(J))*10)
K1=LOOP(I)+NUM1
CALL KMOVAB(TPX(LOOP(I)),TPY(LOOP(I)))
DO 200 N=LOOP(I),LOOP(I)+NUM1
AA=TPY(N+1)-TPY(N)
BB=TPX(N+1)-TPX(N)
IF(AA.EQ.0. .AND. BB.EQ.0.) THEN
ELSE
  DIS=SORT(AA*AA+BB*BB)
  CALL ARROW(TPX(N),TPY(N),DIS.SHII(I),I)
ENDIF
200
CONTINUE
CALL KDRWAB(X,Y)
DO 300 N=LOOP(J)+NUM2+1,LOOP(J)+1-2
AA=TPY(N+1)-TPY(N)
BB=TPX(N+1)-TPX(N)
IF(AA.EQ.0. .AND. BB.EQ.0.) THEN
ELSE
  DIS=SORT(AA*AA+BB*BB)
  CALL ARROW(TPX(N),TPY(N),DIS.SHII(J),O)
ENDIF
300
CONTINUE
CALL KMOVAD(-2100.,250.)
CALL KTEXT('  IS THIS OK? ',1)
CALL KTEXT(' ENTER 'Y/N' ',1)
302
CALL KREADC(HELP,Y)
IF(L.Y.EQ.'N'. .OR. IL.Y.EQ. 'n') GOTO 400
IF(L.Y.EQ.'Y'. .OR. IL.Y.EQ. 'y') GOTO 500
CALL KTEXT(' ERROR. INPUT 'Y/N'. ',1)
GOTO 302
500
CONTINUE
TPX(K1)=X
TPY(K1)=Y
DO 10 N=1+1,MM
LOOP(N)=LOOP(N)-NUM1
10
CONTINUE
LAST=LAST-NUM1
DO 20 N=LOOP(I)+1,LAST
TPX(N)=TPX(N)+NUM1
TPY(N)=TPY(N)+NUM1
20
CONTINUE
TPX(LOOP(J))=X
TPY(LOOP(J))=Y
DO 30 N=J+1,MM
LOOP(N)=LOOP(N)-NUM2
30
CONTINUE
LAST=LAST-NUM2
DO 40 N=LOOP(J)+1,LAST
TPX(N)=TPX(N)+NUM2
TPY(N)=TPY(N)+NUM2
40
CONTINUE
IF(Y.EQ.0. .AND.X.EQ.0.) THEN
  STOP=0.
ELSE
  STOP=ATAH2(Y,X)
ENDIF
PC(J)=Y
RA(J)=X
SUBROUTINE CASERP

THIS SUBROUTINE ELIMINATE RE-ENTRY SEQUENCE FOR TWO INTERSECTING SEQUENCES generated by a RP chain which are partially ON THE BOUNDARY.

SUBROUTINE CASERP(I,K,L,M)
CHARACTER I L
CHARACTER*35 TITLE
LOGICAL RF,IV,COMP,ROTA,INACT
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /XY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999),
     CY3(999),CX4(999),CY4(999),TPX(999),TPY(999),FDIR(999),BD(300)
COMMON /SUM/PC(60),RA(60),PHI(60),SHI(60),SH11(60)
COMMON /ITG/LAST,MM,LOOP(60),NSE,LANG,LL,IP,DIR,ICOUNT(30),JJ
COMMON /LOG/RW.RF.COMP,ROTA(60),INACT(6)
COMMON /PICK/INCP(10),NI
TITLE='MAIN/VOLUME/PICKBD/INTECP/CASERP'
IF(H.EQ.1) GOTO 99
CALL FRAME
CALL BOUNDIT(0,0)
CALL KMOVAB(-2100,500.)
CALL KMOVAB(-150.,500.)
CALL KMOVAB(-2950.,850.)
CALL KTEXT('* PICK THE INTERSECTION POINT*.1)
CALL KTEXT(' WHICH IS ON THE BOUNDARY*:1)
CALL KTEXT(' HIT SPACE BAR -- PICKED*.1)
CALL KTEXT(' HIT "N" -- NONE ON BOUNDARY*.1)
CALL KTEXT(' HIT "C" -- FOR NEW PAGE*.1)
CONTINUE
IF(ROTA(K)) THEN
    AX=PC(K)*COS(PHI(K))
    AY=PC(K)*SIN(PHI(K))
    BX=RA(K)
    BY=PC(K)
    R1=RA(K)
    SLOPE=SH11(K)
ELSE
    AX=PC(K)*COS(PHI(L))
    AY=PC(K)*SIN(PHI(L))
    BX=RA(P)
    BY=PC(P)
    R1=RA(P)
    SLOPE=SH11(K)
ENDIF
IF(ABS(SLOPE)-PI/2.) LT.0.001) THEN
    D1F=R1**2-(EX-AX)**2
    IF(D1F.LT.0.) GOTO 60
    XI=DX
    X2=DX
    Y1=AY+SOPT(D1F)
    Y2=AY-COPT(D1F)
    SLOPE=1.*ED1
ELSE IF(ACOS(SLOPE),LT.0.001) THEN
    D1F=R1**2-(CY-AY)**2
    IF(D1F.LT.0.) GOTO 60
    V1=BY
    V2=BY
    X1=AX+SOPT(D1F)
    X2=AX-COPT(D1F)
ELSE
    SLOPE=TAN(SLOPE)
    AX=SLOPE
    BB=-1.
    CC=BY-SLOPE*BX
    D1=1.*BB*BE/AA/AA
CALL CIRCLE(A1.A2, R1, A1.A2, 360, 0)

END
INTERSECTING SEQUENCES generated by a RR chain which are PARTIALLY ON THE BOUNDARY.

SUBROUTINE CASERR(I,K,L,M)
CHARACTER IL
CHARACTER*35 TITLE
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,TMIN(4),THMAX(4),THS(4),THB(4)
COMMON /XY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999),
       CY3(999),CX1(999),CY4(999),TPX(999),TPY(999),FDIR(999),BD(300)
COMMON /SUM/PC(60),RA(60),PHY(60),SH(60),SH(60),SH(60)
COMMON /ITC/LAST.MM,LOOP(60),NSE,IANG,LL,IP,DIR,ICOUNT(30),JJ
COMMON /PICT/INCIC1(99),N
TITLE='MAIN/VOLUME/PICKBD/INTECP/CASERR'
IF(M.EQ.1) GOTO 99
100 CALL FRAME(TITLE)
   CALL KMOVAD(-2100.,500.)
   CALL KMOVAD(-1150.,500.)
   CALL KMOVAD(-2050.,850.)
   CALL KTEXT(* PICK THE INTERSECTION POINT*,1)
   CALL KTEXT(* WHICH IS ON THE BOUNDARY*,1)
   CALL KTEXT(* HIT SPACE BAR -- PICKED*,1)
   CALL KTEXT(* HIT "N" -- NONE ON BOUNDARY*,1)
   CALL KTEXT(* HIT "C" -- FOR NEW PAGE*,1)
   CALL BOUND(0,0)
   LAG=0
   AX=PC(K)*COS(PHY(K))
   AV=PC(K)*SIN(PHY(K))
   BX=PC(L)*COS(PHY(L))
   BV=PC(L)*SIN(PHY(L))
   R1=RA(K)
   R2=RA(L)
   D=SQRT((AV-BV)**2+(AX-BX)**2)
   RMX=MAXM(1,R2)
   RMX=MINM(1,R2)
   IF(DIS.LT.RMX.RAND.ABS(RA(K)-RA(L)),LT.0,.01) LAG=LAG-1
   IF(DIS.LT.(R1+R2)) LAG=LAG-1
   IF(DIS.GT.(RMX-RMX)) LAG=LAG+1
   IF(LAG.EQ.2) THEN
     AX=R1-R2-BX*BX-AX*AX-AY*AY
     IF(AABS(EX-AX),LT.0.01) THEN
       B=2.*((AY-BV)
     IF(AAS(EE),LT.0.01) THEN
       M=1
       GOTO 200
     ELSE
       X1=AA/BB
       X2=AY/V
       D1=(-(V1-AV)**2-R1)**2
       IF(D1.LE.0.) THEN
         X1=X
       ELSE
         X1=AX/SORT(D1)
         X2=AX/SORT(D1)
       ENDIF
     ENDIF
   ENDIF
BB=2.*(BX-AK)
AA=AA/BB
BB=2.*(BY-AV)/BB
D1=1.*CB*BB
D2=2.*(AA*BB-BB*AK-AV)
D3=AA*AK+AY*AY+AA-2.*AA*AK-R1*R1
D4=D2-D2-B1*D1*D2
IF(D4.LE.0.) THEN
  Y1=0.
  Y2=0.
ELSE
  Y1=(-D2+SQRT(D4))/(2.*D1)
  Y2=(-D2-SQRT(D4))/(2.*D1)
ENDIF
X1=AA*BB*Y1
X2=AA*BB*Y2
ENDIF
CALL CIRCLE(AX,AY,R1,1.,360.,1.)
CALL CIRCLE(BX,BY,R2,1.,360.,1.)
CALL KUPDAT
CALL KMOVAB(-2100.,400.)
CONTINUE
CALL KTEXT('CHOOSE ONE INTERSECTING PT.'
CALL KLOCATE(X1,Y1)
IF(IL.EQ.'N'.OR.IL.EQ.'n') THEN
  M=0
  GOTO 200
ELSE IF(IL.EQ.'C'.OR.IL.EQ.'c') THEN
  GOTO 100
ELSE IF(IL.EQ.'') THEN
  M=0
  D1=SQRT((X1-X)**2+(Y1-Y)**2)
  D2=SQRT((X2-X)**2+(Y2-Y)**2)
  IF(D1.LT.D2) THEN
    X=X1
    Y=Y1
  ELSE
    X=X2
    Y=Y2
  ENDIF
  CALL CIRCLE(X,Y,30.,1.,360.,20)
  ETA=ATAN2(Y-BY,X-BX)
  BETA=ATAN2(Y-AY,X-AK)
  CALL CLIPRR(K,L,BETA,ETA,X,Y)
  I=I+1
ELSE
  GOTO 300
ENDIF
ELSE
  M=1
ENDIF
RETURN
END
*******************************************************************************

CIRCLE.FOR

DESCRIPTION:
  THIS ROUTINE HAS A CAPABILITY OF
  DRAWING A CIRCLE OR A ARC.

MAIN VARIABLES:
SUBROUTINE CIRCLE(X,Y,RR,TH1,TH2,IT)
CHARACTER*10 TYPE
IF(IT.EQ.1) THEN
  TYPE='DOTTED'
ELSE IF(IT.EQ.2) THEN
  TYPE='DASHED'
ELSE IF(IT.EQ.3) THEN
  TYPE='DASHDOT'
ELSE
  TYPE='SOLID'
ENDIF
N=ABS(TH1-TH2)/6.
P1=ATAN(1.)
THIRAD=TH1+P1/180.
RX=PR*COS(THIRAD)
RY=RR*SIN(THIRAD)
CALL KMOVABIX+RX,Y*RYIDO
100
1*1.NSI-I
RAD=(SI*6.*P1/180.)*THIRAD
XCR=RR*COS(RAD)
YCR=RR*SIN(RAD)
CALL KLNTYP(TYPE)
CALL KORWAES(X*XCR,Y*YCR)CONTINUE
TH2RAD=TH2+P1/180.
RX=PR*COS(TH2RAD)
RY=PR*SIN(TH2RAD)
CALL KDFAES(X*RX,Y*RY)
CALL KLNTYP('SOLID')
RETURN
END

SUBROUTINE CLIPRP

THIS SUBROUTINE ELIMINATE RE-ENTRY SEQUENCE FOR TWO INTERSECTING SEQUENCES (JOINT TYPE R & P) PARTIALLY ON THE BOUNDARY.

K - THE MOTION SEQUENCE GENERATED BY REVOLUTE JOINT.
L - THE MOTION SEQUENCE GENERATED BY PRISMATIC JOINT.
BETA - INTERSECTING ANGLE ASSOC. TO REVOLUTE JOINT
ETA - ANGLE BETWEEN INTERSECTING POINT TO ORIGIN AND HORIZONTAL
(X,Y) - INTERSECT POINT

SUBROUTINE CLIPRP(K,L,BETA,ETA,X,Y)
CHARACTER IL
BYTE AL(2)
LOGICAL I,RF,COMP,ROTA,INACT
COMMON /LENGTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /XY,CX1(999),CY1(999),CX2(999),CY2(999),CY3(999)
COMMON /SUM/PC(60),RA(60),PHY(60),SHI(60),SHI1(60)
COMMON /T/LAST,SH,LOOP(L+G),HSE,ANG.LL,LJ,1O,R,ICOUNT(30),JJ
COMMON /LOG/RF,COMP,ROTA(6G),INACT(6)
COMMON /PICK/INC(10),NI

IF(ROTA(L)) THEN
  M=L
  K=M
ELSE
  ENDIF
  IF((SHI1(K)-SHI0(K)).GT.0.) THEN
    IF(BETA.LT.SH10(K)) THEN
      DANG*ANOM(BETA-SH10(K)+2.*PI)
    ELSE
      DANG*ANOM(BETA-SH10(K))
    ENDIF
  ELSE
    IF(BETA.GT.SH10(K)> THEN
      DANG-ANOM(SH10(K)-BETA)*2.*PI
    ELSE
      DANG-ANOM(SH10(K)-BETA)
    ENDIF
  ENDIF
ENDIF

CALL KMOVAB(TPX(LOOP(K)),TPY(LOOP(K)))
DO 200 1=LOOP(K),LOOP(K)+1,2
AA=TPY(I+1)-TPY(I)
BB=TPX(I+1)-TPX(I)
IF(AA.EQ.0. .AND. BB.EQ.0.) THEN
  ELSE
  DI=SORT(AA*AA+BB*BB)
  CALL ARROW(TPX(I),TPY(I),DIS,ANG,0)
ENDIF
CONTINUE
ENDIF

CALL KDRAW(X,Y)
DO 300 I=LOOP(L)+1,LOOP(L+1)-2
AA=TPY(I+1)-TPY(I)
BB=TPX(I+1)-TPX(I)
IF(AA.EQ.0. .AND. BB.EQ.0.) THEN
  ELSE
  DI=SORT(AA*AA+BB*BB)
  CALL ARROW(TPX(I),TPY(I),DIS,SHI1(L),0)
ENDIF
CONTINUE

CALL KUPDAT
CALL KMOVAC(-2100..350.)
CALL KTEXT(" IS THAT A RIGHT PORTION?",1)
CALL KTEXT(" ENTER "Y/N",1)

CALL KREADC(IL,HELP,"Y")
IF (L.EQ. 'N' .OR. L.EQ. 'n') GOTO 400
IF (L.EQ. 'Y' .OR. L.EQ. 'y') GOTO 500
CALL KTEXT1 (ERROR, INPUT "Y/N", I)
GOTO 302

500 CONTINUE
IF (SHI1(K).GT.SHI0(K)) THEN
   SHI(K)=BETA
   IF (SHI1(K).LT.SHI0(K)) SHI(K)=SHI1(K)+2.*PI
ELSE
   SHI(K)=BETA
   IF (SHI0(K).LT.SHI1(K)) SHI(K)=SHI1(K)-2.*PI
ENDIF

PHY(LI)=DIS2
PC(LI)=Y
RA(LI)=X
SHI0(LI)=ETA
TPX(LOOP(L))=X
TPY(LOOP(L))=Y
DO 30 M=1, LAST
   LOOP(M)=LOOP(M)-12+1
30 CONTINUE

LAST=LAST+2
DO 40 N=1, LAST
   TPX(N)=TPX(NO)
   TPY(N)=TPY(NO)
40 CONTINUE
GOTO 999

CONTINUE
AL(1)='N'
AL(2)='O'
DO 250 I=1, LOOP(K)+1-1, 2
   CALL CFROSS(TPX(I), TPY(I), 30, 30, 30, AL)
250 CONTINUE
DO 350 LOOP(L)+1=1, LOOP(L+1)-2-2
   CALL CFROSS(TPX(I), TPY(I), 30, 30, 30, AL)
350 CONTINUE

IF (SHI1(K).LT.SHI0(K)) THEN
   SHI=BETA
   IF (SHI0(K).LT.SHI1(K)) SHI0(K)=SHI1(K)-2.*PI
   DAIG=ANOM(SHI1(K)-SHI0(K))
ELSE
   SHI=BETA
   IF (SHI1(K)-SHI0(K)).GT.2.*PI SHI0(K)=SHI1(K)-2.*PI
   DAIG=ANOM(SHI0(K)-SHI1(K))
ENDIF
I1=INT(DANG*30./P1)+2
TPX(LOOP(K))=X
TPY(LOOP(K))=Y
NUM=LOOP(K+1)-LOOP(K)-11
DO 55 M=K+1,HM
LOOP(1)=LOOP(1)-NUM
55 CONTINUE
LAST=LAST-11
DO 55 N=K1,LAST
NO=N-11
TPX(N)=TPX(NO)
TPY(N)=TPY(NO)
55 CONTINUE
CALL DRWLOP(K,2,0)
KI=LOOP(L)+11
TPX(KI)=X
TPY(KI)=Y
NUM=LOOP(L+1)-KI-11
DO 35 M=KI+1,HM
LOOP(1)=LOOP(1)-NUM
35 CONTINUE
LAST=LAST-11
DO 55 N=KI+1,LAST
NO=N-11
TPX(N)=TPX(NO)
TPY(N)=TPY(NO)
55 CONTINUE
PHY(L)=DIS1
CALL DRWLOP(L,2,0)
CALL KPAUSED.D)
999 IF(ROTA(L)) THEN
H=L
L=K
K=M
ELSE
ENDIF
RETURN
END

SUBROUTINE CLIPRR
THIS SUBROUTINE ELIMINATE RE-ENTRY SEQUENCE FOR TWO
INTERSECTING SEQUENCES PARTIALLY
ON THE BOUNDARY. (JOINT TYPE R & R)

K - THE MOTION SEQUENCE GENERATED BY 1ST REVOLTE JOINT.
L - THE MOTION SEQUENCE GENERATED BY 2ND JOINT.
BETA - INTERSECTING ANGLE ASSOC. TO REVOLVE JOINT
ETA - ANGLE BETWEEN INTERSECTING POINT TO ORIGIN
AND HORIZONTAL
(X,Y) - INTERSECT POINT

SUBROUTINE CLIPRR(K,L,BETA,ETA,X,Y)
BYTE AL(2)
CHARACTER IL
COMMON /LENGTH/RATIO,A(4),BL(4),P1,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMI(4),THMAX(4),THS(4),THE(4)
COMMON /XY/XY(999),CX1(999),CX2(999),CX3(999),CV1(999),CY1(999),CV2(999),
1 CY3(999),CX4(999),CPX(999),CYP(999),FDIR(999),BD(300)
COMMON /SUM/PC(60),RA(60),PHY(60),SHI0(60),SHII(60)
COMMON /LAST.MM,LOOP(60),HSE.IANG.LL.IP.IDIR.ICOUNT(30),J
COMMON /IPICK/INC(10),NIT
IF((SHII(L)-SHI0(L)).GT.0.) THEN
  IF(BETA.LT.SHII(L)) THEN
    DANG=ANOM(BETA-SHI0(L)-2.*PI)
    ELSE
    DANG=ANOM(ETA-SHI0(L)-2.*PI)
    ENDIF
  ELSEIF(BETA.GT.SHII(L)) THEN
    DANG=ANOM(BETA-SHI0(L)-2.*PI)
    ELSE
    DANG=ANOM(ETA-SHI0(L)-2.*PI)
    ENDIF
  ENDIF
ENDIF

CALL KRndAB(X,Y)
DO 360 I=LOOP(L)+1-LOOP(L)-12
  AA=TPY(I+2)-TPY(I)
  BB=TPX(I+2)-TPX(I)
  IF(AA.EQ.0. .AND. BB.EQ.0.) THEN
    ELSE
    IF(AA.EQ.0. .AND. BB.EQ.0.) THEN
      ELSE
    ENDIF
  ENDIF
CONTINUE
CALL KUPDAT
CALL K.getText("IS THAT RIGHT PORTION?",1)
CALL KgetText(", ENTER 'Y/N',.1)
360 CALL Kreadc(IL,IHELP,'Y')
IF(IL.EQ. 'N' .OR. IL .EQ. 'n') GOTO 400
IF(IL.EQ.'V'.OR.IL.EQ.'v')GOTO500
CALLKTEXT('ERROR, INPUT V/N',I)
GOTO502

500CONTINUE
IF(SHI1(I).GT.SHI0(K))THEN
SHI1(K)=BETA
IF(SHI1(I).LT.SHI0(K))SHI1(K)=SHI1(K)+2.*PI
ELSE
SHI1(K)=BETA
IF(SHI0(K).LT.SHI1(K))SHI1(K)=SHI1(K)-2.*PI
ENDIF
IF(SHI1(L).GT.SHI0(L))THEN
SHI0(L)=ETA
IF(SHI0(L).LT.SHI1(L))SHI0(L)=SHI0(L)-2.*PI
ELSE
SHI0(L)=ETA
IF((SHI0(L)-SHI1(L)).GT.2.*PI)SHI0(L)=SHI0(L)-2.*PI
IF(SHI0(L).LT.SHI1(L))SHI0(L)=SHI0(L)+2.*PI
ENDIF

TPX(LOOP(L))=X
TPY(LOOP(L))=Y
NUM=LOOP(L)+1,MH
LOOP(M)=LOOP(M)-NUM
CONTINUE
LAST=LAST-NUM
DO60N=LOOP(L)+1,LAST
NO=N+NUM
TPX(NO)=TPX(NO)
TPY(NO)=TPY(NO)
CONTINUE
KI=LOOP(K)+I
TPX(KI)=X
TPY(KI)=Y
NUM=LOOP(K)+I-KI-1
DO30M=KI+1,MH
LOOP(M)=LOOP(M)-NUM
CONTINUE
LAST=LAST-NUM
DO40N=KI+1,LAST
NO=N+NUM
TPX(NO)=TPX(NO)
TPY(NO)=TPY(NO)
CONTINUE
GOTO999
CONTINUE
AL(I)=N
AL(2)=O
DO250I=LOOP(K),LOOP(K)+I1-I
CALLCROSS(TPX(I),TPY(I),30.,30.,AL)
CONTINUE
DO350I=LOOP(L)+NUM+1,LOOP(L)+2
CALLCROSS(TPX(I),TPY(I),30.,30.,AL)
CONTINUE
IF(SHI1(K).GT.SHI0(K))THEN
SHI0(K)=BETA
IF(SHI1(K).LT.SHI0(K))SHI0(K)=SHI1(K)-2.*PI
ENDIF=ANOM(SHI1(K)-SHI0(K))
IF($HI0(K)-SH11(K)).GT.2.*PI $HI0(K)=SH11(K)+2.*PI
IF($HI0(K)<SH11(K)) SH11(K)=SH10(K) 
DANG=ANOM($HI0(K)-SH11(K))
ENDIF
110 INT(DANG*30./PI)+2
IF((SH11(L)-SH10(L)).GT.0.) THEN
  SH11(L)=ETA
  IF(SH11(L).LT.SH10(L)) SH11(L)=SH10(L)+2.*PI
  DANG=SH11(L)-SH10(L)
  IF(DANG.GT.2.*PI) DANG=DANG-2.*PI
ELSE
  SH11(L)=ETA
  IF(SH11(L).GT.SH10(L)) SH11(L)=SH10(L)+2.*PI
  DANG=SH11(L)-SH10(L)
  IF(DANG.GT.2.*PI) DANG=DANG-2.*PI
ENDIF
120 INT(DANG*30./PI)+1
TPX(LOOP(K))=X
TPY(LOOP(K))=Y
NUM=LOOP(K)+1-LOOP(K)-11
DO 55 M=K+1,MM
  LOOP(M)=LOOP(M)-NUM
  CONTINUE
LAST=LAST-NUM
DO 65 N=LOOP(K)+1,LAST
  HO=N+NUM
  TPX(N)=TPX(NO)
  TPY(N)=TPY(NO)
  CONTINUE
CALL DROOLP(1,2,D)
  KI=LOOP(K)+12
  TPX(KI)=X
  TPY(KI)=Y
  NUM=LOOP(K)+1-KI-1
  DO 35 M=L+1,MM
    LOOP(M)=LOOP(M)-NUM
  CONTINUE
LAST=LAST-NUM
DO 45 N=K+1,LAST
  HO=N+NUM
  TPX(N)=TPX(NO)
  TPY(N)=TPY(NO)
  CONTINUE
CALL DROOLP(L,2,D)
CALL KPAUSE(D,D)
999 RETURN
END

C:---------------------------------------------------------------
C: CROSS.FOR
C: THIS ROUTINE IS TO DRAW CROSS LINES AT DESIRE POSITION (sx,sy)
C: with length 2*rrr.
C: and write characters at end of lines with ascii code (al)
C: if rrr=rrrr then draw arrow head on both end of cross line.
C:---------------------------------------------------------------
C
SUBROUTINE CROSS(SX,SY,RR,RRR,AL)
BYTE AL(2)
100 CALL PHOVAB(SX,SY+RRR)
SUBROUTINE DEMO

THIS SUBROUTINE provides data for four example robots. Variables used please see the main program.

CALL KDRWAB(SX, SY-RRR)
CALL KMOVAB(SX-RRR, SY)
CALL KDRWAB(SX+RRR, SY)
CALL KTEXT('X', 0)
IF (PR .EQ. RRR) THEN
  CALL KMOVAB(SX+RRR/2, SY+RRR/2)
  IF (AL(1).EQ. 'Y') THEN
    CALL KTEXT('Y', 0)
  ELSE IF (AL(1).EQ. 'Z') THEN
    CALL KTEXT('Z', 0)
  ELSE
    ENDIF
ELSE
  CALL KMOVAB(SX-RRR-10., SY+10.)
  CALL KDRWAB(SX+10., SY-10.)
  CALL KMOVAB(SX+10., SY-10.)
ENDIF
RETURN
END

SUBROUTINE DEMO IN

CHARACTER*1, TYPE
CHARACTER*20, MAKER
CHARACTER*35, TITLE
CHARACTER*10, MODELCOMMON /ANKLE/SUE_EP.THMINU, THMAX(4), THS(4), THB(4)
COMMON /LENGTH/RAT 10., AM 11, BL 14, PI, RTMS(4), TH(4), TWIST(4)
COMMON /CHP/MODEL, MAKER, TYPE, TITLE
COMMON /PAR/ALPHA(6), THETA(6), OFFSET(6), HLEN(6)
TITLE = 'INPUT/DEMO'
CALL FRAME(TITLE)
CALL KMOVAB('1, 307.1')
CALL KDRWAB('1022., 307.1')
CALL KMOVAB('510., 750.1')
CALL KDRWAB('1010., 30.1')
CALL KTEXS12('025')
CALL KMOVAB('50., 650.1')
CALL KTEXT('CINCINNATI MILACRON', 0)
CALL KMOVAB('50., 650.1')
CALL KTEXT('CINCINNATI MILACRON', 0)
CALL KMOVAB(550.,600.)
CALL KTEXT('T3 - 363*.0')
CALL KMOVAC(550.,550.)
CALL KTEXT('JOINT TYPE: 1R-2P-3R'.0)
CALL KMOVAB(58.,300.)
CALL KTEXT('CINCINNATI MILACRON'.0)
CALL KMOVAB(550.,250.)
CALL KTEXT('T3 - 800'.0)
CALL KMOVAC(58.,200.)
CALL KTEXT('JOINT TYPE: 3P-3R'.0)
CALL KMOVAB(2.,8.)
CALL KTEXT('REIS MACHINE CORPORATION'.0)
CALL KLOCAT(X,Y,IL)
IF(X.LT.510 .AND. Y.GT.387) THEN
  GOTO 40
ELSE IF(X.LT.510 .AND. Y.LT.367) THEN
  GOTO 50
ELSE IF(X.GT.510 .AND. Y.GT.367) THEN
  GOTO 60
ELSE
  GOTO 70
ENDIF
70 MAKER="REIS MACHINE CORP."
MODEL="RR-625/650"
DO 41 JTYPE(1)='P'
41 JTYPE(1)='P'
  SWEEP=48.
  THS(1)=-90.
  THB(1)=180.
  THS(2)=-175.
  THB(2)=175.
  THS(3)=-135.
  THB(3)=135.
  BL(1)=20.
  BL(2)=20.
  BL(3)=16.941176
  TWIST(1)=0.
  TWIST(2)=0.
  TWIST(3)=90.
  HLENGT(2)=BL(1)
  HLENGT(3)=BL(2)
  ALPHA(4)=TWIST(3)*PI/180.
  RETURN
60 MAKER="CINCINNATI MILACRON"
MODEL="T3-363"
DO 61 JTYPE(1)='R'
61 JTYPE(1)='R'
  SWEEP=300.
  THS(1)=17.
  THB(1)=41.
THS(2) = 3.0
THB(2) = 57.0
THS(3) = 90.0
THB(3) = 0.0
BL(1) = 41.0
BL(2) = 57.0
BL(3) = 0.0
TWIST(1) = 90.0
TWIST(2) = 90.0
TWIST(3) = 90.0
ALPHA(3) = P1/2
ALPHA(4) = P1/2
THETA(2) = P1/2
THETA(3) = P1/2
RETURN

MAKE='CINCINNATI MILACRON'
MODEL='T3-300'
DO 51 I = 1, 3
   JTYPE(I) = 'R'
   SWEEP = 0.0
   THS(I) = 0.0
   THB(I) = 0.0
   THS(I) = 0.0
   THB(I) = 0.0
END

51
THS(1) = 36.0
THB(1) = 36.0
THS(2) = 36.0
THB(2) = 36.0
THS(3) = 36.0
THB(3) = 36.0
ALPHA(1) = P1/2
ALPHA(2) = P1/2
ALPHA(3) = P1/2
ALPHA(4) = P1/2
THETA(1) = P1/2
THETA(2) = P1/2
THETA(3) = P1/2
RETURN

MAKE='CINCINNATI MILACRON'
MODEL='T3-746'
DO 42 I = 1, 6
   JTYPE(I) = 'R'
END

42
CONTINUE
SWEEP = 270.0
THS(1) = 60.0
THB(1) = 147.0
THS(2) = 30.0
THB(2) = 135.0
THS(3) = -119.0
THB(3) = 119.0
BL(1) = 44.0
BL(2) = 55.0
BL(3) = 6.0
TWIST(1) = 90.0
TWIST(2) = 60.0
TWIST(3) = 60.0
HLENGTH(2) = CL(1)
SUBROUTINE TO CLEAR THE SCREEN AND DRAW A FRAME

SUBROUTINE FRAME(TITLE)
CHARACTER*35 TITLE
CHARACTER*15 RTIME
CHARACTER*9 RDATE
CALL KCLEAP
CALL KWPRT(0..1..0..0..76171875)
CALL HKOVAB(0..0.)
CALL KORVAB(1024..0.)
CALL KORVAB(1024..780.)
CALL KORVAB(0..780.)
CALL KORVAB(0..0.)
CALL KMOVAB(0..30.)
CALL KORVAB(1024..0.)
CALL KMOVAB(0..750.)
CALL KORVAB(1024..750.)
CALL KTSIZE(0..622)
CALL KMOVAB(5..755.)
CALL KTEXT('WORK SPACE - ')
CALL KTEXT(TITLE,0)
CALL KTEXT(RTIME,0)
CALL KDATE(RDATE)
CALL KUPDAT
RETURN
END

SUBROUTINE HAND(1)
THIS SUBROUTINE DRAW the ROBOT's end-effector with the orientation along the last non-zero link.

SUBROUTINE HAND (1)
LOGICAL IW,RF,COMP,ROTA,INACT
COMMON /SUM/PC(0).RA(60).PHY(60).SHI0(60).SHI1(60)
COMMON /LY/CH(LAST,MM.LOOP(60).NSE.IANG.LL.IP.IDIR.ICOUNT(30))
COMMON /LOG/IW.RF,COMP,ROTA(60).INACT(6)
IF(CL(3).EQ.0.) THEN
IF(CL(3).EQ.0.) THEN
X=CX2(Loop(1))=CX1(Loop(1))
Y=CY2(Loop(1))=CY1(Loop(1))
ELSE IF(BL(4).EQ.0.) THEN
X=CX3(Loop(1))=CX2(Loop(1))
Y=CY3(Loop(1))=CY2(Loop(1))
ELSE
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDIF
ENDER

170
SUBROUTINE HANDRF

THIS SUBROUTINE ALLOWS THE USER TO ASSIGN HAND ORIENTATION FOR THE MANIPULATOR. THE ORIGINAL STORAGE ARRAYS WILL BE DESTROYED BY THIS SUBROUTINE.

The angle of hand orientation is to the horizontal direction of the generating plane.

