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.

UMI Dissertation Information Service
University Microfilms International
A Bell & Howell Information Company
300 N Zeeb Road, Ann Arbor, Michigan 48106
Langer, Franz Dieter

KINEMATICS AND ROTATIONAL DYNAMICS OF MULTI-LINKAGE SYSTEMS
AND THE CONTROL OF A PLANAR TWO-LINK SYSTEM IN THE AIR

The Ohio State University

Copyright 1986
by
Langer, Franz Dieter

All Rights Reserved
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
KINEMATICS AND ROTATIONAL DYNAMICS OF MULTI-LINKAGE SYSTEMS AND
THE CONTROL OF A PLANAR TWO-LINK SYSTEM IN THE AIR

DISSERTATION

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

by
Franz Dieter Langer

* * * * *

The Ohio State University
1986

Reading Committee:
Professor H. Hemami
Professor R. Moses
Professor S. Yurkovich

Approved by

Advisor
Department of Electrical Engineering
Dedicated to my parents,
Franz and Elisabeth Langer.
ACKNOWLEDGMENTS

I would like to express my appreciation to my adviser, Professor Hooshang Hemami, for his continuous guidance, encouragement and support. I am especially grateful to him for his invaluable assistance and constructive criticism without which this research project could not have been completed.

I would like to express my gratitude to Professor Randy Moses, to Professor Steve Yurkovich, and to Professor Don W. Miller for the careful reading and valuable comments on this dissertation.

I also would like to thank Todd Pienkowski for the coding of the algorithms of the graphics package, which perform database generation for animation.

I would also like to thank Ms. Emily Baird and LuCinda Flores for the excellent typing of this manuscript.

This work was supported in part by the National Science Foundation under Grant Number ECS-820-1240, and in part by the Department of Electrical Engineering, The Ohio State University, Columbus, Ohio 43210.
VITA

June 1, 1956

Born: Leverkusen, Federal Republic of Germany (FRG).

June 1977 - Jan. 1979

Teaching Assistant, Department of Nachrichtentechnik, Fachhochschule Cologne, FRG.

January 1979

Ingenieur (Grad.) Nachrichtentechnik, Fachhochschule Cologne, FRG.

March 1979 - June 1980

Systems Engineer for Avionic Systems at Messerschmitt - Boelkow-Blohm, Munich, FRG.

July 1980 - June 1981

Systems Engineer for Avionic Systems at Elektronik System Gesellschaft, Munich, FRG.

August 1981 - June 1982

Recipient of a Fulbright Scholarship for Graduate Studies at The Ohio State University, Columbus, Ohio, USA.

August 1982 - June 1986

Research and Teaching Assistant, Departments of Electrical Engineering and Mathematics, The Ohio State University.

December 1983

M.S., Department of Electrical Engineering, The Ohio State University.

PUBLICATIONS


FIELDS OF STUDY

Department: Electrical Engineering, The Ohio State University

Major Field: Studies in Control: Professor H. Hemami

Professor R. Scherzinger

Minor Fields: Studies in Mathematics: Professor T. Scheick

Studies in Statistics: Professor T. Notz

Studies in Antennas: Professor R. Rudduck
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ACKNOWLEDGMENTS</th>
<th>ii</th>
</tr>
</thead>
<tbody>
<tr>
<td>VITA</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td></td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td></td>
</tr>
</tbody>
</table>

## CHAPTERS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td></td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td>1.1 Scope of the Dissertation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Organization of the Dissertation</td>
<td>3</td>
</tr>
<tr>
<td>II.</td>
<td></td>
</tr>
<tr>
<td>SURVEYS OF PREVIOUS WORK</td>
<td></td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Computer Graphics Applied to Movement Studies</td>
<td>4</td>
</tr>
<tr>
<td>2.3 The Dynamics of Multi-Body Systems with Emphasis on Rotational Motion</td>
<td>8</td>
</tr>
<tr>
<td>2.3.1 The Formulation of the Dynamics of Multi-Body Systems</td>
<td>8</td>
</tr>
<tr>
<td>2.3.2 Methods for the Projection of the Equations of Motion of Multi-Body System</td>
<td>12</td>
</tr>
<tr>
<td>2.3.3 The Dynamics of Multi-Body Systems in the Air</td>
<td>14</td>
</tr>
<tr>
<td>2.3.4 The Symbolic Generation of the Equations of Motion of Multi-Body Systems</td>
<td>15</td>
</tr>
<tr>
<td>2.4 Numerical Instabilities Induced by the Imposition of Holonomic and Simple Nonholonomic Constraints</td>
<td>15</td>
</tr>
<tr>
<td>2.5 The Attitude Control of Multi-Body Systems</td>
<td>16</td>
</tr>
<tr>
<td>2.6 Nonlinear Controllability Concepts</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td>17</td>
</tr>
<tr>
<td>2.7 The Automation of the Nonlinear Controllability Analysis</td>
<td>23</td>
</tr>
<tr>
<td>2.8 Summary</td>
<td>24</td>
</tr>
</tbody>
</table>

### III. A GRAPHICS SOFTWARE PACKAGE FOR KINEMATIC STUDIES

<p>| 3.1 Introduction | 25   |
| 3.2 The Hardware Arrangement and Software Structure | 26   |
| 3.2.1 Hardware Arrangement | 26   |
| 3.2.2 Software Structure | 25   |
| 3.3 Basic Graphics Software | 29   |
| 3.3.1 The Three-Dimensional Viewing Process | 29   |
| 3.3.1.1. Normalizing Transformation | 29   |
| 3.3.1.2. Three Dimensional Clipping | 36   |
| 3.4 Hidden Surface Removal | 39   |
| 3.4.1 Hidden Surface Removal for Convex Polyhedra | 41   |
| 3.4.2 Hidden Surface Removal for Concave Polyhedra | 41   |
| 3.4.2.1 Decision Tests | 42   |
| 3.4.2.2. Warnock Algorithm | 43   |
| 3.4.2.3. Weiler Atherton Algorithm | 47   |
| 3.5 Object Manipulation and Data Structure | 54   |
| 3.5.1 Data Structure for Kinematic Manipulation | 54   |
| 3.5.2 Implementation of the Hidden Surface Removal Algorithms and Their Data Structure | 55   |
| 3.5.3 Data Structure for Animation Studies | 57   |
| 3.6 Convex Polyhedra | 57   |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7 Concave Polyhedra</td>
<td>59</td>
</tr>
<tr>
<td>3.7.1 Humanoid: ADAM</td>
<td>59</td>
</tr>
<tr>
<td>3.7.1.1 Program and Data Structures</td>
<td>59</td>
</tr>
<tr>
<td>3.7.1.2 Kinematics</td>
<td>64</td>
</tr>
<tr>
<td>3.7.1.3 Animation</td>
<td>68</td>
</tr>
<tr>
<td>3.8 Summary</td>
<td>68</td>
</tr>
<tr>
<td>IV. THE GENERATION AND THE NONLINEAR CONTROLLARILITY ANALYSIS OF MECHANICAL SYSTEMS USING THE SYMBOLIC ALGEBRAIC MANIPULATION (SAM) LANGUAGE MACSYMA</td>
<td></td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>69</td>
</tr>
<tr>
<td>4.2 The Symbolic Generation of the Rotational Equations of Motion and the Computation of Their First Integrals</td>
<td>70</td>
</tr>
<tr>
<td>4.2.1 The Rotational Equations of Motion for N Rigid Links</td>
<td>70</td>
</tr>
<tr>
<td>4.2.2 The Derivation of the Rotational Equations of Motion Using the Newton Euler Method and a Decoupling Transformation</td>
<td>72</td>
</tr>
<tr>
<td>4.2.3 The Derivation of the Rotational Equations of Motion from the Interconnection Diagram</td>
<td>73</td>
</tr>
<tr>
<td>4.2.4 First Integrals of the Rotational Equations of Motion</td>
<td>77</td>
</tr>
<tr>
<td>4.3 The Automation of the Nonlinear Controllability Analysis</td>
<td>79</td>
</tr>
<tr>
<td>4.4 The Control of the Rotational Equations of Planar Motion</td>
<td>86</td>
</tr>
<tr>
<td>4.4.1 The Controllability Analysis of Two Planar Rigid Links Which are Subject to a Constant Angular Momentum</td>
<td>86</td>
</tr>
<tr>
<td>4.4.2 A Solution to the Inverse Attitude Control Problem of Two Planar, Rigid Links Which are Subject to a Constant Angular Momentum</td>
<td>92</td>
</tr>
<tr>
<td>4.5 Summary</td>
<td>118</td>
</tr>
</tbody>
</table>
V. APPLICATIONS, DERIVATIONS AND COMPUTATIONS OF HOLONOMIC CONSTRAINT FORCES IN MECHANICAL SYSTEMS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Introduction</td>
<td>119</td>
</tr>
<tr>
<td>5.2 Derivation of Constraint Forces</td>
<td>120</td>
</tr>
<tr>
<td>5.3 Computation of Constraint Forces</td>
<td>125</td>
</tr>
<tr>
<td>5.3.1 Computational Complexity</td>
<td>126</td>
</tr>
<tr>
<td>5.3.2 Estimates of Round-Off Errors</td>
<td>127</td>
</tr>
<tr>
<td>5.4 Illustrative Examples</td>
<td>133</td>
</tr>
<tr>
<td>5.4.1 Planar One Link</td>
<td>133</td>
</tr>
<tr>
<td>5.4.2 Planar Double Pendulum</td>
<td>136</td>
</tr>
<tr>
<td>5.4.3 One Link in Three Dimensions</td>
<td>143</td>
</tr>
<tr>
<td>5.5 Conclusions</td>
<td>150</td>
</tr>
</tbody>
</table>

VI. DISCUSSIONS AND CONCLUSIONS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Summary</td>
<td>151</td>
</tr>
<tr>
<td>6.2 Recommendations for Future Research</td>
<td>153</td>
</tr>
</tbody>
</table>
# APPENDIX

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Computer Graphics Procedures</td>
<td>154</td>
</tr>
<tr>
<td>A.1</td>
<td>The Modules of the Basic Graphics Package</td>
<td>154</td>
</tr>
<tr>
<td>A.2</td>
<td>The Modules for Generating the Humanoid, ADAM</td>
<td>176</td>
</tr>
<tr>
<td>A.3</td>
<td>The Modules for Animation</td>
<td>194</td>
</tr>
<tr>
<td>A.4</td>
<td>Help Documents for Using the Graphics Package</td>
<td>216</td>
</tr>
<tr>
<td>B</td>
<td>The Symbolic Generation of the Rotational Dynamics of N Rigid Links in Three-Dimensional Space and the Computation of Their First Integrals</td>
<td>210</td>
</tr>
<tr>
<td>B.1</td>
<td>The Symbolic Projection of the Newton Euler Equations of Motion Using MACSYMA</td>
<td>210</td>
</tr>
<tr>
<td>B.2</td>
<td>A General Expression for the Rotational Equations of Motion and the Symbolic Evaluation for the Case of Three Links (N=3)</td>
<td>224</td>
</tr>
<tr>
<td>B.3</td>
<td>The Derivation of the Hamiltonian Form of the Rotational Equations of Motion</td>
<td>230</td>
</tr>
<tr>
<td>B.4</td>
<td>The Derivation of the Angular Momentum Expression for N Arbitrary Rigid Bodies</td>
<td>233</td>
</tr>
<tr>
<td>B.5</td>
<td>The Derivation of the Angular Momentum Expressions for N Rigid Bodies Subject to Holonomic (Connection) Constraints</td>
<td>238</td>
</tr>
<tr>
<td>C</td>
<td>The Symbolic Generation and the Controllability Analysis of the Rotational Dynamics of Two Rigid Planar Links</td>
<td>245</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>255</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Fig.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>ADAM: A Twenty-three Segment Humanoid</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>The Assembly Robot Consisting of a Robot Arm and a Conveyor Belt</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>The i'th Body of a Kinematic Chain and the Body and the Inertial Coordinate System</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Simulation Diagram of a Constrained Mechanical System</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>Illustration of the Lie Bracket Operation</td>
<td>19</td>
</tr>
<tr>
<td>3.1</td>
<td>Graphics Hardware Arrangement</td>
<td>27</td>
</tr>
<tr>
<td>3.2</td>
<td>Dependence Tree of the Basic Graphics Software Package: GRAPH</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Three-dimensional Viewing Process</td>
<td>30</td>
</tr>
<tr>
<td>3.4</td>
<td>World and Eye Space Coordinate Systems</td>
<td>31</td>
</tr>
<tr>
<td>3.5</td>
<td>Visual Realism by Perspective Projection</td>
<td>34</td>
</tr>
<tr>
<td>3.6</td>
<td>Eye Space Coordinate System with Truncated Viewing Pyramid</td>
<td>38</td>
</tr>
<tr>
<td>3.7</td>
<td>Window Edge Coordinates</td>
<td>38</td>
</tr>
<tr>
<td>3.8</td>
<td>Clipping of a Polyonal Mesh</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>(a) Open Polygon Clipping</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(b) Closed Polygon Clipping</td>
<td></td>
</tr>
<tr>
<td>3.9</td>
<td>Test for Containment of One Polygon in Another Polygon</td>
<td>44</td>
</tr>
<tr>
<td>3.10</td>
<td>Degenerate Case of Edge Intersection</td>
<td>44</td>
</tr>
<tr>
<td>3.11</td>
<td>Flowchart of the Warnock Algorithm</td>
<td>45</td>
</tr>
<tr>
<td>3.12</td>
<td>Flowchart of the Weiler Atherton Algorithm</td>
<td>48</td>
</tr>
<tr>
<td>3.13</td>
<td>Flowchart of the Polygon Clipping Algorithm</td>
<td>49</td>
</tr>
<tr>
<td>3.14a</td>
<td>Polygon Clipping of a Clip Polygon Against a Subject Polygon</td>
<td>52</td>
</tr>
<tr>
<td>3.14b</td>
<td>Generation of the Inside List from the Modified Vertex List Clip and Subject List</td>
<td>52</td>
</tr>
<tr>
<td>Fig.</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.14c</td>
<td>Generation of the Outside List from the Modified Vertex List of Slip and Subject Polygon</td>
<td>53</td>
</tr>
<tr>
<td>3.15</td>
<td>Viewing Pipelines and Object Data Formats</td>
<td>56</td>
</tr>
<tr>
<td>3.16</td>
<td>Convex Polyhedra:</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>(a) Generalized, Truncated Cone</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(b) Sphere</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(c) Generalized Bar</td>
<td></td>
</tr>
<tr>
<td>3.17</td>
<td>Pointer Structures for Body Segment Description</td>
<td>60</td>
</tr>
<tr>
<td>3.18</td>
<td>Diagram of the Fall Structure of ADAM's Segments</td>
<td>61</td>
</tr>
<tr>
<td>3.19</td>
<td>Coordinate Systems of Body Segments and Definition of Position Vectors</td>
<td>62</td>
</tr>
<tr>
<td>3.20</td>
<td>Program Dependencies in Software Model ADAM</td>
<td>63</td>
</tr>
<tr>
<td>3.21</td>
<td>Kinematic Study of a Human Long Jump</td>
<td>65</td>
</tr>
<tr>
<td>4.1</td>
<td>A Collection of N Rigid, Interconnected Links and Their Respective Body Coordinate Systems and the Inertial Coordinate System</td>
<td>71</td>
</tr>
<tr>
<td>4.2</td>
<td>A Three Link Interconnected System with Centers of Mass $CM_i$ ($i = 1, 2, 3$) and Connection-Barycenters $B_i$ ($i = 1, 2, 3$)</td>
<td>74</td>
</tr>
<tr>
<td>4.3</td>
<td>A Planar, One Link System, Which is Hinged to the Origin</td>
<td>84</td>
</tr>
<tr>
<td>4.4</td>
<td>Two Rigid, Planar Links and the Associated Connection Barycenters and Centers of Mass</td>
<td>88</td>
</tr>
<tr>
<td>4.5</td>
<td>Simulation Diagram for the Attitude Control of the Planar, Reduced Two Link System</td>
<td>95</td>
</tr>
<tr>
<td>4.6</td>
<td>The Flight Parabola of a Point Mass which Starts up at the Origin and Lands on an Inclined Plane</td>
<td>96</td>
</tr>
<tr>
<td>4.7</td>
<td>Simulation Results for an Under-Rotated Dive, $H_0 = 2$ ($kg \cdot m^2 \cdot rad/sec$)</td>
<td>99</td>
</tr>
<tr>
<td>4.8</td>
<td>Simulation Results for an Optimal Dive, $H_0 = 24$ ($kg \cdot m^2 \cdot rad/sec$)</td>
<td>103</td>
</tr>
<tr>
<td>Fig.</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.9</td>
<td>Simulation Results for an Over-Rotated Dive, $H_0 = 35$ (kg * m$^2$ * rad/sec)</td>
<td>107</td>
</tr>
<tr>
<td>4.10</td>
<td>Simulation Diagram for the Attitude Control of the Planar, Unreduced Two-Link System</td>
<td>112</td>
</tr>
<tr>
<td>4.11</td>
<td>Simulation Diagram for the Attitude Control of the Planar, Unreduced Two-Link System</td>
<td>114</td>
</tr>
<tr>
<td>4.12</td>
<td>Trajectories for the Optimal Dive Using as State $Z = (\theta, \dot{\theta}_1, \dot{\theta}_2)^T$</td>
<td>115</td>
</tr>
<tr>
<td>5.1</td>
<td>The Simple, Planar Pendulum Subject to Two Holonomic Contact Constraints</td>
<td>121</td>
</tr>
<tr>
<td>5.2</td>
<td>The Pendulum Embedded in $\theta, x, y$ Space</td>
<td>124</td>
</tr>
<tr>
<td>5.3</td>
<td>Comparison of the Operation Count of Smith's, Kane's and Hemami's Method for Computing Constraint Force</td>
<td>131</td>
</tr>
<tr>
<td>5.4</td>
<td>Double Pendulum Subject to Holonomic Constraints</td>
<td>137</td>
</tr>
<tr>
<td>5.5</td>
<td>Double Pendulum Embedded in a Larger State Space</td>
<td>140</td>
</tr>
<tr>
<td>5.6</td>
<td>Three-Dimensional Pendulum with Body and Inertial Coordinate Systems Corresponding to Axes $(X^1_B, X^2_B, X^3_B)$ and $(X^1_I, X^2_I, X^3_I)$</td>
<td>145</td>
</tr>
<tr>
<td>8.1</td>
<td>System of n Rigid Bodies with a Discrete Mass Distribution in an Inertial Coordinate System Which Originates in the Center of Mass C</td>
<td>234</td>
</tr>
<tr>
<td>8.2</td>
<td>Body i with Its Own Body Coordinate System $B_iCS$ in an Inertial Frame</td>
<td>236</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.1</td>
<td>Body Segment Parameters for the Planar, Two-Link System</td>
<td>97</td>
</tr>
<tr>
<td>5.1</td>
<td>Computational Complexity of Calculating an Inverse Matrix and Solving a System of Simultaneous Equations</td>
<td>128</td>
</tr>
<tr>
<td>5.2</td>
<td>Computational Complexity of Calculating Constraint Forces Assuming the Inertia Matrix is Diagonal</td>
<td>129</td>
</tr>
<tr>
<td>5.3</td>
<td>Computational Complexity of Calculating Constraint Forces Assuming the Inertia Matrix is Nondiagonal</td>
<td>130</td>
</tr>
<tr>
<td>5.4</td>
<td>Operation Count for the Methods of Calculating Constraint Forces for the Planar One Link</td>
<td>134</td>
</tr>
<tr>
<td>5.5</td>
<td>Upper Bounds for Round-Off Errors for Computed Constraint Forces of the Planar One Link</td>
<td>135</td>
</tr>
<tr>
<td>5.6</td>
<td>Operation Count for the Methods of Calculating Constraint Forces for the Planar Two Link</td>
<td>138</td>
</tr>
<tr>
<td>5.7</td>
<td>Upper Bounds for Round-Off Errors for Computed Constraint Forces of the Planar Double Pendulum</td>
<td>144</td>
</tr>
<tr>
<td>5.8</td>
<td>Operation Count for Computing Holonomic Constraint Forces of the Three-Dimensional Pendulum</td>
<td>148</td>
</tr>
<tr>
<td>5.9</td>
<td>Operation Count for Computing Holonomic and Simple Nonholonomic Constraint Forces of the Three-Dimensional Pendulum</td>
<td>149</td>
</tr>
<tr>
<td>7.1</td>
<td>Help Document for Plotting a Sphere Using the Graphics Package</td>
<td>207</td>
</tr>
<tr>
<td>7.2</td>
<td>Help Document for Plotting A Cylinder Using the Graphics Package</td>
<td>208</td>
</tr>
<tr>
<td>7.3</td>
<td>Help Document for the Use of the Animation Routines</td>
<td>209</td>
</tr>
<tr>
<td>8.1</td>
<td>Elements of the Coupled Newton Euler Equations of Motion (Equ. 4.1) for N Rigid Links</td>
<td>211</td>
</tr>
<tr>
<td>8.2</td>
<td>Elements of the Decoupling Transformation (Equ. 4.2)</td>
<td>213</td>
</tr>
<tr>
<td>8.3</td>
<td>MACSYMA Program Listing for the Symbolic Projection of an Unconstrained Three-Link System Onto a Reduced State Space</td>
<td>215</td>
</tr>
</tbody>
</table>
### LIST OF TABLES
(continued)

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4</td>
<td>The Unconstrained Equations of Motion and the Decoupling Transform for Three Links in Three-Dimensional Space</td>
</tr>
<tr>
<td>8.5</td>
<td>The Projected Equations of Motion for Three Links in Three-Dimensional Space</td>
</tr>
<tr>
<td>8.6</td>
<td>A General Expression for the Decoupled Equations of Motion for N Rigid Interconnected Bodies (Eqn. 4.3)</td>
</tr>
<tr>
<td>8.7</td>
<td>MACSYMA Program Listing for the Derivation of the Rotational Dynamics for Three Links in Three-Dimensional Space from the Interconnection Diagram</td>
</tr>
<tr>
<td>8.8</td>
<td>The Projected Equations of Rotational Motion for Three Links in Three-Dimensional Space Which Are Derived from the Interconnection Diagram</td>
</tr>
<tr>
<td>8.9</td>
<td>The General Expression for the Angular Momentum $\mathbf{M}_N$ of N Links Which Are Subject to Holonomic Constraints</td>
</tr>
<tr>
<td>8.10</td>
<td>Listing of the Program Which Symbolically Calculates the Angular Momentum of Three Links</td>
</tr>
<tr>
<td>8.11</td>
<td>The Angular Momentum $\mathbf{H}_3$ and Related Quantities of Three Holonomic Links</td>
</tr>
<tr>
<td>9.1</td>
<td>The Symbolic Generation of the Rotational Dynamics of Two Planar, Rigid Links from the Dynamics of the Same System in Three-Dimensional Space</td>
</tr>
<tr>
<td>9.2</td>
<td>Linear and Nonlinear Symbolic Controllability Analysis of Two Planar Links</td>
</tr>
<tr>
<td>9.3</td>
<td>The Involutive Basis Set of Vector Fields for the Determination of the Nonlinear Controllability of the Rotational Dynamics of Two Planar Links</td>
</tr>
<tr>
<td>9.4</td>
<td>The Controllability Matrix of the Linearized Rotational Dynamics of Two Planar Links and Test for Controllability Defects</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

1.1 Scope of the Dissertation

Considerable attention has recently been given to the study of multi-linkage, rigid body systems. These systems can be employed to both describe the dynamics of robotic mechanisms and to approximate some mechanical attributes of human and animal locomotion.

This dissertation primarily deals with the attitude control problem of the above mentioned systems. It is divided into the major areas of the rotational dynamics of multi-linkage systems, the controllability analysis of the equations of rotational motion, and the display of the kinematics of motion of multi-linkage systems. The studies in the above mentioned major areas are done on different types of multi-body systems. The kinematics are studied on the complex model of a human body. The rotational dynamics are derived for the general case of an arbitrary number of rigid links which are connected in an open kinematic chain, and move in three-dimensional space. The controllability of multi-linkage systems is analyzed via a planar, two-link system. Finally, the reduction by projection and the imbedding of multibody systems, which are important concepts in dynamics, are illustrated by one and two link systems which move in two-and three-dimensional space.

The human body is represented by twenty-three body segments, which are made up of wire-basked solids of revolution and are produced by a general purpose graphics program package for the display of multi-body systems. The graphics program accepts the kinematic information on the multi-linkage system as input parameters, solves the associated hidden surface problem and does a transformation from the three-dimensional world space to the two-dimensional image space. The user can input the data, which are the orientations of the individual links, by means of an input file or during an interactive dialogue. The major emphasis of this study is placed on the implementation and optimization of computer graphic related algorithms, on software organization and on data base generation for animation.
The determination and systematic symbolic generation of the dynamics of the multi-linkage systems in the air, and on the ground, two important issues in dynamics, are investigated. The dynamics in the air are decoupled into the translational dynamics of the center of mass (COM) of the composite system and the rotational dynamics of the individual links about the center of mass. Constraints that are due to the angular momentum being constant are included in the formulation.

The controllability of the rotational dynamics of a planar, two link system are analyzed using methods of linear and nonlinear control theory. It is shown that the above mentioned two approaches produce similar results. The inverse attitude control problem is defined for the planar, two-link system and is solved by transforming the corresponding ill-posed boundary value problem into a well-posed one. Digital computer simulations are performed to demonstrate that linear state feedback ensures that the nonlinear two-link system is capable of tracking a prespecified reference trajectory. It is also shown that the above mentioned, partially controllable system does not exhibit numerical instabilities during simulations if the constraint, to which the system is subjected, is a linear function of the state variables of the system. The complexity of the controllability of the multi-linkage systems increases with the rising number of links. Although there is, in principle, no difficulty in performing the nonlinear controllability analysis by hand, the lengthy, time consuming computations, which are involved, become increasingly prone to human error. For this reason algorithms for an automation of the nonlinear controllability analysis are discussed. The study comprises properties of the Lie algebra under a change of the frames of references and under a triangularization of elements of the Lie algebra.

The mechanism of projecting a multi-linkage system to a reduced state space is investigated. This space is characterized by the imposition of holonomic and simple nonholonomic constraint forces or torques. In the multi-linkage systems, which are studied here, holonomic constraints have to be enforced, if the individual bodies of the composite system are jointed together in a kinematic chain. If these connections are such that rotational degrees of freedom are lost, constraint torques are employed, to model these particular joint characteristics. The analysis comprises a comparison of three different algorithms of numerically computing the above type of constraint forces and torques. Several examples demonstrate merits of these different techniques and compare them.
1.2 Organization of the Dissertation

This dissertation consists of six chapters and three appendices. In chapter two the literature is surveyed pertaining to:

- computer graphics applied to movement studies
- dynamics of multi-linkage systems in the air and the symbolic generation of the equations of motion
- mathematical concepts in nonlinear control theory and the automation of the nonlinear controllability analysis
- the mechanism of projection of the multi-linkage system and associated instabilities.

In chapter three a number of essential components of a computer graphics package are reviewed. Two algorithms for the solution of the hidden surface problem are outlined. Animation studies of a human-like model, denoted ADAM, are shown. In chapter four the rotational dynamics of multi-linkage systems are derived by using two different approaches. As an example, the controllability of a constrained planar two link system is analyzed using concepts of linear and nonlinear control theory. The attitude control problem of the above system is defined and a control strategy is discussed. In chapter five the mechanism of projection of the multi-linkage system is addressed. Three algorithms are discussed, which can be employed to perform this projection numerically. The merits of these algorithms are illustrated by several examples. Finally, a summary is presented in Chapter six and direction of future research is discussed.
Chapter 2

SURVEY OF PREVIOUS WORK

2.1 Introduction

In this chapter the literature relevant to the subject matter of this dissertation is surveyed. In section 2.2, some fundamentals of computer graphics and the graphical representation of the kinematics of multi-body systems are discussed. In section 2.3 the dynamics of the above mentioned systems are presented and the mechanism of projection by the imposition of holonomic and simple nonholonomic constraint forces or torques is reviewed. In section 2.4 the instabilities which are caused by confining the system to a reduced state space are described. In section 2.5 the attitude control problem of multi-body systems is defined. Finally, nonlinear controllability concepts are presented in section 2.6.

2.2 Computer Graphics Applied to Movement Studies

An essential part of the study of dynamics of mechanical systems is the graphical representation of simulation results, i.e., the transformation and projection of trajectories of the object onto a viewing screen. This provides for a visual verification of simulation results, which can, in turn, be shown to a wider, not necessarily technically inclined, audience. Furthermore, computer graphics provides for means of data reduction or image enhancement, if necessary. In fact, in movement studies it is sometimes desired to see the object in the top, side, or frontal plane instead of having a three-dimensional image of it [1]. In gait analysis, on the other hand, it might be important to focus on the motion of the hip, for example, to identify asymmetries of gait. There is also an associated inverse problem; i.e., the generation of three-dimensional objects from a two-dimensional image. The inverse problem is of interest, for example, in the reconstruction of the three-dimensional trajectories of dummies in a crash test, which is recorded on two-dimensional photographic material [2].

Computer graphics is extensively applied in robotics, especially in CAD/CAM systems [3 - 14]. It is, for example, sometimes necessary to visualize the working envelope of an assembly robot [11]. On the other hand, computer graphics can be applied to verify application programs of robots by displaying the motion of the robotic mechanism [12].
In this dissertation a graphics software package is presented which displays the model of a humanoid and an assembly robot as a set of wireframe segments. This program provides interactive and file-driven input. The corresponding input parameters are the orientations of the individual segments of the composite system. The software package supports animation capabilities as well. In the present implementation the user is able to display a cyclic motion of the multi-body system at various speeds on the screen. An introduction to computer graphics and suggestions for the structure of such a software package are given in [15 - 17].

A number of the essential components of the basic graphic package are reviewed below. This package is primarily designed for the use of a vector display, but the objects may be shown on a raster scan display as well [16]. It generates a database of standard geometric objects, such as a sphere, a cone, a cylinder, and a bar, which are represented as wire frame baskets and stored in polygon list format [15 - 18]. The individual polygons of these objects are subject to program modules, which do either open polygon clipping [15 - 17] or closed polygon clipping using the Sutherland Hodgeman re-entrant clipping algorithm [19]. The hidden surfaces of the object can be removed upon request. There are essentially two types of algorithms [20], which accomplish this task. If the objects are convex polyhedra the above types of algorithms take a particular simple form and operate in object space, which is the space that describes the object before the clipping and projection operation. However, if the objects are concave polyhedra, the corresponding hidden-surface removal algorithms are more elaborate. Two different image space algorithms, which operate on the description of the object in image (or screen) space coordinates, are implemented in the graphics software package and are optimized for the present configuration. These are the Warnock [21] and Weiler-Atherton [22] algorithms. The former algorithm exploits area coherence of the objects, i.e., the tendency for small areas to be contained in at most a single polygon. It, consequently, subdivides the initially large display area into smaller but equal subareas until a correct decision on the visibility or invisibility of polygons is reached. The Weiler-Atherton algorithm uses object area coherence. It recursively subdivides polygons along boundaries of other polygons until a conclusive decision on visibility or invisibility is reached. After the clipping and hidden surface removal operation, the object is displayed perspectively on the screen [15 -17].

The modules of the basic graphics software package are used to display the more complex objects, such as the assembly robot and the humanoid, called ADAM. Some researchers concentrated on the simulation of human movement [23, 24]. The design of the humanoid is taken from [25], see Fig. 2.1. The humanoid is a 23 segment model, where the individual segments are represented as wire-basket solids of revolution.
Figure 2.1. ADAM: A Twenty-three Segment Humanoid
Figure 2.2: The Assembly Robot Consisting of a Robot Arm and a Conveyor Belt
Each of the legs consists of four segments: thigh, lower leg, foot, and toes. Similarly, the arms consist of four segments each: the upper arm, lower arm, the palm and the fingers. The main body consists of the head and a six-segmented torso; the individual segments are denoted: the lower torso, rib 4 to rib 1, and the upper torso. Each body segment has at most 6 degrees of freedom: three angles specifying its orientation and three corresponding angular velocities. The assembly robot, see Fig. 2.2, which is studied here, consists of a robot, which is modelled according to the specification of the Rhino robot arm [26], and a conveyor belt. In the animation studies, involving either the humanoid or the assembly-robot, a database is created, which consists of intermediate positions of the whole motion cycle. These intermediate picture frames are displayed on the vector graphics display to create a movie-like scene.

2.3 The Dynamics of Multi-Body Systems with Emphasis on Rotational Motion

In this section the dynamics of multi-body systems are reviewed with emphasis on rotational motion. In section 2.3.1 the general formulation of the above mentioned systems is presented using Bryant angles and Euler parameters as frames of reference. In section 2.3.2 several methods are introduced which serve to project the equations of motion of mechanical systems to a reduced subspace. The derivation of the rotational dynamics of multi-body systems is reviewed in section 2.3.3. Finally, in section 2.3.4 the symbolic generation of the dynamics of mechanical systems is discussed.

2.3.1 The Formulation of the Dynamics of Multi-Body System

In this dissertation mechanical systems, which are a collection of rigid bodies hinged together at ideal ball-and-socket joints, and which impose holonomic or simple nonholonomic connection constraints, are studied. The above mentioned systems are called multi-body systems in the sequel. These systems are used to represent bipeds [27-31], quadrupeds [32], and robotic mechanisms [33]. The dynamics of multi-body systems can be derived using the Hamiltonian formalism, or the Newton Euler or Lagrangian mechanics [34-38]. As the complexity of the multibody system increases, it is advantageous to resort to graph theory in specifying the mutual interconnection of bodies [39]. In what follows the construction of rotational and translational dynamics of the multi-body system are reviewed. Consider Fig. 2.3, which shows the i\textsuperscript{th} body of a kinematic chain in a body\textsubscript{i} coordinate system (R\textsuperscript{i}(CS)) with axes \(X_{1\textsubscript{i}}, X_{2\textsubscript{i}}, X_{3\textsubscript{i}}\), relative to a world (or inertial) coordinate system (ICS) with axes \(X_W, Y_W, Z_W\). The origin of the i\textsuperscript{th} body is denoted \(O_i\). The position vectors \(K_i\) and \(L_i\) point to the hinge point with the (i+1)\textsuperscript{th}
Figure 2.3. The $i^{th}$ Body of a Kinematic Chain and the Body $i$ and the Inertial Coordinate System
body and the \((i-1)\)th body relative to \(0_i\). Other quantities, which are not shown in Fig. 2.1, are vectors \(\Gamma_i\) and \(N_i\) of reaction forces acting on body \(i\) and body \((i-1)\) and vector of external torques acting on the respective joint. The orientation and angular velocity of the \(i\)th body are specified in minimal state space form by maps \(A_i\) and \(B_i\) using Bryant angles [39]. Then the equations of motion for the \(i\)th link are [39, 40]:

\[
\begin{align*}
\dot{m}_i \dot{x}_i &= -m_i g y_i + \Gamma_i - \Gamma_{i+1} \\
\dot{\theta}_i &= B_i w_i \tag{2.1}
\end{align*}
\]

\[J_i \ddot{\theta}_i = -w_i J_i w_i + K K_i A_i^{-1} \Gamma_i - L L_i A_i^{-1} \Gamma_{i+1}\]

with

\[
X_i = (x_1, i, x_2, i, x_3, i)^T
\]

\[\theta_i = (\theta_1, i, \theta_2, i, \theta_3, i)^T\]

\[W_i = (w_1, i, w_2, i, w_3, i)^T\]

\[
WW_i = \begin{bmatrix}
0 & -w_3, i & w_2, i \\
+ w_3, i & 0 & -w_1, i \\
- w_2, i & w_1, i & 0
\end{bmatrix}
\]

\[
A_i = \begin{bmatrix}
C_2 C_3 & -C_2 S_3 & S_2 \\
C_1 S_3 + S_1 S_2 C_3 & C_1 C_3 - S_1 S_2 S_3 & -S_1 C_2 \\
S_1 S_3 - C_1 S_2 C_3 & S_1 C_3 + C_1 S_2 S_3 & C_1 C_2
\end{bmatrix}
\]

\[
B_i = \begin{bmatrix}
C_3 / C_2 & -S_3 / C_2 & 0 \\
S_3 & C_3 & 0 \\
-C_3 S_2 / C_2 & S_2 S_3 / C_2 & 1
\end{bmatrix}
\]

\[
C_1 = \cos(\theta_1, i) ; C_2 = \cos(\theta_2, i) ; C_3 = \cos(\theta_3, i) \\
S_1 = \sin(\theta_1, i) ; S_2 = \sin(\theta_2, i) ; S_3 = \sin(\theta_3, i)
\]
Alternatively, the orientation and angular velocity can be specified in nonminimal state space form by maps $A_i^*$ and $B_i^*$ using Euler parameters. The equations of motion for the $i^{th}$ link are then given by:

\[
\begin{align*}
\ddot{m}_i \hat{X}_i &= -m_ig\hat{y}_w + \Gamma_i - \Gamma_{i+1} \\
\dot{Q}_i &= B_i^* W_i^* \\
J_i \ddot{W}_i &= -WW_i J_i \dot{W}_i + KK_i(A_i^*)^{-1} \Gamma_i - LL_i(A_i^*)^{-1} \Gamma_{i+1} + N_i - N_{i+1}
\end{align*}
\]

with $W_i^* = (0, W_{1,i}, W_{2,i}, W_{3,i})^T$.

\[
A_i^* = \begin{bmatrix}
2(q_0^2 + q_1^2) - 1 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\
2(q_1q_2 + q_0q_3) & 2(q_0^2 + q_2^2) - 1 & 2(q_2q_3 - q_0q_1) \\
2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 2(q_0^2 + q_3^2) - 1
\end{bmatrix}
\]

\[
B_i^* = \begin{bmatrix}
q_0 & -q_1 & -q_2 & -q_3 \\
q_1 & q_0 & -q_3 & q_2 \\
q_2 & q_3 & q_0 & -q_1 \\
q_3 & -q_2 & q_1 & q_0
\end{bmatrix}
\]

There is a relation between the Euler Parameters and the Bryant angles [39]:

\[
\begin{align*}
q_{0,i}^2 &= \frac{\text{tr} A_i + 1}{4} \\
q_{j,i}^2 &= \frac{A_{ij}}{2} - q_{0,i}^2, \quad (j = 1, 2, 3)
\end{align*}
\]
Nonminimal state space formulations are advantageous for computational reasons [37, 39], redundant computer simulations and digital computer simulations of redundant systems.

2.3.2 Methods for the Projection of the Equations of Motion of Multi-Body Systems

The mechanism of projections is an integral part in the study of the dynamics of multi-body systems. The projection is accomplished by the imposition of holonomic or simple nonholonomic constraint forces or torques. Holonomic constraints are, by definition, represented by algebraic equations, in the generalized coordinates and the time. All the other constraints, which do not fall in the above category, are denoted as non-holonomic constraints [34 - 36]. The term simple, non-holonomic constraints refers to those non-holonomic constraints that are linear in the generalized velocities. In mechanics, further subgroups of constraints are defined [39], which are not mentioned here. In the multi-body systems, which are studied in this dissertation, holonomic constraints have to be enforced, if the individual bodies of the composite system, are jointed together in a kinematic chain [40]. If these connections are such that rotational degrees of freedom are lost, constraint torques are engaged, which model these particular joint characteristics [40].

The multi-body systems and the corresponding constraints can be modelled in the following way. Consider Fig. 9.4, which shows a simulation diagram of a constrained mechanical system. The vectors of state, input, and constraints are denoted X, U, and r. The dynamics of the individual bodies of the composite system, which are specified by Eq. (2.1), or (2.2), are represented by the block:

\[ X = X(X, U, r) \]

It is shown [14] that the constraint forces or torques can be represented as a function of the state and input of the system:

\[ r = r(X, U) \]

The constraint forces serve different purposes depending upon the application considered. In robotics [41], locomotion [33], or in biomechanical systems [42], the constraint forces are controlled such that kinematic constraints are maintained or deliberately violated. Hence, for the synthesis of control laws, it is desired to derive expressions for the constraint forces as functions of the state and the input of the mechanical system, and to derive the sensitivity of these forces with respect to the state and the input [43]. The holonomic constraints can be grouped into model-intrinsic and model-environment constraints. The former group reflects the interconnection of
Fig. 2.4 Simulation Diagram of a Constrained Mechanical System.

\[ r = r(X, U) \]

Forces of constraint

\[ \dot{X} = f(X, U, F) \]

Input \rightarrow \dot{X} \rightarrow \int \rightarrow state
individual rigid links in a system [44]. The above constraints are permanently engaged. They can be implemented into the system's dynamic formulation by a method which projects an unconstrained system to a lower dimensional, constrained subspace [40]. If the interconnection between links does not allow any movement at all; the corresponding links are locked [45].

The other group of holonomic constraints describe the interaction of the mechanical model with its environment, such as ground contact of legs [16-49], skiing [50], sliding [42], or tactile sensing [51-53] where the sensors detect the partial shape, characteristics, and relative motion of an unknown object by surface contact.

The computation of constraint forces as proposed by Smith [54] is suitable primarily for analysis. In this method, the n dynamic and m constraint equations are solved simultaneously for the unknown accelerations and forces of constraint.

In the second method [55], suitable for control and simulation, the dynamic equations are substituted back in the constraint equations to eliminate the accelerations. The remaining system of m equations is then solved for the forces as a function of the state and input (Fig. 2.2).

In the third method, proposed by Kane [34], the mechanical system is embedded in a larger state space. This space is chosen here in such a way that the constraint equations, corresponding to the unknown constraint forces are linear functions of the new state of the system. The imposition of these simpler constraints reduces this larger dimensional system to the original one and renders appropriate expressions for the constraint forces. It is shown on several examples that the calculation of these forces results in some computational savings over the first and second method.

The computational complexity of each method is assessed by provided a count of arithmetic operation involved. This can further be used to estimate the computation time on various computers in a Fortran environment [56]. The accuracy of the individual method is inferred from the condition number of the respective coefficient matrix of the constraint forces. This provides an upper bound of the relative residual and the relative error [57-63].

2.3.3 The Dynamics of Multi-Body Systems in the Air

In the study of attitude control of multi-body systems, see section 2.6, the equations of motion of the multi-body system are conveniently described by the translational motion of the center of mass of the composite system and by the rotational equations of motion.
Those multi-body systems are of interest in the study of spacecraft dynamics [64-71] or in analyzing the motion of human [71] or animals under free fall conditions. The cat, for example attracted considerable attention, since it always lands safely on all four legs [73-80]. Several studies focus on astronauts, who are maneuvering in outer space [81-87], or on various disciplines in sport, where the athletes are subject to free fall conditions, for example diving [30], hurdle running, or pole vaulting.

The derivation of the rotational equations of motion is considered next. Many researchers have addressed this problem in the past [39, 64, 72]. In order to ensure correct results, two independent methods of deriving the rotational equations of motion are used. In the first method [32] a free body analysis is done on the system and a decoupling transform is used to extract the rotational dynamics. In the second method [39, 64] the rotational equations of motion are derived based upon the system's interconnection structure.