SUBROUTINE HANDRF(\*'\*)
LOGICAL IW,RF,CMP,ROTA,INACT
COMMON /ITG/LAST,MM,LOOP(50),NSE,IANG,LI,IP,IDIR,JCOUNT(30),JJ
COMMON /LOG/IW,RF,CMP,ROTA(50),INACT(5)
RF=.TRUE.
100 CALL KTXSIZ(0,12)
CALL KMOVAB(0.5,45.0)
CALL KTEXT("HAND ORIENTATION (DEG)\*'0")
CALL KREAD1(IANG,0,88,0)
IF(IW) THEN
WRITE(2,122) IANG
122 FORMAT(//" ANGLE OF HAND TO HORIZONTAL\*'I4,' DEGREE"//)
ELSE
ENDIF
IP=1
JCOUNT(I)=1
LL=1
MM=1
LAST=1
CALL SEQUE2(*170,*180)
170 RETURN 1
180 RETURN 2

X=CX4(LOOP(I))-CX3(LOOP(I))
Y=CV4(LOOP(I))-CV3(LOOP(I))
ENDIF
ANGL=ATAN2(Y,X)
HLEN=0.14*BL(3)
IF(HLEN.LT.50.0) HLEN=30.
DY1=HLEN*SIN(ANGL+PI/4.)
DY2=HLEN*SIN(ANGL+3*PI/4.)
DY3=HLEN*SIN(ANGL-PI/4.)
DY4=HLEN*SIN(ANGL-3*PI/4.)
DX1=HLEN*COS(ANGL+PI/4.)
DX2=HLEN*COS(ANGL+3*PI/4.)
DX3=HLEN*COS(ANGL-PI/4.)
DX4=HLEN*COS(ANGL-3*PI/4.)
122 CALL KMOVAB(0,0)
CALL KDRWAB(CX4(LOOP(I)),CY4(LOOP(I)))
CALL KDRWAB(CX2(LOOP(I)),CY2(LOOP(I)))
CALL KDRWAB(CX3(LOOP(I)),CY3(LOOP(I)))
IF(BL(4).EQ.0.) THEN
ELSE
CALL KDRWAB(CX4(LOOP(I)),CY4(LOOP(I)))
ENDIF
CALL KMOVRL(DX1,DY1)
CALL KDRWRL(DX2-DX1,DY2-DY1)
CALL KDRWRL(DX3-DX2,DY3-DY2)
CALL KDRWRL(DX4-DX3,DY4-DY3)
RETURN
SUBROUTINE INTECP

THIS SUBROUTINE SOLVES intersecting points for two motion sequences which are partially on the workspace boundary. Depending on the joint type used, the program calls the CASE**

to solve the intersecting points. The portions which lie inside the workspace are then removed by calling CLIP** subroutine.

The program checks an identified intersecting sequence with every other identified intersecting sequence.

I: counter for 1st sequence
J: counter for 2nd sequence
II: label to avoid double check
K: the 1st sequence to be checked
L: the 2nd sequence to be checked
M: INC(1): the identified intersecting sequence number
N: total number of identified intersecting sequences

SUBROUTINE INTECP
LOGICAL IV,RF,COMP,ROTA,INACT
COMMON /LOG/IV,RF,COMP,ROTA(IN),INACT(IN)
COMMON /PICK/INC(I),N
I=0
J=0
10 I=I+1
J=J+1
20 K=INC(I)
L=INC(J)
IF(ROTA(K) .AND. ROTA(L)) THEN
   CALL CASERR(I,K,L,M)
ELSE IF(.NOT.ROTA(K) .AND. .NOT.ROTA(L)) THEN
   CALL CASERR(I,K,L,M)
ELSE
   CALL CASEPP(I,K,L,M)
ENDIF
IF(I.LT.K-1) GOTO 10
IF(J.LT.N-1) GOTO 20
30 GOTO 30
   CALL KMOVAB(-2100,0)
   CALL KTETXT('INTERSECTED SEQUENCES','I')
   CALL KTETXT('SORTING FINISHED','I')
   CALL KPAUSE(0,0)
RETURN
END

SUBROUTINE KINPUT

THIS SUBROUTINE ACCEPT DATA FROM KEYBOARD INPUT

The joint parameters are to be arranged so that joints 2, 3, 4 lie in the generating plane. Total link length is normalized to 1 unit.

JTYPE: joint type, either R for revolute or P for prismatic
MAKER: the robot manufacturer

MODEL: the model of the robot

S: intermediate character value used to hold data

I: logical variable, if true, write a data file

R: logical variable, if true, Hand orientation is specified

ROTA: logical array, if true, the motion sequence is generated by a revolute joint

INACT: logical array, if true, the joint is inactive

COMP: logical variable, if true, the robot is in an unstable configuration under the applied force

RATIO: the total physical link length

A(i): the array to hold lengths of joint members

BL(i): the array to hold maximum lengths of joint members

PI: constant PI=3.14159

RTH(I): the motion range of joint I

Th: the angle between joint 1-1 to 1

Twist(I): constant angle between two prismatic joints

LAST: counter for each small motion increment, variable

MAX: the maximum step of motion increment

M: counter for each motion sequence, variable

NSE: maximum number of motion sequences

LOOP(I): the starting position of motion sequence I

JANG: the degree of hand orientation to the horizontal

I: number of joint

SWEEP: the motion range of 1st joint

THMIN(I): the minimum motion position of joint I

THMAX(I): the maximum motion position of joint I

THS(I): the minimum motion position of joint I (degree)

THU(I): the maximum motion position of joint I (degree)

ALPHA(I): the twist angle between joint i-1 to I

THETA(I): the Joint angle of joint I

OFFSET(I): the offset of joint I

HLEN(I): the link length I

SUBROUTINE KINPUT

EXTERNAL ANOM

CHARACTER 1L,JTYPE

CHARACTER*1 S,MODEL

CHARACTER*35 TITLE

LOGICAL IV,RF,COMP,ROTA,INACT

COMMON / LENGTH/PATIO,A(I),BL(I),RTH(I),TH(I),TWIST(I)

COMMON / LAGT,HM,LOOP(I),NSE,JANG,LL,IP,1DIR,1COUNT(I)

JNUM,NSC,1ANG,LL,IP,1DIR,1COUNT(I)

COMON /ANGLE/SWEEP,THMIN(I),THMAX(I),THS(I),THU(I)

COMON /CHF/MODEL,MAKER,JTYPE(6)

COMON /OBJ/IV,RF,COMP,ROTA(6),INACT(6)

COMMON /PAR/ALPHA(I),THETA(I),OFFSET(I),HLEN(I)

IV=.FALSE.

DO 123 I=1,6

ALPHA(I)=0.

THETA(I)=0.

OFFSET(I)=0.

HLEN(I)=0.

CONTINUE

123 CONTINUE

DO 456 I=1,6

LL(I)=0.

1456 CONTINUE

A(I)=0.

BL(I)=0.

TWIST(I)=0.
CONTINUE
TITLE='MAIN/KINPUT'
CALL FRAME(TITLE)
CALL KMOVAB(SO., 700.)
CALL KTTEXT('PLEASE CHOOSE ONE:',0)
CALL KMOVAB(SO., 640.)
CALL KTTEXT(' 1 - FOR DEMOSTRATION',0)
CALL KMOVAB(SO., 580.)
CALL KTTEXT(' 2 - ENTER YOUR OWN DATA',0)
CALL KMOVAB(SO., 520.)
CALL KTTEXT(' 3 - FOR INTRODUCTION...<1>',0)
CALL KREADC(IL,HELP, '1')
CALL KINFD(I)
IF(IL.EQ. '1') THEN
   CALL DEMOIN
   ALPHA(5)=PI/2.
   THETA(5)=PI/2.
   OFFSET(6)=DL(3)
   GOTO 1000
ELSE IF(IL.EQ. '2') THEN
   GOTO 14
ELSE IF(IL.EQ. '3') THEN
   CALL WELCOM
   GOTO 10
ELSE
   CALL KTTEXT(' INVALID NUMBER',1)
   GO TO 33
ENDIF
CALL FRAME(TITLE)
CALL KMOVAB(SO., 700.)
CALL KTTEXT('DO YOU WANT TO KEEP A DATA FILE <N>..',0)
CALL KREADC(IL,HELP, 'N')
CALL KINFD(I)
IF(IL.EQ. 'N' OR. IL.EQ. 'y') THEN
   IV=TRUE.
   OPEN(UNIT=2,NAME="WORKSP.TYPE="NEW")
   CALL KTTEXT( ' THE DATA FILE WILL BE "WORKSP.DAT",1)
ELSE
   ENDIF
CALL KMOVAB(SO., 600.)
CALL KTTEXT(' ENTER MANUFACTURER OF THE ROBOT:',0)
CALL KREADC(MAKER,HELP, ' ')
CALL KINFD(I)
CALL KTTEXT(' ENTER MODEL OF THE ROBOT:',0)
CALL KREADC(MODEL,HELP, ' ')
CALL KINFD(I)
CALL KTTEXT(' ENTER HOW MANY JOINTS THE ROBOT HAS:<6>',0)
CALL KREADC(JNUH,08,6)
CALL KTTEXT(' ENTER TYPES OF FIRST 3 JOINTS IN ORDER',0)
CALL KTTEXT(' R -- REVOLUTE JOINT',1)
CALL KTTEXT(' P -- PRISMATIC JOINT',1)
CALL KTTEXT( ' EXAMPLE: RPP ',0)
CALL KREADC(JTYPE,HELP, 'RRRPRR')
CALL KINFD(I)
DO 30 I=1,3
JTYPE(I)= 'R'
IF(JTYPE(I).EQ. 'R') JTYPE(I)= 'R'
IF(JTYPE(I).EQ. 'p') JTYPE(I)= 'p'
IF(JTYPE(I).EQ. 'P') JTYPE(I)= 'p'
CONTINUE
CALL FRAMETITLE)
CALL KMVAB(5.,705.)
CALL KTEXT(" ENTER PARAMETERS O F  JOINT 1",1)
IF(JTYPE(1).EQ.'P') THEN
  IF(JTYPE(1).EQ.'P') THEN
    CALL KTEXT(" JOINT ANGLE = 0",1)
    CALL KTEXT(" TWIST ANGLE = 90",1)
    CALL KTEXT(" LINK LENGTH = 0",1)
    ALPHA(1)=PI/2.
  ELSE
    CALL KTEXT(" JOINT ANGLE = 0",1)
    CALL KTEXT(" TWIST ANGLE = 0",1)
    CALL KTEXT(" LINK LENGTH = 0",1)
  ENDIF
ELSE
  IF(JTYPE(1).EQ.'P') THEN
    CALL KTEXT(" LINK LENGTH = 0",1)
    CALL KTEXT(" TWIST ANGLE = 90",1)
    CALL KTEXT(" OFFSET = 0",1)
  ELSE
    CALL KTEXT(" OFFSET = 0",1)
    CALL KTEXT(" TWIST ANGLE = 90",1)
    CALL KTEXT(" LINK LENGTH = 0",1)
  ENDIF
ENDIF
CALL KTEXT(" ENTER JOINT SWEEP ANGLE = 0")
ENDIF
CALL KREAD(1,SWEEP,'1001.0.,)
CALL KLIFD1)
CALL KTEXT(" ENTER PARAMETERS O F  JOINT 2",1)
IF(JTYPE(2).EQ.'P') THEN
  IF(JTYPE(1).EQ.'P') THEN
    CALL KTEXT(" JOINT ANGLE = 90",1)
    CALL KTEXT(" LINK LENGTH = 0",1)
    CALL KREAD(1,TWIST(1),'1002.90.,)
    CALL KLIFD1)
    CALL KTEXT(" TWIST(1),=90.
    ALPHA(2)*TWIST(1)*PI/160.
  ELSE
    CALL KTEXT(" JOINT ANGLE = 90",1)
    CALL KTEXT(" LINK LENGTH = 0",1)
    CALL KTEXT(" TWIST(1),=90.
    THETA(2)=PI/2.
  ENDIF
THETA(2)=PI/2.
ELSE
  IF(JTYPE(3).EQ.'P') THEN
    CALL KREAD(1,TWIST(1),'1005.90.,)
    CALL KLIFD1)
    CALL KTEXT(" TWIST(1),=90.
    ALPHA(2)*TWIST(1)*PI/160
  ELSE
    CALL KTEXT(" JOINT ANGLE = 0",1)
    CALL KTEXT(" LINK LENGTH = 0",1)
CALL KTEXT(* TWIST ANGLE = 90°.1)
TWIST(1)=90.
ALPHA(2)=PI/2.
ENDIF

1003
CALL KTEXT(* ENTER TRAVEL (OFFSET) LENGTH = *,0)
CALL KREADR(1,BL(1),*1003,1.)
CALL KLIND(1)
ELSE
IF(JTYPE(1).EQ.'P') THEN
IF(JTYPE(2).EQ.'P') THEN
CALL KTEXT(* TWIST ANGLE = 90°.1)
CALL KTEXT(* OFFSET = 0°.1)
CALL KTEXT(* LINK LENGTH = 0°.1)
ALPHA(2)=PI/2.
TWIST(1)=90.
ELSE
CALL KTEXT(* OFFSET = 0°.1)
CALL KTEXT(* TWIST ANGLE = 0°.1)
CALL KTEXT(* ENTER LINK LENGTH = *,0)
CALL KREADR(1,BL(1),*1004,1.)
CALL KLIND(1)
ENDIF
ELSE
IF(JTYPE(3).EQ.'P') THEN
CALL KTEXT(* TWIST ANGLE = 90°.1)
CALL KTEXT(* OFFSET = 0°.1)
CALL KTEXT(* LINK LENGTH = 0°.1)
ALPHA(2)=PI/2.
TWIST(1)=90.
ELSE
CALL KTEXT(* OFFSET = 0°.1)
CALL KTEXT(* TWIST ANGLE = 0°.1)
CALL KTEXT(* ENTER LINK LENGTH = *,0)
CALL KREADR(1,BL(1),*1006,1.)
CALL KLIND(1)
ENDIF
ENDIF

1007
CALL KTEXT(* ENTER MOTION RANGE = *,1)
CALL KTEXT(* THE MIN. JOINT 2 = *,0)
CALL KREADR(1,THS(1),*1007,0.)
1008
CALL KTEXT(* THE MAX. JOINT 2 = *,0)
CALL KREADR(1,THB(1),*1008,1.)
CALL KLIND(1)
CALL KTEXT(* ENTER PARAMETERS OF JOINT 3 = *,1)
IF(JTYPE(3).EQ.'P') THEN
IF(JTYPE(2).EQ.'P') THEN
CALL KTEXT(* JOINT ANGLE = 90°.1)
CALL KTEXT(* LINK LENGTH = 0°.1)
CALL KTEXT(* TWIST ANGLE = 90°.1)
1009
CALL KTEXT(* ENTER TOTAL TRAVEL=OFFSET) LENGTH = *,0)
CALL KREADR(1,BL(2),*1009,1.)
CALL KLIND(1)
THETA(3)=PI/2.
ELSE
CALL KTEXT(* TWIST ANGLE = 90°.1)
CALL KTEXT(* LINK LENGTH = 0°.1)
CALL KTEXT(* JOINT ANGLE = 0°.1)
1010
CALL KTEXT(* ENTER TOTAL TRAVEL=OFFSET) LENGTH = *,0)
CALL KREADR1.BL(1),*1010,1.)
CALL KLINFD(1)
ENDIF

1029

TWIST(2)=90.
ALPHA(3)=p/2.
ELSE
 CALL KTEXT:* TWIST ANGLE = 0°,1)  
 CALL KTEXT:* OFFSET = 0,1) 
 CALL KTEXT:* ENTER LINK LENGTH *=0.1)  
 CALL KREADR1.BL(2),*1029,1.) 
 CALL KLINFD(1) 
 TWIST(2)=90.
ENDIF

1012

CALL KTEXT:* THE MOTION RANGE*:1)  
 CALL KTEXT:* THE MIN. JOINT 3*:0) 
 CALL KREADR1.THS(2),*1012,0.)
1013

CALL KTEXT:* THE MAX. JOINT 3*:0)  
 CALL KREADR1.THB(2),*1013,1.)
CALL FRAME(TITLE) 
IF:JNUM.GT.6 THEN
 CALL KMOVAB(5,720.)
 CALL KTEXT:* ENTER PARAMETERS OF JOINT 4*:1)  
 CALL KTEXT:* TWIST ANGLE = 0°,1) 
 CALL KTEXT:* OFFSET = 0°,1) 
 CALL KTEXT:* LINK LENGTH = 0°,1) 
 TWIST(3)=0.
ALPHA(4)=0.
1015

CALL KTEXT:* THE MOTION RANGE*:1)  
 CALL KTEXT:* THE MIN. JOINT 4*:0) 
 CALL KREADR1.THS(3),*1015,0.)
1016

CALL KTEXT:* THE MAX. JOINT 4*:0)  
 CALL KREADR1.THB(3),*1016,1.)
CALL KLINFD(2)
 CALL KTEXT:* PARAMETERS OF JOINT 5 ARE 1*:1) 
 CALL KTEXT:* TWIST ANGLE = 90°,1) 
 CALL KTEXT:* LINK LENGTH = 0°,1) 
 CALL KTEXT:* JOINT ANGLE = 0°,1) 
 ALPHA(5)=p/2.
1021

CALL KTEXT:* ENTER OFFSET*:0)  
 CALL KREADR1.BL(3),*1021,0.) 
 offset(5)=bl(3)
 CALL KLINFD(2)
 CALL KTEXT:* PARAMETERS OF JOINT 6*:1)  
 CALL KTEXT:* TWIST ANGLE = 90°,1) 
 CALL KTEXT:* LINK LENGTH = 0°,1) 
 CALL KTEXT:* JOINT OFFSET = 0°,1) 
 alpha(6)=p/2.
1025

CALL KTEXT:* THE MOTION RANGE*:1)  
 CALL KTEXT:* THE MIN. JOINT 6*:0) 
1026

CALL KREADR1.THS(4),*1026,0.)
 CALL KTEXT:* THE MAX. JOINT 6*:0) 
 CALL KREADR1.THB(4),*1026,1.)
 CALL KLINFD(2) 
 CALL KTEXT:* PARAMETERS OF JOINT 7*:1)  
 CALL KTEXT:* TWIST ANGLE = 90°,1) 
 CALL KTEXT:* LINK LENGTH = 0°,1) 
 CALL KTEXT:* JOINT ANGLE = 0°,1) 
 CALL KTEXT:* ENTER OFFSET*:0)  
 CALL KREADR1.BL(4),*1027,0.)
ELSE
CALL KMOVAB15,.708.)
CALL KTEXT(' ENTER PARAMETERS OF JOINT 4:1)
CALL KTEXT(' TWIST ANGLE = 90'.1)
CALL KTEXT(' OFFSET = 0'.1)
TWIST(3)=90.
ALPHA(4)=PI/2.
IF(JTYPE(3).EQ.'P') THEN
  IF(JTYPE(2).EQ.'R') THEN
    CALL KTEXT(' ENTER LINK LENGTH = *.0)
    CALL KREADR(1,BL(2),*1024.1.)
    CALL KLINFD(1)
    HLEN(T(1)=BL(2)
  ELSE
    CALL KTEXT(' LINK LENGTH = 0'.1)
ENDIF
ELSE
  CALL KTEXT(' LINK LENGTH = 0'.1)
ENDIF
1022
CALL KTEXT(' THE MOTIVON RANGE *.1)
CALL KREADP 1.* THIS (3).0)
1023
CALL KTEXT(' THE MIN. JOINT 4:1)
CALL KREADR(1,THD(3),*1023.1.)
CALL KLINFD(1)
CALL KTEXT(' ENTER LINK LENGTH = 90'.1)
CALL KTEXT(' TWIST ANGLE = 0'.1)
CALL KTEXT(' LINK LENGTH = 0'.1)
CALL KTEXT(' JOINT ANGLE = -90'.2)
ALPHA(5)=PI/2.
CALL KTEXT(' USE STANDARD LENGTH FOR END-EFFECTOR?*.1)
CALL KTEXT(' IT IS 0.15 OF TOTAL LENGTH <Y> = *.0)
CALL KREADC(IL,HELP,.Y')
IF(IL.EQ.'Y'.OR.[L.EQ.'N') THEN
  IF(JTYPE(1).EQ.'P') THEN
    BL(3)=0.1764705*(BL(1)+BL(2)+SWEEP)
  ELSE
    BL(3)=0.1764705*(BL(1)+BL(2))
  ENDIF
ELSE
  CALL KTEXT(' ENTER LENGTH OF END-EFFECTOR = *.0)
  CALL KREADR(1,BL(3),*1018.0.15)
  CALL KLINFD(1)
ENDIF
1018
CALL KTEXT(' ENTER OTHER PARAMETERS OF JOINT 6:1)
CALL KTEXT(' TWIST ANGLE = 0'.1)
CALL KTEXT(' LINK LENGTH = 0'.1)
CALL KTEXT(' JOINT ANGLE = 0'.1)
THETA(6)=PI/2.
OFFSET(6)=BL(3)
ENDIF
CALL KMOVAC(2,.8.)
CALL KTEXT(' PLEASE WAIT ....'.0)
CALL KUPDAT
1020
IF(JTYPE(1).EQ.'P') THEN
  RATIO=SWEEP+BL(1)+BL(2)+PL(3)+PL(4)
  SWEEP=SWEEP*1000./RATIO
ELSE
  RATIO=BL(1)+BL(2)+BL(3)+BL(4)
ENDIF
DO 80 I = 1, 4
TWIST(I) = TWIST(I) * PI / 180.
BL(I) = 1800. * BL(I) / RATIO
CONTINUE
DO 81 J = 1, 6
HLENGT(I) = HLENGT(I) / RATIO
OFFSET(I) = OFFSET(I) / RATIO
CONTINUE

IF (JTYPE(2) .EQ. 'P') THEN
  THMIN(I) = THS(I) * 1000. / RATIO
  THMAX(I) = THB(I) * 1000. / RATIO
  TEMP = THMAX(I) - THMIN(I)
  IF (ABS(TEMP - BL(I) * GT. 1.) THEN
    THMAX(I) = BL(I)
    THMIN(I) = BL(I) - TEMP
  ELSE ENDIF
  A(I) = BL(I) - TEMP
  THB(I) = THMAX(I) / 1000.
  THS(I) = THMIN(I) / 1000.
  RTH(I) = THMAX(I) - THMIN(I)
ELSE ENDIF
IF (JTYPE(3) .EQ. 'P') THEN
  THMIN(2) = THS(2) * 1000. / RATIO
  THMAX(2) = THB(2) * 1000. / RATIO
  TEMP = THMAX(2) - THMIN(2)
  IF (ABS(TEMP - BL(2) * GT. 1.) THEN
    THMAX(2) = BL(2)
    THMIN(2) = BL(2) - TEMP
  ELSE ENDIF
  A(2) = BL(2) - TEMP
  THB(2) = THMAX(2) / 1000.
  THS(2) = THMIN(2) / 1000.
  RTH(2) = THMAX(2) - THMIN(2)
ELSE ENDIF
IF (JTYPE(4) .EQ. 'P') THEN
  THMIN(3) = THS(3) * 1000. / RATIO
  THMAX(3) = THB(3) * 1000. / RATIO
  TEMP = THMAX(3) - THMIN(3)
  IF (ABS(TEMP - BL(3) * GT. 1.) THEN
    THMAX(3) = BL(3)
    THMIN(3) = BL(3) - TEMP
  ELSE ENDIF
  A(3) = BL(3) - TEMP
  THB(3) = THMAX(3) / 1000.
  THS(3) = THMIN(3) / 1000.
  RTH(3) = THMAX(3) - THMIN(3)
ELSE
LISPAR FOR

This subroutine lists the input parameters to screen.

SUBROUTINE LISPAR
LOGICAL IV, RF, COMP, POTA, INACT
CHARACTER*30 S
CHARACTER*20 MAKER
CHARACTER*10 MODEL
CHARACTER JTYPE COMMON /LENGTH/RATIO, A(4), CL(4), PI, RTH(4), TH(4), TWIST(4)
COMMON /ANGLE/SWEEP, THMIN(4), THMAX(4), THS(4), THB(4)
COMMON /ITG/LAST, MM, LOOP(60), NSE, IANG, LL, IP, IDIR, ICOUNT(30), JJ
COMMON /PAR/ALPHA(6), THETA(6), OFFSET(6), LENGTH(6)
COMMON /CHP/MODEL, MAKER, JTYPE(6)
COMMON /LOG/RF, COMP, WOA(60), INACT(6)
CALL KVPRT(0., 1.0, 0.76171875)
CALL KWIN(0., 1.0, 0.76171875)
CALL KTSIZ(.012)
CALL KMVAF(0., 300.)
CALL KDPWAC(304., 360.)
DO 100 I = 1, 12
  CALL KMVAF(1., 30., *24., *1)
  CALL KDPWAC(304., 30., *24., *1)
100 CONTINUE
CALL KMVAF(10., 270.)
CALL KDPWAC(83., 314.)
CALL KMVAF(83., 150.)
CALL KDPWAC(83., 38.)
CALL KMVAF(170., 270.)
CALL KDPWAC(170., 174.)
CALL KMVAF(170., 150.)
CALL KDPWAC(170., 30.)
DNI = BL111/1000.
DII2 = BL12/1000.
DIII = BL31/1000.
IF (RF) THEN
  CALL KMVAF(2., 480.)
  WRITE (UNIT = S, FMT = 12) IANG
  FORMAT (* CURRENT ORIENTATION: *, I4, * DEG *)
  CALL KTEXT(S, B)
ELSE
ENDIF
CALL KMOVAB(0., 360.)
CALL KTEXT(‘ MAKE>, *0)
CALL KMOVAB(MAKER, 0.)
CALL KMOVAB(0., 330.)
CALL KTEXT(‘ MODEL: *, 0)
CALL KTEXT(MODEL, 0.)
CALL KMOVAB(25., 302.)
CALL KTEXT(‘ MANIPULATOR PARAMETERS’, 0)
CALL KMOVAB(20., 270.)
CALL KTEXT(‘ EFFECTIVE LENGTH / ANGLE’, 0)
CALL KMOVAB(80., 254.)
CALL KTEXT(‘ NORMALIZED EFF. ANGLE’, 0)
CALL KMOVAB(5., 238.)
WRITE(UNIT=S, FMT=102)DN1, ALPHA(2)
102  FORMAT( ‘LINK 2*: G6, F6.4, 2X, F7.3)
CALL KTEXT(S. 0)
CALL KMOVAB(5., 205.)
WRITE(UNIT=S, FMT=103)DN2, ALPHA(3)
103  FORMAT( ‘LINK 3*: G6, F6.4, 2X, F7.3)
CALL KTEXT(S. 0)
CALL KMOVAB(5., 182.)
WRITE(UNIT=S, FMT=104)DN3, ALPHA(4)
104  FORMAT( ‘LINK 4*: G6, F6.4, 2X, F7.3)
CALL KTEXT(S. 0)
CALL KMOVAB(10., 158.)
CALL KTEXT(‘ JOINTS DISPLACEMENT’, 0)
CALL KMOVAB(10., 134.)
CALL KTEXT(‘ # - TYPE FROM TO’, 0)
CALL KMOVAB(20., 110.)
WRITE(UNIT=S, FMT=107) JTYP1(1), -SWEEP/2, SWEEP/2
107  FORMAT( ‘ 1 -*, AI, 4X, F6.1, 5X, F6.1)
CALL KTEXT(S. 0)
CALL KMOVAB(20., 86.)
WRITE(UNIT=S, FMT=108) JTYP1(2), THS(1), THB(1)
108  FORMAT( ‘ 2 -*, AI, 4X, F6.1, 5X, F6.1)
CALL KTEXT(S. 0)
CALL KMOVAB(20., 62.)
WRITE(UNIT=S, FMT=109) JTYP1(3), THS(2), THB(2)
109  FORMAT( ‘ 3 -*, AI, 4X, F6.1, 5X, F6.1)
CALL KTEXT(S. 0)
CALL KMOVAB(20., 38.)
WRITE(UNIT=S, FMT=111) JTYP1(4), THS(3), THB(3)
111  FORMAT( ‘ 4 -*, AI, 4X, F6.1, 5X, F6.1)
CALL KTEXT(S. 0)
CALL KUPDAT
RETURN
END

C  -----------------------------------------------------
C
C SUBROUTINE LISRLT
C
C This subroutine lists the workspace properties to screen
C
C     SUBROUTINE LISRLT
C     LOGICAL RF, COM, ROTA, INACT
C     CHARACTER*70 S
C     CHARACTER*20 MAKER
C     CHARACTER*10 MODEL
C     CHARACTER*10 JTYPE
C     COMMON /LENGTH/RATIO, A(4), EL(4), PI, RTH(4), TH(4), TWIST(4)
COMMON /ANGLE/SWEET,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /ITG/LAST,MM,LOOPI60),NSE,IANG,LL,IP,DIR,ICOUNT(30),JJ
COMMON /VOLUME/TVOL1,TAREA,TRCX,TVX,TAV,TAY,TAZ
COMMON /MOMENT/TVMIX,TVMIX,TVMIX,TVMIX,RKX,RKY,RKZ
COMMON /PAR/ALPHA(6),THETA(6),OFFSET(6),HLENGT(6)
COMMON /CHP/MODEL,MAKER,JTYPE(6)
COMMON /LOG/IV.RF,COMP,INACT(6)
CALL KMOVAB(0..54.)
DO 100 I=1,6
CALL KMOVAB(1..30.+24.*1)
CALL KMOVAB(1024..30.+24.*1)
CONTINUE
CALL KMOVAB(70.,150.)
CALL KDRWAB(70.,30.)
CALL KMOVAB(100..150.)
CALL KDRWAB(100..30.)
CALL KMOVAB(210.,174.)
CALL KDRWAB(210.,30.)
CALL KMOVAB(280.,150.)
CALL KDRWAB(280.,30.)
CALL KMOVAB(345.,174.)
CALL KDRWAB(345.,30.)
CALL KMOVAB(412.,150.)
CALL KDRWAB(412.,30.)
CALL KMOVAB(474.,174.)
CALL KDRWAB(474.,30.)
CALL KMOVAB(599.,150.)
CALL KDRWAB(599.,30.)
CALL KMOVAB(724.,150.)
CALL KDRWAB(724.,30.)
CALL KMOVAB(824.,150.)
CALL KDRWAB(824.,30.)
CALL KMOVAB(924.,150.)
CALL KDRWAB(924.,30.)
CALL OUTPUT(5)
CALL KTXY57.(.012)
CALL KMOVAC(640.,150.)
CALL KTEXT('GAUSSIAN DIVERGENCE THEOREM','.0)
CALL KMOVAB(479.,134.)
WRITE(UNIT=S,FMT=114)
114 FORMAT('TOTAL VOLUME',5X,'ABOUT AXIS',8X,'X',12X,'Y',12X,'Z')
1 CALL KTEXT(S,0)
CALL KMOVAC(480.,110.)
WRITE(UNIT=S,FMT=115) TVOL1,TVMIX,TVMIX,TVMIX
115 FORMAT(E12.5,4X,'MOMENT OF IN.',31X,E12.5))
CALL KTEXT(S,0)
CALL KMOVAC(479.,86.)
WRITE(UNIT=S,FMT=116) RKY,RKY,RKZ
116 FORMAT('CROSS-SEC. AREA RADIUS OF GY.'3(1X,E12.5))
CALL KTEXT(S,0)
117 CALL KMOVAC(480.,38.)
WRITE(UNIT=S,FMT=117) TAREA,TAY,TAZ
117 FORMAT(E12.5,4X,'VOL. CENTROID',2X,E12.5,4X,'0,0',6X,E12.5)
CALL KTEXT(S,0)
CALL KMOVAC(480.,38.)
WRITE(UNIT=S,FMT=118) TRCX,TAZ
118 FORMAT('SCALE',X,F7.2,2X,'AREA CENTROID',2X,E12.5,4X,'---')
1 CALL KTEXT(S,0)
SUBROUTINE OUTPUT(M)

EXTERNAL A T L O M
LOGICAL IW.RF,COMP,ROTA,INACT
CHARACTER*8 S
CHARACTER*20 MAKER
CHARACTER*10 MODEL
CHARACTER JTYPE

DIMENSION CONST(3),COMMON /LENGTH/RATIO.A(4),CL(4),PI.RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /ITG/LAST,LOOP(60),NSE,IAHN,LL,IP,DIR,ICOUNT(30),JJ
COMMON /PAp/ALPHAtG),THETA(6),OFFSET(6),HLNGTH(6)
COMMON /CHR/MODEL,MAKER,JTYPE(6)
COMMON /LOG/IW.RF,COMP,ROTA,INACT