2.3.4 The Symbolic Generation of the Equations of Motion of Multi-Body Systems

The symbolic generation of the equations of motion of multi-body systems has attracted many researchers [88-101]. In this dissertation the rotational equations of motion are generated by two different methods using the symbolic language MACSYMA [100]. Thus, the unreduced expressions, which arise through the imposition of constraints [40] can be reduced to their minimal form by symbolically applying suitable trigonometric identities and by cancelling of common factors, for example. The symbolic manipulation eliminates truncation errors which are caused by the finite number representation of intermediate results. The symbolic computations offer additional advantages over numeric computations [99], which are, however, at the expense of the speed of computation. The latter problem can be overcome by translating the Lisp code, which is the language used to code MACSYMA, into FORTRAN code [101] for the purpose of iteratively solving the dynamic equations.

2.4 Numerical Instabilities Induced by the Imposition of Holonomic and Simple Nonholonomic Constraints

The imposition of holonomic and simple, nonholonomic constraints on a multi-body system changes its stability properties [102-105]. These instabilities can be compensated by numerical stabilization methods [102-110]. These methods, which are originally used to stabilize autonomous systems, can be applied to nonautonomous systems as well, but with relaxed requirements, i.e., it suffices that the latter systems are made stabilizable. Stabilization methods are suggested by Baumgarte and others [102-115], and agree very well with concepts in control theory.
In the latter formulation the numerical stabilization methods are equivalent to controlling the unconstrained system, subject to holonomic and simple nonholonomic constraints. The system is acted on by additional hypothetical inputs, whose only function is to maintain the constraints and assure stability.

2.5 The Attitude Control of Multi-Body Systems

In this section the attitude control of multi-body systems, which are subject to a constant angular momentum, is discussed.

The angular momentum principle very often serves as an analytical tool to bear on questions in the field of sports and space exploration. As an illustration of this principle consider the following question. Assume that a man is freely falling after losing ground contact. How can he move various parts of his body relative to each other to change his orientation in space? A solution to this problem is proposed below. Assuming that the man has no means of producing any external forces, he is subject to the gravity force only, while the angular momentum about his center of gravity remains constant. By virtue of the angular momentum principle he can apply internal forces to change the relative position of body parts, so that his orientation is changed for the purpose of keeping the angular momentum constant.

Trampolinists, trapeze performers and divers are some of the athletes who exploit this principle. In fact, a diver [80] who was trained to perform the same righting movements that a cat does when it is released from an arbitrary position above ground. The physical phenomenon, which was analyzed in this way, is that a cat always lands safely on its feet and never on the back, as one might suspect. Other examples which illustrate the use of this principle are hurdle runners, long jumpers, and astronauts maneuvering in free space. Astronauts have to be able to change the orientation of their bodies in a state of weightlessness to perform certain tasks in connection with manned space flight [85-87]. If they are not in direct contact with the space vehicle they have the choice of using either a gas gun to produce the desired change or moving various parts of their body to accomplish the same task. In the former case the angular momentum does not remain constant while in the latter case it does. The latter case is attractive because it can be applied in case of a mechanical failure of the force producing device. It also eliminates the need for astronauts to carry excessive amounts of supply fuel, since certain maneuvers can be done by exploiting the angular momentum principle.

In the study of the attitude control of multi-body systems a direct and an inverse attitude control problem are defined as follows [82]. The variables which specify the orientation of the composite system relative to an inertial system, are denoted external variables, while those, which specify the relative position of body parts to each other,
are called internal variables. In the direct attitude control problem the internal variables are completely specified, while the external ones are unknown. This leads to three coupled, first-order differential equations with three unknowns in three dimensions, which can be solved by numerical integration. In the inverse attitude control problem, the situation is reversed, i.e., the external variables are known and the internal ones are unknown. This problem occurs in manned spacecraft, for example, where it is necessary for the astronaut to know, how to move his limbs to bring about a change of his orientation. The inverse attitude control problem also leads to three coupled, differential equations but in the unknown external variables, which usually exceed the number of three. Hence, this is an ill-posed boundary value problem, which requires considerable effort to find a solution [81].

2.6 Nonlinear Controllability Concepts

The controllability analysis of a system determines whether the inputs to the system are able to drive the respective state vectors to any arbitrary point in state space. For linear systems one of several standard tests [116] can be used to determine their controllability. The same analysis is more elaborate for nonlinear systems. However, the standard tests for the controllability of linear systems can be applied to nonlinear systems as well, when the latter ones are approximated by a linear system. This can be done by expanding the nonlinear system in a Taylor series and by retaining only the first order terms. The latter test sometimes fails to determine the exact controllability properties of the system [117]. Therefore, nonlinear controllability tests are designed to overcome the above mentioned difficulty. The Lie bracket operation is an integral part of the nonlinear controllability analysis. Their derivation is given below for a scalar system, but without loss of generality. Consider the system

\[ \dot{x} = \sum_{i=1}^{m} u_i(t) f_i[x(t)], \quad x_0 \in \mathbb{R}^n \]  

(2.1)

which evolves on an n-dimensional state space and has controls \( u_i(i = 1, \ldots, m) \in \{0, 1\} \), which can be switched on and off. A solution to Eq. (2.1) is denoted

\[ x(t) = \exp[tf]x_0 \]  

(2.2)

regardless of whether Eq. (2.1) represents a linear or nonlinear system. The question of controllability of the system, given by Eq. (2.1), can now be reformulated as to whether the terms \( f_i(i = 1, \ldots, m) \) in Eq. (2.1) span the space in \( \mathbb{R}^n \), which is tangent to \( x(t) \), specified by Eq. (2.2). Hence, it is desired to systematically identify basis
vector fields for this tangent space. At the end of this process it has
to be determined, whether these basis vector fields, span the whole
tangent space. These basis vector fields can be found as explained
below. Consider Fig. 2.1, which shows the trajectory of the state
vector in the respective state space. The trajectory starts at \( x_0 \),
which is the initial value of \( x(t) \), Eq. (2.2). It is assumed that all
but the second control, Eq. (2.1), are switched off, so that \( x(t) \) moves
along the vector field \( f_2 \) for \( t \) units of time in forward direction until
point \( x_1 \) is reached. At this point all but the first control are
switched off and the state vector moves along the vector field \( f_1 \) in
forward direction up to point \( x_2 \). Now, the same vector fields are
traversed in opposite directions until point \( x_4 \) is reached, see Fig.
2.1. If point \( x_4 \) does not coincide with the starting point \( x_1 \), a new
direction in state space is created by the above sequence of traversal
of vector fields \( f_1 \) and \( f_2 \). However, if points \( x_1 \) and \( x_4 \) coincide, no
new direction in state space is generated.

A Taylor series expansion for the system

\[
\dot{x}(t) = f_1(x(t)) \quad x(0) = x_0
\]

about \( x_0 \), assuming that it exists, is:

\[
\dot{x}(t) = f_1(x_0) + \frac{df_1(x_0)}{dt} t + \frac{1}{2} \frac{d^2f_1(x_0)}{dt^2} t^2 + O(t^3)
\]

(2.3)

The integration of Eq. (2.3) yields:

\[
x(t) = x_0 + f_1(x_0) t + \frac{t^2}{2} \frac{df_1(x_0)}{dt} + O(t^3)
\]

Hence, a second order expression for \( x(t) \) of Eq. (2.3) is:

\[
x(t) = x_0 + tf_1(x_0) + \frac{t^2}{2} \frac{df_1(x_0)}{dt} + O(t^3)
\]

(2.4)

where \( f_1(x_0) = \frac{dx_0}{dt} \)

The above second order expansion is used to find an expression for
\( x_1, x_2, x_3, \) and \( x_4 \) as a function of \( x_0 \), Fig. 2.1. The value of
\( x_1 \) is, using Eq. (2.4):
Figure 2.5. Illustration of the Lie Bracket Operation
\[ x_1(t) = \exp(tf_2)x_0 \]
\[ = x_0 + tf_2(x_0) + \frac{t^2}{2} \frac{\partial f_2(x_0)}{\partial x} f_2(x_0) + O(t^3) \]

Similarly, the value of \( x_2 \) is:

\[ x_2(t) = \exp(tf_1)x_1 = \exp(tf_1) \exp(tf_2)x_0 \]
\[ = x_1 + tf_1(x_1) + \frac{t^2}{2} \frac{\partial f_1(x_1)}{\partial x} f_1(x_1) + O(t^3) \]  \hspace{1cm} (2.5)

Since

\[ f_1(x_1) = f_1(x_0) + \frac{\partial f_1(x_0)}{\partial x} (x_1-x_0) + \frac{1}{2} \frac{\partial^2 f_1(x_0)}{\partial x^2} (x_1-x_0)^2 + O(x_1-x_0)^3 \]
\[ x_2(t) = x_0 + tf_2(x_0) + \frac{t^2}{2} \frac{\partial f_2(x_0)}{\partial x} f_2(x_0) + O(t^3) \]
\[ + t\{f_1(x_0) + \frac{\partial f_1(x_0)}{\partial x} (x_1-x_0) + \frac{1}{2} \frac{\partial^2 f_1(x_0)}{\partial x^2} (x_1-x_0)^2 + O[(x_1-x_0)^3] \}
\[ + O(t^3) \]

The value of \( x_3(t) \), Fig. 2.1, is found to be:

\[ x_3(t) = \exp(-tf_2)x_2 \]
\[ = x_2 - tf_2(x_2) + \frac{t^2}{2} \frac{\partial f_2(x_2)}{\partial x} f_2(x_2) + O(t^3) \]
\[ = x_0 + tf_1(x_0) + \frac{t^2}{2} \left[ \frac{\partial f_1(x_0)}{\partial x} f_1(x_0) + 2\frac{\partial f_1(x_0)}{\partial x} f_2(x_0) \right.
\[ - 2\frac{\partial f_2(x_0)}{\partial x} f_1(x_0) \left] + O(t^3) + O[(x_1-x_0)^3] + O[(x_2-x_0)^3] \right. \]

Similarly,

\[ x_4(t) = \exp(-tf_1)x_2 \]
\[ = x_3 - tf_1(x_3) + \frac{t^2}{2} \frac{\partial f_1(x_3)}{\partial x} f_1(x_3) + O(t^3) \]
\[ = x_0 + t^2 \left[ \frac{\partial f_1(x_0)}{\partial x} f_2(x_0) - \frac{\partial f_2(x_0)}{\partial x} f_1(x_0) \right]
\[ + O(t^3) + O(x_1-x_0)^3 + O(x_2-x_0)^3 + O(x_3-x_0)^3 \]
With \( \frac{\partial f_1}{\partial x} f_2 - \frac{\partial f_2}{\partial x} f_1 \triangleq [f_1, f_2] \): Lie bracket of \( f_1, f_2 \)

\[ x_4(t) = x_0 + t^2[f_1, f_2] + \text{higher order terms} \quad (2.6) \]

Under the assumptions that the higher order terms in Eq. (2.6) vanish, it follows that points \( x_0 \) and \( x_4 \) in Fig. 2.1 coincide if the Lie bracket of fields \( f_1 \) and \( f_2 \), denoted \([f_1, f_2]\), vanishes. Hence, the Lie bracket operation serves as a tool to identify basis vector fields which span the tangent space for \( x(t) \), Eq. (2.2). Further references on differential geometry and concepts of nonlinear control theory are found in [118-124].

In this dissertation the above concepts are applied to mechanical systems. Crouch [125] used geometric control theory to analyze a rigid one link system in three-dimensional space. Khosravi [4] analyzed the controllability of a planar linkage system in the air and on the ground.

In what follows the notation of Marino and Cesareo [126] is used, who considered a nonlinear system

\[ \dot{x} = f(x) + G(x)u = f(x) + \sum_{i=1}^{m} g_i(x) u_i \quad (2.7) \]
\[ y = h(x) \]

where \( x \in \mathbb{R}^n \), \( u \in \mathbb{R}^m \), and \( y \in \mathbb{R}^r \) denote the state, input, and output of the system. They define two basic distributions for the analysis of structural properties of Eq. (2.7):

" - the strongly accessible distribution \( L_0 \), which is the smallest involutive distribution which contains

\[ \text{span} \{ \text{ad}^l f g_i : l > 0, 1 < i < m \} \]

where

\[ \text{ad} f g_i = g_i, \text{ad} f g_i = [f, g_i] \]
\[ \text{ad}^{k+1} f g_i = \text{ad} f (\text{ad}^k f g_i) \]

" - the accessible distribution \( L \) which is the smallest involutive distribution which contains

\[ \text{span} \{ f_1, g_1, \cdots, g_m \} \]

where \( L(x) = \text{span} \{ L_0(x) f(x) \} \).

Further definitions are:
\[ G^0 = \text{span}\{g_1, \ldots, g_m\} \]
\[ G^j = \text{span}\{G^{j-1}, [G_f, G^{j-1}]\} : G_f = f + G^0 \]
\[ [G_f, G^{j-1}] = \text{span}\{[x, y] : x \in G_f, y \in G^{j-1}\} \]
\[ M^0(x) = G^0(x) \]
\[ M^j(x) = \text{span}\{M^{j-1}(x) [f, M^{j-1}(x)]\} \]

The following proposition [126] states the involutivity property of sets \( G^k(x) \) and \( M^k(x) \):

"a. If for some integer \( k \), \( G^k(x) = G^{k+1}(x) \) for every \( x \in V \), open subset in \( \mathbb{R}^n \), then \( G^k(x) = G^{k+1}(x) \) for any integer \( l > 0 \) and any \( x \in V \).

"b. If, for some integer \( k \), \( M^k(x) = M^{k+1}(x) \) for every \( x \in V \), open subset in \( \mathbb{R}^n \), then \( M^k(x) = M^{k+1}(x) \) for any integer \( l > 0 \) and any \( x \in V \)."

The next proposition [126] states the relationship between \( G^k(x) \) and \( L_0(x) \), which are given above:
\[ L_0(x) = G^{n-1}(x) \] for every \( x \in N \), open and dense submanifold in \( \mathbb{R}^n \).

Let
\[ n_a = \sup_{x \in \mathbb{R}^n} \dim L(x) \]
\[ n_{sa} = \sup_{x \in \mathbb{R}^n} \dim L_0(x) \]
\[ n_1 = \sup_{x \in \mathbb{R}^n} \dim M_{\text{max}}(x) \]
\[ n = \dim(x) \]

The tuple \( (n, n_a, n_{sa}, n_1) \) gives a preliminary classification of the system, Eq. (2.7) [126]:

If \( n = n_a \), the system is accessible.

If \( n = n_a = n_{sa} \), the system is strongly accessible.

If \( n > n_{sa} \), the quantity \( (n - n_{sa}) \) gives the number of (strongly) nonaccessible directions. If \( n = n_1 \), the system is feedback equivalent to a linear controllable system whose state space has dimension \( n_{sa} \).
Next some theorems of [127] are stated which allow the determination of the controllability properties of the mechanical systems considered here:

Theorem 1 [127, pg. 27]: Accessibility is a necessary but not sufficient condition for the controllability of the system, Eq. (2.7)

Theorem 2 [127, p. 61]: Assume the system, Eq. (2.7) is accessible with unconstrained controls and \( f(x) \) is a Poisson stable vector field, i.e., \( \text{div} \ (f(x)) = 0 \) [126, pg. 60]. Then the system is controllable.

Theorem 3 [127, pg. 27]: The accessible system given by Eq. (2.7) is strongly accessible if there exists some time \( T \), such that \( R(t, x_0) \subset S^r \), and \( R(T, x_0) \) is not contained in an analytic surface \( S^s \) of dimension \( s < r \), where

\[
R(t, x_0) : \text{time } T \text{ reachable set from the initial state } x_0 \\
S^r : \text{analytic surface of dimension } r
\]

The existence of a reachable set \( R(T, x_0) \subset S^r \) is interchangeably used with the controllability of the system on an \( r \)-dimensional manifold.

2.7 The Automation of the Nonlinear Controllability Analysis

The concepts of nonlinear control theory are largely applied to either scalar systems or to non-scalar systems of low dimensionality, which merely serve to illustrate the underlying principles. The nonlinear controllability analysis of systems with many states, for example, rapidly increases in complexity due to the characteristics of the matrices involved. Even though there is no difficulty in principle, the probability of making mistakes while performing tedious hand computations which require a large amount of time is very high. This problem can only be overcome with the help of a digital computer which is able to execute a symbolic manipulation program.

The above mentioned programs, which can be used to symbolically derive the equations of motion of a system, see section 2.3.2, are also applied for the evaluation of nonlinear controllability or observability properties [96, 126, 128]. Claude and Dufresne [128] use MACSYMA [84] to obtain decoupling of nonlinear systems by means of state-feedback. Marino and Cesareo [126] use REDUCE [81] to check feedback (or state) equivalence to linear controllable systems and to analyze the accessibility and strong accessibility properties of nonlinear systems, which are linear in the controls. They provide some algorithms, which are written in pseudocode, and can, therefore, be coded in any of the symbolic manipulation systems available.
2.8 Summary

In this chapter a number of previous works in the areas of kinematics, dynamics and control of multi-linkage system are surveyed. A twenty-three segment human model for kinematic studies is introduced. The dynamics of multi-linkage systems in the air and the symbolic generation of the equations of motion are discussed. Concepts in nonlinear control theory are reviewed. The mechanism of projection of multi-linkage systems and the associated instabilities are addressed.
Chapter 3

A GRAPHICS SOFTWARE PACKAGE FOR MOVEMENT STUDIES

3.1 Introduction

An important part of the study of dynamics of mechanical systems is the graphical representation of simulation results, i.e., the transformation and projection of the object onto a viewing screen. This provides for a visual verification of simulation results, which can, in turn, be presented to a wider, not necessarily technically inclined, audience. Computer graphics provides the option of data reduction or even image enhancement for the purpose of focusing on features of the object being investigated. In the study of animal or human movement [115], it might be desired to display a front, side, or top view of the object. The specification of an arbitrary viewing plane, however, provides the experimenter with more flexibility. Computer graphics is extensively employed in CAD/CAM applications [3-14]. It can be used, for example, to display the working envelope of an assembly robot [11]. Furthermore, it can be employed to verify application programs for robots by displaying the motion of the robotic mechanism [12].

In this chapter a graphics software package is introduced which displays the model of a humanoid (ADAM) and an assembly robot as a set of wire frame segments. This package provides interactive and file-driven input. The corresponding input parameters are the orientations of the individual segments of the composite system. This software package supports animation capabilities, as well. In the present implementation the user is able to display a cyclic motion of the multi-body system at various speeds on the screen. The graphics software package consists of three main modules. The basic graphics module contains a number of elementary graphics operations, including clipping, hidden surface removal, and perspective transformation [103-105]. The second main module contains a set of procedures which compute the two-dimensional image of a 23-segment humanoid, see Fig. 2.3. It uses the procedures of the former module as software primitives. Finally, the third module contains a number of subprograms which calculate the two-dimensional image of the assembly robot, see Fig. 2.4. Similarly, it uses the routines of the basic graphics modules as software primitives. The three major graphics software modules are listed in Appendix A1, A2, and A3. They are implemented on an IBM-PC (personal computer) using the language PASCAL. Some important information on how to use this graphics package is provided in Appendix A4, which shows the HELP-document that is installed in the computer system.
It is pointed out that some of the algorithms in this section were designed as projects for independent studies in the Department of Electrical Engineering, and were supervised by the author. These algorithms include the hidden surface removal routines and the procedures for animation.

The organization of this chapter is as follows. In section 3.2, the structure of the software and functional blocks of the hardware are described. In section 3.3 and 3.4, software elements, which perform elementary graphical functions, are discussed. Section 3.5 describes kinematic manipulations and object data structures. Sections 3.6 and 3.7 focus on plotting of polyhedra, which are wire-basket solids representing the object's segments, and on the implementation of animation capabilities.

3.2 The Hardware Arrangement and Software Structure

3.2.1 Hardware Arrangement

The hardware arrangement for the graphics system is shown in Fig. 3.1. The graphics software is general enough, so that the output can be displayed on both a raster-scan and a random-scan display. Consequently, the IBM-PC, which is the main processor in the present design, is interfaced to both types of output devices. The random scan display is the vector graphics system, which consists of a graphics generator and a graphics display. The graphics generator, which accepts input data in serial form, is connected to the parallel output of the IBM-PC via a special purpose interface. This interface transmits serial data at a high baud rate in order to achieve the transmission speed of a parallel interface.

The raster-scan display in the above configuration is the control display of the IBM, which can be switched between normal mode, i.e., text mode, and graphics mode. Finally, the IBM-PC is connected to a PDP11, which drives an X-Y plotter. This serves to get a hard copy of the final picture in the following way: A plot file which contains the display code for the final image, is sent to the PDP 11. The PDP 11 activates a module which translates the display code in equivalent commands for the X-Y plotter.

3.2.2 Software Structure

An overview of the basic graphics software is provided in Fig. 3.2, which indicates program dependencies as well. The function of the modules, whose names are largely self-explanatory, is explained below.
Figure 3.1. Hardware Arrangement for Graphics
Figure 3.2. Dependence Tree of the Basic Graphics

Software Package: GRAPH

28
The branch of programs, which are dependent on Init Perspective, does the clipping operations, removes hidden surfaces, and accomplishes the projection onto the display area. The block, denoted Utility, provides the essential mathematical operations, like matrix and file operation and initialization of kinematic transformation matrices. The above mentioned elementary routines are not listed by their individual names. The branch of programs, which depends on the program Init Graphic, consists of all graphic primitives [15, 16], which generate display code. The programs Gen TruncCone (generalized, truncated cone), sphere, pyramid, and GenBar (generalized bar) create a database of elementary geometric objects, represented by wire-basket solids. Finally, the programs dependent on animation contain all modules, which support the animation capabilities of the graphics package. All programs shown in Fig. 3.2 are listed in Appendix A.

3.3 Basic Graphics Software

3.3.1 Three-dimensional Viewing Process

A block diagram illustrating the three-dimensional viewing process is shown in Fig. 3.3. An object which is to be displayed on the screen enters the viewing process at the top of Fig. 3.3. It is typically described by a mesh of polygons, in which each individual polygon is represented by a set of vertices which are specified in world (or absolute) coordinates. These world coordinates are subject to a normalizing transformation, which consists of a world to eye space transformation, an aspect ratio adjustment, and a transformation to window edge coordinates. The normalizing transformation is described in section 3.3.1.1. The transformed object is then clipped in three dimensional space, which is discussed in section 3.3.1.2. The object, which is now in canonical view description, is transformed into screen space and its hidden lines are removed. This is described in section 3.4. Finally, the transformed segment is subject to two dimensional image space calculations, after which it is displayed on the screen [15, 16].

3.3.1.1 Normalizing Transformation

The normalizing transformation consists of a world to eye space transformation, an aspect ratio adjustment, and a transformation to window edge coordinates. The world to eye space transformation is explained below.

Consider Fig. 3.4a which shows a right-handed world coordinate system with axes $X_w$, $Y_w$, and $Z_w$ and origin $O_w$. An observer is specified by a local vector $EP$ (eye point) relative to $O_w$. A center of interest (or center of projection) is specified by a local vector COI relative to
Output Primitives In Three-Dim. Coord.

World To Eye Space Transformation

Aspect Ratio Adjustment

Transformation To Window-Edge Coordinates

Clipping Against Canonical View

Screen Space Transformation

Hidden Line Removal

Screen Space To Viewport Transformation

Display

Figure 3.3. Three-Dimensional Viewing Process
Figure 3.4. World and Eye Space Coordinate Systems

(a) World Space Coordinate System

(b) Eye Space Coordinate System
The direction of view is given by the difference vector $C_{OI}-E_{P}$, its unit normal is denoted by $P_{en}$, the direction pointing upwards relative to the direction of view is called $P_{eu}$. The above world space coordinate system is mapped into the eye coordinate system, which is shown in Fig. 3.4b, with axes $X_{e}$, $Y_{e}$, and $Z_{e}$. It is a left-handed system since the projection of its axes $X_{e}$ and $Y_{e}$ onto an imaginary viewing plane is supposed to represent horizontal and vertical extension of the image, while the coordinate $Z_{e}$, which is pointing into the plane, transmits the depth perception. In the eye coordinate system the observer position $E_{P}$ is at the origin. The $Y_{e}$ axis is aligned with the vector $P_{eu}$. Similarly, the $Z_{e}$ axis is aligned with the vector $P_{en}$. The transformation from the world to the eye coordinate system consists of a sequence of mappings [15 - 18], which are concatenated into a single matrix

$$(X_{e}', Y_{e}', Z_{e}', 1)^{T} = N(X_{W}, Y_{W}, Z_{W}, 1)$$

with $(X_{e}', Y_{e}', Z_{e}', 1)^{T}$: homogeneous coordinates of the eye coordinate system, without aspect ratio adjustment

$(X_{W}, Y_{W}, Z_{W}, 1)^{T}$: homogeneous world coordinates

$$N = \begin{bmatrix}
C_{\theta} & 0 & -S_{\theta} & -C_{OI_x}C_{\theta} + C_{OI_z}S_{\theta} \\
-S_{\phi}S_{\theta} & C_{\phi} & S_{\phi}C_{\theta} & C_{OI_x}S_{\phi}S_{\theta} - C_{OI_y}C_{\phi} + C_{OI_z}S_{\phi}C_{\theta} \\
-C_{\phi}S_{\theta} & -S_{\phi} & C_{\phi}C_{\theta} & C_{OI_x}C_{\phi}S_{\theta} + C_{OI_y}S_{\phi} + C_{OI_z}C_{\phi}C_{\theta} + d \\
0 & 0 & 0 & 1
\end{bmatrix}$$

$C_{OI} - E_{P} = (a, b, c)^{T}$: difference vector of center of interest and eye point

$$d = a^2 + b^2 + c^2 : \text{ length of } C_{OI} - E_{P}$$

$$f = \frac{a^2 + c^2}{d}$$

$$C_{\theta} = \cos \theta = \frac{c}{f}$$

$$S_{\theta} = \sin \theta = \frac{a}{f}$$

$$C_{\phi} = \cos \phi = \frac{f}{d}$$

$$S_{\phi} = \sin \phi = \frac{b}{d}$$
The aspect ratio adjustment compensates for the effect that distinctive vertical and horizontal line resolutions have on the final image. It consists of a scaling which is represented by a matrix $S$ s.t.:

$$\begin{pmatrix} x, y, z, 1 \end{pmatrix} = S \begin{pmatrix} x', y', z', 1 \end{pmatrix}$$

with

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & AR & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$\text{AR: aspect ratio} = \frac{\text{horizontal lines}}{\text{vertical lines}}$

The perspective transformation, which is considered next, is done in two steps. Initially, the vertices of the segment, which are now in eye space coordinates are transformed into coordinates of a canonical view. In the latter form, the subsequent clipping algorithm is particularly straightforward to implement. In the second and final step the vertices of the segment are transformed into the screen space coordinates. The perspective transformation has to reach two goals. Firstly, the final image has to be visually realistic, i.e., edges or polygons, which are far away from the observer have to appear smaller on the screen. Secondly, points in eye space are not simply projected onto a viewing plane but are transformed to it. This transformation is such that the z-depth information is preserved and that planes in eye space remain planes in screen space. This implies that 2 distinct points in eye space, which possess the same vertical and horizontal screen space coordinates, can be distinguished by their z-depth. Furthermore, the z-depth of an arbitrary point on a known polygon can be found by linear interpolation. The expressions for the perspective transformation are given below. First, the perspective projection is discussed, see Fig. 3.5a. In this figure a screen is shown, with height and width equal to a quantity $S$, which is located a distance $D$ relative to the origin of the eye coordinate system. The screen is parallel to the $X_e-Y_e$ plane and has its own coordinate system with axes $X_s$ and $Y_s$. A point $P$ with eye coordinates $(x_e, y_e, z_e)$ is projected onto the screen to point $P'$, which has coordinates $(x_s, y_s, D)$ in the eye coordinate system. The projection equations are found from Fig. 3.5b which shows the $X_e-Z_e$ plane of Fig. 3.5a. A similar figure for the $Y_e-Z_e$ plane can be obtained from Figure 3.5b by replacing $X_e$ by $Y_e$. It is assumed that the observer has a viewing angle denoted by $\theta$, i.e., the observer cannot see objects which are beyond the line $X_e = \pm Z_e \tan \frac{\theta}{2}$ and $Y_e = \pm Z_e \tan \frac{\theta}{2}$ in $X_e-Z_e$ plane and $Y_e-Z_e$ plane. From Fig. 3.5b:
a) eye coordinate system

b) $x_e, z_e$ plane of the eye coordinate system

Figure 3.5. Visual Realism by Perspective Projection
\[
x_s = \frac{x_e}{z_e/D}
\]

Normalizing the above equation with \(X_{smax} = D\tan \theta/2 = S/2\)

\[
\frac{x_s}{X_{smax}} = \frac{x_e}{z_e\tan \theta/2}
\]

Similarly, for the \(Y_e - Z_e\) plane

\[
\frac{y_s}{Y_{smax}} = \frac{y_e}{z_e\tan \theta/2}
\]

The property that planes in the eye coordinate system, denoted by,

\[ax_e - by_e + cz_e + d = 0\]  

(3.6a)

transform into planes in the screen coordinate system, denoted by

\[a'x_s + b'y_s + c'z_s + d' = 0\]  

(3.6b)

is satisfied if:

\[a' = a,\]
\[b' = b,\]
\[c' = \frac{d(H - Y)}{HY\tan \theta/2},\]
\[d' = \frac{c - d/H}{\tan \theta/2},\]
\[z_s = \frac{Y}{Y - H} + \frac{HY}{(H - Y)z_e}.
\]

where

\[a, b, c, d: \text{coefficients of the equation of a plane in the coordinate system.}\]
\[a', b', c', d': \text{coefficients of the equation of a plane in the screen coordinate system.}\]
\[(x_s, y_s, z_s): \text{screen coordinates.}\]
\[H: \text{location of the front clipping plane relative to } O_W.\]
\[Y: \text{location of the back clipping plane relative to } O_W.\]

The front (hither) clipping plane is due to the fact that objects which are too close to the observer cannot be displayed on the screen due to
its finite resolution. For the same argument a back (yon) clipping plane is introduced, which eliminates all points which are too far away from the observer.

The coefficients \(a, b, c,\) and \(d\) in (3.6a) can be found using elementary linear algebra [15]. They are related to the coefficients \(a', b', c',\) and \(d'\) of Eq. (3.6b) [15]. Equations (3.4) and (3.6) can written into a single matrix \(P,\) such that:

\[
(x_s', y_s', z_s', w_s')^T = P(x_e, y_e, z_e, 1)^T
\]

where \((x_s', y_s', z_s, w_s')^T:\) intermediate screen coordinates

\[
P = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & \alpha & \beta \\
0 & 0 & \delta & 0
\end{bmatrix}
\]

\[
\alpha = \frac{Y}{Y - H}; \quad \beta = \frac{Y \cdot H}{H - Y}; \quad \delta = \tan \frac{\theta}{2}
\]

The matrices \(N, S,\) and \(P\) of eq. (3.1), Eq. (3.2), and Eq. (3.7) are concatenated into a single matrix \(Q,\) s.t.:

\[
(x_s', y_s', z_s', w_s')^T = Q(x_w, y_w, z_w, 1)^T
\]

with

\[
Q = \begin{bmatrix}
C_\phi & 0 & -S_\phi & -C_{\phi}C_\theta + COI_2S_\theta \\
-ARS_\phi S_\theta & C_\phi AR & -S_\phi C_\theta AR & (COI_2S_\phi S_\theta - COI_1C_\phi + COI_2S_\phi S_\theta)AR \\
-C_\phi S_\phi \alpha \delta & -S_\phi \alpha \delta & -C_{\phi}C_{\phi} \alpha \delta & (COI_1C_\phi S_\phi + COI_1S_\phi + COI_2C_\phi C_\phi + d)\alpha \delta + \beta \delta \\
-C_\phi S_\theta \delta & -S_\phi \delta & -C_{\phi}C_\theta \delta & (COI_1C_\phi S_\theta + COI_1S_\phi + COI_2C_\phi C_\theta + d)\delta
\end{bmatrix}
\]

### 3.3.1.2 Three Dimensional Clipping

There are two issues of primary interest in clipping in three dimensional space. The first issue deals with the clipping of edges against the respective clipping planes. The second one deals with the application of the above mentioned algorithm to the problem of clipping a closed polygon against a set of clipping planes.
3-D Edge Clipping

The three dimensional edge clipping is explained below.

Consider Fig. 3.6, which shows the truncated viewing pyramid in the eye coordinate system. From the discussion in the previous section it follows that objects which are visible to the observer are bounded by the planes

\[-ze \tan \frac{\theta}{2} < x_e < ze \tan \frac{\theta}{2}\]

\[-ze \tan \frac{\theta}{2} < y_e < ze \tan \frac{\theta}{2}\]

\[H < z_e < y\]

(3.8)

where \(\theta\) is the viewing angle and \(H, Y\) specify the coordinates of the hither and yon clipping plane.

The intersection of a clipping plane with an edge can be calculated using elementary vector analysis as follows: consider a plane, which is specified by

\[N \cdot P + d = 0\]

where \(N\) is the plane's normal, \(d\) is the shortest distance of the plane to the origin, and \(P\) is an arbitrary point. An edge is specified as

\[P = P_1 + t(P_2 - P_1)\]

(3.9)

where \(P_1\) and \(P_2\) are initial and terminal points of the line, and \(P\) is again an arbitrary point on the line, and \(t\) is a scalar variable. The value of \(t\), which corresponds to the intersection of the line with the plane is

\[t = \frac{-d - N \cdot P_1}{N \cdot (P_2 - P_1)}\]

(3.10)

The implementation of Eq. (3.10) and the corresponding tests for edge to clipping plane intersection [17] produce an inefficient algorithm. The use of homogeneous coordinate provides for more efficient clipping routine [15]. Performing the multiplication in Eq. (3.7) produces intermediate (homogeneous) screen coordinates.
Figure 3.6. Eye Space Coordinate System with Truncated Viewing Pyramid

Figure 3.7. Window Eye Coordinates
\[ x_s' = x_e \\
\]
\[ y_s' = y_e \]
\[ z_s' = \frac{y}{\sqrt{h}} \tan \frac{\theta}{2} (z_e - H) \] 
\[ w_s' = z_e \tan \frac{\theta}{2} \]

It follows that the bounds specified in (3.8) can be written as:

\[ -w_s' < x_s' < w_s' \]

\[ -w_s' < y_s' < w_s \] 

\[ 0 < z_s' < w_s \] (3.12)

Consequently, an edge which violates the above inequalities is clipped. Using the quantities \( x_s' \), \( y_s' \), \( z_s' \), and \( w_s' \) as new coordinates introduces the window edge coordinates, Fig. 3.7. In this new space the clipping planes have particularly simple equations. A clipping algorithm based on window edge coordinates is given in [15] and is implemented in the present software package.

Three Dimensional Polygon Clipping

The three-dimensional polygon clipping deals with the implementation of a clipping algorithm to clip the individual edges of a (closed) polygon. Consider Fig. 3.8, which shows a polygon mesh representing the front faces of a sphere. In Fig. 3.8a the individual edges of the mesh are subjected to a clipping algorithm, i.e., edges which intersect clipping planes are consequently clipped and edges which are beyond those planes are discarded. It is obvious that this method produces open polygons as output polygons if a clipping is performed. The effect of this clipping method is seen on the poles of the sphere in Fig. 3.8a, i.e., several edges of the polygon mesh appear to be missing. Consider now Fig. 3.8b, which shows the same object as in Fig. 12a, using an algorithm which produces closed output polygons [19]. The edges, which are missing in Fig. 3.8a, are indeed displayed. It can be seen that the use of the latter algorithm produces a more realistic display image.

3.4 Hidden Surface Removal Algorithms

In the following section hidden surface removal algorithms are discussed. It is distinguished between algorithm for convex polyhedra, i.e., the corresponding interior angles are smaller than or equal to \( \pi \), and
Fig. 3.8. Clipping of a Polygonal Mesh
a) Open Polygon Clipping
b) Closed Polygon Clipping
concave polyhedra, i.e., those, which do not fall in the above category. The hidden surface removal algorithm for convex polyhedra, which are described in section 3.4.1 are fairly straightforward to implement. The corresponding algorithm for concave polyhedra, which are discussed in section 3.4.2 are more complex ones.

3.4.1 Hidden Surface Removal Algorithm for Convex Polyhedra

It is assumed in the following section that polyhedra have the data structure of the polygon list format [16, 18]. In this format polyhedra are redundantly described by a set of vertices, \( v_1, \ldots, v_N \), which consist of the coordinates of the vertices in three-dimensional space, and a set of polygons \( p_1, \ldots, p_M \), where \( p_j \) (\( i = 1, \ldots, m \)) consists of a list of pointers to those vertices which form the polygon's edges. The redundancy is introduced by the fact that edges which belong to several polygons are listed more than once. (Hidden surfaces, i.e., surfaces which are invisible to the observer, can be removed by forming the dot product between the surface normal and the vector of the viewing direction. The dot products of front and back face can be distinguished by their respective signs. These faces, for which the dot product vanishes, are back faces as well [15].)

3.4.2 Hidden Surface Removal for Concave Polyhedra

The algorithms for hidden surface removal of concave polyhedra are described more in detail. In section 3.4.2.1 standard tests, which are used in these algorithms are listed. In section 3.4.2.2 a hidden line removal algorithm is described which is based on area subdivision [21] and operates in image space, i.e., it uses the object's description in image (or screen) space as input data. The so-called Warnock algorithm exploits area coherence, i.e., the tendency for small areas to be contained in at most a single polygon. Consequently, the algorithm recursively subdivides the initial large display area into smaller subareas until a correct decision on visibility of invisibility of a polygon is reached. Another area subdivision algorithm is described in section 3.4.2.3. The so-called Weiler Atherton [22] algorithm exploits object area coherence, i.e., it recursively subdivides the original area along polygon boundaries until a correct visibility or invisibility decision is reached. The current implementation of this algorithm dramatically decreases the number of subdivisions as compared to the former algorithm. However, at each iteration stage, decisions are conducted which are more complex than the corresponding ones of the Warnock algorithm.
3.4.2.1. Decision Tests

In this section decision tests [17] are listed, which are used in the current implementation of the Warnock [21] and Weiler Atherton [22] algorithm.

Intersection of two edges:
Two edges $E$ and $E'$, each described by two points
$P_i = (x_i, y_i) \in E$ [i = 1, 2]
and
$P_i' = (x_i', y_i') \in E'$ [i = 1, 2]
intersect at point $(x_j, y_j)$ if:

$$\min[\min(x_1, x_2), \min(x_1', x_2')] < x_j < \max[\max(x_1, x_2), \max(x_1', x_2')]$$
and

$$\min[\min(y_1, y_2), \min(y_1', y_2')] < y_j < \max[\max(y_1, y_2), \max(y_1', y_2')]$$

(3.13)

Containment of a polygon in a rectangular area

A polygon $P$, which is specified by a tuple of its vertices
$P = (V_1, V_2, \ldots, V_m)$ with $V_j = (x_j, y_j), j \in [1, m]$ is contained in the rectangular area $A$ with

$$A = \{ x, y | x_{\text{min}} < x < x_{\text{max}} \text{ and } y_{\text{min}} < y < y_{\text{max}} \}$$

if

$$x_j \in [x_{\text{min}}, x_{\text{max}}] \bigcup j$$

$$y_j \in [y_{\text{min}}, y_{\text{max}}] \bigcup j$$

(3.14)

Test for non overlap of two closed convex polygons in image space

Two polygons $P_1$ and $P_2$, each specified by a tuple of its vertices,
s.t.

$$P_1 = (V_1, 1, V_1, 2, \ldots, V_1, m)$$
$$P_2 = (V_2, 1, V_2, 2, \ldots, V_2, n)$$

are disjoint in image space if the following minimax tests hold:
\begin{align}
&\max[x_{1,i}] < \min[x_{2,j}] \\
&\text{or} \\
&\max[x_{2,j}] < \min[x_{1,i}] \\
&\text{or} \\
&\max[y_{1,i}] < \min[y_{2,j}] \\
&\text{or} \\
&\max[y_{2,j}] < \min[y_{1,i}] \\
&i \in [1, \ldots, m] \\
&j \in [1, \ldots, n]
\end{align}

### Containment of one polygon in another polygon

Consider Fig. 3.9. A polygon $P_1$ completely surrounds another polygon $P_2$, where $P_1$ and $P_2$ are as specified before, if:

\[
\sum_{j=1}^{m-1} \arg (V_{2,i}, V_{1,j} ; V_{2,i}, V_{1,j+1}) = 2\pi \quad i \in [1, n] \quad (3.16)
\]

where $V_{2,i} V_{1,j}$ is the local vector from the vertex $V_{2,i}$ to vertex $V_{1,j}$.

The implementation of the above tests in the hidden surface algorithm has to be such that decision tests for cases with a high probability of occurrence have to be done prior to those tests for cases with a lower probability of occurrence. For example, the test for complete containment, which is fairly expensive, has to be done only, if the the tests for disjoint polygons has failed. Furthermore, in computing edge intersection, care has to be taken to avoid degenerate cases, which are shown in Fig. 3.10.

### 3.4.2.2. Warnock Algorithm

A flowchart of the Warnock [21] algorithm is shown in Fig. 3.11. The algorithm, which is coded in PASCAL, recursively subdivides the original rectangular display area into four subareas of equal dimensions until a conclusion is reached on the final display of the area element. The subareas can in the limit be such that they either represent a line segment or an individual pixel of the final image. The recursion is terminated if one of the four cases, which are shown at the bottom of Fig. 3.11, occurs. From left to right, in the first case all polygons are disjoint with the area element. Consequently, only the background value of it is displayed. In the second and third case, only one polygon intersects or is completely contained in the area element.
Figure 3.9. Test for Containment of One Polygon in Another Polygon

Figure 3.10. Degenerate Cases of Edge Intersection
Figure 3.11. Flowchart of the Warnock Algorithm
Push (initially)

CenterH, CenterV, SideH, SideV, Polygons : P(i)

Return

Pop

CenterH, CenterV, SideH, SideV, Polygons : P(i)

Return

Push (recursively)

CenterH1 = CenterH/2; CenterV1 = CenterV/2;
CenterH2 = CenterH1 + CenterH2;
CenterV2 = CenterV1 + CenterV2;
CenterH3 = CenterH1 + CenterH3;
CenterV3 = CenterV1 + CenterV3;
SideHNew = SideH/2;
SideVNew = SideV/2

Push:
CenterH1; CenterV1; SideHNew; SideVNew
Polygons

Push:
CenterH2; CenterV2; SideHNew; SideVNew
Polygons

Push:
CenterH3; CenterV3; SideHNew; SideVNew
Polygons

Push:
CenterH4; CenterV4; SideHNew; SideVNew
Polygons

Return

Figure 3.11 (cont'd)
Hence, the contained part of the polygon is displayed together with the background value of the rest of the area element. The fourth case is characterized by the fact that one polygon completely surrounds the area element. Consequently, only the color of the surrounder polygon is displayed. The decision tests, which are discussed in section 3.4.2.1 are implemented according to the relative probability of occurrence of the respective case. In the case of recursive subdivision the area element description and the potentially contained or intersecting polygons are pushed on the stack. The algorithm is terminated if the stack is empty.

3.4.2.3 Weiler-Atherton Algorithm

The discussion of the Weiler Atherton algorithm [22] consists of a description of the actual hidden surface algorithm and a description of the special case of polygon clipping.

Hidden surface removal algorithm:

Consider the flowchart of the Weiler Atherton algorithm in Fig. 3.12. Initially all polygons are roughly presorted according to their z-depth, to increase algorithm efficiency. For example, the polygons can be sorted according to the average z-depth of their corresponding vertices. The polygon on top of the list, which has the lowest approximate z-depth, is taken to be the clip polygon, see next section. The remaining polygons are put on the input list, i.e., they are subject polygons which are clipped against the clip polygon. In what follows an inside list and an outside list are created. The inside list contains all polygons or all fragments of polygons, which are inside the clip polygon. The outside list contains all the remaining polygons or fragmented polygons. In case that the actual clip polygon is not the foremost polygon, the algorithm recursively subdivides the polygons of the inside list into smaller fragments until a z-depth test conclusively determines visibility or invisibility of polygon fragments. After this the inside list is displayed and the whole process is repeated for the outside list, i.e., the polygon on top of the outside list is taken to be the clip polygon and the rest of the outside list is copied into the input list. The algorithm stops if the outside list is exhausted.

Polygon clipping algorithm

A flowchart of the algorithm, which clips a subject polygon against a clip polygon, is shown in Fig. 3.13. In the clipping process the clip polygon remains unchanged while the subject polygon might be fragmented into smaller polygons. An example, which illustrates the above clipping algorithm, is presented in Fig. 3.14a, 3.14b, and 3.14c. Fig. 3.14a shows the clip polygon, specified by a tuple of vertices $C_1, C_2, \ldots, C_6$, and a subject polygon specified by vertices $S_1, S_2, \ldots, S_8$. The
Figure 3.12. Flowchart of the Weiler-Atherton Algorithm
Figure 3.13. Flowchart of the Polygon Clipping Algorithm
Fig. 3.13. (cont'd)
Fig. 3.13. (cont'd)
Figure 3.14a. Polygon Clipping of a Clip Polygon Against a Subject Polygon

Figure 3.14b. Generation of the Inside List from the Modified Vertex List of Clip and Subject Polygon
Figure 3.14c. Generation of the Outside List from the Modified Vertex List of Clip and Subject Polygon
clip and subject polygon intersect at points \( I_1, \ldots, I_4 \). The clipping algorithm modifies the original vertex lists of clip and subject polygon by inserting the intersection points as false vertices in the appropriate places in the two lists. The algorithm generates two additional lists. The first list contains all those intersection points at which the clip polygon border passes to the left of the subject polygon border. If \( S = S_i, S_{i+1} \) and \( C = C_i, C_{i+1} \) denote edges of subject and clip polygon, which are directed in clockwise sense while traversing the corresponding polygon, then the first list is characterized by: \( S \times C > 0 \).

The remaining intersection points, which do not satisfy the above condition, are put on the second list. After this the inside and outside list are created as illustrated in Fig. 3.14b and Fig. 3.14c.

3.5 Object Manipulation and Data Structures

In this section the data formats at the different stages of object manipulation are described. Consider the viewing pipelines in Fig. 3.15 through which objects have to pass in order to be displayed on the screen. The kinematics operations which are applied to the object's master files to produce the corresponding world files are described in section 3.5.1. The data structures of the objects which pass through the hidden surface removal algorithms are discussed in section 3.5.2. The data structure for performing animation is described in section 3.5.3.

3.5.1 Kinematic Operations and Data Structures

In the present graphic software package a body segment, which is modelled by a convex polyhedron, is completely specified by its kinematic parameters and by a list of the vertices of the polyhedron.

The kinematic parameters are considered below. Consider Fig. 2.3 which shows the \( i \)th body segment in a body coordinate system with axes \( X_1,i, X_2,i, X_3,i \), relative to a world coordinate system with axes \( X_W, Y_W, Z_W \). The origin of the \( i \)th body is denoted by \( O_i \). The position vectors \( K_i \) and \( L_i \) point to the hinge point with the \((i+1)\)th body and the \((i-1)\)th body relative to \( O_i \). The orientation of the \( i \)th body is specified by Bryant angles [39].

In the current implementation the kinematic information of the \( i \)th body, which consists of its orientation, the position vectors to the hinge points with adjacent segments and the location of the origin \( O_i \), is stored as a pointer structure (in PASCAL).
The vertices of the polyhedron are stored in polygon list format [103, 106]. In this format each polygonal face of the polyhedron is redundantly described by a set of vertex pointers, which point to the coordinates of the respective vertices. The above mentioned polygon list of a body segment is denoted a master file, see Fig. 3.15. It is transformed to a world file by applying the respective kinematic transformation to which it is subjected.

3.5.2 Implementation of the Hidden Surface Removal Algorithms and Their Data Structures

In the current implementation of the hidden surface algorithms different data structures are used to optimize algorithm efficiency. The hidden surface algorithm for convex polyhedra is applied to the world (or object) space description of the respective polyhedron, which is the polygon list format, since this algorithm operates in object space, see Fig. 3.15. The concave polyhedra, which are studied in this thesis, are composed of a set of convex polyhedra. Hence, the above object space algorithm is applied to the individual convex polyhedra to eliminate all totally invisible surfaces in a presorting phase. The individual polyhedra are then entered to the clipping algorithm in three-dimensional space and transformed into image space. In the final phase either the Warnock or the Weiler Atherton algorithm can be applied, since both procedures operate in image space. If the Warnock algorithm is used all individual polyhedra, which are stored in different polygon lists, are copied into a single polygon list and entered to the hidden surface algorithm. Hence, the data structures essentially remain unchanged.

If the Weiler-Atherton algorithm is used the description of the individual polyhedra is changed from polygon list format to a matrix interconnection structures, in which the matrix elements represent vertices of the polyhedra's polygon mesh. The latter form is a non-redundant surface description. It increases the object's area coherence, since the boundary of polygon mesh serves as either clip or subject polygon, while interior polygons render the surface's texture. In fact, the change from polygon list to matrix interconnection format reduces the number of subdivisions, which are necessary to compute the final display. However, during each iteration stage, more intermediate calculations are necessary since a possible fragmentation of the subject polygon includes a recalculation of the surface's texture as well.

The implementation of the Warnock and Weiler-Atherton algorithm as a recursive procedure requires a considerable amount of memory space, if the final image is very complex. There are two ways of reducing the actual memory space of the algorithm. If the final image contains many objects which are evenly scattered on the screen area, then the screen is subdivided into smaller subareas, each of which is treated as a
Figure 3.15. Viewing Pipelines and Object Data Formats
separate hidden surface problem. However, if the objects are concentrated on a particular area of the screen and are evenly scattered in (z-) depth, the above approach is not necessarily effective. In this case the rectangular image volume is subdivided into several slabs along the z-axis. Each of the slabs is now treated as a different hidden surface problem. After this, the solutions of the intermediate hidden surface problems are used as inputs to the actual hidden surface problem.

3.5.3 The Data Structure for Animation Studies

Consider Fig. 3.15. At the end of the viewing pipeline segmented display files are created which contain the display code for the object's image. In the present graphics environment, see section 2.1, a total of 64 display files are permitted. A rough estimate of the data transmission speed from the computer to the graphics generator suggests that for animation purposes all display files have to be loaded into the memory of the graphics generator, i.e., it is not feasible to intermediatedly store data on the hard disk of the computer and transfer it to the graphics generator while performing animation. Therefore, the organization of these display files is such that all stationary segments of an object are stored in a single display file. This leaves a total of 63 display files for specifying intermediate positions of a movement cycle. The effect of a continuous, periodic motion is created by sequentially blanking one display file while unblanking the remaining nonstationary ones.

3.6 Convex Polyhedra

Generalized Truncated Cone:
Consider Fig. 3.16, which shows a generalized, truncated cone in a body reference system with axes X, Y, Z. The top and bottom surface of it are ellipses with major axes atop, btop, and abottom, bbottom. The program allows surface details of the side of the cone to be represented by a polygonal mesh with an arbitrary number of horizontal and vertical subdivisions.

Sphere:
Consider Fig. 3.16b, which shows a sphere. It is specified by its radius r. Its surface details are represented by a polygonal mesh with an arbitrary number of vertical and horizontal subdivisions.

Generalized Bar:
Consider Fig. 3.16c, which shows the generalized bar.
Figure 3.16. Convex Polyhedra: a) Generalized, Truncated Cone; b) Sphere; c) Generalized Bar
3.7 Concave Polyhedra

In what follows concave polyhedra are shown, which are composed of a set of convex polyhedra. A model of a humanoid, called ADAM, is described in section 3.7.1. The model of an assembly robot is presented in section 3.7.2.

3.7.1 Humanoid: ADAM

3.7.1.1 Program and Data Structures

The data, which is necessary to display the humanoid as a wire grid model, see Fig. 2.1, is separated into kinematic information and body segment description.

The kinematic information is stored in pointer structures which contain a record of the following form in PASCAL:

```pascal
Type pointer = ^ node;
node = record
  name: string;
  a-abs, a-rel: array;
  l, k, l-right, k-right,
  l-abs, k-abs, l-right-abs, l-left-abs: array;
  theta: array
  up, down, left, right: pointer
end;
```

where

- **name**: name of the body segment
- **a-abs**, **a-rel**: absolute and relative orientation of the segment
- **l**, ..., **l.left-abs**: absolute and relative position vectors to hinge points with adjacent segments
- **theta**: orientation of the body segment
- **up**, ..., **right**: points to the upper, lower, left and right record

After initialization the pointer structures have the configuration, which is shown in Fig. 3.17. The lower torso (LOTORSO), i.e., the hip, is chosen to be the reference segment. Then the individual segments exhibit the nested tree structure, which is shown in Fig. 3.18. Note, however, that the branch from LOTORSO to head is traversed in the opposite direction as compared to the remaining branches. Consider now Fig. 3.19, which shows an exploded view of ADAM. It shows the body (or master) coordinate systems of the individual segments and the respective position vectors which point to the hinge points with adjacent links.
Figure 3.17. Pointer Structures for Body Segment Description
Figure 3.18. Diagram of the Tree Structure of ADAM's Segments
Figure 3.19. Coordinate Systems of Body Segments and Definitions of Position Vectors
Figure 3.20. Program Dependencies in Software Module ADAM
All segments except LOTORSO and UPTORSO are connected to at most two adjacent segments; it suffices, therefore, to specify the position vectors $L_i, K_i$ or both $L_i$ and $K_i$, see section 3.4.2. The segments LOTORSO and UPTORSO are connected to more than two adjacent segments; hence, it is necessary to introduce additional position vectors, denoted $L$-LEFT, AND $L$-RIGHT, which specify the segments' interconnections.

As pointed out earlier, the body segment description, which contains the boundary polygon and the surface details of the polygonal mesh, which represents the body segments, are stored in either polygon list format or in matrix interconnection form, section 3.4.3.

Various programs that are necessary to compute the wire grid model of ADAM, Fig. 2.1, are shown in Fig. 3.20. Note that the basic graphic routines are not listed here.

3.7.1.2 Kinematics

The kinematics of the humanoid are explained below. The absolute locations of the respective origins are calculated first. The origin of the reference segment is set to zero:

$$O_{LOTORSO} = (0, 0, 0)$$

In the branch from LOTORSO to HEAD the origin $O_i$ of the $i^{th}$ segment relative to the previous segment, which is pointed out by pointer DOWN is:

$$O_i = O_{DOWN} + K_{DOWN} - L_i \; i \in \{LOTORSO, RR1B4, ..., P1B1, UPTORSO, HEAD\} \; (3.18)$$

Similarly in the remaining branches, the origin $O_j$ of the $j^{th}$ segment relative to the previous segment which is pointed at by pointer UP is:

$$O_j = O_{UP} + L_{UP} - K_j \; j \neq u \; (3.19)$$

The specification of the orientation of the individual body segment is considered next. The relative orientation specifies the orientation of the actual segment relative to the adjacent segment which is closer to the LOTORSO, while traversing the interconnection tree. The absolute orientation of a body segment is the orientation of it relative to the LOTORSO. Referring to Fig. 3.19 the absolute orientation $A_{ABS_i}$ of a segment is

$$A_{ABS_i} = \prod_{K \in P} A_{Rel_k} \; (3.20)$$

64
Figure 3.21. Kinematic Study of a Human Long Jump.
Figure 3.21. (cont.)
Figure 3.21. (cont.)
where the set $P$ contains all body segments which are encountered while traversing the interconnection tree from the actual segment to the LOTORSO.

The above formulas are implemented in a program called World Files, see Fig. 3.2.1. This program is called by a routine denoted recursive tree walker (RecTree Walker), which traverses the directed graph, Fig. 3.18 in the correct direction.

3.7.1.3 Animation

The graphics package supports animation capabilities as well. This is done in the following way. The program produces a database of up to 64 intermediate positions of the object under consideration. The display files, which contain the above mentioned positions, are loaded into the graphics generator, see section 3.2.1, and are sequentially blanked or unblanked to create a movie-like image. The user has the choice of creating the animation database by either linear interpolation or by cubic-spline interpolation. In the former one the user has to input the initial and terminal posture of the object. In the latter type of interpolation the user has to specify several intermediate (or pivot) postures of the object. The user can influence the speed of the movie-like image with the speed of blanking and unblanking of the display files. An animation study with the humanoid, which does a long jump, is shown in Fig. 3.21. The postures of humanoid shown in this figure represent the pivot positions for the cubic spline interpolation. Information on how to operate the program is provided in Appendix A3.

3.8 Summary

In this chapter a particular hardware configuration and a software structure for the kinematic analysis of multi-linkage system are described. For that purpose a computer graphics program is developed, which accepts the kinematic information of the multi-linkage system as input parameters, solves the associated hidden surface problem, and does a transformation from three-dimensional world space to a two-dimensional image space. Although the graphics package consists of a set of elementary computer graphics functions, such as perspective transformations, open and closed polygon clipping, it contains advanced algorithms, which perform hidden surface removal, as well. The graphics package is structured such that the objects are displayed on a vector display but can also be shown on a raster scan display in case of a failure of the first mentioned output device. Major contributions of this study in the area of computer graphics are the implementation and optimization of the above mentioned algorithms in a minicomputer environment and the definition of particular data structure which accomplishes this task. Finally, an animation study for the analysis of the human long jump is shown.
CHAPTER IV

CONTROLLABILITY ANALYSIS OF MECHANICAL SYSTEMS

4.1 Introduction

This chapter is focused on the generation of the equations of motion and the nonlinear controllability analysis of mechanical systems using symbolic algebraic manipulation (SAM) language, which is called MACSYMA. Special emphasis is placed on the derivation of the rotational equations of motion of systems of connected rigid bodies. Advantages and disadvantages of SAM systems are discussed below:

The use of symbolic manipulation systems for the generation of the dynamics of mechanical systems [89-95] and for the subsequent application to nonlinear controllability studies [126, 128] is well understood. High level symbolic manipulation programs like MACSYMA [100] or REDUCE [97] are especially suited for the above mentioned problems. Their suitability relies on direct commands for the following operations:

- arithmetic matrix operations like matrix addition, multiplication, computation of the inverse and the determinant, rank test and orthogonalization methods, where the matrix entries are allowed to be rational polynomials,
- differential calculus,
- pattern matching capabilities for the simplification of trigonometric expressions, for example,
- definition of user defined algebraic rules, which are useful, for example, for multiplying matrices whose entries are themselves matrices.

The potential advantage of symbolic manipulation programs is their capability of performing large numbers of algebraic operations in finite time. These operations can also be done by hand but the probability of making mistakes while performing these calculations, which also require a large amount of time, is fairly high. However, it should be pointed out here that certain checks have to be performed to verify the correctness of the results which are obtained from a SAM system. If, for
example, the dynamics of a mechanical system are derived symbolically [89-96] such a check can be done using both a Newton-Euler approach and a Lagrangian mechanics method, or if the controllability of a constrained system is analyzed, an upper bound could be provided for the dimension of the controllable subspace.

The special type of mechanical system, whose rotational dynamics are investigated here, are described next. The movements of these multi-body systems are specified by the dynamics of the center of mass of the composite system and by a set of rotational equations of motion [32, 39]. The decoupled form of a system description is useful in the study of spacecraft dynamics [64-71] or in analyzing the motion of animals or humans under free fall conditions [72-79]. In these studies the motion of the system's center of mass is influenced by gravity forces alone so that the resulting trajectory is a function of the corresponding initial conditions only. Hence, the dynamics of the center of mass are those of the point mass, which is an elementary problem in physics and is not further pursued here.

The inverse attitude control problem is defined for a planar, two-link system and is solved by transforming the corresponding ill-posed boundary value problem into a well-posed one under suitable assumptions. Digital computer simulations are performed to demonstrate that linear state feedback ensures that the nonlinear two link system is capable of tracking a prespecified reference trajectory. It is also shown that the above mentioned, partially controllable system does not exhibit numerical instabilities during simulations if the constraint, to which the system is subjected is a linear function of the state variables of the system.

The organization of this chapter is as follows. In section 4.2 the symbolic generation of the rotational equations of motion are discussed using two different approaches. In section 4.3 nonlinear control concepts, which are applicable to nonlinear systems with linear inputs, are outlined. The automation of the nonlinear controllability analysis is treated in section 4.4.

4.2 The Symbolic Generation of the Rotational Equations of Motion and the Computation of Their First Integrals

4.2.1 The Rotation Equations of Motion for N Rigid Links

In the following sections two methods are presented for deriving the rotational equations of motion for a system of N rigid bodies (Fig. 4.1). Fig. 4.1 shows N rigid bodies connected together at ideal ball-and-socket joints, which impose holonomic or simple nonholonomic connection constraints. Each body \( i \) \((i = 1, \ldots, N)\) has associated with it a body coordinate system \( \mathbb{B}_i \mathbb{C}_S \), which is aligned with the principal axes of body \( i \) and originates in the center of mass \( \mathbb{C}_M(i) \) of body \( i \). The local vectors \( \mathbf{K}_i \) and \( \mathbf{L}_i \) describe initial and terminal connection
Figure 4.1. A Collection of N Rigid, Interconnected Links and Their Respective Body Coordinate Systems and the Inertial Coordinate System.
point of link $i$ relative to $CM_i$. An inertial coordinate system $ICS$, relative to which the $N$ bodies move, is assumed. The axes of $B_iCS$ are denoted $(X_i, Y_i, Z_i)$ and those of the $ICS$ are $(X, Y, Z)$. The maps $A_i$

$A_i : B_iCS \rightarrow ICS$

and $B_i$, 

$B_i : B_iCS \rightarrow ICS$

map position and angular velocity vectors from the respective body to the inertial coordinate system. Bryant angles [39] are chosen for the maps $A_i$ and $B_i$. Sections 3.2.1 and 3.2.2 describe the two methods of deriving the rotational equations of motion for the system, shown in Fig. 4.1.

4.2.2 The Derivation of the Rotation Equations of Motion Using the Newton-Euler Method and the Decoupling Transformation

In the first method of deriving the rotation equations of motion, a free body analysis is done separately for each of the $N$ rigid bodies [34] to arrive at Eq. (2.1). Eq. (2.1) can be compactly written in vector-matrix form as:

$$U_1 \cdot \ddot{Z} = U_2 \Gamma + U_3 + U_4 M$$  

(4.1)

where

$$\ddot{Z} = (\ddot{X}_1, \ldots, \ddot{X}_n, \ddot{W}_1, \ldots, \ddot{W}_n)$$

$$\Gamma = (\Gamma_1, \ldots, \Gamma_{n-1})$$

$$M = (M_1, \ldots, M_{n-1})$$

The other quantities of Eq. (4.1) are specified in Appendix B1, Table B1. Constraint equations, which express the holonomic connection constraints of the $N$-bodies of Fig. 4.1, are differentiated twice to arrive at:

$$\ddot{Z} = U_5 \ddot{Y} + U_6$$  

(4.2)

with

$$\ddot{Y} = (\ddot{X}, \ddot{W}_1, \ldots, \ddot{W}_n)$$

A general expression for the matrices $U_5$ and $U_6$ is given in Appendix B1, Table 8.2. Eq. (4.2) is used to project Eq. (4.1) with base vector $Z$ to a reduced space with base vector $Y$ and to decouple rotational and translational motion [32]. The equations of the new, reduced system are:
\[ V_1 \ddot{Y} = V_2 + V_3 + V_4 + V_5 \, M \]  
(4.3)

with

\[ \begin{align*}
V_1 &= U^t_5 U_1 U_5 \\
V_2 &= -U^t_5 U_1 U_6 \\
V_3 &= U^t_5 U_3 = 0 \, !! \\
V_4 &= U^t_5 U_2 \\
V_5 &= U^t_5 U_4
\end{align*} \]

where \( t \) denotes the transpose operation. A general expression for the elements of equ. (4.3) is given in Appendix B2, Table 8.6. Equations (4.1), (4.2), and (4.3) are programmed in the symbolic language MACSYMA and evaluated for the case of three links in three-dimensional space. The corresponding program listing is shown in Appendix B1, Table 8.3. After the execution of the program the results, which consist of the unconstrained (Equ. (4.1), the projected equations of motion (equ. (4.3)), and the decoupling transformation (equ. (4.2)) are displayed in Appendix B1, Tables 8.4 and 8.5.

4.2.3 The Derivation of the Rotational Equation of Motion from the Interconnection Diagram

In the second method of deriving the rotational equations of motion the results, which were published by Braume [72] in 1906 and rediscovered independently by Hooker [64], and Roberson and Wittenburg [39], are directly applied to the system of Fig. 4.1. This method first proves the validity of the general expression of Eq. (4.1) and of the transformed Eq. (4.3). Second, the method provides a straightforward way to derive the elements of Eq. (4.3) without performing the projection, as outlined before. In what follows the notation of [51], is used, which is illustrated below on the example of a three link system, see Fig. 4.2. Fig. 4.2 shows a three-link interconnected system with centers of mass \( C_M_i \) \((i = 1, 2, 3)\) and the corresponding connection barycenters \( C_B_j \) \((i = 1, 2, 3)\) [39,64]. The barycenter \( C_B_j \) is the center of mass of the augmented body \( i \), which is formed by assuming that the masses of the bodies connected to it are concentrated in the respective interconnection joints. Hooker [64] further defines a set \( S \) which contains all unique labels of the bodies and a set \( F_\lambda \) which contains all unique labels of joints, which are connected to body \( \lambda \) \((\lambda = 1, 2, 3)\). Finally \( S_\lambda j \) is the set of labels of those bodies which are connected to body \( \lambda \) via joint \( j \) either directly or via a chain of intermediate bodies. For the three link system of Fig. 4.2. \( S, F_\lambda \) and \( S_\lambda j \) \((\lambda = 1,2,3 ; j = 2, 3)\) are

\[ S = \{1, 2, 3\} \]
Figure 4.2. A Three-Link Interconnected System with Centers of Mass $CM_i$ ($i = 1,2,3$) and Connection-Barycenters $CB_i$ ($i = 1,2,3$).
\[ F_1 = \{2\}, \ F_2 = \{2, 3\}, \ F_3 = \{3\} \]
\[ S_{12} = \{2, 3\}, \ S_{13} = \{3\} \]
\[ S_{22} = \{1\}, \ S_{23} = \{3\} \]
\[ S_{32} = \{1\}, \ S_{33} = \{1, 2\} \]

The vectors \( O L_{\lambda j} \) are vectors from \( CM_j \) to joint \( j \). Furthermore,
\[ O L_{\lambda j} = O L_{\lambda \mu} \text{ for any } \mu \in S_{\lambda j}, \text{ i.e.,} \]
\[ O L_{12} = O L_{13} ; \ O L_{32} = O L_{31} \]

The vectors \( D_{\lambda} \) and \( D_{\lambda H} \) describe the center of mass of body \( \lambda \), \( CM_\lambda \), and the hinge point of body \( \lambda \) to joint \( j \) relative to the barycenter of body \( \lambda \).
\[ D_{\lambda} = -\frac{1}{m} \sum_{\mu \neq \lambda} m_\mu O L_{\lambda \mu} \quad (4.4) \]
\[ D_{\lambda \mu} = D_{\lambda} + O L_{\lambda \mu} \]
\[ \text{with } m = \sum_{\mu=1}^{n} m_\mu \quad (4.5) \]

The main result of [64] is the set of rotational equations of motion:
\[ \phi_\lambda^* \dot{\omega}_\lambda + \sum_{\mu \neq \lambda} D_{\lambda \mu} \times D_{\mu \lambda} \times m \times \omega_\mu \]
\[ = \omega_\lambda \times \phi_\lambda^* - \omega_\lambda + \sum_{\mu \neq \lambda} D_{\lambda \mu} \times m \omega_\mu \times (\omega_\mu \times D_{\mu \lambda}) \]
\[ + T_\lambda + \sum_{j \in F_\lambda} T_{\lambda j}^H + D_{\lambda} \times F_\lambda + \sum_{\mu \neq \lambda} D_{\lambda \mu} \times F_\mu \quad (4.6) \]
\[ \lambda = 1, \ldots, n \]

with
\[ \phi_\lambda^* = \phi_\lambda + (m_\lambda D_{\lambda}^2 + \sum_{\mu \neq \lambda} m_\mu D_{\lambda \mu}^2) I \]
\[ - (m_\lambda D_{\lambda} D_{\lambda} + \sum_{\mu \neq \lambda} m_\mu D_{\lambda \mu} D_{\lambda \mu}) \]
\[ \phi_\lambda^* : \text{moment of inertia of body } \lambda \]
\[ \phi_\lambda^* : \text{moment of inertia of augmented body } \lambda \]
\[ \omega_\lambda : \text{angular velocity of body } \lambda \]
The rotational equations of motion, which are specified for each individual link $\lambda \in [1, n]$ in Eq. (4.6), can be more compactly written in vector matrix form, see Eq. (4.3). The elements of Eq. (4.3) are specified in Appendix B2, Table 8.6. Equation (4.3) is programmed in MACSYMA for the case of three (N=3) links. The program listing is shown in Appendix B2, Table 8.7 and the results, which are obtained by executing the above program on a DEC 20 computer, are shown in Appendix B2, Table 8.8. It can be shown that the two approaches of computing rotational dynamics for three links in three-dimensional space produce the same results, see Appendix B2. This strongly supports the validity of the rotational equations of motion for $N = 3$. 

$T_{\lambda}$: external torque on body $\lambda$ ($T_{\lambda} = 0$)

$F_{\lambda}$: external force on body $\lambda$ ($F_{\lambda} = 0$)

$T_{\lambda_j}^H$: constraint torque on body $\lambda$

$N$: number of links
4.2.4 First Integrals of the Rotational Equations of Motion

In this section it is shown that the angular momentum constraint is implicit in the rotational equations of motion, Eq. (4.6). Consider Eq. (4.3) whose rotational part is rewritten as a set of first order differential equations in a common reference frame, the ICS:

\[ v_1 \dot{w} = v_2 + v_4 + v_5 M \]  \hspace{1cm} (4.7)

where

\[ w = (w_1, \ldots, w_n)^T \]

\( v_i \) (i = 1, ..., 5): the portion of the augmented matrices \( V_j \) (i = 1, ..., 5), which correspond to rotational motion.

A first integral of Eq. (4.7) is a scalar function \( h(\theta, W, t) \), such that whenever the quantities \( \theta \) with \( \theta = (\theta_1, \ldots, \theta_n)^T \) and \( W \) satisfy Eq. (4.7), then \( h(\theta, W, t) \) is equal to a constant.

Lemma: The first integral of Eq. (4.3) is a vector function \( H(\theta, W, t) \) in the inertial coordinate system ICS.

\[ H(\theta, W, t) = \sum_{i=1}^{N} (A_i I_{ij} W_i + \sum_{j=1}^{N} (A_i D D_{ij} A_j D D_{ji} W_j)) \]

with \( I_{ij} = J_i - m_i D D_i^2 - \sum_{j=1}^{N} m_j D D_{ij} \) \hspace{1cm} (4.8)

Proof: It is shown below that the time derivative of \( H(\theta, W, t) \) is equal to the expression, which is obtained by adding the individual differential equations of Eq. (4.7) in the ICS. The quantity \( \dot{H} \) is:

\[ \dot{H}(\theta, W, t) = \sum_{i=1}^{N} (A_i W W_i I_{ij} W_i + A_i I_{ij} W_i) \]

\[ + \sum_{i=1}^{N} \sum_{j=1}^{N} m_i \left\{ A_i W W_i D D_{ij} A_j T A_j D D_{ji} W_j - A_i D D_{ij} W W_i A_j T A_j D D_{ji} W_j \right\} \]

\[ - A_i D D_{ij} A_j T A_j W W_j D D_{ji} + A_i D D_{ij} A_i T A_j D D_{ji} W_j \]
The sum of the equations of motion, Eq. (4.7) is:
\[ H(\hat{\theta}, \omega, T) = \sum_{i=1}^{N} (A_i \dot{W}_i \dot{I}_{ii} W_i + A_i I_{ii} \dot{W}_i) \]
\[ + m \sum_{i=1}^{N} \sum_{j=1}^{N} \left\{ A_i DD_{ij} A_i T A_j DD_{ji} \dot{W}_j - A_i DD_{ij} A_i T A_j WW_j^2 D_{ji} \right\} \]

An inspection of the above equations reveals that \[ H(\hat{\theta}, \omega, t) = H_1(\hat{\theta}, \omega, t) \]
if their difference vanishes
\[ \Delta = \hat{H} - \hat{H}_1 = \sum_{i=1}^{N} \sum_{j=1}^{N} m(A_i (WW_i DD_{ij} - DD_{ij} WW_i) A_i T A_j DD_{ji} W_j) = 0 \]  \hspace{1cm} (4.9)

For the purpose of simplifying the above proof let
\[ A_i WW_i = WW_i^* \]
\[ A_i DD_{ij} = DD_{ij}^* \]
\[ A_i W_i = W_i^* \]  \hspace{1cm} (i, j=1, ... N)
such that:
\[ \Delta = \hat{H} - \hat{H}_1 = \sum_{i=1}^{N} \sum_{j=1}^{N} m(WW_i^* DD_{ij}^* - DD_{ij}^* WW_i^*) DD_{ji} W_j^* \]  \hspace{1cm} (4.10)

Furthermore, the crossproduct operators appearing in Eq. (4.10) are designated by the "X" operator and the superscripts "*" are dropped from now on, so that
\[ WW_i^* + x W_i x \]
\[ DD_{ij}^* + x D_{ij} x \]

Hence \[ \Delta = \hat{H} - \hat{H}_1 = \sum_{i=1}^{N} \sum_{j=1}^{N} m(W_i x [D_{ij} x (D_{ji} x W_j)] - D_{ij} [W_i x (D_{ji} x W_j)]) \]  \hspace{1cm} (4.11)
Using elementary vector calculus Eq. (4.10) reduces to

\[ \Delta = \sum_{i=1}^{N} \sum_{j=1}^{N} (D_{ij} \cdot W_j) W_i \times D_{ji} - (D_{ij} \cdot D_{ji}) W_i \times W_j \]

\[- (W_i \cdot W_j) D_{ij} \times D_{ji} + (W_i \cdot D_{ji}) D_{ij} \times W_j \]

By symmetry

\[ \Delta = \sum_{i=1}^{N} \sum_{j=1}^{N} (D_{ij} \cdot W_j) W_i \times D_{ji} - (D_{ij} \cdot D_{ji}) W_i \times W_j \]

\[- (W_i \cdot W_j) D_{ij} \times D_{ji} + (W_i \cdot D_{ji}) D_{ij} \times W_j \]

\[+ (D_{ji} \cdot W_i) W_j \times D_{ij} - (D_{ji} \cdot D_{ij}) W_j \times W_i \]

\[-(W_j \cdot W_i) D_{ji} \times D_{ij} + (W_j \cdot D_{ij}) D_{ji} \times W_i = 0 \]

Hence \( \Delta = 0 \), since like terms in the above equation cancel each other.

q.e.d.

Next, it is shown that Eq. (4.8) is indeed the expression for the angular momentum. In Appendix A4, the angular momentum expression is derived from its definition [8] for N arbitrary rigid bodies. This expression is specified in Appendix B5 for N holonomic rigid bodies. In Appendix B6, the above mentioned expression is programmed in the symbolic language MACSYMA [100] and is evaluated for N = 2 and N = 3, see Appendix B, Tables 8.10 and 8.11. The angular momenta for N = 2 and N = 3 are the same expressions, which are obtained by summing up the columns of the vector \( v, w \) for N = 2 and N = 3, see Appendix B1, Tables 8.4, and 8.6. This proves the validity of Eq. (4.8) for N = 2 and N = 3.

4.3 The Automation of the Nonlinear Controllability Analysis

It is shown, see section 2.8, that in the nonlinear controllability analysis, one has, in general, to deal with a large number of algebraic manipulations due to the evaluation of Lie brackets and due to a set of rank tests in order to find the dimension of the controlled subspace. The theorems and definitions of nonlinear controllability concepts, which are stated in section 2.8, are already in a suitable form to be programmed in the SAM language MACSYMA. Two additional theorems are stated here, which prove to be very powerful in the automated nonlinear controllability analysis; they are denoted theorem 4 and theorem 5.
Theorem 4 [121]

The Lie bracket of two vector fields \( f \) and \( g \), denoted \([f, g]\), has tensor characters, i.e., it is invariant under a change of the local coordinate system. The original proof appears in [121]. An easy proof is provided here.

Proof: Let \( f = f(x) = f(y) \) and \( g = g(x) = g(y) \) be two vector fields on an \( n \)-dimensional manifold which are expressed in two different frames of references, which are

\[
x = (x_1, \ldots, x_n) \in \mathbb{R}^n
\]

and

\[
y = (y_1, \ldots, y_n) \in \mathbb{R}^n
\]

Let the coordinates of the second frame be expressible in those of the first frame, s.t.

\[
X = \rho(y)
\]

Then the Jacobian, denoted \( \rho*(y) = \frac{\partial x}{\partial y} \), which is assumed to be a nonsingular transformation is the following map:

\[
\rho*(Y) : X \rightarrow Y
\]

The Lie bracket of vector fields \( f \) and \( g \), denoted \([f, g]\), is invariant under a change of the frame of reference if:

\[
[f(Y), g(Y)] = [\rho*f(x), \rho*g(x)] = \rho*[f(x), g(x)]
\]

By definition \([f(x), g(x)] = \frac{\partial f(x)}{\partial x} g(x) - \frac{\partial g(x)}{\partial x} f(x)\)

\[
\rho*[f(x), g(x)] = \frac{\partial x}{\partial Y} (\frac{\partial f(x)}{\partial x} g(x) - \frac{\partial g(x)}{\partial x} f(x))
\]

\[
= \frac{\partial x}{\partial Y} \frac{\partial f(x)}{\partial x} g(x) - \frac{\partial x}{\partial Y} \frac{\partial g(x)}{\partial x} f(x)
\]

\[
= [\rho*f(x), \rho*g(x)]
\]

\[
= \frac{\partial f(\rho(Y))}{\partial Y} g(\rho(Y)) - \frac{\partial g(\rho(Y))}{\partial Y} f(\rho(Y))
\]

\[
= \frac{\partial f(Y)}{\partial Y} g(Y) - \frac{\partial g(Y)}{\partial Y} f(Y)
\]

\[
= [f(Y), g(Y)]
\]

q.e.d.
The next theorem deals with the span and the Lie brackets of an original and a triangularized set of vector fields, which is given in [126] without a proof.

**Theorem 5**

Consider a collection $S_1$ of vector fields

$$S_1 = \{f^1, \ldots, f^m\}$$

on an $n$-dimensional manifold and let the set $S_2$

$$S_2 = \{Gf_1, \ldots, Gf_m\}$$

denote the set which is obtained by triangularizing the set $S_1$. Then

$$\text{span} \{f^1, \ldots, f^m\} = \text{span} \{Gf_1, \ldots, Gf_m\}.$$ 

Furthermore, if

$$[f^i, f^j] \in \text{span} \{f^1, \ldots, f^m\}, i, j \in [0, m]$$

$$[Gf^i, Gf^j] \in \text{span} \{Gf_1, \ldots, Gf_m\} = \text{span} \{f^1, \ldots, f^m\}$$

**Proof:** Assume, without loss of generality, that the set $S_1$ does not need to be pivoted so that it can be triangularized by premultiplication of a lower triangular matrix $A(x)$ as follows, see [126] and section 5.3:

$$A(x) \begin{bmatrix} f^1 \\ f^2 \\ \vdots \\ f^m \end{bmatrix} = \begin{bmatrix} Gf^1 \\ Gf^2 \\ \vdots \\ Gf_m \end{bmatrix}$$

where $A(x) = \begin{bmatrix} a_{11}(x) & 0 \\ a_{21}(x) & \vdots \\ \vdots & \ddots \\ a_{n1}(x) & \cdots & a_{nn}(x) \end{bmatrix}$
x: state of the system

From Eq. (4.11) it follows that

\[ \text{span} \{ f^1, \ldots, f^m \} = \text{span} \{ G f^1, \ldots, G f^m \} \]

Let an arbitrary triangularized vector of Eq. (4.110 be expressed as:

\[ G f^i = \sum_{k=1}^{m} a_{ik}(x) f^k(x) \]

Then the Lie brackets of vectors \( G f^i \) and \( G f^j \), denoted \([G f^i, G f^j](x)\) is:

\[ [G f^i, G f^j](x) = \sum_{k=1}^{m} \sum_{l=1}^{m} \left[ a_{ik}(x) f^k(x), a_{jl}(x) f^l(x) \right] \]

\[ = \sum_{k=1}^{m} \sum_{l=1}^{m} \left[ a_{ik}(x) f^k(x), a_{jl}(x) f^l(x) \right] \text{ (by linearity)} \]

From [122, pg. 283]

\[ [G f^i, G f^j](x) = \sum_{k=1}^{m} \sum_{l=1}^{m} \left\{ a_{ik}(x) a_{jl}(x) [f^k(x), f^l(x)] \right\} \]

\[ + a_{ik}(x) (L f^k(x) a_{jl}(x)) \cdot f^l(x) \]

\[ - a_{jl}(x) (L f^l(x) a_{ik}(x)) \cdot f^k(x) \]

with \( (L f^l(x) a_{ik}(x)) = \langle \text{grad } a_{jl}(x), f^k(x) \rangle \)

Let \( a_{ik}(x) a_{jl}(x) = b_{ijk}(x) \)

\( a_{ik}(x) (L f^k(x) a_{jl}(x)) = c_{ijk}(x) \)

\( -a_{jl}(x) (L f^l(x) a_{ik}(x)) = d_{ijk}(x) \)

Then, it follows that

\[ [G f^i, G f^j](x) = \sum_{k=1}^{m} \sum_{l=1}^{m} \left\{ b_{ijk}(x)[f^k(x), f^l(x)] \right\} \]

\[ + c_{ijk}(x) \cdot f^l(x) + d_{ijk}(x) f^k(x) \]

(4.12)
Hence it follows that if
\[ [f^i, f^j](x) \in \text{span}\{f^1, \ldots, f^m\} \] then
\[ [Gf^i, Gf^j](x) \in \text{span}\{Gf^1, \ldots, Gf^m\} = \text{span}\{f^1, \ldots, f^m\}.\]

Theorem 4 implies that one can construct a state space representation for a system, in which the Lie brackets take a particular simple form. Theorem 5 implies that involutivity properties of a set of vector fields are invariant under a triangularization of the set.

The use of theorems 4 and 5 is illustrated below via an example of a planar, one link system, which is hinged to the origin see Fig. 4. When using Euler parameters its normalized equations of motion are:
\[ \dot{X} = f + gu \quad (4.13) \]
with \( X = (q_0, q_1, w)^T \)
\[ q_0 = \sin \frac{\theta}{2} \]
\[ q_1 = \cos \frac{\theta}{2} \]
\[ w = \dot{\theta} \]
\[ f = (q_0 \frac{w}{2}, -\frac{q_0 w}{2}, 2q_0 q_1)^T \]
\[ g = (0, 0, -1)^T \]

The above system is controllable if its structural parameters, see section 2.8, are:
\[ (n, n_a, n_s, n_l) = (3, 2, 2, 2) \quad (4.14) \]
Equation (4.14) implies that the maximal dimension of the controllable subspace is the number of states \( n = 3 \) less the number of constraints, which is one in Eq. (4.13).

A subset \( S \subset \{f, g\} \) is:
\[ S = \{ f_1, g_1, [f, g], [[f, g], f], [[f, g], g] \} \]
\[
\begin{pmatrix}
\frac{1}{2} q_1 w \\
-\frac{1}{2} q_0 w \\
2 q_0 q_1
\end{pmatrix}
= \begin{pmatrix}
0 \\
0 \\
-1
\end{pmatrix}
= \begin{pmatrix}
\frac{1}{2} q_1 \\
-\frac{1}{2} q_0 \\
0
\end{pmatrix}
= \begin{pmatrix}
0 \\
0 \\
1
\end{pmatrix}
= \begin{pmatrix}
0 \\
0 \\
0
\end{pmatrix}
\]
\[ (4.15) \]
Figure 4.3. A Planar, One Link System Which is Hinged to the Origin.
By applying theorem 4, for example, the set S, Eq. (4.15) can be transformed to a set $S^*$, which has as states $(\theta, \dot{\theta})$. Let $\phi^*(q_0, q_1, w)$ be the map from the old state to the new state of the system:

$$
\phi^*(q_0, q_1, w) : (q_0, q_1, w)^T \rightarrow (\theta, \dot{\theta})^T
$$

Hence

$$
S^* = \phi^* S = \left\{ \begin{bmatrix} \dot{\theta} \\
\sin\theta \end{bmatrix}, \begin{bmatrix} 0 \\
-1 \end{bmatrix}, \begin{bmatrix} 1 \\
0 \end{bmatrix}, \begin{bmatrix} 0 \\
1 \end{bmatrix}, \begin{bmatrix} 0 \end{bmatrix} \right\}
$$

(4.16)

The set $S^*$, which exhibits the same controllability properties as set S, allows an efficient determination of the structural parameters, since it contains two independent, constant vector fields. The set $S$, on the other hand, contains two independent, nonconstant vector fields, which implies that one has to check this for possible controllability defects. The structural parameters of the set $S^*$, denoted $(n^*, n_s^*, n_{sa}^*, n^*) = (2, 2, 2, 2)$, i.e., the system, Eq. (4.13) is controllable.

Alternatively, one can apply theorem 5 to Eq. (4.15) in order to produce a set $G^S$, which is formed from set S by triangularization. This is done by premultiplying S by a lower triangular matrix as follows:

$$
\begin{bmatrix} 1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix} [f, g] \\
f \\
g \\
[f, g], f \end{bmatrix} = \begin{bmatrix} G[f, g] \\
Gf \\
Gg \\
G[f, g], f \end{bmatrix}
$$

(4.17)
An inspection of Eq. (4.17) reveals that $\sup(n_s) = \sup(n_{sa}) = 2$. There are no controllability defects, however, since $q_0$ and $q_1$ can each vanish, but they do not do so simultaneously. Hence,

$$(n, n_s, n_{sa}, n_1) = (3, 2, 2, 2)$$

and the system is controllable. Note that the checks for the determination of involutivity are less complex for the triangularized set, Eq. (4.17), than for the original set, Eq. (4.15).