DO I = 1,3
  IF(JTYPE(I)<>P) THEN
    CONST(I)=ALPHAtG(I)*180./P
  ENDIF
  CONTINUE
  IF(M.EQ.5) THEN
    CALL KMOVAB(40.,156.)
    CALL KTEXT('HAND ORIENTATION',0)
    CALL KMOVAB(360.,158.)
    CALL KTEXT('DISPLACEMENT',0)
    IF(RF) THEN
      CALL KMOVAB(260.,158.)
      WRITE(UNIT=S,FMT=101) IANG
    ENDIF
  ELSE
    CALL KMOVAB(270.,158.)
    CALL KTEXT('NO',0)
  ENDIF
  CALL KMOVAB(158.,134.)
  WRITE(UNIT=S,FMT=103)
  FORMAT(*-TYPE LINK L.N., OFFSET JT. ANG TWIST TO FROM')
  CALL KTEXT(S,0)
  CALL KMOVAB(15.,110.)
  IF(JTYPE(1)<>P) THEN
    WRITE(UNIT=S,FMT=105)
    FORMAT(*-P 0.000 VAR 0.0 0.0 0.2F8.3)
    CALL KTEXT(S,0)
  ELSE
    WRITE(UNIT=S,FMT=105)
    FORMAT(*-R 0.000 0.00 VAR 90.0 0.2F8.3)
    CALL KTEXT(S,0)
  ENDIF

END
ENDIF
CALL KMOVAB(5.,86.)
IF(JTYPE(2).EQ.'P' .AND. JTYPE(2).EQ.'P') THEN
WRITE(UNIT=S,FMT=107) CONST(1),THS(1),THB(1)
FORMAT(2-P 0.000 VAR 'F6.1,5X ' 0.0 ' ,2F8.3)
CALL KTEXT(S,0)
ELSE IF(JTYPE(2).EQ.'P' .AND. JTYPE(2).EQ.'P') THEN
WRITE(UNIT=S,FMT=108) CONST(1)-90.,THS(1),THB(1)
FORMAT(2-P 0.000 VAR 'F6.1,2X,2F8.3)
CALL KTEXT(S,0)
ELSE IF(JTYPE(2).EQ.'R' .AND. JTYPE(2).EQ.'P') THEN
WRITE(UNIT=S,FMT=109) CONST(1),THS(1),THB(1)
FORMAT(2-R ',F8.3,' 0.00 VAR 'F6.1,2X,2F8.3)
CALL KTEXT(S,0)
ENDIF
ENDIF
CALL KMOVAB(5.,62.)
IF(JTYPE(2).EQ.'P' .AND. JTYPE(2).EQ.'P') THEN
WRITE(UNIT=S,FMT=110) CONST(1),THS(1),THB(1)
FORMAT(2-R ',F8.3,' 0.00 VAR 'F6.1,2X,2F8.3)
CALL KTEXT(S,0)
ELSE IF(JTYPE(2).EQ.'P' .AND. JTYPE(2).EQ.'R') THEN
WRITE(UNIT=S,FMT=111) CONST(2),THS(2),THB(2)
FORMAT(2-P 0.000 VAR 'F6.1,2X,2F8.3)
CALL KTEXT(S,0)
ENDIF
ELSE IF(JTYPE(3).EQ.'P' .AND. JTYPE(2).EQ.'R') THEN
WRITE(UNIT=S,FMT=112) CONST(2),THS(2),THB(2)
FORMAT(2-P 0.000 VAR 'F6.1,5X ' 0.0 ' ,2F8.3)
CALL KTEXT(S,0)
ELSE IF(JTYPE(3).EQ.'P' .AND. JTYPE(2).EQ.'P') THEN
WRITE(UNIT=S,FMT=113) CONST(2),THS(2),THB(2)
FORMAT(2-P 0.000 VAR 'F6.1,2X,2F8.3)
CALL KTEXT(S,0)
ENDIF
ELSE
WRITE(UNIT=S,FMT=114) CONST(2),THS(2),THB(2)
FORMAT(2-R ',F8.3,' 0.00 VAR 'F6.1,2X,2F8.3)
CALL KTEXT(S,0)
ENDIF
END IF
CALL KMOVAB(5.,30.)
IF(JTYPE(4).EQ.'P' .AND. JTYPE(3).EQ.'P') THEN
WRITE(UNIT=S,FMT=115) CONST(3),THS(3),THB(3)
FORMAT(2-P 0.000 VAR 'F6.1,2X,2F8.3)
CALL KTEXT(S,0)
ELSE IF(JTYPE(4).EQ.'P' .AND. JTYPE(3).EQ.'R') THEN
WRITE(UNIT=S,FMT=116) CONST(3),THS(3),THB(3)
FORMAT(2-P 0.000 VAR 'F6.1,5X ' 0.0 ' ,2F8.3)
CALL KTEXT(S,0)
ELSE IF(JTYPE(4).EQ.'P' .AND. JTYPE(3).EQ.'R') THEN
WRITE(UNIT=S,FMT=117) CONST(3),THS(3),THB(3)
FORMAT(2-R ',F8.3,' 0.00 VAR 'F6.1,5X,2F8.3)
CALL KTEXT(S,0)
ELSE
WRITE(UNIT=S,FMT=118) CONST(3),THS(3),THB(3)
FORMAT(2-R ',F8.3,' 0.00 VAR 'F6.1,5X,2F8.3)
CALL KTEXT(S,0)
ELSE
ENDIF

ELSE
WRITE(2,220) MAKEr
FORMAT(2,221) TITLE MAKEr ',.A20)
WRITE(2,222) MODEL
FORMAT(2,223) TITLE MODEL ',.A18)
WRITE(2,224) (JTYPE(I),I=1,4)
1 ' ( R - REVOLUTE, P - PRISMATIC')
WRITE(2,225)
201 FORMAT(/ ' HAND ORIENTATION 1')
IF(RF) THEN
WRITE(2,226) IANG
FORMAT(2,227) TITLE IANG ',.A20>
ELSE
WRITE(2,228)
FORMAT(2,229) TITLE ',.A20'
ENDIF
WRITE(2,230) RATIO
FORMAT(3X,'SCALE: ANGLE (DEGREE)/')
1 14X,'TOTAL LENGTH NORMALIZED TO UNIT, SCALE FACTOR = X',FB.3)
WRITE(2,231)
219 FORMAT(5X,7.2( )
WRITE(2,232)
204 FORMAT(' LINK OFFSET JOINT TWIST')
1 1
2 JTYPE LENGTH ANGLE ANGLE'
WRITE(2,233) JTYPE(I),EO.,'.P') THEN
WRITE(2,234) -Sweep/2,SWEEP/2
FORMAT(1,2F11.4)
ELSE
WRITE(2,235) -Sweep/2,SWEEP/2
FORMAT(1,2F11.4)
ENDIF
IF(JTYPE(2),EO.,'.P') AND. JTYPE(1),EQ.,'.P') THEN
WRITE(2,236) CONST(1),THS(1),THB(1)
FORMAT(2,2F11.4)
ELSE
WRITE(2,237) CONST(1),THS(1),THB(1)
FORMAT(2,2F11.4)
ENDIF
IF(JTYPE(2),EO.,'.P') AND. JTYPE(1),EQ.,'.P') THEN
WRITE(2,238) CONST(1),THS(1),THB(1)
FORMAT(2,2F11.4)
ELSE
WRITE(2,239) CONST(1),THS(1),THB(1)
FORMAT(2,2F11.4)
ENDIF
IF(JTYPE(3),EO.,'.P') AND. JTYPE(2),EQ.,'.P') THEN
WRITE(2,240) CONST(2),THS(2),THB(2)
FORMAT(2,2F11.4)
ELSE
WRITE(2,241) CONST(2),THS(2),THB(2)
FORMAT(2,2F11.4)
ENDIF
WRITE(2,242) 3 - P 0.0000 VAR 90.0000'.3F11.4
SUBROUTINE OVERLP

EXTERNAL ANOM

LOGICAL IWF, RF, COMP, ROTA, INACT, CWI, CWJ

COMMON /LENGTH/RATIO, A(4), BL(4), PI, TH(4), TWIST(4)

COMMON /ANGLE/SWEEP, THMIN(4), THMAX(4), THS(4), THB(4)

COMMON /XY/CX1(999), CX2(999), CX3(999), CY1(999), CY2(999),

COMMON /SUM/PC(60), PA(60), PHY(60), SHI0(60), SH11(60)

COMMON /ITG/LAST, NM, LOOP(60), NSE, IANG, LI, IDIR, ICOUNT(30), JJ

COMMON /LOG/IWF, RF, COMP, ROTA(60), INACT(6)

COMMON /PICK/INC(10), NI

 Title='MAIN/VOLUME/OVERLP'

CALL FRAME1(TITLE)

CALL SCREEN

CALL BOUNDT(0,1)

ICNT=0
DO 100 N=1,MM-1
  DEF=SHI(N)=SHI0(N)
  IF(SHIB(N).GE.PI .OR. SHI0(N).LT.PI) THEN
    SHIB(N)=ANOM(SHIB(N))
    SHI(N)=SHI0(N)*DEF.
  ELSE
    ENDIF
  CONTINUE
  I=0
  J=0
  I=I+1
  J=J+1
  CW0=.FALSE.,
  CW1=.FALSE.,
  K=0
  IF(ROTA(I).AND.ROTA(J)) THEN
    IF(ABS(PC(I)-PC(J)).LT.1.E-2) K=K+1
    IF(ABS(ROA(I)-ROA(J)).LT.1.E-2) K=K+1
    IF(ABS(PHY(I)-PHY(J)).LT.1.E-2) K=K+1
    IF(K.LE.3) THEN
      STOP=SHI(N)=SHI0(I)
      STOPJ=SHI(J)-SHI0(J)
      IF(STOPJ.LT.0.) CWIJ=.TRUE.
      IF(STOPJ.LT.0.) CWIJ=.TRUE.
      DEGIJ=ANOM(ANOM(SHIB(N))-ANOM(SHIB(J)))
      ENDIJJ=ANOM(SHIB(N))-ANOM(SHIB(J))
      AOIJ=SHIB(J)-SHIB(I)
      OLVJ=STOPJ-DEGIJ
      OLVI=STOPJ-DEGIJ
      CCW0=AOIJ
      IF(CW0) THEN
        IF(CCWI.GT.-.0001) CCWJ=CCWJ-2.*PI
        IF(CCWI.GT.-.0001) CCW0=CCWJ-2.*PI
        CCWJ=STOPJ+CCW0
        IF(CCWI.GE.0.) THEN
          CALL DPVLOP(J,1,0)
          CALL REDUCE(J)
          J=I-1
        ELSE
          ENDIF
        ELSE IF(BEIJ.GE.0. .AND. ENDIJ.LE.0. .AND. OLVJ.GE.STP) THEN
          CALL DRWLOP(J,1,0)
          CALL REDUCE(I)
          J=I-1
        ELSE IF(BEIJ.LE.0. .AND. ENDIJ.GE.0. .AND. OLVJ.LE.STP) THEN
          CALL DRWLOP(I,1,0)
          CALL REDUCE(I)
          I=I-1
        ELSE IF(BEIJ.GE.0. .AND. OLVI.LT.0.) THEN
          THEN
            SHI0(J)=ANOM(ANOM(SHIB(N)))
            IF((SHI0(J)-SHIB(J)).GT.2.*PI) SHI0(J)=SHIB(J)-2.*PI
            NUM=INTACS(STOPJ-SHIB(N)+SHIB(J))*30./PI
            DO 20 N=J+1,MM
            LOOP(N)=LOOP(N)-NUM
            20 CONTINUE
CONTINUE
LAST=LAST-NUM
DO 40 N=LOOP(J),LAST
TPX(N)=TPX(N+NUM)
TPY(N)=TPY(N+NUM)
40 CONTINUE
CALL DRWLOP(J,1,0)
CALL DRWLOP(J,1,0)
ELSE IF(BEGI.J.LE.0 .AND. OLVI.LT.0.) THEN
SHII(J)=SHI0(I)
IF(SHII(J).GT.SHI0(J)) SHII(J)=SHII(J)-2.*PI
NUM=JNT(ADS(STOPJ-SHII(J)+SHI0(J))*30./PI)
DO 50 N=J+1,NUM
CONTINUE
LAST=LAST-NUM
TPX(N)=TPX(N+NUM)
TPY(N)=TPY(N+NUM)
50 CONTINUE
CALL DRWLOP(1,1,0)
CALL DRWLOP(J,1,0)
ELSE ENDIF
ELSE IF(CCWO.LT.0.0001) CCWO=CCWO+2.*PI
CCWJ=STOPJ-CCWJ
IF(CWJ) THEN
IF(CCWO.LE.0.) THEN
CALL DRWLOP(J,1,0)
CALL REDUCE(J)
J+J=1
ELSE ENDIF
ELSE IF(BEGIJ.GE.0 .AND. ENDIJ.LE.0 .AND. OLVI.LE.STOPJ
AND.OLVI.GT.0.) THEN
CALL DRWLOP(J,1,0)
CALL REDUCE(J)
J+J=1
ELSE IF(BEGIJ.LE.0 .AND. ENDIJ.LE.0 .AND. OLVI.LE.STOPJ
AND.OLVI.GE.0 .AND. OLVI.LE.STOPJ
AND.OLVI.GE.0 .AND. OLVI.LE.STOPJ
CALL DRWLOP(J,1,0)
CALL REDUCE(J)
J+J=1
ELSE IF(BEGIJ.LE.0 .AND. OLVI.GT.0.) THEN
SHII(J)=SHI0(J)
NUM=JNT(STOPJ-SHII(J)+SHI0(J))*30./PI)
DO 35 N=J+1,NUM
CONTINUE
LAST=LAST-NU
DO 45 N=LOOP(J),LAST
TPX(N)=TPX(N+NUM)
TPY(N)=TPY(N+NUM)
45 CONTINUE
CALL DRWLOP(1,1,0)
CALL DRWLOP(J,1,0)
ELSE IF (BEGI\(\text{.GE.}\) .AND. OLVI.GT.\(\text{.O.}\))

\[
\begin{align*}
\text{I} & \quad \text{THEN} \\
\text{SHI1}(J) & \quad \text{SHI0}(I) \\
\text{IF} & \quad \text{SHI1}(J) \quad \text{LT.} \quad \text{SHI0}(J) \quad \text{SHI1}(J) \quad \text{SHI0}(J) \quad 2 \quad \text{.}\times \text{PI} \\
\text{NUM} & \quad \text{JNTR}((\text{STOP1} \quad \text{SHI1}(J) \quad \text{SHI0}(J) \quad 30 \quad \text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\text{.}\t
CALL DRWLOP(J,1,0)
call reduce(J)
J=J-1
else if (begij.ge.0. and endij.le.0.) then
  call drwlop(J,1,0)
call reduce(J)
J=J-1
else if (begij.le.0. and endij.ge.0.) then
  call drwlop(J,1,0)
call reduce(J)
J=J-1
else if (jceq.1. and begij.lt.0. and endij.lt.0.) then
  dis=sort((x1-ra(j))**2+(y1-pc(j))**2)
  num=joint(dis*10/phy(j))
  phy(j)=phy(j)-dis
  ra(j)=x1
  pc(j)=y1
  sh(j)=stop
  do 33 n=j+1,mm
    loop(n)=loop(n)-num
    continue
  last=last-num
  do 43 n=loop(j),last
    txp(n)=txp(n)-num
    tpy(n)=tpy(n)-num
    continue
  call drwlop(J,1,0)
call drwlop(J,1,0)
else if (jceq.1. and begij.gt.0. and endij.gt.0.) then
  dis=sort((x1-ra(j))**2+(y1-pc(j))**2)
  num=joint(dis*10/phy(j))
  phy(j)=phy(j)-dis
  do 53 n=j+1,mm
    loop(n)=loop(n)-num
    continue
  last=last-num
  do 63 n=loop(j+1),last
    txp(n)=txp(n)-num
    tpy(n)=tpy(n)-num
    continue
  call drwlop(J,1,0)
call drwlop(J,1,0)
else
  if (.not.cwj .and. endij.lt.0.) then
    call drwlop(J,1,0)
call reduce(J)
J=J-1
else if (begij.ge.0. and endij.le.0.) then
  call drwlop(J,1,0)
call reduce(J)
J=J-1
else if (begij.le.0. and endij.ge.0.) then
  call drwlop(J,1,0)
call reduce(J)
J=J-1
else if (jceq.1. and begij.gt.0. and endij.gt.0.) then
  dis=sort((x1-ra(j))**2+(y1-pc(j))**2)
  num=joint(dis*10/phy(j))
SUBROUTINE PICKBD

THIS SUBROUTINE lets THE USER TO IDENTIFY THE BOUNDARIES FOR
inside or on the boundary INTERACTIVELY BY MEANS OF A CURSOR.
THEN ELIMINATE THE overlapped VOLUMES ACCORDING TO THE
overlapped type. If the motion sequence occur ENTIRELY
WITHIN THE WORKSPACE THE PROGRAM CALL REDUCE TO REMOVE THIS
SEQUENCE.

IF A MOTION SEQUENCE INTERSECTS OTHER SEQUENCE AND THEY ARE
PARTIALLY ON THE BOUNDARY, THEN CALL INTECP TO SOLVE THE
INTERSECTING POINTS AND REMOVE THE INTERIOR PORTIONS.
If a motion sequence coincides with other sequence, then call
OVERLP to solve this problem.

SUBROUTINE PICKBD
LOGICAL IV, RF, COMP, ROTA, INACT
CHARACTER*27 S
CHARACTER*35 TITLE
CHARACTER ID
COMMON /ITG/LAST/MM, LOOP(60), NSE, lANG, ll, IP, IDIR, ICOUNT(30), JJ
COMMON /LOG/IW, RF, COMP, ROTA(60), INACT(6)
COMMON /PICK/INC(10), NI
TITLE='MAIN/VOLUME/PICKBD'
NI=0
N=1
NN=0
DO 11 I=1, 10
INC(I)=0
11 CONTINUE
30 CALL FRAME(TITLE)
CALL SCREEN
NN=1
CALL BOUND(T, 0, 1)
CALL KMOVAB(-2100..500.)
CALL KMOVAB(-1150..500.)
CALL KMOVAB(-2160..300.)
CALL KTEXT('THE ARC IS ON THE BOUNDARY', 1)
CALL KTEXT('ENTER -- Y', 1)
CALL KTEXT('PARTIALLY ON THE BOUNDARY', 1)
CALL KTEXT('ENTER -- P', 1)
CALL KTEXT('NOT ON THE BOUNDARY AT ALL', 1)
CALL KTEXT('ENTER -- N', 1)
CALL KTEXT('NEED A CLEAR NEW PAGE', 1)
CALL KTEXT('ENTER -- C', 1)
160 CONTINUE
CALL DROLOP(N, 2, 0)
200 CONTINUE
YM=450.-55.*NN
IF(YM.LT.-1000.) GOTO 30
CALL KMOVAB(-2100..YM)
WRITE(UNIT=5, FMT=12) N
12 FORMAT('IDENTIFY SEQUENCE', 'Y', 'P', 'N')
CALL KREADC(ID, INHELP, 'Y')
IF(ID.EQ. 'P' .OR. ID.EQ. 'p') THEN
NI=NI+1
INC(N)+N
N=NI+1
NN=NN+1
ELSE IF(ID.EQ. 'N' .OR. ID.EQ. 'n') THEN
CALL REDUCE(N)
NN=NN+1
ELSE IF(ID.EQ. 'C' .OR. ID.EQ. 'c') THEN
GOTO 30
ELSE IF(ID.EQ. 'Y' .OR. ID.EQ. 'y') THEN
N=NI+1
NN=NN+1
ELSE
YM=500.-50.*NN
CALL KMOVAB(-2050., VM)
CALL KTEXT(' ERROR INPUT. ENTER Y/P/N/C: ', B)
GOTO 20
ENDIF
IF(N.LT/MM) GOTO 100
CALL OVERL
IF(INC(I)=NE, B) THEN
CALL INTECP
ELSE
ENDIF
RETURN
END

SUBROUTINE PMSTOP
TO FIND A STOP POSITION FOR A SLIDING JOINT
IF NUM=1 the next stop position is the other end of joint limit
IF NUM=2, 3 the next stop position is decided by the locations
of inner joints in the generating plane.

SUBROUTINE PMSTOP(FIDX, NUM, ASTOP)
LOGICAL IF, RF, COMP, ROTA, INACT
COMMON /JTC/LAST, MM, LOOP(60), NSE, IANG, IP, IDIR, ICOUNT(30), JJ
COMMON /LOC/IF, RF, COMP, ROTA(60), INACT(6)
DIMENSION BETA(3), ANG(6), ROX(4), ROY(4), RO(6), AFA(10)

CALL CALCU1(POX, POY, RO, AFA)
IF(NUM.EQ.0) THEN
COPR=A(2)*SIN(FIDX-AFA)
AE=POY(I)-FOY(1)
CC=ROX(I)-FOX(2)
EE=AA*POY(I)+BB*POY(I)
ELSE
DD=ABS(CE/SQRT(EE))
ENDIF
X1=DD*C05(FIDX)
Y1=DD*31N(FIDX)
GOTO 10
IF(DY.EQ.0) AND (DX.EQ.0) GOTO 10
PHAI=ATA12(DY, DX)
DY=ROV(2)-V1
DX=ROK(2)-X1
GOTO 10
IF(DY.EQ.0) AND (DX.EQ.0) GOTO 10
PHAI=ATA12(DY, DX)
DELTA=ABS(PHA1-PHA2)
IF(DELTA.GT.5.1) THEN
X1=DD*C0S(FIDX-P1)
V1=DD*SIN(FIDX-P1)
IF (DY.EQ.0. .AND. DX.EQ.0.) GOTO 10

PHA1 = ATAN2(DY, DX)

ELSE

ENDIF

DEL1 = ABS(PHA1 - PHA2)

10

DELT1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)

ENDIF

ELSE

DEL1 = ABS(PHA1 - PHA2)
DY = ROY(2) - Y1
DX = ROX(2) - X1
IF(DY, EQ. 0. . AND. DX, EQ. 0.) GOTO 28
PHA2 = ATAN2(DY, DX)
DELT1 = ABS(PHA1 - PHA2)
ELSE
ENDIF
DELT2 = ACS(AFA(4) - PHA1) - PI
IF(DELT2, LT. 0. . AND. DELT2, LT. 0. .) N1 = 1
E1 = SORT((ROX(2) - X1)**2 + (ROY(2) - Y1)**2)
E2 = SORT((ROX(3) - X1)**2 + (ROY(3) - Y1)**2)
IF((E1, DL(3), LT. 1) N2 = 1
IF(E1, LT. 1) E1 = PTH(3)
E2 = PTH(3)
ELSE IF(N2, EQ. 1) THEN
E1 = RTH(3)
ELSE IF(N1, EQ. 1) THEN
E2 = RTH(3)
ELSE ENDIF
ELSE
E1 = RTH(3)
E2 = RTH(3)
ENDIF
ELSE
EE = AA*AE + EC*GB
IF(EE, EQ. 0.) THEN
DD = 0.
ELSE
DD = ACS((AA*ROX(1) + BB*ROY(1) + CC)/SORT(EE))
ENDIF
X1 = DD*COS(FDX) + ROX(1)
Y1 = DD*SIN(FDX) + ROY(1)
DY = ROY(2) - Y1
DX = ROX(2) - X1
IF(DY, EQ. 0. . AND. DX, EQ. 0.) GOTO 30
PHA3 = ATAN2(DY, DX)
DY = ROY(3) - Y1
DX = ROX(3) - X1
IF(DY, EQ. 0. . AND. DX, EQ. 0.) GOTO 30
PHA4 = ATAN2(DY, DX)
DELT3 = ACS(PHA1 - PHA2)
IF(DELT3, GT. 0. .) THEN
X1 = DD*COS(FDX) - ROX(1)
Y1 = DD*SIN(FDX) - ROY(1)
DY = ROY(2) - Y1
DX = ROX(2) - X1
IF(DY, EQ. 0. . AND. DX, EQ. 0.) GOTO 30
PHA3 = ATAN2(DY, DX)
DY = ROY(3) - Y1
DX = ROX(3) - X1
IF(DY, EQ. 0. . AND. DX, EQ. 0.) GOTO 30
PHA4 = ATAN2(DY, DX)
DELT4 = ABS(PHA1 - PHA2)
ELSE
ENDIF
DELT4 = ACS(AFA(5) - PHA3) - PI
IF(DELT4, LT. 0. . AND. DELT4, LT. 0. .) N3 = 1
E3 = SORT((ROX(2) - X1)**2 + (ROY(2) - Y1)**2)
SUBROUTINE PRIVOL

THIS SUBROUTINE CALCULATE VOLUME GENERATED BY ROBOT ARM whose FIRST JOINT IS PRISMATIC.

BYTE AL(2)
LOGICAL IW,RF,COMP,ROTA,INACT

COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /XY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999),CY3(999)
COMMON /SUM/PC(6P),FPHY(60),PHYR(60),PHY1(60),PHY2(60),PHY3(60)
COMMON /1TG/LAST,MIN,LOOP(60),NSE,ANGLL,IP,IDIR,ICOUNT(30),JJ
COMMON /LC/IV,PF,COMP,ROTA(60),INACT(6)
COMMON /VOLUME/TVOI,TAREA,TPCX,TVZ,TMZ,TVX
COMMON /MOIEN/TV1,TV1X,TV1Y,TV1Z,TV2,TV2X,TV2Y,TV2Z
DIMENSION VZ(60),TVX(60),TVY(60),TVZ(60),TV1X(60),TV1Y(60),TV1Z(60)

DO 100 I=1,MM-1
IF (IFCXM1, .NE., TFUE.) THEN
   X1=RA(I)
   V1=PC(I)
   AFA=SH1(I)
   PL=PHY(I)
   H=QRT((X1*X1)+(Y1*Y1))*SIN(PI/4)*PI
   APEA(I)=H*PL/2.66
   IF (AREA(I).LE.0.0) THEN
      SIGN=-1.
   ELSE
      SIGN=1.
   ENDIF
   IF (SAE(RAREA(I)),LT.1.E-6) THEN
      AREA(I)=0.0
      ASZ(I)=0.0
   ENDIF
100 CONTINUE

END
ELSE
  ASZ(1) = PL*H*(Y1*PL/2.*SIN(AFA))/3.E9
  AZ(1) = ASZ(1)/AREA(1)
  VZ(1) = AZ(1)
  VOL(1) = AREA(1)*SWEEP/1000.
  ASX(1) = PL*H*(X1*PL/2.*SIN(AFA))/3.E9
  RCX(1) = ASX(1)/AREA(1)
  VMIX(1) = H*PL*SWEEP*X1**2*X2*PL*COS(AFA)+PL**2*COS(AFA)**2/3.)
        /4.E15
  VMIV(1) = H*PL*SWEEP*(V1**2+V1*PL*SIN(AFA)+PL**2*SIN(AFA)**2/3.)
        /4.E15
  ZMI = H*PL*SWEEP**3/2.4E16
ENDIF
ELSE
  DCOS1 = COS(SHI1(1) - COS(SHI0(1))
  DSIN1 = SIN(SHI1(1) - SIN(SHI0(1))
  DSIN2 = SIN(2.*SHI1(1) - SIN(2.*SHI0(1))
  DCOP1 = COS(SHI1(1) - PHY1) - COS(SHI0(1) - PHY1)
  DCOP2 = COS2(2.*SHI1(1) - PHY1) - COS(2.*SHI0(1) - PHY1)
  DSIP1 = SHI1(1) - PHY1 - SIN(SHI0(1) - PHY1)
  DSIP2 = SHI1(1) - PHY1 - SIN(2.*SHI0(1) - PHY1)
  DSIP3 = SHI1(1) - PHY1 - SIN(3.*SHI0(1) - PHY1)
  DSPIH1 = -SHI1(1)
  AREA1 = 5.*RA1***(PC(1)*DSIP1+RA(1)*DSH1)/1.E6
IF(ABS(AREA1).LT.1.E-5) THEN
  AREA1 = 0.0
  ASZ(1) = 0.0
  AZ(1) = 0.0
  ZMI = 0.0
  VOL(1) = 0.0
  RCX(1) = 0.0
  VMIX(1) = 0.0
  VMIV(1) = 0.0
ELSE
  ASZ(1) = RA1***(PC(1)*PC(1)*SIN(PHY1)) + DSIP1**1.5*RA1***(PC(1)**2.5*DCOP2 -
        2
    RA1***(PC(1))**3/3.E9
  AZ(1) = ASZ(1)/AREA1
  ASX(1) = RA1***(PC(1))**2*PC(1)*SIN(PHY1)) + DSIP1**RA1***(PC(1)
    = DSIN1**1.5*PC(1)**RA1***(PC(1)**COS(PHY1))**DSH1**PC(1)
    2
    *FA1***(DCIP2/4.1/3.E9
    RCX(1) = ASX(1)/AREA1
    VOL(1) = AREA1*SWEEP/1000.
    VZ(1) = AZ(1)
    AA**PC(1)**3*SIN(PHY1)) + 2*DSIP1
    DD**PC(1)**3/2**SIN(PHY1)) + (DCIP2-4.)
    1
    SIN(PHY1))**DSH1
    1
    DD**PC(1)**3/2**DSIP3-3.*DSIP1*
    1
ZM1-RA1)**SWEEP**3*(PC1)*DSIP1*(RA1)*DSHIP1)**2.4E16
VHIV1*(RA1)**SWEEP**(AA+BB-CC-DD)/4.E15
AA*PC1)**3*COS(PHY1)**2*DSIP1
BB*RA1)**3.4**(2.**DSHIP1)**DSHIP2***(DSIP2)**4.*
CC*PC1)**2*(RA1)*COS(PHY1)**2.*DSHIP1
DD*PC1)**2*DSHIP1)**212.*(DSIP3)**3*DSIP1*
VHIX1)**SWEEP*(RA1)**(AA+BB+CC+DD)/4.E15

ENDIF

ENDIF

VMIZ1**SIGN**ABS(VMIZ1)
VMIV1**SIGN**ABS(VMIV1)
VMIX1**SIGN**ABS(VMIX1)
VMIZ2*VMIZ1**TVIIZ
VMIV2*VMIV1**TVIIV
VMIX2*VMIX1**TVIX
TAREA=TAREA+AREA1
TASZ+ASZ1
TASX+ASX1
AL1=AL1**2
AL2=AL2**2
PX=TPX**1000.
PY=TPY**1000.
CALL KMOVAB0.00.
CALL KLMTY?DOTTED'
CALL KDRWACIPX.PY)
CALL KMOVAB0.00.
CALL KTXSIZ0.12)
IFIIU^RWR22TVOLI1TRCX^TVZ.TAREA.TRCX.TAZ.TVMIX.RKX.TVMIY.RIZ

188 CONTINUE

TAZ=TASZ/TAREA
TVZ=TAZ
TRCX=TAZ/TAREA
TVX=TRCX
RXY=SORT(ABS(TVX)/TVOL1)
RZY=SORT(ABS(TVY)/TVOL1)
RIZ=SORT(ABS(TVZ)/TVOL1)
TVOL1=TVOL1/4.18879
CALL KTXSIZ.(12)
CALL KMOVAB0.00.
CALL KMOVAB0.00.
CALL KTXSIZ(11)
IFIIU^RWR22TVOLI1TRCX^TVZ.TAREA.TRCX.TAZ.TVMIX.RKX.TVMIY.RIZ
SUBROUTINE RCTFY

THIS SUBROUTINE DETECTS ANY ARC GENERATED BY A MOTION SEQUENCE INTERSECTING THE Z AXIS. THEN DIVIDE THIS SEQUENCE INTO TWO PARTS TO ELIMINATE SIGN CONFUSING FOR VOLUME INTEGRATION.