4.4 The Control of the Rotational Equations of Motion of Two Planar, Rigid Links

In what follows the methods which are described in sections 4.2 and 4.3 are applied to a planar, two-link system. Its rotational dynamics are derived from the interconnection diagram, section 4.2.3. The motion of the system is restricted to translations in the Y-Z plane and to rotations about the X-axis, i.e., entries in the system matrices, which do not correspond to the above type of motion are set to zero.

It is shown that the constant angular momentum constraint has quasi-holonomic character, i.e., it can be used to eliminate one state variable of the original system which corresponds to the loss of half a degree of freedom.

The controllability of the system is analyzed using linear and nonlinear controllability concepts. It is then shown that a simple, time-varying state feedback can be implemented to control the system about a nominal trajectory. The organization of section 4.4 is as follows. In section 4.4.1 the equations of motion of the system are specified and their controllability is analyzed. In section 4.4.2 a solution to the direct attitude control problem is proposed for two planar, rigid links, which are subject to a constant angular momentum. Numerical instabilities, which occur during the simulation of the unreduced, partially controllable system are discussed.

4.4.1 The Rotational Equations of Motion and the Controllability Analysis of Two Planar, Rigid Links

In this section the rotational equations of motion are specified for a planar two link system. Its controllability is analyzed using concepts of linear and nonlinear control theory. It is shown in Appendix B.3 that the rotational equations of motion of a rigid body system, with ideal torque actuators as inputs, can be written in the following form:
\[ \dot{X} = f(X) + gU \]  \hspace{1cm} (4.18)

where \( X \) and \( U \) are the state and the input to the system and \( f(X) \) and \( g \) are respectively the nonconstant drift term and the constant input matrix. The elements of Eq. (4.18) are derived for two planar, rigid links in Appendix C, starting from the equivalent system in three-dimensional space. Consider Fig. 4.4, which shows the above mentioned system consisting of two links, joined together at a common hinge point. The state of the system is

\[ X = (\theta_1, \theta_2, P_1, P_2)^T \]

\( \theta_i \): angle between link \( i \) and the vertical \((i = 1, 2)\)

\( P_i \): angular momentum of link \( i \) relative to the center of mass of the composite system \((i = 1, 2)\)

The drift term \( f(X) \) and the input matrix \( g \) are given below:

\[
f(X) = \begin{bmatrix}
\frac{1}{\det} (J_2 P_1 + D C_{21} P_2) \\
\frac{1}{\det} (J_1 P_2 + D C_{21} P_1) \\
\frac{DS_{21}}{\det^2} [P_1 P_2 (J_1 J_2 + D^2 C_{21}^2) + D C_{21}(J_1 P_2^2 + J_2 P_1^2)] \\
\frac{DS_{21}}{\det^2} [P_1 P_2 (J_1 J_2 + D^2 C_{21}^2) + D C_{21}(J_1 P_2^2 + J_2 P_1^2)]
\end{bmatrix}
\]

\[ g = (0, 0, 1, -1)^T \]

\( \det = J_1 J_2 - C_{21}^2 D^2 \)

\( C_{21} = \cos(\theta_2 - \theta_1) \)

\( S_{21} = \sin(\theta_2 - \theta_1) \)

\( D = \frac{m_1 m_2}{m} D_{12} D_{21} \)

\( m_i \): mass of link \( i \) \((i = 1, 2)\)

\( J_i \): inertia matrix of link \( i \) relative to the connection barycenter of link \( i \) \((i = 1, 2)\)

\( U \): torque between links 1 and 2.
Figure 4.4. Two Rigid, Planar Links and the Associated Connection Barycenters and Centers of Mass.
The quantities, which appear in Fig. 4.3, are already described in section 4.2.3 and are listed below:

- \( C_{M_i} \): center of mass of link \( i \)
- \( C_{B_i} \): connection barycenter of link \( i \)
- \( D_i \): position vector to \( C_{M_i} \) relative to \( C_{B_i} \)
- \( D_{ij} \): position vector of joint \( i \) relative to \( C_{B_j} \)
- \( O_{1ij} \): position vector of joint \( i \) relative to \( C_{M_j} \)
- \( J_i \): inertia matrix of body \( i \) relative to \( C_{B_i} \)

\((i, j = 1, 2, i \neq j)\)

Hence, the new, reduced system is:

\[
\dot{Y} = f_1(Y) + g_1U
\]

with \( Y = (\theta_1 \theta_2 P) \)

\[ P = P_1 \]

\[
f_1(Y) = \begin{bmatrix}
    J_2P_1 + DC_{21}(H_0 - P) \\
    J_1J_2 - C_{21}^2D_2 \\
    J_1(H_0 - P) + PDC_{21} \\
    J_1J_2 - C_{21}^2D_2 \\
    -DS_{21}P(H_0 - P)(J_1J_2 + D_2^2C_{21}^2) + DC_{21}[J_1(H_0 - P)^2 + J_2P_1^2] \\
    (J_1J_2 - C_{21}^2D_2)^2
\end{bmatrix}
\]

\[ g_1 = (0, 0, 1)^T \]

Note that is a relationship between the angular moments \( P_i \) and the angular velocities \( \theta_i \) (\( i = 1, 2 \)):

\[
\begin{bmatrix}
    P_1 \\
    P_2
\end{bmatrix} = \begin{bmatrix}
    J_1 & -DC_{21} \\
    -DC_{21} & J_2
\end{bmatrix} \begin{bmatrix}
    \dot{\theta}_1 \\
    \dot{\theta}_2
\end{bmatrix}
\]

(4.19)

It is assumed that the two links, see Fig. 4.3 are not subject to any external force besides the gravity force. Hence, the total angular momentum of the system, denoted \( H_0 \), remains constant, so that

\[ P_1 + P_2 = H_0 = \text{constant} \]

(4.20)
The nature of this constraint renders the rotational dynamics partially controllable. The maximum dimension of the controllable subspace, \( \text{sup}(\text{dim } C) \), is, due to the presence of the constraint:

\[
\text{sup}(\text{dim } C) = 4 \text{ (states)} - 1 \text{ (constraint)} = 3
\]

The angular momentum constraint, Eq. (4.20), can be implemented into the dynamic system, Eq. (4.18) by replacing \( P_2 \), for example, by:

\[
P_2 = H_0 - P_1
\]

After its implementation the constraint renders the third and fourth equation of Eq. (4.18) identical, so that one of them can be dropped from the dynamic formulation, see Appendix C.

The above mentioned constraint has quasi-holonomic character, since it can be employed to eliminate only one of the state variables instead of two [34], corresponding to the loss of half a degree of freedom [34]. The controllability analysis of Eq. (4.21) is considered below. In linear controllability analysis the system, Eq. (4.21), is approximated by the first terms of a Taylor series:

\[
\delta Y = A \delta Y + B \delta U \tag{4.22}
\]

with

\[
A = \frac{\delta f(Y)}{\delta Y}; B = g
\]

\( \delta Y \): incremental state
\( \delta U \): incremental input (torque)

After that, the controllability matrix \( C \) is formed, which is analyzed for possible controllability defects.

\[
C_1 = \{B, AB, A^2B\} \tag{4.23}
\]

The largest, potentially nonzero, (3 x 3) determinant of \( C \) is, see Appendix C:

\[
\det_2 = \frac{-S_{21}}{(J_1 J_2 - C_{21}^2 D^2)^3} \left\{ [(D J_2^2 - 2C_{21} D^2) J_2 - J_1^2 + 2C_{21} D^2 J_1] \\
+ H_0[(D J_1 + C_{21} D^2) J_2 + D J_1^2 - C_{21} D^2 J_1 - 2C_{21} D^3] \right\} \tag{4.24}
\]
which vanishes for the following cases:

a) The angles $\theta_1$ and $\theta_2$ are equal: $\theta_2 - \theta_1 = 0$

b) The two link system degenerates into a one link system.

c) At the points where $P = H_0 = 0$

d) The remaining cases where $\det C_1 = 0$

It follows that the linearized system is completely controllable on a three-dimensional subspace with the exception of some isolated points. The results of the nonlinear analysis are considered next. It is shown in Appendix C that the set

$$L_0 = \{g, [f, g], [[f, g], f]\}$$  \hspace{1cm} (4.25)

has structural parameters, see section 2.6,

$$(n_a, n_{sa}, n, n) = (3, 3, 3, 3)$$

i.e., it is (locally) controllable. The largest potentially nonzero determinant of $L_0$, denoted $\det$, is:

$$\det_1 = \frac{-2D_1 H_0 S_{21}}{(J_1 J_2 - C_2^2 D_2)^2}$$  \hspace{1cm} (4.26)

Controllability defects occur at the points where $C_3$ vanishes, which are at the points where:

a) the angular momentum $H_0$ vanishes: $H_0 = 0$

b) the angles $\theta_2$ and $\theta_1$ are equal: $\theta_2 - \theta_1 = 0$

c) the two link system degenerates into a one link system: $D = 0$

Hence the nonlinear system is (locally) controllable in the three-dimensional manifold with the exception of some isolated points. The determinants $\det_1$, Eq. (4.25), and $\det_2$, Eq. (4.23), which reflect the controllability defects of the nonlinear and the approximated linear systems have, in fact, very terms in common. The reason for this is given below. The controllability matrix of the approximated linear system is:

$$C = \{B, AB, A^2B\} = \{g, \frac{\partial f(Y)}{\partial Y} g, \frac{\partial f(Y)}{\partial Y} g\} Y = Y_0$$

Similarly, $L_0$ is:

$$L_0 = \{g, [f, g], [[f, g], f]\}$$

$$= \{g, \frac{\partial f}{\partial Y} g, \frac{\partial}{\partial Y} \left[\frac{\partial [f, g]}{\partial Y} f\right] - \left(\frac{\partial f}{\partial Y}\right)^2 g\} Y = Y.$$
Hence, both sets differ by the term

\[ \frac{\partial}{\partial Y} [\frac{\partial f}{\partial Y}] \]

which, in turn, yields distinctive values for the corresponding determinants.

Conclusion: The linear and nonlinear controllability analysis provide identical results with respect to the maximum dimension of the controllable subspace. The results are different, however, with respect to controllability defects. Hence, the nonlinear controllability does not provide much more information than the linear analysis.

4.4.2 A Solution to the Attitude Control Problem of Two Planar, Rigid Links, Whose Angular Momentum is Constant

In this section it is shown that a controller can be devised for the planar, two-body system, whose sole purpose is to make the system follow some prespecified nominal trajectories. This is a direct consequence of the fact that the above system is completely controllable on a constrained subspace, see section 4.4.2. Hence, the computation of the reference trajectories and the implementation of the controller are discussed below. Finally, digital computer simulations are presented to verify the functioning of the controller.

Computation of Reference Trajectories

The attitude control problem of the planar two-body system can be reformulated as the following two-point boundary value problem by using the notation of the previous section: Find \( \theta_1(t) \) and \( \theta_2(t) \), which satisfy the angular momentum constraint:

\[ H_0 = (J_1 - D C_2) \dot{\theta}_1 + (J_2 - C_2 D) \dot{\theta}_2 \]  (4.26)

subject to the following boundary conditions:

\[ \theta_1(t_0) = \theta_{10} \quad \theta_1(t_f) = \theta_{1f} \]
\[ \theta_2(t_0) = \theta_{20} \quad \theta_2(t_f) = \theta_{2f} \]

where \( t_0 \) and \( t_f \) are the time at the left and right boundary.

A solution to the direct attitude control problem is considered next. The form of the solution is that of [82 -84], with the exception that a controller is proposed and implemented here. Let the angle \( \theta_1(t) \)
denote external variable of the two-body system, which specifies its orientation relative to an inertial (or reference) frame. Furthermore, let \( \theta_{21}(t) = \theta_2(t) - \theta_1(t) \). The direct attitude control problem consists of prespecifying the internal variables and solving for the external ones by using the constraint equation, Eq. (4.26). In this process the ill-posed boundary value problem, Eq. (4.26), is, thus, converted to a well-posed one.

The internal variable \( \theta_{21}(t) \) is assumed to be of the form

\[
\theta_{21}(t) = a_0 \cos(a_1 t) + a_2
\]  

(4.27)

where the constants \( a_i \) (\( i = 0, 1, 2 \)) are specified later. By substituting Eq. (4.27) into Eq. (4.26), the latter one can be solved for the quantity \( \dot{\theta}_1(t) \):

\[
\dot{\theta}_1(t) = \frac{H_0 - (J_2 - C_{21} D) \dot{\theta}_{21}}{J_1 + J_2 - 2C_{21} D}
\]  

(4.28)

Hence

\[
\theta_1(t) = \int_{t_0}^{t_f} \dot{\theta}_1(t) \, dt + \theta_{10} = \int_{t_0}^{t_f} \frac{H_0 - (C_{21} D) \dot{\theta}_{21}}{J_1 + J_2 - 2C_{21} D} \, dt + \theta_{10}
\]  

(4.29)

Choosing the quantity \( P = P_1 \) to be:

\[
P = J_1 \dot{\theta}_1 - DC_{21} \dot{\theta}_2 = (J_1 - DC_{21}) \dot{\theta}_1 - DC_{21} \dot{\theta}_{21}
\]  

(4.30)

eliminates the need of computing bias torques, i.e., an inverse plant, as suggested in [32], is not necessary.

Controller Implementation

The controller is implemented as follows. Starting from the linearized system, Eq. (4.22), a closed loop system, denoted

\[
\delta \hat{Y} = (A + BK(t)) \delta Y
\]  

(4.31)

is formed which has eigenvalues at \((-20, -25, -30)\). This is accomplished by transforming Eq. (4.22) into the controllable canonical form, computing the transformed feedback gains \( K^*(t) \), and by applying an inverse transform to obtain the actual feedback gains \( K(t) \).
Digital Computer Simulations

A simulation diagram, illustrating the above mentioned design, is shown in Fig. 4.5. The body segment parameters are taken from [30] and are listed in Table 4.1. The left and right boundary values are [30]:

\[
\begin{align*}
\theta_1(0) &= -0.2 \text{ rad} & \theta_2(0) &= 0.12 \text{ rad} \\
\theta_1(1.5) &= \pi & \theta_2(1.5) &= \pi \\
\end{align*}
\]

with \( t_0 = 0 \text{ sec} \) and \( t_f = 1.5 \text{ sec} \).

The above boundary values determine the constants \( a_i \) \((i = 0, 1, 2)\) the internal variable \( \theta_2(t) \), Eq. (4.27)

\[
\theta_2(t) = a_0 \cos (a_1 t + a_2)
\]

with \( a_0 = \frac{1}{2} \{ [\theta_2(o) - \theta_1(o)] - [\theta_2(1.5) - \theta_1(1.5)] \} = 0.16 \)

\[
a_1 = 2.0944 ; T = 1.5 \text{ sec} \\
a_2 = \frac{1}{2} \{ [\theta_2(o) - \theta_1(o)] - [\theta_2(1.5) - \theta_1(1.5)] \} = 0.16
\]

The simulations, which are done here, are used to represent the diving motion of a human being [30] which is modelled as a planar, two-link system. The diver jumps from a nine meter elevated platform, travels two meters in a horizontal direction and lands in the water. The above assumptions determine the shape of the flight parabola of the center of mass of the composite system. Consider Fig. 4.6, which shows the flight parabola of a point mass. The point mass has an initial velocity vector \( v_0 \), which makes an angle \( \phi \) with the horizontal and lands at point \( R \), which lies on an inclined plane \( B \), which makes an angle \( \alpha \) with the horizontal. Recall that the range \( R \) of the point mass is

\[
R = \frac{2|v_0|^2 \sin(\beta - \alpha) \cos \beta}{g \cos^2 \alpha} \quad |v_0|: \text{magnitude of } v_0
\]

The maximum range \( R_{\text{max}} \) is reached at \( \beta = \frac{\pi}{4} + \frac{\alpha}{2} \)

\[
R_{\text{max}} = \frac{|v_0|^2}{g(1 + \sin \alpha)}
\]

A position vector \( r(t) \) which shows the location of the point mass at time \( t \) relative to the origin is:
Figure 4.5. Simulation Diagram for the Attitude Control of the Reduced, Planar Two Link System
Figure 4.6. Flight Parabola of a Point Mass Which Starts at the Origin and Lands on an Inclined Plane R.
Table 4.1. Body Segment Parameters for The Two-Link System

<table>
<thead>
<tr>
<th></th>
<th>Link No. 1</th>
<th>Link No. 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$/Kg-m²</td>
<td>1.30</td>
<td>2.35</td>
</tr>
<tr>
<td>$J$/kg-m²</td>
<td>4.0567</td>
<td>3.6567</td>
</tr>
<tr>
<td>$l$/meter</td>
<td>0.4</td>
<td>0.32</td>
</tr>
<tr>
<td>$k$/meter</td>
<td>0.6</td>
<td>0.28</td>
</tr>
<tr>
<td>$D$</td>
<td>-1.8667</td>
<td>-1.8667</td>
</tr>
</tbody>
</table>
\[ r(t) = (|v_0| \cos \beta) t \hat{y} + \left( (|v_0| \sin \beta) t - \frac{1}{2} gt^2 \right) \hat{z} \]  

where \( \hat{y} \) and \( \hat{z} \) are unit vectors in the horizontal and vertical directions. The time \( t_f \), at which \( r(t) = \beta \) is

\[ t_f = \frac{2|v_0| \sin(\beta - \alpha)}{g \cos \alpha} \]  

Using equations (4.33) - (4.35) the position \( r(t) \) corresponding to \( r(0) = 0\hat{y} + 0\hat{z} \) and \( r(1.5) = -g\hat{y} + 2\hat{z} \) is:

\[ r(t) = 1.3329t \hat{y} + (0.1464 - 4.0965)t \hat{z} \]  

In the simulations, which are shown next, it is assumed that the only variable, which the diver can influence, is the total angular momentum \( H_0 \). Since the quantity \( H_0 \) is determined by the corresponding initial conditions, the diver has to try to generate the particular value of \( H_0 \), denoted \( H_{opt} \), to achieve a correct dive. The diver over-rotates if

\[ H_0 > H_{opt} \]

and underrotates if

\[ H_0 < H_{opt} \]

The trajectories for the three cases of under-rotation, correct rotation, and over-rotation are shown in Fig. (4.7), Fig. (4.8), and Fig. (4.9). Fig (4.7) shows that the angles \( \theta_1(t) \) and \( \theta_2(t) \), the corresponding angular velocities, and the angular momentum are tracked very well with the controller shown in Fig (4.5). However, the time derivative of the angular momentum, denoted \( \dot{P} \), differs considerably from its reference value and so does the input torque \( U \) whose reference value is equal to zero. The above mentioned difference and the peak values of \( U \) and \( \dot{P} \) at the left and right hand sides of the respective graphs are due to the fact that at time \( t = 0 \) sec. the two links start moving relative to each other while at \( t = 1.5 \) sec. they stop to do so. Fig. (4.8) and (4.9) show similar characteristics. However, only the angles \( \theta_1(t) \) and \( \theta_2(t) \) are tracked very well. The corresponding angular velocities and the angular momentum \( P \) show large errors between actual and reference value at left and right boundaries while the tracking in between the boundaries is very well. A close inspection of the input torques reveals that they exhibit the same characteristic as the one in
Figure 4.7. Simulation Results for an Under-Rotated Dive,
\[ H_0 = 2 \text{ (KG METER}^2 \text{ RAD/SEC)} \]
Figure 4.7. (cont.)
Figure 4.7. (cont.)
Figure 4.7. (cont.)
Figure 4.8. Simulation Results for an Optimal Dive,

\[ H_0 = 24 \text{ (kg \cdot meter}^2 \cdot \text{rad/ sec)} \]
Figure 4.8. (cont.)
Figure 4.8. (cont.)
Figure 4.8. (cont.)
Figure 4.9. Simulation Results for an Over-Rotated Dive, 
$H_0 = 35 \text{ (KG * METER}^2 \text{ *RAD / SEC)}$. 
Figure 4.9. (cont.)
Figure 4.9. (cont.)
Figure 4.9. (cont.)
Fig. 4.7 with the exception that the peak values at the respective boundaries are more pronounced in Fig. (4.8) and Fig. (4.9). Therefore, the cause of these discrepancies is the same as discussed above. The relatively bad tracking properties of the controller, Eq. (4.31), suggests that it might be too slow, so that new eigenvalues, which are further to the left than the present ones, will probably improve the performance of the controller.

Simulation of the Unconstrained System

The simulation of the reduced system, Eq. (4.21) is shown in Fig. 4.7 - 4.9. It is known from the physics of the problem involved that the unreduced system, Eq. (4.18), behaves in exactly the same way as the reduced one. However, the simulation of the unreduced system, might pose numerical problems, see Section 2.4, since it is only partially controllable. It is shown below that the constraint has to be artificially stabilized during the simulation of the unreduced system, if it is being violated in this process.

Consider Fig. 4.10, which shows a diagram for the simulation of the unreduced system, Eq. (4.18). This configuration is used to re-simulate the optimal dive, which is shown in Fig. 4.8 with the trajectories for the reduced system. Note, from Fig. 4.10, that the feedback law is derived from the reduced state $Y = (\theta_1, \theta_2, P_1)^T$. The simulation of this particular configuration does not pose any numerical problems, since the angular momentum constraint, Eq. (4.20) is a linear function of the state and, consequently, is not violated during a numerical integration process. Hence, for this case the trajectories for the optimal dive are identical for the reduced and the unreduced system. As a consequence for the fact that the angular momentum constraint is not violated at all, the value of the angular momentum of the second link can be recovered from the knowledge of the angular momentum of the first link and the total angular momentum:

$$ P_2(t) = H_0 - \dot{P}_1(t) $$

$$ \dot{P}_2(t) = -P_1(t) $$

Next, it is shown that if the constraint is a nonlinear function of the state, numerical problems are encountered, so that the constraint
Figure 4.10. Simulation Diagram for the Attitude Control of the Unreduced, Planar Two Link System
has to be artificially stabilized. For that purpose a new state $Z$ is defined as follows:

$$Z = (\theta_1 \theta_2 \dot{\theta}_1 \dot{\theta}_2)^T$$

so that the new system is

$$\dot{Z} = f_2(Z) + g_2(Z) U$$

(4.37)

$$f_2(Z) = \begin{bmatrix}
\dot{\theta}_1 \\
\dot{\theta}_1 \\
\begin{bmatrix} J_1 & -DC_{12} \\
-DC_{12} & J_2 \end{bmatrix}^{-1} \\
\begin{bmatrix} D S_{12} & \dot{\theta}_2 \\
-D S_{12} & \dot{\theta}_1 \end{bmatrix}
\end{bmatrix}$$

$$g_2(Z) = \begin{bmatrix}
0 \\
0 \\
\begin{bmatrix} J_1 & -DC_{12} \\
-DC_{12} & J_2 \end{bmatrix}^{-1} \\
\begin{bmatrix} 1 \\
-1 \end{bmatrix}
\end{bmatrix}$$

The constraint is now a nonlinear function of the state

$$H_0 = (J_1 - D C_{12}) \dot{\theta}_1 + (J_2 - D C_{12}) \dot{\theta}_2$$

(4.38)

A simulation diagram for the above system is shown in Fig. 4.11. Note that the total angular momentum is that of the optimal dive, so that

$$H_0 = 24 \text{ (kg * rad * m}^2\text{/sec).}$$

Furthermore, the input torque $U$ for the new unreduced system, Eq. (4.37), is taken to be the same as that for the optimal dive, Fig. (4.8). The simulation results for the above mentioned system are shown.
Figure 4.11. Simulation Diagram for the Attitude Control of the Unreduced, Planar Two Link System with State $Z = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)^T$. 

$$\dot{Z} = f_2(Z) + g_2(Z) U$$

equ. (4.37)
Figure 4.12. Trajectories for the Optimal Dive Using as State $Z = (\theta_1, \theta_2, \dot{\theta}_1, \ddot{\theta}_1)^T$.  

115
Figure 4.12. (cont.)
Figure 4.12.  (cont.)
in Fig. 4.12. A close inspection of Fig. 4.12 reveals that the tracking of the angles \( \theta_1(t) \) and \( \theta_2(t) \) is fairly good, since deviations of the corresponding actual values from the reference trajectories are compensated by the much larger errors in actual and reference values, which occur in the respective angular velocities. Note also that the constraint \( H_0 \) is severely violated. It follows that this simulation can be of further use since the numerical instabilities which are introduced into the system through the process of numerical integration, do not cause a significant divergence of the angles \( \theta_1(t) \) and \( \theta_2(t) \) from the respective reference values. This is due to the fact that the dive has a finite duration which is specified by the time it takes to dive from a nine meter elevated platform into the water. If the set up shown in Fig. 4.11 were used to simulate the motion of a satellite in outer space corresponding to a not necessarily finite amount of time for simulation, the instabilities introduced by the above mentioned mechanism had a more severe effect. In the latter case additional, hypothetical inputs are necessary to maintain the constraint and assure stability of the system.

4.5 Summary

In this chapter the rotational equations of motion and their first integrals are derived for multi-linkage systems in three-dimensional space. It is shown that the above task can be considerably simplified through the use of a general-purpose, symbolic manipulation program. The rotational equations of motion are specified for a planar, two link system. Its controllability properties are analyzed using methods of linear and nonlinear control theory. It is shown that in the above nonlinear system the nonlinear controllability analysis does not provide much more information as the corresponding linear analysis. The automation of the nonlinear controllability analysis is discussed. The inverse attitude control problem is illustrated via a diving motion. A solution to this problem is implemented for the planar, two-link system. Digital computer simulations are performed for the unreduced system and for the reduced system, where the constant angular momentum constraint is implemented as a quasi-holonomic constraint. The unreduced system, which is only partially controllable, is not subject to numerical instabilities during digital computer simulation, if the constraint equation is a linear function of the state of the system. However, it is shown, that for nonlinear constraint equations in the state, hypothetical controls have to be devised whose sole purpose is to maintain the constraints and assure stability.
Chapter V

COMPUTATIONS OF HOLONOMIC AND SIMPLE NONHOLONOMIC CONSTRAINT FORCES OR TORQUES IN MULTI-BODY SYSTEMS

5.1 Introduction

Constraint forces or torques are an integral part of the study of multi-body systems. The terms simple nonholonomic constraints refers to those nonholonomic constraints whose corresponding constraint equations have the same structure as holonomic ones if both types of equations are differentiated twice with respect to time. In the multi-body systems, which are studied in this dissertation, holonomic constraints have to be enforced, if the individual bodies of the composite system are jointed together in a kinematic chain [40]. If these connections are such that rotational degrees of freedom are lost, constraint torques are engaged, which model these particular joint characteristics [40].

The constraint forces of a multi-body system serve different purposes. In robotic and locomotion systems, implementing, maintaining, and deliberately violating contact and support conditions require either on-line sensing of constraint forces or indirect computation of them. In this chapter three different methods of calculating constraint forces are compared with respect to computational complexity and accuracy, and computation time. In the first two methods, the nonlinear constraint equations and the system dynamic equations are combined such that the resulting set of equations is uniquely solvable for the constraint forces. In the third method, the mechanical system is embedded in a higher dimensional state space, chosen in such a way that the constraint equations are linear in the new state of the system. Consequently the computation of the constraint force simplifies. The complexity of the three methods and computation time for a particular computer can be inferred from an operation count. The accuracy of the individual method is estimated by providing an upper bound of the relative error and the relative residual of the computed constraint forces.

The organization of this chapter is given below. In section 5.2 and 5.3 the derivation and computation of constraint forces or torques, via the above mentioned methods, is outlined. Illustrative examples are given in section 5.4. Finally, the results of this study are discussed in section 5.5.
5.2 Derivation of Constraint Forces

The above three methods are applied in this section to a simple inverted pendulum for comparison purposes. Other examples can be found in references [16, 17, and 18]. Consider a simple planar pendulum (Fig. 5.1). Its center of mass is at point $0$ with coordinates $(x, y)$ and its contact point with the ground is $P$. The system has an input $U$ and is subject to holonomic contact forces $\Gamma_x$ and $\Gamma_y$. Let $X = [x, y, \theta]$, and let $m$ and $I$ be respectively the mass and moment of inertia of the pendulum about its center of gravity. The equations of motion are:

\[
\dot{u}X = u_2 \Gamma^T + u_3 + u_4 U
\]  \hspace{1cm} (5.1)

with

\[
u_1 = \text{diag}(m, m, I)
\]

\[
u_2 = \begin{bmatrix}
1 & 0 \\
0 & 1 \\
-k \cos \theta & k \sin \theta
\end{bmatrix}
\]

\[
u_3 = (0, -mg, 0)^T
\]

\[
u_4 = (0, 0, 1)^T
\]

The two holonomic constraints are:

\[x - k \sin \theta = 0\]

\[y - k \cos \theta = 0\]

The are differentiated twice to arrive at:

\[
u_2^T X = u_5
\]  \hspace{1cm} (5.2)
Figure 5.1. The Simple, Planar Pendulum Subject to Two Holonomic Contact Constraints
with \( u_5 = -k(\sin \theta, \cos \theta)^T \).

The first method consists of solving Eqs. (5.1, 5.2) simultaneously for the unknowns. In matrix form

\[
A_1 Y_1 = B_1 \tag{5.3}
\]

with

\[
A_1 = \begin{pmatrix} -u_2 & u_1^T \\ 0 & u_2^T \end{pmatrix}
\]

\[
B_1 = \begin{pmatrix} u_3 + u_4U \\ u_5 \end{pmatrix}
\]

\[
Y_1 = (\Gamma, X)^T
\]

The above system is solved by forming an augmented matrix \((A_1; B_1)\), and transforming it to \((I, A_1^{-1}B_1)\) by elementary row operations [57, 58] where the quantity \( I \) represents the identity matrix. This method requires less arithmetic operations than computing the inverse matrix \((A_1)^{-1}\) for solution [57, 58].

In the second method the accelerations are eliminated. The new system of equations is:

\[
A_2 Y_2 = B_2 \tag{5.4}
\]

with

\[
A_2 = u_2^T (u_1)^{-1} u_2
\]

\[
B_2 = u_5 - u_2^T (u_1)^{-1} (u_3 + u_4U)
\]

\[
Y_2 = \Gamma
\]

In the above derivation, the matrix \( A_2 \) is positive definite [40], and therefore invertible. The dimension of \( A_2 \) is equal to the number of constraints \( m \).

In the third method, the original system is changed to the system depicted in Fig. 5.2. The state vector of the system is \( W = (X_1, Y_1, \theta, \dot{X}_1, \dot{Y}_1, \dot{\theta})^T \), where \( X_1 \) and \( Y_1 \) are the coordinates of the contact point \( P \). The equations of the new system are:

\[
v_1 \ddot{Z} = v_2 \Gamma + v_3 + v_4U + v_5 \tag{5.5}
\]
with

\[
\begin{align*}
\mathbf{v}_1 &= \begin{bmatrix} m & 0 & mkC \\ 0 & m & -mkS \\ mkC & -mkS & 1 + mk^2 \end{bmatrix} \\
\mathbf{v}_2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \\
\mathbf{v}_3 &= \mathbf{u}_3 \\
\mathbf{v}_4 &= \mathbf{u}_4 \\
\mathbf{v}_5 &= (kS \ddot{\theta}^2, kC\dot{\theta}^2, 0)^T \\
\ddot{\mathbf{z}} &= (\ddot{x}_1, \ddot{y}_1, \ddot{\theta})^T \\
\mathbf{S} &= \sin \theta; \quad \mathbf{C} = \cos \theta
\end{align*}
\]

The constraints are:

\[
\begin{align*}
\dot{x}_1 &= 0 \\
\dot{y}_1 &= 0
\end{align*}
\]  
(5.6)

The constraint forces can be calculated very efficiently, from

\[
\begin{align*}
A_3 \mathbf{y}_3 &= B_3 
\end{align*}
\]  
(5.7)

with

\[
A_3 = \begin{bmatrix} -\mathbf{v}_2 & \mathbf{v}_1 \\ 0 & -\mathbf{v}_2^T \end{bmatrix} \\
B_3 = \begin{bmatrix} \mathbf{v}_3 + \mathbf{v}_4 \mathbf{u} + \mathbf{v}_5 \\ 0 \end{bmatrix}
\]

Eq. (5.7) is similar to Eq. (5.3) but offers considerable computational
Figure 5.2. The Pendulum Embedded in $0, x, y$, Space.
advantages. Matrix $A_3$ is already of triangular form, so that its
inverse can be computed without using the forward iteration. The lower
portion of the vector $B_3$ is equal to zero, such that during back
substitution, additional computational savings occur. Eqs. (5.1 and
5.5) are related by the nonsingular transformation [6]:

$$\ddot{X} = D_1\ddot{Z} + D_2$$ (5.8)

with

$$D_1 = \begin{bmatrix} 1 & 0 & kC \\ 0 & 1 & -kS \\ 0 & 0 & 1 \end{bmatrix}$$

$$D_2 = \begin{bmatrix} -kS\dot{\theta}^2 \\ -kC\dot{\theta}^2 \\ 0 \end{bmatrix}$$

Using Eq. (5.8) the matrices of Eq. (5.5) are:

$$v_1 = D_1^T u_1 D_1$$
$$v_2 = D_1^T u_2$$
$$v_3 = D_1^T u_3$$
$$v_4 = D_1^T u_4$$
$$v_5 = -D_1^T D_2$$

5.3 Computation of Constraint Forces

The last step in each of the above three derivations is to solve a
set of linear equations in the forces of constraint $\Gamma$. The methods to be
used for this purpose are Gaussian elimination with partial pivoting
[57-63] - a series of row operations consisting of two parts: the
forward elimination and the back substitution. Consider the equation

$$Ay = B$$ (5.9)

During forward iteration, the matrix $A$ is transformed to an upper
triangular matrix $R$, according to:

$$R = L^{-1} PA$$, (5.10)
The multipliers, which perform this transformation, form a lower triangular matrix \( L \), which has 1's along its diagonal. The matrix \( P \) is:

\[
P = P_{n-1} P_{n-2} \ldots P_1,
\]

where the \( P_i \) (\( i = 1, \ldots, n-1 \)) are Frobenius matrices \([31]\), which accomplish a row permutation of matrix \( A \) in the \( i \)th iteration step. During back substitution the diagonal elements of \( R \), which are \( r_{ii} \) (\( i = 1, \ldots, p \)), are normalized by premultiplication of \( R \) by \( 0 \):

\[
0 = \text{diag} \left( \frac{1}{r_{n}}, \ldots, \frac{1}{r_{pp}} \right).
\]

The solution to Eq. (5.9) is:

\[
Y = (OR)^{-1}(OL^{-1})B \quad (5.11)
\]

Due to the form of matrices \((OR)\) and \( L \), their inverses are found by changing the sign of the off-diagonal elements. The operations involved in calculating Eq. (5.11) are listed in Table 5.1. The searches for a maximum, which are listed in Table 5.1, are done during pivoting.

5.3.1 Computational Complexity

The number of additions and multiplications for all three methods as a function of the \( n \) dynamic and \( m \) constraint equations of the system are derived here. These formulas are based on the Gauss elimination technique and are shown in Tables 5.2 and 5.3 for two cases: diagonal and nondiagonal inertia matrices. These formulas are based on Eqs. (5.3), (5.4), and (5.7) for the general case. The expressions in Tables 5.2 and 5.3 for the third method contain an operation for solving Eq. (5.7), provided that the corresponding matrices \( v_i \) (\( i = 1, \ldots, 5 \)) are known. However, if the matrices of the old system, Eq. (5.1), are known only, they can be transformed to those of the new one, Eq. (5.5), using the transformation Eq. (5.8). Consequently, a separate operation count is provided for the above transformation.

The computational savings, due to the blocks of zero elements in matrices \( A_1 \) of Eq. (5.3) and \( B_3 \), in Eq. (5.7), are accounted for in Table 5.3. Additional computational savings due to a diagonal inertia matrix \( u_1 \), in Eq. (5.3), (5.4), and (5.8) are reflected in Table 5.2. The remaining matrices, however, are assumed to contain only nonzero entries.

The computational complexity of the three methods is summarized in Fig. 5.3. In Fig. 5.3 the number of additions and multiplications of the three methods is shown as a function of the \( n \) dynamic and \( m \) constraint equations of the system. Fig. 5.3a and b represent a system with few constraints, where \( m = 0.2 \times n \). Fig. 4c and d show a system with a medium number of constraints, s.t.,

126
m = 0.5 * n

Figs. 5.3 e and f represent a system with many constraints, where

m = 0.9 * n

Figs. 5.3a, c, and e are graphed based on the expressions of Table 5.2, Figs. 5 b, d, and f are based on Table 5.3. An interpretation of Fig. 5.3 reveals the applicability of the three methods:

**Method I (Smith):**
This method is efficient if the system is known in the form specified by Eq. (5.1) and if the inertia matrix \( u_1 \) is non-diagonal.

**Method II (Hemami):**
This method is efficient if the system is known in the form specified by Eq. (5.1) but with a diagonal inertia matrix.

**Method III (Kane):**
This method requires the least amount of computations, if the system is known in the form specified by Eq. (5.5). If the old system, Eq. (5.1), is known only, this method is still efficient for a system with many constraints.

### 5.3.2 Estimates of Round-Off Errors

The following expressions are upper bounds of the round-off errors [57 -63] which occur by applying the method of Gaussian elimination for solving a system of linear equations. These round-off errors are due to the representation of real numbers as floating point numbers with finite mantissa in the memory of a computer.

Consider a linear system, Eq. (5.9), with a true solution \( Y \) and a computed solution \( Y^* \), which is obtained by applying the method of Gaussian elimination. It is assumed that the matrices \( A \) and \( B \) in Eq. (5.9), are expressed in exact floating point numbers and that no overflow nor underflow occurs during the solution of (5.9). Let

\[
||C|| = \sum_{i=1}^{n} |c_i| \tag{5.12}
\]

denote the norm of a vector \( c \) with elements \( c_i \) \( (i = 1, \ldots, n) \).

Then the relative residual is:

\[
\frac{||B - AY^*||}{||A|| \cdot ||Y^*||} < \rho \beta^{-t} \tag{5.13}
\]
Table 5.1. Computational Complexity of Calculating an Inverse Matrix and Solving a System of Simultaneous Equations.

<table>
<thead>
<tr>
<th></th>
<th>Number of Additions</th>
<th>Number of Multiplications</th>
<th>Other operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>solving a system of p simultaneous</td>
<td>$\frac{p^3}{3} + \frac{p^2}{2} - \frac{5p}{6}$</td>
<td>$\frac{p^3}{3} + \frac{3}{2} p^2 - \frac{5}{6} p$</td>
<td>$\frac{1}{2} (p^2 + p)$ searches for a maximum</td>
</tr>
<tr>
<td>equations via Eq. (5.11)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>computing the pxp inverse matrix via</td>
<td>$\frac{2}{3} p^3 - p^2 + \frac{1}{3} p$</td>
<td>$\frac{2}{3} p^3 + p^2 + \frac{1}{3} p$</td>
<td>$\frac{1}{2} (p^2 + p)$ searches for a maximum</td>
</tr>
<tr>
<td>$A^{-1} = (OR)^{-1} (QL^{-1}p)$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5.2. Computational Complexity of Calculating Constraint Forces Assuming the Inertia Matrix is Diagonal.

<table>
<thead>
<tr>
<th>Calculation of Constraint Forces</th>
<th>Number of Additions</th>
<th>Number of Multiplications</th>
<th>Other operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Via Method I Eq. (5.3) (Smith)</td>
<td>$m^2n + n^2m$</td>
<td>$m^2n + n^2m$</td>
<td>$rac{1}{n}\left[(n+m)^2 - n+m\right]$</td>
</tr>
<tr>
<td></td>
<td>$+ \frac{1}{2} (n+m)^2 + \frac{1}{2} n^2$</td>
<td>$+ \frac{3}{2} (n+m)^2 + \frac{n^2}{2}$</td>
<td>$-m^2 - 1$</td>
</tr>
<tr>
<td></td>
<td>$-n - \frac{m}{2}$</td>
<td>$- n - \frac{m}{2}$</td>
<td>$- \frac{1}{2}[ (n-m)^2 - n+m \right]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>searches</td>
</tr>
<tr>
<td>Via Method II Eq. (5.4) (Hemami)</td>
<td>$\frac{1}{3} m^3 + \frac{m^2n}{2} - \frac{m^2}{2}$</td>
<td>$\frac{1}{3} m^3 + \frac{m^2n}{2} + \frac{3}{2} m^2$</td>
<td>$\frac{1}{2} (m^2 + m) - 1$</td>
</tr>
<tr>
<td></td>
<td>$- \frac{11}{6} m + mn$</td>
<td>$+ 2 mn - \frac{5}{6} m$</td>
<td>searches</td>
</tr>
<tr>
<td>Via Method III Eq. (5.7) (Kane)</td>
<td>$\frac{1}{3} (n-m)^3 + \frac{1}{2} (n^2 - n)$</td>
<td>$\frac{1}{3} (n-m)^3 + \frac{1}{2} (n-m)^2 + \frac{1}{2} [(n-m)^2 + n+m]$</td>
<td>$+(n-m)(n-m-1)m - 1$</td>
</tr>
<tr>
<td></td>
<td>$+ (n-m)(n-m-1)m$</td>
<td>$+ \frac{3}{2} n^2 - m^2 + \frac{5}{6} m$</td>
<td>searches</td>
</tr>
<tr>
<td></td>
<td>$- \frac{1}{3} (n-m)$</td>
<td>$- \frac{4}{3} n$</td>
<td></td>
</tr>
<tr>
<td>Transformation of state vector Eq. (5.8)</td>
<td>$n^3 + n^2 - 2n$</td>
<td>$n^3 + 3n^2$</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 5.3. Computational Complexity of Calculating Constraint Forces Assuming the Inertia Matrix is Nondiagonal.

<table>
<thead>
<tr>
<th>Calculation of Constraint Forces</th>
<th>Number of Additions</th>
<th>Number of Multiplications</th>
<th>Other operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Via Method I Eq. (5.3) (Smith)</td>
<td>$\frac{1}{3}(n+m)^3 - \frac{1}{3}m^3$</td>
<td>$\frac{1}{3}(n+m)^3 - \frac{1}{3}m^3$</td>
<td>$\frac{1}{n}[(n+m)^2 - n-m]$</td>
</tr>
<tr>
<td></td>
<td>$+ \frac{1}{2}(n+m)^2 - \frac{5}{6}n - \frac{m}{2}$</td>
<td>$+ \frac{3}{2}(n+m)^2 - \frac{5}{6}n$</td>
<td>$-m^2 -1$ searches</td>
</tr>
<tr>
<td>Via Method II Eq. (5.4) (Hemami)</td>
<td>$\frac{1}{3}(m+n)^3 + \frac{1}{3}n^3-n^2$</td>
<td>$\frac{1}{3}(m+n)^3 + \frac{1}{3}n^3+n^2$</td>
<td>$\frac{1}{2}(m^2 + m) - 1$ searches</td>
</tr>
<tr>
<td></td>
<td>$- \frac{m^2}{2} + \frac{1}{3}n - \frac{11}{6}m$</td>
<td>$+ \frac{3}{2}m^2 + mn + \frac{n}{3}$</td>
<td></td>
</tr>
<tr>
<td>Via Method III Eq. (5.7) (Kane)</td>
<td>$\frac{1}{3}(n-m)^3 + \frac{1}{2}(n^2-n)$</td>
<td>$\frac{1}{3}(n-m)^3 + \frac{1}{2}(n-m)^2$</td>
<td>$\frac{1}{2}[(n-m)^2+n+m]$</td>
</tr>
<tr>
<td></td>
<td>$(n-m)(n-m-1)m$</td>
<td>$+(n-m)(n-m-1)m$</td>
<td>$-1$</td>
</tr>
<tr>
<td></td>
<td>$- \frac{1}{3}(n-m)$</td>
<td>$+ \frac{3}{2}n^2 - m^2 + \frac{5}{6}m$</td>
<td>$\frac{11}{6}n$ searches</td>
</tr>
<tr>
<td>Transformation of state vector</td>
<td>$2n^3 - 2n$</td>
<td>$2n^3 + 2n^2$</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.3. Comparison of the Operation Count of Smith's, Kane's, and Hemami's Method of Computing Constraint Forces.

- + - + - + - : Method I (Smith)
- x - x - x - : Method II (Hemami)
- # - # - # - : Method III (Kane)
- 0 - 0 - 0 - : Method III Including a Transformation of the Basis

a) Few Constraints (M = 0.2 N), Diagonal Inertia Matrix
b) Few Constraints, Non-diagonal Inertia Matrix
c) Medium Number of Constraints (M = 0.5 N), Diagonal Inertia Matrix
d) Medium Number of Constraints, Non-diagonal Inertia Matrix
e) Many Constraints (M = 0.9 N), Diagonal Inertia Matrix
f) Many Constraints, Non-diagonal Inertia Matrix
and the relative error is:

\[ \frac{|Y - Y^*|}{|Y^*|} \leq p \text{ and } \beta - t \]  \tag{5.14}

where:

- \( \beta \): base of the floating point number system
- \( t \): number of significant digits in the mantissa of the floating point number
- \( \rho \): \( 0 < \rho < \beta \) in most cases \[24\]

\[ ||A|| = \max_Y \frac{||AY||}{|Y|} \]  \tag{5.15}

and

\( \text{cond } (A) \): condition number of A

5.4 Illustrative Examples

In this section the three methods are compared for three illustration examples.

5.4.1. Planar One Link

The first example is the planar one link of Fig. 5.1. The constraint forces, in analytical form, are

\[ \Gamma_x = a_1 S\dot{e}_2^2 + a_2 SC + a_3 CU \]
\[ \Gamma_y = a_1 C\dot{e}_2^2 + a_4 + a_2 C^2 - a_3 SU \]  \tag{5.16}

with

\[ a_1 = -mk \]
\[ a_2 = k^2mg \frac{1}{1 + mk^2} \]
\[ a_3 = \frac{mk}{1 + mk^2} \]
\[ a_4 = \frac{\text{Im}g}{1 + mk^2} \]
Table 5.4. Operation Count for the Methods of Calculating Constraint Forces for the Planar One Link.

<table>
<thead>
<tr>
<th>Calculation of Constraint Forces</th>
<th>Additions</th>
<th>Multiplications</th>
<th>Searches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytic Method Eq. (5.16)</td>
<td>5</td>
<td>12</td>
<td>-</td>
</tr>
<tr>
<td>Initializing $u_2$, $u_5$ [Eq. (5.1, 5.2)]</td>
<td>-</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>Method I (Smith) ($m = 2, n = 3$)</td>
<td>43</td>
<td>68</td>
<td>9</td>
</tr>
<tr>
<td>Method II (Hemami):</td>
<td>15</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>Method III (Kane) +</td>
<td>3 + 30</td>
<td>8 + 54</td>
<td>0</td>
</tr>
<tr>
<td>(transformation of base vector)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initializing $v_1$, $v_5$ [Eq. (5.5)]</td>
<td>3</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>and $B_3$ [Eq. (5.7)]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Method III (Kane)</td>
<td>3</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.5. Upper Bounds for Round-Off Errors for Computed Constraint Forces of the Planar One Link

Simulation: The numerical values of the parameters for this example are 
\( m = 49 \text{ kg}; \ I = 2.35 \text{ kg m}^2; \ l = 0.312 \text{ m}; \ \theta = 0^\circ, 45^\circ; \ \beta = 2. \)

<table>
<thead>
<tr>
<th>Calculation of Constraint Forces via the Method of</th>
<th>Single Precision (t=24)</th>
<th>Double Precision (t=54)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith (Eq. (5.3))</td>
<td>(&lt; 11574 \times 10^{-7})</td>
<td>(&lt; 2694 \times 10^{-16})</td>
</tr>
<tr>
<td>Hemami, (Eq. (5.4))</td>
<td>(&lt; 408 \times 10^{-7})</td>
<td>(&lt; 95 \times 10^{-16})</td>
</tr>
<tr>
<td>Kane, (Eq. (5.7))</td>
<td>(&lt; 8 \times 10^{-7})</td>
<td>(&lt; 2 \times 10^{-16})</td>
</tr>
</tbody>
</table>
where $S$ and $C$ respectively mean $\sin \theta$ and $\cos \theta$. The operations, which are necessary to calculate $\Gamma_X$ and $\Gamma_Y$ via Eq. (5.16), are counted assuming that the constants $a_i$ ($i = 1, \ldots, 4$) are known and that repeated terms are computed only once, i.e., they are stored in memory. An operation count for calculating constraint forces using the numerical methods, section 5.3.1 is performed based on the expressions of Table 5.2. The results are shown in Table 5.4. The round off errors are given in Table 5.5; it is assumed here that the numbers which are used to compute the constraint forces are represented either in single (24 bit mantissa) or double (54 bit mantissa) precision format in a digital computer.

5.4.2 Planar Double Pendulum

A planar double pendulum is considered next (Fig. 5.4). The corresponding centers of mass are at points $O_i$ ($i = 1, 2$) with coordinates $(x_i, y_i)$ ($i = 1, 2$). The hinge points are denoted by $P_i$ and $P_i'$ ($i = 1, 2$). The distances $P_iO_i$ and $O_iP_i'$ are called $k_i$ and $l_i$ ($i = 1, 2$). The angle $\theta_i$ ($i = 1, 2$) is measured between the line $P_iO_i$ and the vertical. The system has inputs $u_1$ and $u_2$ and is subject to holonomic forces $\Gamma_1 = (\Gamma_{1x}, \Gamma_{1y})^T$ and $\Gamma_2 = (\Gamma_{2x}, \Gamma_{2y})^T$, which constrain the translational motion of the system. The matrices of the dynamic and constraint equations, Eq. (5.1) and Eq. (5.2) which are denoted $u_i$ ($i = 1, \ldots, 5$); are given below:

$$u_1 = \text{diag}(m_1, m_1, m_2, m_2, l_1, l_2)$$

$$u_2 = \begin{pmatrix}
+1 & 0 & -1 & 0 \\
0 & +1 & 0 & -1 \\
0 & 0 & +1 & 0 \\
0 & 0 & 0 & +1 \\
-k_1C_1 & +k_1S_1 & -l_1C_1 & +l_1S_1 \\
0 & 0 & -k_2C_2 & +k_2S_2
\end{pmatrix}$$

$$C_i = \cos \theta_i; \quad S_i = \sin \theta_i \quad (i = 1, 2)$$

$$u_3 = (0, -m_1g, 0, -m_2g, 0, 0)^T$$

$$u_4 = \begin{pmatrix}
-0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 1 & -1
\end{pmatrix}^T$$
Figure 5.4. Double Pendulum Subject to Holonomic Constraints.
Table 5.6. Operation Count for the Methods of Calculating Constraint Forces for the Planar Two Link.

<table>
<thead>
<tr>
<th>Calculation of Constraint Forces</th>
<th>Additions</th>
<th>Multiplications</th>
<th>Searches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytic Expression Eq. (5.21)</td>
<td>16</td>
<td>28</td>
<td>-</td>
</tr>
<tr>
<td>Initializing ( u_2, u_5 ) [Eq. (5.1, 5.2)]</td>
<td>2</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>Method I (Smith) ((m = 4, n = 6))</td>
<td>355</td>
<td>455</td>
<td>27</td>
</tr>
<tr>
<td>Method II (Hemami)</td>
<td>356</td>
<td>488</td>
<td>9</td>
</tr>
<tr>
<td>Method III (Kane) + (transformation of base vector)</td>
<td>25 + 420</td>
<td>46 + 504</td>
<td>1</td>
</tr>
<tr>
<td>Initializing ( A_3 ) and ( B_3 ) [Eq. (5.7)]</td>
<td>6</td>
<td>19</td>
<td>-</td>
</tr>
<tr>
<td>Method III (Kane)</td>
<td>25</td>
<td>46</td>
<td>1</td>
</tr>
</tbody>
</table>
\[
U_5 = \begin{bmatrix}
-k_1 S_1 \dot{\theta}_1^2 ; -k_1 C_1 \dot{\theta}_1^2 ; (-l_1 S_1 \dot{\theta}_1^2 - k_2 S_2 \dot{\theta}_2^2) ; \\
-1 \dot{C}_1 \dot{\theta}_1^2 - k_2 C_2 \dot{\theta}_2^2
\end{bmatrix}
\]

(5.17)

The base vector \( X \) of the system is:

\[
X = (x_1 \ y_1 \ x_2 \ y_2 \ \theta_1 \ \theta_2)^T .
\]

An operation count for computing the constraint forces \( \Gamma_1 \) and \( \Gamma_2 \) numerically, using the formulas of Table 5.2, is given in Table 5.6, row 2, 3, and 4. The computation of the above forces via the third method, see section 5.2.2 is shown below. First, the original system, which is shown in Fig. 5.4 is embedded in a larger state space, see Fig. 5.5. In that space the original system has four additional translational degrees of freedom which correspond to the variables \( (x_{11}, y_{11}, x_{21}, y_{21}) \). The local vector \((x_{11}, y_{11})^T\) specifies the lower hinge point \( P_1 \) of the first link relative to the origin. The vector \((x_{21}, y_{21})^T\) specifies the lower hinge point \( P_2 \) of the second link relative to the upper hinge point \( P_1 \) of the first link. From Fig. 5.5 it can be seen that the imposition of constraints

\[
x_{11} = y_{11} = x_{21} = y_{21} = 0
\]

(5.18)

reduces this larger dimensional system to the original one. The matrices of the dynamic equations of the new system, Eq. (5.5) are denoted by \( V_i \) \((i = 1, \ldots , 5)\) and are given below:

\[
C_{12} = \cos(\theta_1 - \theta_2)
\]

\[
V_2 = \begin{bmatrix}
1 & 0 & -1 & 0 \\
0 & 1 & 0 & -1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]
Figure 5.5. Double Pendulum Embedded in a Larger State Space.
The basic vector $Z$ of the new system is

$$Z = (x_{11}, y_{11}, x_{21}, y_{21}, \theta_1, \theta_2)^T$$

The constraint forces $\Gamma$ in Eq. (5.5), are:

$$\Gamma = (\Gamma_{1x} \ \Gamma_{1y} \ \Gamma_{2x} \ \Gamma_{2y})^T$$

The constraint forces $\Gamma$ are found in the following way. The equations of the new system, Eq. (5.19), and the constraint equation, Eq. (5.18), which are differentiated twice, are written in the form of Eq. (5.7). A permutation of columns and a reduction of the system produces the following almost upper triangular system of equations:

$$\begin{bmatrix}
1 & 0 & -1 & 0 & a_1 & 0 \\
0 & 1 & 0 & -1 & a_2 & 0 \\
0 & 0 & 1 & 0 & a_3 & a_7 \\
0 & 0 & 0 & 1 & a_4 & a_8 \\
0 & 0 & 0 & 0 & a_5 & a_6 \\
0 & 0 & 0 & 0 & a_6 & a_9
\end{bmatrix}
\begin{bmatrix}
\Gamma_{1x} \\
\Gamma_{1y} \\
\Gamma_{2x} \\
\Gamma_{2y} \\
\ddot{\theta}_1 \\
\ddot{\theta}_2
\end{bmatrix}
= \begin{bmatrix}
b_1 \\
b_2 \\
b_3 \\
b_4 \\
b_5 \\
b_6
\end{bmatrix}$$

with

$$a_1 = m_1 k_1 C_1$$
$$a_2 = -m_1 k_1 S_1$$
$$a_3 = m_2 (l_1 + k_1) C_1$$
\[
a_4 = -m_2(l_1 + k_1)s_1 \\
a_5 = I_1 + m_1k_1^2 + m_1(l_1 + k_1)^2 \\
a_6 = m_2(l_1 + k_1)k_2c_{12} \\
a_7 = m_2k_2c_2 \\
a_8 = -m_2k_2s_2 \\
a_9 = I_2 + m_2k_2^2 \\
b_1 = -k_1s_1^2 \\
b_2 = -k_1c_1^2 - m_1g \\
b_3 = -(l_1 + k_1)s_1^2 - k_2s_2^2 \\
b_4 = -(l_1 + k_1)c_1^2 - k_2c_2^2 - m_2g \\
b_5 = -k_1s_1m_1g - (l_1 + k_1)s_1m_2g - u_1 + u_2 \\
b_6 = -k_2s_2m_2g - u_2
\]

An operation count for doing this is given in Table 5.6, row 6 and 7, where it is assumed that the equations of the new system, Eq. (5.5), are known. However, if the equations of the old system are known only, they can be transformed, via Eq. (5.8), to those of the new system as stated earlier. The operation count for the latter transformation is shown in Table 5.6, row 5.

Elementary row operations are done on Eq. (5.20) to find an analytic expression for the unknown constraint forces:

\[
\Gamma_{1x} = l_1s_1\ddot{\theta}_1^2 + k_2s_2\ddot{\theta}_2^2 - \frac{m_1k_1c_1}{a_5} [k_1s_1m_1g - (l_1 + k_1)s_1m_2g - u_1 + u_2] \\
\Gamma_{1y} = l_1c_1\ddot{\theta}_1^2 + k_2c_2\ddot{\theta}_2^2 - \frac{m_1k_1s_1}{a_5} [k_1s_1m_1g - (l_1 + k_1)s_1m_2g - u_1 + u_2] \\
+ (-m_1 + m_2)g \\
\Gamma_{2x} = b_3 + \frac{a_3b_5}{a_5} + a_5a_7 \frac{b_6 - b_5}{a_5a_7 - a_6} \\
\Gamma_{2y} = b_4 + \frac{a_4b_5}{a_5} + a_5a_8 \frac{b_6 - b_5}{a_5a_7 - a_6}
\]

(5.21)
An operation count for computing the constraint forces via Eq. (5.21) is performed assuming that quantities, which are needed repeatedly, are stored in the memory of a computer. The results are shown in Table 5.6 row 1.

Upper bounds for the round-off error for the computed constraint forces are provided in Table 5.7.

5.4.3 One Link in Three Dimensions

The third example is a one link system in three dimensions (Fig. 5.1). The one link system has its own body fixed coordinate system BCS with axes \((X_1B, X_2B, X_3B)\) and which moves relative to an inertial coordinate system ICS with axes \((X_1I, X_2I, X_3I)\). The center of mass of this system is located at the origin of the body coordinate system, which is aligned with the principal axes of the body [40]. It is assumed that the one link is subject to three holonomic constraints, i.e., it is hinged to the origin. Furthermore, two additional simple nonholonomic constraints [14] are imposed on it, i.e., the rotational motion of the system is confined to rotations about the \(X_3I\) axis. It should be noted that, for the computation of constraint forces, the simple nonholonomic constraints can be treated as holonomic ones [40].

It is shown here that the computation of holonomic constraint forces is very efficient if the third method is used. Furthermore, the computation of the additional nonholonomic constraint forces does not significantly increase the number of operations involved, if the third method is used for this purpose. The equations of motion of this system are:

\[
\begin{align*}
\dot{\boldsymbol{\omega}} &= \mathbf{B} \mathbf{W} \\
\dot{\mathbf{X}} &= \dot{\mathbf{X}} \\
\mathbf{JW} &= \mathbf{WW} \mathbf{FW} + \mathbf{M} + \mathbf{A} \mathbf{TRAD} + \mathbf{K} \mathbf{K} \mathbf{ATR} \\
\mathbf{MX} &= \mathbf{G} + \mathbf{r}
\end{align*}
\]  

(5.17)

where

- \(\dot{\boldsymbol{\omega}}\): angular velocity vector in the ICS
- \(\mathbf{W}\): angular velocity vector in the BCS
- \(\mathbf{WW}\): matrix representation of cross product operator [14]
- \(\mathbf{X}\): location of the center of mass of the body in the ICS.
- \(\mathbf{A}\): map of vectors from BCS to ICS
### Table 5.7. Upper Bounds for Round-Off Errors for Computed Constraint Forces of the Planar Double Pendulum.

The parameters of the digital computer simulation are:

- \( m_1 = 49 \) kg
- \( m_2 = 7.63 \) kg
- \( I_1 = 2.35 \) kg m²
- \( I_2 = 0.089 \) kg m²
- \( l_1 = 0.312 \) m
- \( k_1 = 0.312 \) m
- \( l_2 = 0.2476 \) m
- \( k_2 = 0.184 \) m

Calculation of Constraint Forces via the Method of Single Precision (t=24) Double Precision (t=54)

<table>
<thead>
<tr>
<th>Constraint Forces</th>
<th>Single Precision (t=24)</th>
<th>Double Precision (t=54)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Relative Error</td>
<td>Relative Residual</td>
</tr>
<tr>
<td>Smith (Eq. (5.3))</td>
<td>(&lt; 340 \times 10^{-6})</td>
<td>(&lt; 12 \times 10^{-8})</td>
</tr>
<tr>
<td>Hemami, (Eq. 5.4))</td>
<td>(&lt; 40 \times 10^{-6})</td>
<td>(&lt; 12 \times 10^{-8})</td>
</tr>
<tr>
<td>Kane, (Eq. 5.7))</td>
<td>(&lt; 2 \times 10^{-6})</td>
<td>(&lt; 12 \times 10^{-8})</td>
</tr>
</tbody>
</table>
Figure 5.6. Three-Dimensional Pendulum with Body and Inertial Coordinate Systems Corresponding to Axes \((x_{1B}, x_{2B}, x_{3B})\) and \((x_{1I}, x_{2I}, x_{3I})\).
B: map of the vector \( \dot{W} \) to the vector \( \dot{\alpha} \)
J: Inertia dyadic of the system
Γ: constraint forces due to holonomic constraints
Δ: constraint forces due to simple nonholonomic constraints
G: \((0, 0, -mg)^T\): gravity vector
M: vector of external torques

The holonomic constraints of the system \( X + A K = 0 \) are differentiated twice to arrive at:

\[
X + A(WW)^K - A K K \dot{W} = 0 \tag{5.18}
\]

The two holonomic constraints, which imply that the vector \( W \) is both orthogonal to \( X_2 \) and \( X_3 \), are formulated as

\[
R^T A W = 0, \text{ with } R^T = \begin{bmatrix} 0 & 1 & 1 \\
0 & 0 & 1 \end{bmatrix},
\]

and are differentiated once:

\[
R^T A (WW) W + R^T A W = 0 \tag{5.19}
\]

The application of the first method of calculating constraint forces, Eq. (5.3), gives

\[
\begin{bmatrix}
-I & 0 & M & 0 & \Gamma \\
-K K A^T & -A^T R & 0 & J & \Delta \\
0 & 0 & I & -A K K & X \\
0 & 0 & 0 & R^T A & \dot{W} \\
\end{bmatrix} = \begin{bmatrix}
G \\
-W W J W + M \\
-A(WW)^K \\
0 \\
\end{bmatrix}
\]

The application of the second method, as specified by Eq. (4), renders
For the application of the third method, Eq. (5.7), a vector $Z$ is defined as:

$$X = Z - AK$$

which introduces three additional degrees of freedom into the constrained system. Then the new, larger dimensional system reduces to the original one by setting

$$Z = 0$$

The formulation of the third method of calculating constraint forces is:

$$\begin{bmatrix}
-I & 0 & MA(KK) & M \\
0 & -A^T R & J-M(KK)^2 & -MKKA^T \\
0 & 0 & -RTA & 0 \\
0 & 0 & 0 & -I
\end{bmatrix}
\begin{bmatrix}
\Gamma \\
\Delta \\
\dot{\Delta} \\
Z
\end{bmatrix} =
$$

$$\begin{bmatrix}
MA(KK)^2k_2 \\
-MKK (WW)^2 K + WW JW + M \\
0 \\
0
\end{bmatrix}$$

(5.22)

If the holonomic constraints of the pendulum are to be computed only, the rows and columns corresponding to the nonholonomic constraints are deleted from Eq. (5.20), (5.21), and Eq. (5.22). An operation count of
Table 5.8. Operation Count for Computing Holonomic Constraint Forces of the Three Dimensional Pendulum

<table>
<thead>
<tr>
<th>Method of Computation</th>
<th>Total Number of</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Additions</td>
<td>Multiplications</td>
<td>Searches</td>
<td></td>
</tr>
<tr>
<td>Smith (Eq. (5.3))</td>
<td>268</td>
<td>349</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>Hemami (Eq. (5.4))</td>
<td>271</td>
<td>382</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Kane (Eq. (5.7))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>old system known (Eq. (5.1))</td>
<td>461</td>
<td>435</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>new system known (Eq. (5.5))</td>
<td>41</td>
<td>15</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
Table 5.9. Operation Count for Computing Holonomic and Simple Nonholonomic Constraint Forces of the Three Dimensional Pendulum

<table>
<thead>
<tr>
<th>Method of Computation</th>
<th>Additions</th>
<th>Multiplications</th>
<th>Searches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith (Eq. (5.3))</td>
<td>455</td>
<td>576</td>
<td>29</td>
</tr>
<tr>
<td>Hemami (Eq. (5.4))</td>
<td>460</td>
<td>617</td>
<td>29</td>
</tr>
<tr>
<td>Kane (Eq. (5.7))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>old system known</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Eq. (5.1))</td>
<td>575</td>
<td>530</td>
<td>0</td>
</tr>
<tr>
<td>new system known</td>
<td>71</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>(Eq. (5.3))</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
computing the holonomic forces is shown in Table R. An operation count for computing both the holonomic and simple nonholonomic constraint forces is given in Table 5.9. In Tables 5.8 and 5.9 the number of operations, which are necessary to initialize the respective matrices are not listed since both tables only demonstrate the additional computational effort of calculating non-holonomic constraint forces.

5.5 Conclusion

In this paper three different methods of computing holonomic constraint forces are discussed. The first method, as suggested by Smith, is suitable primarily for the analysis of linkage systems, since the unknown accelerations of the mechanical system are solved simultaneously with the unknown constraint forces. An analysis of the round off errors, which are due to the number representation in a digital computer, reveals that the relative error in computing the above forces is fairly high for the two examples considered. It is, therefore, appropriate to use a double precision format for programming the first method on a digital computer.

The second method of computing constraint forces is suitable primarily for control purposes, since the constraint forces are expressed as functions of the state and the input of the mechanical system. An operation count for this method shows, that the computation of constraint forces is relatively efficient if the inertia matrix of the system is diagonal. Furthermore, if the mechanical system satisfies the above condition and is, in addition, subject to few constraints, this method is the most efficient one. The round off error, inherent in this method, is considerably lower than the one of the first method.

The third method of computing holonomic constraint forces is suitable for the analysis of control systems, since the system of equations is essentially solved for the unknown constraint forces. This method is computationally least costly for most of types of mechanical systems considered. The above method also has the smallest upper bound for round-off errors. It is suitable for deriving analytical expressions for constraint forces, as well. It is computationally efficient if simple nonholonomic constraint forces are computed simultaneously with holonomic forces.

The high cost of computing the constraint forces via methods I, II, and III (provided that the system is known in the old formulation, Eq. (5.1)) is due to the size of the matrices involved. It is evident, however, that the computation of constraint forces via analytical expressions requires a minimal computational effort. The reason for this is, that symbolic simplification is done when an analytical expression for constraint forces is derived, i.e., many terms involving trigonometric expressions are combined to a single expression.
Chapter 6

Discussions and Conclusions

6.1 Summary

In this dissertation, the attitude control of multi-linkage systems is analyzed. In this framework, the rotational dynamics of multi-body systems are derived, the controllability properties of these systems are studied, and their graphical representation is investigated by means of a kinematic model. The studies in the above mentioned areas are done on different types of multi-body systems.

The kinematics of the above mentioned systems are explored on the model of a human-being represented by twenty-three body segments which are made up of wire-basket solids of revolution. For that purpose a computer graphics program is developed, which accepts the kinematic information of the humanoid as input parameters, solves the associated hidden surface problem, and does a transformation from three-dimensional world space to a two-dimensional image space. Although the graphics package consists of a set of elementary computer graphics functions, such as perspective transformations, open and closed polygon clipping, it contains advanced algorithms, which perform hidden surface removal as well. The graphics package is structured such that the objects are displayed on a vector display but can also be shown on a raster scan display in case of a failure of the first mentioned output device. Major contributions of this study in the area of computer graphics are the implementation and optimization of the above mentioned algorithms in a minicomputer environment and the definition of particular data structure which accomplishes this task. It is shown that the humanoid can be represented by a set of convex polyhedra whose respective polygons are stored in polygon list format. However, for processing the object through the Weiler Atherton hidden surface algorithm it is advantageous to change the data structure to matrix interconnection format to speed up the computation of the final display.

Two important issues in dynamics, the determination and systematic symbolic generation of the rotational equations of motion are investigated. It is obvious that the derivation of the dynamics of multi-body systems is prone to human errors, especially, if the number of bodies involved are high. For that reason a general purpose symbolic manipulation program is employed which derives the above mentioned
dynamics automatically. It is shown that the symbolic manipulation program can automatically check itself by deriving the dynamics via two distinct approaches and comparing the results. First integrals of motion can be derived in a similar way.

The controllability of the rotational dynamics of a planar, two link system are analyzed using methods of linear and nonlinear control theory. It is shown that the above mentioned two approaches produce similar results. It is pointed out that the constant angular momentum constraint, to which the above mentioned system is subjected, has quasi-holonomic character. This stems from the fact the quasi-holonomic constraint can be used to eliminate only one state variable of the system instead of two, which is the characteristic of a holonomic constraint in position state variables. This corresponds to the loss of only half a degree of freedom instead of the loss of one. The above mentioned quasi-holonomic constraint can be used to project the planar two-link system, which is partially controllable in the original space, to a reduced space, in which it is completely controllable.

The inverse attitude control problem is defined for the planar two link system. It is solved by transforming the corresponding ill-posed boundary value problem into a well-posed one using a well-known method. Digital computer simulations are performed to demonstrate that linear state feedback ensures that the nonlinear two link system is capable of tracking a prespecified trajectory. It is shown that the above mentioned, partially controllable, unreduced dynamic system does not exhibit numerical instabilities which are induced during a numerical simulation if the quasi-holonomic constraint is a linear function of the state variables of the system. In case of nonlinear constraints, additional hypothetical inputs have to be defined whose only purpose is to maintain the constraint and assure stability.

The derivation and computation of holonomic and simple non-holonomic constraint forces or torques in imbedded multi-body systems is studied. In fast and efficient computation of the above mentioned quantities is an important issue in robotics applications under real-time constraints. The analysis comprises a comparison of three different algorithms of computing the above type of constraint forces or torques. Several examples demonstrate merits of these different techniques and compare them.
6.2 **Recommendations for Future Research**

The work outlined in this dissertation can be extended in the following major areas:

1) **Symbolic generation of the equations of motion:**

Symbolic algebraic manipulation (SAM) programs can be effectively employed to derive the dynamics of multi-body systems. However, major shortcomings in this application are due to the finite amount of data space for the storage of intermediate results. Hence, some research can be focussed in reducing the amount of this space and, thereby, making the program more useful. This can be accomplished by devising additional rules which are characteristic of the system under investigation, for example for rigid, pin-connected multi-body systems a characteristic relationship is:

\[ \dot{A} = AWW \]  
(See Eqn. 5.18)

2) **Automated controllability analysis:**

The problem of a finite amount of data space occurs in the automated controllability analysis as well. Hence, some research effort can be directed in making the analysis more efficient. Further research can be directed in developing an expert system for the nonlinear controllability analysis of mechanical systems in which all applicable controllability theorems for the above mentioned systems are implemented.

3) **Graphical representation of multi-body system:**

Movie like images of multi-body systems can be produced by computing a set of postures off-line by loading them into the memory of the graphics generator, and by sequentially blanking and unblanking the display files which contain the respective postures. Hence, some research can be focussed on devising a dedicated computer architecture for a real-time display of the motion of multi-body systems.

4) **Numerical instabilities during the simulation of constrained system:**

Simulation of the unreduced, partially controllable planar two link system does not produce nor indicate numerical instabilities if the respective constraints of the system are linear functions of the state variables of the system. Hence, future research can be directed in investigating whether these findings hold for other types of constrained systems as well.
APPENDIX A

COMPUTER GRAPHICS PROCEDURES

In Appendix A program listings are provided for the procedures of the Graphics Package which is described in Chapter 3. The organization of this appendix is as follows. In Appendix A.1, A.2, and A.3 program listings are provided for the basic graphics package, see Fig. 3.2, the modules which generate the humanoid, see Fig. 3.20, and the modules which accomplish the animation. Finally, in Appendix A4 some help documents are provided for using the graphics program package.

A.1 The Modules of the Basic Graphics Package

A program listing of the basic graphics package, see Fig. 3.2, is given below:

```
VERSION OF GRAPHMOD.PAS: VERSION 2, JUNE 4, 1986

VERSION: GRAPHMOD.PAS

GRAPHMOD.PAS contains elementary graphics subroutines, which are written
in PASCAL.

USAGE: GRAPHMOD.PAS is included in the main program using the
compiler directive: graphmod.

COMPILED: MAIN program: name.pas

WORK program: GRAPHMOD.PAS

this package contains the following procedures

PROCEDURE NAME, VERSION No., DATE, MODIFICATIONS

Global 1 Jan-7-B6
InfPerspective 1 Feb-15-86
InitGraphics 1 Feb-15-86
GenCylinder 1 Jan-20-86
GenSphere 1 Jan-21-86
... MoveTo 1
... LineTo 1
... OpenSegment 1
... CloseSegment 1
... PostSegment 1
... UnPostSegment 1
... ConcatFile 1 Feb-10-86
... Trun_Rot 1 Feb-07-86
... FhMatVec 1
... FhMutMat 1
... Wm3 1
... CopyMat 1
... CopyVec 1
... VwAddVecSubVec 1
... Bry_ang 1
... ClipEdgesin3d 1
... ClipPolyin3d 1
... ClFowlClip3c 1

******************************************************************************
```
dieter langer

7-Jan-86

this file provides the global variables, which can be compiled with all procedures

CON:

pi=3.141592654

epsilon=1E-5

TYPE

arr3=AARRAY[1..33] OF integer;
arr3=AARRAY[1..33] OF real;
arr4=AARRAY[1..43] OF real;
arr4=AARRAY[1..43] OF integer;
arr5=AARRAY[1..53] OF integer;
arr6=AARRAY[1..63] OF real;
arr6=AARRAY[1..63] OF char;
arr101=AARRAY[1..101] OF integer;
arr50=AARRAY[1..50] OF real;
arr2k41=AARRAY[1..21,1..42] OF integer;
arr3k3=AARRAY[1..3,1..33] OF real;
arr2k41=AARRAY[1..21,1..42] OF real;
arr4k41=AARRAY[1..4,1..43] OF real;
arr4k41=AARRAY[1..4,1..43] OF integer;
arr20k5r=AARRAY[1..20,1..52] OF real;
outcode=AARRAY[1..4] OF boolean;
str=string[203];
str2=string[123];
str4=string[143];
str8=string[183];
str14=string[141];
UAR

Coi,eu:arr3; (center of interest, eye position)

vadeg, arr4: real; (viewing angle(deg.), aspect ratio, hither, yon)

viewart:arr4;

hp,yp:2u,y2p:integer;

fronti,backface:char;

testprogram, paraminit, mapWldtoScrready:boolean; (test of the whole program)

plotGorFile,afortFc:char; (used in all plot routines)

inputFile;

drive_ws,drive_wld,drive_plt,drive_dat: str2;

ws:arr4r; (disk drives for master, world, plot and data files)

ws2:arr4r; (map WLD --> SCR)

viewport:arr4;

LastXFlotted, LastYFlotted: integer; (used in MoveTo, LineTo)

ok:boolean;

function: dieter langer

date: 15-feb-86

function: initialization of parameters for perspective transformation

PROCEDURE InitPerspective;

UAR

...
BEGIN < InitPerspective >
pi:=3.1415927;
write(' Subroutine InitPerspective ');}
write( viewing angle, deg. ) = ? ' ); readln( vadeg ) ;
write( eye position : radius = ? ' ); readln( radius )
Repeat:
write(' input of (theta,phi) or ');
write(' (top, side, front view) [T,V] = ? ');
readln( ch ) ;
Until ch in ['T', 'V']( 'CASE ch of:
'T': begin:
REPEAT
write( elevation angle [0, 90] = ? ' );
readln( theta ) ;
UNTIL ( ( theta >= 0.0 ) AND ( theta < 90.0 ) ) ;
REPEAT
write( azimuth angle [0, 360] = ? ' );
readln( phi ) ;
UNTIL ( ( phi >= 0.0 ) AND ( phi < 360.0 ) ) ;
end;
'V': begin:
REPEAT
write( Front, Top, or Side view [F, T, S] = ? ' );
readln( cmdch ) ;
Until cmdch in ['F', 'T', 'S'] {
CASE cmdch of:
'F': begin
theta:=89.99; phi:=0.1
end;
'T': begin
theta:=0.1; phi:=0.1
end;
'S': begin
theta:=89.99; phi:=89.99
end
end \{ case statement \}
end \{ begin \}
end \{ case statement \}
theta:=theta*pi/180; phi:=phi*pi/180;
ep[1]:=radius*sin(theta)*sin(phi);
ep[2]:=radius*cos(theta);
eu[3]:=radius*sin(theta)*cos(phi);
REPEAT
write( init. of other persp. paramet. [Y/N] = ? ' ); readln( cmdch ) ;
UNTIL cmdch IN ['Y', 'N'] ;
CASE cmdch OF:
'Y':
BEGIN
write( aspect ratio [GT:1.0 : PC:1.24] = ? ' ); readln( ) ;
write( center of interest : y-comp. = ? ' ); readln( ) ;
write( x-comp. = ? ' ); readln( ) ;

end \{ begin \}
end \{ case statement \}

156
write(' testprogram=false
REPEAT
  write(' plot on Gt, or Fc, or write to File [G/F/F] ? '); readln(ch);
  UNTIL ch IN ['G','F','F'];
  IfPlotGtorFileOrFc:=ch
END;
'N';
BEGIN
  cos[1]=0;cos[2]=0;cos[3]=0;h=0.5;yo=500;
estprogram:=false;PlotGtorFileorFc:=F
END;
writeln;
REPEAT
  write(' frontorbackface [F / B / A]:(#F#)=(#B#)=? '); readln(frontorbackface);
  UNTIL frontorbackface IN ['F','B','A'];
< other fixed parameter of the graphics terminal >
if ((PlotGtorFileOrFc='F') or (PlotGtorFileorFc='p')) then
  begin
  end
ELSE
  begin
  end
initinit:=true; < perspective parameters are initialized >
mapWtolScrready:=false  < map :: WLD --> SCR must be initialized >
END; < InitPerspective >

< function : initialization of graphics terminal >
< date : 15-feb-86 >

PROCEDURE InitGraphics;
VAR
et,dc4,cr,lf,chr;
BEGIN
  et:=chr(3); dc4:=chr(20); cr:=chr(13); lf:=chr(10);
  case PlotGtorFileOrFc of
    'G': begin
      writeln(et;chr(dc4;cr);lf); writeln(et;en;en;en;en;en;en;en);
    end;
    'F': begin
      writeln(out;et;chr(dc4;cr);lf); writeln(out;en;en;en;en;en;en;en);
    end;
    'F': begin
      hires:hirescolor(yellow) < 640 * 200 dots black + one color >
    end
  end; (case )
END;
PROCEDURE ellipse(xaxislen, zaxislen: real; ndiv: integer; VAR xcoord, ycoord: arr[50, integer]; VAR pi: real; i: integer;
BEGIN
    pi := 3.14159271;
    FOR i := 0 TO ndiv-1 DO
        BEGIN
            xcoord[i+1] := xaxislen * cos(2*pi/ndiv*i);
            ycoord[i+1] := zaxislen * sin(2*pi/ndiv*i);
        END;
    END;  \{ ellipse \}

PROCEDURE gencylinder(xaxistop, xaxisbottom, zaxistop, zaxisbottom, height: real; ndivhorizontal, ndivvertical: integer; drive: str2; outfile: str8; extension: str4);
VAR
    npoly, nedge, nvert: i; j, k: integer;
    xcoord, ycoord: arr[50, integer];
    xaxisactual, zaxisactual, yactual: real;
    vtiarr3i: pi: arr[50, i];
    outfile: str8;
BEGIN \{ gencylinder \}
    npoly := ndivvertical*ndivhorizontal;
    nvert := (ndivvertical+1)*ndivhorizontal;
    nedge := 4;
    outfile := concat(drive, outfile, extension); assign(outfile, outfile);
    rewrite(outfile);
    write(outfile, npoly); write(outfile, nvert); writeln(outfile);
    FOR i := 0 TO ndivvertical DO
        BEGIN
            xaxisactual := xaxistop + (xaxisbottom-xaxistop)/ndivvertical * i;
            zaxisactual := zaxistop + (zaxisbottom-zaxistop)/ndivvertical * i;
            yactual := height/2-height/ndivvertical * i;
            \{ computation of vertices \}
            ellipse(xaxisactual, zaxisactual, ndivhorizontal, xcoord, ycoord);
            FOR j := 1 TO ndivhorizontal DO
                BEGIN
                    vti(j) := xcoord[j];
                    vt[j+3] := yactual;
                    write(outfile, vti(j));
                    write(outfile, vt[j+3]);
                    writeln(outfile);
                END;
        END;
    END;
< computation of polygon faces >

FOR i:=1 TO ndivvertical DO
BEGIN
FOR j:=1 TO ndivhorizontal DO
BEGIN
k:=(i-1)*ndivhorizontal+j;
pn[1]:=edge1;
pn[2]:=k; pn[3]:=k*ndivhorizontal1;
IF j*ndivhorizontal1 THEN
BEGIN
pn[4]:=1*ndivhorizontal1+1;
pn[5]:=(i-1)*ndivhorizontal1+1
END;
ELSE
BEGIN
pn[4]:=pn[3]+1;
pn[5]:=pn[2]+1
END;
write(out,pn[1]:6);write(out,pn[2]:6);
write(out,pn[3]:6);write(out,pn[4]:6);
write(out,pn[5]:6);writein(out);
END;
close(out) < close file OUT >
END; < gen cylinder >
PROGRAMMER:  Dieter Longer

DATE:  21-Jan-86

FUNCTION:  generate master coordinates for plotting a sphere

PROCEDURE gensphere(ndivh oriz,ndivvertdiv2:integer;radius:real;drive:std2;
outfile:std4;extension:std4;var out:ext);

VAR
x,y,z,th,phi:real;  nvert,npoly:integer;
infilenout:std2; p: real;
infilename, outputfilename: std;

PROCEDURE sphere(radius,th,phi:real; VAR x,y,z:real);
BEGIN
x:=radius*sin(th)*sin(phi);
y:=radius*sin(th)*cos(phi);
z:=radius*cos(th);
END;

BEGIN
pi:=3.141592654;
nvert:=(ndivvertdiv2+1)*ndivh oriz+1;
npoly:=ndivvertdiv2*ndivh oriz;
outputfilename:=concat(drive,outfile,extension);assign(out,outputfilename);
rewrite(output);
write(out,nvert);write(out,npoly);writeln(out);
x:=-radius;write(output,0);write(output,x,12:3);
write(output,y,12:3);write(output,z,12:3);
FOR i:=1 TO ndivvertdiv2-1 DO
BEGIN
th:=pi/ndivvertdiv2+1;
FOR j:=0 TO ndivh oriz-1 DO
BEGIN
phi:=2*pi/ndivh oriz+j;
sphere(radius,th,phi,x,y,z);
write(output,x,12:3);write(output,',');
write(output,y,12:3);write(output,',');
write(output,z,12:3);writeln(output);
END;
END;
x:=0;y:=-radius;z:=0;writeln(output,x,12:3,y,12:3,z,12:3);

< computation of polygon list >

FOR i:=1 TO ndivvertdiv2 DO
BEGIN
FOR j:=1 TO ndivh oriz DO
BEGIN
IF (j=1)
THEN
BEGIN  < first set of triang. faces >
p[13]:=3;p[23]:=1;p[33]:=j+1;
END
END

< (programmer :  dieter longer )
< (date :  21-Jan-86 )
< (function :  generate master coordinates for plotting a sphere )

000000160
IF (j=ndivhoriz)
THEN
BEGIN
  p[43] := 2;
END;
ELSE
BEGIN
  p[43] := j+2;
END;
WRITE(out,p[13]:6); WRITE(out,p[23]:6);
WRITE(out,p[33]:6); WRITE(out,p[43]:6); WRITEin(out);
END; (* first set of triangle faces *)
ELSE
BEGIN
  IF (j=ndivvertdiv2)
  THEN
    BEGIN
      { terminal set of triangle faces }
      p[33] := nvert;
      IF (j=ndivhoriz)
      THEN
        BEGIN
          p[43] := (ndivvertdiv2-2)*ndivhoriz+2;
        END;
      ELSE
        BEGIN
          p[43] := p[23]+1;
        END;
    END; (* terminal set of triangle faces *)
  ELSE
  BEGIN
    { intermediate faces }
    BEGIN
      p[33] := p[23]+ndivhoriz;
      IF (j=ndivhoriz)
      THEN
        BEGIN
          p[53] := (j-2)*ndivhoriz+2;
          p[43] := p[53]+ndivhoriz;
        END;
      ELSE
        BEGIN
          p[53] := p[23]+1;
        END;
    END; (* intermediate faces *)
  END; (* end of if (j=ndivvertdiv2) *)
END; (* else *)
END; (* end of if (j=ndivhoriz) *)
close(out);
END;
<programmer : dieter longer
<date : 16-jan-86
<function : graphic primitives

PROCEDURE moveto(x,y:integer);
BEGIN
  case FloatGtorFileorFc of
    'G': write(LST,'pe0;pa',x:4','y:4','!'); writeln(LST);
    'F': write(DUT,'pe0;pa',x:4','y:4','!'); writeln(DUT);
    'P': begin
      LastXPlotted:=x;LastYPlotted:=y
      end;  
  end(end case);  
END;

PROCEDURE lineto(x,y:integer);
BEGIN
  case FloatGtorFileorFc of
    'G': begin
      write(LST,'pa',x:4','y:4','!'); writeln(LST)
      end;
    'F': begin
      write(DUT,'pa',x:4','y:4','!'); writeln(DUT)
      end;
    'P': begin
      draw(LastXPlotted,LastYPlotted,x,y,1);
      LastXPlotted:=x;LastYPlotted:=y
      end
  end(end case);  
END;

procedure OpenSegment(i:integer);
var outfile:istr14;
  value_of_i:istr2;
begin
  { open output file : SEGM1.FLT }
  str(1,value_of_i);
  outfile:=conc(drive_plt,'segm',value_of_i,'.plt');
  assign(outfile);rewrite(outfile);  
  case FloatGtorFileorFc of
    'G': begin
      write(lst,'nf',14,'!'); writeln(lst)
      end;
    'F': begin
      write(out,'nf',14,'!'); writeln(out)
      end;
    'P': begin end { no segmentation }
  end(end case);  
end;

procedure CloseSegment;
begin
  case FloatGtorFileorFc of
    'G': begin
      write(lst,'sn!'); writeln(lst)
      end;
end;
procedure PostSegment(i:integer);
begin
    case FloatGforFileorFc of
        'G': begin
            writeln(lst,'uf',i:4,'i',''); writeln(lst)
        end;
        'F': begin
            writeln(out,'uf',i:4,'i',''); writeln(out)
        end;
        'P': begin end < no segmentation >
    end;
end;

procedure UnPostSegment(i:integer);
begin
    case FloatGforFileorFc of
        'G': begin
            writeln(lst,'bf',i:4,'i',''); writeln(lst)
        end;
        'F': begin
            writeln(out,'bf',i:4,'i',''); writeln(out)
        end;
        'P': begin end < no segmentation >
    end;
end;

procedure DeleteSegment(i:integer);
begin
    case FloatGforFileorFc of
        'G': begin
            writeln(lst,'ef',i:4,'i',''); writeln(lst)
        end;
        'F': begin
            writeln(out,'ef',i:4,'i',''); writeln(out)
        end;
        'P': begin end < no segmentation >
    end;
end;

procedure concatfiles(drive_in1:char; str2:name_in1; str4; drive_in2:char; name_in2; str8; ext_in2; str4; drive_out:char; str5; name_out; str8; ext_out: str4;
offset1:offset2;arr3);

< programmer : dieter longer >
< date : 10-feb-86 >
< function : read file name_in1 ,add offset1 to the vertices >
< read file name_in2 , add offset2 to the vertices >
< copy both files into file name_out >

163
var inp1,inp2,outfile:integer;
inplfile,inp2file,outfile:std_string;
1,2,3,nvert1,nvert2,npoly1,npoly2,nvertout,npolyout:integer;
vtlarr[1..nvert1];

begin

inp1file:=concat(drive_in1,name_in1,ext_in1);assign(inp1,inp1file);
reset(inp1);
inp2file:=concat(drive_in2,name_in2,ext_in2);assign(inp2,inp2file);
reset(inp2);
outfile:=concat(drive_out1,name_out1,ext_out1);assign(outfile,outfile);
rewrite(outfile);

read(inp1,nvert1);read(inp1,npoly1);readln(inp1);
read(inp2,nvert2);read(inp2,npoly2);readln(inp2);
nvertout:=nvert1+nvert2;npolyout:=npoly1+npoly2;
write(outfile,nvertout);write(outfile,npolyout);writeln(outfile);

< copy vertices of 1st file + offset1 into outfile >
for i:=1 to nvert1 do
begin
vt1[i,1]:=read(inp1,vt1[i,1]);vt1[i,2]:=read(inp1,vt1[i,2]);vt1[i,3]:=read(inp1,vt1[i,3]);
vt1[i,1]x:=vt1[i,1]x+offset1[1,i];vt1[i,2]x:=vt1[i,2]x+offset1[2,i];vt1[i,3]x:=vt1[i,3]x+offset1[3,i];
write(outfile,vt1[i,1]x,vt1[i,2]x,vt1[i,3]x);
end;

< copy vertices of 2nd file + offset2 into outfile >
for i:=1 to nvert2 do
begin
vt2[i,1]:=read(inp2,vt2[i,1]);vt2[i,2]:=read(inp2,vt2[i,2]);vt2[i,3]:=read(inp2,vt2[i,3]);
vt2[i,1]x:=vt2[i,1]x+offset2[1,i];vt2[i,2]x:=vt2[i,2]x+offset2[2,i];vt2[i,3]x:=vt2[i,3]x+offset2[3,i];
write(outfile,vt2[i,1]x,vt2[i,2]x,vt2[i,3]x);
end;

< copy polygons of 1st file into outfile >
for i:=1 to npoly1 do
begin
read(inp1,pn[i,1]);write(outfile,pn[i,1]);
for j:=2 to pn[i,1]+1 do
begin
read(inp1,pn[i,j]);write(outfile,pn[i,j]);
end;
readln(inp1);writeln(outfile);
end;

< copy polygons of 2nd file into outfile >
for i:=1 to npoly2 do
begin
read(inp2,pn[i,1]);write(outfile,pn[i,1]);
for j:=2 to pn[i,1]+1 do
begin
read(inp2,pn[i,j]);pn[i,j]:=pn[i,j]+nvert1;write(outfile,pn[i,j]);
end;
readln(inp2);writeln(outfile);
end;
The body is rotated using matrix \texttt{rotmat} and translated by \texttt{orig}.