SUBROUTINE RCTFY

LOGICAL IW.RF, COMP.ROTA.INACT
COMMON /ANGLE/SWEEP.THMIN(4), THMAX(4), THS(4), THB(4)
COMMON /XY/CX1(999), CX2(999), CX3(999), CY1(999), CY2(999),
   CY3(999), CY4(999), TPX(999), TPY(999), FDUR(999), BD(300)
COMMON /SUM/PC(60), PA(60), PHY(60), SHI(60), SHI(60)
COMMON /TG/LAST.MINitude.Loop(60), NSE.ANG.LL, IP.IDIR.ICOUNT(30), JJ
COMMON /LOG/IW.RF, COMP.ROTA(60), INACT(6)

J=0
1 IF(J.LT.MH) THEN
   DO 20 I=LOOP(J), LOOP(J+1)-1
   IF(TPX(I+1).LT.-0.1) THEN
      IS=0
   ELSE IF(TPX(I+1).GT.0.1) THEN
      IS=1
   ELSE IS=2
   ENDIF
   IF(TPX(I).LT.-0.1) THEN
      IM=0
   ELSE IF(TPX(I).GT.0.1) THEN
      IM=1
   ELSE IF(I.EQ.Loop(1) .OR. IS.EQ.2) THEN
      GOTO 20
   ELSE CALL SHIFT(I,J)
      GO TO 20
   ENDIF
   IF(IM.NE.IS .AND. IS.NE.2) THEN
      CALL SHIFT(I,J)
      GOTO 20
   ELSE E N D I F
   CONTINUE
   GOTO 1
   ELSE E N D I F
   IF(IW) THEN
\begin{verbatim}
WRITE(2,45)  
        FORMAT(//,5X,'--------- DATA FOR VOLUME INTEGRAL ------',  
             /5X,'NUMBER PC RA PHY',  
             /5X,'SH10 SH11')  
DO 210 I=1,MM-1  
WRITE(2,211) I,PC(I),RA(I),PHY(I),SH10(I),SH11(I)  
211 FORMAT(7X,12.3X,6F11.3)  
CONTINUE  
ELSE  
ENDIF  
RETURN  
END

SUBROUTINE REDUCE(M)

THIS SUBROUTINE DELETES THE DATA POINTS of a motion sequence which is entirely NOT ON THE WORKSPACE BOUNDARY.

SUBROUTINE REDUCE(M)
LOGICAL IW.RF,COMP,ROTA,INACT
COMMON /ANGLE/SWEEP.THMINU1,THMAX1,THS1,THE1
COMMON /XY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999)
COMMON /SUM/PC(60),RA(60),PHY(60),SH10(60),SH11(60)
COMMON /LOG/IW.RF,COMP,INACT(60)
COMMON /PICK/INC(10)

DO 5 I=1,NI  
   IF(M.EQ.INC(I)) THEN  
      NI=NI-1  
      DO 6 J=1,NI  
         INC(J)=INC(J+1)  
      CONTINUE  
   ELSE  
      ENDIF  
6 CONTINUE
5 CONTINUE

HUM=LOOP(M+1)-LOOP(M)
LAST=LAST-NUM
DO 4B L=LOOP(M),LAST
   TPX(L)=TPX(L+NUM)
   TPY(L)=TPY(L+NUM)
   CONTINUE
4B DO 10 I=1,MM-1
   IF(ROTA(I+1)) THEN
      ROTA(I)=.TRUE.
   ELSE
      ROTA(I)=.FALSE.
   ENDIF
   PC(I)=PC(I+1)
   RA(I)=RA(I+1)
   PHY(I)=PHY(I+1)
   SH10(I)=SH10(I+1)
   SH11(I)=SH11(I+1)
   IF(I.EQ.MM-1) THEN
      LOOP(I+1)=LAST
   ELSE
      LOOP(I+1)=LOOP(I+2)-NUM
   ENDIF
END
\end{verbatim}
SUBROUTINE ROSTOP

FIND THE STOP POSITION FOR A REOLUTE JOINT IN A MOTION SEQUENCE
The stop position depends on the locations of inner joints in
the generating plane. The smallest rotating angle to an inner
joint which is reciprocal to the force is the stop angle.

ANGLE(I): ANGLE BETWEEN FORCE (FIDX) TO ANGLE I
BETA(I): ANGLE BETWEEN FIDX AND THE ORTHOGONAL DIRECTION OF
ANGLE I
GAMA*: THE EFFECTIVE ANGLE TO BE COMPARED

SUBROUTINE ROSTOP(FIDX, NUM, ASTOP)
LOGICAL IW, RF, COMP, ROTA, INACT
CHARACTER*20 MAKER
CHARACTER*10 MODEL
CHARACTER JTYPE
COMMON /LENGTH/RAT10, A(4), BL(4), PI, RTH(4), TH(4), TWIST(4)
COMMON /ANGLE/SWEEP, THMIN(4), THMAX(4), THS(4), THB(4)
COMMON /XY/CY1(999), CX2(999), CX3(999), CY2(999), CY3(999), TPX(999), TPY(999), FDIR(999), BD(300)
COMMON /ITG/LAST, MIP, LOOP(6), USE, JANG, LL, IP, IDIR, ICOUNT(30), J
COMMON /CHC/MODEL, MAKER, JTYPE(6)
COMMON /LOG/IW, RF, COMP, ROTA(6), INACT(6)
DIMENSION BETA(3), ANG(10), FIDX(4), ROY(4), RO(6), AFA(10)
CALL CALCUL(POX, ROY, RO, AFA)
DO 20 K=1,10
IF(ANG(K).LE.-2*PI) ANG(K)=3.*PI+ANG(K)
ELSE
ANG(K)=ANG(K)-FIDX
ENDIF
IF(IF(IDIR.EQ.1) THEN
ANG(K)=AFA(K)-FIDX
ELSE
ENDIF
IF(ANG(K).LE.-2*PI) ANG(K)=3.*PI+ANG(K)
IF(IF(JTYPE(2).EQ.*P*) THEN
20 CONTINUE
IF(IF(IDIR.EQ.1) THEN
BETA(1)=PI+ANG(K)-FIDX-P1/2.
BETA(2)=AFA(4)-FIDX-P1/2.
BETA(3)=AFA(5)-FIDX-P1/2.
ELSE
BETA(1)=FIDX-AFA(1)+PI/2.
BETA(2)=FIDX-AFA(4)+PI/2.
BETA(3)=FIDX-AFA(5)+PI/2.
ENDIF
DO 22 K=1,3
IF(ANG(K).LE.-2*PI) BETA(K)=3.*PI+BETA(K)
IF(BETA(K).LE.-PI) BETA(K)=2.*PI+BETA(K)
IF(BETA(K).LE.0.0) BETA(K)=PI+BETA(K)
IF(COCK(2).EQ.*P*) THEN
22 CONTINUE
IF(IF(IDIR.EQ.2) THEN
IF(JTYPE(2).EQ.*P*) THEN
CONTINUE
SUBROUTINE ROTAT

THIS SUBROUTINE executes the motion sequence
AND CALL SUBROUTINE 'SUM' TO STORE THE DATA.
THE PARAMETERS TRANSFERRED INTO ARE:
NUM: ROTATING JOINT NUMBER
SUBROUTINE ROTAT(NUM,ASTOP,FIDX)
EXTERNAL ANOM
LOGICAL I,RF,COMP,ROTA,INACT
CHARACTER*20 MAKER
CHARACTER*10 MODEL
CHARACTER JTYPE
COMMON /LEL/'GTH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /ITG/LAST,MH,LOOP(60),NSE,IANG,LL,IP,DIR,ICOUNT(30),J3
COMMON /CHP/MODEL,MAKER,JTYPE(6)
COMMON /LOG/IU,RF,COMP,ROTA(60),INACT(6)
DIMENSION ROX(4),ROY(4),RO(5),AFA(10)
IF(JTYPE(NUM+1).EQ. "P") THEN
  THN=TH(NUM)
  IF(DIR.EQ.1) THEN
    DO 150 I=1,11
      ANG=I*ASTOP/10.
      DIR=IDX
      ANUM=ANG+THN
      CALL C-LCU1(ROX,ROY,RO,AFA)
      CALL SUM1(ROX,ROY,AFA,FIDX,DIR,NUM,ASTOP)
      IF(LAST.GE.999) RETURN
      IF(MM,EQ.60) RETURN
      CONTINUE
  150 CONTINUE
  ELSE
    DO 160 I=1,11
      ANG=I*ASTOP/10.
      DIR=IDX
      ANUM=THN-ANG
      CALL C-LCU1(ROX,ROY,RO,AFA)
      CALL SUM1(ROX,ROY,AFA,FIDX,DIR,NUM,ASTOP)
      IF(LAST.GE.999) RETURN
      IF(MM,EQ.60) RETURN
      CONTINUE
  160 CONTINUE
  ELSE
    L=3*INT(ASTOP*30./PI)
    THN=TH(NUM)
    IF(RF) AFA(5)=IANG*PI/180.
    IF(DIR.EQ.1) THEN
      DO 180 I=1,L
        ANG=K*6*PI/180.
        CALL C-LCU1(ROX,ROY,RO,AFA)
        CALL SUM1(ROX,ROY,RO,AFA,FIDX,DIR,NUM,ASTOP)
        IF(LAST.GE.999) RETURN
        IF(MM,EQ.60) RETURN
        CONTINUE
  180 CONTINUE
  ELSE
    L=3*INT(ASTOP*30./PI)
    THN=TH(NUM)
    IF(RF) AFA(5)=IANG*PI/180.
    IF(DIR.EQ.1) THEN
      DO 180 I=1,L
        ANG=K*6*PI/180.

TH(NUM) = ANOM(THN + ANG)
DIR = FIDX + ANG
IF(L.EQ.0) TH(NUM) = THN + ASTOP
IF(RF) THEN
CALL CALCUL2(ROX, ROY, RO, AFA)
IF(THMAX(3).GT.0) THEN
IF(TH(3).GT.THMAX(3)) OR TH(3).LT.THMIN(3)) IDIR = 3
ELSE
IF(TH(3).GT.THMAX(3)) AND TH(3).LT.THMIN(3)) IDIR = 3
ENDIF
K = INT(IDIR/3.)
IF(JJ.NE.KK) THEN
IP = IP + 1
IF(IP.EQ.30) RETURN
IF(IDIR.EQ.3) THEN
ICOUNT(IP) = LAST - 1
ELSE
ICOUNT(IP) = LAST
ENDIF
ELSE
ENDIF
ELSE
CALL CALCUL1(ROX, ROY, RO, AFA)
ENDIF
CALL SUMI(ROX, ROY, RO, AFA, FIDX, DIR, NUM, ASTOP)
IF(LAST.EQ.999) RETURN
IF(MM.EQ.60) RETURN
CONTINUE
TH(NUM) = ANOM(THN + ASTOP)
CALL CALCUL1(ROX, ROY, RO, AFA)
FIDX = FIDX + ASTOP
IF(FIDX.GE.PI) FIDX = FIDX - 2.*PI
DIR = FIDX
CALL SUMI(ROX, ROY, RO, AFA, FIDX, DIR, NUM, ASTOP)
ELSE IF(IDIR.EQ.2) THEN
DO 200 I = 1, L + 2
II = I - 1
ANG = K*6.*PI/180,
TH(NUM) = THN - ANG
DIR = FIDX + ANG
IF(L.EQ.0) TH(NUM) = THN - ASTOP
TH(NUM) = ANOM(THNUM)
IF(RF) THEN
CALL CALCUL2(ROX, ROY, RO, AFA)
IF(TH(X1(3)).GT.0) THEN
IF(TH(3).GT.THMAX(3) OR TH(3).LT.THMIN(3)) IDIR = 4
ELSE
IF(TH(3).GT.THMAX(3) AND TH(3).LT.THMIN(3)) IDIR = 4
ENDIF
K = INT(IDIR/3.)
IF(JJ.NE.KK) THEN
IP = IP + 1
IF(IP.EQ.30) RETURN
IF(IDIR.EQ.4) THEN
ICOUNT(IP) = LAST - 1
ELSE
ICOUNT(IP) = LAST
ENDIF
200 CONTINUE
ENDIF
ELSE
   ELSE
    CALL CALCUI(ROX,ROY,RO,AFA)
   ENDIF
    CALL SUMII(ROX,ROY,RO,AFA,FIDX,DIR,NUM,ASTOP)
   IF(LAST.EQ.999) RETURN
   IF(MM.EQ.60) RETURN
    CONTINUE
   TH(NEW) = THM(THM-ASTOP)
    CALL CALCUI(ROX,ROY,RO,AFA)
    IF(FIDX.LT.-PI) FIDX = 2.*PI-FIDX
    DIR = FIDX
    CALL SUMII(ROX,ROY,PO,AFA,FIDX,DIR,NUM,ASTOP)
   ELSE
   ENDIF
   RETURN
END

C:******************************
C SCREEN.FOR
C******************************
C
C THIS ROUTINE IS FOR DRAWING THE X-Y AXIS AND
C PUTTING SCALE ON THOSE TWO AXES.
C
C SUBROUTINE SCREEN,
BYTE AL(2)
CHARACTER*3 SC(1D)
DATA SC/'0.1','0.2','0.3','0.4','0.5','0.6','0.7','0.8','0.9','1.0'
IFLAG=0
CALL KVFPT(0,0,0.0292,0.7324)
CALL KWINDOW(-2100,-1100,-1150,1100)
CALL KMOVAC(-1150,1150,1100)
GO TO 12C
ENTRY SCALE(AL)
IFLAG=2
CALL KWINDOW(-1150,1100,-1150,1100)
CALL KMOVAC(-1150,1150,1100)
XLMN=1150.
XLMN=1050.
XRANGE=1050.
YLMN=1050.
YRANGE=1050.
YLMN=1150.
CALL KTXSIZ(0,12)
CALL KMOVAC(XLMN,YLMN)
CALL KDRWAC(XLMN,YLMN)
CALL KMOVAC(XRANGE,YLMN)
CALL KDRWAC(XRANGE,YLMN)
CALL KMOVAC(XRANGE,YRANGE)
CALL KDRWAC(XRANGE,YRANGE)
CALL KMOVAC(0,YRANGE)
CALL KDRWAC(0,YRANGE)
CALL KNTYP('DOTTED')
CALL KDRWAD(0.,YUEDGE)
CALL KMOVAD(XLEDGE,0.)
CALL KDRWAD(XREDEGE,0.)
NY1=ABS(YUEDGE/100.)
NY2=ABS(YLEDGE/100.)
NYT=NY1+NY2+1
YSTAR=100.*NY1
CALL KLNTYP('SOLID')
DO 100 I=1,NYT
SI=1
DIS=100.*(SI-1.)
CALL KMOVAD(XLEDGE,YSTAR-DIS))
CALL KDRWAD(XLEDGE+30.,(YSTAR-DIS))
100 CONTINUE
NX1=ABS(XLEDGE/100.)
NX2=ABS(XREDEGE/100.)
NXT=NX1+NX2+1
XSTAR=100.*NX1
DO 101 I=1,NXT
SI=1
DIS=100.*(SI-1)
CALL KMOVAD(XLEDGE,YLEDGE)
CALL KDRWAD(XSTAR+DIS,YLEDGE)
101 CONTINUE
NY1=ABS(YUEDGE/50.)
DO 103 I=1,NY1,2
SI=1
DIS=50.*SI
CALL KMOVAD(XLEDGE,DIS)
CALL KDRWAD(XLEDGE+15.,DIS)
103 CONTINUE
NY2=ABS(YLEDGE/50.)
DO 104 I=1,NY2,2
SI=1
DIS=50.*SI
CALL KMOVAD(XLEDGE,-DIS)
CALL KDRWAD(XLEDGE+15.-DIS)
104 CONTINUE
NX1=ABS(XLEDGE/50.)
DO 105 I=1,NX1,2
SI=1
DIS=50.*SI
CALL KMOVAD(-DIS,YLEDGE)
CALL KDRWAD(-DIS,YLEDGE+15.)
105 CONTINUE
NX2=ABS(XLEDGE/50.)
DO 106 I=1,NX2,2
SI=1
DIS=50.*SI
CALL KMOVAD(DIS,YLEDGE)
CALL KDRWAD(DIS,YLEDGE+15.)
106 CONTINUE
IF (IFLAG .EQ. 2) GO TO 999
CALL KTXSIZ(.012)
CALL KMOVAD(KRIH+15.,-7.)
CALL KTEXT('U,D','D')
CALL KMOVAD(-25.,VPIH+40.)
CALL KTEXT('U,D','D')
DO 107 I=1,NY1
DIS=100.*I
CALL KMOVAB(XRIM+15..(DIS-7.))
CALL KTEXT(SC(I),0)
107 CONTINUE
DO 109 I=1,NY2
DIS=100.*I
CALL KMOVAB(XRIM+15..(-DIS-7.))
CALL KTEXT(SC(I),0)
109 CONTINUE
DO 111 I=1,NX1
DIS=100.*I
CALL KMOVAB(-DIS-25.,YRIM+40.)
CALL KTEXT(SC(I),0)
111 CONTINUE
DO 113 I=1,NX2
DIS=100.*I
CALL KMOVAB(DIS-25.,YRIM+40.)
CALL KTEXT(SC(I),0)
113 CONTINUE
CALL CROSS(850.,900.,42.,40.,'Z')
999 CONTINUE
CALL KUPDAT
RETURN
END

-----------------------------------------------
SUBROUTINE SELECT
THIS SUBROUTINE PLOTS THE main SELECTING MENU
-----------------------------------------------
SUBROUTINE SELECT
CALL KVVPRT(0.,1.,0.,1.,76171875)
CALL KWINDO(0.,1024.,0.,760.)
DO 10 I=1,6
CALL KMOVAB(0.,750.,37.*I)
CALL KDRPAB(304.,750.,-37.*I)
10 CONTINUE
CALL KMOVAB(0.,475.)
CALL KUPAB(304.,475.)
CALL KXSIZ(0.022)
CALL KMOVAB(10.,720.)
CALL KTEXT(' EXIT',0)
CALL KMOVAB(10.,683.)
CALL KTEXT(' PEDESIGN',0)
CALL KMOVAB(10.,646.)
CALL KTEXT(' DRAW LINK',0)
CALL KMOVAB(10.,609.)
CALL KTEXT(' VOLUME',0)
CALL KMOVAB(10.,572.)
CALL KTEXT(' DRAW BOUNDARY',0)
CALL KMOVAB(10.,535.)
CALL KTEXT(' MANIPULABILITY',0)
CALL KMOVAB(10.,500.)
CALL KTEXT(' HAND ORIENTATION',0)
CALL KUPDAT
RETURN
END
SUBROUTINE SEQUE2

THIS SUBROUTINE TO PROCESS THE ROTATION SEQUENCE
FOR HAND ORIENTATION SPECIFIED MANIPULATOR

SUBROUTINE SEQUE2(*,*)
EXTERNAL ANOM
LOGICAL RP(2,2),IR(2,2),ROTA,RF,COMP,INACT
CHARACTER*20 MAKER
CHARACTER*10 MODEL
CHARACTER*20 OTYPE

COMMON /LENGTH/RATIO,AL(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON /LIMIT/RIGHT(4),LEFT(4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),TH(4)
COMMON /SIN/PC(60),RA(60),PA(60),SHI(60),SH(60)
COMMON /ITC/LAST.MM.LOOP(60),NSE.IANG.LL.IP.IDIR.ICOUNT(38),J0
COMMON /CHR/MODEL,MAKER,OTYPE(6)
COMMON /LOG/IV,RF,COMP,ROTA(60),INACT(6)
DIMENSION AFA(10),ANG(10),ROX(4),ROY(4),RO(6)

IF(JTYPE(2).EQ.*R*) THEN
TH(1)=THMIN(1)
LEFT(1)=TH(1)-THMIN(1)
IF(LEFT(1).LT.0.0) LEFT(1)=LEFT(1)+2.*PI
ELSE
TH(1)=THMAX(1)
LEFT(1)=LEFT(1)-THMIN(1)
ENDIF

IF(JTYPE(3).EQ.*R*) THEN
TH(2)=THMIN(2)
LEFT(2)=TH(2)-THMIN(2)
IF(LEFT(2).LT.0.0) LEFT(2)=LEFT(2)+2.*PI
ELSE
TH(2)=THMAX(2)
LEFT(2)=LEFT(2)-THMIN(2)
ENDIF

IF(OTYPE(2).EQ.*R*) THEN
TH(3)>THMIN(3)
TLEFT(3)<TH(3)-THMIN(3)
IF(TLEFT(3).LT.0.0) TLEFT(3)=TLEFT(3)+2.*PI
ELSE
IF(TLEFT(3).GT.0.0) TLEFT(3)=TLEFT(3)-2.*PI
ENDIF

AFA(5)=IANG*PI/180.
CALL CALCU2(ROX,ROY,RO,AFA)

DO 19 I=1,2
RPI(1,3)=.FALSE.
IRI(1,3)=.FALSE.
K=0
19 CONTINUE
DO 18 J=1,2
18 CONTINUE

M=0
N=0
IL=0
IFLAG=0
RIGHT(1)=RTH(1)-TLEFT(1)
RIGHT(2)=RTH(2)-TLEFT(2)

CHECK THE MOBILITY OF EACH JOINT
XIDX1=COS(FIDX*0.005)
YIDX1=SIN(FIDX*0.005)
DEL2=ABS(FIDX-AFA(2))

CHECK MOBILITY OF JOINT 3
IF(JTYPE(3).EQ."P") THEN
  AK=ANOM(FIDX-AFA(4))
  IF(AK.GT.-1.56 .AND. AK.LT.1.56) THEN
    FCR2=-1
  ELSE
    FCR2=1
  ENDF
ENDIF
IF(FCR2.LT.0 .AND. RIGHT(2).GT.0.1) THEN
  RP(2,1)=.TRUE.
  M=2
ELSE IF(FCR2.GT.0 .AND. TLEFT(2).GT.0.1) THEN
  RP(2,2)=.TRUE.
ENDIF
IF(FCR2.GT.0 .AND. TLEFT(2).GT.0.1) THEN
  M=2
ENDIF
ELSE
  FCR2=XIDX1*(ROY(1)-ROY(1))-YIDX1*(ROX(1)-ROX(1))
  IF(FCR2.LT.0 .AND. RIGHT(2).GT.0.1) THEN
    RP(2,2)=.TRUE.
  ELSE IF(FCR2.GT.0 .AND. TLEFT(2).GT.0.1) THEN
    RP(2,2)=.TRUE.
  ELSE
    M=2
  ENDF
ENDIF
IF(IP(2,2).EQ.1) THEN
  IF(IP(2,1).EQ.1)
  CHECK IF JOINT 2 MOVABLE
  DELT6=AC(S(FIDX-AFA(1))
  DELT7=AC(S(DELT6-P1))
  DELT8=AC(S(FIDX-AFA(4)-FIDX-P1))
  DELT11=AC(S(DELT6-P1/2))
  DELT12=AC(S(DELT11-P1))
  IF(DELT6.LT.0.01 .OR. DELT7.LT.0.01) IFLAG=2
  IF(IP(2).EQ.2 .AND. DELT6.LT.0.05) IL=2
  IF(JTYPE(2).EQ."P") THEN
    AK=ANOM(FIDX-AFA(1))
    IF(AK.GT.-1.56 .AND. AK.LT.1.56) THEN
      FCR1=-1
    ELSE
      FCR1=1
    ENDF
    IF(IP(1,2).EQ.1) FCR1=FCR1
    IF(FCR1.GT.0 .AND. TLEFT(1).GT.0.01) RP(1,2)=.TRUE.
    IF(FCR1.LT.0 .AND. RIGHT(1).GT.0.01) RP(1,1)=.TRUE.
  ELSE
    N=2
ELSE
FCR1=XIDX1*ROV(1)-XIDX1*ROX(1)
IF(M.EQ.2 .AND. DEL2.LT.0.1) FCR1=-FCR1
IF(IFLAG.EQ.2 .AND. IL.EQ.2) FCR1=-FCR1
IF(FCR1.LT.0 .AND. RIGHT(1).GT.0.01) THEN
  IR(1,1)=.TRUE.
  N=1
ELSE
  ENDIF
IF(FCR1.GT.0 .AND. TLEFT(1).GT.0.01) THEN
  IR(1,2)=.TRUE.
  N=1
ELSE
  ENDIF
ENDIF

SORTING FOR PRIVILEGE TO MOTION
IF(M.EQ.2) THEN
  IF(IN.EQ.1 .AND. IFLAG.EQ.2) THEN
    IR(1,1)=.FALSE.
    IR(1,2)=.FALSE.
  ELSE
    ENDIF
ELSE
  ENDIF
ENDIF

NOW DOING THE BOUNDARY GENERATING SEQUENCE
IF(INV2.EQ.2 .OR. INV3.EQ.2) COMP=.TRUE.
IF(RP(1,1) .OR. IR(1,1)) THEN
  IDIR=1
  ASTOP=RTH(1)
  TLEFT(1)=TLEFT(1)-ASTOP
ELSE IF(RP(1,2) .OR. IR(1,2)) THEN
  NUM=1
  IDIR=2
  ASTOP=RTH(1)
  TLEFT(1)=TLEFT(1)-ASTOP
ELSE IF(RP(2,1) .OR. IR(2,1)) THEN
  NUM=2
  IDIR=1
  IF(RP(2,1)) CALL PMSTOP(FIDX,NUM,ASTOP)
  IF(IR(2,1)) CALL FOSTOP(FIDX,NUM,ASTOP)
  IF((ASTOP).GT.(TLEFT(2)).GT.RTH(2)) ASTOP=RTH(2)-TLEFT(2)
  TLEFT(2)=TLEFT(2)-ASTOP
ELSE IF(RP(2,2) .OR. IR(2,2)) THEN
  NUM=2
  IDIR=2
  IF(RP(2,2)) THEN
    CALL PMSTOP(FIDX,NUM,ASTOP)
  ELSE
    CALL ROSTOP(FIDX,NUM,ASTOP)
  ENDIF
ELSE IF((TLEFT(2)-ASTOP).LT.0.0) ASTOP=TLEFT(2)
  TLEFT(2)=TLEFT(2)-ASTOP
ELSE
  IDIR=1
  NUM=5
ENDIF
CALL ROSTOP(FIDX,NUM,ASTOP)
FDX=ASTOP+FDX
IF(FIDX.GE.PI) FIDX=FDX-2.*PI
GOTO 20
END IF
CALL ROTAT(NUM,ASTOP,FDX)
CALL CALCUL(ROX,ROY,RO,ANG)

NOW CHECK DONE OR NOT
20 IF(JTYPE(2).EQ.,*P*) THEN
  ANGIS=ABS(A(1))-THMAX(1))
ELSE
  ANGIS=ABS(TH(1))-THMIN(1))
ENDIF
IF(JTYPE(3).EQ.,*P*) THEN
  ANG2S=ABS(A(2))-THMAX(2))
ELSE
  ANG2S=ABS(TH(2))-THMIN(2))
ENDIF
ANG(S=AMAXI(ANGIS,ANG2S)
IF(ANG(S,LT.0.1) ) THEN
  IF(ROTAIN).EQ..FALSE.> I  SJ  I  t  (  N H  )  *ATAN2I (  P . O V (  4  )  -  P C I  N N  )  ) .  I R O X I  4)-RA(NN)  )  )  RETURN 1
ELSE
  IF(LAST.GE.999) RETURN 2
  IF(MH.EQ.60) RETURN 2
  GO TO 10
ENDIF
END

SUBROUTINE SE0UE4
THIS ROUTINE GENERATE MOTION SEQUENCES FOR a 4 links kinematic
chain in the generating plane.
THOSE JOINTS CAN BE EITHER REVOLUTE OR PRISMATIC JOINTS.

Except for the variable used in COMMON, which listed in the
main program "WORKSP", the variables used locally are:

IR(J,J): true for the revolute joint I can move in J direction
RP(J,J): true for the prismatic joint I can move in J direction
I: joint I
J: J - move in increase joint variable sense, 2 - decrease
ASTOP: motion range for a motion sequence (stop position)
RIGHT(I): range from current position to right limit of joint I
TLEFT(I): range from current position to left limit of joint I
IDIR: rotating direction which is the same as J
INV(I): condition to meet for moment of inner joint to be inv.

SUBROUTINE SE0UE4(*.*,)
EXTERNAL A'ON
LOGICAL RP(4,2),IR(4,2),ROTA,RF,INACT,IIW,COMP,INV(3)
CHARACTER*5 MAKER
CHARACTER*10 MODEL
CHARACTER JTYPE
COMMON /LENGTH/RAT10,AI4),CL<4),PT,RTH(4) . THt4),TWIST( 4 )
COMMON / L I I I I T / R  I G H T I  4  ) ,  T l  E FTI4)
COMMON /ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /SUM/PC(60),RA(60),PHV(60),SH10(60),SHI1(60)
COMMON /ITG/LAST,THMIN(4),THMAX(4),THS(4),THB(4)
COMMON /CHP/MODEL,MAKER,VTYPE
COMMON /LOS/IW.RF.SE.COMP.RF.ROTA(60),INACT(6)
DIMENSION ROX(4),ROY(4),RO(6),AS(4),TS(4),AFA(10)
CALL STARTS
DO 50 I=1,4
AS(I)=A(I)
TS(I)=TH(I)
50 CONTINUE
CALL CALC(U1(ROX,ROY,RO,AFA)
IF(JTYPE(2).EQ.'P') THEN
TLEFT(I)=A(I)-THMIN(I)
ELSE
TLEFT(I)=TH(I)-THMIN(I)
ENDIF
IF(JTYPE(3).EQ.'P') THEN
TLEFT(2)=A(2)-THMIN(2)
ELSE
TLEFT(2)=TH(2)-THMIN(2)
ENDIF
IF(JTYPE(4).EQ.'P') THEN
TLEFT(3)=A(3)-THMIN(3)
ELSE
TLEFT(3)=TH(3)-THMIN(3)
ENDIF
TLEFT(4)=TH(4)-THMIN(4)
IF(TLEFT(4).GT.2.-PI) TLEFT(4)=TLEFT(4)-2.*PI
ENDIF
IF(BL(4).EQ.0.) THEN
FIDX=AFA(I)
ELSE
FIDX=AFA(I)
ENDIF
IF(BL(4).EQ.0.) THEN
FIDX=AFA(I)
ELSE
FIDX=AFA(I)
ENDIF
DO 19 J=1,2
IF(JM(J).EQ.0.) THEN
FIDX=AFA(I)
ENDIF
19 CONTINUE
BEGINNING LOOP
DO 10 I=1,4
DO 19 J=1,2
IF(JM(J).EQ.0.) THEN
FIDX=AFA(I)
ENDIF
BEGF-FIDX
RIGHT(1)=TH(1)-TLEFT(1)
RIGHT(2)=TH(2)-TLEFT(2)
RIGHT(3)=TH(3)-TLEFT(3)
RIGHT(4)=TH(4)-TLEFT(4)
CONTINUE
CONTINUE
I=4
K=5
INV(1)=.FALSE.
INV(2)=.FALSE.
INV(3)=.FALSE.
COMP=.FALSE.

CHECK THE MOBILITY OF EACH JOINT

XIDX1=cos(FIDX+0.01)
YIDX1=sin(FIDX+0.01)

CHECK MOBILITY OF JOINT 5

IF (BL(3).EQ.0) GOTO 2
IF (JNUM.LE.5) GOTO 6
FCR4=XIDX1*(ROY(4)-ROY(3))-YIDX1*(ROX(4)-ROX(3))
IF (FCR4.LT.0 .AND. RIGHT(4).GT.0.01) THEN
I=2
IR(4,1)=.TRUE.
ELSE IF (FCR4.GT.0.0 .AND. TLEFT(4).GT.0.01) THEN
I=2
IR(4,2)=.TRUE.
ELSE
ENDIF
DEL3=ABS(Abs(FIDX-AFA(10))-P1)
IF (I.EQ.3 .AND. DEL3.LE.0.0001) INV(3)=.TRUE.

CHECK MOBILITY OF JOINT 4

IF (JTYPE(4).EQ.'P') THEN
AK=ANGM(FIDX-AFA(5))
IF (AK.GT.-1.58 .AND. AK.LT.1.56) THEN
FCR3=1
ELSE
FCR3=1
ENDIF
IF(Abs(Abs(AK)-P1/2.) .LT. 0.001) THEN
IF(INV3) FCR3=-FCR3
ELSE
ENDIF
IF (FCR3.LT.0 .AND. RIGHT(3).GT.0.01) THEN
PP(3,1)=.TRUE.
ELSE IF (FCR3.GT.0 .AND. TLEFT(3).GT.0.01) THEN
PP(3,2)=.TRUE.
ELSE
ENDIF
ELSE
FCR3=XIDX1*(ROY(1)-ROY(2))-YIDX1*(ROX(1)-ROX(2))
IF(Abs(Abs(FIDX-AFA(9))-P1) .LT. 0.001) THEN
IF (INV3) FCR3=-FCR3
ELSE
ENDIF
IF (FCR3.LT.0 .AND. RIGHT(3).GT.0.01) THEN
I=2
IR(3,1)=.TRUE.
ELSE IF (FCR3.GT.0 .AND. TLEFT(3).GT.0.01) THEN
I=2
IF(1.EQ.2 .AND. DEL2.LT.0.0001) INV(2)*.TRUE.
CHECK MOBILITY OF JOINT 3
IF(JTYPE(3).EQ.'P') THEN
  AK=ANOM(FIDX-AFA(4))
  IF(AK.LT.-1.56 .AND. AK.LT.1.56) THEN
    FCR2=-1
  ELSE
    FCR2=1
  ENDIF
  IF(INV(3)).GT.0.001 THEN
    FCR2=-FCR2
  IF(INV(2)).GT.0.001 THEN
    FCR2=-FCR2
  ELSE
    FCR2=-FCR2
  ENDIF
ELSE
  FCR2=(IDX1)*ROY(1)-ROY(1)*YIDY(1)*ROY(1)-ROX(1))
  IF(INV(2)).GT.0.001 THEN
    FCR2=-FCR2
  IF(INV(3)).GT.0.001 THEN
    FCR2=-FCR2
  ELSE
    FCR2=-FCR2
  ENDIF
ELSE
  FCR2=IFCR1
  IF(FCR2.LT.0. .AND. FCR2.GT.0.001) THEN
    FCR2=-FCR2
  ELSE
    FCR2=-FCR2
  ENDIF
ELSE
  FCR1=(IDX1)*ROY(1)-ROY(1)*YIDY(1)*ROY(1)-ROX(1))
  IF(INV(3)).GT.0.001 THEN
    FCR1=-FCR1
  IF(INV(2)).GT.0.001 THEN
    FCR1=-FCR1
  ELSE
    FCR1=-FCR1
  ENDIF
ELSE
  FCR1=-FCR1
ENDIF
CHECK IF JOINT 2 MOVABLE
IF(JTYPE(2).EQ.'P') THEN
  AK=ANOM(FIDX-AFA(1))
  IF(AK.LT.-1.56 .AND. AK.LT.1.56) THEN
    FCR1=-1
  ELSE
    FCR1=1
  ENDIF
  IF(AK.LT.-1.56 .AND. AK.LT.1.56) THEN
    FCR1=-FCR1
  IF(AK.LT.-1.56 .AND. AK.LT.1.56) THEN
    FCR1=-FCR1
  ELSE
    FCR1=-FCR1
  ENDIF
ELSE
  FCR1=-FCR1
ENDIF
ENDIF
IF (FCR1.LT.0 . AND. RIGHT(1).GT.0.01) THEN
RP(1,2)=.TRUE.
ELSE IF (FCR1.GT.0 . AND. TLEFT(1).GT.0.01) THEN
RP(1,1)=.TRUE.
ELSE
ENDIF
ELSE
FCR1=XIDX1*ROY(1)-YIDX1*ROX(1)
IF (ABS(ACS(FIDX-AFX(7))-P1).LT.0.001) THEN
IF (INV(1)) FCR1=FCR1
IF (INV(2)) FCR1=FCR1
ELSE
ENDIF
IF (FCR1.LT.0 . AND. RIGHT(1).GT.0.01) THEN
IR(1,1)=.TRUE.
ELSE IF (FCR1.GT.0 . AND. TLEFT(1).GT.0.01) THEN
IR(1,2)=.TRUE.
ELSE
ENDIF
ENDIF

C CHECK THE UNSTABLE SITUATION

DO 3 M=1,3
IF (INV(M)) K=K+1
CONTINUE
IF (K.GE.2) COMP=.TRUE.

C DOING THE BOUNDARY GENERATING SEQUENCE

IF (IR(1,1) .OR. RP(1,1)) THEN
IDIR=1
NUM=1
ASTOP=RTH(1)
TLEFT(1)=TLEFT(1)+ASTOP
RIGHT(1)=RIGHT(1)-TLEFT(1)
ELSE IF (IR(1,2) .OR. RP(1,2)) THEN
IDIR=2
NUM=1
ASTOP=RTH(1)
TLEFT(1)=TLEFT(1)+ASTOP
RIGHT(1)=RIGHT(1)-TLEFT(1)
ELSE IF (IR(2,1) .OR. RP(2,1)) THEN
IDIR=1
NUM=2
IF (IR(2,1)) CALL PHSTOP(FIDX,NUM,ASTOP)
IF (IR(2,2)) CALL ROSTOP(FIDX,NUM,ASTOP)
IF ((ASTOP+TLEFT(2)).GT.RTH(2)) ASTOP=RTH(2)-TLEFT(2)
TLEFT(2)=TLEFT(2)+ASTOP
RIGHT(2)=RIGHT(2)-TLEFT(2)
ELSE IF (IR(2,2) .OR. RP(2,2)) THEN
NUM=2
IDIR=2
IF (RP(2,2)) THEN
CALL PHSTOP(FIDX,NUM,ASTOP)
ELSE
CALL ROSTOP(FIDX,NUM,ASTOP)
ENDIF
IF((TLEFT(2) .EQ. ASTOP) .AND. (TLEFT(2) .LT. 0.0)) ASTOP = TLEFT(2)
TLEFT(2) = TLEFT(2) - ASTOP
RIGHT(2) = RTH(2) - TLEFT(2)
ELSE IF((IR(3,1) .OR. RP(3,1)) THEN
NUM = 3
IDIR = 1
IF(RP(3,1)) THEN
CALL PHSTOP(FIDX,NUM,ASTOP)
ELSE
CALL ROSTOP(FIDX,NUM,ASTOP)
ENDIF
IF((ASTOP + TLEFT(3)) .GT. RTH(3)) ASTOP = RTH(3) - TLEFT(3)
TLEFT(3) = TLEFT(3) - ASTOP
RIGHT(3) = RTH(3) - TLEFT(3)
ELSE IF((IR(3,2)) .OR. RP(3,2)) THEN
NUM = 3
IDIR = 2
IF(RP(3,2)) THEN
CALL PHSTOP(FIDX,NUM,ASTOP)
ELSE
CALL ROSTOP(FIDX,NUM,ASTOP)
ENDIF
IF((TLEFT(3) .EQ. ASTOP)) THEN
CALL ROSTOP(FIDX,NUM,ASTOP)
IF((ASTOP + TLEFT(4)) .GT. RTH(4)) ASTOP = RTH(4) - TLEFT(4)
TLEFT(4) = TLEFT(4) - ASTOP
RIGHT(4) = RTH(4) - TLEFT(4)
ELSE IF((IR(4,4)) THEN
NUM = 4
IDIR = 1
CALL ROSTOP(FIDX,NUM,ASTOP)
IF((TLEFT(4) .EQ. ASTOP) .AND. (TLEFT(4) .LT. 0.0)) ASTOP = TLEFT(4)
TLEFT(4) = TLEFT(4) - ASTOP
RIGHT(4) = RTH(4) - TLEFT(4)
ELSE
IDIR = 1
NUM = 5
CALL POSTOP(FIDX,NUM,ASTOP)
FIDX = ASTOP + FIDX
IF(FIDX .GE. PI) FIDX = FIDX - 2.*PI
GOTO 120
END IF
CALL ROTAT(NUM,ASTOP,FIDX)
CALL CALCU1IROX,ROY,RO,AF1A
END IF
NOW CHECK DONE OR NOT
120 IF(JTYPE(2) .EQ. *P*) THEN
ANG1 = ABS(A1) - AS(1))
ELSE
ANG1 = AS(TH(1) - TS(1))
ENDIF
IF(JTYPE(3) .EQ. *P*) THEN
ANG2 = ABS(A2) - AS(2))
ELSE

ANG2S = ABS(TH(2) - TS(2))
ENDIF

IF(JTYPE(4).EQ. 'P') THEN
  ANG3S = ABS(A(3) - AS(3))
ENDIF

ANG4S = ABS(TH(4) - TS(4))
ANGF = ABS(ECGF - FIDX)
ANGS = AMAX1(ANGS, ANG2S, ANG3S, ANG4S, ANGF)

IF(ANGS .LT. 0.1) THEN
  NN = NN - 1
  IF(ROTA(NN).EQ. FALSE.) THEN
    SH11(NN) = ATAN2((ROY(4) - PC(NN)), (ROX(4) - RA(NN)))
    RETURN
  ELSE
    RETURN 1
  ENDIF
ENDIF

RETURN
END

SUBROUTINE SHIFT(I, J)

THIS SUBROUTINE CALLED BY SUBROUTINE RCTFY TO INSERT A SEQUENCE AND SHIFT DATA AFTER THAT A POSITION BACKWARD.

LOGICAL IW, RF, COMP, ROTA, INACT
COMMON /ANGLE/SWEEP, THHIN(4), THMAX(4), THS(4), THB(4)
COMMON /X Y/ CX(999), CY(999), CX1(999), CX2(999), CY1(999), CY2(999),

1 CY3(999), CY4(999), TPX(999), TPY(999), TDIR(999), BD(300)
COMMON /SUM/ PC(60), PA(60), PHY(60), SH10(60), SHI(60)
COMMON /ITG/ LAST, MM, LOOP(60), NGE, IANG, LL, IP, IDIR, ICOUNT(30), JJ
COMMON /LOG/IW, RF, COMP, ROTA(4), INACT(6)

IF(POTA(I)) THEN
  XC = PC(I) * COS(PHY(I))
  YC = PC(I) * SIN(PHY(I))
  RZ = ACS(PA(I))**2 - XC**2
ELSE
  ENDIF
  DO 30 K = MM + 1, J + 1, -1
  IF(ROTA(K)) THEN
    ROTA(J) = .TRUE.
  ELSE
    ROTA(J) = .FALSE.
  ENDIF
  PC(J) = PC(N)
  RA(J) = RA(N)
  PHY(J) = PHY(N)
  SH11(J) = SH11(N)
  SHI(J) = SHI(N)
  LOOP(J) = LOOP(K) + 1
  CONTINUE
  LAST = LAST + 1
  DO 40 L = LAST, I + 1, -1
    TPX(L) = TPX(L - 1)
  30 CONTINUE
  RETURN
END
SUBROUTINE STARTSFIND A STARTING POSITION FOR THE PCBOT ARM TO GENERATE A MOTION SEQUENCE SO THAT IT WILL NOT GO INFINITE.
 THE STARTING POSITION IS THAT ALL JOINTS ARE RECIPROCAL.
TO THE APPLIED FORCE.

THE VARIABLES USED HERE ARE THE SAME AS THAT OF "SEQU4".

SUBROUTINE STARTS

EXTERNAL ANOM

LOGICAL RF(3,2),IR(3,2),ROTA,RF,INACT,IN.W,COMP

CHARACTER*20 MAKER

CHARACTER JTYPE

COMMON /LENGTH/RATIO,A(4),BL(4),P1,RTH(4),TH(4),TWIST(4)

COMMON /LIMIT/RIGHT(4),LEFT(4)

COMMON /ANGLE/SP,THMIN(I),THMAX(I),THS(I),THB(I)

COMMON /ITC/LAST.MM,LOOP(I),NSE.IANG.LL,IP.IDIR.ICOUNT(30),JJ

COMMON /CHR/MAKER,MODEL,JTYPE(I)

COMMON /LOG/RF,COMP,ROTA(60),INACT(6)

DIMENSION ROX(4),ROV(4),ROA(10)

IF(JTYPE(2).EQ."P") THEN
    TH(I) = TWIST(I)
    A(I) = THMIN(I)
    LEFT(I) = A(I) - THMIN(I)
ELSE
    TH(I) = THMIN(I)
    A(I) = CL(I)
    LEFT(I) = TH(I) - THMIN(I)
    IF(LEFT(I).LT.0.0) LEFT(I) = LEFT(I) + 2.*PI
    IF(LEFT(I).GT.2.*PI) LEFT(I) = LEFT(I) - 2.*PI
ENDIF

IF(JTYPE(3).EQ."P") THEN
    A(2) = THMIN(2)
    TH(2) = TWIST(2)
    LEFT(2) = A(2) - THMIN(2)
ELSE
    TH(2) = THMIN(2)
    A(2) = CL(2)
    LEFT(2) = TH(2) - THMIN(2)
    IF(LEFT(2).LT.0.0) LEFT(2) = LEFT(2) + 2.*PI
    IF(LEFT(2).GT.2.*PI) LEFT(2) = LEFT(2) - 2.*PI
ENDIF

IF(JTYPE(4).EQ."P") THEN
    A(3) = THMIN(3)
    TH(3) = TWIST(3)
    LEFT(3) = A(3) - THMIN(3)
ELSE
    TH(3) = THMIN(3)
    A(3) = CL(3)
    LEFT(3) = TH(3) - THMIN(3)
    IF(LEFT(3).LT.0.0) LEFT(3) = LEFT(3) + 2.*PI
    IF(LEFT(3).GT.2.*PI) LEFT(3) = LEFT(3) - 2.*PI
ENDIF

CALL CALCUL(ROX,ROY,ROA)

IF(CL(3).EQ."A") then
    IF(JTYPE(3).EQ."P") THEN
        FDAX = ANOM(AFA(4) + PI/2.)
ELSE
   FIDX=AFA(4)
ENDIF
else IF(CL(4).EQ.0.) THEN
   FIDX=AFA(5)
ELSE
   FIDX=AFA(10)
ENDIF
RIGHT(1)=RTH(1)-TLEFT(1)
RIGHT(2)=RTH(2)-TLEFT(2)
RIGHT(3)=RTH(3)-TLEFT(3)
RIGHT(4)=RTH(4)-TLEFT(4)

BEGINNING LOOP
   DO 18 J=1,3
   DO 19 J=1,2
   RP(I,J)=.FALSE.
   IR(I,J)=.FALSE.
   CONTINUE
   CONTINUE

CHECK RECIPROCITY OF EACH JOINT

XIDX1=COS(FIDX)
YIDX1=SIN(FIDX)
IF(BL(3).EQ.0.) GOTO 1
IF(JNUM.LE.6) GOTO 2

CHECK RECIPROCITY OF JOINT 4

IF(JTYPE(4).EQ. 'P') THEN
   AK=ANOM((FIDX-AFA(5))
   IF(AK.GT.-1.58 .AND. AK.LT.1.56) THEN
      FCP3=1
   ELSE
      FCP3=1
   ENDIF
   IF(FCP3.LT.0. .AND. RIGHT(3).GT.0.1) THEN
      RP(3,1)=.TRUE.
   ELSE IF(FCP3.GT.0. .AND. TLEFT(3).GT.0.1) THEN
      RP(3,2)=.TRUE.
   ELSE
      ENDIF
   ELSE
      FCP3=XIDX1*(ROV(4)-ROV(2))-YIDX1*(ROX(4)-ROX(2))
      IF(FCP3.LT.0. .AND. RIGHT(3).GT.0.01) THEN
         RP(3,1)=.TRUE.
      ELSE IF(FCP3.GT.0. .AND. TLEFT(3).GT.0.01) THEN
         RP(3,2)=.TRUE.
      ELSE
         ENDIF
   ELSE
      ENDIF

CHECK RECIPROCITY OF JOINT 3

IF(JTYPE(3).EQ. 'P') THEN
   AK=ANOM((FIDX-AFA(4))
   IF(AK.GT.-1.58 .AND. AK.LT.1.56) THEN
      FCR2=1

---

---
ELSE
FCR2=1
ENDIF
IF((FCR2.LT.0 . AND. RIGHT(2).GT.0.1) THEN
RP(2,1)=.TRUE.
ELSE IF((FCR2.GT.0 . AND. TLEFT(2).GT.0.1) THEN
RP(2,2)=.TRUE.
ELSE ENDIF
ELSE
FCR2=XIDX1*ROY(4)-ROY(1)-YIDX1*(ROX(4)-ROX(1))
ENDIF
IF((FCR2.LT.0 . AND. RIGHT(2).GT.0.1) THEN
IR(2,1)=.TRUE.
ELSE IF((FCR2.GT.0 . AND. TLEFT(2).GT.0.1) THEN
IR(2,2)=.TRUE.
ELSE ENDIF
ENDIF
CHECK RECIPROCITY OF JOINT 2
IF(JTYPE(2).EQ.'P') THEN
AK=A*(FXD-AFA(1))
ENDIF
IF((AK.GT.-1.58 . AND. AK.LT.1.56) THEN
FCRI=-1
ELSE
FCRI=1
ENDIF
IF((FCR1.LT.0 . AND. TLEFT(1).GT.0.1) THEN
RP(1,2)=.TRUE.
ELSE IF((FCR1.GT.0 . AND. RIGHT(1).GT.0.1) THEN
RP(1,1)=.TRUE.
ELSE ENDIF
ELSE
FCR1=XIDX1*ROY(4)-YIDX1*ROX(4)
ENDIF
IF((FCR1.LT.0 . AND. RIGHT(1).GT.0.1) THEN
IR(1,1)=.TRUE.
ELSE IF((FCR1.GT.0 . AND. TLEFT(1).GT.0.1) THEN
IR(1,2)=.TRUE.
ELSE ENDIF
ENDIF
NOW MOVE THE UU-RECIPROCAL JOINTS
IF(IR(1,1) . OR. RP(1,1)) THEN
TH(1)=TMAX(1)
THD=2*A*(FXD-RTH1)
ELSE
TH(1)=TMAX(1)
ENDIF
TLEFT(1)=TLEFT(1)*RTH(1)
TH(1)=TH(1)+TLEFT(1)
ELSE IF(IR(1,2) . OR. RP(1,2)) THEN
TH(1)=TMIN(1)
THD=2*A*(FXD-RTH1)
ELSE
A(1)=THMIN(1)
ENDIF
TLEFT(1)=TLEFT(1)-RTH(1)
RIGHT(1)=RTH(1)-TLEFT(1)
ELSE IF IFR(2,1) .OR. RP(2,1)) THEN
NUH=2
IDIR=1
IF(RP(2,1)) THEN
CALL FISTOP(FIDX,NUM,ASTOP)
A(2)=A(2)+ASTOP
ELSE
CALL ROSTOP(FIDX,NUM,ASTOP)
TH(2)=TH(2)+ASTOP
FIDX=ANOM(FIDX+ASTOP)
ENDIF
TLEFT(2)=TLEFT(2)+ASTOP
RIGHT(2)=RTH(2)-TLEFT(2)
ELSE IF IR(2,2).OR. RP(2,2)) THEN
NUH=2
IDIR=2
IF(RP(2,2)) THEN
CALL FISTOP(FIDX,NUM,ASTOP)
A(2)=A(2)+ASTOP
ELSE
CALL ROSTOP(FIDX,NUM,ASTOP)
TH(2)=TH(2)+ASTOP
FIDX=ANOM(FIDX+ASTOP)
ENDIF
TLEFT(2)=TLEFT(2)+ASTOP
RIGHT(2)=RTH(2)-TLEFT(2)
ELSE IF IR(3,1).OR. RP(3,1)) THEN
NUH=3
IDIR=1
IF(RP(3,1)) THEN
CALL FISTOP(FIDX,NUM,ASTOP)
A(3)=A(3)+ASTOP
ELSE
CALL ROSTOP(FIDX,NUM,ASTOP)
TH(3)=TH(3)+ASTOP
FIDX=ANOM(FIDX+ASTOP)
ENDIF
TLEFT(3)=TLEFT(3)+ASTOP
RIGHT(3)=RTH(3)-TLEFT(3)
ELSE IF IR(3,2).OR. RP(3,2)) THEN
NUH=3
IDIR=2
IF(RP(3,2)) THEN
CALL FISTOP(FIDX,NUM,ASTOP)
A(3)=A(3)+ASTOP
ELSE
CALL ROSTOP(FIDX,NUM,ASTOP)
TH(3)=TH(3)+ASTOP
FIDX=ANOM(FIDX+ASTOP)
ENDIF
TLEFT(3)=TLEFT(3)+ASTOP
RIGHT(3)=RTH(3)-TLEFT(3)
ELSE
RETURN
ENDIF
TH(1)=ANOM(TH(1))
TH(2)=ANOMITH(2))
TH(3)=ANOMITH(3))
TH(4)=ANOMITH(4))
CALL CALCUI(ROX,ROY,RO,AFA)
GO TO 100
END

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

SUBROUTINE SUM

THIS SUBROUTINE COLLECTS CALCULATED RESULTS AND PUT
THEM INTO GROUPS OF DATA AREAS FOR SUBSEQUENCE USE.

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

SUBROUTINE SUM(IROX,ROY,RO,AFA,FIDK,DIR,NUM,ASTOP)
EXTERNAL ANOM
LOGICAL IV.RF,COMP,ROTA,INACT,LIMIT
CHARACTER*10 MAKER
CHARACTER*10 MODEL
CHARACTER JTYPE
COMMON/LENGH/RATIO,A(4),BL(4),PI,RTH(4),TH(4),TWIST(4)
COMMON/L1:MIT/RIGHT(4),TLEFT(4)
COMMON/ANGLE/SWEEP,THMIN(4),THMAX(4),THS(4),THI(4)
COMMON/KY/CX1(999),CX2(999),CX3(999),CY1(999),CY2(999),
1 CY3(999),CY4(999),CY5(999),TPX(999),TPY(999),FDI(100),BD(100)
COMMON/SUM/PC(60),RA(60),FY(60),SH1(60),SH2(60)
COMMON/1 TO/LAST.MM.LOOP(60),N1N.MM,PP,J,JJ
1 :MAX,JNUM
COMMON/CHR/MODEL,MAKER,JTYPE(6)
COMMON/VARV1(999),VAR2(999),VAR3(999)
COMMON/IAT/NUM(60),NMS(60),NL(300)
DIMENSION AFA(10),POX(4),ROY(4),RO(6),LIMIT(4)
CX1(LAST)=POX(1)
CX2(LAST)=POX(2)
CX3(LAST)=POX(3)
CX4(LAST)=POX(4)
CY1(LAST)=ROY(1)
CY2(LAST)=ROY(2)
CY3(LAST)=ROY(3)
CY4(LAST)=ROY(4)
FDI(LAST)=ANOM(DIR)
IF(JTYPE(2),EO.'P') THEN
VAR1(LAST)=A1(1)
ELSE
VAR1(LAST)=TH(1)
ENDIF
IF(JTYPE(3),EO.'P') THEN
VAR2(LAST)=A2(2)
ELSE
VAR2(LAST)=TH(2)
ENDIF
VAR3(LAST)=TH(3)
IF(KI.MODI(LAST*4),5) THEN
VAR3(LAST)=TH(3)
ENDIF
IF(KI.MODI(LAST*4),5) THEN
VAR3(LAST)=TH(3)
ENDIF
IF(OX,EO.1) THEN
VAR3(LAST)=TH(3)
ENDIF
IF(ATOM) BD(L)=ATOM(BD(LLL)*P)
1 L=L+1
ELSE
ENDIF

TPX(LAST)=ROX(4)
TPY(LAST)=ROY(4)

IF (1.EQ.1) THEN
  LOOP(MM)=LAST
  NUM(MM)=NUM+1
  ROT(MM)=.TRUE.
  IF (RIGHT(1).EQ.0 .OR. TLEFT(1).EQ.0.) LIMIT(1)=.TRUE.
  IF (RIGHT(2).EQ.0 .OR. TLEFT(2).EQ.0.) LIMIT(2)=.TRUE.
  IF (RIGHT(3).EQ.0 .OR. TLEFT(3).EQ.0.) LIMIT(3)=.TRUE.
  IF (RIGHT(4).EQ.0 .OR. TLEFT(4).EQ.0.) LIMIT(4)=.TRUE.
  IF (RIGHT(3).EQ.0 .OR. BL(3).EQ.0.) LIMIT(3)=.FALSE.
  IF (RIGHT(4).EQ.0 .OR. BL(4).EQ.0.) LIMIT(4)=.FALSE.
  IF (LIMIT(1).AND. NUM.NE.1) THEN
    NUM(MM)=2
  ELSE
   ENDIF
  IF (LIMIT(2).AND. NUM.NE.2) THEN
    NUM(MM)=10*NUM(MM)+3
  ELSE
   ENDIF
  IF (LIMIT(3).AND. NUM.NE.3) THEN
    NUM(MM)=10*NUM(MM)+4
  ELSE
   ENDIF
  IF (LIMIT(4).AND. NUM.NE.4) THEN
    NUM(MM)=10*NUM(MM)+6
  ELSE
   ENDIF
  IF (JTYPE(NUM+1),EQ.,"P") THEN
    PC(MM)=ROY(4)
    POTA(MM)=.FALSE.
    RA(MM)=ROX(4)
    PHY(MM)=ASTOP
    IF (ROY(4).EQ.0 .AND. ROX(4).EQ.0.) THEN
      PHI(MM)=0.
    ELSE
      PHI(MM)=ATAN2(ROY(4),ROX(4))
    ENDIF
    ELSE IF (F .AND. JTYPE(NUM+1),EQ.,"R") THEN
      IF (NUM.EQ.1) THEN
        PC(MM)=A(3)
        RA(MM)=ROI(1)
        PHY(MM)=AFA(5)
        SHI(MM)=AFA(2)
        IF (IDIR.EQ.1 .OR. IDIR.EQ.3) THEN
          SHI(MM)=AFA(2)+ASTOP
        ELSE
          SHI(MM)=AFA(2)-ASTOP
        ENDIF
      ELSE
        XA=A(1)*COS(AFA(1))+A(3)*COS(AFA(5))
        YA=A(1)*SIN(AFA(1))+A(3)*SIN(AFA(5))
        PC(MM)=SORT(XA*XA+YA*YA)
        RA(MM)=A(2)
        IF (YA.EQ.0 .AND. XA.EQ.0.) THEN
          PHY(MM)=0.
        ELSE
          PHY(MM)=ATAN2(YA,XA)
        ENDIF
      ENDIF
SUBROUTINE VOLUME

THIS SUBROUTINE CALCULATE
1. AREA BOUNDED BY OUTER BOUNDARY
2. VOLUME BOUNDED BY THE BOUNDARY
3. CENTROID POSITION FOR AREA
4. CENTROID POSITION FOR VOLUME
5. MOMENT OF INERTIA OF VOLUME ABOUT Z-AXIS.

NOTICE: OVERLAPPED REGIONS CAN BE REMOVE BY ASSIGNING 'NET VOLUME'.

The variables used in this subroutine are:
- TVOL: total volume (TVOL)
- TAREA: total cross-sectional area in the generating plane
- TRCX: x location of total area centroid (radius of rotation)
- TAZ: z location of total area centroid
- TVX: x location of total volume centroid
- TVVIZ: total volume moment of inertia about y axis
- TVHIZ: total volume moment of inertia about z axis
- RIX: radius of gyration about x axis
- RIZ: radius of gyration about z axis
- VDX(I): volume dot x of sub-volume generated by sequence I
- VZ(I): centroid location along z-axis of sub-volume I
- VDZ(I): volume dot z of sub-volume generated by sequence I
- VZC(I): sub-volume centroid along z-axis of sub-volume I
- VDI(I): integrated sub-volume I
- VDIZ(I): sub-volume I with sign compiled with area integral
- VMIX(I): volume moment of inertia of sub-volume I about x axis
- VMIV(I): volume moment of inertia of sub-volume I about y axis
- VMIZ(I): volume moment of inertia of sub-volume I about z axis
- ASZ(I): area time z of sequence I
- AZ(I): area centroid at z direction of sequence I

SUBROUTINE VOLUME.
BYTE AL(12)
LOGICAL IV,RF,COMP,ROTA,INACT
CHARACTER*1 RF,COMP,ROTA,RF,MKAER
CHARACTER*25 TITLE
COMMON /LENG/RATIO,PI,TH,THST(4)
COMMON /ANGLE/SWEEP,THM(4),THM(4),THS(4),THB(4)
COMMON /PARAM/(999),CZ(999),CY(999),CV(999),CV(999)
COMMON /SUM/PC,(CP),PX(60),PY(60),SH(60),SH(60)
COMMON /LOG/RF,COMP,ROTA,IV,RF,COMP,INACT(9)
COMMON /VOLUME/TVAL,TAREA,TRCX,TZ,TAZ,TMX
COMMON /DIM/TVX,TVY,TVMIX,TVMIZ,RKX,RKZ
DIMENSION VDX(I),VZ(I),VMIX(I),VMIV(I),VMIZ(I),VOL(I),VOL(I),VOL(I),AREA(I),ASZ(I),AZ(I),VZ(I)
DOUBLE PRECISION XX,YY,ZZ
CALL KMODAL(0,-450.)
CALL KTEXT1(-1,NET VOLUME*.1)
CALL KMOVAB(0,-430.)
CALL KTEXT1(2,GIROSS VOLUME*.1)
CALL KMOVAB(0,-410.)
CALL KTEXT1(3,NET VOLUME, ALL SURFACES*.1)
CALL KREADCIL,HELP,0)
FORMAT(1)
TITLE='MAIN/VOLUME'
IF L=2 THEN CALCULATE GROSS VOLUME and plot all surfaces
ELSE calculate net volume and plot boundary. In addition:
IF L=0 PLOT CENTROID location
L=1 PLOT ALL GENERATED SURFACES

IF L.LT.2 THEN
CALL PICKBD
CALL FRAME(TITLE)
CALL SCREEN
CALL PINTD(1.0)
CALL KMOVAB(-2100.0,620.0)
CALL KDEWAB(-1150.0,620.0)
CALL KXSIZE(0.02)
CALL KM0VAB(-2050.0,980.0)
CALL KTEXT('OK 1'.0)
CALL KMOVAB(-2050.0,980.0)
CALL KTEXT('GO AHEAD 1'.0)
CALL KM0VAB(-2050.0,580.0)
CALL KTEXT('WRONG 1'.0)
CALL KMOVAB(-2050.0,400.0)
CALL KTEXT('RE-GENERATE THE 1'.0)
CALL KM0VAB(-2050.0,300.0)
CALL KTEXT('MOTION SEQUENCE 1'.0)
CALL KUPDAT
CALL L0CAT(X,Y,IL)
IF(Y.LT.500.) RETURN 1
ELSE ENDF
ELSE ENDF
CALL DOMTVJ
CALL LVVPRD(0.04,472.0,259.0,691.0)
CALL KWINDOW(-1.15,1.15,-1.15,1.15)
IF(L.EQ.'2' .OR. L.EQ.'3') CALL BOUND4(Ø)
CALL RCTFY
TVOL=0.
TVOL1=0.
TVHIX=0.
TVHIY=0.
TAREA=0.
TVDZ=0.
TVD=0.
TADA=0.
TAC=0.
IF(/TYPE(1).EQ.*P*) THEN
CALL PRIVOL
ELSE
QUAD=2.*THMIN(I)*RTH(I)-PI
DO 10 I=1,NM-1
IF(THETA(I).NE.TRUE.) THEN
THETA=S5E*PI/180.
X=RAT(I)
Y=PC(I)
AFA=SHI(I)
FL=PHY(I)
M=SORT(X)*X+Y*Y*SIN(AFA-SHI(I))
AREA(I)=H*PL/2.0E
IF(AREA(I).GE.0.) THEN
10 CONTINUE
END
SIGN=1.
ELSE
SIGN=-1.
ENDIF
IF(ABS(AREA(I)) LT 1.E-5) THEN
AZ(I)=0.
AREA(I)=0.
ELSE
ASZ(I)=PL*H*(Y1+PL/2.*SIN(AFA))/3.E9
AZ(I)=ASZ(I)/AREA(I)
ENDIF
VOL(I)=PL*H*THETA*(X1+PL/2.*COS(AFA))/3.E9
IF(ABS(VOL(I)) LT 1.E-6) THEN
VOL(I)=0.
VZ(I)=0.
RCX(I)=0.
VX(I)=0.
VMIX(I)=0.
VMIV(I)=0.
VMIZ(I)=0.
ELSE
VDZ(I)=PL*H*THETA*(X1*Y1+PL*X1/2.*SIN(AFA)+Y1*PL/2.*COS(AFA)+PL**2/3.*COS(AFA))/4.E12
AA=H*PL*(X1**2*X1*PL*COS(AFA)+PL**2/3.*COS(AFA))*2/3.E9
VDX(I)=AA*(SIN(THETA/2.)*SIN(-THETA/2.))/4.E12
RCX(I)=VOL(I)/AREA(I)/THETA
VX(I)=VDX(VOL(I))
VZ(I)=VDZ(VOL(I))
RX(I)=PL*H*(X1**2*X1**2*PL*COS(AFA)+PL**2/3.*COS(AFA))/4.E15
ZH(I)=PL*THETA*(X1*Y1**2*Y1**2*PL/2.*COS(AFA)+X1*Y1*PL
*SIN(AFA)+PL/3.0*(X1*SIN(AFA)+Y1*PL)*SIN(2.0*AFA))
==PL**3/8.*SIN(2.*AFA)*SIN(AFA)+PL**3/8.*SIN(2.*AFA)*SIN(AFA)+PL**3/8.*SIN(2.*AFA)/3.E15
COS=J*(THETA-5.*(SIN(THETA)-SIN(-THETA)))
SIN=5.*(THETA-5.*(SIN(THETA)-SIN(-THETA)))
VMIX(I)=SIN*RM1+ZMI/1.E15
VMIV(I)=COS*RM1+ZMI/1.E15
VMIZ(I)=THETA*RM1
ENDIF
ELSE
IF(SWEEP GT 100) THEN
POS=*TPX(LOOP(I)+1)
IF(POS<0. AND. QUAD<0.) THEN
THETA=1.0-SWEEP/360.*PI*2.
ELSE IF(POS LT 0. AND. QUAD LT 0.) THEN
THETA=1.0-SWEEP/360.*PI*2.
ELSE
THETA=SWEEP*PI/180.
ENDIF
ELSE
THETA=SWEEP*PI/180.
ENDIF
AA=PC(I)*PC(I)*COS(PHY(I))+(SIN(SHI(I)-PHY(I))-SIN(SHI(I)-PHY(I))
BB=3.0*PC(I)*PC(I)*SIN(SHI(I)-SHI(I))*COS(PHY(I))
CC=RA(I)*PC(I)*4.*SIN2(SHI(I))-PHY(I))*SIN2(SHI(I))-PHY(I))
DD=RA(I)*RA(I)*SIN(SHI(I))=SIN(SHI(I))
EE=PA(I)*THETA/3
VOL(I)=EE*(AA+CC+DD)/1.E9
AA=12*RA(I)*PC(I)*PC(I)*(SHII1(I)-SHI0(I))*SIN(2*PHY(I))
BB=15*RA(I)*RA(I)*PC(I)*PHY(I)*SIN(2*SHI1(I))
CC=3*RA(I)*RA(I)*PC(I)*PC(I)*(COS(2*SHI1(I))
-DD=-PC(I)*PC(I)*PC(I)*SIN(2*PHY(I))
EE=RA(I)*RA(I)*PC(I)*(COS(3*SHII1(I)-PHY(I))
FF=RA(I)*THETA/48.
VDZ1=FF*(AA-DD-CC-DD-EE)/1.E12
AA=Sin(THETA/2.)-SIN(-THETA/2.)
BB=RA(I)**3*(2.*SHI1(I)-SIN(2.*SHI1(I))
CC=PC(I)**2*RA(I)**2.25*SIN(2.*PHY(I))
-DD=-PC(I)*RA(I)**2*SHI1(I)*SIN(2.*PHY(I))
EE=-PC(I)**3*COS(PHY(I))**2*(S1N(2.*SHI0(I))))
VF(X)=RA(I)*AA+BB+CC-DD-EE)/1.42.