The transformed body is stored in the file \texttt{drive:inpfile.wld}.

```plaintext
BEGIN
<read in vertices>
read\(\texttt{inp}\),\(\texttt{nvert}\);read\(\texttt{inp}\),\(\texttt{npoly}\);read\(\texttt{inp}\);
write\(\texttt{out}\),\(\texttt{nvert}\);write\(\texttt{out}\),\(\texttt{npoly}\);write\(\texttt{out}\);
for \(i=1\) to \(nvert\) do
begin
read\(\texttt{inp}\),\(\texttt{vt}\[1]\);read\(\texttt{inp}\),\(\texttt{vt}\[2]\);read\(\texttt{inp}\),\(\texttt{vt}\[3]\);read\(\texttt{inp}\);
for \(j=1\) to \(3\) do
begin
\texttt{vtnew}[][]=0;
for \(k=1\) to \(3\) do
begin
\texttt{vtnew}[][]=\texttt{vtnew}[][]+\texttt{rotmat}[][]\texttt{vt}[][]
end;
<add vector \texttt{vt} to transformed vertices>
\texttt{vtnew}[][]=\texttt{vtnew}[][]+\texttt{orig}[][]
end;
write\(\texttt{out}\),\(\texttt{vtnew}\)[1:12:3];write(\texttt{out},
" ");write\(\texttt{out}\),\(\texttt{vtnew}\)[2:12:3];write(\texttt{out},
" ");write\(\texttt{out}\),\(\texttt{vtnew}\)[3:12:3];write\(\texttt{out}\);
end;
<copy polygon indices into output file>
for \(i=1\) to \(npoly\) do
begin
read\(\texttt{inp}\),\(\texttt{pn}\)[1];write\(\texttt{out}\),\(\texttt{pn}\)[1:6];
for \(k=2\) to \(\texttt{pn}[1]+1\) do
begin
read\(\texttt{inp}\),\(\texttt{pn}\)[k];write\(\texttt{out}\),\(\texttt{pn}\)[k:6]
end;
read\(\texttt{inp}\);write\(\texttt{out}\)
end;
end;
```

<date: 10-feb-86>
<function: matrix subroutines>
procedure preotvec(mat:arr3k3r; vec:arr3r; res:arr3r);
< function : product of a (3 x 3) matrix with a (3 x 1) vector >
var row, col: integer;
begin
  for row:=1 to 3 do
  begin
    res[row]:=0;
    for col:=1 to 3 do
    begin
      res[row]:=res[row]+mat[row,col]*vec[col]
    end
  end
end;

procedure preomat(matfirst:matsec:arr3k3r; var matres:arr3k3r);
< function : product of a (3 x 3) matrix with a (3 x 3) matrix >
var row, col, i: integer;
begin
  for row:=1 to 3 do
  begin
    for col:=1 to 3 do
    begin
      matres[row,col]:=0;
      for i:=1 to 3 do
      begin
        matres[row,col]:=matres[row,col]+matfirst[row,i]*matsec[i,col]
      end
    end
  end
end;

procedure umat3(var mat:arr3k3r);
< function : setting up a unity matrix >
var row, col: integer;
begin
  for row:=1 to 3 do
  begin
    for col:=1 to 3 do
    begin
      if (row=col) then begin
        mat[row,col]:=1
      end
      else begin
        mat[row,col]:=0
      end
    end
  end
end;
procedure copymat(matin;arr3x3;ivar matcopy;arr3x3);
var row;col:integer;
begin
for row:=1 to 3 do
begin
for col:=1 to 3 do
begin
matcopy[row,col]:=matin[row,col]
end
end;
end;

procedure copyvec(vecin;arr3x1;ivar veccopy;arr3x1);
var row:integer;
begin
for row:=1 to 3 do
begin
vecopy[row]:=vecin[row]
end;
end;

procedure vecaddvecsubvec(vec1;vec2;vec3;arr3x1;ivar vecres;arr3x1);

<function : vec1+vec2-vec3=vecres>
var row:integer;
begin
for row:=1 to 3 do
begin
vecres[row]:=vec1[row]+vec2[row]-vec3[row]
end;
end;

<function : setup a rotation matrix for bryant angles>
procedure bry_ang(th1;th2;th3:real;ivar rotmat;arr3x3);
var c1;c2;c3;s1;s2;s3:real;
begin
c1:=cos(th1);c2:=cos(th2);c3:=cos(th3);
s1:=sin(th1);s2:=sin(th2);s3:=sin(th3);
rotmat[1,1]:=c2*c3;
rotmat[1,2]=-c2*s3;
rotmat[1,3]:=s2;
rotmat[2,1]:=c1*s3+s1*s2*c3;
rotmat[2,3]=-s1*c2;
rotmat[3,1]=-s1*s3+c1*s2*s3;
rotmat[3,2]=s1*c3+s1*s2*s3;
rotmat[3,3]=c1*c2
end;
<programmer>distancelanger</programmer>
<date>8-jun-86</date>
<function>mapping of straight lines in 3d world space into screen space</function>
<input param.>x1,y1,z1, x2,y2,z2 endpoints of line</input param.>
<output param.>x1p,y1p,x2p,y2p endpoints of line on the screen</output param.>
<global var.>viewport=(v1,vr,vt,vb) of viewport</global var.>

PROCEDURE clipedgesin3d(x1,y1,z1,x2,y2,z2)
VAR xlp,ylp,x2p,y2p:integer;
VAR MapWtDataScrReady: boolean; VAR wsi:array(4,4) of real;

VAR
p1w:p2w:w1s:p2s:arr4r;
VAR

PROCEDURE perspe(VAR wsi:array(4,4) of real);
(function) mapping from world space to screen space using
<definitions>
AR aspect ratio
H Viewing angle
V viewpoint
CDI center of inter.

VAR
ws:array(4,4) of real;
BEGIN
BEGIN

VAR
ub,dc,d,f,alp,bet,del,sth,cph,sph,va:real;
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN

VAR
BEGIN
PROCEDURE clip(x1,y1,z1,w1,x2,y2,z2,w2:real;VAR x1p,y1p,x2p,y2p:integer);

TYPE
  edge=1..6;
  outcode=SET OF edge edge takes values 1,2,3,4,5,6 and outcode is
  a set with at most 6 elements

VAR
  wc:ARRAY[1..2,1..6] OF real;
  cl,c2:outcode;
  dx,dy,dz,dw,t1,t2:real;
  li:integer;

PROCEDURE showline(xx1,yy1,zz1,ww1,xx2,yy2,zz2,ww2:real;VAR x1p,y1p,x2p,y2p:integer);

VAR
  v1,v2,v3,v4:integer;

BEGIN
  < showline >
  if (abs(w1) < epsilon) then
  begin
    x1=(v1+v2)/2; y1=(v3+v4)/2;
  end
  ELSE
  begin
    x1=xx1/w1*(v1-v2)/2+(v1+v2)/2;
    y1=yy1/w1*(v3-v4)/2+(v3+v4)/2;
  end;
  if (abs(w2) < epsilon) then
  begin
    x2=(v1+v2)/2; y2=(v3+v4)/2;
  end
  ELSE
  begin
    x2=xx2/w2*(v1-v2)/2+(v1+v2)/2;
    y2=yy2/w2*(v3-v4)/2+(v3+v4)/2;
  end;
  x1p=round(x1); y1p=round(y1); x2p=round(x2); y2p=round(y2);
</ showline >

PROCEDURE makewindowcoords(pi:integer;xx,yy,zz,ww:real;VAR c:outcode);

VAR
  li:integer;

BEGIN
  < makewindowcoord >
  wc[pi,1]=xx+wc[pi,2];
  wc[pi,3]=yy+wc[pi,4];
  wc[pi,5]=zz+wc[pi,6];
  li[i] (c = empty set);
  FOR I=1 TO 6 DO
  IF wc[pi,i] < 0
  THEN
    IF wc[pi,i] < 0
THEN c:=c+{1}
(c is the set of elements of wc, which are outside clip limits)

END;

(makewindowcoords)
BEGIN (clip)
makewindowcoords(1,1,y1,z1,w1,c1);
makewindowcoords(2,2,y2,z2,w2,c2);
IF (c1∩c2) = {} THEN

< if both endpoints are outside clipping plane throw edge out >
BEGIN (c1∩c2 : set c1 intersect with set c2)
t1:=0; t2:=1;
FOR i:=1 TO 6 DO
IF (wc1[1,i] < 0) OR (wc2[1,i] < 0) THEN BEGIN
BEGIN
t:=wc1[1,i]/(wc1[1,i]-wc2[1,i]);
IF wc1[1,i] < 0 THEN BEGIN
BEGIN IF t > t1 THEN t1:=t
END ELSE BEGIN IF t < t2 THEN t2:=t
END
END;
IF t2 >= t1 THEN BEGIN
BEGIN
ax:=x2-x1; ay:=y2-y1;
az:=z2-z1; aw:=w2-w1;
IF t2 <= 1 THEN BEGIN < if point 2 is outside clipping plane, clip pair >
BEGIN
x2:=x1+t1*ax; y2:=y1+t1*ay;
z2:=z1+t1*az; w2:=w1+t1*aw
END;
IF t1 < 0 THEN BEGIN < if point 1 is outside clipping plane, clip pair >
BEGIN
x1:=x1+t2*ax; y1:=y1+t2*ay;
z1:=z1+t2*az; w1:=w1+t2*aw
END;
showline(x1,y1,z1,w1,x2,y2,z2,w2)
END;
END;
END;
END;
END;
END;
END;
END;
END;
BEGIN <clip3d>
IF (NOT parminit)
THEN
BEGIN
writeln(con,'### perspective parameters are not initialized ###
writeln(con,'### call procedure InitPerspective  ###
END;
ELSE
BEGIN
<initialize transformation matrix WS>
if (not MapWldtoScrready) then begin perspe(ws) end;
<now map : WLD --> SCR is initialized>
MapWldtoScrready:=true;
<initialize vectors p1w and p2w>
p1w[13]:=x1p1w[13]:=y1p1w[13]:=z1p1w[13]:=1;
p2w[13]:=x2p2w[13]:=y2p2w[13]:=z2p2w[13]:=1;
<mapping from world to screen space>
FOR i:=1 TO 4 DO
BEGIN
p1s[i3]:=0;p2s[i3]:=0;
FOR j:=1 TO 4 DO
BEGIN
p1s[i3]:=p1s[i3]+p1w[j3]*ws[i,j];
p2s[i3]:=p2s[i3]+p2w[j3]*ws[i,j];
END
END;
<clipping screen space coordinates>
clip(p1s[13],p1s[23],p1s[33],p1s[43],p2s[13],p2s[23],p2s[33],p2s[43],
::l1p,x1p,y1p,z1p;p2p,x2p,y2p);
IF testprogram
THEN
BEGIN
writeln;
writeln('----------procedure clipedgesin3d-----------
writeln('
p1screen x,y,z
writeln(p1s13,p1s23,p1s33,p1s43,p2s13,p2s23,p2s33,p2s43,
pworld x,y,z
writeln(p1w[13],p1w[23],p1w[33],p1w[43],p2w[13],p2w[23],p2w[33],p2w[43]
writeln('x1p,y1p,z1p;x2p,y2p');
END
END; <clip3d>
PROCEDURE cippolyin3d;

TYPE
  ptr := vertex;
  vertex := RECORD
    vt: ARRAY[1..100,1..33] OF real;
    nxt: ptr;
    bck: ptr
  END;

VAR
  npoly, nvrt, nedge, i, j, k, m, lastx, lasty: integer;
  pp, q, pq, p2, p2eqp3, face: boolean;
  parr1q; pt, pr, q, p2eqp3, vtl, vtyl, vtzt, vtx2, vty2, vtzt2: real;
  nrecords: integer;
  infile, outfile: str;
  abs_of_d: real;

PROCEDURE findvertex(VAR p: ptr; VAR vtx, vty, vtz: real; index: integer);

VAR
  nrecord, actindex: integer;
  i: integer;
BEGIN
  reset pointer to first record
  WHILE p^.bck < NIL DO
    BEGIN
      p := p^.bck
    END;
  END;

  calculate actual record
  nrecord := trunc(index/100);
  FOR i := 1 TO nrecord DO
    BEGIN
      p := p^.nxt
    END;
  IF ((index-nrecord*100)=0) THEN
    BEGIN
      actindex := index-(nrecord-1)*100
    END;
  ELSE
    BEGIN
      actindex := index-nrecord*100
    END;
  vtx := p^.vt[actindex,1];
  vty := p^.vt[actindex,2];
  vtz := p^.vt[actindex,3];
END;
FUNCTION backface(pn2,pn3,pn4:integer;var abs_of_d:real):boolean;
VAR
  a,b,c:real;i:integer;
  vt:ARRAY[1..3,1..3] OF real;
BEGIN
  findvertex(p,vt[1,1],vt[1,2],vt[1,3],pn2);
  findvertex(p,vt[2,1],vt[2,2],vt[2,3],pn3);
  findvertex(p,vt[3,1],vt[3,2],vt[3,3],pn4);
  FOR i=1 TO 3 DO
    BEGIN
      a[i]:=vt[3,i]-vt[2,i];
      b[i]:=vt[2,i]-vt[1,i];
      c[i]:=ep[i]-co[i];
    END;
    < a = v3 - v2 ; b = v2 - v1 ; c = ep - co >
    < form d = ( a cross b ) dot c ) >
    < front face if d > 0 >
    < back face if d < 0 >
    d:=(a[2][2]*b[3][3]-b[2][2]*a[3][3])*c[1][1]+(a[1][2]*b[3][3]-a[3][2]*b[1][3])*c[2][2]
    +(a[1][3]*b[2][2]-a[2][3]*b[1][2])*c[3][3];
    abs_of_d:=abs(d);
    IF d > 0
    THEN
      BEGIN
        backface:=true
      END
    ELSE
      BEGIN
        backface:=false
      END;
  END;
END;  < function backface >

BEGIN
  read(inp,vert);read(inp,poly);readln(inp);
  new(p);  < create new record >
  p^.back:=NIL;  < EOR to back field of 1st record >
  nrecords:=trunc(vert/100)+1;
  FOR i:=1 TO nrecords DO
    BEGIN
      IF (i=1)AND(1<(nrecords-1))
      THEN
        BEGIN
          FOR j:=1 TO 100 DO
            BEGIN
              read(inp,p^.vt[j,1]);read(inp,p^.vt[j,2]);read(inp,p^.vt[j,3]);
              readln(inp)
            END;
        END;
      ELSE
        BEGIN
          FOR j:=1 TO (vert-(nrecords-1)*100) DO
            BEGIN
              read(inp,p^.vt[j,1]);read(inp,p^.vt[j,2]);read(inp,p^.vt[j,3]);
              readln(inp)
            END;
        END;
    END.
END.
IF $i \leq n$ THEN
BEGIN
new(q); \text{ open new record } \\begin{align*}
p \cdot \text{nxt} & = q; \text{ pointer to next record} \\
a \cdot \text{bck} & = p; \text{ pointer to prev. record} \\
F & = q \text{ reset pointer}
\end{align*}
END;
\text{reading vertices}\n\begin{align*}
p \cdot \text{nxt} & = \text{NIL}; \text{ EOR to next field of last record } \\
\text{start plotting polygon by polygon}
\end{align*}
eps := 0.0001;
FOR $i = 1$ TO npoly DO
BEGIN
\begin{align*}
\text{read}(\text{inp}, \text{pn}[i]); \\
\text{for } n = 2 \text{ to } \text{pn}[i+1] \text{ do begin }
\text{read}(\text{inp}, \text{pn}[n])
\end{align*}
END;
\begin{align*}
\text{pn}[\text{pn}[i]+2] & = \text{pn}[2]; \\
k & = 0;
\end{align*}
\text{repeat}
\begin{align*}
\text{check for degenerate polygons, while backfacing} \\
\text{deg. polyg. : } \text{vertex}[i] = \text{vertex}[i+1] != \\
k & = k + 1; \\
\text{face} & = \text{backface}(\text{pn}[k+1], \text{pn}[k+2], \text{pn}[k+3], \text{abs of d});
\end{align*}
\text{until}(\text{abs of } d > 0.00001 \text{ or } (k = \text{pn}[i]-1));
\begin{align*}
\text{IF} (\text{abs of } d < 0.00001) \text{ THEN }
\text{BEGIN}
\text{END};
\text{IF} \begin{align*}
((\text{frontorbackface} = 'A') \text{ OR } \\
((\text{frontorbackface} = 'B') \text{AND face}) \text{ OR } \\
((\text{frontorbackface} = 'F') \text{AND}(\text{NOT face}))
\end{align*}
\text{THEN}
\text{BEGIN}
\text{END};
\text{BEGIN}
\text{lastx} := 0; \text{lasty} := 0;
\text{FOR } j = 2 \text{ TO } \text{pn}[i]+1 \text{ DO }
\begin{align*}
\text{single polygons} \\
k & = \text{pn}[i]+1; \\
\text{findvertex}(p, vtx1, vty1, vtz1, k); \\
\text{findvertex}(p, vtx2, vty2, vtz2, m); \\
\text{clipedges}(vtx1, vty1, vtz1, vtx2, vty2, vtz2, \\
xlp, ylp, x2p, y2p, mapWidth, scrready, ws); \\
p2eqp3 & = ((\text{abs(lastx-ylp)} < \text{eps}) \text{AND(}\text{abs(lasty-ylp)} < \text{eps)}) \\
p2eqp3 & = ((\text{abs(x2p-xlp)} < \text{eps}) \text{AND(}\text{abs(y2p-ylp)} < \text{eps}))
\end{align*}
\text{IF}(\text{NOT p2eqp3}) \text{ THEN }
\text{BEGIN}
\text{draw a line from p2 -- p3} \\
\text{IF}(\text{NOT p2eqp3}) \text{ THEN }
\text{BEGIN}
\text{moveto}(xlp, ylp) \\
\text{END;} \\
\text{lineto}(x2p, y2p)
\text{END.}
<draw a line from p2 --> p3>

END;

< single polygons >

END;

< delete vertex arrays >

WHILE p^.bck <> NIL DO
BEGIN
  pi=p^.bck
END;
WHILE p^.nxt <> NIL DO
BEGIN
  pi=p^.nxt;
  dispose(p^.bck)
END;
dispose(p)
END;
A.2 The Modules for Generating the Humanoid, ADAM

A program listing of the modules which generate the humanoid, see Fig. 3.20, is given below:

```pascal
version 1.1 MAY 13, 1986

<purpose>
<compiler directive>

<users guide>
main file: P:ADAM.PAS
work file: P:GRAPHMOD.PAS
GRAPHMOD.PAS contains the following procedures

\graph\miscello:
  InitPerspective
\graph\concavpo:
  GenCylinder
  GenSphere
\graph\primiv:
  MOVETO
  LINETO
  OpenSegment
  CloseSegment
  PostSegment
  UnPostSegment
\graph\fileop:
  PlotFile
  ConcatFile
  TransMat
\graph\matop:
  FrMatVec
  FrMatMat
  Umat3
  CopyMat
  CopyVec
  VecAddVecSubVec
  Try_angle
\graph\basics:
  ClipVertices
  ClipEdges

<programmer>
dieter longer
<date>
10-Feb-86
<function>
main program for plotting ADAM

type pointer = node:
  record
    name: str8;          < name of the record >
    o_abs, o_rel: arr3k3r; < abs. and rel. orientation mat.>
    origin_abs: arr3r;   < origin of actual node >
    l_k, r_k, r_left, l_left: arr3k3r;
    l_k_abs, r_k_abs, r_left_abs, l_left_abs: arr3r; < local vectors to initial and terminal joint >
    tharr3r;            < rel. or abs. orientation angles >
    up, down, left, right: pointer; < pointer to upper, lower, right and left record >
  end;

<users guide>
ADAM
ADAMINIT: initialization of pointer structures and position vect.
MasterFiles: creation of master files
ReadAngles: read angles from disk C, which describe the body segments' orientation
WorldFile: creation of world files
RectTreeWalker: recursively computes new world files
FindPointer: determines the pointer corresponding to a body segment's name
```
Print_Record : prints out all the information in the node records of ADAM
PlotFile : generates plot commands
DisplayFiles : generates segmented display files
InteractiveInp: starts interactive operation

//**************************

var latorsoptr,p,q,segmentptr:pointer;
i: integer;
relorabs_angles:boolean;
cmdch:char;
segmentno:integer;
done:boolean;
segmentName:str8;

<programmer : dieter longen >
<date : 10-feb-86 >
<function : this program reads file_body_parm and sets up the master files for ADAM, it initializes the pointer structures, which describe body segments >

procedure odom init(file_body_parm:str4;var latorsoptr:pointer);

var outx:outx{text;
outfile:str8texture:se4;
x+iarr3;
indivh,indivv:integer;
uptop,abottom,btop,bbottom,htoop:real;
body_sgm_parm:arr20k5r;
dist,dist_rib:real;
ursorsoptr,p,q:pointer;
offset,offset2:arr3r;

procedure MasterFiles(var body_sgm_parm:arr20k5r);

begin < MasterFiles >
  <open file_body_parm>
  extension:='.mast';assign(inp,file_body_parm);
  <check whether file_body_parm exists, see pg.116 Pascal Manual>
  readln(inp);<i+>
  ok:=(iresult = 0);
  if not ok then
  begin
    writeln;writeln;
    writeln('****** copy the file ','file_body_parm,' to drive ','drive_dat,'******');
    halt;
  end  
  <end I/O check >
  readln(inp);<neglect first (comment) line in file_body_parm >
  for i:=1 to 19 do
  begin
    read(inp,outfile);readln(inp);<segment name>
read(inp,height);read(inp,atop);read(inp,btop);read(inp,abottom);
read(inp,bbottom);read(inp,ndivh);read(inp,ndivv);read(inp);
body_segm_parm[1,13]=height;
body_segm_parm[1,23]=atop;
body_segm_parm[1,33]=btop;
body_segm_parm[1,43]=abottom;
body_segm_parm[1,53]=bbottom;

if ((outfile <> 'head') and (outfile <> 'HEAD')) then
begin
gencylinder(atop,abottom,btop,bbottom,height,ndivh,ndivv,drive_mst,
outfile,extension);
end
else
begin
gensphere(ndivh,ndivv,height,drive_mst,outfile,extension,out)
end
end;
close(inp);

{ combine files L0T0R501+L0T0R502 —> L0T0R50 }
offset[1][13]=0;offset[1][23]=0;offset[1][33]=0;
offset[2][33]=0;
concatfiles(drivemst,'lotorso1','mst',drivemst,'lotorso2','mst',
drivemst,'lotorso','mst',offset[1],offset[2]);

{ combine files UFT0R501+UFT0R502 —> UFT0R50 }
offset[2][23]=0;
offset[1][23]=(body_segm_parm[6,13]+body_segm_parm[7,13])/2;
concatfiles(drivemst,'uptorso1','mst',drivemst,'uptorso2','mst',
drivemst,'uptorso','mst',offset[1],offset[2]);

{ combine files LARM1+LARM2 —> LARM }
offset[1][23]=body_segm_parm[14,13]/2;
offset[2][23]=body_segm_parm[15,13]/2;
concatfiles(drivemst,'larm1','mst',drivemst,'larm2','mst',
drivemst,'larm','mst',offset[1],offset[2]);

{ combine files LLEG1+LLEG2 —> LLEG }
offset[1][23]=body_segm_parm[2,13/2];
offset[2][23]=body_segm_parm[3,13/2];
concatfiles(drivemst,'leg1','mst',drivemst,'leg2','mst',
drivemst,'leg','mst',offset[1],offset[2])
end; { MasterFiles }

begin { adoinit main procedure body }
MasterFiles(body_segm_parm);
{ initialization of pointer structures }

{ L0T0R50 }
dist:=0.2*body_segm_parm[8,13];
dist_rib:=0.2*body_segm_parm[8,13];

178
new(p) | l o t o r s o p t r = p;  
with p do  
begin  
down := nil; name := 'l o t o r s o p t r ;
k[1][1] := 0; k[2][2] := body_segm_parm[1,1,12]+dist_rib;k[3][3] := 0;  
l_left := body_segm_parm[1,1,2]+body_segm_parm[1,2,2];  
l_left := body_segm_parm[1,1,12];  
l_left[3][3] := 0;  
l_right := l_left;  
l_left[2][2] := l_left[2][2];  
l_right[3][3] := l_left[3][3];  
end;  
if (body_segm_parm[1,1,4] < 2*body_segm_parm[1,1,2]) then  
begin  
writeln;  
writeln('--------- PROCEDURE ADAMINIT ---------');  
writeln('legs do not fit under LOWER TORSO');  
writeln('body_segm[1,1,4] < 2*body_segm_parm[1,1,2]');  
writeln('1/2 width of lower torso');  
writeln('upper leg');  
end;  
{ upper left(p) and right(q) leg }  
new(p) | l o t o r s o p t r . l e f t := p;  
with p do  
begin  
right := nil; left := nil;  
n = 'u l l e g ' ; up := l o t o r s o p t r ;  
n = 'q'; l o t o r s o p t r . r i g h t := q; . right := nil; q . left := nil;  
k[1][1] := 0; k[2][2] := body_segm_parm[1,1,12]+dist_rib;k[3][3] := 0;  
k[1][1] := k[1][1]; k[2][2] := k[2][2]; k[3][3] := k[3][3];  
l[1][1] := 0; l[1][1] := body_segm_parm[1,1,12];  
end;  
{ upper right(p) and left(q) leg }  
new(p) | l o t o r s o p t r . l e f t := p;  
with p do  
begin  
right := nil; left := nil;  
n = 'q'; l o t o r s o p t r . r i g h t := q; . right := nil; q . left := nil;  
k[1][1] := k[1][1]; k[2][2] := k[2][2]; k[3][3] := k[3][3];  
l[1][1] := 0; l[1][1] := body_segm_parm[3,1,12];  
end;  
{ lower left(p) and right(q) foot }  
new(p) | l o t o r s o p t r . l e f t := p;  
with p do  
begin  
right := nil; left := nil;  
n = 'q'; l o t o r s o p t r . r i g h t := q; . right := nil; q . left := nil;  
k[1][1] := 0; k[2][2] := body_segm_parm[3,1,12];  
l[1][1] := 0; l[1][1] := body_segm_parm[3,1,12];  
end;  
{ lower right(p) and left(q) foot }  
new(p) | l o t o r s o p t r . l e f t := p;  
with p do  
begin  
right := nil; left := nil;  
n = 'q'; l o t o r s o p t r . r i g h t := q; . right := nil; q . left := nil;  
k[1][1] := 0; k[2][2] := body_segm_parm[3,1,12];  
l[1][1] := 0; l[1][1] := body_segm_parm[3,1,12];  
end;
new(p); ltorsooptr^.left^.down^.down:=p;
with p* do
begin
  name := 'lfoot';
  up := ltorsooptr^.left^.down;
  right := nil; left := nil;
end;

new(q); ltorsooptr^.right^.down^.down:=q; name := 'rfoot';
q^.up := ltorsooptr^.right^.down;
q^.right := nil; q^.left := nil;

k[13] := 0; k[23] := body_segm_parm[4, 13/2 + dist_k[33]] := 0;
q^.k[13] := k[13]; q^.k[23] := k[23]; q^.k[33] := k[33];
end;

< left (p) and right(q) toes >

new(p); ltorsooptr^.left^.down^.down:=p;
with p* do
begin
  name := 'ltoes';
  up := ltorsooptr^.left^.down;
  right := nil; left := nil;
end;

new(q); ltorsooptr^.right^.down^.down:=q; name := 'rtoes';
q^.up := ltorsooptr^.right^.down;
q^.right := nil; q^.left := nil;

k[13] := 0; k[23] := body_segm_parm[19, 13/2 + dist_k[33]] := 0;
q^.k[13] := k[13]; q^.k[23] := k[23]; q^.k[33] := k[33];
end;

{ RIB4 }

new(p); ltorsooptr^.up:=p;
with p* do
begin
  name := 'rib4';
  right := nil; left := nil; ltorsooptr;
end;

k[13] := 0; k[23] := body_segm_parm[10, 13/2 + dist_rib_k[33]] := 0;
end;

{ RIB3 }

new(p); ltorsooptr^.up^.up:=p;
with p* do
begin
  name := 'rib3';
  right := nil; left := nil; ltorsooptr^.up;
end;

k[13] := 0; k[23] := body_segm_parm[10, 13/2 + dist_rib_k[33]] := 0;
end;

{ RIB2 }

new(p); ltorsooptr^.up^.up^.up:=p;
with p* do
begin
end;

180
begin
name:='rib2';
right:=nil;left:=nil;down:=lotorsoptr.^up.^up.^up;
K[1][1]=0;K[2][1]=body_segm_parm[9,13/2]+dist_rib;K[3][1]=0;
L[1][1]=0;L[2][1]=body_segm_parm[9,13/2];L[3][1]=0
end;

< RIB1 >

new(p);lotorsoptr.^up.^up.^up.^up:=p;
with p do
begin
name:='rib1';
right:=nil;left:=nil;down:=lotorsoptr.^up.^up.^up.
K[1][1]=0;K[2][1]=body_segm_parm[8,13/2]+dist_rib;K[3][1]=0;
L[1][1]=0;L[2][1]=body_segm_parm[8,13/2];L[3][1]=0
end;

< upper torso >

new(p);lotorsoptr.^up.^up.^up.^up:=p;
with p do
begin
name:='uptorso';
down:=lotorsoptr.^up.^up.^up.^up;
uptorsoptr:=p;
K[1][1]=0;K[2][1]=body_segm_parm[6,13/2]+body_segm_parm[7,13/2]+dist;
K[3][1]=0;
L[1][1]=body_segm_parm[7,13/2];L[2][1]=body_segm_parm[7,13/2];L[3][1]=0;
L_left[1][1]=body_segm_parm[7,4]+body_segm_parm[13,23];
L_left[2][1]=body_segm_parm[7,13/2];L_left[3][1]=0;
L_right[1][1]=1_left[1][1];
L_right[2][1]=1_left[2][1];L_right[3][1]=1_left[3][1]
end;

if (body_segm_parm[7,4] < body_segm_parm[8,23]+body_segm_parm[13,23])
then
begin
writeln writeln('------------------- PROCEDURE ADAMINIT -------------------');
writeln(' ARMS do not fit under UPPER TORSO '); writeln(' or 
writeln(' decrease body_seg[8,2] : 1/2 width of rib
writeln END;

< HEAD >

new(p);uptorsoptr.^up:=p;
with p do
begin
name:='head';
down:=uptorsoptr.^up:=nil;left:=nil;right:=nil;
K[1][1]=0;K[2][1]=body_segm_parm[5,13/2];K[3][1]=0
end;
< UPPER LEFT(p) and RIGHT(q) ARM >

new(p):!uptorsooptr^\left!=p
\textbf{with} p^ do
\textbf{begin}
\textbf{right}=nil^\left!=nil;
\textbf{name}='ularm'!\uparrow!\textbf{uptorsooptr};
\textbf{new}(q):!uptorsooptr^\right!=qi^\right!=nil^\right!=nil;
\textbf{name}='urarm'!\uparrow!\textbf{uptorsooptr};
\textbf{end};
\textbf{k1}_1=0;\textbf{k2}_1=\textbf{body}_seg_porm[13,13/2]+\textbf{dist}[k3]=0;
\textbf{k}^\_\textbf{qi}_1=k\_\textbf{qi}_1^\_\textbf{qi}_1\textbf{k}_2^\_\textbf{qi}_2^\_\textbf{qi}_2\textbf{.k}_3^\_\textbf{qi}_3^\_\textbf{qi}_3;
\textbf{if}_1=0;\textbf{if}_2=\textbf{body}_seg_porm[13,13/21]+\textbf{dist}[k3]=0;
\textbf{qi}^\_\textbf{qi}_1=\textbf{qi}^\_\textbf{qi}_1^\_\textbf{qi}_1\textbf{qi}^\_\textbf{qi}_2^\_\textbf{qi}_2\textbf{qi}^\_\textbf{qi}_3^\_\textbf{qi}_3^\_\textbf{qi}_3;
\textbf{end};
\textbf{< LOWER LEFT(p) and RIGHT(q) ARM >}

new(p):!uptorsooptr^\left^\downarrow!=nil;
\textbf{with} p^ do
\textbf{begin}
\textbf{right}=nil^\left!=nil;
\textbf{name}='llarm'!\uparrow!\textbf{uptorsooptr}^\left^\downarrow;
\textbf{new}(q):!uptorsooptr^\right^\downarrow!=qi^\right^\downarrow!=nil^\right^\downarrow!=nil;
\textbf{name}='lurarm'!\uparrow!\textbf{uptorsooptr}^\right^\downarrow;
\textbf{end};
\textbf{k1}_1=0;\textbf{k2}_1=\textbf{body}_seg_porm[14,13/2]+\textbf{dist}[k3]=0;
\textbf{k}^\_\textbf{qi}_1=k\_\textbf{qi}_1^\_\textbf{qi}_1\textbf{k}_2^\_\textbf{qi}_2^\_\textbf{qi}_2\textbf{.k}_3^\_\textbf{qi}_3^\_\textbf{qi}_3;
\textbf{if}_1=0;\textbf{if}_2=\textbf{body}_seg_porm[15,13/21]+\textbf{dist}[k3]=0;
\textbf{qi}^\_\textbf{qi}_1=\textbf{qi}^\_\textbf{qi}_1^\_\textbf{qi}_1\textbf{qi}^\_\textbf{qi}_2^\_\textbf{qi}_2\textbf{qi}^\_\textbf{qi}_3^\_\textbf{qi}_3^\_\textbf{qi}_3;
\textbf{end};
\textbf{< LEFT(p) and RIGHT(q) PALM >}

new(p):!uptorsooptr^\left^\downarrow^\downarrow!=nil;
\textbf{with} p^ do
\textbf{begin}
\textbf{right}=nil^\left!=nil;
\textbf{name}='lpalarm'!\uparrow!\textbf{uptorsooptr}^\downarrow^\downarrow;
\textbf{new}(q):!uptorsooptr^\downarrow^\downarrow!=qi^\downarrow^\downarrow!=nil^\downarrow^\downarrow!=nil;
\textbf{name}='rpalm'!\uparrow!\textbf{uptorsooptr}^\downarrow^\downarrow;
\textbf{end};
\textbf{k1}_1=0;\textbf{k2}_1=\textbf{body}_seg_porm[16,13/2]+\textbf{dist}[k3]=0;
\textbf{k}^\_\textbf{qi}_1=k\_\textbf{qi}_1^\_\textbf{qi}_1\textbf{k}_2^\_\textbf{qi}_2^\_\textbf{qi}_2\textbf{.k}_3^\_\textbf{qi}_3^\_\textbf{qi}_3;
\textbf{if}_1=0;\textbf{if}_2=\textbf{body}_seg_porm[16,13/21]+\textbf{dist}[k3]=0;
\textbf{qi}^\_\textbf{qi}_1=\textbf{qi}^\_\textbf{qi}_1^\_\textbf{qi}_1\textbf{qi}^\_\textbf{qi}_2^\_\textbf{qi}_2\textbf{qi}^\_\textbf{qi}_3^\_\textbf{qi}_3^\_\textbf{qi}_3;
\textbf{end};
\textbf{< LEFT(p) and RIGHT(q) FINGERS >}

new(p):!uptorsooptr^\left^\downarrow^\downarrow^\downarrow!=nil;
\textbf{with} p^ do
\textbf{begin}
\textbf{right}=nil^\left!=nil;
\textbf{name}='lpalarm'!\uparrow!\textbf{uptorsooptr}^\downarrow^\downarrow^\downarrow;
\textbf{new}(q):!uptorsooptr^\downarrow^\downarrow^\downarrow!=qi^\downarrow^\downarrow^\downarrow!=nil^\downarrow^\downarrow^\downarrow!=nil;
\textbf{name}='rfingers'!\uparrow!\textbf{uptorsooptr}^\downarrow^\downarrow^\downarrow;
\textbf{end};
<program>

{ function : read angles of body segments from file }
{ order : }
{ 1st line : LTOHSG/R1H3/R1B2/R1B1/UTORSG/ }
{ HEAT/ULLEG/LLEG/RFDOT/RFOES/ULLEG/ }
{ LLEG/LFDOOT/RDOES/ULARM/LLARM/LHAND/ }
{ LFINGER/URARM/LRAM/RRHAND/RFINGER }
{ 5th line : ------empty------- }
{ 6th line : th1,th2,th3,.... }
{ line : ... }
{ nrecords indicate how many lines are read from }
{ the input file }

procedure readangles(input_angles: string; leftorsoptr:pointer; nrecords: integer);

var
  i: integer;
procedure readinput(leftorsoptr:pointer);
var
  p, up: torsoptr; pointer;
begin
  p:=leftorsoptr;
  < read first three values as center of interest values >
  read(inp,coi[1]);read(inp,coi[2]);read(inp,coi[3]);
  < read angles from complete torso and head >
  while p <> nil do
    begin
      read(inp,p^.th[1233]);read(inp,p^.th[2333]);read(inp,p^.th[3333]);
      p:=p^.up
      end;
  < read angles for right leg >
p:=leftorsoptr^.right;
  while p <> nil do
    begin
      read(inp,p^.th[1233]);read(inp,p^.th[2333]);read(inp,p^.th[3333]);
      p:=p^.down
      end;
  < read angles for left leg >
p:=leftorsoptr^.left;
end;
while p <> nil do
begin
  read(inp,p.th[13]);read(inp,p.th[23]);read(inp,p.th[33]);
  p:=p.down
end;

uptorsoptr:=lotosoptr.up^up^up^up^up;

<read angles for left arm>

p:=uptorsoptr^left;
while p <> nil do
begin
  read(inp,p.th[13]);read(inp,p.th[23]);read(inp,p.th[33]);
  p:=p.down
end;

<read angles for right arm>

p:=uptorsoptr^right;
while p <> nil do
begin
  read(inp,p.th[13]);read(inp,p.th[23]);read(inp,p.th[33]);
  p:=p.down
end;
readln(inp)
end;  <read input>

begin  <read angles>
  assign(inp,input_angles);
  <check whether file_body_pars exists, see pg.116 Pascal Manual>
  <$I-> reset(inp)<$I>
  ok:=(iresult = 0);
  if not ok then
  begin
    writeln;
    writeln('**** File ',input_angles,' does not exist ****');
    halt
  end;
  <end I/O check>

  readln(inp)readln(inp)readln(inp)readln(inp)readln(inp);
  <neglect 5 comment lines in file input_angles>
  for i:=1 to nrecords do
  begin
    readinput(lotosoptr)
  end;
  close(inp)
end;  <read angles>

-----------------------------------------------------------------------------------
<programmer : dieter langner>
<date    : 04-march-1986>
<function  : reset file Name.Hst for input
             open file Name.Wld for output
             compute world files
             close input and output channel>
procedure WorldFile(pipointer);
var done:boolean;
name, mstrFileName: strB;
inFile, outFile: strB;

procedure world_file(p: pointer):;
var
d: arr3; lptr, upr: pointer; n: strB;
beg
with p do
begin
<record loop>
if name<> name then
<compute orientation matrix>
bry_ang(th11, th23, th33, a.re1); lptr= i slaughterForm(u=pi, up=up, up=up, up=up);<pointer to uptorso>
if (n=lptr.name) then
begin
<LOWER TORSO>
copymat(a.re1, a.abs);
prmatvec(a.abs, k, k.abs);
prmatvec(a.abs, l, l.abs);
prmatvec(a.abs, l.right, l.right.abs);
prmatvec(a.abs, l.left, l.left.abs);<origin of LOWER TORSO in the origin of world coord.>
tran_rot(origin.abs, a.abs)
end
ELSE
begin
IF ((n=up.name) or (n=up.name) or (n=up.name) or (n=up.name) or (n=up.name) or (n=up.name))
THEN
begin
<KIK4, RIK3, RIK2, RIK1, UPTORSO, and HEAD>
if relorabs_angles then
begin
begin prmatmat(a.re1, down.abs, a.abs)
end
else
begin copymat(a.re1, a.abs) end;
prmatvec(a.abs, l, l.abs);
prmatvec(a.abs, k, k.abs);
if name<>u.name then
begin
prmatvec(a.abs, l.left, l.left.abs);
prmatvec(a.abs, l.right, l.right.abs)
end;
vecaddvecsubvec(down.abs, down.abs, k.abs, l.abs, origin.abs); tran_rot(origin.abs, a.abs)
end
ELSE
begin
<ULL, LLLEG, RFoot, LTOES, UUPARM, URARM, RHAND, RFINGERS>
<ULL, LLLEG, LFOOT, LTOES, ULPARM, ULARM, LHAND, LFINGERS>
if relorabs_angles then
begin
begin prmatmat(a.re1, up.up, a.abs, a.abs) end
else
begin copymat(a.re1, a.abs) end;
prmatvec(a.abs, k, k.abs);
prmatvec(a.abs, l, l.abs);
if ((n=up.left.name) or (n=lptr.left.name)) then
begin copyvec(up^l_left_abs,d) end
else
begin
IF ((n=^l.right.name) or (n^lptr^r.right.name)) then
begin copyvec(up^l_right_abs,d) end
else
begin copyvec(up^l_abs,d) end
end
{ origin^up + d^up = k^abs[1] = origin[1] } { origin_abs^l^left_abs^l_left_d^l_left^d^l_left^l^left_abs^d }
vecaddvecsubvec(up^origin_abs^d^k^abs^origin_abs) { setting up master files }{ tran_rot(origin_abs^d^k^abs) }{ record loop }
end
end<world_files>
begin <main procedure body World File>
{ reset Master files and open World Files }
imstfilename="/name=n^p^name;
if ((name=\'lotorso\') or (name=\'rib4\') or (name=\'rib3\') or (name=\'rib2\') or (name=\'rib1\') or (name=\'uptorso\') or (name=\'head\')) THEN begin mstfilename=\'name end
ELSE
begin
if ((name=\'urleg\') or (name=\'ulleg\')) then begin mstfilename=\'uleg\' end
ELSE
begin
if ((name=\'l1leg\') or (name=\'r1leg\')) then begin mstfilename=\'lleg\' end;
if ((name=\'lfoot\') or (name=\'rfoot\')) then begin mstfilename=\'foot\' end;
if ((name=\'ltoes\') or (name=\'rtoes\')) then begin mstfilename=\'toes\' end;
if ((name=\'larm\') or (name=\'larm\')) then begin mstfilename=\'larm\' end;
if ((name=\'rarm\') or (name=\'rarm\')) then begin mstfilename=\'rarm\' end;
if ((name=\'lfo\') or (name=\'rfo\')) then begin mstfilename=\'fo\' end;
if ((name=\'lpalm\') or (name=\'rpalm\')) then begin mstfilename=\'palm\' end;
if ((name=\'lfinger\') or (name=\'rfinger\')) then begin mstfilename=\'finger\' end
end
if (length(mstfilename) > 0 ) then
begin <the name is an actual Segment Name >
inpfile=concat(drive\_ms\_mstfilename,\'.\_mst\');
assign(inp,inpfile);reset(inp);
outfile=concat(drive\_wd\_name,\'.\_wd\');
assign(out,outfile);rewrite(out);
world_file=p;
close(inp);close(out)
end
ELSE
begin { the name is not a Segment Name }
write(( '---------------------- WORLDFILE ----------------------'))
write(( 'you did it again !--->F^\text{name},<- inputted to worldfile ') )
write(( 'is not the name of a bodysegment '))
end

end! < WorldFile >

-------------------------------------------------------------------------------------------
< 3-march-1986
< function ! input of segment name, the program then calculates
< all world files , which are affected if the angles
< in the actual segment are changed

procedure RecTreeWalker(p:pointer);

var 1,up:pointer;

function StopRecursion(q:pointer):boolean;

begin stopRecursion=false
with q do
begin
if (left=nil) and (right=nil) and
((up=nil) or (down=nil)))
then begin stopRecursion=true end
end
end! < stopRecursion >

begin { main procedure body RecTreeWalker >
if stopRecursion(p) ( head, toes, fingers )
THEN
begin worldFile(p) end
ELSE
begin { recursion }
1:lower
1^\text{lower torso}:1^\up,1^\up,1^\up,1^\up,1^\up,1^\up;
if ((p=1) or (p=1^\up) or (p=1^\up,1^\up) or (p=1^\up,1^\up,1^\up) or
(p=1^\up,1^\up,1^\up,1^\up,1^\up,1^\up)
THEN
begin { upward directed segments }
if ((p=1) ( lower torso ) or (p=1) ( upper torso ))
THEN
begin
worldFile(p);RecTreeWalker(p^\text{left});
RecTreeWalker(p^\text{right});
RecTreeWalker(p^\text{up})
end
ELSE
begin { rib4 -- rib1 }
worldFile(p);RecTreeWalker(p^\text{up})
end
end { upward directed segments }
ELSE
begin { downward directed segments }
worldFile(p);RecTreeWalker(p^\text{down})
end { downward directed segments }
end { recursion }
end! < RecTreeWalker >

{-------------------------------------------------------------------------------------------}
procedure FindFointer(segmentName: string; var p:pointer);
var headp: pointer;

procedure RecFindFointer(q:pointer);
begin
  if ((p<>nil) and (q<>nil)) (termination condition) then
  begin
    if (q^ name=segmentName) then
      begin p:=q end
    ELSE
      begin (recursion)
        RecFindFointer(q^ .down);
        RecFindFointer(q^ .left);
        RecFindFointer(q^ .right);
      end (recursion)
  end;
begin (main procedure body FindFointer)
p:=nil;headptr:=loptr^ .up^ .up^ .up^ .up^ .up;
RecFindFointer(headptr);
end (FindFointer)

{FUNCTION : testing of the contents of the records containing the}
{body segment description}

procedure print_records;

procedure printrecord(p:pointer);
var row,col:integer;
begin
  with p do
  begin
    writeln(lst,'************ record : ');
    writeln(lst);writeln(lst,'A_ABS');
    for row:=1 to 3 do
      begin
        for col:=1 to 3 do
          begin
            write(lst,a_abs[row,col]:12:3)
            writeln(lst);
          end;
    writeln(lst);writeln(lst,'A_REL');
    for row:=1 to 3 do
      begin
        for col:=1 to 3 do
          begin
            write(lst,a_rel[row,col]:12:3)
            writeln(lst);
          end;
  end;
begin write ln (lst); write ln (lst, ' L K L_RIGHT L_LEFT ORIGIN_');
FOR row = 1 to 3 do
begin
write ln (lst, row:12:3, K[row]:12:3, L[row]:12:3, origin_obs[row]:12:3);
end; write ln (lst);
write ln (lst, ' L_ABS K_ABS L_RIGHT_ABS L_LEFT_ABS');

FOR row = 1 to 3 do
begin
write ln (lst, row:12:3, K_ABS[row]:12:3, L_ABS[row]:12:3, origin_obs[row]:12:3);
end; write ln (lst);

angles th1, th2, th3;
write ln (lst, ' angles th1, th2, th3 ');
write ln (lst, 'p o i n t e r U P — > ', up*.name);
write ln (lst, 'p o i n t e r D O U N — > ', down*.name);
write ln (lst, 'p o i n t e r LEFT — > ', left*.name);
write ln (lst, 'p o i n t e r RIGHT — > ', right*.name);

end;
procedure plotfile(point); var inpfile: file of string;
begin
  (PlotFile)
  (open input file)
  inpfile:=concat(drive, p.name); assign(inp, inpfile);
  reset(inp);
  clippedin3d;
  (close input file)
  close(inp);
end; (PlotFile)

procedure DisplayFiles(segmentno: integer);
begin
  (generate segmented plot files)
  (1st segment : TOKSO and HEAD)
  OpenSegment(segmentno);
  (initialize graphics generator first)
  initgraphics;
  p:=LOTOKSOFTK;
  plotfile(p);
  plotfile(p.up); plotfile(p.up.up);
  plotfile(p.up.up.up); plotfile(p.up.up.up.up);
  plotfile(p.up.up.up.up.up); plotfile(p.up.up.up.up.up.up);
  (CloseSegment;)
  (2nd segment : LEFT LEG)
  OpenSegment(segmentno+1);)
  plotfile(p.left); plotfile(p.left.down);
  plotfile(p.left.down.down); plotfile(p.left.down.down.down);
  (CloseSegment;)
  (3rd segment : RIGHT LEG)
  OpenSegment(segmentno+2);)
  plotfile(p.right); plotfile(p.right.down);
  plotfile(p.right.down.down); plotfile(p.right.down.down.down);
  (CloseSegment;)
  (4th segment : LEFT ARM)
  OpenSegment(segmentno+3);)
  p:=lotorsoptr.up.up.up.up.up;
procedure SetAngles(var segmentName: str8; var segmentPtr: pointer; var i: integer; done: boolean);
begin
  done := false;
  while not done do
  begin
    writeln(' Set Segment Angles interactively; segments are named ');
    writeln(' as shown below ');
    writeln(' ltorso[rib4|rib3|rib2|rib1|uptorsoihead: ');
    writeln(' urleg|llleg|foot|toes: ');
    writeln(' ularm|llarm|palm|fingers: ');
    writeln(' urarm|llorm|palm|fingers: ');
    write(' Segment name = ? ');
    repeat
      buffer := 8;
      readln(segmentName);
      until length(segmentName) > 0; (segmentName contains at least 1 char)
    if segmentName <> '' then done := true
  end;
  FindPointer(segmentName, segmentPtr);
  with segmentPtr^ do
  begin
    for i := 1 to 3 do
    begin
      write(' theta[', i, ' ] = '); readln(th[i]);
      th[i] := th[i] * pi / 180
    end
  end;
end;

procedure InteractiveInp;
var flname : packed array [1..64] of str14;
  name : str8;
  nfiles : integer;
begin
  while not done do
  begin
    writeln(' Set Segment Angles interactively; segments are named ');
    writeln(' as shown below ');
    writeln(' ltorso[rib4|rib3|rib2|rib1|uptorsoihead: ');
    writeln(' urleg|llleg|foot|toes: ');
    writeln(' ularm|llarm|palm|fingers: ');
    writeln(' urarm|llorm|palm|fingers: ');
    write(' Segment name = ? ');
    repeat
      buffer := 8;
      readln(segmentName);
      until length(segmentName) > 0; (segmentName contains at least 1 char)
    if segmentName <> '' then done := true
  end;
  FindPointer(segmentName, segmentPtr);
  with segmentPtr^ do
  begin
    for i := 1 to 3 do
    begin
      write(' theta[', i, ' ] = '); readln(th[i]);
      th[i] := th[i] * pi / 180
    end
  end;
end;
repeat
  writeln('Begin SetAngles Change Fersp. Read File',
    'Print records Quit');
  write('B/S/C/R/F/Q = '); readln(cmdch);
  until cmdch in ['B','S','C','R','F','Q'];
case cmdch of
'F':
  print_records;
  'K':
    case cmdch of
      'B':
        begin
          // initialization of body segment parameters and computation
          // (of master files)
          // adminit('clssegm.dat', lots soptr);
          writeln(Enter file name to read data from filename.dat : )
          readln(name);
          // filename[1] := concat(name, '.dat');
          // read one record of file biseg_ang.dat
          readangles(filename[1], lots soptr[1]);
          // (读取文件) read angles from filename[1], lots soptr[1]
          // initialize all worldfiles
          RecTreeWalker(lots soptr[1]);
          // PRINT_RECORDS:
          repeat
            write('segment number [1..64] = ? ');
            readln(segmentno);
            until segmentno in [1..64];
            displayfiles(segmentno);
          end;
        end;
'S':
  begin
    SetAngles(segmentName, segmentptr);
    RecTreeWalker(segmentptr);
    repeat
      write('segment number [1..64] = ? ');
      readln(segmentno);
      until segmentno in [1..64];
      displayfiles(segmentno);
    end;
'C':
  begin
    InitPerspective;
    // PRINT_RECORDS:
    RecTreeWalker(lots soptr[1];
    repeat
      write('segment number [1..64] = ? ');
      readln(segmentno);
      until segmentno in [1..64];
      displayfiles(segmentno);
    end;
'O':
  begin
    done:=true
  end;
end;

192
'R' begin repeat write('Enter number of file you would like read in [1..64] : '); readln(nofiles) until (nofiles >= 1) and (nofiles <= 64); for i := 1 to nofiles do begin write('Enter file name to read data from [filename.dat] : '); readln(name); filename[i] := concat(name, '.dat') end; for i := 1 to nofiles do begin < read one record of file > readangles(filename[i],latorsoptr,i); mapWlatoScrready := false; < initialize all worldfiles > RecTreeWalker(latorsoptr); < PRINT_RECORDS > segmentno := i; displayfiles(segmentno) end end ( case statement ) end ( while loop ) end; ( InteractiveInp ) (-----------------------------------------------) (-----------------------------------------------) begin ( ADAM main program body ) drive_dot := 'c:'; drive_ extinguisher := 'c:'; drive_wld := 'c:'; drive_plt := 'd:'; < relative angles are inputted > relorabs_angles := true; done := false; InteractiveInp; textcolor(9) end. ( adam main program body ) (-----------------------------------------------)
A.3 The Modules for Animation

A program listing of the subroutines which accomplish the animation of the humanoid are given below:

```
program interpolation(input,output,inp, out);

const
  maxpivots = 20;
  pi = 3.141592654;

type
  str = string[20];
  io_type = text;
  pointertype = ^nodetype;
  nodetype = record
    value : real;
    right : pointertype;
    down : pointertype;
  end;
  vec_type = array [1..maxpivots] of real;
  mat_type = array [1..maxpivots] of vec_type;

var
  inp : text;
  out : text;
  inpfile : str;
```
filesout : array [1..64] of str;
datapntr,dpntr,trcpntr,vec2pntr,pivpntr : pointer;
tracks, msteps, npivots, val_number : integer;
fourx4 : mat_type;
matrix : mat_type;
vector, vec4 : vec_type;
tangent : vec_type;
inter_type : char;
input_dev : char;
disk : string (2);
infile, outfile : string [8];

< Reads data in from a previously assigned file >
< or from the keyboard >

procedure read_in_data;
var
  pointer, pointer1, pointer2 : integer;
  temp : integer;

begin

  { loop to read in each pivot of data }
  for pointer2 := 1 to npivots do begin
    temp := round(ntracks/18 + 0.5);
    pointer := 1;

    { loop to read in data points for each pivot }
    { reads 18 values off each line }
    for pointer1 := 1 to temp do begin
      while (pointer <= (18:pointer1)) and not (pointer > ntracks) do begin
        new(datapntr);

        { if first pivot then set up pointer values }
        if pointer = 1 then begin
          if (pointer2 = 1) then begin
            dpntr := datapntr;
            trcpntr := datanptr;
          end
          else begin
            vec2pntr := pivpntr;
            pivpntr := datapntr;
            trcpntr := datanptr;
          end;

          { read in a data value }
          if input_dev = 'F' then
            read(inp, datapntrA.value)
          else begin
            write('enter next data value : ');
            read(datapntrA.value);
          end
        end
      end;
  end;

end;}
\begin{verbatim}
writeln
end;

{increment variable pointer and pointer structures}

pointer := pointer + 1;
if (pointer2 <> 1) then
  if pointer2 <> npivots then
    begin
      trc2pntr^.right := datapntr;
      trc2pntr := trc2pntr^.down
    end
  else
    begin
      trc2pntr^.right := datapntr;
      trc2pntr := trc2pntr^.right^.right = nil;
      trc2pntr := trc2pntr^.down
    end;

  trcpntr^.down := datapntr;
  trcpntr := datapntr
end;

{set last pointer in each pivot to nil}

trcpntr^.down := nil
end
end;

{reads interpolation type, number of tracks, number of step, and}
{number of pivots if nonlinear, from a file previously defined}

procedure read_info;
  var one_char : char;
begin
reset(inp);
  read(inp, one_char);
  writeln('reading in file');
  while not((one_char = 'L') or (one_char = 'N')) do
    begin
      readln(inp);
      read(inp, one_char)
    end;

  inter_type := one_char;
  if inter_type = 'N' then
    begin
      read(inp, ntracks, nsteps, npivots);
      writeln('number of tracks = ', ntracks, 'number of steps = ', nsteps,
               'number of pivots = ', npivots)
    end
  else
    begin
      read(inp, ntracks, nsteps);
      writeln('number of tracks = ', ntracks, 'number of steps = ', nsteps);
      npivots := 2
    end;
end;
\end{verbatim}
read_in_data
end;

< creates all files needed for output >
< the number of files needed is >
< (number of pivots - 1) times number of steps + 1 >

procedure create_files;

   type
      add_type = string [2];

   var
      I : integer;
      ser_num : add_type;

   < receives a number "in number" and converts it to an alphanumeric >
   < string >

   procedure characterize(number : integer; var addition : add_type);
      var
         times, loop, num : integer;
         char1 : char;

      begin
         addition := '';
         if number > 9 then loop := 2 else loop := 1;
         for times := 1 to loop do
            begin
               num := number - trunc(number/10)*10;
               case num of
                  0: char1 := '0';
                  1: char1 := '1';
                  2: char1 := '2';
                  3: char1 := '3';
                  4: char1 := '4';
                  5: char1 := '5';
                  6: char1 := '6';
                  7: char1 := '7';
                  8: char1 := '8';
                  9: char1 := '9';
               end;
               number := trunc(number/10);
            end;
         end;

      begin
         for I := 1 to nsteps*(n pivots-1) + 1 do
            begin
               < get alphanumeric string of I's value >
               characterize(I, ser_num);
               filesout[1] := concat(disk, out file, ser_num, '.dat');
               assign(out, filesout[1]);

               rewrite(out); < creates file and clears contents >
               writeln(out, ''); writeln(out, ''); writeln(out, ''); writeln(out, ''); writeln(out, '');
               close(out);
            end;
         end;

   end;

197
procedure get_info;
begin
repeat
write ('Input device : keyboard or file [K/F] : '); readln(input_dev);
until (input_dev = 'K') or (input_dev = 'F'); writeln;
if input_dev = 'F' then
begin
write ('Drive [A:/B:/C:/D:] = '); readln(disk);
write ('Name of input file [in_file.dat], in_file = '); readln(in_file);
assign(in_file, in_file);
write ('Name of output file [out_file.dat], out_file = '); readln(out_file);
writeln;
read_info
end
else
begin
repeat
write ('Linear or Non-linear interpolation [L/N] : '); readln(inter_type);
until (inter_type = 'N') or (inter_type = 'L'); writeln;
write ('Number of tracks [1..ntracks], ntracks = '); readln(ntracks);
writeln;
write ('Number of interpolation steps [1..nsteps], nsteps = '); readln(nsteps);
writeln;
if inter_type = 'N' then
begin
write ('Number of pivot points [1..npivots], npivots = '); readln(npivots);
writeln
end
else
npivots := 2;
write ('Name of output file [out_file.dat], out_file = '); readln(out_file);
read_in_data
end;
end;

{ gets information as to where data is coming from; }
{ keyboard or file, and the interpolation type, tracks }
{ steps and pivots if input is from keyboard }

procedure form_4_..4;
var
  count1, count2 : integer;
begin
  fourx4[1,1] := 2;
  fourx4[1,2] := -2;
end.
for count1 := 1 to max pivots do
  for count2 := 5 to max pivots do
    fourx[count1, count2] := 0;
for count1 := 5 to max pivots do
  for count2 := 1 to 4 do
    fourx[count1, count2] := 0;
end;

< multiplies a matrix by a vector to return a vector >

procedure multiply(size: integer; mat: mat_type; vec: vec_type; var tan: vec_type); var row: integer;
col: integer;
begin
  < clear tan vector >
  for row := 1 to size do
    tan[row] := 0;
< go through all rows of matrix >
  for row := 1 to size do
    begin
      tan[row] := 0;
      for col := 1 to size do
      end;
  end;
< forms a matrix depending on the number of pivots >

procedure form_matrix; var count1: integer;
count2: integer;

(* SUBROUTINE: MATINV.FAS *)
(* PROGRAMMER: HELMUT BUCHNER, DIETER LANGER *)
(* DATE: JUNE 4, 1982 *)
(* FUNCTION: INVERT THE INERTIA MATRIX */
(* USER GUIDE: MATINT(A,I))
(* PROGRAMMER
(* GUIDE: )FAS MATINU=MATINU
(* >MAC MATINT=MATINT
(*
(*
(* PROCEDURES
(* CALLED: NONE
(*
(* GLOBAL VARIABLES
(* REFERENCED: NONE
(*
(* GLOBAL VARIABLES
(* MODIFIED: NONE
(*
*******************************************************************************************/

PROCEDURE invmatrix(VAR A: mat_type);
LABEL 100,110;
VAR  I,IJ,IK,IZ,J,J1,JF,JG,K,K1,KJ,KK,N,NK: INTEGER;
    B: INTEGER;
    L,M: ARRAY[1..npivots] OF INTEGER;
    D : REAL;
BEGIN: ( MATINV )

(* SEARCH FOR THE LARGEST ELEMENT *)

N := npivots-2;
D := 1.0;
Nh := -N;
FOR K := 1 TO N DO
  BEGIN
    Nh := Nh + N;
    L[K]:= K;
    M[N]:= K;
    Nh := Nh + N;
    BIGA := ACler[KK];
  END
FOR J:=K TO N DO
  BEGIN
    IZ := N # (J-1);
    FOR I:=K TO N DO
      BEGIN
        IJ := IZ + I;
        IF ABS(BIGA) = ABS(A[I,J])-0.0 THEN
BEGIN
BIGA := A[I,J]
L[K] := I;
M[K] := J;
END;
END;

(* INTERCHANGE ROWS *)

J := L[K];
IF J-K > 0 THEN
BEGIN
K1 := K - N;
FOR I := 1 TO N DO
BEGIN
K1 := K1 + N;
HOLD := -A[K1,J];
J1 := K1 - K + J;
A[K1,J] := A[J1,J];
A[J1,J] := HOLD;
END;
END;

(* INTERCHANGE COLUMNS *)

I := M[K];
IF I - K > 0 THEN
BEGIN
JP := N * (I-1); 
FOR J := 1 TO N DO
BEGIN
JN := NK + J;
J1 := JP + J1;
HOLD := -A[J,J1];
A[J1,J] := HOLD;
END;
END;

(* DIVIDE COLUMNS BY MINUS PIVOT (VALUE OF PIVOT ELEMENT IS CONTAINED IN BIGA) *)

IF BIGA = 0.0 THEN
BEGIN
D := 0.0;
GOTO 110;
END ELSE
BEGIN
FOR I := 1 TO N DO
BEGIN
IF I-K < 0 THEN
BEGIN
IN := NK + I;
AC[I,K] := AC[I,K] / (-BIGA);
END;
END;

201
(* REDUCE MATRIX *)

FOR I:=1 TO N DO
BEGIN
  IH := IH + I;
  HOLD := A[I,H];
  IJ := I-N;
  FOR J:=1 TO N DO
  BEGIN
    IJ := IJ + N;
    IF I-K <> 0 THEN
    BEGIN
      IF J-K <> 0 THEN
      BEGIN
        KJ := IJ - I + K;
      END;
      END;
    END;
  END;
END;

(* DIVIDE ROW BY PIVOT *)

KJ := K - N;
FOR J:=1 TO N DO
BEGIN
  KJ := KJ + N;
END;

(* PRODUCT OF PIVOTS *)

D := D * BIGA;

(* REPLACE PIVOT BY RECIPROCAL *)

ACK,KJ := 1.0 / BIGA;
END;

(* FINAL ROW AND COLUMN INTERCHANGE *)

K := N;
K := K-1;
IF K <> 0 THEN
BEGIN
  I := L[K];
  IF I-K <> 0 THEN
  BEGIN
    J0 := N * (K-1);
    JR := N * (I-1);
    FOR J:=1 TO N DO
    BEGIN
      JN := J0 + J;
      HOLD := A[J,J];
      JJ := JR + J;
      ACK,KJ := -ACK,JJ;
      ACK,J := HOLD;
    END;
  END;
END;

END;
Ji = M(K)j
IF J - K > 0 THEN
BEGIN
K1 := K - Nj
FOR I := I TO N DO
BEGIN
K1 := K1 + Nj
HOLD := AMK,13;
JI := K1 - K + j;
AK,13 := -AK,13;
AK,13 := HOLD;
END;
GOTO 100;
END;

BEGIN
{ form the matrix (see cubic spline calculations) }
for count1 := 1 to npivots-2 do
BEGIN
for count2 := 1 to npivots-2 do
BEGIN
if count1 = count2 then
matrix[count1,count2] := 4
else if (count1 = count2 - 1) or (count1 = count2 + 1) then
matrix[count1,count2] := 1
else matrix[count1,count2] := 0
END;

{ clear all unneeded locations }
for count1 := npivots-1 to maxpivots do
matrix[count1,count2] := 0
END;

for count1 := npivots - 1 to maxpivots do
for count2 := 1 to maxpivots do
matrix[count1, count2] := 0;
iwmatrix(matrix) { take inverse of matrix }
END;

procedure form_vector;
var
count : integer;
d2pntr : pointertype;
BEGIN
{ start with first track }
d2pntr := dpntr;
vector[i] := 3*(d2pntr^.right^.right^.value - d2pntr^.value);
for count := 2 to npivots-1 do
BEGIN
d2pntr := d2pntr^.right;
vector[count] := 3*(d2pntr^.right^.right^.value - d2pntr^.value);
END;
END;
procedure calc_all_data;
var
    bees: vec_type;
    tanpntr: integer;
    datapntr: pointer_type;
    count, file_num, i: integer;
    val: real;

procedure store_val;
begin
    if val_number > 3 then val := (val*pi)/180; (* first three values are center of interest not angles *)
    assign(out, filesout[count]);
    append(out); (* open file and save data to add more *)
    (* write only 18 values on a line *)
    if trunc(val_number/18)*18 = val_number then
        writeln(out, val:12);
    else write(out, val:12);
    close(out)
(* will store values in files *)
end;

procedure calc_steps(b: vec_type; steps: integer; pivots: integer);
var
    number: real;
    stepper: integer;
begin
    number := 0;
    stepper := 0;
    (* calculates nsteps values between each two values of data *)
    while (stepper <> nsteps) do
        stepper := stepper + 1;
        store_val (* store value in appropriate file *)
        count := count + 1;
        (* increment for next step *)
        number := number + 1/(nsteps)
    end;

begin
    for count := 1 to max_pivots do (* initially clear vec4 *)
        vec4[count] := 0;
    tanpntr := 1;
    datapntr := dpntr;
    count := 1;
    repeat (* calc a cubic function for each two points in a row in one track *)
        vec4[1] := datapntr^..value;
        vec4[2] := datapntr^..right^..value;
        datapntr := datapntr^..right;
        vec4[3] := tangent[tanpntr];
        vec4[4] := tangent[tanpntr+1];
    end;
tonptr := tonptr + 1;
multiply(4, fourx4, vec4, becs); /* multiply vec4 by fourx4 to get constants for cubic function */
calc_steps(becs, nsteps, npivots);
until (dataptrA.right = nil) or (tonptr = npivots);
val := dataptrA.value;
store_val /* store last value in file */
end;

procedure process;
var
  loop : integer;
  trck : integer;
begin
  val_number := 11 /* to make sure only 18 values per line */
  trck := 1; /* tells which track of data points being calculated */
  form_4x4; /* form matrix only once */
  for_matrix; /* will form and take inverse */
  writeln writeln('calculating track number '); /* loop to calculate all data once it has been entered or read in */
  for loop := 1 to (ntracks) do
    begin
      writeln(trck:4);
      form_vector; /* calculates data from present track */
      multiply(npivots-2, matrix, vector, tangent); /* calc tangent vector */
      calc_all_data;
      dptr := dptrA.down; /* next track data set */
      trck := trck + 1;
      val_number := val_number + 1
    end
  end;
begin
  clrscr;
  get_info;
  writeln writeln(' creating files *');
  create_files;
  writeln writeln(' processing data, please wait *');
  process;
  writeln writeln(' done *')
end.
A.4 Help Documents for Using the Graphics Package

Several help documents for using the graphics package are listed in Tables 7.1 to 7.3. Tables 7.1 and 7.2 show a listing of the program and the interactive dialogue for plotting a sphere and a cylinder.

Table 7.3 illustrates the use of the animation routines to generate a set of intermediate postures of the human-like model.
Table 7.1. Help Document for Plotting a Sphere Using the Graphics Package.

```
VAR
radius:REAL; ndihor:real; ndivertdiv21:integer;
inputfile:CHAR; extension:CHAR;
BEGIN
(main program body)
  Iniferspective:

  radius:=200;
  outfile:=‘sph’;
  writeln;
  writeln;
  Subroutine [eep, sph ‘‘]
  writeln
  drive_width:=cl ‘drive pt’:=‘d’;
  outfile:=‘sph’; extension:=‘.wld’;
  ‘generate a world file which contains a sphere in polygon ‘
  ‘list format ‘
  ‘gmesh(p_xhor) ndivertdiv21, radius, drive_width, outfile’
  Extension.out)
  ‘open input channel for further processing of the world file of the ‘
  ‘sphere ‘
  Inputfile:=concat (drive_width, outfile, extension)
  assignment:=inputfile[‘resetline’];
  ‘open an output file which contains the final display code ‘
  ‘for the sphere with name ‘
  ‘Gmeshfile3(l)
  OpenSegment3;
  InitGraphical ‘initialize the graphics output device ‘
  CloseSegment;
  CloseInput;
  end;

DEND_SPH

Subroutine Iniferspective

----------
viewing angle(deg.) = ° 30
eye position 1/ radius = ° 500
input of (theta, phi) or
(top, side, front view) [T,U] = T
front, top, or side view [F,T,S] = T
init. of other persp. parameter [Y/M] = U
aspect ratio [111.0] I FC1(2.4) = Y
1.24
center of interest 1 [X, Y] = 0
center of interest 1 [X, Y] = 0
hither size =
you-size =
1000
put on Gt. or f.c. or write to file [G/f/F] = F
front-backface if [b/p/ (M=sph)] = F
----------
```

207
Table 7.2. Help Document for Plotting a Cylinder Using the Graphics Package.

<table>
<thead>
<tr>
<th>Programmer</th>
<th>Dieter Longer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>20-Jan-86</td>
</tr>
<tr>
<td>Function</td>
<td>To plot a general cylinder which has an ellipse as a ground plane</td>
</tr>
</tbody>
</table>

( the following compiler directive links the program package )
( GRAPHMOD.FAS to the demonstration program )

(*I GRAPHMOD.FAS*)

```pascal
var atop:bottom,btop:bottom,heigh:real;
ndivhorizontal,ndivvertical,i,j:integer;
outfile,extension,str14:
begin  ( main program body )
  initPerspective:
  atop:=100;bottom:=400;
  btop:=50;bottom:=150;heigh:=500;
  drive_width:=c;drive_plt:=d;
  extension:=i;wld'outfile:=cyl';
  writeln;
  writeln(' Subroutine Demo_cyl ');writeln;
  writeln( ' -------------------------- ' );
  write(' ndivhorizontal = ? ');readln(ndivhorizontal);
  write(' ndivvertical = ? ');readln(ndivvertical);

  ( generate a world file which contains a sphere in polygon )
  ( list format )
gencylinder(atop, bottom, btop, bottom, height, ndivhorizontal, ndivvertical, drive_wld, outfile, extension);
  inputfile:=concat(drive_wld, outfile, extension);
  assign(inputfile);reset(inputfile);

  ( open an output file, which contains the final display )
  outfile:=concat(drive_wld, outfile, extension);
  assign(outfile);reset(outfile);

  ( for a cylinder, with name : SEGMI.FLT )
  OpenSegment[1];

  initgraphics; ( initialize the graphics output device )
  ClipPolygon3D;
  CloseSegment;
  CloseInput; ( close input channel )
end.

"Subroutine initPerspective"

viewing angle(deg.) = ? 40
eye position : radius = ? 500
input of (theta, phi) or
   (top,side,front view) [T,V] = ? V
   Front, Top or Side view [F, T, S] = ? F
init. of other persp. paramet. [Y/N] ? Y
aspect ratio [GT:1.0 ; FC:1.24] = ? 1.24
center of interest : x-comp.=? 0
center of interest : y-comp.=? 0
center of interest : z-comp.=? 0
hither plane = 1
van-viewer ?
plot on Gl, or Fc, or write to File [G/F] ? F
front or backface [F/P/A; Eff] = ? F

Subroutine Demo_cyl"

ndivhorizontal = ? ;
ndivvertical = ? ;
```

208
Table 7.3. Help Document for the Use of the Animation Routines.

```
Input file:
# data for jumping motion
# first three values are center of interest
#
# col 1: input 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
# col 2: input 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
# col 3: input 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
# col 4: input 61 62 63 64 65 66 67 68 69 70 71 72
#
N 72 1 0
```

Interactive dialogue:

```
Input device: keyboard of file [n/f] : f
Drive [A/B/C/D/G] : D
Name of input file [infil.dat]: inlin = MAN.FOS
Name of output file [outfil.dat]: outfil = LONO,JUMP
reading in file number of tracks = 72 number of steps = 1 number of pivots = 0

# creating file #

# processing data, please wait ##
```

```
calculating track number
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72

### done ###
```
Appendix B

THE SYMBOLIC GENERATION OF THE ROTATIONAL DYNAMICS OF THREE
RIGID LINKS IN THREE-DIMENSIONAL SPACE AND THE
COMPUTATION OF THEIR FIRST INTEGRALS

In Appendix B the two methods of deriving the rotational dynamics
of an open kinematic chain of rigid bodies, see section 4.2, are
illustrated for a three-link system in three-dimensional space. It is
shown how the first integrals of the above mentioned systems are derived
and computed. All the analytical expressions for the rotational
dynamics and the first integrals, with the exception of Appendix B3 and
B4 are programmed in the symbolic manipulation language MACSYMA and are
evaluated for the case of three (N=3) rigid links. Therefore, the
individual sections of this appendix contain the corresponding (MACSYMA)
program listing as well the results which are obtained by executing
these programs. The organization of this appendix is as follows.
In Appendix B1 the equations of motion of three rigid links are
projected onto a subspace, in which rotational and translational
dynamics are decoupled, see section 4.2.2. In Appendix B2 the
rotational dynamics of the same system are derived from an
interconnection diagram, see section 4.2.3. In Appendix B3 a general
expression for the rotational dynamics in Hamiltonian form is derived.
In Appendix B4 a general expression is derived from the angular momentum
of N arbitrary, rigid bodies, which is the first integral of the above
mentioned mechanical systems. In Appendix B5 a general expression is
derived from an open chain composed of N rigid, holonomically
constrained bodies.

B.1. The Symbolic Projection of the Newton-Euler Equations of Motion
Using MACSYMA

This appendix refers to the main section 4.2.2. Consider Table
8.1, which shows the elements of Eq. (4.1) for a system of N, holonomic,
rigid links, which are interconnected in an open chain. Table 8.2
depicts the elements of the decoupling transformation, which are
specified by Eq. (4.2). It is pointed out that the matrices F, G of
Table 8.3 are a function of N, the number of links of the composite
system.
Equations (4.1) and (4.2) are programmed for the case of three (\( N = 3 \)) links using MACSYMA. The corresponding program listing is shown in Table B.3. This program was executed on a DEC20 computer at The Ohio State University. The results, which consist of a listing of the unconstrained and projected equations of motion for three links in three-dimensional space are shown in Tables 8.4 and 8.5. The program listing can easily be modified to mechanical systems with more than three links in three-dimensional space.

Table 8.1. Elements of the Coupled Newton Euler Equations of Motion (Eq. 4.1) for \( N \) Rigid, Holonomic Bodies

\[
U_1 = \text{block diag}(m_1 E, m_2 E, \ldots, m_n E, J_1, J_2, \ldots, J_n): \\
\begin{bmatrix}
E & 0 & 0 & \cdots & 0 \\
-E & E & 0 & \cdots & 0 \\
0 & -E & E & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \cdots & \cdots & 0 & -E
\end{bmatrix}
\]

\[
U_2 = \begin{bmatrix}
-L L_1 A_1^T & 0 & 0 \\
K K_2 A_2^T & -L L_2 A_2^T & \cdots \\
0 & K K_3 A_3^T & \cdots \\
\vdots & \vdots & \ddots & \vdots \\
0 & \cdots & \cdots & 0 \\
0 & 0 & \cdots & -L L_n A_n^T & -L L_n A_n^T & \cdots \\
0 & 0 & \cdots & 0 & K K_n A_n^T
\end{bmatrix}
\]

\[
U_3 = \begin{bmatrix}
-m_1 g R_3 \\
\vdots \\
-m_n g R_3 \\
f_1(w_1) \\
\vdots \\
f_n(w_n)
\end{bmatrix}
\]

with \( R_3 = (0, 0, 1)^T \); \( f_i(w_i) = -W W_i J_i W_i \)
Table 8.1. (cont'd)

\[
U_4 = \begin{bmatrix}
0 & 0 & \cdots & 0 \\
0 & \ddots & & \\
0 & & 0 & 0 \\
-A_1^T & 0 & 0 & 0 \\
A_2 & -A_2^T & \ddots & \ddots \\
0 & A_3 & \ddots & \ddots \\
& \ddots & 0^T & \ddots \\
0 & \ddots & -A_{n-1} & A_n \\
0 & & & \\
\end{bmatrix} : [3n \times 3(n-1)]
\]
Table 8.2. Elements of the Decoupling Transformation, Eq. (4.2).

\[
\begin{bmatrix}
X_1 \\
\vdots \\
X_n \\
\end{bmatrix}
= 
\begin{bmatrix}
E & F_{11} & F_{12} & \cdots & F_{n1} \\
E & F_{21} & \cdots & \cdots & \cdots \\
E & \cdots & \cdots & \cdots & \cdots \\
\end{bmatrix}
\begin{bmatrix}
X \\
w_1 \\
\vdots \\
w_n \\
\end{bmatrix}
\]

with \( E: [3 \times 3] \) Identity matrix

\[
F_{I,I} = F_{I,I}(N) + F_{DIAG_I,N}: [3 \times 3] \text{ matrix}
\]

\[
F_{I,J} = F_{I,J}(N) + F_{LODIAG_{I,J},N}: [3 \times 3] \text{ matrix}
\]

\[
G_I = G_I(N) + G_I,N \quad I = 1, \ldots, N: [3 \times 1] \text{ matrix}
\]
Table 8.2. (cont'd)

\[
\begin{array}{c}
\text{FDIAG} = \begin{cases}
1, N & \text{if } \text{FDIAG} = 1, N \\
 & \text{otherwise}
\end{cases}
\end{array}
\]

\[
\begin{array}{c}
\text{FUPDIAG} = \begin{cases}
1, J, N & \text{if } \text{FUPDIAG} = 1, J, N \\
 & \text{otherwise}
\end{cases}
\end{array}
\]

\[
\begin{array}{c}
\text{FLDUDIAG} = \begin{cases}
1, J, N & \text{if } \text{FLDUDIAG} = 1, J, N \\
 & \text{otherwise}
\end{cases}
\end{array}
\]

\[
\begin{array}{c}
\text{G} = \begin{cases}
1, N & \text{if } \text{G} = 1, N \\
 & \text{otherwise}
\end{cases}
\end{array}
\]

\[
\begin{array}{c}
\text{I} = \begin{cases}
1, N & \text{if } \text{I} = 1, N \\
 & \text{otherwise}
\end{cases}
\end{array}
\]

\[
\begin{array}{c}
\text{J} = \begin{cases}
1, N & \text{if } \text{J} = 1, N \\
 & \text{otherwise}
\end{cases}
\end{array}
\]

\[
\begin{array}{c}
\text{K} = \begin{cases}
1, N & \text{if } \text{K} = 1, N \\
 & \text{otherwise}
\end{cases}
\end{array}
\]

214
Table 8.3.  MACSYMA Program Listing for the Symbolic Projection of a
Three-Link System in Three-Dimensional Space

/* symbolic projection of equations of motion */
/* for a 3-link system */
/* orthonormal ann matrices */
block(
   gll:1111, 12:1122, 13:1133,
   11:1111, 12:1122, 13:1133,
   1111:1111, 1212:1122, 1313:1133,
   k:k111, k2:k22, k3:k33,
   expr1: x111, expr2: x122, expr3: x133,
   /* summation results */
   declare(t1,t2,t3), ncont: 11, expr1: x111,
   declare(t1,t2,t3), ncont: 11, expr2: x122,
   declare(t1,t2,t3), ncont: 11, expr3: x133,
   t1:11, t2:12, t3:13,
   /* define sin,litification rules */
   block(
      defrule(rule1, transpose(11), -111),
      defrule(rule2, transpose(12), -122),
      defrule(rule3, transpose(13), -133),
      defrule(rule4, transpose(a1), a1**(-1)),
      defrule(rule5, transpose(a2), a2**(-1)),
      /* end of block */
   )

/* elements of projection matrices U and U^t */
l1,n:=(1/sum(ratsun1(a1),kk11)*sum(n(kj,k,j,1,j-1)*
   m(kk11,j11j)*sum(n(kk1,k,j+1,j))),
   n:=(1/sum(ratsun1(a1),kk11)*sum(n(kj,k,j,1,j-1)*
   m(kk11,j11j)*sum(n(kk1,k,j+1,j)));

/* invoke rules for scalar products */
doctrues: true; notswing: true; destrue: true;
block(
   ul:rats1(11), u2:rats1(12), u3:rats1(13),
   v1:rats1(11), v2:rats1(12), v3:rats1(13),
   w1:rats1(11), w2:rats1(12), w3:rats1(13),
   /* declare transposes(a1), a2, a3 */
   ul, v2, v3, w1, w2, w3),
   215
Table 8.3. (cont.)

```cpp
129x690
145x612
169x428
216`
Table 8.3. (cont'd)

```c
/* vs := ubt . u4
   vs1:=[ratsimp(sum(u5tlj,k),u4lj,k)]x1,0:0,n),
   vs1:=[ratsimp(sum(u5lj,ne_n,ins_n,1,1)); /* end of block */
/* matrices of original system */
display(u1, u2, u3, u4, v6); 
/* matrices of transformed (coupled) system */
display(v1, v2); misu(v3, v4, v5); 
closefile("neweul.txt");
```
Table 8.4. The Original Equations of Motion and the Decoupling Transform for 3 links in 3-D.