IF(ABS(VLO(I)).LT.1.E-15) THEN
   VOL(I)=0.
   VZ(I)=0.
   VX(I)=0.
ENDIF
IF(ASA(A)=II.E-10) THEN
   RCX(I)=B.
   AREA(I)=0.
ELSE
   PCX(I)=VLO(I)/AREA(I)/THETA
ENDIF
AA=PC(I)*PC(I)*SIN(PHY(I))
BB=RA(I)*PC(I)**2*(SHII1(I)-SHI0(I))
CC=RA(I)*PC(I)**2*COS(2*SHI1(I)-PHY(I))
-DD=-RA(I)*RA(I)*COS(2*SHI0(I))
ASZ=RA(I)/3*(AA+BB-CC-DD)/1.42.

IF(AREA(I)=E.0.) THEN
   AZ(I)=0.
ELSE
   AZ(I)=ASZ(I)/AREA(I)
ENDIF
AA=0.2*(PC(I)**2*(PC(I)**2-9.*RA(I)**2)*SIN(2.*PHY(I)*SHI1(I))
BB=3.*PC(I)*RA(I)**3.*PC(I)**2.7.*RA(I)**2*SIN(PHY(I))

1

229
CC=3.*PC(1)*RA(1)*(6.*PC(1)**2+9.*RA(1)**2)*(SIN(PHY(1))
1 2-9.*RA(1)**2)*SIN(PHY(1)**2)*SHI(1))
1 DD=6.*PC(1)**2*(3.*PC(1)**2+12.*RA(1)**2)*(SIN2.*PHY(1)
1 -SHI(1))*SIN(2.*PHY(1)-SHI(1)))
1 EE=2.*RA(1)**2*(3.*PC(1)**2+2.*RA(1)**2)*(SIN3.*SHI(1))
1 FF=6.*PC(1)*PC(1)**2*SHI(1)**2*PC(1)*RA(1)**2*COS(3.*PHY(1))
1 3.*PC(1)**2+2.*RA(1)**2)*COS(PHY(1)))
1 GG=6.*PC(1)**2*2.*PC(1)**2*2.*RA(1)**2+6.*RA(1)**4)*(SIN(1
1 SHI(1)-SHI(1)))
1 HH=6.*PC(1)*PC(1)**2*(SIN4.*PHY(1)-SHI(1)**2)*SIN4.*PHY(1)-
1 SHI(1)**2)*25*RA(1)**2*(SIN(4.*SHI(1)-PHY(1))
1 2-5IN(4.*SHI(1)-2.*PHY(1))
1 PP=RA(1)**2*THETA/240.
1 QQ=AA+CC*DD*EE*FF*GG*HH*OO
1 AA=1.*PC(1)*PC(1)**2+4.*RA(1)**2)*SIN(2.*PHY(1))
1 BB=-30.*PC(1)**2+2.*RA(1)**2*(SIN(SHI(1))**2-2.*PHY(1)-
1 SHI(1)**2+2.*PHY(1))
1 CC=6.*PC(1)**2+12.*PC(1)**2)*FA(1)**2)*SIN(PHY(1))
1 DD=-3.*PC(1)*RA(1)*(PC(1)**2+4.*RA(1)**2)*(SIN2.*SHI(1)**
1 PHY(1)-SIN(2.*SHI(1)**2-2.*SHI(1)**2+2.*PHY(1))
1 EE=-4.*SHI(1)-SIN(2.*SHI(1)-2.*PHY(1))
1 FF=6.*PC(1)**2*FA(1)**2*(SIN(SHI(1)-3.*PHY(1))
1 GG=30.*PC(1)**2*FA(1)**2)*COS(PHY(1))
1 HH=THETA*RA(1)**2/240.
1 QQ=6.*PC(1)**2+2.*FA(1)**2)*SIN(SHI(1)**2+2.*FA(1)**2)*
1 (SIN(SHI(1)**2)-SIN(SHI(1))))
1 PP=3.*PC(1)**2+2.*FA(1)**2)*(SIN3.*SHI(1)**2-3.*SHI(1)**
1 RR=RA(1)**2*(SIN(SHI(1)**2)-SIN(SHI(1)))*QQ/240.
1 SS=AA+BB*CC*DD+EE+FF+GG+OO+PP
1 TT=RA(1)**2*(SIN(SHI(1)**2)-SIN(SHI(1)))*QQ/240.
1 VMIX=THETA**SS/TT1/1.E15
1 END IF
1 IF(ABS(VHIX(1))>1.E-15) VMIX(1)=0.
1 IF(ABS(VHIX(1))>1.E-15) VMIX(1)=0.
1 IF(ABS(VHIX(1))>1.E-15) VMIX(1)=0.
1 VOL(1)=SIN*ABS(VOL(1))
1 VMIX(1)=SIN*ABS(VHIX(1))
1 VHIX(1)=SIN*ABS(VHIX(1))
1 VOL(1)=VOL+VOL(1)
1 TVAL=VHIX(1)+VHIX(1)
1 VMIX=VHIX(1)+VHIX(1)
1 TAREA=TAREA+AREA(1)
1 TASZ=TASZ+ASZ(1)
IF(L.EQ.'.'I') THEN
  AL(I)=64*I
  ALI=AL(I)/1000.
  PX=TPX(LOOP(I))/1000.
  PY=TPY(LOOP(I))/1000.
  CALL KMOVAB(0.,0.)
  CALL KNTYP('DOTTED')
  CALL KDRWAB(PX,PY)
  CALL KNTYP('SOLID')
  CALL CROSS(RCX(I),AZ(I),01.,01.,AL)
ELSE
  ENDIF
IF(LW)
  WRITE(11,1) VOL(I),VZ(I),VX(I),AREA(I),AZ(I),RCX(I),VMIX(I)
  VMIY(I),VHIZ(I)
11 FORMAT('-------- SECTION ',I2,' --------')
  /'.5X,' TOTAL VOLUME =',E13.5,'X,' CENTROID V-Z =',E13.5,
  /'.5X,' CENTROID V-X =',E13.5,
  /'.5X,' AREA =',E13.5,'X,' CENTROID A-Z =',E13.5,
  /'.5X,' MT. OF IN. K =',E13.5,
  /'.5X,' MT. OF IN. Y =',E13.5,
  /'.5X,' MT. OF IN. Z =',E13.5)
  VDZ(I)=VOL(I)*VZ(I)
  VX(I)=VOL(I)*VX(I)
TVX=TVOL+VDX(I)
  TAZA=AREA(I)*THETA
  TRCX=TVOL/TADA
  RXX=SORTA(VMIX)/TVOL
  RYY=SORTA(VMIY)/TVOL
  RZZ=SORTA(VHIZ)/TVOL
  TVOL=TVOL/4.10079
  CALL KVXSZI(.012)
  CALL CROSS(TVX,TVZ,02.,02.,'AC')
  CALL CROSS(TRCX,TAZ,02.,02.,'AC')
IF(LW) WRITE(12,22) TVOL,TVX,TVZ,AREA,TRCX,TAZ,VMIX,RXX,VMIIY,RYK,VHIZ,RKZ
22 FORMAT('..... TOTAL ..........')
  /'.5X,' TOTAL VOLUME =',E13.5,'X,' CENTROID V-K =',E13.5,
  /'.5X,' CENTROID V-Z =',E13.5,
  /'.5X,' CENTROID A-Z =',E13.5,
  /'.5X,' MT. OF IN. K =',E13.5,'X,' CENTROID A-Z =',E13.5,
  /'.5X,' MT. OF IN. Y =',E13.5,'X,' CENTROID A-Z =',E13.5,
  /'.5X,' MT. OF IN. Z =',E13.5,'X,' CENTROID A-Z =',E13.5)
ENDIF
CALL KVXPS(.518,.95,.259,.691)
CALL KWINDO(-1.15,1.1,-1.15,1.1)
CALL CROSS(TVX,0,.02,.02,'VC')
CALL KVXPIO(0.,0.0,.76171879)
CALL KWINDO(0.,1.024,.0,.780)
CALL LISRL
CALL KMOVAB(12.,8.)
CALL KTEXT('HIT <CR> TO CONTINUE',0)
CALL KPAUSE(0,0)
RETURN 2
END

SUBROUTINE WELCOM

SUBROUTINE WELCOM
CHARACTER*35 TITLE
TITLE='MAIN/INTRODUCTION'
CALL FRAME(TITLE)
CALL KTEX12(.025)
CALL KMOVAD(10.,730.)
CALL KTEXT(' INTRODUCTION:',1)
CALL KTEX12(.022)
CALL KTEXT
1(' 'This program is capable of generating workspace for some',1)
CALL KTEXT
2(' industrial robots. In general, this program can handle the',1)
CALL KTEXT
3(' robot having the following manipulator parameters:',1)
CALL KTEXT
4(' Joint Angle: variable when revolute joint.',1)
CALL KTEXT
5(' constant for intersecting sliding joints.',1)
CALL KTEXT
6(' 90 degree for prismatic joint intersecting.',1)
CALL KTEXT
7(' constant for revolute joint.',1)
CALL KTEXT
8(' Link Length: zero for prismatic joint.',1)
CALL KTEXT
9(' constant for revolute joint.',1)
CALL KTEX12(.025)
CALL KTEXT
10(' Twist Angle: 90 degree for 1st joint when revolute.',1)
CALL KTEXT
11(' 0 degree for all other revolute joints.',1)
CALL KTEX12(.025)
CALL KTEXT
12(' constant for prismatic joint intersecting.',1)
CALL KTEX12(.022)
CALL KTEXT
13(' with revolute joint.',1)
CALL KTEX12(.025)
CALL KTEXT
14(' Offset: zero for revolute joints.',1)
CALL KTEX12(.025)
CALL KTEXT
15(' variable for prismatic joints.',1)
CALL KMOVAD(2.,8.)
CALL KTEX12('HIT <CP> TO CONTINUE',0)
CALL KPAUSE(0.0)
CALL FRAME(TITLE)
CALL KMOVAD(1.,730.)
CALL KTEXT
1(' That is, the first two joints intersected, and all',1)
CALL KTEXT
1(' other links remain on a plane. If the first joint is',1)
CALL KTEXT
1(' revolute, the plane contains the first joint axis; if',1)
CALL KTEXT
1(' the first joint is prismatic, the plane cut the first',1)
CALL KTEX12(.025)
CALL KTEXT
1(' joint axis at right angles.',1)
CALL KTEXT
1(*) The work space boundary generation is based on finding'.1)
CALL KTEXT
2(*) singular surfaces by applying a wrench on the tip of'.1)
CALL KTEXT
3(*) robot hand. When the wrench intersects all the joint'.1)
CALL KTEXT
4(*) axes, the robot loss at least one degree of freedom -'.1)
CALL KTEXT
5(*) a singular position. By rotating the wrench, the'.1)
CALL KTEXT
6(*) singular surface can be found on a specified plane.'.1)
CALL KTEXT
7(*) The cross-sectional area as well as the volume bounded'.1)
CALL KTEXT
8(*) by the generated surface are calculated, and the'.1)
CALL KTEXT
9(*) overlapped regions due to excessive motion range can'.1)
CALL KTEXT
1(*) be eliminated interactively. The moments of inertia'.1)
CALL KTEXT
11(*) and radii of gyration about three principal axes can'.1)
CALL KTEXT
2(*) also calculated.'.1)
CALL KHVAD(2.,0.)
CALL KTEXT('HIT <CR> TO CONTINUE',0)
CALL KPAUSE(0.0)
RETURN
END

SUBROUTINE JACOBN
This subroutine form the Jacobian matrix by providing joint
transformation matrix U, V, and S. Matrix U and V are in
(6,3,3) dimension, matrix S is (6,3), the result Jacobian
matrix Bj is (6,6).
All other matrices are intermediate.

SUBROUTINE JACOBN
CHARACTER JTYPE
CHARACTER*10 MODEL
CHARACTER*20 MAKER
COMMON /TRHS/U(6,3,3),V(6,3,3),S(6,3),BJ(6,6)
COMMON /CHR/ MODEL, MAKER, JTYPE(6)
DIMENSION UV(3,3),US(3),Q1(3,3),R(3),W(6,3),RO(6,3),PAMDA(6,3)
DIMENSION QUV(3,3),QUS(3)
DO 10 I=1,3
R(I)=0.
DO 11 J=1,3
Q(I,J)=0.
IF(I.EQ.J) Q(I,J)=1.
11 CONTINUE
10 CONTINUE
DO 20 K=1,6
W(K,1)=Q(1,1)
W(K,2)=Q(2,3)
W(K,3)=Q(3,3)
RO(K,1)=R(1)
RO(K,2)=R(2)
RO(K,3)=R(3)
PAMDA(K,1)=RO(K,2)*W(K,3)-RO(K,3)*W(K,2)
SUBROUTINE TRSMAT

This subroutine forms transformation matrix \( U.V \) and \( S \) from the provided joint parameters \( \alpha, \theta, h, \) and \( \xi \). The dimension of \( U.V \) is \( (6,3,3) \), and \( S \) is \( (6,3) \).

The input-output are all given in block common named \( /PAR/ \) and \( /TRN/ \).

SUBROUTINE TRSMAT

COMMON /PAR/ alpha(6),theta(6),offset(6),h(6),
COMMON /TRN/ u(6,3,3),v(6,3,3),s(6,3),bj(6,6)

DO 100 K=1,6
  U(K,1,1)=cos(theta(K))
  U(K,1,2)=cos(theta(K))
  U(K,1,3)=0.
  U(K,2,1)=-sin(theta(K))
  U(K,2,2)=cos(theta(K))
  U(K,2,3)=0.
  U(K,3,1)=0.
  U(K,3,2)=0.
  U(K,3,3)=1.
  V(K,1,1)=1.
  V(K,1,2)=0.
  V(K,1,3)=0.

END
V(K,2,1)=B
V(K,2,2)=COS(ALPHA(K))
V(K,2,3)=-SIN(ALPHA(K))
V(K,3,1)=B
V(K,3,2)=SIN(ALPHA(K))
V(K,3,3)=COS(ALPHA(K))
S(K,1)=LENGTH(K)
S(K,2)=B
S(K,3)=OFFSET(K)
188 CONTINUE
RETURN
END

C SUBROUTINE MUL633
C This subroutine make a multiple of two matrix (3,3). The matrices
C are embedded in (6,3,3) matrices.
C Input: A(6,3,3) x B(6,3,3) in their Nth (3,3) matrices.
C Output: C(3,3)

DIMENSION A(6,3,3),B(6,3,3),C(3,3)
DO 10  I=1,3
DO 20  J=1,3
SUM=0
DO 30  K=1,3
SUM=SUM+A(I,K)*B(K,J)
30 CONTINUE
C(I,J)=SUM
20 CONTINUE
10 CONTINUE
RETURN
END

C SUBROUTINE MUL33
C This subroutine make a multiple of two matrix (3,3).
C Input: A(3,3) x B(3,3)
C Output: C(3,3)

DIMENSION A(3,3),B(3,3),C(3,3)
DO 10  I=1,3
DO 20  J=1,3
SUM=0
DO 30  K=1,3
SUM=SUM+A(I,K)*B(K,J)
30 CONTINUE
C(I,J)=SUM
20 CONTINUE
10 CONTINUE
RETURN
END

C SUBROUTINE MUL631
C This subroutine make a multiple of two matrix (3,3) AND (3,1).
C The matrices are embedded in (6,3,3) and (6,3,1) matrices.
C Input: A(6,3,3) x B(6,3,1) in their Nth (3,3) and (3,1) matrices.
C Output: C(3,1)

DIMENSION A(6,3,3),B(6,3,1),C(3,1)
DO 10  I=1,3
DO 20  J=1,3
SUM=0
DO 30  K=1,3
SUM=SUM+A(I,K)*B(K,J)
30 CONTINUE
C(I,J)=SUM
20 CONTINUE
10 CONTINUE
RETURN
END
SUBROUTINE MUL31(A,B,C,N)
DIMENSION A(3,3),B(3),C(3)
DO 10 I=1,3
SUM=C(I)
DO 20 J=1,3
SUM=SUM+A(I,J)*B(J)
20 CONTINUE
C(I)=SUM
10 CONTINUE
RETURN
END

SUBROUTINE MUL31(A,B,C)
DIMENSION A(3,3),B(3),C(3)
DO 10 I=1,3
SUM=C(I)
DO 20 J=1,3
SUM=SUM+A(I,J)*B(J)
20 CONTINUE
C(I)=SUM
10 CONTINUE
RETURN
END

SUBROUTINE MENU2
CHARACTER*CG TITLE
TITLE="MAIN/MAIN/MAIN/MAIN"
CALL FRAME(TITLE)
CALL KMOVAC(304.,750.)
CALL KDRVAB(304.,30.)
CALL KMOVAC(304.,150.)
CALL KDRVAB(304.,150.)
CALL KMOVAC(304.,270.)
CALL KDRVAB(304.,270.)
CALL KMOVAC(304.,390.)
CALL KDRVAB(304.,390.)
CALL KMOVAC(304.,510.)
CALL KDRVAB(304.,510.)
CALL KMOVAC(304.,630.)
CALL KDRVAB(304.,630.)
CALL KTEXT("UPDATE LINK PARAMETERS\",0)
CALL KMOVAC(0.,550.)
CALL KTEXT("ALTER VARIABLE FOR LINKS\",0)
CALL KMOVAC(0.,430.)
CALL KTEXT("CHANGE INACTIVE JOINTS\",0)
CALL KMOVAC(0.,310.)
CALL KTEXT("LIST JOINT PARAMETERS\",0)
CALL KMOVAC(0.,190.)
CALL KTEXT(' CLEAR THE SCREEN', 'B')
CALL KMOVAB(D, ' .75, )
CALL KTEXTI(' EXIT INPUT AND GO AHEAD', 'B')
RETURN

END

C Subroutine PARINP
C This subroutine lets user input or modify joint parameters for
C robot to calculate manipulability and Jacobian matrix.
C New variables used are:
C TLEHG: Total link length of new manipulator

C-------------------------------------------------------------------
SUBROUTINE PARINP
CHARACTER JTVPE, I, I, I
CHARACTER*10 MODEL
CHARACTER*20 MAKER
LOGICAL RF, IW, COMP, ROTA, INACT
COMMON /PAP/ALPHA(I), THETA(I), OFFSET(I), HLENGTH(I)
COMMON /CH/MODEL, MAKER, JTVPE(I)
COMMON /LAT/NUMI(I), NUMSI(I), NL(I), NDO(I)
COMMON /L05/IW, RF, COMP, ROTA(I), INACT(I)
CALL KTEXTR(0, ' .3')
CALL MENU2
CALL KLOCAT(X, Y, IC)
IF (IC .NE. ' ' ) GOTO 99
CALL KTXMP(320, 150)
CALL KTXSIZ(0, 0, 12)
IF (Y GT 650.) THEN
10 CALL KTEXT(' ENTER LINK NUMBEF TO BE CHANGED', ' .3')
CALL KREAD(I, NO, '10.0')
CALL KLIFDI(1)
IF (NO .LT. I .AND. NO .GT. 6) GOTO 10
IF (NO .LE. 3) THEN
11 CALL KTEXT(' ENTER JOINT TYPE (R or P)', ' .3')
CALL IRCADJ(JTVPE(I, NO), HELP, ' R ')
CALL ILINFDO(I)
IF (JTVPE(I, NO) EQ ' P ') JTVPE(I, NO) = ' P '
IF (JTVPE(I, NO) EQ ' R ') JTVPE(I, NO) = ' R '
IF (JTVPE(I, NO) NE ' P ' .AND. JTVPE(I, NO) NE ' R ') THEN
CALL KTEXT(' INVALID JOINT TYPE', ' .1')
GOTO 11
ELSE
ELSE END IF
ELSE END IF
20 CALL KTEXT(' ENTER JOINT ANGLE (DEGREES)', ' .3')
CALL KREAD(I, THETA(I, NO), '20.0. ')
CALL KLIDFi(I)
30 CALL KTEXT(' ENTER TWIST ANGLE (DEGREES)', ' .3')
CALL KREAD(I, ALPHA(I, NO), '30.0. ')
CALL KLIDFi(I)
40 CALL KTEXT(' ENTER LINK LENGTH', ' .3')
CALL KREAD(I, HLENGTH(I), '40.0. ')
CALL KLIDFi(I)
50 CALL KTEXT(' ENTER OFFSET ', ' .3')
CALL KREAD(I, OFFSET(I), '50.0. ')
CALL KLIDFi(I)
THETA(I, NO) = THETA(I, NO) + P1/180.
ALPHA(I, NO) = ALPHA(I, NO) + P1/180.
ELSE IF (V.CT.510.) THEN
ENDIF
CALL KTEXT(' ENTER LINK NUMBER TO MODIFY:'.0)
CALL KREADI1,1,NO.,*60,0)
CALL KLINFD1)
IF(NO.LT.1 .OR. NO.GT.6) GOTO 60
IF(TYPE(NO).EQ.'P') THEN
   CALL KTEXT(' ENTER NEW OFFSET:'.0)
   CALL KREADI1,OFFSET(NO),*70,0.)
ELSE
   CALL KTEXT(' ENTER NEW JOINT ANGLE (DEGREE):'.0)
   CALL KREADI1,THETA(NO),*80,0.)
   THETA(NO)=THETA(NO)*PI/180.
ENDIF
CALL KLINFD1)
ELSE IF(Y.GT.390.) THEN
   CALL KTEXT(' A -- ADD AN INACTIVE JOINT'.1)
   CALL KTEXT(' D -- ACTIVE A JOINT..<A>',0)
   CALL KREADC11,IHELP,'A')
   CALL KLINFD1)
   IF(NO.EQ.'A') THEN
      CALL KTEXT(' ENTER A INTERACTIVE JOINT:'.0)
      CALL KREADI1,NO.,*90,0)
   ELSE IF(NO.GE.6 .AND. NO.LE.6) THEN
      INACT(NO)=.TRUE.
      GOTO 90
   ELSE IF(NO.GT.6) THEN
      CALL KTEXT(' INVALID JOINT NUMBER'.1)
      GOTO 90
   ELSE
      ENDIF
   ELSE IF(NO.EQ.'D') THEN
      CALL KTEXT(' ENTER AN ACTIVE JOINT:'.0)
      CALL KREADI1,NO.,*92,0)
      CALL KLINFD1)
      IF(NO.EQ.'B' .AND. NO.LE.6) THEN
         INACT(NO)=.FALSE.
      GOTO 92
   ELSE IF(NO.GT.6) THEN
      CALL KTEXT(' INVALID JOINT NUMBER'.1)
      GOTO 92
   ELSE
      ENDIF
   ELSE
      GOTO 91
   ENDIF
ELSE IF(Y.GT.270.) THEN
   CALL KIQOSI(X,Y)
   IF(Y.LT.120.) THEN
      CALL MENUS
   ELSE
      ENDIF
   CALL KTEXT(' LINK, TYPE, LENGTH, OFFSET, ANGLE, TWIST:'.1)
   DO 100 I=1,6
      WRITE(C,101) 1.,TYPE(I),HLENCT(I),OFFSET(I),THETA(I),ALPHA(I)
   101 FORMAT(3X,12,2.,A1,2.,4(2X,F8.3))
   CALL KIQOSI(X,Y)
CALL KTEXT(S,1)
CONTINUE
ELSE IF(Y.GT.150.) THEN
CALL MENU2
ELSE
TLENG=R
DO 93 N=1,6
TLENG=TLENG+HLENGT(N)+OFFSET(N)
CONTINUE
DO 94 L=1,6
HLENGT(L)=HLENGT(L)/TLENG
OFFSET(L)=OFFSET(L)/TLENG
CONTINUE
ENDIF
RETURN
ENDIF
GOTO 99
END

SUBROUTINE SUBSTK(I,L)
This subroutine substitutes the joint variable for each configuration of robot-arm into joint parameter arrays and sorts inactive joints.

Input:
- I - the number of sequence or configuration
- JTYPE - joint type supplied in block common /chr/
- VAR* - in block common /var/ for substitute values
- IUACT - inactive joint number

Output:
- INACT - inactive joint number in individual manipulation

SUBROUTINE SUBSTK(I,L)
CHARACTER JTYPE,IL
CHARACTER*10 MODEL
CHARACTER*20 MAKER.
LOGICAL RF,IV,COMP,ROTA,INACT
COMMON /PAR/ALPHA(6),THETA(6),OFFSET(6),HLENGT(6)
COMMON /CHP/MODEL,MAKER,JTYPE(6)
COMMON /VAR/VAR(999),VAR2(999),VAR3(999)
COMMON /AT/NUM(60),NUM3(60),NL(300)
COMMON /LOG/RF,COMP,ROTA(60),INACT(6)

SUBSTITUTE JOINT VARIABLE INTO JOINT PARAMETER ARRAYS

IF(JTYPE(2),.EQ.,'P') THEN
OFFSET(2)=VAR1(I)/1000.
ELSE
HLENGT(2)=HLENGT(2)
THETA(2)=VAR1(I)
ENDIF
IF(JTYPE(3),.EQ.,'P') THEN
OFFSET(3)=VAR2(I)/1000.
ELSE
HLENGT(3)=HLENGT(3)
THETA(3)=VAR2(I)
ENDIF
THETA(4)=VAR3(I)

SORTING INACTIVE JOINTS

DO 12 K=1,6
INACT(K)=.FALSE.
CONTINUE
IF (IL.EQ.'Y', OR, IL.EQ.'Y') THEN
  IF (INUM(1).EQ.2) THEN
    INACT(2)=.TRUE.
  ELSE IF (INUM(1).EQ.3) THEN
    INACT(3)=.TRUE.
  ELSE IF (INUM(1).EQ.4) THEN
    INACT(4)=.TRUE.
  ELSE IF (INUM(1).EQ.23) THEN
    INACT(23)=.TRUE.
    INACT(2)=.TRUE.
    INACT(3)=.TRUE.
  ELSE IF (INUM(1).EQ.24) THEN
    INACT(24)=.TRUE.
  ELSE IF (INUM(1).EQ.234) THEN
    INACT(234)=.TRUE.
    INACT(2)=.TRUE.
    INACT(3)=.TRUE.
    INACT(4)=.TRUE.
ELSE
  ENDDO
ELSE
ENDIF
RETURN
END

SUBROUTINE MENU
THIS SUBROUTINE PLOT A SELECTION MENU FOR 'MANIPL'

CHARACTER*35 TITLE
TITLE='MAIN/MANIPL'
CALL FRAME(TITLE)
CALL KMOVAB(304.,750.)
CALL KDPWAB(304.,318.)
CALL KMOVACI(304.,462.)
CALL KMOVACI(304.,500.)
CALL KMOVACI(304.,606.)
CALL KMOVACI(304.,606.)
CALL KTEXTP('USE PROGRAM DATA',0)
CALL KMOVAC(5.,606.)
CALL KTEXTP('INPUT NEW DATA',0)
CALL KMOVAC(5.,375.)
CALL KTEXTP('RETURN',0)
CALL KMOVAC(5.,226.)
CALL KTEXTP('EXIT',0)
RETURN
END

SUBROUTINE MANIPL
This subroutine serves as the main program of Jacobian matrix,
singular value decomposition, and manipulability calculation.
It calls subroutines PARIMP, TRSMAT, JACOBN, and LSVDF.

LSVD is in the IMSL scientific subroutine.

SUBROUTINE LSVDF(IN, B, J)

Input: B(J,J) the Jacobian matrix.

Output: VM(J,J) - the orthogonal matrix for B = VM * SIG * UT

The singular value decomposition.

S.append(6) - the singular values of B

UT(J,J) - the orthogonal matrix containing eigenfunctions

KRA1 - the rank of B

ALBILTY - THE MANIPULABILITY

SUBROUTINE MANIPL

CHARACTER IC, IL, JTYPE

INTEGER NPTS(I), LOGICAL RF, INACT, VR

COMMON /LENGTH/RATIO(A), B(J,J), PI, RTH(4), TH(4), TWIST(4)

COMMON /ITC/LAST.MM, LOOP(I), NSE, IANG, LL, IP, IDIR, ICOUNT(J), JJ

COMMON /CH/MODEL, LMK(6), JTYPE(I)

COMMON /TRM/UT(J,J), SIG, UT(J,J)

COMMON /VAR/VAR(1), VA(2), VAR(3)

COMMON /PAR/ALPHA, THETA, OFFSET, HLEN(6)

COMMON /IAT/IUM, NDO, NLD(J), J J

COMMON /ID/RF, IN(6), ICOUNT(J)

DIMENSION X(J), Y(J), SIG(I)

CALL KTEXT(0)

CALL KLOCAT(PX, PY, IC)

IF(IC.EQ. 'A' OR IC.EQ. 'b')) GOTO 50

CALL KTEXT(32)

IF(PY.GT.520.) THEN

WRITE(STRING, 33) NSE

FORMAT(1 TOTAL SEQUENCES: '12)

CALL KTEXT(STRING, 1)

CALL KTEXT( 'A -- USE ALL SEQUENCES: ')

CALL KTEXT( 'B -- USE ONE SEQUENCE: ')

CALL KTEXT( 'C -- WRITE DATA FILE (')

CALL KREAD(1)

CALL KINF(I)

CALL KINF(1)

CALL KINF(I)

CALL KINF(1)

IF(IC.EQ. 'A' OR IC.EQ. 'b') THEN

VR = .TRUE.

DO 30 I = 1, NSE

CALL SUBST(I, IL)

CALL TRSMAT

CALL JACOBN

CALL HBLTY(VR, KRA1, ALBILTY, SIG)

CONTINUE

ELSE IF(IC.EQ. 'B' OR IC.EQ. 'b') THEN

CALL KINF('WRITE DATA FILE (')

CALL KINF(' ')

IF(IL.EQ. 'Y' OR IC.EQ. 'd') THEN

VR = .TRUE.

ELSE

DO 30 I = 1, NSE

CALL SUBST(I, IL)

CALL TRSMAT

CALL JACOBN

CALL HBLTY(VR, KRA1, ALBILTY, SIG)

CONTINUE

ELSE IF(IC.EQ. 'B' OR IC.EQ. 'b') THEN

CALL KINF('WRITE DATA FILE (')

CALL KINF(' ')

IF(IL.EQ. 'Y' OR IC.EQ. 'd') THEN

VR = .TRUE.