```plaintext
(1) batchload("new_eul3.sym");
Batching the ill file "new_eul3.sym"
Watching core...
Time= 135.970 sec.
(1) "new_eul3.sym"
(2) display(u1,u2,u3,u4,u5,w0);

| 0 0 0 0 0 0 |
| 0 0 0 0 0 |
| 1 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| -1 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |

u2 = [-LL * TRANSF(A) 0]
| 1 0 0 0 0 |
| 1 0 0 0 0 |
| -1 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |

u3 = [-LL * TRANSF(A) - LL * TRANSF(A) 0]
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |

u4 = [0]
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |

u5 = [0]
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |

w0 = [0]
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
| 0 0 0 0 0 |
```

218
Table 8.4. (cont'd)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U</td>
</tr>
</tbody>
</table>

\[ u_4 = \{
\begin{align*}
&= \text{TRANSP}(A) \quad u \\
&1 \\
&\text{TRANSP}(A) = \text{TRANSP}(A) \\
&2 \\
&\text{TRANSP}(A) \\
&3 \\
\end{align*}
\]
Table 8.4. (con'd)

\[
X_{ij} = \text{main effect of } (A, \text{ wa}, \text{ wa} - 1)
\]

\[
\begin{array}{cccccc}
3 & 3 & 3 & 3 & 2 & 2 \\
2 & 2 & 2 & 2 & 1 & 1 \\
1 & 1 & 1 & 1 & 3 & 3 \\
2 & 2 & 2 & 2 & 1 & 1 \\
1 & 1 & 1 & 1 & 3 & 3 \\
2 & 2 & 2 & 2 & 1 & 1
\end{array}
\]
Table 8.5. The Projected Equations of Motion for 3 Links in 3D.
Table 8.5. (cont'd)

\[
\begin{align*}
V_3 &= \begin{bmatrix}
0 & 0 \\
0 & 0 \\
\end{bmatrix} \\
\end{align*}
\]

\[
\begin{align*}
\begin{bmatrix}
-1 & 0 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
V_4 &= \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
V_5 &= \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
-\text{TRANSPOSE}(A) & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
V_6 &= \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\text{TRANSPOSE}(A) & -\text{TRANSPOSE}(A) & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
V_7 &= \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
2 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
V_8 &= \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\end{bmatrix}
\end{align*}
\]
Table 8.5. (cont'd)

\[
\begin{array}{cccc}
\alpha & \beta & \gamma & \delta \\
\epsilon & \zeta & \eta & \theta \\
\iota & \kappa & \lambda & \mu \\
\nu & \xi & \omicron & \pi \\
\rho & \sigma & \tau & \upsilon \\
\phi & \chi & \psi & \omega \\
\end{array}
\]
Appendix B.2. A General Expression for the Rotational Equations of Motion and the Symbolic Evaluation for the Case of Three Links (N = 3)

This appendix refers to the main sections 4.2.2 and 4.2.3. Consider Table 8.6, which specifies the elements of Eq. (4.3) for a system of N, holonomic, rigid bodies, which are interconnected in an open chain. Eq. (4.3) is programmed in MACSYMA for the case of three (N = 3) links. A program listing is shown in Table 8.7. The program was executed on a DEC-20 system at The Ohio State University. The corresponding results are shown in Table 8.8. Note that Table 8.5 describes both the translational and the rotational dynamics of three links in three-dimensional space. However, Table 8.5 describes both the translational and the rotational dynamics of three links in three-dimensional space. However, Table 8.8 describes only the rotational dynamics of the same system. A comparison of the rotational dynamics of the system using the two different approaches shows that the corresponding entries of the same matrices are equal, when it is assumed that the mechanical parameters of the three links, which are shown in Table 8.5, are specified in the inertial coordinate system, i.e., the $A_i$ ($i = 1, 2, 3$) in the unity matrices.
Table 8.6. A General Expression for the Decoupled Equations of Motion for N Rigid, Interconnected Bodies, Eq. (4.3)

\[
V_1 = \begin{bmatrix}
M & 0 & 0 & \cdots & 0 \\
0 & I_{11} & \text{mDD}_{12}A_1^TA_2^T & \text{DD}_{21} & \cdots & \text{mDD}_{1n}A_1^TA_n^T & \text{DD}_{n1} \\
0 & \text{mDD}_{21}A_2^T & I_{22} & \cdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \text{mDD}_{n1}A_n^T & \text{DD}_{1n} & \cdots & \text{I}_{nn}
\end{bmatrix}
\]

\[I_{ii} = J_{ii} - \text{mDD}_i^2 - \sum_{j \neq 1} \text{m}_j \text{DD}_{ij}^2\]

\[
V_2 = \begin{bmatrix}
0 \\
\text{m} \sum_{j \neq 1} \text{DD}_{ij}A_1^TA_j^T & \text{WW}_1^2 D_{j1} \\
\vdots \\
\text{m} \sum_{j \neq 1} \text{DD}_{nj}A_n^TA_j^T & \text{WW}_j^2 D_{jn}
\end{bmatrix}
\]

\[
V_4 = \begin{bmatrix}
\text{-Mg} \\
\text{-WW}_1 I_{11} \text{W}_1 \\
\vdots \\
\text{-WW}_n I_{nn} \text{W}_n
\end{bmatrix}
\]

\[
V_5 = \begin{bmatrix}
0 & \cdots & 0 \\
-A_1^T & 0 \\
A_2^T & -A_2^T & \cdots & 0 \\
\vdots & A_3^T \\
0 & \vdots & \vdots & -A_{n-1}^T \\
0 & \vdots & A_n^T
\end{bmatrix}
\]

225
Table 8.7. MACSYMA Program Listing for the Derivation of the Rotational Dynamics for Three Links in Three-Dimensional Space from the Interconnection Diagram.

```macsyma
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate(true);
allocate true;
Table 8.7. (cont'd)

/* evaluation of local vectors */

n_link: l; maxsum(r[k], k, 1, n_link);

ratsinp(uo[i]); ratsump(uo[i]); ratsinp(uo[i]);

ratsinp(dci[l, 1]); ratsinp(dci[l, 2]); ratsinp(dci[l, 3]);

ratsinp(oou[l, 1]); ratsinp(oou[l, 2]); ratsinp(oou[l, 3]);

ratsinp(gou[l, 1]); ratsinp(gou[l, 2]); ratsinp(gou[l, 3]);

ratsinp(fou[l, 1]); ratsinp(fou[l, 2]); ratsinp(fou[l, 3]);

ratsinp(dou[l, 1]); ratsinp(dou[l, 2]); ratsinp(dou[l, 3]);

ratsinp(too[l, 1]); ratsinp(too[l, 2]); ratsinp(too[l, 3]);

ratsinp(poo[l, 1]); ratsinp(poo[l, 2]); ratsinp(poo[l, 3]);

ratsinp(loo[l, 1]); ratsinp(loo[l, 2]); ratsinp(loo[l, 3]);

/* computation of system matrices */

/* matrix vi */

v1: gen = matrix(v1, n_link, n_link, 1, 1);

for l = 1 thru n_link do
    for j = 1 thru n_link do
        if l = j then w[i, l, j, m, l, m]
            else w[i, l, j, m, l, m] = 0;

v1: full ratsinp(v1);

/* matrix v2 */

v2: gen = matrix(v2, n_link, n_link, 1, 1);

v2: full ratsinp(v2);

/* matrix v4 */

v4: gen = matrix(v4, n_link, n_link, 1, 1);

v4: full ratsinp(v4);

/* display of functions and definitions */

gdisplay(v1, v2, v4);

closefile("verycent.txt");
Table 8.8. The Projected Equations of Motion for 3 Links in 3 D Derived from the Interconnected Diagram.

```c
/* display of functions and definitions */
display(v1, v2, v3);

LL M * LL M
1 3 1 <2>
V1 = H11 H12 (------------) + (------------) + J,
1 3 2 1 1

(VL = A1 A2 A3 A4)
2 2 1 1

LL M , ----------------, LL P , ----------------,
1 1 1 1

3 2 1 3 2 1

LL P
1 1

((LL = LL) P + PH M ), ---------------------
2 2 2 2 2 2

LL = LL PH
1 1 1 1

((LL = LL) P + PH M ) , ---------------------
2 2 2 2 2 2

(LL = PH) P - PH M 
2 2 1 2 1 2

M (--------------) - M (--------------)
1 1 1 1

3 2 1 3 2 1

LL P + LL M
1 1 1 1

((PH = PH) PH M ), ---------------------
3 3 3 3

((PH = PH) PH M ) M
3 3 3 3

(M - M) A2
1 3 1 1

(M - M) A2
1 3 1 1
```

228
Table 8.8. (cont'd)

<table>
<thead>
<tr>
<th>LL</th>
<th>m</th>
<th>&lt;z&gt;</th>
<th>m</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&lt;z&gt;</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(L = h) m = h m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(L = h) p = h p</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(L = h) q = h q</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>(L = h) r = h r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ V = P A(i) A((s + m + m) \text{ (---)} + (m + p + p) \text{ (---)}) + (s - m - m) \text{ (---)} + (m + h + h) \text{ (---)}) \]

<table>
<thead>
<tr>
<th>LL</th>
<th>m</th>
<th>&lt;z&gt;</th>
<th>m</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&lt;z&gt;</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ (L - h) m = h m \]

<table>
<thead>
<tr>
<th>LL</th>
<th>m</th>
<th>&lt;z&gt;</th>
<th>m</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&lt;z&gt;</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ (L - h) p = h p \]

<table>
<thead>
<tr>
<th>LL</th>
<th>m</th>
<th>&lt;z&gt;</th>
<th>m</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&lt;z&gt;</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ (L - h) q = h q \]

<table>
<thead>
<tr>
<th>LL</th>
<th>m</th>
<th>&lt;z&gt;</th>
<th>m</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&lt;z&gt;</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ (L - h) r = h r \]

<table>
<thead>
<tr>
<th>LL</th>
<th>m</th>
<th>&lt;z&gt;</th>
<th>m</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>&lt;z&gt;</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ (L - h) s = h s \]
Appendix B.3. Derivation of the Hamiltonian Form of the Rotational Equations of Motion.

In this appendix the Hamiltonian form of the rotational equations of motion is derived, which is useful in the controllability analysis of the system, Eq. (4.3), or Eq. (4.6). The rotational equations of motion, Eq. (4.6), Table 8.7 are given below in a common reference frame, the ICS system.

\[ v_1 \dot{W} = v_2 + v_4 + v_5 M \quad \text{(R.1)} \]

with

\[
v_1 = \begin{bmatrix}
A_1 I_{11} & \vdots & m A_1 DD_{12} A_1^T A_2 DD_{21} \\
m A_2 DD_{21} A_2^T A_1 DD_{12} & \vdots & \vdots \\
m A_n DD_{n1} A_n^T A_1 DD_{n1} & \vdots & A_n I_{nn}
\end{bmatrix}
\]

\[
v_2 = \begin{bmatrix}
A_1 DD_{12} A_1^T A_j WW_j^2 D_{j1} \\
\vdots \\
m \sum_{j \neq 1} A_n DD_{nj} A_n^T A_j WW_j^2 D_{jn}
\end{bmatrix}; \quad v_4 = \begin{bmatrix}
-A_1 WW_1 I_{11} W_1 \\
\vdots \\
-A_n WW_n I_{nn} W_n
\end{bmatrix}
\]

\[
v_5 = \begin{bmatrix}
-E & 0 & \ldots & 0 \\
E & -E & \vdots \\
0 & E & \vdots \\
\vdots & 0 & -E \\
0 & 0 & E
\end{bmatrix}
\]

The angular momentum vector \( P = (P_1 \ldots P_n)^T \) is, in the ICS: \( P = v, W \)
The time derivative of \( P \) is

\[
\dot{P} = v_1 \dot{W} + v_6 + v_4 + v_8 + v_9 \quad \text{(B.2)}
\]
\[ v_6 = M \begin{bmatrix} \sum_{j \neq 1} A_j DD_{1j} A_1 T A_j DD_{1j} W_j \\ \vdots \\ \sum_{j \neq n} A_n DD_{nj} A_n T A_j DD_{nj} W_j \end{bmatrix} = M \begin{bmatrix} \sum_{j \neq 1} A_j WW_{1j} DD_{1j} A_1 T A_j DD_{1j} W_j \\ \vdots \\ \sum_{j \neq n} A_n WW_{nj} DD_{nj} A_n T A_j DD_{nj} W_j \end{bmatrix} \]

\[ v_7 = M \begin{bmatrix} \sum_{j \neq 1} A_j DD_{1j} A_1 T A_j DD_{1j} W_j \\ \vdots \\ \sum_{j \neq n} A_n DD_{nj} A_n T A_j DD_{nj} W_j \end{bmatrix} = M \begin{bmatrix} \sum_{j \neq 1} A_j WW_{1j} DD_{1j} A_1 T A_j DD_{1j} W_j \\ \vdots \\ \sum_{j \neq n} A_n WW_{nj} DD_{nj} A_n T A_j DD_{nj} W_j \end{bmatrix} \]

\[ v_8 = M \begin{bmatrix} \sum_{j \neq 1} A_j DD_{1j} A_1 T A_j DD_{1j} W_j \\ \vdots \\ \sum_{j \neq n} A_n DD_{nj} A_n T A_j DD_{nj} W_j \end{bmatrix} = M \begin{bmatrix} \sum_{j \neq 1} A_1 WW_{1j} I_n W_1 \\ \vdots \\ A_n WW_{nj} I_{nn} W_n \end{bmatrix} \]

Substituting Eq. (B.1) into Eq. (B.2) and realizing that \( v_2 = -v_8; \)
\( v_4 = -v_9 \) gives:

\[ \dot{p} = v_{10} + v_5 M \]  

(B.3)

with
\[ v_{10} = v_6 + v_9 = M \begin{bmatrix} \sum_{j \neq 1} A_1 \left( WW_1 DD_1 j - DD_1 j WW_1 \right) A_1^T A_j DD_j W_j \\ \vdots \\ \sum_{j \neq 1} A_n \left( WW_n DD_n j - DD_n j WW_n \right) A_n^T A_j DD_j W_j \end{bmatrix} \]

Also: \[ \dot{\theta} = BW = B(v^{-1} P) \] with \( B = \text{block diag}(B_1, \ldots, B_n) \) (B.4)

Equations (B.3) and (B.4) are the Hamiltonian state equations for the rotational equations of motion.
Appendix B4. Derivation of the Angular Momentum Expression for n Arbitrary Rigid Bodies.

In the following section the expression for the angular momentum of n arbitrary rigid bodies is derived, which is shown in [34] for the case of a single rigid body. For the sake of simplicity it is first assumed that each rigid body has a discrete mass distribution. It is shown that the results, obtained under these restrictions, apply to rigid bodies with continuous mass distributions as well. Consider Figure 8.1, which shows n rigid bodies with their respective centers of mass $C_i$ ($i = 1, ..., n$) in an inertial frame RCS with basis \{R_x, R_y, R_z\}. The origin of frame R coincides with the center of mass C of the compound system. Each rigid body has a total mass $m_i$ ($i = 1, ..., n$) and consists of p particles with mass $m_{ij}$ ($i = 1, ..., n; j = 1, ..., p$) s.t.:

$$m_i = \sum_{j=1}^{p} m_{ij} ; i = 1, ..., n$$

It has associated with it a local vector $p_i$ to its own center of mass $c_i$ relative to C. A vector $r_{ij}$ points from $c_i$ to the particle $m_{ij}$. From Fig. B.1 it follows that

$$z_{ij} = p_i + r_{ij} \quad \text{(B.5)}$$

Since $C_i$ is the center of mass of body i the following relationship holds:

$$\sum_{j=1}^{p} m_{ij} r_{ij} = 0 \quad \text{(B.6)}$$

$$\frac{d}{dt} \left( \sum_{j=1}^{p} m_{ij} r_{ij} \right) = \sum_{j=1}^{p} m_{ij} \dot{r}_{ij} = 0$$

Similarly, since C is the system's center of mass:

$$\sum_{i=1}^{n} m_i p_i = 0 \quad \text{(B.7)}$$

$$\frac{d}{dt} \left( \sum_{i=1}^{n} m_i p_i \right) = \sum_{i=1}^{n} m_i \dot{p}_i = 0$$

The angular momentum of a single body is relative to C is [26]:

$$H_i = \sum_{j=1}^{p} m_{ij} z_{ij} \dot{z}_{ij}$$

233
Figure 8.1. System of n Rigid Bodies with Discrete Mass Distribution in an Inertial Coordinate System which Originates in the Center of Mass C.
where \( \mathbf{zz}_ij \) is the crossproduct operator. The angular momentum of the compound system relative to \( C \) is:

\[
H = \sum_{i=1}^{n} H_i = \sum_{i=1}^{n} \sum_{j=1}^{p} m_{ij} \mathbf{zz}_ij \mathbf{z}_ij
\]  

(B.8)

The time derivative of Eq. (B.5) is:

\[
\dot{\mathbf{z}}_{ij} = \dot{\mathbf{p}}_i + \mathbf{r}_{ij}
\]  

(B.9)

Inserting equations (B.5), (B.6), (B.7) and (B.9) into (B.8) gives:

\[
H = \sum_{i=1}^{n} \sum_{j=1}^{p} m_{ij} \mathbf{p}_i \dot{\mathbf{p}}_i + \sum_{i=1}^{n} \sum_{j=1}^{p} m_{ij} \mathbf{r}_{ij} \mathbf{r}_{ij}
\]  

(B.10)

The expression for the angular momentum \( H \) consists of two terms. The first term on the right hand side of Eq. (B.10) represents the angular momentum of the points \( C_i \) \( (i = 1, \ldots, n) \) relative to \( C \) assuming that the total mass of body is is concentrated in \( C_i \) [26]. The second term in Eq. (B.10) represents the angular momenta of all \( i \) bodies relative to their center of mass \( C_i \) \( (i = 1, \ldots, n) \). The latter term can be further simplified. Consider, therefore, Fig. 8.2 which shows the \( j \)th body \( (i = 1, \ldots, n) \) of Fig. B.1 with an associated body coordinate system \( B_j \) with axes \( (X_j, Y_j, Z_j) \) whose origin coincides with \( C_j \). Body \( i \) is assumed to rotate with angular velocity \( \omega_i \) in frame \( R \). The term \( \mathbf{r}_{ij} \) in Eq. (B.10) which is the time derivative relative to frame \( R \), can be expressed as:

\[
\mathbf{r}_{ij} \bigg|_R = (\mathbf{z}_{ij} - \mathbf{p}_i) \bigg|_R = \mathbf{r}_{ij} \bigg|_{B_{iCS}} + \omega_i \mathbf{r}_{ij}
\]

where \( \mathbf{r}_{ij} \bigg|_{B_{iCS}} \) is the derivative with respect to the frame \( B_{iCS} \).

However, since body \( i \) is assumed to be a rigid body, the location of \( m_{ij} \) \( (i = 1, \ldots, n; j = 1, \ldots, p) \) is fixed in \( B_{iCS} \). Hence, the term \( \mathbf{r}_{ij} \bigg|_{B_{iCS}} \) vanishes:

\[
\sum_{i=1}^{m} \sum_{j=1}^{p} m_{ij} \mathbf{r}_{ij} \mathbf{r}_{ij} = \sum_{i=1}^{m} \sum_{j=1}^{p} m_{ij} \mathbf{r}_{ij} (\omega_i \mathbf{r}_{ij})
\]

\[
= \sum_{i=1}^{n} \sum_{j=1}^{p} (m_{ij} \mathbf{r}_{ij} \mathbf{r}_{ij} \omega_i - \mathbf{r}_{ij} \omega_i \mathbf{r}_{ij})
\]

235
Figure 8.2. Body i With Its Own Body Coordinate System $R_iCS$ in an Inertial Frame.
where the sign (*) symbolizes the scalar product between vectors. Introducing a unit dyadic U, s.t.,
\[ U \cdot v = v \cdot U = v \]

where \( v \) is an arbitrary vector gives

\[
\sum_{i=1}^{m} \sum_{j=1}^{p} m_{ij} r_{ij} \dot{r}_{ij} = \sum_{i=1}^{m} \sum_{j=1}^{p} m_{ij} (r_{ij} \cdot r_{ij} \cdot U - r_{ij} \cdot r_{ij}) \dot{\omega}_i \\
= \sum_{i=1}^{n} I_i \omega_i \tag{B.11}
\]

The quantity \( I_i \) denotes the inertia dyadic of body \( i \) relative to its center of mass \( C_i \). Finally, the angular momentum formula, Eq. (B.10) for \( n \) rigid bodies with a discrete mass distribution becomes:

\[
H = \sum_{i=1}^{n} I_i \omega_i + m_i p_{pi} \dot{p}_i \tag{B.12}
\]

with

\[
\sum_{i=1}^{n} m_i p_{pi} p_i = \sum_{i=1}^{n} \sum_{j=1}^{p} m_{ij} p_{pi} \dot{p}_i
\]

Equation (B.12) applies to a system of rigid bodies with continuous mass distribution as well. To show this, take the limit of Eq. (B.12)

\[
H' = \lim_{r \to \infty} H = \sum_{i=1}^{n} \int_{m_i}^{p_{pi}} \dot{p}_i \, d_m + \sum_{i=1}^{n} \int_{m_i}^{p_{pi}} r_{rij} \dot{r}_{ij} \, d_m \\
= \sum_{i=1}^{n} I_i \omega_i + \sum_{i=1}^{n} \int_{m_i}^{p_{pi}} (r_{ij} \cdot r_{ij} \cdot U - r_{ij} \cdot r_{ij}) \, d_m \tag{B.13}
\]

This simplifies for rigid bodies to:

\[
H' = H = \sum_{i=1}^{n} m_i p_{pi} \dot{p}_i + I_i \omega_i \tag{B.14}
\]

where \( I_i = \int_{m_i}^{p_{pi}} (r_{ij} \cdot r_{ij} \cdot U - r_{ij} \cdot r_{ij}) \, d_m \), which is the inertia dyadic of body \( i \) relative to its center of mass \( C_i \).
Appendix B.5. Derivation of the Angular Momentum Expression for N Rigid Bodies Subject to Holonomic (Connection) Constraints.

In this section the formula for the angular momentum of a collection of N arbitrary rigid bodies, Eq. (B.14), is applied to N interconnected rigid bodies. The latter bodies and their associated body coordinate are shown in Fig. B.1. The terminology is the same as outlined in Appendix B4. A local vector $p_i$ shows the center of mass of body $i$ ($i = 1, ..., N$) relative to the center of mass of the compound system CM. Using the appropriate results from [32] the vector $p_i = p_i(N)$, where $N$ represents the number of links, can be written as:

$$p_i(N) = \frac{1}{M} \sum_{j=1}^{i-1} \{(-K_j + L_j) \sum_{k=1}^{j-1} m_k + L_j m_j\} - \frac{1}{M} \{ K_i \sum_{k=1}^{i-1} m_k + L_i \sum_{k=i+1}^{N} m_k \} - \frac{1}{M} \sum_{j=i+1}^{N} \{(-K_j m_j + (-K_j + L_j) \sum_{k=j+1}^{N} m_k) \}

$$

where

$K_i(L_i)$: local vector to the initial (terminal) hinge point of the i-th link relative to the center of mass of the i-th link
$m_i$: mass of the i-th link

It should be noted that the time derivative of vectors $K_i$ with respect to the RCS is:

$$\dot{K}_i \bigg|_{RCS} = \dot{K}_i \bigg|_{B_iCS} + W_i K_i$$

with $W_i$: angular velocity of body $i$ in $B_iCS$

$WW_i$: cross product operator of $W_i$

However, the quantity $K_i$ is a constant in the $B_iCS$ s.t. $\dot{K}_i \bigg|_{B_iCS}$ vanishes. The vector $K_i$ is conveniently [40] expressed in the $B_iCS$ and transforms from the $B_iCS$ to the RCS via the map $A_i$:

$$K_i \bigg|_{RCS} = A_i K_i \bigg|_{B_iCS}$$
Taking the time derivative on both sides and noting that $\dot{A}_i = A_i W W_i$ gives automatically the previously mentioned result:

$$\dot{k}_i \mid_{RCS} = A_i W W_i k_i \mid_{B_i CS}$$

The above also applies to the vector $\mathbf{L}_i$. With these definitions the expressions for the angular momentum $H_n$ of $n$ rigid, holonomic links can be evaluated, (see Table 8.8). In Table 8.8 the expression for the angular momentum $H_n$ and for vectors $p_{p_i(n)}$ and $p_{i(n)} = p_{dt_1(n)}$ is given. For reasons of convenience the maps

$$A_i: B_i CS \rightarrow RCS$$
$$A_i T: B_j CS \rightarrow B_i CS$$

were omitted from the expressions of $H_n$. A program listing for computing $H_n$, see Table 8.8, symbolically using MACSYMA [100] is given in Table 8.9. The angular momentum $H_n$ is evaluated for $H_3 (n = 3)$ and the results are shown in Table 8.10.

The program, see Table 8.9, can easily be modified to derive the angular momentum expression of an arbitrary number of links. The number $(N)$ of links is only limited by the implementation of MACSYMA on the computer system, i.e., the amount of available space for data storage.
Table 8.9. The General Expression for the Angular Momentum $H_N$ of $N$ Links Which are Subject to Holonomic Constraints.

$H_N = \sum_{J=1}^{J=N} (PP \cdot PDT) + \sum_{J=1}^{J=N} (1 \cdot J)$

For $N > M$ and $PP > (I \cdot w)$

$H_N \leq J_N \leq J_N'$

For $J = 1$ to $J = N$

$PP = \sum_{I=1}^{I=M} \sum_{J=1}^{J=M} (M - KK) \geq M - KK$

$I \geq M + LL$

$J = 1$ to $J = J$
Table 8.9. (cont'd)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 8.10. Listing of the Program which Symbolically Calculates the Angular Momentum of Three Links.

```plaintext
/* symbolic computation of the angular momentum of n links */
(30), which are subject to holonomic constraints.
notation: the dot '.' indicates a matrix product
definitions of scalars and matrices

l11(l1) 12:l1(l2) 13:l1(l3)
l12(l1) 12:l2(l2) 13:l2(l3)
l13(l1) 12:l3(l2) 13:l3(l3)
lk2(kk(2) 1kk(kk(3) 1111(l1) 1121(l2)
lk3(kk(2) 1kk(kk(3) 1111(l1) 1121(l2)

declare(l1,l2,l3, ml, a2,a3),nonscalar);
declare((k2,kk, 111,112),nonscalar);
declare(lk2,lk3),nonscalar);
declare((m,m2,m3), scalar);

/* declare rules for scalar products */
dot0simpl:true; dot0simp:true; dot0simp:true;

expr1(j):=(-kk[j]*l1[j]+l1[i]*sum(mik,k,1,i-1)*l1[j]*m[j]);
expr2(1,n):=1/m(kk[l]*sum(mik,k,1,i-1)*l1[j]*m[j]));
expr3(j,n):=kk[j]*l1[j]+(kk[j]*l1[j])*sum(a[k],k,J-1)*l1[j]*m[j]);
pp(l,n):=1/m*sum(expr1(j),j,1,n-1)+expr2(1,n)-1/m*sum(expr3(j,n),j,1,n);

expr4(j):=-(kk[j]*l1[j]+l1[j]*sum(mik,k,1,i-1)*l1[j]*m[j]));
expr6(1,n):=1/m*sum(kk[l]*sum(mik,k,1,i-1)*l1[j]*m[j]);
expr6(j,n):=kk[j]*l1[j]+(kk[j]*l1[j])*sum(a[k],k,J-1)*l1[j]*m[j]);
pdt(l,n):=1/m*sum(expr4(j),j,1,n-1)+expr6(1,n)-1/m*sum(expr6(j,n),j,1,n));

expr5(j,n):=-kk[j]*l1[j]+l1[j]*sum(mik,k,1,i-1)*l1[j]*m[j]);
expr6(l,n):=1/m*sum(kk[l]*sum(mik,k,1,i-1)*l1[j]*m[j]);
expr6(j,n):=kk[j]*l1[j]+(kk[j]*l1[j])*sum(a[k],k,J-1)*l1[j]*m[j]);
pdt(l,n):=1/m*sum(expr4(j),j,1,n-1)+expr6(1,n)-1/m*sum(expr6(j,n),j,1,n));

/* angular momentum H[N] w/r to center of mass */

n2l2:=sum((i(l1),l1,1,l,n)*sum(m[j]*pp(l,n),pdt(l,n),j,1,n));
n2l2:=sum((i(l1),l1,1,l,n)*sum(m[j]*pp(l,n),pdt(l,n),j,1,n));

n3rev(n2);
n3revexpand(n2);
n3l3simp(n2);

n3rev(n3);
n3revexpand(n3);
n3l3simp(n3);

display(pp(1,2),pp(2,2),pdt(1,2),pdt(2,2),
pp(3,3),pp(2,3),pp(3,3),pdt(1,3),pdt(2,3),pdt(3,3),
expr1,expr2(1,n),expr1(n),expr(1,j),expr6(1,n),expr(6,j),
pp(1,n),pdt(1,n),n(n,2,n));
```

242
Table 8.11. The Angular Momentum $H_3$ and Related Quantities of Three Holonomic Constraints.

<table>
<thead>
<tr>
<th>1, 3</th>
<th>3 3 2 2 3 2 1 3 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2, 3</td>
<td>3 3 2 2 3 1 2 1 1</td>
</tr>
<tr>
<td>3, 3</td>
<td>3 3 2 2 3 1 1 2 1 3</td>
</tr>
</tbody>
</table>

**PDT**

<table>
<thead>
<tr>
<th>1, 3</th>
<th>3 3 2 2 3 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2, 3</td>
<td>3 3 2 2 3 2 1 2 2 2 2 2 2 2 2 2 2 2 2</td>
</tr>
<tr>
<td>3, 3</td>
<td>3 3 2 2 3 2 1 2 2 2 2 1 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2, 3</th>
<th>3 3 2 2 3 2 1 2 1 3 3 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3, 3</td>
<td>3 3 2 2 3 2 2 2 2 2 1 1 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2, 3</th>
<th>3 3 2 2 3 2 1 2 1 3 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3, 3</td>
<td>3 3 2 2 3 2 2 2 2 2 1 1 1</td>
</tr>
</tbody>
</table>

243
Table 8.11. (cont'd)

<table>
<thead>
<tr>
<th>( t )</th>
<th>2</th>
<th>2</th>
<th>2</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>3</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s )</td>
<td>2</td>
<td>( &lt; )</td>
<td>2</td>
<td>( &gt; )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( m )</td>
<td>(- M )</td>
<td>(- M )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
</tr>
<tr>
<td>( n )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
</tr>
<tr>
<td>( k )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
</tr>
<tr>
<td>( m )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
</tr>
<tr>
<td>( n )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
</tr>
<tr>
<td>( k )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
</tr>
<tr>
<td>( m )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
</tr>
<tr>
<td>( n )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
</tr>
<tr>
<td>( k )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
</tr>
<tr>
<td>( m )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
</tr>
<tr>
<td>( n )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
</tr>
<tr>
<td>( k )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
</tr>
<tr>
<td>( m )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
</tr>
<tr>
<td>( n )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
</tr>
<tr>
<td>( k )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
</tr>
<tr>
<td>( m )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
</tr>
<tr>
<td>( n )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
</tr>
<tr>
<td>( k )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
</tr>
<tr>
<td>( m )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( - M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
<td>( M )</td>
</tr>
<tr>
<td>( n )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( - N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
<td>( N )</td>
</tr>
<tr>
<td>( k )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( - K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
<td>( K )</td>
</tr>
</tbody>
</table>
Appendix C

THE GENERATION OF THE ROTATIONAL DYNAMICS OF TWO PLANAR LINKS AND
THE CORRESPONDING, AUTOMATED CONTROLLABILITY ANALYSIS

This appendix is on the derivation of the rotational dynamics of the planar, two link system, which is shown in Fig. 4.3, and on the corresponding controllability analysis. The appendix refers to the main section 4.3.

Consider Table 9.1, which shows a (MACSYMA) program listing for the symbolic generation of the rotational dynamics of two links from their interconnection diagram, see section 4.2.3. The notation used is identical to that of the above mentioned section. The maps for the transformation of orientation vectors and of angular velocity vectors are chosen according to the Bryant convention [39]. The motion of the system is restricted to the Y-Z plane and to notations about the X-axis only, i.e., matrix entries which do not correspond to the above type of motion are set to zero. After this a data compression is done, i.e., the system matrices of \( f \) and \( g \) of Eq. (4.18), which are 12 x 1 matrices for the three-dimensional system, are reduced to 4 x 1 matrices for planar motion. The resulting system matrices are shown in Eq. (4.20).

Note that the angular momentum constraint, see section 4.3.:

\[
P_2 = H_0 - P_1
\]

is already implemented into the system matrices. This implies that the 3rd and 4th equation of the system are identical so that one of the two equations can be dropped from now on. Furthermore

\[P_1 = p\]

A program listing for the automated controllability analysis is shown in Table 9.2. The nonlinear controllability analysis is done first. A basis set \( L_0 \) c { \( f, g \) }LA is:

\[L_0 = \{g, [f, g], [[f, g], f]\}\]

and has structural parameters, see section 2.6,

\[(n_a, n_s, n_l, n) = (3, 3, 3, 3)\]

i.e., it is (locally) controllable. The set \( L_0 \) is shown in Table 9.3.
Table 9.1. The Symbolic Generation of the Rotational Dynamics of Two Limbs.

```plaintext
print("nlinsys.n");

// definition of the dynamics of 2 limbs in z-u
// from the dynamics of the base system in z-v
// definitions of mechanical parameters
// m: mass of the system
// i: pos. vector of joint rel. to c1
// ti: inertia tensor of beam rel. to c1
// i2: inertia tensor of limb rel. to c2
array(tu2,1); array(ta2,0,hinv),);

// definition of Bryant angles

w1 := block([c1,c2,c3,s1,s2,s3],
  c1:cos(tu1,1), c2:cos(tu1,2), c3:cos(tu1,3),
  tu1:matrix([c1/c3], [c1*c3], [-c1*c3]),
  tu2:matrix([c1/c2], [c1*c2], [-c1*c2]),
  hinv1:matrix([c1/c2], [c1*c2], [-c1*c2]),
  [s1,s2,s3], [s1,s2,s3], [s1,s2,s3]),
  block(// simplification of system matrices for z-u motion//
    tu1, tu2, 0, 0, 0, 0,
    d1, d2, 0, 0, 0, 0,
    u1, u2, 0, 0, 0, 0,
    "1", "2", "3", "3", "3", "3",
    ev(tu1,w1), simp, outcot, 1,3),
    ev(tu2,w1), simp, outcot, 1,3),
    e2t1 := [1,2,3,4,5,6],
    e2t2 := [1,2,3,4,5,6],
    tu1 := matrix([1,0,0],[0,0,0],[0,0,0]),
    tu2 := matrix([0,1,0],[0,0,0],[0,0,0]),
    display(tu1, tu2, hinv1, hinv2),
    [1,2,3,4,5,6], [1,2,3,4,5,6],
    tu1, tu2, hinv1, hinv2),
    [1,2,3,4,5,6], [1,2,3,4,5,6],
    tu1, tu2, hinv1, hinv2),
    [1,2,3,4,5,6], [1,2,3,4,5,6],
    tu1, tu2, hinv1, hinv2),
    [1,2,3,4,5,6], [1,2,3,4,5,6],
    tu1, tu2, hinv1, hinv2)
); /* end of block */
```

246
block
jj'lljj;jz
| then
| (il j <= j then trireduce(l1,ll1,11-j)))
| else
| (it j <= j then trireduce(l2,ll1,11-j)))
jj:=ematrix(jj), (t, t)

ll,jj:=
| then
| (it j <= j then (u[j], w[j]-j)))
| else
| (if 1 <= y then trireduce((u[j+1], w[j+1])) else
| (if 1 <= y then trireduce((u[j], w[j]-j)))
2:=ematrix(t, t, 1, 1)

end of block

save system matrices

tosave("/system_out", 12, 170);
Table 9.2 Linear and Nonlinear Symbolic Controllability Analysis of Two Planar Links.
Table 9.3. Involutive Basis Set of Vectors for the Determination of Nonlinear Controllability of the Rotational Dynamics of Two Planar Links.

<table>
<thead>
<tr>
<th>Vector</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{v}_1 )</td>
<td>( \mathbf{v}_1 = -C_1 \mathbf{v}_1 )</td>
</tr>
<tr>
<td>( \mathbf{v}_2 )</td>
<td>( \mathbf{v}_2 = -C_2 \mathbf{v}_2 )</td>
</tr>
<tr>
<td>( \mathbf{v}_3 )</td>
<td>( \mathbf{v}_3 = 4 C_1 (C_1 \mathbf{v}_1 + \mathbf{v}_1) + 4 C_2 (C_2 \mathbf{v}_2 + \mathbf{v}_2) )</td>
</tr>
<tr>
<td>( \mathbf{v}_4 )</td>
<td>( \mathbf{v}_4 = 4 C_1 (\mathbf{v}_1 - C_1 \mathbf{v}_1) + 4 C_2 (\mathbf{v}_2 - C_2 \mathbf{v}_2) )</td>
</tr>
<tr>
<td>( \mathbf{v}_5 )</td>
<td>( \mathbf{v}_5 = 4 C_1 (\mathbf{v}_1 + C_1 \mathbf{v}_1) - 4 C_2 (\mathbf{v}_2 + C_2 \mathbf{v}_2) )</td>
</tr>
<tr>
<td>( \mathbf{v}_6 )</td>
<td>( \mathbf{v}_6 = 4 C_1 (\mathbf{v}_1 - C_1 \mathbf{v}_1) - 4 C_2 (\mathbf{v}_2 + C_2 \mathbf{v}_2) )</td>
</tr>
<tr>
<td>( \mathbf{v}_7 )</td>
<td>( \mathbf{v}_7 = 4 C_1 (\mathbf{v}_1 + C_1 \mathbf{v}_1) + 4 C_2 (\mathbf{v}_2 - C_2 \mathbf{v}_2) )</td>
</tr>
<tr>
<td>( \mathbf{v}_8 )</td>
<td>( \mathbf{v}_8 = 4 C_1 (\mathbf{v}_1 - C_1 \mathbf{v}_1) + 4 C_2 (\mathbf{v}_2 - C_2 \mathbf{v}_2) )</td>
</tr>
</tbody>
</table>

Note: \( C_i \) represents the involutive operator for the \( i \)-th component, and \( \mathbf{v}_1, \mathbf{v}_2, \ldots \) are vectors defining the basis set.
<table>
<thead>
<tr>
<th>Table 9.3 (cont'd)</th>
</tr>
</thead>
</table>
| \[ V = \left( \begin{array}{c}
V_1 \\
V_2 \\
\vdots \\
V_n
\end{array} \right) \] |
| where \( V_i \) is the \( i \)-th coordinate of \( V \) and \( n \) is the dimension of \( V \). |
| In the context of a vector space, \( V \) is a set of vectors that can be added together and multiplied by scalars. The coordinate of \( V \) in the \( i \)-th position is denoted by \( V_i \). |
250
Table 9.3. (cont'd)

(C24) c1:fullratsimp(deeterminant(submatrix(1,c)));
Time = 125 msec.

(C25) c1:fullratsimp(deeterminant(submatrix(2,c)));
Time = 117 msec.

(C26) c1:fullratsimp(deeterminant(submatrix(3,c)));
Time = 115 msec.

(C27) c1:fullratsimp(deeterminant(submatrix(4,c)));
Time = 117 msec.

\[
\begin{align*}
&\text{Time} = \frac{115}{2} \text{ msec.} \\
&\text{Time} = \frac{117}{2} \text{ msec.} \\
&\text{Time} = \frac{117}{2} \text{ msec.} \\
&\text{Time} = \frac{117}{2} \text{ msec.}
\end{align*}
\]
Table 9.4. Controllability Matrix of the Linearized Rotational Dynamics of Two Planar Links and Test for Controllability Defects.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( 1 )</th>
<th>( 2 )</th>
<th>( 3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( j )</td>
<td>( 1 )</td>
<td>( 2 )</td>
<td>( 3 )</td>
</tr>
<tr>
<td>( 1 )</td>
<td>( -1 )</td>
<td>( 0 )</td>
<td>( 4 )</td>
</tr>
<tr>
<td>( 2 )</td>
<td>( 4 )</td>
<td>( -1 )</td>
<td>( 0 )</td>
</tr>
<tr>
<td>( 3 )</td>
<td>( 0 )</td>
<td>( 4 )</td>
<td>( -1 )</td>
</tr>
</tbody>
</table>

*Note: The matrix entries represent the controllability conditions.*
Table 9.4. (cont'd)

<table>
<thead>
<tr>
<th>Expression</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (110) ) c: = ( \det \left( \text{submatrix}(1, c) \right) )</td>
<td>9.54</td>
</tr>
<tr>
<td>( (110) ) c: = ( \det \left( \text{submatrix}(1, c) \right) )</td>
<td>9.54</td>
</tr>
<tr>
<td>( (110) ) c: = ( \det \left( \text{submatrix}(1, c) \right) )</td>
<td>9.54</td>
</tr>
<tr>
<td>( (110) ) c: = ( \det \left( \text{submatrix}(1, c) \right) )</td>
<td>9.54</td>
</tr>
</tbody>
</table>

Note: The table continues with more expressions and their corresponding times.
Possible rank defects of $L_0$ are found by sequentially eliminating one of its four rows and be evaluating the determinant of the remaining square matrix. The potentially nonzero determinant $\det_1$ is:

$$\det_1 = \frac{2D_0 S_{21}}{(J_1J_2 - C_{21}^2D^2)}$$

The linear controllability analysis is considered next. The controllability matrix $C$, see section 4.4,

$$C = \{B, AB, A^2B\}: 4 \times 3$$

is shown in Table C.5. Controllability defects are found in a similar way as described before. The largest potentially nonzero determinant $\det_2$ of $d$ is:

$$\det_2 = \frac{-S_{21}}{(J_1J_2 - C_{21}^2D^2)^3} \left\{ (D_2^2 - 2C_{21}D^2)J_2 - DJ_1^2 + 2C_{21}^2D_1J_1 \right\} P_1$$

$$+ D_0 \left[ (DJ_1 + C_{21}D^2)J_2 + DJ_1^2 - C_{21}^2D_1J_1 - 2C_{21}^2D^3 \right]$$
REFERENCES


259