ELSE

...
VR=.FALSE.
ENDIF
32 CALL KTXSIZ(0.022)
CALL KMOVAB(.8.)
CALL KTEXT(' ENTER 0 TO EXIT',0)
CALL KMOVAB(1..48.)
CALL KTEXT(' ENTER SEQUENCE NO<(0>',0)
CALL KREAD(1,NO.*31,8)
CALL KLINFD(1)
IF(NO.GT.NSE) THEN
CALL KTEXT(' INVALID NUMBER',1)
GOTO 31
ELSE IF(NO.NE.0) THEN
33 J=1
DO 34 J=LOOP(NO). LOOP(NO+1) 
CALL SUBST(1,IL)
CALL TRSMAT
CALL JACOBN
CALL MABLTY(1,WR,KRAN,ABILTY,SIG)
IF(JTYPE(NUMS(NO)).EQ.'P') THEN
X(J)=OFFSET(NUMS(NO))*RATIO
ELSE
X(J)=THETA(NUMS(NO))/PI*100.
ENDIF
Y(J)=ABILTY
J=J+1
CONTINUE
34 IF(JTYPE(NUMS(NO)).EQ.'P') THEN
WRITE(LX,20) NUMS(NO)
ELSE
WRITE(LX,27) NUMS(NO)
ENDIF
20 F0RMAT(' JOINT NO. ',11,' PRISMATIC')
21 F0RMAT( ' JOINT NO. ',11,' REVOLUTE')
LT=' MANIPULABILITY'
LINE(1)=' SOLID'
XLEFT=300.
XRIGHT=760.
YBOTTM=250.
YTLP=600.
TITLE=' PLOT KANIPULABILITY'
CALL FRAME( TITLE)
CALL KGRAPH(X,Y,60.1,60.1,XLEFT,XRIGHT,YBOTTM,YTOP,LX,LY, 
LT,XMINS,XMAXS,IXEXP,YMINS,YMAXS,IYEXP,NPTS,LINE, 
CHAR,5000,500000,1ERR)
CALL KMOVAB(200.,220.)
WRITE(STRING,41) MAKER
41 FORMAT(' MAKER: ',20A)
CALL KTEXT(STRING,0)
CALL KMOVAB(200.,190.)
WRITE(STRING,42) MODEL
42 FORMAT(' MODEL: ',10A)
CALL KTEXT(STRING,0)
IF(INACT(1)) THEN
IAC=1
ELSE
   IACT=0
ENDIF
IF(INACT(2)) IACT=IACT*10+2
IF(INACT(3)) IACT=IACT*10+3
IF(INACT(4)) IACT=IACT*10+4
IF(INACT(5)) IACT=IACT*10+5
IF(INACT(6)) IACT=IACT*10+6
CALL KMOVAB(200,160.)
WRITE(STRING,43) IACT
FORMAT(' THE INACTIVE JOINTS ARE: ',I3)
CALL KTEXT(STRING,0)
CALL KMOVAB(2,0.)
CALL KTEXT('HIT <CR> TO CONTINUE',0)
CALL KPAUSE(0,0)
CALL MENU1
GOTO 32
ELSE
ENDIF
ELSE
CALL FIELD
ENDIF
ELSE IF(PY.GT.462. .AND. PY.LT.606.) THEN
   N=1
   WR=.TRUE.
   CALL PARINP
   CALL TRSMAT
   CALL JACOBN
   CALL MABLTYIN(WR,KRAN,ABILTY,SIG)
   N=N+1
   GOTO 40
ELSE IF(PY.LT.462. .AND. PY.GT.318.) THEN
   RETURN
ELSE IF(PY.LT.316.) THEN
   STOP
ENDIF
CALL KTMMRG(0.,0.)
GOTO 10
END

-----------------------------------------------------------------------
SUBROUTINE MABLTY
-----------------------------------------------------------------------
This subroutine calls the LSVDF in IMSL scientific library to
perform singular value decomposition and output the results.
Input: BJ(6,6) - the jacobian matrix
       INACT(6) - the inactive joints
Output: SIG(6) - singular values of BJ
       ABILTY - manipulability
       KR,N - rank of BJ
       UT(6,6) - the orthogonal matrix produced by LSVDF
       VM(6,6) - ditto
-----------------------------------------------------------------------
SUBROUTINE MABLTYIN(WR,KRAN,ABILTY,SIG)
CHARACTER*132 STRING
CHARACTER*20 MODEL
CHARACTER*10 JTYPE
DOUBLE PRECISION VM,UT,DSIG,WK
LOGICAL RF,FW,COMP,ROTA,INACT,WR
COMMON /CIW/MODEL,MAKER,JTYPE(6)
COMMON /PAR/ALPHA(6),THETA(6),OFFSET(6),HLENGTH(6)
COMMON /STH/UY(6,3,3),UY(6,3,3),SU(6,3,3),BU(6,3,3),C3(6,6)
COMMON /LST/NUM(60),NUMS(60),NL(300)
COMMON /LOG/IVRF,COMP,ROTA(60),INACT(6)
DIMENSION VM(6,6),UT(6,6),SIG(6),WK(12),DSIG(6)

COPY JACOBIAN MATRIX TO MATRIX VM

DO 10 J=1,6
DO 20 I=1,6
VM(I,J)=BJ(I,J)
UT(I,J)=0.
IF (I.EQ.0) UT(I,J)=1.
20 CONTINUE
10 CONTINUE

STRIKE OUT THE INACTIVE JOINTS FROM THE NEW JACOBIAN MATRIX

DO 70 I=1,6
IF (INACT(I)) THEN
DO 60 J=1,6
VM(K,I)=0.
60 CONTINUE
ELSE
ENDIF
70 CONTINUE

IF (INACT(1)) THEN
IACT=1
ELSE
IACT=0
ENDIF

IF (INACT(2)) IACT=IACT*10+2
IF (INACT(3)) IACT=IACT*10+3
IF (INACT(4)) IACT=IACT*10+4
IF (INACT(5)) IACT=IACT*10+5
IF (INACT(6)) IACT=IACT*10+6

DOING SINGULAR VALUE DECOMPOSITION

CALL LSVDF(VM,6,6,UT,6,6,DSIG,WK,IER)
KRAI=6
DO 33 I=6,1,-1
SIG(I)=DSIG(I)
33 CONTINUE
DO 30 I=1,6,1
IF(SIG(I).LT.1.D-6) KRAI=I-1
30 CONTINUE
ADJLTY=SIG(KRAI)/SIG(1)
IF(.NOT.WK) RETURN

OUTPUT RESULTS

WRITE(11,61) IMAK
FORMAT(1X,'/AX,' MAKER: ',A20)
WRITE(11,62) IMAK
FORMAT(4X,'/AX,' MODEL: ',A10)
WRITE(11,71) N
FORMAT(1X,'/AX,' JOINT PARAMETERS FOR SEQUENCE: ',A20)
1  'LINK:TYPE:LENGTH,BX,ANGLE,BX,OFFSET,BX, Twist')
DO 73 I=1,6
WRITE(1,72) 1,JTYPE(I),HLENGTH(I),THETA(I),OFFSET(I),ALPHA(I)
72 FORMAT(6X,12X,A1,4X,F10.5,3(4X,F10.5))
73 CONTINUE
WRITE(1,85)
85 FORMAT(/ THE JACOBIAN MATRIX:)
DO 86 J=1,6
WRITE(1,87) (BJ(I,J),J=1,6)
86 CONTINUE
WRITE(1,94) IACT
94 FORMAT(/ THE JACOBIAN MATRIX*)
DO 87 I=1,6
WRITE(1,95) I,B(I,I),I,J,1,6>
87 CONTINUE
WRITE(1,96) IACT
96 FORMAT(/ THE INACTIVE JOINTS ARE: *)
IF(IER.EQ.129) THEN
CALL KTEXT(' NOT CONVERGENCE'). N
WRITE(1,80) N
80 FORMAT(4X, 'SEQUENCE '.12,' IS NOT CONVERGENCE')
ELSE IF(IER.EQ.34) THEN
CALL KTEXT(' INACTIVE RANKS'). N
WRITE(1,91) N
91 FORMAT(4X, 'SEQUENCE '.12,' HAS INVALID RANKS')
ELSE IF(IER.EQ.33) THEN
WRITE(1,92) N
92 FORMAT(4X, 'SEQUENCE '.12,' RANK REDUCED OR ILL-CONDITIONED')
ELSE
WRITE(1,97) N
97 FORMAT(4X, 'SEQUENCE '.12,' SVD COMPLETED')
ENDIF
CALL KTXSIZ(0.022)
IF(N.EQ.1) THEN
CALL KTXMRG(340.,0., )
CALL KM0VAB(340.,730.)
ELSE IF(CN.EQ.21) THEN
CALL KTXMRG(600.,0., )
CALL KM0VAB(600.,730.)
ELSE
ENDIF
WRITE(STRING,31) N,KEAN,ABILITY
31 FORMAT(2X,1X,F10.5))
WRITE(1,63) I,RAN
63 FORMAT(4X, 'RANK='.12)
WRITE(1,93) ABILITY
93 FORMAT(4X, 'MANIPULABILITY='.E12.7)
WRITE(1,84) (SIG(I,I),I=1,6)
84 FORMAT(4X, 'SINGULAR VALUES: ',/ 3X,6(2X,D11.5))
WRITE(1,98) I,J
98 FORMAT(/ THE V MATRIX:*)
DO 89 I=1,6
WRITE(1,97) (VM(I,J),J=1,6)
89 CONTINUE
WRITE(1,90)
90 FORMAT(/ THE U MATRIX:*)
DO 91 I=1,6
WRITE(1,97) (UT(I,J),J=1,6)
91 CONTINUE
ENDIF
WRITE(2,68)
68 FORMAT(2X,6(1X,F10.5))
RETURN
END
THIS SUBROUTINE GENERATES POINTS INSIDE THE WORKSPACE BOUNDARY ON THE GENERATING PLANE. THE CORRESPONDING JOINT VARIABLE FOR JOINTS 2 AND 3 ARE SUBSTITUTED INTO JOINT PARAMETER ARRAYS. THEN MANIPULABILITIES OF THOSE POINTS ARE COMPUTED.

YOU CAN USE CONT2.EXE FOR TWO DIMENSIONAL CONTOUR PLOTTING OR USE CONT3.EXE FOR THREE DIMENSIONAL CONTOUR PLOTTING.

SUBROUTINE FIELD

CHARACTER IC, IL, JTYPE
CHARACTER*35 STRING, TITLE
CHARACTER*20 MAKER
CHARACTER*10 MODEL,FNAME
LOGICAL RF, IV, COMP, ROTA, INACT, WR
COMMON /ANGLE/ SWEEP, THMIN(4), THMAX(4), THS(4), THB(4)
COMMON /LENGTH/RAT10, A(4), B(4), P1, RTH(4), TH(4), TWIST(4)
COMMON /XY/CX1(999), CX2(999), CX3(999), CY1(999), CY2(999), CY3(999), 0(999), CY4(999), TPX(999), TPY(999), FD1(999), SD1(300)
COMMON /ITG/LAST, LL, LOOP(60), NSE, IANG, LL, IP, IDIR, ICOUNT(30), JJ
COMMON /CHF/ MODEL, MAKER, JTYPE
COMMON /PAR/ALPHA(6), THETA(6), OFFSET(6), HLENGTH(6)
COMMON /LOG/IW, RF, COMP, ROTA(60), INACT(6)
DIMENSION ICX1(4), ICY1(4), ICY2(4), JTYPE(4), ICY3(4), ICY4(4), TPX(4), TPY(4), FD1(300), SD1(6)

TITILE='HANUI/HANDL/FIELD-WRITE DATA FILE'
CALL FRAME(TITLE)
CALL SCREEN
CALL KM0VADICX2(1), CY2(1))
DO I=K-1,LAST
CALL KDRWA2(CX2(I), CY2(I))
CONTINUE
WR=.FALSE.
CALL KMOVAD(-200..1000.)
CALL KTEXT(STRING,3) SWEEP/2.-SWEEP/2
3 FORMAT(* BETWEEN*,F10.4,* AND *,F10.4)
CALL KLINFD(2)
CALL KTEXT(STRING,0)
CALL KREADD(1,VARI,2,PI)
IF(JTYPE(1).EQ.,PI) THEN
LENGTH(1)=VARI/1000.
ELSE
THETA(1)=VARI*PI/180.
ENDIF
CALL KLINFD(3)
CALL KTEXT(STRING,0)
CALL KREADD(2,VAR2,2,PI)
ELSE
THETA(2)=VAR2*PI/180.
ENDIF
CALL KTEXT(STRING,0)
CALL KREADD(3,VAR3,2,PI)
ELSE
THETA(3)=VAR3*PI/180.
ENDIF
CALL KTEXT(STRING,0)
CALL KREADD(4,VAR4,2,PI)
ELSE
THETA(4)=VAR4*PI/180.
ENDIF
CALL KTEXT(STRING,0)
IF(JTYPE(2).EQ.,PI) THEN
N2=1
D2=1.*RTH(1)
ELSE
VAR2=THS(1)*PI/180.
VAR3=THS(2)*PI/180.
N2=INT((THB(1)-THS(1))/6.)
E2=MOD(THB(1)-THS(1),6.)
ELSE
N2=VAR2*180./PI
ENDIF
ELSE
   N2=N2+2
ENDIF
D2=6.*PI/180.
ENDIF
IF(JTYPE(3).EQ.*P*) THEN
   N3=1
   D3=1.RTH(2)
ELSE
   VAR3S=THS(2)*PI/180.
   VAR3B=THU(2)*PI/180.
   IF(THS(2)*GE.0.) THEN
      N3=JINT(THB(2)-THS(2)/6.)
      E3=MOD(THB(2)-THS(2),6.)
      IF(E3.LT.2.) THEN
         N3=N3+1
      ELSE
         N3=N3+2
      ENDIF
   ELSE
      N3=N3+2
   ENDIF
ENDIF
DO THE CONFIGURATION WHEN JOINT 3 POSITIVE
C IF(THB(2),GE.0.) THEN
   OPEN(UNIT=3,FILE=NAME,TYPE='NEW')
   WRITE(3,30) MATER
   WRITE(3,31) MODEL
   WRITE(3,32) HLENCT(K),OFFSET(K),THETA(K),ALPHA(K)
   WRITE(3,33) H2,N3,VARI
   WRITE(3,34) N2,N3,VARI
   FORMAT(215,10.5)
   DO 10 I=1,N3
      IF(JTYPE(3),EQ.*P*) THEN
         A(2)=THMNN(2)-(J-1)*D3
         OFFSET(3)=A(2)/1000.
         ELSE
            IF(THS(2),GE.0.) THEN
               TH(2)=VAR3S-(J-1)*D3
               THETA(3)=TH(2)
            ELSE
               TH(2)=(J-1)*D3
               THETA(3)=TH(2)
            ENDIF
            IF(TH(2),GT.VAR3B) THEN
               TH(2)=VAR3B
               THETA(3)=VAR3B
ELSE
ENDIF
ENDIF

C BEGIN THETA(2) LOOP

DO 20 I=1,N2
IF (TYPE(3), EQ, "P") THEN
  A(I) = THMIN(I) * (1 - 1) * D2
  OFFSET(2) = A(2) / 1000.
ELSE
  TH(1) = VAR2S * (1 - 1) * D2
  THETA(2) = TH(1)
  IF (TH(I) .GT. VAR2B) THEN
    TH(1) = VAR2B
    THETA(2) = VAR2B
  ELSE
    THETA(2) = VAR2B
  ENDIF
ENDIF
CALL CALCU1(ROX, ROY, RO, AFA)

C DOING MANIPULABILITY CALCULATION

CALL TRSMAT
CALL JACOBNCALL MABLTYCM.WR, KF, ABLITY.SIG)
CALL KPHTAB(FOX(2), ROY(2))
X = ROX(I) / 1000.
V = ROY(I) / 1000.
WRITE(3, 50) X, V, SIG
50 FORMAT(2F8.5, 6I10.6)
20 CONTINUE
CONTINUE
10 CONTINUE
CLOSE(UNIT=3)
CALL KMOVAC(-210C, 0)
WRITE(STRING, 11) FNAME
11 FORMAT(4A) MODEL
40 FORMAT(20A)
WRITE(3, 41) MODEL
41 FORMAT(10A)
DO 42 K=1,6
WRITE(3, 43) HLEN(K), OFFSET(K), THETA(K), ALPHA(K)
43 FORMAT(6F10.7)
42 CONTINUE
WRITE(3, 44) N2, NJ, VAR1
44 FORMAT(2I5, F10.5)
IF (TYPE(3), EQ, "P") THEN
  N3 = I1
  D3 = -1 * RTH(2)
ELSE
  N3 = INT(6 - THS(2) / 6.)
E3 = MOD(1 - TH1(2), 6.)
IF(E3.LT.2.) THEN
  N3 = N3 + 1
ELSE
  N3 = N3 + 2
ENDIF
D3 = 6.*P1/180.
ENDIF
DO 15 J = 1,N3
  IF(ITYPE(3).EQ."P") THEN
    A(2) = (J-1)*D3
    OFFSET(3) = A(2)/180.
  ELSEIF(TH(2).LE.0.) THEN
    TH(2) = VAR3B*(J-1)*D3
    THETA(3) = TH(2)
  ELSE
    TH(2) = (J-1)*D3
    THETA(3) = TH(2)
  ENDF
  IF(IT(TH(2).LT.VAR3S)) THEN
    TH(2) = VAR3S
    THETA(3) = VAR3S
  ELSE
  ENDF
ENDIF
BEGIN THETA(2) LOOP
DO 25 I = 1,N2
  IF(ITYPE(2).EQ."P") THEN
    A(1) = THIN(I)+(I-1)*D2
    OFFSET(2) = A(1)/180.
  ELSE
    TH(I) = VAR2S*(I-1)*D2
    THETA(2) = TH(I)
    IF(TH(I).GT.VAR2B) THEN
      TH(I) = VAR2B
      THETA(2) = VAR2B
    ELSE
  ENDF
  ENDIF
ENDIF
CALL CALCUL(ROX,ROY,PO,AFA)
BEGIN MANIPULABILITY CALCULATION
CALL TRSMAT
CALL JACOB
CALL MABLTY(M,WK,KRAN,ABLITY,SIG)
CALL KPINTAS(ROX(2),ROY(2))
X = ROX(2)/180.
Y = ROY(2)/180.
WRITE(13,55) X,Y,SIG
55 FORMAT(2F8.5,6F10.6)
CONTINUE
CLOSE(UNIT=3)
CALL KMOVAB(-2100.,-500.)
WRITE(STRING,13) FILE
13 FORMAT(10A.,"WRITTEN")
CALL KTEXT(STRING.2)
ELSE
ENDIF
CALL KMOVA3(-2100.,-1000.)
CALL KTEXT('PUSH <CR> TO CONTINUE',0)
CALL KPAUSE(0,0)
CALL MENU1
RETURN
END
APPENDIX B

List of program "DSURF.FOR"
This is a program to generate workspace of a manipulator with 3 links in the generating plane by two methods. Similar to the program "WORKSP", this program accepts the same joint parameters and generates the workspace using aspect mapping method. Therefore, this program should be linked with the library "WORKSP.OLB" created by the program "WORKSP.FOR". All the variables used here are the same as corresponding block /COMMON/ in program "WORKSP.FOR".

C ******************************************************************************
C PROGRAM DSURF
LOGICAL RF, IW, COMP, ROTA, INACT
CHARACTER*16 MODEL
CHARACTER*20 MAKER
CHARACTER*35 TITLE
CHARACTER JTYPE, COMMON /ANGLE/SWEEP, THMIN(4), THMAX(4), THS(4), THB(4)
COMMON /XY/CX1(999), CX2(999), CX3(999), CY1(999), CY2(999), CY3(999),
I TPX(999), TPY(999), FDIR(999), BD(300)
COMMON /SUM/PCGO), PTAGO), PHY10), SH100, SHI100)
COMMON /1TG/LAST, MM, LOOP(60), NSE, IANG, LL, IP, IDIR, ICOUNT(30), JJ
COMMON /CHR/MODEL, MAKER, JTYPE(6)
COMMON /LOG/RF, IW, COMP, ROTA(60), INACT(6)
CALL KINITZI5, 00)
TITLE*'MAIN PROGRAM'*
PI=4.*ATAH(1. )
100 CALL KINPUT
300 IP=IP+1
ICOUNT(I)=1
LL=1
MM=1
RF=.FALSE.
IF(BL(3), EO.0.) THEN
IANG=0
CALL SEQUE2(*170, *200)
ELSE
CALL PRISEQ(*170, *200)
ENDIF
170 IP=IP+1
LOOP(MM)=LAST
LAST=LAST-1
ICOUNT(IP)=LAST
NSE=MM-1
MAX=LAST -
150 CALL FRAMEI(TITLE)
CALL SCREEN
CALL SELEC
110 CALL KLOCAT(X, Y, LL)
IF(I.L.NE. ' ') GO TO 110
IF(Y.YT.528, .AND. Y.GT.475.) THEN
GOTO 150
ELSE IF(Y.YT.565, .AND. Y.GT.528.) THEN
GOTO 150
ELSE IF(Y.YT.602, .AND. Y.GT.565.) THEN
GOTO 130
ELSE IF(Y.YT.639, .AND. Y.GT.602.) THEN
GOTO 180
ELSE IF(Y.LE.676. .AND. Y.GT.639.) THEN
GOTO 160
ELSE IF(Y.LE.713. .AND. Y.GT.676.) THEN
GOTO 180
ELSE IF(Y.GT.713.) THEN
GOTO 120
ELSE
GOTO 120
ENDIF
130 CALL BOUND
GO TO 110
160 CALL ANLINK(MAX)
GO TO 110
200 CALL KTXSIZ(0.925)
CALL KMOVAS(20.,100.)
CALL KTEXT('WARNING! CHECK ROTATING SEQUENCE!',0)
CALL KPAUSE(-1.0)
CALL KTEXT('EXIT',0)
GO TO 170
180 CALL SURF
GOTO 110
120 CONTINUE
CALL KFINISH
STOP
END

SUBROUTINE SELEC1
CALL KVWPRY(0.,1.,0.,76171875)
CALL KVIND(0.,1024.,0.,780.)
DO 10 I=1,6
CALL KMOVAS(0.,750.,-37.,1)
CALL KDRWAS(304.,750.,-37.,1)
10 CONTINUE
CALL KMOVAS(0.,475.)
CALL KDPVAS(304.,475.)
CALL KTXSIZ(0.022)
CALL KMOVAS(10.,720.)
CALL KTEXT('EXIT',0)
CALL KMOVAS(10.,683.)
CALL KTEXT('REDESIGN',0)
CALL KMOVAS(10.,646.)
CALL KTEXT('DRAW LINK',0)
CALL KMOVAS(10.,609.)
CALL KTEXT('SURFACES',0)
CALL KMOVAS(10.,573.)
CALL KTEXT('DRAW BOUNDARY',0)
CALL KMOVAS(10.,535.)
CALL KTEXT('NEW PAGE',0)
CALL KMOVAS(10.,505.)
CALL KTEXT('NEW PAGE',0)
CALL KUPDAT
RETURN
END
SUBROUTINE SURFACE

This subroutine gets the data from program WORKSP by block
common and plots the three families of hyper-surfaces of a
manipulator in Cartesian space.

THF(I): FIXED ANGLE OF JOINT I AT INCORPORATION I

AN*(I,J): THE CORRESPONDING ANGLE OF JOINT I FOR CONDITION J

CUBE SURFACE

EXTERNAL ANOM

CHARACTER*20 MAKER

CHARACTER JTYPE MODEL

COMMON /ANGLE/SWEEP, THMIN(I), THMAX(I), THO(I), TH1(I), TH2(I), TH3(I)

COMMON /XY/CX1(I), CX2(I), CX3(I), CY1(I), CY2(I), CY3(I)

COMMON /ITG/LAST, MH, LOOP(6), NSE, ANG, LL(I), IP, IDIR, 1COUNT(30), JJ

DIMENSION ROX(I), ROY(I), ROZ(I), AFA(I), AN(I)

CUBE SURFACE

DIMENSION AN(I,1), AN(I,2)

SINGULARITY 1: US2*VS23-0

1. FIXED THETA 2 AND FIND THETA 3

DO 800 I=1,121

SI=1

THF2(I)=THMIN(I)+SI-1.)/360.*PI

IF(THF2(I).GT.THMAX(I)) THEN

SI=-1

GOTO 801

ELSE

V1=V*SIN(THF2(I))

V2=V*COS(THF2(I))

U1=U*SIN(THF2(I))

UVU=U-V1

SV=V*V*V1-U1

IF(SV.LT.0.) THEN

AN3(I,1)=1000.

AN3(I,2)=1000.

ELSE IF(UVU.ED.0.) THEN

IF(I.NE.1) THEN

IF(AN3(I,1).LT.0.) THEN

AN3(I,1)=-PI

AN3(I,2)=PI

ELSE

AN3(I,1)=PI

AN3(I,2)=-PI

ENDIF

ELSE

AN3(I,1)=PI

AN3(I,2)=-PI

ENDIF

ELSE

TEMP1=SQRT(SV)

TEMP2=(-V2-TEMP1)/UVU

TEMP1=(-V2-TEMP1)/UVU

AN3(I,1)=2.*ATAN(TEMP1)

AN3(I,2)=2.*ATAN(TEMP2)

ENDIF
255

```fortran
800  CONTINUE
C
C  2. FIX THETA3 AND CALCULATE THETA2
C
801  DO 700 1=1,121
  s1 = 1
  thf3(1)=THMIN(3)+(s1-1)*PI/60. 
  IF(thf3(1).GT.THMAX(3)) THEN 
    n2=-1 
    GOTO 701
  ELSE 
    VI = V*COS(THF3(1))
    V2 = V*SIN(THF3(1))
    IF(V1.EQ.-V1) THEN 
      IF(AN2(1,1).GT.Pi/2.) THEN 
        AN2(1,1)=Pi/2.
      ELSE 
        AN2(1,1)=Pi/2 
      ENDIF
    ELSE 
      AN2(1,1)=ATAN(-V2/(U*V1))
      ENDIF
    IF(AN2(1,1).LT.Pi) THEN 
      AN2(1,2)=AN2(1,1)-Pi
    ELSE 
      AN2(1,2)=AN2(1,1)+Pi
    ENDIF
  endif
  CONTINUE
C
C SINGULARITY 2: S23+US3=0
C
700  CONTINUE
C
701  DO 600 1=1,n1
  VI = V1=1
  V2 = V2=1
  IF(V1.EQ.-V2) THEN 
    IF(1.GE.2) THEN 
      IF(AN3(1,1).GT.Pi/2.) THEN 
        AN3(1,1)=Pi/2.
      ELSE 
        AN3(1,1)=Pi/2 
      ENDIF
    ELSE 
      AN3(1,1)=ATAN(-V1/(U+V1))
      ENDIF
    IF(AN3(1,1).LT.Pi) THEN 
      AN3(1,2)=AN3(1,1)-Pi
    ELSE 
      AN3(1,2)=AN3(1,1)+Pi
    ENDIF
  endif
  CONTINUE
```

END
2. FIX THETA3 AND FIND THETA2

DO 500 I=1,n2
V1=COS(THF3(I))
V2=SIN(THF3(I))
U1=U*V2
UMV=U1-V2
SV1=-U1*U1
IF(SV1.LT.0.) THEN
AN2(I,3)=100.
AN2(I,4)=100.
ELSE IF(UMV.EQ.0.) THEN
IF(I.NE.1) THEN
IF(AN2(I-1,3).LT.0.) THEN
AN2(I,3)=-PI
AN2(I,4)=PI
ELSE
AN2(I,3)=PI
AN2(I,4)=-PI
ENDIF
ELSE
AN2(I,3)=PI
AN2(I,4)=-PI
ENDIF
ELSE TEMP1=SORT(SV1)
TEMP2=(-V1+TEMP1)/UMV
THMIN2=2.*ATAN(TEMP2)
AN2(I,3)=THMIN2
AN2(I,4)=THMIN2
TH2=THMIN2
TH3=THF3(I)
CALL CALCUL(rox,roy,ro,afa)
CALL KPNTAB(rox(4),roy(4))
ENDIF
IF (AN2(I+1,3).LT.THMIN2 .OR. AN2(I,k).GT.THM(2)) THEN
ELSE
TH1=THMIN2
TH2=AN2(I,k)
TH3=THF3(I)
CALL CALCUL(rox,roy,ro,afa)
CALL KPNTAB(rox(4),roy(4))
ENDIF
IF (AN2(I+1,3).LT.THMIN2 .OR. AN2(I,k+1).GT.THM(2)) THEN
ELSE
TH1=THMIN2
TH2=AN2(I+1,k+1)
TH3=THF3(I)
CALL CALCUL(rox,roy,ro,afa)
CALL KPNTAB(rox(4),roy(4))
ENDIF
IF (AN3(I+1,3).LT.THMIN3 .OR. AN3(I,k).GT.THM(3)) THEN
ELSE
TH1=THMIN3
TH2=THF2(I)
C
C PLOTTING THE SINGULAR POINTS
C
CALL KVUPRT(0.,0.0292,0.7324)
CALL KWINDO(-2100.,1100.,-1150.,1100.)
do 990 k=1,3,2
DO 910 I=1,111
IF (AN2(I,k).LT.THMIN(2) .OR. AN2(I,k).GT.THM(2)) THEN
ELSE
TH1=THMIN(2)
TH2=AN2(I,k)
TH3=THF3(I)
CALL CALCUL(rox,roy,ro,afa)
CALL KPNTAB(rox(4),roy(4))
ENDIF
IF (AN2(I,k+1).LT.THMIN(2) .OR. AN2(I,k).GT.THM(2)) THEN
ELSE
TH1=THMIN(2)
TH2=AN2(I,k+1)
TH3=THF3(I)
CALL CALCUL(rox,roy,ro,afa)
CALL KPNTAB(rox(4),roy(4))
ENDIF
IF (AN3(I,k).LT.THMIN(3) .OR. AN3(I,k).GT.THM(3)) THEN
ELSE
TH1=THMIN(3)
TH2=THF2(I)
C
TH(3)=AN3(1,k)
call calcui(rox,roy,ro,a)
call kpntab(rox(4),roy(4))
endif
if (AN3(1,k+1).lt.thi(3) .or. an3(1,k+1).gt.th(3)) then
else
TH(1)=THI(1)
TH(2)=THI(2)
TH(3)=AN3(1,k+1)
call calcui(rox,roy,ro,a)
call kpntab(rox(4),roy(4))
endif
if (AN2(1,k).lt.thi(2) .or. AN2(1,k).gt.th(2)) then
else
TH(1)=THMAX(1)
TH(2)=AN2(1,k)
TH(3)=THF2
CALL CALCUI(ROX,ROY,RO,FA)
call kpntab(rox(4),roy(4))
endif
if (AN2(1,k+1).lt.thi(2) .or. AN2(1,k+1).gt.th(2)) then
else
TH(1)=THMAX(1)
TH(2)=AN2(1,k+1)
TH(3)=AN2(1,k)
call calcui(rox,roy,ro,a)
call kpntab(rox(4),roy(4))
endif
if (AN3(1,k).lt.thi(3) .or. AN3(1,k).gt.th(3)) then
else
TH(1)=THMAX(1)
TH(2)=THF3
TH(3)=AN3(1,k+1)
call calcui(rox,roy,ro,a)
call kpntab(rox(4),roy(4))
endif
if (AN3(1,k+1).lt.thi(3) .or. AN3(1,k+1).gt.th(3)) then
else
TH(1)=THMAX(1)
TH(2)=THF3
TH(3)=AN3(1,k)
call calcui(rox,roy,ro,a)
call kpntab(rox(4),roy(4))
endif
continue
100 do s10=1,121
TH(1)=THI(1)+(S1-1).*(THMAX(1)-THMIN(1))/120.
if (AN2(1,k).lt.thi(2) .or. AN2(1,k).gt.th(2)) then
else
TH(3)=THF2
TH(2)=AN2(1,k)
call calcui(rox,roy,ro,a)
call kpntab(rox(4),roy(4))
endif
if (AN2(1,k+1).lt.thi(2) .or. AN2(1,k+1).gt.th(2)) then
else
TH(3)=THF3
TH(2)=AN2(1,k+1)
CALL CALCUL(ro,roy,ro,a)
CALL KPNTAB(ro,roy)
ENDIF
C
IF(AN3(1,K),LT,THMIN(3),OR,AN3(1,K),GT,THMAX(3)) THEN
ELSE
TH(3)=AN3(1,K)
TH(2)=THF2(1)
CALL CALCUL(ro,roy,ro,a)
CALL KPNTAB(ro,roy)
ENDIF
C
IF(AN3(1,K+1),LT,THMIN(3),OR,AN3(1,K+1),GT,THMAX(3)) THEN
ELSE
TH(3)=THF2(1)
TH(3)=AN3(1,K+1)
CALL CALCUL(ro,roy,ro,a)
CALL KPNTAB(ro,roy)
ENDIF
C
IF(AN3(121,K),LT,THMIN(3),OR,AN3(121,K),GT,THMAX(3)) THEN
ELSE
TH(3)=THF2(121)
TH(3)=AN3(121,K)
CALL CALCUL(ro,roy,ro,a)
CALL KPNTAB(ro,roy)
ENDIF
C
IF(AN3(121,K+1),LT,THMIN(3),OR,AN3(121,K+1),GT,THMAX(3)) THEN
ELSE
TH(3)=THF3(1)
TH(3)=AN3(121,K+1)
CALL CALCUL(ro,roy,ro,a)
CALL KPNTAB(ro,roy)
ENDIF
C
IF(AN(121,K),LT,THMIN(2),OR,AN(121,K),GT,THMAX(2)) THEN
ELSE
TH(2)=AN2(121,K)
TH(3)=THF3(121)
CALL CALCUL(ro,roy,ro,a)
CALL KPNTAB(ro,roy)
ENDIF
C
IF(AN2(121,K+1),LT,THMIN(2),OR,AN2(121,K+1),GT,THMAX(2)) THEN
ELSE
TH(2)=AN2(121,K+1)
TH(3)=THF3(121)
CALL CALCUL(ro,roy,ro,a)
CALL KPNTAB(ro,roy)
ENDIF
B10 CONTINUE
920 CONTINUE
C
DO 710 I=1,121
SI=1
TH(2)=THMIN(2)*((SI-1.)*(THMAX(2)-THMIN(2))/120)
TH(1)=THMIN(1)
TH(3)=THMIN(3)
CALL CALCUL(ro,roy,ro,a)
CALL KPNTAB(rox(4), roy(4))

C
TH(3) = THMAX(3)
CALL CALCUL(rox, roy, ro, afa)
CALL KPNTAB(rox(4), roy(4))
C
TH(1) = THMAX(1)
CALL CALCUL(rox, roy, ro, afa)
CALL KPNTAB(rox(4), roy(4))
C
TH(3) = THMIN(3)
CALL CALCUL(rox, roy, ro, afa)
CALL KPNTAB(rox(4), roy(4))
C
IF(THMIN(3), LE, 0, .AND. THMAX(3), GE, 0.) THEN
   TH(3) = 0.
   CALL CALCUL(rox, roy, ro, afa)
   CALL KPNTAB(rox(4), roy(4))
   ELSE
   ENDIF
   CONTINUE
   DO 720 I = 1, 121
      SI = I
      TH(1) = THMIN(1) + (SI - 1.)*(THMAX(1) - THMIN(1))/120
      TH(2) = THMIN(2)
      TH(3) = THMIN(3)
      CALL CALCUL(rox, roy, ro, afa)
      CALL KPNTAB(rox(4), roy(4))
C
TH(3) = THMAX(3)
CALL CALCUL(rox, roy, ro, afa)
CALL KPNTAB(rox(4), roy(4))
C
TH(2) = THMAX(2)
CALL CALCUL(rox, roy, ro, afa)
CALL KPNTAB(rox(4), roy(4))
C
TH(3) = THMIN(3)
CALL CALCUL(rox, roy, ro, afa)
CALL KPNTAB(rox(4), roy(4))
C
IF(THMIN(3), LE, 0, .AND. THMAX(3), GE, 0.) THEN
   TH(3) = 0.
   CALL CALCUL(rox, roy, ro, afa)
   CALL KPNTAB(rox(4), roy(4))
   ELSE
   ENDIF
   CONTINUE
   CALL KVWPRT(0., 1., 0., 0., 76171875)
   CALL KWINDO(0., 1824., 0., 785.)
RETURN
END
APPENDIX C

List of program "CONT3.FOR"
program Cont3
this program reads a data file for x,y coordinates of a set of points and the function values of those points. The x,y coordinates of all points will be kept so that the shape of the object will not change.

The first three lines are dummies.
The fourth line gives number of element in x direction NX, number of element in Y direction NY, the z coordinate of the x-y plane VAR1, and the ranges of x and y.
The program then reads arrays of x,y and 6 function values and divide the arrays into 2-dimension shapes by given information. i.e. this program can handle any shape as long as the shape can be sub-divided into 2 coordinates.

Then the two-dimensional shape of object is divided into elements of four corners, each corner has its own coordinates. The elements is further divided into 4 triangles with each vertex located at the mass center of the 4 corner element.

A contour line is then drawn in each triangle if there exist such a value by linear interpolation.

A contour line is drawn in each triangle if there exists such a value by linear interpolation.

coordinates of each corner

function value at each corner of element

the desired contour line value

coordinates of 4 corners of element

the coordinates of each corner

coordinates of 4 corners of element

the ZOOM ratio for each coordinate in order to make a readable plot in screen.

the minimum height of contour plot.

---

PROGRAM CONT3
CHARACTER*50 MAKER
CHARACTER*10 MODEL, FNAME
DOUBLE PRECISION EP(3,1),F0
DIMENSION XI(1505), Y(1505), Z(1505), F(4), EP(3,4)
DIMENSION SIG(6), HLENGTH(6), OFFSET(6), THETA(6), ALPHA(6), VALUE(15)
WRITE(6,1008)
1008 FORMAT(‘ENTER DATA FILE NAME:,A) READ(5,1009) FNAME
1009 FORMAT(10A) OPEN(UNIT=2, NAME=FNAME, TYPE=‘OLD’) READ(3,1002) MAKER
1002 FORMAT(2A) READ(3,1003) HLENGTH(J), OFFSET(J), THETA(J), ALPHA(J)
1003 FORMAT(6F10.6) CONTINUE
1004 FORMAT(10A) NX, NY, VAR1, THMIN, THMAX, TH2MIN, TH2MAX
1005 FORMAT(215,6F10.5)
1006 IF(TH2MIN.LT.0.) TH2MIN=0.
1007 RANG2=TH2MAX-TH2MIN
N=1
XMAX=-1000.  
XMIN=1000.  
YMAX=1000.  
YM1N=1000.  
ZMAX=10.  
ZMIN=100.  

READ(3,1005,END=1006,ERR=1026) X(N),Y(N),(SIGIKI,K=1,6)  
1005
FORMAT(2F8.4,6F10.6)  
N2=M00D(N,NX)  
N3=N/NX+1  
IF(N2.EQ.D) THEN  
N2=NX  
N3=N3-1  
ELSE  
ENDIF  
VAR2=RANG1/(NX-1)*(N2-1)  
VAR3=RANG2/(NY-1)*(N3-1)  
TEMP1=ABS(VAR2*(RANG1-VAR2))/RANG1**2  
TEMP2=ABS(VAR3*(RANG2-VAR3))/RANG2**2  
CONST1=1.-EXP(-100.*TEMP1*TEMP2)  
Z(N)*SIG(6)/SIG(1)*CONST1  
IF(XMIN.LT.X(N)) XMIN=X(N)  
IF(YMIN.LT.Y(N)) YMIN=Y(N)  
IF(ZMIN.LT.Z(N)) ZMIN=Z(N)  
N=N+1  
GOTO 66  
1006
CLOSE(UNIT=3)  
C
NORMALIZE THE THREE COORDINATIONS  
C  
DEFX=XMAX-XMIN  
DEFY=YMAX-YMIN  
DEFZ=ZMAX-ZMIN  
IF(DEFZ.EQ.0.) THEN  
WRITE(5,999)  
FORMAT: 'ERROR! YOU GOT constant FUNCTION VALUES')  
999
GOTO 998  
ELSE IF(DEFZ.LT.0.25) THEN  
RATIOZ=0.25/DEFZ  
ELSE  
RATIOZ=1.  
ENDIF  
IF(DEFX.GT.DEFY) THEN  
RATIO=1./DEFX  
ELSE  
RATIO=1./DEFY  
ENDIF  
HELFX=(XMIN+XMAX)/2.*RATIO  
HELFW=(YMIN+YMAX)/2.*RATIO  
IF(ZMAX*RATIOZ.GT.1.) THEN  
SK1RT=ZMAX*RATIOZ-1.  
ELSE IF(ZMIN*RATIOZ.LT.0.2) THEN  
SK1RT=ZMIN*RATIOZ-.2  
ELSE  
SK1RT=0.  
ENDIF  
DO 100 I=1,N
\[\begin{align*}
X(I) &= X(I) \times \text{RATIO-HELFX} \\
Y(I) &= Y(I) \times \text{RATIO-HELXY} \\
Z(I) &= Z(I) \times \text{RATIOZ-.3-SKIRT}
\end{align*}\]

```
100 CONTINUE
   DO 12 I = 1, 4
      EP(3, I) = 0.
   CONTINUE

C INQUIRE NUMBER OF CONTOURS TO BE PLOTTED
   WRITE(6, 34) ZMIN, ZMAX
   FORMAT(1X, 'THE FUNCTION VALUE FROM', E12.5, ' TO', E12.5)
   WRITE(6, 33)
   FORMAT(1X, 'ENTER HOW MANY CONTOUR LINES TO BE PLOTTED/*
   1   ENTER 0 TO PLOT SPECIFIC CONTOUR VALUE/*$)', 5X)
   READ(6, *) NLINE
   ID = 0
   IF(NLINE .EQ. 0) THEN
      ID = 1
      WRITE(6, 35)
      FORMAT(1X, 'ENTER AT MOST 15 VALUES, SEPARATED BY *:', 5X, ')$)
      READ(6, *, ERR=37) (VALUE(K), K = 1, 15)
      DO 36 I = 1, 15
         IF(VALUE(I) .GT. 0.) THEN
            NLINE = NLINE + 1
         ELSE
            GO TO 40
         ENDIF
      CONTINUE
   ELSE
      GO TO 40
   ENDIF

C INITIALIZE THE PLOTTING DEVICE AND DRIVE. THEN SET UP THE TERMINAL
   CALL OBEGIN
   CALL JBEGIN
   CALL JDINIT(I)
   CALL JDEVON(I)
   CALL JASPEK(1, ASP)
   CALL JVSPAC(-1., 1., -ASP, ASP)
   CALL JPORT(-1., 1., -ASP, ASP)
   CALL JVINDO(-1.5, 1.5, -ASP*1.5, ASP*1.5)
   CALL JWCLIP(1., TRUE.)
   CALL JOPEN

C DRAW WORKSPACE BASE BOUNDARY
   CALL MOVE(X(I), Y(I), -0.3)
   DO 23 I = 1, 2, NX
      CALL DRAW(X(I), Y(I), -0.3)
   CONTINUE
   DO 24 J = 1, 2, NY
      J = NX + 1
      CALL DRAW(X(I), Y(J), -0.3)
   CONTINUE
   J = NX
   K = NY - NX + 1
   DO 25 I = J, K - 1
      CALL DRAW(X(I), Y(I), -0.3)
   CONTINUE
   J = NX * (NY - 1) + 1
```
DO 26 I=J,1,-NX
   CALL DRAW(X(I),Y(I),-.3)
CONTINUE

26 DRAW THE SKIRT AND UPPER BOUNDARY

CALL MOVE(X(1),Y(1),-.3)
CALL DRAW(X(1),Y(1),Z(1))
DO 43 I=2,NX
   INTV=MOD(I,2)
   IF (INTV.EQ.1) THEN
      CALL DRAW(X(I),Y(I),Z(I))
      CALL DRAW(X(I),Y(I),-.3)
      CALL MOVE(X(I),Y(I),Z(I))
   ELSE
      CALL DRAW(X(I),Y(I),Z(I))
   ENDF
43 CONTINUE

DO 44 J=1,NY
   INTV=MOD(J,2)
   IF (INTV.EQ.1) THEN
      CALL DRAW(X(I),Y(J),Z(I))
      CALL DRAW(X(I),Y(J),-.3)
      CALL MOVE(X(I),Y(J),Z(I))
   ELSE
      CALL DRAW(X(I),Y(J),Z(I))
   ENDF
44 CONTINUE

DO 45 J=NX*NY-NX+1,NX*NY-1
   K=NX*NY-NX+1-J
   DO 46 I=J,NX-1
      INTV=MOD(I,2)
      IF (INTV.EQ.1) THEN
         CALL DRAW(X(I),Y(J),Z(I))
         CALL DRAW(X(I),Y(J),-.3)
         CALL MOVE(X(I),Y(J),Z(I))
      ELSE
         CALL DRAW(X(I),Y(J),Z(I))
      ENDF
46 CONTINUE
45 CONTINUE

DO 30 NUM=1,NLINE
   IF (ID.EQ.1) THEN
      FO=VALUE(NUM)*RATIO=.3-SKIRT
   ELSE
      FO=RATIO*((NUM-1)*(ZMAX-ZMIN)/NLINE-ZMIN)-.3-SKIRT
30 CONTINUE
SUBROUTINE ODCONT(X,F,FO)
C Subroutine to draw contour of value FO through
C a quadrilateral whose corners j=1,2,3,4 have
C coordinates (x,y,z)(j) = (X(1,j),X(2,j),X(3,j)) and
C the value of the function at corner j is F(j)
C ---------------
DIMENSION X(3,4),F(4),Y(3),Z(3)
DIMENSION Y(3,3),H(3),XM(3)
DOUBLE PRECISION X,F,FO,Y,H,XM
C Compute position of midpoint of the quadrilateral
DO 1 I=1,3
XH(I) = (X(I,1)+X(I,2)+X(I,3)+X(I,4))/4.0DB
CONTINUE
C Compute value of the function at the midpoint
FM = (F(I)+F(2)+F(3)+F(4))/4.0DB
C Find coordinates and function values at the vertices of the first triangle in the quadrilateral
DO 2 I=1,3
Y(I,1) = X(I,1)
Y(I,2) = X(I,2)
Y(I,3) = X(I,4)
CONTINUE
2 CONTINUE
H(1) = F(1)
H(2) = F(2)
H(3) = FM
CALL TRCON((Y,H,F0))
C Find coordinates and function values at the vertices of the second triangle in the quadrilateral
DO 3 I=1,3
Y(I,1) = X(I,2)
Y(I,2) = X(I,3)
Y(I,3) = X(I,4)
CONTINUE
3 CONTINUE
H(1) = F(2)
H(2) = F(3)
H(3) = FM
CALL TRCON((Y,H,F0))
C Call the triangle routine to draw the contour in the triangle.
CALL TRCON((Y,H,F0))
C Find coordinates and function values at the vertices of the third triangle in the quadrilateral
DO 4 I=1,3
Y(I,1) = X(I,3)
Y(I,2) = X(I,4)
Y(I,3) = X(I,1)
CONTINUE
4 CONTINUE
H(1) = F(3)
H(2) = F(4)
H(3) = FM
CALL TRCON((Y,H,F0))
C Call the triangle routine to draw the contour in the triangle.
CALL TRCON((Y,H,F0))
C Find coordinates and function values at the vertices of the fourth triangle in the quadrilateral
DO 5 I=1,3
Y(I,1) = X(I,1)
Y(I,2) = X(I,4)
Y(I,3) = X(I,3)
CONTINUE
5 CONTINUE
H(1) = F(1)
H(2) = F(4)
H(3) = FM
CALL TRCON((Y,H,F0))
RETURN
END
C SUBROUTINE TRCON((X,F,F0))
C Subroutine to draw contour of value FO through a triangle whose vertices j=1,2,3 have
C coordinates (x,y,z)(j) = (X(j,1),X(j,2),X(j,3)) and
the value of the function at vertex J is \( F(j) \)

```
DIMENSION X(3,3),F(3)
DIMENSION IO(3),Y1(3),Y2(3)
DOUBLE PRECISION X,F,F0

C Find IO(1) which is the vertex number at which F is maximum
C Find IO(2) which is the vertex number at which F is minimum
C Find IO(3) which is the vertex number at which F is in between

IF (F(IO(2)).GT.F(IO(1))) THEN
    J=IO(2)
    IO(2)=IO(3)
    IO(3)=J
ENDIF

IF (F(IO(3)).GT.F(IO(2))) THEN
    J=IO(2)
    IO(2)=IO(1)
    IO(1)=J
ENDIF

IF (F(IO(1)).GT.F(IO(2))) THEN
    J=IO(2)
    IO(2)=IO(3)
    IO(3)=J
ENDIF

IF (F(IO(1)).GE.F0) .AND. (F0.GE.F(IO(2)))) THEN
    Compute the intersection
    A=(F(IO(1))-F0)/(F(IO(2))-F(IO(1)))
    B=(F(IO(3))-F0)/(F(IO(3))-F(IO(1)))
    DO 1 I=1,3
        Y1(I)=X(I,IO(1))*A*X(I,IO(2))*A-B
        Y2(I)=X(I,IO(3))*B*X(I,IO(3))*B-A
    CONTINUE
    Draw the contour
    This part will have to be changed depending on which graphics package is being used.
    For a two-dimensional contour plot, make the parameter list (Y1(1),Y1(2)),
    for three-dimensional contour plot in which the third direction is the function value, make
    the parameter list (Y1(1),Y1(2),FO).

    CALL MOVE(Y1(1),Y1(2),FO)
    CALL DRAW(Y2(1),Y2(2),FO)

ELSE IF (F(IO(2)).GT.F0) .AND. (F0.GE.F(IO(3)))) THEN
    Compute the intersection
    A=(F(IO(2))-F0)/(F(IO(2))-F(IO(1)))
    B=(F(IO(3))-F0)/(F(IO(3))-F(IO(1)))
    DO 2 I=1,3
        Y1(I)=X(I,IO(3))*A*X(I,IO(2))*A-B
        Y2(I)=X(I,IO(3))*B*X(I,IO(3))*B-A
    CONTINUE
    Draw the contour
    This part will have to be changed depending on
```
which graphics package is being used.

For a two-dimensional contour plot, make the parameter list \((V1(1), V1(2))\).

For three-dimensional contour plots in which the third direction is the function value, make the parameter list \((V1(1), V1(2), F0)\).

```c
CALL MOVE(V1(1), V1(2), F0)
CALL DRAW(Y2(1), Y2(2), F0)
```

```c
ENDIF
RETURN
END
```

```c
SUBROUTINE DRAW(X, Y, Z)

THIS SUBROUTINE ACCEPT THREE DIMENSIONAL COORDINATE OF A POINT
AND DRAW FROM CURRENT POSITION TO THIS POINT AT TWO DIMENSIONAL
COORDINATE BY ISOPARAMETRIC PROJECTION.

\[ \alpha: \text{angle from the horizontal } x_2\text{-axis to } x\text{-axis} \]
\[ \beta: \text{angle from the horizontal } -x_2\text{-axis to } y\text{-axis} \]

\[ \pi = \sqrt{\frac{1}{2}} \]
\[ \alpha = \pi/6 \]
\[ \beta = \alpha \]
\[ x_2 = x \cos(\alpha) - y \cos(\beta) \]
\[ y_2 = x \sin(\alpha) + y \sin(\beta) + Z \]

CALL JDRAW(X2, Y2)
RETURN
END
```

```c
SUBROUTINE MOVE(X, Y, Z)

THIS SUBROUTINE ACCEPT THREE DIMENSIONAL COORDINATE OF A POINT
AND MOVE FROM CURRENT POSITION TO THIS POINT AT TWO DIMENSIONAL
COORDINATE BY ISOPARAMETRIC PROJECTION.

\[ \alpha: \text{angle from the horizontal } x_2\text{-axis to } x\text{-axis} \]
\[ \beta: \text{angle from the horizontal } -x_2\text{-axis to } y\text{-axis} \]

\[ \pi = \sqrt{\frac{1}{2}} \]
\[ \alpha = \pi/6 \]
\[ \beta = \alpha \]
\[ x_2 = x \cos(\alpha) - y \cos(\beta) \]
\[ y_2 = x \sin(\alpha) + y \sin(\beta) + Z \]

CALL JMOVE(X2, Y2)
RETURN
END
```
APPENDIX D

List of program "ASPECT.FOR"
Program ASPECT

This program accepts three link lengths and their joint limits and solves for three conditions for which the 2x2 Jacobian minors equal 0.

Then the three families of hyper-surfaces are plotted in joint space to find the aspects of this manipulator. The plotting is in DL3000 subroutines, and in isoparametric view.

A1: Link length 1
A2: Link length 2
A3: Link length 3

THM(i): Joint limits, 1 = thimin, 2 = thimax etc...

THF(i,1): temporary angle array of angle increments

TH*(i,j): "#11 for angle increment, j#4 for 2 conditions
each condition has two possible solutions

---------------------------------------------------------

PROGRAM ASPECT

DIMENSION TH2(6,4), TH3(6,4), THF(6), THM(6)

9 FORMAT(6,900)
900 FORMAT(*) ENTER LINK LENGTHES!,$)
READ(5,*) A1, A2, A3
WRITE(6,901)
901 FORMAT(*) ENTER JOINT LIMITS!,$)
READ(5,*) THM
U=A2/A1
V=A3/A1
PI=ATAN(1.0)*4.
DO 902 I=1,6
THM(I)=THM(I)/100.*PI
902 CONTINUE

C SINGULARITY 1: VS2*V523=0

1. FIXED THETA 2 AND FIND THETA3

WRITE(1,200)
200 FORMAT(*) singular condition 1, case 1 fix 2 find 3*)
DO 800 I=1,6
SI=I
THF(I)=(SI-3.1)/30.*PI
V1=SIN(THF(I))
V2=V*COS(THF(I))
U1=V*SIN(THF(I))
UW=U-V1
SV=V-U1*U1
IF(SV.LT.0.) THEN
TH3(I,1)=1000.
TH3(I,2)=1000.
ELSE IF(UW.EQ.0.) THEN
IF(I.NE.1) THEN
IF(TH3(I-1,1).LT.0.) THEN
TH3(I,1)=-PI
TH3(I,2)=PI
ELSE
TH3(I,1)=PI
TH3(I,2)=-PI
ENDIF
ELSE
TH3(I,1)=PI
TH3(I,2)=-PI
ENDIF
ENDIF
ELSE
TH3(I,1)=PI

TH3(1,2)=-PI

ENDIF

ELSE

TH3(1,1)=2.*ATAN(TEMP1)
TH3(1,2)=2.*ATAN(TEMP2)

ENDIF

888 CONTINUE

C 2. FIX THETA3 AND CALCULATE THETA2

write(1,202)

format(' singular condition 1. case 2 fix 3 find 2')
DO 700 I=1,61
V1=V*COS(THF(I))
V2=V*SIN(THF(I))
IF(U.EQ.-V1) THEN
  IF(I.ge.2) THEN
    IF(th2(I-1,1),GT.0.) THEN
      TH2(I,1)=PI/2.
    ELSE
      TH2(I,1)=-PI/2.
    ENDIF
    ELSE
      TH2(I,1)=PI/2.
  ENDIF
ELSE
  TH2(I,1)=-ATAN(-V2/(U*V1))
ENDIF
ENDIF IF(TH2(I,1).GT.0.) THEN
TH2(I,2)=TH2(I,1)-PI
ELSE
TH2(I,2)=TH2(I,1)+PI
ENDIF
700 CONTINUE

C SINGULARITY 2: S23+US3=0

C 1: FIX THETA2 AND FIND THETA3

write(1,204)

format(' singular condition 2. case 1 fix 2 find 3')
DO 600 I=1,61
V1=SIN(THF(I))
V2=COS(THF(I))
IF(-U.EQ.V2) THEN
  IF(I.ge.2) THEN
    IF(th3(I-1,1),GT.0.) THEN
      TH3(I,1)=PI/2.
    ELSE
      TH3(I,1)=-PI/2.
    ENDIF
    ELSE
      TH3(I,1)=PI/2.
  ENDIF
ELSE
  TH3(I,1)=ATAN(-V1/(U+V2))
ENDIF
ENDIF IF(TH3(I,1),GT.0.) THEN
272

TH3(1,4) = TH3(1,3) - PI
ELSE
TH3(1,4) = TH3(1,3) + PI
ENDIF

C
C 2. FIX THETA3 AND FIND THETA2
C
WRITE(1,205)
FORMAT('singular condition 2, case 2 fix 3 find 2')
DO 500 I = 1, n
V1 = COS(TH1(I))
V2 = SIN(TH1(I))
U1 = U*V2
UMV = U1 - V2
SV = 1 - U1*U1
IF(SV .LT. 0.) THEN
TH2(I,3) = 1.000.
TH2(I,4) = 1.000.
ELSE IF(UMV .EQ. 0.) THEN
TH2(I,3) = TH2(I,4)
ELSE IF(TH2(I,3) .LT. 0.) THEN
TH2(I,3) = PI
TH2(I,4) = PI
ENDIF
ELSE
TH2(I,3) = -PI
TH2(I,4) = -PI
ENDIF

500 CONTINUE

C INITIALIZE THE PLOTTING DEVICE AND DRIVE, THEN SET UP THE TERMINAL
C
CALL JBEGIN
CALL JDINIT(1)
CALL JDEVON(1)
CALL JASPEK(1, ASP)
CALL JVSPAC(-1.1, -1.1, -ASP, ASP)
CALL JVPORT(-1.1, -1.1, -ASP, ASP)
CALL JWINO(-10., 10., -ASP*10., ASP*10.)

C DRAW THE BOUNDARY FOR JOINT MOTION CUBE
C
CALL JOPEN
CALL MOVE(-PI, -PI, -PI)
call dash(-PI, -PI, -PI)
call draw(-PI, PI, PI)
call draw(-PI, PI, PI)
call draw(PI, PI, PI)
call draw(PI, PI, PI)

C
CALL MOVE(-P1,P1,P1)
call draw(-P1,-P1,-P1)
call dash(-P1,-P1,-P1)
CALL MOVE(P1,P1,P1)
call draw(P1,P1,P1)
call MOVE(P1,P1,P1)
call draw(P1,P1,P1)

DRAW THREE AXES

CALL MOVE(0.,0.,0.)
call draw(0.,0.,0.)
CALL MOVE(0.8,-0.08,0.)
CALL DRAW(0.,0.,0.)
call move(0,0.,0.)
call draw(0.,0.,0.)
CALL MOVE(-0.08,0.8,0.)
CALL DRAW(0.,0.,0.)
call move(-0.08,0.8,0.)
call draw(-0.08,0.8,0.)
CALL MOVE(0.,0.,0.)
CALL JDRAW(0.,1.)
CALL JDRAW(0.8,0.8)

DRAW THE JOINT LIMITS

CALL MOVE(thm1),thm3),thm5))
call dash(thm1),thm3),thm5))
call draw(thm1),thm4),thm6))
call draw(thm2),thm4),thm6))
call draw(thm2),thm4),thm6))
call draw(thm2),thm4),thm6))
call dash(thm1),thm3),thm5))
call dash(thm1),thm3),thm5))
call MOVE(thm1),thm3),thm5))
call MOVE(thm2),thm3),thm5))
call MOVE(thm2),thm3),thm5))
call MOVE(thm2),thm3),thm5))
call move(thm1),thm4),thm6))
call draw(thm1),thm4),thm6))
call draw(thm1),thm4),thm6))

DRAW SINGULAR CONDITION 1

DO 300 K=1,3,2
DO 13 I=1,6
CALL point(pl,thf1),th3(I,K))
CONTINUE
DO 14 I=61,1,-1
CALL point(-pl,thf1),th3(I,K))
CONTINUE

DRAW SINGULAR CONDITION 2
DO 16 I=1,61
CALL point(p1,th2(1,K),thf(1))
CONTINUE
DO 16 I=61,1,-1
CALL point(-p1,th2(1,K),thf(1))
CONTINUE

DRAW SINGULAR CONDITION 3

CALL MOVE(p1,p1,0.)
CALL DRAW(p1,-p1,0.)
CALL DASH(-p1,-p1,0.)
CALL DASH(-p1,p1,0.)
CALL DRAW(p1,p1,0.)

DRAW 2ND SINGULAR CONDITION 1

DO 23 I=1,61
CALL point(p1,thf(I),th3(I,K+1))
CONTINUE
DO 24 I=61,1,-1
CALL point(-p1,thf(I),th3(I,K+1))
CONTINUE

DRAW 2ND SINGULAR CONDITION 2

DO 25 I=1,61
CALL point(p1,th2(1,K+1),thf(I))
CONTINUE
DO 26 I=61,1,-1
CALL point(-p1,th2(1,K+1),thf(I))
CONTINUE
388 CONTINUE
CALL JCLOSE
CALL JPAUSE(1)
999 CALL JDEVOF(1)
CALL JEND
STOP
END

SUBROUTINE DRAW(X,Y,Z)
C THIS SUBROUTINE ACCEPT THREE DIMENSIONAL COORDINATE OF A POINT
AND DRAW FROM CURRENT POSITION TO THIS POINT AT TWO DIMENSIONAL
COORDINATE BY ISOPARAMETRIC PROJECTION.

ALPHA: ANGLE FROM THE HORIZONTAL -X2 AXIS TO X AXIS
BETA: ANGLE FORM THE HORIZONTAL -X2 AXIS TO Y AXIS
PI=4.*ATAN(1.)
IF(Y.GT.PI .OR. Y.LT.-PI) RETURN
IF(Z.GT.PI .OR. Z.LT.-PI) RETURN
ALPHA=PI/12.
BETA=PI/4.
X2=-X*COS(ALPHA)+Y*COS(BETA)
Y2=-X*SIN(ALPHA)-Y*SIN(BETA)+Z
CALL JDRAW(X2,Y2)
RETURN
END

SUBROUTINE DASH(X,Y,Z)
C THIS SUBROUTINE ACCEPT THREE DIMENSIONAL COORDINATE OF A POINT
AND DRAW FROM CURRENT POSITION TO THIS POINT AT TWO DIMENSIONAL
COORDINATE BY ISOPARAMETRIC PROJECTION.

\[
\begin{align*}
\text{ALPHA: } & \text{ANGLE FROM THE HORIZONTAL } -X_2 \text{ AXIS TO } X \text{ AXIS} \\
\text{BETA: } & \text{ANGLE FROM THE HORIZONTAL } +X_2 \text{ AXIS TO } Y \text{ AXIS} \\
\phi & = \arctan(1) \\
\text{IF} (Y \text{ GT } \phi , \text{ OR } Y \text{ LT } -\pi) \text{ RETURN} \\
\text{IF} (Z \text{ GT } \phi , \text{ OR } Z \text{ LT } -\pi) \text{ RETURN} \\
\text{ALPHA} & = \frac{\pi}{12} \\
\text{BETA} & = \frac{\pi}{4} \\
X_2 & = -X \cos(A) + Y \cos(B) \\
Y_2 & = Y \sin(A) + Z \\
\text{CALL JLISTYL(1)} \\
\text{CALL JDRAW(X2,Y2)} \\
\text{CALL JLISTYL(0)} \\
\text{RETURN} \\
\end{align*}
\]

END

SUBROUTINE MOVE(X,Y,Z)
THIS SUBROUTINE ACCEPT THREE DIMENSIONAL COORDINATE OF A POINT
AND MOVE FROM CURRENT POSITION TO THIS POINT AT TWO DIMENSIONAL
COORDINATE BY ISOPARAMETRIC PROJECTION.

\[
\begin{align*}
\text{ALPHA: } & \text{ANGLE FROM THE HORIZONTAL } -X_2 \text{ AXIS TO } X \text{ AXIS} \\
\text{BETA: } & \text{ANGLE FROM THE HORIZONTAL } +X_2 \text{ AXIS TO } Y \text{ AXIS} \\
\phi & = \arctan(1) \\
\text{IF} (Y \text{ GT } \phi , \text{ OR } Y \text{ LT } -\pi) \text{ RETURN} \\
\text{IF} (Z \text{ GT } \phi , \text{ OR } Z \text{ LT } -\pi) \text{ RETURN} \\
\text{ALPHA} & = \frac{\pi}{12} \\
\text{BETA} & = \frac{\pi}{4} \\
X_2 & = -X \cos(A) + Y \cos(B) \\
Y_2 & = Y \sin(A) + Z \\
\text{CALL JDRAW(X2,Y2)} \\
\text{RETURN} \\
\end{align*}
\]

END

SUBROUTINE POINT(X,Y,Z)
THIS SUBROUTINE ACCEPT THREE DIMENSIONAL COORDINATE OF A POINT
AND MOVE FROM CURRENT POSITION TO THIS POINT AT TWO DIMENSIONAL
COORDINATE BY ISOPARAMETRIC PROJECTION.

\[
\begin{align*}
\text{ALPHA: } & \text{ANGLE FROM THE HORIZONTAL } -X_2 \text{ AXIS TO } X \text{ AXIS} \\
\text{BETA: } & \text{ANGLE FROM THE HORIZONTAL } +X_2 \text{ AXIS TO } Y \text{ AXIS} \\
\phi & = \arctan(1) \\
\text{IF} (Y \text{ GT } \phi , \text{ OR } Y \text{ LT } -\pi) \text{ RETURN} \\
\text{IF} (Z \text{ GT } \phi , \text{ OR } Z \text{ LT } -\pi) \text{ RETURN} \\
\text{ALPHA} & = \frac{\pi}{12} \\
\text{BETA} & = \frac{\pi}{4} \\
X_2 & = -X \cos(A) + Y \cos(B) \\
Y_2 & = Y \sin(A) + Z \\
\text{CALL JDRAW(X2,Y2)} \\
\text{RETURN} \\
\end{align*}
\]

END